Software Requirements Specification

for

Walker RPG

Version 1.0

Cameron Fullerton, Andrew Buikema

Phillip Buckreis, Peter Hansen

CAPP

10/16/17

[**Introduction**](#_x9p6eqnrjxh6) **4**

[Purpose](#_m3bjk9736i) 4

[Document Conventions](#_x6hagja674qq) 4

[Intended Audience and Reading Suggestions](#_mafx3fhx26wa) 5

[Project Scope](#_24zcnfiuk54c) 5

[References](#_elgyhfqzuymo) 5

[**Overall Description**](#_ec4y4gl56ggd) **5**

[Product Perspective](#_ft1wk1sfp7zj) 5

[Product Features](#_axh7tmg7h4k1) 6

[User Classes and Characteristics](#_pfpc15at0v1p) 6

[Operating Environment](#_r9shde0yzpq) 8

[Design and Implementation Constraints](#_83qsn8oaeax1) 8

[User Documentation](#_cm9c50j6ydk0) 8

[Assumptions and Dependencies](#_3dgn6f2nlsh1) 8

[**System Features**](#_g31u97l4bkh3) **8**

[Save File System](#_89t8oelpfzcx) 8

[Description and Priority](#_2dffqroha0jd) 8

[Stimulus/Response Sequences](#_cgrqjt2db0k8) 9

[Functional Requirements](#_uccgod9h3elp) 9

[Inventory System](#_lq2jkz2ediwr) 9

[Description and Priority](#_s6vv5tcaxhnt) 9

[Stimulus/Response Sequences](#_ik3mb3qu5rj9) 9

[Functional Requirements](#_mhyscxglqaay) 9

[Pet System](#_fuo73n15eofr) 10

[Description and Priority](#_7uiqvj4wxpk) 10

[Stimulus/Response Sequences](#_36mvxgkvmvv) 10

[Functional Requirements](#_8pb09ynau6yh) 10

[Zone System](#_yv2jcujo44qi) 10

[Description and Priority](#_apfkmgfwp5u3) 10

[Stimulus/Response Sequences](#_gw4o4vgeeye4) 11

[Functional Requirements](#_85inm7lcfxqb) 11

[Equipment System](#_8xew1cpbq7lv) 11

[Description and Priority](#_nv7sv1nx4ndv) 11

[Stimulus/Response Sequences](#_xlj8kax30brf) 11

[Functional Requirements](#_y4d871ht1qit) 11

[Loot Box System](#_n6yanr2q26xy) 11

[Description and Priority](#_zfm2bsfsltvi) 11

[Stimulus/Response Sequences](#_bpdblpbo1cjy) 12

[Functional Requirements](#_umxjifsaygti) 12

[Step Counter System](#_sk7zrwd7osfv) 12

[Description and Priority](#_7nvk2hd807wd) 12

[Stimulus/Response Sequences](#_v566x1j76doa) 12

[Functional Requirements](#_1jxq5ygs3gfw) 13

[Random Encounter System](#_pjeglluiv9h5) 13

[Description and Priority](#_kydiqjkrqrpp) 13

[Stimulus/Response Sequences](#_go4abtpp53xl) 13

[Functional Requirements](#_30cngj85mwaw) 13

[Battle System](#_71wwaeq1yz3x) 13

[Description and Priority](#_4dxc3p4pdar) 13

[Stimulus/Response Sequences](#_o6xj39aght2q) 14

[Functional Requirements](#_a9ol8w24rzlw) 14

[**External Interface Requirements**](#_qb6k4lzd74mh) **16**

[Use Interfaces](#_jodrt6epr6ho) 16

[Hardware Interfaces](#_nzzpx0si6wyx) 17

[Software Interface](#_8c3m7u789q2x) 17

[**Other Nonfunctional Requirements**](#_vv5v13ga9sjb) **17**

[Performance Requirements](#_tg4on8tly7k6) 17

[Safety Requirements](#_dgps2olkjvsz) 17

[Software Quality Attributes](#_xrzcv2oybh7m) 17

[**Project Management**](#_6igdncs720up) **18**

[Timeline](#_astqxxl31vgt) 18

[Gantt Chart](#_rplqxsi66eck) 19

[**Test Plan**](#_b9beba7a07ia) **19**

[Developer Mode](#_4zdny8fm09s5) 19

[Tester Mode](#_aqlc0c4rrxug) 20

[User Mode](#_i95xqkk7zd13) 20

[**Risk Management**](#_51ymwdjl31e) **20**

[**Meeting Minutes**](#_aipjkyblldhz) **21**

[**Contribution Table**](#_w5rpurji50pk) **21**

# Introduction

## Purpose

This document details the functionality required for a new game called Walker RPG.

