Software Requirements Specification

for Advanced Tic Tac Toe

1. Introduction

1.1 Purpose

- The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive and detailed description of the functional and non-functional requirements for the Tic-Tac-Toe game application. This document serves as a foundational reference for developers, testers, and stakeholders to understand the system's intended behavior, constraints, and performance expectations. It aims to ensure that all parties involved in the development process have a clear and agreed-upon set of requirements to guide the design, implementation, testing, and deployment of the application. The SRS will facilitate effective communication, reduce ambiguity, and provide a basis for verifying that the final product meets the specified needs.

1.2 Scope

The Tic-Tac-Toe game is a desktop application developed using C++ with the Qt framework, targeting cross-platform compatibility across Windows, macOS, and Linux. The scope of this project includes the following key features:

- **User Authentication**: Support for user registration, login, and a guest mode, where registered users can save their game history, and guests cannot.
- **Gameplay Modes**: Implementation of player vs. player and player vs. AI game modes, with a basic AI opponent.
- **Game History**: Management of game history for registered users, including saving, loading, and replaying past games in a user-specific file.
- **User Interface**: A graphical user interface (GUI) with windows for login, game mode selection, gameplay, settings, and history viewing. The application is designed as an offline, single-user experience with no online multiplayer or cloud integration. Future enhancements, such as advanced AI difficulty levels or graphical improvements, are outside the current scope but may be considered in subsequent releases.

1.3 Definitions, Acronyms, and Abbreviations

- **Tic-Tac-Toe**: A two-player game played on a 3x3 grid, where players take turns marking cells with 'X' or 'O', aiming to align three of their symbols in a row, column, or diagonal to win, or resulting in a tie if the board fills without a winner.
- **AI**: Artificial Intelligence, referring to the computer-controlled opponent in the player vs. AI mode, implemented with a simple strategy (random moves).
- **SRS**: Software Requirements Specification, this document.

- **UI**: User Interface, the graphical components through which users interact with the application.
- **Guest Mode**: A temporary play option that allows users to play without registration, with no persistent data saved and no data history.

1.4 References

- **Qt Documentation**: https://doc.qt.io/qt-6/, the official reference for the Qt framework used in development.
- **IEEE 830-1998**: IEEE Recommended Practice for Software Requirements Specifications, providing the standard structure for this document.
- **Google C++ Style Guide**: https://google.github.io/styleguide/cppguide.html, the coding standard to be followed for the project's source code.
- **Tic-Tac-Toe Rules**: General knowledge of the game, as commonly understood in gaming culture.

1.5 Overview

This SRS document is organized into several sections to provide a complete specification of the Tic-Tac-Toe game. Following this introduction, **Section 2: Overall Description** provides the product perspective, functions, user characteristics, constraints, assumptions, and apportioning of requirements. **Section 3: Specific Requirements** details the external interface, functional, non-functional, and design constraints, forming the core of the system's requirements. **Section 4: Supporting Information** includes additional documentation, a glossary, and appendices for supplementary details. This structure ensures that all aspects of the project are covered, from high-level goals to specific implementation details, serving as a guide throughout the development lifecycle.

2. Overall Description

2.1 Product Perspective

The Tic-Tac-Toe game is a standalone desktop application built using C++ with the Qt framework, designed to operate offline without external system dependencies. It enhances the classic two-player game by incorporating user authentication, allowing both registered users and guests to play, and introduces a graphical user interface with multiple windows for seamless navigation. The application stores game history locally for registered users, using user-specific files, and includes a basic AI opponent for single-player mode. It builds on a modular design with distinct components for game logic, user management, and history tracking, aiming to provide a robust and user-friendly experience across Windows, macOS, and Linux platforms.

2.2 Product Functions

- **User Authentication**: Supports registration with a username and password, login for registered users to access their history, and a guest mode for temporary play without data persistence.
- **Game Mode Selection**: Provides options for player vs. player and player vs. AI modes, accessible through a dedicated window, with the ability to set player names or AI difficulty.
- **Gameplay**: Features a 3x3 grid where players or the AI alternate turns with 'X' or 'O', including move validation, win or tie detection, and real-time score tracking displayed on the game board.

- **Game History Management**: Saves game details (player names, moves, outcome, timestamp) for registered users with options to view, replay, or clear history.
- **Settings and Navigation**: Includes a settings window for account switching or history access, and a "Return" button to navigate back to the game mode selection without creating duplicate windows.

2.3 User Characteristics

The target audience includes casual gamers of varying ages and skill levels, from children to adults, with basic computer literacy and familiarity with Tic-Tac-Toe rules. Users are expected to navigate the GUI using mouse or keyboard inputs, with no need for advanced technical knowledge. Registered users, who value tracking their game history, contrast with guest users seeking a quick, commitment-free experience. The interface is designed to be intuitive, catering to both competitive players who track scores and casual players enjoying a simple pastime.

