**Software Requirements and Design Document**

**For**

**Group 18**

04/28/2019

Version 2.0

**Authors**:

Matthew Wix

Riley Garrison

Michael Tesfaye

Ryan Kenney

# Overview

We are creating a web-based open world 2D RPG game that uses frontend and backend technologies. Our game takes place in a fantasy kingdom and utilizes weapon-based “hack and slash” gameplay. People who play the game have their own profile with their saved data and customized information that is stored in an online database. The game has a complex story with multiple branching paths based on dialogue options and character actions that result in different potential endings.

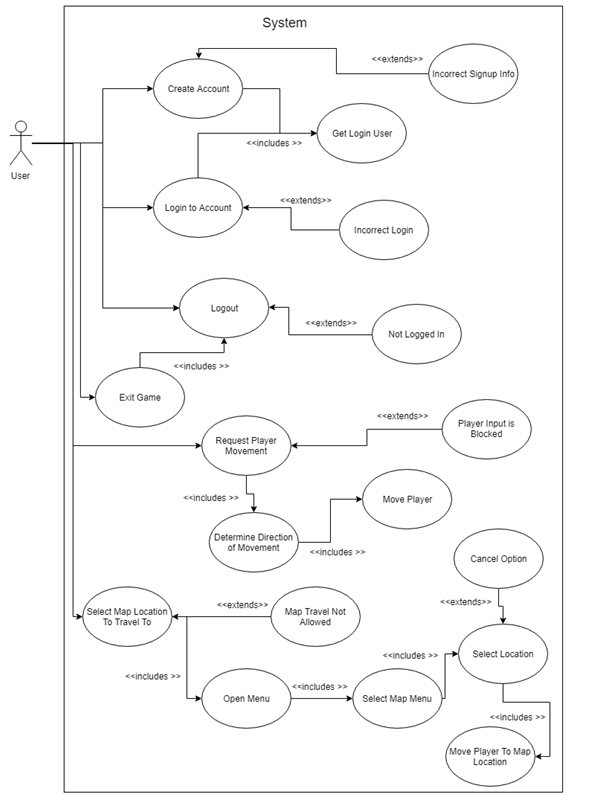
# Functional Requirements

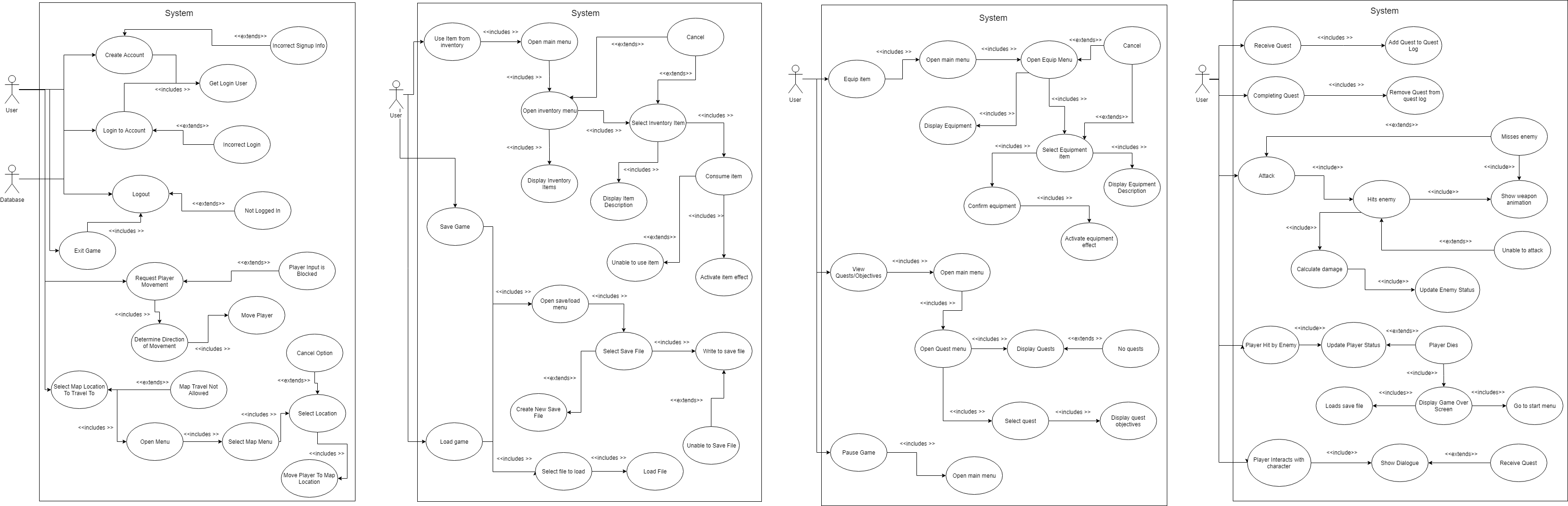
1. **Player movement (high) –** The system will allow the player sprite to move in any direction based upon the arrow key that the user presses (up arrow key for upward movement, left arrow key for leftward movement, etc.). The system will check to make sure the player is in a state where movement is allowed first before initiating movement (i.e. not moving in the direction of a solid block, not engaged in dialogue, not in the pause menu).
2. **Selecting a map location to travel to (medium) –** The system will allow the player to open the pause menu, select the map menu option, and select a location from the map that they wish to travel to and then update their location to the option they have chosen.
3. **Receiving quests (high) –** The system will allow the player to receive quests with objectives that need to be completed. The quests can be received from NPCs or as a result of completing another quest. When the quest is received, it will be stored in the quest log and able for the player to view in the pause menu under the quests menu option.
4. **Branching Story Paths (high) –** The system will allow players to choose between different objectives given to them by a quest and will choose a corresponding story path based on that choice.
5. **Player uses weapon (high) –** The system will allow the player to use their weapon provided they are not in the pause menu or engaged in dialogue. When the weapon is used, a corresponding animation will be played and if the player is in range of an enemy, damage calculation will be performed.