## Document Conventions

This document follows a standard report convention, including a table of contents and headings for primary sections (eg. Introduction, Software Requirements, etc), and secondary headers (eg. Purpose, Document Conventions, Intended Audience, etc).

Primary headings are bolded in the Table of Contents to denote major sections of the report, and secondary sections below them are assumed to inherit the same priority level.

## Intended Audience and Reading Suggestions

This document is intended for developers, project managers, users, and testers of the Walker RPG and provides information on the foundation of the game, descriptions of the general gameplay aspects, and insight into the design and implementation of key features.

The rest of this SRS details the scope of the project and which users the game is intended for, as well as an explanation of the various features to be included and assumptions made in development. Additionally, requirements and constraints are listed for project managers to further understand the design scope and create a reasonable development timeline with attainable goals.

This document is intended to be read in order, though not all sections are necessarily important for all readers. For example, users would benefit from reading the introduction, followed by the Overall Description and System Features sections, while developers might read the Introduction and then the External and Nonfunctional Requirements sections.

## Project Scope

Walker RPG is a game designed to promote fitness for mobile device users. It is an RPG game for users to play on their mobile phones, and encourages productive fitness routines such as walking or jogging. It is intended to provide a creative motivation for users who do not regularly engage in fitness activities, and to encourage those who already do to diversify and expand upon their current activities.

## References

This document references Use Case and Class diagrams to illustrate a high level, overarching view of the game functions. Additionally, Step-Action and Battle Sequence diagrams are referenced to further illustrate the basic game mechanics.

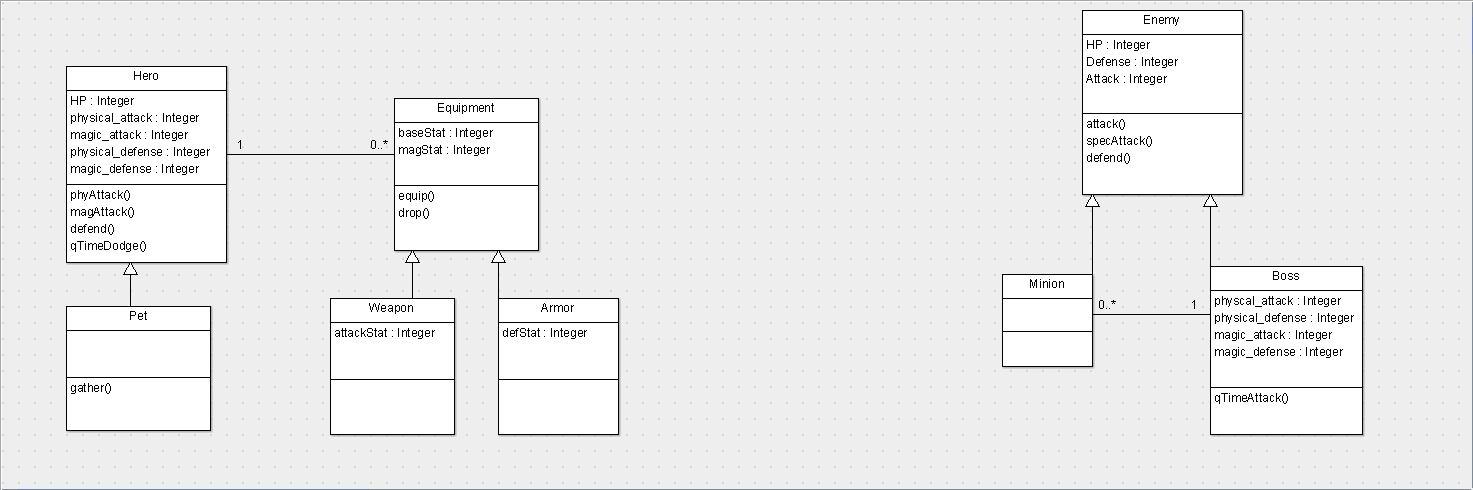
# Overall Description

## Product Perspective

Walker RPG is a new progress based RPG made for people who enjoy fitness. It will have a turn based combat system in which the player and enemies take turns attacking and blocking to defeat each other. Progress through the game will be made by walking through paths and encountering enemies. To walk through paths in the game, a player will walk in real life and the steps taken will be transferred into the game to progress through the path. Systems within the game will consist of the combat system, walk tracking system, and a player interface system.

