C# Minor Boss – Console App

MARVEL DB - USE CASE ANALYSIS, UI DESIGN and TEST DOCUMENT

# OVERVIEW

You are a software engineer working for a software development company specialising in information processing solutions. Your client is Marvel Comics and they have requested a solution to help their writers manage the details of the thousands of superhero characters they have created over the years. This is tentatively called “Project Marvel Database” or “Marvel DB”.

Your company has sent in a team of analysts to work with Marvel Comics and they have documented a detailed analysis of the use cases relevant to Marvel DB. A [use case](https://en.wikipedia.org/wiki/Use_case) is the description of how a user uses a system, information which can serve to specify the features that your software solution must have.

## Your tasks (later highlighted in Yellow)

1. TASK 1:   
   Study the use cases carefully and complete the User Interface (UI) design sections. Since you are creating a console app, the UI will be text-based, and your design task involves creating the text that shows exactly how the interactions between the user and the app will take place (i.e. what the screen would look like when the app is running). This is a realistic and invaluable exercise that will help you identify the text strings for the app, as well as the flow of the interactions.
2. TASK 2:  
   Using Visual Studio, you are to develop the app to meet all the use cases. You will also need to learn how to work with CSV files (learning materials will be provided). This is a normal part of professional software development: there will always be something new to learn in every project; remember, it’s not how much you know, it’s how fast you can learn!  
   Finally, you will need to go through the use cases again and test to make sure that your app meets all the use cases. To test effectively, you will need to generate at least 10 Marvel superheroes in the CSV file covering a range of possibilities.

# TASK 1 – USE CASES AND UI DESIGN

|  |  |
| --- | --- |
| **UC Name:** Start and View Choices of Possible Actions |  |
| **Primary Actor:** Operator |
| **Scope:** Marvel DB |
| **Trigger:** Upon starting the system, the choices of possible actions are shown |
| **Flow:**  Operator starts System  System loads data from MarvelDB.csv into memory (into an appropriate data structure)  Systems shows a welcoming text with a brief introduction to the app, what it is for and how to use it.  System shows a list of possible actions in dot-points:   * View Marvel DB * Add Superhero * Edit Superhero * Remove Superhero * Exit System   **Note:** changes to Marvel DB are saved to the local hard drive immediately. |
| **UI DESIGN**  TODO: Complete this section for this use case | |

|  |  |
| --- | --- |
| **UC Name:** View Marvel DB |  |
| **Primary Actor:** Operator |
| **Scope:** Marvel DB |
| **Trigger:** Upon viewing the list of possible actions, the Operator chooses the “View Marvel DB” action |
| **Flow:**  System shows list of choices  Operator chooses “View Marvel DB”  System displays data in a text-based table on the screen.  (Then System displays choices of actions again.)  **Note:** *Displayed*data items must be *sorted* alphabetically based on the superheroes’ names (not real names). |
| **UI DESIGN**  TODO: Complete this section for this use case | |

|  |  |
| --- | --- |
| **UC Name:** Add Superhero |  |
| **Primary Actor:** Operator |
| **Scope:** Marvel DB |
| **Trigger:** Upon viewing the list of possible actions, the Operator chooses the “Add Superhero” action |
| **Flow:**  System shows list of choices  Operator chooses “Add Superhero”  System asks for information about the new superhero:   * Superhero’s name * Real name * Height (in cm) * Weight (in kg) * Race (one of “Human”, “Alien”, “Animal”, “Supernatural”) * Whether the superhero has appeared in a movie yet (no need to be a main character).   System validates the input:   * Superhero’s name must not already exist. * Height and Weight must be in the correct format (numeric, rounded to 2 decimal places for storage). * The choice for Race must be within the available races.   If validation fails: System offers unlimited retries (informing that the Operator could type “**cancel**” to return to the choices of possible actions).  If validation passes:   * System confirms success * Saves the changes to the “MarvelDB.csv” file * and return to displaying the choices of possible actions. |
| **UI DESIGN**  TODO: Complete this section for this use case | |

