PEGLEG PIRATES
SRS DOCUMENT

Night and Day Games, LLC.

Jonathan Adams
Matthew Bradley
Angela Lollis
Mikal Wessel

Contents

1 Introduction

1.1 Document Revision Tracking

This document is tracked and organized by the table listing below. Any action made in this document is performed under strict logging to this tracking table. Access to the document is performed in good faith that no action shall be taken on the document without expressly acknowledging the changes performed in the tracking log.

Name	Role	Date	Action
Jonathan Adams	System Engineer	11/26/2016	Initial Creation
Jonathan Adams	System Engineer	12/01/2016	Addition of Individual Use-Cases.
Jonathan Adams	System Engineer	12/03/2016	Expanded Product Functions section.
Jonathan Adams	System Engineer	12/03/2016	Incorporated Edits and Review Comments for Draft version 2.
Jonathan Adams	System Engineer	12/07/2016	Fixed formatting of document after PDF failure during Draft 2.
Jonathan Adams	System Engineer	12/08/2016	Incorporated settings use case from Mikal Wessel and Angela Lollis.
Jonathan Adams	System Engineer	12/09/2016	Incorporated settings use cases from Matthew Bradley.
Jonathan Adams	System Engineer	12/10/2016	Input requirements outlined during development.
Jonathan Adams	System Engineer	12/10/2016	Performed final polish review of document. Appendix and Index removed for lack of necessity.

Table 1 - Document Revision Matrix

1.2 Purpose

The purpose of this document is to give a detailed description of the requirements for the game software “PegLeg Pirates”. This document will describe the features, functional requirements, nonfunctional requirements, development of the game, the environment under which the game must operate, and the planned release of the game. This document is intended for customer(s) approval and as a reference for the development team.

1.3 Scope

Below is information regarding the project scope and expected deliverables. This outline is highly subject to change once more information in understood about the deliverables expected in the subsequent semester of CEN 4021.

1.3.1 Project Objectives

The project objective is to develop and deliver an entertainment software product.

1.3.2 Goals

The project goals are listed below.

- Develop a prototype for the designed entertainment software product.
- Design and document the design for the entertainment software product.
- Provide requirements documentation and other documents as expected from the class.

1.3.3 Sub-Phases

The project phases shall follow the design listed in the swim-lane development diagram. Since long-term project phase cannot be fully determined until the expectations of CEN 4021 are established, this swimlane diagram outlines the prototype deliverable only.

Week	**11-5**	**11-12**	**11-19**	**11-26**	**12-3**	**12-10**
Activity						
Prospectus Documentation						
User Interface & Landing Scene Development						
Individual Market Research Document						
Individual Market Research Presentation						
Game Character & Controls Development						
Game Design Document & Outline						
Draft of the SRS (1)						
Draft of the SRS (2)						
Final SRS Deliverable						
Demo Level Design & Development						
Game Back-End Code (Combat, Multiplayer, Other Functionality, etc.)						
Software Prototype Testing						
Prototype Deliverable and Presentation						

Table 2 - Software Phase Schedule

1.3.4 Tasks

There is currently only one major deliverable task exterior to documentation requirements for the academic semester. A prototype development of the software shall be delivered at the end of the term, in addition to the requirements design documents, game design documents, and market research documentation for the software product.

Expectations for the full development of the software are not fully understood at this time; however, the implied expectation is for the development of a deliverable product by the end of the subsequent semester. Once these expectations are defined, the Software Prototype will be revised and expanded iteratively, and a new phase schedule will be outlined.

1.3.5 Resources

This software shall be developed utilizing free-use software tools. Unity3D shall be the software tool utilized for developing the code and game. Unity3D provides the capability to compile, build, edit, and develop code as well as create and associate game functionality and objects. The developed code shall be configured, controlled, distributed, and hosted by the GitHub code repository. All members of the project shall use an independent tool of their choosing to manage code retrieval from the code repository; however, it is also expected that they will edit and upload code to the repository in a safe and documented manner. Additionally, the project communication and coordination shall be managed through Slack, G-mail, Google Calendar, Google Chat, Asana, and Skype software tools. Project documentation and related information shall be managed with Google Documents, and Asana.

1.3.6 Budget

There is no budget for this project. This project is developed without funding, and is limited as an academic development.

1.3.7 Schedule

The schedule for this project is determined by the availability of class electives and deadlines provided by the educator. See the sub-phases section, table 2, for a strongly defined schedule for the prototype deliverable.

1.3.8 Project Architecture

The software game project “PegLeg Pirates” will be developed using all the free-to-use software tools that were decided upon during the initial individual research and group decision during the planning stage of development. The free-to-use software tools of choice for development are as follows in the section below.

1.3.8.1 Game Engine Framework

Unity3D will be the game engine software used for developing all of the code and the game objects and levels. Unity3D will allow team members to do all of the necessary tasks that come along with game

development such as: IDE for code editing & developing, object relation and mapping, level & UI designing, build, compile, and constructing game functionality.

1.3.8.2 Language

Since the game engine chosen is Unity3D, we will be adhering to the programming language of development for Unity3D. Unity3D uses C# and JavaScript coupled with scripting API's that Unity Engine provides developers to use to tie into gaming objects that are created. There is a wealth of information and examples on the scripting languages on Unity3D's website as well as a large internet community where we will be able to learn, and find help for all of our development using C#, JavaScript, and Unity Engine API's.

1.3.8.3 Platform

The platform for development will be a combination of Windows, Mac OS, and Linux platforms. Unity3D is cross platform for development so the developers were able to choose their operating system of choice for development.

The intended target platforms are limited to the compile-and-build capabilities of the game engine framework Unity3D. Unity3D supports compile-and-building of software for Windows, Linux, Mac, as well as mobile platforms. The game software PegLeg Pirates will target desktop environments Windows, Linux, and Mac until either scale of the software increases, or challenges during software development are met that limit the systems.

1.4 Glossary

These common terms are used throughout the document.

Term	Definition
Game	A game is a final product produced from the software framework utilized. As a final product, it is an executable software which provides entertainment to the target user.
Player	A player is a synonymous term for a user. A player is a user who is currently using the software and participating in interactions with the software.
Character	A character is a user's avatar, their controllable actor represented in the game while it executes. A player may have many characters, but they may only access, control, and use one character at a time.
Game Engine	A Game Engine is a software framework used to develop the target product.

