2D Basketball Game – “Hoop Town”

1. Revision History

Version	Date	Description	Authors
0.1	2025-04-13	Initial draft for Tuesday check-in	Evan Gallagher,Brandon Mountan,Thomas Ho

2. Table of Contents

---

3. Introduction

3.1 Purpose

This SRS defines the requirements for “Hoop Townr,” a 2D basketball game that will be developed as a group project. The game aims to build a framework and exercise our software engineering skills while supporting user account management. Users may create an account to track their progress, view historical scores, and play against another player locally.

3.2 Scope

“Hoop Town” is a 2D basketball game in which players make baskets to score points. The game includes the following major components:

- **User Management:** Registration (sign up), login, and guest play.
- **Game Mechanics:** Physics-based shooting with scoring, and stealing the ball.
- **Progress Tracking:** Each player’s account will store the highest level reached and historical scores.
- **User Interface:** A graphical interface that presents options to play as a guest, sign in, or sign up, and displays a personalized birthday greetings if appropriate.

This SRS covers the two sprints planned for the project:

- **Sprint 1 (Product Backlog 1):** Account management, profile, and basic game history functionality.
- **Sprint 2 (Product Backlog 2, Option 3):** The design and implementation of a fully featured 2D basketball game with creative gameplay elements.

3.3 Definitions, Acronyms, and Abbreviations

- **UI:** User Interface
- **SRS:** Software Requirements Specification
- **SDK:** Software Development Kit
- **JSON:** JavaScript Object Notation
- **TXT:** Plain text file
- **Account:** A unique user registration for saving game progress
- **Guest Play:** Option for playing without registration

3.4 References

- IEEE Standard for Software Requirements Specifications
- Qt Creator Documentation
- Relevant online resources about 2D game development and user account design (e.g., tutorials and sample projects)

3.5 Overview

This document outlines the functional and non-functional requirements for “Hoop Town.” It describes how players will interact with the game, the key features required for account management and game mechanics, design constraints, and assumptions regarding the development environment. Additionally, it discusses recommended data storage methods (using JSON or TXT files) for saving user data and historical game scores.

---

4. Overall Description

4.1 Product Perspective

“Hoop Town” is a standalone application that combines a game engine with an account management system. It will run on desktop platforms using the Qt framework and modern C++ for game logic and UI design.

- **System Context:** The product will connect to local storage for saving user profiles and game history.
- **Interfaces:** It will use graphical interfaces for account management and the game, and store data via files (using JSON or TXT formats) on disk.

4.2 Product Functions

The application will support the following core functions:

- **User Registration and Login:**
  - Sign up with first name, last name, date of birth, gender (optional), username (unique), and password (minimum 8 characters with a number, upper and lower case).
  - Login to view a personalized dashboard.
  - Guest play option.
- **Birthday Greetings:**
  - On login, if the current date matches the user’s birth date, a personalized birthday greeting is displayed.
- **Game History:**
  - Store and display historical game scores, comparing the user’s scores to global best scores.
- **Game Mechanics (2D Basketball):**
  - Players shoot a basketball with physics-based gameplay.
  - Players can play with another local player in an arcade style.
- **Persistent Data:**
  - User account and game history stored in files (JSON or TXT).

4.3 User Characteristics

- **Novice to Intermediate Gamers:** Familiar with standard computer interfaces and games.
- **Account Holders and Guest Users:** The system supports registered users and guests.
- **Team Environment:** The application is designed for team-based development with distributed responsibilities.

4.4 Assumptions and Dependencies

- It depends on the Qt framework (version 6.9.0 as an example) and a modern C++ compiler.

- The local file system will be used for persistent storage (via JSON or TXT files).
- The team has defined sprint time frames and has clear responsibilities for account management, game mechanics, and UI development.

---

5. Specific Requirements

5.1 Functional Requirements

FR1: User Account Management

- **FR1.1:** The system shall allow users to sign up by entering first name, last name, date of birth, gender (optional), profile picture, username, and password.
- **FR1.2:** The system shall ensure that the username is unique.
- **FR1.3:** The system shall enforce password rules (minimum 8 characters, including at least one number, upper-case and lower-case letter).
- **FR1.4:** Upon login, the system shall display the user's name, and the current date.
- **FR1.5:** If today is the user's birthday, the system shall display a birthday greeting.

FR2: Game History and Progress Tracking

- **FR2.1:** The system shall record game scores and levels reached.
- **FR2.2:** The system shall allow the user to view a history of their game scores and compare them to the global best score.

FR3: 2D Basketball Game Mechanics

- **FR3.1:** The game shall allow a player to shoot a basketball on a 2D court.
- **FR3.2:** The game shall provide visual feedback (animations) and sound effects when the shot is made.
- **FR3.3:**The game shall allow players to steal and block the ball from their opponent..

FR4: Persistent Data Storage

- **FR4.1:** The system shall store user data and game history in external files.
- **FR4.2:** The system shall support data storage in either JSON or TXT formats.

5.2 Non-Functional Requirements

- **NFR1: Performance:**
  - The game should maintain a steady frame rate (e.g., 30–60 FPS) on supported hardware.
- **NFR2: Usability:**
  - The user interface must be clear, intuitive, and responsive.
  - Account creation and game controls should require minimal training.

- **NFR3: Reliability:**
  - User data and game history should be reliably stored and retrieved.
- **NFR4: Portability:**
  - The game should run on multiple desktop platforms (macOS, Windows, Linux) with minimal changes.
- **NFR5: Security:**
  - Passwords should be stored securely (ideally hashed if stored in files) and user information protected.
- **NFR6: Scalability:**
  - The design shall allow for adding additional game features or modes in future sprints.

5.3 Interface Requirements

- **User Interface:**
  - The application will use the Qt Widgets framework for a graphical interface.
  - The main window will feature tabs or panels for signing up/in, profile display, game area, and history tracking.
- **External Interfaces:**
  - File I/O interfaces for persistent storage in JSON/TXT format.
  - Optional external APIs for global high score updates (if implemented in future sprints).

---

6. Design Constraints

- **Development Framework:** The application must be built using Qt (e.g., Qt 6.9.0).
- **Programming Language:** The code will be written in C++.
- **Data Storage:** Persistent data should be stored locally using either JSON or plain text files.
- **Team Collaboration:** The project will use version control (e.g., Git) with pull requests, and work will be divided across sprints.

---

7. Data Storage and File Management

- **User Data and Game History:**
  - All user data (profile information, historical scores, progress) will be stored in external JSON files. JSON is preferred because of its structured format and ease of parsing in C++.
  - Alternatively, TXT files can be used if simple string storage is sufficient.

- **Resource Files:**
    - Images (profile pictures, game textures) and sound files will be stored in designated resource folders.
- **Configuration Files:**
    - Game configuration settings (such as level difficulty, game speed, etc.) may be loaded from an external file using the “Load File” functionality.

---

8. Appendices

Appendix A: Glossary

- **Account:** A user’s profile including login credentials and game progress.
- **2D Basketball Game:** A side or top-down basketball game where players shoot, score, and progress through levels.
- **Sprint:** A set period during which specific features are developed.

Appendix B: References

- IEEE SRS standard guidelines
- Qt documentation
- JSON specification
