**Software requirements Specification SRS**

**1. Introduction**

**1.1 Purpose:** The purpose of this document is to define all functional and non-functional requirements for the TicTacToe game, a desktop application developed using Qt. This SRS serves as a reference for developers, testers, and stakeholders to ensure the game meets user expectations and project goals.

**1.2 Scope:** The Advanced Tic Tac Toe Game allows two players or a player and an AI to compete on a 3x3 grid. It includes a secure user authentication system, a game history feature for reviewing past games, and an intelligent AI opponent powered by a minimax algorithm. The system uses SQLite for data management and GitHub Actions for CI/CD.

**1.3 Definitions** Player: A human user or AI, represented as X or O. Game Board: A 3x3 grid where players place their symbols.Auth: The authentication system for user registration and login. History: The system for saving and retrieving match results.

**2. Overall Description**

**2.1 Product Perspective:** TicTacToe is a desktop application built with Qt 6.9.1 and C++17, designed to run on Windows. It offers an interactive gaming experience with a simple interface for playing the classic X O game.

**2.2 Target Audience:** Casual gamers who enjoy strategy games. Developers learning to build applications with Qt.

**2.3 Assumptions** Users have Windows 10 or 11 installed. Qt 6.9.1 and MinGW 64-bit are available for development and execution.

**3. Functional Requirements**

**3.1 User Registration**

FR1: Users shall register a new account by providing a unique username and password.

FR2: The system shall store user credentials in a SQLite database (users.db).

**3.2 User Login**

FR3: Users shall log in by entering their username and password.

FR4: The system shall validate credentials and display an error message for invalid inputs.

**3.3 Game Mode Selection**

FR5: Users shall choose between "Player vs. AI" or "Player vs. Player" modes.

FR6: The system shall open the appropriate game window (MainWindow for AI mode or PlayerVsPlayerWindow for player mode).

**3.4 Play vs. AI**

FR7: Users shall play against an AI opponent using the Minimax algorithm.

FR8: The system shall display the game outcome (win, loss, or draw).

**3.5 Play vs. Player**

FR9: Two players shall take turns playing on the same device.

FR10: The system shall detect and announce the winner or a draw.

**3.6 Game History**

FR11: The system shall save match results in a text file (history\_username.txt) for each user.

FR12: Users shall view their match history.

**4. Non-Functional Requirements**

**4.1 Performance**

NFR1: The game shall respond to user moves in less than 0.5 seconds.

NFR2: The AI shall select a move in less than 1 second.

**4.2 Security**

NFR3: Passwords shall be stored encrypted in the SQLite database.

NFR4: Match history files shall be protected from unauthorized modifications.

**4.3 Usability**

NFR5: The user interface shall be simple and intuitive.

NFR6: Buttons shall be at least 100x100 pixels for easy interaction.

**4.4 Compatibility**

NFR7: The game shall run on Windows 10 or 11 with Qt 6.9.1.

**5. User Interface**

**5.1 Startup Window**

Contains two buttons: "Login" and "Register".

**5.2 Login Window**

Includes fields for username and password, and a "Login" button.

**5.3 Game Mode Window**

Offers two buttons: "Player vs. AI" and "Player vs. Player".

**5.4 Game Window**

Displays a 3x3 grid for gameplay, a "New Game" button, and the game status (win/loss/draw).

**6. Assumptions and Constraints**

**6.1 Assumptions**

Users have basic knowledge of operating Windows applications.

**6.2 Constraints**

The game does not support online multiplayer.

The SQLite database supports a maximum of 1000 user accounts.