Software Requirements Specification

Version: 1.0

Advanced Tic-Tac-Toe Game

Submitted by

Your Next Move Team

Team members:

Name	Section	ID
Mariam Ahmed Hamed	4	9220812
Mariam Mohamed Saeed	4	9220828
Malak Waleed Fouad	4	9220867
Menna Moutaz Elhosseiny	4	9220874
Maha Yasser Mohamed	4	9220878

Submitted to

Dr. Omar Nasr

Table of contents:

1. Introduction

1.1 Purpose:

The purpose of this document is to outline the functional and non-functional requirements for the development of our advanced Tic-Tac-Toe game. This project will incorporate user authentication, personalized game history, and an intelligent AI opponent.

1.2 Scope:

The scope of our project is to build a Tic-Tac-Toe game with many advanced features and a user-friendly GUI to ensure the satisfaction of the users. This game is developed using C++. It supports both player-vs-player and player-vs-AI modes and allows users to log in, track their game history, and analyze past games. The project also includes secure user management system and it underwent rigorous testing procedures.

1.3 Definitions, Acronyms and Abbreviations:

AI: Artificial Intelligence

GUI: Graphical User Interface

SRS: Software Requirements Specification

CI/CD: Continuous Integration/Continuous Deployment

Git: Version control system

SQLite: Lightweight database management system

1.4 References:

[1] "GeeksforGeeks," 16 Janruary 2023. [Online]. Available: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/. [Accessed 25 June 2024].

[2] "Great Learning," 30 April 2024. [Online]. Available: https://www.mygreatlearning.com/blog/alpha-beta-pruning-in-ai/. [Accessed 25 June 2024].

[3] "GeeksforGeeks," 13 June 2022. [Online]. Available: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/. [Accessed 25 June 2024].

[4] "SQLite," [Online]. Available: https://www.sqlite.org/cintro.html. [Accessed 25 June 2024].

[5] KDAB, "Unit Testing from Qt Creator," 22 March 2022. [Online]. Available: https://www.youtube.com/watch?v=N4pvvCToogM. [Accessed 25 June 2024].

[6] "ChatGPT".

[7] "GitHub," [Online]. Available: https://education.github.com/git-cheat-sheet-education.pdf. [Accessed 25 June 2024].

1.5 Overview:

This document is structured to detail the system's requirements, including the functional and non-functional aspects, user interfaces, system interfaces, and constraints.

2. Overall Description

2.1 Product Perspective:

The advanced Tic-Tac-Toe game is a standalone application that will provide an enhanced gaming experience with user authentication, personalized game history, and an AI opponent. It will be developed using C++ with a Qt-based GUI and will utilize SQLite for managing user data and game histories.

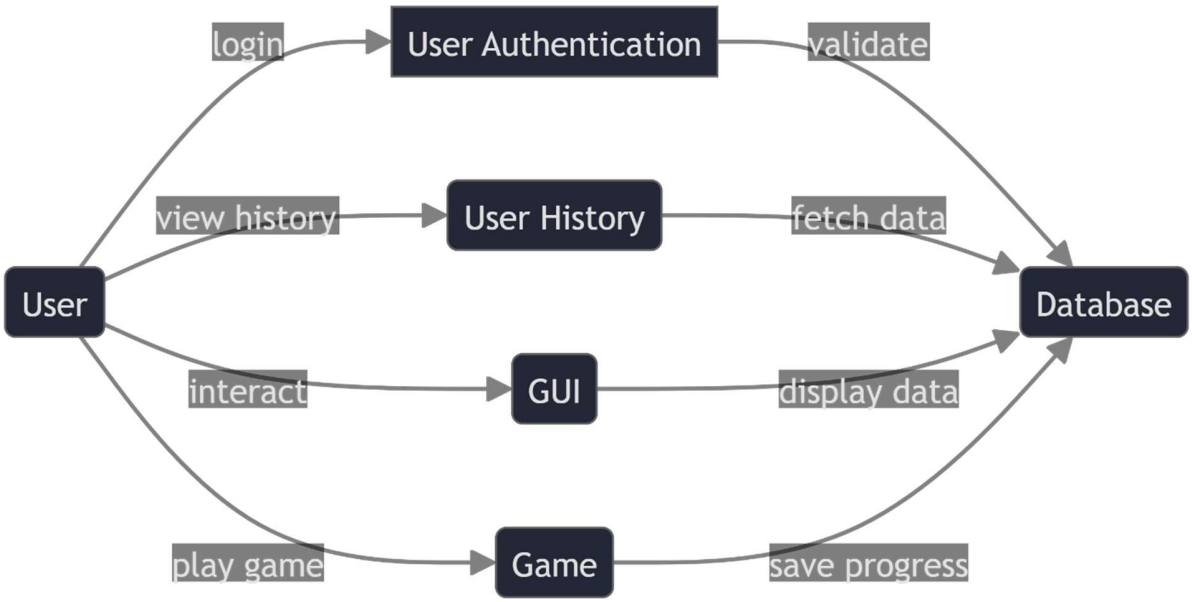

Figure 1 Main Blocks

2.2 Product Functions:

- User authentication: Secure login and registration system.
- Game modes: Player-vs-player and player-vs-AI.
- AI opponent: Implemented using the minimax algorithm with alpha-beta pruning.
- Game history: Saving and viewing past game sessions.
- GUI: Interactive and user-friendly interface.