2.4 Constraints

The development and operation of the Tic-Tac-Toe game are constrained by:

- **Platform Support**: Requires Qt 6.x and must function consistently on Windows 10+, macOS 10.15+, and Ubuntu 20.04+.
- **Storage**: Limits history and user data to local files in the user's home directory, with no cloud or network storage.
- **Performance**: Must run efficiently on minimal hardware (1 GHz processor, 1 GB RAM, 100 MB storage), restricting AI complexity.
- **Coding Standards**: Adheres to the Google C++ Style Guide for maintainability and readability.
- **Offline Use**: Operates without internet, focusing on local functionality.

2.5 Assumptions and Dependencies

The project assumes and depends on the following:

- **Assumptions**:
    - Users have Qt installed and configured on their systems.
    - The file system allows read/write access to the home directory for history files.
    - Users typically provide valid inputs, though the system handles exceptions.
- **Dependencies**:
    - Relies on Qt 6.x for GUI and core functionality.

- Requires a C++ compiler (e.g., g++) and qmake for building.
- Assumes a stable desktop environment for consistent UI behavior.

2.6 Apportioning of Requirements

Requirements are divided between the initial release and potential future enhancements:

- **Initial Release**: Covers user authentication (login/register/guest), both game modes (player vs. player, player vs. AI with basic AI), gameplay with win/tie detection and score tracking, and history management for registered users. The AI uses a simple strategy, with history isolated per user and disabled for guests.
- **Future Versions**: May introduce advanced AI with difficulty levels (Easy, Medium, hard using Minimax), performance optimizations, depending on additional resources and feedback.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The application features a multi-window GUI with the following components:

- **Main Window**: Displays login, registration, and guest mode options with text fields for username and password, and buttons labeled "Login," "Register," and "Play as Guest."
- **Game Mode Window**: Presents buttons for "Player vs. Player" and "Player vs. AI," along with a "Settings" button on the left side.
- **Game Board**: A 3x3 grid of clickable buttons for gameplay, a "Reset" button, a "Return" button, a status label showing the current player, and a score label in the top right displaying player scores.
- **History Window**: Lists past games with a "Replay" button and a "Clear History" button, accessible only for registered users.
- **Settings Window**: Includes buttons for "Switch Account" and "History," with a layout supporting future additions like sound controls.
- All windows use Qt widgets (QPushButton, QLabel, QLineEdit) for consistency and responsiveness.

3.1.2 Hardware Interfaces

- **Minimum Requirements**: 1 GHz processor, 1 GB RAM, 100 MB free storage.
- **Recommended Requirements**: 2 GHz processor, 2 GB RAM for smoother performance.
- **Mouse and keyboard** for interaction
- No specialized hardware is required beyond standard desktop capabilities.

3.1.3 Software Interfaces

- **Framework**: Built with Qt 6.x for GUI and core functionality.
- **Operating Systems**: Compatible with Windows 10+, macOS 10.15+, and Ubuntu 20.04+.
- **Build Tools**: Requires qmake and a C++ compiler (e.g., g++).
- **File System**: Uses local file I/O for user data (registered_users.txt) and history files (tictactoe_<username>_history.json).

3.1.4 Communications Interfaces

- No network communication is supported; the application operates offline.

3.2 Functional Requirements

This section specifies the functionalities the Tic-Tac-Toe game must provide

3.2.1 User Authentication

- **FR1**: The system allows users to register with a unique username and password, hashing the password with SHA-256 before storing it in "registered_users.txt".
- ***Description*: Registration creates a new user entry, preventing duplicates.**
- **FR2**: The system shall enable users to log in with their registered username and password, directing them to the game mode window upon success.
- ***Description:* Failed login attempts display an error message**.
- **FR3**: The system should provide a guest mode option that allows gameplay without registration, with no game history saved or displayed.
- ***Description:* Guest mode sets an empty username, disabling history features**.

3.2.2 Gameplay

- **FR4**: The system shall support player vs. player and player vs. AI game modes, selectable from the game mode window.
- ***Description:* AI mode uses a basic random move strategy.**
- **FR5**: The system shall validate moves by rejecting placements on occupied cells and alternating turns between players or player and AI.
- ***Description:* Invalid moves trigger a status update, and turns switch automatically.**
- **FR6**: The system shall detect a win when three identical symbols ('X' or 'O') are aligned in a row, column, or diagonal, or declare a tie when the board is full, displaying a message box with the result.
- ***Description:* Win/tie detection updates scores and triggers a board reset.**
- **FR7**: The system shall track and display the score for both players in player vs. player mode and for the player vs. AI mode on a label in the top right of the game board.
- ***Description:* Scores increment on a win and reset with the board.**

3.2.3 Game History

- **FR8**: The system shall save game details (player names, game mode, outcome, moves, winner symbol, timestamp) for registered users in tictactoe_<username>_history.json after each game ends.
- ***Description*: Guests have no history saved; registered users' files are user-specific.**
- **FR9**: The system shall load and display the game history for the current registered user in the history window, with a replay option for past moves.
- ***Description:* Displays "Please create an account to save your played games" for guests.**
- **FR10**: The system shall allow the current registered user to clear their game history via a button in the history window.
- ***Description:* *Clearing resets the history file to empty.***

