**Software Requirements Specification Template**

The following annotated template shall be used to complete the Software Requirements Specification (SRS). The instructor must approve any modifications to the overall structure of this document.

**Template Usage:**

Text contained within angle brackets (‘<’, ‘>’) shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either ‘Smart Home’ or ‘Sensor Network’.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

**Acknowledgements:**

Sections of this document are based upon the IEEE Guide to Software Requirements Specification (ANSI/IEEE Std. 830-1984).

*Wing it!*

Software Requirements Specification

Version 1.0

2/9/2020

John Nguyen

Team Leader/Programmer

Brad Mancini

Story Writer/Level Designer

Zachary Loch

Digital Artist/UI Designer

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| <date> | <Version 1> | <Your Name> | <First Revision> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | John Nguyen | Leader/Programmer | 2/9/2020 |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[**1. Introduction**](#_heading=h.1fob9te) **6**

[1.1 Purpose](#_heading=h.3znysh7) 6

[1.2 Scope](#_heading=h.2et92p0) 6

[1.3 Definitions, Acronyms, and Abbreviations](#_heading=h.tyjcwt) 6

[1.4 Overview](#_heading=h.3dy6vkm) 6

[**2. General Description**](#_heading=h.1t3h5sf) **6**

[2.1 Product Perspective](#_heading=h.4d34og8) 6

[2.2 Product Functions](#_heading=h.17dp8vu) 6

[2.3 Users and Characteristics](#_heading=h.26in1rg) 7

[2.4 General Constraints](#_heading=h.35nkun2) 7

[2.5 Assumptions and Dependencies](#_heading=h.1ksv4uv) 7

[2.6 Operating Environment](#_heading=h.44sinio) 7

[**3. Specific Requirements**](#_heading=h.2jxsxqh) **7**

[3.1 External Interface Requirements](#_heading=h.z337ya) 8

[3.1.1 User Interfaces](#_heading=h.3j2qqm3) 8

[3.1.2 Hardware Interfaces](#_heading=h.1y810tw) 8

[3.1.3 Software Interfaces](#_heading=h.4i7ojhp) 8

[3.2 Functional Requirements](#_heading=h.1ci93xb) 8

[3.2.1 <Jumping>](#_heading=h.3whwml4) 8

[3.2.1.1 Description and Priority](#_heading=h.lreycje5l67y) 8

[3.2.1.2 Stimulus/Response Sequences](#_heading=h.m8oj8ciymbhz) 8

[3.2.1.3 Functional Requirements](#_heading=h.qszrovs187ed) 8

[3.2.2 <Basic Movement/Running>](#_heading=h.qsh70q) 8

[3.2.2.1 Description and Priority](#_heading=h.q609mzaedjkr) 8

[3.2.2.2 Stimulus/Response Sequences](#_heading=h.lbfnbua4rzgb) 8

[3.2.2.3 Functional Requirements](#_heading=h.vaknje3fxeed) 8

[3.2.3 <Attacking>](#_heading=h.qstnr23d3sj9) 9

[3.2.3.1 Description and Priority](#_heading=h.pu8sfrdf7o2t) 9

[3.2.3.2 Stimulus/Response Sequences](#_heading=h.pzuq7tchbzov) 9

[3.2.3.3 Functional Requirements](#_heading=h.btqwq54a3mmy) 9

[3.2.4 <Saving the game>](#_heading=h.tj07w0sgfl16) 9

[3.2.4.1 Description and Priority](#_heading=h.28kljnhng5my) 9

[3.2.4.2 Stimulus/Response Sequences](#_heading=h.69x6qgdx25um) 9

[3.2.4.3 Functional Requirements](#_heading=h.nlz7jtnw368a) 9

[3.2.5 <Health/HP>](#_heading=h.b9ejfg5nrvir) 9

[3.2.5.1 Description and Priority](#_heading=h.ws5s37hq55w6) 9

[3.2.5.2 Stimulus/Response Sequences](#_heading=h.sprvkboah7lp) 9

[3.2.5.3 Functional Requirements](#_heading=h.hhkkigcs0pnx) 9

[3.2.6 <Collectibles>](#_heading=h.l5aa4cptuesq) 9

[3.2.6.1 Description and Priority](#_heading=h.1hxn3j6a7hnk) 9

