**Analysis Document**

1. Introduction
2. Proposed System
   1. Overview
   2. Functional Requirements
   3. Non-Functional Requirements
   4. System Models
      1. Use case model
         1. Use cases description
         2. Use case diagrams
      2. Object Model
         1. Class Diagram
         2. Class Diagram description
      3. Dynamic Model
         1. Sequence Diagrams
      4. User interface
         1. Screen mockups (Extra Credit)
         2. Navigation

* 3. Glossary

**1. Introduction**

Zombie Hospital™ is a Java implemented text based adventure game that will be composed of three levels. The game will be executed on a desktop, with the ability to save and load user profiles. Users will be able to navigate rooms in each level, interact with monsters, solve puzzles, and pick up and use items. The user will also be able to save their progress at the end of each level. The System will save and retrieve information from a text file. The text file will hold information on which rooms have been visited, which puzzles have been solved, which level the user is currently on, and which monsters have been encountered. The administrator will have access to game system editing, the user will be interacting with the game system.

**2. Proposed System**

**2.1 Overview**

Zombie Hospital is a zombie adventure game where the player attempts to rid a hospital of a zombie infestation. Players will traverse three levels each containing ten rooms. The player will be able to equip items such as weapons and armor. Players will solve seven puzzles, two during the first two levels and three during the last level. Each level will contain two monsters with a seventh appearing at the end of the game as a final boss. A new weapon or piece of armor will be found after each monster and the player will start with a basic weapon equipped. A level ends when the player has solved each puzzle and defeated each zombie. When a level ends the player is teleported to the beginning of the next level. At the start of each play session users will be prompted by the game to either load a saved state or begin a new adventure. At the end of each level the user will have the option to save their progress and end their session. Players will be able to attack or flee from monsters in a menu. Players will be able to enter into an equipment menu that will show their equipped items and offer them the option to swap out and drop items. Players can only carry two weapons and two pieces of armor at a time.

**2.2 Functional Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **ID** | **Description** | **Dependency** |
| Functional Requirement 1 | FEA1001 | The Saving and Mapping System must have read access to the player’s current health status. | -none- |
| Functional Requirement 2 | FEA1002 | The Saving and Mapping System must be able to display health of player. | -none- |
| Functional Requirement 3 | FEA1003 | The Saving and Mapping System must have read/write access to the artifacts in the player’s possession. | -none- |
| Functional Requirement 4 | FEA1004 | The Saving and Mapping System must have read/write access to the Player’s location or room and level ID’s. | -none- |
| Functional Requirement 5 | FEA1005 | The Saving and Mapping System must have read/write access to the artifacts in a room. | -none- |
| Functional Requirement 6 | FEA1006 | The Saving and Mapping System must have write access to the player’s current health status. | FEA1001 |
| Functional Requirement 7 | FEA1007 | The Saving and Mapping System must be able to receive input from the user. | -none- |
| Functional Requirement 8 | FEA1008 | The Saving and Mapping System must have read/write access to the player’s profile. | FEA1007 |
| Functional Requirement 9 | FEA1009 | The Saving and Mapping System must have access to known user-input commands. | FEA1007, FEA1010 |
| Functional Requirement 10 | FEA1010 | The Saving and Mapping System must have access to all player usernames. | -none- |
| Functional Requirement 11 | FEA2001 | System checks the currently location of the character. | -none- |
| Functional Requirement 12 | FEA2002 | System checks to see if there is a zombie placed within the current room. | -none- |
| Functional Requirement 13 | FEA2003 | System begins the combat sequence under the conditions that the player is in a room with a zombie. | FEA2001, FEA2002 |
| Functional Requirement 14 | FEA2004 | The system takes in a string from the console and checks to see if it matches the word Attack, ignoring case. If this is done than an attack is initiated. | FEA2003 |
| Functional Requirement 15 | FEA2005 | The system checks and stores the strength attribute of the player object. | -none- |
| Functional Requirement 16 | FEA2006 | System checks and stores the health attribute of the Zombie. | -none- |
| Functional Requirement 17 | FEA2007 | System subtracts the current zombie health by the user’s strength then sets the current zombie health attribute equal to the difference. | FEA2005, FEA2006 |
| Functional Requirement 18 | FEA2008 | The system generates a random number. | -none- |
| Functional Requirement 19 | FEA2009 | System checks a randomly generated number to see if the number is greater 4. | FEA2008 |
| Functional Requirement 20 | FEA2010 | System checks and stores the zombies held item. | -none- |
| Functional Requirement 21 | FEA2011 | The system places the zombies item in the current room. | FEA2001, FEA2010 |
| Functional Requirement 22 | FEA2012 | The system removes the zombie from the current room. | FEA2001 |
| Functional Requirement 23 | FEA2013 | If the zombie’s health is at or below 0 the zombie is removed from the room and its held item is placed in the room. | FEA2010, FEA2012 |
| Functional Requirement 24 | FEA2014 | The system checks the user’s current health and stores the information. | -none- |
| Functional Requirement 25 | FEA2015 | If the user’s health drops to 0 the user is sent to the Start menu. | FEA2014 |
| Functional Requirement 26 | FEA2016 | The system determines which attack a zombie performs and then determines the damage associated. | -none- |
| Functional Requirement 27 | FEA2017 | The system subtracts the zombies attack damage from the user's health and sets the difference to the user's new health. | FEA2014, FEA2016 |
| Functional Requirement 28 | FEA3001 | Artifact System must have read access to the player’s current inventory and equipment. | -none- |
| Functional Requirement 29 | FEA3002 | Artifact System must have record of different classifications of artifacts and the attributes of them. | -none- |
| Functional Requirement 30 | FEA3003 | Artifact System must have access to specific artifact’s attributes. | -none- |
| Functional Requirement 31 | FEA3004 | Artifact System must be able to trigger the effects of items. | -none- |
| Functional Requirement 32 | FEA3005 | Artifact System must be able to add and remove artifacts from rooms and player inventory. | FEA3001 |
| Functional Requirement 33 | FEA3006 | Artifact System must be able to equip and unequipped items from the player. | FEA3001 |
| Functional Requirement 34 | FEA3007 | Artifact System must have read/write access to all artifacts in database. | -none- |
| Functional Requirement 35 | FEA3008 | Artifact System must be able to include and exclude specific artifacts from in-game, with the exception of certain key items such as boss items or puzzle-related props. | FEA3007 |
| Functional Requirement 36 | FEA4001 | Dialogue System must have the ability to communicate and pass information between all Subsystems in the System. | -none- |
| Functional Requirement 37 | FEA4002 | Dialogue system will be able to read input from the console | -none- |
| Functional Requirement 38 | FEA4003 | Dialogue system will be able to output data to the console. | -none- |

**2.3 Non-Functional Requirements**

Design Constraints

* Game should be developed using Java.

