Architectural description of the system

- **Detailed System Description**

The system is designed as a role-playing game (RPG) where players navigate through a virtual world, interact with various elements, and engage in combat and quests. The game is structured to provide a rich and interactive experience.

- **Requirements and Usage Scenarios**

1. Player Management:

- Attributes: Players have health points (HP), stamina, and other attributes that affect gameplay.

- Actions: Players can perform actions such as moving, attacking, using items, and interacting with the environment.

- Inventory: Players can manage their inventory by adding, removing, and using items. They can also equip and unequip weapons and shields.

2. Combat System:

- Engagement: Players can initiate combat with enemies present in the game world.

- Mechanics: Combat involves calculating damage based on player and enemy attributes, weapons, and shields.

- Outcomes: Combat results in changes to player and enemy health, potentially leading to victory or defeat.

3. Quest System:

- Objectives: Quests have specific objectives that players must complete to progress.

- Progression: Players can start, progress, and complete quests, with progress tracked and rewards given upon completion.

4. Environment Interaction:

- Locations: Players can move between different locations, each with its own set of interactive elements.

- Interactive Elements: Players can interact with things like doors, chests, and beds, which may reveal items or trigger events.

- **Technical Requirements for the Game**

1.Platform Compatibility

- Operating System: The game must run on Ubuntu 20.04 LTS or later.

