Faculty of Engineering

Department of Electronics and Electrical Communications

ELC2080 Project

Tic Tac Toe game

Software Requirements Specifications (SRS)

Under the supervision of:

Dr. Omar Ahmed Nasr

Contents

1.Introduction

1.1 Purpose

This document defines the software requirements for the Tic Tac Toe Game. It is intended to specify the functional and non-functional requirements, constraints, and external interfaces of the game. The intended audience includes software developers, testers, project supervisors and students involved in the development and evaluation of the software.

1.2 Background

Traditionally, Tic Tac Toe has been played on paper, but the increasing demand for digital and easily accessible entertainment highlights the need for a modernized version. This project addresses that demand by delivering a contemporary, feature-rich application that transforms the classic game into an engaging digital experience.

1.3 Target audience

The Tic Tac Toe desktop application is designed to appeal to a wide and diverse audience, including users of different age groups and technical proficiencies:

- **Families:** the game's simplicity and intuitive interface make it ideal for family-friendly usage, including children.
- **Friends**: it creates an environment full of cheer and happiness while playing. So, it is good to be a fun game that is played with friends.
- **Strategic thinkers**: players interested in challenging the AI to test and improve their tactical decision-making skills.

By catering to this diverse user base, the game aims to deliver an enjoyable experience that balances accessibility with strategic depth.

1.4 Scope

The Tic Tac Toe Game is a C++ application with a graphical user interface (GUI) that allows users to play either against another human or an AI opponent. It includes features such as user authentication, session management, personalized game history, and replay capability. The AI has different levels of difficulty. The system will be tested using Google Test and development will follow version control and CI/CD practices using GitHub and GitHub Actions.

1.5 Definitions, Acronyms, and Abbreviations

- **AI:** Artificial Intelligence.
- **GUI:** Graphical User Interface.
- **CI/CD:** Continuous Integration and Continuous Deployment.
- **SRS:** Software Requirements Specification.
- **Qt:** Cross-platform application development framework for GUI.
- **SQL:** Database engine.
- **UX:** User Experience.
- **UI:** User Interface.
- **OS:** Operating System.

1.6 Overview

This project aims to make the user experiences the enduring strategy and challenge of Tic Tac Toe in a modern and user-friendly format. This application offers a captivating and convenient way to test your skills, challenge others or AI opponent and experience the enduring fun of Tic Tac Toe.

The rest of this document details the system's overall functionality, detailed feature specifications, performance constraints, user interaction interfaces, and additional technical appendices that support the development and implementation of the game.

2.Overall description

2.1 Product perspective

The advanced Tic Tac Toe game is developed as a standalone desktop application, designed to operate independently without requiring internet connectivity. All core functionality, including gameplay, user interaction and AI computation, is executed locally on the user's device. For data persistence, the application supports the use of a local SQLite database, which securely manages user profiles and game history.

This project reimagines the classic Tic Tac Toe game by integrating modern software features that enhance both usability and user engagement. Key enhancements include:

- **User Authentication**: Secure login and registration to enable personalized access.
- **Game History Tracking**: Persistent storage of game sessions, allowing users to review and analyze past gameplay.
- **Strategic AI Opponent**: An intelligent AI agent built using different algorithms for different levels of difficulty each level has it algorithm.

2.2 Core functionalities

- **User registration and login**
    - User registration: new users can create a new account on the application by providing a unique username and password.
    - User login: registered players that have an account can login to their account to take advantage of the game history feature to show their past games.
- **Password storage with secure hashing**
    - Ensure user password is stored safely by applying cryptographic hashing algorithms, protecting this sensitive data from unauthorized access or breaches.
- **Interactive game interface**
    - When starting a game a new 3x3 empty grid is created allowing the player to play on the empty cell the starting player is always the "X" and the other is "O".
    - Each player should play on the empty cell otherwise the move won't be played. After each move it detects if there is a win or draw case and shows the result of the game.
- **Game mode detection**
    - The game can be player vs player mode or player vs AI mode
        - Player vs player mode: allow 2 players to play on the game each one has its turn after the other player. The player can be a guest player so no data will be saved in history for him, or it can be a registered player so he enters his username and password, and the game played will be saved in his database.
        - Player vs AI mode: the player choose the difficulty level "easy, normal or hard" then he plays and what for the AI to automatically play against it.
- AI opponent
    - The AI has 3 difficulty levels, each level works by a different algorithm
        - Easy level: the AI generates a random place from the empty places to play on it.
        - Normal level: the AI checks 2 possibilities if there is a winning move for it to play it or if there is a potential winning move for the human player and block it. Otherwise, if the 2 possibilities aren't found it plays randomly on allowable space.
        - Hard level: the AI plays using minimax algorithm with alpha and beta pruning to calculate the best move to be played.
- **Store and retrieve personalized game history**
    - Store each game the registered user played and show the list of games played and the result of each game also the time when it is played.
    - It also can show a replay of a match which is played before.