Reliability

* Load times should be kept to under 2 seconds.
* Inputs should receive near instantaneous responses.

Security

* Save files should not be stored as a plain text document.

Maintainability

* System should take advantage of object oriented design.

Logical Database Requirements

* No Database is to be used.

Other Requirements

* Have an appropriate level of white space and readability.

**2.4 System Models**

**2.4.1 Use Case Model**

**2.4.1.1 Use Case Description**

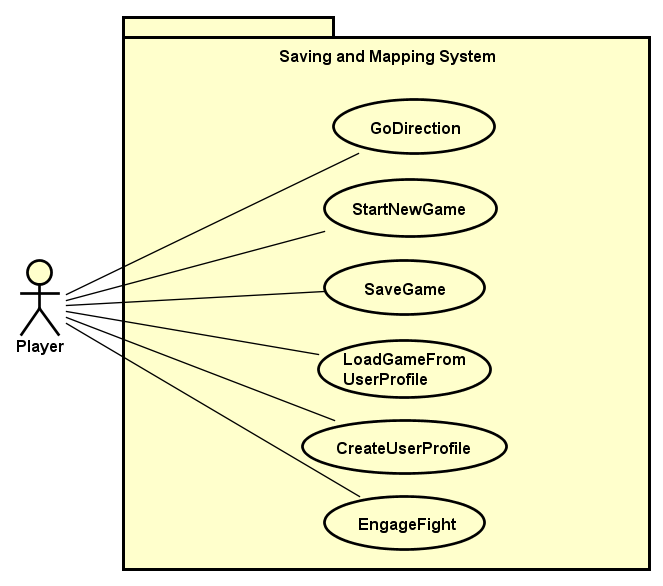
Saving and Mapping System (Janna Timmer)- The Saving and Mapping System will interact with the administrator and the player to determine functions related to saving, loading, user profiles, and location. If the player is interacting with the system they will be able to start a new game, load a game from a user profile, save game, and create a user profile.

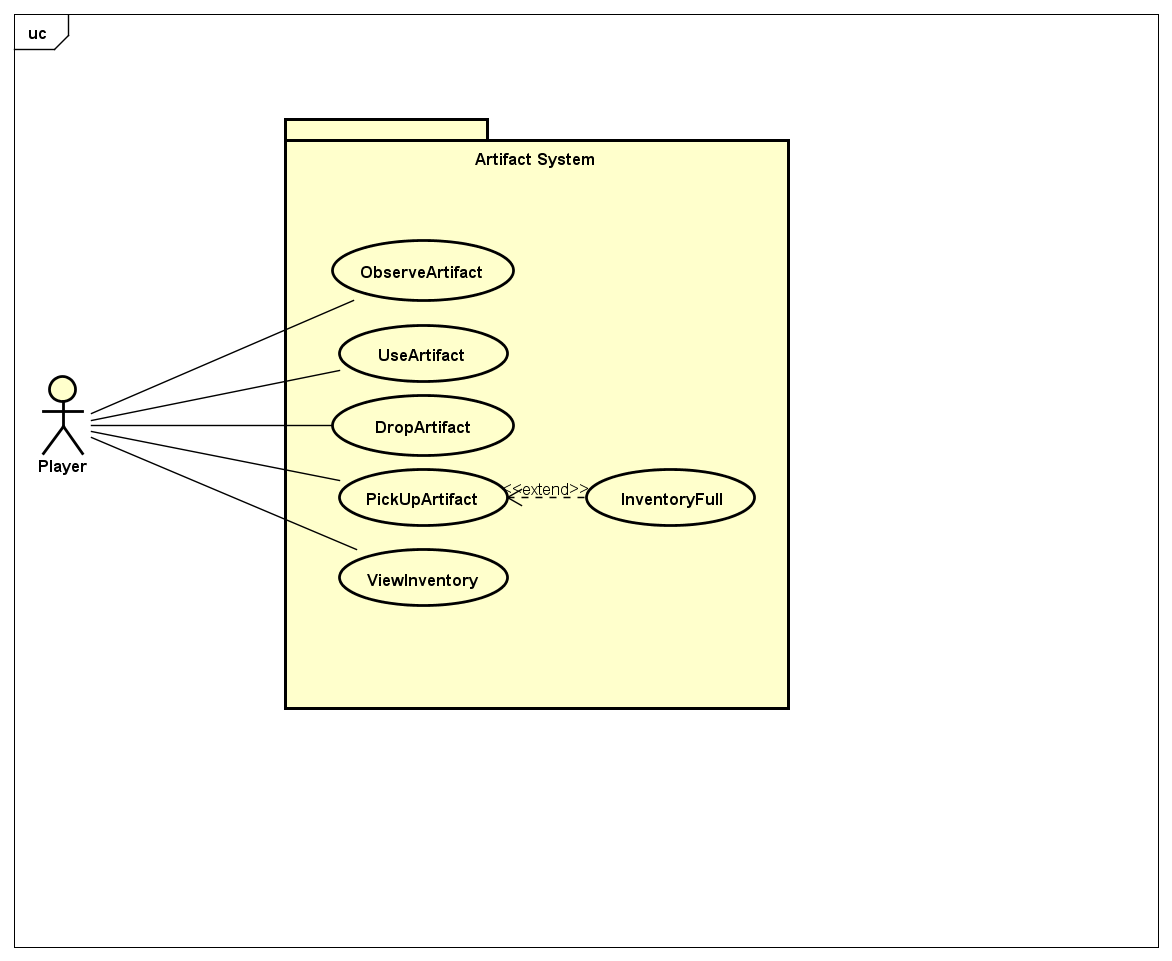
Artifact System (Matthew Nelson)- the Artifact System allows the player to interact with, equip, use and observe items encountered during the game.

