Student Project Brief: Game Character Manager

A CRUD Application Using Python and SQLite3

Project Context

You've been hired as a junior developer for a game studio working on a fantasy RPG. Your task is to build a command-line tool that helps game designers manage a list of playable characters. This project will help you understand how to use **SQLite3**, a lightweight database engine, with **Python** to perform **CRUD operations**: *Create, Read, Update, and Delete*.

Design a **text-based menu system** that allows users to interact with the database through a simple shell interface.

Learning Objectives

- ❒ Learn how to connect Python to an SQLite3 database.
- ❒ Understand how to create and manage database tables.
- ❒ Implement CRUD operations using SQL commands in Python.
- ❒ Build a user-friendly shell interface for interacting with data.
- ❒ Practice writing modular, readable, and maintainable code.

Project Requirements

You must create a Python program that:

1. **Creates a database** (if it doesn't already exist) with a table called characters.
2. Allows the user to:
    - ❒ **Add a new character** (name, class, level, health).
    - ❒ **View all characters** in a readable format.
    - ❒ **Update a character's stats** using their ID.
    - ❒ **Delete a character** using their ID.
    - ❒ **Exit** the program safely.

Database Table Structure

Column Name	Data Type	Description
id	INTEGER	Primary Key (auto-incremented)
name	TEXT	Character's name
class	TEXT	Character class (e.g., Warrior, Mage, Rogue)
level	INTEGER	Character level

User Interface

The program should display a menu like this:

```
Welcome to the Game Character Manager

1. Add Character
2. View All Characters
3. Update Character
4. Delete Character
5. Exit

Enter your choice:
```

Each option should guide the user through the necessary steps, including input prompts and confirmation messages.

Success Criteria

- ❒ The program runs without errors.
- ❒ The database is created and updated correctly.
- ❒ All CRUD operations work as expected.
- ❒ The user interface is clear and easy to use.
- ❒ Code is well-commented and logically structured.

Extension Challenges (Optional)

- ❒ Add input validation (e.g., level must be between 1 and 100).
- ❒ Add a “status” field (e.g., Alive, Injured, Defeated).
- ❒ Allow searching for characters by class or level.
- ❒ BONUS: Add a feature to simulate a battle between two characters.