**IT8118 Advanced Programming Project**

**Semester 1 2018-19**

**Project Design due (Group):** Thur 29th Nov 23:55

**Project Code due (Group):** Mon 31st Dec 14:00

**Critical Reflection due (Individual)**: Mon 31st Dec 23.55

**Grade:** 35% of Course Grade

**Important Notes:**

* A Word document template will be provided for the Project Design. The completed document should be uploaded to Moodle (one per group) by the deadline.
* A Project Group folder will be provided on the STUDSQL server for the Project Code. Work should be completed on the server by the deadline.
* A Word document template will be provided for the Critical Reflection. The completed document should be uploaded to Moodle (one per student) by the deadline.
* Requests for extensions should be made before the deadline to Course Coordinator. Extensions will only be approved with valid reasons, up to a maximum of 2 calendar days. Extensions are for the whole group. You are only permitted a maximum of one extension per course per semester. There is no extension available for the Project Design sub-component
* If the assessment is submitted late the maximum result the group can achieve is 60%
* The cut off time for submitting an assessment will be 3 calendar days after the assessment is due. A group submitting after 3 calendar days will get 0%.

**Learning Outcomes assessed:**

|  |
| --- |
| 1. Design and implement desktop applications that access and manipulate a database |
| 2. Design and implement web applications that access and manipulate a database |
| 3. Use object oriented techniques to design and implement custom database entity classes |
| 4. Manage a multi-tier application which uses custom entity classes to access and manipulate a database |

**Project Description**

**Project Overview:**

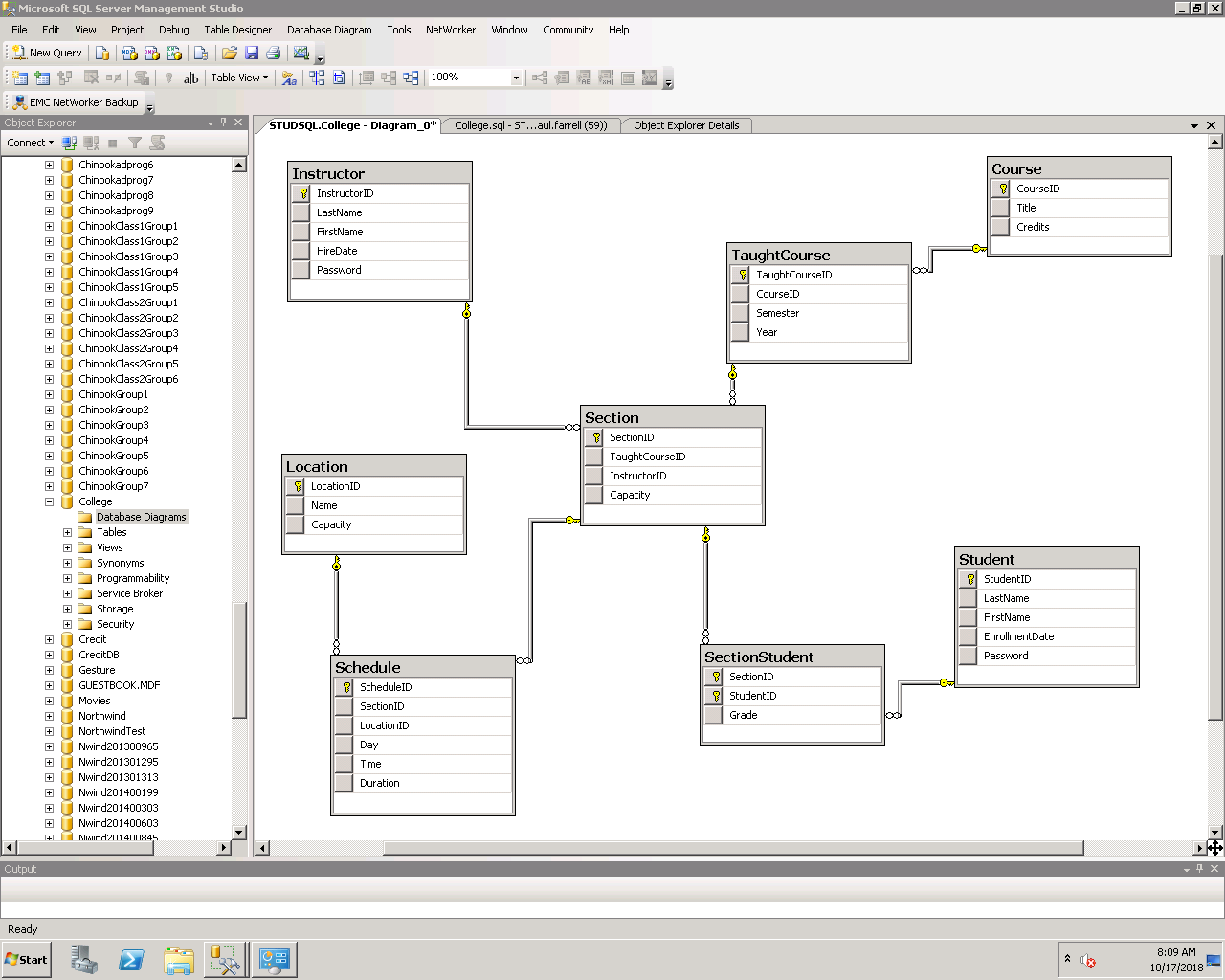
ACME College is an imaginary new third level technical college. The college requires a system which will allow administrative users to manage data for the college and also allow instructors and students to view and manage their own data.