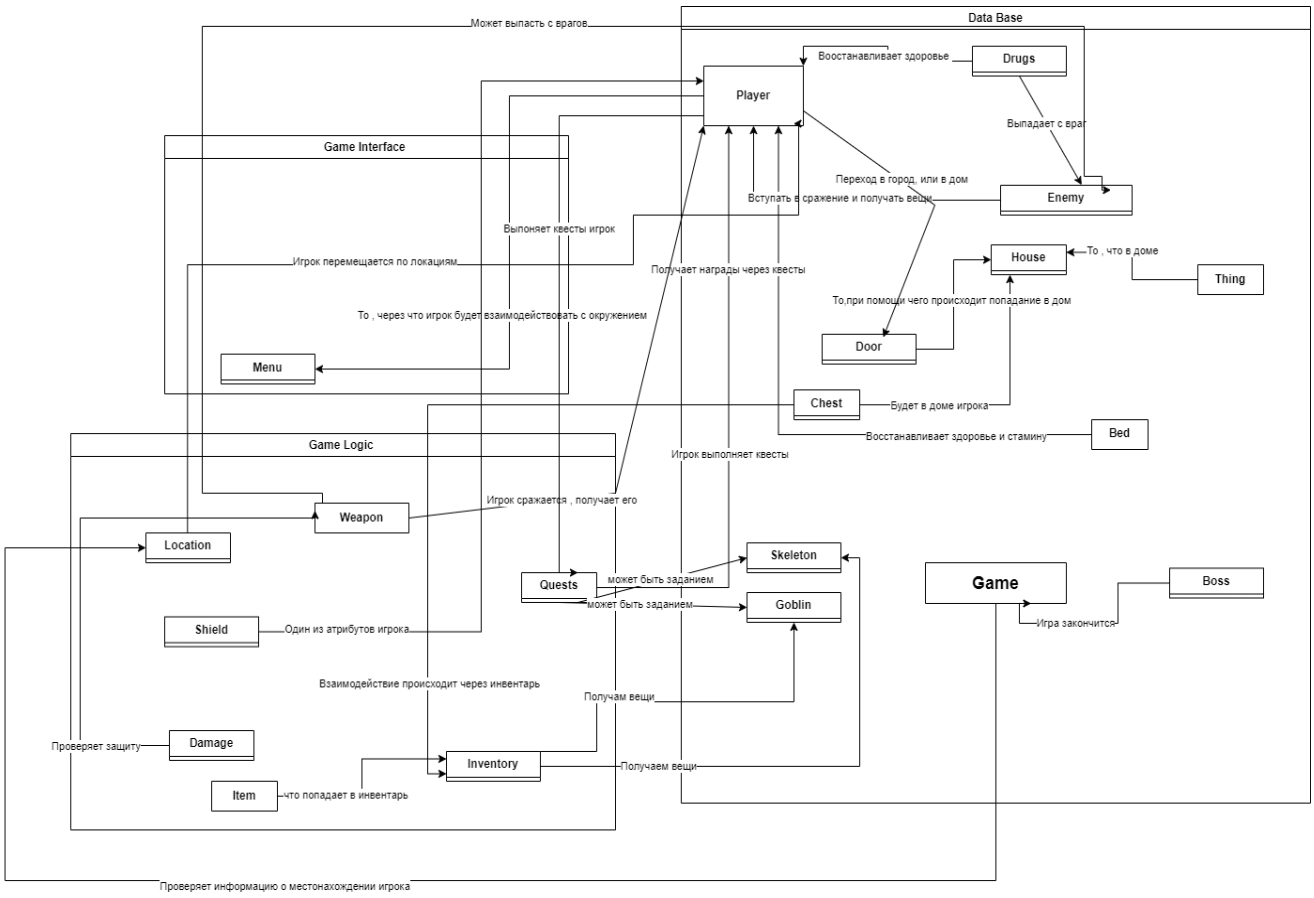
2. System Requirements

- C++ (google unit testing framework, version C++11 and later).

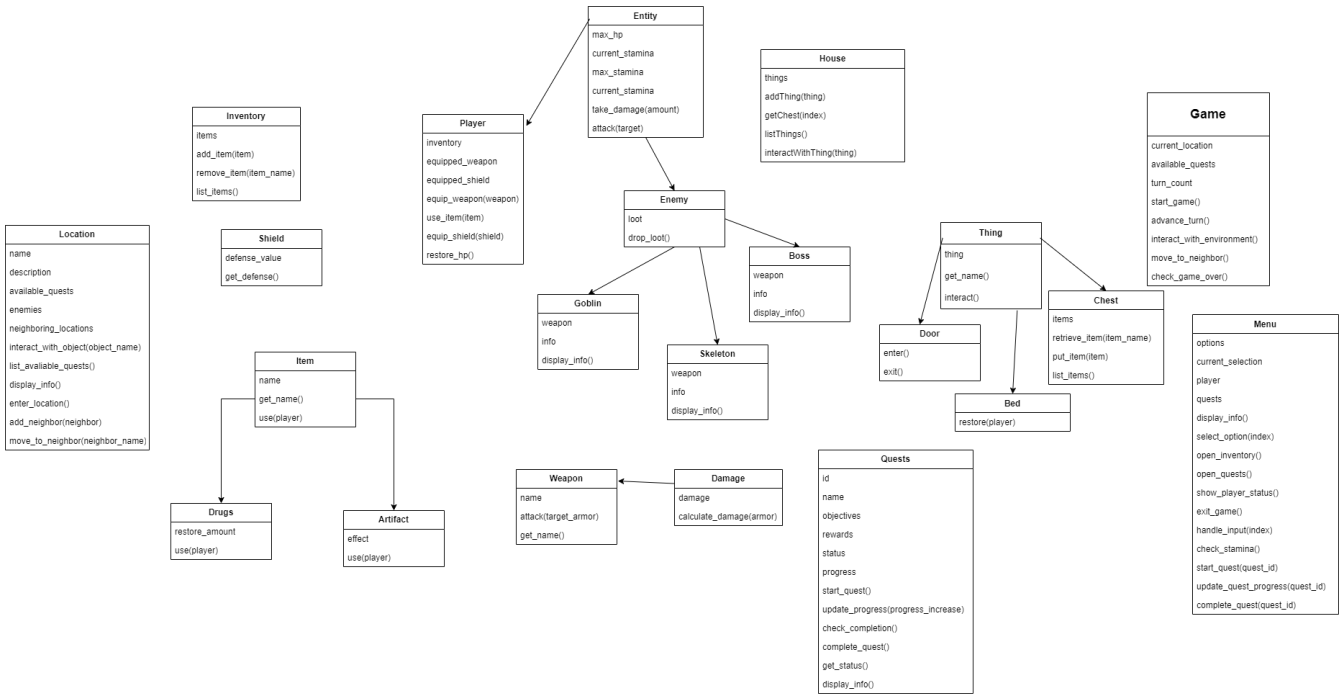
This list covers the technical requirements for the game, including its compatibility with Ubuntu, system requirements, and necessary software dependencies.

- Component Diagram Description

The component diagram provides a high-level view of the system architecture, highlighting the main components and their interactions.



- Class Diagram Description



The class diagram offers a detailed look at the system's classes, their attributes, methods, and relationships.

- **Detailed Test Plan**

1. Unit Testing:

- Classes and Methods: Test each class and its methods individually to ensure they function correctly.

- Coverage: Include tests for inventory management, combat calculations, quest progression, and environment interactions.

2. Integration Testing:

- Component Interactions: Test how different components interact with each other.

- Scenarios: Ensure that player actions correctly update the game state and are reflected in the interface.