Project 3 – Course Registration System

This project will give you more experience building multi-tier web applications. In this project you will develop a small scale course registration system. This system should be a multi-page web application. You will also gain experience working with databases and designing a good data model for your web application to use. You will demonstrate data modelling skills learned from the pre-requisite database class by creating a good data model and implementing it by creating database tables.

Database Table Requirements:

1. Create a student table named "**Students**" in your database with the necessary fields.

Field	Data Type
StudentID	int
Name	varchar
TuitionOwed	float(50)
… other fields related	

* StudentID should be the Identity (primary key) of the table, and you should let the database handle creating a value for this field for all new records.
* TuitionOwed should reflect the cost of the courses a student is enrolled in. The cost is typically calculated based on the credit hours x cost per credit hours.

2. Create the necessary tables to store a college "**Course**" along with "**Sections**" of a course and their related information in your database. You will need a number of database tables to accomplish this: one table for the course, one table for the sections of a course, another to associate the semesters, and possibly others to deal with the data related to a course. I listed a number of important fields in the list below, but your job is to determine where they belong and what other important data fields need to be stored for these entities. You will need to design a data model and implement it. The information I provided is only a guide of the type of data you will need to store.

Field	Data Type
CRN	varchar
CourseTitle	varchar
DepartmentID	varchar(3)
Semester	varchar
Section	varchar
ProfessorID	varchar
DayCode	varchar
TimeCode	varchar
CreditHours	float(50)
NumberofSeatsAvailable	int
MaximumSeats	int
Prerequisites	???

Hints and suggestions:
* Prerequisites shouldn't be a field because you could a variable number of prereqs.
* Department is the 3 character code representing a department the class is in (CIS, BUS,...).
* Semester indicates the semester the course is offered (Fall 2017, Spring 2017, etc…)
* Section is the specific section in the course offerings.
* DayCode refer to a record or records that would associate this class record with the days it runs (use a good data model).
* TimeCode refer to the record(s) that would associate a class record with the times. Alternatively, you could use start and end times as fields.

3. Create the remaining tables needed to implement this system. For example, you will need to record the student's registration to record the classes they added for their class roster and for the financial billing. You will be graded on how well you designed and implemented a data model for this system. **You will need to create and submit a UML Entity Relational Diagram (ERD) before beginning this project.**

Course Registration System Requirements:

Your project needs to implement the following transactions.

1. Create a new student.
   If the database is handling the creation of StudentID for each record, then you need to retrieve it and display it to the user upon successfully creating the account.

2. Add & Delete Courses.
   Create an ASPX page to allow the user to add a record for a new course and delete an existing one.

3. Modify an existing course's Information.
   *You can use the same ASPX page to implement the **add and delete course transactions** (2) and the **modify existing course transaction** (3), because they both require that a single course be accessed and displayed.*

4. Perform a search for Courses based on a Department for a particular semester.
   A user should be able to enter the initial letters of a department's name and see a complete list of courses that match the department or select a department from the list. This can be done with a drop-down box or list, and it should be part of the registration process. This isn't something that requires its own page.

5. Register for a Course:
   a. A user can enter his/her StudentID, select a semester, select a Department, and possibly select a DayCode, and see a display of all classes that meet the criteria. This display is to be presented in a dynamic display using a GridView, but you can use a Repeater or manually build the content without a control like we learned in class.

b. You need to display the **student's name** and **department name** in a header to the table or at the top of the list of classes. Then, the dynamic display will contain the CRN, Course name, description, number of seats available, maximum seats, credit hours, days, times, professor, and any other relevant information for each course that satisfies the request. Include a Label with a status in the dynamic display. The label will indicate whether there are any seats by saying whether it is "closed" or "open".

c. Provide some mechanism to allow the user to select one or more classes in the list to enroll.

d. Registration considerations:
    i. A student that isn't in the database shouldn't be able to register for a course.
    ii. A student should not be able to register for the same course in a particular semester.
    iii. Check that the student meets the pre-requisites for each class.

e. Update the database to complete the registration:
    i. The **NumberofSeatsAvailable** field in the table related to a section of a course.
    ii. The student's **TuitionOwed**. This is based on the total credit hours for the classes the student has registered. This should update to reflect more classes being added for a student. You can determine the price per credit for undergrad and graduate classes.
    iii. Record the registration in the database.

6. Drop a course from a student's roster.
    a. Update the database to complete the dropping of a class:
        i. The **NumberofSeatsAvailable** field in the table related to a section of a course.
        ii. The student's **TuitionOwed**. This is based on the total credit hours for the classes the student has registered. This should update to reflect the removal of the class.
        iii. Remove the registration for the particular course from the database.

7. Review a student's roster based on semester and bill for all the courses they are registered.
    This may be a good place to allow the student to drop a course.

8. You need to use Stored Procedures for all queries and database operations.

9. Design principles:
    a. This application will be used by an adviser or an employee of the university to complete tasks like adding/modifying/deleting courses. The student will be able to add/drop courses, view their roster and tuition bill. Think of these perspectives when designing the application.
    b. Make the pages of your web application professional looking and presentable. This means they should make use of images, colors, and proper alignments to present the content. You're welcome to use tools like Bootstrap to enhance the presentation of your pages.
    c. Provide a consistent and logical navigation system. The user should never have to use the browser's Back and Forward buttons to move between pages.

d. The user should be presented with an opening screen that presents the various transactions with links to respective pages to perform the selected transaction.
e. Make your presentation clear to the user, providing on-screen instructions wherever needed both for data entry and error correction. If required data is omitted or entries are incorrect, the user should not have to re-enter data that is already correct.
f. Create a good data model and implement the data model by creating the necessary tables in the database. The tables I listed above are just for explanation purposes. You are free to make a completely different set of tables since I expect you to implement your own data model. You will be graded on the implementation of your data model.
g. You need to use a proper naming convention for all controls and in your code and use good programming practices when writing your code (i.e. use C# naming conventions, spacing, indentation, naming, etc…). I expect you to properly name your classes, variables, functions, etc… according to the C# naming convention, not Java!
h. Your programs should not crash for any reason; it's poor design to have a program crash. Make sure to implement exception handling in appropriate places that can cause errors and handle them gracefully.
i. You cannot use the SQLDataSource control or any other similar controls to work with the database. You need to handle the code manually using ADO.NET programming or my DBConnect class.
j. You must use component-based software design. This means writing as much code in classes and functions of classes instead of in the GUI.

10. Use server-side input validation for all transactions where it's necessary.