Project PRG2782

Create a Version-Controlled Superhero Database System using Windows Forms and GitHub.

Your team plays a major role in the special HQ unit for Belgium Campuses' sister campus, **One Kick Heroes Academy**. While the academy focuses on training aspiring heroes through hero courses with exams determining ranks and abilities to handle various threats - your team at HQ steps in to help manage and coordinate all the trainee records. Up until this point, assessors have struggled with paper-based documents during the entry exams, assessing students abilities and feeding the results into the old system. It has been chaotic, seeing that they can't keep up with processing the reports. Build a C# app to track hero details, auto-calculate ranks based on exam scores, and generate reports. It's your unit's way to support the heroes-in-training, making sure they're prepped for everything from pop quizzes to finals week mayhem without any data disasters.

Develop a C# Windows Forms application that manages superhero records using text files and incorporates Git version control. The application should include the following features:

1. **Add New Superhero**: Through a form interface, allow the user to input superhero details (At least a Hero ID, Name, Age, Superpower, and Hero Exam Score – more fields can be added to do in-app exam scores/assessments) and save these details to a file called superheroes.txt. Automatically determine and store the hero's rank and the threat level they can handle based on the exam score.
2. **View All Superheroes**: Display all superhero records from superheroes.txt in a DataGridView, including calculated ranks and threat levels.
3. **Update Superhero Information**: Enable the user to search for a superhero by ID, update their details through a form (including exam score), recalculate rank and threat level, and save changes.
4. **Delete a Superhero**: Allow the user to select a superhero from the DataGridView and delete the corresponding record from the file.
5. **Generate a Summary Report**: Calculate the total number of superheroes, the average age, average exam score, number of heroes per rank, and display the results on the form, saving this summary to a file named summary.txt.
6. **Version Control with Git**:
   - Initialize a Git repository for the project.
   - Stage and commit changes after each major modification (add, update, delete, or report generation).
   - Push the repository to GitHub.

The ranking system is for heroes in the One Kick Heroes Academy. The hero's rank is determined by their exam score as follows:

S-Rank: 81-100
A-Rank: 61-80
B-Rank: 41-60
C-Rank: 0-40

The threat level they can handle is:

S-Rank: Finals Week (threat to the entire academy)

A-Rank: Midterm Madness (threat to a department)

B-Rank: Group Project Gone Wrong (threat to a study group)

C-Rank: Pop Quiz (potential threat to an individual student)

The requirements for the project are as follows:

1. **Create a C Windows Forms Application**:

- The application should be designed with an interface that allows users to interact with the Superhero Database System, including input fields and action buttons.

2. **Implement Core Functionalities**:

- **Add New Superhero**:
  - Use input fields (TextBox controls) for Hero ID, Name, Age, Superpower, and Hero Exam Score (numeric, 0-100).
  - Calculate rank and threat level based on the exam score.
  - Save new superhero details, including calculated rank and threat level, to a file called superheroes.txt.
- **View All Superheroes**:
  - Display the list of superheroes in a DataGridView control on the form, showing Hero ID, Name, Age, Superpower, Exam Score, Rank, and Threat Level.
  - Load superhero data from superheroes.txt and populate the DataGridView.
- **Update Superhero Information**:
  - Allow selection of a superhero record from the DataGridView, and populate the input fields for editing.
  - If exam score is updated, recalculate rank and threat level.
  - Save the updated information, including new rank and threat level if applicable, back to superheroes.txt.
- **Delete a Superhero**:
  - Enable the user to delete a selected superhero from the DataGridView.
  - Remove the corresponding record from superheroes.txt.
- **Generate a Summary Report**:
  - Calculate and display the total number of superheroes, average age, average exam score, and number of heroes per rank (S, A, B, C) on the form.
  - Save these details in a file called summary.txt.

3**. Implement Version Control Using Git**:

- **Initialize a Git Repository**:
  - In the project directory, initialize a Git repository.
- **Commit Changes**:
  - After implementing each major feature (Add, Update, Delete, Generate Report), stage and

commit the changes.

- Ensure clear, meaningful commit messages that describe each change.

4. **Integrate with GitHub**:

- **Create a GitHub Repository**:
  - Set up a new repository on GitHub and push your local commits to this remote repository.
- **Push Changes**:
  - After completing each task, push all commits to the GitHub repository.

5. **Error Handling and Documentation**:

- Implement error handling for file I/O operations and input validation (including for exam score) to ensure a smooth user experience.
- Include inline comments explaining key parts of the code for better readability and maintainability.

