University of Southern Queensland

Faculty of Engineering & Surveying

‘Online Learning Environment’ using ASP.NET

A dissertation submitted by

Timothy Ross Vriesema

in fulfilment of the requirements of

ENG4111/2 Research Project

towards the degree of

Bachelor of Engineering (Software Engineering)

Bachelor of Business (Management and Leadership)

Submitted: October, 2004

Abstract

This dissertation focuses on the management, design, implementation, testing, and deployment of a learning environment for educational institutions. The management activities of this project are based on the techniques outlined by Bruegge & Dutoit (2004) and Sommerville (2001), involving the scheduling of project tasks, management of the possible risks, and documentation of the project's design. The design of the project involves the integration and alteration of documents generated from the management activities, while implementation consists of using the Microsoft .NET Framework class library via the programming languages, ASP.NET, C# .NET, ADO.NET, XML, and SQL. The activities revolving around testing focuses on its documentation via the management activities. As well, a small-scale deployment of this project is introduced into a Microsoft Windows Server 2003 operating system environment.

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof G Baker

Dean

Faculty of Engineering and Surveying

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Timothy Ross Vriesema

Student Number: 0011122446

_____

Signature

_____

Date

Acknowledgments

I would like to take this opportunity to thank the many people that influenced and encouraged me throughout this project.

Firstly, I would like to thank my loving girlfriend, Cherie, and all the support she has given me to continue the rigorous work required to complete this project. Her undivided attention to my incessant ramblings on the technical development helped me overcome problematic issues.

Secondly, I would like to thank my family for their ongoing support in my University studies.

Thirdly, my supervisor, Dr. Hong Zhou. I would like to thank her for her constant advice, feedback, and experience in our weekly meetings to allow my ideas to flow and not kept inside my head.

Finally, in the research and implementation of this project, I was faced with many programming language issues where I could not find the answers. The *Most Valuable Professional* (MVP) developers within the Microsoft newsgroups promptly answered my many of my questions. Thank-you to Jon Skeet [C# MVP], David Portas [SQL Server MVP], and Oleg Tkachenko [XML MVP].

Timothy Ross Vriesema

University of Southern Queensland

October 2004

Contents

Abstract **i**

Certification of Dissertation **iii**

Acknowledgments **iv**

List of Figures **x**

List of Tables **xii**

List of Listings **xix**

Chapter 1 Introduction **1**

1.1 Structure of the Dissertation 2

Chapter 2 Methodology and Techniques **3**

2.1 Software Project Development 4

2.2 Software Project Management 5

2.2.1 Software Process Models 7

2.2.2 Software Project Documentation 8

2.2.3 Software Project Schedule Management 13

2.2.4 Software Project Risk Management 14

2.3 Programming Languages 14

2.3.1 Microsoft Active Server Pages .NET 1.1 17

2.3.2 Microsoft C# .NET 1.1 18

2.3.3 Microsoft Active Data Objects .NET 1.1 21

2.3.4 Extensible Markup Language 21

2.3.5 Structured Query Language 22

2.3.6 Hyper-Text Markup Language 24

2.3.7 Cascade Style Sheeting 24

2.4 Development Environments and Tools 24

2.4.1 Microsoft Visual Studio .NET 2003 24

2.4.2 Microsoft ASP.NET Web Matrix 25

2.4.3 Microsoft Windows Server 2003 25

2.4.4 Microsoft Internet Information Service 6.0 25

2.4.5 Microsoft SQL Server 2000 and 2005 26

2.4.6 Microsoft Developer Network 26

2.5 Software Project Implementations 27

2.5.1 Web Applications 28

Chapter 3 Software Design **31**

3.1 System Architecture 31

3.1.1 System-to-Data Relationship 33

3.1.2 Data Storage Techniques 33

3.2 Database Architecture 35

3.3 Design using the Microsoft .NET Framework 46

Chapter 4 Software Implementation **50**

4.1 Implementation Issues 51

4.1.1 XML in Microsoft SQL Server 2000 51

4.1.2 Registry and File System Permissions 51

4.1.3 Native XML Architecture 52

4.2 Implementation using ASP.NET 52

4.2.1 Authentication 52

4.2.2 Web Forms and User Controls 54

4.3 Implementation using XML 54

4.3.1 XML Data Structures and Database Compatibility 55

4.3.2 Serialization and Deserialization 55

4.4 The Incomplete Web Application 55

Chapter 5 Software Deployment **58**

5.1 Configuration of Microsoft Windows Server 2003 58

5.2 Installation - The Setup Application 59

Chapter 6 Software Project Management 66

6.1 Documentation 66

6.2 Schedule Management 67

6.3 Risk Management 69

6.4 Resource Requirements 72

6.5 Implementation Management using Microsoft Visual Studio .NET . . . 73

6.5.1 `Database Source Code` Project 75

6.5.2 `OLEWebSetup` Project 77

Chapter 7 Conclusions 79

7.1 Further Work 80

7.2 The Future Project 81

References 83

Appendix A Project Specification 88

Appendix B Requirements Analysis Document 90

Appendix C System Design Document 122

Appendix D Object Design Document 135

Appendix E Test Manual **166**

Appendix F Web Application Model **186**

Appendix G Project Gannt Chart **188**

Appendix H Source Code **195**

H.1 `OLEWebApplication` Project 195

H.2 `OLEBusinessLogic` Project 293

H.3 `Database Source Code` Project 403

List of Figures

2.1 The *Waterfall* Process Model 7

2.2 The *Evolutionary Development* Process Model 8

2.3 The outline of the Requirements Analysis Document (RAD) 10

2.4 The outline of the System Design Document (SDD) 11

2.5 The outline of the Object Design Document (ODD) 12

2.6 The outline of the Test Manual 13

2.7 The Microsoft .NET Framework's components 16

2.8 The tier application architectural designs 29

3.1 The three-tier application architecture 32

3.2 The steps in the Database Design Methodology (Steps 1 to 2) 36

4.1 *Login* page 56

4.2 *Home* page for user 'Jonathan Trinder' (`userID`: 26) 56

4.3 'Assignment 1' for 'Research Project 1' (`courseID`: ENG4111) within the *Class* page 57

5.1 The *Application Server* window from the *Windows Component Wizard*, under *Windows Server 2003* 59

5.2 Welcome screen (first step) 60

5.3 SQL Server name setup (second step) 61

5.4 IIS virtual folder and port setup (third step) 62

5.5 Confirm installation (forth step) 63

5.6 Setup installing (fifth step) 64

5.7 Setup finished (sixth step) 65

6.1 Microsoft Visual Studio .NET 2003's Solution Explorer (for the OLE *solution*) 74

6.2 Microsoft Visual Studio .NET 2003's *Debug* Configuration Manager (for the OLE *solution*) 75

6.3 Microsoft Visual Studio .NET 2003's *Release* Configuration Manager (for the OLE *solution*) 75

6.4 Microsoft Visual Studio .NET 2003's database table creation/alteration (for the OLE *solution*) 76

6.5 Microsoft Visual Studio .NET 2003's database relationship creation/alteration (for the OLE *solution*) 77

6.6 The *User Interface* configuration 78

6.7 The *File System* configuration 78

6.8 The *Registry* configuration 78

7.1 New database concept. 81

List of Tables

2.1 Software processes within the *IEEE 1074: Software Developing Life Cycle Processes* 6

3.1 The attributes within the entity types *Users*, *AETQuestions*, and *AETAnswers* 38

3.2 The attributes within the entity types *Classes*, *Roles*, and *ThreadSubscriptions* 38

3.3 The attributes within the entity types *ClassUsers* and *OtherUsers* . . . 39

3.4 The attribute domains for the attributes in Table 3.1 40

3.5 The attribute domains for the attributes in Table 3.2 40

3.6 The attribute domains for the attributes in Table 3.3 41

3.7 The highlighted and created primary and composite keys from Table 3.1 42

3.8 The highlighted and created primary and composite keys from Table 3.2 42

3.9 The highlighted and created primary and composite keys from Table 3.3 42

6.1 The Documentation, Requirements Elicitation and Analysis, and Design work structure 68

6.2 The Training, Implementation, Testing, and Deployment work structure 69

6.3 Risk types and the risks that may be encountered (Risk Identification) . 70

6.4 Risks, their probability and effects (Risk Analysis) 71

6.5 Risks and their strategies (Risk Strategy) 72

6.6 Resource requirements 73

List of Listings

2.1 Interface example in C# 19

2.2 C/C++ example of how and where memory is allocated and deallocated 20

2.3 C# example of where memory is allocated and deallocated 20

2.4 `FOR XML EXPLICIT` stored procedure 23

2.5 Output from the stored procedure in Listing 2.4 23

H.1 Class.aspx 195

H.2 Class.aspx.cs 196

H.3 Configuration.aspx 200

H.4 Configuration.aspx.cs 201

H.5 Default.aspx 202

H.6 Default.aspx.cs 204

H.7 Default.css 204

H.8 Footer.ascx 214

H.9 Footer.ascx.cs 214

H.10 Global.asax 214

H.11 Global.asax.cs 214

H.12 Header.ascx 218

H.13 Header.ascx.cs 219

H.14 Login.aspx 221

H.15 Login.aspx.cs 222

H.16 Management.aspx 222

H.17 Management.aspx.cs 223

H.18 Menu.ascx 224

H.19 Menu.ascx.cs 225

H.20 Submission.aspx 226

H.21 Submission.aspx.cs 227

H.22 Web.config 228

H.23 ClassControls/AETSubmissionList.ascx 228

H.24 ClassControls/AETSubmissionList.ascx.cs 228

H.25 ClassControls/ClassAnnoucements.ascx 229

H.26 ClassControls/ClassAnnoucements.ascx.cs 232

H.27 ClassControls/InformationAET.ascx 233

H.28 ClassControls/InformationAET.ascx.cs 234

H.29 ClassControls/MultipleChoiceAET.ascx 236

H.30 ClassControls/MultipleChoiceAET.ascx.cs 237

H.31 ClassControls/ShortResponseAET.ascx 242

H.32 ClassControls/ShortResponseAET.ascx.cs 243

H.33 ClassControls/ShortResponseAETAnswers.ascx 247

H.34 ClassControls/ShortResponseAETAnswers.ascx.cs 247

H.35 ClassControls/ThreadPosts.ascx 248

H.36 ClassControls/ThreadPosts.ascx.cs 250

H.37 ClassControls/Threads.ascx 252

H.38 ClassControls/Threads.ascx.cs 253

H.39 ConfigurationControls/ApplicationConfiguration.ascx 254

H.40 ConfigurationControls/ApplicationConfiguration.ascx.cs 254

H.41 ConfigurationControls/ClassConfiguration.ascx 255

H.42 ConfigurationControls/ClassConfiguration.ascx.cs 255

H.43 ConfigurationControls/ClassEnrollmentConfiguration.ascx 256

H.44 ConfigurationControls/ClassEnrollmentConfiguration.ascx.cs 256

H.45 ConfigurationControls/InstitutionConfiguration.ascx 256

H.46 ConfigurationControls/InstitutionConfiguration.ascx.cs 257

H.47 ConfigurationControls/RoleAssignmentConfiguration.ascx 257

H.48 ConfigurationControls/RoleAssignmentConfiguration.ascx.cs 257

H.49 ConfigurationControls/UserConfiguration.ascx 258

H.50 ConfigurationControls/UserConfiguration.ascx.cs 258

H.51 HomeControls/ErrorMessages.ascx 259

H.52 HomeControls/ErrorMessages.ascx.cs 261

H.53 HomeControls/ForumThreadSubscriptions.ascx 262

H.54 HomeControls/ForumThreadSubscriptions.ascx.cs 262

H.55 HomeControls/InstitutionAnnouncements.ascx 263

H.56 HomeControls/InstitutionAnnouncements.ascx.cs 266

H.57 HomeControls/QuickLinks.ascx 267

H.58 HomeControls/QuickLinks.ascx.cs 268

H.59 HomeControls/UpcomingDates.ascx 269

H.60 HomeControls/UpcomingDates.ascx.cs 271

H.61 LoginControls/Authentication.ascx 273

H.62 LoginControls/Authentication.ascx.cs 274

H.63 ManagementControls/ClassAnnouncementManagement.ascx 275

H.64 ManagementControls/ClassAnnouncementManagement.ascx.cs 276

H.65 ManagementControls/ClassManagement.ascx 276

H.66 ManagementControls/ClassManagement.ascx.cs 277

H.67 ManagementControls/ClassNoteManagement.ascx 277

H.68 ManagementControls/ClassNoteManagement.ascx.cs 277

H.69 ManagementControls/InformationAETManagement.ascx 278

H.70 ManagementControls/InformationAETManagement.ascx.cs 278

H.71 ManagementControls/InstitutionAnnouncementManagement.ascx . . 279

H.72 ManagementControls/InstitutionAnnouncementManagement.ascx.cs 279

H.73 ManagementControls/MultipleChoiceAETManagement.ascx 280

H.74 ManagementControls/MultipleChoiceAETManagement.ascx.cs 280

H.75 ManagementControls/ShortResponseAETManagement.ascx 281

H.76 ManagementControls/ShortResponseAETManagement.ascx.cs 281

H.77 ManagementControls/UserManagement.ascx 281

H.78 ManagementControls/UserManagement.ascx.cs 282

H.79 MenuControls/Assignments.ascx 282

H.80 MenuControls/Assignments.ascx.cs 283

H.81 MenuControls/Classes.ascx 284

H.82 MenuControls/Classes.ascx.cs 285

H.83 MenuControls/Exams.ascx 286

H.84 MenuControls/Exams.ascx.cs 286

H.85 MenuControls/Notes.ascx 288

H.86 MenuControls/Notes.ascx.cs 288

H.87 MenuControls/Tutorials.ascx 289

H.88 MenuControls/Tutorials.ascx.cs 290

H.89 SubmissionControls/Class.ascx 291

H.90 SubmissionControls/Class.ascx.cs 291

H.91 SubmissionControls/User.ascx 292

H.92 SubmissionControls/User.ascx.cs 292

H.93 Address.cs 293

H.94 AETAnswer.cs 294

H.95 AETQuestion.cs 309

H.96 Authentication.cs 330

H.97 Class.cs 332

H.98 Configuration.cs 338

H.99 DataBridge.cs 342

H.100 ErrorAndSecurity.cs 358

H.101 Limitations.cs 366

H.102 MD5Encryption.cs 367

H.103 Post.cs 368

H.104 Role.cs 382

H.105 Thread.cs 387

H.106 User.cs 392

H.107 oledb.sql 403

Chapter 1

Introduction

The idea of the *'Online Learning Environment' using ASP.NET* (OLE) was spawned from an interest in software engineering. Software engineering is quite a broad area, so the idea was to involve as many software engineering tasks as possible. These include software project management and enterprise-level application architectural design, implementation, testing, and deployment.

The initial ideas and research required a small amount of experience in application design and implementation. Throughout the writer's own technical experience and University studies, experience within Microsoft Active Server Pages (ASP), C and C++, and databases (SQL) in design, implementation, testing, and deployment had already been achieved. For the purposes of this project however, the writer sought to expand his current level of experience in these fields, to new, modern, and exciting fields.

The main inspiration for this project came from using an already established enterprise-level Web application from the company *WebCT, Inc.* and their educational environment application, *Vista*.

1.1 Structure of the Dissertation

The dissertation begins with Chapter 2, introducing methodologies and techniques researched in order to gain an idea as to how development of the project would take place. Chapter 6 then outlines the activities involved in software project management. The design of the OLE project is then detailed in Chapter 3, while Chapter 4 displays how the OLE project was implemented. Chapter 5 details the deployment of the OLE project. The final chapter, Chapter 7, provides project conclusions and ideas for future projects to follow on from the OLE.

The appendices of this dissertation include (in order):

A. *Project Specification* - A contract between the University of Southern Queensland and the writer to ensure a pass grade (minimum) within the courses ENG4111 and ENG4112, Research Project 1 and 2.

B. *Requirements Analysis Document* - A document detailing the requirements of the OLE project.

C. *System Design Document* - A document detailing the system design of the OLE project.

D. *Object Design Document* - A document detailing the object design of the OLE project.

E. *Test Manual* - A document detailing the test procedures for the OLE project.

F. *Web Application Model* - A Unified Modeling Language model of the OLE Web application.

G. *Project Gantt Chart* - A chronological gantt chart that lists tasks and their completion times.

H. *Source Code* - The source code generated from the OLE project's implementation.

Chapter 2

Methodology and Techniques

Throughout the OLE project, different methodologies and techniques into software engineering were required to be researched. The areas researched within this project consisted of software project management, programming languages, development environments and tools, and software project implementation methods and techniques.

Application Programming Interface (API) 'The set of programming language constructs or statements that can be coded in an application program to invoke the specific functions and services provided by an underlying operating system or service program.' (Microsoft Developer Network 2003, his/snagloss_1b4x.htm)

Common Language Runtime (CLR) 'The engine at the core of managed code execution. The runtime supplies managed code with services such as cross-language integration, code access security, object lifetime management, and debugging and profiling support.' (Microsoft Developer Network 2003, p. Netstart/html/cpgloc.htm)

Database Management System (DBMS) 'A software system that enables users to define, create, maintain, and control access to the database.' (Connolly & Begg 2002, p. 16)

Dynamic Link Library (DLL) A file containing a library of functions that can be dynamically linked to many applications. When called upon, it allocates itself to

memory only once, so it can be accessed by other applications.

Technical Preview Beta An application that allows developers to understand the application before it is released to the public. It also contains software bugs due to the fact that it is still under development.

2.1 Software Project Development

Software Project Development consists of processes that make up the development of a software application. Some of these processes are:

- Requirements Elicitation
- Requirements Analysis
- System Design
- Object Design
- Implementation
- Testing

Firstly, the *Requirements Elicitation* process focuses on defining the purpose of the software system, generating functional and non-functional requirements from an initial problem statement. A functional requirement is an 'area of fuctionality the system must support' (Bruegge & Dutoit 2000, p. 718), whereas a non-functional requirement is a 'user-visable constraint on the system' (Bruegge & Dutoit 2000, p. 724).

Secondly, the *Requirements Analysis* process uses the generated products from the *Requirements Elicitation* process that generates an analysis model that is composed of three other model:s:

- *Functional Model* - Use cases and scenarios.
- *Analysis Object Model* - Class and object diagrams.

- *Dynamic Model* - Statechart and sequence diagrams.

Thirdly, the *System Design* process uses the generated products from the *Requirements Analysis* and *Requirements Elicitation* process to generate design goals and decompose the subsystems. Fourthly, the *Object Design* process uses the subsystem decompositions to generate a class diagram. Finally, the *Implementation* process uses the generated class diagram to produce working source code, which can then be tested via the *Testing* process.

2.2 Software Project Management

Software Project Management consists of activities and techniques to monitor and maintain a certain level of performance in the development of a software project. Some of the activities in software project management are:

- Scheduling the workload over a period of time;
- Dividing the workload over a range of human resources;
- Defining and manage the risks throughout the project; and
- Documenting the progress, technical specifications, and functional/non-functional requirements.

The life cycles for software projects vary from organisation-to-organisation, however, the activities involved are consistent throughout. The *Institute of Electrical and Electronics Engineers* (IEEE) have constructed a standard for the development life cycle within software projects known as the *IEEE 1074: Standard for Developing Life Cycle Processes.*

Bruegge & Dutoit (2000, pp. 460-468) describes the *IEEE 1074: Standard for Developing Life Cycle Processes*, from the IEEE (1997). Table 2.1 gives the process groups and processes within each group under the IEEE 1074 specification.

Table 2.1: Software processes within the *IEEE 1074: Software Developing Life Cycle Processes*

Process Group	***Process***
Project Management	Project Initiation Project Management and Control Software Quality Management
Pre-development	Concept Exploration System Allocation
Development	Requirements Design Implementation
Post-development	Installation Operation and Support Maintenance Retirement
Integral Processes	Verification and Validation Software Configuration Management Document Development Training

The first process group from Table 2.1, *Project Management*, is designed to monitor and control the project throughout the software life cycle. The next process, *Pre-development* researches the concept through different ways in order to understand and appreciate the project in more detail. The *Development* process follows by defining the software's requirements, then designing and implementing the software. *Post-development* includes the installation of the software, while offering support and maintenance. The last processes under *Integral Processes* are performed throughout the duration of the project, which include software configuration management, documentation development, and training.

2.2.1 Software Process Models

The implementation of a software application has a number of models to approach the process of software project management. Several software process models were detailed by Sommerville (2001, pp. 44-50). Specifically, two of these models are:

- *The waterfall model* - The fundamental activities within the development of a software project are represented as phases such as requirements specification, software design, implementation, and testing.
- *Evolutionary development* - Interleaves the fundamental activities of a software project. Abstract specifications then create an initial system to be refined with customer input to satisfy the requirements of the customer.

The main difference between these two models processes is the order in which each activity occurs. To clarify, using the *waterfall* process model (Figure 2.1) in an ideal situation only allows back-tracking to the previous activities when the last activity has been reached. Whereas, the *evolutionary development* process model (Figure 2.2) allows back-tracking between all activities.

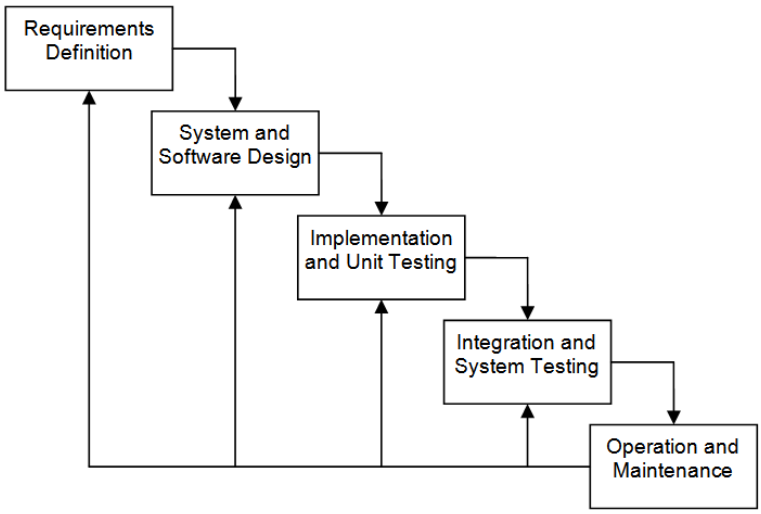


Figure 2.1: The *Waterfall* Process Model

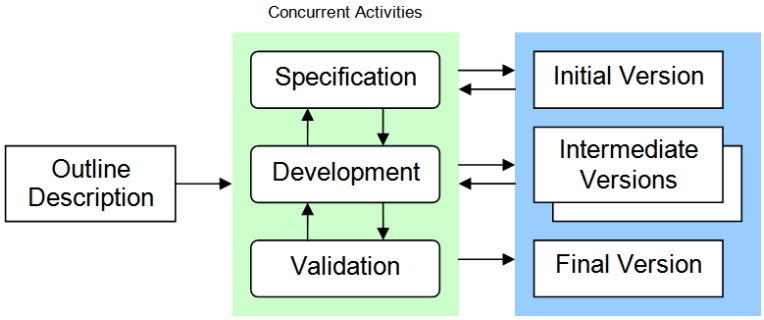


Figure 2.2: The *Evolutionary Development* Process Model

2.2.2 Software Project Documentation

Documentation of a software application is an important component within every software project. It relates to software project management by allowing project managers to record alterations throughout the project. Within an organisation, executives monitor and keep progress reports and information relating to all projects. Another reason for keeping documentation is so that if project staff leave the organisation, new project staff can research the documentation and understand the project.

The four documents that will be described in detail are:

- *Requirements Analysis Document* (RAD) - Describes system's functional and non-functional requirements and serves as a contract between the developers and the client. The audience for the RAD includes the client, the users, project management, the system analysts, and the system designers (Bruegge & Dutoit 2004, pp. 151-153).
- *System Design Document* (SDD) - Describes the design by subsystem decomposition (with the use of UML class diagrams), hardware and software mapping (with the use of UML deployment diagrams), access control, control flow mechanisms, data management, and boundary conditions. The audience for the SDD includes system architects, project management, and the developers who implement each subsystem (Bruegge & Dutoit 2004, pp. 282-284).
- *Object Design Document* (ODD) - Describes object design and the guidelines

followed for subsystem interfaces, decomposition of subsystems into packages and classes, and class interfaces. The audience for the ODD includes system architects, developers who implement the subsystem, and testers (Bruegge & Dutoit 2004, pp. 373-379).

- *Test Manual* - Describes the scope, approach, and records the differences between the expected output and the results of the test. The audience for the test manual includes developers who implement the system, testers, and system architects (Bruegge & Dutoit 2004).

- *User Manual* describes the instructions in using the software application. This is so that users can refer to it for information and instructions on the software application. The audience for the User Manual includes users that interact with the system, developers, and testers (Bruegge & Dutoit 2004).

From Section 2.2.1, it was mentioned that this project would follow the *evolutionary development* process model. This means that these four documents are not completed sequentially. That is, the SDD does not have to commence before the RAD, or the ODD before the SDD. To clarify, these documents will interleave between each other until completed.

Commonly, however, the document that commences first is the RAD, since it defines the functions and features that will exist within the software project.

Requirements Analysis Document

Figure 2.3 outlines the template used for the RAD. It has been modified from its original outline, specified by Bruegge & Dutoit (2004, p. 152), so that its section headings fit the OLE project.

Requirements Analysis Document

1. Introduction
    - 1.1 Purpose of the System
    - 1.2 Scope of the System
    - 1.4 Definitions, Acronyms, and Abbreviations
    - 1.6 Overview
2. Current System
    - 2.1 Overview
    - 2.2 Functions and Features
3. Proposed System
    - 3.1 Overview
    - 3.2 Functional Requirements
    - 3.3 Non-Functional Requirements
        - 3.3.1 Usability
        - 3.3.2 Reliability
        - 3.3.3 Performance
        - 3.3.4 Supportability
        - 3.3.5 Implementation
        - 3.3.6 Interface
        - 3.3.7 Operation
    - 3.4 System Models
        - 3.4.1 Scenarios
        - 3.4.2 Use Case Models
        - 3.4.3 Object Model
        - 3.4.4 Dynamic Models
        - 3.4.5 User Interface

Glossary

Figure 2.3: The outline of the Requirements Analysis Document (RAD)

Section 1 of the RAD involves an introduction into the system, providing information about its purpose and scope. *Section 2* provides an analysis of a current system's functions and features. *Section 3* is the largest section of this document. It lists the functional and non-functional requirements that will exist for the proposed system. The non-functional requirements are made up of the system's usability, reliability, performance, supportability, implementation, interface, and operation requirements. It

details system models using UML diagrams, which describe the system with respect to the functional and non-functional requirements.

System Design Document

Figure 2.4 outlines the template used for the SDD. It has been modified from its original outline, specified by Bruegge & Dutoit (2004, p. 283), so that its section headings fit the OLE project.

System Design Document

1. Introduction
    - 1.1 Purpose of the System
    - 1.2 Design Goals
    - 1.3 Definitions, Acronyms, and Abbreviations
    - 1.5 Overview
2. Current Software Architecture
3. Proposed Software Architecture
    - 3.1 Overview
    - 3.2 Subsystem decomposition
    - 3.3 Hardware/Software Mapping
    - 3.4 Persistent Data Management
    - 3.5 Access Control and Security
    - 3.6 Global Software Control
    - 3.7 Boundary Conditions
4. Subsystem Services

Figure 2.4: The outline of the System Design Document (SDD)

Section 1 of the SDD is similar to the RAD's introduction. It provides an introduction to the system's design goals, in addition to describing purpose of the system. *Section 2* briefly details the current system's architecture. *Section 3* details the proposed system's overall architecture. This consists of decomposing and detailing the subsystems, mapping hardware and software, defining how persistent data is managed, security and access control features, how the software is controlled, and the conditions of start-up and errors. *Section 4* then defines the subsystems and the services they provide.

Once the first revision of this document started to take shape, the implementation of this project was commence.

Object Design Document

Figure 2.5 outlines the template used for the ODD. It has been modified from its original outline, specified by Bruegge & Dutoit (2004, p. 376), so that its section headings fit the OLE project.

Object Design Document

1. Introduction
    - 1.1 Object Design Trade-Offs
    - 1.2 Interface Documentation Guidelines
    - 1.3 Definitions, Acronyms, and Abbreviations
    - 1.4 Overview
2. Packages
3. Class Interfaces

Figure 2.5: The outline of the Object Design Document (ODD)

Section 1 of the ODD introduces the reader to the object design trade-offs and documentation guidelines for the interface. *Section 2* provides a detailed description of packages within the system. *Section 3* details the class interfaces, which are used throughout the system.

This document flows through a number of revisions in an attempt to increase efficiency and decrease complexity of the class interfaces.

Test Manual

Figure 2.6 outlines the sections used for the *Test Manual*. It is a combination of the two documents, *Test Plan* and *Test Case Specification*, specified by Bruegge & Dutoit (2004, pp. 476-477).

Test Manual

1. Introduction

2. Testing

    2.1 Unit Testing

    2.2 Integrated Testing

Appendix A. Source Code

Figure 2.6: The outline of the Test Manual

Section 1 provides a brief introduction of how the testing will be carried out. *Section 2* is split into unit and integrated testing. Unit testing tests the classes individually and integrated testing tests the user interfaces, which integrates the classes from the project.

2.2.3 Software Project Schedule Management

A software project's schedule is vital to its development. Deadlines can be created so that developers stay on time. This project activity can only be set by individuals with experience in the respective software field. For this project, seven components were fashioned:

- *Documentation* - Lists the documents that are required to be completed and in which order.
- *Requirements Elicitation and Analysis* - Lists the process required to produce the requirements and analyse them.
- *Design* - Lists the process required to produce the design of the system.
- *Training* - Lists the languages required to be learnt before implementation.
- *Implementation* - Lists the steps involved in implementing the system.
- *Testing* - Lists the process required to test two other components. They are:
    - *Unit Testing* - The back-end components of the system.

  - *Integration Testing* - All of the components with the system, working together.

- *Deployment* - Lists the process required to deploy the system.

2.2.4 Software Project Risk Management

All projects, software or other, have unexpected issues that may arise throughout its development. *Risk Management* is a common safeguard that ensures that these issues are minimised or averted. Sommerville (2001, pp. 84-91) describes some of the risks involved when developing a software project. He divides risk management into three sections:

- *Risk Identification* - Identifies and lists the risks that may be encountered within the design and implementation.
- *Risk Analysis* - Lists and analyses the risks that may affect or jeopardise the project; along with the probability of affect upon, and the effect it may impose on the project.
- *Risk Strategy* - Lists the risks from the risk analysis section and strategies that may be carried out in order to avoid such a danger.

2.3 Programming Languages

This section introduces the programming models and languages that were researched in order to implement a modern software application. Firstly, Microsoft .NET and Microsoft .NET Framework will be detailed, while then describing ASP.NET, C# .NET, ADO.NET, XML, SQL, HTML, and CSS. Their acronyms will also be defined in each respective sub-section.

The primary languages will be described in greater depth, introducing their main features and components.

Microsoft .NET

The *Microsoft .NET* platform is used for building, running, and experiencing the next generation of distributed applications. It extends over clients, servers, and services. Microsoft Corporation (2001*a*) outlines Microsoft .NET and that it consists of:

- *A unified programming model* - Enables developers to build applications and XML Web services;
- *Client software* - This helps developers deliver a deep and compelling user experience across a family of devices;
- *Servers* - Integrates, executes, operates, and manages applications and XML Web services;
- *XML Web services* - Helps developers create a simple and integrated user experience; and
- *Tools* - Are used to develop Windows and Web applications, and XML Web services.

Microsoft .NET Framework 1.1

The *Microsoft .NET Framework* is the programming model of the Microsoft .NET platform. It allows for building, deploying, and running applications and XML Web services. It also manages much of the inner workings of .NET, enabling developers to concentrate on writing business logic code for their software applications (Microsoft Corporation 2001*a*). The Microsoft .NET Framework consists of two main components:

- Common Language Runtime (CLR)
- Class Libraries

Figure 2.7 depicts the Microsoft .NET Framework and how its components exist with each other (Microsoft Corporation 2001*a*).

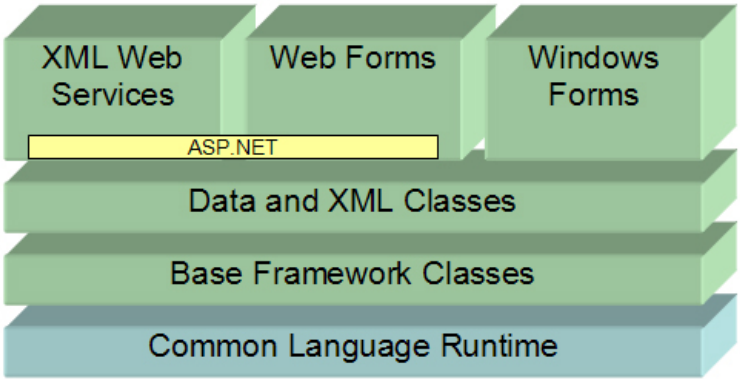


Figure 2.7: The Microsoft .NET Framework's components

The Common Language Runtime (CLR) is the core engine responsible for services such as language integration, and memory, process, and thread management. It also has the role of managing development time features such as; strong type naming, cross language exception handling, and dynamic binding. This reduces the amount of code to be written by developers in order to turn business logic into reusable components (Microsoft Corporation 2001*b*).

The Class Libraries provide what the name suggests, a library of classes for the use of developers. Microsoft Corporation (2001*b*) lists some of these as:

- *Base classes* - Provide standard functionality for developers such as, string manipulation, input/output, network communications, thread management, text management, and many others;
- *Data classes* - Support data management and include SQL classes for manipulating data stores through a standard SQL interface;
- *XML classes* - Enable XML data manipulation, searching, and translations;
- *Web Forms classes* - Include the classes that enable developers to rapidly create Web graphical user interface (GUI) applications; and
- *Windows Forms classes* - Support developers in creating GUI for Windows. This provides a common, consistent development interface across all languages supported by the Microsoft .NET Framework.

2.3.1 Microsoft Active Server Pages .NET 1.1

Microsoft Active Server Pages .NET (ASP.NET) is more than the next version of Microsoft Active Server Pages 3.0 (ASP); it is a Web development platform that provides the services that are required for developers to design and build enterprise-class Web applications (Microsoft Corporation 2004*a*).

The following sub-sections will briefly describe some of the main components within ASP.NET.

Web Forms

Microsoft Developer Network (2003, p. Netstart/html/cpglow.htm) defines a *Web Form* as an 'ASP.NET page framework, which consists of programmable Web pages ... that contain reusable server controls'. To clarify, a Web form is a Web page that is compiled dynamically when it is requested by a client. Once a client requests the page, the ASP.NET Web server executes the methods defined by the programmer, with its result sent back to the client.

Web User Controls

Microsoft Developer Network (2003, p. Netstart/html/cpglou.htm) defines a *Web User Control* as:

> *'A server control that is authored declaratively using the same syntax as an ASP.NET page and is saved as a text file with an .ascx extension. User controls allow page functionality to be partitioned and reused. Upon first request, the page framework parses a user control into a class that derives from System.Web.UI.UserControl and compiles that class into an assembly, which it reuses on subsequent requests.'*

To clarify, a Web user control is identical to a Web form, however, they are designed to be included within Web forms. For example, most Web pages use a menu to navigate

throughout the site. So that the menu does not have to be repeated for each page, a Web user control (`menu.ascx`, for example) is included at a specific location within all ASP.NET pages.

Code-Behind

Code-Behind can be defined in two forms, files and classes. A *code-behind file* is 'a code file containing the page class that implements the program logic of a Web Forms or ASP.NET mobile Web Forms application' (Microsoft Developer Network 2003, p. Netstart/html/cpgloc.htm). A *code-behind class* is 'a class that is accessed by an .aspx file, but resides in a separate file (such as a .dll or .cs file). For example, you can write a code-behind class that creates an ASP.NET custom server control, contains code that is called from an .aspx file, but does not reside within the .aspx file' (Microsoft Developer Network 2003, p. Netstart/html/cpgloc.htm).

This defines a very important advantage for programmers since the logic of an ASP.NET page can exist outside of the page's presentation code. Linking pages to classes also allows programmers to develop powerful methods and properties that interact with the page in a more logical manner.

2.3.2 Microsoft C# .NET 1.1

Microsoft C# .NET is the most common programming/scripting language for ASP.NET and other methods. It is a modern, object-oriented, and type-safe language and enables programmers to build a wide range of different applications for the new Microsoft .NET platform. C#, pronounced "see-sharp", is designed to enable rapid development to the C++ programmer without sacrificing the power and control that have been the trademarks of C and C++ (Microsoft Corporation 2004*e*).

In 2001, the International Standards Office (ISO) received Microsoft C# .NET to be standardised. In April of 2003, it was ratified as ISO/IEC 23270 (Microsoft Corporation 2004*d*).

In the following sub-sections, a brief description will be given to some of the main concepts and components within C# .NET.

Value-Types versus Reference-Types

In order to explain value and reference-types correctly, their definitions must be described. A *value-type* is a 'data type that is represented by the type's actual value' (Microsoft Developer Network 2003, Netstart/html/cpglov.htm). To clarify, changing a value-type variable changes the actual value of the variable. Some value-typed data-types are `int`, `char`, enumerations, and structures.

A *reference-type* is a 'data type that is represented by a reference (similar to a pointer) to the type's actual value' (Microsoft Developer Network 2003, Netstart/html/cpglor.htm). To clarify, a reference-type pointed to a place in memory where the actual value was held. This allows multiple reference-type variables to point to the same location, each changing the same actual value. All classes are reference-types.

Interfaces

An interface provides a framework to which a class must conform. That is, it supplies a definition of properties and/or methods which must be implemented by any derived class. Listing 2.1 shows an example of an interface.

Listing 2.1: Interface example in C#

```
public interface IManage
{
    bool isReadonly
    {
        get;
        set;
    }
    void Create();
    void Delete();
    void Update();
```

```
}
```

Classes

A class within C# is made up of methods (virtual and abstract), attributes, properties, and static members. These class members are identical to C++, with minor changes. One of these changes is that in C++, a class can only inherit one other class. In C#, his ability has been replaced by enabling a class to inherit an infinite amount of interfaces.

Garbage Collection

Garbage collection is not a new concept within programming languages. It is 'the process of transitively tracing through all pointers to actively used objects in order to locate all objects that can be referenced, and then arranging to reuse any heap memory that was not found during this trace' (Microsoft Developer Network 2003, p. Netstart/html/cpglog.htm).

C and C++ do not have garbage collection. When memory was allocated to a variable, it must be deallocated. Listing 2.2 demonstrates an example of this.

Listing 2.2: C/C++ example of how and where memory is allocated and deallocated

```
int main()
{
    int mem[] = new int[100] // Allocated memory of 100 ints
    // mem = null; // Would not deallocate the memory (no
        garbage collection)
    delete mem; // Allocated memory now deallocated
}
```

C#, however, has this feature, and the allocated memory is deallocated once it is no longer referenced by a variable. Listing 2.3 demonstrates an example of this.

Listing 2.3: C# example of where memory is allocated and deallocated

```
static void Main(string[] args)
```

```
2     {
3         int[] mem = new int[100]; // Allocated memory of 100 ints
4         mem = null; // Allocated memory now deallocated (garbage
              collection)
5     }
```

2.3.3 Microsoft Active Data Objects .NET 1.1

Microsoft Active Data Objects .NET, also known as ADO.NET, is defined by Microsoft Developer Network (2003, p. Netstart/html/cpglog.htm) to be:

> *'The suite of data access technologies included in the .NET Framework class libraries that provide access to relational data and XML. ADO.NET consists of classes that make up the DataSet (such as tables, rows, columns, relations, and so on), .NET Framework data providers, and custom type definitions (such as SqlTypes for SQL Server).'*

To clarify, ADO.NET consists of data connectivity features via the .NET Framework in the form of database and XML. The .NET Framework namespace that provides these class libraries is `System.Data`.

2.3.4 Extensible Markup Language

Extensible Markup Language (XML) is a simple and very flexible text format derived from the language, Standard Generalized Markup Language (SGML). It is playing an increasingly successful and important role in the exchange of a wide variety of data on the Web and elsewhere.

Microsoft Developer Network (2003, p. Netstart/html/cpgloe.htm) defines XML as providing 'a uniform method for describing and exchanging structured data that is independent of applications or vendors.' XML specification also provides more complex tools to maintain and manipulate XML.

Extensible Markup Language Schema Definitions

Extensible Markup Language Schema Definitions (XSD) are a tool to structure and maintain that structure (through validation) in XML data files. This is done by defining data-types to the different nodes' attributes and elements. Also, it defines the structure of the node by having a set of rules that determine how many elements and attributes exist under the root node.

Extensible Stylesheet Language Transformations

Extensible Stylesheet Language Transformations (XSLT) is another tool for XML files that can manipulate the way in which the data is displayed to the user. It allows the XML data to be unchanged so that other XSLT files can manipulate the data dynamically.

2.3.5 Structured Query Language

Structured Query Language (SQL) is the standard language for database operations and commands. Simple programming syntaxes allow tables, rows, and columns to be added to a database easily and efficiently. SQL also allows fast and efficient listings of database entries to be viewed by users.

Stored Procedures (Microsoft SQL Server specific)

Microsoft Developer Network (2003, architec/8_ar_da_0nxv.htm) defines a *stored procedure* as a 'precompiled collection of Transact-SQL statements stored under a name and processed as a unit'. To clarify, it stores a precompiled query within the database as the database changes. This increases server effeciency by stopping the requirement to process and compile query requests.

XML Support

The support given by Microsoft SQL Server 2000 for XML is provided by transforming database data in the form of an XML document. This XML document can then by read and transformed by the application that requests the data. Listing 2.4 demonstrates a stored procedure that implements a `FOR XML EXPLICIT` mode. Listing 2.5 details an example XML document that would be compiled from Listing 2.4.

Listing 2.4: `FOR XML EXPLICIT` stored procedure

```
CREATE PROCEDURE dbo.FindClass
    (
        @classID CHAR(10)
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [Classes!1],
        NULL AS [Class!2!classID],
        NULL AS [Class!2!Name!element],
        NULL AS [Class!2!denyOtherUsers]
    UNION ALL
    SELECT
        2 AS Tag,
        1 AS Parent,
        NULL,
        Classes.classID,
        Classes.className,
        Classes.denyOtherUsers
    FROM Classes
    WHERE classID=@classID
FOR XML EXPLICIT
```

Listing 2.5: Output from the stored procedure in Listing 2.4

```
<Classes>
    <Class classID="ENG4111" denyOtherUsers="1">
        <Name>Research Project 1</Name>
```

```
4            </Class>
5        </Classes>
```

2.3.6 Hyper-Text Markup Language

HyperText Markup Language (HTML) is made up of tags such as `<h1>` and `</h1>` to structure text into headings, paragraphs, images, lists as well as hypertext links so that data and information can be displayed in an appropriate and eye catching method. HTML is another language that is derived from the language SGML.

2.3.7 Cascade Style Sheeting

Cascading Style Sheets (CSS) is a simple programming mechanism used for adding styles to HTML Web documents. These include colours, fonts, spacing, table sizing, as well as a number of other style attributes. It was designed to speed up Web page development by specifying a style class in place of typing code repeatedly.

2.4 Development Environments and Tools

Every software project requires an environment under which development occurs. Tools are also important in assisting in the development of the project. The following subsections will briefly describe the environments and tools utilized within this project.

2.4.1 Microsoft Visual Studio .NET 2003

Microsoft Visual Studio .NET 2003 delivers the developer productivity required to deliver a wide range of professional software applications. Its *Integrated Development Environment* (IDE) provides a consistent interface for all languages, including *Microsoft Visual Basic .NET*, *Microsoft Visual C++ .NET*, *Microsoft Visual C# .NET*, and *Microsoft Visual J# .NET*. Developers can take advantage of shared visual designers

to design and build rich Windows-based applications and dynamic Web applications that render in any browser (Microsoft Corporation 2004*f*).

2.4.2 Microsoft ASP.NET Web Matrix

Microsoft ASP.NET Web Matrix allows developers to write ASP.NET code productively and efficiently. Designed and developed by some of the team that created ASP.NET, it has been completely written using the Microsoft .NET Framework and Microsoft C# .NET (Microsoft Corporation 2004*b*). Although it is not as powerful and diverse as Microsoft Visual Studio .NET 2003, it offers subtle advantages.

The *Matrix Class Browser* is one of these advantages as it lists all the classes (along with their properties, methods, and events) within the Microsoft .NET Framework.

2.4.3 Microsoft Windows Server 2003

Microsoft Windows Server 2003 is a multipurpose operating system capable of handling a diverse set of server roles. Some of these server roles include the ability to be a Web server with Web application services and a mail server. Microsoft Windows Server 2003 is the first operating system to completely integrate the Microsoft .NET Framework on installation (Microsoft Corporation 2002*b*).

2.4.4 Microsoft Internet Information Service 6.0

Internet Information Services 6.0 (IIS) is a powerful Web server which is included with all Microsoft Windows Server 2003 editions. It provides a highly reliable, manageable, scalable, and secure Web application infrastructure and environment. IIS allows developers to quickly and easily deploy Web sites as well as providing a high-performance platform for applications built using Microsoft ASP.NET and the Microsoft .NET Framework (Microsoft Corporation 2003*d*).

2.4.5 Microsoft SQL Server 2000 and 2005

Microsoft SQL Server 2000 is an industry leader in database development and deployment. Its speed and efficiency in adding, deleting, altering, and searching data has made its mark in the *Database Management System* (DBMS) industry (Microsoft Corporation 2002*c*).

The *Microsoft Desktop Engine 2000* (MSDE) is a data engine built and based on core Microsoft SQL Server 2000 technology for database and application developers. It is a reliable storage engine and query processor for desktop extensions of enterprise applications. The common technology base shared between Microsoft SQL Server 2000 and MSDE allows developers to build applications that can scale seamlessly from portable computers to multiprocessor clusters. An example is the Microsoft SQL Server 2000 Enterprise Edition database server (Microsoft Corporation 2003*c*).

Microsoft has recently released a technical preview beta version of their upcoming *Microsoft SQL Server 2005*. This technical preview has been released as an *Express Edition* which is the equivalent to MSDE, described above, with its core engine their primary difference. The SQL Server 2005 engine offers (Microsoft Corporation 2003*b*):

- *.NET Framework Hosting* - Developers will be able to create database objects using languages such as Microsoft Visual C# .NET.
- *XML Technologies* - Native support for storage and query of XML documents.
- *Transact-SQL Enhancements* - Including error handling and recursive query capabilities.

2.4.6 Microsoft Developer Network

Throughout the history of Microsoft, a very important and vital tool has been offered (free of charge) to all public developers. The *Microsoft Developer Network* (MSDN) offers a substantial information database for Microsoft products and *Application Programming Interfaces* (APIs), including Windows, MS-DOS, DirectX, SQL Server, and

recently, the .NET Framework. This information is offered in the form of white-papers, case studies, source code examples, and class interface descriptions.

Microsoft makes this library available on disc and on the Internet (Microsoft Developer Network 2004). With this information, programmers can develop their software so that it works within the Windows platform.

As an extension to MSDN, Microsoft offers discussion newsgroups where Microsoft employees, Microsoft's *Most Valuable Professionals* (MVPs), and volunteers offer their advice and help in the situations where developers encounter difficulties or problems.

2.5 Software Project Implementations

Throughout the software industry, there are three implementation types:

- Executable Applications
- Web Site
- Web Applications

Executable Applications

An *Executable Application* is an executable file that can be a versatile, dynamically formed computer program that allows for full interaction with data. These programs are subject to acute synchronisation errors when accessing and communicating with databases. Each executable application is platform dependent, meaning that it requires a specific operating system in order to operate correctly. In addition, each computer must have an executable file available to access and manipulate data. An Executable Application can be identified by the file extension `.exe`.

Web Sites

A *Web Site* is a neat and orderly set out page that statically displays data and allows a limited interaction with data. These pages have a restricted range of data manipulation techniques. A Web site can be used via the Internet or an Intranet network by using a third party Web browser and can be identified by the file extensions `.htm` and `.html`.

2.5.1 Web Applications

A *Web Application* is a combination of a Web site and an executable application. This means that Web application can be a neat and orderly laid out page that dynamically displays and manipulates data. It can be used via the Internet or an Intranet network by the use of a third party Web browser. A Web application can be identified by the file extension `.asp`, `.aspx`, `.jsp`, `.cgi`, and `.php`.

Web application's are more complex than they appear. When implementing, they can be divided into a *tier application architecture*, which can range from a *one-tier* to an nth*-tier* application. The most common are *two-tier* and *three-tier* application architectures. Figure 2.8 details these architectures.

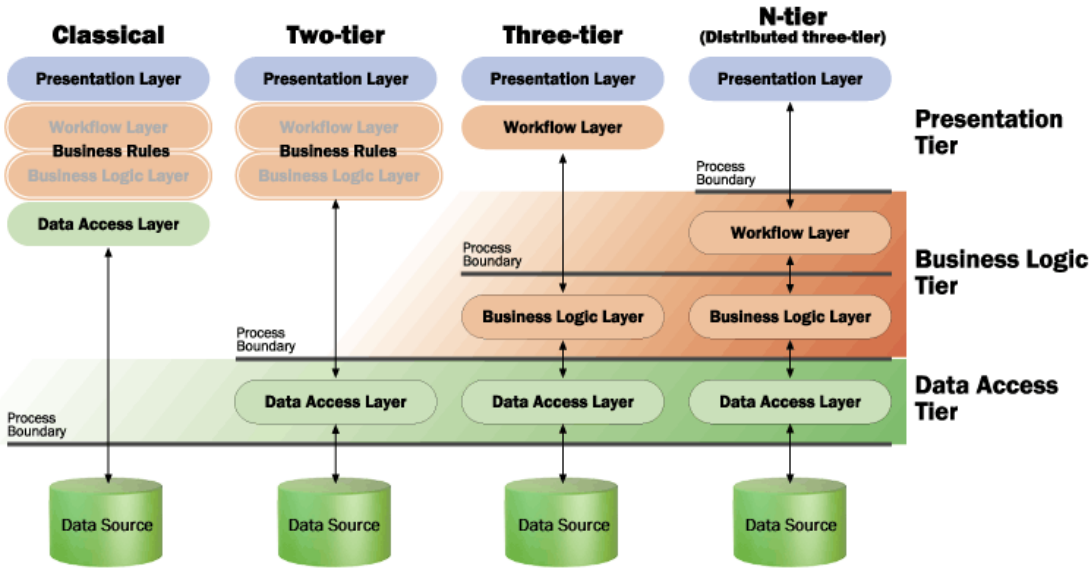


Figure 2.8: The tier application architectural designs

Source: Coleridge (2002, Figure 1)

The business rules component shown above (Figure 2.8) manipulates data from the data source so that other components can use the data. Common to all tier architectures are the different layers each possess. The four layers are:

- *Presentation Layer* - This layer presents data to the user via a user interface.
- *Workflow Layer* - The business rules on the client-side. That is, it controls the user's input and other processes on the client.
- *Business Logic Layer* - The business rules on the server-side. That is, it controls the manipulation and flow of data on the server.
- *Data Access Layer* - Allows the business rules access to the data source.

One-Tier Architecture

The one-tier architecture contains all layers inside the application itself. In a large-scale environment, this type of application would be difficult to maintain. This type of architecture would be acceptable for small systems such as a small retail operation.

Two-Tier Architecture

The two-tier architecture splits the application by creating the *Data Access Layer* (second tier). This allows the programmer to implement an application with multi-user access more easily than with a one-tier system. This type of architecure is suitable for small to medium systems that allows multiple users to access the system at the same time.

Three-Tier Architecture

The three-tier architecture splits the application again by creating the *Business Logic Layer* (second tier) alongside the *Data Access Tier* (third tier). This layer not only contains the code to determine the data's purpose, but also code that determines how and when to manipulate the data.

This architecture was chosen as the back-bone of this project, because it allows:

- Users to interact with the application
- The server to manipulate the data to suit the database
- The database to store and serve data
- Users to access data simultaneously

N-Tier Architecture

The n-tier architecture is very similar to the three-tier architecture. The only difference is, in its decision to split its layers into a greater number, the work load is decreased in different processes. One example is the spliting of the *Data Access Layer* up into an SQL Server database and an XML Web Service.

Chapter 3

Software Design

This chapter describes the design behind the system and database architectures of the OLE project. It also identifies the Microsoft .NET Framework components and

Section 3.1 explains the thought behind the system architecture, while Section 3.2 explains the database architecture. Finally, Section 3.3 describes the different namespaces and members of the Microsoft .NET Framework.

Database Management System (DBMS) 'A software system that enables users to define, create, maintain, and control access to the database.' (Connolly & Begg 2002, p. 16)

3.1 System Architecture

In Chapter 2, specifically Section 2.5.1, the different application architectures were discussed. The OLE utilized the *three-tier* architecture, shown in Figure 3.1, which can also be found and described in the *System Design Document* (Appendix C).

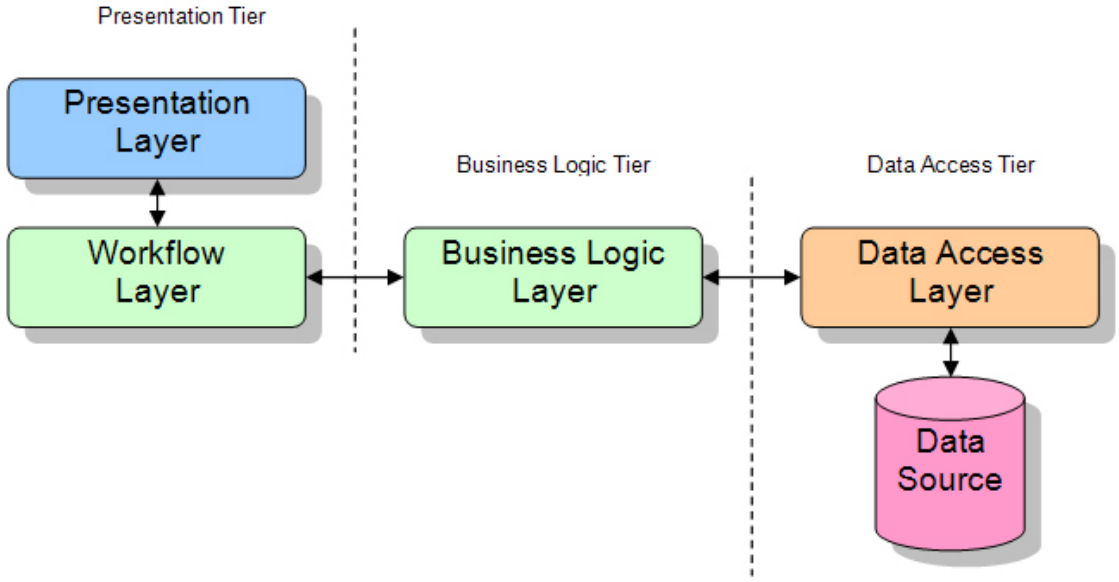


Figure 3.1: The three-tier application architecture

Presentation Tier

This tier (subsystem) consists of two layers. They are the *Presentation* and *Workflow* layers. The *Presentation* layer consists of code that presents the data to the user. The languages used under this layer were HTML and CSS.

The *Workflow* layer consists of code that creates and manipulates data to fit within the *Presentation* layer. The language used was combination of C# .NET and ASP.NET

Business Logic Tier

This tier involves a layer called the *Business Logic* layer. This layer consists of classes that are used within the OLE. The language used here was C# .NET.

Data Access Tier

This tier consists of the *Data Access* layer. It provides classes that can realise the storage, retrieval, and query of persistent objects within the *Data Source*. The languages used here were ADO.NET and SQL.

3.1.1 System-to-Data Relationship

Since the OLE was also a research project, a hybrid method was taken towards the relationship between the system and data. Two methods were used to manage data utilized, specifically XML files and an SQL Server database.

In order to create this hybrid method, a `sealed` class was designed to offer properties and methods relating to both SQL and XML data management. The name given to this class reflected its purpose; to offer a bridge between the system and data. The name, `DataBridge` was created.

Since SQL is very broad in its method calls, its connection class was offered as a property, so that its calls can be manually invoked. XML was used quite differently. It held two properties that gave the class and institution directories. These directories sorted the XML files. The class also offered many methods that returned *next files' paths* and *existing files' paths*.

3.1.2 Data Storage Techniques

Two types of data storage techniques were used within the OLE project. These two techniques are the most used methods within the development industry. They are:

- SQL databases; and
- XML files.

In order for these two techniques to exist within the project, a programming model that utilized these two techniques was required.

Section 3.2 describes the SQL database's design, but not its connection to the XML file design. It will be noted here that within *Section 3.4* within the SDD (Appendix C), the design and structure of the XML file and SQL database is documented. The SQL database is used for effeciently recovering data, since the database only holds superficial and important information (i.e. small amounts of information or information that will constantly be accessed). The XML files hold all of the information so that both techniques do not have to be called in order to get retrieve the requested information.

An example:

> An assignment will have an identifier (ID), a due date, and a question to accompany it (simplified example). The SQL database will hold the:
>
> - *ID* - A unique number to create a link between programming calls.
> - *Due Date* - A small value that will be constantly called apon.
>
> The XML file will hold all the values so:
>
> - *ID* - As seen above.
> - *Due Date* - Holds and the SQL database does not have to be called in order to retrieve this value.
> - *Question* - Is a large field that can be any size.

There are two main reasons why this programming model was designed. They are:

- To learn both data storage techniques; and
- As a way to lessen the complication of the SQL database since they can only hold up to 8000 characters within a `VARCHAR` field.

3.2 Database Architecture

Before a database can be completed, it must pass through a number of design steps. Connolly & Begg (2002, pp. 420-421) describes the database design methodology, which was used in order to generate a database for the OLE. This methodology is divided into three main sections, each containing steps:

- *Conceptual Database Design* - 'The process of constructing a model of the information used in an enterprise, independent of all physical considerations.' (Connolly & Begg 2002, p. 419)
  - *Step 1* - Build a local conceptual data model.
- *Logical Database Design* - 'The process of constructing a model of the information used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.' (Connolly & Begg 2002, p. 419)
  - *Step 2* - Build and validate a local logical data model.
  - *Step 3* - Build and validate a global logical data model.
- *Physical Database Design* - 'The process of producing a description of the implementation of the database on secondary storage...' (Connolly & Begg 2002, p. 419)
  - *Step 4* - Translate the global logical data model.

The steps shown above also split into sub-steps. Figure 3.2 display these steps in further detail, however, only steps 1 and 2 will be described.

Step 1 - Build a local conceptual data model

- Step 1.1 - Identify entity types
- Step 1.2 - Identify relationship types
- Step 1.3 - Identify and associate attributes with entity or relationship types
- Step 1.4 - Determine attribute domains
- Step 1.5 - Determine candidate and primary key attributes

Step 2 - Build and validate a local logical data model

- Step 2.1 - Derive relations for local logical data model
- Step 2.2 - Validate relations using normalization
- Step 2.3 - Define integrity constraints

Figure 3.2: The steps in the Database Design Methodology (Steps 1 to 2)

This section will follow through the described steps, informing how the OLEs database was generated.

Step 1.1 - Identify entity types

Connolly & Begg (2002, p. 331) defines an *entity type* as, 'a group of objects with the same properties, which are identified by the enterprise as having an independent existence'. Using this definition, the OLE has the following entity types:

- *Users* - An individual that contributes to the system.
- *Classes* - A course for users to enrol in and contribute.
- *Roles* - Three roles that define a user's type of contributions.
- *AETQuestions* - An Assignment, Exam, or Tutorial question within a class.
- *AETAnswers* - An answer to an AETQuestion
- *ThreadSubscriptions* - A subscription to a forum thread, defined by the user.

Step 1.2 - Identify relationship types

Connolly & Begg (2002, p. 334) defines a *relationship type* as, 'a set of meaningful associations among entity types'. With this definition, the OLE has the following relationship types:

Users (1..1) Creates ▷ (0..*) AETQuestions

*A **User** can create zero-to-many **AETQuestion**(s).*

*An **AETQuestion** can have one **User**.*

Users (0..*) Maintains ▷ (0..*) Classes

*A **User** can maintain zero-to-many **Class**(es).*

*A **Class** can have zero-to-many **User**(s).*

Users (0..*) Supervises ▷ (0..*) Users

*A **User** can supervise zero-to-many **User**(s).*

*A **User** can be supervised by zero-to-many **User**(s).*

Users (1..1) Owns ▷ (0..*) AETAnswers

*A **User** can own zero-to-many **AETAnswer**(s).*

*An **AETAnswer** can have one **User**.*

Users (0..*) Has ▷ (0..*) ThreadSubscriptions

*A **User** can have zero-to-many **ThreadSubscription**(s).*

*A **ThreadSubscription** can have zero-to-many **User**(s).*

Classes (0..*) Has ▷ (0..*) Users

*A **Class** can have zero-to-many **User**(s).*

*A **User** can have zero-to-many **Class**(es).*

Classes (1..1) Has ▷ (0..*) AETQuestions

*A **Class** can have zero-to-many **AETQuestion**(s).*

*An **AETQuestion** can have one **Class**.*

Roles (0..3) Assigns ▷ (0..*) Users

*A **Role** can assign zero-to-many **User**(s).*

*A **User** can have zero-to-many **Role**(s).*

Step 1.3 - Identify and associate attributes with entity or relationship types

Connolly & Begg (2002, p. 338) defines an *attribute* as, 'a property of an entity or a relationship type'. With this definition, Table 3.1, 3.2, and 3.3 will detail the attributes that were used within each entity.

Table 3.1: The attributes within the entity types *Users*, *AETQuestions*, and *AETAnswers*

Users	AETQuestions	AETAnswers
userID	aetQID	aetQID
firstName	classID	classID
middleName	aetQType	aetQType
lastName	aetQTitle	userID
password	aetQDescription	aetASubmittedDate
dob	aetQQuestionType	aetAMarked
postalStreet	aetQViewQuestionDateTime	
postalSuburb	aetQDueDateTime	
postalState	aetQHaltSubmissionDateTime	
postalPostcode	aetQViewMarkedAnswersDateTime	
postalCountry	aetQAmountOfQuestions	
phoneNumber		
mobileNumber		
emailAddress		

Table 3.2: The attributes within the entity types *Classes*, *Roles*, and *ThreadSubscriptions*

Classes	Roles	ThreadSubscriptions
classID	rolesID	classID
className		threadID
denyOtherUsers		classID

Table 3.3: The attributes within the entity types *ClassUsers* and *OtherUsers*

ClassUsers	OtherUsers
classID	userID
userID	roleID
roleID	

Step 1.4 - Determine attribute domains

Connolly & Begg (2002, p. 338) defines an *attribute domain* as, 'the set of allowable values for one or more attributes'. These values are defined by SQL data types:

- `BIGINT` (BI) - An integer with a set character length of 8.
- `BIT` (B) - Equivilent to a boolean value with a set character length of 1.
- `CHAR` (C) - A variable character length that stores values of a specific length.
- `DATETIME` (DT) - A date and time value with a set character length of 8.
- `INT` (I) - An integer with a set character length of 4.
- `VARCHAR` (VC) - A variable character length that stores values of a variable length.

Tables 3.4, 3.5, and 3.6 lists the values beside each attribute within [ ] parenthesises. For example, a `VARCHAR` of character length 50, for the attribute `firstName`, would be [VC50].

Table 3.4: The attribute domains for the attributes in Table 3.1

Users	AETQuestions	AETAnswers
userID [I]	aetQID [BI]	aetQID [BI]
firstName [VC50]	classID [C7]	classID [C7]
middleName [VC50]	aetQType [VC20]	aetQType [VC20]
lastName [VC50]	aetQTitle [VC100]	userID [I]
password [C47]	aetQDescription [VC30]	aetASubmittedDate [DT]
dob [DT]	aetQQuestionType [VC20]	aetAMarked [B]
postalStreet [VC50]	aetQViewQuestionDateTime [DT]	
postalSuburb [VC50]	aetQDueDateTime [DT]	
postalState [VC50]	aetQHaltSubmissionDateTime [DT]	
postalPostcode [VC10]	aetQViewMarkedAnswersDateTime [DT]	
postalCountry [VC50]	aetQAmountOfQuestions [I]	
phoneNumber [VC15]		
mobileNumber [VC15]		
emailAddress [VC50]		

Table 3.5: The attribute domains for the attributes in Table 3.2

Classes	Roles	ThreadSubscriptions
classID [C7]	rolesID [VC50]	classID [C7]
className [VC50]		userID [I]
denyOtherUsers [B]		threadID [I]

Table 3.6: The attribute domains for the attributes in Table 3.3

ClassUsers	OtherUsers
classID [C7]	userID [I]
userID [I]	roleID [VC50]
roleID [VC50]	

Step 1.5 - Determine candidate and primary key attributes

Within a database entity, there are three types of key attributes. They are:

- *Candidate Key* - 'The minimal set of attributes that uniquely identifies each occurrence of an entity type.' (Connolly & Begg 2002, p. 340)
- *Primary Key* - 'The candidate key that is selected to uniquely identify each occurrence of an entity type.' (Connolly & Begg 2002, p. 341)
- *Composite Key* - 'A candidate key that consists of two or more attributes.' (Connolly & Begg 2002, p. 341)

With these definitions, Tables 3.7, 3.8, and 3.9 will highlight the primary and composite keys from the set attributes detailed in Tables 3.1, 3.2, and 3.3. Primary keys are defined by having a **{PK}** notation beside an attribute. A composite key is defined by having more than one attribute with the **{PK}** notation.

Table 3.7: The highlighted and created primary and composite keys from Table 3.1

Users	AETQuestions	AETAnswers
userID **{PK}**	aetQID **{PK}**	aetQID **{PK}**
firstName	classID **{PK}**	classID **{PK}**
middleName	aetQType **{PK}**	aetQType **{PK}**
lastName	aetQTitle	userID **{PK}**
password	aetQDescription	aetASubmittedDate
dob	aetQQuestionType	aetAMarked
postalStreet	aetQViewQuestionDateTime	
postalSuburb	aetQDueDateTime	
postalState	aetQHaltSubmissionDateTime	
postalPostcode	aetQViewMarkedAnswersDateTime	
postalCountry	aetQAmountOfQuestions	
phoneNumber		
mobileNumber		
emailAddress		

Table 3.8: The highlighted and created primary and composite keys from Table 3.2

Classes	Roles	ThreadSubscriptions
classID **{PK}**	rolesID **{PK}**	classID **{PK}**
className		userID **{PK}**
denyOtherUsers		threadID **{PK}**

Table 3.9: The highlighted and created primary and composite keys from Table 3.3

ClassUsers	OtherUsers
classID **{PK}**	userID **{PK}**
userID **{PK}**	roleID **{PK}**
roleID **{PK}**	

Step 2.1 - Derive relations for local logical data model

Within a database , there are two types of entities. They are:

- *Strong Entity Types* - 'An entity type that is not existence-dependent on some other entity type.' (Connolly & Begg 2002, p. 342)
- *Weak Entity Types* - 'An entity type that is existence-dependent on some other entity type.' (Connolly & Begg 2002, p. 343)

From these definitions, the strong and weak entity types within the database will be defined. Once defined, the relationships between entities will be detailed.

Strong Entity Types:

Users (userID, ...)

Primary Key userID

Classes (classID, ...)

Primary Key classID

Roles (roleID)

Primary Key roleID

Weak Entity Types:

AETAnswers (aetQID, classID, aetQType, userID, ...)

Alternate Key aetQType

Foreign Key aetQID

Foreign Key classID

Foreign Key userID

AETQuestions (aetQID, classID, aetQType, ...)

Primary Key aetQID

Alternate Key aetQType

Foreign Key classID

ClassUsers (classID, userID, roleID)

Foreign Key classID

Foreign Key userID

Foreign Key roleID

OtherUsers (userID, roleID)

Foreign Key userID

Foreign Key roleID

ThreadSubscriptions (threadID, classID, userID)

Primary Key threadID

Foreign Key classID

Foreign Key userID

Relationships:

AETAnswers (aetQID, classID, aetQType, userID, ...)

Foreign Key aetQID **references** AETQuestions(aetQID)

Foreign Key classID **references** Classes(classID)

Foreign Key userID **references** Users(userID)

AETQuestions (aetQID, classID, aetQType, ...)

Foreign Key classID **references** Classes(classID)

ClassUsers (classID, userID, roleID)

Foreign Key classID **references** Classes(classID)

Foreign Key userID **references** Users(userID)

Foreign Key roleID **references** Roles(roleID)

OtherUsers (userID, roleID)

Foreign Key userID **references** Users(userID)

Foreign Key roleID **references** Roles(roleID)

ThreadSubscriptions (threadID, classID, userID)

Foreign Key classID **references** Classes(classID)

Foreign Key userID **references** Users(userID)

Step 2.2 - Validate relations using normalization

Connolly & Begg (2002, p. 376) defines a *normalization* as, 'a technique for producing a set of relations with desirable properties given the data requirements of an enterprise'. With this definition, the entities within the OLE are already in its maximum *normal form* (with the exception of *AETAnswers(aetQType)* and *AETQuestions(aetQType)*).

Step 2.3 - Define integrity constraints

This step will define the referential integrity between entities within the OLE. That is, how records are updated/deleted when a referenced record is updated/deleted.

Relationships:

AETAnswers (aetQID, classID, aetQType, userID, ...)

Foreign Key aetQID **references** AETQuestions(aetQID)

ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key classID **references** Classes(classID)

ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key userID **references** Users(userID)

ON UPDATE CASCADE, ON DELETE CASCADE

AETQuestions (aetQID, classID, aetQType, ...)

Foreign Key classID **references** Classes(classID)

ON UPDATE CASCADE, ON DELETE CASCADE

ClassUsers (classID, userID, roleID)

Foreign Key classID **references** Classes(classID)

ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key userID **references** Users(userID)

ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key roleID **references** Roles(roleID)

ON UPDATE CASCADE, ON DELETE CASCADE

OtherUsers (userID, roleID)

Foreign Key userID **references** Users(userID)

ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key roleID **references** Roles(roleID)

ON UPDATE CASCADE, ON DELETE CASCADE

ThreadSubscriptions (threadID, classID, userID)

Foreign Key classID **references** Classes(classID)

ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key userID **references** Users(userID)

ON UPDATE CASCADE, ON DELETE CASCADE

3.3 Design using the Microsoft .NET Framework

This section describes the main .NET Framework components (classes, enumerations, and interfaces) that were used in the OLEs implementation. Microsoft Developer Network (2003) describes the main namespaces that were used in the OLE:

- `Microsoft.Win32` - 'The Microsoft.Win32 namespace provides two types of classes: those that handle events raised by the operating system and those that manipulate the system registry.' (Microsoft Developer Network 2003, p. cpref/html/frlrfMicrosoftWin32.htm)
- `System` - 'The System namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystem.htm)
- `System.Collections` - 'The System.Collections namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hashtables and dictionaries.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemCollections.htm)
- `System.Data` - 'The System.Data namespace consists mostly of the classes that constitute the ADO.NET architecture. The ADO.NET architecture enables you

to build components that efficiently manage data from multiple data sources.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemData.htm)

- `System.Data.SqlClient` - 'The System.Data.SqlClient namespace is the .NET Framework Data Provider for SQL Server.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemDataSqlClient.htm)
- `System.IO` - 'The System.IO namespace contains types that allow reading and writing to files and data streams, and types that provide basic file and directory support.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemIO.htm)
- `System.Security.Cryptography` - 'The System.Security.Cryptography namespace provides cryptographic services, including secure encoding and decoding of data, as well as many other operations, such as hashing, random number generation, and message authentication.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemSecurityCryptography.htm)
- `System.Security.Principal` - 'The System.Security.Principal namespace defines a principal object that represents the security context under which code is running.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemSecurityPrincipal.htm)
- `System.Text` - 'The System.Text namespace contains classes representing ASCII, Unicode, UTF-7, and UTF-8 character encodings; abstract base classes for converting blocks of characters to and from blocks of bytes; and a helper class that manipulates and formats String objects without creating intermediate instances of String.' (Microsoft Developer Network 2003,)
- `System.Xml` - 'The System.Xml namespace provides standards-based support for processing XML.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemXml.htm)
- `System.Xml.Serialization` - 'The System.Xml.Serialization namespace contains classes that are used to serialize objects into XML format documents or streams.' (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemXmlSerialization.htm)

The following classes, enumerations, and interfaces were used:

- `Microsoft.Win32.RegistryKey` *class* - Opens access to the Win32 registry and encapsulates any or new registry keys, allowing alterations, deletions, and creations.

- `System.BitConverter` *class* - Converts base data types to an array of bytes and also converts an array of bytes to base data types.

- `System.Collections.ArrayList` *class* - Offers methods and properties that manipulate object arrays. That is, the inserting, removing, and relocating of objects within the list.

- `System.Data.CommandType` *enumeration* - Lists the types of commands that can be executed by an `System.Data.SqlClient.SqlCommand` class.

- `System.Data.SqlDbType` *enumeration* - Lists the data types that exist within an SQL Server database (e.g. `CHAR`, `VARCHAR`, `INT`, etc.).

- `System.Data.SqlClient.SqlConnection` *class* - Creates a connection to an existing SQL Server database server when supplied a correct connection string.

- `System.Data.SqlClient.SqlCommand` *class* - The ability to execute query and non-query commands through an `System.Data.SqlClient.SqlConnection` connection.

- `System.Data.SqlClient.SqlDataAdapter` *class* - The ability to read data which is retured from an `System.Data.SqlClient.SqlCommand` command execution.

- `System.IO.Directory` *class* - Creates, deletes, and renames a directory within the local fileystem. It can also return a array of files that exist within the directory.

- `System.IO.File` *class* - Creates, deletes, and renames a file within the local fileystem. It can also provide detailed information about the file.

- `System.Security.Cryptography.MD5CryptoServiceProvider` *class* - Converts a `string` to an MD5 encrypted `byte[]`.

- `System.Security.Principal.IPrincipal` *interface* - Provides methods and properties that define a .NET Framework user. The interface is used in many ASP.NET authentication processes.

- `System.Text.ASCIIEncoding` *class* - Converts an ASCII character set to other character encodings.
- `System.Xml.XmlDocument` *class* - The ability to load an XML file and manipulate its contents.
- `System.Xml.XmlTextReader` *class* - The ability to read through an XML file.
- `System.Xml.XmlTextWriter` *class* - The ability to write an XML file.
- `System.Xml.Serialization.XmlSerializer` *class* - The ability to convert attributed class properties into an XML document, using the `Serialize()` method. This process can also be reversed by using the `Deserialize()` method
- `System.Xml.Serialization.XmlArrayAttribute` *class* - Defines a property as an XML node with items (`System.Xml.Serialization.XmlArrayItem`).
- `System.Xml.Serialization.XmlArrayItem` *class* - Defines an array property's items as XML nodes within the `System.Xml.Serialization.XmlArrayAttribute` node.
- `System.Xml.Serialization.XmlTextAttribute` *class* - Defines a property to exist as text within an XML element.
- `System.Xml.Serialization.XmlRootAttribute` *class* - Defines a class to be the root XML node of an XML document.
- `System.Xml.Serialization.XmlElementAttribute` *class* - Defines a property to be an XML element under the XML root node.
- `System.Xml.Serialization.XmlAttributeAttribute` *class* - Defines a property to be an attribute of an XML element.

Chapter 4

Software Implementation

The implementation of the OLE project involved the use of all programming languages listed under Section 2.3. However, the two most significant programming languages were ASP.NET and XML.

This chapter will start by explaining some of the problematic issues revolving around the project's implementation, in Section 4.1. Section 4.2 will then explain some of the decisions behind the implementation using ASP.NET, while Section 4.3 will explain the decisions made using XML.

Windows Registry 'The registry acts as a central repository of information for the operating system and the applications on a computer. The registry is organized in a hierarchical format, based on a logical ordering of the elements stored within it...' (Microsoft Developer Network 2003, cpref/html/frlrfmicrosoftwin32 registrykeyclasstopic.htm)

Win32 Microsoft Windows in 32-bit executable code.

4.1 Implementation Issues

This section contains some implementation issues that could not be resolved, researched (due to lack of time), or implemented due to the lack of support of features within the software.

4.1.1 XML in Microsoft SQL Server 2000

In Section 2.3.5, Microsoft SQL Server's support for XML was discussed. SQL Server 2000 however, does not support an XML document formed by the SQL data-type, `BINARY` (formed by a stored procedure using `FOR XML EXPLICIT` mode). SQL Server 2005 provides native support for XML, allowing this problem to be bypassed.

Originally, a user's password was hashed into a C# `byte[]` data-type so that it could be stored in an SQL `BINARY` field. Instead, the C# `byte[]` data-type was converted to a string of hexadecimal values and then stored in a an SQL `CHAR` field. This was decided since SQL Server 2000's support for XML was not as required and SQL Server 2005 had not been completely developed by Microsoft.

4.1.2 Registry and File System Permissions

During the implementation process of the setup application for the OLE project, a few issues were encountered during post-installation testing of the Web application. It was noticed that if the data files were created under a directory without full permissions given to *Everyone* (a default NTFS user, defining every user under the operating system), files were unable to be created, altered, or removed. Read-only access was the only permission given. This was also consistent with the values within the Windows registry.

4.1.3 Native XML Architecture

The choice in only implementing a small portion of XML's features and capabilities was based on the time required to research against the time required for all the other activities within the project. However, foundation knowledge for all of XML's features and capabilities was researched.

4.2 Implementation using ASP.NET

This section describes the primary impacts on the OLE project's implementation while using ASP.NET. Some of these implementation impacts were discovered while implementing ASP.NET authentication, and creating Web forms and user controls.

4.2.1 Authentication

Authentication under ASP.NET works differently to most other programming languages. A number of procedures must have been completed in order for ASP.NET authentication to be enabled. J. D. Meier & Vasireddy (2002*a*) defines these procedures in its *How-To* article. The main steps as provided in this article are:

- *Step 1* - Configure the Web application for Forms Authentication.
- *Step 2* - Generate an Authentication Ticket for the authenticated users.
- *Step 3* - Create a class that implements the IPrincipal interface.
- *Step 4* - Create the CustomPrincipal object.

The obvious reason for Microsoft to publish an article such as this one is to ensure that ASP.NET is practiced the way it was designed to be. It also allows ASP.NET developers to move from one ASP.NET application to another without having to learn a the previous developers' implementation.

Step 1 - Configure the Web application for Forms Authentication

This step involves the configuration of the Web.config source code to enable *Forms Authentication*. In essence, the only change required is the setting of the *Uniform Resource Locator* (URL) of the user login page, changing the <`authentication`> element's `mode` attribute to `Forms`, and defining an XML element under the <`authentication`> node called <`deny`>, with its `users` attribute set as `?`.

This step's implementation can be found within Listing H.22, between lines 6 and 11.

Step 2 - Generate an Authentication Ticket for the authenticated users

This step involves the addition of a `bool` valued variable called `isAuthenticated`. A method is called to check for a user's valid password, and its result is returned to `isAuthenticated` (successful as `true` and unsuccessful as `false`). On the success of a user's authentication, a `System.Web.Security.FormsAuthenticationTicket` is created with the user's details, encrypted, and stored as a Web cookie on the user's computer. The user is then redirected to the page the requested before being redirected to the login page.

This step's implementation can be found within Listing H.62, between lines 39 and 83.

Step 3 - Create a class that implements the IPrincipal interface

This step involves the integration of the `IPrincipal` interface with the OLE's `User` class. The integration consists of a method, `bool IsInRole(string)`, and a property, `System.Security.Principal.IIdentity Identity`. The `IsInRole()` method is used by ASP.NET to identify the user's assigned roles (i.e. Student, Instructor, and/or Administrator). The `Identity` property is made up of the decrypted cookie, created by the previous step.

