**Diploma in Software and Design**

**Assignment Cover Sheet**

|  |  |
| --- | --- |
| **Course name:**  Diploma in Software and Design | **Student’s name:**  Amarbir SIngh |
| **Module Name /or number: Advanced Web Development** | |
| **Assignment title and/or number** DSD-09 Web API | |
| **Assessment weighting** | *Need to pass the assessment to complete the course* |
| **Passing Criteria:** | Need to score over 50 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:** | This assessment can be added to an existing project.  https://github.com/amar751/BookCollectionAPI |
| **Learning Outcomes:** | * 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; * 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). |

**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.

Amarbir Singh

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

# Create a WebAPI using a Controller and a JS Framework.

This assessment can be done in conjunction with another project. Such as DSD-06, DSD-07, DSD-08. Or it can be standalone with a simple Database backend.

Generate an API Controller in your Visual Studio Web Project.

Create CRUD to a simple Model / Table in a View using JQuery or a similar JS framework, such as React, connected to that API Controller.

**This cannot use the same model fields as the one in the manual. Use your own.**

**Marking Schedule**

**AchieveTick *Presentation features***

1.1 Suitable signage and design

***Class Operations***

2.2 Use MVC

## Database Features

3.1 The program will have a database back end

3.2 The database must be operated on using Entity Framework, with Linq and at least one lambda.

3.3 Create a full CRUD (Create, Read, Update, and Delete) front end

3.4 Uses a different model to the one used in the Class activity.

## 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** | | | Significant details of the specification are violated, program often exhibits incorrect behaviour. | Program only functions correctly in very limited cases or not at all. |
| 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. |
| **Mark** | 50 | 40 | 30 | 0 |
| **Reada** | **bility** |  | At least one major issue with layout, variable names, or organization. | Major problems with at three or four of the readability subcategories. |
| 20% | No errors, code is clean, understandable, and well-organized. | Minor issues with layout, variable naming, or general organization. |
| **Mark** | 20 | 16 | 12 | 0 |
| **Docum** | **entation** |  | Complicated lines or sections of code uncommented or lacking meaningful comments. | No comments present. |
| 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. |
| **Mark** | 20 | 16 | 12 | 0 |
| **Code E** | **fficiency** |  | 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. |
| 5% | No errors, code uses the best approach in every case. | *N/A* |
| **Mark** | 5 | 4 | 3 | 0 |
| **Assign** | **ment Specifications** |  | 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. |
| 5% | No errors *N/A* | |
| **Mark** 5 4 | | | 3 | 0 |