Table 3 - Glossary of Terms

1.5 Reference

- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
- Section 508 Software and Website Accessibility Compliance Regulations. GSA Government-wide Section 508 Accessibility Program, 2016. http://www.section508.gov/

2 Game Design Overview

2.1 Description

2.1.1 Elevator Pitch

Avast! The seas call to you in *PegLeg Pirates*! Embark with your fellow ne'er-do-wells aboard the great ship, the *Kraken Killer*! Take to arms and sabers with your fellow eye-patched powder monkeys in your quest to stop other freebooters from plundering your hard earned booty! There's nothing better than the salty spray of the open seas and gold-filled treasure chests; so raise your Jolly Roger, and set your sails for the nearest ship!

2.1.2 Project Description Overview

This game is a first-person multiplayer game in which players join a team of other players over the internet. The team of players is formed online, through a server-lobby system. When the team joins a server, they compete against another team that has joined the same server. The two teams fight against each other through various scenarios in a game level to protect their resources (called a "match").

Players may select roles to play which allow them to access unique styles of tactics and gameplay. The roles are represented as types of characters, and each role has access to special tools and weapons to use during the match. Players may change their role at the start of a match.

Once the match is completed, the player's character gains experience points and other rewards that they may use to customize their character. They are returned to the lobby system and may join a new server and subsequently a new team.

2.1.3 Influences, Tones, and Themes

This software is designed similarly to competitor products. Other products within the genre of this software are Overwatch, Call of Duty, Star Wars: Battlefront, and TeamFortress2. This game is not designed to copy these products, but will utilize similar features and mechanics.

The art and style of this product shall be cartoony, comedic pirates. The characters, levels, user interface, and all artwork associated with the final product shall be designed with these tones. This choice was made in order to target the largest consume audience with the least amount of violence and limit material which might not be suitable for children and young adults.

2.2 Product Perspective

PegLeg Pirates is a stand-alone executable produced from development with the Game Engine.

2.2.1 System interfaces

This product is a stand-alone executable. No other system interfaces are required.

2.2.2 User interfaces

2.2.2.1 Landing Screens

The Landing Screens are a series of screens displayed at the entry and termination of the game, and is synonymous with the Main Menu. In these screens, the user may navigate to separate portions of the game, and to interact with different components.

2.2.2.2 Heads-Up Display (HUD)

The Heads-Up Display (HUD) is the main interface for the player to play the game. The HUD is very slim and covers a very small amount of the screen. This display shows players which abilities they have slotted, what special items or potions they have equipped, and a small map that they may toggle off and on to display. As well, the HUD also shows the player's health, stamina, and intellect pools in the form of bars.

2.2.2.3 In-Game Interfaces

In the game, there are other interfaces which are not part of the Main Menu or the HUD. These interfaces show specific information to the player, such as who is speaking on voice chat, or the current objective. These interfaces are usually billboard effects and are displayed in game as game objects rather than 2D interfaces to the player. These interface features help the play navigate through levels in the game or notify them of important information relevant to decision making and playing.

2.2.3 Hardware interfaces

This product is a stand-alone executable. No other hardware interfaces are required.

2.2.4 Software interfaces

This product shall interface with the Steam Client to manage transactions, hosting, updates, and other features provided through Steam.

Software Name	Steam Client
Mnemonic	N/A
Specification	N/A
Version	Steam API v017
Source	N/A

2.2.5 Communications interfaces

This product shall maintain a network interface that allows the product to communicate over a network. The specific design of the network interface shall use a client-server architecture. Clients and Servers shall be separate running modes of the primary product, and users shall be able to initialize and operate servers from their game executable.

2.3 Product Functions

This game shall use a multiplayer Team FPS pattern for design, and this pattern will be developed using the Unity3D engine.

The functional process of this game follows a simple program flow; players shall start the game, and arrive at series of UI landing screens, with buttons to navigate through various menu choices. By selecting a specific option, players are brought to a screen with a list of available gameplay servers.

The player may select a server and join it, or view properties of the server. The tabled list of servers shall show the location of the server, the name of the server, the volume of player currently on the server, the maximum player volume for the server, password-limited access, and other features.

One a player selects a server to join, they are shown a landing screen with the current arena-gameplay camera as a background. Players may select from an available list of characters which character they wish to use, and then also which team they wish to join. When they select join, the character is spawned at the spawn point, and they are given control of the character and can participate in the game.

The game will display the current scenario, and various other HUD features while playing. The player will continue playing in the server through various time-limited scenarios or they may quit the game and leave the server. Once they leave the server, they are shown the server selection list. They may exit this list, and quit the game or they may select another server.

2.3.1 Gameplay Scenarios

There are a few gameplay scenarios available to the player in a server. The initial scenario is selected when a user initializes a server. Then, the scenario is played when players join the server. Once the scenario is finished, all players who remain in the server and wish to continue playing in the next game scenario are prompted with a vote. All players then vote on which scenario to play next with ties broken by selected from the most popular at random. The scenario begins, and is played by users until the end, and the selection of the next scenario by vote is repeated.

Scenarios are based on time and objective, and terminate when the specific limitation has been met. Some scenario objectives may be played by one team, and then repeated by another team, and the winner determined by speed of completing the objective. The specific limiter of a scenario is described in each scenario definition below.

2.3.1.1 Scenario: Raid the Pirate Cover

2.3.1.1.1 Limiter

Teams are split into attackers and defenders. Attackers are timed on their ability to complete the main objective. Once the objective is completed, the scenario is reset and teams are switched. The new attackers must complete the objective in a shorter time to be considered the winners.

2.3.1.1.2 Description

In this scenario, players are challenged with entering an enemy's secret cove, reaching their storage room, and then stealing all the loot in the room. With elements similar to capture the flag styled games, the players must seek the room and return the contained items to their specific starting point (i.e. their ship in the cove). However, in order to breach the cover and enter the room, they must pass through a series of defenses organized as a set run-the-gauntlet.

Once the cove has been raided, and all the loot stolen from the defending team, the scenario ends. Teams are then swapped, and the team that was the defenders now replays the scenario as the attackers, and must beat the scenario in a time that is less the predecessors'. Once the scenario is finished, the winners receive loot and experience for winning.

2.3.1.2 Scenario: Team vs. Team Ship-to-Ship Free For All

2.3.1.2.1 Limiter

Teams are limited by time in this scenario. Teams are given a total of 20 minutes to collect as many points for their team as possible. The team with the highest amount of points after 20 minutes receives the win, loot, and experience.

2.3.1.2.2 Description

In this scenario, players are split onto one of two teams. They are placed on board a ship corresponding to their team. The ships start at opposite sides of the map, and may be controlled. The objective initially is to disable each other's rudders or sails so that the ship stops moving. Once this ship is disabled, it may be boarded. Once the ship is boarded, players fight against each other. A ship may be boarded in various ways: by raft, by rope swing, by plank, or simply by swimming to it and climbing up a ladder.

