**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, Space Invaders, and Jet Fighter.

# 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:** Highscores will be viewed on HTML pages 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. 4 Games the user can select to play: Flappy Bird, Space Invaders, Snake, Pong. - **high priority**

# Non-functional Requirements

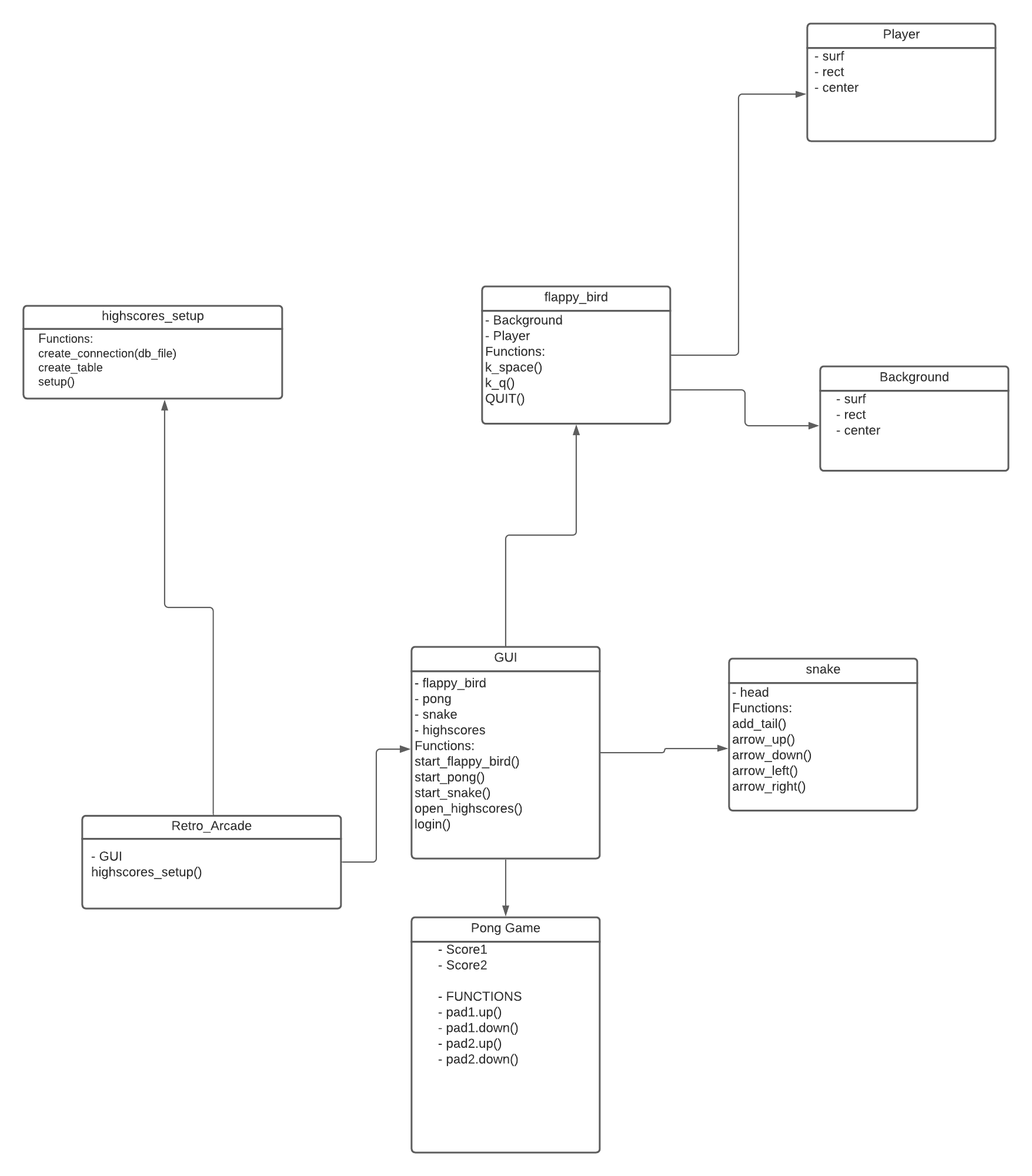
1. Ensure that the system will not affect the overall performance of a user's machine, allowing the user to use our system as well as other applications simultaneously.
2. Encrypt user profile’s password using **bcrypt** encryption method.
3. System stability, prevent crashes throughout the functions 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.