6. **Damage calculation in combat (high) –** The system will perform a calculation of the damage that will be inflicted by the player on the enemy or vice versa depending on the stats for the weapon used by the attacker and the armor points of the entity being attacked. If the resulting damage causes the entity’s health to drop to 0 or less, then the death flag will be triggered for that entity.
7. **Enemy death (high) –** The system will remove an enemy from the map if it is detected that its health is at 0 or below.
8. **Player death (high) –** The system will trigger a Game Over screen if it is detected that the player’s health is at 0 or below and will then transition to the start menu.
9. **Dialogue (medium) –** The system will display the dialogue associated with an NPC if the user is in range of an NPC and presses the appropriate button to initiate dialogue.
10. **User logs in (high) –** The system will connect with a Google Firebase database that contains a list of registered users. If the user exists and the correct credentials are entered, the information associated with the user’s profile will be loaded in from the database. Otherwise an error message will be displayed.
11. **User signs up (high) -** The system will connect with a Google Firebase database and attempt to add a new user to the existing list so long as valid credentials are entered (i.e. the user does not already exist and the password has a long enough length)
12. **Add item to inventory (high) –** The system will allow the player to interact with an item and add it to their inventory list. The inventory menu option on the pause menu will also be updated to reflect this.
13. **Using items (high) –** The system will allow players to use items in their inventory by selecting it from the list of items displayed inventory menu option under the pause menu. Once selected, the item’s effect will be triggered.
14. **Pausing the game (medium) –** The player will be able to press the escape key to pause the game any time they are not engaged in dialogue, and a pause menu will be displayed upon doing so.
15. **Load graphics (high) –** The system will wait until the graphics have been loaded by the browser before displaying them. The system will constantly refresh the graphics being displayed on the screen and react to any changes such as input from the user.
16. **Handle keyboard input (high) –** The system will take in keyboard input from the user and use that to perform the appropriate operation associated with the key that was pressed.