Players earn points for their teams by doing specific actions, or killing other players. Disabling a ship earns a team 1000 points. Killing another player earns 50 points, and is multiplies the previous kill points by the current kill number for every subsequent kill within 10 seconds (ex. Kill number 1 is 50 points, kill

number 2 is 100 points, kill number 3 is 300 points, kill number four is 1200 points, etc.). Using a powerup earns 10 points. Using a ship-based weapon, such as a cannon emplacement, to complete an objective doubles the objective's rewarded points. Entering the booty room and stealing loot by returning it to their team's own booty earns 5000 points each time this is completed (players are marked and receive a time-incrementing debuff until they have deposited the gold or drop it). Destroying a team's ship-emplacement weapons earns 300 points. There are other easter egg objectives which will be added dependent on project development time.

2.3.1.3 Scenario: Capture the Village

2.3.1.3.1 Limiter

In this scenario, players are limited by flag-captures and points; the first team to reach the specific pointflag value is considered the winner and receives the loot and experience.

2.3.1.3.2 Description

This scenario is a ship-to-village capture the flag match. Players in both the ship and the village struggle to capture and contain the space between the team's primary flag points. Players begin on either the ship, or at the "barracks" of the village. In each starting location is the team's flag. Between the barracks and the ship is a small maze-organized village, with specific *capture points* in the maze set in larger open squares, with towers in the square and a gun emplacement in the tower.

Players must compete to capture these village squares with the tower – called capture points -- and are rewarded with ownership of the tower and access to the gun emplacement. A tower may be taken once a team has defeated all enemy players in the tower, and has waited at the capture point for 20 seconds (note: this time is reduced for each player standing at the tower flag by 2 seconds, with a minimum of 5 seconds to capture the tower flag). Capturing 3 towers is considered a full flag capture, and teams are rewarded as such; loosing 2 towers is considered a flag loss, and teams are penalized as such.

Teams must also seek to capture the flag from the opposing teams starting location. A player picks up the opposing team's flag, and then attempts to return it to their own starting location. Players may only turn in the flag they are carrying if their flag is at its starting location. While a player is holding a flag, they receive an incremental debuff making them more vulnerable to other players.

A team is considered the winner when the team is counted as having won a total of 5 flag captures. Teams are rewarded a flag capture point when they turn in the opposing team's flag to their flag location. Teams that have their flag captured and turned in by the opposing team do not have a flag point removed from their flag point capture total; rather, if they have a flag captured and turned in, and then a **single** capture point is lost within 5 minutes, then they lose a flag point. After 5 minutes, the single capture point penalty is removed, and it required a full two capture points to lose a flag capture point.

2.3.2 Characters

In this game, there are five different types of characters a player may select from. Each character has a general appearance that is shared; however, the unique appearance and design of a character can be determine by a player. As the player unlocks experience and loot, they gain the ability to improve or purchase appearance options to uniquely design their character.

When a player enters a game, they select from a list what type of character they wish to play. While in this character type, players earn experience for that character, and any experience earned for that character may be spent on that character. Experience is used to purchase new weapon types, upgrades to gear, new active abilities, and improve passive abilities.

Loot is gold the player earns at the end of a match for winning the game. Loot is collected over time and stored in a bank associated with the user account. A player may spend earned and banked loot on any character, unlike experience which is specific to a character. Loot can be used to purchase new appearance items, new visual effects, new animations, new taunts and voice packages, and various other cosmetic character features.

The types of characters available to play are described in the section below. Players have access to each character type, and may freely switch between a character at respawn or death.

2.3.2.1 Parrot Pete

Parrot Pete is a short pirate with a thick white beard, and a bulbous nose. He wears a scarf-like hat, and a small shirt that is ripped and ragged at the bottom. He has knee-length dirty trousers, and is barefooted. Pete has the unique ability of having parrot companions who sit on his shoulders. Pete's abilities allow him to summon parrots to make attacks. He also wields a pistol, and a small mallet, which he can switch between using.

2.3.2.2 The Cook

The Cook is a giant, tank-like melee fighter. He dual wields butcher's cleavers, and has a comically large weight problem. The Cook is designed to appear as a heavyset man. He is also armed with a sack of potatoes, which he may use as an AOE or group control weapon. He will slam the bag of potatoes into an enemy or the ground, exploding potatoes out around him. He can also individually throw potatoes at people. His attacks are strongly area based, but he has a very close range for these attacks.

2.3.2.3 Hand Cannon Henry

Hand Cannon Henry is a comically muscular man, which stands with an almost ape-like posture and a grimace. He has a pencil-thin handlebar mustache, and carries a massive cannon on his shoulder, and a satchel of various cannon balls. Henry's special moves are tied to the various types of cannon balls he can fire from his hand cannon. He can fire grape and cluster shots which can ensnare opponents, or hit an area. He can also swing his cannon around like a very heavy bat. He has very low defense capabilities, and a long duration between reloads, but can stand at long distances and soak up large amounts of damage.

2.3.2.4 Captain No-Beard

Captain No-Beard is a pirate with a peg leg and a hook, and a large long coat. She hobbles around with a grimace, and has a giant assortment of pistols strapped to belts and bandoleers all over her. She can use her hook as a melee weapon. Her moves are focused on crowd control, and she can use abilities like fear, stun, and other buffs. She wears a comical fake beard, and her animations often show her trying to reattach the fake beard. She also has a long-refresh-timer cannon-grenade which she can use as a flash bang to blind other players.

2.3.2.5 Scallywag Sue

Scallywag Sue is a female pirate that has stealth abilities. She dual wields kukris, and speaks with a nearAustralian accent. She is hunched over, and her animations make her look like a long-limbed climbing monkey. She is covered in tattoos, and wears big hoop earrings, and a head scarf. She can use her kukris for backstabbing and stealth attacks, or she may throw small daggers. She has special moves that allow her to dive bomb people from perches on buildings and ships above other players.

2.3.2.6 Voodoo Jim

James is a young British man with a Jekyll-and-Hyde mechanism. While in his young man form, he may act as a healer. He can throw AOE potions at team mates to heal them, or use a steam-punk gun that can steadily heal people by “shooting” them with the ray. He can also give temporary damage health buffs to other players by using a stimulant he injects in them with a needle. When he converts to his Hyde-form “Voodoo Jim,” he becomes a disease unit, infecting enemies with large diseased claws. His moves focus on using dark magic to suck the life from others, infect them with disease and poisons, and to use debuffs that decrease their health.

