Software Requirements Specification

For

Advanced Tic Tac Toe

Prepared by TicTacTactics

Submitted to: Dr. Omar Nasr

**Table of Contents**

[1. Introduction 1](#_Toc170132310)

[1.1 Purpose 1](#_Toc170132311)

[1.2 Document Conventions 1](#_Toc170132312)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc170132313)

[1.4 Project Scope 1](#_Toc170132314)

[1.5 References 1](#_Toc170132315)

[2. Overall Description 2](#_Toc170132316)

[2.1 Product Perspective 2](#_Toc170132317)

[2.2 Product Features 2](#_Toc170132318)

[2.3 User Classes and Characteristics 2](#_Toc170132319)

[2.4 Operating Environment 2](#_Toc170132320)

[2.5 Design and Implementation Constraints 2](#_Toc170132321)

[2.6 Assumptions and Dependencies 2](#_Toc170132322)

[3. System Features 3](#_Toc170132323)

[3.1 System Feature 1: User Authentication 3](#_Toc170132324)

[3.1.1 Description: 3](#_Toc170132325)

[3.1.2 Functional Requirements: 3](#_Toc170132326)

[3.1.3 User Interface Requirements: 3](#_Toc170132327)

[3.1.4 Non-Functional Requirements: 3](#_Toc170132328)

[3.2 System Feature 2: Mode Selector 3](#_Toc170132329)

[3.2.1 Description: 3](#_Toc170132330)

[3.2.2 Functional Requirements: 3](#_Toc170132331)

[3.2.3 User Interface Requirements: 3](#_Toc170132332)

[3.2.4 Non-Functional Requirements: 3](#_Toc170132333)

[3.3 System Feature 3: Game Board 4](#_Toc170132334)

[3.3.1 Description: 4](#_Toc170132335)

[3.3.2 Functional Requirements: 4](#_Toc170132336)

[3.3.3 User Interface Requirements: 4](#_Toc170132337)

[3.3.4 Non-Functional Requirements: 4](#_Toc170132338)

[3.4 System Feature 4: Player vs Player Mode 4](#_Toc170132339)

[3.4.1 Description: 4](#_Toc170132340)

[3.4.2 Functional Requirements: 4](#_Toc170132341)

[3.4.3 User Interface Requirements: 4](#_Toc170132342)

[*3.*4.4 Non-Functional Requirements: 4](#_Toc170132343)

[3.5 System Feature 5: Player vs AI Mode 5](#_Toc170132344)

[3.5.1 Description: 5](#_Toc170132345)

[3.5.2 Functional Requirements: 5](#_Toc170132346)

[3.5.3 User Interface Requirements: 5](#_Toc170132347)

[3.5.4 Non-Functional Requirements: 5](#_Toc170132348)

[3.6 System Feature 6: Game History 5](#_Toc170132349)

[3.6.1 Description: 5](#_Toc170132350)

[3.6.2 Functional Requirements: 5](#_Toc170132351)

[3.6.3 User Interface Requirements: 5](#_Toc170132352)

[3.6.4 Non-Functional Requirements: 5](#_Toc170132353)

[4. External Interface Requirements 6](#_Toc170132354)

[4.1 User Interfaces 6](#_Toc170132355)

[4.2 Hardware Interfaces 6](#_Toc170132356)

[4.3 Software Interfaces 6](#_Toc170132357)

[5. Other Nonfunctional Requirements 6](#_Toc170132358)

[5.1 Performance Requirements 6](#_Toc170132359)

[5.2 Security Requirements 6](#_Toc170132360)

[5.3 Software Quality Attributes 6](#_Toc170132361)

# Introduction

## Purpose

This document specifies the requirements for the XO (Tic Tac Toe) game project, developed using Qt and C++ with SQLite for database storage. It is intended to provide a comprehensive overview of the game's functionalities, features and design constraints.

## Document Conventions

* XO: Tic-Tac-Toe
* Qt: Open-Source widget toolkit for creating graphical user interfaces (GUI).
* GUI: Graphical User Interface.
* AI: Artificial Intelligence.
* SQLite: A C-language library that implements a small, fast, self-contained, high-reliability SQL database engine.

## Intended Audience and Reading Suggestions

* Users: to seek information on game features to enhance their gameplay experience.
* Developers: For understanding the requirements and design of the game.
* Testers: For creating test cases based on the requirements.
* Project Managers: For understanding the scope and features of the project.

## Project Scope

The project includes an XO game with user authentication, mode selection (Player vs Player, Player vs AI), AI difficulty levels, game board interaction, move validation, win/lose/draw conditions, game history tracking, and database integration using SQLite.

## References

Qt Documentation: <https://doc.qt.io/>

SQLite Documentation: <https://www.sqlite.org/docs.html>

# Overall Description

## Product Perspective

The XO game is a standalone desktop application that allows users to sign up, log in, select game modes, and play against another player or AI. The application stores user data, game history, and tracks game statistics using an SQLite database.

## Product Features

* User authentication (sign-up and sign-in).
* Game modes: Player vs Player, Player vs AI (Easy, Medium, Hard)
* Game board with interactive cell selection and turn indicator.
* Win/lose/draw condition checking after each move.
* Game history tracking with detailed game records.
* Data storage in SQLite database including player statistics (win, draw, lose counts)

and individual game details.

