**Software Requirements and Design Document**

**For**

**Group 13**

Version 2.0

**Authors**:

Thomas Marconi

Jack Hayes

Tyler Flaherty

Thomas Cano

# Overview (5 points)

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

Bounties and Blades is our take on a turn-based Strategy RPG, which is inspired by games such as Fire Emblem and Dungeons and Dragons. In Bounties and Blades, you choose the team of three heroes that you will take on your quest, going through various rooms filled with dangerous enemies and magical treasures.

You’ll choose a party of three heroes from a cast of various types (i.e. fighter, ranger, rogue, etc.), control this team, and progress through a dungeon fighting enemies of similar classes. You’ll be able to move your heroes around in a turn-based combat where enemies also move to attack you. There will be items and an inventory management system where you can equip items to your characters to give them increases to their stats.

# 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 what). 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 game must have 30+ player models, low priority.

The game must have at least five unique classes, medium priority.

The game must support a party size of three, high priority.

# 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 game must be able to handle user input for movement and combat.

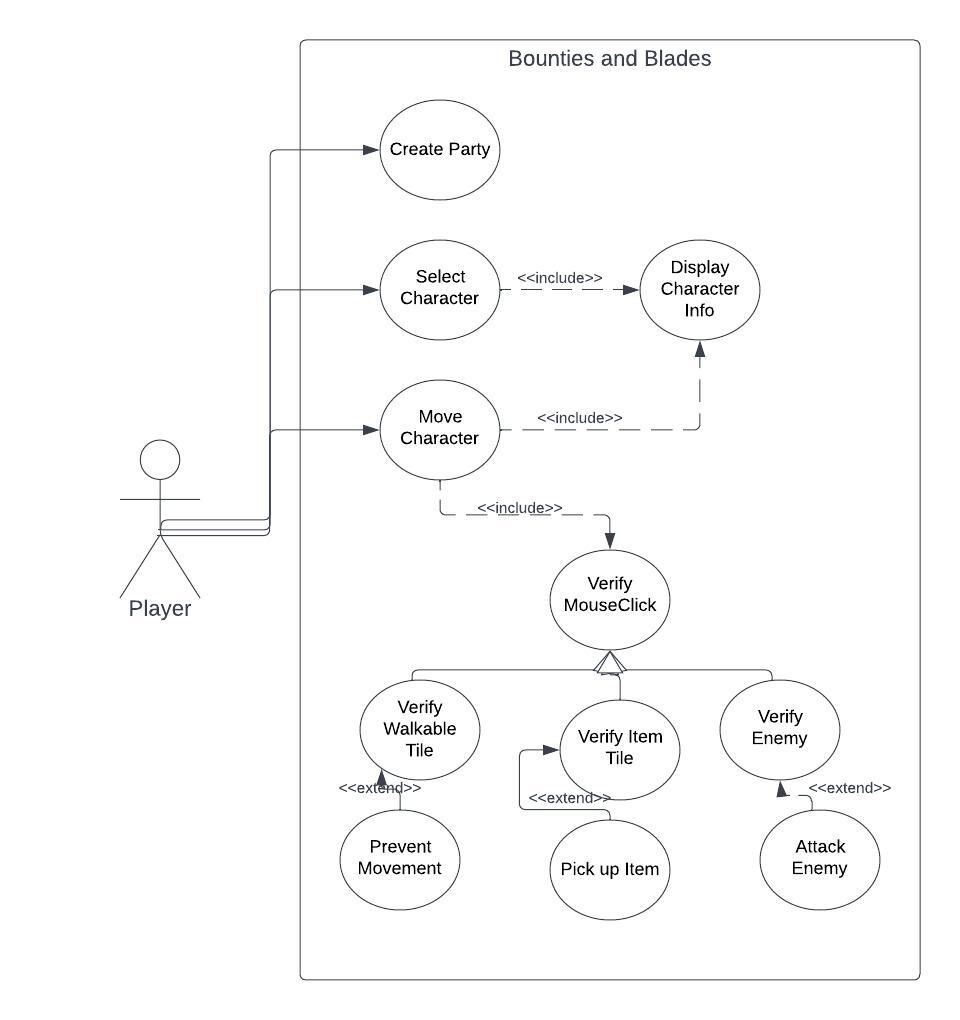
The game has to run smoothly on most computers, i.e. not have memory leaks or mismanage system resources.

The player must be able to restart the game either upon death or by the player's choice.

