

# 1. Introduction

## 1.1 Purpose

This Software Requirements Specification (SRS) defines the key requirements for a cross-platform Tic Tac Toe application. It outlines both functional and non-functional needs to guide the development of a modern, user-friendly game with classic and enhanced features. Serving as a central reference, the SRS ensures that all stakeholders—developers, testers, and users—share a clear understanding of the system's objectives and limitations.

## 1.2 Scope

This project covers the design and implementation of a feature-rich Tic Tac Toe application with the following capabilities:

- Multiple game modes: Player vs Player (PvP) and Player vs AI (PvAI) with selectable AI difficulty levels.
- Secure user authentication, registration, and guest mode for instant play.
- Persistent match history with replay functionality, including series-based gameplay (e.g., best-of-N format).
- Customizable user interface with a wide selection of visual themes.
- Local database storage for user accounts and match results.
- Support for account management features such as account deletion and logout.

The application is intended to run on desktop platforms and aims to provide an accessible and engaging experience for all users

## 1.3 Definitions

- Player vs Player (PvP): Two human players compete against each other.
- Player vs AI (PvAI): A human player competes against the computer, with selectable difficulty (Easy, Medium, Hard).
- Series: A set of games played in succession, where the first player to reach a specified number of wins is the overall winner.
- Match History: A record of completed games, including player names, results, and timestamps, available for review and replay.
- Theme: A selectable visual style for the application's interface.
- Guest Mode: Allows users to play without registration, with limited access to persistent features.
- Database: Local storage system used for saving user accounts and match data.
- Authentication: The process of verifying user identity via username and password.

## 1.4 Intended Audience

This SRS is intended for:

- Developers: To guide the implementation of the application according to the specified requirements.
- Testers: To verify that the application meets all functional and non-functional requirements.
- Project Stakeholders: To understand the application's features, constraints, and objectives.
- End Users: To gain insight into the capabilities and options provided by the Tic Tac Toe application

## 2. Overall Description

### Product Perspective

The Tic Tac Toe application is a standalone, cross-platform game designed to offer both classic and advanced gameplay experiences. It can be played on any device with a compatible operating system, such as Windows, macOS, or Linux. The application is self-contained, requiring no external dependencies beyond standard system libraries and a local SQLite database for user accounts and match history storage. The game supports both local multiplayer (Player vs Player) and single-player modes against an AI opponent with selectable difficulty levels. The interface is designed to be intuitive and visually customizable through a variety of themes.

### Product Features

- Classic 3x3 Tic Tac Toe gameplay with alternating turns for X and O.
- Two main game modes: Player vs Player (PvP) and Player vs AI (PvAI), with three AI difficulty levels (Easy, Medium, Hard).
- User authentication system supporting registration, login, and guest access.
- Persistent match history tracking, including series-based gameplay (best-of-N format) and replay functionality.
- Customizable user interface with multiple visual themes.
- Scoreboard displaying wins, losses, and ties for each player.
- Secure local storage of user data and match results.