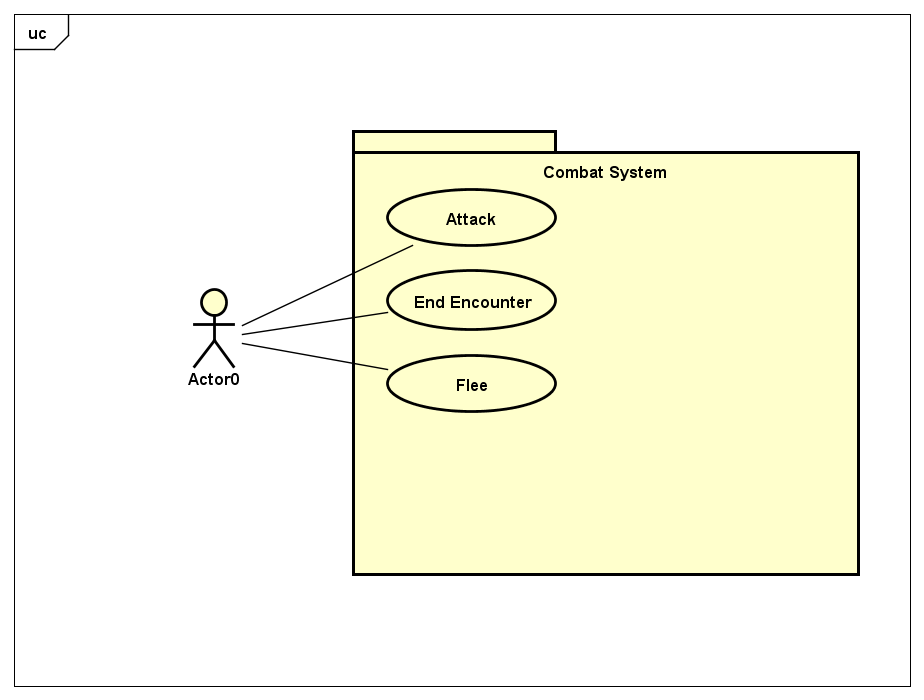
Combat System (Matthew Xiong) - the Combat System keep track of all relevant data for players to engage in combat with zombies. Players will be able to attack and flee from zombies. A player can either win or lose an encounter with a zombie. Engagement happens when a player enters a room with a zombie.

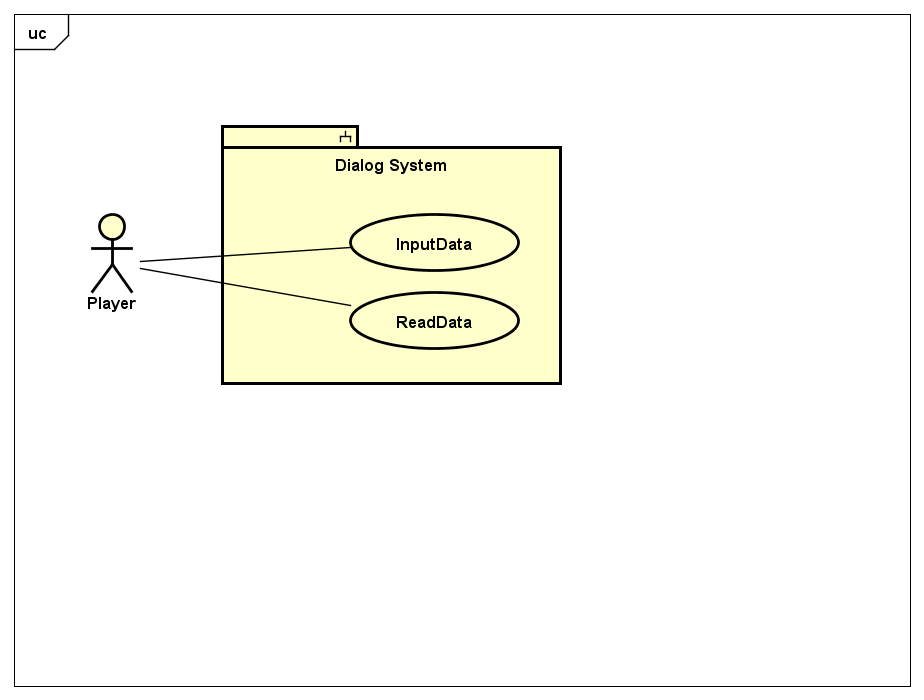
Dialog System (Matthew Xiong) - the Dialog System takes user input and displays pertinent information to the user.

**2.4.1.2 Use Case Diagrams**



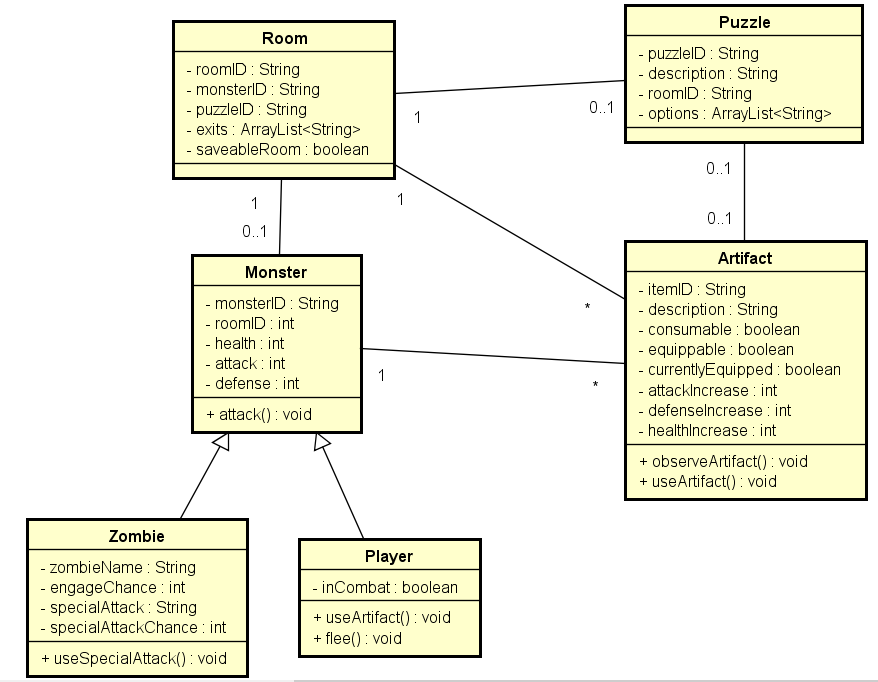






**2.4.2 Object Model**

**2.4.2.1 Class Diagram**



**2.4.2.2 Class Diagram Description**

Room-Has a unique ID, and holds all the monsters and puzzles. Each room has exits, and players may find artifacts on the ground.

Puzzle-Is placed in different rooms to halt the advancement of the player. The player must solve the puzzle to move on.

