Software Requirements Specification (SRS) for Tic-Tac-Toe Game

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Tic-Tac-Toe game application. The purpose of this SRS is to provide a comprehensive overview of the functionality, performance, and design constraints of the system. It is intended for stakeholders, developers, and testers involved in the project.

1.2 Scope

The Tic-Tac-Toe game is a classic two-player game played on a 3x3 grid. This application aims to provide an interactive and engaging user experience, allowing players to compete against each other or against an AI opponent. The scope of this document covers the features, user interfaces, and system functionalities required for a complete and enjoyable gaming experience.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **SDS**: Software Design Specification
- **UI**: User Interface
- **AI**: Artificial Intelligence
- **Tic-Tac-Toe**: A game played on a 3x3 grid where two players take turns marking the spaces in a 3x3 grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

1.4 References

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

1.5 Overview

This document is organized into several sections. Section 1 provides an introduction to the SRS, including its purpose, scope, definitions, references, and an overview of the document structure. Section 2 describes the overall description of the software, including product perspective, product functions, user characteristics, and general constraints. Section 3 details the specific requirements, covering functional requirements, non-functional requirements, and design constraints.

2. Overall Description

2.1 Product Perspective

The Tic-Tac-Toe game is a standalone application designed to run on a desktop environment. It does not depend on any other external software systems for its core functionality. However, it may interact with a local database for user authentication, game history, and statistics. The application will provide a graphical user interface (GUI) for user interaction.

2.2 Product Functions

The primary functions of the Tic-Tac-Toe game include: * **User Management**: Allow users to register, log in, and manage their profiles. * **Game Modes**: Support single-player (Player vs. AI) and multi-player (Player vs. Player) game modes. * **Gameplay**: Enable players to make moves, validate moves, detect win/draw conditions, and reset the game. * **AI Difficulty**: Provide different difficulty levels for the AI opponent (e.g., Easy, Medium, Hard). * **Game History and Statistics**: Record game outcomes and display player statistics. * **User Interface**: Provide an intuitive and responsive graphical user interface.

2.3 User Characteristics

- **Casual Gamers**: Users who enjoy playing simple and quick games.
- **New Users**: Users who are new to the application and need a straightforward registration and login process.
- **Experienced Users**: Users who are familiar with the game and may be interested in advanced features like statistics and different AI difficulties.

2.4 General Constraints

- **Operating System**: The application is developed using Qt framework, which supports cross-platform deployment. However, the current build environment

suggests a Windows-based development (MinGW_64_bit). The target operating system for deployment will be specified later.
- **Database**: A local database (e.g., SQLite) will be used for storing user data and game statistics.
- **Performance**: The game should respond to user inputs promptly, and AI moves should be calculated within a reasonable time frame.
- **Security**: User credentials should be stored securely. (Though for a simple Tic-Tac-Toe game, this might be a lower priority).

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Management

- **FR1.1 Registration**: The system shall allow new users to register by providing a unique username and password. The system shall validate the uniqueness of the username.
- **FR1.2 Login**: The system shall allow registered users to log in using their username and password. The system shall authenticate user credentials.
- **FR1.3 Profile Management**: The system shall allow logged-in users to view and update their profile information (e.g., change password).

3.1.2 Game Modes

- **FR2.1 Player vs. Player**: The system shall allow two human players to play Tic-Tac-Toe against each other on the same device.
- **FR2.2 Player vs. AI**: The system shall allow a human player to play Tic-Tac-Toe against an AI opponent.

3.1.3 Gameplay

- **FR3.1 Game Board Display**: The system shall display a 3x3 Tic-Tac-Toe game board.
- **FR3.2 Player Turn Management**: The system shall clearly indicate whose turn it is.
- **FR3.3 Move Input**: The system shall allow players to make moves by clicking on an empty cell on the game board.
- **FR3.4 Move Validation**: The system shall validate each move to ensure it is made on an empty cell.
- **FR3.5 Win Condition Detection**: The system shall detect when a player has achieved three of their marks in a horizontal, vertical, or diagonal row.

- **FR3.6 Draw Condition Detection**: The system shall detect when the game ends in a draw (all cells filled, no winner).
- **FR3.7 Game Reset**: The system shall allow players to reset the game after a win or draw.

3.1.4 AI Difficulty (for Player vs. AI mode)

- **FR4.1 Easy AI**: The system shall provide an 'Easy' AI opponent that makes random valid moves.
- **FR4.2 Medium AI**: The system shall provide a 'Medium' AI opponent that attempts to block immediate player wins and makes some strategic moves.
- **FR4.3 Hard AI**: The system shall provide a 'Hard' AI opponent that plays optimally, aiming to win or draw every game.

3.1.5 Game History and Statistics

- **FR5.1 Game History Storage**: The system shall store the outcome of each game (win, loss, draw) for each player.
- **FR5.2 Player Statistics Display**: The system shall display player statistics, including total games played, wins, losses, and draws.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- **NFR1.1 Response Time**: The system shall respond to user input (e.g., clicking a cell) within 100 milliseconds.
- **NFR1.2 AI Move Time**: The AI opponent shall make a move within 500 milliseconds, regardless of difficulty level.

3.2.2 Security Requirements

- **NFR2.1 Password Storage**: User passwords shall be stored in a hashed and salted format.
- **NFR2.2 Authentication**: The system shall prevent unauthorized access to user accounts.

3.2.3 Usability Requirements

- **NFR3.1 Intuitive Interface**: The user interface shall be intuitive and easy to navigate for new users.
- **NFR3.2 Clear Feedback**: The system shall provide clear visual feedback for valid and invalid moves, as well as game outcomes.

3.2.4 Reliability Requirements

- **NFR4.1 Data Integrity**: The system shall ensure the integrity of game history and user profile data.
- **NFR4.2 Error Handling**: The system shall handle unexpected inputs or errors gracefully, providing informative messages to the user.

3.2.5 Maintainability Requirements

- **NFR5.1 Modularity**: The codebase shall be modular, allowing for easy modification and extension of features.
- **NFR5.2 Readability**: The code shall be well-commented and follow consistent coding standards.

3.2.6 Portability Requirements

- **NFR6.1 Cross-Platform Compatibility**: The application shall be capable of running on different operating systems (e.g., Windows, macOS, Linux) with minimal modifications.

3.3 Design Constraints

- **DC1.1 Programming Language**: The application shall be developed using C++.
- **DC1.2 Framework**: The application shall utilize the Qt framework for GUI development.
- **DC1.3 Database System**: The application shall use SQLite for local data storage.