- **Errors handling using exceptions**
  - Each error like playing in wrong cell or a requesting to show the results is handled through exceptions and each possible wrong case is handled also

2.3 User classification

The first player who is the host of the game can be a registered user "has an account on the app" or a guest. When choosing to play against another player the other player can be a guest or a registered user so before starting the game both of them choose to play at which state if chosen to be a registered player the game played saved to his history of games also if guest it does not.

2.4 Operating environmental

For this initial development phase, the Tic Tac Toe game will be built specifically for the Windows operating system. While as a future work, we can work on to make it applicable on another operating systems like macOS and Linux

2.5 Constraints

- **The game is implemented in C++**

  C++ is used for performance and object-oriented design as it is a powerful and versatile programming language, will be the foundation for building the game's logic and functionality. This choice offers precise control over the game's performance.

- **The GUI is developed using the Qt framework**

  Qt provides a cross-platform framework for building responsive and modern user interfaces. It also provides a wide range of pre-built components and tools, streamlining the development process for creating menus, buttons, and other visual elements within the game.

- **User data stored using SQLite**

  SQLite comes into play here as the chosen method for data storage as it is a lightweight database management system, essentially a software tool that organizes and secures information. Also, it is used to store the user's credentials by secure cryptographic hash functions to protect against unauthorized access and data breaches.

- **CI/CD practices integrated using GitHub Actions**

  CI/CD, which stands for continuous integration and continuous delivery, refers to a set of practices that streamline the process of building, testing, and deploying the game. It specifies using GitHub Actions, a built-in automation feature within the popular code-sharing platform GitHub.

2.6 Assumptions

- **Users are familiar with basic Tic Tac Toe game rules**

  As the won't provide tutorials for the game, it is assumed that users understand the fundamental rules of Tic Tac Toe, including the concept of alternating turns, winning by forming a line "horizontal, vertical or diagonal", and the possibility of a draw.

- **Users interact using a mouse or similar pointing device**

  The application is designed primarily for use with a desktop or laptop computer equipped with a mouse, trackpad, or equivalent input device. It assumes that the user can interact with graphical UI elements by clicking or selecting options on the screen.

2.7 Dependencies

- **Operating system compatibility**

  The game requires a compatible desktop operating system such as Windows or Linux "mainly windows". It relies on OS-level support for GUI libraries, threading, and local file or database access.

- **Display environment**

  The application is designed to run on standard desktop/laptop display environments. Mobile devices, tablets, or small embedded systems are not supported. A minimum screen resolution is assumed for proper GUI rendering.

- **Language support:**

  The application will be developed in English. It is expected that the user can read and interact with English-language menus, messages, and labels and the desktop supports English language.

- **Qt framework availability**

  The Qt framework is essential for rendering the GUI and handling user interactions. The system requires Qt runtime libraries to be available on the host machine. These libraries support features such as windows, buttons, text inputs, and game board visualization.

2.8 System Behavior

The system is designed to provide responsive and deterministic behavior during all game interactions. The following outlines expected behaviors in key functional areas:

- **User Authentication**
  - Upon launching the application, the user is presented with options to log in or register or to play as a guest with no account.
  - Successful login grants access to the main menu. Failed login displays an error message.
  - Registration creates a new user profile with a hashed password stored securely.
- **Main Menu Navigation**
  - After login, the main menu offers options: Play vs. Player, Play vs. AI, View History, and Logout.
  - Each button routes the user to the respective module.
- **Gameplay**
  - In either game mode (Player vs. Player or Player vs. AI), the system initializes an empty 3x3 board.
  - Players alternate turns (X and O).
  - After each move, the system evaluates the game state to determine:
    - If a player has won.
    - If the game is a draw.
    - Or if the game should continue.
  - The system disables further input once a result is determined and displays the outcome.
- **AI Response**
  - The AI computes its move using the minimax algorithm with alpha-beta pruning.
  - The AI move is executed within 500ms and adheres to optimal gameplay strategy.
- **Game History**
  - At the end of each game, the system automatically records the result, players involved, and sequence of moves.
  - Users can access and replay any game from their history.
  - Replays show moves step-by-step with pause and skip controls

2.9 Technology stack

Component	Technology
Language	C++
GUI framework	Qt
Database	SQLite
Testing framework	Google test
CI/CD	GitHub actions
Security	cryptographic hash function

3.Specific requirements

3.1 Functional requirements

3.1.1 Game environment

The game is developed exclusively for Windows OS (Windows 10 and later), ensuring full compatibility and optimized performance. The Qt framework is utilized to deliver a polished graphical user interface (GUI) with smooth rendering and responsiveness. Cross-platform expansion to macOS and Linux may be considered in future updates due to Qt's inherent portability.

