**Diploma in Software and Design**

**Assignment Cover Sheet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student’s name:** AMARBIR SINGH | | | | |
| **Module 3 Advanced Web Development (25 credits)** | | | | |
| **Assignment title and/or number**:  DSED-06 Asp.net Design pattern | | | | |
| **Assessment weighting** | | | *Need to pass the assessment to complete the course* | |
| **Passing Criteria:** | | | Need to score 50 or more marks to pass the assessment.  **Total Marks : 100** | |
| **Due date**: | | | **Date submitted**:  (late submissions incur 10% penalty, after 7 days late, the assessment will not be marked) | |
| **Assessment conditions:** | | | This is a resource-based assessment. This means that you may have access to any relevant resources to assist you. This could include, for example, your learning materials, information on the Internet, and so on. However, all work must be your own with no assistance from any other person. | |
| **Submission requirements:** | | | You’re required to upload the following on Cloud Campus:   * This document, completed where appropriate * Visual Studio project files * Upload your project on Github and paste the link below * <https://github.com/amar751/CarRentalSystem> | |
| **Learning Outcomes:** | * User experience (Ux) design including user interface (UI), HCI principles, and universal accessibility; * Creating accurate and clear technical and user documentation; * Application of the core interaction design concepts and practice, underpinned in the third outcome of the New Zealand Certificate in Information Technology (Level 5)[Ref: 2595]; * Application of the core information systems skills and knowledge underpinned in the second outcome in the New Zealand Certificate in Information Technology [Ref: 2595]. * Coding – object oriented, procedural; * Facility in multiple common programming languages and integrated development environments (IDEs), which fosters the ability to migrate to new languages, tools and systems; * Construct software with complex, multi-element architectures and abstract data types (ADTs), such as general graphs, trees, tables; * Writing code following design patterns and software development standards * Source and version control; * Optimisation concepts and techniques; * Automated software builds; * Program maintenance techniques; * Application of the core software development concepts and practice, underpinned in the fourth outcome of the New Zealand Certificate in Information Technology (Level 5) [Ref: 2595]. * Testing on a range of platforms e.g. multiple devices and environments; * Debugging, which includes debugging utilities, managing bug reports and issue tracking. * Designing and implementing appropriate application data access, management, and storage technologies to match the application domain; * Digital asset management and storage technologies appropriate to match the application domain e.g. source and version control, artefact repository * Application security principles, including current best practices in IT security e.g. OWASP; * Encryption and privacy; * Protecting data integrity, data validation techniques; data access permissions; * Authentication and authorization. * Tool selection and architecture; * Understanding service orientation and using external services e.g. simple object access protocol (SOAP), representational state transfer (REST); * Software architectural patterns including model view controller (MVC) and model view presenter (MVP). * Information representation design for multiple situations e.g. data visualisation; technical writing - help documents, user instructions, specifications; | | | |
| **Assignment Checklist:** | | |  |  | | --- | --- | | **Requirement** | **Completed** | | Database | [Symbol][Symbol] | | User interface | [Symbol][Symbol] | | Functionality | [Symbol][Symbol] | | Coding | [Symbol][Symbol] | | Testing | [Symbol][Symbol] | | |

**Disclaimer of Plagiarism and Collusion**

I declare that, to the best of my knowledge, this assessment is my own work, and has not been copied from any other student's work or from any other source.

Enter your name here to indicate you agree to the above statement.

AMARBIR SINGH

# DSED-06 Asp.net Design pattern

## Instructions:

* The assessment is an open book exercise – students may consult with others, but finally must present their own work
* BE CREATIVE, this project has a ton of potential and a variety of interpretations.

## Complete the following specifications:

1. The program will have a database back end created in SQL Management Server and connected remotely to your program. Save the DB files in the same folder as your program to make it easy to move around.
2. The database will contain at least 4 tables connected with relationships of sufficient complexity. Fields will include Strings, Integers, and Boolean (bit).
3. The database must be operated on using **Entity Framework**, with **LINQ** and at least one **lambda**.
4. Create a full CRUD (Create, Read, Update, and Delete) front end in Visual Studio.
5. Use Classes to hold the methods and the variables.
6. To the best of your ability separate out the 3 layers of **Presentation** (stuff you see on the forms), **Business** (all the coding in classes) and **Database** (all the database calls) in their own classes. Better yet, into their own Projects.
7. Use any front end you like to display your data.
8. The program must be fit for purpose (i.e.: it must work) although it can be simplified from a ‘real’ program.

## Front End features

* Data entry text boxes or listboxes, and labels
* Buttons or Radio buttons or any other clickable event to manipulate data
* Adequate signage and titles to make it easy to understand.

### Marking Schedule

|  |  |  |
| --- | --- | --- |
| **Tick Achieved** | Form features | |
| **✓** | 1.1 | Use a CRUD frontend to operate on your data with the relevant form feature |
|  | Class Operations | |
| **✓** | 2.1 | Use Classes to hold the methods and the variables |
| **✓** | 2.2 | To the best of your ability separate out the 3 layers of Presentation (stuff you see on the forms), Business (all the coding in classes) and Database (all the database calls) in their own classes. |
| **✓** | 2.3 | For multiple Business classes create a folder to keep them in |
|  | Database Features | |
| **✓** | 3.1 | The program will have a database back end created in SQL Management Server and connected remotely to your program. |
| **✓** | 3.2 | The database will contain at least 3 tables connected with relationships of sufficient complexity. |
| **✓** | 3.3 | The database must be operated on using Entity Framework, with Linq and at least one lambda. |
|  | Observation and Explanation | |
| **✓** | 4.1 | The student has been observed creating the program in the class |
| **✓** | 4.2 | The student has verbally explained and visually shown the program logic to the tutor, verifying a thorough understanding of the code and architecture. |
| **✓** | 4.3 | The student has commented the code in important non repeating sections providing a concise explanation of what each section does using Summary and comments. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Grade %** | **Excellent 100%** | **Adequate 80%** | **Poor 60%** | **Not Met 0%** |
|
| **Program Specifications / Correctness** | | |  |  |
| 50% | No errors, program always works correctly and meets the specification. | Minor details of the program specification are violated, program functions incorrectly for some inputs. | Significant details of the specification are violated, program often exhibits incorrect behaviour. | Program only functions correctly in very limited cases or not at all. |
| **Mark** | 50 | 40 | 30 | 0 |
| **Readability** | | |  |  |
| 20% | No errors, code is clean, understandable, and well-organized. | Minor issues with layout, variable naming, or general organization. | At least one major issue with layout, variable names, or organization. | Major problems with at three or four of the readability subcategories. |
| **Mark** | 20 | 16 | 12 | 0 |
| **Documentation** | | |  |  |
| 20% | No errors, code is well-commented. | One or two places that could benefit from comments are missing them **or** the code is *overly* commented. | Complicated lines or sections of code uncommented or lacking meaningful comments. | No comments present. |
| **Mark** | 20 | 16 | 12 | 0 |
| **Code Efficiency** | | |  |  |
| 5% | No errors, code uses the best approach in every case. | *N/A* | Code uses poorly-chosen approaches in at least one place. | Many things in the code could have been accomplished in an easier, faster, or otherwise better fashion. |
| **Mark** | 5 | 4 | 3 | 0 |
| **Assignment Specifications** | | |  |  |
| 5% | No errors | *N/A* | Minor details of the assignment specification are violated, such as files named incorrectly or extra instructions slightly misunderstood. | Significant details of the specification are violated, such as extra instructions ignored or entirely misunderstood. |
| **Mark** | 5 | 4 | 3 | 0 |