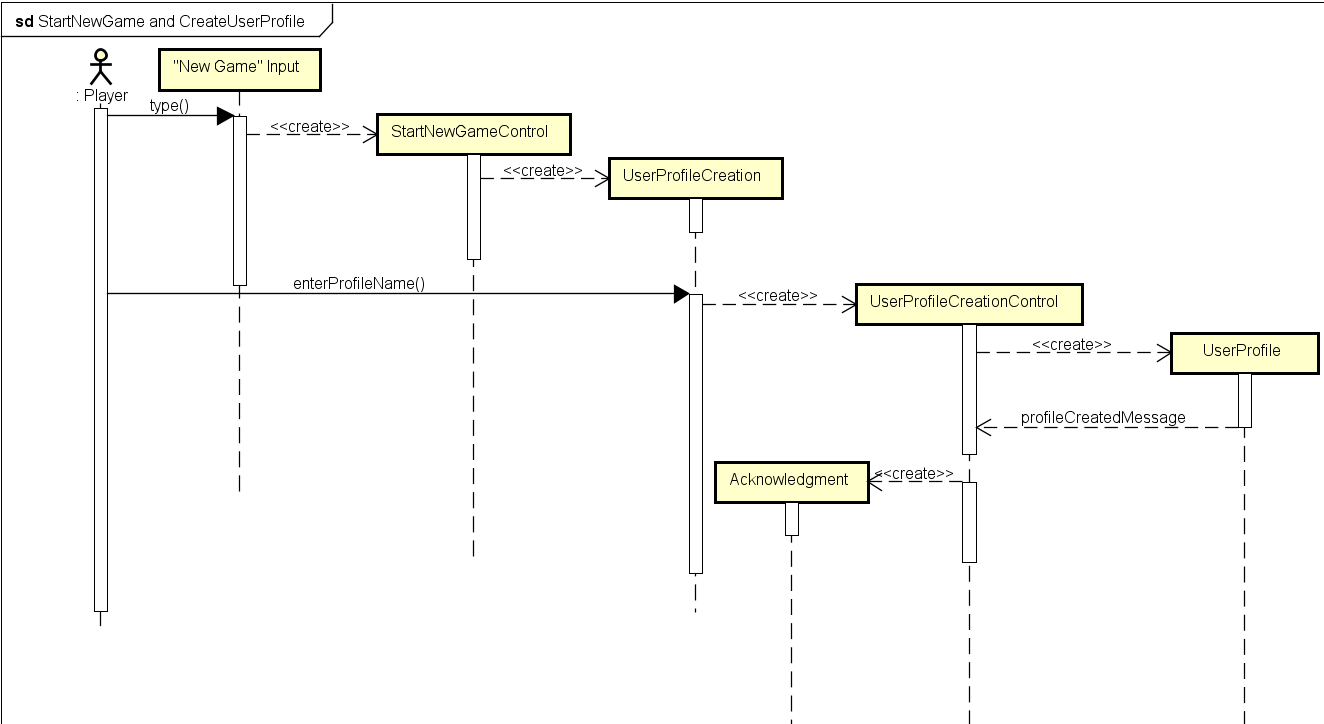
Monster-There are 8 monster throughout the game. The player must defeat 7 monster and fight the last boss to complete the game.

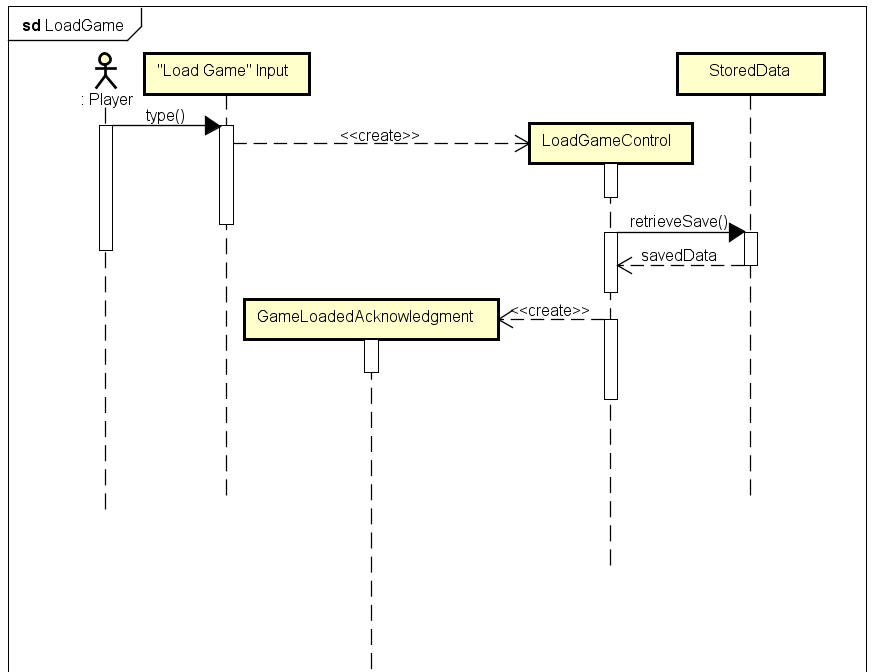
Zombie-The type of monster that is presented in this game

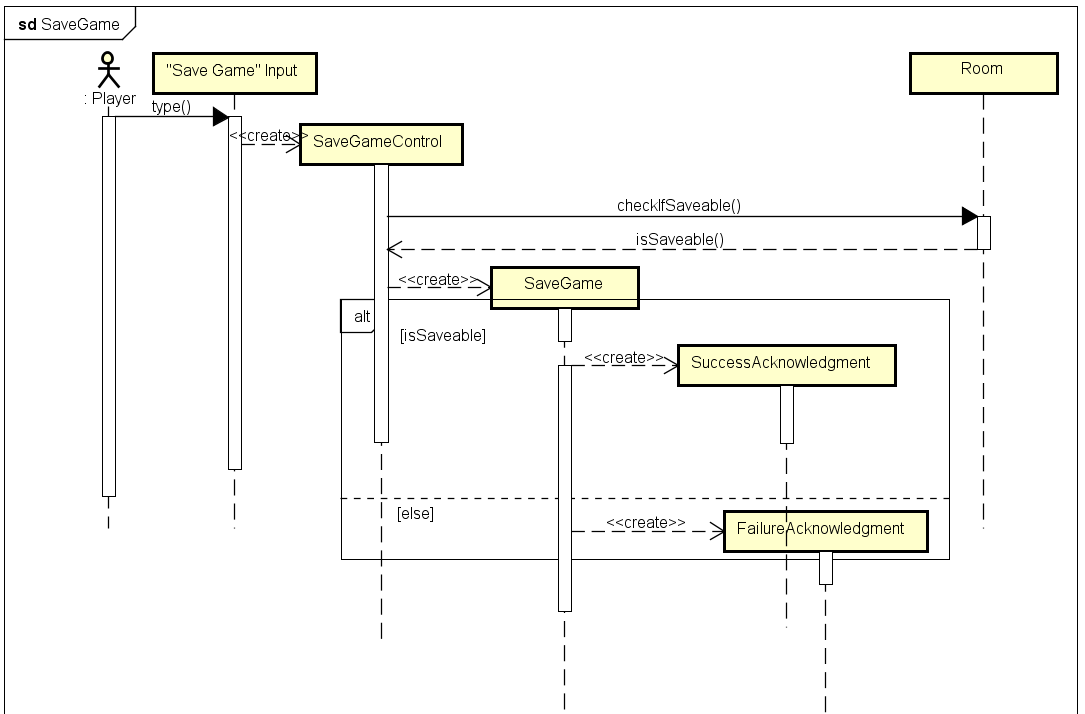
Player-The main hero/heroine of the game. The player must fight through monsters and solve cryptic puzzle and escape the nightmare.