3.2.4 Settings and Navigation

- **FR11**: The system shall provide a settings window accessible from the game mode window, with options to switch accounts or view history.
- ***Description:* Switching accounts returns to the main window; history opens the history window.**
- **FR12**: The system shall allow users to return to the game mode window from the game board using a "Return" button, reusing the existing instance to prevent duplication.
- ***Description*: Hides the game board and shows the parent game mode window.**

3.3 Non-Functional Requirements

3.3.1 Performance Requirements

- **NFR1**: The system shall process and display a player or AI move within 500 milliseconds on minimum hardware.
- ***Description*: Ensures smooth gameplay, measured during testing.**
- **NFR2**: The system shall load the main window within 2 seconds on a 1 GHz processor with 1 GB RAM.
- ***Description*: Targets quick startup for user satisfaction.**

3.3.2 Security Requirements

- **NFR3**: The system shall hash passwords with SHA-256 before storing them in registered_users.txt.
- ***Description*: Protects user credentials from plain-text exposure.**
- **NFR4**: The system shall restrict access to tictactoe_<username>_history.json files to the respective user, stored in the home directory.
- **Description: Prevents unauthorized history access.**

3.3.3 Usability Requirements

- **NFR5**: The system shall provide an intuitive interface with clear labels (e.g., "Player vs. AI") and real-time status updates (e.g., current player).
- ***Description*: Enhances user experience for all skill levels.**
- **NFR6**: The system shall support mouse and keyboard input for all interactive elements.
- ***Description*: Offers flexible control options.**

3.3.4 Reliability Requirements

- **NFR7**: The system shall handle invalid inputs (e.g., empty login fields) with error messages and maintain stability.
- ***Description*: Prevents crashes and informs users.**
- **NFR8**: The system shall ensure no data loss during history save operations, logging errors if writes fail.
- ***Description*: Guarantees history integrity.**

3.3.5 Maintainability Requirements

- **NFR9**: The system shall use a modular design with separate classes (GameBoard, GameHistory) following the Google C++ Style Guide.
- ***Description*: Facilitates future updates and debugging.**

3.4 Design Constraints

- The application must be developed using Qt 6.x and adhere to the Google C++ Style Guide for code consistency.
- All UI elements must be implemented with Qt widgets, avoiding external libraries.
- History storage is limited to local JSON files, with no database integration.

3.5 Software System Attributes

- **Availability**: Operates offline with 100% availability on supported platforms.
- **Maintainability**: Modular code structure allows for easy bug fixes and feature additions.
- **Portability**: Designed to run on Windows, macOS, and Linux with minimal adjustments.

4. Supporting Information

4.1 Documentation

This section references additional documents that support the development and testing of the Tic-Tac-Toe game:

- **Software Design Specification (SDS)**: Details the architecture, including UML class and sequence diagrams, and design decisions for the Qt-based GUI, game logic, and history management. This document will be developed following the completion of the SRS and will reference the functional requirements (FR1-FR12).

- **Testing Documentation**: Includes test plans, unit test cases using Google Test, integration test results, and code coverage reports. This will validate requirements such as move validation (FR5), history saving (FR8), and performance (NFR1).
- **CI/CD Pipeline Configuration**: Specifies the GitHub Actions workflow for building, testing, and deploying the application across Windows, macOS, and Linux, ensuring continuous integration and deployment as outlined in the project plan.

4.2 Glossary

This glossary defines key terms used throughout the SRS to ensure consistent understanding:

- **Move**: A player's action of placing an 'X' or 'O' on a specific cell of the 3x3 game board, recorded as a pair of row and column indices.
- **Session**: A single instance of a Tic-Tac-Toe game, encompassing player names, moves, outcome (win or tie), timestamp, and winner symbol, saved as a GameSession structure for history.
- **AI Difficulty**: A setting in player vs. AI mode that determines the AI's strategy ("Easy" for random moves), with potential future levels like "Medium" or "Hard."
- **Guest Mode**: A play option that allows users to start a game without registration, disabling history features and using an empty username.
- **History File**: A user-specific JSON file (e.g., tictactoe_<username>_history.json) storing all past game sessions for registered users.

4.3 Appendix

This appendix includes supplementary materials to enhance the SRS:

- **Sample UI Mockups**: Text-based representations of key windows (to be replaced with actual images or screenshots post-development):
- **Main Window**: Displays "Username:" and "Password:" fields, "Login," "Register," and "Play as Guest" buttons, and a status label.
- **Game Mode Window**: Shows "Player vs. Player" and "Player vs. AI" buttons, with a "Settings" button on the left.
- **Game Board**: Features a 3x3 grid of buttons, "Reset" and "Return" buttons, a status label, and a score label.
- **History Window**: Lists game history with "Replay" and "Clear History" buttons, and a replay grid.
- **Settings Window**: Includes "Switch Account" and "History" buttons.