Software requirements specification:

1. Overview

This document details the requirements for a Tic Tac Toe game application. The game will support two modes of playing 2 players & 1 player vs AI, and include a login/sign-up feature with a history log for users to view and replay past games.

2. Functional Requirements

Sign-Up:

- Users can create an account by providing a username, email and password.
- Validation for unique usernames and email addresses.
- Validation for a valid password.
- System validates input and creates a new user account.

Login:

- Users can log in using their username and password.
- System validates credentials and logs the user in.

User Profile:

- Users can view their profile information.
- Users can view their past games and replay them.

Game Modes:

- Two Players Mode:

  - Two users can play against each other on the same device.
  - Alternating turns between players.

- One Player vs AI Mode:

  - A single user can play against the AI.
  - The AI has a Hard difficulty levels.

Game Board:

- A 3x3 grid where users can place their marks (X or O).

- Players alternate turns to place their mark on an empty cell.

Game Rules:

- One player is 'X', and the other player is 'O'. Players take turns placing their marks in an empty cell.

- The first player to get three of their marks in a row (horizontal, vertical, or diagonal) wins the game.

- If all nine cells are filled and neither player has three marks in a row, the game is a draw.

- In single-player mode, the user always plays as 'X', and the AI plays as 'O'.

- The game provides feedback on the game's result (win, loss, draw).

Game Flow

- User selects game mode (Two Players or One Player vs AI).

- User enters the players' names.

- Player 1 plays as 'X', and player 2 plays as 'O'.

- The system initializes the game board and indicates the current players.

- Players take turns placing their marks on the grid.

- System checks for a win or draw after each turn.

- The system announces the winner or a draw.

- The game result is saved in the user's history.

Game History:

- User navigates to the history section.

- Users can view a list of their past games, including date, opponent, result and a replay button.

- Users can replay any past game to review the moves made by each player.

- When the user selects a game to replay the system replays the game move-by-move as the user clicks on "Next move" button.

- System displays a "no more moves" message when the game ends.

Performance Requirements

- The system must handle up to 10,000 concurrent users without performance degradation.

- The application should immediately respond to user inputs .

```
move play time in milliseconds: 0
AI move play time in milliseconds: 8
move play time in milliseconds: 0
AI move play time in milliseconds: 0
move play time in milliseconds: 0
AI move play time in milliseconds: 0
move play time in milliseconds: 0
QSqlDatabasePrivate::addDatabase: duplicate
Game history saved successfully.
AI move play time in milliseconds: 1437
```

Fig 1: Response time of AI player

```
move play time in milliseconds: 0
move play time in milliseconds: 0
move play time in milliseconds: 0
move play time in milliseconds: 0
move play time in milliseconds: 0
move play time in milliseconds: 0
QSqlDatabasePrivate::addDatabase: duplicate
Game history saved successfully.
move play time in milliseconds: 1318
```

Fig 2: Response time for player moves

- The database should support rapid retrieval of game history and user information, with queries completing within 2 seconds.

```
Database connected
Error loading user profile: ""
Incorrect email or password.
log in time in milliseconds: 3
QSqlDatabasePrivate::addDatabase: duplicate
Database connected
User logged in successfully! User ID: 22
log in time in milliseconds: 23
```

Fig 3: Response time of logging in

```
QSqlDatabasePrivate::addDatabase: duplicate
sign up time in milliseconds: 0
QSqlDatabasePrivate::addDatabase: duplicate
sign up time in milliseconds: 1937
QSqlDatabasePrivate::addDatabase: duplicate
sign up time in milliseconds: 1938
```

Fig 4: Response time of signing up

3. Non-Functional Requirements

User Interface:

- Intuitive and user-friendly interface.

- Clear and responsive buttons for game actions.

- Handling of errors with user-friendly messages.

Data Protection:

- User data, including login credentials are securely stored and encrypted.

Authentication:

- Ensure secure login mechanisms to prevent unauthorized access.