Faculty of Engineering
Department of Electronics and Electrical Communications
ELC2080 Project

Advanced Tic-Tac-Toe

System Requirements Specifications

Name	ID
Ziad Mostafa	9231055
Ziad Emad	9230401
Ziad Abdel Hafeez	9230399
Sandra Atef	9230428
Robir Tamer	9230378

Under the supervision of:

Dr. Omar Ahmed Nasr

Table of Contents

1 Document Release History 4
2 Authors, Reviewers and Stakeholders' 4
Client Stakeholders Register 4
3 Introduction 5
3.1 Background 5
3.2 Intended Audience 5
3.3 Purpose 5
3.4 Scope 5
3.5 Definitions, Acronyms, and Abbreviations 5
3.6 Overview 6
4 Overall Description 7
4.1 Product perspective 7
4.2 Product functions 7
4.2.1 User login and registration 7
4.2.2 Interactive game interface for Tic-Tac-Toe 7
4.2.3 AI opponent 7
4.2.4 Game history tracking 7
4.2.5 Secure user management 7
4.3 User Classes and Characteristics: 7
4.4 Operating Environment 7
4.4.1 Nowadays Requirement 7
4.4.2 Future Expansion Potential 8
4.5 Design and Implementation Constraints 8
4.5.1 The game must be implemented in C++. 8
4.5.2 The GUI must be developed using the Qt framework. 8
4.5.3 User data must be stored securely using SQLite. 8
4.5.4 CI/CD practices must be integrated using GitHub Actions 8
4.6 Assumptions and dependencies of the system: 8
4.6.1 Assumptions 8
4.6.2 Dependencies 8
5 Specific Requirements 9
5.1 Functional Requirements 9
5.1.1 Game Environment 9
5.1.2 User Registration and login 9
5.1.3 Game Play 9
5.1.4 AI Opponent 9
5.1.5 Game History 9
5.1.6 User Interaction 9

5.1.7 Additional Options 10
5.2 Non-Functional Requirements 10
5.2.1 Performance 10
5.2.2 Security 10
5.2.3 Usability 10
5.3 Interface Requirements 10
5.3.1 User Interface 10
5.3.2 System Interface 10
6 Appendix 11
6.1 Glossary 11
6.2 Issues List 11
6.3 Game Rules 11
6.3.1 Objective 11
6.3.2 Players 11
6.3.3 Gameplay 11
6.3.4 Winning Conditions 11
6.3.5 Tie Conditions 12
6.3.6 Turn Management 12
6.3.7 AI Opponent 12
6.3.8 Game End 12
6.4 Detailed References 12

1 Document Release History

#	Version No.	Release Date	Prepared By	Reviewed By	Preparation Comments
1	V1.0	17/6/2024	Sandra Atef	Robir Tamer	Initial Publication

2 Authors, Reviewers and Stakeholders’

Client Stakeholders Register

#	Name	Role in the project	Influence	Contact	Email
1	Dr. Omar Ahmed Nasr	Program Manager	SPOC	01143941832	omaranasr.phone@gmail.com

3 Introduction

This document is designed for the project development team, including software engineers responsible for the code base and algorithms, project managers, and quality assurance and testing teams. It is also applicable for stakeholders involved in project evaluation and validation for our solution.

3.1 Background

While the classic Tic Tac Toe game has traditionally been enjoyed using pen and paper, there's a growing demand for convenient and accessible digital entertainment options. This project aims to address this need by developing an application that offers a modern and engaging Tic Tac Toe experience.

3.2 Intended Audience

The Tic Tac Toe application is designed for a broad audience encompassing individuals of various ages and technical backgrounds. Here's a breakdown of the primary target users:

- **Casual Gamers:** Individuals who enjoy classic games like Tic Tac Toe for casual entertainment. They may not be avid gamers but appreciate a simple and accessible game for passing time.
- **Strategic Players:** Users who enjoy the strategic aspects of Tic Tac Toe. They may be interested in playing against the AI opponent to test their skills and improve their strategic thinking.
- **Families and Children:** The application is suitable for families and children due to the simple rules, turn-based nature of Tic Tac Toe, and the user-friendly interface designed for easy navigation.
- **Social Gamers:** Users who enjoy playing games with friends and family. The multiplayer functionality caters to this group, adding a social element to the traditional Tic Tac Toe experience.