## User Classes and Characteristics

* Players: Users who interact with the game, play against each other or against AI.
* Developers: Individuals responsible for maintaining and updating the game.

## Operating Environment

* *Supported on Windows operating system.*
* Developed using Qt and SQLite libraries.

## Design and Implementation Constraints

* *Must follow the Qt framework for a consistent and functional UI.*
* SQLite database is used for efficient data storage and retrieval.

## Assumptions and Dependencies

* *Users have basic familiarity with Tic-Tac-Toe game rules.*
* Qt and SQLite libraries are installed on the user's machine.

# System Features

## System Feature 1: User Authentication

### Description:

Allows users to sign up for a new account or sign in with existing credentials.

### Functional Requirements:

*-* ***Sign-up****: Username, password (entered twice for confirmation).*

*-* ***Sign-in****: Validate username and hashed password from SQLite database.*

### User Interface Requirements:

*- Sign-up screen with fields for username and password.*

*- Sign-in screen with fields for username and password.*

*- Error messages for incorrect credentials or sign-up validation.*

### Non-Functional Requirements:

*-* ***Security:*** *Hashed storage of passwords in SQLite.*

*-* ***Usability:*** *Intuitive and user-friendly authentication process.*

## System Feature 2: Mode Selector

### Description:

*Allows users to choose between Player vs Player or Player vs AI modes.*

### Functional Requirements:

* *Selection buttons for Player vs Player and Player vs AI modes.*
* *Transition to selected mode upon user input.*

### User Interface Requirements:

* *Mode selection screen with buttons for different game modes.*

### Non-Functional Requirements:

* ***Usability:*** *Clear and straightforward mode selection process*.

## System Feature 3: Game Board

### Description:

Displays a 3x3 grid where players can place their marks (X or O).

### Functional Requirements:

*- Interactive grid cells for player input.*

*- Update board display after each valid move.*

*- Display current player turn.*

### User Interface Requirements:

*- 3x3 grid layout for game.*

*- Highlighted cells indicating current player turn*.

### Non-Functional Requirements:

*-* ***Performance:*** *Smooth update of board state within 0.5ms per move.*

*-* ***Usability:*** *Clear visualization of game board and turn indicator.*

## System Feature 4: Player vs Player Mode

### Description:

Allows two players to play against each other on the same device.

### Functional Requirements:

*- Alternate turns between players.*

*- Check for win/lose/draw conditions after each move.*

*- Update player statistics (wins, losses, draws) in Data Base (SQLite).*

### User Interface Requirements:

*- Display game outcome (win/lose/draw).*

*- Button to start a new game session.*

### Non-Functional Requirements:

*-* ***Usability:*** *Seamless gameplay experience for local multiplayer*.

## System Feature 5: Player vs AI Mode

### Description:

Allows a player to play against an AI opponent with three difficulty levels (easy, medium, hard).

### Functional Requirements:

*- AI opponent makes moves based on selected difficulty.*

*- Implement AI logic for easy (random), medium (blocking, winning tactics), and*

*hard (minimax) levels.*

*- Update player statistics and game history in SQLite.*

### User Interface Requirements:

*- Option to select AI difficulty level.*

*- Display game outcome (win/lose/draw) against AI.*

### Non-Functional Requirements:

*-* ***Performance:*** *Responsive AI moves within 92ms, especially in hard mode.*

*-* ***Usability:*** *Engaging gameplay experience against AI.*

## System Feature 6: Game History

### Description:

Displays player statistics (wins, losses, draws) and a list of all games played with detailed results.

### Functional Requirements:

*- Retrieve and display player statistics from our SQLite database.*

*- List all games with results (win/lose/draw), game level, date/time and last game*

*move shown.*

### User Interface Requirements:

*- History screen with statistics and game list.*

*- Button to display game board and option to do/undo moves.*

### Non-Functional Requirements:

*-* ***Performance:*** *Quick retrieval and display of game history.*

*-* ***Usability:*** *Clear presentation of player achievements and game details.*

# External Interface Requirements

## User Interfaces

* ***Sign-Up/Sign-In Screens:*** *For user authentication.*
* ***Mode Selector Screen:*** *For choosing game modes (Player vs Player, Player vs AI).*
* ***Game Board Screen:*** *Displays the 3x3 grid for gameplay.*
* **History Screen:** Shows player statistics and game history.

## Hardware Interfaces

* *Standard computer hardware (keyboard, mouse, display).*

## Software Interfaces

* *Qt framework for UI components.*
* *SQLite for data storage and retrieval.*

# Other Nonfunctional Requirements

## Performance Requirements

* *Response time: User interactions processed within 0.5ms.*
* *AI moves: AI logic executes moves within 92ms.*

## Security Requirements

* *Secure storage of user credentials (hashed passwords) in SQLite.*
* *Protection against unauthorized access to game data.*

## Software Quality Attributes

* ***Reliability:*** *Stable operation without crashes.*
* ***Usability:*** *Intuitive and easy-to-navigate user interface.*
* **Portability:** Runs on Windows platform.