|  |  |
| --- | --- |
| **UC Name:** Edit Superhero |  |
| **Primary Actor:** Operator |
| **Scope:** Marvel DB |
| **Trigger:** Upon viewing the list of possible actions, the Operator chooses the “Edit Superhero” action |
| **Flow:**  System shows list of choices  Operator chooses “Edit Superhero”  System asks for the full superhero name of the superhero to be edited.  System asks for information about the changes to be made to the superhero (informing that if the Operator types “**same**” for an item, it will be unchanged):   * Superhero’s name * Real name * Height (in cm) * Weight (in kg) * Race (one of “Human”, “Alien”, “Animal”, “Supernatural”) * Whether the superhero has appeared in a movie yet (no need to be a main character).   System validates the input:   * Superhero’s name must not already exist elsewhere in Marvel DB. * Height and Weight must be in the correct format (numeric, rounded to 2 decimal places for storage). * The choice for Race must be within the available races.   If validation fails: System offers unlimited retries (informing that the Operator could type “**cancel**” to *undo all the changes* and return to the choices of actions).  If validation passes:   * System confirms success * Saves the changes to the “MarvelDB.csv” file * and return to displaying the choices of possible actions. |
| **UI DESIGN**  TODO: Complete this section for this use case | |

|  |  |
| --- | --- |
| **UC Name:** Remove Superhero |  |
| **Primary Actor:** Operator |
| **Scope:** Marvel DB |
| **Trigger:** Upon viewing the list of possible actions, the Operator chooses the “Remove Superhero” action |
| **Flow:**  System shows list of choices  Operator chooses “Remove Superhero”  System asks for the full superhero name of the superhero to be removed.  System asks for confirmation that the Operator wants to remove the superhero permanently and offers a way to cancel the operation.  If operation goes ahead: System removes the superhero and update MarvelDB.csv.  If operation is cancelled: System returns to displaying the choices of actions. |
| **UI DESIGN**  TODO: Complete this section for this use case | |

|  |  |
| --- | --- |
| **UC Name:** Exit System |  |
| **Primary Actor:** Operator |
| **Scope:** Marvel DB |
| **Trigger:** Upon viewing the list of possible actions, the Operator chooses the “Exit System” action |
| **Flow:**  System shows list of choices  Operator chooses “Exit System”  System acknowledges that it has stopped. |
| **UI DESIGN**  TODO: Complete this section for this use case | |

## Programming Requirements

To ensure the quality of the program, you must follow these best software engineering practices:

* refactor your code into multiple reusable methods (hint: the use case diagrams are very helpful here – go with the bubbles 😊).
* handle exceptions and errors with meaningful messages and appropriate actions (don’t just crash the program).
* Include proper documentation using the “triple slash” style at the top of every method, as well as block comments and in-line comments where appropriate.

# TASK 2 – RESEARCH, DEVELOPMENT & TEST

## RESEARCH – CSV File IO

CSV (Comma-Separated Values) file is a very popular file format to store data in a tabular form (in a table). Even simple Excel spreadsheets could be saved as CSV files!

Save the file below to your computer and double-click to open it, you will probably see Excel showing it as a table of data



Open the file again in Visual Studio (dragging and dropping) or in a text editor and you will see that the CSV file is simply a text file with these lines:

Name,DoB,Height,Weight

Person 1,1/01/1990,172,83

Person 2,4/12/2000,145,52

Person 3,31/05/2001,167,65

Notice how every row contains the values of each column in the table, separated by commas, hence the name CSV! (Also: the first row is the row of column headers.)

To implement the solution, you will need to write information from some data structure(s) into a CSV file and to read information from a CSV file to populate some data structure(s). Naturally, this is not an easy task but thankfully, it is a very common task and there are many existing C# modules out there to help you.

In the ZIP file containing the C# code for this assessment, you will find some existing code to handle CSV file IO, as well as some examples of how to use them. Just study how you can use the existing code in “ReadWriteCsv.cs”, not how it actually works, that’s the whole idea of “reusing code”!

## Development

Implement the use cases in C# with Visual Studio.

## Test

TODO: Run the program and use the program to generate a CSV file with at least 10 Marvel superheroes. Choose your superheroes to cover a range of possible input values (e.g. all the different names, heights, weights, races and appearance-or-not in a movie etc.).

TODO: Complete the table below to prove that you have tested the use cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Test Result (Passed/Failed) | Test Date | Comments  (why failed, incomplete features, etc.) |
| Start And View Choices of Possible Actions |  |  |  |
| View Marvel DB |  |  |  |
| Add Superhero |  |  |  |
| Edit Superhero |  |  |  |
| Remove Superhero |  |  |  |
| Exit System |  |  |  |

# SUBMISSION

TODO: You are to hand in a single ZIP file containing:

1. The Visual Studio solution folder (and everything within it)
2. This Word document, all filled in
3. The CSV file *with at least 10 Marvel superheroes* (OR make sure it is in the bin\Debug\... folder of the Visual Studio solution folder that is going into the ZIP file)