### Operating Environment

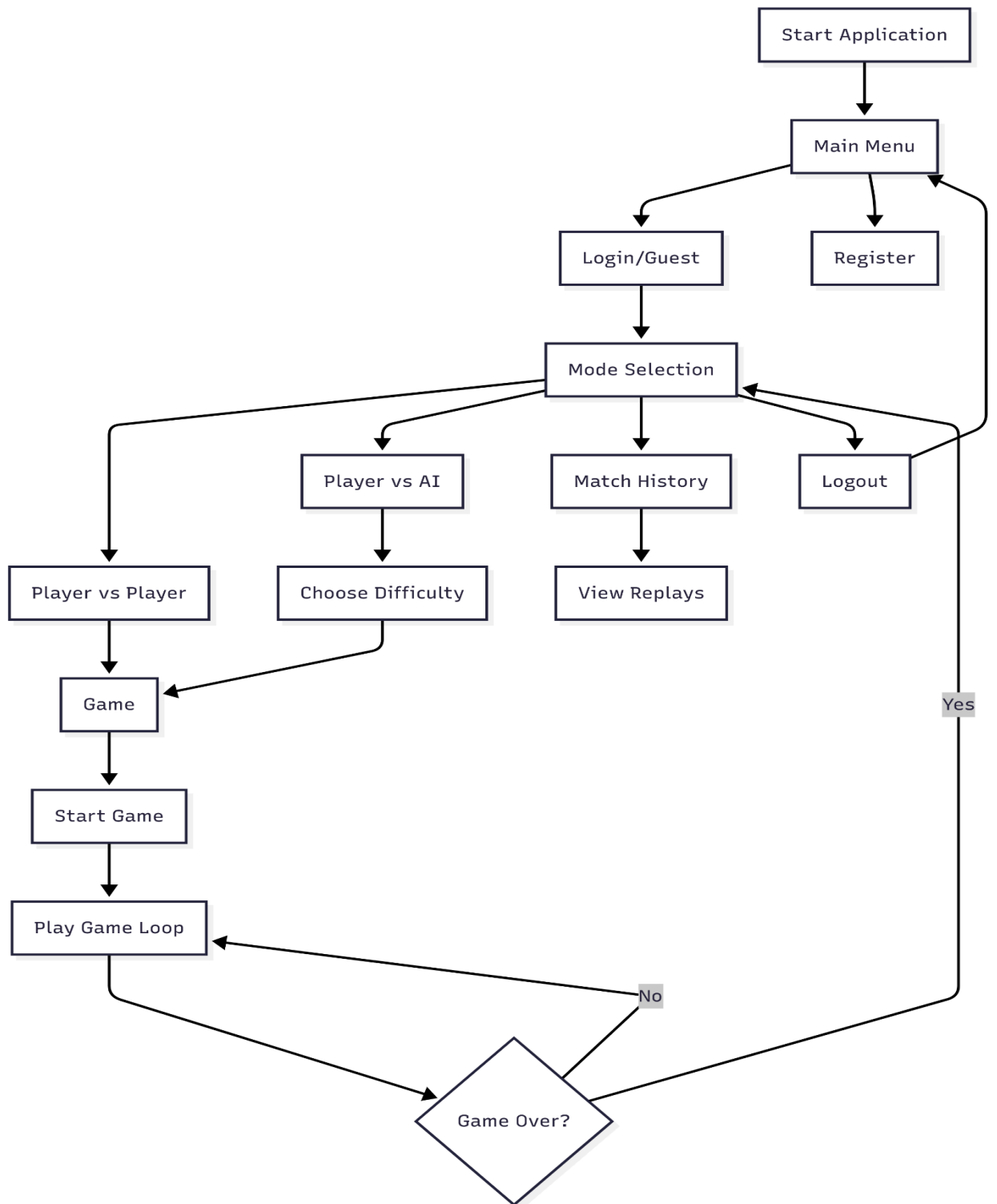
- The application runs on desktop operating systems (Windows, macOS, Linux) and requires basic hardware resources.
- No internet connection is required for core gameplay, as all data is stored locally via SQLite.
- The user interface is built using Qt, ensuring a responsive and visually consistent experience across platforms.

### Design and Implementation Constraints

- The game must be easy to use, with clear instructions and accessible controls.
- Efficient use of memory and processing power is required, especially for AI computations at higher difficulty levels.
- The design must support extensibility for adding new features, themes, or modes in the future .