The roles that the user is assigned to can be a problem. For example, a user could have a role Student, but a student of which class? This problem is overcome by invoking a

method that confirms their role status for a particular class when a class function is accessed.

This step's implementation can be found within Listing H.106, between lines 21 and 58.

Step 4 - Create the CustomPrincipal object

Creating a `IPrincipal` object involves the implementation of the `User` object, created from the `FormsAuthenticationTicket` in Step 2, within the application authentication event handler. The event handler is initiated by decrypting the cookie kept on the user's computer. If this cookie is not found, the authentication process is cancelled and the user must be asked for their authentication details once again. However, if the cookie is found and has not expired, the user will then be redirected to the page they first requested.

This step's implementation can be found within Listing H.11, between lines 86 and 134.

4.2.2 Web Forms and User Controls

Within the OLE project, a Web form is a subsystem, and a Web user control was a component of its respective subsystem. For example, the menu is a component that exists within every subsystem. So the menu was implemented as a Web user control.

This created a powerful implementation, allowing the Web user controls to be reused, rather than having to be recoded on each Web form.

4.3 Implementation using XML

This section details some examples of the XML data structures that were used within the OLE project in order to store data. It also explains how these data structures were assembled and disassembled using the .NET Framework.

4.3.1 XML Data Structures and Database Compatibility

The data structures designed in the OLE project were based on the tables inside the SQL Server database's tables. These tables then formed the classes inside the *Business Logic* tier of the application. The reason for this was to ensure a compatibility between the database and Web application.

This compatibility was achieved by the .NET Framework's serialisation and deserialisation class, which is discussed in the following sub-section.

4.3.2 Serialization and Deserialization

Within the Microsoft's .NET Framework, an XML based class known as `XmlSerializer` is found under the `System.Xml.Serialization` namespace. This class offers the ability to 'serialise' a class and its members into an XML document that can then be 'deserialised' into a new instance of that same class.

This method was used to form the XML data structures easily and effectively. It allowed the SQL stored procedures that were converted into XML documents to be easily 'deserialised' into a new instance of its respective class.

4.4 The Incomplete Web Application

This section provides several screenshots of the OLE Web application within Microsoft Internet Explorer 6.0. Since the OLE Web application was not completely implemented, only a few screenshots of the application are shown.

The first screenshot, Figure 4.1, shows the *Login* page for users to access the Web application. Figure 4.2 depicts the *Home* page for user 'Jonathan Trinder' (`userID`: 26). Finally, Figure 4.3 shows the *Class* page, with 'Assignment 1' for class, 'Research Project 1' (`courseID`: ENG4111).

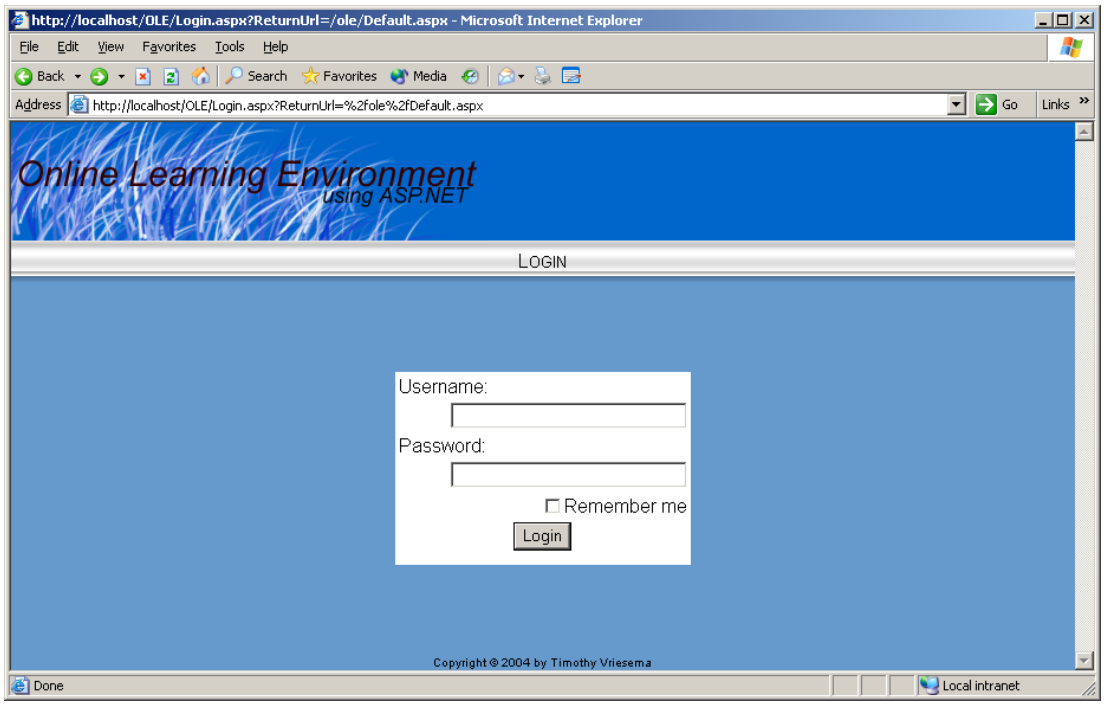


Figure 4.1: *Login* page

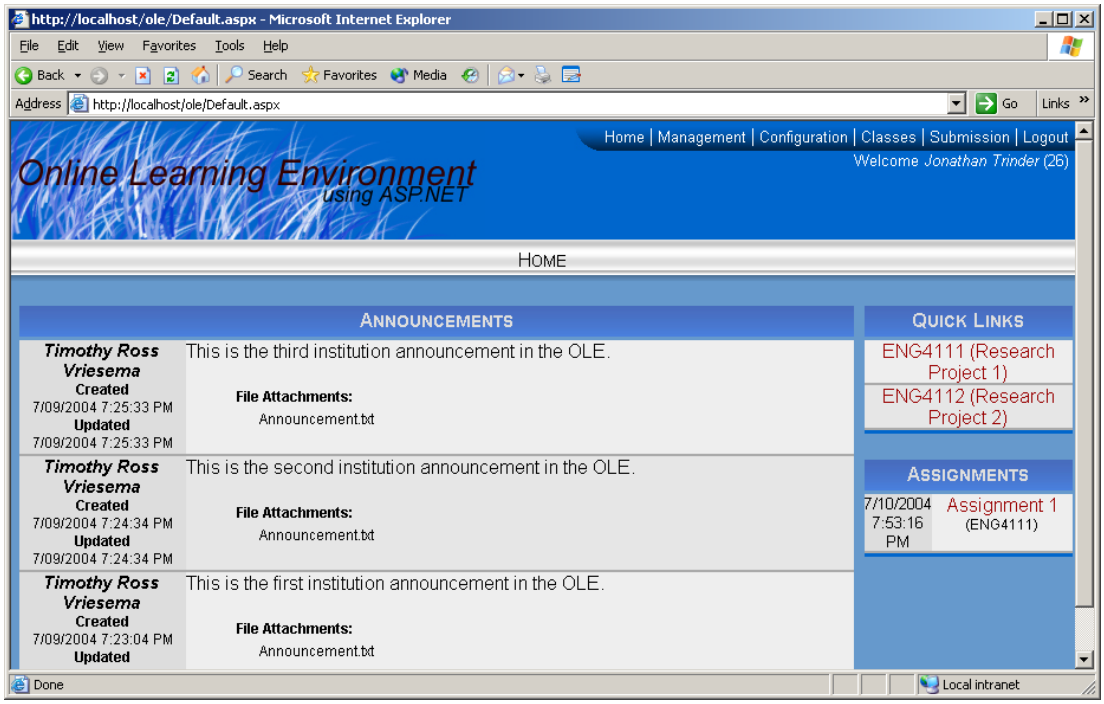


Figure 4.2: *Home* page for user 'Jonathan Trinder' (`userID`: 26)

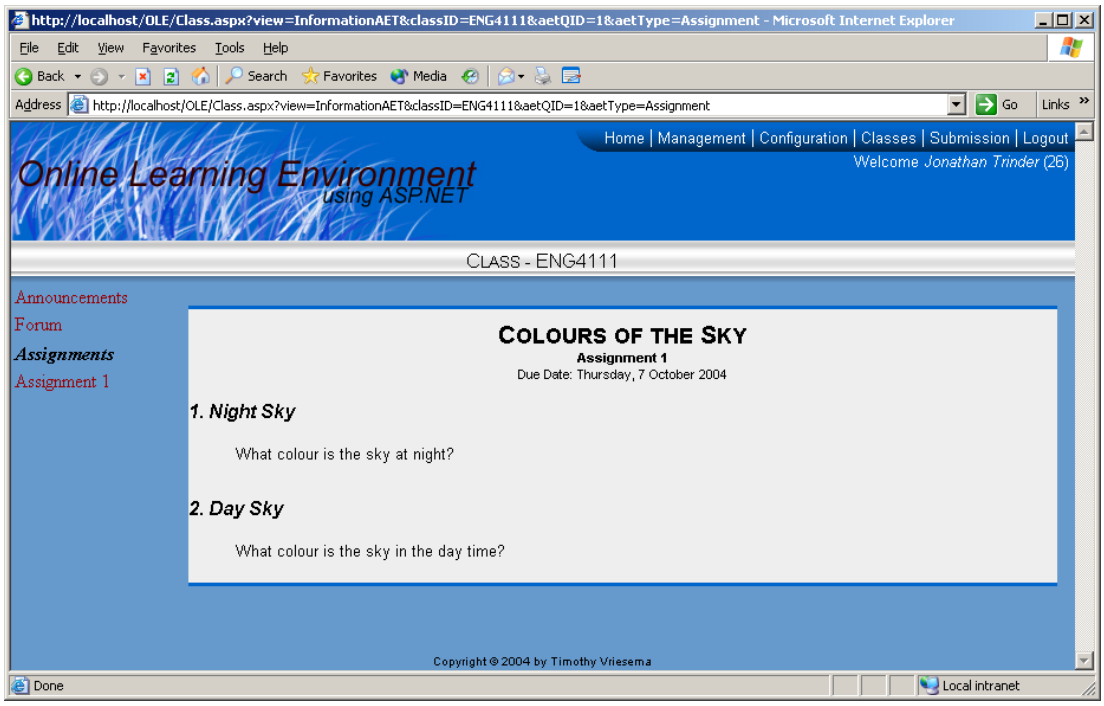


Figure 4.3: 'Assignment 1' for 'Research Project 1' (`courseID`: ENG4111) within the *Class* page

Chapter 5

Software Deployment

This chapter describes the step-by-step procedures required to deploy the 'Online Learning Environment' in a Microsoft Windows Server 2003 environment. It will first display a prerequisite option that is required for the application to run successfully in Microsoft Internet Information Services 6.0 (Section 5.1). The chapter will then explain the installation process of the OLE within Microsoft Windows Server 2003 (Section 5.2).

Virtual Folder A folder that is treated by Microsoft Windows as a normal folder, but does not correspond to a physical directory.

Server/Machine Name A unique name that identifies a computer to the network. A computer can also be referred to as having the name, `localhost`.

5.1 Configuration of Microsoft Windows Server 2003

Within *Windows Server 2003*, ASP.NET support is disabled throughout the operating system (specifically IIS). The OLE requires ASP.NET and therefore ASP.NET support must to be enabled.

In order to enable ASP.NET, navigation to Windows' *Control Panel* occurred, then *Add/Remove Programs* was opened. A button inside this window called the *Windows*

Component Wizard was clicked and then the *Application Server* was selected and the *Details* button was clicked. Figure 5.1 shows this window and the option to check 'enabled'.

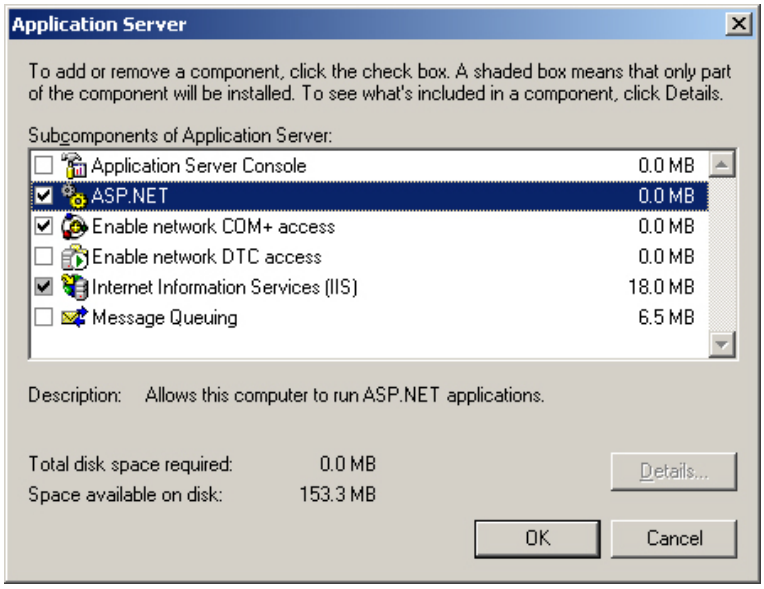


Figure 5.1: The *Application Server* window from the *Windows Component Wizard*, under *Windows Server 2003*

5.2 Installation - The Setup Application

This section describes the steps involved in the installation the OLE. The installation fails when the presence of *Microsoft Internet Information Services 5.0+* (ISS) and *Microsoft .NET Framework 1.0+* is not found. This is by design because the OLE uses both components.

The files required for the setup application to execute, are:

- `Setup.exe` - Required to execute the setup application.
- `OLEWebSetup.msi` - Holds the data and files required for installation. This can be executed on some Windows operating systems without the use of `Setup.exe`. However, using `Setup.exe` to execute the installation is recommended.

- `Setup.ini` - Holds meta-data for the `Setup.exe` file that redirects the user to the Microsoft .NET Framework's download page in the case that the user does not have it installed. It also describes which `.msi` file holds the installation data.

Step 1 - Welcome Screen

Figure 5.2 details the first interactive screen for the user. It is usually shown when all pre-requisites are met. To proceed, the user clicks on *Next*. Otherwise, the user clicks *Cancel*.

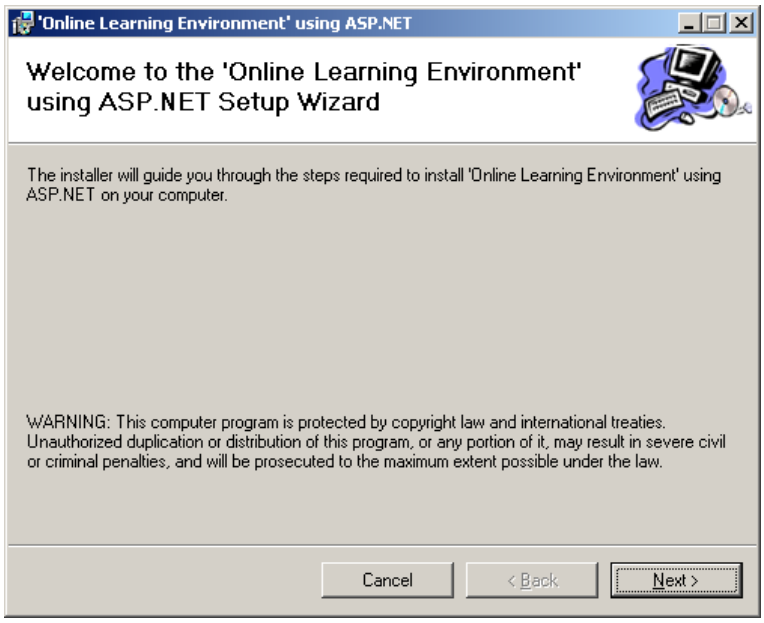


Figure 5.2: Welcome screen (first step)

Step 2 - Configuration

Figure 5.3 details the next screen the user encounters. It contains a textbox that asks for the server name of their SQL Server application. By default, the textbox contains `localhost`, which is the name of the machine in which they executed the setup application.

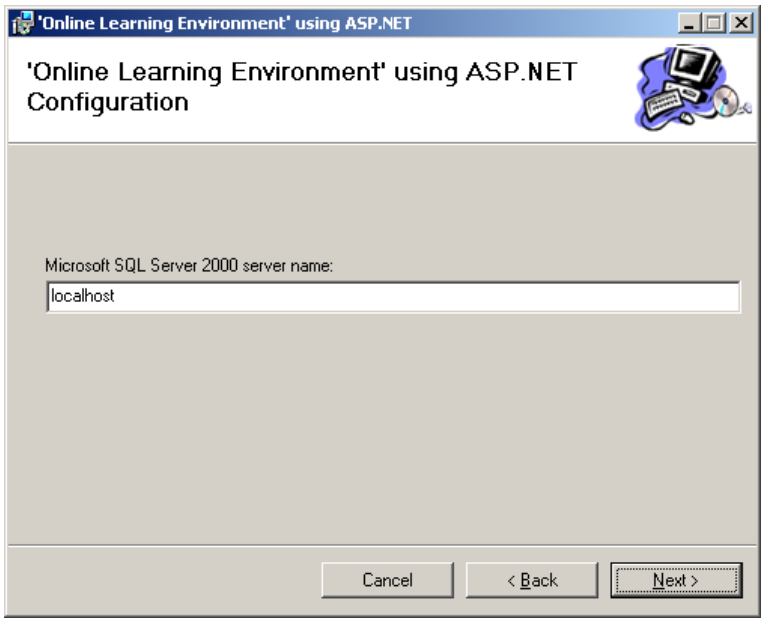


Figure 5.3: SQL Server name setup (second step)

Step 3 - Installation Address

Figure 5.4 details the screen following the previous step. It asks for which virtual folder should the files be copied to. Also, it asks what port the application should set the virtual folder to use.

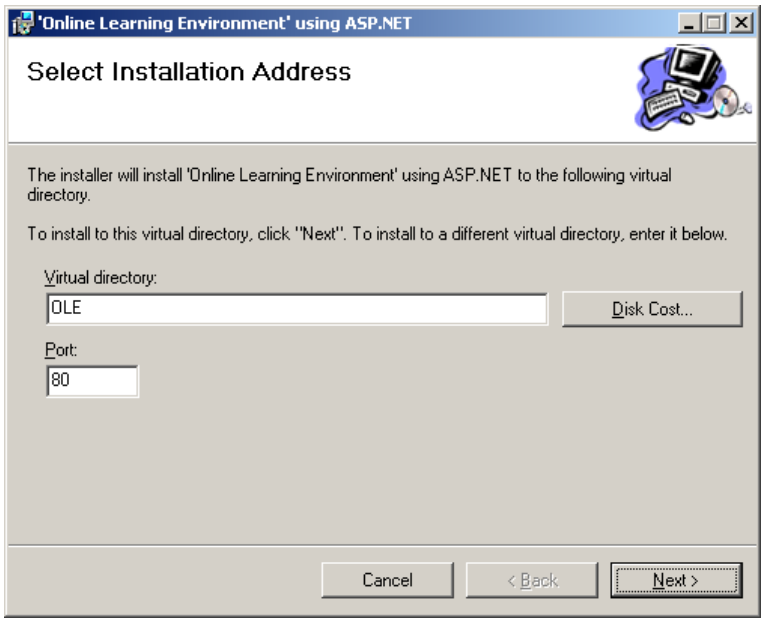


Figure 5.4: IIS virtual folder and port setup (third step)

Step 4 - Installation Confirmation

Figure 5.5 details the screen before installation commences. This screen appears in the occurrence of a mistake by a user, providing the ability to use *Back* so that the settings can be altered.

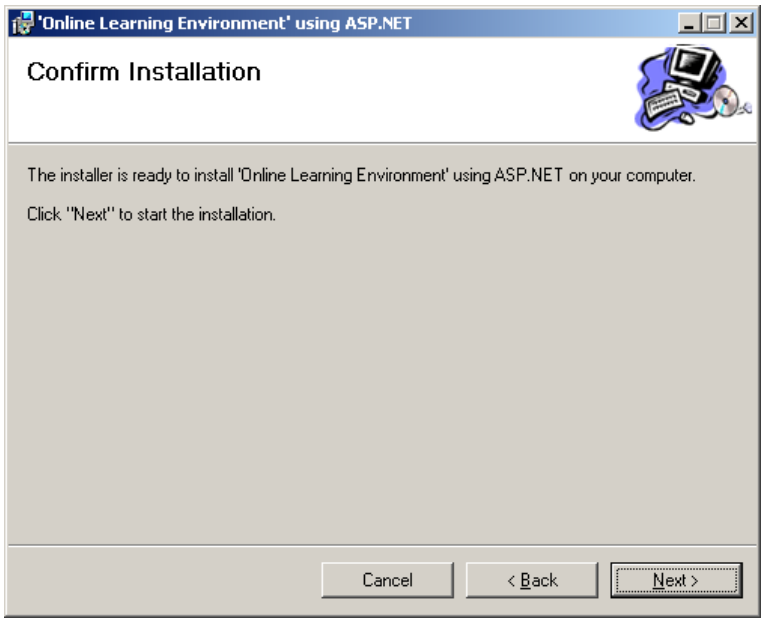


Figure 5.5: Confirm installation (forth step)

Step 5 - Installation Progress

Figure 5.6 details the installation's progress by the use of a progress bar. It continues once the bar has been completely filled.

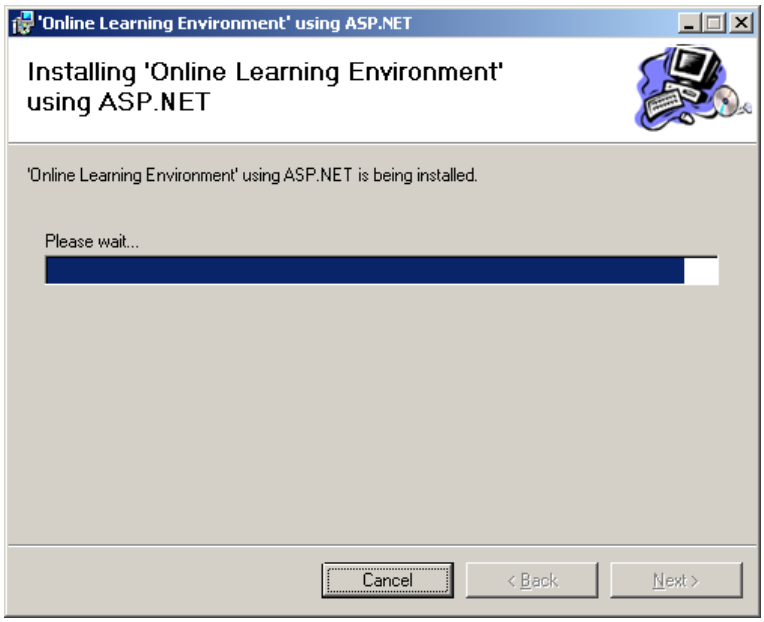


Figure 5.6: Setup installing (fifth step)

Step 6 - Installation Complete

Figure 5.7 details the installation's status. If the installation does not encounter a problem or error, this screen will appear.

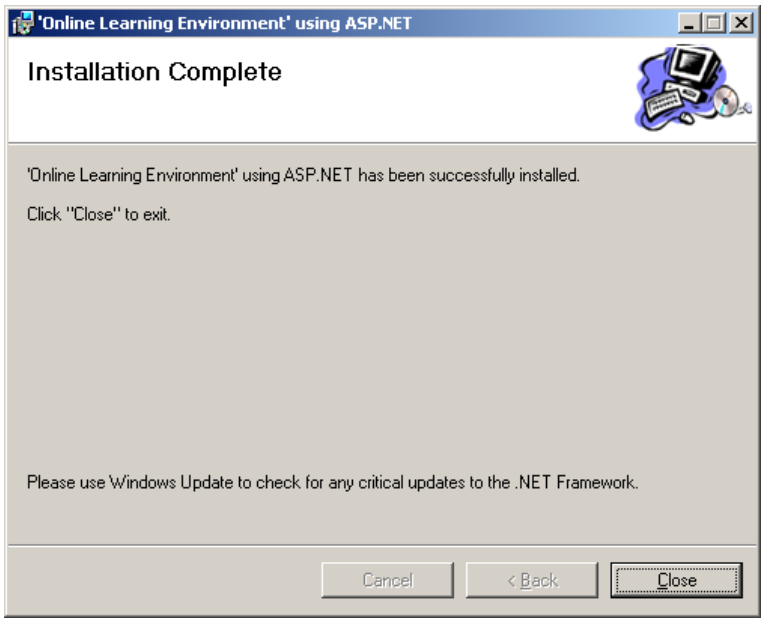


Figure 5.7: Setup finished (sixth step)

Step 7 - Database Installation

Once the setup application has successfully installed the OLE, the user must execute a `.sql` file within their *Microsoft SQL Server 2000+* database, installing the database required for the OLE to work correctly. This file is located in the virtual directory, specified in *Step 3*.

Chapter 6

Software Project Management

The software project management activities presented in this chapter make up the 'nuts and bolts' that construct the management of the OLE project. It allows the progress of its development to be followed, in addition to mapping out the project development's structure.

The first section of this chapter, Section 6.1, describes how documents were maintained throughout the project's development. Section 6.2 details the project's schedule, while Section 6.3 lists the possible risks that may have posed unwanted issues towards the project. Section 6.4 identifies the resources required in order for the project to be implemented. Finally, Section 6.5 details the project's key development environment, Microsoft Visual Studio .NET 2003, and how it managed the implementation of the OLE project's programming source code.

6.1 Documentation

This project generated documents (outlined in Section 2.2.2) as *draft revisions*. A small change in the document structure implies a small change in the revision title. For example, the revision title of *Draft 1.1* increases to *Draft 1.2*. If a large change in the document occurs, the revision title increases from *Draft 1.2* to *Draft 2.1*. Once a document has been completed, the revision title is removed.

This system was introduced to keep a record and back-up of the direction these documents were moving. If questions arose to why a feature was taken out from later document revisions, but existed in earlier revisions, the earlier revisions would be consulted to investigate why it was removed.

6.2 Schedule Management

Tables 6.1 and 6.2 shows the project's work structure and an estimation of completion times. The headings include:

- *Step* - Provides the step identification number.
- *Description* - Provides the name of the task.
- *Estimate* - Provides the estimated time to complete the task.

Table 6.1: The Documentation, Requirements Elicitation and Analysis, and Design work structure

Step	***Description***	***Estimate***
1	Documentation	
1.1	Project Specification	5 hours
1.2	Project Appreciation	50 hours
1.3	Requirements Analysis Document (RAD)	30 hours
1.4	System Design Document (SDD)	9 days
1.5	Object Design Document (SDD)	9 days
1.6	Project Dissertation	5 weeks
1.7	Test Manual	8 days
1.8	User Manual	10 hours
2	Requirements Elicitation and Analysis	
2.1	Needs analysis	3 days
2.2	Secure required resources	1 day
2.3	Obtain supervisor approval	2 hours
3	Design	
3.1	Review RAD	1 day
3.2	Design Presentation Tier	3 days
3.3	Design Business Logic Tier	3 days
3.4	Design Data Access Tier	3 days

Table 6.2: The Training, Implementation, Testing, and Deployment work structure

Step	***Description***	***Estimate***
4	Training	
4.1	C# .NET	6 days
4.2	ASP.NET	6 days
4.3	SQL	3 days
4.4	XML	6 days
5	Implementation	
5.1	Review RAD, SDD, and ODD	5 days
5.2	Develop code	30 days
5.3	Implementation testing (Debugging)	10 days
6	Testing	
6.1	Develop unit test plans	2 days
6.2	Develop integration test plans	2 days
6.3	Unit Testing	
6.3.1	Test component modules	2 days
6.3.2	Modify code	1 day
6.3.3	Re-test modified code	1 day
6.4	Integration Testing	
6.4.1	Test module integration	2 days
6.4.2	Modify code	1 day
6.4.3	Re-test modified code	1 day
7	Deployment	
7.1	Deploy software	2 days

Since the programming and development experience of the writer was limited to different languages and applications, the time estimation was approximated as best as possible.

6.3 Risk Management

Briefly discussed in Section 2.2.4, risk management is split into three section:

- Risk Identification
- Risk Analysis
- Risk Strategy

Risk Identification

Table 6.3 identifies and lists the risks that may be encountered within the design and development of this project (Sommerville 2001, pp. 87-88). They are organised into a *Risk Type* column outlined in the table below (Table 6.3).

Table 6.3: Risk types and the risks that may be encountered (Risk Identification)

Risk Type	***Risks***
Technology	Computer hardware failure Computer security compromised Data loss
People	Illness
Organisational	Not Applicable
Tools	Software unavailability Software evaluation expiration
Requirements	Changes to requirements (major design rework)
Estimation	Time required for development (underestimate) Rate of bug fixing (underestimate) Size of the software (underestimate)

Risk Analysis

Table 6.4 lists and analyses the risks that may affect or jeopardise the project; along with the probability of affect upon, and the effect it may impose on the project (Sommerville 2001, pp. 88-89).

Each risk has a probability rating as either:

- Very low (<10%)
- Low (10-25%)
- Medium (25-50%)
- High (50-75%)
- Very high (>75%)

The effects of each risk are assessed as either:

- Catastrophic
- Serious
- Tolerable
- Insignificant

Table 6.4: Risks, their probability and effects (Risk Analysis)

Risk	***Probability***	***Effects***
Computer hardware failure	Medium	Catastrophic
Computer security compromised	Medium	Serious
Data loss	Medium	Catastrophic
Illness	High	Insignificant
Software unavailability	Low	Catastrophic
Software evaluation expiration	Very High	Tolerable
Changes to requirements	Very Low	Serious
Time required for development (underestimate)	Medium	Serious
Rate of bug fixing (underestimate)	Medium	Tolerable
Size of the software (underestimate)	High	Catastrophic

Risk Strategy

Table 6.5 lists the risks from Table 6.4 and strategies that may be carried out in order to avoid such a danger. These strategies need to be followed in order for work momentum

to be maintained.

Table 6.5: Risks and their strategies (Risk Strategy)

Risk	***Strategy***
Computer hardware failure	Archive constant backups of data using compact disc writing and/or secondary hard drive.
Computer security compromised	Deploy a software firewall to prevent any unauthorised access into computer system.
Data loss	Archive constant printings of documentation.
Illness	Leave an opening at the end of the project due date to allow for such an event.
Software unavailability	Locate and deploy a different software source.
Software evaluation expiration	Backup data and format computer system to allow the software to be used again (this cancels the expiration date).
Changes to requirements (design rework)	Allow for an opening at the end of the project due date to allow for such an event.
Time required for development (underestimate)	Develop and document the system with the ability to add support for certain, non-crucial, components.
Rate of bug fixing (underestimate)	Develop and document the system with the ability to add support for certain, non-crucial, components.
Size of the software (underestimate)	Develop and document the system with the ability to add support for certain, non-crucial, components.

6.4 Resource Requirements

Table 6.6 lists the resources required to develop this project. The total budget came to $159, and was not exceeded at any point within the project's development.

Table 6.6: Resource requirements

Need	***Resources***	***Status***	***Cost***	***Comment***
Development Workstation	1GHz+, 256Mb+ RAM	Acquired	N/A	Own workstation
Database Server	1GHz+, 256Mb+ RAM	Acquired	N/A	Same as Development Workstation
Server Operating System	Microsoft Windows Server 2003	Acquired	Free	120-day limit
Database Server	Microsoft SQL Server 2000 (MSDE)	Acquired	Free	
Development Environment	Microsoft Visual Studio .NET 2003 Academic	Acquired	$159	

6.5 Implementation Management using Microsoft Visual Studio .NET

In Section 2.4.1, Microsoft Visual Studio .NET 2003 (VS.NET) was briefly described. In this section, the ways in which VS.NET is used will be displayed.

VS.NET files are managed by a *solution*, which contains a number of *projects*. The *projects* are usually the different components that exist within a software project. Figure 6.1 displays the *Solution Explorer* that exists within VS.NET for the OLE *solution*.

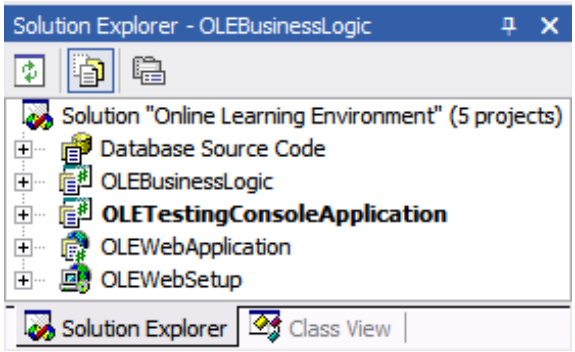


Figure 6.1: Microsoft Visual Studio .NET 2003's Solution Explorer (for the OLE *solution*)

Within the OLE *solution*, four projects exist:

- `Database Source Code` - Maintains a connection to the development database.
- `OLEBusinessLogic` - The main class library.
- `OLEWebApplication` - Manages the Web pages and Web components.
- `OLETestingConsoleApplication` - Used to test the main class library (`OLEBusinessLogic`).
- `OLEWebSetup` - The installation application.

VS.NET's *Configuration Manager* allows the developer to define the compilation/build process of the *solution*'s *projects* so that *project* dependencies can be maintained. Figures 6.2 and 6.3 show the two configurations that exist within the OLE *solution*:

- *Debug* (Figure 6.2) - Optimized compilation/build for testing and debugging.
- *Release* (Figure 6.3) - Optimized compilation/build for speed in a release environment.

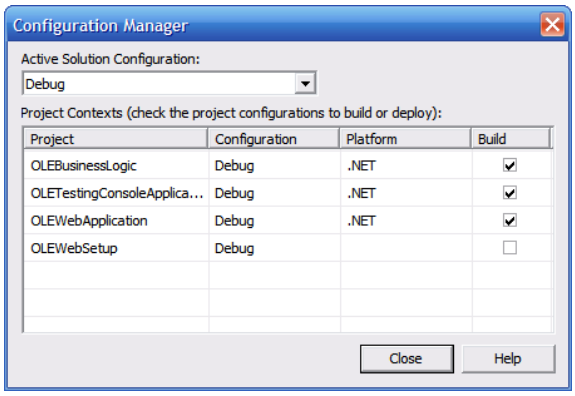


Figure 6.2: Microsoft Visual Studio .NET 2003's *Debug* Configuration Manager (for the OLE *solution*)

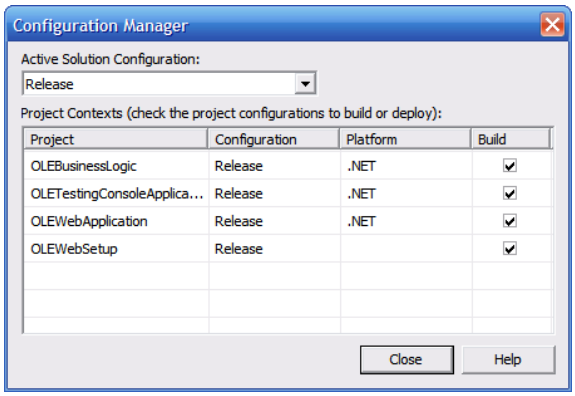


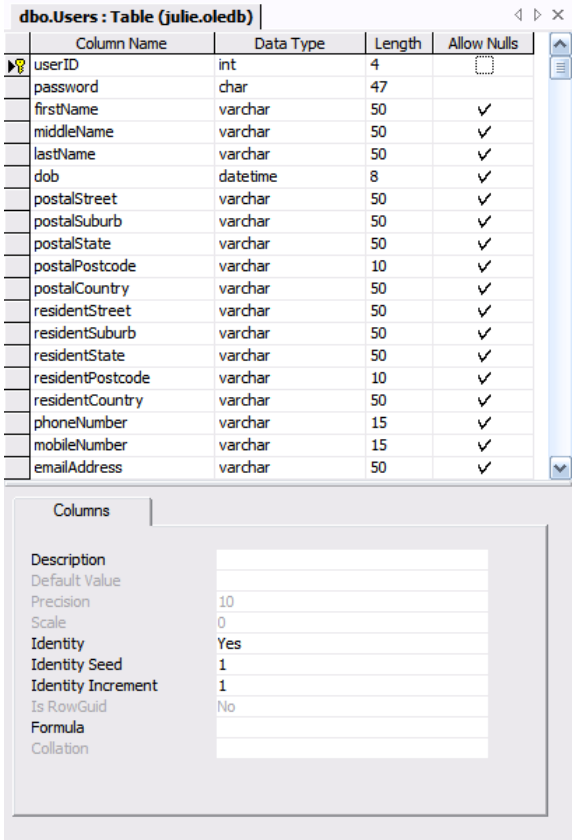
Figure 6.3: Microsoft Visual Studio .NET 2003's *Release* Configuration Manager (for the OLE *solution*)

The following two sub-sections below will describe the *projects* that effectively managed the database and setup application.

6.5.1 `Database Source Code` Project

The database for SQL Server was created without the use of programming code in VS.NET (excluding stored procedures). This allowed more time to be given to the *projects* that required programming code.

Figure 6.4 details the design of a table within VS.NET. It allows the definition of primary keys, data types, and null values to the table's fields.



dbo.Users : Table (julie.oledb)

Column Name	Data Type	Length	Allow Nulls
userID	int	4	
password	char	47	
firstName	varchar	50	✓
middleName	varchar	50	✓
lastName	varchar	50	✓
dob	datetime	8	✓
postalStreet	varchar	50	✓
postalSuburb	varchar	50	✓
postalState	varchar	50	✓
postalPostcode	varchar	10	✓
postalCountry	varchar	50	✓
residentStreet	varchar	50	✓
residentSuburb	varchar	50	✓
residentState	varchar	50	✓
residentPostcode	varchar	10	✓
residentCountry	varchar	50	✓
phoneNumber	varchar	15	✓
mobileNumber	varchar	15	✓
emailAddress	varchar	50	✓

Columns

Description	
Default Value	
Precision	10
Scale	0
Identity	Yes
Identity Seed	1
Identity Increment	1
Is RowGuid	No
Formula	
Collation	

Figure 6.4: Microsoft Visual Studio .NET 2003's database table creation/alteration (for the OLE *solution*)

Figure 6.5 details the referential integrity (relationships) between the tables in the database. A one-to-infinity relationship is shown by a key symbol (one) and an infinite symbol (infinity).

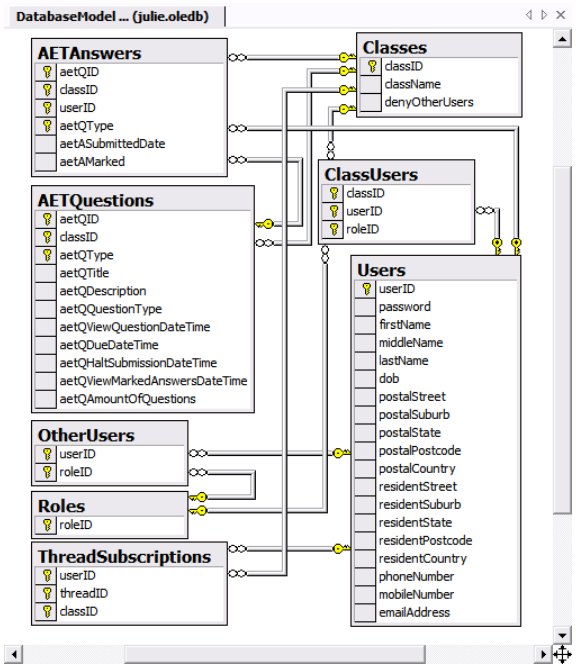


Figure 6.5: Microsoft Visual Studio .NET 2003's database relationship creation/alteration (for the OLE *solution*)

6.5.2 `OLEWebSetup` Project

A setup application was created using VS.NET via a *project* named `OLEWebSetup`. Below in Figures 6.6, 6.7, and 6.8, the configuration of the setup application is shown. By default, a Web setup project automatically detects whether Microsoft Internet Information Services (IIS) is installed.

Figure 6.6 details the screens shown as the user proceeds through the setup process. All screens are included by default, except the `Textboxes (A)` screen. It has been included so that the user can provide a server name for where the SQL Server database will exist.

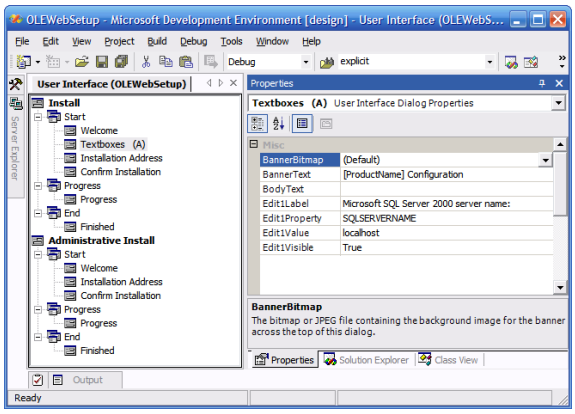


Figure 6.6: The *User Interface* configuration

Figure 6.7 details where the OLE's files are to be installed during the setup. The *Web Application Folder* is the virtual folder created by the installation, within IIS. VS.NET *solutions* allow its *projects* to interact with each other, which is shown by the `OLEWebApplication`'s *Content Files* and *Primary Output* existing within the setup application's file system.

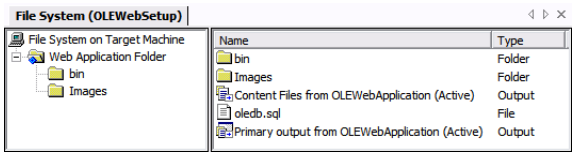


Figure 6.7: The *File System* configuration

Figure 6.8 shows the registry values that will be created when the installation of the OLE occurs. It allows the OLE to retrieve information after installation.

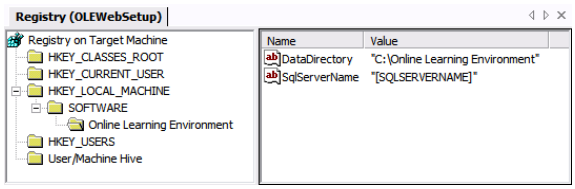


Figure 6.8: The *Registry* configuration

Chapter 7

Conclusions

Throughout this project, a large amount of research has been performed, allowing a large portion of the 'Online Learning Environment' to be completed, as well as allowing the acquirement of experience in:

- *Development of an enterprise-level software application* - Designing the architecture, and implementation and deployment the software application.
- *Small-scale project management* - Documenting the requirements and design of the software application, and scheduling the tasks and managing the risks of the project.
- *Research and incorporation of different programming models and languages* - Including the .NET Framework, ASP.NET, C# .NET, SQL, XML, HTML, and CSS.

While the result of the 'Online Learning Environment' was not as successful as anticipated, the journey of learning new and exciting skills within the field of software engineering was invaluable.

7.1 Further Work

Benefits of this project would not occur solely from continuation of the current progress. Rather, it would benefit the projectee to re-design the entire system, learning from the ideas in this project and improving on them.

The following sub-sections will provide a more detailed scope for a future project, derived from the 'Online Learning Environment'. It will propose a larger system perspective, not confined to students and instructors. It will also propose a system architecture with focus on the database and data retrieval.

XML Web Services

An XML Web Service acts as a bridge between applications and data via the HyperText Transfer Protocol (HTTP). Under the .NET Framework, an application or Web application can access an XML Web Service via a network. This technology is used from the Internet and has the ability to provide excellent grounds to commence further research and development.

New Data Storage and Retrieval Concept

The OLE utilized two techniques in order to store data (Section 3.1.2). As mentioned before, that programming model was designed so that the two techniques could be learnt. However, the following programming model provides an easier alternative to implementation of a future project as well as providing greater efficiency.

Since an SQL database field can only hold 8000 characters using the SQL data-type, `VARCHAR`. An improvement is to create multiple entries for a field that exceed the character limitation. Figure 7.1 details this new concept.

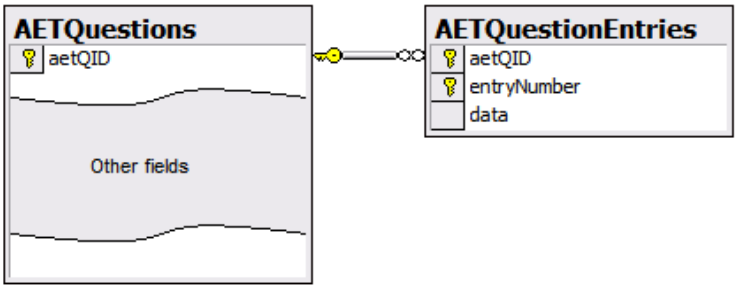


Figure 7.1: New database concept.

The existence of XML within this new architecture will no have to be thrown away either. Unlike Microsoft SQL Server 2000, Microsoft SQL Server 2005 has native XML support. Meaning that the new programming model can use XML in order to retrieve data from the new database concept.

7.2 The Future Project

The idea for this future project comes from initial project plans. Lack of knowledge however, in Web development forced the complexity of this project to be reduced to the project described in this dissertation.

The idea for the project involves the process of merging different institution departments, such as retail stores, financial and human resource management, and, learning environment (this project). A small example of this project's potential is described in a use case (below).

Bookshop Use Case

A student accesses their class Web site (learning environment site) at the start of the semester, and then clicks on a menu item called 'Text Books'. This would displays a page that lists the books required for the course, and allows the student to add them to the Bookshop's (retail store) shopping cart. The Bookshop's printer can then print this order and allow employees to package the books and send them to the student.

Due to the scale of the project, an option for future work is to take a collaborative approach among students.

References

Ahmed, M., Garrett, C., Faircloth, J., Payne, C. & DotThatCom.com (2002), *ASP.NET Web Developer's Guide*, Syngress Publishing Inc., Rockland, MA, United States of America.

Alhir, S. S. (2003), *Learning UML*, O'Reilly & Associates Inc., California, United States of America.

Anderson, J. & Poole, M. (2001), *Assignment and Thesis Writing*, forth edn, John Wiley and Sons, Brisbane, Australia.

Archer, T. (2001), *Inside C#*, Microsoft Press, Washington, United States of America.

Bell, J., Clark, M., Elmhorst, A., Gibbs, M., Homer, A., Lee, B., Milner, M., Narkiewicz, J., Rehan, A. & Schenken, J. (2001), *ASP.NET Programmer's Reference*, Wrox Press, Birmingham, United Kingdom.

Bruegge, B. & Dutoit, A. H. (2000), *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*, Prentice Hall, New Jersey, United States of America.

Bruegge, B. & Dutoit, A. H. (2004), *Object-Oriented Software Engineering Using UML, Patterns, and Java*, second edn, Pearson Prentice Hall, New Jersey, United States of America.

Chaffee, A. (2000), *One, two, three, or n tiers?*, JavaWorld.com, http://www.javaworld.com/javaworld/jw-01-2000/jw-01-ssj-tiers-p4.html.

Coleridge, R. (2002), An introduction to the duwamish books sample application,

in 'MSDN Library - Visual Studio .NET 2003', Microsoft Corporation, ms-help://MS.MSDNQTR.2003FEB.1033/dnduwam/html/dw1intro.htm.

CollabNet Inc. (2003), *readyset.tigris.org*, CollabNet Inc., http://readyset.tigris.org/.

Connolly, T. & Begg, C. (2002), *Database Systems*, third edn, Pearson Education Limited, Essex, England.

Grimes, R. (2002), *Developing Applications with Visual Studio.NET*, Pearson Education Limited, Essex, England.

IEEE (1997), *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Computer Society, New York, United States of America.

J. D. Meier, A. Mackman, M. D. & Vasireddy, S. (2002*a*), *How To: Implement Iprincipal*, Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/SecNetHT06.asp.

J. D. Meier, A. Mackman, M. D. & Vasireddy, S. (2002*b*), *How To: Use Forms Authentication with SQL Server 2000*, Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/SecNetHT03.asp.

J. D. Meier, A. Mackman, M. D. & Vasireddy, S. (2004), *Building Secure ASP.NET Applications*, Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/secnetlpMSDN.asp.

Malcolm, G. (2001), *Programming Microsoft SQL Server 2000 with XML*, Microsoft Press, Washington, United States of America.

McDonald, L. (2003), *Advantages of Using ASP.Net*, Brilliance Web Design Inc., http://www.brillianceweb.com/betterwebdesign/tips_52.aspx.

McManus, J. P. & Kinsman, C. (2002), *C# Developer's Guide to ASP.NET, XML, and ADO.NET*, Pearson Education Limited, Essex, England.

Microsoft Corporation (2001*a*), *Microsoft .NET Framework Datasheet*, number 2, Microsoft Corporation, http://www.gotdotnet.com/team/framework/DotNET Framework DataSheet v2.doc.

Microsoft Corporation (2001*b*), *Microsoft .NET Framework Technical Overview*, number 3, Microsoft Corporation, http://www.gotdotnet.com/team/framework/DotNet Framework Technical Overview v3.doc.

Microsoft Corporation (2002*a*), *Inside the .NET Framework*, number 1.1.0, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconInsideNETFramework.asp.

Microsoft Corporation (2002*b*), *Overview of Windows Server 2003, Web Edition*, Microsoft Corporation, http://www.microsoft.com/windowsserver2003/evaluation/overview/web.mspx.

Microsoft Corporation (2002*c*), *SQL Server 2000 Product Overview*, Microsoft Corporation, http://www.microsoft.com/sql/evaluation/overview/default.asp.

Microsoft Corporation (2003*a*), *HOW TO: Use ASP.NET to Protect File Types*, Microsoft Corporation, http://support.microsoft.com/default.aspx?scid=kb;en-us;815152&Product=aspnet.

Microsoft Corporation (2003*b*), *Microsoft SQL Server: Top 30 Features of SQL Server 2005*, Microsoft Corporation, http://www.microsoft.com/sql/2005/productinfo/top30features.asp#B.

Microsoft Corporation (2003*c*), *MSDE 2000 Product Overview*, Microsoft Corporation, http://www.microsoft.com/sql/msde/productinfo/overview.asp.

Microsoft Corporation (2003*d*), *Technical Overview of Internet Information Services (IIS) 6.0*, Microsoft Corporation, http://download.microsoft.com/download/8/a/7/8a700c68-d1af-4c8d-b11e-5f974636a7dc/IISOverview.doc.

Microsoft Corporation (2004*a*), *ASP.NET Overview*, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconInsideNETFramework.asp.

Microsoft Corporation (2004*b*), *The ASP.NET Web Matrix Project (Reloaded!)*, Microsoft Corporation, http://www.asp.net/webmatrix/default.aspx?tabIndex=4&tabId=46.

Microsoft Corporation (2004*c*), *How To Upload a File to a Web Server in ASP.NET by Using Visual C# .NET*, Microsoft Corporation, http://support.microsoft.com/default.aspx?scid=kb;en-us;323246&Product=aspnet.

Microsoft Corporation (2004*d*), *MSDN ECMA Standardisation*, Microsoft Corporation, http://msdn.microsoft.com/net/ecma/.

Microsoft Corporation (2004*e*), *Product Information for Visual C# .NET 2003*, Microsoft Corporation, http://msdn.microsoft.com/vcsharp/productinfo/.

Microsoft Corporation (2004*f*), *Product Overview for Visual Studio .NET 2003*, Microsoft Corporation, http://msdn.microsoft.com/vstudio/productinfo/overview/default.aspx.

Microsoft Developer Network (2003), *MSDN Library - Visual Studio .NET 2003*, Microsoft Corporation, ms-help://MS.MSDNQTR.2003FEB.1033/.

Microsoft Developer Network (2004), *Microsoft Developer Network*, Microsoft Corporation, http://msdn.microsoft.com/.

Onion, F. (2003), *Essential ASP.NET with Examples in C#*, Pearson Education Limited, Essex, England.

Sheriff, P. D. (2002), *Designing a .NET Application*, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/designnetapp.asp.

Sommerville, I. (2001), *Software Engineering*, sixth edn, Pearson Education Limited, Essex, England.

Wahlin, D. (2002), *XML for ASP.NET Developers*, Sams, Indiana, United States of America.

Worley, S. (2001), *Inside ASP.NET*, New Riders Publishing, Indiana, United States of America.

Wright, C. (2002), *C# Tips & Techniques*, The McGraw-Hill Companies, California, United States of America.

Appendix A

Project Specification

UNIVERSITY OF SOUTHERN QUEENSLAND
FACULTY OF ENGINEERING AND SURVEYING
ENG4111/2 – RESEARCH PROJECT

PROJECT SPECIFICATION

FOR: Timothy Ross **Vriesema**

TOPIC: 'Online Learning Environment' using ASP.NET

SUPERVISORS: Dr. Hong Zhou & Dr. John Leis

PROJECT AIM: This project aims to design and develop a generalised learning environment application for educational institutions that endeavours to use modern programming languages, application environments, and software project management techniques (and methodologies). The software will include features such as:

- Assignment, tutorial, and exam management; and
- Discussion forums.

PROGRAMME: *Issue F*, 26 August 2004

1. Research of the Microsoft .NET Framework and development languages required for the software project's development. These languages include C# .NET, ASP.NET, XML, and SQL.

2. Research the different processes for managing software projects and select an appropriate means to do so.

3. Investigate the project requirements and generate a *Requirements Analysis Document* (RAD).

4. Design the software project's overall and internal (object) architecture and generate a *System Design Document* (SDD) and an *Object Design Document* (ODD).

If time permits:

5. Develop the software application and alter the *SDD* and *ODD* documents at the discretion of the development process.

6. Produce a testing procedure and then perform the tests while generating a *Test Manual*.

7. Generate a *User Manual* for information on how to operate the software application.

AGREED:

_____ (Student Signature) _____ (Supervisor Signature)

____ / ____ / ____ (Date) ____ / ____ / ____ (Date)

_____ (Supervisor Signature)

____ / ____ / ____ (Date)

Appendix B

Requirements Analysis Document

‘Online Learning Environment’ using ASP.NET

REQUIREMENTS ANALYSIS DOCUMENT

Thursday, 14 October 2004

NAME

Timothy Vriesema

tripix@msn.com

STUDENT NUMBER

0011122446

UNIVERSITY PROGRAM

Bachelor of Engineering

(Software Engineering)

AND

Bachelor of Business

(Management and Leadership)

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **I**

1 INTRODUCTION **1**

1.1 PURPOSE OF THE SYSTEM 1
1.2 SCOPE OF THE SYSTEM 1
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 1
1.4 OVERVIEW 1

2 CURRENT SYSTEM **3**

2.1 OVERVIEW 3
2.2 FUNCTIONS AND FEATURES 3
2.2.1 Multi-Level Administrative Roles *3*
2.2.2 Announcements *3*
2.2.3 Assessments *3*
2.2.4 Notes *4*
2.2.5 Discussion Forum *4*
2.2.6 Internal Mail *4*
2.2.7 File Manager *4*

3 PROPOSED SYSTEM **5**

3.1 OVERVIEW 5
3.2 FUNCTIONAL REQUIREMENTS 5
3.2.1 Components Shared by All Roles *5*
3.2.1.1 Login Authentication 5
3.2.1.2 Quick Links 5
3.2.1.3 Institution Announcements 5
3.2.1.4 Personal Information Configuration 5
3.2.2 Components for the Administrator Role *6*
3.2.2.1 Recent Errors 6
3.2.2.2 Database Configuration 6
3.2.2.3 E-Mail Configuration 6
3.2.2.4 Institution Configuration 6
3.2.2.5 Terminology Configuration 6
3.2.2.6 Web Application Startup and Shutdown 6
3.2.2.7 Error Logging 7
3.2.2.8 Login Logging 7
3.2.2.9 User Management 7
3.2.2.10 Institution Announcements Management 7
3.2.3 Components Shared by Instructor and Student Roles *7*
3.2.3.1 Assignments, Exams, and Tutorials 7
3.2.3.2 Assignment, Exam and Tutorial Solutions 7
3.2.3.3 Upcoming Important Dates 7
3.2.3.4 Class Announcements 8
3.2.3.5 Notes 8
3.2.3.6 Forums 8
3.2.4 Components for the Instructor Role *8*
3.2.4.1 Assignment, Exam, and Tutorial Management 8
3.2.4.2 Class Authorisation 9
3.2.4.3 Class Announcements Management 9
3.2.4.4 Notes Management 9
3.2.4.5 Assignment, Exam and Tutorial Answer Viewing and Downloading 10
3.2.4.6 Assignment, Exam and Tutorial Submission List 10
3.2.5 Components for the Student Role *10*
3.2.5.1 Forum Thread Subscriptions 10
3.2.5.2 Assignment, Exam and Tutorial Submission 10
3.3 NON-FUNCTIONAL REQUIREMENTS 10

Copyright © 2004 by Timothy Vriesema

3.3.1 Usability *10*
3.3.2 Reliability *11*
3.3.3 Performance *11*
3.3.4 Supportability *11*
3.3.5 Implementation *11*
3.3.6 Interface *11*
3.3.7 Operation *11*
3.4 SYSTEM MODELS 11
3.4.1 Scenarios *12*
3.4.2 Use Case Models *16*
3.4.2.1 Use Case Diagram – Class Subsystem 17
3.4.2.2 Use Case Diagram – Configuration Subsystem 18
3.4.2.3 Use Case Diagram – Submission and Management Subsystems 19
3.4.2.4 Use Case Diagrams – Login and Home Subsystems 20
3.4.3 Object Model *21*
3.4.3.1 Class Diagram 21
3.4.3.2 Class – User 21
3.4.3.3 Abstract Class – AETAnswer 22
3.4.3.4 Inherited Class – AETInformationAnswer : AETAnswer 22
3.4.3.5 Inherited Class – AETMultipleChioceAnswer : AETAnswer 22
3.4.3.6 Inherited Class – AETShortResponseAnswer : AETAnswer 22
3.4.3.7 Abstract Class – AETQuestion 22
3.4.3.8 Inherited Class – AETInformationQuestion : AETQuestion 23
3.4.3.9 Inherited Class – AETMultipleChioceQuestion : AETQuestion 23
3.4.3.10 Inherited Class – AETShortResponseQuestion : AETQuestion 23
3.4.3.11 Class – Class 23
3.4.3.12 Class – Role 23
3.4.3.13 Class – Thread 24
3.4.3.14 Abstract Class – Post 24
3.4.3.15 Inherited Class – ClassAnnouncement : Post 24
3.4.3.16 Inherited Class – ClassThreadPost : Post 24
3.4.3.17 Inherited Class – ClassNote : Post 24
3.4.3.18 Inherited Class – InstitutionalAnnouncement : Post 25
3.4.4 Dynamic Models *25*
3.4.4.1 Sequence: Assignment, Exam, and Tutorial Posting and Answering 25
3.4.4.2 Sequence: Assignment, Exam, and Tutorial Marking 26
3.4.5 User Interface *26*

4 GLOSSARY 28

1 INTRODUCTION

The following subsections will provide an introduction into the document's purpose, the scope of the proposed system, and an overview of the rest of this document.

1.1 Purpose of the System

The system that this document is generated from is the 'Online Learning Environment' using ASP.NET (OLE). It aims to develop a system for educational institutions and their needs. This primarily includes class management by instructors, for enrolled students.

1.2 Scope of the System

The OLE will be utilized by students of an educational institution to:

- View and submit class assignments, tutorials, and/or exams online;
- Edit personal details (i.e. residential and email addresses);
- View announcements from both instructors and the educational institution; and
- Discussing events, questions, and statements via a dynamic class forum.

Instructors will also have distinct advantages from this system. These include:

- Assignment, tutorial, and exam management (i.e. creating, altering, and deleting questions);
- Assignment, tutorial, and exam marking from online submissions from students;
- Edit personal details; and
- Interacting with students via the dynamic class forum.

1.3 Definitions, Acronyms, and Abbreviations

Before reading the *Requirements Analysis Document*, some abbreviations are presented below that will be used.

AET	**A**ssignment, **E**xam, and **T**utorial
ADO.NET	Microsoft **A**ctive **D**ata **O**bjects **.NET**
ASP.NET	Microsoft **A**ctive **S**erver **P**ages **.NET**
IIS	Microsoft **I**nternet **I**nformation **S**ervice 6.0
OLE	**'O**nline **L**earning **E**nvironment**'** using ASP.NET

1.4 Overview

This document gives a detailed description of the system's perspective, functions, user characteristics, constraints, and

Copyright © 2004 by Timothy Vriesema

assumptions and dependencies. Following this general description, it documents the functional and non-functional requirements within the system. This involves a detailed description of what features are expected.

Footnotes have been used throughout this document to help refer the relationships between each component and feature.

2 CURRENT SYSTEM

Within the University of Southern Queensland, a system already exists with features similar to the OLE. This system is called Vista, designed and developed by WebCT.

2.1 Overview

WebCT Vista offers a number of features that will far exceed the expectations of the OLE. This is due to budget and time constraints.

Vista has features that allow students and instructors to meet in a neutral environment.

2.2 Functions and Features

This section will point out some functions and features not provided by the OLE.

2.2.1 Multi-Level Administrative Roles

Vista has many different Administrator roles that allow different people access to certain areas of the system. Some of these roles are:

- System Administrator
- Backup Administrator
- Database Administrator

2.2.2 Announcements

Announcements allow Section Instructors to provide Students with information about the class.

2.2.3 Assessments

This feature within Vista allows Section Instructors to manage assessment items within their respective classes. These assessment items consist of questions such as:

- Multiple choice
- Fill-in blanks
- Match-up definitions
- Complete paragraphs

Section Instructors define the correct answers to these questions before they become available to Students so that the system can automatically mark Students' answers.

Copyright © 2004 by Timothy Vriesema

2.2.4 Notes

The notes feature that exists within Vista allows Section Instructors to manage notes about the class so that Students can view the Instructors' view and perspective on certain class modules.

2.2.5 Discussion Forum

Vista provides a discussion forum so that Section Instructors and Students can communicate about the class. Vista allows Section Instructors to delete messages that violate class rules (e.g. the answer to an assessment item). Students do not have this ability.

2.2.6 Internal Mail

Each class section has an internal mail account where e-mail is passed between Section Instructors and/or other Students. It cannot be used to communicate with outside e-mail addresses.

2.2.7 File Manager

Vista has a built-in file manager for instructors to add files in order for students to download and view.

Copyright © 2004 by Timothy Vriesema

3 PROPOSED SYSTEM

This section provides preliminary information and design for the OLE.

3.1 Overview

The OLE system's preliminary information and design has been described below showing its functional and non-functional requirements, and its system models. Some system models include:

- Scenarios
- Use Case models
- Object models
- Dynamic models
- User Interfaces

3.2 Functional Requirements

The following sub-sections detail the different functional requirements for each user role. These roles are:

- Administrators
- Instructors
- Students

3.2.1 Components Shared by All Roles

This section details the components that will be accessed by all users and roles.

3.2.1.1 Login Authentication

All users will be able to access the Web application by the use of their issued username and password.

3.2.1.2 Quick Links

All users will have access to a list of Web links, linking to other parts of the Web application that will be used on a regular basis. The Web links for each user will be automatically chosen for them.

3.2.1.3 Institution Announcements

This allows all users to be kept up-to-date through displays of news, events, and developments that are happening within the institution.

3.2.1.4 Personal Information Configuration

All users are required to supply personal information in order for their username to exist within the Web application's database. Users will be able to alter this information simply and quickly. Such information that users will be able to alter are:

- Residential address
- E-Mail address

Copyright © 2004 by Timothy Vriesema

- Postal address

3.2.2 Components for the Administrator Role

This section details the components that will be accessed by the *Administrators* within the Web application.

3.2.2.1 Recent Errors

This allows *Administrators* to be shown recent errors so that they can be aware of recent problems within the Web application.

3.2.2.2 Database Configuration

This page allows *Administrators* to set the server name or IP address that holds a *Microsoft SQL Server* database for the Web application to access. The will also be able to set the Web application to create the database on the database server.

3.2.2.3 E-Mail Configuration

Administrators will be able to set the e-mail server POP3 address so that e-mails can be sent.

3.2.2.4 Institution Configuration

This page will allow Administrators to personalise the Web application by specifying educational institution information that may show up throughout it. Some information that will be changeable is:

- Institution name
- Institution abbreviation
- Institution logo image

3.2.2.5 Terminology Configuration

Many educational institutions have different terminology. For example, an instructor could be a teacher, examiner, or tutor. This functional requirement allows the entire Web application to be adapted to individual institutions by dynamically altering the terminology set by *Administrators*. The terms that will be varied include:

- Institution
- Class
- Instructor
- Student
- Assignment
- Exam
- Tutorial

Administrators will be the only users that will view the generalised terms. All other roles will view the terminology set by *Administrators*.

3.2.2.6 Web Application Startup and Shutdown

This allows the *Administrators* to start and stop the Web application. A reason for stopping the Web application may be for database upgrades, configuration changes, or error repairs.

Copyright © 2004 by Timothy Vriesema

3.2.2.7 Error Logging

This log will show errors that have occurred within the Web application. It will display the page the error occurred on, the date and time the error occurred and the error exception or message.

3.2.2.8 Login Logging

This log will display successful and unsuccessful logins to the Web application. It will display the IP address that attempted the login, the date and time the attempt was made, along with the username that was being used.

3.2.2.9 User Management

This allows *Administrators* to create, delete, and alter all user information. When the *Administrators* create a user, that user must be assigned to a certain role. *Students* will then be assigned to the classes they are attending and *Instructors* assigned to the classes they control.

Administrators can also delete users that no longer need to exist within the database and alter a user's role association and information.

3.2.2.10 Institution Announcements Management

This gives the *Administrators* an ability to manage the institution announcements[1] and management features such as posting, editing, and deleting.

3.2.3 Components Shared by Instructor and Student Roles

This section details the components that will be accessed by all users within the roles *Instructors* and *Students*.

3.2.3.1 Assignments, Exams, and Tutorials

This allows *Instructors* and *Students* to view assignment, exam, and tutorial questions for their class.

3.2.3.2 Assignment, Exam and Tutorial Solutions

Instructors and *Students* will be able to download and view solutions to an assignment, exam, or tutorial[2] question. The *Instructors* will upload these solutions[3].

3.2.3.3 Upcoming Important Dates

This allows *Instructors* and *Students* to view a list of due dates for certain assignments, exams, and tutorials. Each entry in the list will be a link to the question.

---

[1] See *Institution Announcements*, section 3.2.1.3
[2] See *Assignments, Exams, and Tutorials*, section 3.2.3.1
[3] See *Assignment, Exam, and Tutorial Management*, section 3.2.4.1

Copyright © 2004 by Timothy Vriesema

3.2.3.4 Class Announcements

This allows *Instructors* and *Students* to view news, events, and developments related to a certain class.

3.2.3.5 Notes

Instructors and *Students* will be able to download notes. The *Instructors* will upload these notes[4].

3.2.3.6 Forums

Instructors and *Students* will be able to post, read, and reply to messages on certain threads within the forum. This allows all users to communicate in a neutral environment, viewed by all.

Each post to the forum cannot be edited or deleted. However, *Instructors* will be able to delete any post as it may violate collusion rules within the educational institution.

3.2.4 Components for the Instructor Role

This section details the components that will be accessed by the *Instructors* within the Web application.

3.2.4.1 Assignment, Exam, and Tutorial Management

All assignments, exams, and tutorials[2] can be added, deleted, and altered. Each question will have an 'Updated' date at the bottom that displays the date it was last modified. *Instructors* will also be able to manually sort these questions so that they are displayed in an order they are satisfied with. Assignments, exams, and tutorials[2] will also have no upper or lower limit, allowing the amount of questions to be completely flexible.

Different assignment, exam, and tutorial[2] types may also exist within the Web application. The different types are:

- *Question* – Created by *Instructors*, giving *Students* a question for them to submit online or by another method;
- *Multiple Choice* – Consisting of a number questions, each displaying a series of answers which may be chosen; and
- *Short Answer* – Consisting of a number of questions and space provided under each for *Students* to input an answer.

The *Instructors* will also be able to input links and images to support the questions.

The *Multiple Choice* type varies from the other types of assignment, exam, and tutorials[2]. *Instructors* input the correct answers into the Web application for each of the questions; thereby allowing the Web application to assign a mark automatically after the task has been submitted.

[4] See *Notes Management*, section 3.2.4.4

Copyright © 2004 by Timothy Vriesema

Each type will also have a mode of submission. They are:

- *None* – *Instructors* will give their own description of how to submit (i.e. hand in at a certain office); and
- *File Submission* – *Students* will upload a file via the Web application.

The *Multiple Choice* and *Short Answer* types, however, have no submission mode. These are automatically submitted. When the *File Submission* mode is used, *Instructors* will set the allowed file types so that *Students* do not submit the wrong file type.

Each assignment, exam, and tutorial[2] will require a due date to be set. Alongside this, *Instructors* may set limits on the questions. These limits include:

- *Time Allow* – *Instructors* can set a date to which the question may be viewed. The question is automatically configured to be displayed to users after this date;
- *Time Between* – *Instructors* can set two date and times for the question to be accessed between and as above, the question is automatically configured to be displayed to users between these dates; and
- *Time Extend* – *Instructors* can enable or disable submissions after the due date.

Instructors can upload the assignment, exam, and tutorial solutions[5] via the Web application as a file that may be downloaded by *Students*. *Students* can also view it as typed text. These solutions can have the option to set the *Time Allow* limit (detailed above) for the Web application to make them available for all users. The solution can be made available immediately instead of using a limit.

3.2.4.2 Class Authorisation

This allows *Instructors* to choose if other *Students*, not enrolled in the particular class, can visit the *Instructors*' pages. This is to prevent assignment, exam, and tutorial solutions[5] becoming readily available to the public.

3.2.4.3 Class Announcements Management

This gives the *Instructors* an ability to manage the class announcements[6]. This includes posting, editing, and deleting.

3.2.4.4 Notes Management

This will allow *Instructors* to upload files to allow other users the ability to download and view them[7]. This gives the *Instructors* the ability to add, delete, and update files.

---

[5] See *Assignment, Exam and Tutorial Solutions*, section 3.2.3.2
[6] See *Class Announcements*, section 3.2.3.4
[7] See *Notes*, section 3.2.3.5

Copyright © 2004 by Timothy Vriesema

3.2.4.5 Assignment, Exam and Tutorial Answer Viewing and Downloading

Instructors will be able to view the *Short Answer*[2] type submissions within the class site and assign a mark to *Students*. The *Multiple Choice*[2] type will not be able to be viewed because the Web application assigns a mark automatically. *Instructors* can download files submitted by *Students* and manually enter a mark via the Web application.

3.2.4.6 Assignment, Exam and Tutorial Submission List

Instructors will be able to view a list containing names of *Students*. This list will also contain a statement (`true` or `false`) as to whether a submission was made by a *Students* user and if the submission was before the due date.

3.2.5 Components for the Student Role

This section details the components that will be accessed by the *Students* within the Web application.

3.2.5.1 Forum Thread Subscriptions

This allows *Students* to subscribe to certain threads within forums[8] so that when a message is posted, extensive searching for its location does not have to occur. Each subscription will be a link to a thread and will show the number of new postings.

3.2.5.2 Assignment, Exam and Tutorial Submission

This allows *Students* to submit an assignment, exam, or tutorial via the Web application. If *File Submission*[2] mode has been chosen, *Students* will be required to supply a file with their answer on it. An error will be given if *Students* attempts to submit the incorrect file type.

3.3 Non-Functional Requirements

This section describes the aspects of the Web application that are not directly related to its functional behaviour. These aspects are:

- Usability
- Reliability
- Performance
- Supportability
- Implementation
- Interface
- Operation
- Packaging

3.3.1 Usability

The users of this Web application will range from all departments within an educational system and therefore suggests that most users will not be completely computer literate. This will then require the Web application to be logical, coherent, and easy-to-use.

[8] See *Forums*, section 3.2.3.6

Copyright © 2004 by Timothy Vriesema

The system will also need to keep the standards that most popular user interfaces implement to allow users to feel comfortable while using and interacting with it.

A *User Manual* will be included so that *Administrators*, *Instructors*, and *Students* can follow instructions on how to use all features and components.

3.3.2 Reliability

This system requires a security feature such as a username and password model so that unauthorised users cannot access it.

3.3.3 Performance

This system requires an unlimited amount of concurrent users to operate it so that it can be accessed any time, every time.

3.3.4 Supportability

This section defines the supportability and maintainability of the Web application by users. It details which users are required to maintain the system.

3.3.5 Implementation

This system requires all users to utilize a Web browser. Since Web browsers are deployed on almost all operating system platforms, the Web application can be accessed anywhere.

The Web application requires a Web server that is *ASP.NET* compatible in order to execute and operate correctly.

3.3.6 Interface

Data within the system should be kept inside a database. This data would include information about the users of the system and their personal details.

3.3.7 Operation

This system will require users that configure many of the system's settings. Some of these settings would affect the:

- Database[9]
- Institution variables[10]

3.4 System Models

This section gives scenarios and models based on the OLE software system. Only the key functions have been given due to the amount of functions the OLE provides.

[9] See *Database Configuration*, section 3.2.2.2
[10] See *Institution Configuration*, section 3.2.2.4

3.4.1 Scenarios

This section details the primary features within the OLE as scenarios. The scenarios consist of a name, relevant actors, and the flow of events to take place.

Scenario Name	`Login Authentication`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits the OLE Web application. 2. The OLE prompts Julie to enter her username and password to gain access. 3. Julie then types her username and password (where the password is hidden on the screen so others do not see it). 4. The OLE then gives Julie access to the Web application.

Scenario Name	`Mangage Institution Announcements`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses the Institution Announcements management. 3. Julie then adds/deletes/alters an announcement.

Scenario Name	`View Institution Announcement`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim accesses and views the Institution Announcements.

Scenario Name	`Alter Personal Information`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim accesses and alters his personal information.

Scenario Name	`Configure Database`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses and alters the database's server name.

Scenario Name	`Configure E-Mail`
Participating Actors	`julie:Administrator`

Copyright © 2004 by Timothy Vriesema

Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses and alters the e-mail server name.

Scenario Name	`Manage Roles`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses the Roles Management page, and adds a new role called 'Guest'. 3. Julie then configures 'Guest' to view all pages within the OLE Web application.

Scenario Name	`Configure Institution`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses and alters the institution name and logo.

Scenario Name	`Configure Terminology`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses and alters the terminology to be used within the OLE. One alteration includes Instructor to Teacher.

Scenario Name	`Web Application Running Status`
Participating Actors	`julie:Administrator`
Flow of Events	1. Julie visits and logs into the OLE Web application. 2. Then, Julie accesses and alters the OLE Web application's running state to 'stop'. This will stop all users from using the server-side scripts.

Scenario Name	`Error Logging`
Participating Actors	`julie:Administrator` `tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application and receives an error when he tries to access a page. 2. The server then records the time, date, the username that Tim had used, and the error given.

Copyright © 2004 by Timothy Vriesema

	3. Julie then accesses the `OLE`, views the log, and then repairs the error.

Scenario Name	`Login Logging`
Participating Actors	`julie:Administrator`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. The server then records the time, date, and username that Tim had used. 3. Julie then accesses the `OLE` and views the log.

Scenario Name	`Manage Assignment, Exam, and Tutorial`
Participating Actors	`john:Instructor`
Flow of Events	1. John visits and logs into the OLE Web application. 2. Then, John adds an assignment to his class. He types the question and posts it immediately. 3. John then uploads a file with the questions solution.

Scenario Name	`View Assignment, Exam, and Tutorial`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim visits and views the assignment question page.

Scenario Name	`View Assignment, Exam, and Tutorial Solution`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim visits and views the assignment question page. He then clicks and downloads the question's solution.

Scenario Name	`Submit Assignment, Exam, and Tutorial Answer`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim visits and views the assignment question page. 3. Tim then uploads his assignment file via the page.

Copyright © 2004 by Timothy Vriesema

Scenario Name	`View Assignment, Exam, and Tutorial Submission List`
Participating Actors	`john:Instructor`
Flow of Events	1. John visits and logs into the OLE Web application. 2. Then, John adds an assignment to his class. He types the question and posts it immediately. 3. John then uploads a file with the questions solution.

Scenario Name	`View Assignment, Exam, and Tutorial Answer`
Participating Actors	`john:Instructor`
Flow of Events	1. John visits and logs into the OLE Web application. 2. Then, John adds an assignment to his class. He types the question and posts it immediately. 3. John then uploads a file with the questions solution.

Scenario Name	`Manage Class Announcement`
Participating Actors	`john:Instructor`
Flow of Events	1. John visits and logs into the OLE Web application. 2. Then, John adds an announcement to the class page. 3. He then alters the announcement by adding some more information.

Scenario Name	`View Class Annoucement`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim visits the class page and views the announcements.

Scenario Name	`Manage Note`
Participating Actors	`john:Instructor`
Flow of Events	1. John visits and logs into the OLE Web application. 2. Then, John adds a note to the class page. 3. He then alters the note by adding some more information.

Scenario Name	`View Note`
Participating Actors	`tim:Student`

Copyright © 2004 by Timothy Vriesema

Flow of Events	1. Tim visits and logs into the OLE Web application. 2. Then, Tim visits the class page and views the notes.

Scenario Name	`View Forum Thread`
Participating Actors	`john:Instructor` `tim:Student`
Flow of Events	1. Both, John and Tim visit and log into the `OLE` Web application. 2. Then, Tim and John visit the class page and view the forum. 3. John posts on the forum. 4. Tim, then replies to John's post.

Scenario Name	`Class Authorisation`
Participating Actors	`john:Instructor`
Flow of Events	1. John visits and logs into the OLE Web application. 2. Then, John visits and alters the class authorisation configuration.

3.4.2 Use Case Models

This section generates use case diagrams from scenarios above. It involves certain actors that interact with the OLE and its components. These actors are:

- `Administrators` – The actors that administrate and manage the software system.
- `Students` – The actors that use and interact with the software system, guided by an `Instructor`.
- `Instructors` – The actors that use and interact with the software system, guiding more than or one `Student`.

Copyright © 2004 by Timothy Vriesema

3.4.2.1 Use Case Diagram – Class Subsystem

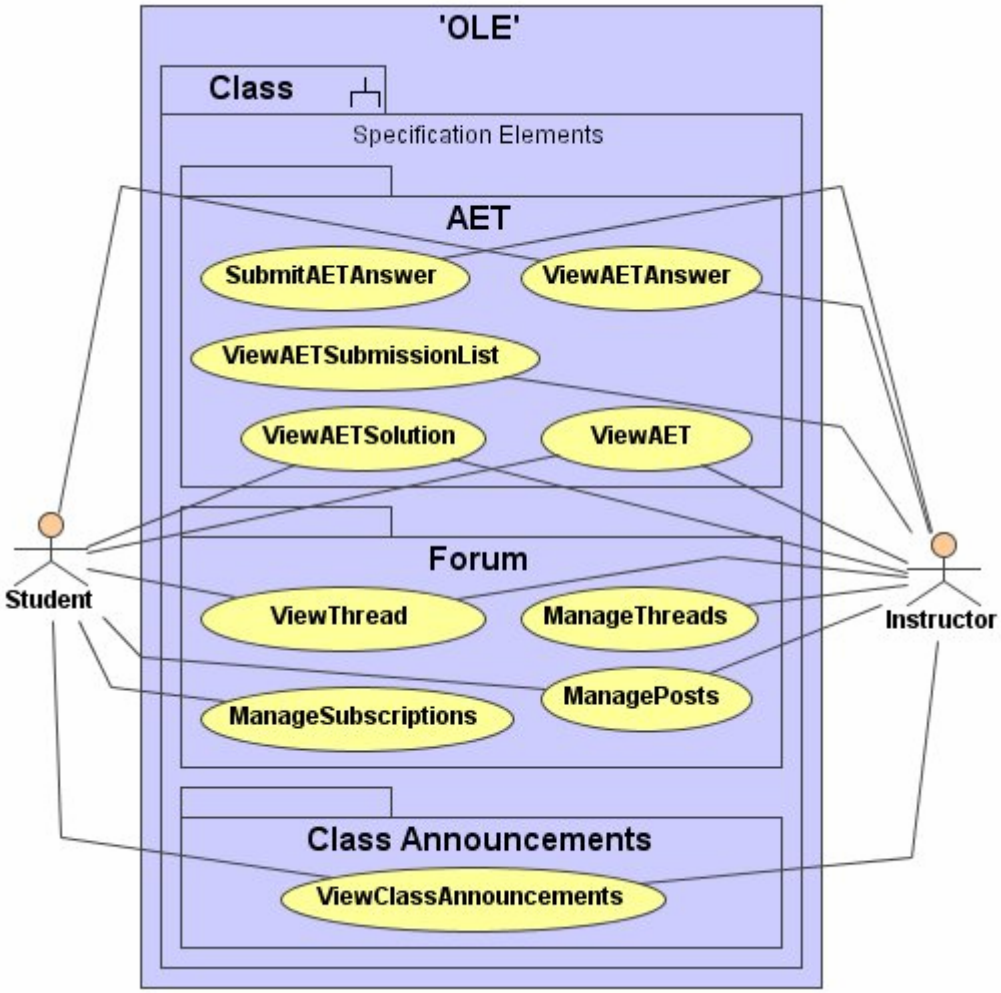


Figure 3.1: The use case diagram of the Class subsystem (UML)

Use Case Name	`ViewClassAnnoucements`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits the class announcements page and reads the announcement Jon has entered.
Entry Conditions	• The `Student` is logged into the OLE.
Exit Conditions	None

Use Case Name	`ViewAET`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits the Assignments page and clicks on the first assignment. 2. The OLE then displays the questions that are associated with that assignment.
Entry Conditions	• The `Student` is logged into the OLE.
Exit Conditions	• Receive an acknowledgement of success or fail.

