

Software Design Specification (SDS) For Advanced Tic Tac Toe Game

Supervised by: Dr/ Omar Nasr

1. Introduction:

1.1 Purpose

This document outlines the design, architecture, and functionality of an advanced Tic Tac Toe game using Qt and C++. The purpose of this game is to provide an engaging and interactive experience with features such as AI opponents, personalized game history, and user authentication.

1.2 Scope

The advanced Tic Tac Toe game is designed for players of all skill levels and offers single-player (against AI) and multiplayer modes. It includes the following features:

- AI opponent with adjustable difficulty levels.
- User authentication and registration.
- Game history and personalized statistics.
- Intuitive graphical user interface (GUI).

2. System Overview:

1. User Authentication

- User registration with secure password handling.
- Login functionality with error handling for invalid credentials.

2. AI Opponent

- Adjustable difficulty levels: Easy, Medium, and Hard.
- AI behaviour based on strategic algorithms (Minimax).

3. Game History

- Stores and retrieves game outcomes and statistics per user.
- Tracks win/loss ratio and most-played game modes.

4. Graphical User Interface

- The GUI must display:
 - The game board.
 - Notifications for invalid moves.
 - Game state (e.g., ongoing, win, draw).
- Buttons for starting new games, changing settings, and logging out.

5. Multiplayer Mode

- Local two-player mode with alternating turns.

6. Tools and Technologies

- **Programming Language:** C++
- **Libraries:**
 - **GUI:** SFML or Qt for graphical interface.
 - **Database:** SQLite for managing user and game data.
 - **Security:** crypt for password hashing.
- **Development Environment:** Visual Studio Code or Qt Creator.

3. Minimax Algorithm:

The **Minimax Algorithm** is a decision-making strategy used in two-player games like Tic Tac Toe. It ensures that the AI opponent plays optimally by considering all possible moves and their outcomes, maximizing its chance of winning while minimizing the player's chance to win.

How It Works

1. **Tree Representation:** The algorithm constructs a game tree where:
 - Each node represents a possible state of the game.
 - Each edge represents a move by the player or the AI.

2. Scoring:

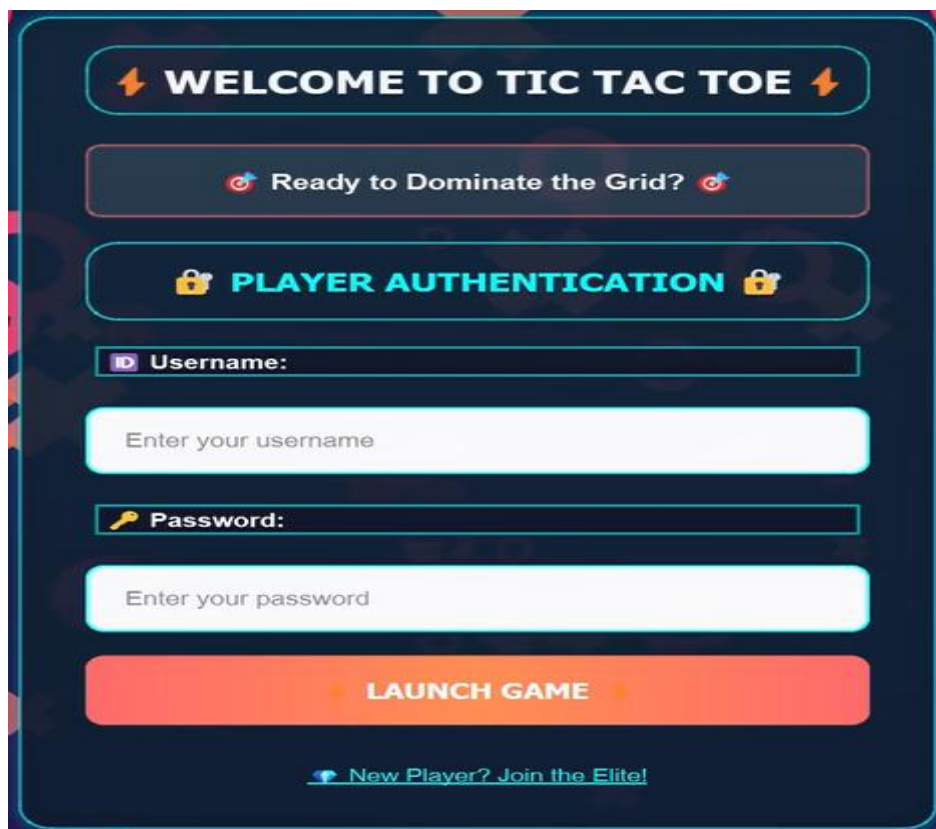
- A win for the AI is assigned a score of +10.
- A loss for the AI is assigned a score of -10.
- A draw is assigned a score of 0.