[3.2.6.2 Stimulus/Response Sequences](#_heading=h.wz611pwcw3) 9

[3.2.6.3 Functional Requirements](#_heading=h.9goofiyziiij) 10

[3.2.7 <Item Drops/Pickups>](#_heading=h.2r959ngsbg4) 10

[3.2.7.1 Description and Priority](#_heading=h.z8quz4hb4kd4) 10

[3.2.7.2 Stimulus/Response Sequences](#_heading=h.almguhu8a4t2) 10

[3.2.7.3 Functional Requirements](#_heading=h.pnm3oqtcaikv) 10

[3.2.8 <Upgrades>](#_heading=h.3sixft31kkkr) 10

[3.2.8.1 Description and Priority](#_heading=h.kf6s9mna9vxe) 10

[3.2.8.2 Stimulus/Response Sequences](#_heading=h.14mzp7lphqlw) 10

[3.2.8.3 Functional Requirements](#_heading=h.ndgk6zvzq8sz) 10

[3.2.9 <Enemies>](#_heading=h.o7pervw6nllo) 10

[3.2.9.1 Description and Priority](#_heading=h.3rz7f98kjo9s) 10

[3.2.9.2 Stimulus/Response Sequences](#_heading=h.tpw06v3ppjg0) 10

[3.2.9.3 Functional Requirements](#_heading=h.25o9ms4qpzl6) 10

[3.2.10 <Stage Effects/Hazards>](#_heading=h.thf2vz18wyhn) 10

[3.2.10.1 Description and Priority](#_heading=h.x5z2nobt5hd1) 10

[3.2.10.2 Stimulus/Response Sequences](#_heading=h.r38nzyp7nsfq) 10

[3.2.10.3 Functional Requirements](#_heading=h.vf236b1fovyk) 11

[3.2.11 <Currency>](#_heading=h.oalnmnam9ys0) 11

[3.2.11.1 Description and Priority](#_heading=h.t8o8c7kb5sxc) 11

[3.2.11.2 Stimulus/Response Sequences](#_heading=h.6g2ctkrcvtuj) 11

[3.2.11.3 Functional Requirements](#_heading=h.b1nauacfizqp) 11

[3.3 Use Cases](#_heading=h.3as4poj) 11

[3.3.1 Use Case #1](#_heading=h.1pxezwc) 11

[3.3.2 Use Case #2](#_heading=h.2p2csry) 11

[3.4 Non-Functional Requirements](#_heading=h.147n2zr) 12

[3.5.1 Performance](#_heading=h.3o7alnk) 12

[3.5.2 Reliability](#_heading=h.23ckvvd) 12

[3.5.3 Availability](#_heading=h.ihv636) 12

[3.5.4 Security](#_heading=h.32hioqz) 12

[3.5.5 Maintainability](#_heading=h.1hmsyys) 12

[3.5.6 Portability](#_heading=h.41mghml) 12

[3.5 Design Constraints](#_heading=h.2grqrue) 12

[3.6 Logical Database Requirements](#_heading=h.vx1227) 12

[3.7 Other Requirements](#_heading=h.3fwokq0) 12

[**4. Analysis Models**](#_heading=h.1v1yuxt) **12**

[Sequence Diagrams](#_heading=h.4f1mdlm) 13

[**5. Change Management Process**](#_heading=h.3tbugp1) **13**

**List of Figures**

[Figure 1: Data Flow Diagram Example 1 6](#_heading=h.2u6wntf)

[Figure 2 Data Flow Diagram Example 2 6](#_heading=h.19c6y18)

**List of Tables**

[Table 1: Use case 1 4](#_heading=h.49x2ik5)

# 1. Introduction

The purpose of this document is to project the development of a recreational program, a 2D platforming game with elements of exploration and collectables.

## 1.1 Purpose

The goal/purpose of this game is to provide a fun, light-hearted, mostly linear gameplay experience. Secrets may be hidden here and there for a little bit of exploration.

## 1.2 Scope

The software being produced is an entirely recreational program, a 2D platformer video game called “*Wing It!”* and will feature a variety of game mechanics. In the game the player will be able to traverse various levels by jumping and running. Another function the player will be able to do is to fight enemies with the given weapon (Butter knife).

## 1.3 Definitions, Acronyms, and Abbreviations

2D: Two-dimensional

Drops: Items or collectables enemies drop when they are killed. Drops can also be found on the ground naturally

throughout the game.

