**E-Learning Facility - n-tier web application**

***1.1. Project Overview***

**1.1.1. Outline and Objectives**

The purpose of this project is to build a dynamic data-driven web application written in ASP.NET and C# using the techniques that you have learned so far throughout the module. The main focuses of the project are persistent state, access control, ADO.NET database operations, validation and AJAX. Another important requirement is that your web application has an underlying n-tier architecture, and utilises principles of abstraction, decomposition and modularisation, as has been taught during the module.

The web application that you will develop will form the foundations of an E-Learning Facility for students and tutors. For the purpose of this project, you have been specified to develop some of the basic essential functionality for this web application, which may then be further developed as necessary.

You have been provided with a prebuilt SQL Server Database (*see Section 1.2*) that you must use as a means of data storage and retrieval. Most of the required functionality has been comprehensively documented for you as a requirements specification consisting of use cases, and user interface designs (*see Section 2*). You should strictly adhere to this documentation and use it as a basis for completing PART A of the project. PART B of the project provides you with more freedom to show creativity and innovation.

Following submission, you will be required to demonstrate your achievements to your tutor. This will allow your tutor to more easily assess your progress, and clarify any additions you may have added (during PART B). You will also be asked questions about the syntax and semantics of your source code. The level of understanding you display and the technical correctness of your answers will help to inform your tutors decision for a given assessment criteria. Once the viva has finished, you should get immediate feedback about your provisional (pre-moderated) mark for this project

*Please turn over...*

**1.1.2. Task Overview and Assessment Criteria**

***PART A (85%)***

**Core use cases (70%)**

You have been provided with several well documented use cases (*see Section 2*), that inform you of the required functionality for the web application:-

|  |  |  |  |
| --- | --- | --- | --- |
| (UC001) Login User |  | (UC005A) Load Tutor Details |  |
| (UC002A) Load Available Courses |  | (UC005B) Remove Student |  |
| (UC002B) Register Student |  | (UC006) Update User Password |  |
| (UC003) Display User Account |  | (UC007A) Load Tutor Courses |  |
| (UC004) Logout User |  | (UC007B) Update Tutor Course |  |
|  |  | (UC008A) Load Student Details |  |
|  |  | (UC008B) Show Tutor Email Address |  |
|  |  | (UC008C) Load Student Modules |  |

You should attempt to develop all of these use cases. Should you do so, your mark will be based on their design, correctness, and fitness for purpose. Should you not be able to develop any use cases, then you will lose some marks for this area of PART A.

**Note**: Whilst you are recommended to get the registration and login facilities working at first, you do not have to develop the other use cases in the order that they are specified. For example, you may struggle at first to develop UC005A and UC005B but be able to develop UC006 and UC007A. Saying this, whilst the additional use cases do not have to be developed in any given order, ***where a use case number has multiple use cases (e.g. 8A, 8B, 8C), these will generally need to be developed in order***. For example, take UC007A and UC007B - UC007B requires UC007A to have been developed, otherwise it will be impossible to develop UC007B.

**N-tier architecture design and comments (15%)**

You should make use of component-based programming and logically separate different components into an underlying n-tier design as taught during this module.

*A typical n-tier design with a dedicated data access channel*

You will be assessed on your ability to correctly deploy this approach within your solution, and identify logical areas of abstraction, which will ultimately make your code less repetitive, more easily interchangeable, and simpler to maintain.

You should also write comments within the classes of your Presentation Tier (code-behind files), Middle Tier, and Data Access Layer. These comments should give a brief overview of the class, explain the purpose of each constructor and method (including their parameters and any return value), and outline any key areas of source code within your functions/methods that you feel require further elaboration.

***PART B (15%)***

***Note***: Part B will require you to create and submit a Microsoft Word document (*report.docx*).

**Applying AJAX (5%)**

You should apply Asynchronous JavaScript and XML (AJAX) functionality in appropriate places within the E-Learning web application. The way you should go about this is purposely specified in less formal detail to give you a chance to show creativity and innovation.

Make use of the AJAX Extensions web server controls, in particular the UpdatePanel control and where appropriate the UpdateProgress control. Where you apply AJAX is entirely up to you. It should be an appropriate use of AJAX carried out in a correct way, e.g. ensuring an external trigger is associated with an UpdatePanel correctly. You should be careful to not introduce AJAX behaviour where it is inappropriate, and should aim as a minimum to have it applied in two locations within the application.