2.3.3 Game Mechanics

2.3.3.1 Software Structure

PegLeg Pirates software structure is based around the menu option and user interface controls. The player accesses various parts of the software and functionality through menu options presented in the Landing Page / Main Menu GUI menus. This is displayed in the activity diagram below. This diagram shows the software initialization process, and access to basic game options. Each of the menu options selected allows the user to access the functionality options and are defined below.

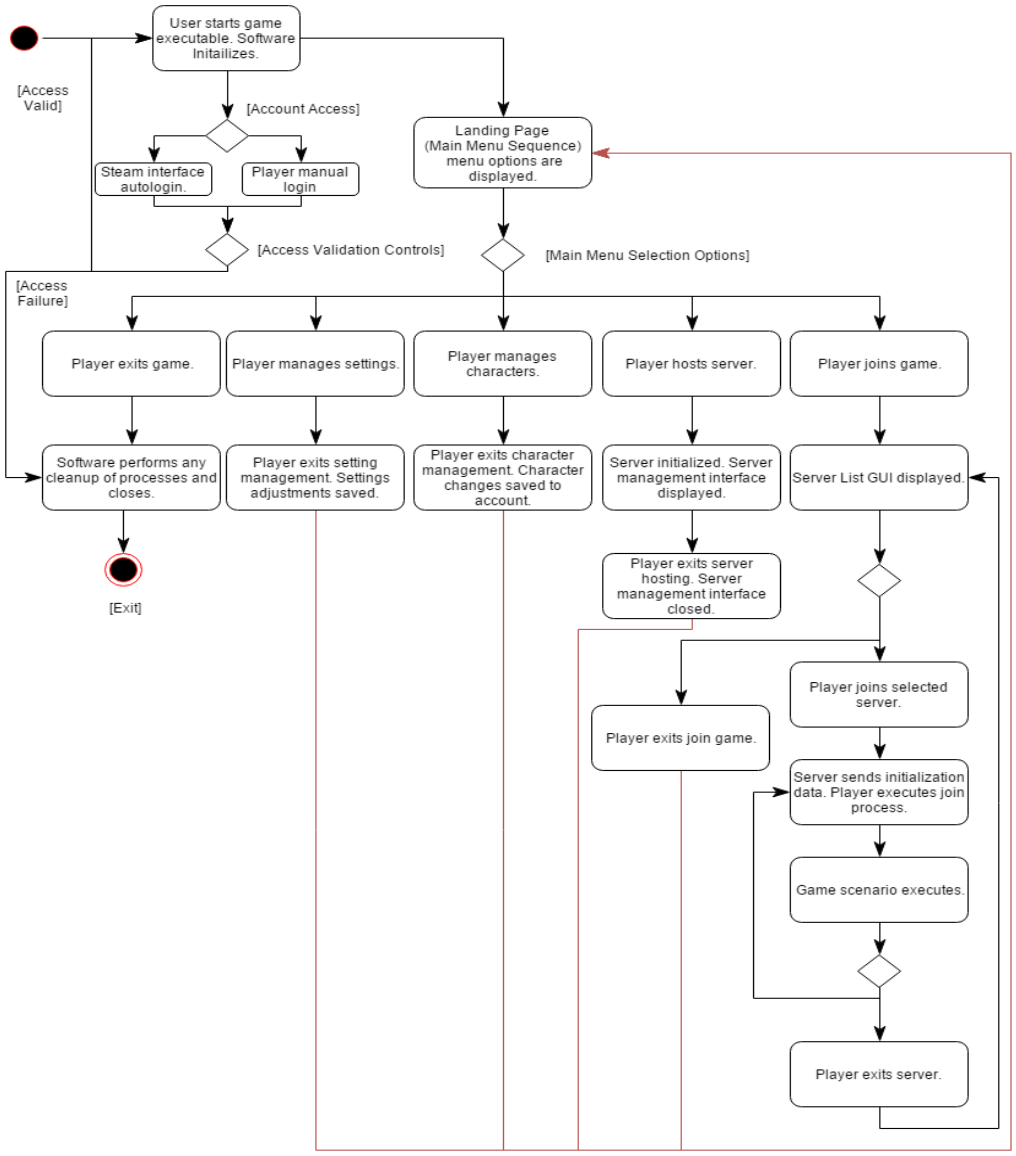

Figure 1 - PegLeg Pirates Software Activity Diagram [1]

[1] The red lines do not signify any specific difference in processing, and it is only meant to group processes for readability of the diagram.

2.3.3.1.1 Join Game

The Join Game Menu Option represents one of the core game functions. Players select the Join Game option and are led through a series of GUI menus that allow them to enter and play the game. This functionality can be represented in four-state sequence; a Join Server state, and a Playing state, and Idle state, and an Exit state.

In the Join Server state, the user moves through the menu options to select a valid and available server to join and then attempts to join. Background processes and checks execute which validate the player's accessibility constraints, and the player joins the server. Once a player has joined the server, they transition to the playing state.

In the Playing state, the player has performed all the necessary functions and passed through all checks to validate their ability to join a server. In this state, it is assumed the player will continue to play until they choose to exit. In the playing state, the player joins with other players and loops through scenarios of play. When the player exits, they return to the Join Server state, or may exit the Join Game menu returning them to the Main Menu options.

In the Idle state, the player resides on the server list choosing to do nothing. They can transition from the Idle state to the Join Server state, or to the Exit state by interacting with the Server List User Interface. When the player selects the "Join Game" option from the Main Menu, they are brought to the Idle state and the Server List is displayed.

Finally, the Exit state is the functional process of exiting the Join Game GUI menus and returning to the Main Menu options. The Exit state performs any leftover cleanup required to return to the Main menu.

