**Vision College Logo**

# Diploma in Software Engineering and Design

# Assignment Cover Sheet

|  |  |
| --- | --- |
| Course name: Diploma in Software Engineering and Design | **Student’s name:**  Rupinder Singh |
| **Module Name -** Database Design and Analysis (*15 credits)* | |
| **Assignment title and/or number**: SQL Server Assessment Project | |
| **Assessment weighting :** | *12.5% of the overall programme.* |
| **Passing Criteria :** | Need to score 50 or more marks to pass the assessment.  **Total Marks : 100** |
| **Due date** **: 12th Dec 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 submit the following into Cloud Campus:   * This document, completed where appropriate * Database files * Visual Studio project files |
| **Learning Outcomes** | * Develop computer programs in a GUI environment * Demonstrate knowledge of data types and structures * Demonstrate knowledge of user interface controls and user interface animation * Design, develop and test a computer program * Incorporate comments into code * Compile, debug and test computer programs |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assignment Checklist:** | |  |  | | --- | --- | | **Requirement** | **Completed** | | Database | ✓🗶 | | User interface | ✓🗶 | | Functionality | ✓🗶 | | Coding | ✓🗶 | | Testing | ✓🗶 | |

# 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.

|  |
| --- |
| Rupinder Singh |

Github

https://github.com/rupinder8699/VideoRentalSystem

# C#.Net and SQL Server Assessment Project

Assessment Overview

In this assessment, you need to plan, develop and evaluate a software application for a Video store. You need to create the UI with C#.NET and have it interact with the database in SQL Server using ADO.NET.

Scenario

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. A customer can only rent two movies at a time.

# Assessment Requirements

## Database Requirements

You can use the sample database design as shown below.



You can modify the above design to add more functionality in your software application. However you need a minimum of three tables, **Movies**, **Customer**, and **Rented Movies.**

Note that they all have Primary keys, and are joined to each other by foreign keys.

You will be provided with a script to import sample data in to your app.

## User Interface requirements

UI design should have the following features

* A simple, easy to use and intuitive UI.
* Use of appropriate controls such as DataGridView, List view, Combo box as necessary.
* Use of Adequate signage and titles to make it easy to understand.

Sample UI layout are shown below



## Functionality Requirements

* Add, Delete, Search and Update Movies
* Add, Delete, Search and Edit Customers
* Issue Movie Functionality
* Return Movie Functionality
* Movies Currently Rented Out
* Stats Page showing who borrows the most videos and the most popular videos
* Add fees for the videos, if videos are older than 1 year (Release Date) then they cost $2 otherwise they cost $5

## Coding requirements

Use a Database class to manage all the database operations

Appropriate naming conventions used and coding standards followed, use the following checklist to assess your code

1. Good class names?
2. Good function/method names?
3. Good variable names?
4. Easily understandable?
5. As simple as possible?
6. Follows principle of least astonishment?
7. Extracts commonality & minimizes duplication?
8. No unnecessary cuteness or cleverness?
9. Proper names for Controls, Forms (txtName, lbluser etc.)

## Marking Schedule

|  |  |
| --- | --- |
| **Parameter** |  |
| User Interface | 20 |
| Functionality | 30 |
| Coding | 40 |
| Testing | 10 |

## Testing Requirements

 Testing assesses the quality of the product, you can perform a black box or a unit test to test the following scenarios for a video store. Fill out the following testing sheet with the results of your test and submit it.

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Test Scenario** | **Result (Pass /Fail)** |
| Adding Movies | * *Add a Movie and check if it adds to the database and shows in the Data-grid* * *The user should not be able to add the same movie twice, the system should give appropriate error message* * *The system should not store a Blank Movie* * *Movie Genre should have proper value* |  |
| Adding Customers | * *Add a customer and check if it adds to the database and shows in the data-grid* * *The system should not store a blank customer name* * *Customer name should not accept special characters or numbers* |  |
| Search Movies / Customer | * *Search by Movie name / Customer name* * *Search should give out all results that matches the search criteria* |  |
| Update Movies / Customer | * *Should update details of the respective movie and customer only* |  |
| Delete Movies | * *Should not be able to delete a movie that is rented out* |  |
| Delete Customer | * *Should not be able to delete a customer that has rented out a movie* |  |
| Issue Movie | * *Issue a movie to a customer and check if it adds a record in the Rented movies table, Date rented should be the current system date* * *System should not issue a movie that is already been rented out* * *Give an error message if there are no copies of the movies* * *Cannot issue more than two movies to the same customer* |  |
| Return Movie | * *Return movie should update the Rented Movies Table* * *Cannot return a movie that has not been rented out* * *Date returned should reflect the current system date in the Rented movies table* |  |

## Marking criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **% of Grade** | **Excellent 100%** | **Adequate 80%** | **Poor 60%** | **Not Met 0%** |
| User Interface  20% | Simple easy to use intuitive UI, no errors spelling mistake and good color schemes used | Minor errors with the UI, minor layout issues | Major UI errors making it hard to understand | Significant UI errors with no logical sense and program crash |
| Functionality  30% | 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 |
| Coding  40% | No errors, code uses the best approach in every case and follows the coding standards | Minor errors or repetition of code, coding and naming standards not followed in some occasions | Code uses poorly-chosen approaches in some places. Naming standards not followed in some places. | Many things in the code could have been accomplished in an easier, faster, or otherwise better fashion. Poor naming and coding standards |
| Testing  10% | Program is well tested to identify and fix bugs and errors. No major bugs or defects in the program. Testing results matches the actual program. | Program is well tested to identify most of the bugs, but some bugs still exist.  Some test cases marked pass which are false | Program has a lot of major bugs and is not tested. Testing sheet incorrect or incompletely filled out. | Program is not at all tested and testing sheet is not filled. |