Use Case Name	`SubmitAETAnswer`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits the Assignments page and clicks on the first assignment. 2. The OLE then displays the questions that are associated with that assignment. 3. He then fills out the questions and clicks 'Submit'.
Entry Conditions	• The `Student` is logged into the OLE.
Exit Conditions	• Receive an acknowledgement of success or fail.

3.4.2.2 Use Case Diagram – Configuration Subsystem

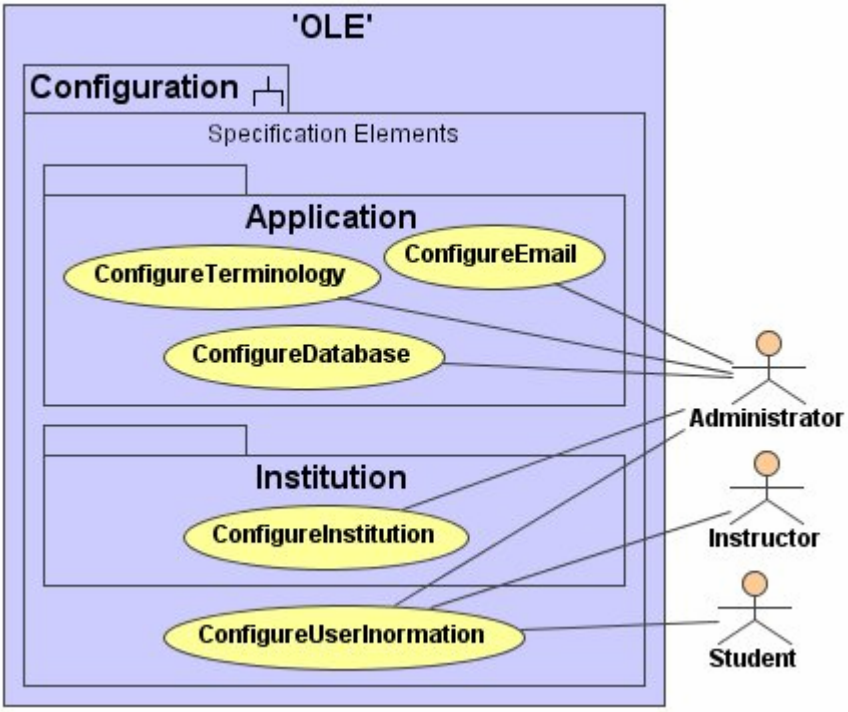


Figure 3.2: The use case model of the Configuration subsystem (UML)

Use Case Name	`ConfigureUserInformation`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits the Configuration page and configures his user information. 2. He changes the current home address to his new home address.
Entry Conditions	• The `Student` is logged into the OLE.
Exit Conditions	• Receive an acknowledgement of success or fail.

3.4.2.3 Use Case Diagram – Submission and Management Subsystems

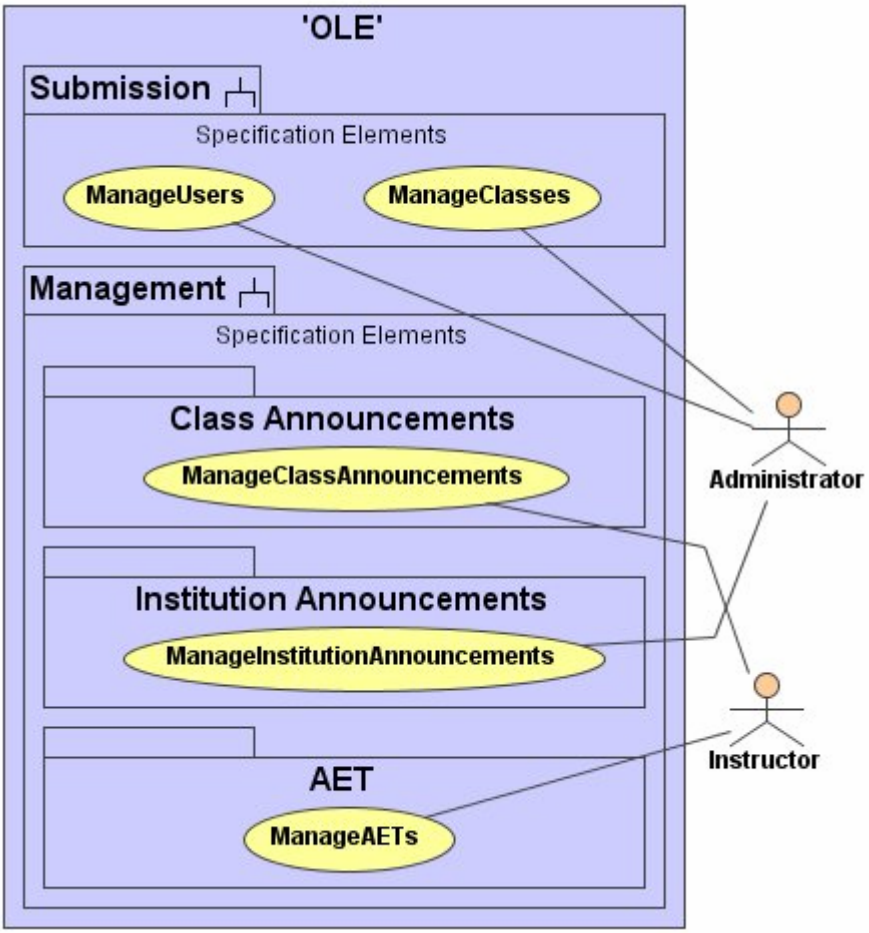


Figure 3.3: The use case model of the Submission and Management subsystems (UML)

Use Case Name	`ManageInstitutionAnnouncements`
Participating Actors	`john:Administrator`
Flow of Events	1. Jon adds two announcements by entering information and file attachments. 2. He notices a problem with one and decides to delete it completely. 3. He then notices a minor spelling error in the other announcement so he alters its information.
Entry Conditions	• The `Administrator` is logged into the OLE.
Exit Conditions	• Receive an acknowledgement of success or fail.

Use Case Name	`ManageClassAnnouncements`
Participating Actors	`john:Instructor`
Flow of Events	1. Jon adds two announcements by entering information and file attachments. 2. He notices a problem with one and

	decides to delete it completely. 3. He then notices a minor spelling error in the other announcement so he alters its information.
Entry Conditions	• The `Instructor` is logged into the OLE.
Exit Conditions	• Receive an acknowledgement of success or fail.

3.4.2.4 Use Case Diagrams – Login and Home Subsystems

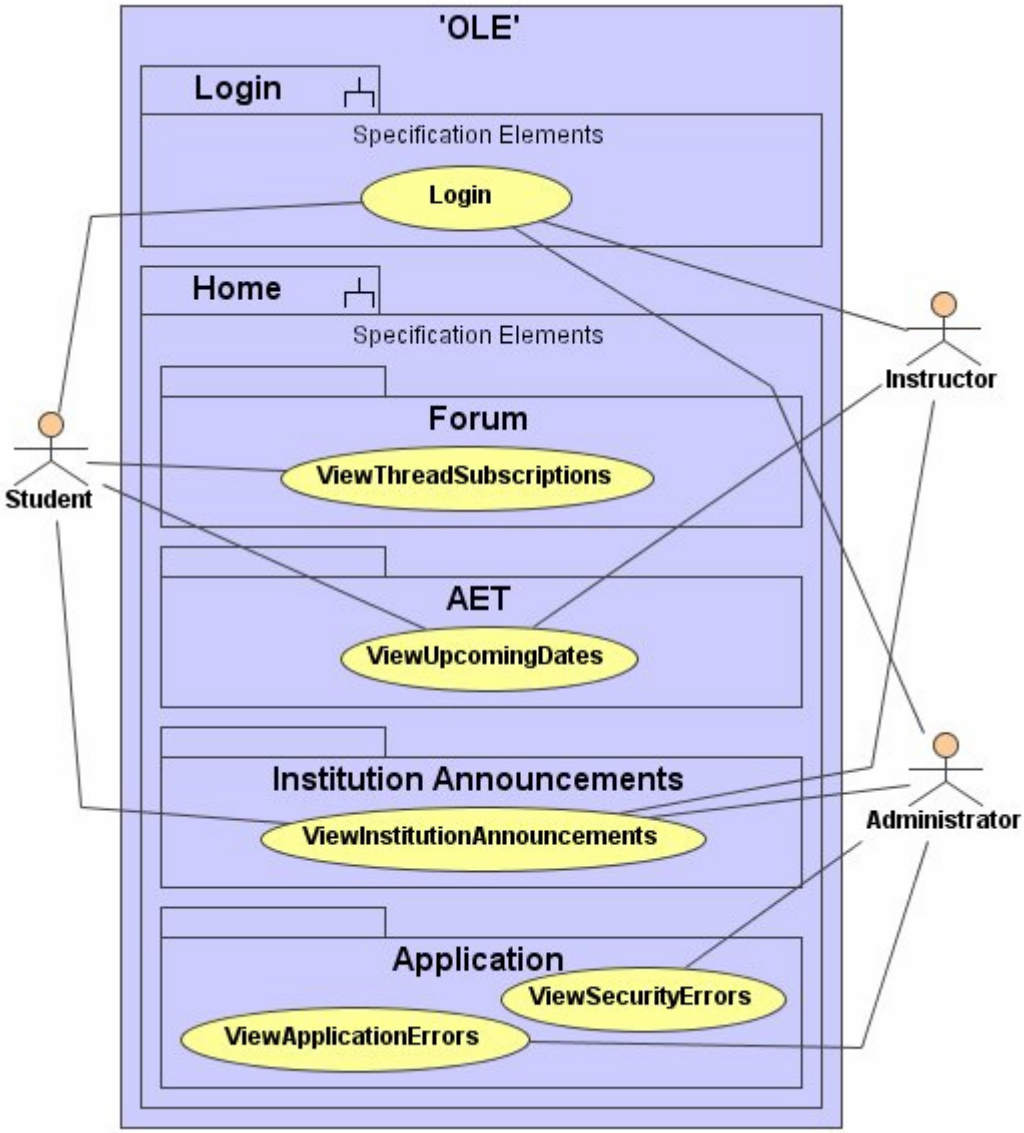


Figure 3.4: The use case model of the Login and Home subsystems (UML)

Use Case Name	`Login`
Participating Actors	`tim:AllRoles`
Flow of Events	1. Tim visits the OLE and views the Login page. 2. He types his user ID and password. 3. The OLE issues an error describing his username or password is incorrect.

	4. He then re-types them and enters.
Entry Conditions	• The `Student` is *not* logged into the OLE.
Exit Conditions	• Receive an acknowledgement of success or fail.

Use Case Name	`ViewInstitutionAnnoucements`
Participating Actors	`tim:Student`
Flow of Events	1. Tim visits the Home page and reads the institutional announcements.
Entry Conditions	• The `Student` is logged into the OLE.
Exit Conditions	None

3.4.3 Object Model

This section will detail the classes that will be utilized within the 'Online Learning Environment' Web application. Section 3.4.3.1 details the class model diagram. The sections following the class model diagram describe, in detail, each class and its attributes and methods.

3.4.3.1 Class Diagram

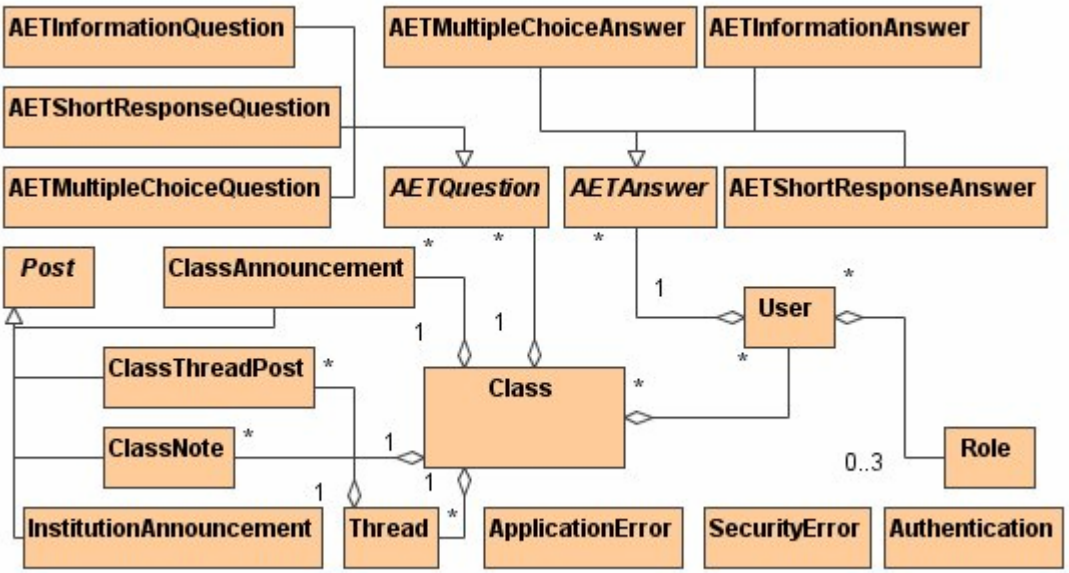


Figure 3.5: Class Diagram for the OLE Web application (UML)

3.4.3.2 Class – User

This class contains properties and methods for users.

Properties (not all listed):

- `FirstName` – The user's first name.
- `LastName` – The user's surname.
- `ResidentAddress` – The user's residential address.
- `PostalAddress` – The user's postal address.
- `EmailAddress` – The user's e-mail address.
- `PhoneNumber` – The user's phone number.

Copyright © 2004 by Timothy Vriesema

- `Password` – The user's password within the OLE.
- `UserID` – The user's unique identifier.

3.4.3.3 Abstract Class – AETAnswer

This abstract class provides base methods and properties for all assignment, exam, and tutorial answers.

Properties:

- `AETQID` – The AET's unique identifier.
- `AETType` – The type of AET.
- `ClassID` – The associated class ID.
- `Marked` – True if the answer has been marked and false if not.
- `SubmittedDateTime` – The date and time the user submitted their answer.
- `UserID` – The associated user ID.

3.4.3.4 Inherited Class – AETInformationAnswer : AETAnswer

This inherited class defines properties and methods for Information AET answers.

Properties:

- `AETAFilename` – The filename that contains the user's answer.

3.4.3.5 Inherited Class – AETMultipleChioceAnswer : AETAnswer

This inherited class defines properties and methods for Multiple Choice AET answers.

Properties:

- `Answers` – Structures that include the user's answers.

3.4.3.6 Inherited Class – AETShortResponseAnswer : AETAnswer

This inherited class defines properties and methods for Short Response AET answers.

Properties:

- `Answers` – Structures that include the user's answers.

3.4.3.7 Abstract Class – AETQuestion

This abstract class provides base methods and properties for all assignment, exam, and tutorial questions.

Properties:

- `AETQID` – The AET's unique identifier.
- `AETType` – The type of AET.
- `AmountOfQuestions` – The amount of questions within the AET.
- `ClassID`

Copyright © 2004 by Timothy Vriesema

- `Description` – The description of the AET.
- `DueDateTime` – The date and time the AET is due.
- `HaltSubmissionDateTime` – The date and time to stop allowing answers from students.
- `QuestionType` – Assignment, Exam, or Tutorial?
- `Title` – The title of the question
- `ViewMarkedAnswersDateTime` – The date and time students can view their marked answers.
- `ViewQuestionDateTime` – The date and time students can view the question

3.4.3.8 Inherited Class – AETInformationQuestion : AETQuestion

This inherited class defines properties and methods for Information AET questions.

Properties:

- `AETSFilename` –The filename of the solutions.
- `Questions` – The questions associated with the AET.

3.4.3.9 Inherited Class – AETMultipleChioceQuestion : AETQuestion

This inherited class defines properties and methods for Multiple Choice AET questions.

Properties:

- `Questions` – The questions associated with the AET.

3.4.3.10 Inherited Class – AETShortResponseQuestion : AETQuestion

This inherited class defines properties and methods for Short Response AET questions.

Properties:

- `Answers` – The correct answers entered by an instructor.
- `Questions` – The questions associated with the AET.

3.4.3.11 Class – Class

This class defines properties and methods for classes.

Properties:

- `ClassID` – The class' unique string identifier.
- `ClassName` – The class' name.
- `DenyOtherUsers` – If true, non-enrolled users cannot view the class and its functions.

3.4.3.12 Class – Role

This class defines properties and methods for roles.

Properties:

- `RoleID` – The role's name (unique identifier).

3.4.3.13 Class – Thread

This class defines properties and methods for class threads.

Properties:

- `ClassID` – The associated class ID.
- `CreatedDateTime` – The date and time created.
- `CreatorName` – The creator's name.
- `ThreadID` – The thread's unique identifier.
- `Title` – The thread's title.

3.4.3.14 Abstract Class – Post

This abstract class provides base methods and properties for all posts.

Properties:

- `AttachmentFilenames` – The file attachments associated with the post.
- `CreatedDateTime` – The date and time created.
- `CreatorName` – The creator's name.
- `Information` – The information associated with the post.
- `PostID` – The post's unique identifier.
- `UpdatedDateTime` – The date and time altered/updated.

3.4.3.15 Inherited Class – ClassAnnouncement : Post

This inherited class defines properties and methods for class announcements.

Properties:

- `Title` – The announcement's title.
- `ClassID` – The associated class ID.

3.4.3.16 Inherited Class – ClassThreadPost : Post

This inherited class defines properties and methods for class thread posts.

Properties:

- `ClassID` – The associated class ID.
- `ThreadID` – The associated thread ID.
- `Withhold` – If true, it won't be displayed to students.

3.4.3.17 Inherited Class – ClassNote : Post

This inherited class defines properties and methods for class notes.

Properties:

- `ClassID` – The associated class ID.

- `Description` – The description of the note.
- `Title` – The note's title.

3.4.3.18 Inherited Class – InstitutionalAnnouncement : Post

This inherited class defines properties and methods for institutional announcements.

Properties:

- `Title` – The announcement's title.

3.4.4 Dynamic Models

This section details dynamic models generated from the functional requirements[11].

3.4.4.1 Sequence: Assignment, Exam, and Tutorial Posting and Answering

Figure 3.6 shows the sequence in which an Instructor posts and assignment, exam, or tutorial and then how a Student interacts receives the question and posts an answer.

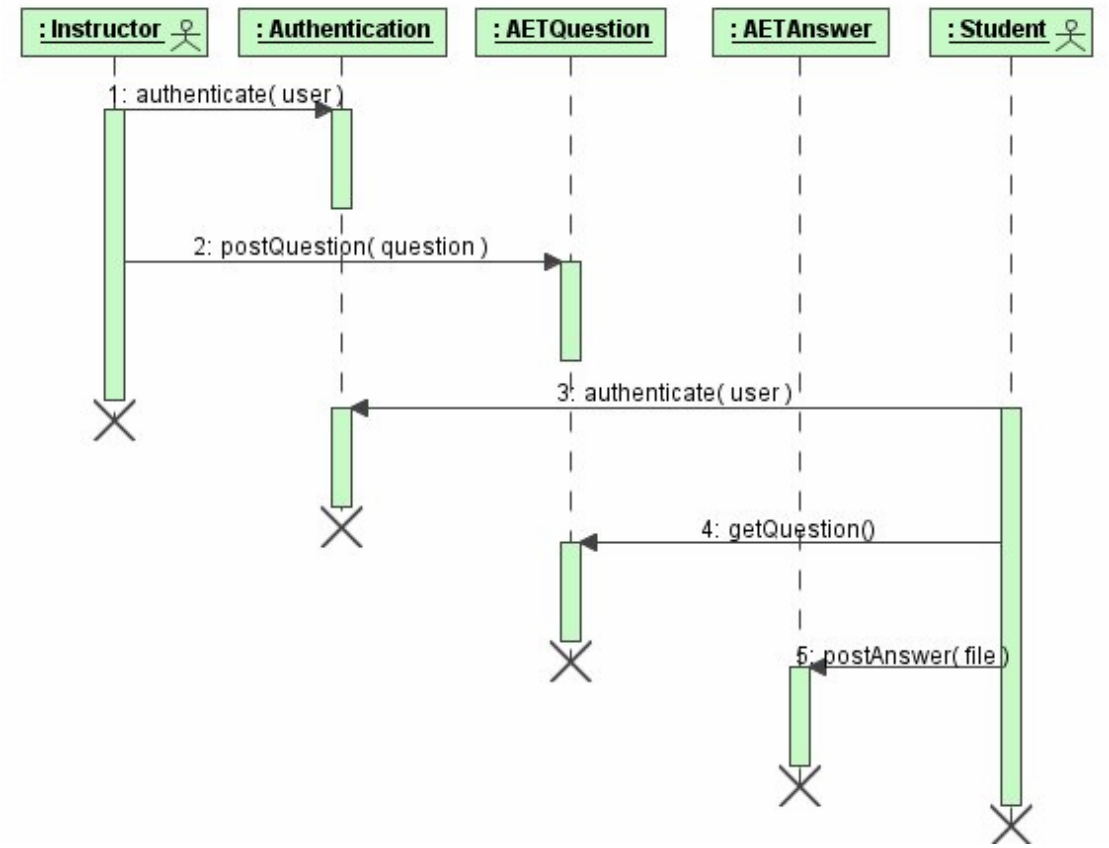


Figure 3.6: Sequence Diagram for Posting and Answering AETs (UML)

[11] See *Functional Requirements*, section 3.2

3.4.4.2 Sequence: Assignment, Exam, and Tutorial Marking

Figure 3.7 shows the sequence in which an Instructor receives an AET submission list, receives the answer file, and then posts a mark for the answer. The Student then visits and receives the mark assigned to that particular answer.

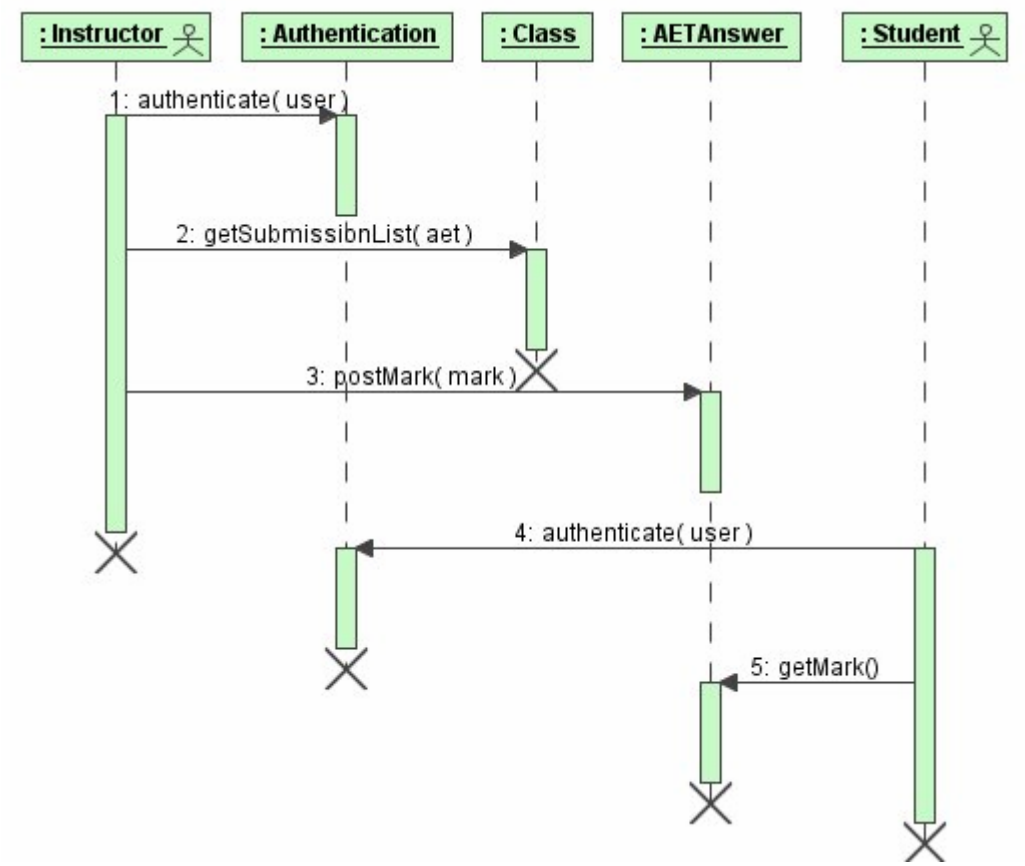


Figure 3.7: Sequence Diagram for Marking AETs (UML)

3.4.5 User Interface

Figure 3.8 shows the Web page layout for the login page when a user first visits the OLE Web application. Once passed the login procedure, Figure 3.9 shows the Web page layout for the rest of the OLE Web application. These pages will consist of hyperlinks, pictures, and input forms[12].

Figure 3.10 shows the components that will be used within the Web page forms in order to input data. They consist of:

- *Text box* – This allows users to type text that does not require more than one line.
- *Button* – This allows users to execute an operation.
- *Combo box* – This allows users to select a choice from within a box.
- *Multi-line Text box* – This allows users to type text that requires one or more lines.

[12] See, Figure 3.10: Components for Web Forms

- *Check box* – This allows users to select an option.
- *Radio button* – This allows users to choose one option from several options, similar to the *Comb box*, but all options are shown.
- *Tooltips* – This allows users to view information about a certain object as their mouse pointer pauses above it.

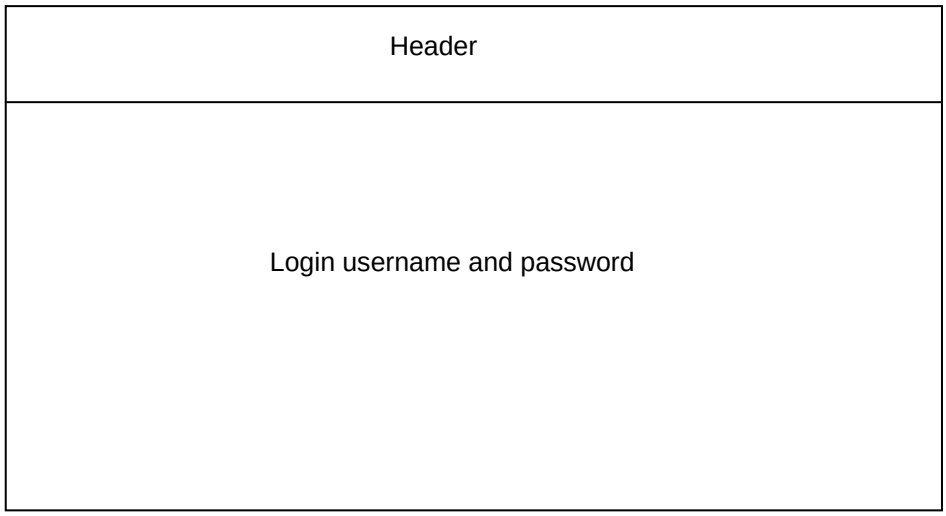


Figure 3.8: Web layout for the OLE Web application's login page

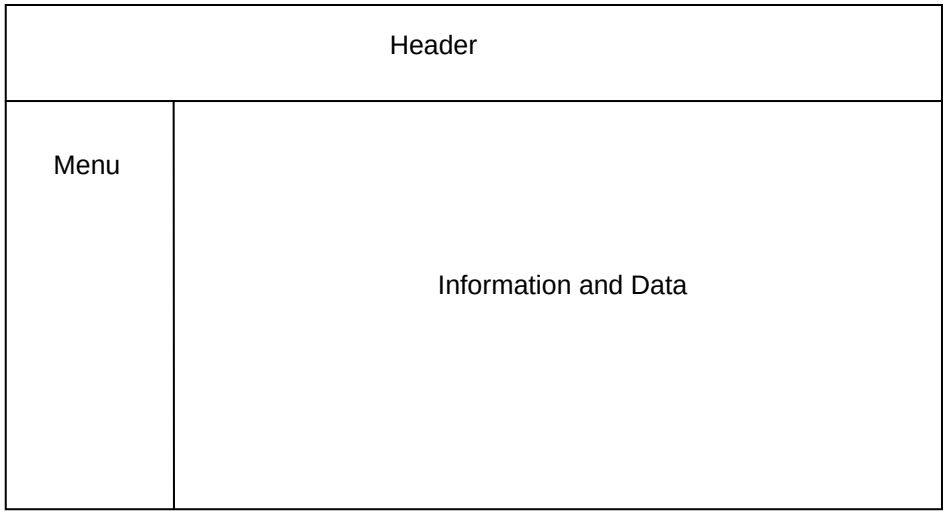


Figure 3.9: Web layout of the OLE Web application

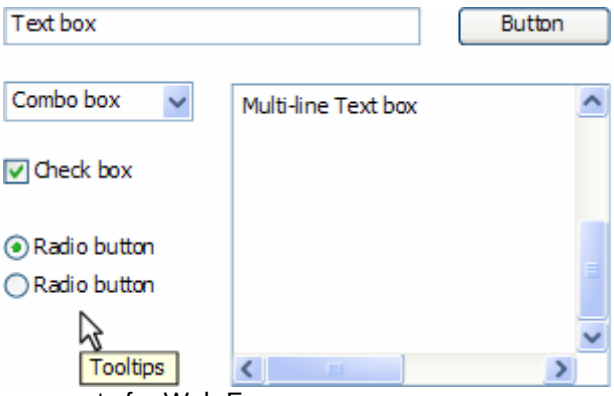


Figure 3.10: Components for Web Forms

4 GLOSSARY

Administrator	A system user that manages the software systems.
Button	This allows users to execute an operation.
Check box	This allows users to select an option.
Class	
Combo box	This allows users to select a choice from within a box.
Instructor	An individual that operates a class within the educational institution.
Multi-line text box	This allows users to type text that requires one or more lines.
Radio button	This allows users to choose one option from several options, similar to the Comb box, but all options are shown.
Role	A name describing what a group of people do.
Student	An individual that populates a class and is lead by an Instructor.
Text box	This allows users to type text that does not require more than one line.
Tooltips	This allows users to view information about a certain object as their mouse pointer pauses above it.
Web application	An application that is used via the Internet or internal Intranet.

Copyright © 2004 by Timothy Vriesema

Appendix C

System Design Document

‘Online Learning Environment’ using ASP.NET

SYSTEM DESIGN DOCUMENT

Sunday, 24 October 2004

NAME

Timothy Vriesema

tripix@msn.com

STUDENT NUMBER

0011122446

UNIVERSITY PROGRAM

Bachelor of Engineering

(Software Engineering)

AND

Bachelor of Business

(Management and Leadership)

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **I**

1 INTRODUCTION **1**

1.1 PURPOSE OF THE SYSTEM 1
1.2 DESIGN GOALS 1
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 1
1.4 OVERVIEW 1

2 PROPOSED SOFTWARE ARCHITECTURE **2**

2.1 OVERVIEW 2
2.1.1 Presentation Tier *2*
2.1.2 Business Logic Tier *3*
2.1.3 Data Access Tier *3*
2.2 SUBSYSTEM DECOMPOSITION 3
2.3 HARDWARE/SOFTWARE MAPPING 3
2.4 PERSISTENT DATA MANAGEMENT 3
2.4.1 Database *4*
2.4.2 XML File Storage *5*
2.5 ACCESS CONTROL AND SECURITY 5
2.5.1 ASP.NET Forms-Based Authentication *5*
2.5.2 Encryption *6*
2.6 GLOBAL SOFTWARE CONTROL 6
2.7 BOUNDARY CONDITIONS 6
2.7.1 Start-up Behaviours *7*
2.7.2 Error Behaviours *7*

3 SUBSYSTEM SERVICES **8**

3.1 LOGIN SUBSYSTEM 8
3.2 HOME SUBSYSTEM 8
3.2.1 Institution Announcements *8*
3.2.2 Quick Links *8*
3.2.3 Upcoming Dates *8*
3.2.4 Application and Security Errors *8*
3.2.5 Thread Subscriptions *8*
3.3 CLASS SUBSYSTEM 8
3.3.1 Assignments, Exams, and Tutorials *9*
3.3.1.1 Submission List 9
3.3.2 Class Announcements *9*
3.3.3 Notes *9*
3.3.4 Forum *9*
3.4 SUBMISSION SUBSYSTEM 9
3.5 MANAGEMENT SUBSYSTEM 9
3.5.1 Assignment, Exam, and Tutorial Management *9*
3.5.2 Class Announcement Management *9*
3.5.3 Institution Announcement Management *10*
3.5.4 User Management *10*
3.5.5 Class Management *10*
3.6 CONFIGURATION SUBSYSTEM 10
3.6.1 Application Configuration *10*
3.6.2 Institution Configuration *10*
3.6.3 Class Configuration *10*
3.6.4 User Configuration *10*
3.6.5 Role Assignment Configuration *10*
3.6.6 Class Enrolment Configuration *10*

Copyright © 2004 by Timothy Vriesema

1 INTRODUCTION

The following sub-sections will provide an introduction to the purpose of the system and design goals towards the OLE project.

1.1 Purpose of the System

The system that this document is generated from is the 'Online Learning Environment' using ASP.NET (OLE). It aims to develop a system for educational institutions and their needs. This primarily includes class management by instructors, for enrolled students.

1.2 Design Goals

Throughout the design process, design goals were set in order to give a deeper purpose to the process. The OLE project is not designed to be profitable or to be sold to organisations; it is designed as a learning experience for its developers.

1.3 Definitions, Acronyms, and Abbreviations

Before reading the *System Design Document*, some abbreviations are presented below that will be used.

AET	**A**ssignment, **E**xam, and **T**utorial
ADO.NET	Microsoft **A**ctive **D**ata **O**bjects **.NET**
ASP.NET	Microsoft **A**ctive **S**erver **P**ages **.NET**
IIS	Microsoft **I**nternet **I**nformation **S**ervice 6.0
OLE	**'O**nline **L**earning **E**nvironment**'** using ASP.NET

1.4 Overview

This document provides detail on the system design of the OLE project. It details the proposed software architecture for the project and then details the services to be provided, sorted by subsystem.

Copyright © 2004 by Timothy Vriesema

2 PROPOSED SOFTWARE ARCHITECTURE

This section details the overall architecture to the OLE. It gives an overview of the system, along with its subsystem decomposition, hardware and software mappings, persistent data management, access control and security, global software control, and boundary conditions.

2.1 Overview

The OLE has been split into a three-tier architecture. These tiers are (Figure 2.1):

- Presentation Tier
- Business Logic Tier
- Data Access Tier

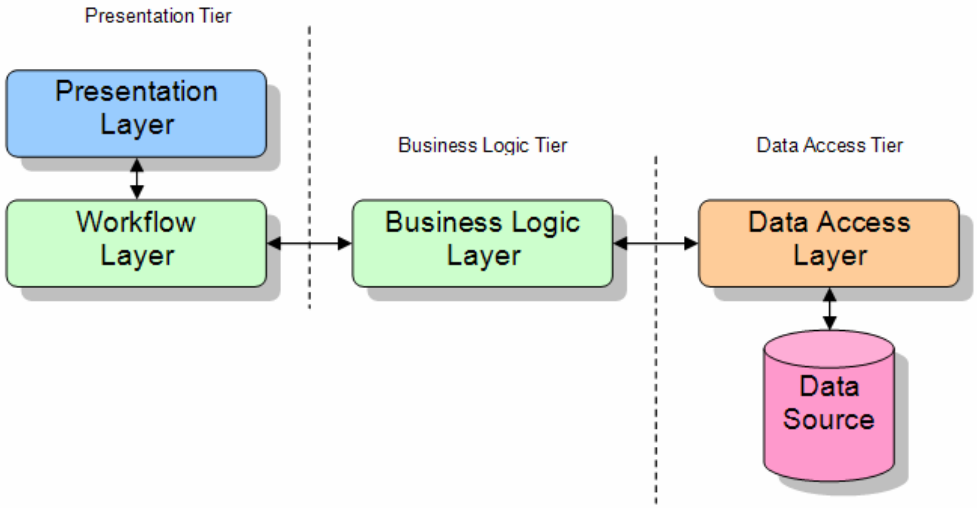


Figure 2.1: The three tiers of the 'Online Learning Environment'

2.1.1 Presentation Tier

This tier (subsystem) consists of two layers. They are:

- Presentation Layer
- Workflow Layer

The *Presentation Layer* consists of code that presents the data to the user. The languages used will be HTML and CSS.

The *Workflow Layer* consists of code that creates and manipulates data to fit within the Presentation Layer. The language used here is a mixture of C# .NET and ASP.NET

2.1.2 Business Logic Tier

This tier involves a layer called the *Business Logic Layer*. This layer consists of classes that are used within the OLE. The language used here is C# .NET.

2.1.3 Data Access Tier

This tier consists of the *Data Access Layer*. It provides classes that can realise the storage, retrieval, and query of persistent objects within the *Data Source*. The languages used here is ADO.NET and SQL.

2.2 Subsystem Decomposition

The *Presentation Tier* (section 2.1.1) is the only tier that has been split into subsystems. These subsystems are:

- *Login* – Provides an interface for users to enter username and password information in order to access the OLE.
- *Home* – Provides an interface for users to see an overview of the OLE.
- *Submission* – Provides an interface for Administrators to add users and classes to the OLE.
- *Class* – Provides an interface for users to view class related materials.
- *Management* – Provides an interface for users to manage institution and/or class materials.
- *Configuration* – Provides an interface for users to configure application, institution, user, and/or class materials.

These subsystems and their services are described in section 3.

2.3 Hardware/Software Mapping

The OLE is split between two primary systems, the client, and server. The client users access the OLE via a Web browser and the server holds all data and code to manipulate the client's responses.

2.4 Persistent Data Management

The persistent data within the OLE is stored in either or both of two ways. They are:

- Database; and/or
- XML files.

This design was chosen to allow for research in both persistent data storage methods. If the OLE were to be re-programmed, the database would be used because of its quick methods of accessing data.

Below, these two methods will be described as to how they are implemented within the OLE.

Copyright © 2004 by Timothy Vriesema

2.4.1 Database

There are three core tables within the database. They are `Classes`, `Roles`, and `Users`. This means that all other tables use the primary keys from within those three tables to make up their own primary keys.

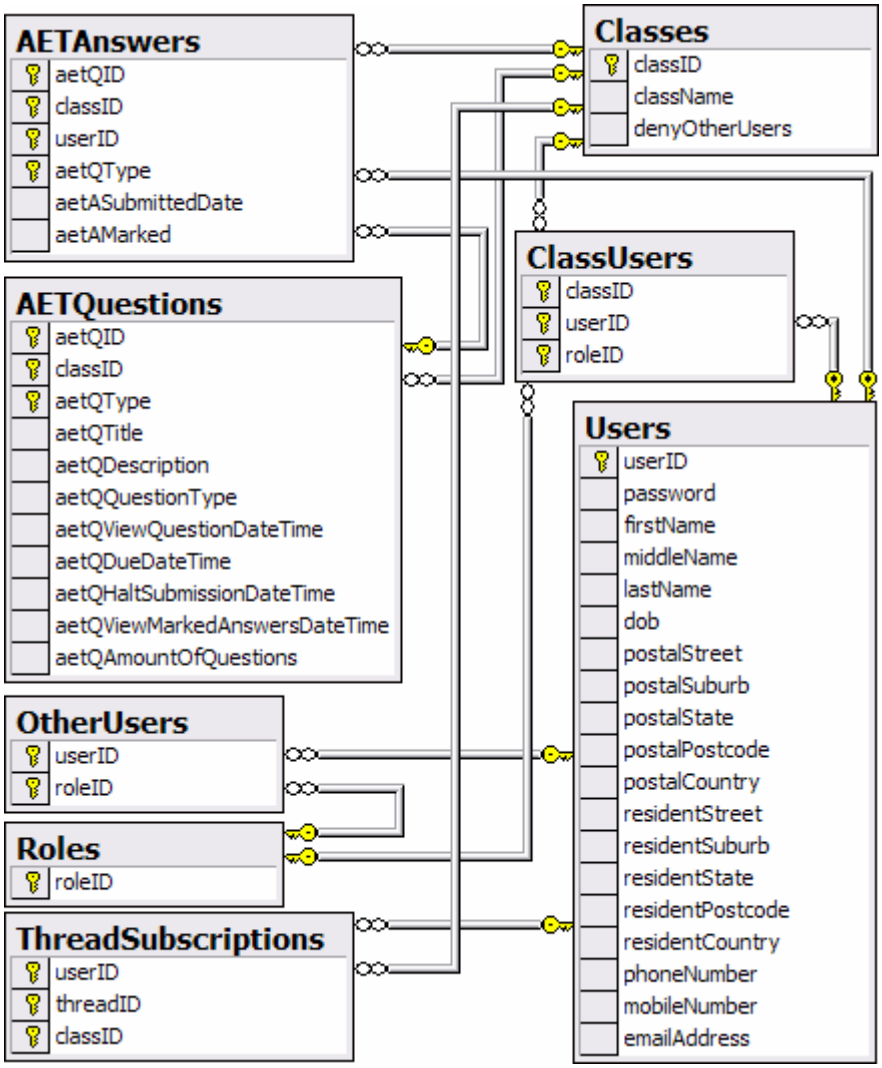


Figure 2.2: The OLE database

From Figure 2.2, there exist eight tables. They are (with descriptions):

- `AETAnswers` – Stores the submitted date and its marked status for an AET answer.
- `AETQuestions` – Stores information about an AET question (i.e. title, description, due date, etc.).
- `Classes` – Stores information about the classes that exist.
- `ClassUsers` – Stores which users belong to which classes.
- `OtherUsers` – Stores which users are Administrators.
- Roles – Stores the roles that exist.
- `ThreadSubscriptions` – Stores a user's subscription to certain threads.
- `Users` – Stores information about existing users.

2.4.2 XML File Storage

The OLE creates and manipulates many XML files. These files have to be kept in a directory on the system. This directory is unavailable to all remote users (i.e. Web clients). Figure 2.3 displays an example of the OLE directory's structure.

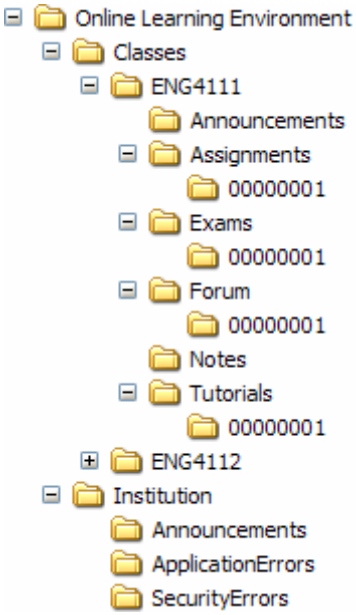


Figure 2.3: An example of the OLE data directory's structure

From the above example, two classes exist with IDs *ENG4111*, and *ENG4112*. The folder structure of *ENG4112* is identical to *ENG4111* so it has been omitted from the figure.

It can be seen that ENG4111 has six subfolders, Announcements, Assignments, Exams, Form, Notes, and Tutorials. Within each of these directories, a list of XML files exists. These XML files have incremented filenames from 1 to 99999999 (padded with 0). For example, the fifth XML file within the directory would be '00000005.xml'. The 125th file would be '00000125.xml'.

Each XML file is accompanied with a directory containing the same number as the filename. For example, '00000001.xml' would have a directory with the name '00000001'.

Note that the above is just and example of the OLEs directory structure. Some subfolders' method differs from one to another.

2.5 Access Control and Security

The following sections will describe the authentication methods used within the OLE.

2.5.1 ASP.NET Forms-Based Authentication

ASP.NET provides an authentication method through web page forms. This is known as Forms-Based Authentication.

The Forms-Based Authentication within ASP.NET can be implemented in conjunction with the Microsoft .NET interface, `System.Security.Principal.IPrincipal`. This interface is inherited by user classes so it is passed around the application.

2.5.2 Encryption

The Microsoft .NET Framework provides many different cryptography classes that can be utilised. The method that will be used for encrypting passwords is MD5.

2.6 Global Software Control

As described in section 2.1, three tiers exist within the OLE. Therefore, there must be a way for all tiers to synchronise. This is done by two methods. They are:

- Dynamic Link Libraries (DLL); and
- SQL Stored Procedures.

A SQL Stored Procedure is code that pre-caches database entries so that the data is ready for retrieval. These procedures are kept in the *Data Access Tier* and referenced by the *Business Logic Tier* via two ADO.NET classes, `System.Data.SqlClient.SqlConnection` and `System.Data.SqlClient.SqlCommand`.

The *Business Logic Tier* (Business Logic layer) is compiled into a DLL and is referenced by the *Presentation Tier*.

The *Presentation Tier* is a little more complex than the other two tiers. It has two layers, Presentation Layer and Workflow Layer. The Workflow Layer is a compiled into a DLL for the Presentation Layer to access.

The Presentation Layer is dynamically compiled by a Web server (ASP.NET compatible). It compiles the Presentation Layer by automatically referencing DLL files located within the application's `\bin` directory.

2.7 Boundary Conditions

The OLE is a Web application that not only uses a Web server; it uses the Windows registry to access configuration values. These registry values are set upon the installation of the OLE. These registry values are held under the windows registry key, `HKEY_LOCAL_MACHINE\SOFTWARE\Online Learning Environment`.

The registry values consist of:

Copyright © 2004 by Timothy Vriesema

- `DataDirectory` – The directory in which all OLE data files will be kept (default value, `C:\Program Files\Online Learning Environment`).
- `SqlServerName` – The computer name of where the Microsoft SQL Server database exists (default name, `localhost`).

2.7.1 Start-up Behaviours

For the OLE to start, the Web server requires a client to access the OLE's address. This then starts by retrieving registry values (described above) and detecting if the data directory exists. If the directory does not exist, the OLE assumes it is starting for the first time and creates the necessary directories.

If the OLE is executing for the first time, a username and password is required to logon and create new users and classes. This username is, *Administrator*. The user, *Administrator*, has the password, `OnlineLearningEnvironment` (case-sensitive).

2.7.2 Error Behaviours

All errors within the OLE are thrown by exceptions. These exceptions are captured and displayed in an easy to read manner to users other than *Administrators*. The full exception message is stored as an Application Error and shown to all *Administrators* when they visit the application.

Copyright © 2004 by Timothy Vriesema

3 SUBSYSTEM SERVICES

This section details the services that exist within each subsystem. Many of these services can be found in more detail under the *Requirements Analysis Document*.

3.1 Login Subsystem

The Login subsystem contains only one service called the Authentication service. It allows users to enter a username and password to allow access to the OLE.

3.2 Home Subsystem

The Home subsystem is where users are directed to by default after authentication by the Login subsystem. It is designed to give an overview of the other subsystems' services. These overview services are described in the sections below.

3.2.1 Institution Announcements

All users can view these announcements. They give information on events throughout the institution.

3.2.2 Quick Links

This allows the users within the Instructors and/or Students roles to have a list of links to their enrolled classes.

3.2.3 Upcoming Dates

Users within the roles Instructors and/or Students can view their upcoming AET due dates. This allows the users to plan and schedule their semester effectively.

3.2.4 Application and Security Errors

This service is only available to users within the Administrators role. It displays security and application errors.

3.2.5 Thread Subscriptions

Users within the Students role can add a thread to a list (subscribing to it) so that they can be notified of new responses quickly and without re-navigating to the thread again.

3.3 Class Subsystem

The Class subsystem consists of services that assist a class within the institution. These services allow users to interact with class activities such as assessment items and discussion forums.

Copyright © 2004 by Timothy Vriesema

3.3.1 Assignments, Exams, and Tutorials

The AET service allows students to visit and take the assessment. Instructors can then calculate the completed assessments, issuing students their performance score.

3.3.1.1 Submission List

This service allows users within the Instructors role to view whether or not students have submitted their AET and their submission dates and times.

3.3.2 Class Announcements

The class announcements service is a periodical update on events that are current within the class. All users within a class can view these announcements.

3.3.3 Notes

This service is quite similar to the class announcements service. However, notes are information on difficult questions, frequently asked questions, or class changes.

3.3.4 Forum

The forum is where instructors and students visit and create threads based on certain categories. Within these threads, any user can place a question and await an answer from another user.

3.4 Submission Subsystem

The Submission subsystem is designed to allow access to users within the Administrators role. It consists of two services:

- Class submission; and
- User submission.

These allow Administrators to add new classes and users to the OLE.

3.5 Management Subsystem

This subsystem deals with the management side of the application. This can include institution management along with class management.

3.5.1 Assignment, Exam, and Tutorial Management

This service allows instructors to manage AETs by creating, altering, and deleting them.

3.5.2 Class Announcement Management

Instructors of a class can manage these announcements by creating, altering, and deleting them.

Copyright © 2004 by Timothy Vriesema

3.5.3 Institution Announcement Management

Administrators can manage these announcements by creating, altering, and deleting them.

3.5.4 User Management

Administrators can manage all users within the institution by creating, altering, and deleting them.

3.5.5 Class Management

Administrators can manage all classes within the institution by creating, altering, and deleting them.

3.6 Configuration Subsystem

This subsystem contains services that configure different parts of the institution and application.

3.6.1 Application Configuration

This service allows administrators to alter application dependent variables such as the SQL Server database

3.6.2 Institution Configuration

The institution configuration service allows administrators to alter institutional terminology such as assignment, class, and tutorial.

3.6.3 Class Configuration

This allows instructors to alter class configurations such as whether or not to allow non-enrolled users to access particular areas.

3.6.4 User Configuration

The user configuration service allows all users to alter their personal information.

3.6.5 Role Assignment Configuration

This allows administrators to assign different roles to different users.

3.6.6 Class Enrolment Configuration

Administrators can enrol users to any class under the Instructors or Students roles.

Copyright © 2004 by Timothy Vriesema

Appendix D

Object Design Document

‘Online Learning Environment’ using ASP.NET

OBJECT DESIGN DOCUMENT

Monday, 25 October 2004

NAME

Timothy Vriesema

tripix@msn.com

STUDENT NUMBER

0011122446

UNIVERSITY PROGRAM

Bachelor of Engineering

(Software Engineering)

AND

Bachelor of Business

(Management and Leadership)

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **I**

1 INTRODUCTION **1**

1.1 OBJECT DESIGN TRADE-OFFS 1
1.2 INTERFACE DOCUMENTATION GUIDELINES 1
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 1
1.4 OVERVIEW 1

2 PACKAGES **2**

2.1 LOGIN SUBSYSTEM 2
2.2 HOME SUBSYSTEM 2
2.3 CLASS SUBSYSTEM 2
2.4 SUBMISSION SUBSYSTEM 3
2.5 MANAGEMENT SUBSYSTEM 3
2.6 CONFIGURATION SUBSYSTEM 4
2.7 MENU PACKAGE 4

3 CLASS INTERFACES **5**

3.1 ONLINELEARNINGENVIRONMENT 5
3.1.1 AddressLimitations *5*
3.1.2 AddressStructure *5*
3.1.3 AETAnswer *6*
3.1.4 AETQuestion *6*
3.1.5 AETInformationAnswer *7*
3.1.6 AETInformationQuestion *8*
3.1.7 AETLimitations *8*
3.1.8 AETMultipleChoiceAnswer *9*
3.1.9 AETMultipleChoiceQuestion *9*
3.1.10 AETShortResponseAnswer *9*
3.1.11 AETShortResponseQuestion *10*
3.1.12 AETType *10*
3.1.13 ApplicationError *11*
3.1.14 Authentication *11*
3.1.15 Class *12*
3.1.16 ClassAnnouncement *13*
3.1.17 ClassLimitations *13*
3.1.18 ClassNote *13*
3.1.19 ClassThreadPost *14*
3.1.20 Configuration *14*
3.1.21 DataBridge *15*
3.1.22 FileLimitations *18*
3.1.23 InformationQuestion *18*
3.1.24 InstitutionAnnouncement *18*
3.1.25 MD5Encryption *19*
3.1.26 MultipleChoiceAnswer *19*
3.1.27 MultipleChoiceChoice *19*
3.1.28 MultipleChoiceQuestion *19*
3.1.29 Post *20*
3.1.30 QuestionType *20*
3.1.31 Role *21*
3.1.32 RoleID *21*
3.1.33 RoleLimitations *22*
3.1.34 SecurityError *22*
3.1.35 ShortAETAnswer *23*

Copyright © 2004 by Timothy Vriesema

3.1.36 ShortAETQuestion 23
3.1.37 ShortResponseAnswer 24
3.1.38 ShortResponseQuestion 24
3.1.39 Thread 25
3.1.40 User 26
3.1.41 UserLimitations 27

1 INTRODUCTION

The following sub-sections will provide an introduction to the object design trade-offs and interface documentation guidelines towards the OLE project.

1.1 Object Design Trade-Offs

While designing the OLE project, particular attention was given to the persistent storage component. Since the project was undertaken as a learning activity, efficiency was sacrificed by using two storage techniques rather than one.

1.2 Interface Documentation Guidelines

Throughout the implementation of the OLE project, different naming conventions carry different components within the source code.

- C# Namespaces – `OnlineLearningEnvironment`
- C# Structure – `OnlineLearningEnvironment`
- C# Enumeration – `OnlineLearningEnvironment`
- C# Classes – `OnlineLearningEnvironment`
- C# Properties – `OnlineLearningEnvironment`
- C# Attribute – `onlineLearningEnvironment`
- XML Element – `OnlineLearningEnvironment`
- XML Attribute – `onlineLearningEnvironment`
- SQL Table – `OnlineLearningEnvironment`
- SQL Attribute – `onlineLearningEnvironment`

1.3 Definitions, Acronyms, and Abbreviations

Before reading the *Test Manual*, some abbreviations are presented below that will be used.

AET	**A**ssignment, **E**xam, and **T**utorial
ADO.NET	Microsoft **A**ctive **D**ata **O**bjects **.NET**
ASP.NET	Microsoft **A**ctive **S**erver **P**ages **.NET**
IIS	Microsoft **I**nternet **I**nformation **S**ervice 6.0
OLE	**'O**nline **L**earning **E**nvironment**'** using ASP.NET

1.4 Overview

This document provides detail on the object design of the OLE project. Firstly it details the packages inside each subsystem and secondly it describes the class interfaces.

Copyright © 2004 by Timothy Vriesema

2 PACKAGES

This section details the packages found inside the 'OLE'. However, since the application is split into three tiers, only the Presentation tier contains packages.

In the following sub-sections, each subsystem will be split into their respective packages. Note that since the Presentation tier is, in essence, an ASP.NET Web application, each subsystem is an .aspx file. Packages are split into .ascx files that are included within the subsystem's .aspx file.

2.1 Login Subsystem

The Login subsystem is contained in the file Login.aspx. The following packages are located within this subsystem:

- *Authentication* – Authenticates a user to access the OLE.
    - Authentication.ascx

2.2 Home Subsystem

The Home subsystem is contained in the file Default.aspx. The following packages are located within this subsystem:

- *Error Messages* – Displays recent error messages (Administrators only).
    - ErrorMessages.ascx
- *Forum Thread Subscriptions* – Displays a user's subscriptions to threads.
    - ForumThreadSubscriptions.ascx
- *Institition Announcements* – Displays announcements for the institution.
    - InstititionAnnouncements.ascx
- *Quick Links* – Displays the classes a user is enrolled in.
    - QuickLinks.ascx
- *Upcoming Dates* – Displays upcoming AET dates.
    - UpcomingDates.ascx

2.3 Class Subsystem

The Class subsystem is contained in the file Class.aspx. The following packages are located within this subsystem:

- *AET Submission List* – Displays the users who have submitted for a specific AET.
    - AETSubmissionList.ascx
- *Class Announcements* – Displays the announcements for a class.
    - ClassAnnouncements.ascx
- *Information AET Question* – Display an Information AET question for a specific class.

Copyright © 2004 by Timothy Vriesema

    - InformationAETQuestion.ascx
- *Multiple-Choice AET Question* – Display a Multiple-Choice AET question for a specific class.
    - MultipleChoiceAETQuestion.ascx
- *Short-Response AET Question* – Display a Short-Response AET question for a specific class.
    - ShortResponseAETQuestion.ascx
- *Short-Response AET Answers* – Displays the answers for a Short-Response AET question.
    - ShortResponseAETAnswers.ascx
- *Thread Posts* – Displays thread posts for a particular thread.
    - ThreadPosts.ascx
- *Threads* – Displays the threads for a particular class.
    - Threads.ascx

2.4 Submission Subsystem

The Submission subsystem is contained in the file Submission.aspx. The following packages are located within this subsystem:

- *User Submission* – Submit a new user.
    - User.ascx
- *Class Submission* – Submit a new class.
    - Class.ascx

2.5 Management Subsystem

The Management subsystem is contained in the file Management.aspx. The following packages are located within this subsystem:

- *Class Announcement Management* – Manages the class announcements.
    - ClassAnnouncementManagement.ascx
- *Class Management* – Manages the classes.
    - ClassManagement.ascx
- *Class Note Management* – Manages the class notes.
    - ClassNoteManagement.ascx
- *Information AET Management* – Manages the Information AETs for a class.
    - InformationAETManagement.ascx
- *Institution Announcement Management* – Manages the institution announcements.
    - InstitutionAnnouncementManagement.ascx
- *Multiple-Choice AET Management* – Manages the Multiple-Choice AETs for a class.
    - MultipleChoiceAETManagement.ascx
- *Short-Response AET Management* – Manages the Short-Response AETs for a class.
    - ShortResponseAETManagement.ascx
- *User Management* – Manages the users.
    - UserManagement.ascx

2.6 Configuration Subsystem

The Configuration subsystem is contained in the file Configuration.aspx. The following packages are located within this subsystem:

- *Application Configuration* – Configure the Web application.
  - ApplicationConfiguration.ascx
- *Class Configuration* – Configure a class.
  - ClassConfiguration.ascx
- *Class Enrollment Configuration* – Configure class user enrolments.
  - ClassEnrollmentConfiguration.ascx
- *Institution Configuration* – Configure the institution.
  - InstitutionConfiguration.ascx
- *Role Assignment Configuration* – Configure a user's roles.
  - RoleAssignmentConfiguration.ascx
- *User Configuration* – Configure a user's personal details.
  - UserConfiguration.ascx

2.7 Menu Package

The Menu package is contained in the file Menu.ascx. The following packages are located within this package:

- *Assignments* – List a user's assignments for a specific class.
  - Assignments.ascx
- *Classes* – List a user's classes.
  - Classes.ascx
- *Exams* – List a user's exams for a specific class.
  - Exams.ascx
- *Notes* – List a user's notes for a specific class.
  - Notes.ascx
- *Tutorials* – List a user's notes for a specific class.
  - Tutorials.ascx

Copyright © 2004 by Timothy Vriesema

3 CLASS INTERFACES

This section details all classes that exist within the 'OLE', along with descriptions on constructors, destructors, attributes, properties, and methods.

3.1 OnlineLearningEnvironment

The `OnlineLearningEnvironment` namespace contains classes from within the Business Logic Tier. They are concerned with describing and storage of data information.

3.1.1 AddressLimitations

Filename	Limitations.cs
Type	C# Structure
Description	Property limitations for structure, AddressStructure.

Attributes

`const int Street`	
	The maximum string length of the combined street number, name, and type
`const int Suburb`	
	The maximum string length of the suburb name
`const int State`	
	The maximum string length of the state name
`const int Postcode`	
	The maximum string length of the postcode number
`const int Country`	
	The maximum string length of the country name

3.1.2 AddressStructure

Filename	Address.cs
Type	C# Structure
Description	The structure of a postal/residential Address.

Attributes

`string Street`	
	Street number and name.
`string Suburb`	
	Suburb/City.
`string State`	
	Suburb/City.
`string Postcode`	
	Postcode of the country.
`string Country`	
	Country.

Copyright © 2004 by Timothy Vriesema

3.1.3 AETAnswer

Filename	AETAnswer.cs
Type	C# Abstract Class
Description	An abstract class, which provides shared properties and methods for AET answers

Constructors

`AETAnswer()`	
	Used for XmlSerializer purposes
`AETAnswer(DataBridge)`	
	Abstract constructor

Properties

`int UserID`	
	The UserID associated with the AETAnswer
`int AETQID`	
	The ID of the AETQuestion associated with the AETAnswer
`AETType AETType`	
	The AETType of the associated AETQuestion
`string ClassID`	
	The ClassID associated class
`DateTime SubmittedDateTime`	
	The date that the answer was submitted
`bool Marked`	
	Whether or not the AETAnswer has been completely marked
`DataBridge DataBridge`	
	The DataBridge that connects to the data

Methods

`Create()`	
	Uses current instance's properties and stores the data
`Alter()`	
	Uses current instance's properties and re-stores the data
`Delete()`	
	Uses the current instance's properties and deletes the stored data

3.1.4 AETQuestion

Filename	AETQuestion.cs
Type	C# Abstract Class
Description	An abstract class which provides shared properties and methods for AETQuestions.

Constructors

`AETQuestion(QuestionType)`	
	Abstract constructor.
`AETQuestion(AETType, QuestionType, DataBridge)`	
	Abstract constructor

Copyright © 2004 by Timothy Vriesema

Properties

`int AETQID`	
	The unique ID.
`AETType AETType`	
	The associated type.
`string AmountOfQuestions`	
	The ClassID associated class
`string ClassID`	
	The associated class ID.
`QuestionType QuestionType`	
	The associated question type.
`string Title`	
	The associated title.
`string Description`	
	The associated description (ususally 'Assignment', 'Exam', or 'Tutorial').
`DateTime DueDateTime`	
	The date & time the answers are due.
`DateTime HaltSubmissionsDateTime`	
	The date & time no more answers are accepted.
`DateTime ViewMarkedAnswersDateTime`	
	The date & time students are able to see their marked answers.
`DateTime ViewQuestionDateTime`	
	The date & time students can view the question.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

Methods

`Create()`	
	Uses current instance's properties and stores the data.
`Alter()`	
	Uses current instance's properties and re-stores the data.
`Delete()`	
	Uses the current instance's properties and deletes the stored data.

3.1.5 AETInformationAnswer

Filename	AETAnswer.cs
Type	C# Sealed Class
Description	An inherited class of AETAnswer for Information answers
Inherits	`AETAnswer`

Constructors

`AETInformationAnswer()`	
	Used for XmlSerializer purposes
`AETInformationAnswer(DataBridge)`	
	Main constructor

Copyright © 2004 by Timothy Vriesema

Properties

`int AETAFilename`	
	The filename of where the answer exists

Static Methods

`FindAETInformationAnswer()`	
	Finds an AETInformationAnswer

3.1.6 AETInformationQuestion

Filename	AETQuestion.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of an Information Question.
Inherits	`AETQuestion`

Constructors

`AETInformationQuestion()`	
	Used for XmlSerializer purposes
`AETInformationQuestion(AETType, DataBridge)`	
	Main constructor

Properties

`InformationQuestion[] Questions`	
	An array of information questions.
`string AETSFilename`	
	The filename of the provided solutions file.

Static Methods

`FindAETInformationQuestion()`	
	Find a specific AETInformationQuestion.

3.1.7 AETLimitations

Filename	Limitations.cs
Type	C# Structure
Description	Property limitations for classes, AETQuestion and AETAnswer.

Attributes

`const int Title`	
	The maximum string length of the title.
`const int Description`	
	The maximum string length of the description.
`const int AETType`	
	The maximum string length of the AETType.
`const int AETQuestionType`	
	The maximum string length of the AETQuestionType.

Copyright © 2004 by Timothy Vriesema

3.1.8 AETMultipleChoiceAnswer

Filename	AETAnswer.cs
Type	C# Sealed Class
Description	An inherited class of AETAnswer for Multiple Choice answers
Inherits	`AETAnswer`

Constructors

`AETMultipleChoiceAnswer()`

Used for XmlSerializer purposes

`AETMultipleChoiceAnswer(DataBridge)`

Main constructor

Attributes

`MultipleChoiceAnswer[] answers`

An array of answers

Static Methods

`FindAETMultipleChoiceAnswer()`

Finds an AETMultipleChoiceAnswer

3.1.9 AETMultipleChoiceQuestion

Filename	AETQuestion.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of a Multiple Choice Question.
Inherits	`AETQuestion`

Constructors

`AETMultipleChoiceQuestion()`

Used for XmlSerializer purposes

`AETMultipleChoiceQuestion(AETType, DataBridge)`

Main constructor

Properties

`MultipleChoiceQuestion[] Questions`

An array of multiple choice questions.

Static Methods

`FindAETMultipleChoiceQuestion()`

Find a specific AETMultipleChoiceQuestion.

3.1.10 AETShortResponseAnswer

Filename	AETAnswer.cs
Type	C# Sealed Class
Description	An inherited class of AETAnswer for Short Response answers
Inherits	`AETAnswer`

Copyright © 2004 by Timothy Vriesema

Constructors

`AETShortResponseAnswer()`	
	Used for XmlSerializer purposes
`AETShortResponseAnswer(DataBridge)`	
	Main constructor

Attributes

`ShortResponseAnswer[] answers`	
	An array of answers

Static Methods

`FindAETShortResponseAnswer()`	
	Finds an AETShortResponseAnswer

3.1.11 AETShortResponseQuestion

Filename	AETQuestion.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of a Short Response Question.
Inherits	`AETQuestion`

Constructors

`AETShortResponseQuestion()`	
	Used for XmlSerializer purposes
`AETShortResponseQuestion(AETType, DataBridge)`	
	Main constructor

Properties

`ShortResponseQuestion[] Questions`	
	An array of short response questions.
`ShortResponseAnswer[] Answers`	
	An array of short resposnse question's answers (entered by Instructors).

Static Methods

`FindAETShortResponseQuestion()`	
	Find a specific AETShortResponseQuestion.

3.1.12 AETType

Filename	AETQuestion.cs
Type	C# Enumeration
Description	The type of the AETQuestion.

Enumerates

`Assignment`	
	An assignment.
`Exam`	
	An exam.

Copyright © 2004 by Timothy Vriesema

`Tutorial`	
	A tutorial.
`NotSet`	
	Enum hasn't be set.

3.1.13 ApplicationError

Filename	ErrorAndSecurity.cs
Type	C# Sealed Class
Description	A class that holds information about application errors.

Constructors

`ApplicationError()`	
	Used for XmlSerializer purposes.
`ApplicationError(DataBridge)`	
	Main constructor.

Static Methods

`FindApplicationError()`	
	Find a specific application error.
`FindApplicationErrors()`	
	Find all application errors.

Methods

`Create()`	
	Creates an instance of the error as an xml file.
`Delete()`	
	Delete the specified error ID.

Properties

`int ErrorID`	
	The ID of the error.
`string Information`	
	The ID of the associated user.
`DateTime CreatedDateTime`	
	The date and time the error was created.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

3.1.14 Authentication

Filename	Authentication.cs
Type	C# Sealed Class
Description	A sealed class that contains static methods to check authentication.

Constructors

`Authentication()`	
	Disabled constructor.
`Authentication(DataBridge)`	

Main constructor

Properties

`DataBridge DataBridge`

The DataBridge that connects to the data.

Static Methods

`CheckAuthentication()`

Check username and password.

3.1.15 Class

Filename	Class.cs
Type	C# Sealed Class
Description	A sealed class that contains methods and properties for classes.

Constructors

`Class()`

Used for XmlSerializer purposes.

`Class(string, DataBridge)`

Main Constructor.

Properties

`string ClassID`

The unique ID.

`string ClassName`

The associated class name.

`bool DenyOtherUsers`

Whether or not non-enrolled users can visit the class' section.

Methods

`Create()`

Uses current instance's properties and stores the data.

`Alter()`

Uses current instance's properties and re-stores the data.

`Delete()`

Uses the current instance's properties and deletes the stored data.

`EnrolUser()`

Enrol a user to the class.

`DeEnrolUser()`

DeEnrol a user to the class.

Static Methods

`IsInClass()`

Find all roles a user is enrolled in.

`FindClass()`

Find a class.

`FindClasses()`

Copyright © 2004 by Timothy Vriesema

Find the classes associated with a user.

3.1.16 ClassAnnouncement

Filename	Post.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of a class announcement.

Constructors

`ClassAnnouncement()`

Used for XmlSerializer purposes.

`ClassAnnouncement(string, DataBridge)`

Main Constructor.

Properties

`string ClassID`

The associated class ID.

`string Title`

The associated title.

Static Methods

`FindClassAnnouncement()`

Find a specific class announcement.

`FindClassAnnouncements()`

Finds all class announcements for a specific class.

3.1.17 ClassLimitations

Filename	Limitations.cs
Type	C# Structure
Description	Property limitations for class, Class

Attributes

`const int ClassID`

The maximum string length of the class ID

`const int ClassName`

The maximum string length of the class name

3.1.18 ClassNote

Filename	Post.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of a class note.

Constructors

`ClassNote()`

Used for XmlSerializer purposes.

`ClassNote(string, DataBridge)`

Main Constructor.

Properties

`string ClassID`	
	The associated class ID.
`string Title`	
	The associated title.
`string Description`	
	The associated description.

Static Methods

`FindClassNote()`	
	Find a specific class note.
`FindClassNotes()`	
	Finds all class notes for a specific class.

3.1.19 ClassThreadPost

Filename	Post.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of a class thread post.

Constructors

`ClassThreadPost()`	
	Used for XmlSerializer purposes.
`ClassThreadPost(string, int, DataBridge)`	
	Main Constructor.

Properties

`string ClassID`	
	The associated class ID.
`int ThreadID`	
	The associated thread ID.
`bool Withhold`	
	Whether or not to show this thread post.

Static Methods

`FindClassThreadPost()`	
	Find a specific class thread post.
`FindClassThreadPost()`	
	Finds all class thread posts for a specific class.

3.1.20 Configuration

Filename	Configuration.cs
Type	C# Sealed Class
Description	A sealed class that contains methods and properties for configuration.

Constructors

`Configuration()`	

Used for XmlSerializer purposes.

`Configuration(DataBridge)`

Main Constructor.

Properties

`string InstitutionName`

The institution name.

`string InstitutionNameAbbreviation`

The institution name's abbreviation.

`string InstitutionLogoImage`

The institution's logo image for the web site header.

`string Institution`

Institution terminology.

`string Class`

Class terminology.

`string Assignment`

Assignment terminology.

`string Exam`

Exam terminology.

`string Tutorial`

Tutorial terminology.

`string Note`

Note terminology.

`DataBridge DataBridge`

The DataBridge that connects to the data.

Methods

`Create()`

Uses current instance's properties and stores the data.

`Alter()`

Uses current instance's properties and re-stores the data.

Static Methods

`FindConfiguration()`

Find configuration.

3.1.21 DataBridge

Filename	DataBridge.cs
Type	C# Sealed Class
Description	The class that provides properties and methods to connect to Xml files and an SQL Server database.

Constructors

`DataBridge(SqlConnection, string, string, bool)`

Main Constructor.

`DataBridge()`

Disabled Constructor.

Properties

Copyright © 2004 by Timothy Vriesema

SqlConnection SqlConnection	
	The connection to an SQL Server database.
string ClassesDirectory	
	The directory that holds all class critical Xml files.
string InstitutionDirectory	
	The directory that holds all institution critical Xml files.

Methods

GetClassAnnouncementFilePath()	
	Get class announcement's file path.
GetClassAnnouncementsDirectory()	
	Get the class' announcements directory.
GetClassAssignmentAnswerFilePath()	
	Get a 'new' class assignment answer's file path.
GetClassAssignmentAnswersDirectory()	
	Get the class' assignment answer directory.
GetClassAssignmentFilePath()	
	Get class assignment's file path.
GetClassAssignmentsDirectory()	
	Get the class' assignments directory.
GetClassComponentDirectory()	
	Gets a class component's directory.
GetClassDirectory()	
	Get a class' files directory.
GetClassExamAnswerFilePath()	
	Get a 'new' class exam answer's file path.
GetClassExamAnswersDirectory()	
	Get the class' exam answer directory.
GetClassExamFilePath()	
	Get class exam's file path.
GetClassExamsDirectory()	
	Get the class' exams directory.
GetClassNoteFilePath()	
	Get class note's file path.
GetClassNotesDirectory()	
	Get the class' notes directory.
GetClassSubComponentDirectory()	
	Gets a class component's sub-directory.
GetClassThreadFilePath()	
	Get class thread's file path.
GetClassThreadPostFilePath()	
	Get class thread post's file path.
GetClassThreadPostsDirectory()	
	Get the class thread post's directory.
GetClassThreadsDirectory()	
	Get the class' threads directory.
GetClassTutorialAnswerFilePath()	
	Get a 'new' class tutorial answer's file path.
GetClassTutorialAnswersDirectory()	
	Get the class' tutorial answer directory.
GetClassTutorialFilePath()	

Copyright © 2004 by Timothy Vriesema

Get class tutorial's file path.

GetClassTutorialsDirectory()

Get the class' tutorials directory.

GetConfigurationFilePath()

Get configuration file path.

GetFilePath()

Retrieves a file path with a certain name for a file.

GetFilePath()

Retrieves a file path with a certain number.

GetInstitutionAnnouncementFilePath()

Gets a file path within the institution announcement directory.

GetInstitutionAnnouncementsDirectory()

Gets the institution's announcement directory.

GetInstitutionApplicationErrorFilePath()

Gets a file path within the institution application errors directory.

GetInstitutionApplicationErrorsDirectory()

Gets the institution's application errors directory.

GetInstitutionComponentDirectory()

Gets an institution component's directory.

GetInstitutionSecurityErrorFilePath()

Gets a file path within the institution security errors directory.

GetInstitutionSecurityErrorsDirectory()

Gets the institution's security errors directory.

GetNextClassAnnouncementFilePath()

Get next file path to exist within the institution security errors directory.

GetNextClassAssignmentFilePath()

Get a 'new' class assignment's file path.

GetNextClassExamFilePath()

Get a 'new' class exam's file path.

GetNextClassNoteFilePath()

Get a 'new' class note's file path.

GetNextClassThreadFilePath()

Get a 'new' class thread's file path.

GetNextClassThreadPostFilePath()

Get a 'new' class thread post's file path.

GetNextClassTutorialFilePath()

Get a 'new' class tutorial's file path.

GetNextFilePath()

Retrieves a 'new' file path with a numeric number above the last in the directory (+1).

GetNextInstitutionAnnouncementFilePath()

Get next file path to exist within the institution announcement directory.

GetNextInstitutionApplicationErrorFilePath()

Get next file path to exist within the application errors directory.

GetNextInstitutionSecurityErrorFilePath()

Get next file path to exist within the institution security errors directory.

Copyright © 2004 by Timothy Vriesema

`GetRootDirectory()`	
	Gets the root data file directory (the directory before the class and institution directory).

3.1.22 FileLimitations

Filename	Limitations.cs
Type	C# Structure
Description	Limitations for filenames used within the OLE.

Attributes

`const int FileNameLength`	
	The character length of a filename
`const char PadCharacter`	
	The character used to pad the filename

3.1.23 InformationQuestion

Filename	AETQuestion.cs
Type	C# Structure
Description	Properties that belong to an InformationQuestion.

Attributes

`int questionID`	
	The question's unique ID.
`string title`	
	The question's title.
`string question`	
	The question's question.

3.1.24 InstitutionAnnouncement

Filename	Post.cs
Type	C# Sealed Class
Description	A sealed class that contains properties and methods of a institution announcement.

Constructors

`InstitutionAnnouncement()`	
	Used for XmlSerializer purposes.
`InstitutionAnnouncement(DataBridge)`	
	Main Constructor.

Properties

`string Title`	
	The associated title.

Static Methods

`FindInstitutionAnnouncement()`	
	Find a specific institution announcement.
`FindInstitutionAnnouncement()`	

Copyright © 2004 by Timothy Vriesema

	Finds all institution announcements.

3.1.25 MD5Encryption

Filename	MD5Encryption.cs
Type	C# Class
Description	This class is used to encrypt strings of data to an MD5 hash string.

Static Methods

`EncryptString()`	
	Encrypts a string to a MD5 hash in the form of a byte array.
`EncryptStringToByteString()`	
	Encrypts a string into a readable MD5 hash string.

3.1.26 MultipleChoiceAnswer

Filename	AETQuestion.cs
Type	C# Structure
Description	Properties that belong to a MultipleChoiceQuestion.

Attributes

`int questionID`	
	The QuestionID of the AETQuestion.
`int choiceID`	
	The answer choiceID from the AETQuestion's choices.
`bool answeredCorrectly`	
	Assigned true by the application equal to AETQuestion.Questions.correctChoice.

3.1.27 MultipleChoiceChoice

Filename	AETQuestion.cs
Type	C# Structure
Description	Properties that belong to a MultipleChoiceQuestion's choice.

Attributes

`int choiceID`	
	The choice's unique ID.
`string choiceText`	
	The choice's text.

3.1.28 MultipleChoiceQuestion

Filename	AETQuestion.cs
Type	C# Structure
Description	Properties that belong to an MultipleChoiceQuestion.

Attributes

`int questionID`	
	The question's unique ID.

`string title`	
	The question's title.
`string question`	
	The question's question.
`bool correctChoice`	
	True if the answer is correct, false if otherwise.
`MultipleChoiceChoice[] choices`	
	An array of choices to accompany the question.

3.1.29 Post

Filename	Post.cs
Type	C# Abstract Class
Description	An abstract class which provides shared properties and methods for posts.

Constructors

`Post()`	
	Used for XmlSerializer purposes.
`Post(string, DataBridge)`	
	Abstract Constructor.

Properties

`int PostID`	
	The unique ID.
`string CreatorName`	
	The name of the creator.
`DateTime CreatedDateTime`	
	The date & time created.
`DateTime UpdatedDateTime`	
	The date & time updated.
`string Information`	
	The associated information.
`string[] AttachmentFilenames`	
	An array of attachment filenames.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

Methods

`Create()`	
	Uses current instance's properties and stores the data.
`Alter()`	
	Uses current instance's properties and re-stores the data.
`Delete()`	
	Uses the current instance's properties and deletes the stored data.

3.1.30 QuestionType

Filename	AETQuestion.cs
Type	C# Enumeration

Copyright © 2004 by Timothy Vriesema

Description	The AETQuestion's question type.

Enumerates

`Information`	
	Information on the assessment questions.
`ShortResponse`	
	Short response assessment questions.
`MultipleChoice`	
	Multiple choice assessment questions.

3.1.31 Role

Filename	Role.cs
Type	C# Sealed Class
Description	A class which holds properties and methods that define a role.

Constructors

`Role()`	
	Used for XmlSerializer purposes.
`Role(int, DataBridge)`	
	Main constructor.

Static Methods

`FindRoles()`	
	Find all roles associated with a user's ID.
`FindRolesIDs()`	
	Find all roles associated with a user's ID.