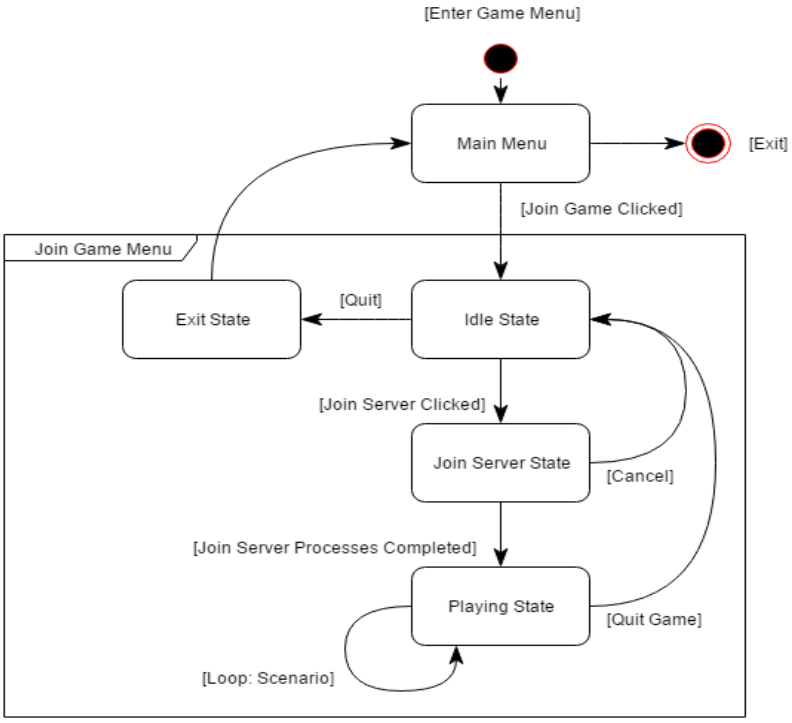

Figure 2 - Join Game State Diagram

2.3.3.1.1.1 Join Game Use-Case

The Join Game Use-Case displays the functional process that is taken by the application to join a Scenario in progress or that is about to begin. Players can access a list of servers and upon selecting a server if there is room they are taken to the Scenario Lobby where they select a character and spawn point and then join the Scenario.

2.3.3.1.1.1.1 User Story

As a Player I would like to have the software keep track of all available servers for me, and not need to maintain a separate list. The game should also track which servers are full and which have space for more players and allow me to search and filter the results. Once the scenario ends I would like to be taken back to the Scenario Lobby and from there able to quit and join a new server, or vote for the next Scenario.

2.3.3.1.1.1.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Play Through a Scenario
Actors	Player, Game, Server, Scenario Lobby, Scenario
Data	The Player selects the *Join Game* option from the main menu. The Game displays a list of available servers and returns the list to the player. Once the Player selects a server the Game checks if the Server is full, if not then the Player joints the Server and is taken to the Scenario Lobby. The Player selects a Character and a Spawn Point and joins the Scenario. Once the Scenario is complete scores and a voting screen for the next Scenario are displayed, or the Player can quit back to the Main Menu and either quit the Game or select a new Server.
Stimulus	The Player selects the *Join Game* option from the Main Menu.
Response	The Game displays a list of Servers and allows the Player to select one. Once a valid Server is selected the player is taken to the Scenario Lobby and allowed to choose a Character and Spawn Point and then enter the Scenario. Once complete the results and voting screens are displayed unless the Player chooses to quit. The Scenario with the most votes is chosen for the next Scenario and launches, taking players back to the Scenario Lobby for the new Scenario. If the Player chooses to Quit they are taken back to the Main Menu.
Comments	The Player is shown an interface that shows the game from various camera views; current scenario; server information; active player info; number of spectators; scenario objectives; player statistics; loot; and experience for each character. There is also a window for the player to chat with other players.

2.3.3.1.1.2 Select Character Use-Case

During the playing state, and during each iteration of character death and scenario start, the player is given the option to select a character to utilize for play. After selecting this option in the user interface screens tied to the state, the player is spawned as this character, and is given control and access to the character's functions.

The associated interface, called the Character Selection Screen, displays the available characters and abilities. Each character's abilities may be modified before entering the game. A selection menu for modifications will accompany each character. When all modifications are complete, the modified character can be saved and enter the game.

2.3.3.1.1.2.1 User Story

As a Player, I would like to choose a character and customize it for the type of gameplay I want at that particular time. This feature should follow the Server selection screen and should be a part of the Scenario Lobby for gameplay.

2.3.3.1.1.2.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Character Selection
Actors	Player, Game, Server, Server Lobby, Scenario Lobby, Scenario
Data	The Player selects the *Character Selection* option from the Scenario Lobby. The Game displays a list of available characters instance, and returns the list to the Player. Once the Player selects a character, the Game displays a list of abilities for that character for the Player to select. Once the Player selects the abilities, the Player can choose to save the character or start over. Once the character is saved, the Player renters the Scenario Lobby.
Stimulus	The Player selects the *Character Selection* option from the scenario lobby menu.
Response	The Game displays the available characters and abilities and displays a management interface to the Player. The Server registers with the Server Lobby and the Scenario Lobby when it the character is created.
Comments	The player is shown the characters from various views, with equipment and items, voices, abilities, and roles. The player must determine the style of gameplay for the scenarios and what particular character fits the style the player wishes to play.

2.3.3.1.2 Host Server

The Host Server Menu Option represents another one of the primary core game functions. Players select the Host Server option and are led through a series of GUI menus that allow them to enter and manage a server hosted from their operating system. The Host Server function allows the user to create a server that will be added to the Server List GUI menu, and players may join then join the server during the Join Game process. The Host Server option creates an instance of a server, which the user may manage.

The user may not both join games and host servers simultaneously in the same executable instance. This is because of a core requisite constraint for processing speed and server reliability. The executable must provide dedicated processes when hosting a server that allow other users to join the server, for the server to process user interactions, and for the server to be accessible. As well, the executable must provide dedicated processes when the player joins a server that allows the player actions to be processed efficiently, and communicated quickly to the server. These conflicting requisites are resolved by denying a player the ability to join a game, and to host a server *in the same executable instance.* A player may open an additional game instance, and simultaneously host or join a game by having parallel executable instances of the complete software.

The Host Server Menu Option can be represented in three separate states; An Initialize Server state, a Manage state, and an Exit state. The Initialize Server state allows the user to enter necessary start-up conditions, and the game executable performs functions to register and start the server. This state transitions to the Manage State, where the server listens for users to join, the User is able to manage the server with administrative options, and game scenarios are executed and communicated to clients. The Manage state subsequently transitions to the Exit state when the user chooses to return to the main menu. This state-process is represented in the following diagram.

