**Software Requirements and Design Document**

**For**

**Group 4: Retro-Arcade**

Version 1.0

**Authors**:

Andrew L

Joseph B

Mackenzie P

Michael T

Seth P

# Overview

Our system is a Retro Arcade application. The user will have 5 games to choose from and will be able to track their high scores depending on the game. We will have a GUI that resembles an old arcade machine where the player can choose which game they wish to play. Along with high scores for individual players, we will also track high scores between different players. The games included will be variations of the games Pong, Flappy-Bird, Snake, and Space Invaders.

# Functional Requirements

1. We will have user profiles that are used to identify different users scores in games as well as update leaderboards/high scores. - **medium priority**
2. SQLite database integration for each game, including capacity for analytics: - **high priority**

**Note:** High scores will be viewed on HTML pages (styled with CSS) using Python3’s *Flask* library to create a (virtual) webserver on the localhost.

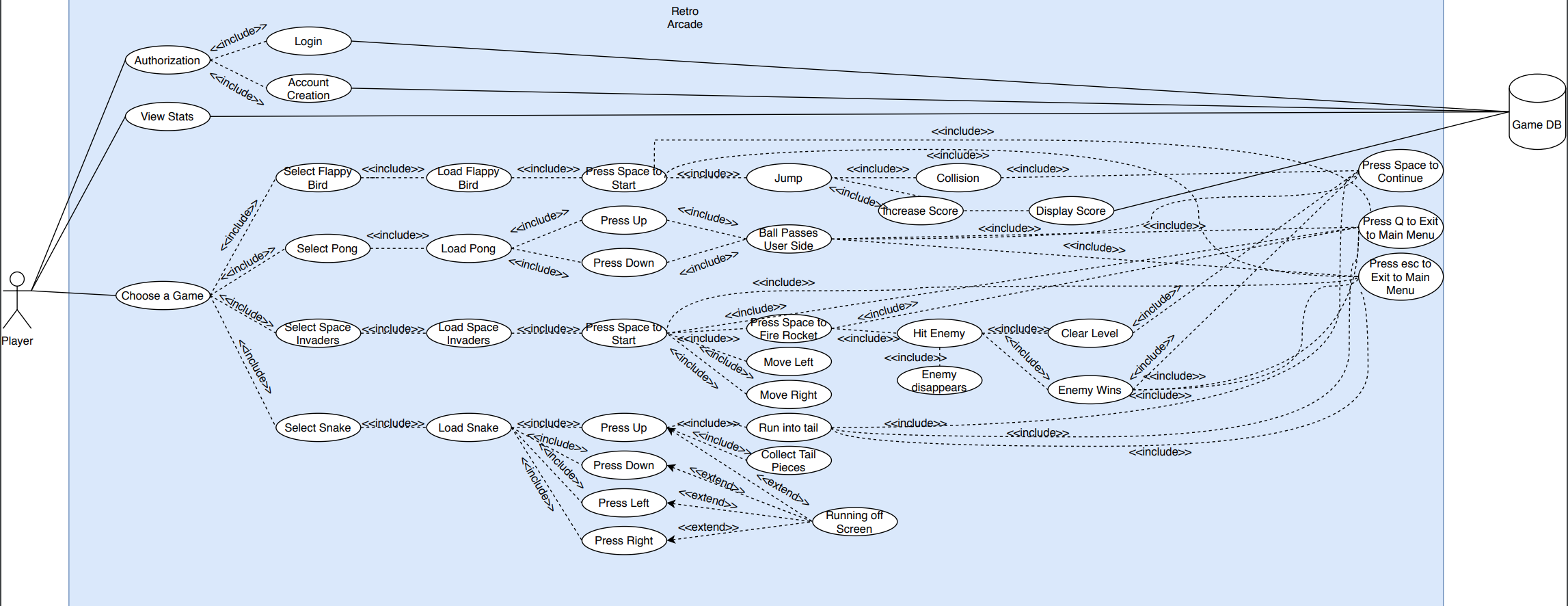
* 1. For an individual user:
     1. All-time highest score for each game.
     2. History of user’s top 5 scores for each game
     3. Aggregate of the user’s highest score for all game (adjusted for different methods of scoring in different games) individual user high score to give them an overall.
  2. Between users:
     1. Capacity to check game’s all-time 10 highest scores and the associated user for each game.
     2. ‘All-time’ Leaderboard with ranking based on the users with the highest composite scores