HP: Health Points/Hit Points; when this reaches 0, the player loses the game

NPC: Non-playable character, characters that the player does not control.

PC: Playable character

Platformer: A sub-genre of game in which gameplay is defined by precise jumping maneuvers.

Side-scroller: A genre of game where the player’s mobility in-game is limited to the X and Y axis.

## 1.4 Overview

<This subsection should:

(1) Describe what the rest of the SRS contains

(2) Explain how the SRS is organized.>

# 2. General Description

“Wing It!” will be a 2D side-scrolling platformer; as such, the gameplay must meet the standard expectations of the genre, as well as be able to compete with other games in the genre.

## 2.1 Product Perspective

“Wing It!” is a stand-alone title, and the first game to be released by our Fowl Play Productions. As such, it’s important to make a good first impression for our studio.

## 2.2 Product Functions

* Allow the player to control the PC using a controller
* The PC is able to run and jump to traverse the level
* The PC is only able to advance to the next level after they have completed the set challenges in each level
* The PC is able to attack enemies to defend themself
* The PC must attack certain enemies to get past them and progress.
* Enemies will try to attack the player to stop them
* A collectible currency will be rewarded to the player for killing enemies and completing various challenges.
* The currency can then be spent on either cosmetic effects for the PC, or upgrades that increase the PC’s capabilities
* The player will be able to save their progress before closing the game.
* The player will be able to load their progress from their save file when restarting the application.

## 2.3 Users and Characteristics

The users of this software, or more accurately, the playerbase of this game is limited to people who use a computer to play video games. A majority of these Computer Gamers use a dedicated gaming computer with advanced specifications. While this means most of our target audience has a disposable income, we do not plan to neglect those who use a market standard, unmodified computer. This program should be able to run on any low end computer without any quality or framerate issues. As far as gaming experience, “Wing it!” caters to fans of 2D platformers, with a particular focus on exploration and collectables.

## 2.4 General Constraints

This program is being built in Unreal Engine Version 4.24. The upper limits of the program are thus defined by this engine, though we do not plan to push the limits of Unreal. More realistically, the production process is constrained by the number of members in our team, and the time limit to complete this project. We only have 3 members, and this program must be completed in 4 months. Not to mention, we have no budget to speak of.

## 2.5 Assumptions and Dependencies

As a 2D platformer, this game must include certain gameplay conventions to fit the genre. Furthermore, since we want the program to run even low end computers, the system requirements must be low, and the graphics must be of a low resolution.

## 2.6 Operating Environment

This game will operate on a Windows environment.

# 3. Specific Requirements

<This will be the largest and most important section of the SRS. The customer requirements are embodied within Section 2 (functions), but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.

Each requirement in this section should be:

* Correct
* Traceable (both forward and backward to prior/future artifacts)
* Unambiguous
* Verifiable (i.e., testable)
* Prioritized (with respect to importance and/or stability)
* Complete
* Consistent (with other requirements)
* Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefully organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.>

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 <Jumping>

#### 3.2.1.1 Description and Priority

The player is able to jump. Jumping is essential to traversing through the levels. Jumping may also be used to transition into attacks or fluttering. Some enemies may also be to jump.

#### 3.2.1.2 Stimulus/Response Sequences

The player presses the button assigned to the jump action. The PC will ascend when the button is pressed. As for enemies, all of their actions will be based on how their AI is scripted.

#### 3.2.1.3 Functional Requirements

A game controller is required to control the PC in any way. There will be no keyboard support.

### 3.2.2 <Basic Movement/Running>

#### 3.2.2.1 Description and Priority

Both playable and non-playable characters are able to do basic movement (left and right). This is must be used in combination with jumping to successfully maneuver through the level.

#### 3.2.2.2 Stimulus/Response Sequences

The player presses the button assigned to the X and Y axis movement (left and right). NPCs will move based on how they are scripted.

#### 3.2.2.3 Functional Requirements

A game controller is required to control the PC in any way. There will be no keyboard support.

### 3.2.3 <Attacking>

#### 3.2.3.1 Description and Priority

Both playable and non playable characters are able to attack. The playable character uses the given weapon to attack

the enemy characters. This is necessary in order to defeat enemies and to traverse to the end of the level.