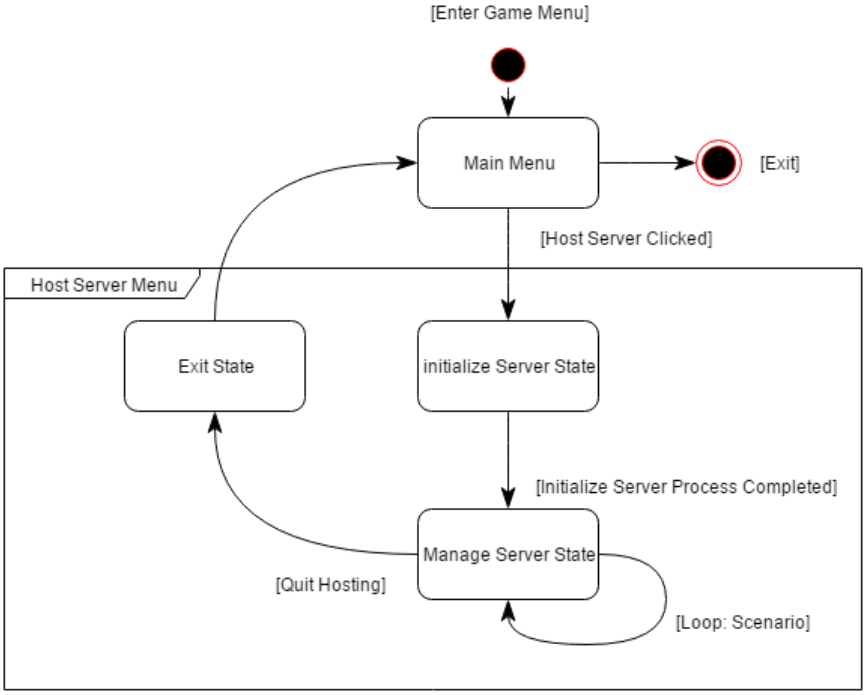

Figure 3 - Host Server State Diagram

2.3.3.1.2.1 Host Server Use-Case

The Host Server Use Case displays the functional process that the application takes to create a server. Servers can be accessed by other players from the Server List menu when they select the Join Game option from the Main Menu.

2.3.3.1.2.1.1 User Story

As a Player, I would like to create, manage, and host a server that should appear on the Server List for players to join. I would like this feature to be part of the game, and I do not want to have separate software to perform this action.

2.3.3.1.2.1.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Host Server
Actors	Player, Game, Server, Server Lobby
Data	The Player selects the *Host Game* option from the main menu. The Game starts a Server instance, and returns the instance to the Player. The Server instance registers itself with the Server Lobby on creation, and it is displayed in the list of servers.
Stimulus	The Player selects the *Host Game* option from the main menu.
Response	The Game creates a Server, and displays a management interface to the Player. The Server registers with the Server Lobby when it is created.
Comments	The player is displayed an interface that shows the game from various selectable camera views; the current scenario; the server type; the number of active players; the number of viewing/watching players; the objectives of the scenario; the kill/death ratios of players, the players' total damage, total health, total healing, character type, objectives achieved, and loot and experience points gained for each character. As well, there is a terminal window for the player to issue commands to the server or chat with other players.

2.3.3.1.3 Manage Characters

The Manage Characters menu option allows the player to access and manage characters tied to their account. In this menu option, players can access the various characters, modify their statistics, purchase new appearance options through the micro-transaction marketplace, and perform any character management options they need to perform between gameplay iterations.

2.3.3.1.3.1 Manage Characters Use-Case

The Manage Characters Use-Case defines the Main Menu option to select and manage a character.

2.3.3.1.3.1.1 User Story

As a Player, I would like the ability to access the characters tied to my account between gameplay iterations. I would like to customize their abilities, visual characteristics, attacks, powers, and rulegoverned statistics.

2.3.3.1.3.1.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Characters
Actors	Player, Game, Server, User Account, Character
Data	The Player selects the *Manage Account* option from the main menu. The Game queries the central Server for the Player's Account. The Character Data for the account is returned to the game with files for modifications to the Characters and account configurations (Loot collected, and other Game-Specific Account Information). The Player accesses menu option to select characters to modify or manage account configurations.
Stimulus	The Player selects the *Manage Account* option from the main menu.
Response	The Central Server validates the account, and sends account information to the game as a series of files. The game interprets the files and displays the data.
Comments	The player is shown a sub-interface where they may select a character to modify, purchase special appearance options, or manage account settings such as purchasing appearance items and the account store of loot. On the character pages, the character's bank of experience points, statistics, and Feat Tree are displayed. Account Configurations, such as password and user e-mail are accessed from a separate Settings menu.

2.3.3.1.4 Manage Settings

The Manage Settings menu option allows the player to control key aspects of the executable software's configurations. The Manage Settings menu provides modification to gameplay configurations, keyboard and control configurations, graphics configurations, audio configurations, account configurations, and software-interface configurations (steam, twitch, and other streaming services).

[Diagram Here]

2.3.3.1.4.1 Manage Settings Use-Case

2.3.3.1.4.1.1 User Story

As a Player, I would like an interface that allows me to access configurations and controls related to gameplay, graphics, audio, account, and software interfaces. I would like the ability to save this information to after it is edited, the ability to revert to a default configuration, and the ability to cancel configuration the current set of changes. The information should be saved with my user account information on a central server so that is consistent between various client instances that I may use.

2.3.3.1.4.1.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Settings
Actors	Player, Game
Data	The Player selects the *Manage Settings* option from the main menu. The Game displays a sub-menu with various options to configure. The player may exit this menu to return to the Main Menu, or they may select one of these configuration options.
Stimulus	The Player selects the *Manage Settings* option from the main menu.
Response	The Game displays a list of configuration menus.
Comments	The Configuration Options are: Gameplay Configurations, Graphics Configurations, Sound and Audio Configurations, User Account Configurations, and Software Interface Configurations.

2.3.3.1.4.2 Manage Gameplay Configurations Use-Case

2.3.3.1.4.2.1 User Story

As a Player, I would like to manage Gameplay configurations related to related to what features are displayed, basic notifications and event monitoring alerts, how user interface data is portrayed and configurations for 508 Compliance.

2.3.3.1.4.2.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Gameplay Configurations
Actors	Player, Game Application
Data	The Player selects the *Settings* from the main menu and then the *Gameplay* option from the settings menu. The Game loads the user interface menu and current gameplay settings for the game.
Stimulus	The Player selects the *Settings* from the main menu and then the *Gameplay* option from the settings menu
Response	If there are changes, the Game updates the current Gameplay settings for the game, and applies the changes immediately.
Comments	The player is displayed a menu that offers them Gameplay configurations which would allow the player to manage configurations related to what features are displayed, basic notifications and event monitoring alerts, how user interface data is portrayed and configurations for 508 Compliance.

2.3.3.1.4.3 Manage Keyboard and Controls Configurations Use-Case

2.3.3.1.4.3.1 User Story

As a Player, I would like to manage Keyboard and Control configurations. I would like to manage what keyboard keys, controllers' buttons, and mouse buttons tie to which functions I have access to during game iterations. I would like this information to be consistent across instances of clients that I may utilize, and the data should be stored with my central account information.