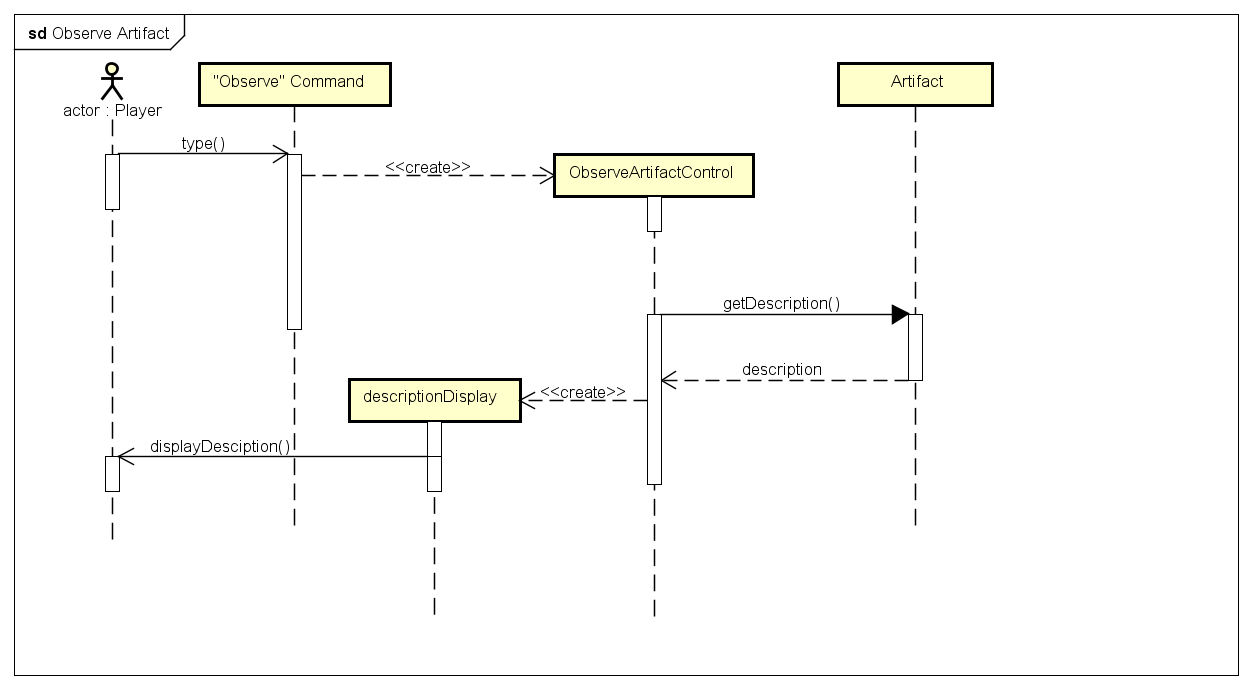
Artifact-Player will be able to find artifacts scattered across the levels. They will need to use some artifacts to help solve puzzles, while some will help the player survive the night.

