# Group 8 - Deliverable #1
# SE 3A04: Software Design II – Large System Design

Anando Zaman
Brian Kibazohi
David Yao
Qiushi Xu
Tamas Leung

## 1 Introduction

This software requirements specification is written for "Ready Player One" arcade game library.

### 1.1 Purpose

The purpose of this document is to outline the overall description and target functionality of the game software to stakeholders. This will cover the design constraints, use cases, scenarios, and requirements aimed towards clients such as gamers and developers. In addition, it will also serve as a reference for verifying correctness and validation of the product during the testing phases.

### 1.2 Scope

The scope of the proposed software, Ready Player One, is an arcade **game library**. It will feature a set of 4 to 5 mini-games where users will compete to achieve a high-score amongst other players. The intention is to encourage friendly competition with a variety of mini-games that will run indefinitely while maintaining information about the high score. The score can be characterized by a measurable aspect such as time, kills, checkpoints, etc.

### 1.3 Definitions, Acronyms, and Abbreviations

a) game library - a collection of games

b) Mini game ending - the time when the user finishes playing a mini game.

### 1.4 References

N/A

### 1.5 Overview

This document is organized into sections with smaller subsections that will aid in describing a given section. The intention is that these sections will work together to provided a more informed description of the product. The document is separated into sections that include the overall description, use case diagrams, functional requirements, and non-functional requirements.

# 2 Overall Description

The overall description details the important factors that impact our final product development process such as development constraints, product functionality, and user characteristics.

## 2.1 Product Perspective

The video game industry is a large and diverse space with a variety of genres. Our software product also falls under the gaming category, specifically for desktop platforms. The product is focused towards targeting a specific niche of the industry, particularly arcade styled games focused on catching market trends. This plays a part in the design, hence why most of the mini-games are simple, fun and intuitive. For example, Flappy-bird and 2048 are great examples of this. In developing a video game, the player's enjoyment needs to be taken into account. Implementing things like high scores and increasing difficulty, for example, serve to provide a sense of progression and competitiveness, encouraging repeated play.

Our product is differentiated by encapsulating several mini-games into a single hub that makes it more convenient to play multiple arcade games in one place. In essence, it imitates a real-life arcade with several games. Our product is self-contained for the most part except that it will interface with an external database for data storage.

## 2.2 Product Functions

### 2.2.1 Function Summary

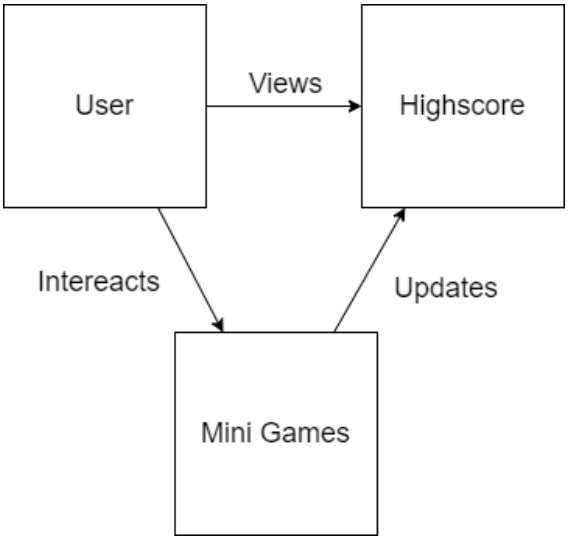
The software functionality is quite simple and is focused primarily around allowing the user to select and play a mini-game to compete for a high-score. The player will be able to login and view their overall score as the sum of the scores of all 5 games. Additionally, the player will be able to view their score for a specific game by clicking on the icon corresponding to the game from the home-screen. The player will also have the ability to stop or pause a game before continuing or exiting to the main screen. While in a game session, the player will use specific keyboard or mouse inputs to interact with the game. Each mini-game is different and will have it's own unique inputs to interface with the game.