2.3 User Classes and Characteristics:

- Players: create accounts, log in, play games, and view their game history.
- Developers: Need access to the main code or certain files and debugging tools.

- Database Administrators: manage the database system, including user accounts and game records.

2.4 Operating Environment:

- Operating Systems: Windows
- Development Tools: C++ compiler, Qt framework, SQLite
- Version Control: Git with GitHub
- CI/CD Tools: GitHub Actions

2.5 Design and Implementation Constraints

- C++ compiler (MinGW)
- C++ editor (Visual Studio code)
- SQLite wrapper
- Qt framework for GUI implementation
- Qt testing
- Git

2.6 Assumptions and Dependencies

- Players should have access to a computer with the necessary operating system
- Developers should have access to the required development tools and resources.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Authentication:

- Sign up: this function is used to register new players, accounts are reached through a unique email and secured by password chosen by the user. User's data is saved in a database file so it could be easily retrieved when needed.
- Log in: this function is used to load existing players' accounts using their emails and passwords matching them.
- Password Hashing: this function hashes users' passwords, so they would be saved hashed to ensure security.
- Secure History: each user can only access their own statistics and the games they participated in without any other games played by other users.

3.1.2 Game Environment:

- 3x3 Grid Displaying: this grid represents the Tic-Tac-Toe game board, each cell is to contain either X or O depending on whose player's turn it is.
- Game Mood Choice: the players have the choice to play against another human player or to play against an AI opponent.
- Turns Alternating: to ensure each player plays only one time a turn.
- Score Displaying: once player has won, the games displays the result and update each players scores
- Playing Again: the players are asked after each game if they want to play again and start a new game

3.1.3 AI Opponent:

- AI Mood: assigns the second player to an AI algorithm for the first player (human player) to play against
- AI move making: AI opponent makes strategic moves based on the current game state, using the minimax algorithm with alpha-beta pruning implemented in the system

3.1.4 Game Rules:

- Win Checking: this function is responsible to apply the Tic-Tac-Toe game rules, first player to have three successive cells with their symbol either in horizontal, vertical or diagonal manner wins. This function is called after each turn to check if the winning conditions occurred.

3.1.5 Game History:

- Game Statistics Saving: the project saves each player's game counts; win count, lose count, player-vs-player games count, player-vs-AI games count and total games count. Users can find such information in their account's history.
- Game Saving: each game played is saved in a database file, through which users are able to retrieve their past games and replay them move by move.

3.1.6 Graphical User Interface (GUI):

- Input Taking: users can interact with the game through clicks to make moves and choose certain options, in addition to using the keyboard to input string data.
- Game Display Graphics: the project provides forms for the game board and push buttons at each cell.
- Sign Up and Log In: the project also provides separate forms for the sign up and log in processes.
- Account Data: there is also forms that show users' data saved in their accounts.

3.2 Non-functional Requirements

3.2.1 Security Requirements:

- This project uses secure hashing algorithms for password storage.
- This project protects users' data from unauthorized access.

3.2.2 Maintainability Requirements:

- This project underwent comprehensive unit and integration tests.
- The game system starts properly even after sudden termination.

4. Other Requirements

4.1 Hardware Requirements:

- This system runs on desktop computers and laptops.
- User needs both a mouse and a keyboard to input data and interact with the game.

4.2 Software Requirements:

- This system uses SQLite database for saving users' data and game history.
- This system needs Qt libraries for GUI implementation.

4.3 Continuous Integration/Continuous Deployment (CI/CD):

- This system uses GitHub Actions for automated testing and deployment.
- The CI/CD pipelines include scripts for building, testing, and deploying the software

4.4 Version Control:

- This project uses Git for version control.
- The project's repository is hosted on GitHub.

5. Appendices

5.1 Glossary:

- Minimax Algorithm: a recursive algorithm used for decision-making in AI, particularly in turn-based games.
- Alpha-Beta Pruning: an optimization technique for the minimax algorithm that reduces the number of nodes evaluated.
- Hashing: a process of converting information into a fixed-size string of characters, which is typically a hash code.
- CI/CD: Continuous Integration and Continuous Deployment, practices in software development to ensure code quality and automate deployment.