1. A GUI modeled after a classic arcade machine will be implemented. - **high priority**
2. Games the user can select to play: Flappy Bird, Space Invaders, Snake, Pong. - **high priority**

# Non-functional Requirements

1. Ensure that the system will not affect the overall performance of a user's machine, allowing the user to use our system as well as other applications simultaneously.
2. Encrypt user profile’s password using **bcrypt** encryption method.
3. System stability: prevent crashes throughout the functions provided in the system.
4. Aesthetic of high-scores pages will be kept to a minimum until all other functionalities are finished. CSS may be added but is low priority.
5. Flask server uses https / SSL 2.0 to launch highscore webpages

# Use Case Diagram



# 

**Use Case Descriptions**

**Authorization** – When the user first accesses the arcade, they are asked to login or create an account, which is directly associated with the Game database

**Login** – login with correct email and password

**Create Account** – user creates account with email and encrypted password (will be implemented in third increment)

**View Stats** – Once logged in, the player will be able to login and see their respective high scores for each game and the overall high scores from other players.

**Choose a game** – once logged in, the user will be given the option to choose from a list of four games.

**Press Space to Start** – Starts the game once space is pressed

- Preconditions – Load Flappy Bird

- Postconditions – Press Q to quit, press esc to quit, jump (using space)

**Collision** – Where the user collides with spear and therefore loses

- Preconditions – jump into spear

- Postconditions – user can press space to continue, if they continue, then they go back to the press space to start menu, where they have the option to quit with esc and ‘Q’

**Increase Score** – Increases the score as the player jumps and the displays the new score

**Display Score** – Score checked with high score in database and saved if new score is higher, once player collides

**Press Up** – moves paddle up.

- Preconditions – load pong after it has been selected as game of choice by user

- Postconditions – ball either hits paddle and remains in play or ends up passing the user’s side and therefore player loses.

**Press Down** – moves paddle down.

- Preconditions – load pong after it has been selected as game of choice by user

- Postconditions – ball either hits paddle and remains in play or ends up passing the user’s side and therefore player loses.

**Ball Passes User Side** – this triggers a loss for the user and will then ask the user if they would wish to play again.

**Press Space to Start** – Spawns the rocket, ship, and enemies for a level of Space Invaders and player is able to start firing rockets at aliens(enemies).

**Press Space to Fire Rocket** – When space is pressed, a rocket is fired upward from the ship toward the aliens

**Move Left** – The user presses the left arrow key to move left and the ship will move that direction.

**Move Right** - The user presses the right arrow key to move right and the ship will move that direction.

**Hit Enemy** – This event is triggered if the rocket crosses paths with the location with an alien.

**Enemy Disappears** – If the enemy is hit, the alien will disappear from the screen.

**Clear Level** – If all aliens disappear from the screen, meaning the player has successfully hit all of the aliens with rockets, the user will be asked if they want to continue to play the next level by hitting the space bar.

- Preconditions – clear all enemies from screen by firing rockets with space bar

- Postconditions – press space to continue or press ‘Q’ or ‘esc’ to exit.

**Enemy Wins** – This event is triggered if the aliens pass the ship before all of the aliens are cleared from the level

- Preconditions – Alien passes ship level

- Postconditions – press space to continue or press ‘Q’ or ‘esc’ to exit.

**Press Up** – snake goes in the upward direction upon hitting the up-arrow key.

- Preconditions – load snake game after user selects it.

- Postconditions – run into tail, collect tail piece

**Press Down** – snake goes in the downward direction upon hitting the down-arrow key

- Preconditions – load snake game after user selects it.

- Postconditions – run into tail, collect tail piece

**Press Left** - snake goes in the left direction upon hitting the left-arrow key

- Preconditions – load snake game after user selects it.

- Postconditions – run into tail, collect tail piece

**Press Right** - snake goes in the right direction upon hitting the right-arrow key

- Preconditions – load snake game after user selects it.