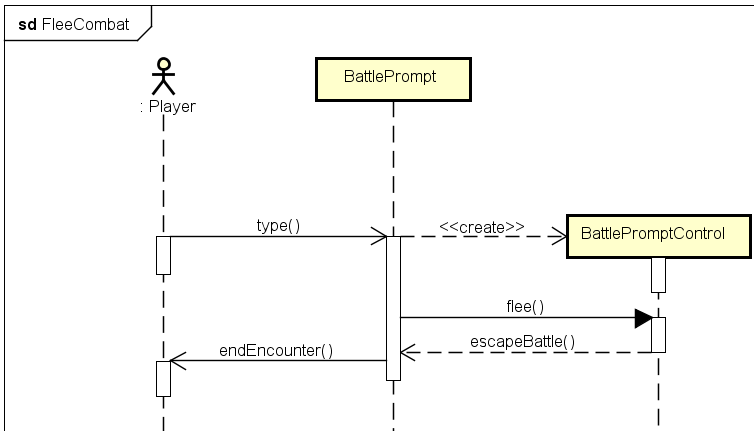
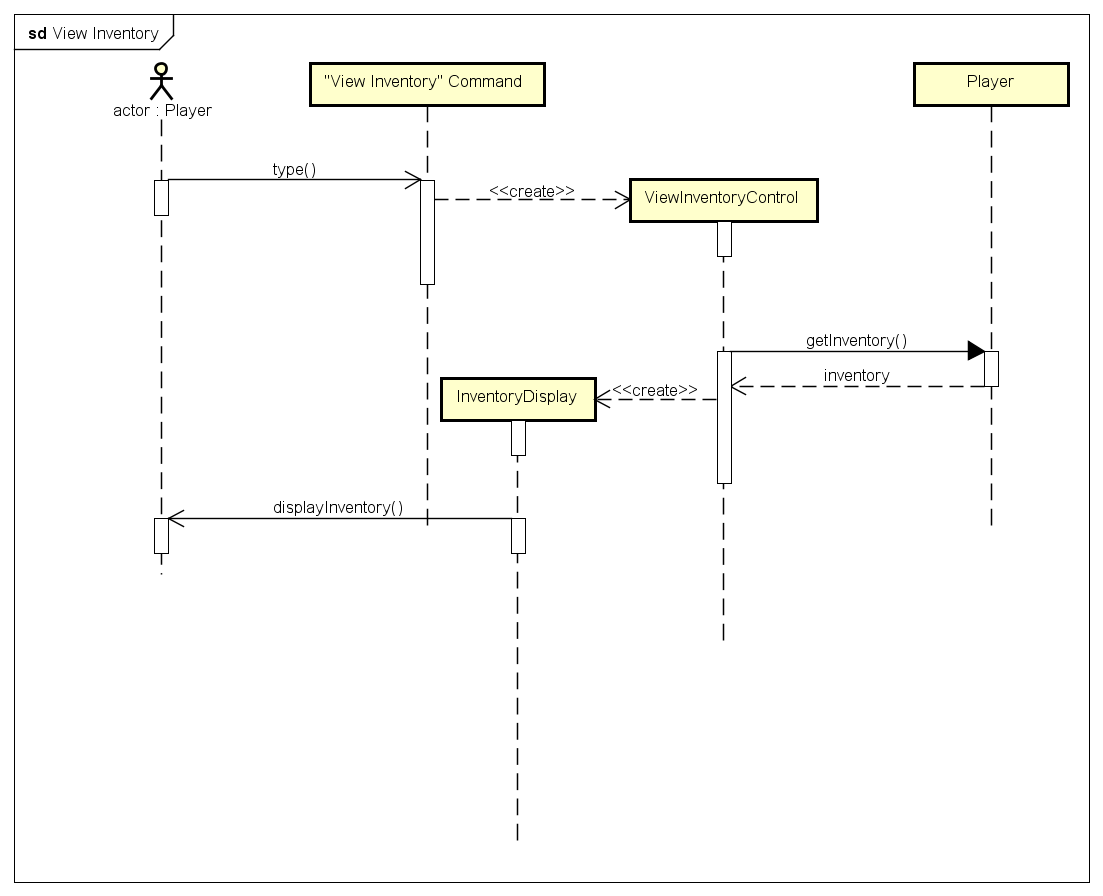
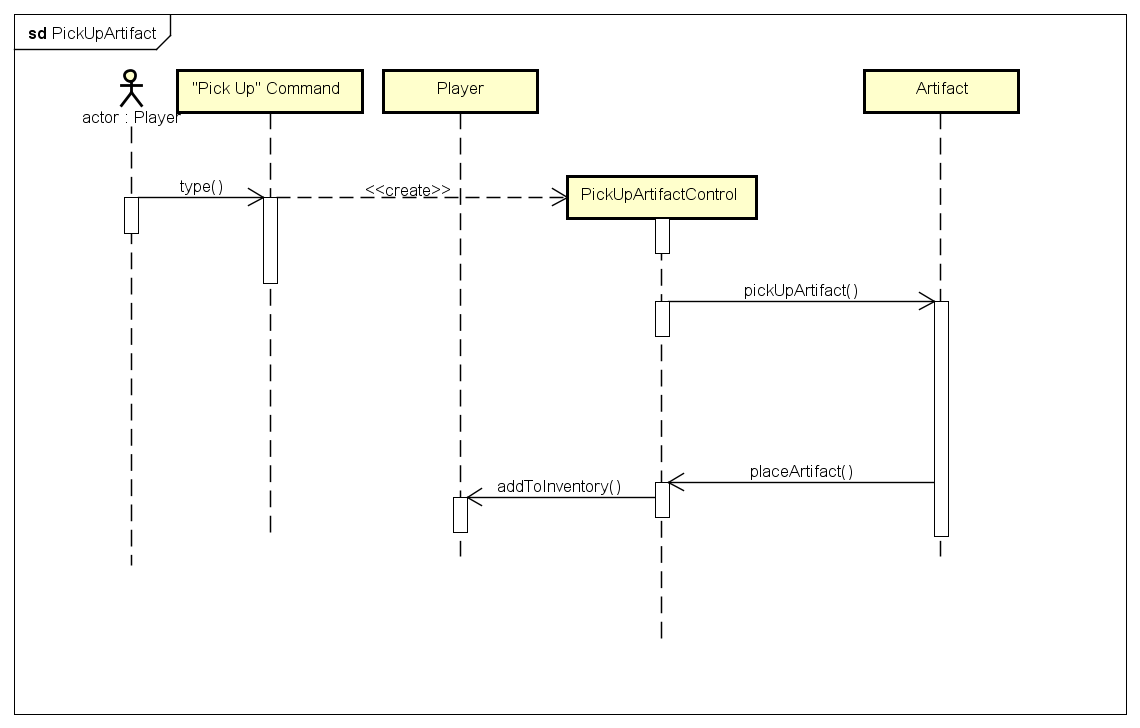