By targeting this diverse user base, the application aims to provide a fun and engaging experience for a wide range of people.

3.3 Purpose

It aims to provide a comprehensive description of the advanced Tic-Tac-Toe game, ensuring user engagement and satisfaction including:

- **User Authentication:** Enabling secure account creation, login, and profile management.
- **Personalized Game History:** Allowing players to track their past games and review details of each session.
- **Intelligent AI Opponent:** Implementing a strategic AI using the minimax algorithm with alpha-beta pruning to provide a challenging and engaging experience for solo players.

These specifications will guide the development process, ensuring all features are seamlessly integrated to deliver a complete and user-friendly Tic Tac Toe application.

3.4 Scope

The scope of the Tic-Tac-Toe project encompasses the creation of an interactive game application with a user-friendly GUI, secure user management, and AI integration. The project will cover software development, testing, CI/CD practices, and performance optimization.

3.5 Definitions, Acronyms, and Abbreviations

This section defines common terms and acronyms used throughout the document to ensure clarity and consistency.

Term	Definition
SRS	Software Requirements Specification
AI	Artificial Intelligence
GUI	Graphical User Interface
CI/CD	Continuous Integration/Continuous Deployment
UX	User Experience
UI	User Interface
OS	Operating System
UML	Unified Modeling Language

3.6 Overview

This project is 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 as family, friends or AI opponent, and experience the enduring fun of Tic Tac Toe.

Key Features:

- **Classic Gameplay:** Enjoy the timeless appeal of Tic Tac Toe with a familiar 3x3 grid and intuitive turn-based mechanics.
- **Multiple Playstyles:** Challenge others to a head-to-head battle or test your strategic prowess against a sophisticated AI opponent.
- **Streamlined Interface:** The intuitive and clean interface fosters a smooth user experience for players of all ages and skill levels.
- **Personalized experience:** Depending on the specific product, you might find features like user profile creation and personalized game history tracking to enhance your experience.

This Tic Tac Toe application is designed to provide a stimulating and convenient way to enjoy the classic game. Sharpen your strategic thinking, challenge your friends, and experience the timeless fun of Tic Tac Toe!

4 Overall Description

4.1 Product perspective

The Professional Tic-Tac-Toe application is a standalone desktop application developed using C++ and the Qt framework. It is designed to provide a modern and engaging digital Tic-Tac-Toe experience. The application integrates user authentication, game logic, AI opponent capabilities, and game history tracking. It is a self-contained system that does not directly interface with other major systems, but it does utilize an SQLite database for local data storage.

4.2 Product functions

The core functions of the Professional Tic-Tac-Toe application include:

4.2.1 User login and registration

Users can create new accounts and log in to existing accounts. This functionality is handled by the `Authentication` class, which manages user data, including hashed passwords, and provides secure login and registration processes. The `User` class defines the structure for user profiles.

4.2.2 Interactive game interface for Tic-Tac-Toe

The application provides a graphical user interface (GUI) for playing Tic-Tac-Toe. The MainWindow class is responsible for setting up and managing the UI elements, including the game board, status messages, and various interactive buttons. The GameLogic class manages the game state, player turns, and win conditions.

4.2.3 AI opponent

The application features an AI opponent that players can challenge. The AIOpponent class implements the AI logic, likely utilizing algorithms such as Minimax with Alpha-Beta Pruning to determine optimal moves based on the selected difficulty level. This provides a challenging experience for solo players.

4.2.4 Game history tracking

The application tracks and stores the history of played games. The GameHistory class

is responsible for recording game outcomes, player statistics, and other relevant game data. This data can be viewed by users to review their past performance.

4.2.5 Secure user management

User data, including credentials and game statistics, is managed securely. The Database class handles the persistence of this data using SQLite, ensuring that user

information is stored and retrieved reliably and securely.