#### 3.2.3.2 Stimulus/Response Sequences

The player presses the button assigned to attacking and the PC will then be animated to show an attack. The NPCs or enemy characters will attack according to their scripting.

#### 3.2.3.3 Functional Requirements

A game controller is required to control the PC in any way. There will be no keyboard support.

### 3.2.4 <Saving the game>

#### 3.2.4.1 Description and Priority

The player will be able to continue from the last save point they saved at if they choose to leave the game and continue later.

#### 3.2.4.2 Stimulus/Response Sequences

The player must interact with the save point and confirm to when trying to overwrite a previous save game. Saving will come at a cost that will be explained later in this section.

#### 3.2.4.3 Functional Requirements

The player must save their game at a save point in the game. They can create a save to load later, or load after they have lost the game. The player must be able to pay with in-game currency for each use.

### 3.2.5 <Health/HP>

#### 3.2.5.1 Description and Priority

The player will take damage when they are either hit by an enemy or run into an enemy without attacking them. After PC takes damage in some way PC’s total health will be decreased by a predetermined amount. If the PC loses all of their HP the game will end and the player will have to load the game from the last point where they saved. HP can be restored by obtaining items or “drops” that restore the health stat.

#### 3.2.5.2 Stimulus/Response Sequences

When they are either hit by an enemy or run into an enemy without attacking them, the player loses health. When the player picks up item drops that restore HP, the player’s health stat will go up but not exceed the set maximum.

#### 3.2.5.3 Functional Requirements

The player must be below the maximum amount of HP but above 0, to restore health. The player cannot lose any more HP after reaching 0 or gain any more at maximum. The player can take damage at any amount of health.

### 3.2.6 <Collectibles>

#### 3.2.6.1 Description and Priority

Collectibles are special items that are placed in the level somewhere. Some are simply given at the end of the level, some may be hidden throughout the level.

#### 3.2.6.2 Stimulus/Response Sequences

Touching said collectible will increase a counter for that collectible.

#### 3.2.6.3 Functional Requirements

Once a collectible has been obtained, it is saved onto the player’s save file when they save. The player will not have to recollect it and it is not re-collectible like regular item drops.

### 3.2.7 <Item Drops/Pickups>

#### 3.2.7.1 Description and Priority

Enemies that are defeated have a chance of dropping items that the player can pick up. Different pickups have different effects.

#### 3.2.7.2 Stimulus/Response Sequences

When the enemy is defeated, a random number will be generated. If that number is within a certain range, a certain item will appear where the enemy’s position as they were defeated. Touching the item will immediately set off its effect.

#### 3.2.7.3 Functional Requirements

If it is a health recovery drop, the player must be below maximum health. If the pickup is a buff/power-up, that buff should not already be in effect in order to pick up the item.

### 3.2.8 <Upgrades>

#### 3.2.8.1 Description and Priority

The player will be able to receive upgrades throughout the game that permanently change their stats. These are one time obtainables that will increase health, or the attack stat. An example of this would be obtaining a health upgrade. once the upgrade is obtained the players health would increase from 5 to 10 for example.

#### 3.2.8.2 Stimulus/Response Sequences

When the player completes a specific task, like purchasing the upgrade or beating a boss they will be awarded with the upgrade.

#### 3.2.8.3 Functional Requirements

The player must have enough currency to purchase an upgrade, or the boss must be defeated.

### 3.2.9 <Enemies>

#### 3.2.9.1 Description and Priority

An enemy is an entity that will actively pursue the player and attempt to deplete the player’s health.

#### 3.2.9.2 Stimulus/Response Sequences

Each enemy will have a set pattern of behavior. In addition, they will start moving towards the player when within their line of sight.

#### 3.2.9.3 Functional Requirements

The player must be in line of sight to be pursued. The enemy must be above 0 HP or they will be removed from the level.

### 3.2.10 <Stage Effects/Hazards>

#### 3.2.10.1 Description and Priority

The stage itself will have effects that will impact the gameplay. These things can be obstacles, or things that can assist with traversing with levels. These can work with or against the player.

#### 3.2.10.2 Stimulus/Response Sequences

An example would be a weather effect such as wind. This effect would remain effective for the entirety of the level. This can work with or against the player in that it can make jumping across large gaps easier, but also hinder control as it is always pushing the player in one direction. An example of a straight up hazard would be thorns placed around the level. Touching them will reduce the player’s HP.