3. Decision Making:

- The algorithm simulates all possible moves.
- For each move, it recursively evaluates the potential future moves of the opponent.
- The AI chooses the move with the highest minimax value.

4. Architectural design:

1. Login/Registration Screen: The login screen ensures secure access for users, with options for new users to register.



⚡ **WELCOME TO TIC TAC TOE** ⚡

🎮 Ready to Dominate the Grid? 🎮

🔒 **PLAYER AUTHENTICATION** 🔒

ID Username:

Enter your username

🔑 Password:

Enter your password

🔥 **LAUNCH GAME** 🔥

👤 [New Player? Join the Elite!](#)

Figure 1 Login/Registration Screen

2. Game History: The history screen showcases user statistics, including win/loss records and game details.

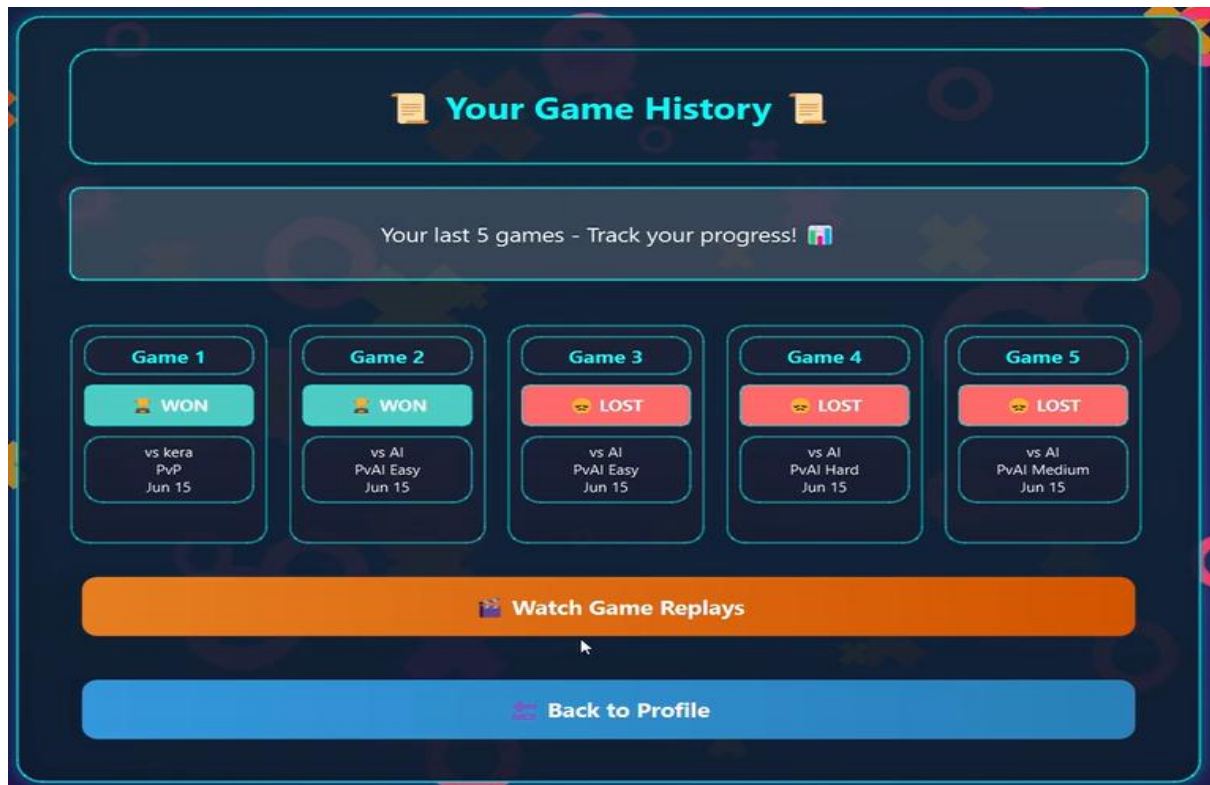


Figure 2 Game History

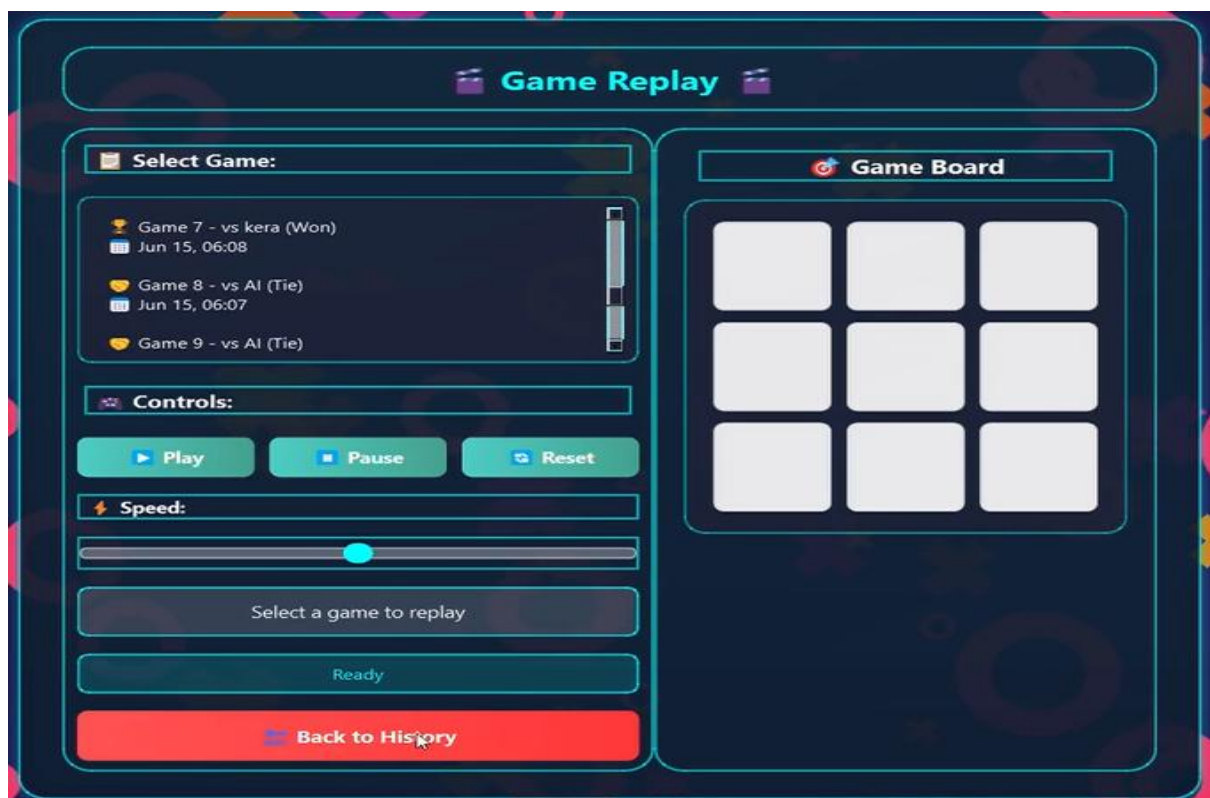


Figure 3 Game details and replay

3. Game window: The main game window serves as the core interface where players interact with the Tic Tac Toe board. It visually represents the board's state, including player and AI moves, and provides feedback on the game's progress.