## Product Features

The goal of Walker RPG is to try to bring fitness into gaming. The player will walk around in real life to move their avatar in the game. This will be accomplished through the step tracking systems on most smartphones. The player will progress through designated paths while also fighting enemies along the way. These enemies will be randomly encountered through a pathway. At the end of each pathway, the player will encounter a boss that will be more powerful than the normal enemies. These bosses will show some mechanics of the previous enemies fought. There will be an inventory system for the player to carry loot gathered along their travels, along with an armor system for equipping new pieces of armor gathered.



**Figure 1: Class Diagram**

## User Classes and Characteristics

Player

Access to their inventory and gear system

Access to paths unlocked.

Able to use steps accumulated while walking to progress through paths.

Able to fight enemies and bosses using skills and attacks unlocked at that time.

Must go through first tutorial as introduction to the game.

Developer

Can access all zones and pathways throughout the game.

Access to inventory and gear acquired.

Access to all skills and attacks.

Options to edit current stats.

Options to generate enemies.

Able to kill enemies instantly.

Options to edit enemy stats.

Infinite steps accumulated.

Can turn off tutorials at any time

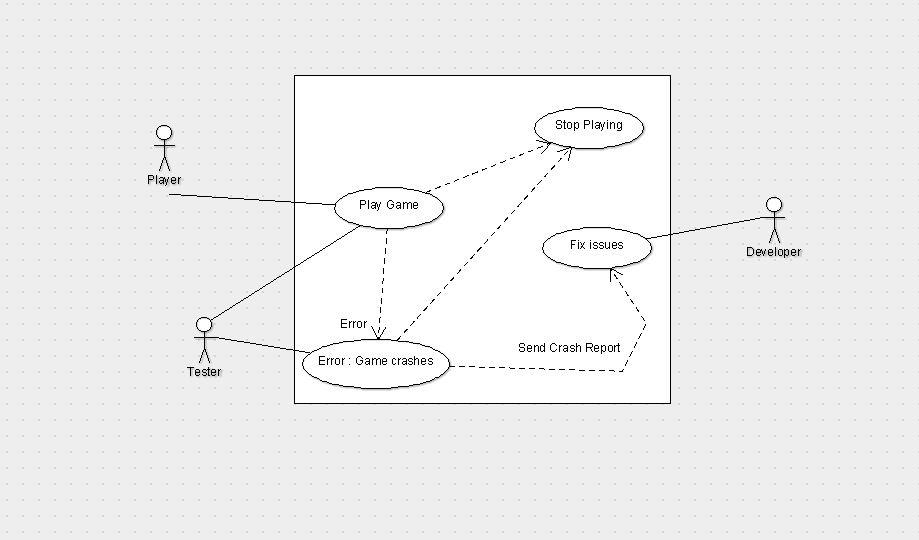
Tester

Access to quick progress system, increasing amount of steps for quicker movement throughout game.

Access to Inventory and gear system loot table to choose gear and equip.

Access to paths unlocked.

Access to skills at earlier levels.



**Figure 2: Use Case Diagram**

## Operating Environment

Walker RPG will run on Android phones as an application, supported on all phones running version 5.0 and newer of the Android mobile operating system. Eventually, additional versions of the game may be developed to run on wearable devices, such as FitBits and Apple Watches, but Walker RPG will initially be released as an Android specific game.

## Design and Implementation Constraints

Implementation constraints on development will primarily be due to hardware limitations. For example, not all developers may have access to modern Android smartphones running the newest versions of Android OS, or with enough memory to support the game. Additionally, security considerations will be a factor when designing the game and may prohibit or restrict the implementation of certain features (for example, security features designed to prevent the unregulated manipulation of a player’s health may slow the process of legally increasing their health (through potions or leveling up) due to the security check and verification process.

## User Documentation

A user tutorial will be given when the game is loaded for the first time, and as new skills and features are unlocked. The tutorial will explain how the combat system works, along with receiving loot boxes and the gear system including inventory.

## Assumptions and Dependencies

It is being assumed that we can access the natural step counter functions in the Android operating system to track the movement of player to translate to the movement available to avatars.

We are depending on using unity to program this game as it has functions and libraries useful in game programming and allows integration of the Android OS functions we plan on using.