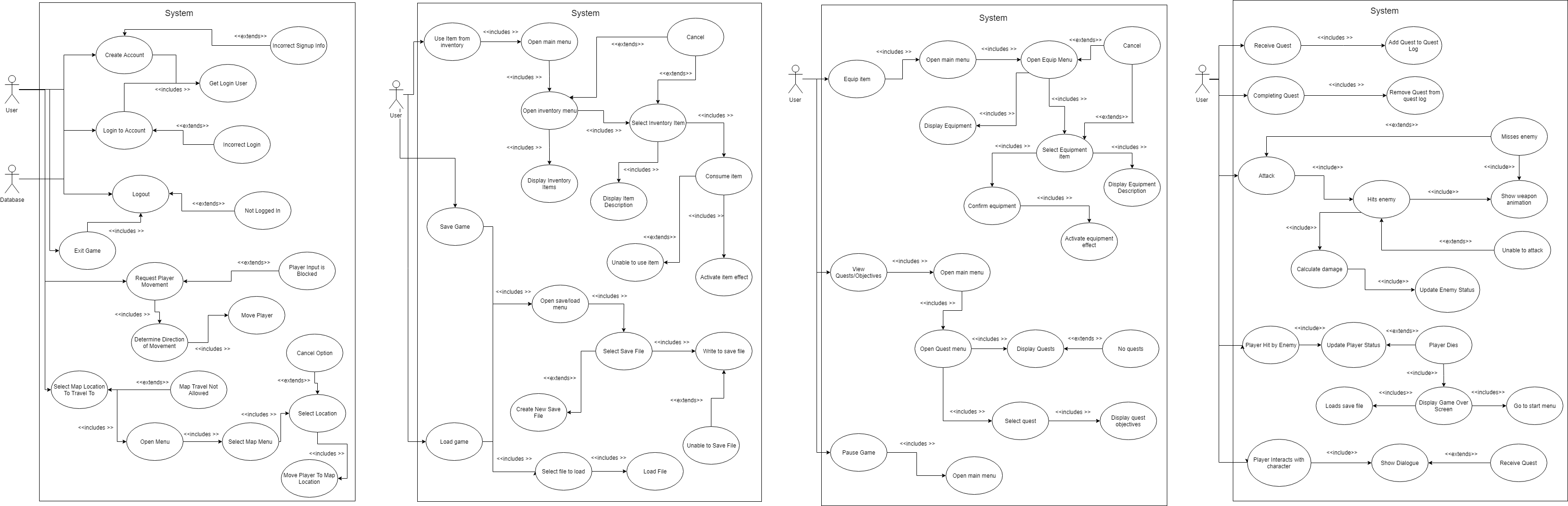
# Non-functional Requirements

1. **Game will run at 60fps:** the game will perform all game state changes at a speed that ensures it is capable of displaying all required parts 60 times per second. This means that complex calculations must not be used and too many calculations should not be performed simultaneously.
2. **Save and load game functions will complete in less than 3 seconds:** the data transferred between the player’s game and the database must be efficiently stored and of a small enough size to ensure that the loading and saving of the game is done quickly. However, it should still be 100% accurate. This is assuming that the user has an internet speed of 1 Mb/s or greater.
3. **Screen will precisely track the player’s movement:** the function that moves the screen must be accurate and capable of fixing temporary errors to ensure that the player’s character never appears off screen.
4. **Passwords must be a minimum of 6 characters long:** a minimum password requirement reduces the chances that an account will be accessed by someone other than its creator.
5. **Database will only have downtime during updates of modifications:** the database should never be inaccessible as a result of errors in the source code.
6. **Database will be able to support 10 concurrent users/players:** the database should be able to support 10 players saving and loading their games at the same time.
7. **Game will function on both Chrome and Firefox browsers:** both browsers should run the game with no errors specific to one.
8. **Player’s inventory will be accurately maintained:** when the player attains an item it should never disappear as a result of accessing an old inventory state, unless the player loads an older save over their current game.
9. **Game will function for screen with widths of 800px or greater:** the minimum screen size required is 800px wide, but the game should be capable of handling screen widths of anything greater.
10. **Keyboard commands will be handled as soon as they are input:** when a user issues a command through keyboard input, the game will acknowledge and begin to interpret it as soon as it is pressed. This means that problems like multiple keys being pressed simultaneously must be addressed to ensure that the user’s inputs are recognized.
11. **Game saves must be preserved in the database:** when a game save file is placed in the database, it is important that it is not erased until it is overwritten by another save file.