Methods

`AssignUser()`	
	Assign a specific user to a role.
`AssignUserToClass()`	
	Assign a specific user to a role within a specific class.
`UnassignUser()`	
	Unassign a specific user to a role.
`UnassignUserToClass()`	
	Unassign a specific user from a role within a specific class.

Properties

`string RoleID`	
	A string ID that defines a role.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

3.1.32 RoleID

Filename	Role.cs
Type	C# Enumeration
Description	An enumeration which holds which roles exist within

Copyright © 2004 by Timothy Vriesema

	the OLE.

Enumerates

`Administrators`	
	The highest authenticated users within the OLE.
`Instructors`	
	Users that controls Students within the OLE.
`Students`	
	The lowest authenticated users within the OLE.

3.1.33 RoleLimitations

Filename	Limitations.cs
Type	C# Structure
Description	Property limitations for class, Role.

Attributes

`const int RoleID`	
	The maximum string length of a role ID.

3.1.34 SecurityError

Filename	ErrorAndSecurity.cs
Type	C# Sealed Class
Description	A class that holds information about security errors.

Constructors

`SecurityError()`	
	Used for XmlSerializer purposes.
`SecurityError(DataBridge)`	
	Main constructor.

Static Methods

`FindSecurityError()`	
	Find a specific security error.
`FindSecurityErrors()`	
	Find all security errors.

Methods

`Create()`	
	Creates an instance of the error as an xml file.
`Delete()`	
	Delete the specified error ID.

Properties

`int SecurityID`	
	The ID of the error.
`int UserID`	
	The ID of the associated user.
`string IPAddress`	

	The value of the associated IP address.
`DateTime CreatedDateTime`	
	The date and time the error was created.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

3.1.35 ShortAETAnswer

Filename	AETAnswer.cs
Type	C# Structure
Description	A structure similar to AETAnswer, but without the answer properties.

Static Methods

`FindShortAETAnswer()`	
	Finds an AETInformationAnswer
`FindShortAETAnswers()`	
	Finds an AETInformationAnswer

Properties

`int UserID`	
	The UserID associated with the AETAnswer
`int AETQID`	
	The ID of the AETQuestion associated with the AETAnswer
`AETType AETType`	
	The AETType of the associated AETQuestion
`string ClassID`	
	The ClassID associated class
`DateTime SubmittedDateTime`	
	The date that the answer was submitted
`bool Marked`	
	Whether or not the AETAnswer has been completely marked
`DataBridge DataBridge`	
	The DataBridge that connects to the data

3.1.36 ShortAETQuestion

Filename	AETQuestion.cs
Type	C# Structure
Description	A structure similar to AETQuestion, but without the question properties.

Static Methods

`FindShortAETQuestion()`	
	Finds a specific ShortAETQuestion.
`FindShortAETQuestions()`	
	Finds the properties of AETQuestions for a particular Class ID (without the questions).

Properties

`int AETQID`	

Copyright © 2004 by Timothy Vriesema

The unique ID.

`AETType AETType`

The associated type.

`string AmountOfQuestions`

The ClassID associated class

`string ClassID`

The associated class ID.

`QuestionType QuestionType`

The associated question type.

`string Title`

The associated title.

`string Description`

The associated description (ususally 'Assignment', 'Exam', or 'Tutorial').

`DateTime DueDateTime`

The date & time the answers are due.

`DateTime HaltSubmissionsDateTime`

The date & time no more answers are accepted.

`DateTime ViewMarkedAnswersDateTime`

The date & time students are able to see their marked answers.

`DateTime ViewQuestionDateTime`

The date & time students can view the question.

`DataBridge DataBridge`

The DataBridge that connects to the data.

3.1.37 ShortResponseAnswer

Filename	AETQuestion.cs
Type	C# Structure
Description	The structure of a short response answer.

Attributes

`int questionID`

The QuestionID of the AETQuestion.

`string answer`

A written answer to the question.

`bool answeredCorrectly`

An Instructor will set true if answer is correct and false if incorrect.

3.1.38 ShortResponseQuestion

Filename	AETQuestion.cs
Type	C# Structure
Description	Properties that belong to a ShortResponseQuestion.

Attributes

`int questionID`

The question's unique ID.

`string title`

Copyright © 2004 by Timothy Vriesema

	The question's title.
`string question`	
	The question's question.

3.1.39 Thread

Filename	Thread.cs
Type	C# Sealed Class
Description	A sealed class that contains methods and properties for classes.

Constructors

`Thread()`	
	Used for XmlSerializer purposes.
`Thread(DataBridge)`	
	Main Constructor.

Properties

`int ThreadID`	
	The ID associated to the thread.
`string ClassID`	
	The ID of the class associated to the thread.
`string Title`	
	The title of the thread.
`string CreatorName`	
	The thread creator's full name.
`DateTime CreatedDateTime`	
	The date and time the thread was created.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

Methods

`Create()`	
	Uses current instance's properties and stores the data.
`Alter()`	
	Uses current instance's properties and re-stores the data.
`Delete()`	
	Uses the current instance's properties and deletes the stored data.
`CreateThreadSubscription()`	
	Creates a subscription to a thread for a user.
`DeleteThreadSubscription()`	
	Deletes a subscription to a thread for a user.

Static Methods

`FindThread()`	
	Find a specific thread that belongs within a specific class.
`FindAllThreads()`	
	Finds all threads that belong to a specific class ID.

Copyright © 2004 by Timothy Vriesema

3.1.40 User

Filename	User.cs
Type	C# Sealed Class
Description	A sealed class that contains methods and properties for classes.
Implemented	`IPrincipal`

Constructors

`User()`	
	Used for XmlSerializer purposes.
`User(DataBridge)`	
	Main Constructor.
`User(IIdentity, DataBridge)`	
	Main Constructor.
`User(int, IIdentity, DataBridge)`	
	Main Constructor.

Properties

`int UserID`	
	The user's numeric ID. Also used as username.
`string Password`	
	The user's password in MD5 encryption form.
`SetEncryptedPassword`	
	Used to set and convert the user's password from a string to an MD5 encryption string.
`string[] RoleIDs`	
	The user's roles.
`string FirstName`	
	The user's first name.
`string MiddleName`	
	The user's middle name.
`string LastName`	
	The user's last name.
`string FullName`	
	Gets the user's full name from the first, middle, and last name properties.
`DateTime DOB`	
	The user's date of birth.
`AddressStructure PostalAddress`	
	The user's postal address.
`AddressStructure ResidentialAddress`	
	The user's residential address.
`string PhoneNumber`	
	The user's home phone number.
`string MobileNumber`	
	The user's mobile phone number.
`string EmailAddress`	
	The user's email address.
`DataBridge DataBridge`	
	The DataBridge that connects to the data.

Copyright © 2004 by Timothy Vriesema

Methods

`Create()`	
	Uses current instance's properties and stores the data.
`Alter()`	
	Uses current instance's properties and re-stores the data.
`Delete()`	
	Uses the current instance's properties and deletes the stored data.

Static Methods

`FindUser()`	
	Find a specific user.
`FindUsers()`	
	Find all users within a specific class.

3.1.41 UserLimitations

Filename	Limitations.cs
Type	C# Structure
Description	Property limitations for class, User

Attributes

`const int Password`	
	The maximum string length of the password
`const int FirstName`	
	The maximum string length of the first name
`const int MiddleName`	
	The maximum string length of the middle name
`const int LastName`	
	The maximum string length of the last name
`const int PhoneNumber`	
	The maximum string length of the phone number
`const int EmailAddress`	
	The maximum string length of the email address

Copyright © 2004 by Timothy Vriesema

Appendix E

Test Manual

‘Online Learning Environment’ using ASP.NET

TEST MANUAL

Saturday, 16 October 2004

NAME

Timothy Vriesema

tripix@msn.com

STUDENT NUMBER

0011122446

UNIVERSITY PROGRAM

Bachelor of Engineering

(Software Engineering)

AND

Bachelor of Business

(Management and Leadership)

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **I**

1 INTRODUCTION **1**

2 TESTING **2**

2.1 UNIT TESTING 5
2.1.1 User *5*
2.1.2 Class *7*
2.1.3 MD5Encryption *10*
2.1.4 Authentication *11*
2.1.5 Configuration *11*
2.1.6 Role *13*
2.1.7 DataBridge *14*
2.1.8 ApplicationError *14*
2.1.9 SecurityError *14*
2.1.10 Thread *14*
2.1.11 AETInformationAnswer : AETAnswer *14*
2.1.12 AETMultipleChoiceAnswer : AETAnswer *14*
2.1.13 AETShortResponseAnswer : AETAnswer *15*
2.1.14 AETInformationQuestion : AETQuestion *15*
2.1.15 AETMultipleChoiceQuestion : AETQuestion *15*
2.1.16 AETShortResponseQuestion : AETQuestion *15*
2.1.17 InstitutionAnnouncement : Post *15*
2.1.18 ClassAnnouncement : Post *15*
2.1.19 ClassNote : Post *15*
2.1.20 ClassThreadPost : Post *15*
2.2 INTEGRATION TESTING 15
2.2.1 Login Site *15*
2.2.2 Default Site *15*
2.2.3 Class Site *15*
2.2.4 Configuration Site *15*
2.2.5 Management Site *15*
2.2.6 Submission Site *16*

APPENDIX A **17**

TESTING SOURCE CODE 17

1 INTRODUCTION

This document has been generated in order to perfect the quality of the release source code. It consists of testing that attempts to ‘break and repair’ faulty source code.

The document starts with testing the different units within the application, while re-coding any faulty portions of the code. Once all units have been tested, the units are integrated into the presentation environment, where faulty presentation or logic code is re-coded to reflect the release result of the application.

Copyright © 2004 by Timothy Vriesema

2 TESTING

This section will test the different components within the application. All tests assume that the data from previous tests exists. For example, the first test is of the `User` class. When the `Class` class is tested, it will assume that the test data created from the User class exists.

Test Data

The test data to be used for these tests are as follows. This data should also exist once all tests are complete.

Type	*Data*
Class 1	ClassID: ENG4111 ClassName: Research Project 1 DenyOtherUsers: true
Class 2	ClassID: ENG4112 ClassName: Research Project 2 DenyOtherUsers: false
Class Announcement 1	ClassID: ENG4111 Information: “This is the first class announcement for ENG4111.” FileAttachments: None CreatorName: Timothy Ross Vriesema CreatedDateTime: Now UpdatedDateTime: Now
Class Announcement 2	ClassID: ENG4111 Information: “This is the second announcement for ENG4111.” FileAttachments: None CreatorName: Timothy Ross Vriesema CreatedDateTime: Now UpdatedDateTime: Now
Assignment 1	ClassID: ENG4111 AETType: Assignment QuestionType: Information AmountOfQuestions: 2 Title: Colours of the Earth Description: Assignment 1 DueDateTime: Now + 1month HaltSubmissionsDateTime: Now + 1month + 1 day ViewMarkedAnswersDateTime: Now + 2months ViewQuestionDateTime: Now
Exam 1	ClassID: ENG4111 AETType: Exam QuestionType: Multiple Choice AmountOfQuestions: 2

Copyright © 2004 by Timothy Vriesema

	Title: Colours of Animals Description: Exam 1 DueDateTime: Now + 1month HaltSubmissionsDateTime: Now + 1month + 1 day ViewMarkedAnswersDateTime: Now + 2months ViewQuestionDateTime: Now
Exam 2	ClassID: ENG4112 AETType: Exam QuestionType: Multiple Choice AmountOfQuestions: 2 Title: Definitions Description: Exam 1 DueDateTime: Now + 1month HaltSubmissionsDateTime: Now + 1month + 1 day ViewMarkedAnswersDateTime: Now + 2months ViewQuestionDateTime: Now
Tutorial 1	ClassID: ENG4111 AETType: Tutorial QuestionType: Short Response AmountOfQuestions: 2 Title: Colours of Materials Description: Tutorial 1 DueDateTime: Now + 1month HaltSubmissionsDateTime: Now + 1month + 1 day ViewMarkedAnswersDateTime: Now + 2months ViewQuestionDateTime: Now
User 1	FirstName: Timothy MiddleName: Ross LastName: Vriesema EmailAddress: tim@home.com DOB: 20/05/1983 Password: tim PhoneNumber: 1234567890 MobileNumber: 1234567890 Street: 1 Main Street Suburb: ASuburb State: SomeState Postcode: 1000 Country: Australia
User 2	FirstName: Jonathan LastName: Trinder EmailAddress: jon@home.com DOB: 18/05/1983 Password: jon PhoneNumber: 0987654321 MobileNumber: 0987654321 Street: 50 Main Street Suburb: ASuburb State: SomeState Postcode: 1000

	Country: Australia
Institution Announcement 1	Information: “This is an institution announcement.” FileAttachments: None CreatorName: Administrator CreatedDateTime: Now UpdatedDateTime: Now

Stored Procedures (Microsoft SQL Server)

Extensive testing of this tier has already been accomplished throughout the Business Logic tier. The SQL Server stored procedures were used within each class. This is mapped out in the Table 2.1 below.

Table 2.1: Mapped SQL stored procedures to Business Logic Tier classes

Stored Procedure	***Associated Business Logic Tier Classes***
AlterAETAnswer	`AETAnswer.Alter()`
AlterAETQuestion	`AETQuestion.Alter()`
AlterClass	`Class.Alter()`
AlterUser	`User.Alter()`
AssignUser	`Role.AssignUser()`
AssignUserToClass	`Role.AssignUserToClass()` `Class.EnrolUser()`
Authentication	`Authentication.CheckAuthentication()`
CreateAETAnswer	`AETAnswer.Create()`
CreateAETQuestion	`AETQuestion.Create()`
CreateClass	`Class.Create()`
CreateThreadSub	`Thread.Create()`
CreateUser	`User.Create()`
UnassignUser	`Role.UnassignUser()`
UnassignUserToClass	`Role.UnassignUserToClass()`
DeleteAETAnswer	`AETAnswer.Delete()`
DeleteAETQuestion	`AETQuestion.Delete()`
DeleteClass	`Class.Delete()`
DeleteThreadSub	`Thread.Delete()`
DeleteUser	`User.Delete()`
FindClass	`Class.FindClass()`
FindClasses	`Class.FindClasses()`
FindRolesIDs	`Role.FindRolesIDs()`
FindShortAETAnswer	`ShortAETAnswer.FindShortAETAnswer()`
FindShortAETAnswers	`ShortAETAnswer.FindShortAETAnswers()`
FindShortAETQuestion	`ShortAETQuestion.FindShortAETQuestion()`
FindShortAETQuestions	`ShortAETQuestion.FindShortAETQuestions()`
FindUser	`User.FindUser()`
FindUsers	`User.FindUsers()`
IsInClass	`Class.IsInClass()`

Copyright © 2004 by Timothy Vriesema

2.1 Unit Testing

This section is designed to test each individual class as a unit, rather than an integrated system. This is to 'iron' out the majority of the programming faults that may exist before integration testing.

If unit testing is not completed prior to the integration testing, faults may be more difficult to locate.

Each sub-section is split into three sections. They are:

- *Specification* – Provides criteria that must succeed in order for the test to be successful.
- *Description* – Provides information on how the test is carried out.
- *Analysis* – Provides an analysis of the test's output, from both the console application and persistent data.

2.1.1 User

This class holds a user's information. Within this test, two users will be added to the database, while one will be altered once it has been entered.

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
    a. `Create();`
    b. `Alter();`
    c. `Delete();`
    d. `FindUser();`
2. User 2's mobile number should change from 1 to 0987654321.
3. User 2's duplicate should be deleted.
4. An exception should be thrown when User 2's duplicate cannot be found.
5. User1 and User 2 must have different (but incremental) user IDs.

Description

The data used for this test is User 1 and User 2.

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```
User user1 = new User(db);
user1.FirstName = "Timothy";
user1.MiddleName = "Ross";
user1.LastName = "Vriesema";
user1.MobileNumber = "1234567890";
user1.PhoneNumber = "1234567890";
user1.EmailAddress = "tim@home.com";
user1.DOB = new DateTime(1983, 5, 20);
```


Copyright © 2004 by Timothy Vriesema

```
user1.SetEncryptedPassword = "tim";
AddressStructure a1 = new AddressStructure();
a1.Street = "1 Main Street";
a1.Suburb = "ASuburb";
a1.State = "SomeState";
a1.Postcode = "1000";
a1.Country = "Australia";
user1.PostalAddress = user1.ResidentAddress = a1;
int userID1 = user1.Create(); // Add user

User user2 = new User(db);
user2.FirstName = "Jonathan";
user2.MiddleName = "";
user2.LastName = "Trinder";
user2.MobileNumber = "1"; // should be 0987654321
user2.PhoneNumber = "0987654321";
user2.EmailAddress = "jon@home.com";
user2.DOB = new DateTime(1983, 5, 18);
user2.SetEncryptedPassword = "jon";
AddressStructure a2 = new AddressStructure();
a2.Street = "1 Main Street";
a2.Suburb = "ASuburb";
a2.State = "SomeState";
a2.Postcode = "1000";
a2.Country = "Australia";
user2.PostalAddress = user2.ResidentAddress = a2;
int userID2 = user2.Create(); // Add user

int userID3 = user2.Create(); // Create a duplicate
User duplicateUser = User.FindUser(userID3, db);
duplicateUser.Delete(); // Delete duplicate
try
{
    User checkUser = User.FindUser(userID3, db);
}
catch // Exception thrown, userID not found
{
    Console.WriteLine("User not found!\n");
}

User alterUser = User.FindUser(userID2, db);
alterUser.MobileNumber = "0987654321";
alterUser.Alter(); // Alter the user's data

User[] foundUsers = new User[2];
foundUsers[0] = User.FindUser(userID1, db);
foundUsers[1] = User.FindUser(userID2, db);
foreach(User u in foundUsers)
{
    Console.WriteLine(u.UserID + " : " + u.FullName + ";" +
        u.MobileNumber);
}
```

Analysis

Below is the screen output from the following test.

```
User not found!

25 : Timothy Ross Vriesema;1234567890
26 : Jonathan Trinder;0987654321
```

It shows that User 1 was given the user ID of 25 and User 2 was given the ID of 26 (different but incremental, specification 5). User 2's mobile number was altered (specification 2).

When the duplicate user was deleted (specification 3), a message was displayed when it could not be found (specification 4).

No errors were found while invoking methods listed under specification 1a through to 1d.

Table 2.2: Data within the database table User after the test

SQL Server Data – User table	
1	64-4B-45-7A-92-E4-66-A1-EB-14-33-0E-CE-BE-A6-B9
25	B1-5D-47-E9-98-31-EE-63-E3-F4-7C-F3-D4-47-8E-9A Timothy Ross Vriesema 20/05/1983 1 Main Street ASuburb SomeState 1000 Australia 1 Main Street ASuburb SomeState 1000 Australia 1234567890 1234567890 tim@home.com
26	00-6C-B5-70-AC-DA-B0-E0-BF-C8-E3-DC-B7-BB-4E-DF Jonathan Trinder 18/05/1983 1 Main Street ASuburb SomeState 1000 Australia 1 Main Street ASuburb SomeState 1000 Australia 0987654321 0987654321 jon@home.com

2.1.2 Class

This class holds details of a certain institution class. Within this test, two classes will be added to the database, while a user is enrolled in both classes.

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
    a. `Create();`
    b. `Alter();`
    c. `Delete();`
    d. `FindClass();`
    e. `FindClasses();`
    f. `EnrolUser();`
    g. `DeEnrolUser();`
2. Class 2's name should change from "Research 2" to "Research Project 2".
3. Duplication of an existing class ID should be denied and an exception should be thrown.
4. An exception should be thrown when a class ID cannot be found.
5. Enrolment of User 1 and User 2 in the two classes has no problems.

Copyright © 2004 by Timothy Vriesema

Description

The data used for this test is Class 1 and Class 2.

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```
Class class1 = new Class("ENG4111", db);
class1.ClassName = "Research Project 1";
class1.DenyOtherUsers = true;
string classID1 = class1.Create(); // Create class

class1.EnrolUser(25, RoleID.Instructors); // Enrol instructor
class1.DeEnrolUser(25, RoleID.Instructors); // DeEnrol instru
class1.EnrolUser(26, RoleID.Students); // Enrol student

Class class2 = new Class("ENG4112", db);
class2.ClassName = "Research 2"; // Left out "Project"
class2.DenyOtherUsers = false;
string classID2 = class2.Create(); // Create class
try
{
    class2.Create(); // Duplicate class
}
catch (Exception e) // Thrown exception, class id found
{
    Console.WriteLine(e.Message);
}

class2.EnrolUser(26, RoleID.Students); // Enrol student

Class alterClass = Class.FindClass(classID2, db);
alterClass.ClassName = "Research Project 2";
alterClass.Alter(); // Alter class name

try
{
    Class foundClass = Class.FindClass("FAKE", db);
}
catch // Thrown exception when class not found
{
    Console.WriteLine("Class not found!");
}

    // Entered and altered classes
Console.WriteLine("Entered and altered classes:");
Class[] foundClasses = new Class[2];
foundClasses[0] = Class.FindClass(classID1, db);
foundClasses[1] = Class.FindClass(classID2, db);
foreach (Class c in foundClasses)
{
    Console.WriteLine(c.ClassID + " : " + c.ClassName);
}

    // User 26's classes
Console.WriteLine("User 26's classes:");
Class[] usersClasses = Class.FindClasses(26, db);
foreach (Class c in usersClasses)
{
    Console.WriteLine(c.ClassID + " : " + c.ClassName);
```

```
}
```

Analysis

Below is the screen output from the following test.

```
ClassID exists!

Class not found!

Entered and altered classes:
ENG4111 : Research Project 1
ENG4112 : Research Project 2

User 26's classes:
ENG4111 : Research Project 1
ENG4112 : Research Project 2
```

The two classes (1 & 2) were added while Class 2's name was altered from "Research 2" to "Research Project 2" (specification 2). When an attempt to duplicate Class 2 was made, an exception was thrown (specification 3).

When a fake class ID was asked to be found, an exception was thrown, satisfying specification 4.

User 1 was enroled as an Instructor to Class 1 and User 2 was enroled as a Student under both Class 1 and Class 2 (specification 5).

No errors were found while invoking methods listed under specification 1a through to 1g.

Table 2.3: Data within the database table Classes after the test

SQL Server Data – Classes table		
ENG4111	Research Project 1	1
ENG4112	Research Project 2	0

Table 2.4: Data within the database table ClassUsers after the test

SQL Server Data – ClassUsers table		
ENG4111	26	Students
ENG4112	26	Students

Table 2.5: Directories found within the \Classes directory after the test

OLE Data Directory – \Classes
\ENG4111
\ENG4111\Announcements
\ENG4111\Assignments
\ENG4111\Exams
\ENG4111\Forum
\ENG4111\Notes
\ENG4111\Tutorials
\ENG4112
\ENG4112\Announcements

\ENG4112\Assignments
\ENG4112\Exams
\ENG4112\Forum
\ENG4112\Notes
\ENG4112\Tutorials

2.1.3 MD5Encryption

This class holds a static method that allows a password to be converted to an encrypted string. Within this test, three strings (two identical and one different) will be compared against each other.

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
    a. `EncryptStringToByteString();`
2. Three encrypted strings that will be tested against each other. They are:
    a. Password
    b. Password
    c. password
3. Case-sensitivity.

Description

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```
string[] e = new string[3];
e[0] = MD5Encryption.EncryptStringToByteString("Password");
e[1] = MD5Encryption.EncryptStringToByteString("Password");
e[2] = MD5Encryption.EncryptStringToByteString("password");
foreach (string s in e)
{
    Console.WriteLine(s);
}
Console.WriteLine("e[0] vs e[1] : " +
    (e[0].CompareTo(e[1]) == 0 ? "Equal" : "Not equal"));
Console.WriteLine("e[0] vs e[2] : " +
    (e[0].CompareTo(e[2]) == 0 ? "Equal" : "Not equal"));
```

Analysis

Below is the screen output from the following test.

```
DC-64-7E-B6-5E-67-11-E1-55-37-52-18-21-2B-39-64
DC-64-7E-B6-5E-67-11-E1-55-37-52-18-21-2B-39-64
5F-4D-CC-3B-5A-A7-65-D6-1D-83-27-DE-B8-82-CF-99
e[0] vs e[1] : Equal
e[0] vs e[2] : Not equal
```

The encrypted strings were identical for the first and second, but not for the third (specification 2). This concludes that there is case-sensitivity (specification 3).

No errors were found while invoking methods listed under specification 1a.

2.1.4 Authentication

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
    a. `CheckAuthentication();`
2. The password for User 1 matches the password that exists in the database.

Description

The data used for this test is User 1.

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```
    // Correct authentication
bool a = Authentication.CheckAuthentication(
    25,
    "tim",
    db);
    // Incorrect authentication
bool b = Authentication.CheckAuthentication(
    25,
    "somefakepassword",
    db);

Console.WriteLine("\"tim\" : " +
    (a ? "Correct" : "Incorrect"));
Console.WriteLine("\"somefakepassword\" : " +
    (b ? "Correct" : "Incorrect"));
```

Analysis

Below is the screen output from the following test.

```
"tim" : Correct
"somefakepassword" : Incorrect
```

The password, “tim”, is the same as the password shown under User 1’s data (specification 2).

No errors were found while invoking methods listed under specification 1a.

2.1.5 Configuration

This class stores information about the application and institution’s configuration, such as terminology values and SQL Server names. Within this test, a configuration file will be created and its institution name will be altered.

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
    a. `Alter();`
    b. `Create();`
    c. `FindConfiguration();`
2. When created without any properties set, no error occurs and default values are created.
3. The XML file containing the data is created under the Data Directory.

Description

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```
Configuration config1 = new Configuration(db);
Console.WriteLine("Default: " + config1.InstitutionName);
config1.Create(); // Create data

    // Retrieve data
Configuration config2 = Configuration.FindConfiguration(db);
config2.InstitutionName = "New Institution Name";
config2.Alter();

    // Re-retrieve data
config2 = Configuration.FindConfiguration(db);
Console.WriteLine("New: " + config2.InstitutionName);
```

Analysis

Below is the screen output from the following test.

```
Default: Online Learning Environment
New: New Institution Name
```

When the file was created, default values were entered without an error (specification 2). This file was also created under the data directory (specification 3).

No errors were found while invoking methods listed under specification 1a through to 1c.

Below displays the new data within the XML file.

```
<?xml version="1.0" encoding="utf-8"?>
<Configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <InstitutionName>New Institution Name</InstitutionName>
    <InstitutionNameAbbreviation>
        OLE
    </InstitutionNameAbbreviation>
    <InstitutionLogoImage>
        Images\header.jpg
    </InstitutionLogoImage>
    <Institution>Institution</Institution>
```

```
    <Class>Class</Class>
    <Assignment>Assignment</Assignment>
    <Exam>Exam</Exam>
    <Tutorial>Tutorial</Tutorial>
    <Note>Note</Note>
</Configuration>
```

2.1.6 Role

This class holds the name of a role. Within this test, users will be added to different roles and then have the roles revoked.

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
    a. `AssignUserToClass();`
    b. `UnassignUserToClass();`
    c. `AssignUser();`
    d. `UnassignUser();`
    e. `FindRoles();`
    f. `FindRoleIDs();`
2. User 1 should be an Instructor of Class 1 and an Administrator.
3. User 1's roles should be retrieved.
4. The roles of User 1 should be taken away.

Description

The data used for this test is User 1 and Class 1.

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```
Role role1 = new Role(RoleID.Administrators, db);
role1.AssignUser(25); // Add user to role

Role role2 = new Role(RoleID.Instructors, db);
role2.AssignUserToClass(25, "ENG4111");

    // Get RoleIDs
Console.WriteLine("25's RoleIDs:");
RoleID[] roleIDs1 = Role.FindRolesIDs(25, db);
foreach (RoleID r in roleIDs1)
{
    Console.WriteLine(r.ToString());
}

    // Get Roles
Console.WriteLine("25's Roles:");
Role[] roles1 = Role.FindRoles(25, db);
foreach (Role r in roles1)
{
    Console.WriteLine(r.RoleID.ToString());
}

role1.UnassignUser(25);
```

```
role2.UnassignUserToClass(25, "ENG4111");
    // Get Roles
Console.WriteLine("25's Roles:");
Role[] roles2 = Role.FindRoles(25, db);
Console.WriteLine(roles2 == null ? "None" : "Error");
```

Analysis

Below is the screen output from the following test.

```
25's RoleIDs:
Administrators
Instructors
25's Roles:
Administrators
Instructors
25's Roles:
None
```

User 1 was put in the role of Instructor for Class 1 and an Administrator (specification 2). The roles User 1' was in were retrieved and displayed to the screen (specification 3) and then the two roles were taken away (specification 4).

No errors were found while invoking methods listed under specification 1a through to 1f.

Table 2.6: Data within the database table ClassUsers after the test

SQL Server Data – ClassUsers table

Table 2.7: Data within the database table OtherUsers after the test

SQL Server Data – OtherUsers table	
1	Administrators

2.1.7 DataBridge

Untested.

2.1.8 ApplicationError

Untested.

2.1.9 SecurityError

Untested.

2.1.10 Thread

Untested.

2.1.11 AETInformationAnswer : AETAnswer

Untested.

2.1.12 AETMultipleChoiceAnswer : AETAnswer

Untested.

2.1.13 AETShortResponseAnswer : AETAnswer

Untested.

2.1.14 AETInformationQuestion : AETQuestion

Untested.

2.1.15 AETMultipleChoiceQuestion : AETQuestion

Untested.

2.1.16 AETShortResponseQuestion : AETQuestion

Untested.

2.1.17 InstitutionAnnouncement : Post

Untested.

2.1.18 ClassAnnouncement : Post

Untested.

2.1.19 ClassNote : Post

Untested.

2.1.20 ClassThreadPost : Post

Untested.

2.2 Integration Testing

This section will integrate the classes into a working environment in order to locate presentation and programming faults.

Each sub-section will be presented in its release code structure in order to perfect the end result.

2.2.1 Login Site

Untested.

2.2.2 Default Site

Untested.

2.2.3 Class Site

Untested.

2.2.4 Configuration Site

Untested.

2.2.5 Management Site

Untested.

2.2.6 Submission Site

Untested.

APPENDIX A

TESTING SOURCE CODE

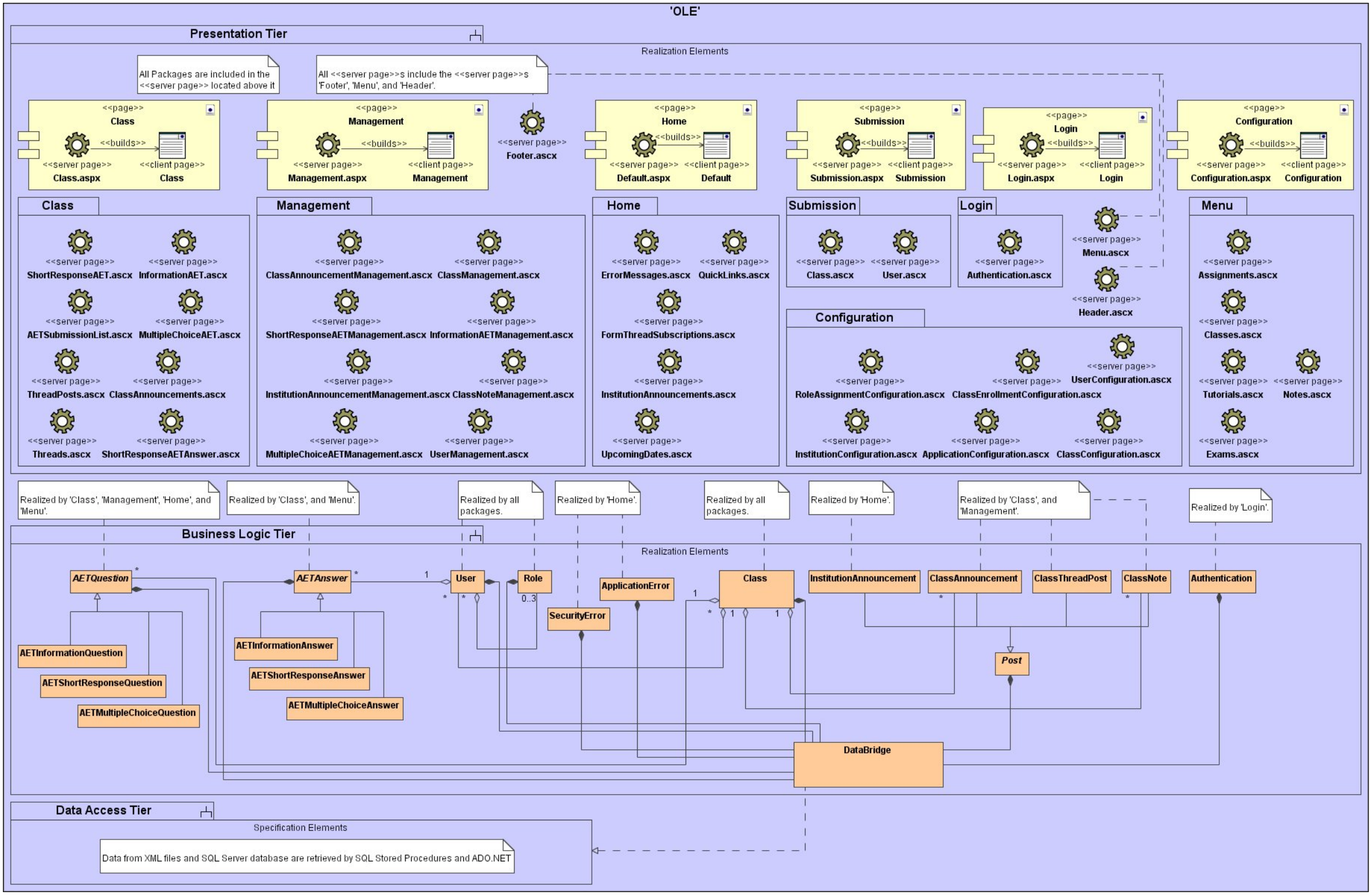
```
using OnlineLearningEnvironment;
using System;

namespace OLETestingConsoleApplication
{
    class OLETest
    {
        static void Main(string[] args)
        {
                // Standard Connection string to the SQL
                // Server database
            string conn = "server=localhost;" +
                "Trusted_Connection=true;database=oledb";
                // The shared DataBridge
                // Folders already exist
            DataBridge db = new DataBridge(
                new System.Data.SqlClient.SqlConnection(conn),
                @"C:\Online Learning Environment\Classes",
                @"C:\Online Learning Environment\Institution",
                true
                );

            try
            {
                //**** Test code is entered in here ****//
            }
            catch (Exception e)
            {
                    // All error data is written to screen
                Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Appendix F

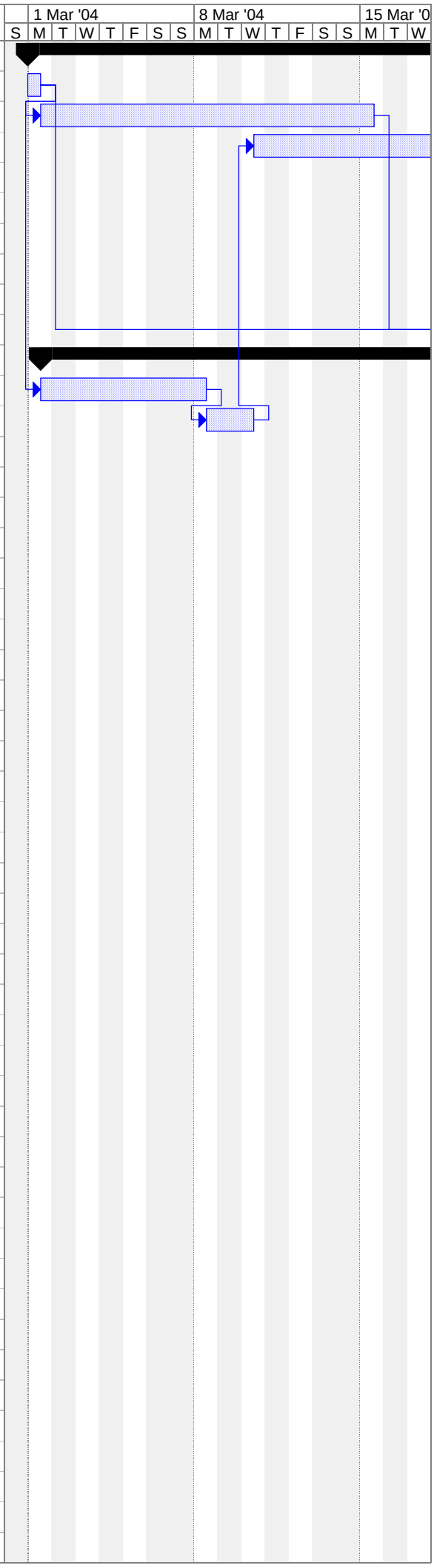
Web Application Model

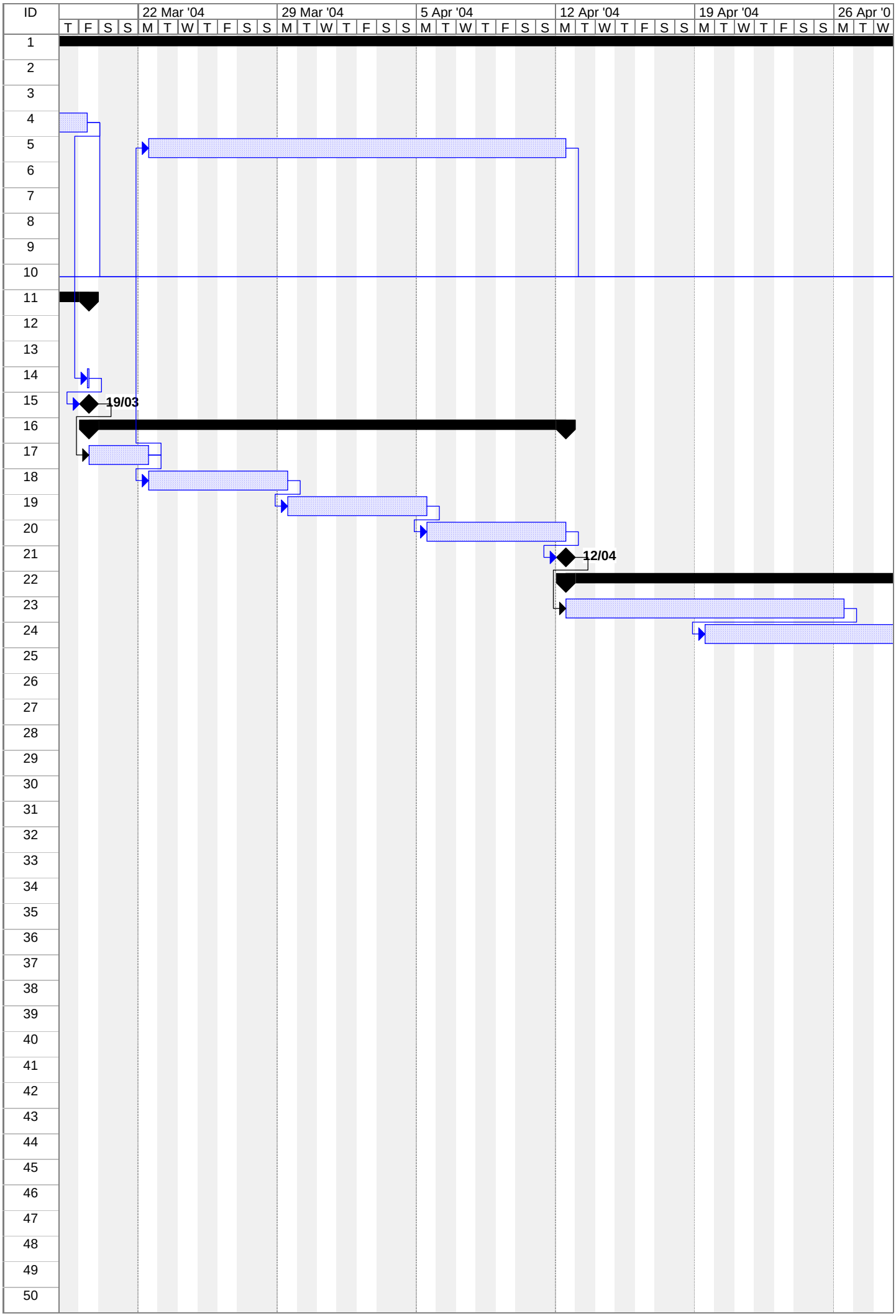

'OLE'
Presentation Tier
Realization Elements
All Packages are included in the <<server page>> located above it
All <<server page>>s include the <<server page>>s 'Footer', 'Menu', and 'Header'.
<<page>> Class
<<builds>>
<<server page>> Class.aspx
<<client page>> Class
<<page>> Management
<<server page>> Management.aspx
<<client page>> Management
<<server page>> Footer.ascx
<<page>> Home
<<server page>> Default.aspx
<<client page>> Default
<<page>> Submission
<<server page>> Submission.aspx
<<client page>> Submission
<<page>> Login
<<server page>> Login.aspx
<<client page>> Login
<<page>> Configuration
<<server page>> Configuration.aspx
<<client page>> Configuration
Class
<<server page>> ShortResponseAET.ascx
<<server page>> InformationAET.ascx
<<server page>> AETSubmissionList.ascx
<<server page>> MultipleChoiceAET.ascx
<<server page>> ThreadPosts.ascx
<<server page>> ClassAnnouncements.ascx
<<server page>> Threads.ascx
<<server page>> ShortResponseAETAnswer.ascx
Management
<<server page>> ClassAnnouncementManagement.ascx
<<server page>> ClassManagement.ascx
<<server page>> ShortResponseAETManagement.ascx
<<server page>> InformationAETManagement.ascx
<<server page>> InstitutionAnnouncementManagement.ascx
<<server page>> ClassNoteManagement.ascx
<<server page>> MultipleChoiceAETManagement.ascx
<<server page>> UserManagement.ascx
Home
<<server page>> ErrorMessages.ascx
<<server page>> QuickLinks.ascx
<<server page>> FormThreadSubscriptions.ascx
<<server page>> InstitutionAnnouncements.ascx
<<server page>> UpcomingDates.ascx
Submission
<<server page>> Class.ascx
<<server page>> User.ascx
Login
<<server page>> Authentication.ascx
<<server page>> Menu.ascx
<<server page>> Header.ascx
Configuration
<<server page>> RoleAssignmentConfiguration.ascx
<<server page>> ClassEnrollmentConfiguration.ascx
<<server page>> UserConfiguration.ascx
<<server page>> InstitutionConfiguration.ascx
<<server page>> ApplicationConfiguration.ascx
<<server page>> ClassConfiguration.ascx
Menu
<<server page>> Assignments.ascx
<<server page>> Classes.ascx
<<server page>> Tutorials.ascx
<<server page>> Notes.ascx
<<server page>> Exams.ascx
Realized by 'Class', 'Management', 'Home', and 'Menu'.
Realized by 'Class', and 'Menu'.
Realized by all packages.
Realized by 'Home'.
Realized by all packages.
Realized by 'Home'.
Realized by 'Class', and 'Management'.
Realized by 'Login'.
Business Logic Tier
Realization Elements
AETQuestion
AETAnswer
User
Role
ApplicationError
Class
InstitutionAnnouncement
ClassAnnouncement
ClassThreadPost
ClassNote
Authentication
SecurityError
AETInformationQuestion
AETShortResponseQuestion
AETMultipleChoiceQuestion
AETInformationAnswer
AETShortResponseAnswer
AETMultipleChoiceAnswer
Post
DataBridge
Data Access Tier
Specification Elements
Data from XML files and SQL Server database are retrieved by SQL Stored Procedures and ADO.NET

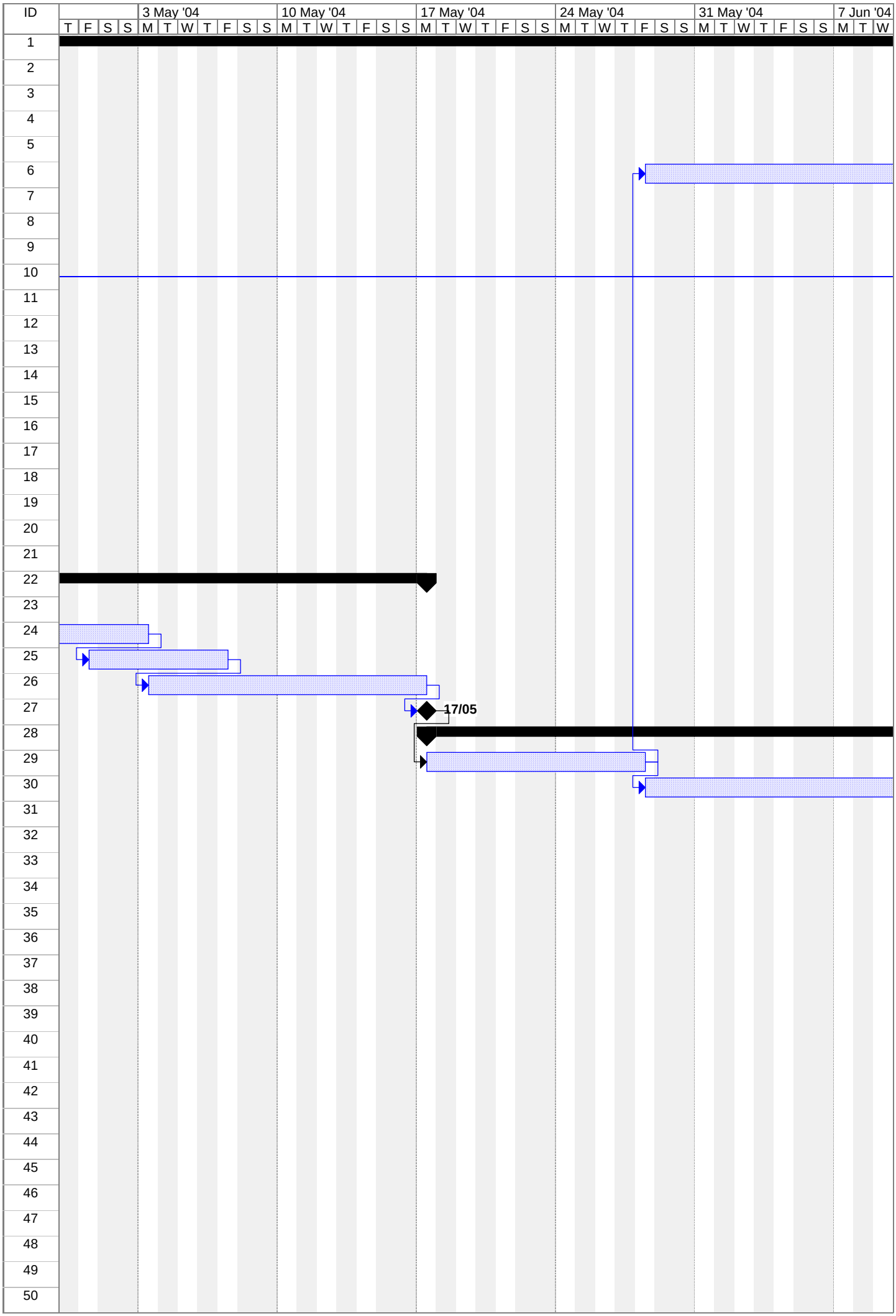
Appendix G

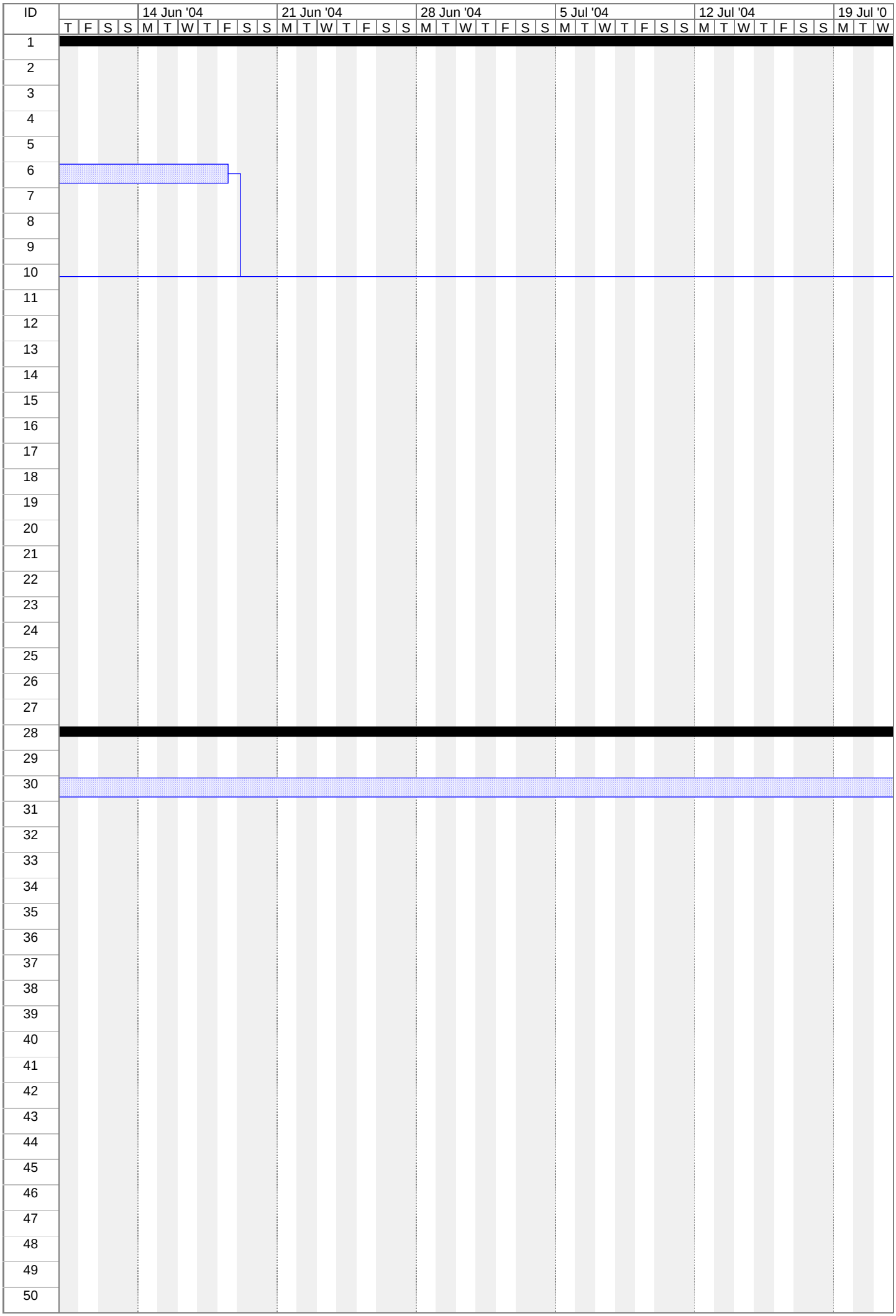
Project Gannt Chart

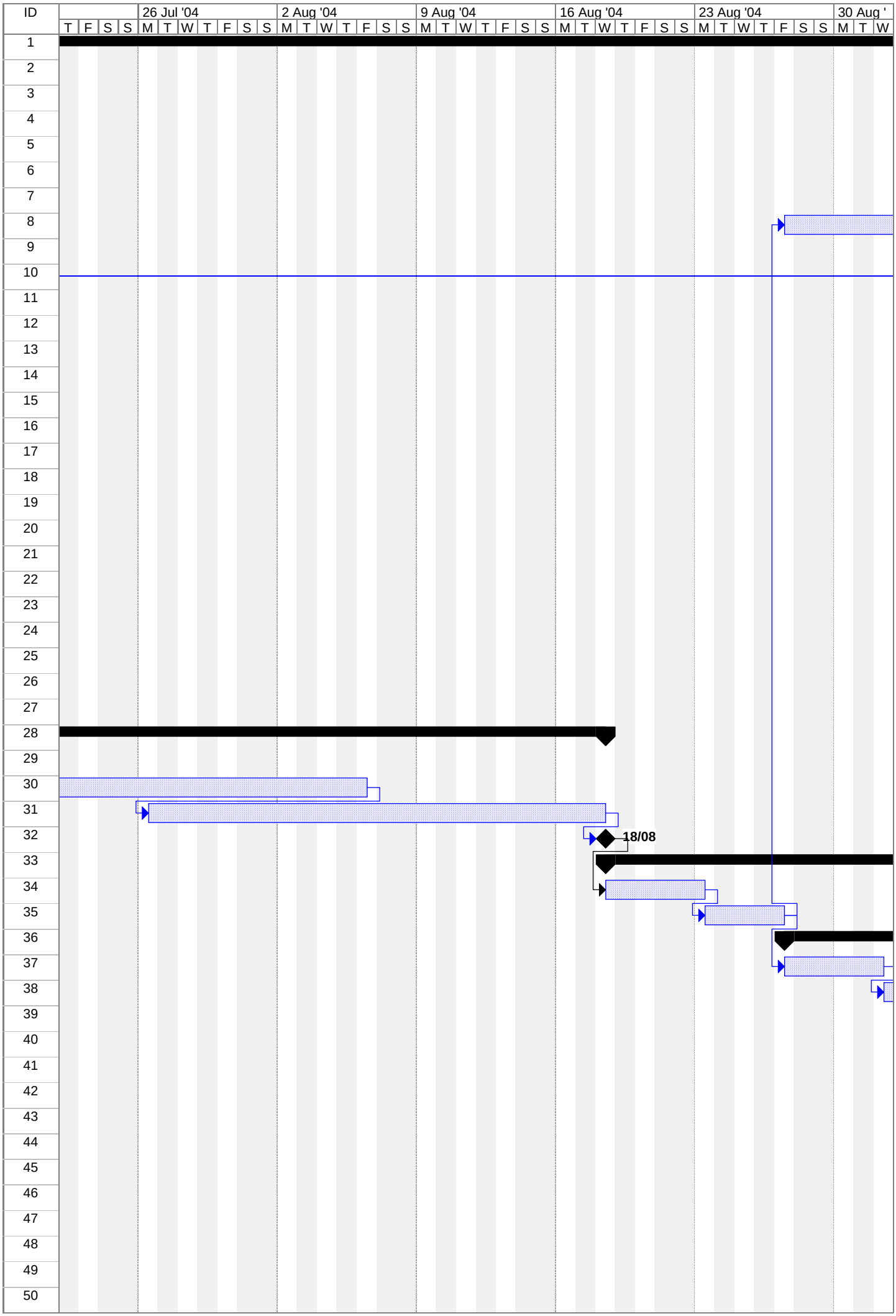
ID	Task Name	Duration	Start
1	**Documentation**	**96 days**	**Mon 1/03/04**
2	Project Specification	5 hrs	Mon 1/03/04
3	Project Appreciation	50 hrs	Mon 1/03/04
4	Requirements Analysis Document (RAD)	30 hrs	Wed 10/03/04
5	System Design Document (SDD)	9 days	Mon 22/03/04
6	Object Design Document (SDD)	9 days	Fri 28/05/04
7	Project Dissertation	5 wks	Mon 6/09/04
8	Test Manual	8 days	Fri 27/08/04
9	User Manual	10 hrs	Wed 15/09/04
10	Documentation Complete	0 days	Fri 8/10/04
11	**Requirements Elicitation and Analysis**	**8 days**	**Mon 1/03/04**
12	Needs analysis	3 days	Mon 1/03/04
13	Secure required resources	1 day	Mon 8/03/04
14	Obtain supervisor approval	2 hrs	Fri 19/03/04
15	Requirements Analysis Complete	0 days	Fri 19/03/04
16	**Design**	**10 days**	**Fri 19/03/04**
17	Review RAD	1 day	Fri 19/03/04
18	Design Presentation Tier	3 days	Mon 22/03/04
19	Design Business Logic Tier	3 days	Mon 29/03/04
20	Design Data Access Tier	3 days	Mon 5/04/04
21	Design Complete	0 days	Mon 12/04/04
22	**Training**	**15 days**	**Mon 12/04/04**
23	C# .NET	6 days	Mon 12/04/04
24	ASP.NET	6 days	Mon 19/04/04
25	SQL	3 days	Fri 30/04/04
26	XML	6 days	Mon 3/05/04
27	Training Complete	0 days	Mon 17/05/04
28	**Implementation**	**40 days**	**Mon 17/05/04**
29	Review RAD, SDD, and ODD	5 days	Mon 17/05/04
30	Implement code	30 days	Fri 28/05/04
31	Implementation testing (Debugging)	10 days	Mon 26/07/04
32	Implementation Complete	0 days	Wed 18/08/04
33	**Testing**	**12 days**	**Wed 18/08/04**
34	Develop unit test plans	2 days	Wed 18/08/04
35	Develop integration test plans	2 days	Mon 23/08/04
36	**Unit Testing**	**4 days**	**Fri 27/08/04**
37	Test component modules	2 days	Fri 27/08/04
38	Modify code	1 day	Wed 1/09/04
39	Re-test modified code	1 day	Fri 3/09/04
40	Unit Testing Complete	0 days	Mon 6/09/04
41	**Integration Testing**	**4 days**	**Mon 6/09/04**
42	Test module integration	2 days	Mon 6/09/04
43	Modify code	1 day	Fri 10/09/04
44	Re-test modified code	1 day	Mon 13/09/04
45	Integration Testing Complete	0 days	Wed 15/09/04
46	Testing Complete	0 days	Wed 15/09/04
47	**Deployment**	**2 days**	**Wed 15/09/04**
48	Deploy software	2 days	Wed 15/09/04
49	Deployment Complete	0 days	Mon 20/09/04
50	Project Complete	0 days	Fri 8/10/04

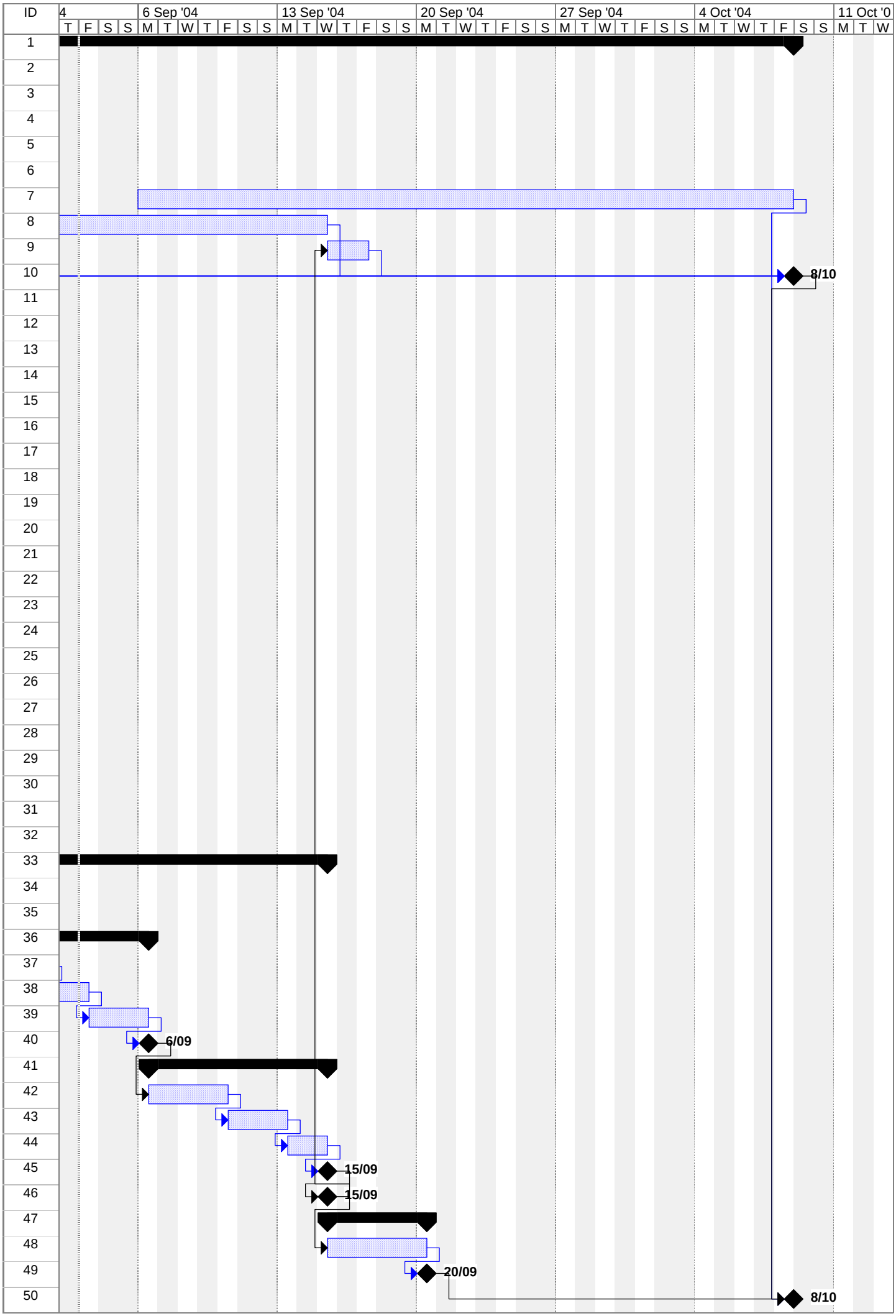



ID
22 Mar '04
29 Mar '04
5 Apr '04
12 Apr '04
19 Apr '04
26 Apr '0
T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W
19/03
12/04


ID
3 May '04
10 May '04
17 May '04
24 May '04
31 May '04
7 Jun '04
T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W
17/05


ID
14 Jun '04
21 Jun '04
28 Jun '04
5 Jul '04
12 Jul '04
19 Jul '0
T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50


ID
26 Jul '04
2 Aug '04
9 Aug '04
16 Aug '04
23 Aug '04
30 Aug '
T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W
18/08


ID
4
6 Sep '04
13 Sep '04
20 Sep '04
27 Sep '04
4 Oct '04
11 Oct '0
T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W
8/10
6/09
15/09
15/09
20/09
8/10

Appendix H

Source Code

H.1 OLEWebApplication Project

Listing H.1: Class.aspx

```
<%@ Register TagPrefix="OLE" TagName="Threads" Src="ClassControls/Threads.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ThreadPosts" Src="ClassControls/ThreadPosts.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ClassAnnoucements" Src="ClassControls/ClassAnnoucements.ascx" %>
<%@ Register TagPrefix="OLE" TagName="InformationAET" Src="ClassControls/InformationAET.ascx" %>
<%@ Register TagPrefix="OLE" TagName="MultipleChoiceAET" Src="ClassControls/MultipleChoiceAET.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ShortResponseAET" Src="ClassControls/ShortResponseAET.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Menu" Src="Menu.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Page language="c#" Codebehind="Class.aspx.cs" AutoEventWireup="false" Inherits="OnlineLearningEnvironment.Interface.ClassPage" codePage="28591" debug="False"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <TITLE>'Online Learning Environment'</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <META name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <META name="CODE_LANGUAGE" Content="C#">
        <META name="vs_defaultClientScript" content="JavaScript">
        <META name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
        <LINK href="Default.css" type="text/css" rel="stylesheet">
    </HEAD>
    <BODY MS_POSITIONING="FlowLayout">
```

```
        <form method="post" runat="server" ID="Form1">
            <table class="PageFramework" cellspacing="0" cellpadding
                ="0">
                <tr>
                    <td class="PageFramework_Header" colspan="2">
                        <OLE:Header id="OLEHeader" runat="server"></
                            OLE:Header>
                    </td>
                </tr>
                <tr>
                    <td class="PageFramework_Menu">
                        <OLE:Menu id="OLEMenu" runat="server"></OLE:
                            Menu>
                    </td>
                    <td class="PageFramework_Body">
                        <OLE:ClassAnnoucements id="
                            OLEClassAnnouncements" runat="server">></
                            OLE:ClassAnnoucements>
                        <OLE:InformationAET id="OLEInformationAET"
                            runat="server">></OLE:InformationAET>
                        <OLE:MultipleChoiceAET id="
                            OLEMultipleChoiceAET" runat="server">></
                            OLE:MultipleChoiceAET>
                        <OLE:ShortResponseAET id="
                            OLEShortResponseAET" runat="server">></OLE
                            :ShortResponseAET>
                        <OLE:ThreadPosts id="OLEThreadPosts" runat="
                            server">></OLE:ThreadPosts>
                        <OLE:Threads id="OLEThreads" runat="server">
                            </OLE:Threads>
                    </td>
                </tr>
                <tr>
                    <td class="PageFramework_Copyright" colspan="2">
                        <OLE:Footer id="OLEFooter" runat="server">></
                            OLE:Footer>
                    </td>
                </tr>
            </table>
        </form>
    </BODY>
</HTML>
```

Listing H.2: Class.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

```

```
12 namespace OnlineLearningEnvironment.Interface
13 {
14     public class ClassPage : System.Web.UI.Page
15     {
16         protected OnlineLearningEnvironment.Interface.HeaderControl
               OLEHeader;
17         protected OnlineLearningEnvironment.Interface.MenuControl
               OLEMenu;
18         protected OnlineLearningEnvironment.Interface.ClassControls.
               ClassAnnoucements OLEClassAnnouncements;
19         protected OnlineLearningEnvironment.Interface.ClassControls.
               Threads OLEThreads;
20         protected OnlineLearningEnvironment.Interface.ClassControls.
               ThreadPosts OLEThreadPosts;
21         protected OnlineLearningEnvironment.Interface.ClassControls.
               InformationAET OLEInformationAET;
22         protected OnlineLearningEnvironment.Interface.ClassControls.
               MultipleChoiceAET OLEMultipleChoiceAET;
23         protected OnlineLearningEnvironment.Interface.ClassControls.
               ShortResponseAET OLEShortResponseAET;
24
25         private string _classID = null;
26         public string ClassID
27         {
28             get { return this._classID; }
29             set { this._classID = value; }
30         }
31
32         private int _threadID = 0;
33         public int ThreadID
34         {
35             get { return this._threadID; }
36             set { this._threadID = value; }
37         }
38
39         private int _aetQID = 0;
40         public int AETQID
41         {
42             get { return this._aetQID; }
43             set { this._aetQID = value; }
44         }
45
46         private AETType _aetType = AETType.NotSet;
47         public AETType AETType
48         {
49             get { return this._aetType; }
50             set { this._aetType = value; }
51         }
52
53         private int _questionID = 1;
54         public int QuestionID
55         {
56             get { return this._questionID; }
57             set { this._questionID = value; }
58         }
```

```
59
60          private void Page_Load(object sender, System.EventArgs e)
61          {
62              this.OLEHeader.PageName = (Application["Configuration"]
                    as Configuration).Class;
63              this.OLEMenu.PageName = (Application["Configuration"] as
                    Configuration).Class;
64              if (Request.QueryString["classID"] != null)
65              {
66                  this.ClassID = Request.QueryString["classID"];
67
68                  if (Request.QueryString["threadID"] != null)
69                  {
70                      try
71                      {
72                          this.ThreadID = Int32.Parse(Request.
                                QueryString["threadID"]);
73                      }
74                      catch {}
75                  }
76                  if (Request.QueryString["aetQID"] != null)
77                  {
78                      try
79                      {
80                          this.AETQID = Int32.Parse(Request.
                                QueryString["aetQID"]);
81                      }
82                      catch {}
83                  }
84                  if (Request.QueryString["questionID"] != null)
85                  {
86                      try
87                      {
88                          this.QuestionID = Int32.Parse(Request.
                                QueryString["questionID"]);
89                      }
90                      catch {}
91                  }
92                  if (Request.QueryString["aetType"] != null)
93                  {
94                      switch (Request.QueryString["aetType"])
95                      {
96                          case "Assignment":
97                              this.AETType = AETType.Assignment;
98                              break;
99                          case "Exam":
100                             this.AETType = AETType.Exam;
101                             break;
102                         case "Tutorial":
103                             this.AETType = AETType.Tutorial;
104                             break;
105                     }
106                 }
107
108                 this.OLEHeader.SubPageName = this.ClassID;
```

```
109                 this.OLEMenu.ClassID = this.ClassID;
110
111                 switch (Request.QueryString["view"])
112                 {
113                     case "Announcements":
114                         this.OLEClassAnnouncements.Enabled = true;
115                         this.OLEClassAnnouncements.ClassID = this.
                                ClassID;
116                         break;
117                     case "Threads":
118                         this.OLEThreads.Enabled = true;
119                         this.OLEThreads.ClassID = this.ClassID;
120                         break;
121                     case "ThreadPosts":
122                         this.OLEThreadPosts.Enabled = true;
123                         this.OLEThreadPosts.ClassID = this.ClassID;
124                         this.OLEThreadPosts.ThreadID = this.ThreadID
                                ;
125                         break;
126                     case "InformationAET":
127                         this.OLEInformationAET.Enabled = true;
128                         this.OLEInformationAET.ClassID = this.
                                ClassID;
129                         this.OLEInformationAET.AETQID = this.AETQID;
130                         this.OLEInformationAET.AETType = this.
                                AETType;
131                         this.OLEInformationAET.QuestionID = this.
                                QuestionID;
132                         break;
133                     case "MultipleChoiceAET":
134                         this.OLEMultipleChoiceAET.Enabled = true;
135                         this.OLEMultipleChoiceAET.ClassID = this.
                                ClassID;
136                         this.OLEMultipleChoiceAET.AETQID = this.
                                AETQID;
137                         this.OLEMultipleChoiceAET.AETType = this.
                                AETType;
138                         this.OLEMultipleChoiceAET.QuestionID = this.
                                QuestionID;
139                         break;
140                     case "ShortResponseAET":
141                         this.OLEShortResponseAET.Enabled = true;
142                         this.OLEShortResponseAET.ClassID = this.
                                ClassID;
143                         this.OLEShortResponseAET.AETQID = this.
                                AETQID;
144                         this.OLEShortResponseAET.AETType = this.
                                AETType;
145                         this.OLEShortResponseAET.QuestionID = this.
                                QuestionID;
146                         break;
147                     case null:
148                     default:
149                         this.OLEClassAnnouncements.Enabled = true;
150                         this.OLEClassAnnouncements.ClassID = this.
```

```
                                ClassID;
                            break;
                        }
                    }
                }

                #region Web Form Designer generated code
                override protected void OnInit(EventArgs e)
                {
                    InitializeComponent();
                    base.OnInit(e);
                }

                private void InitializeComponent()
                {
                    this.Load += new System.EventHandler(this.Page_Load);

                }
                #endregion
            }
}
```

Listing H.3: Configuration.aspx

```
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Menu" Src="Menu.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Page language="c#" Codebehind="Configuration.aspx.cs"
    AutoEventWireup="false" Inherits="OnlineLearningEnvironment.
    Interface.ConfigurationPage" codePage="28591"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <TITLE>'Online Learning Environment'</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=
            iso-8859-1">
        <META name="GENERATOR" Content="Microsoft Visual Studio .NET
            7.1">
        <META name="CODE_LANGUAGE" Content="C#">
        <META name="vs_defaultClientScript" content="JavaScript">
        <META name="vs_targetSchema" content="http://schemas.
            microsoft.com/intellisense/ie5">
        <LINK href="Default.css" type="text/css" rel="stylesheet">
    </HEAD>
    <BODY MS_POSITIONING="FlowLayout">
        <form method="post" runat="server" ID="Form1">
            <table class="PageFramework" cellspacing="0" cellpadding
                ="0">
                <tr>
                    <td class="PageFramework_Header" colspan="2">
                        <OLE:Header id="OLEHeader" PageName="
                            Configuration" runat="server"></OLE:
                            Header>
                    </td>
                </tr>
                <tr>
```

```
                        <td class="PageFramework_Menu">
                            <OLE:Menu id="OLEMenu" PageName="
                                Configuration" runat="server"></OLE:Menu>
                        </td>
                        <td class="PageFramework_Body"></td>
                    </tr>
                    <tr>
                        <td class="PageFramework_Copyright" colspan="2">
                            <OLE:Footer id="OLEFooter" runat="server"></
                                OLE:Footer>
                        </td>
                    </tr>
                </table>
            </form>
        </BODY>
</HTML>
```

Listing H.4: Configuration.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    /// <summary>
    /// Not implemented
    /// </summary>
    public class ConfigurationPage : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.5: Default.aspx

```
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ErrorMessages" Src="
    HomeControls/ErrorMessages.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ForumThreadSubscriptions" Src=
    "HomeControls/ForumThreadSubscriptions.ascx" %>
<%@ Register TagPrefix="OLE" TagName="InstitutionAnnouncements" Src=
    "HomeControls/InstitutionAnnouncements.ascx" %>
<%@ Register TagPrefix="OLE" TagName="QuickLinks" Src="HomeControls/
    QuickLinks.ascx" %>
<%@ Register TagPrefix="OLE" TagName="UpcomingDates" Src="
    HomeControls/UpcomingDates.ascx" %>
<%@ Page language="c#" Codebehind="Default.aspx.cs" AutoEventWireup=
    "false" Inherits="OnlineLearningEnvironment.Interface.HomePage"
    codePage="28591"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <TITLE>'Online Learning Environment'</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=
            iso-8859-1">
        <META name="GENERATOR" Content="Microsoft Visual Studio .NET
            7.1">
        <META name="CODE LANGUAGE" Content="C#">
        <META name="vs_defaultClientScript" content="JavaScript">
        <META name="vs_targetSchema" content="http://schemas.
            microsoft.com/intellisense/ie5">
        <LINK href="Default.css" type="text/css" rel="stylesheet">
    </HEAD>
    <BODY MS_POSITIONING="FlowLayout">
        <form method="post" runat="server">
            <table class="PageFramework" cellspacing="0" cellpadding
                ="0">
                <tr>
                    <td class="PageFramework_Header">
                        <OLE:Header id="OLEHeader" PageName="Home"
                            runat="server"></OLE:Header>
                    </td>
                </tr>
                <tr>
                    <td class="PageFramework_Body" valign="top">
                        <table width="100%" cellspacing="0"
                            cellpadding="0">
                            <tr>
                                <td width="80%" valign="top">
                                    <table cellspacing="0"
                                        cellpadding="0" width="100%">
                                        <tr>
                                            <td align="center"><OLE:
                                                InstitutionAnnouncements
                                                 id="
                                                OLEInstitutionAnnouncements
                                                " Runat="server" /></
                                                td>
```

```
36                                          </tr>
37                                          <tr height="10">
38                                          </tr>
39                                          <tr>
40                                              <td align="center"><OLE:
                                                    ErrorMessages id="
                                                    OLEErrorMessages"
                                                    runat="server" /></td
                                                    >
41                                          </tr>
42                                          <tr height="10">
43                                          </tr>
44                                          <tr>
45                                              <td align="center"><OLE:
                                                    ForumThreadSubscriptions
                                                     id="
                                                    OLEForumThreadSubscriptions
                                                    " runat="server" /></
                                                    td>
46                                          </tr>
47                                      </table>
48                                  </td>
49                                  <td width="20%" valign="top">
50                                      <table cellspacing="0"
                                            cellpadding="0" width="100%">
51                                          <tr>
52                                              <td align="center"><OLE:
                                                    QuickLinks id="
                                                    OLEQuickLinks" Runat=
                                                    "server" /></td>
53                                          </tr>
54                                          <tr height="10">
55                                          </tr>
56                                          <tr>
57                                              <td align="center"><OLE:
                                                    UpcomingDates id="
                                                    OLEUpcomingDates"
                                                    runat="server" /></td
                                                    >
58                                          </tr>
59                                      </table>
60                                  </td>
61                              </tr>
62                          </table>
63                      </td>
64                  </tr>
65                  <tr>
66                      <td class="PageFramework_Copyright">
67                          <OLE:Footer id="OLEFooter" runat="server"></
                                OLE:Footer>
68                      </td>
69                  </tr>
70              </table>
71          </form>
72      </BODY>
```

```
</HTML>
```