# System Features

## Save File System

### Description and Priority

The save file system allows players to save their gameplay data on their device so they can return to the game without losing progress. This is a high priority feature. The benefit of a save system is that it allows the player to turn the game off and then back on and resume playing where they left off. The penalty of not having a save system is that every time the game is closed, the player will have to start from the beginning of the game. The major risk involved is security. The save files are saved locally in the device, which means players can edit their save files to cheat in the game. The cost is low. Implementing a save file system is easy and it doesn’t require much processing power when the player encounters a save.

### Stimulus/Response Sequences

The game will automatically save every 30 seconds the application is open. Alternatively, players can enter an options menu and manually save the game to ensure no progress is lost. If a save file is not found or if a save file is found to be corrupted, a new save file will be created when the application is opened.

### Functional Requirements

The save file system will save steps taken, currency owned, zones unlocked, paths unlocked, current zone, current path, equipment held, pets owned, pet progress, time of save and the operating system’s step counter count. When a player opens the game, the save file system will load the zone and path they were in with the appropriate amount of steps and currency. The game will also check to see how much time has passed since the game was last saved to the current time and rewards players with currency their pets picked up while they were away. During the loading phase of the save file system, we also want to check the operating system’s step counter to see if there is a difference between the amount of steps saved and current amount of steps and add that to the current path.

## Inventory System

### Description and Priority

The inventory system will allow players to collect items along the adventure. As well as sell, equip, drop and swap items. The inventory system will be limited to 20 items. The priority of the inventory system is high. The benefit of an inventory system is that it allows the player to choose items they want to equip and store their favorite items for later use. The penalty of not having an inventory system is that players won’t have the ability to collect items outside of what the player is wearing. The risk of an inventory system is low, but the cost is high. Getting a functioning inventory system is going to take a while.

### Stimulus/Response Sequences

The game will automatically save every 30 seconds the application is open. Alternatively, players can enter an options menu and manually save the game to ensure no progress is lost.

### Functional Requirements

The inventory system will keep track of the player’s weapons, abilities, and other items found. These items can be equipped. The player can freely drop or sell items. The game will keep track of the amount of steps saved and current amount of steps walked and add that to the current path.

## Pet System

### Description and Priority

The pet system allows players to collect pets during the adventure. Pets would be on a timer and once the timer is completed, the player will be rewarded with some currency and the pet will be rewarded with experience. The speed of the timer is dependent on the pet’s level and difficulty of the zone. More difficult zones reward players with more currency and pets with more experience. The priority of the pet system is medium. The benefit of the pet system allows players to receive currency and progress in the game while they aren’t actively playing. Players are bound to get stuck in their adventure and pets give the player currency to allow them to overcome their challenges. The penalty of not having this feature is that players may get stuck and not wish to continue because they can’t get enough currency to buy new equipment. The risks of the pet system is relatively low with only difficulty of implementing the system and balancing the system so pets don’t give too much currency. The cost of the pet system is going to be pretty high. It’s going to be difficult to implement and it depends on many other systems.

### Stimulus/Response Sequences

Player’s pets are acquired either by purchasing them, acquiring them through the story, or randomly acquiring them through random encounters. To access the player owned pets, they’ll need select the “Pets” button at the home screen. In the “Pets” screen, players can see the progress of their pets, what zone their adventuring in, what level they are and how much currency and experience they’re receiving. Players can also call pets back and send them to new zones.

### Functional Requirements

The most important feature of the pet system is that pets give the player currency over time. The purpose of the pets are to let the player gain currency while they aren’t playing to keep pushing them forward in the game. Allowing your pets to go to different zones and how pets are acquired are a secondary requirement.

## Zone System

### Description and Priority

The level system is broken up by two major components: zones and paths. A zone is a unique area that the player can travel, fight unique monsters and bosses. Every time a player completes a path, they have the option of fighting a miniboss. After defeating a mini-boss, the player will be rewarded with a loot box. If the player loses to the mini-boss, they are given the option to either retry to defeat the boss, or return to the path to earn stronger equipment and retry the boss at the player’s leisure. After completing five paths in a zone, the next zone unlocks. Each zone has an unlimited amount of paths, each path getting more and more difficult but rewards becoming more abundant. The zone system is vital to our game, it is a high priority. It challenges and rewards players for completing paths. If we aren’t able to get a zone system completed, the variety of enemies and bosses might bore the player. The most difficult thing about implementing the system is balancing it out so that the player will always feel rewarded for completing a path. Since this is a main feature of our game, it will require a lot of work for our group.