2.3.3.1.4.3.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Keyboard and Controls Configurations
Actors	Player, Game Application
Data	The Player selects the *Settings* from the main menu and then the *Keyboard* option from the settings menu. The Game loads the user interface menu and current Keyboard settings for the game.
Stimulus	The Player selects the *Settings* from the main menu and then the *Keyboard* option from the settings menu
Response	If there are changes, the Game updates the current Keyboard settings for the game, and applies the changes immediately.
Comments	The player is displayed a menu that offers them Keyboard and control configurations which would allow the player to manage what keyboard keys, controllers' buttons, and mouse buttons tie to which functions they have access to during game iterations. The settings would be applied globally to all aspects of the game application.

2.3.3.1.4.4 Manage Graphics Configurations Use-Case

2.3.3.1.4.4.1 User Story

As a Player, I would like to manage Gameplay configurations related to visual settings, sprites, shadows, pixel effects, rendering distance, and various other graphics configurations which would allow me to run the application more efficiently based on my personal client hardware.

2.3.3.1.4.4.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Graphics Configurations
Actors	Player, Game Application
Data	The Player selects the *Settings* from the main menu and then the *Graphics* option from the settings menu. The Game loads the user interface menu and current Graphics settings for the game.
Stimulus	The Player selects the *Settings* from the main menu and then the *Graphics* option from the settings menu
Response	If there are changes, the Game updates the current Graphics settings for the game, and applies the changes immediately.
Comments	The player is displayed a menu that offers them Gameplay configurations related to visual settings, sprites, shadows, pixel effects, rendering distance, and other various graphic configurations that will allow them to run the application more efficiently based on their personal hardware.

2.3.3.1.4.5 Manage Audio Configurations Use-Case

2.3.3.1.4.5.1 User Story

As a Player, I would like to manage Audio configurations, related to sound, echo, vibration, music and soundtrack, voice, and audio effects configurations which would allow me to run the application more efficiently based on my personal client hardware.

2.3.3.1.4.5.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Audio Settings
Actors	Player, Game, User Account, Central Server
Data	The Player selects the *Audio Settings* option from the *Settings* menu, the Game displays the various Audio settings and allows the Player to adjust them. The game saves the results to the configuration file which is then uploaded to the Central Server and stored in the User Account information.
Stimulus	The Player selects the *Audio Settings* option in the *Settings* menu.
Response	The Game displays the Audio settings to the Player and allows them to be adjusted, then saves the results back into the configuration file.
Comments	The Player is shown a menu that has the setting names and current values for things like sound, echo, vibration, music, voice, and sound effects which allow them to be tweaked to run more efficiently on a Player's system or to align with Player preferences.

2.3.3.1.4.6 Manage Account Configurations Use-Case

2.3.3.1.4.6.1 User Story

As a Player, I would like an interface to effectively manage Account configurations related to account and password security, account identity, account permissions, and purchase information (Credit Card) for the micro-transaction market. These configurations should be stored with my central user account information and be consistent across instances of the instances of the application I use.

2.3.3.1.4.6.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Account Settings
Actors	Player, Game, Server, User Account
Data	The Player selects the *Manage Account Settings* option from the *Manage Settings* menu. The game queries the main Server for account data. The server validates the account, and returns a set of configurations that can be modified. The game displays the values of these configurations in the page.
Stimulus	The Player selects the *Manage Account Settings* option from the main menu.
Response	The Game requests the Configurations from the Server. The server processes the request and returns the configuration data for the game. The Game displays the data.
Comments	The Configuration Options consist of the following: Account Name, Account Password, Account E-Mail, Contact Account Address and Information, Account Debit/Credit Card Information, Account Billing Information, Account Multi-Token Dongle Verification

2.3.3.1.4.7 Manage Software Interface Configurations Use-Case

2.3.3.1.4.7.1 User Story

As a Player, I would like an interface to manage and control how external software interacts with the PegLeg Pirates software. This interface should allow me to control configurations per applications relevant and commonly used in conjunction with the software such as Steam, Twitch, Screenshot and Video recording software, and other branded streaming software. These configurations should be stored with my central user account information and be consistent across instances of the instances of the application I use.

2.3.3.1.4.7.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Manage Software Interface
Actors	Player, Game, User Account, Central Server
Data	The Player selects the *Interface* option from the *Settings* menu. The Game allows the Player to view and edit various settings related to how third party applications such as Streaming or VoIP clients interact with the game, the changes are then stored in the configuration file and uploaded to the player account on the central server.
Stimulus	The Player selects the *Software Interface* option in the *Settings* menu.
Response	The Game displays the Interface settings to the Player and allows them to be adjusted, then saves the results back into the configuration file.
Comments	The Player is shown a menu that allows them to change how the Game interfaces with third party applications like named streaming services or VoIP clients, as well as where the default save location for screenshots etc. is.

2.3.3.1.5 Exit Game

The Exit Game menu option provides processing to safely close and exit the game software. The Exit Game option performs processes and functions to perform any needed garbage collection, perform any necessary saving of account data, and logs the user out of the central account server. Any final reliability or resilience actions required to secure the user account from vulnerabilities are performed. The game completes these processes and terminates, returning the user to their desktop or base operating environment.

2.3.3.1.5.1 Exit Game Use-Case

2.3.3.1.5.1.1 User Story

As a Player, I would like a menu option to exit the game safely, securely, and completely without leaving residual game processes and functions on my system. This menu option should terminate all running processes of the game, exit the game process, and return me to my system's desktop or central user interface.

2.3.3.1.5.1.2 Tabulated Use-Case Diagram

System	PegLeg Pirates Game
Use Case	Exit Game
Actors	Player, Game
Data	The Player selects the *Exit* option from the *Main Menue.* The Software shuts down and safely terminates the executable.
Stimulus	The Player selects the Exit option in the in *Main Menu*.
Response	The Game saves any modified information, and closes to the desktop.
Comments	This function is separate from a Player exiting a Server, Scenario, or other menus which contain the option to Exit. This case only occurs from the Main Menu.

2.3.3.2 Controls

This game is focused around combat between players, specifically in an FPS template. In that template, players fight in an FPS style with a first person perspective though they may scroll out to third person perspective. They control their character directly by pressing keys to issue a response. The mouse is often used for controlling the focus point to make attacks and defend against attacks.

Players control their character using the typical WASD movement keys, and they have access to 5 special abilities tied to the 1, 2, 3, 4, and 5 keys. They may also use items with Q, and trigger their special maneuver with E.

2.3.3.3 Attributes

All characters have biological statistics, called Attributes. Attributes are the finite point values that determine how strong, how fast, how tough, how accurate, or how resilient a character is in combat.

