**Vision College Logo**

**Diploma in Software and Design**

**Assignment Cover Sheet**

|  |  |
| --- | --- |
| **Course name:**  Diploma in Software and Design | **Student’s name:**  Jatin Mahajan |
| **Module Name /or number: Object Oriented Programming & Data (25 credits)** | |
| **Assignment title and/or number**:  C#.Net Assessment | |
| **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**:  12thDecember, 2018 | **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   GitHub link below:   https://github.com/jatin9988/video-store |
| **Learning Outcomes:** | User experience (Ux) design including user interface (UI), HCI principles, and universal accessibility;  Coding – object oriented, procedural;  Source and version control;  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]. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.

# <https://github.com/jatin9988/video-store>

# Jatin Mahajan

# DSED-02 SQL Server Assessment Project

## Instructions:

* The due date is end of the First Term break (**we don’t have class time in term 2**)
* The assessment is an open book exercise – students may consult with others, but finally must present their own work

*Write and test an SQL Server database program using a C# application*

1. **Design and write a program in C#** according to the following specification:
   1. **Context:** You have been asked to make a simple **Video / Movie rental system** for a small video shop, before the genre dies off completely because of downloading. The video store needs to keep a record of their videos, as well as a list of their customers and who rents what and when they are returned.

You start with the following tables, **Movies**, **Customer**, and the table that holds all the rentals **Rented Movies**. Note that they all have Primary keys, and are joined to each other by ID Foreign keys.

This may not be all the fields you need, feel free to add and remove fields and tables from your database.

Import the script and run it or add in the existing Database



## Form features

* 1. DataGridView or ListView (or both)
  2. Data entry text boxes or list boxes, and labels
  3. Buttons or Radio buttons or any other clickable event to manipulate data
  4. Adequate signage and titles to make it easy to understand.

I have used a **Tab Control** and put **DataGridViews** on each tab. But you are free to create your own system.

## Form Operations

2.1 **Insert** a new record. **Update** existing records, **Delete** records (CRUD operations)

* 1. Show all videos,
  2. Show just the videos that are out at present.

2.4 Add fees for the videos, if videos are older than 5 years (Release Date) then they cost $2 otherwise they cost $5

* 1. Use a Database class to hold your SQL calls.
  2. Issue, Charge and Return Movies
  3. List **who borrows the most videos** and list **what are the most popular videos**.
  4. Create **Two Unit Tests**, one to test the Connection to the DB and the other of your choice.
  5. Sanitize all your Database Changes to prevent SQL Injection
  6. Use at least one View to return data.
  7. Host the Project on GitHub and send me the link

### Extras

* Keep your database in the same folder as your program files so that you can just drag the program around and the database goes with it. **Attach** the database to the Server Management Studio to use it. Like mine:



### Marking Schedule

|  |  |  |
| --- | --- | --- |
|  | Form features | |
|  | 1.1 | DataGridView or ListView (or both) |
|  | 1.2 | Data entry text boxes or list boxes, and labels |
|  | 1.3 | Buttons or Radio buttons or any other clickable event to manipulate data |
|  | 1.4 | Adequate signage and titles to make it easy to understand. |
|  | Form Operations | |
|  | 2.1 | **Insert** a new record. **Update** existing records, **Delete** records (CRUD operations) |
|  | 2.2 | Show **all** videos |
|  | 2.3 | Show just the videos that are out at present. |
|  | 2.4 | **Add fees** for the videos, if videos are older than 5 years (Release Date) then they cost $2 otherwise they cost $5 |
|  | 2.5 | Use a **Database class** to hold your SQL calls |
|  | 2.6 | **Issue**, **Charge** and **Return** Movies |
|  | 2.7 | List **who borrows the most videos** and List **what are the most popular videos** |
|  | 2.8 | Create **Two Unit Tests**, one to test the Connection to the DB and the other of your choice. |
|  | 2.9 | Sanitize all your Database Changes to **prevent** **SQL** **Injection** |
|  | 2.10 | Use at least one **Procedure** |
|  | 2.11 | Use at least one **View** to return data. |
|  | 2,12 | Host the Project on **GitHub** and send me the link |
|  | Database Operations | |
|  | 3.1 | Tables and relationships Created and filled with data |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **% of 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 behavior. | 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 |