#### 3.2.10.3 Functional Requirements

The game checks to see if the player is making contact with any of these stage hazards and from there the functionality will be the same as what was stated in the Health/HP section of this document.

### 3.2.11 <Currency>

#### 3.2.11.1 Description and Priority

Currency is the player’s main method of making purchases for upgrades and payment for using a save point.

#### 3.2.11.2 Stimulus/Response Sequences

The player will always receive some amount of currency upon defeating an enemy. It is dropped by enemies and when picked up, the player’s currency counter will increase.

#### 3.2.11.3 Functional Requirements

The player must be below maximum to increase their funds further, otherwise it will just add 0 upon pickup.

### 3.2.12 <Level Select Screen>

#### 3.2.12.1 Description and Priority

Between levels, the player will be presented with a level select screen if they so desire to go back to a previous level or continue to the next level. Player can also save on the level select screen.

#### 3.2.12.2 Stimulus/Response Sequences

The player completes the level and thus the level select screen will be presented to them.

#### 3.2.12.3 Functional Requirements

The player must be between levels, the player cannot access the level screen from the middle of a level.

## 3.3 Use Cases

### 3.3.1 Use Case #1

|  |  |
| --- | --- |
| **Use Case Name** | PC-Enemy Interaction |
| **Reference** | Sections 3.2.1, 3.2.2, 3.2.3, 3.2.5, 3.2.7, 3.2.9, 3.2.11 |
| **Trigger** | There is an enemy blocking the player’s progress |
| **Precondition** | The player is familiar with the game’s controls |
| **Basic Path** | 1. The Player controls the PC to direct this encounter 2. They will attack the enemy until it is defeated. 3. The PC collects any rewards or drops from the enemy, then moves on. |
| **Alternative Paths** | If the player chooses not to attack the enemy, they may try to get past another way:   1. The Player controls the PC to jump over the enemy, evading them entirely. 2. The PC moves onward, nothing is lost or gained from this encounter.   The player may fail to attack or evade the enemy   1. The enemy attacks the PC, causing them to lose HP. 2. The PC becomes briefly invincible after taking damage, however no progress was made. 3. Return to step 1. |
| **Postcondition** | The Player has made progress in the level |
| **Exception Paths** | The Player may quit the game at any time, though progress would not be saved |
| **Other** | The scenario provided outlines a very basic interaction between the player and the game. As the player progresses, levels may be designed to make avoiding enemies impossible. |

*Table 1: Use case 1*

### 3.3.2 Use Case #2

### 

|  |  |
| --- | --- |
| **Use Case Name** | Title Screen |
| **Reference** | Sections 3.2.4 |
| **Trigger** | The Player has booted up the game |
| **Precondition** | The game has been downloaded on the player’s computer |
| **Basic Path** | 1. When prompted with “press any button” the player will do so. 2. The player chooses “Game Start” from the selection of “Game Start”, “Load Game”, “Options”, or “Quit”. 3. The story will now commence from the beginning. |
| **Alternative Paths** | The player may already have a save file, and want to start where they left off. In which case:   1. The player chooses “Load Game”. 2. The story resumes from the last save point.   The player may want to customize their options before starting the game, in which case:   1. The player chooses “Options”, and are taken to the options menu. 2. Once there, a number of fields are customizable, such as: display settings, sound settings, and button settings.    1. From “display settings”, the player can customize screen resolution, framerate, and toggle between fullscreen and windowed mode    2. From “sound settings”, the player can set the master volume    3. From “button settings”, the player can set the button configuration to their liking. 3. Once the player has finished adjusting their preferred settings, they may return to the previous menu at step 2.   The player may want to close the game without starting, for whatever reason. In which case:   1. The Player chooses “Quit”, and the game closes. |
| **Postcondition** |  |
| **Exception Paths** | The player can close the game at any time, though it makes no difference than if they chose to “Quit”, from the selection at step 2. |
| **Other** |  |

### 

### 3.3.3 Use Case #3

### 

|  |  |
| --- | --- |
| **Use Case Name** | Game Over Screen |
| **Reference** | Sections 3.2.4 |
| **Trigger** | The Player has been defeated and the game has ended |
| **Precondition** | Player death |
| **Basic Path** | 1. The player has died 2. The player will be presented with the option to continue from their last save point or return to the title screen 3. The game is loaded from the last save the player made |
| **Alternative Paths** | The player may want to return to the title screen instead.   1. The player has died 2. The player will be presented with the option to continue from their last save point or return to the title screen 3. The player returns to the title screen and can quit if they’d like |
| **Postcondition** | Progress lost, the player must start from the last save point |
| **Exception Paths** | The player can close the game at any time. |
| **Other** |  |