Listing H.6: Default.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    public class HomePage : System.Web.UI.Page
    {
        protected OnlineLearningEnvironment.Interface.HomeControls.
            ErrorMessages OLEErrorMessages;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (Context.User.IsInRole(RoleID.Administrators.ToString
                ()))
                this.OLEErrorMessages.Enabled = true;
            else
                this.OLEErrorMessages.Enabled = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.7: Default.css

```
P
{
    font-size: 10pt;
    vertical-align: baseline;
    color: black;
    line-height: normal;
    font-style: normal;
```

```
8       font-family: Sans-Serif;
9       letter-spacing: normal;
10      text-align: left;
11  }
12  P.Copyright
13  {
14      font-size: 8pt;
15      vertical-align: baseline;
16      color: black;
17      line-height: normal;
18      font-style: normal;
19      font-family: Sans-Serif;
20      letter-spacing: normal;
21      text-align: center;
22  }
23
24  H1
25  {
26      font-size: 18pt;
27      vertical-align: baseline;
28      color: black;
29      line-height: normal;
30      font-style: italic;
31      font-family: Sans-Serif;
32      letter-spacing: normal;
33      text-align: left;
34      font-variant: small-caps;
35  }
36
37  H2
38  {
39      font-size: 16pt;
40      vertical-align: baseline;
41      color: black;
42      line-height: normal;
43      font-style: italic;
44      font-family: Sans-Serif;
45      letter-spacing: normal;
46      text-align: left;
47      font-variant: normal;
48  }
49
50  H3
51  {
52      font-size: 14pt;
53      vertical-align: baseline;
54      color: black;
55      line-height: normal;
56      font-style: italic;
57      font-family: Sans-Serif;
58      letter-spacing: normal;
59      text-align: left;
60      font-variant: normal;
61  }
62
```

```
63  BODY
64  {
65      margin: 0px;
66      background-color: #6699CC;
67  }
68
69  A
70  {
71      color: #990000;
72      text-decoration: none;
73  }
74  A:active
75  {
76      color: #990000;
77      text-decoration: none;
78  }
79  A:hover
80  {
81      color: #990000;
82      text-decoration: underline;
83  }
84  A:link
85  {
86      color: #990000;
87      text-decoration: none;
88  }
89
90  TABLE.PageFramework
91  {
92      padding: 0px;
93      margin: 0px;
94      border: none 0;
95      width: 100%;
96      height: 100%;
97  }
98  TD.PageFramework_Header
99  {
100     padding: 0px;
101     margin: 0px;
102     border: none 0;
103     height: 100px;
104 }
105 TD.PageFramework_Menu
106 {
107     padding: 0px;
108     margin: 0px;
109     border: none 0;
110     text-align: left;
111     vertical-align: top;
112 }
113 TD.PageFramework_Body
114 {
115     height: 100%;
116     width: 100%;
117     padding: 0px;
```

```
118     margin: 0px;
119     border: none 0;
120     vertical-align: top;
121 }
122 TD.PageFramework_LoginBody
123 {
124     height: 100%;
125     padding: 0px;
126     margin: 0px;
127     border: none 0;
128 }
129 TD.PageFramework_Copyright
130 {
131     padding: 0px;
132     margin: 0px;
133     border: none 0;
134     text-align: center;
135 }
136
137 A.HeaderFramework_Links
138 {
139     color: #FFFFFF;
140     text-decoration: none;
141 }
142 A:active.HeaderFramework_Links
143 {
144     color: #FFFFFF;
145     text-decoration: none;
146 }
147 A:hover.HeaderFramework_Links
148 {
149     color: #ffffff;
150     text-decoration: underline;
151 }
152 A:link.HeaderFramework_Links
153 {
154     color: #FFFFFF;
155     text-decoration: none;
156 }
157 TABLE.HeaderFramework
158 {
159     padding: 0px;
160     margin: 0px;
161     border: none 0;
162     width: 100%;
163     background-color: #0066CC;
164 }
165 TD.HeaderFramework_TopLeft
166 {
167     height: 100px;
168 }
169 TD.HeaderFramework_TopRightLeft
170 {
171     width: 25px;
172     height: 25px;
```

```
173     background-position: center 50%;
174     background-image: url(Images/topleft.jpg);
175 }
176 TD.HeaderFramework_TopRightMiddle
177 {
178     background-position: center 50%;
179     font-size: 10pt;
180     background-image: url(Images/topright.jpg);
181     color: #FFFFFF;
182     background-repeat: repeat-x;
183     font-family: Arial;
184     height: 25px;
185     text-align: right;
186 }
187 TD.HeaderFramework_TopRightRight
188 {
189     width: 5px;
190     background-position: center 50%;
191     background-image: url(Images/topright.jpg);
192     background-repeat: repeat-x;
193 }
194 TD.HeaderFramework_TopRightBottom
195 {
196     font-size: 10pt;
197     font-family: Arial;
198     color: #FFFFFF;
199     text-align: right;
200 }
201 TD.HeaderFramework_Bottom
202 {
203     background-position: center 50%;
204     font-size: 12pt;
205     background-image: url(Images/border.jpg);
206     background-repeat: repeat-x;
207     font-family: Arial;
208     height: 35px;
209     text-align: center;
210     font-variant: small-caps;
211 }
212
213 TABLE.LoginFramework
214 {
215     background-color: #FFFFFF;
216     width: 250px;
217     padding: 0px;
218     margin: 0px;
219     border: none 0;
220 }
221 TD.LoginFramework_Username
222 {
223     font-size: 12pt;
224     font-family: Arial;
225     text-align: left;
226 }
227 TD.LoginFramework_Password
```

```
228 {
229     font-size: 12pt;
230     font-family: Arial;
231     text-align: left;
232 }
233 TD.LoginFramework_RememberMe
234 {
235     font-size: 12pt;
236     font-family: Arial;
237     text-align: right;
238 }
239 TD.LoginFramework_Login
240 {
241     text-align: center;
242 }
243 TD.LoginFramework_Validators
244 {
245     font-size: 10pt;
246     font-family: Arial;
247     text-align: center;
248 }
249
250 TABLE.PostFramework
251 {
252     font-size: 12pt;
253     font-family: Arial;
254     width: 98%;
255     border: none 0;
256     border-bottom: #0066cc 3px solid;
257 }
258 TH.PostFramework_Header
259 {
260     font-size: 13pt;
261     color: #DDDDDD;
262     border-bottom: #6699cc 3px solid;
263     font-variant: small-caps;
264     border-top: 1px solid #6699cc;
265     vertical-align: middle;
266     height: 30px;
267     background-position: center 50%;
268     background-image: url(Images/header2.jpg);
269     background-repeat: repeat-x;
270 }
271 TD.PostFramework_PostData
272 {
273     vertical-align: top;
274     width: 20%;
275     text-align: center;
276     background-color: #DDDDDD;
277     border-bottom: solid 2px #AAAAAA;
278 }
279 TD.PostFramework_PostInformation
280 {
281     vertical-align: top;
282     width: 80%;
```

```
    background-color: #EEEEEE;
    border-bottom: solid 2px #AAAAAA;
}
TD.PostFramework_AlternatePostInformation
{
    vertical-align: top;
    width: 80%;
    background-color: #e6e6e6;
    border-bottom: solid 2px #AAAAAA;
}
DIV.PostFramework_Name
{
    font-size: 11pt;
    font-weight: bold;
    font-style: italic;
}
DIV.PostFramework_DateDescription
{
    font-size: 9pt;
    font-weight: bold;
}
DIV.PostFramework_Date
{
    font-size: 9pt;
}
DIV.PostFramework_Files
{
    font-size: 9pt;
    font-weight: bold;
}
DIV.PostFramework_Filenames
{
    font-size: 9pt;
}

TABLE.ErrorFramework
{
    font-size: 12pt;
    font-family: Arial;
    width: 98%;
    border: none 0;
    border-bottom: #0066cc 3px solid;
}
TH.ErrorFramework_Header
{
    font-size: 13pt;
    color: #DDDDDD;
    border-bottom: #6699cc 3px solid;
    font-variant: small-caps;
    border-top: 1px solid #6699cc;
    vertical-align: middle;
    height: 30px;
    background-position: center 50%;
    background-image: url(Images/header2.jpg);
    background-repeat: repeat-x;
```

```
338 }
339 TH.ErrorFramework_Columns
340 {
341     text-align: center;
342     background-color: #AAAAAA;
343     border-bottom: solid 3px #888888;
344 }
345 TD.ErrorFramework_Date
346 {
347     width: 20%;
348     text-align: center;
349     background-color: #DDDDDD;
350     border-bottom: solid 2px #AAAAAA;
351 }
352 TD.ErrorFramework_Data
353 {
354     width: 40%;
355     vertical-align: top;
356     text-align: center;
357     background-color: #EEEEEE;
358     border-bottom: solid 2px #AAAAAA;
359 }
360 TD.ErrorFramework_Information
361 {
362     width: 80%;
363     vertical-align: top;
364     text-align: left;
365     background-color: #EEEEEE;
366     border-bottom: solid 2px #AAAAAA;
367 }
368 TD.ErrorFramework_AlternateData
369 {
370     width: 40%;
371     vertical-align: top;
372     text-align: center;
373     background-color: #e6e6e6;
374     border-bottom: solid 2px #AAAAAA;
375 }
376 TD.ErrorFramework_AlternateInformation
377 {
378     width: 80%;
379     vertical-align: top;
380     text-align: left;
381     background-color: #e6e6e6;
382     border-bottom: solid 2px #AAAAAA;
383 }
384
385 TABLE.QLFramework
386 {
387     font-size: 12pt;
388     font-family: Arial;
389     width: 98%;
390     border: none 0;
391     border-bottom: #0066cc 3px solid;
392 }
```

```
TH.QLFramework_Header
{
    font-size: 13pt;
    color: #DDDDDD;
    border-bottom: #6699cc 3px solid;
    font-variant: small-caps;
    border-top: 1px solid #6699cc;
    vertical-align: middle;
    height: 30px;
    background-position: center 50%;
    background-image: url(Images/header2.jpg);
    background-repeat: repeat-x;
}
TD.QLFramework_Class
{
    vertical-align: top;
    text-align: center;
    background-color: #EEEEEE;
    border-bottom: solid 2px #AAAAAA;
}
TD.QLFramework_AlternateClass
{
    vertical-align: top;
    text-align: center;
    background-color: #e6e6e6;
    border-bottom: solid 2px #AAAAAA;
}

TABLE.AEFramework
{
    font-size: 12pt;
    font-family: Arial;
    width: 98%;
    border: none 0;
    border-bottom: #0066cc 3px solid;
}
TH.AEFramework_Header
{
    font-size: 13pt;
    color: #DDDDDD;
    border-bottom: #6699cc 3px solid;
    font-variant: small-caps;
    border-top: 1px solid #6699cc;
    vertical-align: middle;
    height: 30px;
    background-position: center 50%;
    background-image: url(Images/header2.jpg);
    background-repeat: repeat-x;
}
TD.AEFramework_Date
{
    font-size: 10pt;
    text-align: center;
    vertical-align: middle;
    background-color: #DDDDDD;
```

```
448     border-bottom: solid 2px #AAAAAA;
449 }
450 TD.AEFramework_Description
451 {
452     width: 100%;
453     vertical-align: top;
454     text-align: center;
455     background-color: #EEEEEE;
456     border-bottom: solid 2px #AAAAAA;
457 }
458 TD.AEFramework_AlternateDescription
459 {
460     width: 100%;
461     vertical-align: top;
462     text-align: center;
463     background-color: #e6e6e6;
464     border-bottom: solid 2px #AAAAAA;
465 }
466 DIV.AEFramework_ClassID
467 {
468     font-size: 9pt;
469 }
470
471 TABLE.IAETFramework
472 {
473     font-size: 12pt;
474     font-family: Arial;
475     width: 98%;
476     border: none 0;
477     background-color: #EEEEEE;
478     border-top: #0066cc 3px solid;
479     border-bottom: #0066cc 3px solid;
480 }
481 DIV.IAETFramework_Title
482 {
483     padding: 10px 0px 0px 0px;
484     font-size: 16pt;
485     font-variant: small-caps;
486 }
487 DIV.IAETFramework_Description
488 {
489     font-size: 8pt;
490 }
491 TD.IAETFramework_Data
492 {
493     font-size: 8pt;
494     text-align: center;
495 }
496 DIV.IAETFramework_QuestionTitle
497 {
498     font-weight: bold;
499     font-size: 12pt;
500     font-style: italic;
501 }
502 DIV.IAETFramework_Question
```

```
{
    font-size: 10pt;
}
```

Listing H.8: Footer.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Footer
    .ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    FooterControl" TargetSchema="http://schemas.microsoft.com/
    intellisense/ie5"%>
<font color="#000000" size="1" face="Arial">Copyright 2004 by
    Timothy Vriesema</font>
```

Listing H.9: Footer.ascx.cs

```
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class FooterControl : System.Web.UI.UserControl
    {

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.10: Global.asax

```
<%@ Application Codebehind="Global.asax.cs" Inherits="
    OnlineLearningEnvironment.Interface.Global" %>
```

Listing H.11: Global.asax.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
```

```
5  using System.Data.SqlClient;
6  using System.IO;
7  using System.Security.Principal;
8  using System.Web;
9  using System.Web.Security;
10 using System.Web.SessionState;
11 using System.Xml;
12
13 namespace OnlineLearningEnvironment.Interface
14 {
15     public class Global : System.Web.HttpApplication
16     {
17         private System.ComponentModel.IContainer components = null;
18
19         public Global()
20         {
21             InitializeComponent();
22         }
23
24         protected void Application_Start(Object sender, EventArgs e)
25         {
26             Microsoft.Win32.RegistryKey rkOLE = Microsoft.Win32.
                   Registry.LocalMachine.OpenSubKey(
27                 @"SOFTWARE\Online_Learning_Environment"
28                 );
29
30             Application["ClassesDirectory"] = (string)rkOLE.GetValue
                   ("DataDirectory") + "\\" + "Classes";
31             Application["InstitutionDirectory"] = (string)rkOLE.
                   GetValue("DataDirectory") + "\\" + "Institution";
32
33             Application["SqlConnectionString"] = "server=" + rkOLE.
                   GetValue("SqlServerName") + ";Trusted_Connection=true
                   ;database=oledb;";
34
35             DataBridge db = new DataBridge(
36                 new SqlConnection(Application["SqlConnectionString"]
                       as string),
37                 Application["ClassesDirectory"] as string,
38                 Application["InstitutionDirectory"] as string,
39                 false
40                 );
41
42             if (!Directory.Exists((string)rkOLE.GetValue("
                   DataDirectory")))
43             {
44                     // Create Class directory
45                 Directory.CreateDirectory(Application["
                       ClassesDirectory"] as string);
46
47                     // Create Institution directories
48                 Directory.CreateDirectory(Application["
                       InstitutionDirectory"] as string);
49                 Directory.CreateDirectory((Application["
                       InstitutionDirectory"] as string) + "\\" + "
```

```
                    Announcements");
50                  Directory.CreateDirectory((Application["
                        InstitutionDirectory"] as string) + "\\" + "
                        ApplicationErrors");
51                  Directory.CreateDirectory((Application["
                        InstitutionDirectory"] as string) + "\\" + "
                        SecurityErrors");
52
53                  Configuration config = new Configuration(db);
54                  config.Create();
55              }
56
57                  // Configuration for the Application
58              Application["Configuration"] = OnlineLearningEnvironment
                    .Configuration.FindConfiguration(
59                  db
60                  );
61          }
62
63          protected void Session_Start(Object sender, EventArgs e)
64          {
65              Session["DataBridge"] = new DataBridge(
66                  new SqlConnection(Application["SqlConnectionString"]
                        as string),
67                  Application["ClassesDirectory"] as string,
68                  Application["InstitutionDirectory"] as string,
69                  true
70                  );
71
72              Session["SessionVariables"] = new ArrayList();
73              (Session["SessionVariables"] as ArrayList).Add(Session["
                    DataBridge"]);
74          }
75
76          protected void Application_BeginRequest(Object sender,
                EventArgs e)
77          {
78
79          }
80
81          protected void Application_EndRequest(Object sender,
                EventArgs e)
82          {
83
84          }
85
86          protected void Application_AuthenticateRequest(Object sender
                , EventArgs e)
87          {
88              string cookieName = FormsAuthentication.FormsCookieName;
89              HttpCookie httpCookie = Context.Request.Cookies[
                    cookieName];
90
91              if (httpCookie == null)
92              {
```

```
93                  return;
94              }
95              FormsAuthenticationTicket aTicket = null;
96              try
97              {
98                  aTicket = FormsAuthentication.Decrypt(httpCookie.
                        Value);
99              }
100             catch
101             {
102                 return;
103             }
104
105             if (aTicket == null)
106             {
107                 return;
108             }
109
110             FormsIdentity formsIdentity = new FormsIdentity(aTicket)
                    ;
111
112             User user = new User(
113                 Int32.Parse(formsIdentity.Name),
114                 formsIdentity,
115                 new DataBridge(
116                     new SqlConnection(Application["
                            SqlConnectionString"] as string),
117                     Application["ClassesDirectory"] as string,
118                     Application["InstitutionDirectory"] as string,
119                     true)
120                 );
121             Context.User = user;
122
123             OnlineLearningEnvironment.User tempUser =
                    OnlineLearningEnvironment.User.FindUser(
124                 Int32.Parse(Context.User.Identity.Name),
125                 (Context.User as User).DataBridge
126                 );
127             (Context.User as User).FirstName = tempUser.FirstName;
128             (Context.User as User).MiddleName = tempUser.MiddleName;
129             (Context.User as User).LastName = tempUser.LastName;
130             (Context.User as User).RoleIDs = Role.FindRolesIDs(
131                 Int32.Parse(Context.User.Identity.Name),
132                 (Context.User as User).DataBridge
133                 );
134         }
135
136         protected void Application_Error(Object sender, EventArgs e)
137         {
138
139         }
140
141         protected void Session_End(Object sender, EventArgs e)
142         {
143
```

```
144         }
145
146         protected void Application_End(Object sender, EventArgs e)
147         {
148
149         }
150
151         #region Web Form Designer generated code
152         private void InitializeComponent()
153         {
154             this.components = new System.ComponentModel.Container();
155         }
156         #endregion
157     }
158 }
```

Listing H.12: Header.ascx

```
1  <%@ Control Language="c#" AutoEventWireup="false" Codebehind="Header
       .ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
       HeaderControl" TargetSchema="http://schemas.microsoft.com/
       intellisense/ie5"%>
2  <TABLE class="HeaderFramework" cellSpacing="0" cellPadding="0" width
       ="100%">
3      <TR>
4          <TD class="HeaderFramework_TopLeft"><asp:Image id="imageLogo
               " runat="server" Height="100px" Width="420px" /></TD>
5          <TD align="right" valign="top">
6              <asp:Table cellSpacing="0" cellPadding="0" id="
                   tableLinks" Runat="server">
7                  <asp:TableRow Runat="server">
8                      <asp:TableCell CssClass="
                           HeaderFramework_TopRightLeft"></asp:TableCell
                           >
9                      <asp:TableCell CssClass="
                           HeaderFramework_TopRightMiddle">
10                         <asp:HyperLink id="hyperHome" CssClass="
                               HeaderFramework_Links" NavigateUrl="
                               Default.aspx" runat="server">Home</asp:
                               HyperLink>
11                         <font>| </font>
12                         <asp:HyperLink id="hyperManagement" CssClass
                               ="HeaderFramework_Links" NavigateUrl="
                               Management.aspx"
13                                runat="server">Management</asp:HyperLink
                                   >
14                         <font>| </font>
15                         <asp:HyperLink id="hyperConfiguration"
                               CssClass="HeaderFramework_Links"
                               NavigateUrl="Configuration.aspx"
16                                runat="server">Configuration</asp:
                                   HyperLink>
17                         <font>| </font>
18                         <asp:HyperLink id="hyperClasses" CssClass="
                               HeaderFramework_Links" NavigateUrl="Class
                               .aspx" runat="server">Classes</asp:
```

```
                                HyperLink>
                            <font>| </font>
                            <asp:HyperLink id="hyperSubmission" CssClass="HeaderFramework_Links" NavigateUrl="Submission.aspx"
                                runat="server">Submission</asp:HyperLink>
                            <font>| </font>
                            <asp:LinkButton id="buttonLogout" CssClass="HeaderFramework_Links" runat="server">Logout</asp:LinkButton>
                        </asp:TableCell>
                        <asp:TableCell CssClass="HeaderFramework_TopRightRight"></asp:TableCell>
                    </asp:TableRow>
                    <asp:TableRow>
                        <asp:TableCell></asp:TableCell>
                        <asp:TableCell CssClass="HeaderFramework_TopRightBottom">
                            <asp:Label id="labelWelcome" runat="server" />
                        </asp:TableCell>
                        <asp:TableCell width="5"></asp:TableCell>
                    </asp:TableRow>
                </asp:Table>
            </TD>
        </TR>
        <TR>
            <TD class="HeaderFramework_Bottom" colSpan="2"><asp:Label id="labelPageName" runat="server"></asp:Label></TD>
        </TR>
</TABLE>
```

Listing H.13: Header.ascx.cs

```
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Collections;
    using System.Web;
    using System.Web.Security;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class HeaderControl : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Label labelPageName;
        protected System.Web.UI.WebControls.Label labelWelcome;
        protected System.Web.UI.WebControls.HyperLink hyperHome;
        protected System.Web.UI.WebControls.HyperLink hyperConfiguration;
        protected System.Web.UI.WebControls.HyperLink hyperClasses;
        protected System.Web.UI.WebControls.LinkButton buttonLogout;
```

```
20          protected System.Web.UI.WebControls.Image imageLogo;
21          protected System.Web.UI.WebControls.HyperLink
                hyperSubmission;
22          protected System.Web.UI.WebControls.Table tableLinks;
23
24          private string _pageName;
25          public string PageName
26          {
27              get { return this._pageName; }
28              set { this._pageName = value; }
29          }
30
31          private string _subPageName = null;
32          public string SubPageName
33          {
34              get { return this._subPageName; }
35              set { this._subPageName = value; }
36          }
37
38          private void Page_Load(object sender, System.EventArgs e)
39          {
40              this.labelPageName.Text = this.PageName;
41              if (this.SubPageName != null)
42                  this.labelPageName.Text += " - " + this.SubPageName;
43
44              this.imageLogo.ImageUrl = (Application["Configuration"]
                    as Configuration).InstitutionLogoImage;
45
46              if (this.PageName == "Login")
47                  this.tableLinks.Visible = false;
48              else
49                  this.labelWelcome.Text = "Welcome " + "<i>" + (
                        Context.User as User).FullName + "</i>" + " (" +
                        (Context.User as User).UserID + ")";
50          }
51
52          #region Web Form Designer generated code
53          override protected void OnInit(EventArgs e)
54          {
55              InitializeComponent();
56              base.OnInit(e);
57          }
58
59          private void InitializeComponent()
60          {
61              this.Load += new System.EventHandler(this.Page_Load);
62
63          }
64          #endregion
65
66          private void buttonLogout_Click(object sender, System.
                EventArgs e)
67          {
68                  // Kill session variables
69              if (Session["SessionVariables"] != null)
```

```
70                  {
71                      object[] os = (object[])(Session["SessionVariables"
                            ] as ArrayList).ToArray(typeof(object));
72                      for (int i=0; i<os.Length; i++)
73                      {
74                          os[i] = null;
75                      }
76                  }
77
78                  FormsAuthentication.SignOut();
79                  Response.Redirect("Login.aspx");
80              }
81          }
82  }
```

Listing H.14: Login.aspx

```
1  <%@ Page language="c#" Codebehind="Login.aspx.cs" AutoEventWireup="
       false" Inherits="OnlineLearningEnvironment.Interface.LoginPage"
       codePage="28591"%>
2  <%@ Register TagPrefix="OLE" TagName="Authentication" Src="
       LoginControls/Authentication.ascx" %>
3  <%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
4  <%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
6  <HTML>
7      <HEAD>
8          <TITLE>'Online Learning Environment'</TITLE>
9          <meta content="SharePoint.WebPartPage.Document" name="ProgId
               ">
10         <meta content="full" name="WebPartPageExpansion">
11         <META http-equiv="Content-Type" content="text/html; charset=
               iso-8859-1">
12         <META content="Microsoft Visual Studio .NET 7.1" name="
               GENERATOR">
13         <META content="C#" name="CODE_LANGUAGE">
14         <META content="JavaScript" name="vs_defaultClientScript">
15         <META content="http://schemas.microsoft.com/intellisense/ie5
               " name="vs_targetSchema">
16         <LINK href="Default.css" type="text/css" rel="stylesheet">
17     </HEAD>
18     <BODY MS_POSITIONING="FlowLayout">
19         <form id="Form1" method="post" runat="server">
20             <table class="PageFramework" cellSpacing="0" cellPadding
                   ="0">
21                 <tr>
22                     <td class="PageFramework_Header"><OLE:Header id=
                           "OLEHeader" runat="server" PageName="Login"><
                           /OLE:Header></td>
23                 </tr>
24                 <tr>
25                     <td class="PageFramework_LoginBody" align="
                           center"><OLE:Authentication id="OLELogin"
                           runat="server"></OLE:Authentication></td>
26                 <tr>
27                     <td class="PageFramework_Copyright"><OLE:Footer
```

```
                                id="OLEFooter" runat="server"></OLE:Footer></
                                td>
                        </tr>
                    </table>
                </form>
            </BODY>
</HTML>
```

Listing H.15: Login.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    public class LoginPage : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.16: Management.aspx

```
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Menu" Src="Menu.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Page language="c#" Codebehind="Management.aspx.cs"
    AutoEventWireup="false" Inherits="OnlineLearningEnvironment.
    Interface.ManagementPage" codePage="28591"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
```

```
        <TITLE>'Online Learning Environment'</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <META name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <META name="CODE_LANGUAGE" Content="C#">
        <META name="vs_defaultClientScript" content="JavaScript">
        <META name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
        <LINK href="Default.css" type="text/css" rel="stylesheet">
    </HEAD>
    <BODY MS_POSITIONING="FlowLayout">
        <form method="post" runat="server" ID="Form1">
            <table class="PageFramework" cellspacing="0" cellpadding="0">
                <tr>
                    <td class="PageFramework_Header" colspan="2">
                        <OLE:Header id="OLEHeader" PageName="Management" runat="server"></OLE:Header>
                    </td>
                </tr>
                <tr>
                    <td class="PageFramework_Menu">
                        <OLE:Menu id="OLEMenu" PageName="Management" runat="server"></OLE:Menu>
                    </td>
                    <td class="PageFramework_Body"></td>
                </tr>
                <tr>
                    <td class="PageFramework_Copyright" colspan="2">
                        <OLE:Footer id="OLEFooter" runat="server">
                        OLE:Footer>
                    </td>
                </tr>
            </table>
        </form>
    </BODY>
</HTML>
```

Listing H.17: Management.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    /// <summary>
    /// Not implemented
```

```
    /// </summary>
    public class ManagementPage : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.18: Menu.ascx

```
<%@ Register TagPrefix="OLE" TagName="Tutorials" Src="MenuControls/Tutorials.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Notes" Src="MenuControls/Notes.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Exams" Src="MenuControls/Exams.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Assignments" Src="MenuControls/Assignments.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Classes" Src="MenuControls/Classes.ascx" %>
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Menu.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.MenuControl" TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
<asp:Table Height="1px" Width="150px" CellPadding="0" CellSpacing="0" Runat="server">
</asp:Table>
<asp:Table ID="tableClass" Visible="False" Runat="server">
    <asp:TableRow>
        <asp:TableCell><asp:HyperLink ID="hyperAnnouncements" Runat="server">Announcements</asp:HyperLink></asp:TableCell>
    </asp:TableRow>
    <asp:TableRow>
        <asp:TableCell><asp:HyperLink ID="hyperForum" Runat="server">Forum</asp:HyperLink></asp:TableCell>
    </asp:TableRow>
</asp:Table>
<OLE:Notes id="OLENotes" Visible="False" runat="server"></OLE:Notes>
<OLE:Assignments id="OLEAssignments" Visible="False" runat="server">
    </OLE:Assignments>
<OLE:Exams id="OLEExams" Visible="False" runat="server"></OLE:Exams>
```

```
<OLE:Tutorials id="OLETutorials" Visible="False" runat="server"></OLE:Tutorials>
<OLE:Classes id="OLEClasses" Visible="False" runat="server"></OLE:Classes>
```

Listing H.19: Menu.ascx.cs

```
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class MenuControl : System.Web.UI.UserControl
    {
        protected OnlineLearningEnvironment.Interface.MenuControls.Notes OLENotes;
        protected OnlineLearningEnvironment.Interface.MenuControls.Assignments OLEAssignments;
        protected OnlineLearningEnvironment.Interface.MenuControls.Exams OLEExams;
        protected OnlineLearningEnvironment.Interface.MenuControls.Tutorials OLETutorials;
        protected OnlineLearningEnvironment.Interface.MenuControls.Classes OLEClasses;

        protected System.Web.UI.WebControls.Table tableClass;
        protected System.Web.UI.WebControls.HyperLink hyperAnnouncements;
        protected System.Web.UI.WebControls.HyperLink hyperForum;

        private string _pageName;
        public string PageName
        {
            get { return this._pageName; }
            set { this._pageName = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.PageName == (Application["Configuration"] as Configuration).Class)
            {
                if (this.ClassID != null)
                {
                    this.hyperAnnouncements.NavigateUrl = "Class.
```

```
                        aspx?" + "view=" + "Announcements" + "&
                        classID=" + this.ClassID;
                    this.hyperForum.NavigateUrl = "Class.aspx?" + "
                        view=" + "Threads" + "&classID=" + this.
                        ClassID;
                    this.tableClass.Visible = true;
                    this.OLENotes.Enabled = true;
                    this.OLENotes.ClassID = this.ClassID;
                    this.OLEAssignments.Enabled = true;
                    this.OLEAssignments.ClassID = this.ClassID;
                    this.OLEExams.Enabled = true;
                    this.OLEExams.ClassID = this.ClassID;
                    this.OLETutorials.Enabled = true;
                    this.OLETutorials.ClassID = this.ClassID;
                }
                else
                {
                    this.OLEClasses.Enabled = true;
                }
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.20: Submission.aspx

```
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Menu" Src="Menu.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Page language="c#" Codebehind="Submission.aspx.cs"
    AutoEventWireup="false" Inherits="OnlineLearningEnvironment.
    Interface.SubmissionPage" codePage="28591"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <TITLE>'Online Learning Environment'</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=
            iso-8859-1">
        <META name="GENERATOR" Content="Microsoft Visual Studio .NET
            7.1">
        <META name="CODE_LANGUAGE" Content="C#">
        <META name="vs_defaultClientScript" content="JavaScript">
```

```
13          <META name="vs_targetSchema" content="http://schemas.
                microsoft.com/intellisense/ie5">
14          <LINK href="Default.css" type="text/css" rel="stylesheet">
15      </HEAD>
16      <BODY MS_POSITIONING="FlowLayout">
17          <form method="post" runat="server" ID="Form1">
18              <table class="PageFramework" cellspacing="0" cellpadding
                    ="0">
19                  <tr>
20                      <td class="PageFramework_Header" colspan="2">
21                          <OLE:Header id="OLEHeader" PageName="
                                Submission" runat="server"></OLE:Header>
22                      </td>
23                  </tr>
24                  <tr>
25                      <td class="PageFramework_Menu">
26                          <OLE:Menu id="OLEMenu" PageName="Submission"
                                runat="server"></OLE:Menu>
27                      </td>
28                      <td class="PageFramework_Body"></td>
29                  </tr>
30                  <tr>
31                      <td class="PageFramework_Copyright" colspan="2">
32                          <OLE:Footer id="OLEFooter" runat="server"></
                                OLE:Footer>
33                      </td>
34                  </tr>
35              </table>
36          </form>
37      </BODY>
38  </HTML>
```

Listing H.21: Submission.aspx.cs

```
1   using System;
2   using System.Collections;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Web;
7   using System.Web.SessionState;
8   using System.Web.UI;
9   using System.Web.UI.WebControls;
10  using System.Web.UI.HtmlControls;
11
12  namespace OnlineLearningEnvironment.Interface
13  {
14      /// <summary>
15      /// Not implemented
16      /// </summary>
17      public class SubmissionPage : System.Web.UI.Page
18      {
19          private void Page_Load(object sender, System.EventArgs e)
20          {
21          }
22
```

```
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.22: Web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.web>
    <compilation defaultLanguage="c#" debug="true" />
    <customErrors mode="RemoteOnly" />
    <authentication mode="Forms">
      <forms name="OLEAuthentication" loginUrl="Login.aspx"
          protection="All" path="/" />
    </authentication>
    <authorization>
        <deny users="?" />
    </authorization>
    <trace enabled="false" requestLimit="10" pageOutput="false"
        traceMode="SortByTime" localOnly="true" />
    <sessionState
            mode="InProc"
            stateConnectionString="tcpip=127.0.0.1:42424"
            sqlConnectionString="data source=127.0.0.1;
                Trusted_Connection=yes"
            cookieless="false"
            timeout="20"
    />
    <globalization
            requestEncoding="utf-8"
            responseEncoding="utf-8"
   />

 </system.web>

</configuration>
```

Listing H.23: ClassControls/AETSubmissionList.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    AETSubmissionList.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ClassControls.AETSubmissionList" TargetSchema="http://
    schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.24: ClassControls/AETSubmissionList.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class AETSubmissionList : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.25: ClassControls/ClassAnnoucements.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ClassAnnoucements.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ClassControls.ClassAnnoucements" TargetSchema="http://
    schemas.microsoft.com/intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table class="PostFramework" cellspacing="0" cellpadding="0"
            >
            <tr height="20px">
                <td></td>
            </tr>
            <tr>
                <th class="PostFramework_Header" colspan="2">
                    Announcements</th>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td class="PostFramework_PostData">
                <div class="PostFramework_Name"><%#((
                    OnlineLearningEnvironment.ClassAnnouncement)
```

```
                    Container.DataItem).CreatorName%></div>
17                  <div class="PostFramework_DateDescription">Created</
                    div>
18                  <div class="PostFramework_Date"><%#((
                    OnlineLearningEnvironment.ClassAnnouncement)
                    Container.DataItem).CreatedDateTime.ToString()%><
                    /div>
19                  <div class="PostFramework_DateDescription">Updated</
                    div>
20                  <div class="PostFramework_Date"><%#((
                    OnlineLearningEnvironment.ClassAnnouncement)
                    Container.DataItem).UpdatedDateTime.ToString()%><
                    /div>
21              </td>
22              <td class="PostFramework_PostInformation">
23                  <div><%#((OnlineLearningEnvironment.
                    ClassAnnouncement)Container.DataItem).Information
                    %></span></div>
24                  <asp:Repeater ID="fileAttachmentsRepeater1"
                    DataSource="<%#((OnlineLearningEnvironment.
                    ClassAnnouncement)Container.DataItem).
                    AttachmentFilenames%>" Runat="server">
25                      <HeaderTemplate>
26                          <blockquote>
27                              <table width="100%">
28                                  <tr>
29                                      <td colspan="2">
30                                          <div class="
                                            PostFramework_Files">File
                                             Attachments:</div>
31                                      </td>
32                                  </tr>
33                                  <tr>
34                                      <td width="15px"></td>
35                                      <td>
36                                          <div class="
                                            PostFramework_Filenames">
37                      </HeaderTemplate>
38                      <ItemTemplate>
39                          <%#Container.DataItem%>
40                      </ItemTemplate>
41                      <SeparatorTemplate>
42                           
43                      </SeparatorTemplate>
44                      <FooterTemplate>
45                                          </div>
46                                      </td>
47                                  </tr>
48                              </table>
49                          </blockquote>
50                      </FooterTemplate>
51                  </asp:Repeater>
52              </td>
53          </tr>
54      </ItemTemplate>
```

```
55      <AlternatingItemTemplate>
56          <tr>
57              <td class="PostFramework_PostData">
58                  <div class="PostFramework_Name"><%#((
                        OnlineLearningEnvironment.ClassAnnouncement)
                        Container.DataItem).CreatorName%></div>
59                  <div class="PostFramework_DateDescription">Created</
                        div>
60                  <div class="PostFramework_Date"><%#((
                        OnlineLearningEnvironment.ClassAnnouncement)
                        Container.DataItem).CreatedDateTime.ToString()%><
                        /div>
61                  <div class="PostFramework_DateDescription">Updated</
                        div>
62                  <div class="PostFramework_Date"><%#((
                        OnlineLearningEnvironment.ClassAnnouncement)
                        Container.DataItem).UpdatedDateTime.ToString()%><
                        /div>
63              </td>
64              <td class="PostFramework_AlternatePostInformation">
65                  <div><%#((OnlineLearningEnvironment.
                        ClassAnnouncement)Container.DataItem).Information
                        %></span></div>
66                  <asp:Repeater ID="fileAttachmentsRepeater2"
                        DataSource="<%#((OnlineLearningEnvironment.
                        ClassAnnouncement)Container.DataItem).
                        AttachmentFilenames%>" Runat="server">
67                      <HeaderTemplate>
68                          <blockquote>
69                              <table width="100%">
70                                  <tr>
71                                      <td colspan="2">
72                                          <div class="
                                                PostFramework_Files">File
                                                 Attachments:</div>
73                                      </td>
74                                  </tr>
75                                  <tr>
76                                      <td width="15px"></td>
77                                      <td>
78                                          <div class="
                                                PostFramework_Filenames">
79                      </HeaderTemplate>
80                      <ItemTemplate>
81                          <%#Container.DataItem%>
82                      </ItemTemplate>
83                      <SeparatorTemplate>
84                           
85                      </SeparatorTemplate>
86                      <FooterTemplate>
87                                          </div>
88                                      </td>
89                                  </tr>
90                              </table>
91                          </blockquote>
```

```
                        </FooterTemplate>
                    </asp:Repeater>
                </td>
            </tr>
        </AlternatingItemTemplate>
        <FooterTemplate>
            </table>
        </FooterTemplate>
</asp:Repeater>
```

Listing H.26: ClassControls/ClassAnnoucements.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassAnnoucements : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    ClassAnnouncement[] announcements =
                        ClassAnnouncement.FindClassAnnouncements(this
                        .ClassID, Session["DataBridge"] as DataBridge
                        );
                    if (announcements != null)
                    {
```

```
                            Array.Reverse(announcements);

                            this.repeater.DataSource = announcements;
                            this.repeater.DataBind();
                        }
                    }
                }
                else
                    this.Visible = false;
            }

            #region Web Form Designer generated code
            override protected void OnInit(EventArgs e)
            {
                InitializeComponent();
                base.OnInit(e);
            }

            private void InitializeComponent()
            {
                this.Load += new System.EventHandler(this.Page_Load);

            }
            #endregion
        }
}
```

Listing H.27: ClassControls/InformationAET.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    InformationAET.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ClassControls.InformationAET" TargetSchema="http://
    schemas.microsoft.com/intellisense/ie5"%>
<table cellpadding="0" cellspacing="0">
    <tr height="20px"><td></td></tr>
</table>
<table cellpadding="0" cellspacing="0" class="IAETFramework">
    <tr>
        <th>
            <div class="IAETFramework_Title"><asp:Label ID="
                labelTitle" Runat="server"></asp:Label></div>
            <div class="IAETFramework_Description"><asp:Label ID="
                labelDescription" Runat="server"></asp:Label></div>
        </th>
    </tr>
    <tr>
        <td class="IAETFramework_Data">
            <div>Due Date: <asp:Label ID="labelDueDate" Runat="
                server" /></div>
        </td>
    </tr>
    <tr>
        <td height="15px"></td>
    </tr>
    <asp:Repeater ID="repeaterQuestions" Runat="server">
        <ItemTemplate>
```

```
            <tr>
                <td>
                    <div class="IAETFramework_QuestionTitle"><%#((OnlineLearningEnvironment.InformationQuestion)Container.DataItem).questionID%>. <%#((OnlineLearningEnvironment.InformationQuestion)Container.DataItem).title%></div>
                    <blockquote>
                        <div class="IAETFramework_Question"><%#((OnlineLearningEnvironment.InformationQuestion)Container.DataItem).question%></div>
                    </blockquote>
                </td>
            </tr>
        </ItemTemplate>
        <SeparatorTemplate>
            <tr>
                <td height="10px"></td>
            </tr>
        </SeparatorTemplate>
    </asp:Repeater>
</table>
```

Listing H.28: ClassControls/InformationAET.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class InformationAET : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Label labelTitle;
        protected System.Web.UI.WebControls.Label labelDescription;
        protected System.Web.UI.WebControls.Label labelQuestion;
        protected System.Web.UI.WebControls.Label labelDueDate;
        protected System.Web.UI.WebControls.Repeater repeaterQuestions;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
```

```
30          }
31
32          private int _aetQID = 0;
33          public int AETQID
34          {
35              get { return this._aetQID; }
36              set { this._aetQID = value; }
37          }
38
39          private AETType _aetType = AETType.NotSet;
40          public AETType AETType
41          {
42              get { return this._aetType; }
43              set { this._aetType = value; }
44          }
45
46          private int _questionID = 1;
47          public int QuestionID
48          {
49              get { return this._questionID; }
50              set { this._questionID = value; }
51          }
52
53          private AETInformationQuestion currentAETIQ = null;
54
55          private void Page_Load(object sender, System.EventArgs e)
56          {
57              if (this.Enabled)
58              {
59                  this.Visible = true;
60
61                  if (this.AETQID != 0 && this.AETType != AETType.
                        NotSet && this.ClassID != null)
62                  {
63                      if (Request.QueryString["questionID"] != null)
64                      {
65                          try
66                          {
67                              this.QuestionID = Int32.Parse(Request.
                                    QueryString["questionID"]);
68                          }
69                          catch {}
70                      }
71
72                      string uniqueQID = this.ClassID + "|" + this.
                            AETQID.ToString() + "|" + this.AETType.
                            ToString() + "|" + "Question";
73                      if (Session[uniqueQID] == null)
74                      {
75                          Session[uniqueQID] = this.currentAETIQ =
                                AETInformationQuestion.
                                FindAETInformationQuestion(this.ClassID,
                                this.AETQID, this.AETType, Session["
                                DataBridge"] as DataBridge) as
                                AETInformationQuestion;
```

```
                        this.currentAETIQ.DataBridge = Session["DataBridge"] as DataBridge;
                    }
                    else
                        this.currentAETIQ = Session[uniqueQID] as AETInformationQuestion;

                    this.labelTitle.Text = this.currentAETIQ.Title;
                    this.labelDescription.Text = this.currentAETIQ.Description;
                    this.labelDueDate.Text = this.currentAETIQ.DueDateTime.Date.ToLongDateString();

                    this.repeaterQuestions.DataSource = this.currentAETIQ.Questions;
                    this.repeaterQuestions.DataBind();
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.29: ClassControls/MultipleChoiceAET.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="MultipleChoiceAET.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.ClassControls.MultipleChoiceAET" TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
<table cellpadding="0" cellspacing="0">
    <tr>
        <td><asp:Label ID="labelTitle" Runat="server"></asp:Label></td>
    </tr>
    <tr>
        <td><asp:Label ID="labelDescription" Runat="server"></asp:Label></td>
    </tr>
    <tr>
        <td>
            <asp:Repeater ID="repeaterQuestions1" Runat="server">
                <ItemTemplate>
```

```
13                          <a href="Class.aspx?view=MultipleChoiceAET&
                                classID=<%#Request.QueryString["classID"]%>&
                                aetQID=<%#Request.QueryString["aetQID"]%>&
                                aetType=<%#Request.QueryString["aetType"]%>&
                                questionID=<%#Container.DataItem%>"><%#
                                Container.DataItem%></a>
14                      </ItemTemplate>
15                      <SeparatorTemplate>
16                           
17                      </SeparatorTemplate>
18                  </asp:Repeater>
19              </td>
20          </tr>
21          <tr>
22              <td>
23                  <asp:Label ID="labelQuestion" Runat="server"></asp:Label
                        >
24                  <blockquote>
25                      <asp:RadioButtonList id="rblChoices" RepeatLayout="
                            Table" Runat="server"></asp:RadioButtonList>
26                  </blockquote>
27              </td>
28          </tr>
29          <tr>
30              <td>
31                  <asp:Button ID="buttonAnswer" Text="Answer" Runat="
                        server"></asp:Button>
32              </td>
33          </tr>
34          <tr>
35              <td>
36                  <asp:Repeater ID="repeaterQuestions2" Runat="server">
37                      <ItemTemplate>
38                          <a href="Class.aspx?view=MultipleChoiceAET&
                                classID=<%#Request.QueryString["classID"]%>&
                                aetQID=<%#Request.QueryString["aetQID"]%>&
                                aetType=<%#Request.QueryString["aetType"]%>&
                                questionID=<%#Container.DataItem%>"><%#
                                Container.DataItem%></a>
39                      </ItemTemplate>
40                      <SeparatorTemplate>
41                           
42                      </SeparatorTemplate>
43                  </asp:Repeater>
44              </td>
45          </tr>
46      </table>
```

Listing H.30: ClassControls/MultipleChoiceAET.ascx.cs

```
1  namespace OnlineLearningEnvironment.Interface.ClassControls
2  {
3      using System;
4      using System.Data;
5      using System.Drawing;
6      using System.Web;
```

```
7       using System.Web.UI.WebControls;
8       using System.Web.UI.HtmlControls;
9
10      public class MultipleChoiceAET : System.Web.UI.UserControl
11      {
12          protected System.Web.UI.WebControls.Repeater
                repeaterQuestions1;
13          protected System.Web.UI.WebControls.Repeater
                repeaterQuestions2;
14          protected System.Web.UI.WebControls.RadioButtonList
                rblChoices;
15          protected System.Web.UI.WebControls.Button buttonAnswer;
16          protected System.Web.UI.WebControls.Label labelTitle;
17          protected System.Web.UI.WebControls.Label labelDescription;
18          protected System.Web.UI.WebControls.Label labelQuestion;
19
20          private bool _enabled = false;
21          public bool Enabled
22          {
23              get { return this._enabled; }
24              set { this._enabled = value; }
25          }
26
27          private string _classID = null;
28          public string ClassID
29          {
30              get { return this._classID; }
31              set { this._classID = value; }
32          }
33
34          private int _aetQID = 0;
35          public int AETQID
36          {
37              get { return this._aetQID; }
38              set { this._aetQID = value; }
39          }
40
41          private AETType _aetType = AETType.NotSet;
42          public AETType AETType
43          {
44              get { return this._aetType; }
45              set { this._aetType = value; }
46          }
47
48          private int _questionID = 1;
49          public int QuestionID
50          {
51              get { return this._questionID; }
52              set { this._questionID = value; }
53          }
54
55          private AETMultipleChoiceQuestion currentAETMCQ = null;
56          private AETMultipleChoiceAnswer currentAETMCA = null;
57
58          private void Page_Load(object sender, System.EventArgs e)
```

```
59          {
60              if (this.Enabled)
61              {
62                  this.Visible = true;
63                  if (!this.IsPostBack)
64                  {
65                      if (this.AETQID != 0 && this.AETType != AETType.
                            NotSet && this.ClassID != null)
66                      {
67                          if (Request.QueryString["questionID"] !=
                                null)
68                          {
69                              try
70                              {
71                                  this.QuestionID = Int32.Parse(
                                        Request.QueryString["questionID"
                                        ]);
72                              }
73                              catch {}
74                          }
75
76                          string uniqueQID = this.ClassID + "|" + this
                                .AETQID.ToString() + "|" + this.AETType.
                                ToString() + "|" + "Question";
77                          if (Session[uniqueQID] == null)
78                          {
79                              Session[uniqueQID] = this.currentAETMCQ
                                    = AETMultipleChoiceQuestion.
                                    FindAETMultipleChoiceQuestion(this.
                                    ClassID, this.AETQID, this.AETType,
                                    Session["DataBridge"] as DataBridge)
                                    as AETMultipleChoiceQuestion;
80                              this.currentAETMCQ.DataBridge = Session[
                                    "DataBridge"] as DataBridge;
81                          }
82                          else
83                          {
84                              this.currentAETMCQ = Session[uniqueQID]
                                    as AETMultipleChoiceQuestion;
85                          }
86
87                          string uniqueAID = this.ClassID + "|" + this
                                .AETQID.ToString() + "|" + this.AETType.
                                ToString() + "|" + "Answer";
88                          if (Session[uniqueAID] == null)
89                          {
90                              Session[uniqueAID] = this.currentAETMCA
                                    = AETMultipleChoiceAnswer.
                                    FindAETMultipleChoiceAnswer(this.
                                    AETQID, this.ClassID, Int32.Parse(
                                    Context.User.Identity.Name), this.
                                    AETType, Session["DataBridge"] as
                                    DataBridge) as
                                    AETMultipleChoiceAnswer;
91                              if (Session[uniqueAID] == null)
```

```
92                                  {
93                                      Session[uniqueAID] = this.
                                            currentAETMCA = new
                                            AETMultipleChoiceAnswer(Session["
                                            DataBridge"] as DataBridge);
94                                      this.currentAETMCA.AETQID = this.
                                            AETQID;
95                                      this.currentAETMCA.AETType = this.
                                            AETType;
96                                      this.currentAETMCA.ClassID = this.
                                            ClassID;
97                                      this.currentAETMCA.UserID = Int32.
                                            Parse(Context.User.Identity.Name)
                                            ;
98                                      this.currentAETMCA.answers = new
                                            MultipleChoiceAnswer[this.
                                            currentAETMCQ.AmountOfQuestions];
99                                      this.currentAETMCA.Marked = false;
100                                     this.currentAETMCA.SubmittedDateTime
                                            = DateTime.Now;
101                                     this.currentAETMCA.Create();
102                                 }
103                                 else
104                                 {
105                                     this.currentAETMCA.DataBridge =
                                            Session["DataBridge"] as
                                            DataBridge;
106                                 }
107                             }
108                             else
109                             {
110                                 this.currentAETMCA = Session[uniqueAID]
                                        as AETMultipleChoiceAnswer;
111                             }
112
113                             if (this.currentAETMCQ != null)
114                             {
115                                 int[] questionIDs = new int[this.
                                        currentAETMCQ.AmountOfQuestions];
116                                 for (int i=0; i<questionIDs.Length; i++)
117                                     questionIDs[i] = i + 1;
118
119                                 this.repeaterQuestions1.DataSource =
                                        this.repeaterQuestions2.DataSource =
                                        questionIDs;
120                                 this.repeaterQuestions1.DataBind();
121                                 this.repeaterQuestions2.DataBind();
122
123                                 this.labelTitle.Text = this.
                                        currentAETMCQ.Title;
124                                 this.labelDescription.Text = this.
                                        currentAETMCQ.Description;
125
126                                 this.labelQuestion.Text = this.
                                        currentAETMCQ.Questions[this.
```

```
                                QuestionID - 1].question;

                            foreach (MultipleChoiceChoice mcc in this.currentAETMCQ.Questions[this.QuestionID - 1].choices)
                            {
                                this.rblChoices.Items.Add(new ListItem(mcc.choiceText, mcc.choiceID.ToString()));
                            }

                            int answer;
                            if ((answer = this.currentAETMCA.answers[this.QuestionID - 1].choiceID) != 0)
                                this.rblChoices.Items[answer - 1].Selected = true;
                        }
                    }
                }
            }
            else
                this.Visible = false;
        }

        private void ButtonAnswer_Click(object sender, System.EventArgs e)
        {
            string uniqueQID = this.ClassID + "|" + this.AETQID.ToString() + "|" + this.AETType.ToString() + "|" + "Question";
            string uniqueAID = this.ClassID + "|" + this.AETQID.ToString() + "|" + this.AETType.ToString() + "|" + "Answer";
            if (Session[uniqueAID] != null)
            {
                this.currentAETMCQ = Session[uniqueQID] as AETMultipleChoiceQuestion;
                this.currentAETMCA = Session[uniqueAID] as AETMultipleChoiceAnswer;

                this.currentAETMCA.answers[this.QuestionID - 1].questionID = this.QuestionID;
                this.currentAETMCA.answers[this.QuestionID - 1].choiceID = Int32.Parse(this.rblChoices.SelectedValue);
                bool answeredCorrectly = (this.currentAETMCA.answers[this.QuestionID - 1].choiceID == this.currentAETMCQ.Questions[this.QuestionID - 1].correctChoice);
                this.currentAETMCA.answers[this.QuestionID - 1].answeredCorrectly = answeredCorrectly;

                this.currentAETMCA.SubmittedDateTime = DateTime.Now;

                this.currentAETMCA.Alter();
```

```
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
            this.buttonAnswer.Click += new System.EventHandler(this.
                ButtonAnswer_Click);
        }
        #endregion
    }
}
```

Listing H.31: ClassControls/ShortResponseAET.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ShortResponseAET.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ClassControls.ShortResponseAET" TargetSchema="http://
    schemas.microsoft.com/intellisense/ie5"%>
<table cellpadding="0" cellspacing="0">
    <tr>
        <td><asp:Label ID="labelTitle" Runat="server"></asp:Label></
            td>
    </tr>
    <tr>
        <td><asp:Label ID="labelDescription" Runat="server"></asp:
            Label></td>
    </tr>
    <tr>
        <td>
            <asp:Repeater ID="repeaterQuestions1" Runat="server">
                <ItemTemplate>
                    <a href="Class.aspx?view=ShortResponseAET&
                        classID=<%#Request.QueryString["classID"]%>&
                        aetQID=<%#Request.QueryString["aetQID"]%>&
                        aetType=<%#Request.QueryString["aetType"]%>&
                        questionID=<%#Container.DataItem%>"><%#
                        Container.DataItem%></a>
                </ItemTemplate>
                <SeparatorTemplate>
                     
                </SeparatorTemplate>
            </asp:Repeater>
        </td>
    </tr>
    <tr>
        <td><asp:Label ID="labelQuestion" Runat="server"></asp:Label
            ></td>
    </tr>
```

```
    <tr>
        <td><asp:TextBox ID="textboxAnswer" TabIndex="1" Runat="
            server" MaxLength="1000"></asp:TextBox></td>
    </tr>
    <tr>
        <td><asp:LinkButton ID="buttonAnswer" TabIndex="2" Text="
            Answer" Runat="server"></asp:LinkButton></td>
    </tr>
    <tr>
        <td>
            <asp:Repeater ID="repeaterQuestions2" Runat="server">
                <ItemTemplate>
                    <a href="Class.aspx?view=ShortResponseAET&
                        classID=<%#Request.QueryString["classID"]%>&
                        aetQID=<%#Request.QueryString["aetQID"]%>&
                        aetType=<%#Request.QueryString["aetType"]%>&
                        questionID=<%#Container.DataItem%>"><%#
                        Container.DataItem%></a>
                </ItemTemplate>
                <SeparatorTemplate>
                     
                </SeparatorTemplate>
            </asp:Repeater>
        </td>
    </tr>
</table>
```

Listing H.32: ClassControls/ShortResponseAET.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class ShortResponseAET : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater
            repeaterQuestions1;
        protected System.Web.UI.WebControls.Repeater
            repeaterQuestions2;
        protected System.Web.UI.WebControls.Label labelTitle;
        protected System.Web.UI.WebControls.Label labelDescription;
        protected System.Web.UI.WebControls.Label labelQuestion;
        protected System.Web.UI.WebControls.TextBox textboxAnswer;
        protected System.Web.UI.WebControls.LinkButton buttonAnswer;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }
```

```
26
27          private string _classID = null;
28          public string ClassID
29          {
30              get { return this._classID; }
31              set { this._classID = value; }
32          }
33
34          private int _aetQID = 0;
35          public int AETQID
36          {
37              get { return this._aetQID; }
38              set { this._aetQID = value; }
39          }
40
41          private AETType _aetType = AETType.NotSet;
42          public AETType AETType
43          {
44              get { return this._aetType; }
45              set { this._aetType = value; }
46          }
47
48          private int _questionID = 1;
49          public int QuestionID
50          {
51              get { return this._questionID; }
52              set { this._questionID = value; }
53          }
54
55          private AETShortResponseQuestion currentAETSRQ = null;
56          protected System.Web.UI.WebControls.RequiredFieldValidator
                RequiredFieldValidator1;
57          private AETShortResponseAnswer currentAETSRA = null;
58
59          private void Page_Load(object sender, System.EventArgs e)
60          {
61              if (this.Enabled)
62              {
63                  this.Visible = true;
64
65                  if (!this.IsPostBack)
66                  {
67                      if (this.AETQID != 0 && this.AETType != AETType.
                            NotSet && this.ClassID != null)
68                      {
69                          if (Request.QueryString["questionID"] !=
                                null)
70                          {
71                              try
72                              {
73                                  this.QuestionID = Int32.Parse(
                                        Request.QueryString["questionID"
                                        ]);
74                              }
75                              catch {}
```

```
76                          }
77
78                          string uniqueQID = this.ClassID + "|" + this
                                .AETQID.ToString() + "|" + this.AETType.
                                ToString() + "|" + "Question";
79                          if (Session[uniqueQID] == null)
80                          {
81                              Session[uniqueQID] = this.currentAETSRQ
                                    = AETShortResponseQuestion.
                                    FindAETShortResponseQuestion(this.
                                    ClassID, this.AETQID, this.AETType,
                                    Session["DataBridge"] as DataBridge)
                                    as AETShortResponseQuestion;
82                              this.currentAETSRQ.DataBridge = Session[
                                    "DataBridge"] as DataBridge;
83                          }
84                          else
85                          {
86                              this.currentAETSRQ = Session[uniqueQID]
                                    as AETShortResponseQuestion;
87                          }
88
89                          string uniqueAID = this.ClassID + "|" + this
                                .AETQID.ToString() + "|" + this.AETType.
                                ToString() + "|" + "Answer";
90                          if (Session[uniqueAID] == null)
91                          {
92                              Session[uniqueAID] = this.currentAETSRA
                                    = AETShortResponseAnswer.
                                    FindAETShortResponseAnswer(this.
                                    AETQID, this.ClassID, Int32.Parse(
                                    Context.User.Identity.Name), this.
                                    AETType, Session["DataBridge"] as
                                    DataBridge) as AETShortResponseAnswer
                                    ;
93                              if (Session[uniqueAID] == null)
94                              {
95                                  Session[uniqueAID] = this.
                                        currentAETSRA = new
                                        AETShortResponseAnswer(Session["
                                        DataBridge"] as DataBridge);
96                                  this.currentAETSRA.AETQID = this.
                                        AETQID;
97                                  this.currentAETSRA.AETType = this.
                                        AETType;
98                                  this.currentAETSRA.ClassID = this.
                                        ClassID;
99                                  this.currentAETSRA.UserID = Int32.
                                        Parse(Context.User.Identity.Name)
                                        ;
100                                 this.currentAETSRA.answers = new
                                        ShortResponseAnswer[this.
                                        currentAETSRQ.AmountOfQuestions];
101                                 this.currentAETSRA.Marked = false;
102                                 this.currentAETSRA.SubmittedDateTime
```

```
                                = DateTime.Now;
103                                 this.currentAETSRA.Create();
104                             }
105                             else
106                             {
107                                 this.currentAETSRA.DataBridge =
                                        Session["DataBridge"] as
                                        DataBridge;
108                             }
109                         }
110                         else
111                         {
112                             this.currentAETSRA = Session[uniqueAID]
                                    as AETShortResponseAnswer;
113                         }
114
115                         if (this.currentAETSRQ != null)
116                         {
117                             int[] questionIDs = new int[this.
                                    currentAETSRQ.AmountOfQuestions];
118                             for (int i=0; i<questionIDs.Length; i++)
119                                 questionIDs[i] = i + 1;
120
121                             this.repeaterQuestions1.DataSource =
                                    this.repeaterQuestions2.DataSource =
                                    questionIDs;
122                             this.repeaterQuestions1.DataBind();
123                             this.repeaterQuestions2.DataBind();
124
125                             this.labelTitle.Text = this.
                                    currentAETSRQ.Title;
126                             this.labelDescription.Text = this.
                                    currentAETSRQ.Description;
127
128                             this.labelQuestion.Text = this.
                                    currentAETSRQ.Questions[this.
                                    QuestionID - 1].question;
129
130                             if (this.currentAETSRA.answers[this.
                                    QuestionID - 1].answer != null)
131                                 this.textboxAnswer.Text = this.
                                        currentAETSRA.answers[this.
                                        QuestionID - 1].answer;
132                         }
133                     }
134                 }
135             }
136             else
137                 this.Visible = false;
138         }
139
140         private void ButtonAnswer_Click(object sender, System.
                EventArgs e)
141         {
142             string uniqueAID = this.ClassID + "|" + this.AETQID.
```

```
                    ToString() + "|" + this.AETType.ToString() + "|" + "
                    Answer";
            if (Session[uniqueAID] != null)
            {
                this.currentAETSRA = Session[uniqueAID] as
                    AETShortResponseAnswer;

                this.currentAETSRA.answers[this.QuestionID - 1].
                    questionID = this.QuestionID;
                this.currentAETSRA.answers[this.QuestionID - 1].
                    answer = this.textboxAnswer.Text;
                this.currentAETSRA.SubmittedDateTime = DateTime.Now;

                this.currentAETSRA.Alter();
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
            this.buttonAnswer.Click += new System.EventHandler(this.
                ButtonAnswer_Click);
        }
        #endregion
    }
}
```

Listing H.33: ClassControls/ShortResponseAETAnswers.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ShortResponseAETAnswers.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ClassControls.
    ShortResponseAETAnswers" TargetSchema="http://schemas.microsoft.
    com/intellisense/ie5"%>
```

Listing H.34: ClassControls/ShortResponseAETAnswers.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
```

```
    public class ShortResponseAETAnswers : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.35: ClassControls/ThreadPosts.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ThreadPosts.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ClassControls.ThreadPosts" TargetSchema="http://schemas
    .microsoft.com/intellisense/ie5"%>
<table class="PostFramework" cellspacing="0" cellpadding="0">
    <tr height="20px">
        <td></td>
    </tr>
    <tr>
        <th class="PostFramework_Header" colspan="2"><asp:Label ID="
            labelThreadTitle" Runat="server"></asp:Label></th>
    </tr>
    <asp:Repeater id="repeater" runat="server">
        <ItemTemplate>
            <tr>
                <td class="PostFramework_PostData">
                    <div class="PostFramework_Name">><%#((
                        OnlineLearningEnvironment.ClassThreadPost)
                        Container.DataItem).CreatorName%></div>
                    <div class="PostFramework_DateDescription">
                        Created</div>
                    <div class="PostFramework_Date">><%#((
                        OnlineLearningEnvironment.ClassThreadPost)
                        Container.DataItem).CreatedDateTime.ToString
                        ()%></div>
                    <div class="PostFramework_DateDescription">
                        Updated</div>
                    <div class="PostFramework_Date">><%#((
                        OnlineLearningEnvironment.ClassThreadPost)
                        Container.DataItem).UpdatedDateTime.ToString
                        ()%></div>
                </td>
                <td class="PostFramework_PostInformation">
```

```
20                          <div><%#((OnlineLearningEnvironment.
                                ClassThreadPost)Container.DataItem).
                                Information%></span></div>
21                          <asp:Repeater ID="fileAttachmentsRepeater1"
                                DataSource="<%#((OnlineLearningEnvironment.
                                ClassThreadPost)Container.DataItem).
                                AttachmentFilenames%>" Runat="server">
22                              <HeaderTemplate>
23                                  <blockquote>
24                                      <table width="100%">
25                                          <tr>
26                                              <td colspan="2">
27                                                  <div class="
                                                        PostFramework_Files">
                                                        File Attachments:</
                                                        div>
28                                              </td>
29                                          </tr>
30                                          <tr>
31                                              <td width="15px"></td>
32                                              <td>
33                                                  <div class="
                                                        PostFramework_Filenames
                                                        ">
34                              </HeaderTemplate>
35                              <ItemTemplate>
36                                  <%#Container.DataItem%>
37                              </ItemTemplate>
38                              <SeparatorTemplate>
39                                   
40                              </SeparatorTemplate>
41                              <FooterTemplate>
42                                                  </div>
43                                              </td>
44                                          </tr>
45                                      </table>
46                                  </blockquote>
47                              </FooterTemplate>
48                          </asp:Repeater>
49                      </td>
50                  </tr>
51              </ItemTemplate>
52              <AlternatingItemTemplate>
53                  <tr>
54                      <td class="PostFramework_PostData">
55                          <div class="PostFramework_Name"><%#((
                                OnlineLearningEnvironment.ClassThreadPost)
                                Container.DataItem).CreatorName%></div>
56                          <div class="PostFramework_DateDescription">
                                Created</div>
57                          <div class="PostFramework_Date"><%#((
                                OnlineLearningEnvironment.ClassThreadPost)
                                Container.DataItem).CreatedDateTime.ToString
                                ()%></div>
58                          <div class="PostFramework_DateDescription">
```

```
                        Updated</div>
59                      <div class="PostFramework_Date"><%#((
                            OnlineLearningEnvironment.ClassThreadPost)
                            Container.DataItem).UpdatedDateTime.ToString
                            ()%></div>
60                  </td>
61                  <td class="PostFramework_AlternatePostInformation">
62                      <div><%#((OnlineLearningEnvironment.
                            ClassThreadPost)Container.DataItem).
                            Information%></span></div>
63                      <asp:Repeater ID="fileAttachmentsRepeater2"
                            DataSource="<%#((OnlineLearningEnvironment.
                            ClassThreadPost)Container.DataItem).
                            AttachmentFilenames%>" Runat="server">
64                          <HeaderTemplate>
65                              <blockquote>
66                                  <table width="100%">
67                                      <tr>
68                                          <td colspan="2">
69                                              <div class="
                                                    PostFramework_Files">
                                                    File Attachments:</
                                                    div>
70                                          </td>
71                                      </tr>
72                                      <tr>
73                                          <td width="15px"></td>
74                                          <td>
75                                              <div class="
                                                    PostFramework_Filenames
                                                    ">
76                          </HeaderTemplate>
77                          <ItemTemplate>
78                              <%#Container.DataItem%>
79                          </ItemTemplate>
80                          <SeparatorTemplate>
81                               
82                          </SeparatorTemplate>
83                          <FooterTemplate>
84                                              </div>
85                                          </td>
86                                      </tr>
87                                  </table>
88                              </blockquote>
89                          </FooterTemplate>
90                      </asp:Repeater>
91                  </td>
92              </tr>
93          </AlternatingItemTemplate>
94      </asp:Repeater>
95  </table>
```

Listing H.36: ClassControls/ThreadPosts.ascx.cs

```
1  namespace OnlineLearningEnvironment.Interface.ClassControls
2  {
```

```
3       using System;
4       using System.Data;
5       using System.Drawing;
6       using System.Web;
7       using System.Web.UI.WebControls;
8       using System.Web.UI.HtmlControls;
9
10      /// <summary>
11      /// Not implemented
12      /// </summary>
13      public class ThreadPosts : System.Web.UI.UserControl
14      {
15          protected System.Web.UI.WebControls.Repeater repeater;
16          protected System.Web.UI.WebControls.Label labelThreadTitle;
17
18          private bool _enabled = false;
19          public bool Enabled
20          {
21              get { return this._enabled; }
22              set { this._enabled = value; }
23          }
24
25          private int _threadID = 0;
26          public int ThreadID
27          {
28              get { return this._threadID; }
29              set { this._threadID = value; }
30          }
31
32          private string _classID = null;
33          public string ClassID
34          {
35              get { return this._classID; }
36              set { this._classID = value; }
37          }
38
39          private void Page_Load(object sender, System.EventArgs e)
40          {
41              if (this.Enabled)
42              {
43                  this.Visible = true;
44                  if (this.ThreadID != 0)
45                  {
46                      Thread thread = Thread.FindThread(this.ClassID,
                            this.ThreadID, Session["DataBridge"] as
                            DataBridge);
47                      if (thread != null)
48                          this.labelThreadTitle.Text = thread.Title;
49
50                      ClassThreadPost[] classThreadPosts =
                            ClassThreadPost.FindClassThreadPosts(this.
                            ClassID, this.ThreadID, Session["DataBridge"]
                             as DataBridge);
51                      if (classThreadPosts != null)
52                      {
```

```
                        this.repeater.DataSource = classThreadPosts;
                        this.repeater.DataBind();
                    }
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.37: ClassControls/Threads.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    Threads.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    ClassControls.Threads" TargetSchema="http://schemas.microsoft.com
    /intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table width="100%">
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td><a href="Class.aspx?view=ThreadPosts&classID=<%#((
                OnlineLearningEnvironment.Thread)Container.DataItem).
                ClassID%>&threadID=<%#((OnlineLearningEnvironment.
                Thread)Container.DataItem).ThreadID%>"><%#((
                OnlineLearningEnvironment.Thread)Container.DataItem).
                Title%></a></td>
            <td><%#((OnlineLearningEnvironment.Thread)Container.
                DataItem).CreatorName%></td>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
            <td height="5px" bgcolor="#ffffff"></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.38: ClassControls/Threads.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class Threads : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    Thread[] threads = Thread.FindAllThreads(this.
                        ClassID, Session["DataBridge"] as DataBridge)
                        ;

                    if (threads != null)
                    {
                        this.repeater.DataSource = threads;
                        this.repeater.DataBind();
                    }
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
```

```
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.39: ConfigurationControls/ApplicationConfiguration.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ApplicationConfiguration.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ApplicationConfiguration"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.40: ConfigurationControls/ApplicationConfiguration.ascx.cs

```
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ApplicationConfiguration : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
```

```
34          {
35              this.Load += new System.EventHandler(this.Page_Load);
36          }
37          #endregion
38      }
39  }
```

Listing H.41: ConfigurationControls/ClassConfiguration.ascx

```
1  <%@ Control Language="c#" AutoEventWireup="false" Codebehind="
       ClassConfiguration.ascx.cs" Inherits="OnlineLearningEnvironment.
       Interface.ConfigurationControls.ClassConfiguration" TargetSchema=
       "http://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.42: ConfigurationControls/ClassConfiguration.ascx.cs

```
1  namespace OnlineLearningEnvironment.Interface.ConfigurationControls
2  {
3      using System;
4      using System.Data;
5      using System.Drawing;
6      using System.Web;
7      using System.Web.UI.WebControls;
8      using System.Web.UI.HtmlControls;
9
10     /// <summary>
11     /// Not implemented
12     /// </summary>
13     public class ClassConfiguration : System.Web.UI.UserControl
14     {
15         private bool _enabled = false;
16         public bool Enabled
17         {
18             get { return this._enabled; }
19             set { this._enabled = value; }
20         }
21
22         private void Page_Load(object sender, System.EventArgs e)
23         {
24         }
25
26         #region Web Form Designer generated code
27         override protected void OnInit(EventArgs e)
28         {
29             InitializeComponent();
30             base.OnInit(e);
31         }
32
33         private void InitializeComponent()
34         {
35             this.Load += new System.EventHandler(this.Page_Load);
36         }
37         #endregion
38     }
39 }
```

Listing H.43: ConfigurationControls/ClassEnrollmentConfiguration.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ClassEnrollmentConfiguration.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ConfigurationControls.
    ClassEnrollmentConfiguration" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
```

Listing H.44: ConfigurationControls/ClassEnrollmentConfiguration.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassEnrollmentConfiguration : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.45: ConfigurationControls/InstitutionConfiguration.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    InstitutionConfiguration.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.InstitutionConfiguration"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.46: ConfigurationControls/InstitutionConfiguration.ascx.cs

```
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class InstitutionConfiguration : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.47: ConfigurationControls/RoleAssignmentConfiguration.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    RoleAssignmentConfiguration.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ConfigurationControls.
    RoleAssignmentConfiguration" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
```

Listing H.48: ConfigurationControls/RoleAssignmentConfiguration.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{
    using System;
    using System.Data;
```

```
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class RoleAssignmentConfiguration : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.49: ConfigurationControls/UserConfiguration.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    UserConfiguration.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ConfigurationControls.UserConfiguration" TargetSchema="
    http://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.50: ConfigurationControls/UserConfiguration.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
```

```
    /// Not implemented
    /// </summary>
    public class UserConfiguration : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.51: HomeControls/ErrorMessages.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ErrorMessages.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.HomeControls.ErrorMessages" TargetSchema="http://
    schemas.microsoft.com/intellisense/ie5"%>
<asp:Repeater id="repeaterSecurityErrors" runat="server">
    <HeaderTemplate>
        <table class="ErrorFramework" cellspacing="0" cellpadding="0
            ">
            <tr height="20px"><td></td></tr>
            <tr>
                <th class="ErrorFramework_Header" colspan="3">
                    Security Errors</th>
            </tr>
            <tr>
                <th class="ErrorFramework_Columns">Date/Time</th>
                <th class="ErrorFramework_Columns">UserID</th>
                <th class="ErrorFramework_Columns">IP Address</th>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
            <tr>
                <td class="ErrorFramework_Date"><%#((
                    OnlineLearningEnvironment.SecurityError)Container
                    .DataItem).CreatedDateTime.ToString()%></td>
```

```
18                  <td class="ErrorFramework_Data" style="border-right: solid 1px #AAAAAA;"><%#((OnlineLearningEnvironment.SecurityError)Container.DataItem).UserID.ToString()%></td>
19                  <td class="ErrorFramework_Data"><%#((OnlineLearningEnvironment.SecurityError)Container.DataItem).IPAddress%></td>
20              </tr>
21      </ItemTemplate>
22      <AlternatingItemTemplate>
23              <tr>
24                  <td class="ErrorFramework_Date"><%#((OnlineLearningEnvironment.SecurityError)Container.DataItem).CreatedDateTime.ToString()%></td>
25                  <td class="ErrorFramework_AlternateData" style="border-right: solid 1px #AAAAAA;"><%#((OnlineLearningEnvironment.SecurityError)Container.DataItem).UserID.ToString()%></td>
26                  <td class="ErrorFramework_AlternateData"><%#((OnlineLearningEnvironment.SecurityError)Container.DataItem).IPAddress%></td>
27              </tr>
28      </AlternatingItemTemplate>
29      <FooterTemplate>
30          </table>
31      </FooterTemplate>
32  </asp:Repeater>
33  <asp:Repeater id="repeaterApplicationErrors" runat="server">
34      <HeaderTemplate>
35          <table class="ErrorFramework" cellspacing="0" cellpadding="0">
36              <tr height="20px"><td></td></tr>
37              <tr>
38                  <th class="ErrorFramework_Header" colspan="2">Application Errors</th>
39              </tr>
40              <tr>
41                  <th class="ErrorFramework_Columns">Date/Time</th>
42                  <th class="ErrorFramework_Columns">Error Information</th>
43              </tr>
44      </HeaderTemplate>
45      <ItemTemplate>
46              <tr>
47                  <td class="ErrorFramework_Date"><%#((OnlineLearningEnvironment.ApplicationError)Container.DataItem).CreatedDateTime.ToString()%></td>
48                  <td class="ErrorFramework_Information"><%#((OnlineLearningEnvironment.ApplicationError)Container.DataItem).Information%></td>
49              </tr>
50      </ItemTemplate>
51      <AlternatingItemTemplate>
52              <tr>
```

```
53                  <td class="ErrorFramework_Date"><%#((
                        OnlineLearningEnvironment.ApplicationError)
                        Container.DataItem).CreatedDateTime.ToString()%><
                        /td>
54                  <td class="ErrorFramework_AlternateInformation"><
                        %#((OnlineLearningEnvironment.ApplicationError)
                        Container.DataItem).Information%></td>
55              </tr>
56      </AlternatingItemTemplate>
57      <FooterTemplate>
58          </table>
59      </FooterTemplate>
60  </asp:Repeater>
```

Listing H.52: HomeControls/ErrorMessages.ascx.cs

```
1   namespace OnlineLearningEnvironment.Interface.HomeControls
2   {
3       using System;
4       using System.Collections;
5       using System.Data;
6       using System.Drawing;
7       using System.Web;
8       using System.Web.UI.WebControls;
9       using System.Web.UI.HtmlControls;
10
11      public class ErrorMessages : System.Web.UI.UserControl
12      {
13          protected System.Web.UI.WebControls.Repeater
                repeaterSecurityErrors;
14          protected System.Web.UI.WebControls.Repeater
                repeaterApplicationErrors;
15
16          private bool _enabled = true;
17          public bool Enabled
18          {
19              get { return this._enabled; }
20              set { this._enabled = value; }
21          }
22
23          private void Page_Load(object sender, System.EventArgs e)
24          {
25              if (this.Enabled)
26              {
27                  this.Visible = true;
28                  SecurityError[] securityErrors = SecurityError.
                        FindSecurityErrors(Session["DataBridge"] as
                        DataBridge);
29                  if (securityErrors != null)
30                  {
31                      ArrayList tempArray = new ArrayList(
                            securityErrors);
32                      if (tempArray.Count > 5)
33                          tempArray.RemoveRange(5, tempArray.Count -
                                5);
34                      securityErrors = (SecurityError[])tempArray.
```

```
                        ToArray(typeof(SecurityError));
                    Array.Reverse(securityErrors);
                    this.repeaterSecurityErrors.DataSource = securityErrors;
                    this.repeaterSecurityErrors.DataBind();
                }

                ApplicationError[] applicationErrors = ApplicationError.FindApplicationErrors(Session["DataBridge"] as DataBridge);
                if (applicationErrors != null)
                {
                    ArrayList tempArray = new ArrayList(applicationErrors);
                    if (tempArray.Count > 5)
                        tempArray.RemoveRange(5, tempArray.Count - 5);
                    applicationErrors = (ApplicationError[])tempArray.ToArray(typeof(ApplicationError));
                    Array.Reverse(applicationErrors);
                    this.repeaterApplicationErrors.DataSource = applicationErrors;
                    this.repeaterApplicationErrors.DataBind();
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.53: HomeControls/ForumThreadSubscriptions.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="ForumThreadSubscriptions.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.HomeControls.ForumThreadSubscriptions" TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.54: HomeControls/ForumThreadSubscriptions.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.HomeControls
```

```
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ForumThreadSubscriptions : System.Web.UI.
        UserControl
    {
        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.55: HomeControls/InstitutionAnnouncements.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    InstitutionAnnouncements.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.HomeControls.
    InstitutionAnnouncements" TargetSchema="http://schemas.microsoft.
    com/intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table class="PostFramework" cellspacing="0" cellpadding="0"
            >
            <tr height="20px">
                <td></td>
            </tr>
            <tr>
                <th class="PostFramework_Header" colspan="2">
```

```
10                      Announcements</th>
11                  </tr>
12          </HeaderTemplate>
13          <ItemTemplate>
14              <tr>
15                  <td class="PostFramework_PostData">
16                      <div class="PostFramework_Name"><%#((
                            OnlineLearningEnvironment.InstitutionAnnouncement
                            )Container.DataItem).CreatorName%></div>
17                      <div class="PostFramework_DateDescription">Created</
                            div>
18                      <div class="PostFramework_Date"><%#((
                            OnlineLearningEnvironment.InstitutionAnnouncement
                            )Container.DataItem).CreatedDateTime.ToString()%>
                            </div>
19                      <div class="PostFramework_DateDescription">Updated</
                            div>
20                      <div class="PostFramework_Date"><%#((
                            OnlineLearningEnvironment.InstitutionAnnouncement
                            )Container.DataItem).UpdatedDateTime.ToString()%>
                            </div>
21                  </td>
22                  <td class="PostFramework_PostInformation">
23                      <div><%#((OnlineLearningEnvironment.
                            InstitutionAnnouncement)Container.DataItem).
                            Information%></span></div>
24                      <asp:Repeater ID="fileAttachmentsRepeater1"
                            DataSource="<%#((OnlineLearningEnvironment.
                            InstitutionAnnouncement)Container.DataItem).
                            AttachmentFilenames%>" Runat="server">
25                          <HeaderTemplate>
26                              <blockquote>
27                                  <table width="100%">
28                                      <tr>
29                                          <td colspan="2">
30                                              <div class="
                                                    PostFramework_Files">File
                                                     Attachments:</div>
31                                          </td>
32                                      </tr>
33                                      <tr>
34                                          <td width="15px"></td>
35                                          <td>
36                                              <div class="
                                                    PostFramework_Filenames">
37                          </HeaderTemplate>
38                          <ItemTemplate>
39                              <%#Container.DataItem%>
40                          </ItemTemplate>
41                          <SeparatorTemplate>
42                               
43                          </SeparatorTemplate>
44                          <FooterTemplate>
45                                              </div>
46                                          </td>
```

```
47                                  </tr>
48                              </table>
49                          </blockquote>
50                      </FooterTemplate>
51                  </asp:Repeater>
52              </td>
53          </tr>
54      </ItemTemplate>
55      <AlternatingItemTemplate>
56          <tr>
57              <td class="PostFramework_PostData">
58                  <div class="PostFramework_Name"><%#((
                        OnlineLearningEnvironment.InstitutionAnnouncement
                        )Container.DataItem).CreatorName%></div>
59                  <div class="PostFramework_DateDescription">Created</
                        div>
60                  <div class="PostFramework_Date"><%#((
                        OnlineLearningEnvironment.InstitutionAnnouncement
                        )Container.DataItem).CreatedDateTime.ToString()%>
                        </div>
61                  <div class="PostFramework_DateDescription">Updated</
                        div>
62                  <div class="PostFramework_Date"><%#((
                        OnlineLearningEnvironment.InstitutionAnnouncement
                        )Container.DataItem).UpdatedDateTime.ToString()%>
                        </div>
63              </td>
64              <td class="PostFramework_AlternatePostInformation">
65                  <div><%#((OnlineLearningEnvironment.
                        InstitutionAnnouncement)Container.DataItem).
                        Information%></span></div>
66                  <asp:Repeater ID="fileAttachmentsRepeater2"
                        DataSource="<%#((OnlineLearningEnvironment.
                        InstitutionAnnouncement)Container.DataItem).
                        AttachmentFilenames%>" Runat="server">
67                      <HeaderTemplate>
68                          <blockquote>
69                              <table width="100%">
70                                  <tr>
71                                      <td colspan="2">
72                                          <div class="
                                                PostFramework_Files">File
                                                 Attachments:</div>
73                                      </td>
74                                  </tr>
75                                  <tr>
76                                      <td width="15px"></td>
77                                      <td>
78                                          <div class="
                                                PostFramework_Filenames">
79                      </HeaderTemplate>
80                      <ItemTemplate>
81                          <%#Container.DataItem%>
82                      </ItemTemplate>
83                      <SeparatorTemplate>
```

```
                         
                    </SeparatorTemplate>
                    <FooterTemplate>
                                        </div>
                                    </td>
                                </tr>
                            </table>
                        </blockquote>
                    </FooterTemplate>
                </asp:Repeater>
            </td>
        </tr>
    </AlternatingItemTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.56: HomeControls/InstitutionAnnouncements.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.HomeControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class InstitutionAnnouncements : System.Web.UI.
        UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                InstitutionAnnouncement[] announcements =
                    InstitutionAnnouncement.
                    FindInstitutionAnnouncements(Session["DataBridge"
                    ] as DataBridge);
                if (announcements != null)
                {
                    Array.Reverse(announcements);

                    this.repeater.DataSource = announcements;
```

```
                    this.repeater.DataBind();
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.57: HomeControls/QuickLinks.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    QuickLinks.ascx.cs" Inherits="OnlineLearningEnvironment.Interface
    .HomeControls.QuickLinks" TargetSchema="http://schemas.microsoft.
    com/intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table class="QLFramework" cellspacing="0" cellpadding="0">
            <tr height="20px"><td></td></tr>
            <tr>
                <th class="QLFramework_Header">Quick Links</th>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
            <tr>
                <td class="QLFramework_Class"><a href="Class.aspx?
                    classID=<%#((OnlineLearningEnvironment.Class)
                    Container.DataItem).ClassID%>"><%#((
                    OnlineLearningEnvironment.Class)Container.
                    DataItem).ClassID%> (<%#((
                    OnlineLearningEnvironment.Class)Container.
                    DataItem).ClassName%>)</a></td>
            </tr>
    </ItemTemplate>
    <AlternatingItemTemplate>
            <tr>
                <td class="QLFramework_AlternateClass"><a href="
                    Class.aspx?classID=<%#((OnlineLearningEnvironment
                    .Class)Container.DataItem).ClassID%>"><%#((
                    OnlineLearningEnvironment.Class)Container.
                    DataItem).ClassID%> (<%#((
                    OnlineLearningEnvironment.Class)Container.
                    DataItem).ClassName%>)</a></td>
```

```
            </tr>
    </AlternatingItemTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.58: HomeControls/QuickLinks.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.HomeControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class QuickLinks : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                Class[] classes = Class.FindClasses(Int32.Parse(
                    Context.User.Identity.Name), Session["DataBridge"
                    ] as DataBridge);
                if (classes != null)
                {
                    this.repeater.DataSource = classes;
                    this.repeater.DataBind();
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
```

```
46          {
47              this.Load += new System.EventHandler(this.Page_Load);
48          }
49          #endregion
50      }
51  }
```