- Postconditions – run into tail, collect tail piece

**Run into Tail** – When the snake runs into its tail, the game ends and the user is asked if they wish to restart.

**Collect Tail Pieces** – This event makes the snake longer if the user runs over a spot with a tail piece.

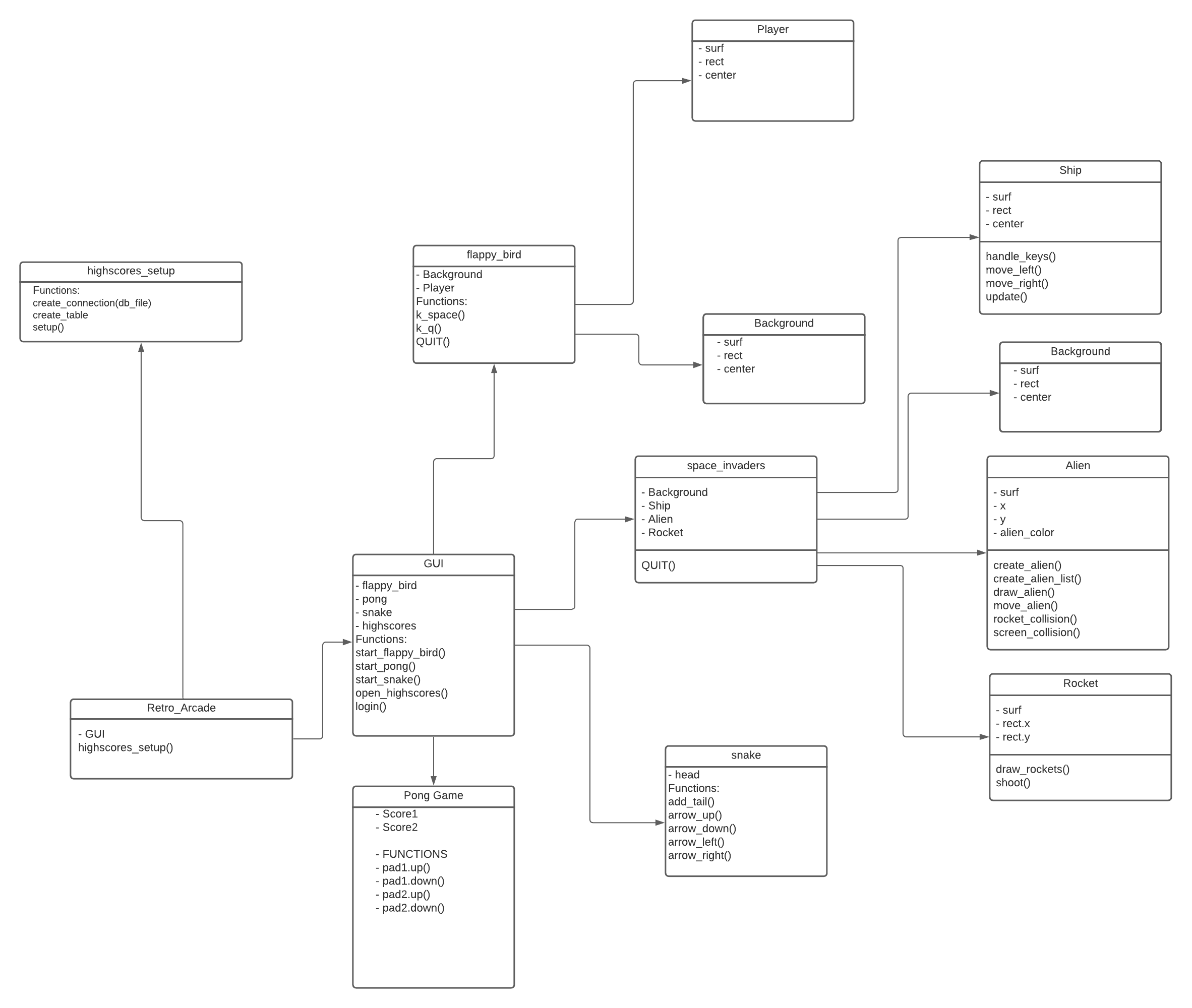
**Running off Screen** – Running off screen does not kill the snake