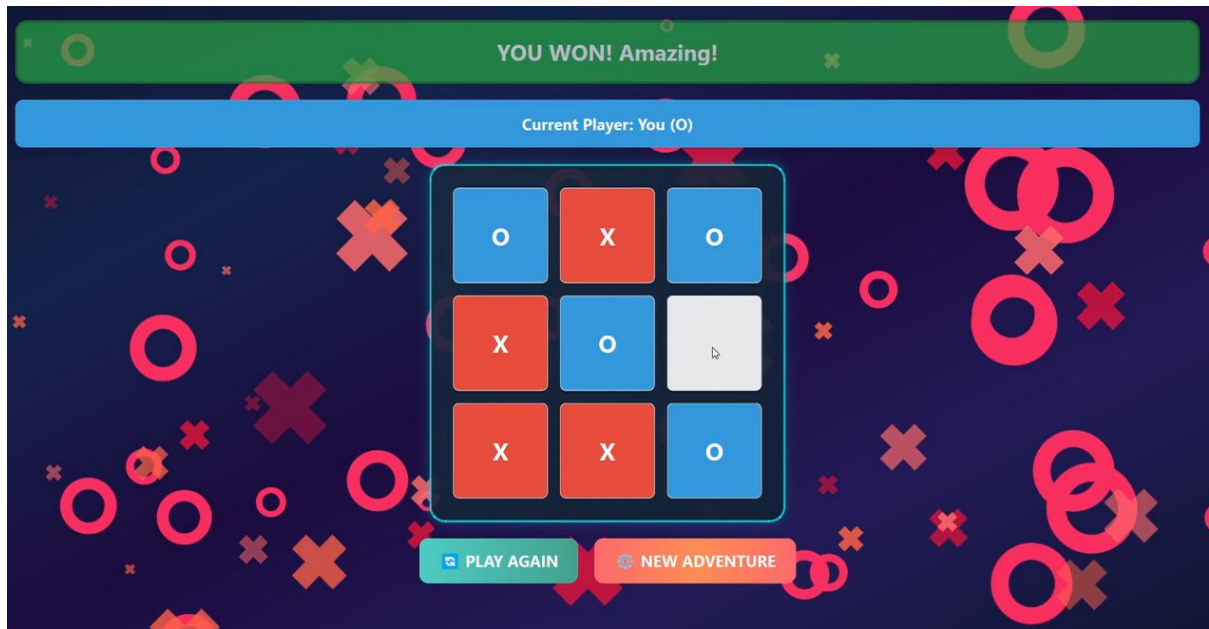


Figure 4 Main Game Window

5. Sequence Diagrams:

1. Login Screen:

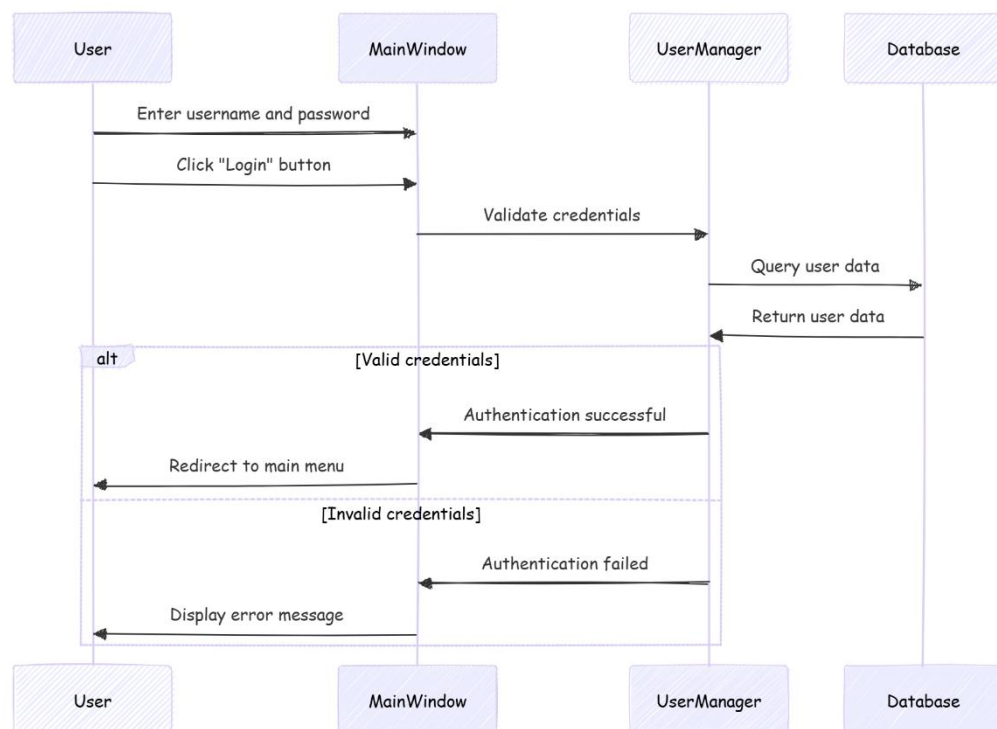


Figure 5 Login Screen Diagram