Listing H.59: HomeControls/UpcomingDates.ascx

```
1   <%@ Control Language="c#" AutoEventWireup="false" Codebehind="
        UpcomingDates.ascx.cs" Inherits="OnlineLearningEnvironment.
        Interface.HomeControls.UpcomingDates" TargetSchema="http://
        schemas.microsoft.com/intellisense/ie5"%>
2   <asp:Repeater id="repeaterAssignments" runat="server">
3       <HeaderTemplate>
4           <table class="AEFramework" cellspacing="0" cellpadding="0">
5               <tr height="20px"><td></td></tr>
6               <tr>
7                   <th class="AEFramework_Header" colspan="2">
                        Assignments</th>
8               </tr>
9       </HeaderTemplate>
10      <ItemTemplate>
11              <tr>
12                  <td class="AEFramework_Date"><%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
                        Container.DataItem).DueDateTime%></td>
13                  <td class="AEFramework_Description">
14                      <div><a href="Class.aspx?view=<%#((
                            OnlineLearningEnvironment.ShortAETQuestion)
                            Container.DataItem).QuestionType%>AET&classID
                            =<%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).ClassID
                            %>&aetQID=<%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).AETQID
                            %>&aetType=<%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).AETType
                            %>"><%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).
                            Description%></a></div>
15                      <div class="AEFramework_ClassID">(<%#((
                            OnlineLearningEnvironment.ShortAETQuestion)
                            Container.DataItem).ClassID%>)</div>
16                  </td>
17              </tr>
18      </ItemTemplate>
19      <AlternatingItemTemplate>
20              <tr>
21                  <td class="AEFramework_Date"><%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
                        Container.DataItem).DueDateTime%></td>
22                  <td class="AEFramework_AlternateDescription">
23                      <div><a href="Class.aspx?view=<%#((
                            OnlineLearningEnvironment.ShortAETQuestion)
                            Container.DataItem).QuestionType%>AET&classID
                            =<%#((OnlineLearningEnvironment.
```

```
                        ShortAETQuestion)Container.DataItem).ClassID
                        %>&aetQID=<%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).AETQID
                        %>&aetType=<%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).AETType
                        %>"><%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).
                        Description%></a></div>
24                  <div class="AEFramework_ClassID">(<%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
                        Container.DataItem).ClassID%>)</div>
25              </td>
26          </tr>
27      </AlternatingItemTemplate>
28      <FooterTemplate>
29          </table>
30      </FooterTemplate>
31  </asp:Repeater>
32  <asp:Repeater id="repeaterExams" runat="server">
33      <HeaderTemplate>
34          <table class="AEFramework" cellspacing="0" cellpadding="0">
35              <tr height="20px"><td></td></tr>
36              <tr>
37                  <th class="AEFramework_Header" colspan="2">Exams</th
                        >
38              </tr>
39      </HeaderTemplate>
40      <ItemTemplate>
41              <tr>
42                  <td class="AEFramework_Date"><%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
                        Container.DataItem).DueDateTime%></td>
43                  <td class="AEFramework_Description">
44                      <div><a href="Class.aspx?view=<%#((
                            OnlineLearningEnvironment.ShortAETQuestion)
                            Container.DataItem).QuestionType%>AET&classID
                            =<%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).ClassID
                            %>&aetQID=<%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).AETQID
                            %>&aetType=<%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).AETType
                            %>"><%#((OnlineLearningEnvironment.
                            ShortAETQuestion)Container.DataItem).
                            Description%></a></div>
45                      <div class="AEFramework_ClassID">(<%#((
                            OnlineLearningEnvironment.ShortAETQuestion)
                            Container.DataItem).ClassID%>)</div>
46                  </td>
47              </tr>
48      </ItemTemplate>
49      <AlternatingItemTemplate>
50              <tr>
51                  <td class="AEFramework_Date"><%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
```

```
                    Container.DataItem).DueDateTime%></td>
                <td class="AEFramework_AlternateDescription">
                    <div><a href="Class.aspx?view=<%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
                        Container.DataItem).QuestionType%>AET&classID
                        =<%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).ClassID
                        %>&aetQID=<%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).AETQID
                        %>&aetType=<%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).AETType
                        %>"><%#((OnlineLearningEnvironment.
                        ShortAETQuestion)Container.DataItem).
                        Description%></a></div>
                    <div class="AEFramework_ClassID">(<%#((
                        OnlineLearningEnvironment.ShortAETQuestion)
                        Container.DataItem).ClassID%>)</div>
                </td>
            </tr>
    </AlternatingItemTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.60: HomeControls/UpcomingDates.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.HomeControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class UpcomingDates : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater
            repeaterAssignments;
        protected System.Web.UI.WebControls.Repeater repeaterExams;

        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                Class[] classes = Class.FindClasses(Int32.Parse(
```

```
                    Context.User.Identity.Name), Session["DataBridge"] as DataBridge);
                if (classes != null)
                {
                    ArrayList assignments = new ArrayList();
                    bool isAssignments = false;
                    ArrayList exams = new ArrayList();
                    bool isExams = false;
                    foreach (Class c in classes)
                    {
                        try
                        {
                            assignments.AddRange(ShortAETQuestion.FindShortAETQuestions(c.ClassID, AETType.Assignment, Session["DataBridge"] as DataBridge));
                            isAssignments = true;
                        }
                        catch
                        {
                            if (!isAssignments)
                                isAssignments = false;
                        }
                        try
                        {
                            exams.AddRange(ShortAETQuestion.FindShortAETQuestions(c.ClassID, AETType.Exam, Session["DataBridge"] as DataBridge));
                            isExams = true;
                        }
                        catch
                        {
                            if (!isExams)
                                isExams = false;
                        }
                    }

                    if (isAssignments)
                    {
                        this.repeaterAssignments.DataSource = assignments;
                        this.repeaterAssignments.DataBind();
                    }
                    if (isExams)
                    {
                        this.repeaterExams.DataSource = exams;
                        this.repeaterExams.DataBind();
                    }
                }
            }
            else
                this.Visible = false;
        }

```

```
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.61: LoginControls/Authentication.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    Authentication.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.LoginControls.Authentication" TargetSchema="http://
    schemas.microsoft.com/intellisense/ie5"%>
<table class="LoginFramework">
    <tr>
        <td class="LoginFramework_Username">Username:</td>
    </tr>
    <tr>
        <td align="right"><asp:textbox id="textBoxUsername" tabIndex
            ="1" runat="server" MaxLength="255" Width="200px"></asp:
            textbox></td>
    </tr>
    <tr>
        <td class="LoginFramework_Password">Password:</td>
    </tr>
    <tr>
        <td align="right"><asp:textbox id="textBoxPassword" TextMode
            ="Password" tabIndex="2" runat="server" MaxLength="255"
                Width="200px"></asp:textbox></td>
    </tr>
    <TR>
        <TD class="LoginFramework_RememberMe"><asp:CheckBox id="
            checkBoxRememberMe" tabIndex="3" runat="server" Text="
            Remember me" /></TD>
    </TR>
    <tr>
        <td class="LoginFramework_Login"><asp:button id="loginButton
            " tabIndex="4" runat="server" Text="Login"></asp:button><
            /td>
    </tr>
    <TR>
        <TD class="LoginFramework_Validators"><asp:Label id="
            labelNotFound" Visible="False" runat="server" ForeColor="
            Red">Invalid username or password!</asp:Label></TD>
    </TR>
    <tr>
        <td class="LoginFramework_Validators"><asp:
            requiredfieldvalidator id="validatorUsernameExists" runat
```

```
            ="server" ErrorMessage="RequiredFieldValidator"
                Display="Dynamic" ControlToValidate="textBoxUsername
                    ">Username field cannot be empty!</asp:
                    requiredfieldvalidator></td>
    </tr>
    <tr>
        <td class="LoginFramework_Validators"><asp:
            requiredfieldvalidator id="validatorPasswordExists" runat
            ="server" ErrorMessage="RequiredFieldValidator"
                Display="Dynamic" ControlToValidate="textBoxPassword
                    ">Password field cannot be empty!</asp:
                    requiredfieldvalidator></td>
    </tr>
</table>
```

Listing H.62: LoginControls/Authentication.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.LoginControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.Security;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Authentication : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Button loginButton;
        protected System.Web.UI.WebControls.TextBox textBoxUsername;
        protected System.Web.UI.WebControls.RequiredFieldValidator
            validatorUsernameExists;
        protected System.Web.UI.WebControls.RequiredFieldValidator
            validatorPasswordExists;
        protected System.Web.UI.WebControls.CheckBox
            checkBoxRememberMe;
        protected System.Web.UI.WebControls.Label labelNotFound;
        protected System.Web.UI.WebControls.TextBox textBoxPassword;

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.loginButton.Click += new System.EventHandler(this.
                loginButton_Click);
            this.Load += new System.EventHandler(this.Page_Load);
```

```
36          }
37          #endregion
38
39          private void loginButton_Click(object sender, System.
                EventArgs e)
40          {
41              bool isAuthenticated;
42
43              try
44              {
45                  isAuthenticated = OnlineLearningEnvironment.
                        Authentication.CheckAuthentication(
46                      Int32.Parse(this.textBoxUsername.Text),
47                      this.textBoxPassword.Text,
48                      Session["DataBridge"] as DataBridge);
49              }
50              catch (System.FormatException)
51              {
52                  // Not authenticated
53                  isAuthenticated = false;
54              }
55
56              // Authenticated?
57              if (isAuthenticated)
58              {
59                  FormsAuthenticationTicket aTicket = new
                        FormsAuthenticationTicket(
60                      1,
61                      this.textBoxUsername.Text,
62                      DateTime.Now,
63                      DateTime.Now.AddMonths(1),
64                      this.checkBoxRememberMe.Checked,
65                      null);
66
67                  string encryptedATicket = FormsAuthentication.
                        Encrypt(aTicket);
68
69                  HttpCookie httpCookie = new HttpCookie(
70                      FormsAuthentication.FormsCookieName,
71                      encryptedATicket);
72                  Response.Cookies.Add(httpCookie);
73
74                  FormsAuthentication.RedirectFromLoginPage(
75                      this.textBoxUsername.Text,
76                      this.checkBoxRememberMe.Checked);
77              }
78              else
79              {
80                  // Display not found message
81                  this.labelNotFound.Visible = true;
82              }
83          }
84      }
85  }
```

Listing H.63: ManagementControls/ClassAnnouncementManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ClassAnnouncementManagement.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ManagementControls.
    ClassAnnouncementManagement" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
```

Listing H.64: ManagementControls/ClassAnnouncementManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassAnnouncementManagement : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.65: ManagementControls/ClassManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ClassManagement.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ManagementControls.ClassManagement" TargetSchema="http
    ://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.66: ManagementControls/ClassManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassManagement : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.67: ManagementControls/ClassNoteManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ClassNoteManagement.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ManagementControls.ClassNoteManagement" TargetSchema="
    http://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.68: ManagementControls/ClassNoteManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassNoteManagement : System.Web.UI.UserControl
```

```
    {

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.69: ManagementControls/InformationAETManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    InformationAETManagement.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ManagementControls.
    InformationAETManagement" TargetSchema="http://schemas.microsoft.
    com/intellisense/ie5"%>
```

Listing H.70: ManagementControls/InformationAETManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class InformationAETManagement : System.Web.UI.
        UserControl
    {

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
```

```
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.71: ManagementControls/InstitutionAnnouncementManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    InstitutionAnnouncementManagement.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ManagementControls.
    InstitutionAnnouncementManagement" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
```

Listing H.72: ManagementControls/InstitutionAnnouncementManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class InstitutionAnnouncementManagement : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
```

```
        }
        #endregion
    }
}
```

Listing H.73: ManagementControls/MultipleChoiceAETManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    MultipleChoiceAETManagement.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ManagementControls.
    MultipleChoiceAETManagement" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
```

Listing H.74: ManagementControls/MultipleChoiceAETManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class MultipleChoiceAETManagement : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.75: ManagementControls/ShortResponseAETManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    ShortResponseAETManagement.ascx.cs" Inherits="
    OnlineLearningEnvironment.Interface.ManagementControls.
    ShortResponseAETManagement" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
```

Listing H.76: ManagementControls/ShortResponseAETManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ShortResponseAETManagement : System.Web.UI.
        UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.77: ManagementControls/UserManagement.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    UserManagement.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.ManagementControls.UserManagement" TargetSchema="http
    ://schemas.microsoft.com/intellisense/ie5"%>
```

Listing H.78: ManagementControls/UserManagement.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class UserManagement : System.Web.UI.UserControl
    {

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.79: MenuControls/Assignments.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    Assignments.ascx.cs" Inherits="OnlineLearningEnvironment.
    Interface.MenuControls.Assignments" TargetSchema="http://schemas.
    microsoft.com/intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table>
            <tr>
                <td><b><i><%#(Application["Configuration"] as
                    OnlineLearningEnvironment.Configuration).
                    Assignment%>s</i></b></td>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td width="80%"><a href="Class.aspx?view=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).QuestionType%>AET&classID=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
```

```
                    DataItem).ClassID%>&aetQID=<%#((
                    OnlineLearningEnvironment.ShortAETQuestion)Container.
                    DataItem).AETQID%>&aetType=<%#((
                    OnlineLearningEnvironment.ShortAETQuestion)Container.
                    DataItem).AETType%>"><%#((OnlineLearningEnvironment.
                    ShortAETQuestion)Container.DataItem).Description%></a
                    ></td>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
            <td height="5px" bgcolor="#ffffff"></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.80: MenuControls/Assignments.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Assignments : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    try
```

```
                        {
                            ShortAETQuestion[] sAETQuestions =
                                ShortAETQuestion.FindShortAETQuestions(
                                this.ClassID, AETType.Assignment, Session
                                ["DataBridge"] as DataBridge);
                            this.repeater.DataSource = sAETQuestions;
                            this.repeater.DataBind();
                        }
                        catch {}
                    }
                }
                else
                    this.Visible = false;
            }

            #region Web Form Designer generated code
            override protected void OnInit(EventArgs e)
            {
                InitializeComponent();
                base.OnInit(e);
            }

            private void InitializeComponent()
            {
                this.Load += new System.EventHandler(this.Page_Load);

            }
            #endregion
        }
}
```

Listing H.81: MenuControls/Classes.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    Classes.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    MenuControls.Classes" TargetSchema="http://schemas.microsoft.com/
    intellisense/ie5" %>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table>
            <tr>
                <td><b><i><%#(Application["Configuration"] as
                    OnlineLearningEnvironment.Configuration).Class%>s
                    </i></b></td>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td><a href="Class.aspx?classID=<%#((
                OnlineLearningEnvironment.Class)Container.DataItem).
                ClassID%>"><%#((OnlineLearningEnvironment.Class)
                Container.DataItem).ClassID%></a></td>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
```

```
            <td height="5px" bgcolor="#ffffff"></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.82: MenuControls/Classes.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Classes : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                try
                {
                    Class[] classes = Class.FindClasses(Int32.Parse(
                        Context.User.Identity.Name), Session["
                        DataBridge"] as DataBridge);
                    this.repeater.DataSource = classes;
                    this.repeater.DataBind();
                }
                catch {}
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

```

```
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.83: MenuControls/Exams.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Exams.
    ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    MenuControls.Exams" TargetSchema="http://schemas.microsoft.com/
    intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table>
            <tr>
                <td><b><i><%#(Application["Configuration"] as
                    OnlineLearningEnvironment.Configuration).Exam%>s<
                    /i></b></td>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td width="80%"><a href="Class.aspx?view=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).QuestionType%>AET&classID=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).ClassID%>&aetQID=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).AETQID%>&aetType=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).AETType%>"><%#((OnlineLearningEnvironment.
                ShortAETQuestion)Container.DataItem).Description%></a
                ></td>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
            <td height="5px" bgcolor="#ffffff"></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.84: MenuControls/Exams.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
```

```
7       using System.Web.UI.WebControls;
8       using System.Web.UI.HtmlControls;
9
10      public class Exams : System.Web.UI.UserControl
11      {
12          protected System.Web.UI.WebControls.Repeater repeater;
13
14          private bool _enabled = false;
15          public bool Enabled
16          {
17              get { return this._enabled; }
18              set { this._enabled = value; }
19          }
20
21          private string _classID = null;
22          public string ClassID
23          {
24              get { return this._classID; }
25              set { this._classID = value; }
26          }
27
28          private void Page_Load(object sender, System.EventArgs e)
29          {
30              if (this.Enabled)
31              {
32                  this.Visible = true;
33                  if (this.ClassID != null)
34                  {
35                      try
36                      {
37                          ShortAETQuestion[] sAETQuestions =
                                ShortAETQuestion.FindShortAETQuestions(
                                this.ClassID, AETType.Exam, Session["
                                DataBridge"] as DataBridge);
38                          this.repeater.DataSource = sAETQuestions;
39                          this.repeater.DataBind();
40                      }
41                      catch {}
42                  }
43              }
44              else
45                  this.Visible = false;
46          }
47
48          #region Web Form Designer generated code
49          override protected void OnInit(EventArgs e)
50          {
51              InitializeComponent();
52              base.OnInit(e);
53          }
54
55          private void InitializeComponent()
56          {
57              this.Load += new System.EventHandler(this.Page_Load);
58          }
```

```
        #endregion
    }
}
```

Listing H.85: MenuControls/Notes.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Notes.
    ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    MenuControls.Notes" TargetSchema="http://schemas.microsoft.com/
    intellisense/ie5" debug="False"%>
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table>
            <tr>
                <td><b><i><%#(Application["Configuration"] as
                    OnlineLearningEnvironment.Configuration).Note%>s<
                    /i></b></td>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td width="80%"><a href="Class.aspx?view=Notes&AET&
                classID=<%#((OnlineLearningEnvironment.ClassNote)
                Container.DataItem).ClassID%>&noteID=<%#((
                OnlineLearningEnvironment.ClassNote)Container.
                DataItem).PostID%>"><%#((OnlineLearningEnvironment.
                ClassNote)Container.DataItem).Description%></a></td>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
            <td height="5px" bgcolor="#ffffff"></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.86: MenuControls/Notes.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Notes : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
```

```
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    ClassNote[] classNotes = ClassNote.
                        FindClassNotes(this.ClassID, Session["
                        DataBridge"] as DataBridge);
                    if (classNotes != null)
                    {
                        this.repeater.DataSource = classNotes;
                        this.repeater.DataBind();
                    }
                }
            }
            else
                this.Visible = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

Listing H.87: MenuControls/Tutorials.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="
    Tutorials.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    MenuControls.Tutorials" TargetSchema="http://schemas.microsoft.
    com/intellisense/ie5"%>
<asp:Repeater id="repeater" runat="server">
```

```
    <HeaderTemplate>
        <table>
            <tr>
                <td><b><i><%#(Application["Configuration"] as
                    OnlineLearningEnvironment.Configuration).Tutorial
                    %>s</i></i></b></td>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td width="80%"><a href="Class.aspx?view=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).QuestionType%>AET&classID=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).ClassID%>&aetQID=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).AETQID%>&aetType=<%#((
                OnlineLearningEnvironment.ShortAETQuestion)Container.
                DataItem).AETType%>"><%#((OnlineLearningEnvironment.
                ShortAETQuestion)Container.DataItem).Description%></a
                ></td>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
            <td height="5px" bgcolor="#ffffff"></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Listing H.88: MenuControls/Tutorials.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Tutorials : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
```

```
22          public string ClassID
23          {
24              get { return this._classID; }
25              set { this._classID = value; }
26          }
27
28          private void Page_Load(object sender, System.EventArgs e)
29          {
30              if (this.Enabled)
31              {
32                  this.Visible = true;
33                  if (this.ClassID != null)
34                  {
35                      try
36                      {
37                          ShortAETQuestion[] sAETQuestions =
                                ShortAETQuestion.FindShortAETQuestions(
                                this.ClassID, AETType.Tutorial, Session["
                                DataBridge"] as DataBridge);
38                          this.repeater.DataSource = sAETQuestions;
39                          this.repeater.DataBind();
40                      }
41                      catch {}
42                  }
43              }
44              else
45                  this.Visible = false;
46          }
47
48          #region Web Form Designer generated code
49          override protected void OnInit(EventArgs e)
50          {
51              InitializeComponent();
52              base.OnInit(e);
53          }
54
55          private void InitializeComponent()
56          {
57              this.Load += new System.EventHandler(this.Page_Load);
58          }
59          #endregion
60      }
61  }
```

Listing H.89: SubmissionControls/Class.ascx

```
1  <%@ Control Language="c#" AutoEventWireup="false" Codebehind="Class.
       ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
       SubmissionControls.Class" TargetSchema="http://schemas.microsoft.
       com/intellisense/ie5"%>
```

Listing H.90: SubmissionControls/Class.ascx.cs

```
1  namespace OnlineLearningEnvironment.Interface.SubmissionControls
2  {
3      using System;
```

```
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class Class : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.91: SubmissionControls/User.ascx

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="User.
    ascx.cs" Inherits="OnlineLearningEnvironment.Interface.
    SubmissionControls.User" TargetSchema="http://schemas.microsoft.
    com/intellisense/ie5"%>
```

Listing H.92: SubmissionControls/User.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.SubmissionControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
```

```
    /// Not implemented
    /// </summary>
    public class User : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

H.2 OLEBusinessLogic Project

Listing H.93: Address.cs

```
using System;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// The structure of a postal/residential Address.
    /// </summary>
    public struct AddressStructure
    {
        #region Properties
        /// <summary>
        /// Street number and name.
        /// </summary>
        private string _street;
        [XmlElement("Street")]
        public string Street
        {
            get { return _street; }
```

```
            set { this._street = value; }
        }

        /// <summary>
        /// Suburb/City.
        /// </summary>
        private string _suburb;
        [XmlElement("Suburb")]
        public string Suburb
        {
            get { return _suburb; }
            set { this._suburb = value; }
        }

        /// <summary>
        /// State of the country.
        /// </summary>
        private string _state;
        [XmlElement("State")]
        public string State
        {
            get { return _state; }
            set { this._state = value; }
        }

        /// <summary>
        /// Postcode of the country.
        /// </summary>
        private string _postcode;
        [XmlElement("Postcode")]
        public string Postcode
        {
            get { return _postcode; }
            set { this._postcode = value; }
        }

        /// <summary>
        /// Country.
        /// </summary>
        private string _country;
        [XmlElement("Country")]
        public string Country
        {
            get { return _country; }
            set { this._country = value; }
        }
        #endregion
    }
}
```

Listing H.94: AETAnswer.cs

```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
```

```
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    #region Structures
    /// <summary>
    /// The structure of a short response answer.
    /// </summary>
    public struct ShortResponseAnswer
    {
        /// <summary>
        /// The QuestionID of the AETQuestion.
        /// </summary>
        [XmlAttribute("questionID")]
        public int questionID;
        /// <summary>
        /// A written answer to the question.
        /// </summary>
        [XmlText()]
        public string answer;
        /// <summary>
        /// An Instructor will set true if answer is correct and
            false if incorrect.
        /// </summary>
        [XmlAttribute("answeredCorrectly")]
        public bool answeredCorrectly;
    }

    /// <summary>
    /// The structure of a multiple choice answer.
    /// </summary>
    public struct MultipleChoiceAnswer
    {
        /// <summary>
        /// The QuestionID of the AETQuestion.
        /// </summary>
        [XmlAttribute("questionID")]
        public int questionID;
        /// <summary>
        /// The answer choiceID from the AETQuestion's choices.
        /// </summary>
        [XmlAttribute("choiceID")]
        public int choiceID;
        /// <summary>
        /// Assigned true by the application equal to AETQuestion.
            Questions.correctChoice.
        /// </summary>
        [XmlAttribute("answeredCorrectly")]
        public bool answeredCorrectly;
    }
    #endregion

```

```
58
59      /// <summary>
60      /// A structure similar to AETAnswer, but without the answer
            properties.
61      /// </summary>
62      [XmlRoot("ShortAETAnswer")]
63      public struct ShortAETAnswer
64      {
65          #region Properties
66          /// <summary>
67          /// The UserID associated with the AETAnswer.
68          /// </summary>
69          private int _userID;
70          [XmlAttribute("userID")]
71          public int UserID
72          {
73              get { return this._userID; }
74              set { this._userID = value; }
75          }
76
77          /// <summary>
78          /// The ID of the AETQuestion associated with the AETAnswer.
79          /// </summary>
80          private int _aetQID;
81          [XmlAttribute("aetQID")]
82          public int AETQID
83          {
84              get { return this._aetQID; }
85              set { this._aetQID = value; }
86          }
87
88          /// <summary>
89          /// The AETType of the associated AETQuestion.
90          /// </summary>
91          private AETType _aetType;
92          [XmlAttribute("aetType")]
93          public AETType AETType
94          {
95              get { return this._aetType; }
96              set { this._aetType = value; }
97          }
98
99          /// <summary>
100         /// The ClassID associated class.
101         /// </summary>
102         private string _classID;
103         [XmlAttribute("classID")]
104         public string ClassID
105         {
106             get { return this._classID; }
107             set { this._classID = value; }
108         }
109
110         /// <summary>
111         /// The date that the answer was submitted.
```

```
112         /// </summary>
113         private DateTime _submittedDateTime;
114         [XmlAttribute("submittedDateTime")]
115         public DateTime SubmittedDateTime
116         {
117             get { return this._submittedDateTime; }
118             set { this._submittedDateTime = value; }
119         }
120 
121         /// <summary>
122         /// Whether or not the AETAnswer has been completely marked.
123         /// </summary>
124         private bool _marked;
125         [XmlAttribute("marked")]
126         public bool Marked
127         {
128             get { return this._marked; }
129             set { this._marked = value; }
130         }
131         #endregion
132 
133         #region Static Data Retrieving Methods
134         /// <summary>
135         /// Finds the properties of an AETAnswer (without the
                answers).
136         /// </summary>
137         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
138         /// <param name="classID">The associated Class ID.</param>
139         /// <param name="userID">The owner of the AETAnswer.</param>
140         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
141         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
142         /// <exception cref="System.Exception">Thrown when an
                AETAnswer isn't found.</exception>
143         /// <returns>Returns the found structure ShortAETAnswer.</
                returns>
144         public static ShortAETAnswer FindShortAETAnswer(int aetQID,
                string classID, int userID, AETType aetType, DataBridge
                dataBridge)
145         {
146             SqlCommand sqlCommand = new SqlCommand("
                    FindShortAETAnswer", dataBridge.SqlConnection);
147             sqlCommand.CommandType = CommandType.StoredProcedure;
148             sqlCommand.Parameters.Add("@aetQID", SqlDbType.Int);
149             sqlCommand.Parameters["@aetQID"].Value = aetQID;
150             sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
151             sqlCommand.Parameters["@classID"].Value = classID;
152             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
153             sqlCommand.Parameters["@userID"].Value = userID;
154             sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                    , AETLimitations.AETType);
155             sqlCommand.Parameters["@aetQType"].Value = aetType;
```

```
156
157             dataBridge.SqlConnection.Open();
158                 // Read the sql server's xml data
159             XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
                    ExecuteXmlReader();
160
161                 // Retrieve the information and deserialize it
162             ShortAETAnswer shortAETAnswer = new ShortAETAnswer();
163             shortAETAnswer.AETQID = 0;
164             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    ShortAETAnswer));
165             while (xmlTextReader.Read())
166                 if (xmlSerializer.CanDeserialize(xmlTextReader))
167                     shortAETAnswer = (ShortAETAnswer)xmlSerializer.
                            Deserialize(xmlTextReader);
168
169             xmlTextReader.Close();
170             dataBridge.SqlConnection.Close();
171
172             if (shortAETAnswer.AETQID == 0)
173                 throw new Exception("ShortAETAnswer not found!");
174
175             return shortAETAnswer;
176         }
177
178         /// <summary>
179         /// Finds the properties of AETAnswers for a particular
                Class ID (without the answers).
180         /// </summary>
181         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
182         /// <param name="classID">The associated Class ID.</param>
183         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
184         /// <exception cref="System.Exception">Thrown when
                AETAnswers aren't found.</exception>
185         /// <returns>Returns an array of ShortAETAnswers that were
                found.</returns>
186         public static ShortAETAnswer[] FindShortAETAnswers(int
                aetQID, string classID, AETType aetType, DataBridge
                dataBridge)
187         {
188             SqlCommand sqlCommand = new SqlCommand("
                    FindShortAETAnswers", dataBridge.SqlConnection);
189             sqlCommand.CommandType = CommandType.StoredProcedure;
190             sqlCommand.Parameters.Add("@aetQID", SqlDbType.Int);
191             sqlCommand.Parameters["@aetQID"].Value = aetQID;
192             sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
193             sqlCommand.Parameters["@classID"].Value = classID;
194             sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                    , AETLimitations.AETType);
195             sqlCommand.Parameters["@aetQType"].Value = aetType;
196
197             dataBridge.SqlConnection.Open();
```

```
                // Read the sql server's xml data
            XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();

                // Retrieve the information and deserialize it
            ArrayList shortAETAnswers = new ArrayList();
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(ShortAETAnswer));
            while (xmlTextReader.Read())
                if (xmlSerializer.CanDeserialize(xmlTextReader))
                    shortAETAnswers.Add(xmlSerializer.Deserialize(xmlTextReader));

            xmlTextReader.Close();
            dataBridge.SqlConnection.Close();

            if (shortAETAnswers.Count == 0)
                throw new Exception("ShortAETAnswers not found!");

            return (ShortAETAnswer[])shortAETAnswers.ToArray(typeof(ShortAETAnswer));
        }
        #endregion
    }

    /// <summary>
    /// An abstract class which provides shared properties and methods for AETAnswers.
    /// </summary>
    public abstract class AETAnswer
    {
        #region Properties
        /// <summary>
        /// The UserID associated with the AETAnswer.
        /// </summary>
        private int _userID;
        [XmlAttribute("userID")]
        public int UserID
        {
            get { return this._userID; }
            set { this._userID = value; }
        }

        /// <summary>
        /// The ID of the AETQuestion associated with the AETAnswer.
        /// </summary>
        private int _aetQID;
        [XmlAttribute("aetQID")]
        public int AETQID
        {
            get { return this._aetQID; }
            set { this._aetQID = value; }
        }

        /// <summary>
```

```
248         /// The AETType of the associated AETQuestion.
249         /// </summary>
250         private AETType _aetType;
251         [XmlAttribute("aetType")]
252         public AETType AETType
253         {
254             get { return this._aetType; }
255             set { this._aetType = value; }
256         }
257 
258         /// <summary>
259         /// The ClassID associated class.
260         /// </summary>
261         private string _classID;
262         [XmlAttribute("classID")]
263         public string ClassID
264         {
265             get { return this._classID; }
266             set { this._classID = value; }
267         }
268 
269         /// <summary>
270         /// The date that the answer was submitted.
271         /// </summary>
272         private DateTime _submittedDateTime;
273         [XmlAttribute("submittedDateTime")]
274         public DateTime SubmittedDateTime
275         {
276             get { return this._submittedDateTime; }
277             set { this._submittedDateTime = value; }
278         }
279 
280         /// <summary>
281         /// Whether or not the AETAnswer has been completely marked.
282         /// </summary>
283         private bool _marked;
284         [XmlAttribute("marked")]
285         public bool Marked
286         {
287             get { return this._marked; }
288             set { this._marked = value; }
289         }
290         #endregion
291 
292         #region Constructors
293         /// <summary>
294         /// Used for XmlSerializer purposes.
295         /// </summary>
296         public AETAnswer() {}
297 
298         /// <summary>
299         /// Abstract constructor.
300         /// </summary>
301         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
```

```
302         public AETAnswer(DataBridge dataBridge)
303         {
304             this.DataBridge = dataBridge;
305         }
306         #endregion
307
308         #region Data Management Properties
309         /// <summary>
310         /// The DataBridge that connects to the data.
311         /// </summary>
312         private DataBridge _dataBridge = null;
313         [XmlIgnore()]
314         public DataBridge DataBridge
315         {
316             get { return this._dataBridge; }
317             set { this._dataBridge = value; }
318         }
319         #endregion
320
321         #region Data Management Methods
322         /// <summary>
323         /// Uses current instance's properties and stores the data.
324         /// </summary>
325         public void Create()
326         {
327             string aetAFilePath = null;
328             string paddedUserID = null;
329
330                 // Get the filepath
331             switch (this.AETType)
332             {
333                 case AETType.Assignment:
334                     aetAFilePath = this.DataBridge.
                            GetClassAssignmentAnswerFilePath(this.ClassID
                            , this.AETQID, this.UserID, out paddedUserID,
                             false);
335                     break;
336                 case AETType.Exam:
337                     aetAFilePath = this.DataBridge.
                            GetClassExamAnswerFilePath(this.ClassID, this
                            .AETQID, this.UserID, out paddedUserID, false
                            );
338                     break;
339                 case AETType.Tutorial:
340                     aetAFilePath = this.DataBridge.
                            GetClassTutorialAnswerFilePath(this.ClassID,
                            this.AETQID, this.UserID, out paddedUserID,
                            false);
341                     break;
342             }
343
344                 // Write information to sql server
345             SqlCommand sqlCommand = new SqlCommand("CreateAETAnswer"
                    , this.DataBridge.SqlConnection);
346             sqlCommand.CommandType = CommandType.StoredProcedure;
```

```
347                 sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
348                 sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
349                 sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                        ClassLimitations.ClassID);
350                 sqlCommand.Parameters["@classID"].Value = this.ClassID;
351                 sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                        , AETLimitations.AETType);
352                 sqlCommand.Parameters["@aetQType"].Value = this.AETType.
                        ToString();
353                 sqlCommand.Parameters.Add("@userID", SqlDbType.BigInt);
354                 sqlCommand.Parameters["@userID"].Value = this.UserID;
355                 sqlCommand.Parameters.Add("@aetASubmittedDate",
                        SqlDbType.DateTime);
356                 sqlCommand.Parameters["@aetASubmittedDate"].Value = this
                        .SubmittedDateTime;
357                 sqlCommand.Parameters.Add("@aetAMarked", SqlDbType.Bit);
358                 sqlCommand.Parameters["@aetAMarked"].Value = this.Marked
                        ;
359                 this.DataBridge.SqlConnection.Open();
360                 sqlCommand.ExecuteNonQuery();
361                 this.DataBridge.SqlConnection.Close();
362
363                     // Write information to xml file
364                 XmlSerializer xmlSerializer = new XmlSerializer(this.
                        GetType());
365                 XmlTextWriter xmlTextWriter;
366                 xmlSerializer.Serialize(xmlTextWriter = new
                        XmlTextWriter(aetAFilePath, Encoding.UTF8), this);
367                 xmlTextWriter.Close();
368             }
369
370             /// <summary>
371             /// Uses current instance's properties and re-stores the
                    data.
372             /// </summary>
373             public void Alter()
374             {
375                 string aetAFilePath = null;
376                 string paddedUserID = null;
377
378                     // Get the filepath
379                 switch (this.AETType)
380                 {
381                     case AETType.Assignment:
382                         aetAFilePath = this.DataBridge.
                                GetClassAssignmentAnswerFilePath(this.ClassID
                                , this.AETQID, this.UserID, out paddedUserID,
                                 true);
383                         break;
384                     case AETType.Exam:
385                         aetAFilePath = this.DataBridge.
                                GetClassExamAnswerFilePath(this.ClassID, this
                                .AETQID, this.UserID, out paddedUserID, true)
                                ;
386                         break;
```

```
387                     case AETType.Tutorial:
388                         aetAFilePath = this.DataBridge.
                                GetClassTutorialAnswerFilePath(this.ClassID,
                                this.AETQID, this.UserID, out paddedUserID,
                                true);
389                         break;
390                 }
391
392                     // Write information to sql server
393                 SqlCommand sqlCommand = new SqlCommand("AlterAETAnswer",
                        this.DataBridge.SqlConnection);
394                 sqlCommand.CommandType = CommandType.StoredProcedure;
395                 sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
396                 sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
397                 sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                        ClassLimitations.ClassID);
398                 sqlCommand.Parameters["@classID"].Value = this.ClassID;
399                 sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                        , AETLimitations.AETType);
400                 sqlCommand.Parameters["@aetQType"].Value = this.AETType.
                        ToString();
401                 sqlCommand.Parameters.Add("@userID", SqlDbType.BigInt);
402                 sqlCommand.Parameters["@userID"].Value = this.UserID;
403                 sqlCommand.Parameters.Add("@aetASubmittedDate",
                        SqlDbType.DateTime);
404                 sqlCommand.Parameters["@aetASubmittedDate"].Value = this
                        .SubmittedDateTime;
405                 sqlCommand.Parameters.Add("@aetAMarked", SqlDbType.Bit);
406                 sqlCommand.Parameters["@aetAMarked"].Value = this.Marked
                        ;
407                 this.DataBridge.SqlConnection.Open();
408                 sqlCommand.ExecuteNonQuery();
409                 this.DataBridge.SqlConnection.Close();
410
411                     // Write information to xml file
412                 XmlSerializer xmlSerializer = new XmlSerializer(this.
                        GetType());
413                 XmlTextWriter xmlTextWriter;
414                 xmlSerializer.Serialize(xmlTextWriter = new
                        XmlTextWriter(aetAFilePath, Encoding.UTF8), this);
415                 xmlTextWriter.Close();
416             }
417
418             /// <summary>
419             /// Uses the current instance's properties and deletes the
                    stored data.
420             /// </summary>
421             public void Delete()
422             {
423                 string aetAFilePath = null;
424                 string paddedUserID = null;
425
426                     // Get the filepath
427                 switch (this.AETType)
428                 {
```

```
429                     case AETType.Assignment:
430                         aetAFilePath = this.DataBridge.
                                GetClassAssignmentAnswerFilePath(this.ClassID
                                , this.AETQID, this.UserID, out paddedUserID,
                                 true);
431                         break;
432                     case AETType.Exam:
433                         aetAFilePath = this.DataBridge.
                                GetClassExamAnswerFilePath(this.ClassID, this
                                .AETQID, this.UserID, out paddedUserID, true)
                                ;
434                         break;
435                     case AETType.Tutorial:
436                         aetAFilePath = this.DataBridge.
                                GetClassTutorialAnswerFilePath(this.ClassID,
                                this.AETQID, this.UserID, out paddedUserID,
                                true);
437                         break;
438                 }
439
440                     // Write information to sql server
441                 SqlCommand sqlCommand = new SqlCommand("DeleteAETAnswer"
                        , this.DataBridge.SqlConnection);
442                 sqlCommand.CommandType = CommandType.StoredProcedure;
443                 sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
444                 sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
445                 sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                        ClassLimitations.ClassID);
446                 sqlCommand.Parameters["@classID"].Value = this.ClassID;
447                 sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                        , AETLimitations.AETType);
448                 sqlCommand.Parameters["@aetQType"].Value = this.AETType.
                        ToString();
449                 sqlCommand.Parameters.Add("@userID", SqlDbType.BigInt);
450                 sqlCommand.Parameters["@userID"].Value = this.UserID;
451                 this.DataBridge.SqlConnection.Open();
452                 sqlCommand.ExecuteNonQuery();
453                 this.DataBridge.SqlConnection.Close();
454
455                     // Delete xml file
456                 File.Delete(aetAFilePath);
457             }
458             #endregion
459         }
460
461         /// <summary>
462         /// An inherited class of AETAnswer for Information answers.
463         /// </summary>
464         [XmlRoot("AETAnswer")]
465         public sealed class AETInformationAnswer : AETAnswer
466         {
467             #region Properties
468             /// <summary>
469             /// The filename of where the answer exists.
470             /// </summary>
```

```
471         private string _aetAFilename;
472         [XmlElement("AnswerFilename")]
473         public string AETAFilename
474         {
475             get { return this._aetAFilename; }
476             set { this._aetAFilename = value; }
477         }
478         #endregion
479
480         #region Constructors
481         /// <summary>
482         /// Used for XmlSerializer purposes.
483         /// </summary>
484         public AETInformationAnswer() : base(null) {}
485
486         /// <summary>
487         /// Main constructor.
488         /// </summary>
489         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
490         public AETInformationAnswer(DataBridge dataBridge) : base(
                dataBridge) {}
491         #endregion
492
493         #region Static Data Retrieving Methods
494         /// <summary>
495         /// Finds an AETInformationAnswer.
496         /// </summary>
497         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
498         /// <param name="classID">The associated Class ID.</param>
499         /// <param name="userID">The owner of the AETAnswer.</param>
500         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
501         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
502         /// <returns>Returns the found class AETInformationAnswer.
                Null if not found.</returns>
503         public static AETInformationAnswer FindAETInformationAnswer(
                int aetQID, string classID, int userID, AETType aetType,
                DataBridge dataBridge)
504         {
505             string aetAFilePath = null;
506             string paddedUserID = null;
507
508             try
509             {
510                     // Get the filepath
511                 switch (aetType)
512                 {
513                     case AETType.Assignment:
514                         aetAFilePath = dataBridge.
                                GetClassAssignmentAnswerFilePath(classID,
                                 aetQID, userID, out paddedUserID, true);
515                         break;
```

```
                        case AETType.Exam:
                            aetAFilePath = dataBridge.
                                GetClassExamAnswerFilePath(classID,
                                aetQID, userID, out paddedUserID, true);
                            break;
                        case AETType.Tutorial:
                            aetAFilePath = dataBridge.
                                GetClassTutorialAnswerFilePath(classID,
                                aetQID, userID, out paddedUserID, true);
                            break;
                    }
                }
                catch
                {
                        // Not found
                    return null;
                }

                    // Retrieve the information from the xml file
                XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    AETInformationAnswer));
                XmlTextReader xmlTextReader;
                AETInformationAnswer newAETA = xmlSerializer.Deserialize
                    (xmlTextReader = new XmlTextReader(aetAFilePath)) as
                    AETInformationAnswer;
                xmlTextReader.Close();

                return newAETA;
            }
            #endregion
        }

        /// <summary>
        /// An inherited class of AETAnswer for Multiple Choice answers.
        /// </summary>
        [XmlRoot("AETAnswer")]
        public sealed class AETMultipleChoiceAnswer : AETAnswer
        {
            #region Properties
            /// <summary>
            /// An array of answers.
            /// </summary>
            [XmlArray("Answers")]
            [XmlArrayItem("Answer")]
            public MultipleChoiceAnswer[] answers;
            #endregion

            #region Constructors
            /// <summary>
            /// Used for XmlSerializer purposes.
            /// </summary>
            public AETMultipleChoiceAnswer() : base(null) {}

            /// <summary>
            /// Main constructor.
```

```
564         /// </summary>
565         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
566         public AETMultipleChoiceAnswer(DataBridge dataBridge) : base
                (dataBridge) {}
567         #endregion
568
569         #region Static Data Retrieving Methods
570         /// <summary>
571         /// Finds an AETMultipleChoiceAnswer.
572         /// </summary>
573         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
574         /// <param name="classID">The associated Class ID.</param>
575         /// <param name="userID">The owner of the AETAnswer.</param>
576         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
577         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
578         /// <returns>Returns the found class AETMultipleChoiceAnswer
                . Null if not found.</returns>
579         public static AETMultipleChoiceAnswer
                FindAETMultipleChoiceAnswer(int aetQID, string classID,
                int userID, AETType aetType, DataBridge dataBridge)
580         {
581             string aetAFilePath = null;
582             string paddedUserID = null;
583
584             try
585             {
586                     // Get the filepath
587                 switch (aetType)
588                 {
589                     case AETType.Assignment:
590                         aetAFilePath = dataBridge.
                                GetClassAssignmentAnswerFilePath(classID,
                                 aetQID, userID, out paddedUserID, true);
591                         break;
592                     case AETType.Exam:
593                         aetAFilePath = dataBridge.
                                GetClassExamAnswerFilePath(classID,
                                aetQID, userID, out paddedUserID, true);
594                         break;
595                     case AETType.Tutorial:
596                         aetAFilePath = dataBridge.
                                GetClassTutorialAnswerFilePath(classID,
                                aetQID, userID, out paddedUserID, true);
597                         break;
598                 }
599             }
600             catch
601             {
602                     // Not found
603                 return null;
604             }
```

```
605
606                 // Retrieve the information from the xml file
607             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    AETMultipleChoiceAnswer));
608             XmlTextReader xmlTextReader;
609             AETMultipleChoiceAnswer newAETA = xmlSerializer.
                    Deserialize(xmlTextReader = new XmlTextReader(
                    aetAFilePath)) as AETMultipleChoiceAnswer;
610             xmlTextReader.Close();
611
612             return newAETA;
613         }
614         #endregion
615     }
616
617     /// <summary>
618     /// An inherited class of AETAnswer for Short Response answers.
619     /// </summary>
620     [XmlRoot("AETAnswer")]
621     public sealed class AETShortResponseAnswer : AETAnswer
622     {
623         #region Properties
624         /// <summary>
625         /// An array of answers.
626         /// </summary>
627         [XmlArray("Answers")]
628         [XmlArrayItem("Answer")]
629         public ShortResponseAnswer[] answers;
630         #endregion
631
632         #region Constructors
633         /// <summary>
634         /// Used for XmlSerializer purposes.
635         /// </summary>
636         public AETShortResponseAnswer() : base(null) {}
637
638         /// <summary>
639         /// Main constructor.
640         /// </summary>
641         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
642         public AETShortResponseAnswer(DataBridge dataBridge) : base(
                dataBridge) {}
643         #endregion
644
645         #region Static Data Retrieving Methods
646         /// <summary>
647         /// Finds an AETShortResponseAnswer.
648         /// </summary>
649         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
650         /// <param name="classID">The associated Class ID.</param>
651         /// <param name="userID">The owner of the AETAnswer.</param>
652         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
```

```
        /// <param name="dataBridge">The DataBridge that connects to the data.</param>
        /// <returns>Returns the found class AETShortResponseAnswer. Null if not found.</returns>
        public static AETShortResponseAnswer FindAETShortResponseAnswer(int aetQID, string classID, int userID, AETType aetType, DataBridge dataBridge)
        {
            string aetAFilePath = null;
            string paddedUserID = null;

            try
            {
                    // Get the filepath
                switch (aetType)
                {
                    case AETType.Assignment:
                        aetAFilePath = dataBridge.GetClassAssignmentAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
                        break;
                    case AETType.Exam:
                        aetAFilePath = dataBridge.GetClassExamAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
                        break;
                    case AETType.Tutorial:
                        aetAFilePath = dataBridge.GetClassTutorialAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
                        break;
                }
            }
            catch
            {
                return null;
            }

                // Retrieve the information from the xml file
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(AETShortResponseAnswer));
            XmlTextReader xmlTextReader;
            AETShortResponseAnswer newAETA = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(aetAFilePath)) as AETShortResponseAnswer;
            xmlTextReader.Close();

            return newAETA;
        }
        #endregion
    }
}
```

Listing H.95: AETQuestion.cs

```
using System;
```

```
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    #region Enums
    /// <summary>
    /// The type of the AETQuestion.
    /// </summary>
    public enum AETType
    {
        /// <summary>
        /// An assignment.
        /// </summary>
        Assignment,
        /// <summary>
        /// An exam.
        /// </summary>
        Exam,
        /// <summary>
        /// A tutorial.
        /// </summary>
        Tutorial,
        /// <summary>
        /// Enum hasn't be set.
        /// </summary>
        NotSet
    }
    /// <summary>
    /// The AETQuestion's question type.
    /// </summary>
    public enum QuestionType
    {
        /// <summary>
        /// Information on the assessment questions.
        /// </summary>
        Information,
        /// <summary>
        /// Short response assessment questions.
        /// </summary>
        ShortResponse,
        /// <summary>
        /// Multiple choice assessment questions.
        /// </summary>
        MultipleChoice
    }
    #endregion

    #region Structures
    /// <summary>
```

```
57      /// Properties that belong to an InformationQuestion.
58      /// </summary>
59      public struct InformationQuestion
60      {
61          /// <summary>
62          /// The question's unique ID.
63          /// </summary>
64          [XmlAttribute("questionID")]
65          public int questionID;
66          /// <summary>
67          /// The question's title.
68          /// </summary>
69          [XmlElement("Title")]
70          public string title;
71          /// <summary>
72          /// The question's question.
73          /// </summary>
74          [XmlElement("Question")]
75          public string question;
76      }
77      /// <summary>
78      /// Properties that belong to an ShortResponseQuestion.
79      /// </summary>
80      public struct ShortResponseQuestion
81      {
82          /// <summary>
83          /// The question's unique ID.
84          /// </summary>
85          [XmlAttribute("questionID")]
86          public int questionID;
87          /// <summary>
88          /// The question's title.
89          /// </summary>
90          [XmlElement("Title")]
91          public string title;
92          /// <summary>
93          /// The question's question.
94          /// </summary>
95          [XmlElement("Question")]
96          public string question;
97      }
98      /// <summary>
99      /// Properties that belong to an MultipleChoiceQuestion.
100     /// </summary>
101     public struct MultipleChoiceQuestion
102     {
103         /// <summary>
104         /// The question's unique ID.
105         /// </summary>
106         [XmlAttribute("questionID")]
107         public int questionID;
108         /// <summary>
109         /// The question's title.
110         /// </summary>
111         [XmlElement("Title")]
```

```
112         public string title;
113         /// <summary>
114         /// The question's question.
115         /// </summary>
116         [XmlElement("Question")]
117         public string question;
118         /// <summary>
119         /// True if the answer is correct, false if otherwise.
120         /// </summary>
121         [XmlAttribute("correctChoice")]
122         public int correctChoice;
123         /// <summary>
124         /// An array of chioces to accompany the question.
125         /// </summary>
126         [XmlArray("Choices")]
127         [XmlArrayItem("Choice")]
128         public MultipleChoiceChoice[] choices;
129     }
130     /// <summary>
131     /// Properties that belong to an MultipleChoiceQuestion's choice
        .
132     /// </summary>
133     public struct MultipleChoiceChoice
134     {
135         /// <summary>
136         /// The choice's unique ID.
137         /// </summary>
138         [XmlAttribute("choiceID")]
139         public int choiceID;
140         /// <summary>
141         /// The choice's text.
142         /// </summary>
143         [XmlText()]
144         public string choiceText;
145     }
146     #endregion
147
148
149     /// <summary>
150     /// A structure similar to AETQuestion, but without the question
         properties.
151     /// </summary>
152     [XmlRoot("AETQuestion")]
153     public struct ShortAETQuestion
154     {
155         #region Properties
156         /// <summary>
157         /// The unique ID.
158         /// </summary>
159         private int _aetQID;
160         [XmlAttribute("aetQID")]
161         public int AETQID
162         {
163             get { return this._aetQID; }
164             set { this._aetQID = value; }
```

```
165         }
166
167         /// <summary>
168         /// The associated class ID.
169         /// </summary>
170         private string _classID;
171         [XmlAttribute("classID")]
172         public string ClassID
173         {
174             get { return this._classID; }
175             set { this._classID = value; }
176         }
177
178         /// <summary>
179         /// The associated title.
180         /// </summary>
181         private string _title;
182         [XmlElement("Title")]
183         public string Title
184         {
185             get { return this._title; }
186             set { this._title = value; }
187         }
188
189         /// <summary>
190         /// The associated description (ususally 'Assignment', 'Exam
            ', or 'Tutorial').
191         /// </summary>
192         private string _description;
193         [XmlElement("Description")]
194         public string Description
195         {
196             get { return this._description; }
197             set { this._description = value; }
198         }
199
200         /// <summary>
201         /// The associated type.
202         /// </summary>
203         private AETType _aetType;
204         [XmlAttribute("aetType")]
205         public AETType AETType
206         {
207             get { return this._aetType; }
208             set { this._aetType = value; }
209         }
210
211         /// <summary>
212         /// The associated question type.
213         /// </summary>
214         private QuestionType _questionType;
215         [XmlAttribute("questionType")]
216         public QuestionType QuestionType
217         {
218             get { return this._questionType; }
```

```
            set { this._questionType = value; }
        }

        /// <summary>
        /// The date & time students can view the question.
        /// </summary>
        private DateTime _viewQuestionDateTime;
        [XmlAttribute("viewQuestionDateTime")]
        public DateTime ViewQuestionDateTime
        {
            get { return this._viewQuestionDateTime; }
            set { this._viewQuestionDateTime = value; }
        }

        /// <summary>
        /// The date & time the answers are due.
        /// </summary>
        private DateTime _dueDateTime;
        [XmlAttribute("dueDateTime")]
        public DateTime DueDateTime
        {
            get { return this._dueDateTime; }
            set { this._dueDateTime = value; }
        }

        /// <summary>
        /// The date & time no more answers are accepted.
        /// </summary>
        private DateTime _haltSubmissionsDateTime;
        [XmlAttribute("haltSubmissionsDateTime")]
        public DateTime HaltSubmissionsDateTime
        {
            get { return this._haltSubmissionsDateTime; }
            set { this._haltSubmissionsDateTime = value; }
        }

        /// <summary>
        /// The date & time students are able to see their marked
            answers.
        /// </summary>
        private DateTime _viewMarkedAnswersDateTime;
        [XmlAttribute("viewMarkedAnswersDateTime")]
        public DateTime ViewMarkedAnswersDateTime
        {
            get { return this._viewMarkedAnswersDateTime; }
            set { this._viewMarkedAnswersDateTime = value; }
        }

        /// <summary>
        /// The amount of questions.
        /// </summary>
        private int _amountOfQuestions;
        [XmlAttribute("amountOfQuestions")]
        public int AmountOfQuestions
        {
```

```
            get { return this._amountOfQuestions; }
            set { this._amountOfQuestions = value; }
        }
        #endregion

        #region Static Data Retrieving Methods
        /// <summary>
        /// Finds a specific ShortAETQuestion.
        /// </summary>
        /// <param name="aetQID">The associated AETQuestion ID.</
            param>
        /// <param name="classID">The associated Class ID.</param>
        /// <param name="aetType">The type of the AETAnswer (
            Assignment, Exam, or Tutorial).</param>
        /// <param name="dataBridge">The DataBridge that connects to
             the data.</param>
        /// <returns>Returns the found ShortAETQuestion. Null if not
             found.</returns>
        public static ShortAETQuestion FindShortAETQuestion(int
            aetQID, string classID, AETType aetType, DataBridge
            dataBridge)
        {
            SqlCommand sqlCommand = new SqlCommand("
                FindShortAETQuestion", dataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
            sqlCommand.Parameters["@aetQID"].Value = aetQID;
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                ClassLimitations.ClassID);
            sqlCommand.Parameters["@classID"].Value = classID;
            sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                , AETLimitations.AETType);
            sqlCommand.Parameters["@aetQType"].Value = aetType;

            dataBridge.SqlConnection.Open();
            XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
                ExecuteXmlReader();

            ShortAETQuestion shortAETQuestion = new ShortAETQuestion
                ();
            shortAETQuestion.AETQID = 0;
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                ShortAETQuestion));
            while (xmlTextReader.Read())
                if (xmlSerializer.CanDeserialize(xmlTextReader))
                    shortAETQuestion = (ShortAETQuestion)
                        xmlSerializer.Deserialize(xmlTextReader);

            xmlTextReader.Close();
            dataBridge.SqlConnection.Close();

            if (shortAETQuestion.AETQID == 0)
                throw new Exception("ShortAETQuestion not found!");

            return shortAETQuestion;
```

```
315         }
316
317         /// <summary>
318         /// Finds the properties of AETQuestions for a particular
                Class ID (without the questions).
319         /// </summary>
320         /// <param name="classID">The associated class ID.</param>
321         /// <param name="aetType">The associated AETType.</param>
322         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
323         /// <returns>Returns an array of ShortAETQuestions that were
                 found.</returns>
324         public static ShortAETQuestion[] FindShortAETQuestions(
                string classID, AETType aetType, DataBridge dataBridge)
325         {
326             SqlCommand sqlCommand = new SqlCommand("
                    FindShortAETQuestions", dataBridge.SqlConnection);
327             sqlCommand.CommandType = CommandType.StoredProcedure;
328             sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
329             sqlCommand.Parameters["@classID"].Value = classID;
330             sqlCommand.Parameters.Add("@aetQType", SqlDbType.Char,
                    AETLimitations.AETType);
331             sqlCommand.Parameters["@aetQType"].Value = aetType.
                    ToString();
332
333             dataBridge.SqlConnection.Open();
334             XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
                    ExecuteXmlReader();
335
336                 // Deserialise questions
337             ArrayList shortAETQuestions = new ArrayList();
338             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    ShortAETQuestion));
339             while (xmlTextReader.Read())
340             {
341                 Console.WriteLine("*");
342                 while (xmlSerializer.CanDeserialize(xmlTextReader))
343                     shortAETQuestions.Add(xmlSerializer.Deserialize(
                            xmlTextReader));
344             }
345
346             xmlTextReader.Close();
347             dataBridge.SqlConnection.Close();
348
349             if (shortAETQuestions.Count == 0)
350                 throw new Exception("ShortAETQuestions not found!");
351
352             return (ShortAETQuestion[])shortAETQuestions.ToArray(
                    typeof(ShortAETQuestion));
353         }
354         #endregion
355     }
356
357     /// <summary>
```

```
358     /// An abstract class which provides shared properties and
            methods for AETQuestions.
359     /// </summary>
360     public abstract class AETQuestion
361     {
362         #region Properties
363         /// <summary>
364         /// The unique ID.
365         /// </summary>
366         private int _aetQID;
367         [XmlAttribute("aetQID")]
368         public int AETQID
369         {
370             get { return this._aetQID; }
371             set { this._aetQID = value; }
372         }
373 
374         /// <summary>
375         /// The associated class ID.
376         /// </summary>
377         private string _classID;
378         [XmlAttribute("classID")]
379         public string ClassID
380         {
381             get { return this._classID; }
382             set { this._classID = value; }
383         }
384 
385         /// <summary>
386         /// The associated title.
387         /// </summary>
388         private string _title;
389         [XmlElement("Title")]
390         public string Title
391         {
392             get { return this._title; }
393             set { this._title = value; }
394         }
395 
396         /// <summary>
397         /// The associated description (ususally 'Assignment', 'Exam
                ', or 'Tutorial').
398         /// </summary>
399         private string _description;
400         [XmlElement("Description")]
401         public string Description
402         {
403             get { return this._description; }
404             set { this._description = value; }
405         }
406 
407         /// <summary>
408         /// The associated type.
409         /// </summary>
410         private AETType _aetType;
```

```
411         [XmlAttribute("aetType")]
412         public AETType AETType
413         {
414             get { return this._aetType; }
415             set { this._aetType = value; }
416         }
417 
418         /// <summary>
419         /// The associated question type.
420         /// </summary>
421         private QuestionType _questionType;
422         [XmlAttribute("questionType")]
423         public QuestionType QuestionType
424         {
425             get { return this._questionType; }
426             set { this._questionType = value; }
427         }
428 
429         /// <summary>
430         /// The date & time students can view the question.
431         /// </summary>
432         private DateTime _viewQuestionDateTime;
433         [XmlAttribute("viewQuestionDateTime")]
434         public DateTime ViewQuestionDateTime
435         {
436             get { return this._viewQuestionDateTime; }
437             set { this._viewQuestionDateTime = value; }
438         }
439 
440         /// <summary>
441         /// The date & time the answers are due.
442         /// </summary>
443         private DateTime _dueDateTime;
444         [XmlAttribute("dueDateTime")]
445         public DateTime DueDateTime
446         {
447             get { return this._dueDateTime; }
448             set { this._dueDateTime = value; }
449         }
450 
451         /// <summary>
452         /// The date & time no more answers are accepted.
453         /// </summary>
454         private DateTime _haltSubmissionsDateTime;
455         [XmlAttribute("haltSubmissionsDateTime")]
456         public DateTime HaltSubmissionsDateTime
457         {
458             get { return this._haltSubmissionsDateTime; }
459             set { this._haltSubmissionsDateTime = value; }
460         }
461 
462         /// <summary>
463         /// The date & time students are able to see their marked
                answers.
464         /// </summary>
```

```
465         private DateTime _viewMarkedAnswersDateTime;
466         [XmlAttribute("viewMarkedAnswersDateTime")]
467         public DateTime ViewMarkedAnswersDateTime
468         {
469             get { return this._viewMarkedAnswersDateTime; }
470             set { this._viewMarkedAnswersDateTime = value; }
471         }
472 
473         /// <summary>
474         /// The amount of questions.
475         /// </summary>
476         private int _amountOfQuestions;
477         [XmlAttribute("amountOfQuestions")]
478         public int AmountOfQuestions
479         {
480             get { return this._amountOfQuestions; }
481             set { this._amountOfQuestions = value; }
482         }
483         #endregion
484 
485         #region Constructors
486         /// <summary>
487         /// Abstract constructor.
488         /// </summary>
489         /// <param name="questionType">The associated question type
            .</param>
490         public AETQuestion(QuestionType questionType)
491         {
492             this._questionType = questionType;
493         }
494         /// <summary>
495         /// Abstract constructor.
496         /// </summary>
497         /// <param name="aetType">The associated AET type.</param>
498         /// <param name="questionType">The associated question type
            .</param>
499         /// <param name="dataBridge">The DataBridge that connects to
             the data.</param>
500         public AETQuestion(AETType aetType, QuestionType
            questionType, DataBridge dataBridge)
501         {
502             this._aetType = aetType;
503             this._questionType = questionType;
504             this._dataBridge = dataBridge;
505         }
506         #endregion
507 
508         #region Data Management Properties
509         /// <summary>
510         /// The DataBridge that connects to the data.
511         /// </summary>
512         private DataBridge _dataBridge = null;
513         [XmlIgnore()]
514         public DataBridge DataBridge
515         {
```

```
            get { return this._dataBridge; }
            set { this._dataBridge = value; }
        }
        #endregion

        #region Data Management Methods
        /// <summary>
        /// Uses current instance's properties and stores the data.
        /// </summary>
        public int Create()
        {
            string aetQFilePath = null;
            string aetAnswerDirectory = null;
            int numericAETQID = 0;
            string paddedAETQID = null;

            switch (this.AETType)
            {
                case AETType.Assignment:
                    aetQFilePath = this.DataBridge.
                        GetNextClassAssignmentFilePath(this.ClassID,
                        out numericAETQID, out paddedAETQID);
                    aetAnswerDirectory = this.DataBridge.
                        GetClassAssignmentAnswersDirectory(this.
                        ClassID, numericAETQID, false);
                    break;
                case AETType.Exam:
                    aetQFilePath = this.DataBridge.
                        GetNextClassExamFilePath(this.ClassID, out
                        numericAETQID, out paddedAETQID);
                    aetAnswerDirectory = this.DataBridge.
                        GetClassExamAnswersDirectory(this.ClassID,
                        numericAETQID, false);
                    break;
                case AETType.Tutorial:
                    aetQFilePath = this.DataBridge.
                        GetNextClassTutorialFilePath(this.ClassID,
                        out numericAETQID, out paddedAETQID);
                    aetAnswerDirectory = this.DataBridge.
                        GetClassTutorialAnswersDirectory(this.ClassID
                        , numericAETQID, false);
                    break;
            }

            this.AETQID = numericAETQID;

                // Write information to sql server
            SqlCommand sqlCommand = new SqlCommand("
                CreateAETQuestion", this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
            sqlCommand.Parameters["@aetQID"].Value = numericAETQID;
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                ClassLimitations.ClassID);
            sqlCommand.Parameters["@classID"].Value = this.ClassID;
```

```
557                 sqlCommand.Parameters.Add("@aetQTitle", SqlDbType.
                        VarChar, AETLimitations.Title);
558                 sqlCommand.Parameters["@aetQTitle"].Value = this.Title;
559                 sqlCommand.Parameters.Add("@aetQDescription", SqlDbType.
                        VarChar, AETLimitations.Description);
560                 sqlCommand.Parameters["@aetQDescription"].Value = this.
                        Description;
561                 sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar
                        , AETLimitations.AETType);
562                 sqlCommand.Parameters["@aetQType"].Value = this.AETType.
                        ToString();
563                 sqlCommand.Parameters.Add("@aetQQuestionType", SqlDbType
                        .VarChar, AETLimitations.AETQuestionType);
564                 sqlCommand.Parameters["@aetQQuestionType"].Value = this.
                        QuestionType.ToString();
565                 sqlCommand.Parameters.Add("@aetQViewQuestionDateTime",
                        SqlDbType.DateTime);
566                 sqlCommand.Parameters["@aetQViewQuestionDateTime"].Value
                         = this.ViewQuestionDateTime;
567                 sqlCommand.Parameters.Add("@aetQDueDateTime", SqlDbType.
                        DateTime);
568                 sqlCommand.Parameters["@aetQDueDateTime"].Value = this.
                        DueDateTime;
569                 sqlCommand.Parameters.Add("@aetQHaltSubmissionDateTime",
                         SqlDbType.DateTime);
570                 sqlCommand.Parameters["@aetQHaltSubmissionDateTime"].
                        Value = this.HaltSubmissionsDateTime;
571                 sqlCommand.Parameters.Add("
                        @aetQViewMarkedAnswersDateTime", SqlDbType.DateTime);
572                 sqlCommand.Parameters["@aetQViewMarkedAnswersDateTime"].
                        Value = this.ViewMarkedAnswersDateTime;
573                 sqlCommand.Parameters.Add("@aetQAmountOfQuestions",
                        SqlDbType.Int);
574                 sqlCommand.Parameters["@aetQAmountOfQuestions"].Value =
                        this.AmountOfQuestions;
575                 this.DataBridge.SqlConnection.Open();
576                 sqlCommand.ExecuteNonQuery();
577                 this.DataBridge.SqlConnection.Close();
578
579                     //Create Answer directory for AETQuestion
580                 if (!Directory.Exists(aetAnswerDirectory))
581                     Directory.CreateDirectory(aetAnswerDirectory);
582
583                     // Write information to xml file
584                 XmlSerializer xmlSerializer = new XmlSerializer(this.
                        GetType());
585                 XmlTextWriter xmlTextWriter;
586                 xmlSerializer.Serialize(xmlTextWriter = new
                        XmlTextWriter(aetQFilePath, Encoding.UTF8), this);
587                 xmlTextWriter.Close();
588
589                 return this.AETQID;
590             }
591
592             /// <summary>
```

```
        /// Uses current instance's properties and re-stores the data.
        /// </summary>
        public void Alter()
        {
            string aetQFilePath = null;
            string aetAnswerDirectory = null;
            string paddedAETQID = null;

            switch (this.AETType)
            {
                case AETType.Assignment:
                    aetQFilePath = this.DataBridge.GetClassAssignmentFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
                    aetAnswerDirectory = this.DataBridge.GetClassAssignmentAnswersDirectory(this.ClassID, this.AETQID, true);
                    break;
                case AETType.Exam:
                    aetQFilePath = this.DataBridge.GetClassExamFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
                    aetAnswerDirectory = this.DataBridge.GetClassExamAnswersDirectory(this.ClassID, this.AETQID, true);
                    break;
                case AETType.Tutorial:
                    aetQFilePath = this.DataBridge.GetClassTutorialFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
                    aetAnswerDirectory = this.DataBridge.GetClassTutorialAnswersDirectory(this.ClassID, this.AETQID, true);
                    break;
            }

                // Write information to sql server
            SqlCommand sqlCommand = new SqlCommand("AlterAETQuestion", this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
            sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char, 7);
            sqlCommand.Parameters["@classID"].Value = this.ClassID;
            sqlCommand.Parameters.Add("@aetQTitle", SqlDbType.VarChar, 100);
            sqlCommand.Parameters["@aetQTitle"].Value = this.Title;
            sqlCommand.Parameters.Add("@aetQDescription", SqlDbType.VarChar, 30);
            sqlCommand.Parameters["@aetQDescription"].Value = this.Description;
            sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, 20);
```

```
629                 sqlCommand.Parameters["@aetQType"].Value = this.AETType.
                        ToString();
630                 sqlCommand.Parameters.Add("@aetQQuestionType", SqlDbType
                        .VarChar, 20);
631                 sqlCommand.Parameters["@aetQQuestionType"].Value = this.
                        QuestionType.ToString();
632                 sqlCommand.Parameters.Add("@aetQViewQuestionDateTime",
                        SqlDbType.DateTime);
633                 sqlCommand.Parameters["@aetQViewQuestionDateTime"].Value
                         = this.ViewQuestionDateTime;
634                 sqlCommand.Parameters.Add("@aetQDueDateTime", SqlDbType.
                        DateTime);
635                 sqlCommand.Parameters["@aetQDueDateTime"].Value = this.
                        DueDateTime;
636                 sqlCommand.Parameters.Add("@aetQHaltSubmissionDateTime",
                         SqlDbType.DateTime);
637                 sqlCommand.Parameters["@aetQHaltSubmissionDateTime"].
                        Value = this.HaltSubmissionsDateTime;
638                 sqlCommand.Parameters.Add("
                        @aetQViewMarkedAnswersDateTime", SqlDbType.DateTime);
639                 sqlCommand.Parameters["@aetQViewMarkedAnswersDateTime"].
                        Value = this.ViewMarkedAnswersDateTime;
640                 sqlCommand.Parameters.Add("@aetQAmountOfQuestions",
                        SqlDbType.Int);
641                 sqlCommand.Parameters["@aetQAmountOfQuestions"].Value =
                        this.AmountOfQuestions;
642                 this.DataBridge.SqlConnection.Open();
643                 sqlCommand.ExecuteNonQuery();
644                 this.DataBridge.SqlConnection.Close();
645
646                     // Write information to xml file
647                 XmlSerializer xmlSerializer = new XmlSerializer(this.
                        GetType());
648                 XmlTextWriter xmlTextWriter;
649                 xmlSerializer.Serialize(xmlTextWriter = new
                        XmlTextWriter(aetQFilePath, Encoding.UTF8), this);
650                 xmlTextWriter.Close();
651             }
652
653             /// <summary>
654             /// Uses the current instance's properties and deletes the
                    stored data.
655             /// </summary>
656             public void Delete()
657             {
658                 string aetQFilePath = null;
659                 string aetAnswerDirectory = null;
660                 string paddedAETQID = null;
661
662                 switch (this.AETType)
663                 {
664                     case AETType.Assignment:
665                         aetQFilePath = this.DataBridge.
                                GetClassAssignmentFilePath(this.ClassID, this
                                .AETQID, out paddedAETQID, true);
```

```
                    aetAnswerDirectory = this.DataBridge.GetClassAssignmentAnswersDirectory(this.ClassID, this.AETQID, true);
                    break;
                case AETType.Exam:
                    aetQFilePath = this.DataBridge.GetClassExamFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
                    aetAnswerDirectory = this.DataBridge.GetClassExamAnswersDirectory(this.ClassID, this.AETQID, true);
                    break;
                case AETType.Tutorial:
                    aetQFilePath = this.DataBridge.GetClassTutorialFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
                    aetAnswerDirectory = this.DataBridge.GetClassTutorialAnswersDirectory(this.ClassID, this.AETQID, true);
                    break;
            }

                // Delete record from sql server
            SqlCommand sqlCommand = new SqlCommand("DeleteAETQuestion", this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
            sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char, 7);
            sqlCommand.Parameters["@classID"].Value = this.ClassID;
            this.DataBridge.SqlConnection.Open();
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();

                // Delete xml file
            File.Delete(aetQFilePath);
                // Delete folder with answers
            Directory.Delete(aetAnswerDirectory, true);
        }
        #endregion
    }

    /// <summary>
    /// A sealed class that contains properties and methods of a Information Question.
    /// </summary>
    [XmlRoot("AETQuestion")]
    public sealed class AETInformationQuestion : AETQuestion
    {
        #region Properties
        /// <summary>
        /// An array of information questions.
        /// </summary>
        private InformationQuestion[] _questions;
```

```
708         [XmlArray("Questions")]
709         [XmlArrayItem("Question")]
710         public InformationQuestion[] Questions
711         {
712             get { return this._questions; }
713             set { this._questions = value; }
714         }
715
716         /// <summary>
717         /// The filename of the provided solutions file.
718         /// </summary>
719         private string _aetSFilename;
720         [XmlElement("SolutionFilename")]
721         public string AETSFilename
722         {
723             get { return this._aetSFilename; }
724             set { this._aetSFilename = value; }
725         }
726         #endregion
727
728         #region Constructors
729         /// <summary>
730         /// Used for XmlSerializer purposes.
731         /// </summary>
732         public AETInformationQuestion() : base(QuestionType.
                Information) {}
733         /// <summary>
734         /// Main Constructor.
735         /// </summary>
736         /// <param name="aetType">The associated AET type.</param>
737         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
738         public AETInformationQuestion(AETType aetType, DataBridge
                dataBridge) : base(aetType, QuestionType.Information,
                dataBridge) {}
739         #endregion
740
741         #region Static Data Retrieving Methods
742         /// <summary>
743         /// Find a specific AETInformationQuestion.
744         /// </summary>
745         /// <param name="classID">The associated class ID.</param>
746         /// <param name="aetQID">The associated AETQuestion</param>
747         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
748         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
749         /// <returns>Return the found AETInformationQuestion. Null
                if not found.</returns>
750         public static AETInformationQuestion
                FindAETInformationQuestion(string classID, int aetQID,
                AETType aetType, DataBridge dataBridge)
751         {
752             string aetQFilePath = null;
753             string paddedAETQID = null;
```

```
754
755             try
756             {
757                     // Find path
758                 switch (aetType)
759                 {
760                     case AETType.Assignment:
761                         aetQFilePath = dataBridge.
                                GetClassAssignmentFilePath(classID,
                                aetQID, out paddedAETQID, true);
762                         break;
763                     case AETType.Exam:
764                         aetQFilePath = dataBridge.
                                GetClassExamFilePath(classID, aetQID, out
                                 paddedAETQID, true);
765                         break;
766                     case AETType.Tutorial:
767                         aetQFilePath = dataBridge.
                                GetClassTutorialFilePath(classID, aetQID,
                                 out paddedAETQID, true);
768                         break;
769                 }
770             }
771             catch
772             {
773                     // Not found
774                 return null;
775             }
776
777                 // Deserialise the question
778             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    AETInformationQuestion));
779             XmlTextReader xmlTextReader;
780             AETInformationQuestion newAETQ = xmlSerializer.
                    Deserialize(xmlTextReader = new XmlTextReader(
                    aetQFilePath)) as AETInformationQuestion;
781             xmlTextReader.Close();
782
783             return newAETQ;
784         }
785         #endregion
786     }
787
788     /// <summary>
789     /// A sealed class that contains properties and methods of a
            Multiple Choice Question.
790     /// </summary>
791     [XmlRoot("AETQuestion")]
792     public sealed class AETMultipleChoiceQuestion : AETQuestion
793     {
794         #region Properties
795         /// <summary>
796         /// An array of mulitple choice questions.
797         /// </summary>
798         private MultipleChoiceQuestion[] _questions;
```

```
799         [XmlArray("Questions")]
800         [XmlArrayItem("Question")]
801         public MultipleChoiceQuestion[] Questions
802         {
803             get { return this._questions; }
804             set { this._questions = value; }
805         }
806         #endregion
807
808         #region Constructors
809         /// <summary>
810         /// Used for XmlSerializer purposes.
811         /// </summary>
812         public AETMultipleChoiceQuestion() : base(QuestionType.
                MultipleChoice) {}
813         /// <summary>
814         /// Main Constructor.
815         /// </summary>
816         /// <param name="aetType">The associated AET type.</param>
817         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
818         public AETMultipleChoiceQuestion(AETType aetType, DataBridge
                 dataBridge) : base(aetType, QuestionType.Information,
                dataBridge) {}
819         #endregion
820
821         #region Static Data Retrieving Methods
822         /// <summary>
823         /// Find a specific AETMultipleChoiceQuestion.
824         /// </summary>
825         /// <param name="classID">The associated class ID.</param>
826         /// <param name="aetQID">The associated AETQuestion</param>
827         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
828         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
829         /// <returns>Return the found AETMultipleChoiceQuestion.
                Null if not found.</returns>
830         public static AETMultipleChoiceQuestion
                FindAETMultipleChoiceQuestion(string classID, int aetQID,
                 AETType aetType, DataBridge dataBridge)
831         {
832             string aetQFilePath = null;
833             string paddedAETQID = null;
834
835             try
836             {
837                     // Find path
838                 switch (aetType)
839                 {
840                     case AETType.Assignment:
841                         aetQFilePath = dataBridge.
                                GetClassAssignmentFilePath(classID,
                                aetQID, out paddedAETQID, true);
842                         break;
```

```
                        case AETType.Exam:
                            aetQFilePath = dataBridge.
                                GetClassExamFilePath(classID, aetQID, out
                                 paddedAETQID, true);
                            break;
                        case AETType.Tutorial:
                            aetQFilePath = dataBridge.
                                GetClassTutorialFilePath(classID, aetQID,
                                 out paddedAETQID, true);
                            break;
                    }
                }
                catch
                {
                        // Not found
                    return null;
                }

                    // Deserialise the question
                XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    AETMultipleChoiceQuestion));
                XmlTextReader xmlTextReader;
                AETMultipleChoiceQuestion newAETQ = xmlSerializer.
                    Deserialize(xmlTextReader = new XmlTextReader(
                    aetQFilePath)) as AETMultipleChoiceQuestion;
                xmlTextReader.Close();

                return newAETQ;
            }
            #endregion
        }

        /// <summary>
        /// A sealed class that contains properties and methods of a
            Short Response Question.
        /// </summary>
        [XmlRoot("AETQuestion")]
        public sealed class AETShortResponseQuestion : AETQuestion
        {
            #region Properties
            /// <summary>
            /// An array of short response questions.
            /// </summary>
            private ShortResponseQuestion[] _questions;
            [XmlArray("Questions")]
            [XmlArrayItem("Question")]
            public ShortResponseQuestion[] Questions
            {
                get { return this._questions; }
                set { this._questions = value; }
            }

            /// <summary>
            /// An array of short resposnse question's answers (entered
                by Instructors).
```

```
889         /// </summary>
890         private ShortResponseAnswer[] _answers;
891         [XmlArray("Answers")]
892         [XmlArrayItem("Answer")]
893         public ShortResponseAnswer[] Answers
894         {
895             get { return this._answers; }
896             set { this._answers = value; }
897         }
898         #endregion
899
900         #region Constructors
901         /// <summary>
902         /// Used for XmlSerializer purposes.
903         /// </summary>
904         public AETShortResponseQuestion() : base(QuestionType.
                ShortResponse) {}
905         /// <summary>
906         /// Main Constructor.
907         /// </summary>
908         /// <param name="aetType">The associated AET type.</param>
909         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
910         public AETShortResponseQuestion(AETType aetType, DataBridge
                dataBridge) : base(aetType, QuestionType.Information,
                dataBridge) {}
911         #endregion
912
913         #region Static Data Retrieving Methods
914         /// <summary>
915         /// Find a specific AETShortResponseQuestion.
916         /// </summary>
917         /// <param name="classID">The associated class ID.</param>
918         /// <param name="aetQID">The associated AETQuestion</param>
919         /// <param name="aetType">The type of the AETAnswer (
                Assignment, Exam, or Tutorial).</param>
920         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
921         /// <returns>Return the found AETShortResponseQuestion. Null
                 if not found.</returns>
922         public static AETShortResponseQuestion
                FindAETShortResponseQuestion(string classID, int aetQID,
                AETType aetType, DataBridge dataBridge)
923         {
924             string aetQFilePath = null;
925             string paddedAETQID = null;
926
927             try
928             {
929                     // Find path
930                 switch (aetType)
931                 {
932                     case AETType.Assignment:
933                         aetQFilePath = dataBridge.
                                GetClassAssignmentFilePath(classID,
```