4.3 User Classes and Characteristics:

The primary user classes for the Professional Tic-Tac-Toe application are:

- **Casual Players:** Users who want to quickly play a game of Tic-Tac-Toe without extensive setup. They prioritize ease of use and a straightforward gaming experience.
- **Competitive Players:** Users who enjoy challenging themselves against an AI or other players. They are interested in tracking their progress, improving their skills, and engaging with more complex game scenarios.
- **New Users:** Individuals who are new to the application and need a clear and intuitive registration and onboarding process.
- **Returning Users:** Users who have previously registered and wish to log in to access their personalized game history and settings.

4.4 Operating Environment

4.4.1 Nowadays Requirement

The application is designed to run on modern desktop operating systems, including Windows, macOS, and Linux, leveraging the cross-platform capabilities of the Qt framework. It requires a standard desktop environment with sufficient CPU, memory, and display resources to run a graphical application.

4.4.2 Future Expansion Potential

Future expansions could include:

- **Mobile Platform Support:** Adapting the application for mobile operating systems (Android, iOS) using Qt for mobile development.
- **Online Multiplayer:** Implementing network capabilities to allow players to compete against each other over the internet.
- **Cloud Integration:** Storing user data and game history in a cloud database for better scalability and accessibility across multiple devices.

4.5 Design and Implementation Constraints

Several constraints guide the design and implementation of the Professional Tic-Tac-Toe application:

4.5.1 The game must be implemented in C++.

All core logic and application components must be developed using the C++ programming language, ensuring high performance and control over system resources.

4.5.2 The GUI must be developed using the Qt framework.

The graphical user interface must be built exclusively with the Qt framework, utilizing its widgets, signals and slots mechanism, and styling capabilities to create a modern and responsive UI.

4.5.3 User data must be stored securely using SQLite.

All persistent user data, including authentication details and game history, must be stored in an SQLite database. Security measures, such as password hashing, must be implemented to protect sensitive user information.

4.5.4 CI/CD practices must be integrated using GitHub Actions

Continuous Integration and Continuous Deployment (CI/CD) practices must be adopted, with GitHub Actions as the chosen platform for automating build, test, and deployment workflows.

4.6 Assumptions and dependencies of the system:

4.6.1 Assumptions

- **Users have a basic understanding of Tic Tac Toe rules:** It is assumed that users have a basic understanding of Tic-Tac-Toe rules.
- **Users have a mouse or other pointing device:** The game will be designed with the expectation of using a mouse or similar tool for clicking on the game board and menu options.
- **The user's computer meets minimum system requirements:** The game will likely have minimal resource requirements, but it's still important to assume users have a computer that can handle basic graphics processing and has enough memory to run the application smoothly.
- **Internet Connection (for future updates):** While the core game is offline, future features like online multiplayer or cloud synchronization would assume an active internet connection.
- **Standard Desktop Environment:** The application assumes a standard desktop operating system environment with necessary graphics drivers and system libraries.

4.6.2 Dependencies

- **Qt Framework:** The application is heavily dependent on the Qt framework for its GUI, event handling, and various utility classes.
- **SQLite Library:** The SQLite library is required for database operations, including storing and retrieving user and game data. C++ Compiler: A C++ compiler (e.g., GCC, Clang, MSVC) compatible with the Qt version used is necessary for building the application.
- **Operating System Specific Libraries:** Standard operating system libraries for file I/O, networking (if applicable), and system calls.

5 Specific Requirements

This section details the specific functional and non-functional requirements of the Professional Tic-Tac-Toe application. These requirements define what the system must do and the quality attributes it must possess.

5.1 Functional Requirements

Functional requirements describe the functions that the system must perform. They define the behavior of the system under various conditions.

5.1.1 Game Environment

- **FR1.1:** The system shall display a 3x3 game board for Tic-Tac-Toe.
- **FR1.2:** The system shall visually represent the state of each cell on the board (empty, 'X', or 'O').
- **FR1.3:** The system shall indicate the current player whose turn it is.
- **FR1.4:** The system shall display status messages to the user (e.g.,"X wins!", "It's a draw!").
- **FR1.5:** The system shall provide a clear and intuitive user interface for navigation and interaction.

