Software Requirements and Design Document

For

Group <17>

Version 1.0

Authors:
Zachary H
Marat B
Aidan S
Andres N

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

The game starts with a title screen, able to adjust different settings and difficulty level, then upon clicking play they start in the lobby scene. Here they learn the tutorial, the lore of the game, the combat style, and other basic stuff. They are then able to go to the next room which contains a bunker and different paths. They learn here they must get to the end of each path to obtain a key to unlock the bunker and defeat the final boss. The player will start out with 8 basic cards and 3 low healing flasks. When the player starts their run they are placed into the map scene where they have three options of rooms to go to, a small, medium, or large sized room. The player does not know which is which and they all lead to the same ending. Each run will consist of 5-10 rooms, campfire rooms in between each, and then a final miniboss room. The campfire room will be where the player can use the cards they found in the previous room in their current deck, they can end their run here or keep going, and the campfire will also give the player back some flasks. Players will be able to obtain coins and cards on the run, all will be gone if the player does not end their run before dying or if they die in the boss room. The player will be able to use the coins to upgrade their flasks and be able to buy some more basic cards at the lobby. More specialty cards will only be obtainable in the forest.

In each combat room there will be a start and end which the player needs to find, trying to avoid all the monsters and opening chests along the way. The monsters will aggro to the player at a certain range and trigger the combat scene. The combat scene will consist of the player, the four tiles they may stand on, the enemy/enemies and the four tiles they may stand on, a movement crystal, your current deck, a discard pile, and your current hand. The battles play out with the player being able to use their movement crystal to either move to one of the two tiles next to them. Then they may play their cards, using up to three elixir, where each card will cost one to three elixir based on how strong it is. Each card consists of the tile/tiles the player must be standing on to use and the tile/tiles the card will deal damage to. The cards the player uses will get discarded and the cards that were not in the starting hand will be randomly chosen to fill up the missing cards that were just used. The enemies will have a set attack and damage move, and will also have a set movement they may do. The enemies will not have cards.

2. Functional Requirements (10 points)

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just <u>what</u>). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

-The user should be able to move their character in four directions using “wasd”, disallowing diagonal movement. (low)
-The system should track the users inventory and stats: all of their cards they own, their current deck, their current health, coin count, etc. (high)
-The system should also track what the player gains on runs and add it on to their main inventory if they end the run or complete it. (high)
-The player should be able to buy new cards and upgrade their healing stats in the lobby shop. (medium)
-The enemies should be able to aggro to the player and trigger a combat scene (high)
-The player should be able to end their run at the campfires (medium)
-The system should display a HUD during the runs showing health, coin count, and zone area (high)
-The combat system will need to get four random cards from the players deck of eight for their first hand (low)

-The combat system will need to monitor the player's and enemies tile spot, work with each cards specific tile attack spot, and the movement crystal which the player uses each turn to move tiles (high)
-The enemies will need to have a set attack ability, the ability to change which tile it will attack, and the ability to move tiles. Each enemy will have its own move/attack set. (high)
-The combat system will need to track the player's and enemies health throughout the fight. (high)
-The system should be able to allow the user to pick the room they want to go in upon start of the run and after each campfire room (low)

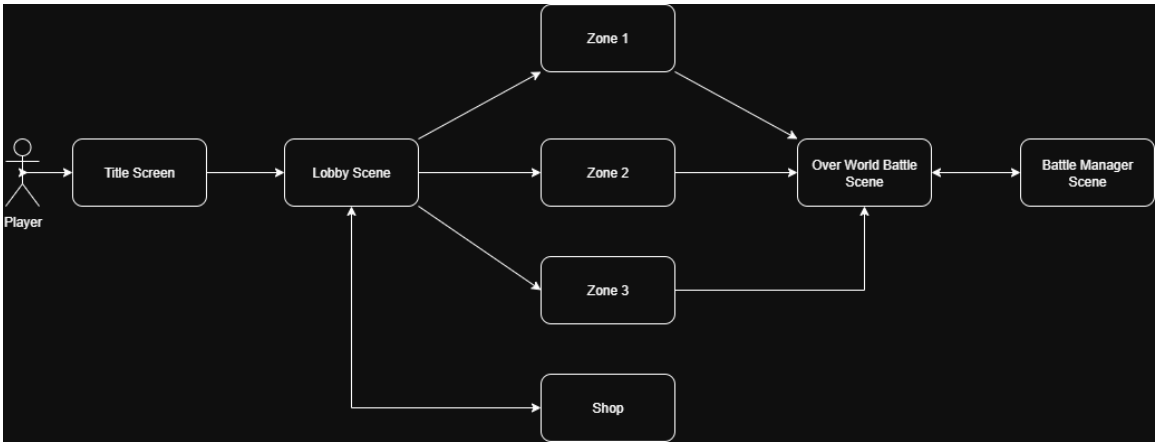
3. Non-functional Requirements (10 points)