# 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.*



A player needs to be able to do 3 basic things: create a party, select a character and move a character. The first of these will be creating a party. When the party is created, this will lead into the game where a player can select a character. When a character is selected, their information (stats, health, etc.) should be displayed. This information also presents itself when a player wants to move a character. Because movement is handled by clicking where you want to go, the game will need to verify wherever you clicked is somewhere where you are able to move or an enemy. If there is an item, your character will pick it up and you will continue to move to that tile.

# Class Diagram and/or Sequence Diagrams (15 points)

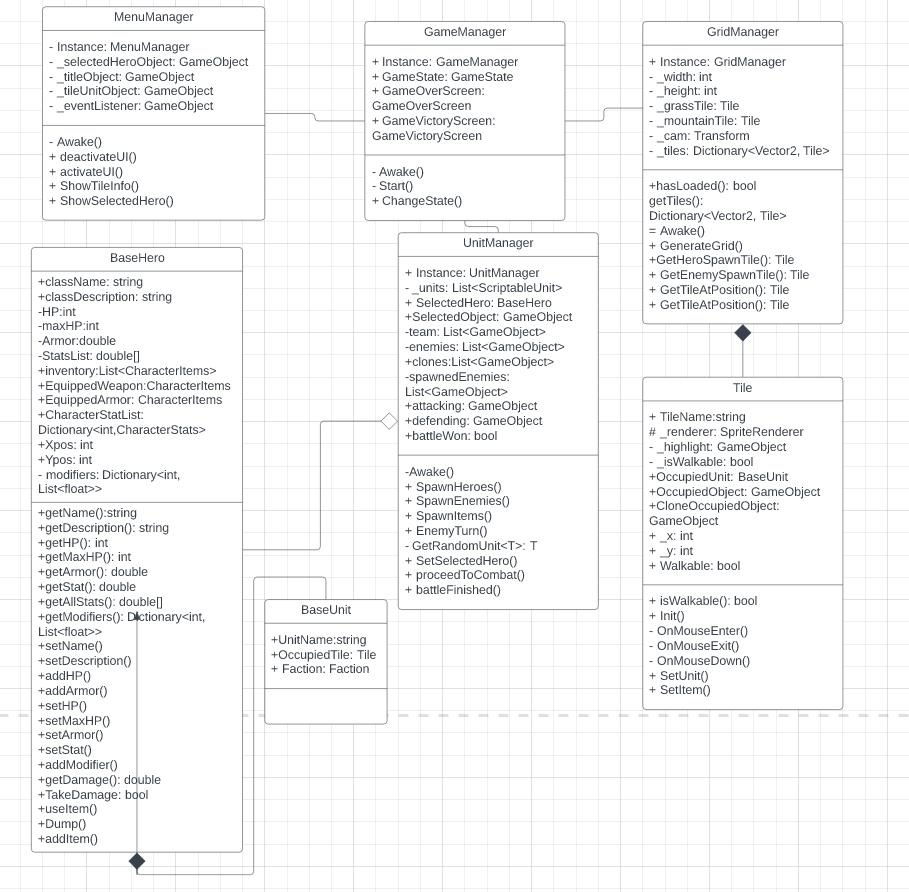
*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

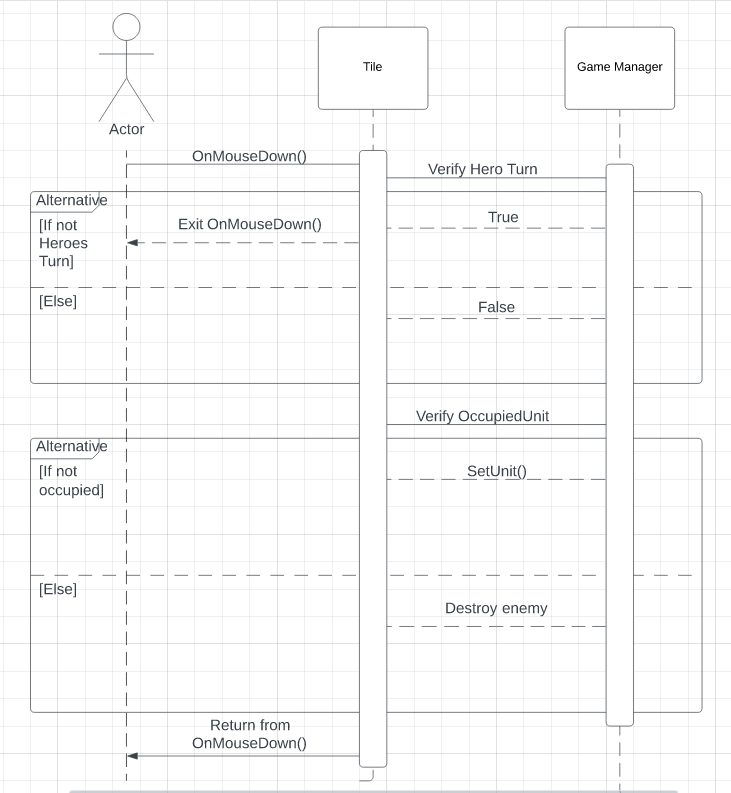
*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*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.*