You should submit a brief overview of how and where (in your web application) you have applied AJAX as part of the **report.docx** attachment inside of your website directory. You may be asked questions about this during your viva.

**Secure password storage (5%)**

You can achieve further marks for integrating a means of secure password storage into the E-Learning web application. To do this you will need to update the database slightly, and apply password hashing and salting procedures appropriately. You will need to research into an appropriate means of achieving this within ASP.NET and C#, and you should reference any source code that you use from elsewhere. It is expected that you will (at the very least) adapt this source code to meet the specific requirements of your own application.

This additional functionality should be integrated into your existing n-tier design, and you should update the relevant use cases from this specification to include additional steps and/or extensions that account for these procedures being undertaken. You should submit these modified use cases as part of the **report.docx** attachment inside of your website directory. You may be asked questions about this during your viva.

**Additional use cases (5%)**

You can add up to THREE additional use cases to those which already exist. The precise behaviour which these use cases define is entirely down to you, however, they should logically have a purpose within the context of the existing E-Learning facility, should not repeat any existing features, and should ideally show further creativity and innovation. As an example, you could think about sorting, filtering and paging through data (e.g. the modules).

The additional functionality encapsulated by these use cases should be integrated into your existing n-tier design. You may add additional Web Forms, or integrate features sensibly into existing Web Forms. You may optionally wish to update the database design, however, if so, it should be done correctly such that all use cases are still designed, behave, and operate in the same way, and principles of normalisation still exist throughout the database design.

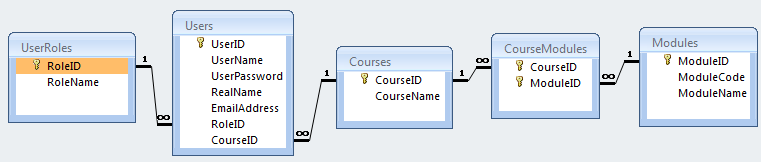
You should submit these new use cases (formally documented) with a parallel brief explanation, and any changes to the database, as part of the **report.docx** attachment inside of your website directory. You may be asked questions about this during your viva.

***1.2. The SQL Server Database***

The SQL Server database (*university.mdf*) that will be used for the project has been created for you. All the required tables (and fields) for Part A have already been generated. Whilst developing the web application you will insert and delete records when testing required functionality.

**1.2.1. The Entity Relationship Diagram (ERD)**

This ERD allows you to see the tables, fields and relationships.



**1.2.2. Table Relationships**

An explanation of the tables and the relationships between them:-

* The User has a User Role, which may either be Student or Tutor.
* A Student studies one Course and a Course is studied by many Students.
* A Tutor teaches on one Course and a Course is taught by many Tutors.
* A Course consists of many Modules and a Module can be taught across many Courses.
* This many-to-many relationship is broken down into the table CourseModules, which has a composite key.

It is worth noting that this database design is simplified compared to that used in practice for such a system in a University. For example, a Tutor could teach on multiple Courses. For the purpose of this project however you should develop the web application based on the database design that has been provided.

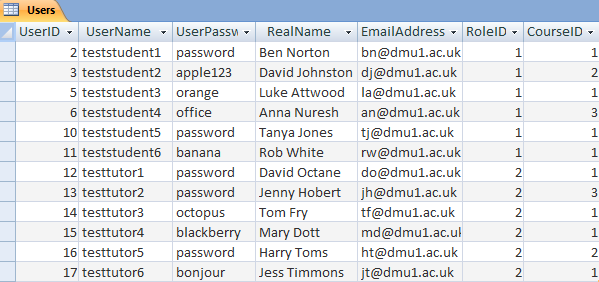
**1.2.3. Field Data Types and Constraints**

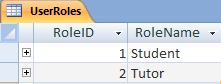
On most occasions the data types chosen for the fields should be fairly obvious, however, to look at this in more detail, you may use the Server Explorer within Visual Studio.