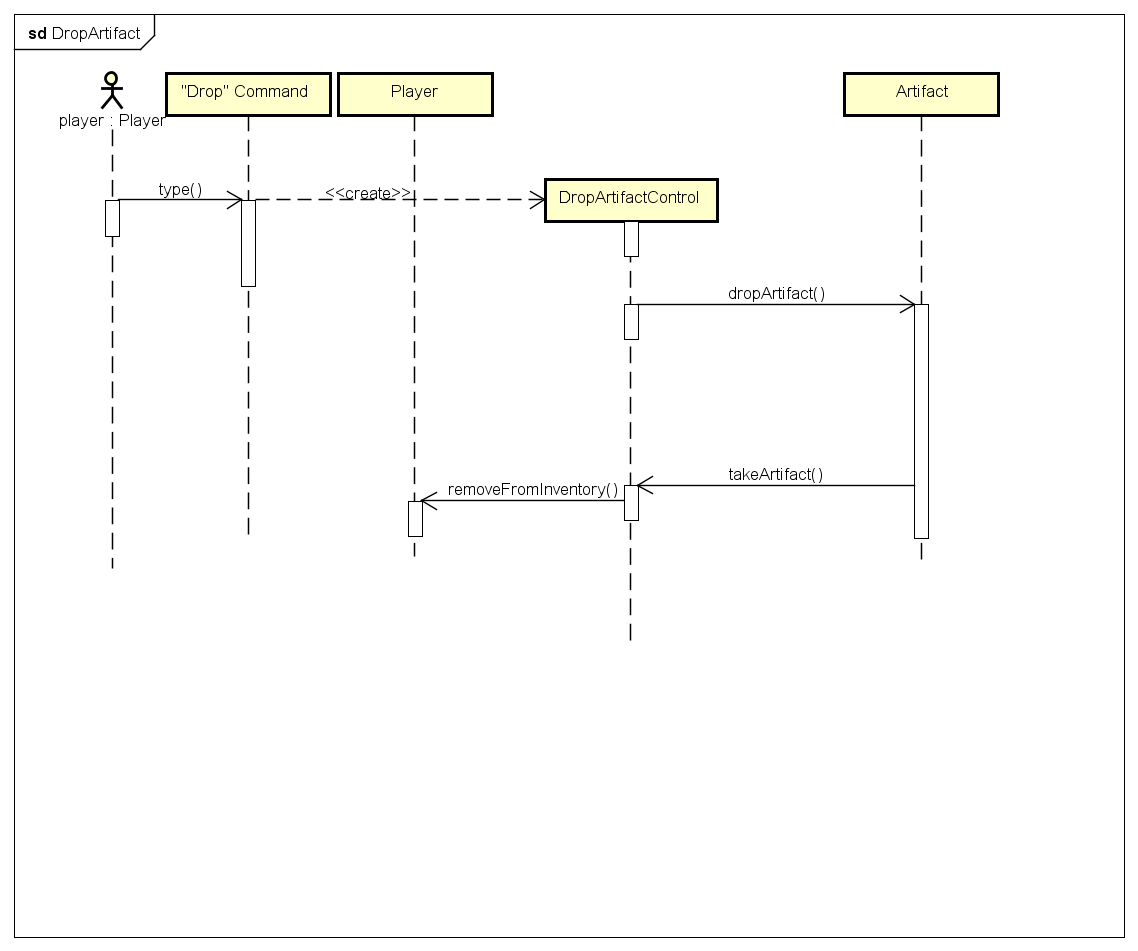
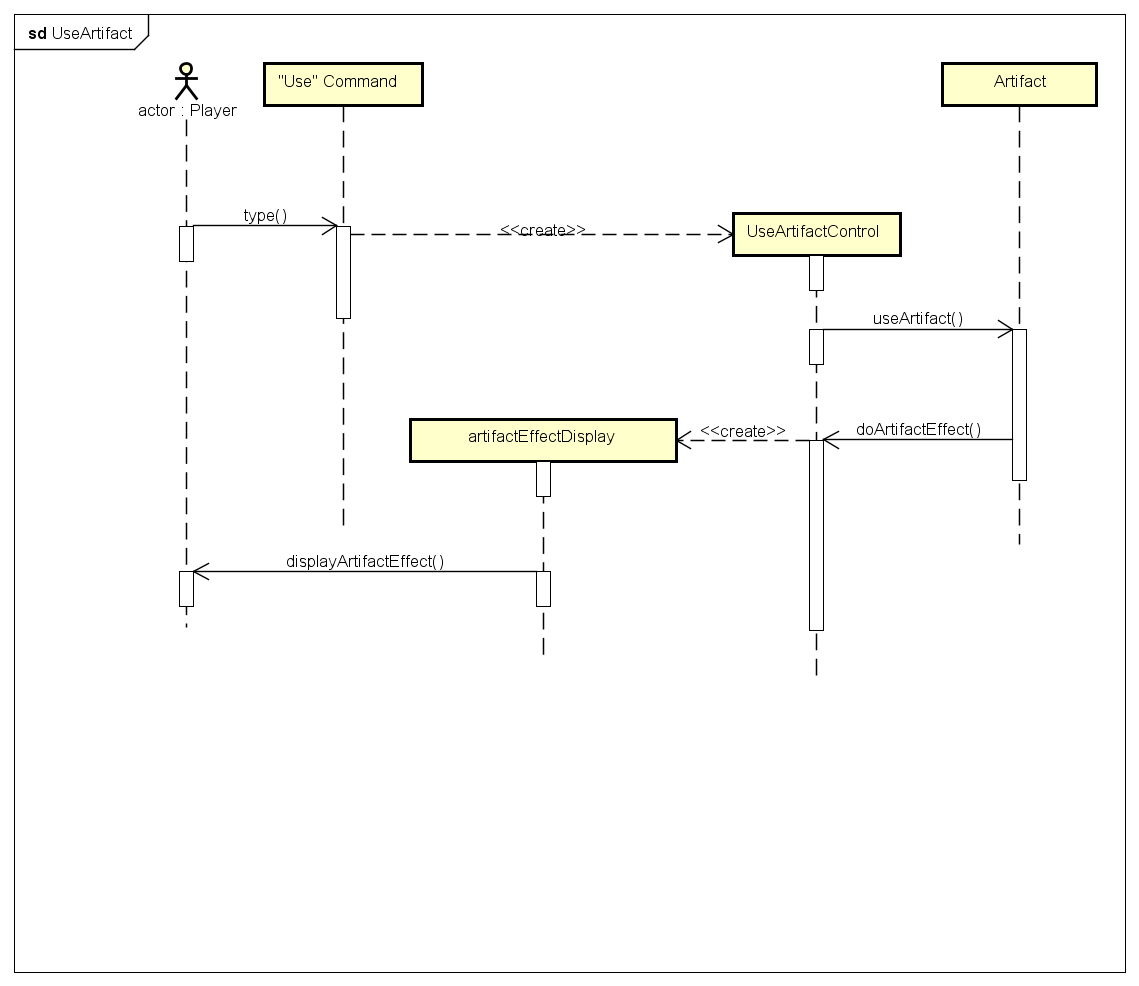
**2.4.3 Dynamic Model**