3.1.2 User registration and login

Account creation: Users register with a unique username and password, which password is securely hashed before storage in an SQLite database to prevent unauthorized access.

Secure login: Users can log in by providing their username and password, which are verified against securely stored, hashed credentials to ensure secure access to their accounts. So, users must provide the correct combination of username and password

3.1.3 Game Play

- **Game Modes**

  Local multiplayer: Two players alternate turns on the same device.

  Single player vs AI: The AI opponent uses different algorithms according to the difficulty level to challenge the player.

- **Game board**

  A 3x3 grid with clear visual distinctions (e.g., bold borders, contrasting colors) ensures readability. The board dynamically updates to reflect moves.

- **Turn management**

  Players alternate turns strictly, with visual cues (e.g., highlighting the current player’s symbol) to prevent confusion.

- **Win/Draw detection**

  The system checks for three matching symbols in a row, column, or diagonal after each move. A draw is declared if the board fills without a winner.

3.1.4 AI Opponent

Adjustable difficulty:

- Easy mode: The AI makes random moves, suitable for beginners.
- Normal mode: The AI makes random moves unless it find a winning move for it the play it or find a winning move for the human player then it blocks it.
- Hard mode: The AI employs full-depth minimax analysis, offering a formidable challenge.

3.1.5 Game history

Tracking: Logged-in users can view past games, showing outcomes (win/loss/draw) and the time of the match played.

Replay feature: Players can replay games to analyze strategies.

3.1.6 User interaction

Input Handling: Mouse clicks are the primary input method, with instant visual feedback (e.g., symbol placement,).

Feedback Mechanisms:

- Turn indicators (e.g., "Player X’s turn").

3.2 Non-Functional Requirements

3.2.1 Performance

Responsiveness: All user actions (e.g., clicks, AI moves) are processed within 1 second or less to maintain fluid gameplay.

Resource Efficiency: Memory usage is capped at 100MB to ensure smooth performance on low-end systems.

3.2.2 Security

Protect passwords by hashing each password with cryptograph hashing function and add to it a salt to safeguard user data.

3.3 Interface Requirements

3.3.1 User interface

Main screen: The central 3x3 grid dominates the UI, flanked by turn indicators and control buttons (New Game, Restart).

Game controls: Buttons for mode selection and difficulty adjustment (for AI) are prominently placed.

3.3.2 System Interface

Database integration: SQLite stores user profiles, game history, and settings, with ACID compliance ensuring data reliability.

External libraries: Qt handles GUI rendering, while GitHub Actions automates CI/CD pipelines for testing and deployment.

4.Appendix

Game rules

➢ Objective

The objective of Tic Tac Toe is to be the first player to place three of your symbols (X or O) in a row, column, or diagonal on the 3x3 grid. You can achieve this by strategically placing your symbols and blocking your opponent's attempts to do the same.

➢ Players

Tic Tac Toe can be played by two players: For Example:

- Player 1: Typically represented by the symbol "X" and takes the first turn.
- Player 2: Typically represented by the symbol "O" and takes turns after Player 1.

➢ Gameplay

The game begins with an empty 3x3 grid.

- Player 1 takes the first turn by placing their "X" symbol in any unoccupied cell on the grid.
- Player 2 then takes their turn by placing their "O" symbol in any unoccupied cell.

Players alternate turns, placing their respective symbols on the grid. After each turn, the game checks for a win or a tie.

➢ Winning condition

A player wins when they successfully place three of their symbols (X or O) in a row, column, or diagonal on the grid.

➢ Draw condition

If all nine cells of the grid are filled with symbols (X and O) without either player achieving a win, the game is declared a tie.

➢ Turn management

The game enforces a turn-based system. Players alternate in their turns, ensuring each player has a fair and strategic opportunity to place their symbols. Taking a turn involves selecting an empty cell on the grid and placing your symbol (X or O) in cell.

➢ Game end

The game ends when one of the following conditions occurs:

- A player achieves a win by placing three of their symbols in a row, column, or diagonal.
- The game board is filled with symbols (X and O) resulting in a draw.