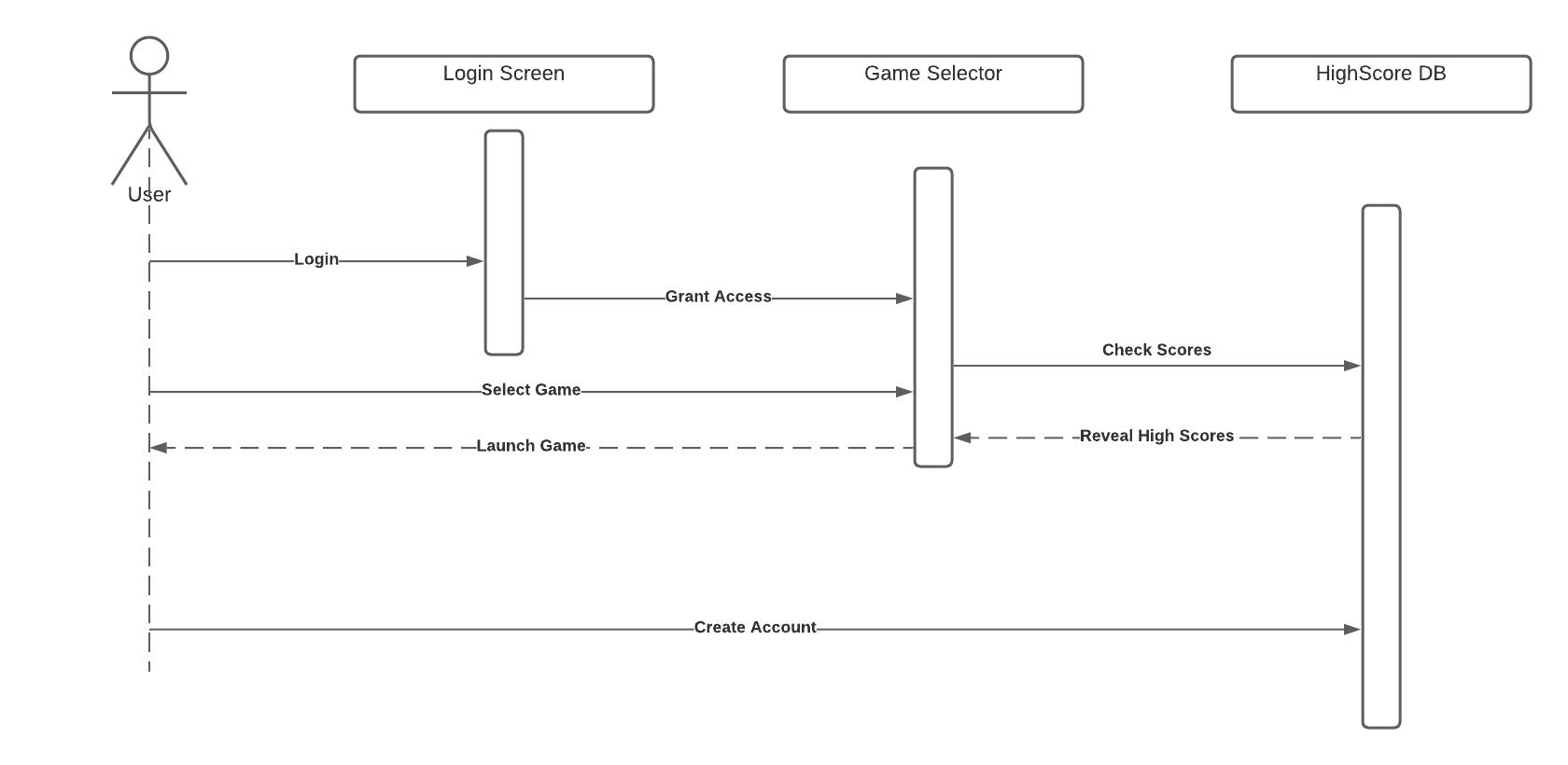
# Use Case Diagram

*Chart

Description automatically generated*

# Class Diagram and/or Sequence Diagrams





# Operating Environment

Our software works in any operating system environment as long as the user has python and the python library pygames 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.