5.1.2 User Registration and login

- **FR2.1:** The system shall allow new users to register with a unique username and password.
- **FR2.2:** The system shall validate user input during registration to ensure strong passwords and unique usernames.
- **FR2.3:** The system shall securely store user credentials (e.g., using hashed passwords).
- **FR2.4:** The system shall allow registered users to log in with their username and password**.**
- **FR2.5:** The system shall authenticate user credentials and provide appropriate feedback (e.g., "Login successful", "Invalid credentials").
- **FR2.6:** The system shall maintain a logged-in session for the user until they explicitly log out or the application is closed.

5.1.3 Game Play

- **FR3.1:** The system shall allow players to choose between playing against an AI opponent or another human player (local multiplayer).
- **FR3.2:** The system shall allow players to make moves by clicking on empty cells on the game board.
- **FR3.3:** The system shall update the game board to reflect the player's move.• FR3.4: The system shall automatically switch turns between players after a valid move.
- **FR3.5:** The system shall detect and declare a winner when a player achieves three of their marks in a row, column, or diagonal.
- **FR3.6:** The system shall detect and declare a draw when the board is full and no player has won.
- **FR3.7:** The system shall provide an option to start a new game at any point.• FR3.8: The system shall provide an option to save the current game state.
- **FR3.9:** The system shall provide an option to load a previously saved game state.

5.1.4 AI Opponent

- **FR4.1:** The system shall provide an AI opponent with multiple difficulty levels (e.g., Easy, Medium, Hard, Expert).
- **FR4.2:** The AI opponent shall make moves automatically based on its difficulty level.
- **FR4.3:** For higher difficulty levels, the AI shall employ advanced algorithms (e.g., Minimax with Alpha-Beta Pruning) to determine optimal moves.
- **FR4.4:** The AI's response time shall be reasonable and not cause significant delays in gameplay.

5.1.5 Game History

- **FR5.1:** The system shall record the outcome of each completed game (win, loss, draw).
- **FR5.2:** The system shall store game history for each registered user.
- **FR5.3:** The system shall allow users to view their game history, including details such as opponent, outcome, and date.
- **FR5.4:** The system shall display overall statistics for each user (e.g., total games played, wins, losses, draws, win rate).

5.1.6 User Interaction

- **FR6.1:** The system shall provide clear visual feedback for user actions (e.g., highlighting selected cells).
- **FR6.2:** The system shall handle invalid user inputs gracefully and provide informative error messages.
- **FR6.3:** The system shall allow users to navigate between different sections of the application (e.g., game, statistics, settings).

5.1.7 Additional Options

- **FR7.1:** The system shall provide an option to exit the game.
- **FR7.2:** The system shall allow users to adjust game settings (e.g., sound, theme) if implemented.

5.2 Non-Functional Requirements

Non-functional requirements specify criteria that can be used to judge the operation of a system, rather than specific behaviours.

5.2.1 Performance

- **NFR1.1:** The application shall launch within 5 seconds on a standard desktop computer.
- **NFR1.2:** The UI shall remain responsive, with user interactions (e.g., button clicks) having a response time of less than 100 milliseconds.
- **NFR1.3:** The AI opponent's move generation time shall not exceed 2 seconds for the highest difficulty level.
- **NFR1.4:** Memory consumption shall remain within reasonable limits (e.g., under 200 MB during active gameplay).
- **NFR1.5:** Database operations (save/load game, user authentication) shall complete within 500 milliseconds.

5.2.2 Security

- **NFR2.1:** User passwords shall be stored securely using industry-standard hashing algorithms (e.g., SHA-256).
- **NFR2.2:** The application shall protect against common security vulnerabilities (e.g., SQL injection, buffer overflows).
- **NFR2.3:** User data stored locally shall be protected from unauthorized access.

5.2.3 Usability

- **NFR3.1:** The user interface shall be intuitive and easy to navigate for users of all technical levels.
- **NFR3.2:** The application shall provide clear and concise error messages.
- **NFR3.3:** The application shall be visually appealing and consistent in its design.

5.3 Interface Requirements

Interface requirements specify how the system interacts with other systems, users, and hardware.

5.3.1 User Interface

- **UI1.1:** The application shall provide a main window with distinct sections for game play, user authentication, and statistics.
- **UI1.2:** The game board shall be visually distinct and clearly indicate occupied and empty cells**.**
- **UI1.3:** Buttons and interactive elements shall be clearly labeled and provide visual feedback upon interaction.
- **UI1.4:** Text and messages displayed to the user shall be legible and easy to understand.