# Use Case Diagram

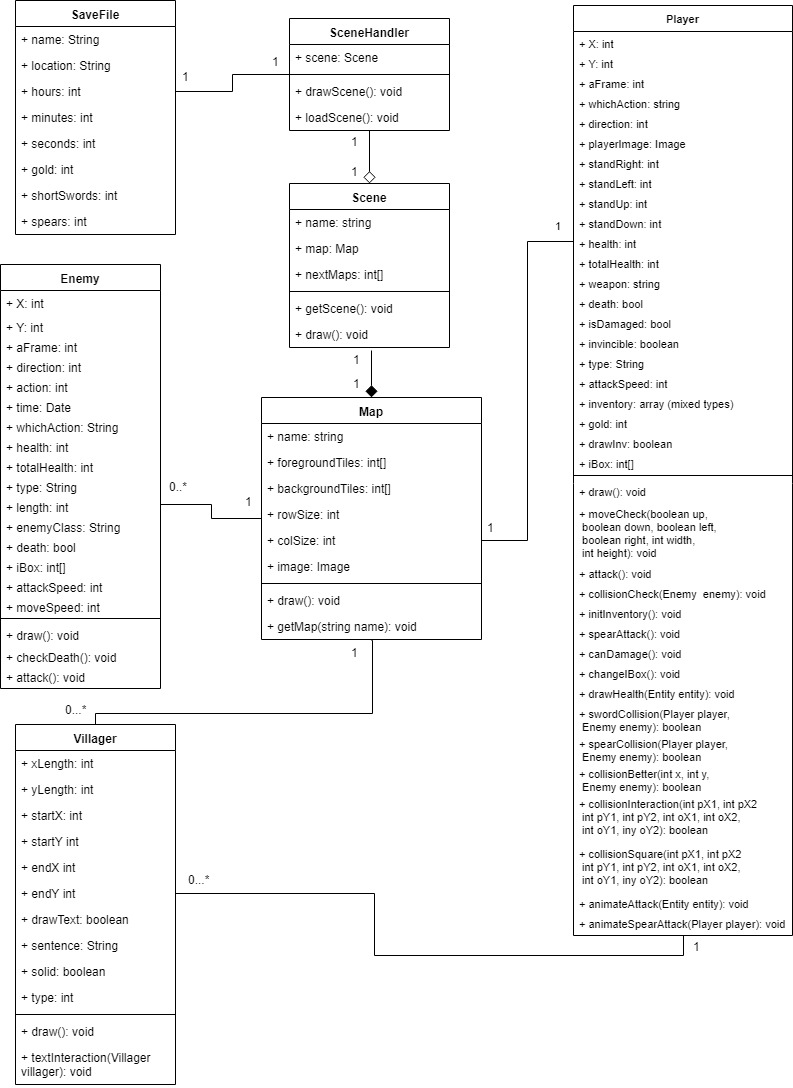






# 

# Class Diagram



# Operating Environment

The game will operate on devices with Windows and MacOS operating systems, specifically on the most current versions (Windows 10 and Mojave, respectively). It is likely to function normally on other recent versions of these operating systems (Windows 8, Sierra, etc), but all testing was performed on the most current versions. On these operating systems it will be accessed through a web browser. Testing was performed on the web browsers Google Chrome and Mozilla Firefox, but Opera, Internet Explorer, and Microsoft Edge should also be capable of running the game normally. JavaScript must be enabled by the browser for the game to function, and an internet connection is required to access the website hosting the game as well as the database that stores user accounts. A mouse/pointer device and a keyboard are required to navigate the site, create an account, and login, but only a keyboard is required to play once in the game.

# Assumptions and Dependencies

Our project assumes that the Firebase database can be connected to and that all internal handling of game data by that database is functioning properly. It is also assumes that the web browser a user is using to play the game functions properly and is capable of interpreting the source code of the website and game. The game assumes the user has an internet connection capable of loading in images ranging from a few kilobytes to a few megabytes in size. The game also relies on the user running the game on a computer with a keyboard rather than a mobile device.