**Project Objective**: Design and develop a distributed application which provides functionality for the ACME College. A database is provided with sample data.

**IDE**: Visual Studio 2017 **Language**: Visual C#

**Database**: College SQL Server database containing sample data. There are no triggers in the database for deletes.

**College Data Model:**

**

**Business Scenario:**

The new college has developed a strategy for the managing its data and a database has been developed and populated with sample data. Management has hired a third party software company to develop the software system which will be needed by the new college. Following internal discussions a meeting was held with the software developer and the following briefing was given:

An internal desktop system will be made available only to administrators and management. The following functionality is required:

* Add/Edit functionality for Courses, TaughtCourses, Instructors, Sections, Students, Schedules and Locations.
* Delete functionality for TaughtCourses, Instructors, Sections, Students and Schedules.
* Management would like to see reports which show:
  + Total scheduled hours. Total scheduled hours by Location, Section, Instructor, Course, Student
  + Average grade for the college. Average grades by Section, Student and Course
* Management would like the application to block double scheduling, i.e.
  + When a Schedule is being added to a Section:
    - The system will not allow a Location to be double scheduled, i.e. if there is already a Section scheduled in that Location on the same Day at the same Time the scheduling will not be allowed
    - The system will not allow an Instructor to be double scheduled, i.e. if the Instructor for the Section is already scheduled to teach on that Day at that time the scheduling will not be allowed
    - The system will not allow a Location to be used for a section where the capacity of the section is greater than the capacity of the location.
* The system may be extended in the future to manage data for multiple semesters, but for now the current system needs to only manage data for the Current Semester.

A website will be required to allow instructors and students to manage their data. The following functionality is required:

* Students and Instructors can log in
* A Student can view his/her timetable and grades
* A student can enrol in any Section which has been scheduled
* Management would like the application to block double scheduling and limit scheduling by location and student, i.e.
  + When a student is being enrolled:
    - The system will not allow a Student to be double scheduled, i.e. if the Student for the Section is already scheduled to attend another class on that Day at that time the enrollment will not be allowed
    - The system will not allow the capacity of the Location for the scheduled section to be exceeded
    - The system will not allow the student to be enrolled for more than 20 hours
* An Instructor can view his/her Timetable and Class List and enter Grades for Students.

In order to enhance the maintainability and extensibility of the system the software developer has proposed an an Object Oriented methodology with implementation of an MVC (Model-View-Controller) architecture. All the core functionality for both applications will be created in a Class Library which will provide a Model containing Entity classes and code to manage interactions with the database. The applications will provide the Views (desktop and web forms containing controls to manage interactions with the user). The code behind each form will provide Controllers to manage interactions between Views and the Model using form controls and objects created from classes in the Class Library.