Presentation Requirements

Group Setup

- **Group Size**: Groups of **3-4 students**. Students may choose their own groups; however, they should be from the same class/module type (on campus or virtual).
- **Time Limit**: Each group will have **10-15 minutes** to present their project, with an additional **5 minutes** for questions from the assessors. Going over your time could result in mark penalties.
- **Presentations Timeslots**: Groups will be allocated timeslots per their lecturer's discretion. There is no guarantee that you will present to your own lecturer, but rather based on the availability of the panel of lecturers running the module.
- **Participation**: All students are expected to present; otherwise, they will receive **0** for the project.
- **Integrity**: Any sign of AI use, plagiarism, or paraphrasing tool use can result in a **0** for the entire group.

Presentation Structure

1. **Introduction (2-3 minutes):**
   - Briefly introduce the project and its purpose.
   - Outline the technologies used (C, Windows Forms, Git, GitHub).

2. **Project Demonstration (5-7 minutes):**
   - Show the interface and demonstrate the core functionalities:
     - Adding a superhero, including entering exam score and showing automatic rank/threat calculation.
     - Viewing all superheroes, with ranks and threat levels displayed.
     - Updating a superhero record, including changing exam score and recalculating rank/threat.
     - Deleting a superhero.
     - Generating a summary report, including ranking statistics.
   - Highlight how they used Git for version control, explaining key commits and how changes were

tracked.

- Show the GitHub repository and explain how they collaborated as a team (if applicable).

3. **Question and Answer Session (5 minutes):**

- The assessors will ask questions related to:
  - The code and implementation choices, including how the ranking system was implemented.
  - The challenges they encountered and how they solved them.
  - How they used Git and GitHub for collaboration and version control.
- Groups should be prepared to explain their code, especially around file handling, error handling, and the ranking logic.

Assessment

- The presentation is assessed based on:
  - **Understanding of the Project**: How well the students understand their own code and the functionality they implemented.
  - **Demonstration of Features**: Clear demonstration of each feature listed in the requirements, including the ranking system.
  - **Use of Git and GitHub**: Evidence of effective use of Git version control, including frequent commits and pushing the project to GitHub.
  - **Responsiveness to Questions**: How well the group responds to questions, demonstrating their knowledge and involvement in the project

BELGIUM CAMPUS iTversity
It's the way we're wired

Project 2025
Programming 2782

Rubric

Criteria	Description	Marks
1. Adding New Superhero	The form correctly accepts input for Hero ID, Name, Age, Superpower, and Exam Score.	5
	Data validation for fields (e.g., age and score are numbers, no empty fields, score 0-100).	3
	Successfully appends the new superhero record, including calculated rank and threat level, to superheroes.txt.	5
	Clear error messages are shown if input is invalid (e.g., empty fields, non-numeric values).	2
Total		15
2. Viewing All Superheroes	Data is successfully read from superheroes.txt.	5
	The DataGridView is correctly populated with the data in a readable, formatted manner, including rank and threat level.	5
	The view is refreshed correctly when the file is updated (e.g., after adding or deleting a superhero).	5
Total		15
3. Updating Superhero Info	The selected superhero record is loaded into the form fields for editing.	5
	Updated information is validated, rank and threat level recalculated if score changes, and saved back to superheroes.txt.	5
	The DataGridView displays updated information without errors or data duplication.	5
Total		15
4. Deleting a Superhero	Allows selection of a superhero from DataGridView and deletion with confirmation prompt.	5
	Removes the corresponding record from superheroes.txt.	5
	The DataGridView is refreshed after deletion, reflecting the changes.	5
Total		15
5. Generating Summary	Calculates total number of superheroes, average age, average exam score, and heroes per rank accurately.	5
	Displays the results on the form in designated text boxes/labels.	3
	Saves the summary to summary.txt with correct formatting.	2
Total		10
6. Version Control with Git	Initializes a Git repository and commits the initial project files.	3
	Properly commits changes with meaningful commit messages for each major feature (add, update, delete, report).	6
	Maintains a clear commit history, with no redundant or unnecessary commits.	3
	Shows evidence of incremental commits during the development process.	3
Total		15

7. GitHub Integration	Creates a GitHub repository and pushes the project files to it.	5
	Links the local Git repository with the GitHub remote and pushes all commits.	5
	Ensures the GitHub repository is public or shared with the lecturer.	5
Total		15
8. Error Handling and Documentation	Provides error handling for file operations (e.g., missing files, permission errors).	4
	Provides input validation and informative error messages for user actions, including exam score.	3
	Includes inline comments in the code, explaining the functionality of each section and purpose of major functions.	3
Total		10
Grand Total		**100**