### Stimulus/Response Sequences

Zone 1 - Path 1 is unlocked at the beginning of the game. To go to Zone 1 - Path 2, the player must walk a specific amount of steps dependent on the zone and path. To unlock additional zones, players must defeat the boss after five completed zones. The player can choose to go to any unlocked zone through a zone selector screen. Tapping on an unlocked zone in the zone select screen returns the player to the last path they were at in the zone.

### Functional Requirements

The Zone System requires, at a minimum, the ability to go to new zones and to complete paths. Bosses aren’t required, but adds variety in the enemy encounters.

## Equipment System

### Description and Priority

The equipment system allows the player to equip items they have. A player can equip up to three different weapons, three different skills and three different pieces of armor. The purpose of the player being able to equip three different skills and weapons is to give the player variety in battle. The equipment system is vital for our project and is rated high priority. Without it, players won’t have any customization of their character’s attacks and battles will be predictable. There is a risk that there isn’t enough variety in equipment and we may not finish the system.

### Stimulus/Response Sequences

The player can go to an equipment screen by pressing “Equipment” in the home screen. The player than can choose which piece of equipment they would like to equip, remove, or swap.

### Functional Requirements

Players must be able to equip and use multiple weapons and armors. They can freely exchange their equipment for anything in their inventory.

## Loot Box System

### Description and Priority

The loot box system generates three random pieces of equipment or skills for the player to add to the player’s inventory. Loot boxes are granted at the end of every boss fight and can be bought with in-game currency. The cost of the loot box is dependent on how strong the loot is that is dropped from the box. The loot’s strength is dependent on the loot the player has equipped and in their inventory. The loot box will check for the player’s three strongest weapons and three armors and give a 50% chance of generating stronger equipment. The loot box system is a requirement for our game. If there is no loot box system, then our game would need to feature another way for players to gain equipment and skills. The risk is properly balancing the drops for the player so that they are casually growing in power, but not at a rate that’s too quick.

### Stimulus/Response Sequences

Loot boxes can be access either by buying them from the shop or after a boss fight. In the shop, players will tap a button to open a loot box for a certain amount of currency. After a successful boss fight, players will tap a treasure box to open a loot box. The loot is then generated and players have the option of accepting or trashing pieces from the box.

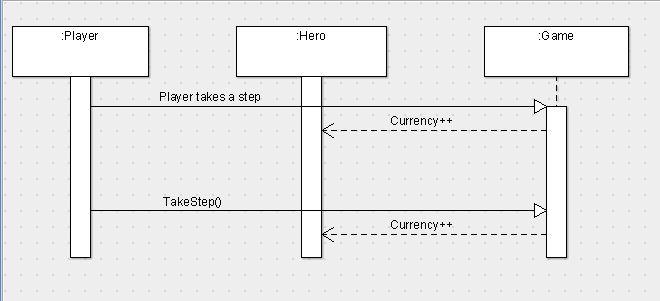
### Functional Requirements

The loot box must drop multiple pieces of equipment or skills. The equipment’s power must be based off of the player’s other equipment’ power.

## Step Counter System

### Description and Priority

The step counter system will allow the game to access an android phone's step tracker and import the data into the game to convert the moved distance since the last update into currency and movement for the player's avatar. The in game steps will be produced from the conversion and can then be spent on moving aa avatar down a pathway to try to encounter enemies and bosses. This system is how the character in game moves and therefore has a high priority as without this the player cannot move his avatar or progress in any way.



**Figure 3: Game Movement Sequence Diagram**

### Stimulus/Response Sequences

Stimulus for this function would be upon opening the game and if the step detector in the Android OS is activated. The response would be to load the steps taken, subtract the last number pulled from this function and add the remaining to the currency and steps in game.

### Functional Requirements

This must load every time the player loads the game and perform a real time update on the step counter and currency in game so the player can see what he has available.

## Random Encounter System

### Description and Priority

The random encounter system is what initiates fights in Walker RPG. Every step the player takes, there is a 2% chance the player will trigger a random encounter. The encounter is typically an enemy fight, but could occasionally be a surprise event. The game will actively count steps when the game is open, and if the player turns their phone off, the amount of steps they have taken while the game is closed will also be counted. When they player turns the game back on we will count the steps they have taken and run it through a formula to determine if they encountered something. Only a single encounter can occur when the phone is off. This is a primary feature of our game and has a high priority. This system is what drives the battle system and it’s important for the game’s flow. Without a random encounter system, players would either manually tell the game when they want to attack. Risks that can occur is an encounter rate that is either too high or too low. We don’t want players to feel overwhelmed if they get an encounter every step or get too bored when they go a whole day without a battle.