**Required Project Components**

1. A Class Library containing custom entity classes which manage interactions between the client applications and the database (Visual Studio Class Library Project using C#)
2. A Windows Forms application which allows administrative users within the college to manage data surrounding the business and provides reporting to management. The application uses the compiled Class Library to interact with the database. (Visual Studio Windows Forms Project using C#)
3. An ASP.NET Website which allows online users to manage their data. The website uses the compiled Class Library to interact with the database. (Visual Studio ASP.NET Web Application Project using C#)

**Project Groups**

* The required number of students in each project group is 3 as far as possible. Each member of a project group should take a leadership role on one of the project components. However all members of the group should work together on all the components, apart from the Critical Reflection which is individual work.
* The project is mainly assessed as a group project, however the critical reflection deliverable is an individual assessment

**Deliverables**

**1: Code Design Document**

Describes the proposed functionality which will need to be added to the Class Library in order to enable implementation of the functionality required in both the desktop and web applications. Provides a list of methods and shows the class that each method should be added to. Each method should have an appropriate name, a description of its functionality and why it would be needed. Any return type or parameters should be mentioned as well as any relevant information regarding overriding or overloading. **[15 marks]**

**2: Distributed Application**

3 components:AClass Library which provides a Data Access layer for the College database. The class library should either make use of the 4 super classes developed throughout the course (DataList, Item, DataListJoin, ItemJoin) or contain new super classes (this is optional: full marks will be awarded for use of the super classes developed in the course). The Class Library must contain Entity sub classes which map to each table in the database.

A Desktop and a Website application which both make use of the Class Library to access data and provide a user interface to manage user interactions. Each component of the project is created as a separate Visual Studio project. The components of the project work together as a functioning distributed application.

Marks will be awarded for working code, but in addition to that marks will also be awarded for code and GUI quality:

**Code Quality Standards**

* Code uses the Class Library to maximise maintainability
* Methods are used to eliminate repeated code
* Class level variables are used to maintain state and reduce repeated code
* Code is as efficient as possible to achieve the task
* Variables and controls have meaningful names
* Code is commented meaningfully
* Validation is provided

**GUI Quality Standards**

* Finding/Selecting data is enabled effectively
* Filtered results are effectively displayed
* User errors are minimised
* Selected data populates related controls effectively
* User choices are 'remembered' between form/page navigation events
* Selected data populates related forms/pages effectively

**[80 marks]**

**3. Critical Reflection Document**

Note: This is an individual requirement which each student has to write and submit separately.

Write a critical reflection which outlines the following using specific examples from the project:

* A summary of the work which you were mainly responsible for
* Alternative approaches which were considered to coding a particular problem and why you selected a particular approach
* Any challenges you faced (group or individual) and how these were overcome
* What you learned about group project work by working on the project
* What you learned technically by working on the project

**[5 marks]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Discussed** | **Marking Criteria** | **Max** | **Grade** |
| A summary of the work which you were mainly responsible for | For each component: Award up to 2 marks for each element in the assessment rubric | 10 |  |
| Alternative approaches which were considered to coding a particular problem and why you selected a particular approach | 10 |  |
| Any challenges you faced (group or individual) and how these were overcome | 10 |  |
| What you learned about group project work by working on the project | 10 |  |
| What you learned technically by working on the project | 10 |  |
| **Total** | | 50 |  |
| 5 |  |

**Assessment Rubric elements**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| Clarity | No example is provided from the project | An example is provided but is not well explained | An example is provided and is fully explained |
| Relevance | Discussion is irrelevant | Discussion has limited relevance | Keeps the discussion specific to the component being discussed |
| Depth | Discussion is too superficial | Limited depth - avoids complexity | Addresses the complexity of the problem; avoids over-simplifying |
| Breadth | Alternatives not considered | Limited discussion of alternatives | Gives meaningful consideration to alternative approaches |
| Logic | Discussion seems confusing/illogical | Limited line of reasoning/logic | Demonstrates a line of reasoning that is logical, with conclusions or goals that follow clearly from it. |

**Marks Summary**

|  |  |
| --- | --- |
| **Component** | **Total** |
| Code Design Document | 15 |
| Class Library | 30 |
| Desktop | 30 |
| Website | 20 |
| Critical Reflection Document | 5 |
| **Total** | 100 |

