Cairo University

Faculty of Engineering

Electronics and Electrical Communications Department

Tic Tac Toe

Software Requirements Specification (SRS)

Name	Code
كريم حسن عاطف على	9230666
محمد كرم محمد أحمد شعراوي	9230797
زينب محمد سيد درويش سيد	9230414
شريفه على قرني على عبدالحفيظ	9230475
أدهم علي حسين علي	9230200

Submitted to :

Dr. Omar Nasr

1. Introduction

1.1 Purpose

The purpose of this document is to specify the complete functional and non-functional requirements for the Tic Tac Toe game system developed using C++ and the Qt framework. This system provides both Player vs Player (1vs1) and Player vs AI (1vsAI) gameplay modes, complete with user authentication, registration, playing as a guest, game history tracking, and a stylized, responsive graphical interface.

1.2 Scope

This application is a desktop-based Tic Tac Toe game with the following features:

- 3x3 board turn-based game between two players
- Player vs AI with an intelligent agent using the Minimax algorithm and alpha-beta pruning.
- User registration, login, and guest mode
- Game history tracking and viewing for registered users
- Custom animations and stylized GUI elements
- Full compatibility with unit testing using Google Test

1.3 Intended Audience

- Developers and testers
- Project team members
- University or faculty evaluators
- Future maintainers of the software

1.4 Definitions

- **Minimax Algorithm**: A decision rule used in AI to minimize the possible loss in a worst-case scenario.
- **Alpha**: The **best score** the **maximizing player (AI)** can guarantee so far
- **Beta**: The **best score** the **minimizing player (opponent)** can guarantee so far
- **Guest Mode**: A mode where a user can play without logging in, but data is not saved.
- **Registered User**: A user with saved data whose games are stored and accessible.
- **1vs1**: Local two-player game mode.
- **1vsAI**: Game mode where a player competes against the AI.

---

2. Overall Description

2.1 Product Perspective

This is a standalone desktop game built with C++ and Qt. It integrates with an SQLite database to manage user data and game histories. It is designed to be self-contained with no external dependencies beyond the Qt framework.

2.2 Product Functions

- Allow users to register and log in securely
- Enable users to play a game in 1vs1 or 1vsAI mode
- Visually display game results with animations
- Automatically detect win and tie conditions after each move
- Save completed games to a database
- Allow registered users to view past games

2.3 User Classes and Characteristics

- **Guest Users**: Can play games but have no persistent data saved
- **Registered Users**: Can play, save, and view history of their games

2.4 Operating Environment

- Desktop environment (Windows, Linux, MacOS)
- Qt 5 or 6 framework
- SQLite database engine
- C++17 or higher compiler

2.5 Design and Implementation Constraints

- Must support both AI and human opponents
- Must be written using the Qt GUI framework
- Must use hashed passwords for authentication for more security

---

3. Functional Requirements

3.1 User Account Features

3.1.1 Authentication

- The system shall allow users to log in with a unique username and password
- The system search for the username and password entered by the user in the database
- If the user is stored in the database, the system advances to the main window and allow the user to play with his account and his history is saved
- If the user is not stored in the database, it prints that there is no such user, and waits for the correct username and password again

3.1.2 Registration

- The system shall allow users to register with a new unique username and password
- The system saves the new user in the database
- The system shall hash passwords before storing securely using **SHA-256 hashing**
- The user enters the username once and the password twice to confirm
- The system checks if the two passwords are the same
- The system checks if the password is strong (has lowercase, uppercase, symbol and digit)
- The system checks if the username and the password are unique or have been used before

3.1.3 Play as Guest

- Option to play without registration
- Guest users can access all game modes
- Game history is not saved for guests

3.2 Game Modes

- The system shall support **1vs1 game mode**: Two humans take turns on the same device
- The system shall support **1vsAI game mode**: Human plays against a computer opponent using the **Minimax algorithm with alpha-beta pruning**, ensuring optimal AI moves.
- The system shall allow players to choose their symbol (X or O)

3.3 Game Play

- The system shall maintain a 3x3 grid board
- The system shall enforce turn-taking rules
- The system shall detect when a player wins: 3 in a row, column, or diagonal → **win**
- The system shall detect a tie if the board is full: All cells filled with no winner → **tie**
- The system shall detect if the board still has free spaces → **Keep Playing**
- The system shall disable input on completed games
- On game end, a **popup dialog** announces the result

3.4 Artificial Intelligence (Minimax)

- The system shall use the Minimax algorithm with alpha-beta pruning for AI decisions
- The AI shall always make the optimal move based on current board state
- Evaluates moves recursively (looks like a tree data structure)
- Balances:

    - Win (10 - depth)
    - Loss (depth - 10)
    - Tie (score 0)

- The **depth** represents the **number of moves**. The less number of moves, the less depth of the tree of evaluations, the higher the score, the more optimal the move
- The **AI (maximizer)** wants the **highest score**.
- The **player (minimizer)** wants the **lowest score**.

- **Alpha-beta pruning** improves Minimax by:

- **Avoiding unnecessary branches**.
- **Skipping moves** that we already know will be bad.

While evaluating moves:

- If `beta <= alpha`, we **cut off** that branch (prune it).
- It means the minimizer has found a better path earlier, and the maximizer won't choose this one.

3.5 History and Persistence

- The system shall save completed games to a database for registered users
- The system shall allow registered users to view their game history including result and board state
- For logged-in users, every finished game (win/loss/tie) is saved, it saves: (Final board state, Game result, Timestamp)

3.6 UI and User Experience

- The system shall display animated transitions on button interactions
- The system shall use visual indicators to show current player turn
- The system shall provide a restart button to reset the game
- The system shall show custom dialogs for symbol selection, game over, and history

---

4. Non-Functional Requirements

4.1 Performance

- The system shall load the main interface within 2 seconds
- The AI shall make decisions in less than 500 ms

4.2 Usability

- The GUI shall be intuitive and accessible
- The game shall be operable with mouse only

4.3 Reliability

- The system shall not crash due to invalid inputs or empty database results

4.4 Security

4.4.1 Password Hashing

- Passwords never stored in plain text
- Uses **SHA-256 hashing** with `QCryptographicHash`

4.4.2 Input Validation

- Minimum username/password lengths
- Disallows special characters in usernames
- Matching password confirmation

4.5 Maintainability

- The codebase shall be modular with clearly separated logic and UI
- The system shall support automated unit and integration testing

4.6 Portability

- The system shall compile and run on Windows, Linux, and MacOS systems with Qt installed

---

5. System Models

5.1 Use Case Summary

- **Register / Login**: Secure user creation and access
- **Play Game (1vs1)**: Local multiplayer game
- **Play Game (1vsAI)**: Single-player game against AI
- **View History**: Browse past results (registered users only)

5.2 Data Model

- **Users Table**: `id`, `username`, `password_hash`
- **Games Table**: `id`, `user_id`, `board`, `result`, `timestamp`

5.3 Sequence (1vsAI)

1. User selects mode: 1vsAI
2. System prompts symbol selection
3. Game begins
4. Player and AI alternate moves
5. The system checks for a winner after each move
6. Result is displayed in the end of the game
7. History is saved (if user is registered)

---

6. Appendix

6.1 Tools Used

- Qt Creator
- C++ (C++17)
- SQLite
- Google Test

6.2 Known Limitations

- Multiplayer is local only; no online mode
- No user profile editing features