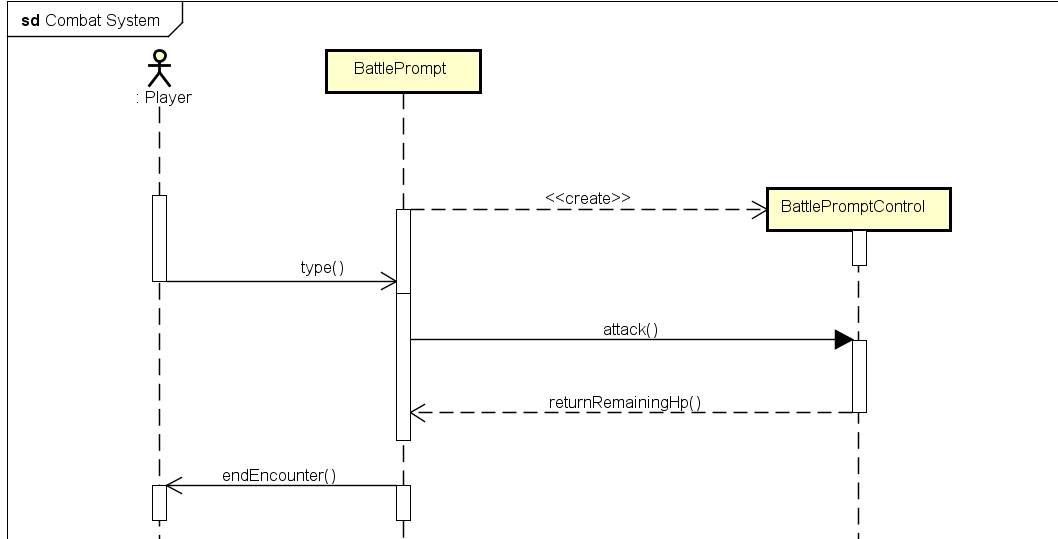
**2.4.3.1 Sequence Diagrams**

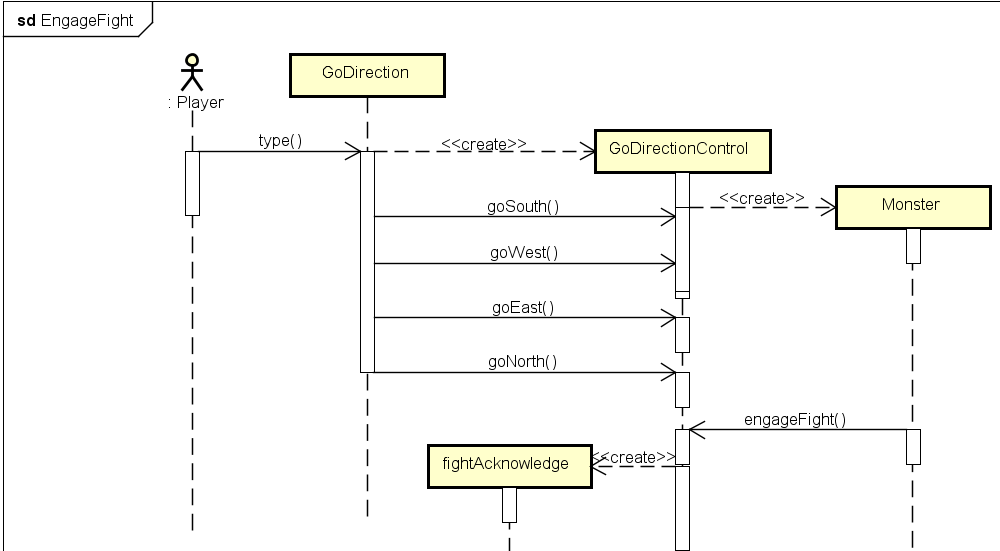


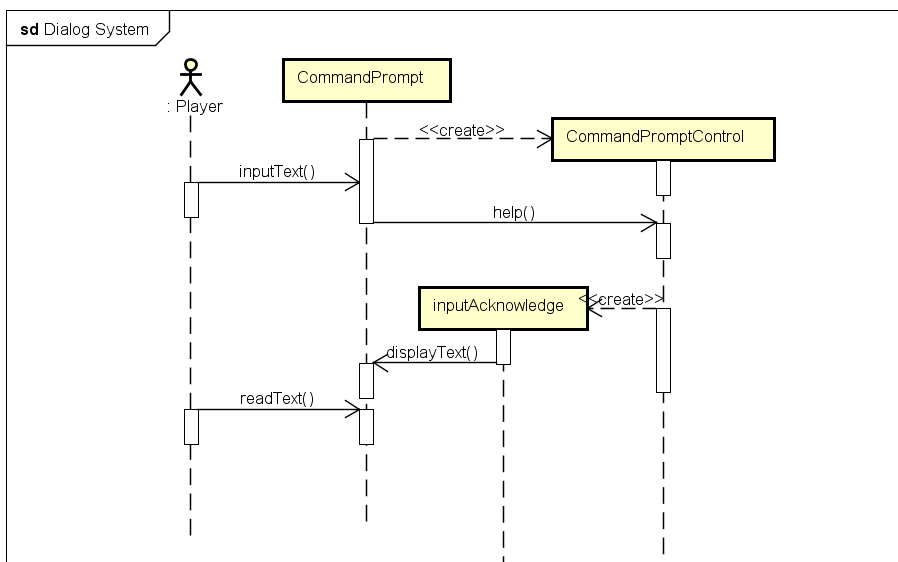












**2.4.4 User Interface**

New Game-Starts a game with no previous save data.

Load Game-Continues the progress of a saved profile.

Go (Direction)-Player is able to type in Go North, South, West, or East

Attack-During a battle the player will strike the target

Flee-Runs from a battle

Use (Artifact)-Player will use said artifact and it will either help the player or hurt the monster.

Observe (Artifact)-Player will be given description of said artifact

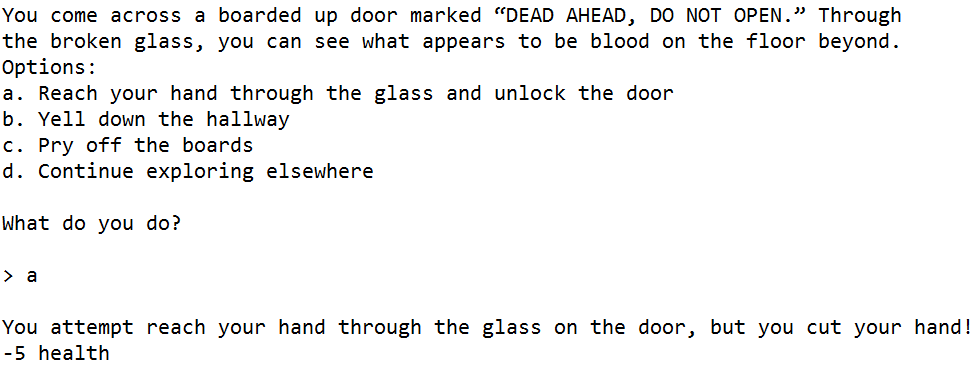
Help-Redisplay all commands given to the player.

Puzzles will also explicitly ask user to interact uniquely with the interface.

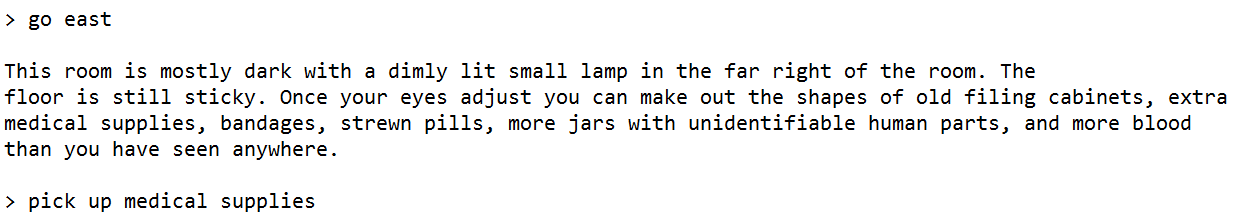
Upon level completion, player will be asked if they wish to save the game.

**2.4.4.1 Screen Mockups**

**Screen Mockup 1**



**Screen Mockup 2**



**2.4.4.2 Navigation**

Player uses command “Go (Direction)” to travel from room to room using their various directions (exits).

**3. Glossary**

Artifact-an item that the player may use in order to solve a puzzle, to equip to increase a player’s temporary max stats, or to increase a player’s health

Player-the monster entity that the user controls to navigate rooms, fight zombies, and collect artifacts

Puzzle-a multiple choice question that the player must answer to progress to the next room of the game

Zombie-a monster entity that shows up in a room that is needed to be defeated by the player