**Final Report**

**1. Analysis and Design**

**a. Main Functionalities of the Software:**

The software aims to replicate an engaging RPG (Role-Playing Game) environment, offering players an immersive experience with a myriad of features:

* **Character Interactions:** Players navigate through an intricate maze of rooms, each presenting unique challenges and rewards. Interactions with characters and monsters shape the gameplay, offering dynamic experiences.
* **Battle System:** A sophisticated turn-based battle mechanism allows players to engage in strategic combat. Players can select from various attacks and defenses, influencing the outcome based on their decisions and character attributes.
* **Music and Settings:** A customizable user interface lets players tailor their gaming experience. Background music adds depth to the gameplay, creating ambiance and enhancing immersion.

**b. Design Diagrams:**

*Use Case Diagram:*  
Visualizes the player's interactions with the game components:

**Use case diagram**

A diagram of a person walking through a dungeon

Description automatically generated

*Sequence Diagram:*  
Details the flow of actions and interactions during gameplay:

1. Player selects an action from the UI.
2. Game processes the selected action.
3. Game updates the player's status and triggers corresponding events.

**c. Adopted Object-Oriented Design Pattern:**

**Singleton Pattern:**  
The Singleton pattern was employed to manage game constants and configurations efficiently.

**Justification:**  
Centralizing game constants in a Singleton **Constants** class ensures consistency across components, simplifying maintenance and reducing redundancy. It also facilitates easier updates and modifications, enhancing scalability.

**2. Implementation Notes**

**a. Problem and Solution Table:**

| **Problem** | **Solution** |
| --- | --- |
| Database Connection Error | Implemented connection pooling to manage database connections efficiently. |
| UI Lagging | Utilized asynchronous tasks for UI rendering to improve responsiveness. |
| Inconsistent Battle Outcomes | Refactored battle logic in **Battle.cpp** to ensure consistent outcomes based on character attributes. |

**Essential Information:**

* Ensure correct integration of all code components to avoid runtime errors.
* Incorporate necessary libraries such as Windows Forms for GUI development.
* Define game constants and configurations in **Constants.h** to maintain centralized control and easy updates.

**3. Testing**

**a. Test Cases:**

1. **Room Visitation Test:**  
   *Objective:* Validate that the room visitation status changes upon player's entry.  
   *Procedure:* Initialize a room, mark it as unvisited, enter with a player, and validate its visitation status change.  
   *Outcome:* The room status should transition from unvisited to visited.
2. **Music Toggle Test:**  
   *Objective:* Confirm that background music starts/stops based on the checkbox state.  
   *Procedure:* Launch **SettingsUI**, check the default music state, toggle the checkbox, and monitor the music status.  
   *Outcome:* Background music should commence when the checkbox is checked and cease when unchecked.
3. **Character Health Test:**  
   *Objective:* Ensure that player health updates correctly during battles.  
   *Procedure:* Simulate a battle scenario, monitor player health changes following monster attacks.  
   *Outcome:* Player health should decrement proportionally to the monster's attack strength.

**4. Evaluation**

**a. Design Evaluation:**

* **Challenges Faced:**
  + **Modularity:** Ensuring each component is modular and can be extended without affecting others.
  + **User Interactions:** Crafting intuitive UI elements and optimizing user engagement.
* **Rectifications:**
  + Adopted interfaces and abstract classes to enhance modularity and extensibility.
  + Incorporated iterative user feedback to refine UI elements and interactions.
* **Future Improvements:**
  + Dynamic dungeon generation to offer diverse and unpredictable gameplay experiences.
  + Player inventory system for managing collected items and enhancing strategic gameplay.

**b. Implementation Evaluation:**

* **Requirement Compliance:**
  + Successfully implemented the core functionalities, including room navigation, battles, and settings customization, aligning with the project objectives.
* **Additional Functionality:**
  + Explored and designed potential enhancements like multiplayer modes, additional character classes, and advanced AI for future iterations.

**Conclusion**The developed software offers a solid foundation for an engaging RPG game environment, blending character interactions, strategic battles, and customizable settings seamlessly. Through systematic design, robust implementation, and rigorous testing, the software promises a captivating gaming experience.

The report delves deep into the software's architecture, elucidating its design principles, challenges encountered during development, comprehensive testing procedures, and evaluations. With a focus on user experience, game mechanics, and scalability, the software aims to captivate a diverse audience of gamers.

Continuous iteration, user feedback, and technological advancements will be pivotal in enhancing gameplay and player satisfaction. This report serves as a transparent and comprehensive overview of the software's development journey, highlighting its strengths, areas of improvement, and future potential.

By adhering to industry best practices, leveraging object-oriented design principles, and embracing a user-centric approach, the software ensures longevity, adaptability, and quality. With the potential to evolve and innovate, the software stands poised to captivate and entertain gamers across the globe.

Through this detailed report, we aim to offer valuable insights into the software's development process, achievements, challenges, and future prospects, fostering a community of passionate gamers and developers alike.