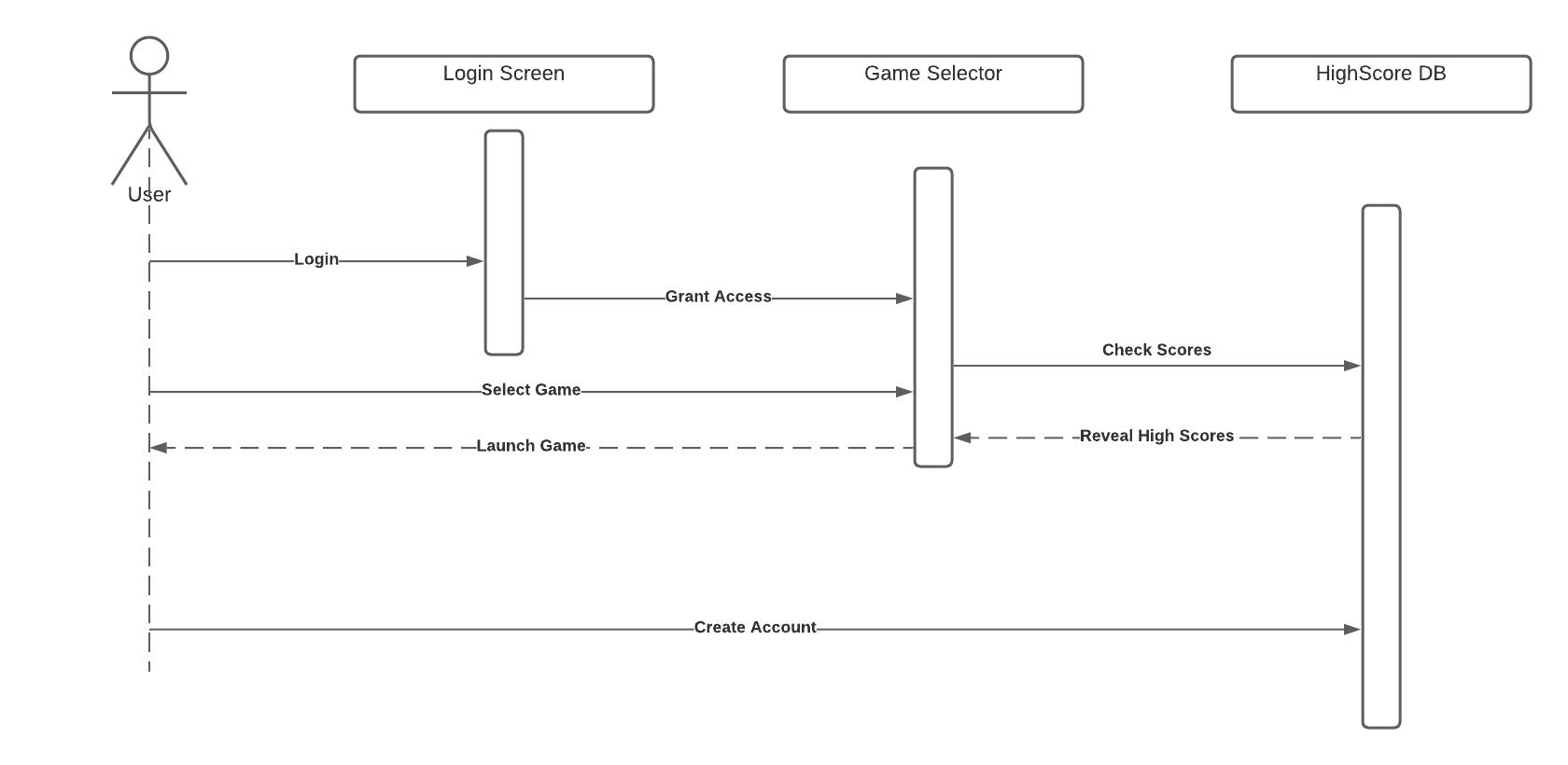
**Press Space to Continue** – Space continues the game in question. This functionality will be used in every game to maintain consistency.

**Press Q to Quit to Main Menu** – ‘Q’ exits the game and will return the user to the main menu where they can once again select another game or view stats. This functionality will be used in every game to maintain consistency.

**Press ‘esc’ to Exit to Main Menu** - ‘esc’ exits the game and will return the user to the main menu where they can once again select another game or view stats. This functionality will be used in every game to maintain consistency.

# Class Diagram and/or Sequence Diagrams





# Operating Environment

Our software works in any operating system environment provided that the user has python and the necessary python libraries installed.

# Assumptions and Dependencies

As of right now, the only issues that our group can see the user having when trying to use our software is not having the proper python libraries installed. We will be coming up with a solution to solve this, such as creating a script that will install such libraries for the user. There is also a possibility that we can “freeze” the python libraries into an app such that all dependencies are bundled within and one executable is able to run the application.