```
                                aetQID, out paddedAETQID, true);
                            break;
                        case AETType.Exam:
                            aetQFilePath = dataBridge.
                                GetClassExamFilePath(classID, aetQID, out
                                 paddedAETQID, true);
                            break;
                        case AETType.Tutorial:
                            aetQFilePath = dataBridge.
                                GetClassTutorialFilePath(classID, aetQID,
                                 out paddedAETQID, true);
                            break;
                    }
                }
                catch
                {
                        // Not found
                    return null;
                }

                    // Deserialise the question
                XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    AETShortResponseQuestion));
                XmlTextReader xmlTextReader;
                AETShortResponseQuestion newAETQ = xmlSerializer.
                    Deserialize(xmlTextReader = new XmlTextReader(
                    aetQFilePath)) as AETShortResponseQuestion;
                xmlTextReader.Close();

                return newAETQ;
            }
            #endregion
        }
}
```

Listing H.96: Authentication.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A sealed class that contains static methods to check
        authentication.
    /// </summary>
    public sealed class Authentication
    {
        #region Constructors
        /// <summary>
        /// Disabled constructor.
        /// </summary>
        private Authentication() {}
        /// <summary>
        /// Main constructor.
```

```
19          /// </summary>
20          /// <param name="questionType">The associated question type
                .</param>
21          public Authentication(DataBridge dataBridge)
22          {
23              this._dataBridge = dataBridge;
24          }
25          #endregion
26
27          #region Data Management Properties
28          /// <summary>
29          /// The DataBridge that connects to the data.
30          /// </summary>
31          private DataBridge _dataBridge = null;
32          public DataBridge DataBridge
33          {
34              get { return this._dataBridge; }
35              set { this._dataBridge = value; }
36          }
37          #endregion
38
39          #region Static Data Retrieving Methods
40          /// <summary>
41          /// Check username and password.
42          /// </summary>
43          /// <param name="username">The associated username.</param>
44          /// <param name="password">The associated password (in non-
                encrypted string form).</param>
45          /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
46          /// <returns>Returns true if password is correct. False if
                otherwise.</returns>
47          public static bool CheckAuthentication(int username, string
                password, DataBridge dataBridge)
48          {
49              SqlCommand sqlCommand = new SqlCommand("Authentication",
                     dataBridge.SqlConnection);
50              sqlCommand.CommandType = CommandType.StoredProcedure;
51              sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
52              sqlCommand.Parameters["@userID"].Value = username;
53              sqlCommand.Parameters.Add("@password", SqlDbType.Char,
                    UserLimitations.Password);
54              sqlCommand.Parameters["@password"].Value = MD5Encryption
                    .EncryptStringToByteString(password);
55              sqlCommand.Parameters.Add("RT_VAL", SqlDbType.Int);
56              sqlCommand.Parameters["RT_VAL"].Direction =
                    ParameterDirection.ReturnValue;
57              dataBridge.SqlConnection.Open();
58              sqlCommand.ExecuteNonQuery();
59              dataBridge.SqlConnection.Close();
60
61                  // Check returned value from SQL
62              if ((int)sqlCommand.Parameters["RT_VAL"].Value == 0)
63                  return false;
64              else
```

```
                return true;
            }
            #endregion
        }
}
```

Listing H.97: Class.cs

```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A sealed class that contains methods and properties for classes.
    /// </summary>
    [XmlRoot("Class")]
    public sealed class Class
    {
        #region Properties
        /// <summary>
        /// The unique ID.
        /// </summary>
        private string _classID;
        [XmlAttribute("classID")]
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        /// <summary>
        /// The associated class name.
        /// </summary>
        private string _className;
        [XmlElement("Name")]
        public string ClassName
        {
            get { return this._className; }
            set { this._className = value; }
        }

        /// <summary>
        /// Whether or not non-enrolled users can visit the class' section.
        /// </summary>
        private bool _denyOtherUsers;
        [XmlAttribute("denyOtherUsers")]
        public bool DenyOtherUsers
        {
```

```
            get { return this._denyOtherUsers; }
            set { this._denyOtherUsers = value; }
        }
        #endregion

        #region Constructors
        /// <summary>
        /// Used for XmlSerializer purposes.
        /// </summary>
        public Class() {}
        /// <summary>
        /// Main Constructor.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="dataBridge">The DataBridge that connects to the data.</param>
        public Class(string classID, DataBridge dataBridge)
        {
            this._classID = classID;
            this._dataBridge = dataBridge;
        }
        #endregion

        #region Data Management Properties
        /// <summary>
        /// The DataBridge that connects to the data.
        /// </summary>
        private DataBridge _dataBridge = null;
        [XmlIgnore()]
        public DataBridge DataBridge
        {
            get { return this._dataBridge; }
            set { this._dataBridge = value; }
        }
        #endregion

        #region Data Management Methods
        /// <summary>
        /// Uses current instance's properties and stores the data.
        /// </summary>
        public string Create()
        {
            if (Directory.Exists(this.DataBridge.GetClassDirectory(this.ClassID, false)))
                throw new Exception("ClassID exists!");

                // Create SQL Server DB entry
            SqlCommand sqlCommand = new SqlCommand("CreateClass",
                this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@classID", SqlDbType.VarChar,
                ClassLimitations.ClassID);
            sqlCommand.Parameters["@classID"].Value = this.ClassID;
            sqlCommand.Parameters.Add("@className", SqlDbType.VarChar, ClassLimitations.ClassName);
```

```
98              sqlCommand.Parameters["@className"].Value = this.
                    ClassName;
99              sqlCommand.Parameters.Add("@denyOtherUsers", SqlDbType.
                    Bit);
100             sqlCommand.Parameters["@denyOtherUsers"].Value = this.
                    DenyOtherUsers;
101             this.DataBridge.SqlConnection.Open();
102             sqlCommand.ExecuteNonQuery();
103             this.DataBridge.SqlConnection.Close();
104
105                 // Create Class folders
106             Directory.CreateDirectory(this.DataBridge.
                    GetClassDirectory(this.ClassID, false));
107             Directory.CreateDirectory(this.DataBridge.
                    GetClassAnnouncementsDirectory(this.ClassID, false));
108             Directory.CreateDirectory(this.DataBridge.
                    GetClassAssignmentsDirectory(this.ClassID, false));
109             Directory.CreateDirectory(this.DataBridge.
                    GetClassExamsDirectory(this.ClassID, false));
110             Directory.CreateDirectory(this.DataBridge.
                    GetClassThreadsDirectory(this.ClassID, false));
111             Directory.CreateDirectory(this.DataBridge.
                    GetClassNotesDirectory(this.ClassID, false));
112             Directory.CreateDirectory(this.DataBridge.
                    GetClassTutorialsDirectory(this.ClassID, false));
113
114             return this.ClassID;
115         }
116
117         /// <summary>
118         /// Uses current instance's properties and re-stores the
                data.
119         /// </summary>
120         public void Alter()
121         {
122             SqlCommand sqlCommand = new SqlCommand("AlterClass",
                    this.DataBridge.SqlConnection);
123             sqlCommand.CommandType = CommandType.StoredProcedure;
124             sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
125             sqlCommand.Parameters["@classID"].Value = this.ClassID;
126             sqlCommand.Parameters.Add("@className", SqlDbType.
                    VarChar, ClassLimitations.ClassName);
127             sqlCommand.Parameters["@className"].Value = this.
                    ClassName;
128             sqlCommand.Parameters.Add("@denyOtherUsers", SqlDbType.
                    Bit);
129             sqlCommand.Parameters["@denyOtherUsers"].Value = this.
                    DenyOtherUsers;
130             this.DataBridge.SqlConnection.Open();
131             sqlCommand.ExecuteNonQuery();
132             this.DataBridge.SqlConnection.Close();
133         }
134
135         /// <summary>
```

```
136         /// Uses the current instance's properties and deletes the
                stored data.
137         /// </summary>
138         public void Delete()
139         {
140             SqlCommand sqlCommand = new SqlCommand("DeleteClass",
                    this.DataBridge.SqlConnection);
141             sqlCommand.CommandType = CommandType.StoredProcedure;
142             sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
143             sqlCommand.Parameters["@classID"].Value = this.ClassID;
144             this.DataBridge.SqlConnection.Open();
145             sqlCommand.ExecuteNonQuery();
146             this.DataBridge.SqlConnection.Close();
147
148             Directory.Delete(this.DataBridge.GetClassDirectory(this.
                    ClassID, true), true);
149         }
150         #endregion
151
152         #region Data Retrieving Methods
153         /// <summary>
154         /// Enrol a user to the class.
155         /// </summary>
156         /// <param name="userID">The associated user's ID.</param>
157         /// <param name="roleID">The associated role ID.</param>
158         public void EnrolUser(int userID, RoleID roleID)
159         {
160             Role role = new Role(roleID, this.DataBridge);
161             role.AssignUserToClass(userID, this.ClassID);
162         }
163
164         /// <summary>
165         /// DeEnrol a user to the class.
166         /// </summary>
167         /// <param name="userID">The associated user's ID.</param>
168         /// <param name="roleID">The associated role ID.</param>
169         public void DeEnrolUser(int userID, RoleID roleID)
170         {
171             Role role = new Role(roleID, this.DataBridge);
172             role.UnassignUserToClass(userID, this.ClassID);
173         }
174
175         /// <summary>
176         /// Find all roles a user is enrolled in.
177         /// </summary>
178         /// <param name="userID">The associated user's ID.</param>
179         /// <returns>An array of associated roles. Null if none
                found.</returns>
180         public RoleID[] IsInClass(int userID)
181         {
182             ArrayList roleIDs = new ArrayList();
183
184             SqlCommand sqlCommand = new SqlCommand("IsInClass", this
                    .DataBridge.SqlConnection);
```

```
185                 sqlCommand.CommandType = CommandType.StoredProcedure;
186                 sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
187                 sqlCommand.Parameters["@userID"].Value = userID;
188                 sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                        ClassLimitations.ClassID);
189                 sqlCommand.Parameters["@classID"].Value = this.ClassID;
190                 this.DataBridge.SqlConnection.Open();
191                 SqlDataReader sqlDataReader = sqlCommand.ExecuteReader()
                        ;
192                 while (sqlDataReader.Read())
193                 {
194                     switch (sqlDataReader["roleID"] as string)
195                     {
196                         case "Administrators":
197                             roleIDs.Add(RoleID.Administrators);
198                             break;
199                         case "Instructors":
200                             roleIDs.Add(RoleID.Instructors);
201                             break;
202                         case "Students":
203                             roleIDs.Add(RoleID.Students);
204                             break;
205                     }
206                 }
207                 this.DataBridge.SqlConnection.Close();
208
209                     // Check what roles
210                 if (roleIDs.Count == 0)
211                     return null;
212                 else
213                     return (RoleID[])roleIDs.ToArray(typeof(RoleID));
214             }
215             #endregion
216
217             #region Static Data Retrieving Methods
218             /// <summary>
219             /// Find a class.
220             /// </summary>
221             /// <param name="classID">The class ID.</param>
222             /// <param name="dataBridge">The DataBridge that connects to
                    the data.</param>
223             /// <returns>Returns the found class. Null if not found.</
                    returns>
224             public static Class FindClass(string classID, DataBridge
                    dataBridge)
225             {
226                 SqlCommand sqlCommand = new SqlCommand("FindClass",
                        dataBridge.SqlConnection);
227                 sqlCommand.CommandType = CommandType.StoredProcedure;
228                 sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                        ClassLimitations.ClassID);
229                 sqlCommand.Parameters["@classID"].Value = classID;
230                 dataBridge.SqlConnection.Open();
231                 XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
                        ExecuteXmlReader();
```

```
232
233             Class c = null;
234             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    Class));
235                 // Deserialise class
236             while (xmlTextReader.Read())
237                 if (xmlSerializer.CanDeserialize(xmlTextReader))
238                     c = xmlSerializer.Deserialize(xmlTextReader) as
                            Class;
239
240             xmlTextReader.Close();
241             dataBridge.SqlConnection.Close();
242
243
244             if (c == null)
245                 return null;
246             else
247             {
248                 c.DataBridge = dataBridge;
249                 return c;
250             }
251         }
252
253         /// <summary>
254         /// Find the classes associated with a user.
255         /// </summary>
256         /// <param name="userID">The associated user ID.</param>
257         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
258         /// <returns>Returns the found classes. Null if not found.</
                returns>
259         public static Class[] FindClasses(int userID, DataBridge
                dataBridge)
260         {
261             SqlCommand sqlCommand = new SqlCommand("FindClasses",
                    dataBridge.SqlConnection);
262             sqlCommand.CommandType = CommandType.StoredProcedure;
263             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
264             sqlCommand.Parameters["@userID"].Value = userID;
265             dataBridge.SqlConnection.Open();
266             XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
                    ExecuteXmlReader();
267
268             ArrayList classes = new ArrayList();
269             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    Class));
270                 // Deserialise classes
271             while (xmlTextReader.Read())
272             {
273                 while (xmlSerializer.CanDeserialize(xmlTextReader))
274                 {
275                     Class tempClass = xmlSerializer.Deserialize(
                            xmlTextReader) as Class;
276                     tempClass.DataBridge = dataBridge;
277                     classes.Add(tempClass);
```

```
                }
            }

            xmlTextReader.Close();
            dataBridge.SqlConnection.Close();

            if (classes.Count == 0)
                return null;
            else
                return (Class[])classes.ToArray(typeof(Class));
        }
        #endregion
    }
}
```

Listing H.98: Configuration.cs

```
using System;
using System.Security.Cryptography;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A sealed class that contains methods and properties for
        configuration.
    /// </summary>
    [XmlRoot("Configuration")]
    public sealed class Configuration
    {
        #region Properties
        #region InstitutionConfiguration
        /// <summary>
        /// The institution name.
        /// </summary>
        private string _institutionName = "Online Learning
            Environment";
        [XmlElement("InstitutionName")]
        public string InstitutionName
        {
            get { return this._institutionName; }
            set { this._institutionName = value; }
        }

        /// <summary>
        /// The institution name's abbreviation.
        /// </summary>
        private string _institutionNameAbbreviation = "OLE";
        [XmlElement("InstitutionNameAbbreviation")]
        public string InstitutionNameAbbreviation
        {
            get { return this._institutionNameAbbreviation; }
            set { this._institutionNameAbbreviation = value; }
        }
```

```
38
39          /// <summary>
40          /// The institution's logo image for the web site header.
41          /// </summary>
42          private string _institutionLogoImage = "Images\\header.jpg";
43          [XmlElement("InstitutionLogoImage")]
44          public string InstitutionLogoImage
45          {
46              get { return _institutionLogoImage; }
47              set { this._institutionLogoImage = value; }
48          }
49          #endregion
50
51          #region TerminologyConfiguration
52          /// <summary>
53          /// Institution terminology.
54          /// </summary>
55          private string _institution = "Institution";
56          [XmlElement("Institution")]
57          public string Institution
58          {
59              get { return this._institution; }
60              set { this._institution = value; }
61          }
62          /// <summary>
63          /// Class terminology.
64          /// </summary>
65          private string _class = "Class";
66          [XmlElement("Class")]
67          public string Class
68          {
69              get { return this._class; }
70              set { this._class = value; }
71          }
72          /// <summary>
73          /// Assignment terminology.
74          /// </summary>
75          private string _assignment = "Assignment";
76          [XmlElement("Assignment")]
77          public string Assignment
78          {
79              get { return this._assignment; }
80              set { this._assignment = value; }
81          }
82          /// <summary>
83          /// Exam terminology.
84          /// </summary>
85          private string _exam = "Exam";
86          [XmlElement("Exam")]
87          public string Exam
88          {
89              get { return this._exam; }
90              set { this._exam = value; }
91          }
92          /// <summary>
```

```
93          /// Tutorial terminology.
94          /// </summary>
95          private string _tutorial = "Tutorial";
96          [XmlElement("Tutorial")]
97          public string Tutorial
98          {
99              get { return this._tutorial; }
100             set { this._tutorial = value; }
101         }
102         /// <summary>
103         /// Note terminology.
104         /// </summary>
105         private string _note = "Note";
106         [XmlElement("Note")]
107         public string Note
108         {
109             get { return this._note; }
110             set { this._note = value; }
111         }
112         #endregion
113         #endregion
114
115         #region Constructors
116         /// <summary>
117         /// Used for XmlSerializer purposes.
118         /// </summary>
119         public Configuration() {}
120         /// <summary>
121         /// Main Constructor.
122         /// </summary>
123         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
124         public Configuration(DataBridge dataBridge)
125         {
126             this._dataBridge = dataBridge;
127         }
128         #endregion
129
130         #region Data Management Properties
131         /// <summary>
132         /// The DataBridge that connects to the data.
133         /// </summary>
134         private DataBridge _dataBridge = null;
135         [XmlIgnore()]
136         public DataBridge DataBridge
137         {
138             get { return this._dataBridge; }
139             set { this._dataBridge = value; }
140         }
141         #endregion
142
143         #region Data Management Methods
144         /// <summary>
145         /// Uses current instance's properties and stores the data.
146         /// </summary>
```

```
147         public void Create()
148         {
149             string configurationFilePath = this.DataBridge.
                    GetConfigurationFilePath(false);
150
151                 // Serialise the data
152             XmlSerializer xmlSerializer = new XmlSerializer(this.
                    GetType());
153             XmlTextWriter xmlTextWriter;
154             xmlSerializer.Serialize(xmlTextWriter = new
                    XmlTextWriter(configurationFilePath, Encoding.UTF8),
                    this);
155             xmlTextWriter.Close();
156         }
157
158         /// <summary>
159         /// Uses current instance's properties and re-stores the
                data.
160         /// </summary>
161         public void Alter()
162         {
163             string configurationFilePath = this.DataBridge.
                    GetConfigurationFilePath(false);
164
165                 // Serialise the data
166             XmlSerializer xmlSerializer = new XmlSerializer(this.
                    GetType());
167             XmlTextWriter xmlTextWriter;
168             xmlSerializer.Serialize(xmlTextWriter = new
                    XmlTextWriter(configurationFilePath, Encoding.UTF8),
                    this);
169             xmlTextWriter.Close();
170         }
171         #endregion
172
173         #region Static Data Retrieving Methods
174         /// <summary>
175         /// Find configuration
176         /// </summary>
177         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
178         /// <returns>Returns a data filled Configuration class. Null
                if not found.</returns>
179         public static Configuration FindConfiguration(DataBridge
                dataBridge)
180         {
181             string configurationFilePath;
182             try
183             {
184                 configurationFilePath = dataBridge.
                        GetConfigurationFilePath(true);
185             }
186             catch
187             {
188                     // Not found
```

```
                return null;
            }

            XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                Configuration));
            XmlTextReader xmlTextReader;
                // Deserialise data
            Configuration newC = (Configuration)xmlSerializer.
                Deserialize(xmlTextReader = new XmlTextReader(
                configurationFilePath));
            xmlTextReader.Close();

            if (newC == null)
                return null;
            else
            {
                newC.DataBridge = dataBridge;
                return newC;
            }
        }
        #endregion
    }
}
```

Listing H.99: DataBridge.cs

```
using System;
using System.Data.SqlClient;
using System.IO;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// The class that provides properties and methods to connect to
    /// Xml files and an SQL Server database.
    /// </summary>
    public sealed class DataBridge
    {
        #region Properties
        private SqlConnection _sqlConnection = null;
        /// <summary>
        /// The connection to an SQL Server database.
        /// </summary>
        public SqlConnection SqlConnection
        {
            get { return this._sqlConnection; }
        }

        private string _classesDirectory = null;
        /// <summary>
        /// The directory that holds all class critical Xml files.
        /// </summary>
        public string ClassesDirectory
        {
            get { return this._classesDirectory; }
        }
```

```
31
32          private string _institutionDirectory = null;
33          /// <summary>
34          /// The directory that holds all insitution critical Xml
                files.
35          /// </summary>
36          public string InstitutionDirectory
37          {
38              get { return this._institutionDirectory; }
39          }
40          #endregion
41
42          #region Methods
43          /// <summary>
44          /// Used to pad a numeric number by a certain length (
                FileLimitations.FilenameLength).
45          /// It also pads the numeric number by a certain character (
                FileLimitations.PadCharacter).
46          /// </summary>
47          /// <param name="numericID">The numeric number to pad.</
                param>
48          /// <returns>Returns a padded number string.</returns>
49          private string PadNumericID(int numericID)
50          {
51              return numericID.ToString().PadLeft(FileLimitations.
                    FilenameLength, FileLimitations.PadCharacter);
52          }
53
54          /// <summary>
55          /// Gets the root data file directory (the directory before
                the class and
56          /// institution directory).
57          /// </summary>
58          /// <returns>Returns a string with the root directory.</
                returns>
59          private string GetRootDirectory()
60          {
61              string institutionDirectory = this.InstitutionDirectory;
62              return institutionDirectory.TrimEnd("Institution".
                    ToCharArray());
63          }
64
65          /// <summary>
66          /// Retrieves a file path with a certain name for a file.
67          /// </summary>
68          /// <param name="directory">The directory where it belongs
                .</param>
69          /// <param name="nameID">The file's name.</param>
70          /// <param name="fileExtension">The extension of the file.</
                param>
71          /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
72          /// <returns>Returns the file path of the file.</returns>
73          private string GetFilePath(string directory, string nameID,
                string fileExtension, bool checkIfExists)
```

```
74          {
75              string filePath = directory + "\\" + nameID + "." +
                    fileExtension;
76              if (!File.Exists(filePath) && checkIfExists)
77                  throw new Exception("File does not exist!");
78
79              return filePath;
80          }
81          /// <summary>
82          /// Retrieves a file path with a certain number.
83          /// </summary>
84          /// <param name="directory">The directory where it belongs
                .</param>
85          /// <param name="numericID">The numeric number of the file
                .</param>
86          /// <param name="paddedID">The padded string number.</param>
87          /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
88          /// <returns>Returns the file path of the file.</returns>
89          private string GetFilePath(string directory, int numericID,
                out string paddedID, bool checkIfExists)
90          {
91              paddedID = PadNumericID(numericID);
92              string filePath = directory + "\\" + paddedID + ".xml";
93              if (!File.Exists(filePath) && checkIfExists)
94                  throw new Exception("File does not exist!");
95
96              return filePath;
97          }
98          /// <summary>
99          /// Retrieves a 'new' file path with a numeric number above
                the last in the directory (+1).
100         /// </summary>
101         /// <param name="directory">The directory where it belongs
                .</param>
102         /// <param name="numericID">The numeric number of the file
                .</param>
103         /// <param name="paddedID">The padded string number.</param>
104         /// <returns>Returns the file path of the file.</returns>
105         private string GetNextFilePath(string directory, out int
                numericID, out string paddedID)
106         {
107             string[] filenames = Directory.GetFiles(directory, "*.
                    xml");
108             if (filenames.Length > 0)
109             {
110                 string[] tempStrings = filenames[filenames.Length -
                        1].Split("\\".ToCharArray(), 100);
111                 numericID = Int32.Parse(tempStrings[tempStrings.
                        Length - 1].TrimEnd(".xml".ToCharArray())) + 1;
112             }
113             else
114             {
115                 numericID = 1;
116             }
```

```
117
118             paddedID = numericID.ToString().PadLeft(FileLimitations.
                    FilenameLength, FileLimitations.PadCharacter);
119
120             return (directory + "\\" + paddedID + ".xml");
121         }
122         #endregion
123
124         #region Constructors
125         /// <summary>
126         /// Disabled Constructor.
127         /// </summary>
128         private DataBridge() {}
129         /// <summary>
130         /// Main Constructor.
131         /// </summary>
132         /// <param name="sqlConnection">The shared connection to an
                SQL Server database.</param>
133         /// <param name="classesDirectory">The directory to the
                class Xml files.</param>
134         /// <param name="institutionDirectory">The directory to the
                institution Xml files.</param>
135         /// <param name="checkIfExists">If true, and directories are
                 not found, an expception is thrown.</param>
136         public DataBridge(SqlConnection sqlConnection, string
                classesDirectory, string institutionDirectory, bool
                checkIfExists)
137         {
138             this._sqlConnection = sqlConnection;
139
140             if (!Directory.Exists(this._classesDirectory =
                    classesDirectory) && classesDirectory != null &&
                    checkIfExists)
141                 throw new Exception("Classes Directory does not
                        exist!");
142             if (!Directory.Exists(this._institutionDirectory =
                    institutionDirectory) && institutionDirectory != null
                     && checkIfExists)
143                 throw new Exception("Institution Directory does not
                        exist!");
144         }
145         #endregion
146
147         #region Other Methods
148         /// <summary>
149         /// Get configuration file path.
150         /// </summary>
151         /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
152         /// <returns>Returns a string of the configuration's file
                path.</returns>
153         public string GetConfigurationFilePath(bool checkIfExists)
154         {
155             string filePath = this.GetRootDirectory() + "
                    Configuration.xml";
```

```
156             if (!File.Exists(filePath) && checkIfExists)
157                 throw new Exception("File does not exist!");
158
159             return filePath;
160         }
161         #endregion
162
163         #region Institution Enums
164         /// <summary>
165         /// Components within the Institution.
166         /// </summary>
167         private enum InstitutionComponent
168         {
169             Announcements,
170             ApplicationErrors,
171             SecurityErrors
172         }
173         #endregion
174
175         #region Institution Methods
176         /// <summary>
177         /// Gets an institution component's directory.
178         /// </summary>
179         /// <param name="institutionComponent">The institution
            component.</param>
180         /// <param name="checkIfExists">If true, and directory not
            found, an expception is throw.</param>
181         /// <returns>Returns a string of the path.</returns>
182         private string GetInstitutionComponentDirectory(
            InstitutionComponent institutionComponent, bool
            checkIfExists)
183         {
184             string directory = this.InstitutionDirectory + "\\" +
                institutionComponent;
185             if (!Directory.Exists(directory))
186                 throw new Exception("Directory does not exist!");
187
188             return directory;
189         }
190
191         /// <summary>
192         /// Gets the institution's announcement directory.
193         /// </summary>
194         /// <param name="checkIfExists">If true, and file is not
            found, an expception is throw.</param>
195         /// <returns>Returns a string of the path.</returns>
196         public string GetInstitutionAnnouncementsDirectory(bool
            checkIfExists)
197         {
198             return this.GetInstitutionComponentDirectory(
                InstitutionComponent.Announcements, checkIfExists);
199         }
200         /// <summary>
201         /// Gets a file path within the institution announcement
            directory.
```

```
202         /// </summary>
203         /// <param name="numericID">The numeric number ID.</param>
204         /// <param name="paddedID">The padded number ID.</param>
205         /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
206         /// <returns>Returns a string with the file path.</returns>
207         public string GetInstitutionAnnouncementFilePath(int
                numericID, out string paddedID, bool checkIfExists)
208         {
209             return this.GetFilePath(this.
                    GetInstitutionComponentDirectory(InstitutionComponent
                    .Announcements, true), numericID, out paddedID,
                    checkIfExists);
210         }
211         /// <summary>
212         /// Get next file path to exist within the institution
                announcement directory.
213         /// </summary>
214         /// <param name="numericID">The numeric number ID.</param>
215         /// <param name="paddedID">The padded number ID.</param>
216         /// <returns>Returns a string of the 'new' file path.</
                returns>
217         public string GetNextInstitutionAnnouncementFilePath(out int
                 numericID, out string paddedID)
218         {
219             return this.GetNextFilePath(this.
                    GetInstitutionComponentDirectory(InstitutionComponent
                    .Announcements, true), out numericID, out paddedID);
220         }
221
222         /// <summary>
223         /// Gets the institution's application errors directory.
224         /// </summary>
225         /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
226         /// <returns>Returns a string of the path.</returns>
227         public string GetInstitutionApplicationErrorsDirectory(bool
                checkIfExists)
228         {
229             return this.GetInstitutionComponentDirectory(
                    InstitutionComponent.ApplicationErrors, checkIfExists
                    );
230         }
231         /// <summary>
232         /// Gets a file path within the institution application
                errors directory.
233         /// </summary>
234         /// <param name="numericID">The numeric number ID.</param>
235         /// <param name="paddedID">The padded number ID.</param>
236         /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
237         /// <returns>Returns a string with the file path.</returns>
238         public string GetInstitutionApplicationErrorFilePath(int
                numericID, out string paddedID, bool checkIfExists)
239         {
```

```
240             return this.GetFilePath(this.
                    GetInstitutionComponentDirectory(InstitutionComponent
                    .ApplicationErrors, true), numericID, out paddedID,
                    checkIfExists);
241         }
242         /// <summary>
243         /// Get next file path to exist within the application
                errors directory.
244         /// </summary>
245         /// <param name="numericID">The numeric number ID.</param>
246         /// <param name="paddedID">The padded number ID.</param>
247         /// <returns>Returns a string of the 'new' file path.</
                returns>
248         public string GetNextInstitutionApplicationErrorFilePath(out
                 int numericID, out string paddedID)
249         {
250             return this.GetNextFilePath(this.
                    GetInstitutionComponentDirectory(InstitutionComponent
                    .ApplicationErrors, true), out numericID, out
                    paddedID);
251         }
252
253         /// <summary>
254         /// Gets the institution's security errors directory.
255         /// </summary>
256         /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
257         /// <returns>Returns a string of the path.</returns>
258         public string GetInstitutionSecurityErrorsDirectory(bool
                checkIfExists)
259         {
260             return this.GetInstitutionComponentDirectory(
                    InstitutionComponent.SecurityErrors, checkIfExists);
261         }
262         /// <summary>
263         /// Gets a file path within the institution security errors
                directory.
264         /// </summary>
265         /// <param name="numericID">The numeric number ID.</param>
266         /// <param name="paddedID">The padded number ID.</param>
267         /// <param name="checkIfExists">If true, and file is not
                found, an expception is throw.</param>
268         /// <returns>Returns a string with the file path.</returns>
269         public string GetInstitutionSecurityErrorFilePath(int
                numericID, out string paddedID, bool checkIfExists)
270         {
271             return this.GetFilePath(this.
                    GetInstitutionComponentDirectory(InstitutionComponent
                    .SecurityErrors, true), numericID, out paddedID,
                    checkIfExists);
272         }
273         /// <summary>
274         /// Get next file path to exist within the institution
                security errors directory.
275         /// </summary>
```

```
        /// <param name="numericID">The numeric number ID.</param>
        /// <param name="paddedID">The padded number ID.</param>
        /// <returns>Returns a string of the 'new' file path.</
            returns>
        public string GetNextInstitutionSecurityErrorFilePath(out
            int numericID, out string paddedID)
        {
            return this.GetNextFilePath(this.
                GetInstitutionComponentDirectory(InstitutionComponent
                .SecurityErrors, true), out numericID, out paddedID);
        }
        #endregion

        #region Class Enums
        /// <summary>
        /// Components within Classes.
        /// </summary>
        private enum ClassComponent
        {
            Announcements,
            Assignments,
            Exams,
            Tutorials,
            Notes,
            Forum
        }
        #endregion

        #region Class Methods
        /// <summary>
        /// Get a class' files directory.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="checkIfExists">If true, and directory is
            not found, an expception is throw.</param>
        /// <returns>Returns a string of the path.</returns>
        public string GetClassDirectory(string classID, bool
            checkIfExists)
        {
            string directory = this.ClassesDirectory + "\\" +
                classID;
            if (!Directory.Exists(directory) && checkIfExists)
                throw new Exception("Directory does not exist!");

            return directory;
        }
        /// <summary>
        /// Gets a class component's directory.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="classComponent">The class component.</param
            >
        /// <param name="checkIfExists">If true, and directory is
            not found, an expception is throw.</param>
        /// <returns>Returns a string of the path.</returns>
```

```
322         private string GetClassComponentDirectory(string classID,
                ClassComponent classComponent, bool checkIfExists)
323         {
324             return (this.GetClassDirectory(classID, checkIfExists) +
                    "\\" + classComponent);
325         }
326         /// <summary>
327         /// Gets a class component's sub-directory.
328         /// </summary>
329         /// <param name="classID">The associated class ID.</param>
330         /// <param name="paddedID">The padded number ID to be the
                sub-directory name.</param>
331         /// <param name="classComponent">The class component.</param
                >
332         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
333         /// <returns>Returns a string of the path.</returns>
334         private string GetClassSubComponentDirectory(string classID,
                 string paddedID, ClassComponent classComponent, bool
                checkIfExists)
335         {
336             return (this.GetClassDirectory(classID, checkIfExists) +
                    "\\" + classComponent + "\\" + paddedID);
337         }
338
339         /// <summary>
340         /// Get the class' announcements directory.
341         /// </summary>
342         /// <param name="classID">The associated class ID.</param>
343         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
344         /// <returns>Returns a string of the path.</returns>
345         public string GetClassAnnouncementsDirectory(string classID,
                 bool checkIfExists)
346         {
347             return this.GetClassComponentDirectory(classID,
                    ClassComponent.Announcements, checkIfExists);
348         }
349         /// <summary>
350         /// Get class announcement's file path.
351         /// </summary>
352         /// <param name="classID">The associated class ID.</param>
353         /// <param name="numericID">The numeric number ID.</param>
354         /// <param name="paddedID">The padded number ID.</param>
355         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
356         /// <returns>Returns a string of the file path.</returns>
357         public string GetClassAnnouncementFilePath(string classID,
                int numericID, out string paddedID, bool checkIfExists)
358         {
359             return this.GetFilePath(this.GetClassComponentDirectory(
                    classID, ClassComponent.Announcements, true),
                    numericID, out paddedID, checkIfExists);
360         }
361         /// <summary>
```

```
362         /// Get a 'new' class announcement's file path.
363         /// </summary>
364         /// <param name="classID">The associated class ID.</param>
365         /// <param name="numericID">The numeric number ID.</param>
366         /// <param name="paddedID">The padded number ID.</param>
367         /// <returns>Returns a string of the 'new' file path.</
                returns>
368         public string GetNextClassAnnouncementFilePath(string
                classID, out int numericID, out string paddedID)
369         {
370             return this.GetNextFilePath(this.
                    GetClassComponentDirectory(classID, ClassComponent.
                    Announcements, true), out numericID, out paddedID);
371         }
372
373         /// <summary>
374         /// Get the class' assignments directory.
375         /// </summary>
376         /// <param name="classID">The associated class ID.</param>
377         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
378         /// <returns>Returns a string of the path.</returns>
379         public string GetClassAssignmentsDirectory(string classID,
                bool checkIfExists)
380         {
381             return this.GetClassComponentDirectory(classID,
                    ClassComponent.Assignments, checkIfExists);
382         }
383         /// <summary>
384         /// Get class assignment's file path.
385         /// </summary>
386         /// <param name="classID">The associated class ID.</param>
387         /// <param name="numericID">The numeric number ID.</param>
388         /// <param name="paddedID">The padded number ID.</param>
389         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
390         /// <returns>Returns a string of the file path.</returns>
391         public string GetClassAssignmentFilePath(string classID, int
                numericID, out string paddedID, bool checkIfExists)
392         {
393             return this.GetFilePath(this.GetClassComponentDirectory(
                    classID, ClassComponent.Assignments, true), numericID
                    , out paddedID, checkIfExists);
394         }
395         /// <summary>
396         /// Get a 'new' class assignment's file path.
397         /// </summary>
398         /// <param name="classID">The associated class ID.</param>
399         /// <param name="numericID">The numeric number ID.</param>
400         /// <param name="paddedID">The padded number ID.</param>
401         /// <returns>Returns a string of the 'new' file path.</
                returns>
402         public string GetNextClassAssignmentFilePath(string classID,
                out int numericID, out string paddedID)
403         {
```

```
404                 return this.GetNextFilePath(this.
                        GetClassComponentDirectory(classID, ClassComponent.
                        Assignments, true), out numericID, out paddedID);
405             }
406             /// <summary>
407             /// Get the class' assignment answer directory.
408             /// </summary>
409             /// <param name="classID">The associated class ID.</param>
410             /// <param name="aetQID">The associated AETQuestion ID.</
                    param>
411             /// <param name="checkIfExists">If true, and directory is
                    not found, an expception is throw.</param>
412             /// <returns>Returns a string of the path.</returns>
413             public string GetClassAssignmentAnswersDirectory(string
                    classID, int aetQID, bool checkIfExists)
414             {
415                 return this.GetClassSubComponentDirectory(classID, this.
                        PadNumericID(aetQID), ClassComponent.Assignments,
                        checkIfExists);
416             }
417             /// <summary>
418             /// Get a 'new' class assignment answer's file path.
419             /// </summary>
420             /// <param name="classID">The associated class ID.</param>
421             /// <param name="aetQID">The associated AETQuestion ID.</
                    param>
422             /// <param name="numericUserID">The associated numeric user
                    ID.</param>
423             /// <param name="paddedUserID">The associated padded user ID
                    .</param>
424             /// <param name="checkIfExists">If true, and directory is
                    not found, an expception is throw.</param>
425             /// <returns>Returns a string of the 'new' file path.</
                    returns>
426             public string GetClassAssignmentAnswerFilePath(string
                    classID, int aetQID, int numericUserID, out string
                    paddedUserID, bool checkIfExists)
427             {
428                 return this.GetFilePath(this.
                        GetClassSubComponentDirectory(classID, this.
                        PadNumericID(aetQID), ClassComponent.Assignments,
                        true), numericUserID, out paddedUserID, checkIfExists
                        );
429             }
430
431             /// <summary>
432             /// Get the class' exams directory.
433             /// </summary>
434             /// <param name="classID">The associated class ID.</param>
435             /// <param name="checkIfExists">If true, and directory is
                    not found, an expception is throw.</param>
436             /// <returns>Returns a string of the path.</returns>
437             public string GetClassExamsDirectory(string classID, bool
                    checkIfExists)
438             {
```

```
439             return this.GetClassComponentDirectory(classID,
                    ClassComponent.Exams, checkIfExists);
440         }
441         /// <summary>
442         /// Get class exam's file path.
443         /// </summary>
444         /// <param name="classID">The associated class ID.</param>
445         /// <param name="numericID">The numeric number ID.</param>
446         /// <param name="paddedID">The padded number ID.</param>
447         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
448         /// <returns>Returns a string of the file path.</returns>
449         public string GetClassExamFilePath(string classID, int
                numericID, out string paddedID, bool checkIfExists)
450         {
451             return this.GetFilePath(this.GetClassComponentDirectory(
                    classID, ClassComponent.Exams, true), numericID, out
                    paddedID, checkIfExists);
452         }
453         /// <summary>
454         /// Get a 'new' class exam's file path.
455         /// </summary>
456         /// <param name="classID">The associated class ID.</param>
457         /// <param name="numericID">The numeric number ID.</param>
458         /// <param name="paddedID">The padded number ID.</param>
459         /// <returns>Returns a string of the 'new' file path.</
                returns>
460         public string GetNextClassExamFilePath(string classID, out
                int numericID, out string paddedID)
461         {
462             return this.GetNextFilePath(this.
                    GetClassComponentDirectory(classID, ClassComponent.
                    Exams, true), out numericID, out paddedID);
463         }
464         /// <summary>
465         /// Get the class' exam answer directory.
466         /// </summary>
467         /// <param name="classID">The associated class ID.</param>
468         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
469         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
470         /// <returns>Returns a string of the path.</returns>
471         public string GetClassExamAnswersDirectory(string classID,
                int aetQID, bool checkIfExists)
472         {
473             return this.GetClassSubComponentDirectory(classID, this.
                    PadNumericID(aetQID), ClassComponent.Exams,
                    checkIfExists);
474         }
475         /// <summary>
476         /// Get a 'new' class exam answer's file path.
477         /// </summary>
478         /// <param name="classID">The associated class ID.</param>
479         /// <param name="aetQID">The associated AETQuestion ID.</
```

```
param>
        /// <param name="numericUserID">The associated numeric user ID.</param>
        /// <param name="paddedUserID">The associated padded user ID.</param>
        /// <param name="checkIfExists">If true, and directory is not found, an expception is throw.</param>
        /// <returns>Returns a string of the 'new' file path.</returns>
        public string GetClassExamAnswerFilePath(string classID, int aetQID, int numericUserID, out string paddedUserID, bool checkIfExists)
        {
            return this.GetFilePath(this.GetClassSubComponentDirectory(classID, this.PadNumericID(aetQID), ClassComponent.Exams, true), numericUserID, out paddedUserID, checkIfExists);
        }

        /// <summary>
        /// Get the class' tutorials directory.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="checkIfExists">If true, and directory is not found, an expception is throw.</param>
        /// <returns>Returns a string of the path.</returns>
        public string GetClassTutorialsDirectory(string classID, bool checkIfExists)
        {
            return this.GetClassComponentDirectory(classID, ClassComponent.Tutorials, checkIfExists);
        }
        /// <summary>
        /// Get class tutorial's file path.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="numericID">The numeric number ID.</param>
        /// <param name="paddedID">The padded number ID.</param>
        /// <param name="checkIfExists">If true, and directory is not found, an expception is throw.</param>
        /// <returns>Returns a string of the file path.</returns>
        public string GetClassTutorialFilePath(string classID, int numericID, out string paddedID, bool checkIfExists)
        {
            return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Tutorials, true), numericID, out paddedID, checkIfExists);
        }
        /// <summary>
        /// Get a 'new' class tutorial's file path.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="numericID">The numeric number ID.</param>
        /// <param name="paddedID">The padded number ID.</param>
        /// <returns>Returns a string of the 'new' path.</returns>
```

```
518         public string GetNextClassTutorialFilePath(string classID,
                out int numericID, out string paddedID)
519         {
520             return this.GetNextFilePath(this.
                    GetClassComponentDirectory(classID, ClassComponent.
                    Tutorials, true), out numericID, out paddedID);
521         }
522         /// <summary>
523         /// Get the class' tutorial answer directory.
524         /// </summary>
525         /// <param name="classID">The associated class ID.</param>
526         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
527         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
528         /// <returns>Returns a string of the path.</returns>
529         public string GetClassTutorialAnswersDirectory(string
                classID, int aetQID, bool checkIfExists)
530         {
531             return this.GetClassSubComponentDirectory(classID, this.
                    PadNumericID(aetQID), ClassComponent.Tutorials,
                    checkIfExists);
532         }
533         /// <summary>
534         /// Get a 'new' class tutorial answer's file path.
535         /// </summary>
536         /// <param name="classID">The associated class ID.</param>
537         /// <param name="aetQID">The associated AETQuestion ID.</
                param>
538         /// <param name="numericUserID">The associated numeric user
                ID.</param>
539         /// <param name="paddedUserID">The associated padded user ID
                .</param>
540         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
541         /// <returns>Returns a string of the 'new' file path.</
                returns>
542         public string GetClassTutorialAnswerFilePath(string classID,
                 int aetQID, int numericUserID, out string paddedUserID,
                bool checkIfExists)
543         {
544             return this.GetFilePath(this.
                    GetClassSubComponentDirectory(classID, this.
                    PadNumericID(aetQID), ClassComponent.Tutorials, true)
                    , numericUserID, out paddedUserID, checkIfExists);
545         }
546
547         /// <summary>
548         /// Get the class' notes directory.
549         /// </summary>
550         /// <param name="classID">The associated class ID.</param>
551         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
552         /// <returns>Returns a string of the path.</returns>
553         public string GetClassNotesDirectory(string classID, bool
```

```
            checkIfExists)
        {
            return this.GetClassComponentDirectory(classID,
                ClassComponent.Notes, checkIfExists);
        }
        /// <summary>
        /// Get class note's file path.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="numericID">The numeric number ID.</param>
        /// <param name="paddedID">The padded number ID.</param>
        /// <param name="checkIfExists">If true, and directory is
            not found, an expception is throw.</param>
        /// <returns>Returns a string of the file path.</returns>
        public string GetClassNoteFilePath(string classID, int
            numericID, out string paddedID, bool checkIfExists)
        {
            return this.GetFilePath(this.GetClassComponentDirectory(
                classID, ClassComponent.Notes, true), numericID, out
                paddedID, checkIfExists);
        }
        /// <summary>
        /// Get a 'new' class note's file path.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="numericID">The numeric number ID.</param>
        /// <param name="paddedID">The padded number ID.</param>
        /// <returns>Returns a string of the 'new' file path.</
            returns>
        public string GetNextClassNoteFilePath(string classID, out
            int numericID, out string paddedID)
        {
            return this.GetNextFilePath(this.
                GetClassComponentDirectory(classID, ClassComponent.
                Notes, true), out numericID, out paddedID);
        }

        /// <summary>
        /// Get the class' threads directory.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="checkIfExists">If true, and directory is
            not found, an expception is throw.</param>
        /// <returns>Returns a string of the path.</returns>
        public string GetClassThreadsDirectory(string classID, bool
            checkIfExists)
        {
            return this.GetClassComponentDirectory(classID,
                ClassComponent.Forum, checkIfExists);
        }
        /// <summary>
        /// Get class thread's file path.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="numericID">The numeric number ID.</param>
```

```
596         /// <param name="paddedID">The padded number ID.</param>
597         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
598         /// <returns>Returns a string of the file path.</returns>
599         public string GetClassThreadFilePath(string classID, int
                numericID, out string paddedID, bool checkIfExists)
600         {
601             return this.GetFilePath(this.GetClassComponentDirectory(
                    classID, ClassComponent.Forum, true), numericID, out
                    paddedID, checkIfExists);
602         }
603         /// <summary>
604         /// Get a 'new' class thread's file path.
605         /// </summary>
606         /// <param name="classID">The associated class ID.</param>
607         /// <param name="numericID">The numeric number ID.</param>
608         /// <param name="paddedID">The padded number ID.</param>
609         /// <returns>Returns a string of the 'new' path.</returns>
610         public string GetNextClassThreadFilePath(string classID, out
                int numericID, out string paddedID)
611         {
612             return this.GetNextFilePath(this.
                    GetClassComponentDirectory(classID, ClassComponent.
                    Forum, true), out numericID, out paddedID);
613         }
614
615         /// <summary>
616         /// Get the class thread post's directory.
617         /// </summary>
618         /// <param name="classID">The associated class ID.</param>
619         /// <param name="threadID">The associated thread ID.</param>
620         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
621         /// <returns>Returns a string of the path.</returns>
622         public string GetClassThreadPostsDirectory(string classID,
                int threadID, bool checkIfExists)
623         {
624             return this.GetClassSubComponentDirectory(classID, this.
                    PadNumericID(threadID), ClassComponent.Forum,
                    checkIfExists);
625         }
626         /// <summary>
627         /// Get class thread post's file path.
628         /// </summary>
629         /// <param name="classID">The associated class ID.</param>
630         /// <param name="threadID">The associated thread ID.</param>
631         /// <param name="numericID">The numeric number ID.</param>
632         /// <param name="paddedID">The padded number ID.</param>
633         /// <param name="checkIfExists">If true, and directory is
                not found, an expception is throw.</param>
634         /// <returns>Returns a string of the file path.</returns>
635         public string GetClassThreadPostFilePath(string classID, int
                threadID, int numericID, out string paddedID, bool
                checkIfExists)
636         {
```

```
            return this.GetFilePath(this.GetClassSubComponentDirectory(classID, this.PadNumericID(threadID), ClassComponent.Forum, true), numericID, out paddedID, checkIfExists);
        }
        /// <summary>
        /// Get a 'new' class thread post's file path.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="threadID">The associated thread ID.</param>
        /// <param name="numericID">The numeric number ID.</param>
        /// <param name="paddedID">The padded number ID.</param>
        /// <returns>Returns a string of the 'new' file path.</returns>
        public string GetNextClassThreadPostFilePath(string classID, int threadID, out int numericID, out string paddedID)
        {
            return this.GetNextFilePath(this.GetClassSubComponentDirectory(classID, this.PadNumericID(threadID), ClassComponent.Forum, true), out numericID, out paddedID);
        }
        #endregion
    }
}
```

Listing H.100: ErrorAndSecurity.cs

```
using System;
using System.Collections;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A class that holds information about security errors.
    /// </summary>
    [XmlRoot("Security")]
    public sealed class SecurityError
    {
        #region Properties
        /// <summary>
        /// The ID of the error.
        /// </summary>
        private int _securityID;
        [XmlAttribute("securityID")]
        public int SecurityID
        {
            get { return this._securityID; }
            set { this._securityID = value; }
        }

        /// <summary>
```

```
29          /// The ID of the associated user.
30          /// </summary>
31          private int _userID;
32          [XmlElement("UserID")]
33          public int UserID
34          {
35              get { return this._userID; }
36              set { this._userID = value; }
37          }
38
39          /// <summary>
40          /// The value of the associated IP address.
41          /// </summary>
42          private string _ipAddress;
43          [XmlElement("IPAddress")]
44          public string IPAddress
45          {
46              get { return this._ipAddress; }
47              set { this._ipAddress = value; }
48          }
49
50          /// <summary>
51          /// The date and time the error was created.
52          /// </summary>
53          private DateTime _createdDateTime;
54          [XmlAttribute("createdDateTime")]
55          public DateTime CreatedDateTime
56          {
57              get { return this._createdDateTime; }
58              set { this._createdDateTime = value; }
59          }
60          #endregion
61
62          #region Constructors
63          /// <summary>
64          /// Used for XmlSerializer purposes.
65          /// </summary>
66          public SecurityError() {}
67
68          /// <summary>
69          /// Main constructor.
70          /// </summary>
71          /// <param name="dataBridge">The data bridge to the data.</
                param>
72          public SecurityError(DataBridge dataBridge)
73          {
74              this._dataBridge = dataBridge;
75          }
76          #endregion
77
78          #region Data Management Properties
79          /// <summary>
80          /// The DataBridge that connects to the data.
81          /// </summary>
82          private DataBridge _dataBridge = null;
```

```
83          [XmlIgnore()]
84          public DataBridge DataBridge
85          {
86              get { return this._dataBridge; }
87              set { this._dataBridge = value; }
88          }
89          #endregion
90
91          #region Data Management Methods
92          /// <summary>
93          /// Creates an instance of the error as an xml file.
94          /// </summary>
95          /// <returns>Returns the ID of the newly created error.</
                returns>
96          public int Create()
97          {
98              int numericID;
99              string paddedID;
100
101                 // Get next file path
102             string securityErrorFilePath = this.DataBridge.
                    GetNextInstitutionSecurityErrorFilePath(out numericID
                    , out paddedID);
103
104                 // Assign ID
105             this.SecurityID = numericID;
106
107                 // Create the file by serializing it to the xml file
108             XmlSerializer xmlSerializer = new XmlSerializer(this.
                    GetType());
109             XmlTextWriter xmlTextWriter;
110             xmlSerializer.Serialize(xmlTextWriter = new
                    XmlTextWriter(securityErrorFilePath, Encoding.UTF8),
                    this);
111             xmlTextWriter.Close();
112
113             return numericID;
114         }
115
116         /// <summary>
117         /// Delete the specified error ID.
118         /// </summary>
119         public void Delete()
120         {
121             int numericID = this.SecurityID;
122             string paddedID;
123
124                 // Get file path of the ID
125             string securityErrorFilePath = this.DataBridge.
                    GetInstitutionSecurityErrorFilePath(numericID, out
                    paddedID, true);
126
127                 // Delete the file
128             File.Delete(securityErrorFilePath);
129         }
```

```
130         #endregion
131
132         #region Static Data Retrieving Methods
133         /// <summary>
134         /// Find a specific security error.
135         /// </summary>
136         /// <param name="securityID">The ID to find.</param>
137         /// <param name="dataBridge">The bridge to the data.</param>
138         /// <returns>
139         /// Returns a SecurityError object or null if not found.
140         /// </returns>
141         public static SecurityError FindSecurityError(int securityID
                , DataBridge dataBridge)
142         {
143             string paddedID;
144
145             string securityFilePath;
146             try
147             {
148                     // Get file path
149                 securityFilePath = dataBridge.
                        GetInstitutionSecurityErrorFilePath(securityID,
                        out paddedID, true);
150             }
151             catch
152             {
153                     // File not found
154                 return null;
155             }
156
157                 // Deserialize the object from the xml file
158             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    SecurityError));
159             XmlTextReader xmlTextReader;
160             SecurityError newSE = (SecurityError)xmlSerializer.
                    Deserialize(xmlTextReader = new XmlTextReader(
                    securityFilePath));
161             xmlTextReader.Close();
162
163             return newSE;
164         }
165
166         /// <summary>
167         /// Find all security errors.
168         /// </summary>
169         /// <param name="dataBridge">The bridge to the data.</param>
170         /// <returns>
171         /// Returns an array of SecurityError objects or null if not
                found.
172         /// </returns>
173         public static SecurityError[] FindSecurityErrors(DataBridge
                dataBridge)
174         {
175             string xmlDirectory = dataBridge.
                    GetInstitutionSecurityErrorsDirectory(true);
```

```
                // Get files to be deserialized
            string[] securityFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");

            ArrayList securityErrors = new ArrayList(securityFilePaths.Length);
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(SecurityError));
            XmlTextReader xmlTextReader;
                // Deserialize files
            foreach (string securityFilePath in securityFilePaths)
            {
                securityErrors.Add(xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(securityFilePath)));
                xmlTextReader.Close();
            }

            if (securityErrors.Count == 0)
                return null;
            else
                return (SecurityError[])securityErrors.ToArray(typeof(SecurityError));
        }
        #endregion
    }

    /// <summary>
    /// A class that holds information about application errors.
    /// </summary>
    [XmlRoot("Error")]
    public sealed class ApplicationError
    {
        #region Properties
        /// <summary>
        /// The ID of the error.
        /// </summary>
        private int _errorID;
        [XmlAttribute("errorID")]
        public int ErrorID
        {
            get { return this._errorID; }
            set { this._errorID = value; }
        }

        /// <summary>
        /// The date and time the error was created
        /// </summary>
        private DateTime _createdDateTime;
        [XmlAttribute("createdDateTime")]
        public DateTime CreatedDateTime
        {
            get { return this._createdDateTime; }
            set { this._createdDateTime = value; }
```

```
225         }
226
227         /// <summary>
228         /// Information about the error.
229         /// </summary>
230         private string _information;
231         [XmlElement("Information")]
232         public string Information
233         {
234             get { return this._information; }
235             set { this._information = value; }
236         }
237         #endregion
238
239         #region Constructors
240         /// <summary>
241         /// Used for XmlSerializer purposes.
242         /// </summary>
243         public ApplicationError() {}
244
245         /// <summary>
246         /// Main constructor.
247         /// </summary>
248         /// <param name="dataBridge">The data bridge to the data.</
            param>
249         public ApplicationError(DataBridge dataBridge)
250         {
251             this._dataBridge = dataBridge;
252         }
253         #endregion
254
255         #region Data Management Properties
256         /// <summary>
257         /// The DataBridge that connects to the data.
258         /// </summary>
259         private DataBridge _dataBridge = null;
260         [XmlIgnore()]
261         public DataBridge DataBridge
262         {
263             get { return this._dataBridge; }
264             set { this._dataBridge = value; }
265         }
266         #endregion
267
268         #region Data Management Methods
269         /// <summary>
270         /// Creates an instance of the error as an xml file.
271         /// </summary>
272         /// <returns>Returns the ID of the newly created error.</
            returns>
273         public int Create()
274         {
275             int numericID;
276             string paddedID;
277
```

```
                // Get next file path
            string applicationErrorFilePath = this.DataBridge.GetNextInstitutionApplicationErrorFilePath(out numericID, out paddedID);

                // Assign ID
            this.ErrorID = numericID;

                // Create the file by serializing it to the xml file
            XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
            XmlTextWriter xmlTextWriter;
            xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(applicationErrorFilePath, Encoding.UTF8), this);
            xmlTextWriter.Close();

            return numericID;
        }

        /// <summary>
        /// Delete the specified error ID.
        /// </summary>
        public void Delete()
        {
            int numericID = this.ErrorID;
            string paddedID;

                // Get file path of the ID
            string applicationErrorFilePath = this.DataBridge.GetInstitutionApplicationErrorFilePath(numericID, out paddedID, true);

                // Delete the file
            File.Delete(applicationErrorFilePath);
        }
        #endregion

        #region Static Data Retrieving Methods
        /// <summary>
        /// Find a specific application error.
        /// </summary>
        /// <param name="applicationErrorID">The ID to find.</param>
        /// <param name="dataBridge">The bridge to the data.</param>
        /// <returns>
        /// Returns a ApplicationError object or null if not found.
        /// </returns>
        public static ApplicationError FindApplicationError(int applicationErrorID, DataBridge dataBridge)
        {
            string paddedID;

            string securityFilePath;
            try
            {
```

```
                    // Get file path
                securityFilePath = dataBridge.GetInstitutionApplicationErrorFilePath(applicationErrorID, out paddedID, true);
            }
            catch
            {
                    // File not found
                return null;
            }

                // Deserialize the object from the xml file
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(ApplicationError));
            XmlTextReader xmlTextReader;
            ApplicationError newAE = (ApplicationError)xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(securityFilePath));
            xmlTextReader.Close();

            return newAE;
        }

        /// <summary>
        /// Find all application errors.
        /// </summary>
        /// <param name="dataBridge">The bridge to the data.</param>
        /// <returns>
        /// Returns an array of ApplicationError objects or null if not
        /// found.
        /// </returns>
        public static ApplicationError[] FindApplicationErrors(DataBridge dataBridge)
        {
            string xmlDirectory = dataBridge.GetInstitutionApplicationErrorsDirectory(true);

                // Get files to be deserialized
            string[] applicationErrorFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");

            ArrayList applicationErrors = new ArrayList(applicationErrorFilePaths.Length);
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(ApplicationError));
            XmlTextReader xmlTextReader;
                // Deserialize files
            foreach (string applicationErrorFilePath in applicationErrorFilePaths)
            {
                applicationErrors.Add(xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(applicationErrorFilePath)));
                xmlTextReader.Close();
```

```
            }

            if (applicationErrors.Count == 0)
                return null;
            else
                return (ApplicationError[])applicationErrors.ToArray
                    (typeof(ApplicationError));
        }
        #endregion
    }
}
```

Listing H.101: Limitations.cs

```
using System;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// Property limitations for classes, AETQuestion and AETAnswer.
    /// </summary>
    public struct AETLimitations
    {
        public const int Title = 100;
        public const int Description = 30;
        public const int AETType = 20;
        public const int AETQuestionType = 20;
    }

    /// <summary>
    /// Property limitations for class, Role.
    /// </summary>
    public struct RoleLimitations
    {
        public const int RoleID = 50;
    }

    /// <summary>
    /// Property limiations for class, User.
    /// </summary>
    public struct UserLimitations
    {
        public const int Password = 47;
        public const int FirstName = 50;
        public const int MiddleName = 50;
        public const int LastName = 50;
        public const int PhoneNumber = 15;
        public const int EmailAddress = 50;
    }

    /// <summary>
    /// Property limitations for structure, AddressStructure.
    /// </summary>
    public struct AddressLimitations
    {
        public const int Street = 50;
```

```
43          public const int Suburb = 50;
44          public const int State = 50;
45          public const int Postcode = 10;
46          public const int Country = 50;
47      }
48
49      /// <summary>
50      /// Property limitations for class, Class.
51      /// </summary>
52      public struct ClassLimitations
53      {
54          public const int ClassID = 7;
55          public const int ClassName = 50;
56      }
57
58      /// <summary>
59      /// Limitations for filenames used within the OLE.
60      /// </summary>
61      public struct FileLimitations
62      {
63          public const int FilenameLength = 8;
64          public const char PadCharacter = '0';
65      }
66  }
```

Listing H.102: MD5Encryption.cs

```
1   using System;
2   using System.Security.Cryptography;
3   using System.Text;
4
5   namespace OnlineLearningEnvironment
6   {
7       /// <summary>
8       /// This class is used to encrypt strings of data to an MD5 hash
            string
9       /// </summary>
10      public sealed class MD5Encryption
11      {
12          /// <summary>
13          /// Encrypts a string to a MD5 hash in the form of a byte
                array.
14          /// </summary>
15          /// <param name="s">The string to encrypt.</param>
16          /// <returns>Returns an encrypted MD5 hash (byte array).</
                returns>
17          private static byte[] EncryptString(string s)
18          {
19              return new MD5CryptoServiceProvider().ComputeHash(new
                    ASCIIEncoding().GetBytes(s));
20          }
21
22          /// <summary>
23          /// Encrypts a string into a readable MD5 hash string.
24          /// </summary>
25          /// <param name="s">The string to encrypt.</param>
```

```
        /// <returns>Returns a readable MD5 hash string.</returns>
        public static string EncryptStringToByteString(string s)
        {
            return BitConverter.ToString(MD5Encryption.EncryptString
                (s));
        }
    }
}
```

Listing H.103: Post.cs

```
using System;
using System.Collections;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// An abstract class which provides shared properties and
        methods for posts.
    /// </summary>
    public abstract class Post
    {
        #region Properties
        /// <summary>
        /// The unique ID.
        /// </summary>
        private int _postID;
        [XmlAttribute("postID")]
        public int PostID
        {
            get { return this._postID; }
            set { this._postID = value; }
        }

        protected string _classID;
        protected int _threadID;
        protected string _title;
        protected bool _withhold = false;

        /// <summary>
        /// The name of the creator.
        /// </summary>
        protected string _creatorName;
        [XmlAttribute("creatorName")]
        public string CreatorName
        {
            get { return this._creatorName; }
            set { this._creatorName = value; }
        }

        /// <summary>
        /// The date & time created.
```

```
45          /// </summary>
46          private DateTime _createdDateTime;
47          [XmlAttribute("createdDateTime")]
48          public DateTime CreatedDateTime
49          {
50              get { return this._createdDateTime; }
51              set { this._createdDateTime = value; }
52          }
53
54          /// <summary>
55          /// The date & time updated.
56          /// </summary>
57          private DateTime _updatedDateTime;
58          [XmlAttribute("updatedDateTime")]
59          public DateTime UpdatedDateTime
60          {
61              get { return this._updatedDateTime; }
62              set { this._updatedDateTime = value; }
63          }
64
65          /// <summary>
66          /// The associated information.
67          /// </summary>
68          private string _information;
69          [XmlElement("Information")]
70          public string Information
71          {
72              get { return this._information; }
73              set { this._information = value; }
74          }
75
76          /// <summary>
77          /// An array of attachment filenames.
78          /// </summary>
79          private string[] _attachmentFilenames = null;
80          [XmlArray("Attachments")]
81          [XmlArrayItem("Filename")]
82          public string[] AttachmentFilenames
83          {
84              get { return this._attachmentFilenames; }
85              set { this._attachmentFilenames = value; }
86          }
87          #endregion
88
89          #region Constructors
90          /// <summary>
91          /// Used for XmlSerializer purposes.
92          /// </summary>
93          public Post() {}
94          /// <summary>
95          /// Abstract constructor.
96          /// </summary>
97          /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
98          public Post(DataBridge dataBridge)
```

```
99          {
100             this._createdDateTime = DateTime.Now;
101             this._dataBridge = dataBridge;
102         }
103         #endregion
104
105         #region Data Management Properties
106         /// <summary>
107         /// The DataBridge that connects to the data.
108         /// </summary>
109         private DataBridge _dataBridge = null;
110         [XmlIgnore()]
111         public DataBridge DataBridge
112         {
113             get { return this._dataBridge; }
114             set { this._dataBridge = value; }
115         }
116         #endregion
117
118         #region Data Management Methods
119         /// <summary>
120         /// Uses current instance's properties and stores the data.
121         /// </summary>
122         public int Create()
123         {
124             string postFilePath = null;
125             int numericID = 0;
126             string paddedID = null;
127
128                 // Get filepath
129             if (this is InstitutionAnnouncement)
130             {
131                     // Is an InstitutionAnnouncement
132                 DataBridge newDataBridge = this.DataBridge as
                        DataBridge;
133                 postFilePath = newDataBridge.
                        GetNextInstitutionAnnouncementFilePath(out
                        numericID, out paddedID);
134             }
135             else
136             {
137                 DataBridge newDataBridge = this.DataBridge as
                        DataBridge;
138                 if (this is ClassAnnouncement)
139                     postFilePath = newDataBridge.
                            GetNextClassAnnouncementFilePath(this.
                            _classID, out numericID, out paddedID);
140                 else if (this is ClassNote)
141                     postFilePath = newDataBridge.
                            GetNextClassNoteFilePath(this._classID, out
                            numericID, out paddedID);
142                 else if (this is ClassThreadPost)
143                     postFilePath = newDataBridge.
                            GetNextClassThreadPostFilePath(this._classID,
                             this._threadID, out numericID, out paddedID)
```

```
                        ;
            }

            this.PostID = numericID;

            XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
            XmlTextWriter xmlTextWriter;
                // Serialise
            xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(postFilePath, Encoding.UTF8), this);
            xmlTextWriter.Close();

            return this.PostID;
        }

        /// <summary>
        /// Uses current instance's properties and re-stores the data.
        /// </summary>
        public void Alter()
        {
            string postFilePath = null;
            string paddedID;

                // Get filepath
            if (this is InstitutionAnnouncement)
            {
                    // Is InstitutionAnnouncement
                DataBridge newDataBridge = this.DataBridge as DataBridge;
                postFilePath = newDataBridge.GetInstitutionAnnouncementFilePath(this.PostID, out paddedID, true);
            }
            else
            {
                DataBridge newDataBridge = this.DataBridge as DataBridge;
                if (this is ClassAnnouncement)
                    postFilePath = newDataBridge.GetClassAnnouncementFilePath(this._classID, this.PostID, out paddedID, true);
                else if (this is ClassNote)
                    postFilePath = newDataBridge.GetClassNoteFilePath(this._classID, this.PostID, out paddedID, true);
                else if (this is ClassThreadPost)
                    postFilePath = newDataBridge.GetClassThreadPostFilePath(this._classID, this._threadID, this.PostID, out paddedID, true);
            }

            XmlSerializer xmlSerializer = new XmlSerializer(this.
```

```
                GetType());
            XmlTextWriter xmlTextWriter;
                // Serialise
            xmlSerializer.Serialize(xmlTextWriter = new
                XmlTextWriter(postFilePath, Encoding.UTF8), this);
            xmlTextWriter.Close();
        }

        /// <summary>
        /// Uses the current instance's properties and deletes the
            stored data.
        /// </summary>
        public void Delete()
        {
            string postFilePath = null;
            string paddedID;

                // Get filepath
            if (this is InstitutionAnnouncement)
            {
                DataBridge newDataBridge = this.DataBridge as
                    DataBridge;
                postFilePath = newDataBridge.
                    GetInstitutionAnnouncementFilePath(this.PostID,
                    out paddedID, true);
            }
            else
            {
                DataBridge newDataBridge = this.DataBridge as
                    DataBridge;
                if (this is ClassAnnouncement)
                    postFilePath = newDataBridge.
                        GetClassAnnouncementFilePath(this._classID,
                        this.PostID, out paddedID, true);
                else if (this is ClassNote)
                    postFilePath = newDataBridge.
                        GetClassNoteFilePath(this._classID, this.
                        PostID, out paddedID, true);
                else if (this is ClassThreadPost)
                    postFilePath = newDataBridge.
                        GetClassThreadPostFilePath(this._classID,
                        this._threadID, this.PostID, out paddedID,
                        true);
            }

                // Delete the file
            File.Delete(postFilePath);
        }
        #endregion
    }

    /// <summary>
    /// A sealed class that contains properties and methods of a
        institution announcement.
    /// </summary>
```

```
    [XmlRoot("Announcement")]
    public sealed class InstitutionAnnouncement : Post
    {
        #region Properties
        /// <summary>
        /// The associated title.
        /// </summary>
        [XmlElement("Title")]
        public string Title
        {
            get { return base._title; }
            set { base._title = value; }
        }
        #endregion

        #region Constructors
        /// <summary>
        /// Used for XmlSerializer purposes.
        /// </summary>
        public InstitutionAnnouncement() {}
        /// <summary>
        /// Main Constructor.
        /// </summary>
        /// <param name="dataBridge">The DataBridge that connects to the data.</param>
        public InstitutionAnnouncement(DataBridge dataBridge) : base(dataBridge) {}
        #endregion

        #region Static Data Retrieving Methods
        /// <summary>
        /// Find a specific institution announcement.
        /// </summary>
        /// <param name="postID">The associated post ID.</param>
        /// <param name="dataBridge">The DataBridge that connects to the data.</param>
        /// <returns>Returns a InstitutionAnnouncement class. Null if not found.</returns>
        public static InstitutionAnnouncement FindInstitutionAnnouncement(int postID, DataBridge dataBridge)
        {
            string paddedPostID;

            string postFilePath;
            try
            {
                postFilePath = dataBridge.GetInstitutionAnnouncementFilePath(postID, out paddedPostID, true);
            }
            catch
            {
                    // Not found
                return null;
```

```
271                 }
272
273                 XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                        InstitutionAnnouncement));
274                 XmlTextReader xmlTextReader;
275                     // Deserialise
276                 InstitutionAnnouncement newIA = (InstitutionAnnouncement
                        )xmlSerializer.Deserialize(xmlTextReader = new
                        XmlTextReader(postFilePath));
277                 xmlTextReader.Close();
278
279                 newIA.DataBridge = dataBridge;
280
281                 return newIA;
282             }
283
284             /// <summary>
285             /// Finds all instititition announcements.
286             /// </summary>
287             /// <param name="dataBridge">The DataBridge that connects to
                    the data.</param>
288             /// <returns>Returns an array of InstitutionAnnouncement
                    classes. Null if none found.</returns>
289             public static InstitutionAnnouncement[]
                    FindInstitutionAnnouncements(DataBridge dataBridge)
290             {
291                 string xmlDirectory = dataBridge.
                        GetInstitutionAnnouncementsDirectory(true);
292
293                     // Get files
294                 string[] postFilePaths = Directory.GetFiles(xmlDirectory
                        , "*.xml");
295
296                 ArrayList posts = new ArrayList(postFilePaths.Length);
297                 XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                        InstitutionAnnouncement));
298                 XmlTextReader xmlTextReader;
299                     // Deserialise
300                 foreach (string postFilePath in postFilePaths)
301                 {
302                     InstitutionAnnouncement tempPost = xmlSerializer.
                            Deserialize(xmlTextReader = new XmlTextReader(
                            postFilePath)) as InstitutionAnnouncement;
303                     tempPost.DataBridge = dataBridge;
304                     posts.Add(tempPost);
305                     xmlTextReader.Close();
306                 }
307
308                     // Any found?
309                 if (posts.Count == 0)
310                     return null;
311                 else
312                     return (InstitutionAnnouncement[])posts.ToArray(
                            typeof(InstitutionAnnouncement));
313             }
```

```
        #endregion
    }

    /// <summary>
    /// A sealed class that contains properties and methods of a
        class announcement.
    /// </summary>
    [XmlRoot("Announcement")]
    public sealed class ClassAnnouncement : Post
    {
        #region Properties
        /// <summary>
        /// The associated class ID.
        /// </summary>
        [XmlAttribute("classID")]
        public string ClassID
        {
            get { return base._classID; }
            set { base._classID = value; }
        }

        /// <summary>
        /// The associated title.
        /// </summary>
        [XmlElement("Title")]
        public string Title
        {
            get { return base._title; }
            set { base._title = value; }
        }
        #endregion

        #region Constructors
        /// <summary>
        /// Used for XmlSerializer purposes.
        /// </summary>
        public ClassAnnouncement() {}
        /// <summary>
        /// Main Constructor.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="dataBridge">The DataBridge that connects to
            the data.</param>
        public ClassAnnouncement(string classID, DataBridge
            dataBridge) : base(dataBridge)
        {
            this.ClassID = classID;
        }
        #endregion

        #region Static Data Retrieving Methods
        /// <summary>
        /// Find a specific class announcement.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
```

```
366         /// <param name="postID">The associated post ID.</param>
367         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
368         /// <returns>Returns a ClassAnnouncement class. Null if not
                found.</returns>
369         public static ClassAnnouncement FindClassAnnouncement(string
                classID, int postID, DataBridge dataBridge)
370         {
371             string paddedPostID;
372
373             string postFilePath;
374             try
375             {
376                 postFilePath = dataBridge.
                        GetClassAnnouncementFilePath(classID, postID, out
                            paddedPostID, true);
377             }
378             catch
379             {
380                     // Not found
381                 return null;
382             }
383
384             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    ClassAnnouncement));
385             XmlTextReader xmlTextReader;
386                 // Deserialise
387             ClassAnnouncement newCA = (ClassAnnouncement)
                    xmlSerializer.Deserialize(xmlTextReader = new
                    XmlTextReader(postFilePath));
388             xmlTextReader.Close();
389
390             newCA.DataBridge = dataBridge;
391
392             return newCA;
393         }
394
395         /// <summary>
396         /// Finds all class announcements for a specific class.
397         /// </summary>
398         /// <param name="classID">The associated class ID.</param>
399         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
400         /// <returns>Returns an array of ClassAnnouncement classes.
                Null if none found.</returns>
401         public static ClassAnnouncement[] FindClassAnnouncements(
                string classID, DataBridge dataBridge)
402         {
403             string xmlDirectory = dataBridge.
                    GetClassAnnouncementsDirectory(classID, true);
404
405                 // Get files
406             string[] postFilePaths = Directory.GetFiles(xmlDirectory
                    , "*.xml");
407
```

```
408                 ArrayList posts = new ArrayList(postFilePaths.Length);
409                 XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                        ClassAnnouncement));
410                 XmlTextReader xmlTextReader;
411                     // Deserialise
412                 foreach (string postFilePath in postFilePaths)
413                 {
414                     ClassAnnouncement tempPost = xmlSerializer.
                            Deserialize(xmlTextReader = new XmlTextReader(
                            postFilePath)) as ClassAnnouncement;
415                     tempPost.DataBridge = dataBridge;
416                     posts.Add(tempPost);
417                     xmlTextReader.Close();
418                 }
419
420                     // Any found?
421                 if (posts.Count == 0)
422                     return null;
423                 else
424                     return (ClassAnnouncement[])posts.ToArray(typeof(
                            ClassAnnouncement));
425             }
426             #endregion
427         }
428
429         /// <summary>
430         /// A sealed class that contains properties and methods of a
                class note.
431         /// </summary>
432         [XmlRoot("Note")]
433         public sealed class ClassNote : Post
434         {
435             #region Properties
436             /// <summary>
437             /// The associated title.
438             /// </summary>
439             [XmlElement("Title")]
440             public string Title
441             {
442                 get { return base._title; }
443                 set { base._title = value; }
444             }
445
446             /// <summary>
447             /// The associated class ID.
448             /// </summary>
449             [XmlAttribute("classID")]
450             public string ClassID
451             {
452                 get { return base._classID; }
453                 set { base._classID = value; }
454             }
455
456             /// <summary>
457             /// The associated description.
```

```
458         /// </summary>
459         private string _description;
460         [XmlElement("Description")]
461         public string Description
462         {
463             get { return this._description; }
464             set { this._description = value; }
465         }
466
467         #endregion
468
469         #region Constructors
470         /// <summary>
471         /// Used for XmlSerializer purposes.
472         /// </summary>
473         public ClassNote() {}
474         /// <summary>
475         /// Main Constructor.
476         /// </summary>
477         /// <param name="classID">The associated class ID.</param>
478         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
479         public ClassNote(string classID, DataBridge dataBridge) :
                base(dataBridge)
480         {
481             this.ClassID = classID;
482         }
483         #endregion
484
485         #region Static Data Retrieving Methods
486         /// <summary>
487         /// Find a specific class note.
488         /// </summary>
489         /// <param name="classID">The associated class ID.</param>
490         /// <param name="postID">The associated post ID.</param>
491         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
492         /// <returns>Returns a ClassNote class. Null if not found.</
                returns>
493         public static ClassNote FindClassNote(string classID, int
                postID, DataBridge dataBridge)
494         {
495             string paddedPostID;
496
497             string postFilePath;
498             try
499             {
500                 postFilePath = dataBridge.GetClassNoteFilePath(
                        classID, postID, out paddedPostID, true);
501             }
502             catch
503             {
504                     // Not found
505                 return null;
506             }
```

```
507
508             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    ClassNote));
509             XmlTextReader xmlTextReader;
510                 // Deserialise
511             ClassNote newCN = (ClassNote)xmlSerializer.Deserialize(
                    xmlTextReader = new XmlTextReader(postFilePath));
512             xmlTextReader.Close();
513
514             newCN.DataBridge = dataBridge;
515
516             return newCN;
517         }
518
519         /// <summary>
520         /// Finds all class notes for a specific class.
521         /// </summary>
522         /// <param name="classID">The associated class ID.</param>
523         /// <param name="dataBridge">The DataBridge that connects to
                 the data.</param>
524         /// <returns>Returns an array of ClassNote classes. Null if
                none found.</returns>
525         public static ClassNote[] FindClassNotes(string classID,
                DataBridge dataBridge)
526         {
527             string xmlDirectory = dataBridge.GetClassNotesDirectory(
                    classID, true);
528
529                 // Get files
530             string[] postFilePaths = Directory.GetFiles(xmlDirectory
                    , "*.xml");
531
532             ArrayList posts = new ArrayList(postFilePaths.Length);
533             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    ClassNote));
534             XmlTextReader xmlTextReader;
535                 // Deserialise
536             foreach (string postFilePath in postFilePaths)
537             {
538                 ClassNote tempPost = xmlSerializer.Deserialize(
                        xmlTextReader = new XmlTextReader(postFilePath))
                        as ClassNote;
539                 tempPost.DataBridge = dataBridge;
540                 posts.Add(tempPost);
541                 xmlTextReader.Close();
542             }
543
544                 // Any found?
545             if (posts.Count == 0)
546                 return null;
547             else
548                 return (ClassNote[])posts.ToArray(typeof(ClassNote))
                        ;
549         }
550         #endregion
```

```
    }

    /// <summary>
    /// A sealed class that contains properties and methods of a class thread post.
    /// </summary>
    [XmlRoot("ThreadPost")]
    public sealed class ClassThreadPost : Post
    {
        #region Properties
        /// <summary>
        /// The associated class ID.
        /// </summary>
        [XmlAttribute("classID")]
        public string ClassID
        {
            get { return base._classID; }
            set { base._classID = value; }
        }

        /// <summary>
        /// The associated thread ID.
        /// </summary>
        [XmlAttribute("threadID")]
        public int ThreadID
        {
            get { return base._threadID; }
            set { base._threadID = value; }
        }

        /// <summary>
        /// Whether or not to show this thread post.
        /// </summary>
        [XmlAttribute("withhold")]
        public bool Withhold
        {
            get { return base._withhold; }
            set { base._withhold = value; }
        }
        #endregion

        #region Constructors
        /// <summary>
        /// Used for XmlSerializer purposes.
        /// </summary>
        public ClassThreadPost() {}
        /// <summary>
        /// Main Constructor.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="threadID">The associated thread ID.</param>
        /// <param name="dataBridge">The DataBridge that connects to the data.</param>
        public ClassThreadPost(string classID, int threadID, DataBridge dataBridge) : base(dataBridge)
```

```
603         {
604             this.ClassID = classID;
605             this.ThreadID = threadID;
606         }
607         #endregion
608
609         #region Static Data Retrieving Methods
610         /// <summary>
611         /// Find a specific class thread post.
612         /// </summary>
613         /// <param name="classID">The associated class ID.</param>
614         /// <param name="threadID">The associated threadID.</param>
615         /// <param name="postID">The associated post ID.</param>
616         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
617         /// <returns>Returns a ClassThreadPost class. Null if not
                found.</returns>
618         public static ClassThreadPost FindClassThreadPost(string
                classID, int threadID, int postID, DataBridge dataBridge)
619         {
620             string paddedPostID;
621
622             string postFilePath;
623             try
624             {
625                 postFilePath = dataBridge.GetClassThreadPostFilePath
                        (classID, threadID, postID, out paddedPostID,
                        true);
626             }
627             catch
628             {
629                     // Not found
630                 return null;
631             }
632
633             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    ClassThreadPost));
634             XmlTextReader xmlTextReader;
635                 // Deserialise
636             ClassThreadPost newCTP = (ClassThreadPost)xmlSerializer.
                    Deserialize(xmlTextReader = new XmlTextReader(
                    postFilePath));
637             xmlTextReader.Close();
638
639             newCTP.DataBridge = dataBridge;
640
641             return newCTP;
642         }
643
644         /// <summary>
645         /// Finds all class thread posts for a specific class.
646         /// </summary>
647         /// <param name="classID">The associated class ID.</param>
648         /// <param name="dataBridge">The DataBridge that connects to
                the data.</param>
```

```
        /// <returns>Returns an array of ClassThreadPost classes. Null if none found.</returns>
        public static ClassThreadPost[] FindClassThreadPosts(string classID, int threadID, DataBridge dataBridge)
        {
            string xmlDirectory = dataBridge.GetClassThreadPostsDirectory(classID, threadID, true);

                // Get files
            string[] postFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");

            ArrayList posts = new ArrayList(postFilePaths.Length);
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassThreadPost));
            XmlTextReader xmlTextReader;
                // Deserialise
            foreach (string postFilePath in postFilePaths)
            {
                ClassThreadPost tempPost = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath)) as ClassThreadPost;
                tempPost.DataBridge = dataBridge;
                posts.Add(tempPost);
                xmlTextReader.Close();
            }

                // Any found?
            if (posts.Count == 0)
                return null;
            else
                return (ClassThreadPost[])posts.ToArray(typeof(ClassThreadPost));
        }
        #endregion
    }
}
```