**Deliverable 1: Code Design Document for Class Library: Template Document**

**Task:** A partial solution to the code design for the Class Library project is shown, with information provided in the shaded areas. Complete the areas which are not shaded

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | GetMaxID |  |  |
| **Description** | A method in the DataList class which finds the current maximum id value in the table and returns that value, or returns the next value |  |  |
| **Rationale** |  | 3 |  |
| **Parameters** | None |  |  |
| **Return type** |  | 1 |  |
| **Total** | | 4 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue |  |  |
| **Description** |  | 2 |  |
| **Rationale** | Needed to calculate the total of all scheduled hours |  |  |
| **Parameters** | *string* column |  |  |
| **Return type** |  | 1 |  |
| **Total** | | 3 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue |  |  |
| **Description** | A method in the DataList class which calculates the total of a column where another column is equal to a value. Overloaded version of previous TotalValue method |  |  |
| **Rationale** |  | 4 |  |
| **Parameters** | *String* sumColumn*, string* column, *string* value |  |  |
| **Return type** |  | 1 |  |
| **Total** | | 5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue |  |  |
| **Description** |  | 5 |  |
| **Rationale** | Needed to calculate the total scheduled hours for an Instructor |  |  |
| **Parameters** |  | 3 |  |
| **Return type** |  | 1 |  |
| **Total** | | 9 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | TotalValue |  |  |
| **Description** |  | 8 |  |
| **Rationale** |  | 4 |  |
| **Parameters** |  | 3 |  |
| **Return type** |  | 1 |  |
| **Total** | | 16 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | AverageValue |  |  |
| **Description** | A method in the DataList class which calculates the average of a column |  |  |
| **Rationale** |  | 2 |  |
| **Parameters** |  | 0.5 |  |
| **Return type** |  | 1 |  |
| **Total** | | 3.5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | AverageValue |  |  |
| **Description** |  | 6 |  |
| **Rationale** | Needed to calculate the average grade for a Section or Student |  |  |
| **Parameters** |  | 1.5 |  |
| **Return type** |  | 1 |  |
| **Total** | | 8.5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | AverageValue |  |  |
| **Description** |  | 8 |  |
| **Rationale** |  | 3 |  |
| **Parameters** |  | 3 |  |
| **Return type** |  | 1 |  |
| **Total** | | 15 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Exists |  |  |
| **Description** | A method in the DataList class which selects two columns which are equal to two values where another column is equal to a value, and returns true if any rows are found |  |  |
| **Rationale** | Needed to determine if a Location is already scheduled for a particular day and time |  |  |
| **Parameters** | *string* column1*, string* value1*, string column2, string* value2*, string* column*,*  string value |  |  |
| **Return type** |  | 1 |  |
| **Total** | | 1 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Exists |  |  |
| **Description** |  | 7 |  |
| **Rationale** | Needed to determine if an Instructor is already scheduled for a particular day and time |  |  |
| **Parameters** |  | 4.5 |  |
| **Return type** |  | 2 |  |
| **Total** | | 13.5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Exists |  |  |
| **Description** |  | 10 |  |
| **Rationale** |  | 4 |  |
| **Parameters** |  | 6 |  |
| **Return type** |  | 2 |  |
| **Total** | | 22 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Delete |  |  |
| **Description** |  | 5 |  |
| **Rationale** | Needed for deleting a Student. As cascading deletes are not implemented in the database, an attempt to delete a Student with related records in the SectionStudent table will throw an exception. First the related records in the SectionStudent table will need to be deleted using this method. Then the Student can be deleted |  |  |
| **Parameters** |  | 1 |  |
| **Return type** |  | 1 |  |
| **Total** | | 7 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Target Solution** | **Max** | **Grade** |
| **Name** | Delete |  |  |
| **Description** |  | 9 |  |
| **Rationale** |  | 10 |  |
| **Parameters** |  | 2.5 |  |
| **Return type** |  | 1 |  |
| **Total** | | 22.5 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Code Quality** |  | **Max** | **Grade** |
| **Overall efficiency of described solution** |  | 20 |  |
| **Total** | | 20 |  |
| **Grand Total** | | 150 |  |
| 15 |  |