Software Requirements Specification

for

Advanced Tic Tac Toe Game

VERSION 1.0

Prepared by:

- Abdelrahman Mohamed
- Mohamed Hisham Mohamed
- Salah Eldin Hassen
- Shahd Gamal
- Shahd Hamad

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the Advanced Tic Tac Toe Game, including its functionalities, performance requirements, and constraints. This document serves as a foundation for the design, development, and testing phases of the project.

1.2 Intended Audience and Reading Suggestions

The intended audience for the Advanced Tic Tac Toe Game includes casual gamers, and individuals seeking to improve their strategic thinking skills. The game is designed for users of all ages who enjoy classic paper and pen games and are looking for a digital version of Tic Tac Toe. Additionally, it targets users who appreciate advanced features such as playing against an AI opponent and tracking their game history. Educational institutions may also find the game beneficial for teaching strategic gameplay and decision-making skills.

1.3 Project Scope

The Advanced Tic Tac Toe Game is a software application developed in C++ with a user-friendly GUI, supporting both player vs player and player vs AI modes. It includes user authentication, personalized game history, and an intelligent AI opponent. The project follows best practices in software engineering, including secure user management, individual unit testing, and CI/CD integration.

1.4 References

- Google C++ Style Guide: Google C++ Style Guide - GitHub
- GitHub Actions Documentation: https://docs.github.com/en/actions
- SQLite Documentation: SQLite Documentation
- YAML Official Website: YAML Tutorial: Everything You Need to Get Started in Minutes - CloudBees

2. Overall Description

2.1 Product Perspective

The Advanced Tic Tac Toe Game is designed to enhance the classic Tic Tac Toe experience with modern features and AI. Built using C++ and the Qt framework, it supports Windows. The game includes user authentication, personalized game history, and an intelligent AI opponent using the minimax algorithm with alpha-beta pruning. The primary components game logic, AI opponent, user management, game history management, and GUI work together seamlessly to offer a fun and beneficial experience for the user.

2.2 Product Features

- **Game Play:** Allow users to play Tic-tac-toe in player-vs-player or player-vs-AI modes.
- **User Authentication:** Enable users to register and manage their profiles securely.
- **Game History:** Store and retrieve user game histories, allowing users to review and replay past games.
- **AI Opponent:** Implement an AI opponent using the minimax algorithm with alpha-beta pruning for strategic gameplay.
- **GUI:** Provide an interactive and user-friendly graphical interface for game interaction and user management.

2.3 User Classes and Characteristics

- **Registered Users:** Users who can log in, manage their profiles, play games, and view game histories.
- **Guests:** Users who can play the game without authentication but cannot save game histories.

2.4 Operating Environment

- **Development Environment:** The game is developed using C++ and Qt framework for the GUI.
- **Target Platforms:** Windows
- **Database:** SQLite for storing user data and game histories.

2.5 Design and Implementation Constraints

- **Programming Language:** C++
- **GUI Framework:** Qt for C++
- **Database:** SQLite or similar lightweight database
- **Testing Framework:** Qtest from Qt framework

3. Functional Requirements

functional requirements specificities actions and behaviors that the software system must do to meet user needs. Each requirement typically includes a description of the functionality, and any conditions or constraints under which it applies.

3.1 User Authentication

- **Registration:** Users shall be able to register by providing a username and password.
- **Login:** Users shall be able to log in using their username and password.
- **Password Hashing:** Passwords shall be securely hashed before storage.
- **Profile Management:** Users shall be able to update their profile information.

3.2 Game Play

- **Player vs Player Mode:** The game shall allow two users to play against each other.
- **Player vs AI Mode:** The game shall allow a user to play against an AI opponent.
- **Turn Management:** The game shall alternate turns between the two opponents
- **Win/Tie Detection:** The game shall detect and announce when a player wins or if the game is a tie.

3.3 AI Opponent

- **Minimax Algorithm:** The AI opponent shall use the minimax algorithm with alpha-beta pruning.
- **Difficulty Levels:** The game shall offer different difficulty levels for the AI opponent.

3.4 Game History

- **Save Game History:** The system shall save the details of each game played by a registered user.
- **View Game History:** Users shall be able to view their past games.
- **Replay Past Games:** Users shall be able to replay their past games from the history.

3.5 GUI

- **Interactive Board:** The GUI shall display a 3x3 grid for the game.
- **User Interaction:** Users shall interact with the game via mouse clicks.
- **Session Management:** The GUI shall include forms for login, registration, and viewing game history.

4. Non-functional Requirements

4.1 Performance Requirements

- **Consistent Response time:** the system should not get slower after saving a lot of sessions and games per user.

4.2 Reliability Requirements

- **Resource Efficiency:** Ensure high number of saved sessions for one user and high number of saved games in for one session without consuming a lot of memory
- **Backup and Recovery:** Ensure saved user information are not lost randomly or due to high traffic from users in one session and that they could be retrieved safely with no errors.

4.3 Security Requirements

- **Authentication and Authorization:** Secure user authentication with a password hashing algorithm, ensuring unauthorized access is prevented.
- **Data Protection:** Hiding the password when signing in or changing password to prevent unwanted personal to see the password.

4.4 Usability Requirements

- **Accessibility and User Interface Design:** Design a graphical user interface (GUI) that provides clear feedback on game state changes and user actions.
- **Platform Support:** Ensure compatibility with Windows OS.
- **Hardware Support:** Ensure compatibility with different computer monitor sizes.

5. Appendices

5.1 Glossary

- **Minimax Algorithm:** A recursive algorithm used for decision-making and game theory.
- **Alpha-Beta Pruning:** An optimization technique for the minimax algorithm that reduces the number of nodes evaluated.
- **Hashing:** converting the password into a fixed-length string of characters using a cryptographic hash function making it difficult to reverse-engineer the original password from its hash.

5.2 Acronyms and Abbreviations

- **AI:** Artificial Intelligence
- **CI/CD:** Continuous Integration/Continuous Deployment
- **GUI:** Graphical User Interface

5.3 References

- Google C++ Style Guide: Google C++ Style Guide - GitHub
- GitHub Actions Documentation: https://docs.github.com/en/actions
- SQLite Documentation: SQLite Documentation
- YAML Official Website: YAML Tutorial: Everything You Need to Get Started in Minutes - CloudBees