### 2.2.2 List of functions

1. User enters their username and moves to the game selection screen.
2. User creates a new mini game instance.
3. User pauses a mini game
4. User exits a mini game
5. User restarts a mini game
6. User can view high scores
7. Mini game ending saves a high score.

### 2.2.3 Function domain

For the functions mentioned above. The user would start the game by entering their username and moves to the game selection screen. At this stage, the user can choose a mini game that he or she wants to play by creating a new mini game instance. Once the user starts a game, he or she can pause, exit or restart as their wish. At the end of each game, the system would save the highest score achieved by the user. Lastly, when the user finished playing all the games, he or she can check their highest score.




## 2.3 User Characteristics

"Ready Player One" is intended to be playable by anyone with a "basic level" of computer literacy. This "basic level" can be defined as being able to interface with the operating system with ease and past experience using a mouse and keyboard. Any past experience with arcade themed games are not required. The user must also be able to comprehend the English language as that will be the main language of the game for navigating across different game scenes.

## 2.4 Constraints

List of items that will limit the developers options

- Development time of this project is limited and may limit the complexity of functions.
- Assets used must be available for public use.
- Pygame platform has bad graphic scaling capabilities limiting the ability to pursue more advanced graphical options. It also doesn't have intuitive ways to form UI components such as textbox inputs which can make specific features involving text input difficult to implement.
- The type of cloud database vendor used may have certain restrictions such as specific timeout periods or hourly quota's which can impact product financials.
- Developer background experience: Some people are new with PyGame or are new to game development which can make it difficult to get started.