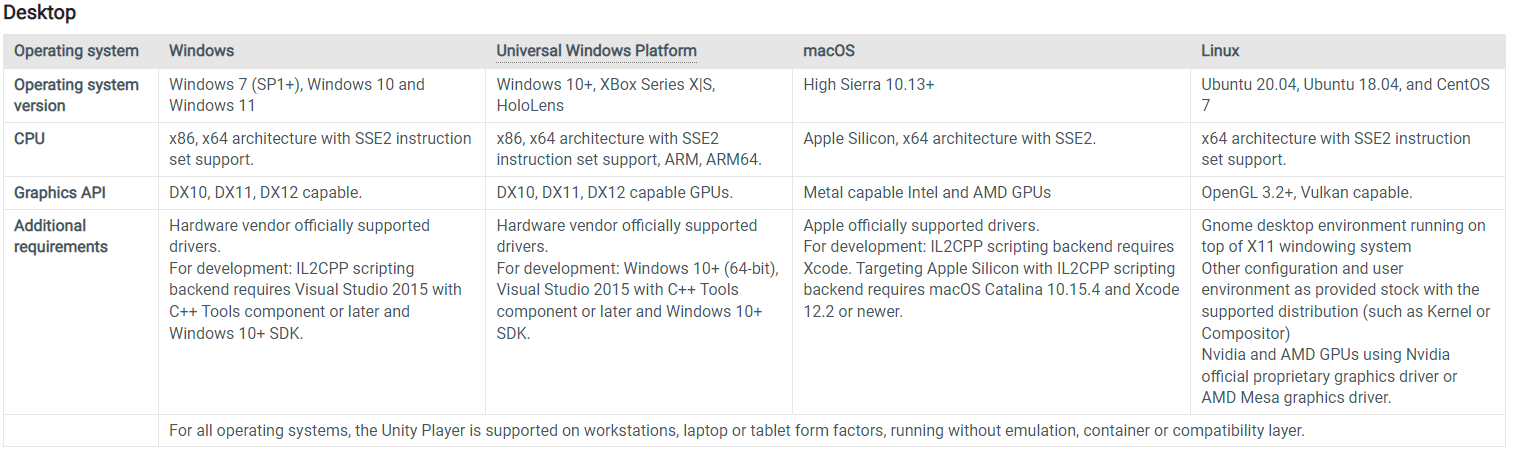
*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*





# 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 game is being built on unity 2021.3.18f1 so it should be compatible with desktop systems that can run that unity version. These are the system requirements specified on the unity website for our version of unity. 

# 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.*

A software component we intend to use is the stats modifier classes from a third-party creator. We are assuming that we will be able to smoothly integrate the code from said classes into our game and make it work with the code we intend to write for our player classes and enemy classes, as well as our inventory system. An issue that could come with this is that if we cannot figure out how to integrate the classes into our system successfully, we would have to rewrite the code for stat modifiers ourselves from scratch, focusing on high cohesion between the classes. The code in question comes from this series of youtube videos: [Character Stats in Unity #1 - Base Implementation](https://www.youtube.com/watch?v=SH25f3cXBVc)

Our project’s movement mechanics and overall function, depends on a project we found from a very helpful content creator that focuses on game development in Unity, in these videos, he touches up on fundamental topics for game development in Unity, such as the extremely important Game Manager script that a majority of Unity games have that keep track of things such as the state of the game, which is the backbone for any functional game. Although our implementation will have differences because our game is different than the one being developed in his videos, there is still a significant chunk of boilerplate code that we depend on having for our game to function properly. The content creator we used is called Tarodev and the following link is the one we used for our implementation: [Create a grid in Unity - Perfect for tactics or turn-based games! Part 2](https://www.youtube.com/watch?v=f5pm29yhaTs)

Our projects battle system, which is a 1-on-1 turn based combat scene, was taken from a youtube channel that has various unity tutorials for game development. This battle system was chosen because it was the simplest to introduce to what we had already implemented in our project. We ideally would’ve had a 3 vs 3 battle system, but that would have required us to scrap our current movement system completely. We changed some sections of the code to work with our project but the majority of the battle scene stems from this project and should be cited:

[Turn-Based Combat in Unity](https://youtu.be/_1pz_ohupPs)