There is a set of six primary attributes which the player may upgrade with experience or by purchasing gear using loot. These attributes are listed in the table below.

Name	Color	Function
Resilience	Orange	How much damage is resisted by a character.
Power	Yellow	How much damage is performed by a character.
Fortitude	Red	How much physical health a character has.
Agility	Blue	How fast a character is.
Stamina	Green	How much energy a character has to perform physical actions.
Intellect	White	How much energy a character has to perform mental actions.
Willpower	Purple	How quickly a special move is generated.

Table 4 - Character Attributes

Intellect and **Stamina** are two point pools that are used to perform abilities, or special actions. As characters perform special actions related to their character, they generate special move points in increments based on their **Willpower**. Melee and ranged combat attacks generally utilize **Stamina** whereas mental and social abilities utilize **Intellect**.

When a player damages another player, the damage is determined from the character's **Power**. Attacks to a specific amount of damage determined by the ability, and then the damage is modified as a function of the character's **Power** score. The speed at which the attack is made is determined from the **Agility** attribute. **Agility** also determines how long the refresh time on a maneuver is, and how quickly a character moves.

When a player receives damage, the damage is reduced by the **Resilience** score. **Resilience** reflects any armor they may be wearing, or how physically tough a character is to damage. Any damage that is not ignored or reduced by **Resilience** is removed from the health pool, which is determined from the **Fortitude** score.

2.3.3.4 Death and Survival

In this game, characters are considered dead when their health bar, determined by their fortitude, reaches 0. First, they **pass out** and require aid from a healer or a player with a medical kit. In this stage,

they may not move, and their character is lying on the ground. They will die within 30 seconds, and this speeds up for any extra damage they take.

Next, they die. When the player dies, they may select a new character and respawn or go into observation mode and leave the match to watch.

2.3.3.5 Abilities

Abilities are special moves that a player may use with their character. Each character has a basic melee or ranged attack, which are used by clicking the LMB. The character may also defend by holding the RMB. Players can swap between melee and ranged attacks with the middle mouse wheel. These actions are separate from **Abilities**.

Abilities are special functions which a player can "slot" into their 5 special hot keys. Each of these hotkeys can be triggered by pressing the key down. Each **Ability** has a cool down which determines how quickly the **Ability** can be reused.

Each character is given a basic set of **Abilities** related to the character design and function. As well, each character has a tree a feats they may unlock in a stylized sequence that improves these abilities; or they may "evolve" an ability to act differently. For example, Captain No-Bear can use cannon grenades that perform a flash-bang effect. This effect can be "evolved" to instead perform a snare on enemies, or can perform a concussion effect to knock enemy players down. Standard abilities for a character, and their ability-feat tree is described in the Character section.

2.3.3.6 Feat Trees

A **Feat Tree** is a special menu that represents passive capabilities of a character. Players use their experience to unlock passive abilities on the **Feat Tree**, and these passive abilities are applied to the character.

Each character's **Feat Tree** is unique to the character. While Voodoo Jim and Scallywag Sue may have different feat trees, all Voodoo Jims in existence have the same **Feat Tree**. **Feat Trees** are usually very specific, and choosing between options on the feat tree will limit the ability to select other options in the **Feat Tree**. For more information, you can view a character's **Feat Tree** in the character section.

2.3.3.7 Experience, Leveling, and Growth

As players compete in scenarios with a character, they also obtain **Experience**. They obtain **Experience** for completing objectives, killing enemy players, and winning a match. Players use this **Experience** to increase the **Attributes** of a character, improve their **Abilities** through the **Feat Tree**, or to unlock new **Ability** functions by evolving the **Ability**.

Players bank **Experience Points** with a character from match-to-match, and level up in set **Experience Point** increments. Once a player has leveled up, their total is rolled over to 0, and any excess point are added to the completion of the next level. When a player receives a level, they earn 1 point, which can be spent to increase an attribute, on abilities in the ability system, or on a passive ability in the Feat Tree.

A player only receives end-of-match experience with the character they were using at the end of the scenario, even though they may freely switch between characters during a match. Experience points gained from killing enemy players or completing objectives are given to the character immediately, and

the player does not need to wait until the end of the scenario to receive Experience Points for these actions.

2.4 User Characteristics

This game is targeted directly towards young adults ranging from 13 - 30 years of age. This age group typically is more geared towards desktop gaming and is familiar with using computers, as well as keyboard and mouse, to operate video games. Also, some of the humor, gameplay, and scenery could involve minor cases of violence which would be appropriate for this age range or higher.

2.5 Constraints

The constraints of this product are defined by compiler and output limitations of the Software Framework. The Target Platform for this product is as follows. It shall be noted that, while these are the minimum requirements to operate the software product, higher quality software and hardware are suggested for an optimal user experience.

Software Product Minimum Requirements	
Minimum Operating System Version	Windows XP SP2 +, MAC OS X 10.8+, Ubuntu 12.04+ (Possible other Linux distros but will not support and cannot guarantee functionality)
Minimum System Network Requirements	10Mbps internet connection
Minimum System Storage Requirements	20GB HDD available space
Minimum RAM Requirements	8GB RAM
Minimum Graphics Card Requirements	Graphics Card with support for DirectX 9.0 or higher or OpenGL
Minimum CPU Requirements	CPU supporting SSE2 instruction set

Table 5 -Minimum System Requirements

2.6 Assumptions and Dependencies

In order to purchase and operate our game, we assume that the individual who purchased the game either owns one of the desktop deployment platforms (Windows, Mac OS, or Linux), or is giving the game to an individual who does.

In order to operate, and play the game to its full capabilities, the user must have a stable internet connection, as well as suitable hardware and software capabilities. These hardware and software capabilities are defined in section 2.5.

3 Specific Requirements

3.1 Functional Requirements

- The software shall be playable on Linux, Mac, and Windows Operating Systems Versions supported by the build component of the Unity3D engine.
- The software shall have the capability to save the user's game file.
- The software shall have the capability to load a user's saved game file.
- The software shall have a GUI control system with user capabilities and functions easily identifiable by the user.
- The software shall receive input from a keyboard and mouse as the default input components.

3.2 Non-Functional Requirements

- The software shall be developed through the Unity3D Engine.
- The software code shall be stored and managed through the GitHub version control system.
- The software code shall be retrieved by project members through Atlassian software.
- The documentation for the software shall be stored and accessed on Google Documents.
- The 3D model art assets for the software shall be developed through the Blender software.

4 Index

The Index provides a reference to all tables and figures in the document.

4.1 Tables

4.2 Figures