## 2.5 Assumptions and Dependencies

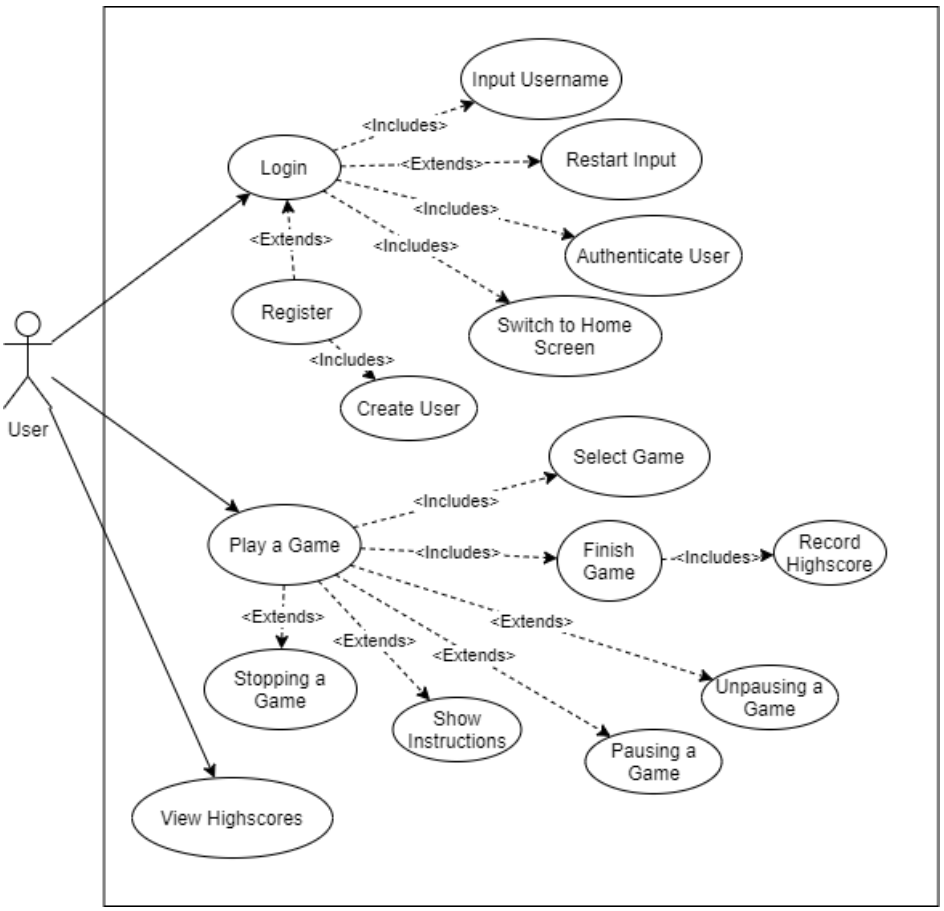
List of assumptions

- Assumption of stable internet connection to query information from a cloud database.
- Assumption that the the program can run on the users' hardware.

### 2.6 Apportioning of Requirements

- Allowing multiple players to play concurrently to compete with each other in real-time.
- Allow users to authenticate themselves may be delayed or not included if the support for multiple PyGame user input textboxes proves to be difficult to implement in a short period of time.

## 3 Use Case Diagram




1. Login/Register: User can enter information to login/register enabling high score recording. The user will input their info and will be authenticate. Upon success, the screen will switch to home screen
2. Play A Game: User can select a game to play. When playing a game, user has options to pause a game, stop a game or show instructions. Upon ending a game, a highscore is recorded
3. View High scores: User can now view a table listing the top high scores achieved by users.

# 4 Functional Requirements

The following Business Events are about the User ViewPoint

VP1. BE1.1 The user wants to login/register a username

i. The system shall allow the user to input a username
ii. The system shall allow the user to confirm selection
iii. The system shall authenticate the user
iv. The system shall have a means to restart/reset their input
v. The system shall transition to the game selection screen if authentication successful.

BE1.2 The user wants to select a game.

i. The system shall allow the user to browse/view the available games.
ii. The system shall prompt for confirmation for the selected game.

BE1.3 The user plays a game.

i. The system shall notify the user what game they are playing.
ii. The system shall alert the user with brief instructions to the game.
iii. The system shall have options to pause/resume the game
iv. The system shall have the option to exit the current game and return to the game selection menu.
v. The system shall record the users' score.
vi. The system shall be able to update the users highscore and the overall maximum highscore

BE1.4 The user wants to view their score.

i. The system shall display text on the top-right corner displaying the score of a particular game.
ii. The system shall display the global max highscore and the corresponding users name.

# 5 Non-Functional Requirements

## 5.1 Look and Feel Requirements

### 5.1.1 Appearance Requirements

LF1. The system shall have a minimalistic design with an classic arcade theme. This means having a neutral background without any fancy patterns or gradients.

### 5.1.2 Style Requirements

LF2. The system shall provide the player a feeling or mood as if they are playing in an arcade.

## 5.2 Usability and Humanity Requirements

### 5.2.1 Ease of Use Requirements

UH1. The UI should be user friendly such that the user is able to navigate to the desired screen or event(ie; game screen) in under 10 seconds.

UH2. The system shall provide the user with descriptive error messages in case an unforeseen event occurs.

UH3. The user shall be able to flow through tasks without being interrupted by external events

### 5.2.2 Personalization and Internationalization Requirements

N/A

#### 5.2.3 Learning Requirements

UH4. The system shall be easy to self-learn such that the user is able to understand how to play a particular game in under 20 seconds.

UH5. The system shall also provide brief instructions to get the player up-to speed if they are confused.

#### 5.2.4 Understandability and Politeness Requirements

UH6. The system shall use buttons with short descriptive text to indicate the function.

UH7. The system shall use icons taken from a common icon asset repository to make the program intuitive.

UH8. The system shall use use alert prompts to ensure that the user is aware about their actions and can confirm upon it.

#### 5.2.5 Accessibility Requirements

N/A

### 5.3 Performance Requirements

#### 5.3.1 Speed and Latency Requirements

PR1. The system is expected to perform and respond to the user at a speed no more than 0.2s.