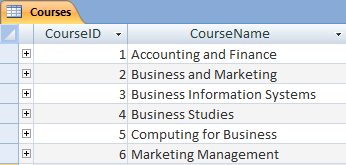
*Please turn over...*

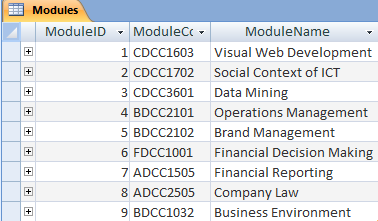
**1.2.4. Existing Table Data**

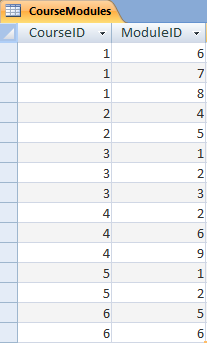
These screenshots show the current data in the SQL Server database:-











*Please turn over..*

**2. Functional Requirements Specification and UI designs**

This section provides all of the documentation that you should need to successfully complete PART A of this project. You have been provided with several Web Forms (e.g. UserLogin.aspx). Each of these has one or more associated use cases, which outline the required functionality for that web page. This has therefore provided a logical way to document the requirements.

The use cases have been written in the format and layout suggested by Martin Fowler 2004 in the book "UML Distilled". Each use case can be looked at as a sequence of interactions between the primary actor (e.g. User, Student, Tutor, etc) and the System. The primary actor will usually interact with the user interface (e.g. input and submit some data) while the system will process and run the application.

The Main Success Scenario lists the steps required for the use case action to be completed successfully. The Extensions are a list of errors that could occur and should thus be validated by your web application. Each extension is directly linked to a specific action by its number (e.g. 3). Where there is more than one error that could occur for a single action, these are separated by using an alphabetic character (e.g. 3a, 3b, 3c, etc).

You should strictly follow the use case guidelines. For a use case to be completed and fully operational the Main Success Scenario should work and all Extensions should be validated as specified. On occasions you may take assumptions about specific design or implementation details.

There is a screenshot taken of Visual Studio showing the design view for each Web Form.

If you wish for any use cases to be clarified, please consult your lab tutor.

*Please turn overleaf to view the start of the documentation.*

***2.1. UserLogin.aspx***

**2.1.1. Use Case(s)**

**(UC001) Login User**

Main Success Scenario:

1. System loads the page.
2. User enters their username and password and then clicks login Button.
3. System validates input fields.
4. System queries database to authenticate the username and password.
5. System creates Session objects for UserID, UserName, CourseID and RoleID.
6. User is redirected to UserAccount.aspx.

Extensions:

3a: Entered username length is less than 5 or greater than 20 characters.

.1: Output appropriate error message.

3b: Entered password length is less than 6 characters .

.1: Output appropriate error message.

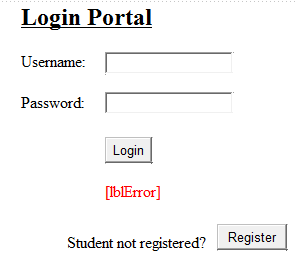
4a: Database query returns no rows - details must be incorrect.

.1: Output appropriate error message.

4b: Catch database error.

.1: Output appropriate error message.

**2.1.2. Design View**



***2.2. StudentRegistration.aspx***

**2.2.1. Use Case(s)**

**(UC002A) Load Available Courses**

Main Success Scenario:

1. System ensures page request is not a postback.
2. System queries database and populates DropDownList with courses.
3. System loads the page.

Extensions:

2a: Catch database error.

.1: Output appropriate error message.

**(UC002B) Register Student**

Main Success Scenario:

1. Student enters a username, password, password confirmation, full name, email address, selects a course and then clicks register Button.
2. System validates input fields.
3. System queries database to ensure username does not already exist.
4. System inserts new student user into the database.
5. Student is redirected to UserLogin.aspx.

Extensions:

2a: Entered username length is less than 5 or greater than 20 characters.

.1: Output appropriate error message.

2b: Entered password length is less than 6 characters.

.1: Output appropriate error message.

2c: Entered password confirmation not equal to password input.

.1: Output appropriate error message.

2d: Entered real name field is empty.

.1: Output appropriate error message.

2e: Entered email field is empty or the email does not contain the string "dmu1.ac.uk".

.1: Output appropriate error message.

3a: Database query returns a record - username must already exist.