## Assumptions and Dependencies

- Users have access to a compatible desktop device with the required operating system and basic hardware.
- The application relies on standard Qt libraries and SQLite for its database functionality.
- All user data, including authentication details and match history, is stored locally and is not shared externally



## System Architecture

### 3. Functional Requirements

#### 1. User Authentication

- Technical Implementation:
  - SQLite database stores `users` table with `username`, `password_hash`, and `salt` fields
  - PBKDF2 with SHA-256 hashing (10000 iterations) for password security
  - Guest mode handled via `guestMode` flag with limited persistence
  - Login screen UI built with `QLineEdit` and `QPushButton` widgets
  - Account deletion cascades to match history removal

#### 2. Game Modes

- Technical Implementation:
  - Mode selection via `QStackedWidget` navigation
  - `mode` variable (1=PvP, 2=PvAI) controls game logic branching
  - AI difficulty levels (1-3) stored in `difficulty` variable

#### 3. Game Settings

- Technical Implementation:
  - `QSpinBox` widgets for numeric input of games/series
  - Validation prevents `gamesToWin > totalGames`
  - Settings persisted via `totalGames` and `gamesToWin` variables

#### 4. Gameplay Mechanics

- Technical Implementation:
  - 9-element `vector<char>` represents 3x3 board
  - Win/draw detection via `checkWin()` and `checkTie()` algorithms
  - Turn management through `currentPlayer` toggle
  - Move validation prevents overwriting occupied cells

#### 5. AI Behavior

- Technical Implementation:
  - Difficulty-based strategy selection:
    - Easy: Avoids immediate loss/win (neutral moves)
    - Medium: Mix of minimax and random moves
    - Hard: Optimal minimax algorithm
  - First-move optimization for corners/center

## 6. Score and Series Management

- Technical Implementation:
  - `player1Wins`, `player2Wins`, `ties` counters
  - Series ID generation using timestamp + random suffix
  - Automatic series termination when win threshold reached

## 7. Match History and Replay

- Technical Implementation:
  - `matches` table stores move sequences, timestamps, and results
  - Replay system uses `replayMoves` vector and `replayIndex` counter
  - Move-by-move reconstruction with original starting player

## 8. User Interface

- Technical Implementation:
  - Qt's `QStackedWidget` for screen management
  - Dynamic theme switching via `applyStyleSheet()`
  - Game board as grid of `QPushButton` objects
  - Scoreboard toggle with `scoreboardVisible` flag

## 9. Data Persistence & Security

- Technical Implementation:
  - SQLite local database with `users` and `matches` tables
  - Password hashing via PBKDF2-SHA256 with unique salts
  - Series/game metadata stored in `matches` table

## 10. Additional Features

- Technical Implementation:
  - Surrender handling with emoji-based result tracking
  - Guest mode limitations enforced via `guestMode` checks
  - Theme persistence through `selectedTheme` variable
  - Scoreboard visibility toggle

## 4. Non Functional Requirements

Performance:

The application responds to user actions and AI moves rapidly, with AI (even on hard mode) making decisions in under 2 seconds. UI updates and button responses are immediate, ensuring a smooth experience.

- **Usability:**  
The interface is intuitive, with clear navigation between login, game modes, settings, and match history. Visual feedback for errors and a wide selection of dynamic themes make the app accessible and visually appealing. Scoreboard toggling and clear status messages enhance user understanding.
- **Reliability:**  
The system robustly handles all game states (win, tie, surrender, series end) and prevents invalid actions (e.g., moves in occupied cells or during replay). All critical scenarios are covered by automated tests to ensure correct operation.
- **Security:**  
User passwords are securely stored using PBKDF2 with SHA-256 and unique salts. Authentication and account deletion are handled safely, and all sensitive operations are confirmed or error-checked.