#### 5.3.2 Safety-Critical Requirements

N/A

#### 5.3.3 Precision or Accuracy Requirements

PR2. The system will ensure accuracy and precision when the leaders-board table requires an update.

PR3. In case of a tie on the leaders-board, the system will use "Time" to the nearest millisecond as a determining factor

#### 5.3.4 Reliability and Availability Requirements

PR4. The system is expected to be ran on the user's local machine. Availability and reliability is dependent on the times they allocate to play the game, and independent of the developer / system.

PR5. The system will get its post and retrieve high score details from an online database, we expect this database to be available and running 24/7, 7 days a week.

#### 5.3.5 Robustness or Fault-Tolerance Requirements

PR5. In response to the user input commands, the system is expected to operate at a highly robust level since there is a finite amount of keys / buttons to be used during the game or when navigating across the web app.

PR6. The system is expected to support one user at a time, hence we will not face any overload with traffic.

PR7. The system is expected to make a low amount of calls to the database server in comparison to threshold provides for online requests. Hence we consider our system stable and fault tolerant.

#### 5.3.6 Capacity Requirements

PR7. The system is expected to be ran on a local machine, such as a PC / laptop. The system is expected to ran smoothly on a PC with at least 100Mb of RAM and 300MB of storage.

#### 5.3.7 Scalability or Extensibility Requirements

PR8. The system is expected to scale if and only if the user choose to download a new version provided by the developer.

#### 5.3.8 Longevity Requirements

N/A

### 5.4 Operational and Environmental Requirements

#### 5.4.1 Expected Physical Environment

N/A

#### 5.4.2 Requirements for Interfacing with Adjacent Systems

N/A

#### 5.4.3 Productization Requirements

OE1. The system will be distributed in the form of python modules and potentially an executable file.

OE2. The system will be available online through a repository.

#### 5.4.4 Release Requirements

OE3. The system will provide development/beta releases on a monthly basis.

OE4. The system will have a final release in April 2021.

### 5.5 Maintainability and Support Requirements

#### 5.5.1 Maintenance Requirements

MS1. The system will allow minigames to be updated individually without affecting any other game.

#### 5.5.2 Supportability Requirements

MS2. Support and instructions about the system are expected to be found on the READme.md file on github, the source.

#### 5.5.3 Adaptability Requirements

MS3. The systems dependencies and packages will be updated to the latest stable version to provide robust and stable experiences for the users.

### 5.6 Security Requirements

#### 5.6.1 Access Requirements

SR1. The system will use social identity providers to authenticate users onto the web-app.

#### 5.6.2 Integrity Requirements

SR2. The system will ensure integrity with data shared across the platform. Data such as score metrics, games played etc. The data presented is expected to be true, and relevant to the user and game.

#### 5.6.3 Privacy Requirements

SR3. The system will ensure to encapsulate all user sensitive and private information. The system will not store or use any irrelevant user data that does not serve towards the game's main objective.

#### 5.6.4 Audit Requirements

N/A

#### 5.6.5 Immunity Requirements

N/A

### 5.7 Cultural and Political Requirements

#### 5.7.1 Cultural Requirements

N/A

#### 5.7.2 Political Requirements

N/A

### 5.8 Legal Requirements

#### 5.8.1 Compliance Requirements

N/A

#### 5.8.2 Standards Requirements

N/A

# A Division of Labour

Table 1: Contributions

| **Name** | **Contribution** |
|---|---|
| Tamas | Use Case diagrams, Function Domain |
| Anando | Most FR, some NFR(Look/feel, usability), user characteristics/constraints |
| David | User characteristics, Some FR, NFA |
| Brian | NFA - performance(and more NFA) |
| Quishi | Function domain, Acronyms |

T.L
Tamas Leung

A.Z
Anando Zaman

D.Y
David Yao

B.K
Brian Kibazohi

Q.X
Quishi Xu