**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 (5 points)

*Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).*

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 (10 points)

*List the* ***functional requirements*** *in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

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**
   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
3. A GUI modeled after a classic arcade machine will be implemented. - **high priority**
4. 5 Games the user can select to play: Flappy Bird, Space Invaders, Snake, Pong, and Jet Fighter. - high priority

# Non-functional Requirements (10 points)

*List the* ***non-functional requirements*** *of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

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.

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

# Class Diagram and/or Sequence Diagrams (15 points)

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

# Operating Environment (5 points)

*Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.*

# Assumptions and Dependencies (5 points)

*List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.*

**Increment 1**

**Progress Report**

During this increment, we accomplished a general breakdown of workload distribution for the project while deciding on which 5 games we were going to implement. The only challenge, a small one, we faced was getting a weekly meeting time set up to allow us to go over what we have accomplished, some challenges, and what needs to happen over the next week. For this first increment, Seth will be handling the GUI and one of the games: pong. Mackenzie and Michael are teaming up to work on a Flappy bird style game, Andrew will be getting started on implementing the high scores for each game and Joe will be getting started on the game, Snake.

Draft/Prototype Schema (SQLite3) for Highscores

A sign on the side of the street

Description automatically generated

Andrew 1