5.3.2 System Interface

- **SI1.1:** The application shall interface with the underlying operating system for file system access (for saving/loading games and user data).
- **SI1.2:** The application shall interface with the SQLite database for persistent data storage.
- **SI1.3:** The application shall utilize the Qt framework's APIs for GUI rendering, event handling, and other system-level interactions.

6 Appendix

6.1 Glossary

This glossary defines key terms and concepts used throughout this Software Requirements Specification (SRS) document to ensure clarity and common understanding among all stakeholders.

Term	Definition
AI (Artificial Intelligence)	The capability of a machine to imitate intelligent human behavior.
GUI (Graphical User Interface)	A user interface that includes graphical elements, such as windows, icons, and buttons.
Qt Framework	A cross-platform application framework used for developing application software with a graphical user interface.
Minimax Algorithm	An algorithm used in decision-making and game theory for minimizing the possible loss in a worst-case scenario.
SQLite	A C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.
Responsive	The ability of a system or component to quickly react to user input.
Hashing	A process of converting an input (or 'message') into a fixed-length string of bytes, typically used for secure data storage.
Authentication	The process of verifying the identity of a user or system.
Authorization	The process of determining what a user or system is allowed to do.
Session Management	Techniques used to manage a user's interaction with a web application over multiple requests.
Cross-Platform	Software that can run on multiple operating systems without modification.
Accessibility	The design of products, devices, services, or environments for people who experience disabilities.
UI/UX	User Interface/User Experience; focuses on the design and usability of a software application.
UML (Unified Modeling Language)	It a general-purpose visual modeling language that is intended to provide a standard way to visualize the design of a system.

6.2 Issues List

This section lists any known issues, open questions, or areas requiring further clarification or decision-making related to the requirements. As of this version, there are no outstanding issues that prevent the core development.

- **Issue 1:** (Placeholder) Further refinement of AI difficulty levels may be required based on user feedback during testing.
- **Issue 2:** (Placeholder) Decision on specific third-party libraries for advanced features (e.g., analytics, crash reporting) is pending.

6.3 Game Rules

This section outlines the fundamental rules of Tic-Tac-Toe as implemented in the application.

6.3.1 Objective

The objective of Tic-Tac-Toe is for players to get three of their marks (either X or O) in a row, column, or diagonal on a 3x3 grid.

6.3.2 Players

The game can be played by two human players (Player vs. Player mode) or one human player against an AI opponent (Player vs. AI mode).

6.3.3 Gameplay

Players take turns placing their marks on an empty square of the 3x3 grid. Player X always goes first.

6.3.4 Winning Conditions

A player wins by successfully placing three of their marks in a horizontal, vertical, or diagonal line.

6.3.5 Tie Conditions

The game is a draw if all nine squares on the grid are filled and neither player has achieved three marks in a row.

6.3.6 Turn Management

The application automatically manages turns, ensuring that players make moves alternately. In Player vs. AI mode, the AI makes its move after the human player.

6.3.7 AI Opponent

The AI opponent will make moves based on its programmed logic and selected difficulty level, aiming to win or block the human player.

6.3.8 Game End

The game ends when either a player wins or the game results in a draw. At this point, the application will display the game outcome and offer options to start a new game or return to the main menu.

6.4 Detailed References

[1] C. Dr. Omar Nasr, "Advanced Tic Tac Toe Game," *Detailed Project Description*, April 2024.

[2] "Github," [Online]. Available: https://google.github.io/googletest/.

[3] "Qt_studio_GUI_index," [Online]. Available: https://doc.qt.io/qt-6/qtgui-index.html.

[4] "Finding optimal move in Tic-Tac-Toe using Minimax Algorithm in Game Theory," 20 Feb 2023. [Online]. Available: https://www.geeksforgeeks.org/finding-optimal-move-in-tic-tac-toe-using-minimax-algorithm-in-game-theory/.

[5] "SQLite Official Documentation," [Online]. Available: https://www.sqlite.org/docs.html.

[6] "Secure Hashing algorithms," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function.

[7] "User Interface Design Basics," [Online]. Available: https://www.interaction-design.org/literature/topics/ui-design.