### 3.3.4 Use Case #4

### 

|  |  |
| --- | --- |
| **Use Case Name** | Saving |
| **Reference** | Sections 3.2.4 |
| **Trigger** | Interact with the save point. |
| **Precondition** | Player has enough currency to use the save point. |
| **Basic Path** | 1. Player interacts with save point 2. Game checks to see if player has enough currency 3. If they have enough, the player can create a save file. |
| **Alternative Paths** | * 1. If they do not have enough, the game will simply tell the player that   2. Player does not get a chance to save |
| **Postcondition** | Player can continue playing as normal |
| **Exception Paths** | Player may opt not to save. |
| **Other** |  |

### 

### 3.3.5 Use Case #5

### 

|  |  |
| --- | --- |
| **Use Case Name** | Loading |
| **Reference** | Sections 3.2.4 |
| **Trigger** | Player selects the Load option on the title screen. |
| **Precondition** | Player has made a save file/used a save point. |
| **Basic Path** | 1. Player has started up the game 2. Player selects the Load option on the title screen 3. Player selects their save file 4. Player continues where they left off |
| **Alternative Paths** | 1. Player has lost the game and has been sent back to the title screen 2. Player selects the Load option on the title screen 3. Player selects their save file 4. Player continues the game where they last saved |
| **Postcondition** | Player continues the game where they left off. |
| **Exception Paths** | The player may decide to start a new game. |
| **Other** |  |

### 

### 3.3.6 Use Case #6

### 

|  |  |
| --- | --- |
| **Use Case Name** | Currency Use |
| **Reference** | Sections 3.2.11 |
| **Trigger** | Player interacts with a merchant or save point |
| **Precondition** | Player has currency |
| **Basic Path** | 1. Player talks to merchant 2. Merchant offers items at a cost 3. Player attempts to purchase    1. Game checks to see if the player has enough currency    2. Game gives player the desired item |
| **Alternative Paths** | 1. Player attempts to use the save point 2. Game checks to see if player has enough currency    1. If yes, let the player save    2. If no, tell the player they need more |
| **Postcondition** | Player has less currency |
| **Exception Paths** |  |
| **Other** |  |

### 

### 3.3.7 Use Case #7

### 

|  |  |
| --- | --- |
| **Use Case Name** | Selecting a Level |
| **Reference** | Sections 3.2.12 |
| **Trigger** | Player finishes a level. |
| **Precondition** | Player is between levels, not actively in a level |
| **Basic Path** | 1. Player finishes level 2. Player is presented with level screen 3. Player selects a level |
| **Alternative Paths** | 1. Player finishes level 2. Player is on level select screen 3. Player saves the game 4. Player selects a level. |
| **Postcondition** |  |
| **Exception Paths** | Player can return to the title screen from the level select screen |
| **Other** |  |

### 

### 3.3.8 Use Case #8

### 

|  |  |
| --- | --- |
| **Use Case Name** | Item Usage/Collection |
| **Reference** | Sections 3.2.6, 3.2.7, 3.2.8 |
| **Trigger** | Player touches the item |
| **Precondition** | There is an item to pick up |
| **Basic Path** | 1. Player touches item 2. Item’s effect activates |
| **Alternative Paths** | 1. Player touches collectible item 2. Counter for collectible goes up 3. Item disappears |
| **Postcondition** | Player cannot recollect the collectible |
| **Exception Paths** | Purchased items are not dropped nor require any interaction to use. The effects are automatically applied after purchase. |
| **Other** |  |

### 

### 3.3.9 Use Case #9

### 

|  |  |
| --- | --- |
| **Use Case Name** | Platforming |
| **Reference** | Sections 3.2.1, 3.2.2 |
| **Trigger** |  |
| **Precondition** |  |
| **Basic Path** |  |
| **Alternative Paths** |  |
| **Postcondition** |  |
| **Exception Paths** |  |
| **Other** |  |

