### **Project Proposal: Rob of the Shire - A Text-based RPG Simulation**

**1. Project Title:** *Rob of the Shire – Character Interaction and Combat Simulation Game*

**2. Problem Statement / Project Idea:**

The aim of this project is to build a simple **text-based RPG** (Role-Playing Game) where the player controls a character named *Rob of the Shire* and engages in various activities like combat, inventory management, and item usage. The game will simulate a dynamic environment where the player can interact with their character’s inventory, fight enemies, use items, and explore the game world.

This project will allow the player to experience turn-based combat with an enemy character, where the player must make decisions based on the items in their inventory and the health status of both the character and the enemy.

**3. Objectives and Goals:**

The primary goals of this project are to:

* Create a character with **basic attributes** such as name, health, and inventory.
* Develop an **inventory system** where items can be added, used, or removed.
* Implement a **combat system** where the character can use weapons to fight enemies.
* Introduce **item types** (weapons, potions, quests) and allow filtering and usage of these items.
* Build a **save/load feature** for game state persistence (e.g., saving character health and inventory).
* Use an **external data file** (JSON) to store item information such as name, type, damage, defense, and healing properties.
* Implement **filters** to allow users to search through their inventory by item type.

**4. Project Scope:**

* The game will be **text-based**, so there is no graphical interface. User interactions will happen through the terminal/console.
* The project will be developed using **Python 3** and will involve at least two primary Python files:  
  + main.py (the main file responsible for the game loop and user interaction).
  + character.py (defining the character's attributes and inventory).
* The project will use an external **JSON file** (items.json) to load item data like weapons and potions, and it will include functionalities such as:  
  + Adding and removing items from the character’s inventory.
  + Using items (healing potions, weapons for combat).
  + Saving and loading the game state using JSON file operations.
* The combat system will simulate a simple **attack** function where the player uses a weapon to fight an enemy. Each weapon will have a specified damage value, and the enemy will have health that decreases upon taking damage.
* The project will also include a **filtering system** for the inventory, allowing the player to view items by type (e.g., weapons, potions).

**5. Deliverables:**

The final project will consist of the following:

* **Python Code**: The complete game code written in Python, with clear documentation and comments.  
  + main.py – Main entry point for user interactions.
  + character.py – Defines the character’s attributes and functionality.
  + getfilter.py – Helper functions for item filtering.
  + itemloader.py – Loads item data from items.json.
  + enemy.py – Defines the enemy class for combat.
  + save\_system.py – Handles saving and loading the game state.
* **External Data File**: items.json to store item data for weapons, potions, and quest items.
* **Game Report**: A final report of 3-5 pages that includes:  
  + An introduction to the game and the project’s goals.
  + Background information on text-based RPGs and inventory management systems.
  + A description of the design and methodology used to implement the game.
  + Results of the testing and any challenges faced during development.
  + A discussion and conclusion on how the game can be improved or expanded.

**6. Methodology / Design:**

The project will be structured into different modules:

1. **Character Class (character.py)**:  
   * This class will hold the character's basic information such as **name**, **health**, and **inventory**.
   * It will provide methods to **add/remove items**, **use items**, **attack enemies**, and display **inventory**.
2. **Item System (items.json)**:  
   * The items will be stored in a JSON file and categorized into types (e.g., **weapons**, **potions**, **quests**).
   * The player can interact with items based on type and purpose.
3. **Combat System (enemy.py)**:  
   * A simple combat system where the player can attack an **enemy** using a weapon from their inventory.
   * Each weapon will have specific damage, and the enemy will have health that decreases after each attack.
4. **Filter System (getfilter.py)**:  
   * Allows players to filter through their inventory by item type.
5. **Save/Load System (save\_system.py)**:  
   * Allows the player to save the game state (character, inventory, health) and load it in future sessions.

**7. Resources:**

* **Python 3.x** installed on local machines.
* **Text editor/IDE**: Visual Studio Code, PyCharm, or any other Python-supported editor.
* **JSON data file** for storing item data.
* **Terminal/Console** for user input and game interaction.

**8. Expected Challenges:**

* **Combat Mechanics**: Implementing a simple combat system may require balancing health values, damage, and attack logic.
* **Inventory Management**: Ensuring that item addition/removal works seamlessly and that the correct item is used at the right time.
* **Save/Load Functionality**: Ensuring the character's state is properly saved and loaded without data loss.

**9. Conclusion:**

This project provides an opportunity to explore **game design principles** such as **inventory management**, **combat systems**, and **data persistence** using Python. It combines object-oriented programming with practical applications in game development. The final product will offer a fun, interactive simulation for users to experience text-based combat and character management.