2. Main Window:

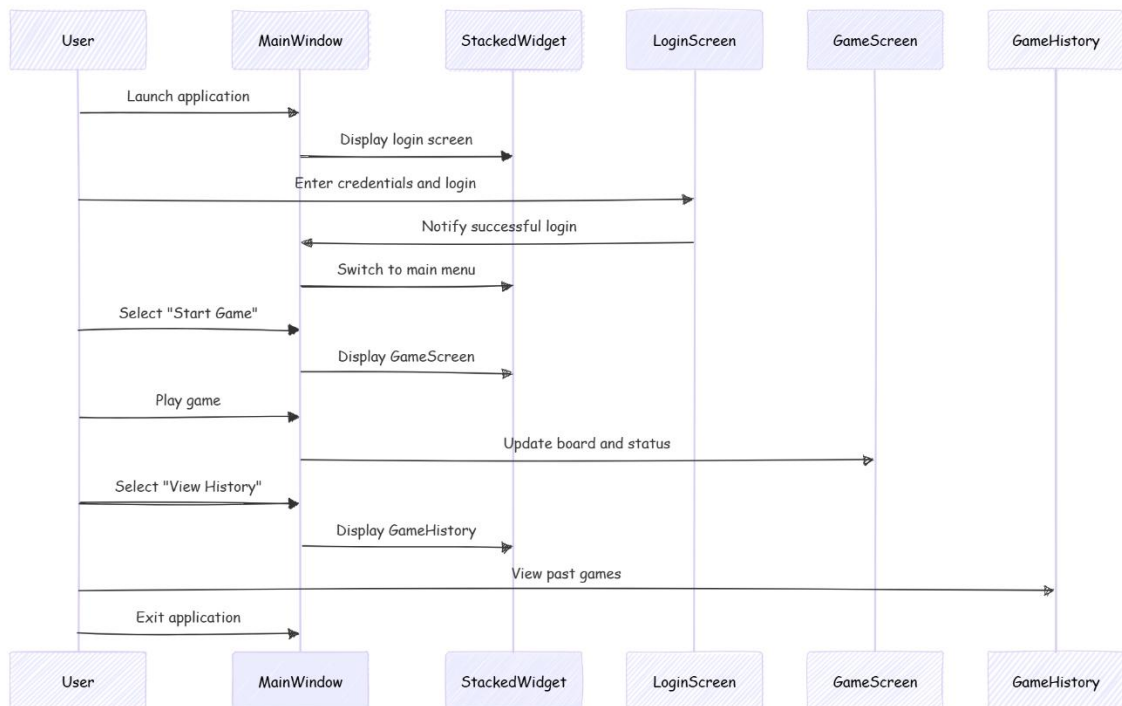


Figure 6 Main Window Diagram

3. Game History:

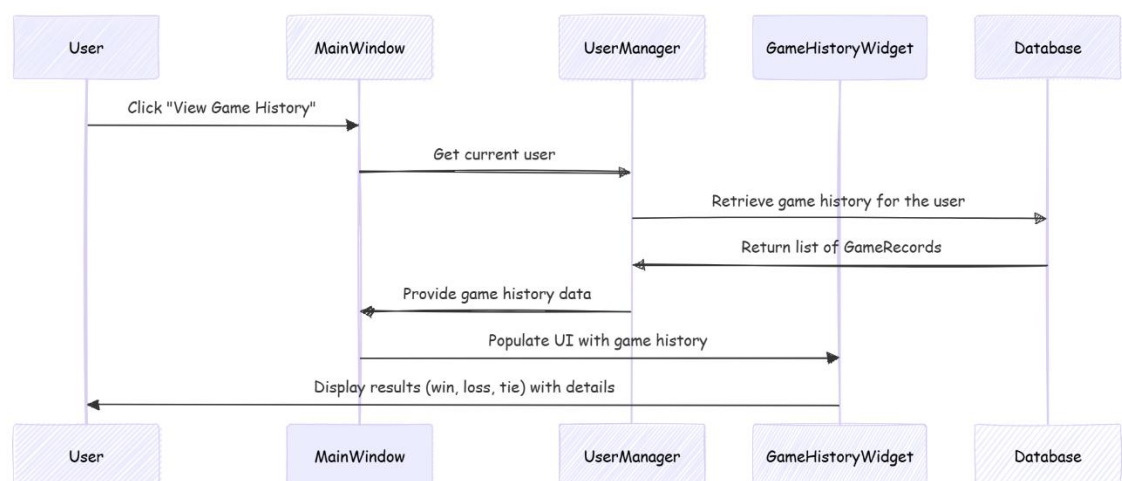


Figure 7 Game History Diagram