### 

### 3.3.10 Use Case #10

### 

|  |  |
| --- | --- |
| **Use Case Name** | Player/Environment Interaction |
| **Reference** | Sections 3.2.10 |
| **Trigger** | Automatically applied when level begins |
| **Precondition** | Player is in the level |
| **Basic Path** | 1. Environmental effects begin 2. Effects are applied to the player 3. Player’s movement/control are affected 4. Player will have to account for these changes as they play |
| **Alternative Paths** |  |
| **Postcondition** | Effects of the environment will cancel when the player exits the level or move to the next level |
| **Exception Paths** |  |
| **Other** |  |

### 

## 3.4 Non-Functional Requirements

### 3.5.1 Performance

Minimum 30fps, Maximum 60fps

### 3.5.2 Reliability

The game should not crash at all. At most, a function may not be working properly.

### 3.5.3 Availability

The game will be free to download/install/play.

### 3.5.4 Security

This game is not a virus. This game has no online capabilities or functions, so there shouldn’t be any security vulnerabilities. Like with most game installation, you need admin rights to install the game like any other program. Most users will likely have admin rights on their own computer.

### 3.5.5 Maintainability

### 3.5.6 Portability

None unless the player is playing this on a Windows laptop.

## 3.5 Design Constraints

The player must have a Windows platform along with the minimum computer specs to run the game. Furthermore, we are adding controller support, so this game is not meant to be played with a mouse and keyboard.

## 3.6 Logical Database Requirements

None. We do not plan to use a database.

## 3.7 Other Requirements

<Catchall section for any additional requirements that did not belong to the previous sections. If there are none, exclude this section>

# 4. Analysis Models

<List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.>

## Sequence Diagrams

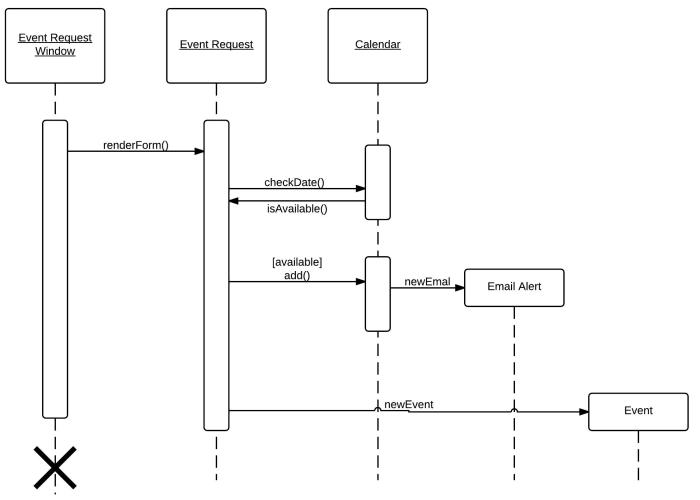
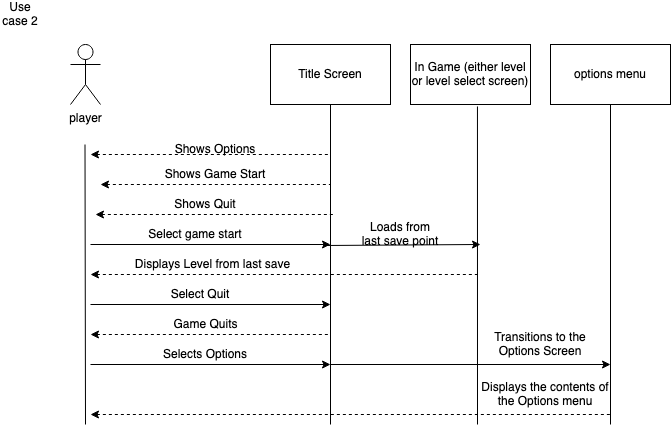


Figure 1: Data Flow Diagram Example 1



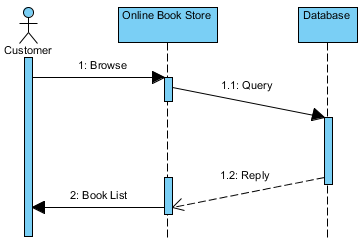


Figure 2 Data Flow Diagram Example 2

<At least one sequence diagram should be included for each requirement or use case.>

# 5. Change Management Process

The document will be updated through Google Drive and any finalized changes will be uploaded to the repository on GitHub. The leader will look over all the changes made by everyone before updating the repository.