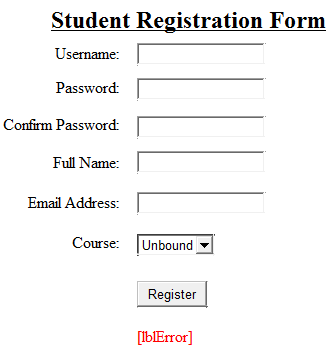
.1: Output appropriate error message.

4a: Catch database error.

.1: Output appropriate error message.

*Please turn over...*

**2.2.2. Design View**



***2.3. UserAccount.aspx***

**2.3.1. Use Case(s)**

**(UC003) Display User Account**

Main Success Scenario:

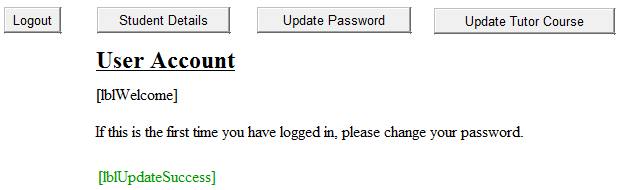
1. System ensures a user is logged in (i.e. a Session exists).
2. System displays appropriate details depending upon Session RoleID value. If Student go to 3, if Tutor go to 4.
3. Student user:-
   * Will display their username and their role name in a Label.
   * Will display a Button that redirects to StudentDetails.aspx.
   * Will display a Button that redirects to ChangePassword.aspx.
   * Will display a Button that redirects to Logout.aspx.
   * Will NOT display change password notification Label.
   * Will NOT display Update Tutor Course Button.
4. Tutor user:-
   * Will display their username and their role name in a Label.
   * Will display a Button that redirects to TutorDetails.aspx.
   * Will display a Button that redirects to ChangePassword.aspx.
   * Will display a Button that redirects to Logout.aspx.
   * Will display change password notification Label.
   * Will display a Button that redirects to UpdateTutorCourse.aspx.
5. System loads the page.

Extensions:

1a: User not logged in.

.1: Redirect user to UserLogin.aspx.

**2.3.2. Design View**



***2.4. Logout.aspx***

**2.4.1. Use Case(s)**

**(UC004) Logout User**

Main Success Scenario:

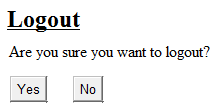
1. User clicks logout Button on UserAccount.aspx page.
2. User is redirected to Logout.aspx.
3. System displays "Yes" and "No" buttons.
4. User clicks "Yes" Button.
5. System removes all Session object information.
6. User is redirected to UserLogin.aspx.

Extensions:

4a: User clicks "No" Button.

.1: Redirect user back to UserAccount.aspx.

**2.4.2. Design View**



***2.5. TutorDetails.aspx***

**2.5.1. Use Case(s)**

**(UC005A) Load Tutor Details**

Main Success Scenario:

1. System ensures a user is logged in (i.e. a Session exists).
2. System ensures logged in user is a Tutor.
3. System ensures page request is not a postback.
4. System queries the Courses table in the database, displaying the tutor's course name in a TextBox.
5. System queries the Users table in the database displaying all student users on the same course as the tutor in a ListBox.
6. System loads the page.

Extensions:

1a: User not logged in.

.1: Redirect user to UserLogin.aspx.

2a: User not a Tutor.

.1: Redirect user to UserAccount.aspx.

4a: Catch database error.

.1: Output appropriate error message.

5a: Catch database error.

.1: Output appropriate error message.

**(UC005B) Remove Student**

Main Success Scenario:

1. Tutor selects student from ListBox and clicks remove student Button.
2. System connects to the database and deletes the selected student
3. System removes the selected item from the ListBox.
4. System outputs a success Label message.

Extensions:

1a: No item selected in ListBox.

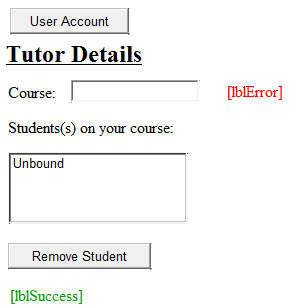
.1: Output appropriate error message.

2a: Catch database error.

.1: Output appropriate error message.