*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

-The system should allow for easy creation of new cards, new enemies and their attack/move set, and new rooms.
-The system should be able to run at 60 fps.
-The system should allow for changeable settings, such as volume for game sounds and game music, different controls, and others like lighting.
-The system should be able to store the players inventory with easy ability to add in new items.
-The system should be able to store everything the user finds on a run and have the ability to add it to the main inventory or lose it all upon death.
-The system should be able to load in new scenes in a reasonable amount of time (2-5 seconds max).
-The system should have autosaves at different points in case of crashes.
-The system should have an easy-to-use interface.
-The system should give the user clear instructions on how to use any new mechanic they may come across.
-The system should provide an easy to follow route of the game the player should be following, giving error messages if the player tries to do something they cannot.
-The system should be able to run on both Windows and MacOS.
-The system should allow the developers to add or update features.
-The codebase should have consistent naming conventions, version control, and proper documentation practices.

4. Use Case Diagram (10 points)

*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*
***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



5. Class Diagram and/or Sequence Diagrams (15 points)

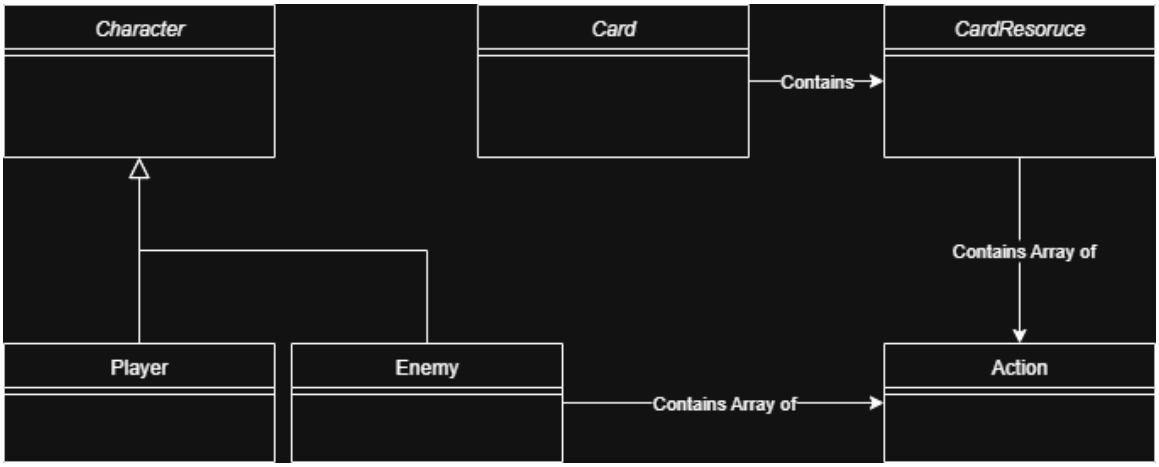
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main **paradigm** in your system is **not Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*



6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

The software will operate on a desktop PC (Windows, MacOS), and will need to peacefully coexist with standard system applications. Because the game is very lightweight, system requirements include: A GPU capable of running the application at 60 fps and rendering 2D, 2-4 GBs of RAM, and a dual-core processor.

7. Assumptions and Dependencies (5 points)

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

Assume that the Godot Engine (version 4.x) will remain stable and supported throughout the development of this project, and that any updates will not break the application. Assume that Windows and MacOS will continue to support executables and packages of Godot Engine. Assume GitHub will provide consistent support to Godot Engine, as the platform is used for version control and team collaboration. Since the game has very low graphical requirements, assume the user has a capable machine of running the software.