### Stimulus/Response Sequences

A random encounter chance is triggered at every step or whenever the player turns the game on and the Android’s step counter is different than the value we saved in our save file system. While the game is active, the step rate encounter is 2% per step. While the game is off, the steps are placed in a logarithmic function that simulates a 2% per step model. If encounter system triggers, then the game will go to the battle system.

### Functional Requirements

The only requirements for the random encounter system is that it must trigger and it must call the battle system.

## Battle System

### Description and Priority

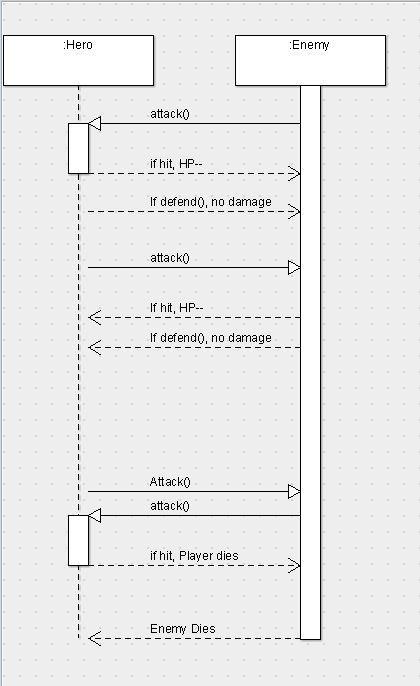
The battle system is turn-based combat. The player and enemy both have a set amount of ability points. Each action, such as swinging a weapon, casting a spell or defending an attack, costs ability points. The amount of ability points an action costs depends on the equipment or skill. Stronger weapons or skills take up more action points than weaker weapons or skills. At the beginning of the battle, players attack first. If the enemy’s health reaches zero, then the player won the battle and earns currency. If the player’s health reaches zero, then the player lost the battle and is returned to the home screen with no reward. This feature is a primary feature of our game. It has high priority. The battle system is the driving force of Walker RPG. Without it, there is little reason to walk or continue playing the game.

### Stimulus/Response Sequences

The battle system is triggered in three ways. The first way is by the random encounter system while the game is active. The second way is when the player turns on the application and the random encounter checker passes the steps taken while away into a logarithmic function to simulate the probability that the player would run into an encounter. The third way is if the player completes a path, the player can press a button to fight a boss.

### Functional Requirements

The battle system needs to be turn based and allow the player to use their own weapons and skills.



# External Interface Requirements

## User Interfaces

When the game starts up it will load into a splash screen displaying our team name, CAPP, and the unity logo, as we are using unity to make the game. This will then automatically move into the start screen.

The Start screen will show a picture and have the words “tap to start” on it, upon tapping the screen, the screen will change to the to the menu screen.

The menu screen will display the user’s character in the background with screen transition buttons consisting of: Character, pet, inventory, battle, map, and stat screens.

Upon touching the Character screen, the screen will change to a view of the users character and items currently equipped. From this screen in the top right hand corner, there will be a “back to menu” button to transfer back to the menu screen. Also on this screen will be a button displaying “Inventory”

Upon touching the inventory screen the user’s inventory will appear displaying the user’s collected items. From this screen in the top right hand corner, there will be a “back to character” button to transfer back to the character screen.

Upon touching the pet button, the screen will transition to a screen showing a list of companion pets found by the user. This screen will also have the ability to send pets on side mission to complete for rewards. From this screen in the top right hand corner, there will be a “back to menu” button to transfer back to the menu screen.

Upon touching the map button the screen will transition to a screen displaying the user’s current zone and path, how far through the path they are and the distance left to travel. This screen will also have a button to display a larger world map to view all paths in the zone and an even larger view of all zones. From this screen in the top right hand corner, there will be a “back to menu” button to transfer back to the menu screen.

Upon touching the Stats button, the screen will change to a display the user’s stats including distance traveled, currency owned, steps taken total, steps take on path. From this screen in the top right hand corner, there will be a “back to menu” button to transfer back to the menu screen.

Upon ending a path, a “Battle” button will appear on the menu screen. Touching the “Battle” button the screen will transition to a battle screen consisting of attack buttons, defense button and a “forfeit” button. Upon hitting the forfeit button the user will go back to the menu screen, and the “battle” button will still appear on the menu screen until the battle is completed.

## Hardware Interfaces

The player will be able to interact with the display screen for displaying the game, the speakers for playing sound, the touch screen for input, and the pedometer for step count. The game will be made for mobile Android devices.