## 5. Interface Requirements

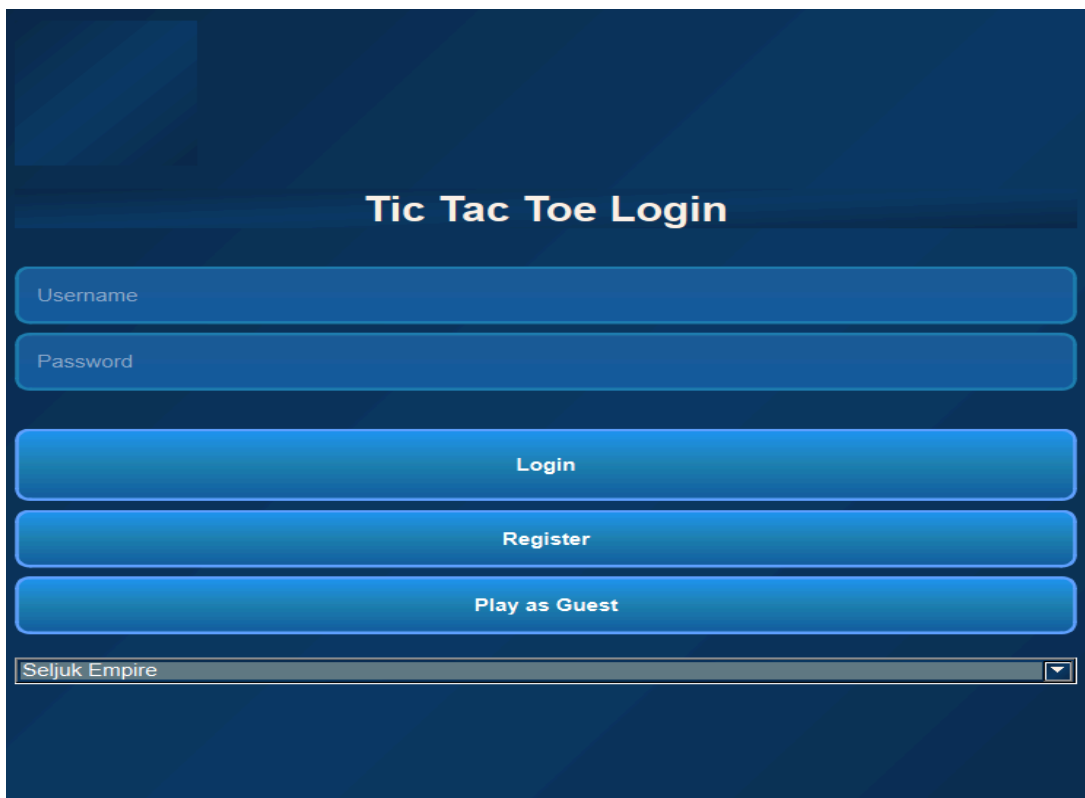
### Main Screens and Components

- **Login Screen**
  - Username and password input fields (`QLineEdit`)
  - Buttons for Login, Register, and Play as Guest
  - Theme selector (`QComboBox`) for choosing from a wide range of visual styles
  - Application logo and title for branding and orientation
- **Mode Selection Screen**
  - Large, clearly labeled buttons for selecting Player vs Player or Player vs AI modes
  - Button to view match history
  - Logout and Delete Account options (visible only for logged-in users)
- **Difficulty Selection Screen (for PvAI)**
  - Buttons for Easy, Medium, and Hard AI difficulty levels
  - Back button for returning to mode selection
- **Player Name Input (for PvP)**
  - Input field for Player 2's name
  - Start Game button
  - Back button to return to settings
- **Game Settings Screen**

- Spin boxes (`QSpinBox`) for configuring total games in a series and games needed to win
- Apply Settings button
- Back button to return to mode selection
- Game Board Screen
  - 3x3 grid of large, responsive buttons representing the board cells
  - Status label showing whose turn it is or the game result
  - Scoreboard toggle button to show/hide current series scores
  - Surrender and Logout buttons
  - Back button to return to mode selection
- Match History Screen
  - Table displaying previous matches with columns for date, players, winner, and result
  - Button to access recorded matches for replay
  - Back button to return to mode selection
- Recorded Match Playback Screen
  - Table listing all recorded matches with metadata
  - Button to watch/replay a selected match move-by-move
  - Back button to return to mode selection