Listing H.104: Role.cs

```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// An enumeration which holds which roles exist within the OLE.
    /// </summary>
    public enum RoleID
```

```
16      {
17          /// <summary>
18          /// The highest authenticated users within the OLE.
19          /// </summary>
20          Administrators,
21
22          /// <summary>
23          /// Users that controls Students within the OLE.
24          /// </summary>
25          Instructors,
26
27          /// <summary>
28          /// The lowest authenticated users within the OLE.
29          /// </summary>
30          Students
31      }
32
33      /// <summary>
34      /// A class which holds properties and methods that define a
            role.
35      /// </summary>
36      public sealed class Role
37      {
38          #region Properties
39          /// <summary>
40          /// A string ID that defines a role.
41          /// </summary>
42          private RoleID _roleID;
43          public RoleID RoleID
44          {
45              get { return this._roleID; }
46              set { this._roleID = value; }
47          }
48          #endregion
49
50          #region Constructors
51          /// <summary>
52          /// Used for XmlSerializer purposes.
53          /// </summary>
54          public Role() {}
55
56          /// <summary>
57          /// Main constructor.
58          /// </summary>
59          /// <param name="roleID">The string ID of a role.</param>
60          /// <param name="dataBridge">The data bridge to the data.</
                param>
61          public Role(RoleID roleID, DataBridge dataBridge)
62          {
63              this._roleID = roleID;
64              this._dataBridge = dataBridge;
65          }
66          #endregion
67
68          #region Data Management Properties
```

```
69          /// <summary>
70          /// The DataBridge that connects to the data.
71          /// </summary>
72          private DataBridge _dataBridge = null;
73          [XmlIgnore()]
74          public DataBridge DataBridge
75          {
76              get { return this._dataBridge; }
77              set { this._dataBridge = value; }
78          }
79          #endregion
80
81          #region Data Retrieving Methods
82          /// <summary>
83          /// Assign a specific user to a role within a specific class
                .
84          /// </summary>
85          /// <param name="userID">The ID of the user.</param>
86          /// <param name="classID">The ID of the class.</param>
87          public void AssignUserToClass(int userID, string classID)
88          {
89              SqlCommand sqlCommand = new SqlCommand("
                    AssignUserToClass", this.DataBridge.SqlConnection);
90              sqlCommand.CommandType = CommandType.StoredProcedure;
91              sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
92              sqlCommand.Parameters["@userID"].Value = userID;
93              sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar,
                    RoleLimitations.RoleID);
94              sqlCommand.Parameters["@roleID"].Value = this.RoleID.
                    ToString();
95              sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
96              sqlCommand.Parameters["@classID"].Value = classID;
97              this.DataBridge.SqlConnection.Open();
98              // Execute the stored procedure
99              sqlCommand.ExecuteNonQuery();
100             this.DataBridge.SqlConnection.Close();
101         }
102
103         /// <summary>
104         /// Unassign a specific user from a role within a specific
                class.
105         /// </summary>
106         /// <param name="userID">The ID of the user.</param>
107         /// <param name="classID">The ID of the class.</param>
108         public void UnassignUserToClass(int userID, string classID)
109         {
110             SqlCommand sqlCommand = new SqlCommand("
                    UnassignUserToClass", this.DataBridge.SqlConnection);
111             sqlCommand.CommandType = CommandType.StoredProcedure;
112             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
113             sqlCommand.Parameters["@userID"].Value = userID;
114             sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar,
                    RoleLimitations.RoleID);
115             sqlCommand.Parameters["@roleID"].Value = this.RoleID.
```

```
                ToString();
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                ClassLimitations.ClassID);
            sqlCommand.Parameters["@classID"].Value = classID;
            this.DataBridge.SqlConnection.Open();
                // Execute the stored procedure
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();
        }

        /// <summary>
        /// Assign a specific user to a role.
        /// </summary>
        /// <param name="userID">The ID of the user.</param>
        public void AssignUser(int userID)
        {
            SqlCommand sqlCommand = new SqlCommand("AssignUser",
                this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
            sqlCommand.Parameters["@userID"].Value = userID;
            sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar,
                RoleLimitations.RoleID);
            sqlCommand.Parameters["@roleID"].Value = this.RoleID.
                ToString();
            this.DataBridge.SqlConnection.Open();
                // Execute the stored procedure
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();
        }

        /// <summary>
        /// Unassign a specific user to a role.
        /// </summary>
        /// <param name="userID">The ID of the user.</param>
        public void UnassignUser(int userID)
        {
            SqlCommand sqlCommand = new SqlCommand("UnassignUser",
                this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
            sqlCommand.Parameters["@userID"].Value = userID;
            sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar,
                RoleLimitations.RoleID);
            sqlCommand.Parameters["@roleID"].Value = this.RoleID.
                ToString();
            this.DataBridge.SqlConnection.Open();
                // Execute the stored procedure
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();
        }
        #endregion

        #region Static Data Retrieving Methods
        /// <summary>
```

```
163         /// Find all roles associated with a user's ID.
164         /// </summary>
165         /// <param name="userID">The ID of the user.</param>
166         /// <param name="dataBridge">The bridge to the data.</param>
167         /// <returns>Returns a Role object array of the associated
                roles.</returns>
168         public static Role[] FindRoles(int userID, DataBridge
                dataBridge)
169         {
170             // Get roles
171             RoleID[] rolesIDs = FindRolesIDs(userID, dataBridge);
172             if (rolesIDs == null)
173                 return null;
174
175             ArrayList roles = new ArrayList(rolesIDs.Length);
176             // Add Roles to an arraylist
177             foreach (RoleID roleID in rolesIDs)
178             {
179                 roles.Add(new Role(roleID, dataBridge));
180             }
181
182             if (roles.Count == 0)
183                 return null;
184             else
185                 return (Role[])roles.ToArray(typeof(Role));
186         }
187
188         /// <summary>
189         /// Find all roles associated with a user's ID.
190         /// </summary>
191         /// <param name="userID">The ID of the user.</param>
192         /// <param name="dataBridge">The bridge to the data.</param>
193         /// <returns>Returns a RoleID enum array of the associated
                roles.</returns>
194         public static RoleID[] FindRolesIDs(int userID, DataBridge
                dataBridge)
195         {
196             ArrayList rolesArray = new ArrayList();
197
198             SqlCommand sqlCommand = new SqlCommand("FindRolesIDs",
                    dataBridge.SqlConnection);
199             sqlCommand.CommandType = CommandType.StoredProcedure;
200             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
201             sqlCommand.Parameters["@userID"].Value = userID;
202             dataBridge.SqlConnection.Open();
203             // Execute stored procedure and retrieve data
204             SqlDataReader sqlDataReader = sqlCommand.ExecuteReader()
                    ;
205             // Read data from DataReader and add to an arraylist
206             while (sqlDataReader.Read())
207             {
208                 switch (sqlDataReader["roleID"] as string)
209                 {
210                     case "Administrators":
211                         rolesArray.Add(RoleID.Administrators);
```

```
                        break;
                    case "Instructors":
                        rolesArray.Add(RoleID.Instructors);
                        break;
                    case "Students":
                        rolesArray.Add(RoleID.Students);
                        break;
                }
            }
            dataBridge.SqlConnection.Close();

            if (rolesArray.Count == 0)
                return null;
            else
                return (RoleID[])rolesArray.ToArray(typeof(RoleID));
        }
        #endregion
    }
}
```

Listing H.105: Thread.cs

```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A class which holds properties and methods that define a
        thread.
    /// </summary>
    [XmlRoot("Thread")]
    public sealed class Thread
    {
        #region Properties
        /// <summary>
        /// The ID associated to the thread.
        /// </summary>
        private int _threadID;
        [XmlAttribute("threadID")]
        public int ThreadID
        {
            get { return this._threadID; }
            set { this._threadID = value; }
        }

        /// <summary>
        /// The ID of the class associated to the thread.
        /// </summary>
        private string _classID;
```

```
34          [XmlAttribute("classID")]
35          public string ClassID
36          {
37              get { return this._classID; }
38              set { this._classID = value; }
39          }
40
41          /// <summary>
42          /// The title of the thread.
43          /// </summary>
44          private string _title;
45          [XmlElement("Title")]
46          public string Title
47          {
48              get { return this._title; }
49              set { this._title = value; }
50          }
51
52          /// <summary>
53          /// The thread creator's full name.
54          /// </summary>
55          private string _creatorName;
56          [XmlAttribute("creatorName")]
57          public string CreatorName
58          {
59              get { return this._creatorName; }
60              set { this._creatorName = value; }
61          }
62
63          /// <summary>
64          /// The date and time the thread was created.
65          /// </summary>
66          private DateTime _createdDateTime;
67          [XmlAttribute("createdDateTime")]
68          public DateTime CreatedDateTime
69          {
70              get { return this._createdDateTime; }
71              set { this._createdDateTime = value; }
72          }
73          #endregion
74
75          #region Constructors
76          /// <summary>
77          /// Used for XmlSerializer purposes.
78          /// </summary>
79          public Thread() {}
80
81          /// <summary>
82          /// Main constructor.
83          /// </summary>
84          /// <param name="dataBridge">The bridge to the data.</param>
85          public Thread(DataBridge dataBridge)
86          {
87              this._dataBridge = dataBridge;
88          }
```

```
89          #endregion
90
91          #region Data Management Properties
92          /// <summary>
93          /// The DataBridge that connects to the data.
94          /// </summary>
95          private DataBridge _dataBridge = null;
96          [XmlIgnore()]
97          public DataBridge DataBridge
98          {
99              get { return this._dataBridge; }
100             set { this._dataBridge = value; }
101         }
102         #endregion
103
104         #region Data Management Methods
105         /// <summary>
106         /// Creates an xml file entry of a thread.
107         /// </summary>
108         /// <returns>Returns the numeric ID of the new thread.</
                returns>
109         public int Create()
110         {
111             int numericThreadID;
112             string paddedThreadID;
113
114                 // Get next filepath and thread's directory ready
                        for storage
115             string threadFilePath = this.DataBridge.
                    GetNextClassThreadFilePath(this.ClassID, out
                    numericThreadID, out paddedThreadID);
116             string threadPostsDirectory = this.DataBridge.
                    GetClassThreadPostsDirectory(this.ClassID,
                    numericThreadID, false);
117
118                 // Assign ID
119             this.ThreadID = numericThreadID;
120
121                 // Serialize thread to the xml file
122             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    Thread));
123             XmlTextWriter xmlTextWriter;
124             xmlSerializer.Serialize(xmlTextWriter = new
                    XmlTextWriter(threadFilePath, Encoding.UTF8), this);
125             xmlTextWriter.Close();
126
127                 // Create directory for posts
128             Directory.CreateDirectory(threadPostsDirectory);
129
130             return numericThreadID;
131         }
132
133         /// <summary>
134         /// Alters the thread's xml file information.
135         /// </summary>
```

```
136         public void Alter()
137         {
138             string paddedThreadID;
139
140                 // Get the filepath of the thread
141             string threadFilePath = this.DataBridge.
                    GetClassThreadFilePath(this.ClassID, this.ThreadID,
                    out paddedThreadID, true);
142
143                 // Serialize the thread back to its xml file with
                        the changes
144             XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                    Thread));
145             XmlTextWriter xmlTextWriter;
146             xmlSerializer.Serialize(xmlTextWriter = new
                    XmlTextWriter(threadFilePath, Encoding.UTF8), this);
147             xmlTextWriter.Close();
148         }
149
150         public void Delete()
151         {
152             string paddedThreadID;
153
154                 // Get the filepath
155             string threadFilePath = this.DataBridge.
                    GetClassThreadFilePath(this.ClassID, this.ThreadID,
                    out paddedThreadID, true);
156
157                 // Delete the xml file
158             File.Delete(threadFilePath);
159         }
160         #endregion
161
162         #region Data Retrieving Methods
163         /// <summary>
164         /// Creates a subscription to a thread for a user.
165         /// </summary>
166         /// <param name="classID">The associated class ID.</param>
167         /// <param name="userID">The associated user ID.</param>
168         public void CreateThreadSubscription(string classID, int
                userID)
169         {
170             SqlCommand sqlCommand = new SqlCommand("CreateThreadSub"
                    , this.DataBridge.SqlConnection);
171             sqlCommand.CommandType = CommandType.StoredProcedure;
172             sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                    ClassLimitations.ClassID);
173             sqlCommand.Parameters["@userID"].Value = classID;
174             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
175             sqlCommand.Parameters["@userID"].Value = userID;
176             sqlCommand.Parameters.Add("@threadID", SqlDbType.Int);
177             sqlCommand.Parameters["@threadID"].Value = this.ThreadID
                    ;
178             this.DataBridge.SqlConnection.Open();
179                 // Execute SQL Stored Procedure
```

```
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();
        }

        /// <summary>
        /// Deletes a subscription to a thread for a user.
        /// </summary>
        /// <param name="classID">The associated class ID.</param>
        /// <param name="userID">The associated user ID.</param>
        public void DeleteThreadSubscription(string classID, int
            userID)
        {
            SqlCommand sqlCommand = new SqlCommand("DeleteThreadSub"
                , this.DataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
                ClassLimitations.ClassID);
            sqlCommand.Parameters["@userID"].Value = classID;
            sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
            sqlCommand.Parameters["@userID"].Value = userID;
            sqlCommand.Parameters.Add("@threadID", SqlDbType.Int);
            sqlCommand.Parameters["@threadID"].Value = this.ThreadID
                ;
            this.DataBridge.SqlConnection.Open();
                // Execute SQL Stored Procedure
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();
        }
        #endregion

        #region Static Data Retrieving Methods
        /// <summary>
        /// Find a specific thread that belongs within a specific
            class.
        /// </summary>
        /// <param name="classID">The associated class' ID.</param>
        /// <param name="threadID">The ID of the thread.</param>
        /// <param name="dataBridge">The bridge to the data.</param>
        /// <returns></returns>
        public static Thread FindThread(string classID, int threadID
            , DataBridge dataBridge)
        {
            string paddedThreadID;

            string threadFilePath;
            try
            {
                    // Get the filepath of the xml file
                threadFilePath = dataBridge.GetClassThreadFilePath(
                    classID, threadID, out paddedThreadID, true);
            }
            catch
            {
                    // Does not exist
                return null;
```

```
            }

                // Deserialize the xml file to a Thread object
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(Thread));
            XmlTextReader xmlTextReader;
            Thread newThread = (Thread)xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(threadFilePath));
            xmlTextReader.Close();

            if (newThread == null)
                return null;
            else
                return newThread;
        }

        /// <summary>
        /// Finds all threads that belong to a specific class ID.
        /// </summary>
        /// <param name="classID">The ID of the associated class.</param>
        /// <param name="dataBridge">The bridge to the data.</param>
        /// <returns>Returns an array of all threads associated with the class.</returns>
        public static Thread[] FindAllThreads(string classID, DataBridge dataBridge)
        {
                // Retrieve its directory
            string xmlDirectory = dataBridge.GetClassThreadsDirectory(classID, true);

            ArrayList threads = new ArrayList();
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(Thread));
            XmlTextReader xmlTextReader;
                // Get file paths and deserialize them into a thread arraylist
            foreach (string filename in Directory.GetFiles(xmlDirectory, "*.xml"))
            {
                threads.Add(xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(filename)));
                xmlTextReader.Close();
            }

            if (threads.Count == 0)
                return null;
            else
                return (Thread[])threads.ToArray(typeof(Thread));
        }
        #endregion
    }
}
```

Listing H.106: User.cs

```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.Security;
using System.Security.Cryptography;
using System.Security.Principal;
using System.Text;
using System.Web.Security;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A class which holds properties and methods that define a user.
    /// </summary>
    [XmlRoot("User")]
    public sealed class User : IPrincipal
    {
        #region IPrincipal Properties
        /// <summary>
        /// Implemented as part of the IPrincipal interface.
        /// </summary>
        private IIdentity _identity;
        [XmlIgnore()]
        public IIdentity Identity
        {
            get { return this._identity; }
        }
        #endregion

        #region IPrincipal Methods
        /// <summary>
        /// Implemented as part of the IPrincipal interface.
        /// </summary>
        /// <param name="roleID">The Role ID.</param>
        /// <returns>Returns true if part of a role and false if otherwise.</returns>
        public bool IsInRole(string roleID)
        {
            if (this._userID == 0)
                throw new Exception("UserID not specified!");

                // Search for role
            if (this.RoleIDs != null)
                switch (roleID)
                {
                    case "Administrators":
                        return (Array.BinarySearch(this.RoleIDs, RoleID.Administrators) >= 0) ? true : false;
                    case "Instructors":
                        return (Array.BinarySearch(this.RoleIDs,
```

```
                                RoleID.Instructors) >= 0) ? true : false;
                        case "Students":
                            return (Array.BinarySearch(this.RoleIDs,
                                RoleID.Students) >= 0) ? true : false;
                    }

            return false;
        }
        #endregion

        #region Properties
        /// <summary>
        /// The user's numeric ID. Also used as username.
        /// </summary>
        private int _userID = 0;
        [XmlAttribute("userID")]
        public int UserID
        {
            get { return this._userID; }
            set { this._userID = value; }
        }

        /// <summary>
        /// The user's password in MD5 encryption form.
        /// </summary>
        private string _password;
        [XmlElement("Password")]
        public string Password
        {
            get { return this._password; }
            set { this._password = value; }
        }
        /// <summary>
        /// Used to set and convert the user's password from a
            string to an MD5 encryption string.
        /// </summary>
        public string SetEncryptedPassword
        {
            set { this._password = MD5Encryption.
                EncryptStringToByteString(value); }
        }

        private RoleID[] _roleIDs = null;
        /// <summary>
        /// The user's roles.
        /// </summary>
        [XmlIgnore()]
        public RoleID[] RoleIDs
        {
            get { return this._roleIDs; }
            set
            {
                RoleID[] roles = value;
                if (roles != null)
                    Array.Sort(roles);
```

```
                this._roleIDs = roles;
            }
        }

        private string _firstName;
        /// <summary>
        /// The user's first name.
        /// </summary>
        [XmlElement("FirstName")]
        public string FirstName
        {
            get { return this._firstName; }
            set { this._firstName = value; }
        }

        private string _middleName;
        /// <summary>
        /// The user's middle name.
        /// </summary>
        [XmlElement("MiddleName")]
        public string MiddleName
        {
            get { return this._middleName; }
            set { this._middleName = value; }
        }

        private string _lastName;
        /// <summary>
        /// The user's last name.
        /// </summary>
        [XmlElement("LastName")]
        public string LastName
        {
            get { return this._lastName; }
            set { this._lastName = value; }
        }

        /// <summary>
        /// Gets the user's full name from the first, middle, and
            last name properties.
        /// </summary>
        [XmlIgnore()]
        public string FullName
        {
            get
            {
                if (this.MiddleName == "")
                    return (this.FirstName + " " + this.LastName);
                else
                    return (this.FirstName + " " + this.MiddleName +
                        " " + this.LastName);
            }
        }

        private DateTime _dob;
```

```
156         /// <summary>
157         /// The user's date of birth.
158         /// </summary>
159         [XmlAttribute("dob")]
160         public DateTime DOB
161         {
162             get { return this._dob; }
163             set { this._dob = value; }
164         }
165
166         private AddressStructure _postalAddress;
167         /// <summary>
168         /// The user's postal address.
169         /// </summary>
170         [XmlElement("PostalAddress")]
171         public AddressStructure PostalAddress
172         {
173             get { return this._postalAddress; }
174             set { this._postalAddress = value; }
175         }
176
177         private AddressStructure _residentAddress;
178         /// <summary>
179         /// The user's residential address.
180         /// </summary>
181         [XmlElement("ResidentAddress")]
182         public AddressStructure ResidentAddress
183         {
184             get { return this._residentAddress; }
185             set { this._residentAddress = value; }
186         }
187
188         private string _phoneNumber;
189         /// <summary>
190         /// The user's home phone number.
191         /// </summary>
192         [XmlElement("PhoneNumber")]
193         public string PhoneNumber
194         {
195             get { return _phoneNumber; }
196             set { this._phoneNumber = value; }
197         }
198
199         private string _mobileNumber;
200         /// <summary>
201         /// The user's mobile phone number.
202         /// </summary>
203         [XmlElement("MobileNumber")]
204         public string MobileNumber
205         {
206             get { return _mobileNumber; }
207             set { this._mobileNumber = value; }
208         }
209
210         private string _emailAddress;
```

```
211         /// <summary>
212         /// The user's email address.
213         /// </summary>
214         [XmlElement("EmailAddress")]
215         public string EmailAddress
216         {
217             get { return _emailAddress; }
218             set { this._emailAddress = value; }
219         }
220         #endregion
221
222         #region Constructors
223         /// <summary>
224         /// Used for XmlSerializer purposes.
225         /// </summary>
226         public User() {}
227
228         /// <summary>
229         /// Main constructor.
230         /// </summary>
231         /// <param name="dataBridge">The data bridge to the data.</
            param>
232         public User(DataBridge dataBridge)
233         {
234             this._dataBridge = dataBridge;
235         }
236
237         /// <summary>
238         /// Main constructor.
239         /// </summary>
240         /// <param name="iIdentity">Used for the IPrincipal
            interface.</param>
241         /// <param name="dataBridge">The DataBridge that connects to
             the data.</param>
242         public User(IIdentity iIdentity, DataBridge dataBridge)
243         {
244             this._identity = iIdentity;
245             this._dataBridge = dataBridge;
246         }
247
248         /// <summary>
249         /// Main constructor.
250         /// </summary>
251         /// <param name="userID">The user's ID.</param>
252         /// <param name="iIdentity">Used for the IPrincipal
            interface.</param>
253         /// <param name="dataBridge">The DataBridge that connects to
             the data.</param>
254         public User(int userID, IIdentity iIdentity, DataBridge
            dataBridge)
255         {
256             this._userID = userID;
257             this._identity = iIdentity;
258             this._dataBridge = dataBridge;
259         }
```

```
260         #endregion
261
262         #region Data Management Properties
263         /// <summary>
264         /// The DataBridge that connects to the data.
265         /// </summary>
266         private DataBridge _dataBridge = null;
267         [XmlIgnore()]
268         public DataBridge DataBridge
269         {
270             get { return this._dataBridge; }
271             set { this._dataBridge = value; }
272         }
273         #endregion
274
275         #region Data Management Methods
276         /// <summary>
277         /// Creates an entry in the database of a user.
278         /// </summary>
279         /// <returns>Returns the user's newly assigned ID.</returns>
280         public int Create()
281         {
282             SqlCommand sqlCommand = new SqlCommand("CreateUser",
                    this.DataBridge.SqlConnection);
283             sqlCommand.CommandType = CommandType.StoredProcedure;
284             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
285             sqlCommand.Parameters["@userID"].Direction =
                    ParameterDirection.Output;
286             sqlCommand.Parameters.Add("@password", SqlDbType.Char,
                    UserLimitations.Password);
287             sqlCommand.Parameters["@password"].Value = this.Password
                    ;
288             sqlCommand.Parameters.Add("@firstName", SqlDbType.
                    VarChar, UserLimitations.FirstName);
289             sqlCommand.Parameters["@firstName"].Value = this.
                    FirstName;
290             sqlCommand.Parameters.Add("@middleName", SqlDbType.
                    VarChar, UserLimitations.MiddleName);
291             sqlCommand.Parameters["@middleName"].Value = this.
                    MiddleName;
292             sqlCommand.Parameters.Add("@lastName", SqlDbType.VarChar
                    , UserLimitations.LastName);
293             sqlCommand.Parameters["@lastName"].Value = this.LastName
                    ;
294             sqlCommand.Parameters.Add("@dob", SqlDbType.DateTime);
295             sqlCommand.Parameters["@dob"].Value = this.DOB;
296             sqlCommand.Parameters.Add("@postalStreet", SqlDbType.
                    VarChar, AddressLimitations.Street);
297             sqlCommand.Parameters["@postalStreet"].Value = this.
                    PostalAddress.Street;
298             sqlCommand.Parameters.Add("@postalSuburb", SqlDbType.
                    VarChar, AddressLimitations.Suburb);
299             sqlCommand.Parameters["@postalSuburb"].Value = this.
                    PostalAddress.Suburb;
300             sqlCommand.Parameters.Add("@postalState", SqlDbType.
```

```
                VarChar, AddressLimitations.State);
            sqlCommand.Parameters["@postalState"].Value = this.PostalAddress.State;
            sqlCommand.Parameters.Add("@postalPostcode", SqlDbType.VarChar, AddressLimitations.Postcode);
            sqlCommand.Parameters["@postalPostcode"].Value = this.PostalAddress.Postcode;
            sqlCommand.Parameters.Add("@postalCountry", SqlDbType.VarChar, AddressLimitations.Country);
            sqlCommand.Parameters["@postalCountry"].Value = this.PostalAddress.Country;
            sqlCommand.Parameters.Add("@residentStreet", SqlDbType.VarChar, AddressLimitations.Street);
            sqlCommand.Parameters["@residentStreet"].Value = this.ResidentAddress.Street;
            sqlCommand.Parameters.Add("@residentSuburb", SqlDbType.VarChar, AddressLimitations.Suburb);
            sqlCommand.Parameters["@residentSuburb"].Value = this.ResidentAddress.Suburb;
            sqlCommand.Parameters.Add("@residentState", SqlDbType.VarChar, AddressLimitations.State);
            sqlCommand.Parameters["@residentState"].Value = this.ResidentAddress.State;
            sqlCommand.Parameters.Add("@residentPostcode", SqlDbType.VarChar, AddressLimitations.Postcode);
            sqlCommand.Parameters["@residentPostcode"].Value = this.ResidentAddress.Postcode;
            sqlCommand.Parameters.Add("@residentCountry", SqlDbType.VarChar, AddressLimitations.Country);
            sqlCommand.Parameters["@residentCountry"].Value = this.ResidentAddress.Country;
            sqlCommand.Parameters.Add("@phoneNumber", SqlDbType.VarChar, UserLimitations.PhoneNumber);
            sqlCommand.Parameters["@phoneNumber"].Value = this.PhoneNumber;
            sqlCommand.Parameters.Add("@mobileNumber", SqlDbType.VarChar, UserLimitations.PhoneNumber);
            sqlCommand.Parameters["@mobileNumber"].Value = this.MobileNumber;
            sqlCommand.Parameters.Add("@emailAddress", SqlDbType.VarChar, UserLimitations.EmailAddress);
            sqlCommand.Parameters["@emailAddress"].Value = this.EmailAddress;
            this.DataBridge.SqlConnection.Open();
                // Execute SQL Stored Procedure
            sqlCommand.ExecuteNonQuery();
            this.DataBridge.SqlConnection.Close();

            this.UserID = (int)sqlCommand.Parameters["@userID"].Value;

            return this.UserID;
        }

        /// <summary>
```

```
333         /// Alters an entry of a user within the database.
334         /// </summary>
335         public void Alter()
336         {
337             SqlCommand sqlCommand = new SqlCommand("AlterUser", this
                    .DataBridge.SqlConnection);
338             sqlCommand.CommandType = CommandType.StoredProcedure;
339             sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
340             sqlCommand.Parameters["@userID"].Value = this.UserID;
341             sqlCommand.Parameters.Add("@password", SqlDbType.Char,
                    UserLimitations.Password);
342             sqlCommand.Parameters["@password"].Value = this.Password
                    ;
343             sqlCommand.Parameters.Add("@firstName", SqlDbType.
                    VarChar, UserLimitations.FirstName);
344             sqlCommand.Parameters["@firstName"].Value = this.
                    FirstName;
345             sqlCommand.Parameters.Add("@middleName", SqlDbType.
                    VarChar, UserLimitations.MiddleName);
346             sqlCommand.Parameters["@middleName"].Value = this.
                    MiddleName;
347             sqlCommand.Parameters.Add("@lastName", SqlDbType.VarChar
                    , UserLimitations.LastName);
348             sqlCommand.Parameters["@lastName"].Value = this.LastName
                    ;
349             sqlCommand.Parameters.Add("@dob", SqlDbType.DateTime);
350             sqlCommand.Parameters["@dob"].Value = this.DOB;
351             sqlCommand.Parameters.Add("@postalStreet", SqlDbType.
                    VarChar, AddressLimitations.Street);
352             sqlCommand.Parameters["@postalStreet"].Value = this.
                    PostalAddress.Street;
353             sqlCommand.Parameters.Add("@postalSuburb", SqlDbType.
                    VarChar, AddressLimitations.Suburb);
354             sqlCommand.Parameters["@postalSuburb"].Value = this.
                    PostalAddress.Suburb;
355             sqlCommand.Parameters.Add("@postalState", SqlDbType.
                    VarChar, AddressLimitations.State);
356             sqlCommand.Parameters["@postalState"].Value = this.
                    PostalAddress.State;
357             sqlCommand.Parameters.Add("@postalPostcode", SqlDbType.
                    VarChar, AddressLimitations.Postcode);
358             sqlCommand.Parameters["@postalPostcode"].Value = this.
                    PostalAddress.Postcode;
359             sqlCommand.Parameters.Add("@postalCountry", SqlDbType.
                    VarChar, AddressLimitations.Country);
360             sqlCommand.Parameters["@postalCountry"].Value = this.
                    PostalAddress.Country;
361             sqlCommand.Parameters.Add("@residentStreet", SqlDbType.
                    VarChar, AddressLimitations.Street);
362             sqlCommand.Parameters["@residentStreet"].Value = this.
                    ResidentAddress.Street;
363             sqlCommand.Parameters.Add("@residentSuburb", SqlDbType.
                    VarChar, AddressLimitations.Suburb);
364             sqlCommand.Parameters["@residentSuburb"].Value = this.
                    ResidentAddress.Suburb;
```

```
365                 sqlCommand.Parameters.Add("@residentState", SqlDbType.
                        VarChar, AddressLimitations.State);
366                 sqlCommand.Parameters["@residentState"].Value = this.
                        ResidentAddress.State;
367                 sqlCommand.Parameters.Add("@residentPostcode", SqlDbType
                        .VarChar, AddressLimitations.Postcode);
368                 sqlCommand.Parameters["@residentPostcode"].Value = this.
                        ResidentAddress.Postcode;
369                 sqlCommand.Parameters.Add("@residentCountry", SqlDbType.
                        VarChar, AddressLimitations.Country);
370                 sqlCommand.Parameters["@residentCountry"].Value = this.
                        ResidentAddress.Country;
371                 sqlCommand.Parameters.Add("@phoneNumber", SqlDbType.
                        VarChar, UserLimitations.PhoneNumber);
372                 sqlCommand.Parameters["@phoneNumber"].Value = this.
                        PhoneNumber;
373                 sqlCommand.Parameters.Add("@mobileNumber", SqlDbType.
                        VarChar, UserLimitations.PhoneNumber);
374                 sqlCommand.Parameters["@mobileNumber"].Value = this.
                        MobileNumber;
375                 sqlCommand.Parameters.Add("@emailAddress", SqlDbType.
                        VarChar, UserLimitations.EmailAddress);
376                 sqlCommand.Parameters["@emailAddress"].Value = this.
                        EmailAddress;
377                 this.DataBridge.SqlConnection.Open();
378                     // Execute SQL Stored Procedure
379                 sqlCommand.ExecuteNonQuery();
380                 this.DataBridge.SqlConnection.Close();
381             }
382
383             /// <summary>
384             /// Delete a user from within the database.
385             /// </summary>
386             public void Delete()
387             {
388                 SqlCommand sqlCommand = new SqlCommand("DeleteUser",
                        this.DataBridge.SqlConnection);
389                 sqlCommand.CommandType = CommandType.StoredProcedure;
390                 sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
391                 sqlCommand.Parameters["@userID"].Value = this.UserID;
392                 this.DataBridge.SqlConnection.Open();
393                     // Execute SQL Stored Procedure
394                 sqlCommand.ExecuteNonQuery();
395                 this.DataBridge.SqlConnection.Close();
396             }
397             #endregion
398
399             #region Static Data Retrieving Methods
400             /// <summary>
401             /// Find a specific user.
402             /// </summary>
403             /// <param name="userID">The ID of the user to find.</param>
404             /// <param name="dataBridge">The bridge to the data.</param>
405             /// <returns>Returns a User class or null if not found.</
                    returns>
```

```
        public static User FindUser(int userID, DataBridge dataBridge)
        {
            SqlCommand sqlCommand = new SqlCommand("FindUser", dataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
            sqlCommand.Parameters["@userID"].Value = userID;

            dataBridge.SqlConnection.Open();
            XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();

            User user = null;
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(User));
                // Read and deserialize xml to a User object
            while (xmlTextReader.Read())
                if (xmlSerializer.CanDeserialize(xmlTextReader))
                    user = xmlSerializer.Deserialize(xmlTextReader) as User;

            xmlTextReader.Close();
            dataBridge.SqlConnection.Close();


            if (user == null)
                return null;
            else
            {
                user.DataBridge = dataBridge;
                return user;
            }
        }

        /// <summary>
        /// Find all users within a specific class.
        /// </summary>
        /// <param name="classID">The ID of the class to search for users.</param>
        /// <param name="dataBridge">The bridge to the data.</param>
        /// <returns>Returns an array of all users or null if not found.</returns>
        public static User[] FindUsers(string classID, DataBridge dataBridge)
        {
            SqlCommand sqlCommand = new SqlCommand("FindUsers", dataBridge.SqlConnection);
            sqlCommand.CommandType = CommandType.StoredProcedure;
            sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
            sqlCommand.Parameters["@classID"].Value = classID;

            dataBridge.SqlConnection.Open();
                // Execute stored procedure and retrieve as XML
```

```
451                 XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
                        ExecuteXmlReader();
452
453                 ArrayList users = new ArrayList();
454                 XmlSerializer xmlSerializer = new XmlSerializer(typeof(
                        User));
455                     // Read and deserialize xml to a User object
                            arraylist
456                 while (xmlTextReader.Read())
457                 {
458                     while (xmlSerializer.CanDeserialize(xmlTextReader))
459                     {
460                         User tempUser = xmlSerializer.Deserialize(
                                xmlTextReader) as User;
461                         tempUser.DataBridge = dataBridge;
462                         users.Add(tempUser);
463                     }
464                 }
465
466                 xmlTextReader.Close();
467                 dataBridge.SqlConnection.Close();
468
469                 if (users == null)
470                     return null;
471                 else
472                     return (User[])users.ToArray(typeof(User));
473             }
474             #endregion
475         }
476 }
```

H.3 Database Source Code Project

Listing H.107: oledb.sql

```
1  /* Generated by Web Data Administrator on 5/08/2004 4:08:22 PM */
2
3  /* Options selected: database drop-commands table-schema table-data
       stored-procedures comments */
4
5  /****** Object: Database oledb Script Date: 5/08/2004 4:08:22 PM
       ******/
6  IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'
       oledb')
7      DROP DATABASE [oledb]
8  GO
9
10 CREATE DATABASE [oledb]
11 GO
12
13 exec sp_dboption N'oledb', N'autoclose', N'true'
14 GO
15
```

```
exec sp_dboption N'oledb', N'bulkcopy', N'false'
GO

exec sp_dboption N'oledb', N'trunc. log', N'true'
GO

exec sp_dboption N'oledb', N'torn page detection', N'true'
GO

exec sp_dboption N'oledb', N'read only', N'false'
GO

exec sp_dboption N'oledb', N'dbo use', N'false'
GO

exec sp_dboption N'oledb', N'single', N'false'
GO

exec sp_dboption N'oledb', N'autoshrink', N'true'
GO

exec sp_dboption N'oledb', N'ANSI null default', N'false'
GO

exec sp_dboption N'oledb', N'recursive triggers', N'false'
GO

exec sp_dboption N'oledb', N'ANSI nulls', N'false'
GO

exec sp_dboption N'oledb', N'concat null yields null', N'false'
GO

exec sp_dboption N'oledb', N'cursor close on commit', N'false'
GO

exec sp_dboption N'oledb', N'default to local cursor', N'false'
GO

exec sp_dboption N'oledb', N'quoted identifier', N'false'
GO

exec sp_dboption N'oledb', N'ANSI warnings', N'false'
GO

exec sp_dboption N'oledb', N'auto create statistics', N'true'
GO

exec sp_dboption N'oledb', N'auto update statistics', N'true'
GO

if( ( (@@microsoftversion / power(2, 24) = 8) and (
    @@microsoftversion & 0xffff >= 724) ) or ( (@@microsoftversion /
    power(2, 24) = 7) and (@@microsoftversion & 0xffff >= 1082) ) )
     exec sp_dboption N'oledb', N'db chaining', N'false'
```

```
GO

use [oledb]
GO

exec sp_adduser 'NT_AUTHORITY\Network_Service', 'NT_AUTHORITY\
    Network_Service', 'db_owner'
GO
exec sp_grantlogin 'NT_AUTHORITY\Network_Service'
GO

/****** Object:  Stored Procedure dbo.AlterAETAnswer    Script Date:
     5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo
    ].[AlterAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[AlterAETAnswer]
GO

/****** Object:  Stored Procedure dbo.AlterAETQuestion    Script
    Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo
    ].[AlterAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') =
    1)
drop procedure [dbo].[AlterAETQuestion]
GO

/****** Object:  Stored Procedure dbo.AlterClass    Script Date:
    5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo
    ].[AlterClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[AlterClass]
GO

/****** Object:  Stored Procedure dbo.AlterUser    Script Date:
    5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo
    ].[AlterUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[AlterUser]
GO

/****** Object:  Stored Procedure dbo.AssignUser    Script Date:
    5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo
    ].[AssignUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[AssignUser]
GO

/****** Object:  Stored Procedure dbo.AssignUserToClass    Script
    Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo
    ].[AssignUserToClass]') and OBJECTPROPERTY(id, N'IsProcedure') =
    1)
drop procedure [dbo].[AssignUserToClass]
GO

```

```
/****** Object:  Stored Procedure dbo.Authentication    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Authentication]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[Authentication]
GO

/****** Object:  Stored Procedure dbo.CreateAETAnswer    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateAETAnswer]
GO

/****** Object:  Stored Procedure dbo.CreateAETQuestion    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateAETQuestion]
GO

/****** Object:  Stored Procedure dbo.CreateClass    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateClass]
GO

/****** Object:  Stored Procedure dbo.CreateRole    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateRole]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateRole]
GO

/****** Object:  Stored Procedure dbo.CreateThreadSub    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateThreadSub]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateThreadSub]
GO

/****** Object:  Stored Procedure dbo.CreateUser    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateUser]
GO

/****** Object:  Stored Procedure dbo.DeleteAETAnswer    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteAETAnswer]
```

```
GO

/****** Object:  Stored Procedure dbo.DeleteAETQuestion    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteAETQuestion]
GO

/****** Object:  Stored Procedure dbo.DeleteClass    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteClass]
GO

/****** Object:  Stored Procedure dbo.DeleteRole    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteRole]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteRole]
GO

/****** Object:  Stored Procedure dbo.DeleteThreadSub    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteThreadSub]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteThreadSub]
GO

/****** Object:  Stored Procedure dbo.DeleteUser    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteUser]
GO

/****** Object:  Stored Procedure dbo.FindClass    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindClass]
GO

/****** Object:  Stored Procedure dbo.FindClasses    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindClasses]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindClasses]
GO

/****** Object:  Stored Procedure dbo.FindRolesIDs    Script Date: 5/08/2004 4:08:23 PM ******/
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindRolesIDs]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindRolesIDs]
GO

/****** Object:  Stored Procedure dbo.FindShortAETAnswer    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETAnswer]
GO

/****** Object:  Stored Procedure dbo.FindShortAETAnswers    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETAnswers]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETAnswers]
GO

/****** Object:  Stored Procedure dbo.FindShortAETQuestion    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETQuestion]
GO

/****** Object:  Stored Procedure dbo.FindShortAETQuestions    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETQuestions]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETQuestions]
GO

/****** Object:  Stored Procedure dbo.FindUser    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindUser]
GO

/****** Object:  Stored Procedure dbo.FindUsers    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FindUsers]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindUsers]
GO

/****** Object:  Stored Procedure dbo.IsInClass    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[IsInClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
```

```
drop procedure [dbo].[IsInClass]
GO

/****** Object:  Table [dbo].[AETAnswers]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[AETAnswers]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [AETAnswers]
GO


/****** Object:  Table [dbo].[AETQuestions]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[AETQuestions]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [AETQuestions]
GO


/****** Object:  Table [dbo].[Classes]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[Classes]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [Classes]
GO


/****** Object:  Table [dbo].[ClassUsers]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[ClassUsers]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [ClassUsers]
GO


/****** Object:  Table [dbo].[OtherUsers]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[OtherUsers]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [OtherUsers]
GO


/****** Object:  Table [dbo].[Roles]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[Roles]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [Roles]
GO


/****** Object:  Table [dbo].[ThreadSubscriptions]    Script Date: 5/08/2004 4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[ThreadSubscriptions]') and OBJECTPROPERTY(id, N'IsUserTable') =
```

```
    1)
drop table [ThreadSubscriptions]
GO


/****** Object:  Table [dbo].[Users]    Script Date: 5/08/2004
    4:08:23 PM ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[
    Users]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [Users]
GO


/****** Object:  Table [dbo].[AETAnswers]    Script Date: 5/08/2004
    4:08:23 PM ******/
CREATE TABLE [AETAnswers] (
    [aetQID] [bigint] NOT NULL ,
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    [userID] [int] NOT NULL ,
    [aetQType] [varchar] (20) COLLATE Latin1_General_CI_AS NOT NULL
        ,
    [aetASubmittedDate] [datetime] NULL ,
    [aetAMarked] [bit] NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[AETQuestions]    Script Date:
    5/08/2004 4:08:24 PM ******/
CREATE TABLE [AETQuestions] (
    [aetQID] [bigint] NOT NULL ,
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    [aetQType] [varchar] (20) COLLATE Latin1_General_CI_AS NOT NULL
        ,
    [aetQTitle] [varchar] (100) COLLATE Latin1_General_CI_AS NULL ,
    [aetQDescription] [varchar] (30) COLLATE Latin1_General_CI_AS
        NULL ,
    [aetQQuestionType] [varchar] (20) COLLATE Latin1_General_CI_AS
        NULL ,
    [aetQViewQuestionDateTime] [datetime] NULL ,
    [aetQDueDateTime] [datetime] NULL ,
    [aetQHaltSubmissionDateTime] [datetime] NULL ,
    [aetQViewMarkedAnswersDateTime] [datetime] NULL ,
    [aetQAmountOfQuestions] [int] NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[Classes]    Script Date: 5/08/2004
    4:08:24 PM ******/
CREATE TABLE [Classes] (
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    [className] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
    [denyOtherUsers] [bit] NULL
) ON [PRIMARY]
```

```
GO


/****** Object:  Table [dbo].[ClassUsers]    Script Date: 5/08/2004
    4:08:24 PM ******/
CREATE TABLE [ClassUsers] (
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    [userID] [int] NOT NULL ,
    [roleID] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[OtherUsers]    Script Date: 5/08/2004
    4:08:24 PM ******/
CREATE TABLE [OtherUsers] (
    [userID] [int] NOT NULL ,
    [roleID] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[Roles]    Script Date: 5/08/2004
    4:08:24 PM ******/
CREATE TABLE [Roles] (
    [roleID] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[ThreadSubscriptions]    Script Date:
    5/08/2004 4:08:24 PM ******/
CREATE TABLE [ThreadSubscriptions] (
    [userID] [int] NOT NULL ,
    [threadID] [int] NOT NULL ,
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[Users]    Script Date: 5/08/2004
    4:08:24 PM ******/
CREATE TABLE [Users] (
    [userID] [int] IDENTITY (1, 1) NOT NULL ,
    [password] [char] (47) COLLATE Latin1_General_CI_AS NOT NULL ,
    [firstName] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
    [middleName] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
    [lastName] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
    [dob] [datetime] NULL ,
    [postalStreet] [varchar] (50) COLLATE Latin1_General_CI_AS NULL
        ,
    [postalSuburb] [varchar] (50) COLLATE Latin1_General_CI_AS NULL
        ,
    [postalState] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
```

```
354     [postalPostcode] [varchar] (10) COLLATE Latin1_General_CI_AS
            NULL ,
355     [postalCountry] [varchar] (50) COLLATE Latin1_General_CI_AS NULL
            ,
356     [residentStreet] [varchar] (50) COLLATE Latin1_General_CI_AS
            NULL ,
357     [residentSuburb] [varchar] (50) COLLATE Latin1_General_CI_AS
            NULL ,
358     [residentState] [varchar] (50) COLLATE Latin1_General_CI_AS NULL
            ,
359     [residentPostcode] [varchar] (10) COLLATE Latin1_General_CI_AS
            NULL ,
360     [residentCountry] [varchar] (50) COLLATE Latin1_General_CI_AS
            NULL ,
361     [phoneNumber] [varchar] (15) COLLATE Latin1_General_CI_AS NULL ,
362     [mobileNumber] [varchar] (15) COLLATE Latin1_General_CI_AS NULL
            ,
363     [emailAddress] [varchar] (50) COLLATE Latin1_General_CI_AS NULL
364 ) ON [PRIMARY]
365 GO
366
367
368 /* Data for table AETAnswers */
369 /* Data for table AETQuestions */
370 /* Data for table Classes */
371 /* Data for table ClassUsers */
372 /* Data for table OtherUsers */
373 INSERT [OtherUsers] ([userID], [roleID]) VALUES (1, 'Administrators'
        )
374 /* Data for table Roles */
375 INSERT [Roles] ([roleID]) VALUES ('Administrators')
376 INSERT [Roles] ([roleID]) VALUES ('Instructors')
377 INSERT [Roles] ([roleID]) VALUES ('Students')
378 /* Data for table ThreadSubscriptions */
379 /* Data for table Users */
380 SET identity_insert [Users] on
381
382 INSERT [Users] ([userID], [password], [firstName], [middleName], [
        lastName], [dob], [postalStreet], [postalSuburb], [postalState],
        [postalPostcode], [postalCountry], [residentStreet], [
        residentSuburb], [residentState], [residentPostcode], [
        residentCountry], [phoneNumber], [mobileNumber], [emailAddress])
        VALUES (1, '64-4B-45-7A-92-E4-66-A1-EB-14-33-0E-CE-BE-A6-B9',
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
         NULL, NULL, NULL, NULL, NULL, NULL)
383 SET identity_insert [Users] off
384 GO
385 /****** Object: Table [dbo].[AETAnswers]    Script Date: 5/08/2004
        4:08:28 PM ******/
386 ALTER TABLE [AETAnswers] WITH NOCHECK ADD
387     CONSTRAINT [PK_AETAnswers] PRIMARY KEY  CLUSTERED
388     (
389         [aetQID],
390         [classID],
391         [userID],
```

```
392             [aetQType]
393         )  ON [PRIMARY]
394 GO
395
396
397 /****** Object:  Table [dbo].[AETQuestions]    Script Date:
        5/08/2004 4:08:28 PM ******/
398 ALTER TABLE [AETQuestions] WITH NOCHECK ADD
399         CONSTRAINT [PK_AETQuestions] PRIMARY KEY  CLUSTERED
400         (
401             [aetQID],
402             [classID],
403             [aetQType]
404         )  ON [PRIMARY]
405 GO
406
407
408 /****** Object:  Table [dbo].[Classes]    Script Date: 5/08/2004
        4:08:28 PM ******/
409 ALTER TABLE [Classes] WITH NOCHECK ADD
410         CONSTRAINT [PK_Classes] PRIMARY KEY  CLUSTERED
411         (
412             [classID]
413         )  ON [PRIMARY]
414 GO
415
416
417 /****** Object:  Table [dbo].[ClassUsers]    Script Date: 5/08/2004
        4:08:28 PM ******/
418 ALTER TABLE [ClassUsers] WITH NOCHECK ADD
419         CONSTRAINT [PK_ClassUsers] PRIMARY KEY  CLUSTERED
420         (
421             [classID],
422             [userID],
423             [roleID]
424         )  ON [PRIMARY]
425 GO
426
427
428 /****** Object:  Table [dbo].[OtherUsers]    Script Date: 5/08/2004
        4:08:28 PM ******/
429 ALTER TABLE [OtherUsers] WITH NOCHECK ADD
430         CONSTRAINT [PK_OtherUsers] PRIMARY KEY  CLUSTERED
431         (
432             [userID],
433             [roleID]
434         )  ON [PRIMARY]
435 GO
436
437
438 /****** Object:  Table [dbo].[Roles]    Script Date: 5/08/2004
        4:08:28 PM ******/
439 ALTER TABLE [Roles] WITH NOCHECK ADD
440         CONSTRAINT [PK_Roles] PRIMARY KEY  CLUSTERED
441         (
```

```
442             [roleID]
443         )  ON [PRIMARY]
444 GO
445
446
447 /****** Object:  Table [dbo].[ThreadSubscriptions]    Script Date:
        5/08/2004 4:08:28 PM ******/
448 ALTER TABLE [ThreadSubscriptions] WITH NOCHECK ADD
449         CONSTRAINT [PK_ThreadSubscriptions] PRIMARY KEY  CLUSTERED
450         (
451             [userID],
452             [threadID],
453             [classID]
454         )  ON [PRIMARY]
455 GO
456
457
458 /****** Object:  Table [dbo].[Users]    Script Date: 5/08/2004
        4:08:28 PM ******/
459 ALTER TABLE [Users] WITH NOCHECK ADD
460         CONSTRAINT [PK_Users] PRIMARY KEY  CLUSTERED
461         (
462             [userID]
463         )  ON [PRIMARY]
464 GO
465
466
467 /****** Object:  Table [dbo].[AETAnswers]    Script Date: 5/08/2004
        4:08:28 PM ******/
468 ALTER TABLE [AETAnswers] ADD
469         CONSTRAINT [FK_AETAnswers_AETQuestions1] FOREIGN KEY
470         (
471             [aetQID],
472             [classID],
473             [aetQType]
474         ) REFERENCES [AETQuestions] (
475             [aetQID],
476             [classID],
477             [aetQType]
478         ),
479         CONSTRAINT [FK_AETAnswers_Classes] FOREIGN KEY
480         (
481             [classID]
482         ) REFERENCES [Classes] (
483             [classID]
484         ) ON DELETE CASCADE  ON UPDATE CASCADE ,
485         CONSTRAINT [FK_AETAnswers_Users] FOREIGN KEY
486         (
487             [userID]
488         ) REFERENCES [Users] (
489             [userID]
490         ) ON DELETE CASCADE  ON UPDATE CASCADE
491 GO
492
493
```

```
/****** Object:  Table [dbo].[AETQuestions]    Script Date:
    5/08/2004 4:08:28 PM ******/
ALTER TABLE [AETQuestions] ADD
    CONSTRAINT [FK_AETQuestions_Classes] FOREIGN KEY
    (
        [classID]
    ) REFERENCES [Classes] (
        [classID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE
GO



/****** Object:  Table [dbo].[ClassUsers]    Script Date: 5/08/2004
    4:08:28 PM ******/
ALTER TABLE [ClassUsers] ADD
    CONSTRAINT [FK_ClassUsers_Classes] FOREIGN KEY
    (
        [classID]
    ) REFERENCES [Classes] (
        [classID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE ,
    CONSTRAINT [FK_ClassUsers_Roles] FOREIGN KEY
    (
        [roleID]
    ) REFERENCES [Roles] (
        [roleID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE ,
    CONSTRAINT [FK_ClassUsers_Users] FOREIGN KEY
    (
        [userID]
    ) REFERENCES [Users] (
        [userID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE
GO


/****** Object:  Table [dbo].[OtherUsers]    Script Date: 5/08/2004
    4:08:29 PM ******/
ALTER TABLE [OtherUsers] ADD
    CONSTRAINT [FK_OtherUsers_Roles] FOREIGN KEY
    (
        [roleID]
    ) REFERENCES [Roles] (
        [roleID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE ,
    CONSTRAINT [FK_OtherUsers_Users] FOREIGN KEY
    (
        [userID]
    ) REFERENCES [Users] (
        [userID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE
GO
```

```
/****** Object:  Table [dbo].[ThreadSubscriptions]    Script Date: 5/08/2004 4:08:29 PM ******/
ALTER TABLE [ThreadSubscriptions] ADD
    CONSTRAINT [FK_ThreadSubscriptions_Classes] FOREIGN KEY
    (
        [classID]
    ) REFERENCES [Classes] (
        [classID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE ,
    CONSTRAINT [FK_ThreadSubscriptions_Users] FOREIGN KEY
    (
        [userID]
    ) REFERENCES [Users] (
        [userID]
    ) ON DELETE CASCADE  ON UPDATE CASCADE
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/****** Object:  Stored Procedure dbo.AlterAETAnswer    Script Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.AlterAETAnswer
    (
        @aetQID BIGINT,
        @classID CHAR(7),
        @userID BIGINT,
        @aetQType VARCHAR(20),
        @aetASubmittedDate DATETIME,
        @aetAMarked BIT
    )
AS
    UPDATE AETAnswers
    SET aetASubmittedDate=@aetASubmittedDate, aetAMarked=@aetAMarked
```

```
599     WHERE aetQID=@aetQID AND classID=@classID AND userID=@userID AND
            aetQType=@aetQType
600 GO
601
602 SET QUOTED_IDENTIFIER OFF
603 GO
604 SET ANSI_NULLS ON
605 GO
606
607 SET QUOTED_IDENTIFIER ON
608 GO
609 SET ANSI_NULLS ON
610 GO
611
612 /****** Object:  Stored Procedure dbo.AlterAETQuestion    Script
        Date: 5/08/2004 4:08:29 PM ******/
613 CREATE PROCEDURE dbo.AlterAETQuestion
614     (
615         @aetQID BIGINT,
616         @classID CHAR(7),
617         @aetQType VARCHAR(20),
618         @aetQTitle VARCHAR(100),
619         @aetQDescription VARCHAR(30),
620         @aetQQuestionType VARCHAR(20),
621         @aetQViewQuestionDateTime DATETIME,
622         @aetQDueDateTime DATETIME,
623         @aetQHaltSubmissionDateTime DATETIME,
624         @aetQViewMarkedAnswersDateTime DATETIME,
625         @aetQAmountOfQuestions INT
626     )
627 AS
628     UPDATE AETQuestions
629     SET aetQTitle=@aetQTitle, aetQDescription=@aetQDescription,
            aetQType=@aetQType, aetQQuestionType=@aetQQuestionType,
            aetQViewQuestionDateTime=@aetQViewQuestionDateTime,
            aetQDueDateTime=@aetQDueDateTime, aetQHaltSubmissionDateTime=
            @aetQHaltSubmissionDateTime, aetQViewMarkedAnswersDateTime=
            @aetQViewMarkedAnswersDateTime, aetQAmountOfQuestions=
            @aetQAmountOfQuestions
630     WHERE aetQID=@aetQID AND classID=@classID AND aetQType=@aetQType
631 GO
632
633 SET QUOTED_IDENTIFIER OFF
634 GO
635 SET ANSI_NULLS ON
636 GO
637
638 SET QUOTED_IDENTIFIER ON
639 GO
640 SET ANSI_NULLS ON
641 GO
642
643 /****** Object:  Stored Procedure dbo.AlterClass    Script Date:
        5/08/2004 4:08:29 PM ******/
644 CREATE PROCEDURE dbo.AlterClass
```

```
    (
        @classID CHAR(7),
        @className VARCHAR(50),
        @denyOtherUsers BIT
    )
AS
    UPDATE Classes
    SET className=@className, denyOtherUsers=@denyOtherUsers
    WHERE classID=@classID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.AlterUser    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.AlterUser
    (
        @userID INT,
        @password CHAR(47),
        @firstName varchar(50),
        @middleName varchar(50),
        @lastName varchar(50),
        @dob DATETIME,
        @postalStreet varchar(50),
        @postalSuburb varchar(50),
        @postalState varchar(50),
        @postalPostcode varchar(10),
        @postalCountry varchar(50),
        @residentStreet varchar(50),
        @residentSuburb varchar(50),
        @residentState varchar(50),
        @residentPostcode varchar(10),
        @residentCountry varchar(50),
        @phoneNumber varchar(15),
        @mobileNumber varchar(15),
        @emailAddress varchar(50)
    )
AS
    UPDATE Users
        SET password=@password, firstName=@firstName, middleName=
            @middleName, lastName=@lastName, dob=@dob, postalStreet=
            @postalStreet, postalSuburb=@postalSuburb, postalState=
            @postalState, postalPostcode=@postalPostcode,
            postalCountry=@postalCountry, residentStreet=
            @residentStreet, residentSuburb=@residentSuburb,
            residentState=@residentState, residentPostcode=
            @residentPostcode, residentCountry=@residentCountry,
```

```
                phoneNumber=@phoneNumber, mobileNumber=@mobileNumber,
                emailAddress=@emailAddress
            WHERE userID=@userID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.AssignUser    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.AssignUser
    (
        @userID INT,
        @roleID VARCHAR(50)
    )
AS
    INSERT INTO OtherUsers
        (userID, roleID)
    VALUES
        (@userID, @roleID)
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.AssignUserToClass    Script
    Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.AssignUserToClass
    (
        @userID INT,
        @classID CHAR(7),
        @roleID VARCHAR(50)
    )
AS
    INSERT INTO ClassUsers
        (userID, classID, roleID)
    VALUES
        (@userID, @classID, @roleID)
GO

SET QUOTED_IDENTIFIER OFF
```

```
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.Authentication    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.Authentication
    (
        @userID INT,
        @password CHAR(47)
    )
AS
    SELECT userID
    FROM Users
    WHERE userID=@userID AND password=@password

    IF @@ROWCOUNT < 1
        RETURN 0
    ELSE
        RETURN 1
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.CreateAETAnswer    Script Date
    : 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.CreateAETAnswer
    (
        @aetQID BIGINT,
        @classID CHAR(7),
        @userID BIGINT,
        @aetQType VARCHAR(20),
        @aetASubmittedDate DATETIME,
        @aetAMarked BIT
    )
AS
    INSERT INTO AETAnswers
        (aetQID, classID, userID, aetQType, aetASubmittedDate,
            aetAMarked)
    VALUES
        (@aetQID, @classID, @userID, @aetQType, @aetASubmittedDate,
            @aetAMarked)
```

```
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.CreateAETQuestion    Script
    Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.CreateAETQuestion
    (
        @aetQID BIGINT,
        @classID CHAR(7),
        @aetQTitle VARCHAR(100),
        @aetQDescription VARCHAR(30),
        @aetQType VARCHAR(20),
        @aetQQuestionType VARCHAR(20),
        @aetQViewQuestionDateTime DATETIME,
        @aetQDueDateTime DATETIME,
        @aetQHaltSubmissionDateTime DATETIME,
        @aetQViewMarkedAnswersDateTime DATETIME,
        @aetQAmountOfQuestions INT
    )
AS
    INSERT INTO AETQuestions
        (aetQID, classID, aetQTitle, aetQDescription, aetQType,
            aetQQuestionType, aetQViewQuestionDateTime,
            aetQDueDateTime, aetQHaltSubmissionDateTime,
            aetQViewMarkedAnswersDateTime, aetQAmountOfQuestions)
    VALUES
        (@aetQID, @classID, @aetQTitle, @aetQDescription, @aetQType,
             @aetQQuestionType, @aetQViewQuestionDateTime,
            @aetQDueDateTime, @aetQHaltSubmissionDateTime,
            @aetQViewMarkedAnswersDateTime, @aetQAmountOfQuestions)
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.CreateClass    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.CreateClass
    (
```

```
841             @classID CHAR(7),
842             @className VARCHAR(50),
843             @denyOtherUsers BIT
844         )
845 AS
846         INSERT INTO Classes
847             (classID, className, denyOtherUsers)
848         VALUES
849             (@classID, @className, @denyOtherUsers)
850 GO
851
852 SET QUOTED_IDENTIFIER OFF
853 GO
854 SET ANSI_NULLS ON
855 GO
856
857 SET QUOTED_IDENTIFIER ON
858 GO
859 SET ANSI_NULLS ON
860 GO
861
862 /****** Object:  Stored Procedure dbo.CreateRole    Script Date:
        5/08/2004 4:08:29 PM ******/
863 CREATE PROCEDURE dbo.CreateRole
864         (
865             @roleID VARCHAR(50)
866         )
867 AS
868         INSERT INTO Roles
869             (roleID)
870         VALUES
871             (@roleID)
872 GO
873
874 SET QUOTED_IDENTIFIER OFF
875 GO
876 SET ANSI_NULLS ON
877 GO
878
879 SET QUOTED_IDENTIFIER ON
880 GO
881 SET ANSI_NULLS ON
882 GO
883
884 /****** Object:  Stored Procedure dbo.CreateThreadSub    Script Date
        : 5/08/2004 4:08:29 PM ******/
885 CREATE PROCEDURE dbo.CreateThreadSub
886         (
887             @classID CHAR(7),
888             @userID INT,
889             @threadID INT
890         )
891 AS
892         INSERT INTO ThreadSubscriptions
893             (classID, userID, threadID)
```

```
894     VALUES
895         (@classID, @userID, @threadID)
896 GO
897
898 SET QUOTED_IDENTIFIER OFF
899 GO
900 SET ANSI_NULLS ON
901 GO
902
903 SET QUOTED_IDENTIFIER ON
904 GO
905 SET ANSI_NULLS ON
906 GO
907
908 /****** Object:  Stored Procedure dbo.CreateUser    Script Date:
        5/08/2004 4:08:29 PM ******/
909 CREATE PROCEDURE dbo.CreateUser
910     (
911         @userID INT OUTPUT,
912         @password CHAR(47),
913         @firstName varchar(50),
914         @middleName varchar(50),
915         @lastName varchar(50),
916         @dob DATETIME,
917         @postalStreet varchar(50),
918         @postalSuburb varchar(50),
919         @postalState varchar(50),
920         @postalPostcode varchar(10),
921         @postalCountry varchar(50),
922         @residentStreet varchar(50),
923         @residentSuburb varchar(50),
924         @residentState varchar(50),
925         @residentPostcode varchar(10),
926         @residentCountry varchar(50),
927         @phoneNumber varchar(15),
928         @mobileNumber varchar(15),
929         @emailAddress varchar(50)
930     )
931 AS
932     INSERT INTO Users
933         (password, firstName, middleName, lastName, dob,
                postalStreet, postalSuburb, postalState, postalPostcode,
                postalCountry, residentStreet, residentSuburb,
                residentState, residentPostcode, residentCountry,
                phoneNumber, mobileNumber, emailAddress)
934     VALUES
935         (@password, @firstName, @middleName, @lastName, @dob,
                @postalStreet, @postalSuburb, @postalState,
                @postalPostcode, @postalCountry, @residentStreet,
                @residentSuburb, @residentState, @residentPostcode,
                @residentCountry, @phoneNumber, @mobileNumber,
                @emailAddress)
936     SET
937         @userID=@@IDENTITY
938 GO
```

```sql

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.DeleteAETAnswer    Script Date
    : 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.DeleteAETAnswer
    (
        @aetQID BIGINT,
        @classID CHAR(7),
        @userID BIGINT,
        @aetQType VARCHAR(20)
    )
AS
    DELETE FROM AETAnswers
    WHERE aetQID=@aetQID AND classID=@classID AND userID=@userID AND
        aetQType=@aetQType
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.DeleteAETQuestion    Script
    Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.DeleteAETQuestion
    (
        @aetQID BIGINT,
        @classID CHAR(7)
    )
AS
    DELETE FROM AETQuestions
    WHERE aetQID=@aetQID AND classID=@classID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.DeleteClass    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.DeleteClass
    (
        @classID CHAR(10)
    )
AS
    DELETE FROM Classes
    WHERE classID=@classID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.DeleteRole    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.DeleteRole
    (
        @roleID VARCHAR(50)
    )
AS
    DELETE FROM Roles
    WHERE roleID=@roleID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.DeleteThreadSub    Script Date
    : 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.DeleteThreadSub
    (
        @classID CHAR(7),
        @userID INT,
        @threadID INT
    )
AS
    DELETE FROM ThreadSubscriptions
```

```
    WHERE classID=@classID AND userID=@userID AND threadID=@threadID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.DeleteUser    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.DeleteUser
    (
        @userID INT
    )
AS
    DELETE FROM Users
    WHERE userID=@userID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.FindClass    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindClass
    (
        @classID CHAR(10)
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [Classes!1],
        NULL AS [Class!2!classID],
        NULL AS [Class!2!Name!element],
        NULL AS [Class!2!denyOtherUsers]
    UNION ALL
    SELECT
        2 AS Tag,
        1 AS Parent,
        NULL,
        Classes.classID,
        Classes.className,
```

```
            Classes.denyOtherUsers
        FROM Classes
        WHERE classID=@classID
FOR XML EXPLICIT
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.FindClasses    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindClasses
    (
        @userID INT
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [Classes!1],
        NULL AS [Class!2!classID],
        NULL AS [Class!2!Name!element],
        NULL AS [Class!2!denyOtherUsers]
    UNION ALL
    SELECT
        2 AS Tag,
        1 AS Parent,
        NULL,
        Classes.classID,
        Classes.className,
        Classes.denyOtherUsers
        FROM ClassUsers CROSS JOIN Classes
    WHERE ClassUsers.userID=@userID AND ClassUsers.classID=Classes.
        classID
FOR XML EXPLICIT
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

```

```
/****** Object:  Stored Procedure dbo.FindRolesIDs    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindRolesIDs
    (
        @userID INT
    )
AS
    SELECT roleID
    FROM ClassUsers
    WHERE userID=@userID
    UNION
        SELECT roleID
        FROM OtherUsers
        WHERE userID=@userID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.FindShortAETAnswer    Script
    Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindShortAETAnswer
    (
        @aetQID INT,
        @userID BIGINT,
        @classID CHAR(7),
        @aetQType VARCHAR(20)
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [ShortAETAnswers!1],
        NULL AS [ShortAETAnswer!2!aetQID],
        NULL AS [ShortAETAnswer!2!classID],
        NULL AS [ShortAETAnswer!2!userID],
        NULL AS [ShortAETAnswer!2!submittedDate],
        NULL AS [ShortAETAnswer!2!marked]
UNION ALL
    SELECT
        2 AS Tag,
        1 AS Parent,
        NULL,
        AETAnswers.aetQID,
        AETAnswers.classID,
        AETAnswers.userID,
        AETAnswers.aetASubmittedDate,
        AETAnswers.aetAMarked
```

```
1201        FROM AETAnswers
1202        WHERE AETAnswers.aetQID=@aetQID AND AETAnswers.classID=@classID
                AND AETAnswers.userID=@userID AND aetQType=@aetQType
1203    FOR XML EXPLICIT
1204    GO
1205
1206    SET QUOTED_IDENTIFIER OFF
1207    GO
1208    SET ANSI_NULLS ON
1209    GO
1210
1211    SET QUOTED_IDENTIFIER ON
1212    GO
1213    SET ANSI_NULLS ON
1214    GO
1215
1216    /****** Object:  Stored Procedure dbo.FindShortAETAnswers    Script
            Date: 5/08/2004 4:08:29 PM ******/
1217    CREATE PROCEDURE dbo.FindShortAETAnswers
1218        (
1219            @aetQID INT,
1220            @classID CHAR(7),
1221            @aetQType VARCHAR(20)
1222        )
1223    AS
1224        SELECT
1225            1 AS Tag,
1226            NULL AS Parent,
1227            NULL AS [ShortAETAnswers!1],
1228            NULL AS [ShortAETAnswer!2!aetQID],
1229            NULL AS [ShortAETAnswer!2!classID],
1230            NULL AS [ShortAETAnswer!2!userID],
1231            NULL AS [ShortAETAnswer!2!submittedDate],
1232            NULL AS [ShortAETAnswer!2!marked]
1233    UNION ALL
1234        SELECT
1235            2 AS Tag,
1236            1 AS Parent,
1237            NULL,
1238            AETAnswers.aetQID,
1239            AETAnswers.classID,
1240            AETAnswers.userID,
1241            AETAnswers.aetASubmittedDate,
1242            AETAnswers.aetAMarked
1243        FROM AETAnswers
1244        WHERE AETAnswers.aetQID=@aetQID AND AETAnswers.classID=@classID
                AND AETAnswers.aetQType=@aetQType
1245    FOR XML EXPLICIT
1246    GO
1247
1248    SET QUOTED_IDENTIFIER OFF
1249    GO
1250    SET ANSI_NULLS ON
1251    GO
1252
```

```
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.FindShortAETQuestion    Script
    Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindShortAETQuestion
    (
        @aetQID INT,
        @classID CHAR(7),
        @aetQType VARCHAR(20)
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [ShortAETQuestions!1],
        NULL AS [ShortAETQuestion!2!aetQID],
        NULL AS [ShortAETQuestion!2!classID],
        NULL AS [ShortAETQuestion!2!Title!element],
        NULL AS [ShortAETQuestion!2!Description!element],
        NULL AS [ShortAETQuestion!2!aetType],
        NULL AS [ShortAETQuestion!2!questionType],
        NULL AS [ShortAETQuestion!2!viewQuestionDateTime],
        NULL AS [ShortAETQuestion!2!dueDateTime],
        NULL AS [ShortAETQuestion!2!haltSubmissionDateTime],
        NULL AS [ShortAETQuestion!2!viewMarkedAnswersDateTime],
        NULL AS [ShortAETQuestion!2!amountOfQuestions]
    UNION ALL
    SELECT
        2 AS Tag,
        1 AS Parent,
        NULL,
        AETQuestions.aetQID,
        AETQuestions.classID,
        AETQuestions.aetQTitle,
        AETQuestions.aetQDescription,
        AETQuestions.aetQType,
        AETQuestions.aetQQuestionType,
        AETQuestions.aetQViewQuestionDateTime,
        AETQuestions.aetQDueDateTime,
        AETQuestions.aetQHaltSubmissionDateTime,
        AETQuestions.aetQViewMarkedAnswersDateTime,
        AETQuestions.aetQAmountOfQuestions
    FROM AETQuestions
    WHERE aetQID=@aetQID AND classID=@classID AND aetQType=@aetQType
    FOR XML EXPLICIT
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

```
1307 SET QUOTED_IDENTIFIER ON
1308 GO
1309 SET ANSI_NULLS ON
1310 GO
1311
1312 /****** Object:  Stored Procedure dbo.FindShortAETQuestions
         Script Date: 5/08/2004 4:08:29 PM ******/
1313 CREATE PROCEDURE dbo.FindShortAETQuestions
1314     (
1315         @classID CHAR(7),
1316         @aetQType VARCHAR(20)
1317     )
1318 AS
1319     SELECT
1320         1 AS Tag,
1321         NULL AS Parent,
1322         NULL AS [ShortAETQuestions!1],
1323         NULL AS [AETQuestion!2!aetQID],
1324         NULL AS [AETQuestion!2!classID],
1325         NULL AS [AETQuestion!2!Title!element],
1326         NULL AS [AETQuestion!2!Description!element],
1327         NULL AS [AETQuestion!2!aetType],
1328         NULL AS [AETQuestion!2!questionType],
1329         NULL AS [AETQuestion!2!viewQuestionDateTime],
1330         NULL AS [AETQuestion!2!dueDateTime],
1331         NULL AS [AETQuestion!2!haltSubmissionDateTime],
1332         NULL AS [AETQuestion!2!viewMarkedAnswersDateTime],
1333         NULL AS [AETQuestion!2!amountOfQuestions]
1334     UNION ALL
1335     SELECT
1336         2 AS Tag,
1337         1 AS Parent,
1338         NULL,
1339         AETQuestions.aetQID,
1340         AETQuestions.classID,
1341         AETQuestions.aetQTitle,
1342         AETQuestions.aetQDescription,
1343         AETQuestions.aetQType,
1344         AETQuestions.aetQQuestionType,
1345         AETQuestions.aetQViewQuestionDateTime,
1346         AETQuestions.aetQDueDateTime,
1347         AETQuestions.aetQHaltSubmissionDateTime,
1348         AETQuestions.aetQViewMarkedAnswersDateTime,
1349         AETQuestions.aetQAmountOfQuestions
1350     FROM AETQuestions
1351     WHERE classID=@classID AND aetQType=@aetQType
1352     FOR XML EXPLICIT
1353 GO
1354
1355 SET QUOTED_IDENTIFIER OFF
1356 GO
1357 SET ANSI_NULLS ON
1358 GO
1359
1360 SET QUOTED_IDENTIFIER ON
```

```
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.FindUser    Script Date:
    5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindUser
    (
        @userID INT
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [Users!1],
        NULL AS [User!2!userID],
        NULL AS [User!2!Password!element],
        NULL AS [User!2!FirstName!element],
        NULL AS [User!2!MiddleName!element],
        NULL AS [User!2!LastName!element],
        NULL AS [User!2!dob],
        NULL AS [PostalAddress!3],
        NULL AS [PostalAddress!3!Street!element],
        NULL AS [PostalAddress!3!Suburb!element],
        NULL AS [PostalAddress!3!State!element],
        NULL AS [PostalAddress!3!Postcode!element],
        NULL AS [PostalAddress!3!Country!element],
        NULL AS [ResidentAddress!4],
        NULL AS [ResidentAddress!4!Street!element],
        NULL AS [ResidentAddress!4!Suburb!element],
        NULL AS [ResidentAddress!4!State!element],
        NULL AS [ResidentAddress!4!Postcode!element],
        NULL AS [ResidentAddress!4!Country!element],
        NULL AS [User!2!PhoneNumber!element],
        NULL AS [User!2!MobileNumber!element],
        NULL AS [User!2!EmailAddress!element]
    UNION ALL
        SELECT
            2 AS Tag,
            1 AS Parent,
            NULL,
            Users.userID,
            Users.password,
            Users.firstName,
            Users.middleName,
            Users.lastName,
            Users.dob,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
```

```
1415                NULL,
1416                NULL,
1417                NULL,
1418                NULL,
1419                 Users.phoneNumber,
1420                 Users.mobileNumber,
1421                 Users.emailAddress
1422            FROM Users
1423            WHERE Users.userID=@userID
1424            UNION ALL
1425                SELECT
1426                     3 AS Tag,
1427                     2 AS Parent,
1428                    NULL,
1429                    NULL,
1430                    NULL,
1431                    NULL,
1432                    NULL,
1433                    NULL,
1434                    NULL,
1435                    NULL,
1436                     Users.postalStreet,
1437                     Users.postalSuburb,
1438                     Users.postalState,
1439                     Users.postalPostcode,
1440                     Users.postalCountry,
1441                    NULL,
1442                     Users.residentStreet,
1443                     Users.residentSuburb,
1444                     Users.residentState,
1445                     Users.residentPostcode,
1446                     Users.residentCountry,
1447                    NULL,
1448                    NULL,
1449                    NULL
1450                FROM Users
1451                WHERE Users.userID=@userID
1452 FOR XML EXPLICIT
1453 GO
1454
1455 SET QUOTED_IDENTIFIER OFF
1456 GO
1457 SET ANSI_NULLS ON
1458 GO
1459
1460 SET QUOTED_IDENTIFIER ON
1461 GO
1462 SET ANSI_NULLS ON
1463 GO
1464
1465 /****** Object:  Stored Procedure dbo.FindUsers    Script Date:
         5/08/2004 4:08:29 PM ******/
1466 CREATE PROCEDURE dbo.FindUsers
1467     (
1468         @classID CHAR(7)
```

```
    )
AS
    SELECT
        1 AS Tag,
        NULL AS Parent,
        NULL AS [Users!1],
        NULL AS [User!2!userID],
        NULL AS [User!2!Password!element],
        NULL AS [User!2!FirstName!element],
        NULL AS [User!2!MiddleName!element],
        NULL AS [User!2!LastName!element],
        NULL AS [User!2!dob],
        NULL AS [PostalAddress!3],
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL AS [ResidentAddress!3],
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL AS [User!2!PhoneNumber!element],
        NULL AS [User!2!MobileNumber!element],
        NULL AS [User!2!EmailAddress!element]
    UNION ALL
        SELECT
            2 AS Tag,
            1 AS Parent,
            NULL,
            Users.userID,
            Users.password,
            Users.firstName,
            Users.middleName,
            Users.lastName,
            Users.dob,
            NULL,
            NULL AS [PostalAddress!3!Street!element],
            NULL AS [PostalAddress!3!Suburb!element],
            NULL AS [PostalAddress!3!State!element],
            NULL AS [PostalAddress!3!Postcode!element],
            NULL AS [PostalAddress!3!Country!element],
            NULL,
            NULL AS [ResidentAddress!3!Street!element],
            NULL AS [ResidentAddress!3!Suburb!element],
            NULL AS [ResidentAddress!3!State!element],
            NULL AS [ResidentAddress!3!Postcode!element],
            NULL AS [ResidentAddress!3!Country!element],
            Users.phoneNumber,
            Users.mobileNumber,
            Users.emailAddress
        FROM Classes CROSS JOIN Users
        WHERE Classes.classID=@classID
```

```
1524            UNION ALL
1525                SELECT
1526                    3 AS Tag,
1527                    2 AS Parent,
1528                    NULL,
1529                    NULL,
1530                    NULL,
1531                    NULL,
1532                    NULL,
1533                    NULL,
1534                    NULL,
1535                    NULL,
1536                    Users.postalStreet,
1537                    Users.postalSuburb,
1538                    Users.postalState,
1539                    Users.postalPostcode,
1540                    Users.postalCountry,
1541                    NULL,
1542                    Users.residentStreet,
1543                    Users.residentSuburb,
1544                    Users.residentState,
1545                    Users.residentPostcode,
1546                    Users.residentCountry,
1547                    NULL,
1548                    NULL,
1549                    NULL
1550                FROM Classes CROSS JOIN Users
1551                WHERE Classes.classID=@classID
1552 FOR XML EXPLICIT
1553 GO
1554
1555 SET QUOTED_IDENTIFIER OFF
1556 GO
1557 SET ANSI_NULLS ON
1558 GO
1559
1560 SET QUOTED_IDENTIFIER ON
1561 GO
1562 SET ANSI_NULLS ON
1563 GO
1564
1565 /****** Object:  Stored Procedure dbo.IsInClass    Script Date:
         5/08/2004 4:08:29 PM ******/
1566 CREATE PROCEDURE dbo.IsInClass
1567     (
1568         @userID INT,
1569         @classID CHAR(7)
1570     )
1571 AS
1572     SELECT roleID
1573     FROM ClassUsers
1574     WHERE userID=@userID AND classID=@classID
1575     ORDER BY roleID
1576 GO
1577
```

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```