*Please turn over...*

**2.5.2. Design View**



***2.6. UpdatePassword.aspx***

**2.6.1. Use Case(s)**

**(UC006) Update User Password**

Main Success Scenario:

1. System ensures a user is logged in (i.e. a Session exists).
2. System loads the page.
3. User enters current password, new password and new password confirmation then clicks update password Button.
4. System validates input fields.
5. System queries database to ensure current password is correct.
6. System updates user's password in database.
7. Student is redirected to UserAccount.aspx?UpdateSuccess=Password.
8. System processes query string and shows a success label message.

Extensions:

1a: User not logged in.

.1: Redirect user to UserLogin.aspx.

4a: Entered current password length is less than 6 characters.

.1: Output appropriate error message.

4b: Entered new password length is less than 6 characters.

.1: Output appropriate error message.

4c: Entered new password confirmation not equal to new password input.

.1: Output appropriate error message.

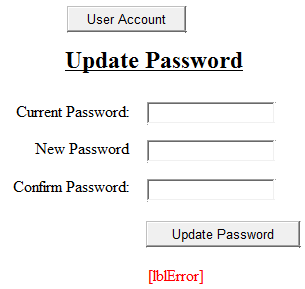
5a: Inputted current password does not match that in database for this user.

.1: Output appropriate error message.

6a: Catch database error.

.1: Output appropriate error message.

**2.6.2. Design View**



***2.7. UpdateTutorCourse.aspx***

**2.7.1. Use Case(s)**

**(UC007A) Load Tutor Courses**

Main Success Scenario:

1. System ensures a user is logged in (i.e. a Session exists).
2. System ensures logged in user is a Tutor.
3. System ensures page request is not a postback.
4. System queries the Courses table in the database, displaying current course in a TextBox and all other courses in a ListBox.
5. System loads the page.

Extensions:

1a: User not logged in.

.1: Redirect user to UserLogin.aspx.

2a: User not a Tutor.

.1: Redirect user to UserAccount.aspx.

4a: Catch database error.

.1: Output appropriate error message.

**(UC007B) Update Tutor Course**

Main Success Scenario:

1. Tutor selects course from ListBox and clicks update course Button.
2. System connects to database and updates tutor's course.
3. System updates Session CourseID value.
4. Tutor is redirected to UserAccount.aspx?UpdateSuccess=Course.
5. System processes query string and shows a success label message.

Extensions:

1a: No item selected in ListBox.

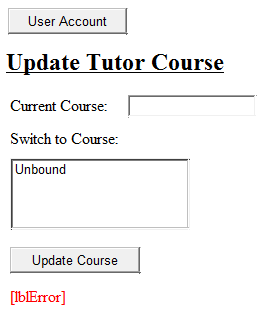
.1: Output appropriate error message.

2a: Catch database error.

.1: Output appropriate error message.

*Please turn over...*

**2.7.2. Design View**



***2.8. StudentDetails.aspx***

**2.8.1. Use Case(s)**

**(UC008A) Load Student Details**

Main Success Scenario:

1. System ensures a user is logged in (i.e. a Session exists).
2. System ensures logged in user is a Student.
3. System ensures page request is not a postback.
4. System queries the Courses table in the database, displaying the student's course name in a TextBox.
5. System queries the Users table in the database, displaying all tutor users on the same course as the student in a ListBox.
6. System loads the page.

Extensions:

1a: User not logged in.

.1: Redirect user to UserLogin.aspx.

2a: User not a Student.

.1: Redirect user to UserAccount.aspx.

4a: Catch database error.

.1: Output appropriate error message.

5a: Catch database error.

.1: Output appropriate error message.

**(UC008B) Show Tutor Email Address**

Main Success Scenario:

1. Student selects tutor from ListBox and clicks show email Button.
2. System retrieves the email address from the selected item in the ListBox.
3. System outputs the email address in a Label.

Extensions:

1a: No item selected in ListBox.

.1: Output appropriate error message.

***PLEASE NOTE (UC008C) IS AN EXTENSION TO (UC008A) AND WOULD OCCUR BETWEEN STEPS 5 AND 6 OF (UC008A).***

**(UC008C) Load Student Modules**

Main Success Scenario:

1. System queries the Modules and CourseModules tables in the database retrieving all modules that are part of the student's course.
2. System displays each module (ModuleName and ModuleCode) in a Repeater.

Extensions:

1a: Catch database error.

.1: Output appropriate error message.

*Please turn over...*

**2.8.2. Design View**