## 6. Appendices

### 6.1 Login



**Tic Tac Toe Login**

Username

Password

Login

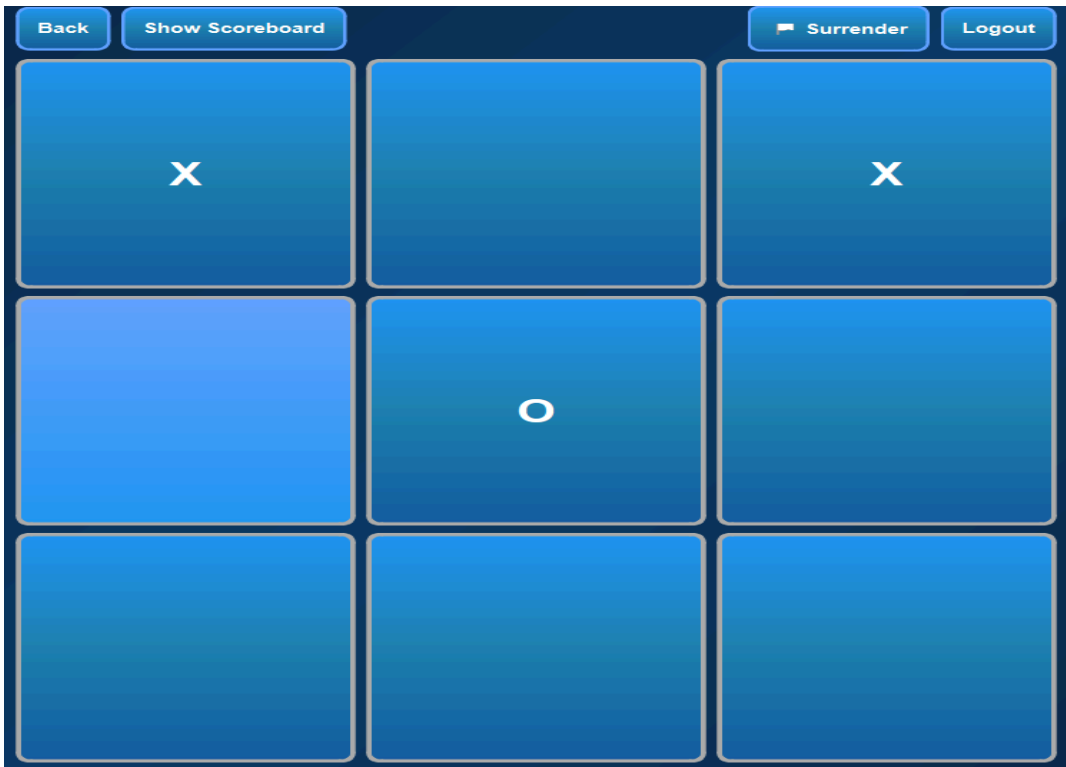
Register

Play as Guest

Seljuk Empire ▼



6.2 Game Board



6.3 History

### Match History

	Date	Player 1	Player 2	Winner	Result
1	2025-06-20T12:0...	amr	Player 2	amr	🏆 Win (Game 1/...
2	2025-06-20T12:0...	AI	amr	-	🤝 Tie (Game 1/...
3	2025-06-20T11:2...	AI	amr	AI	🏆 AI Victory ...
4	2025-06-20T11:2...	AI	amr	-	🤝 Tie (Game 2/...
5	2025-06-20T11:2...	AI	amr	-	🤝 Tie (Game 1/...
6	2025-06-20T09:1...	AI	amr	-	🤝 Tie (Game 3/...
7	2025-06-20T09:1...	AI	amr	AI	🏆 AI Victory ...
8	2025-06-20T09:1...	AI	amr	-	🤝 Tie (Game 1/...
9	2025-06-20T09:1...	AI	amr	AI	🚩 Surrender ...

Replay Recorded Matches

Back