4. Game Logic:

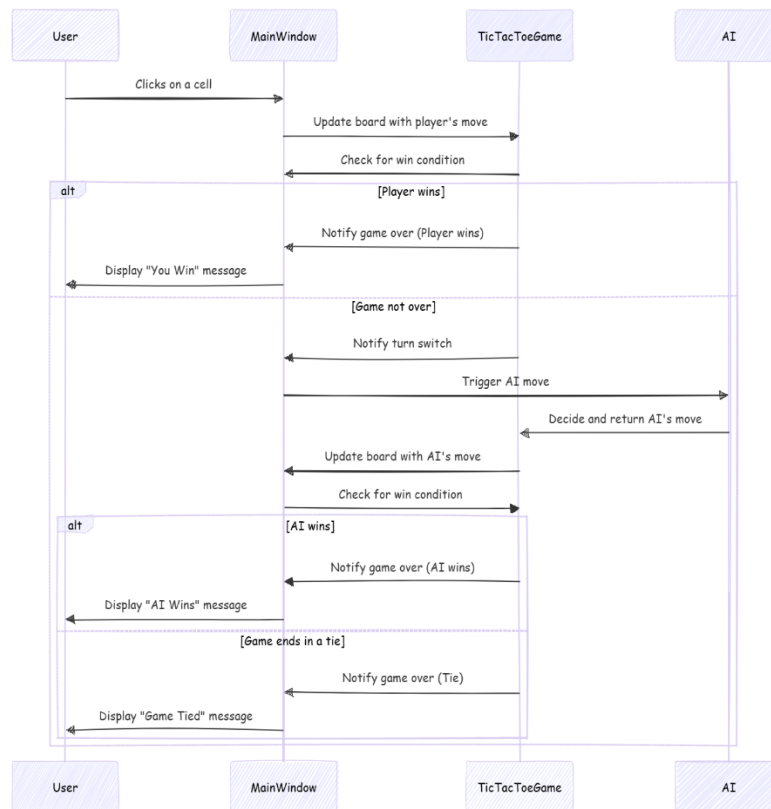


Figure 8 Game Logic Diagram

5. AI Algorithm:

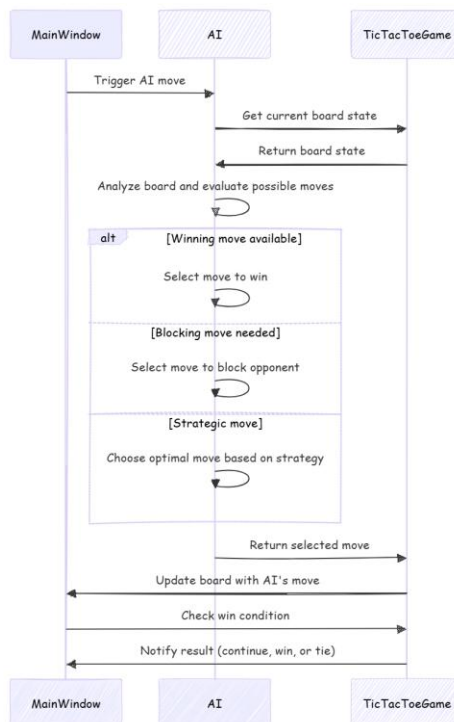


Figure 9 AI Algorithm