## Software Interface

Walker RPG will be using Android’s built-in pedometer so we can keep track of step count, as well as actively know when the player is walking. The pedometer uses the mobile devices’ accelerometers to detect whether the player is walking or not and be able to count every step the player takes.

# Other Nonfunctional Requirements

## Performance Requirements

The game should run at 60FPS. Audio should be synced with video. Pedometer needs to be accurate. Having 60FPS image helps the animation and flow of the game look smoother than less frames per second. If the pedometer is not accurate, the player may walk much further than intended, leading them to stop playing the game. The game will support multiple screen sizes and resolutions.

## Safety Requirements

While playing the game, be aware of your surroundings and obey local laws. Do not play while driving.

## Software Quality Attributes

The game should work when suspended and unsuspended. While suspended, the game will be paused until the game is suspended. When the game is suspended, any steps taken while the game was suspended will be counted.

The game should work on multiple resolutions and screen sizes. Android devices come in many shapes and sizes and Walker RPG should work on all of them.

The home screen should be one or two button presses away from every other screen. This is so players feel grounded and don’t get lost in the application.

# Project Management

## Timeline

9/13 - Deliverable 1

9/14 - Project proposal Presentation

9/23 - Step Counter, Random Encounter, Sample UI, Stats Page

9/30 - Randomized Weapons

10/7 - Inventory System

10/10 - Deliverable 2

10/14 - Loot Box System

10/28 - Battle System, Music/Sound

11/04 - Enemy AI

11/13 - Code inspection

11/18 - Graphics

11/27 - Code completion

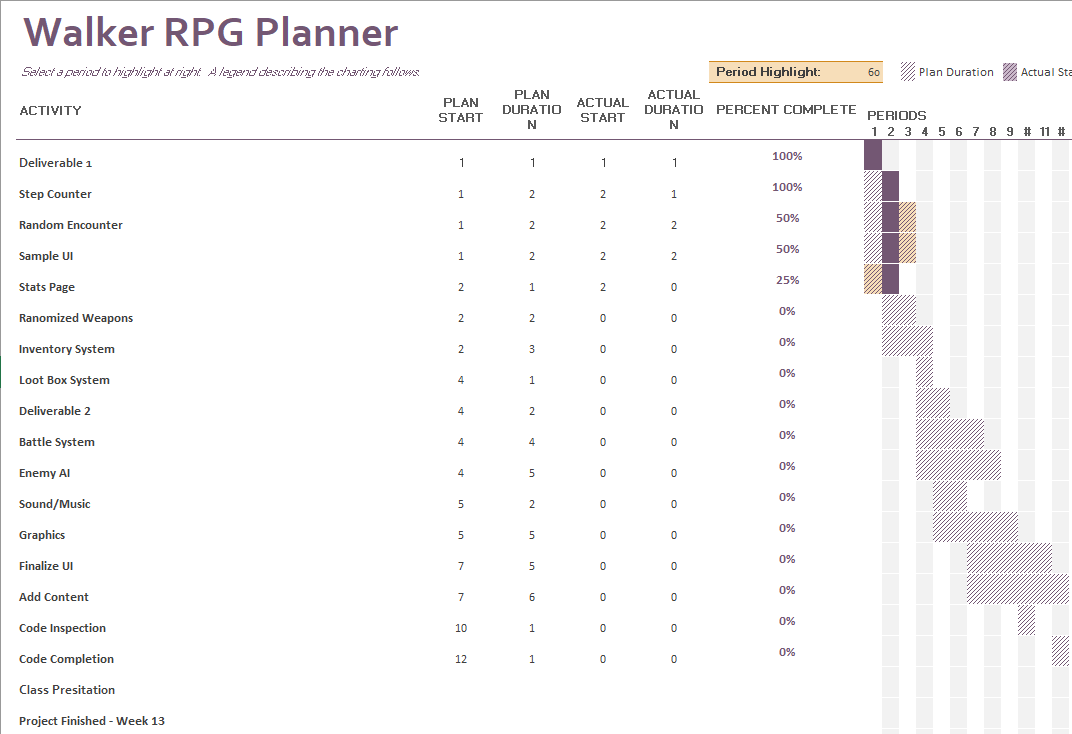
11/27 - Final report

11/28 - Class demo and presentation

12/2 - Finalize UI, Added Content

12/5 - in class usability testing

## Gantt Chart

****

This Gantt chart is an estimation of our workload and duration. Since we’re still early in development of our application, dates and activities are subject to change. We are opting to use a Prototyping development cycle with frequent prototypes and testing to make sure our application is working as intended.

# Test Plan

CAPP’s test plan for Walker RPG involves multiple interfaces depending on the person playing the game. There are three different interfaces: developer, tester, and user. Using these three modes, we will be able to easily test the features of our games and be able to encounter bugs quickly to resolve them.

## Developer Mode

Developer mode has additional buttons that allows the developer to quickly skip zones and paths, immediate encounter button, auto-steppers, infinite currency, and infinite health during battles. This mode is specifically for testing functions of Walker RPG to make sure they are functional.

## Tester Mode

Tester mode has an auto-stepper, the ability to pick loot from a loot table, and earlier access to zones and abilities. This mode is made to test the balance of the game. We want to make sure equipment isn’t ever too expensive, the player is never too weak in the game, the loot system is giving appropriate loot and there aren’t any major bugs that is preventing the player from playing.

## User Mode

User mode is the default mode. It doesn’t contain any additional features that either developer mode or tester mode offers.

# Risk Management

During the initial planning of this project, we came up with a list of possible risks and a system of monitoring, re-evaluation, and possible contingencies for each risk.

* *Knowledge*

No one in our group is familiar with Unity and only Andrew is familiar with C#. Phillip, Andrew and Cameron have been looking at tutorials for C# and Unity. Our group can use both Lynda and Code Academy for tutorials Unity and C#. We can monitor our group members by making sure they’re contributing in the GitHub. If they aren’t contributing, then we can assume the person on our team isn’t knowledgeable in an area. Our contingency is to peer program to help teach each other.

* *Difficulty of the project*

Our project has a large scope and we have a small window of development time. It is possible we can’t get every feature in our game. If we realize our scope is too large, we can either limit features or scrap them altogether. We also can plan for slack days if we need more time to finish features for our game.

* *Team member’s lack of involvement*

Since our project is so large, we need everyone to work their parts. If someone does not participate in the development of our game, our game will suffer. One way we can monitor this is by making sure we all show up to class and we all show up to our team meetings. If we notice a member is not doing their share, the other group members will notify the non-working member of their low-effort and be warned. If the member continue to not work, we will notify the professor.

* *Scheduling conflicts*

Andrew and Cameron both work and are full time students. Peter has monthly ROTC meetings where he may miss class and meetings. Because most of our members have other responsibilities, scheduling for meetings should be made in advance and members should notify the group when they can’t make it to class or meetings. If a member can’t make a meeting in-person, they can join us through Skype or Discord to give us input and help with documentation using Google Docs.

# Meeting Minutes

9/19 - 3:00 - 3:50 Cameron, Andrew, Phillip, and Peter. Conversed about project and options for developing.

9/21 3:00 - 3:50 Cameron, Andrew, Phillip, and Peter. Conversed about feedback given during elicitation and how best to incorporate this info into our game. Such as an immersive story.

9/26 3:00 - 3:50 Cameron, Andrew, Phillip, and Peter. Figure out we were going to be using unity to program the game and this would mean us all learning C# and we decided we would be using lollipop 5.0 as our SDK

9/28 3:00 - 3:50 Cameron, Andrew, Phillip, and Peter. Talked about the Deliverable 2 along with making a use case diagram

10/3 3:00 - 3:50 Cameron, Andrew, Phillip, and Peter. Talked about diagrams needed in deliverable 2 and drew up some drafts

10/5 3:00 - 3:50 Cameron, Andrew, and Phillip. Talked about deliverable 2 and responsibilities.

10/8 4:00 - 7:20 Cameron, Andrew, Phillip, and Peter. Worked on deliverable 2, Andrew drew up diagrams in UML software. Phillip worked on system functions. Cameron worked on system function and overall description. Peter typed up the introduction section

10/9 4:00 - 6:30 Cameron, Andrew, Phillip, and Peter. Finalized deliverable 2. Cameron worked on Interface uses. Phillip worked on external interface requirements and other nonfunctional requirements. Andrew Proofread and edited sections worked on last night while adding diagrams to the Doc. Peter worked on descriptions, scope and proofreading.

# 

# Contribution Table

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Contribution | % | Note |
| Cameron Fullerton | System functions, overall description, interface requirements.  Proofreading after modifications. | 25% |  |
| Andrew Buikema | Drew diagrams, proofreading.  Modification after peer review | 25% |  |
| Phillip Buckreis | System functions, external interface requirements and other nonfunctional requirements.  Modification after peer review. | 25% |  |
| Peter Hansen | Introduction section, descriptions, scope and proofreading. | 25% |  |