Cairo University Faculty of Engineering

Dept. of Electronics and Electrical Communications

Second Year

Advanced Tic Tac Toe Game

Software Requirements Specification (SRS)

NAME	CODE
Islam Essam Salah El Deen	**9230220**
Mohamed Hisham Waheed El Deen	**9230819**
Ahmed Ibrahim Sabry	**9230117**
Amr Khaled Ashour	**9230632**
Mahmoud Mohamed Ezzat	**9230843**
Marwa Ibrahim Fathy	**9230856**

1. Introduction

1.1 Purpose

This document specifies requirements for a Tic Tac Toe implementation featuring an unbeatable AI opponent, user authentication, and game history tracking using SQLite.
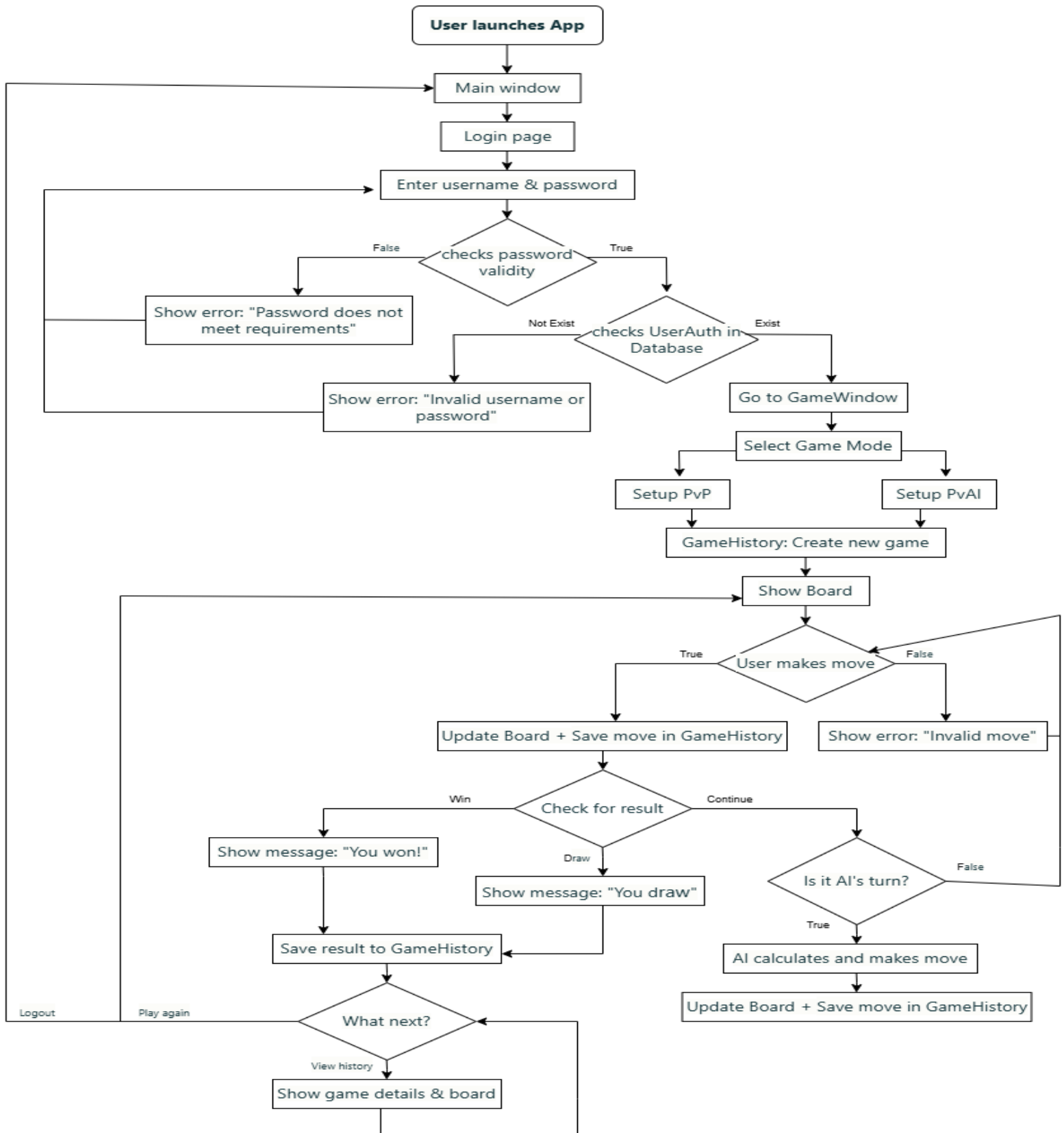
1.2 Scope

The system includes:

- Unbeatable AI using Minimax with Alpha-Beta Pruning

- User authentication system

- Persistent game history storage

- Qt-based GUI

- Move validation and game state management

2. Overall Description

2.1 Key Technologies

- C++17

- Qt 6.x (GUI Framework)

- SQLite3 (Database)

- Minimax Algorithm with Alpha-Beta Pruning

2.1 System Architecture

User launches App

↓

Main window

↓

Login page

↓

Enter username & password

↓

checks password validity

— False → Show error: "Password does not meet requirements"

— True → checks UserAuth in Database

checks UserAuth in Database:
- Not Exist → Show error: "Invalid username or password"
- Exist → Go to GameWindow

↓

Select Game Mode

— Setup PvP
— Setup PvAI

↓

GameHistory: Create new game

↓

Show Board

↓

User makes move

— True → Update Board + Save move in GameHistory
— False → Show error: "Invalid move"

Update Board + Save move in GameHistory

↓

Check for result

- Win → Show message: "You won!"
- Draw → Show message: "You draw"
- Continue → Is it AI's turn?

Is it AI's turn?
- True → AI calculates and makes move → Update Board + Save move in GameHistory
- False →

Save result to GameHistory

↓

What next?

- Logout
- Play again
- View history → Show game details & board

3. Functional Requirements

3.1 Core Gameplay

Requirement	Technical Implementation
3x3 game board	`Board class` with 2D array
Move validation	`isValidMove()` method
Win detection	`checkWinner()` method
Game reset	`resetBoard()` method

3.2 AI System

Requirement	Technical Implementation
Unbeatable AI	`minimax()` with Alpha-Beta pruning
Optimal move selection	`findBestMove()` method

3.3 User Authentication

Requirement	Technical Implementation
Secure login	SHA-256 hashed passwords
Session management	`UserAuth class`
Invalid login feedback	clear error messages

3.4 Game History

Requirement	Technical Implementation
Game recording	`GameHistory class` with SQLite
Game replay feature	Replay button in GameHistoryGUI to step through previous games

4. Non-Functional Requirements

4.1 Performance

We have written this report in a separate report.
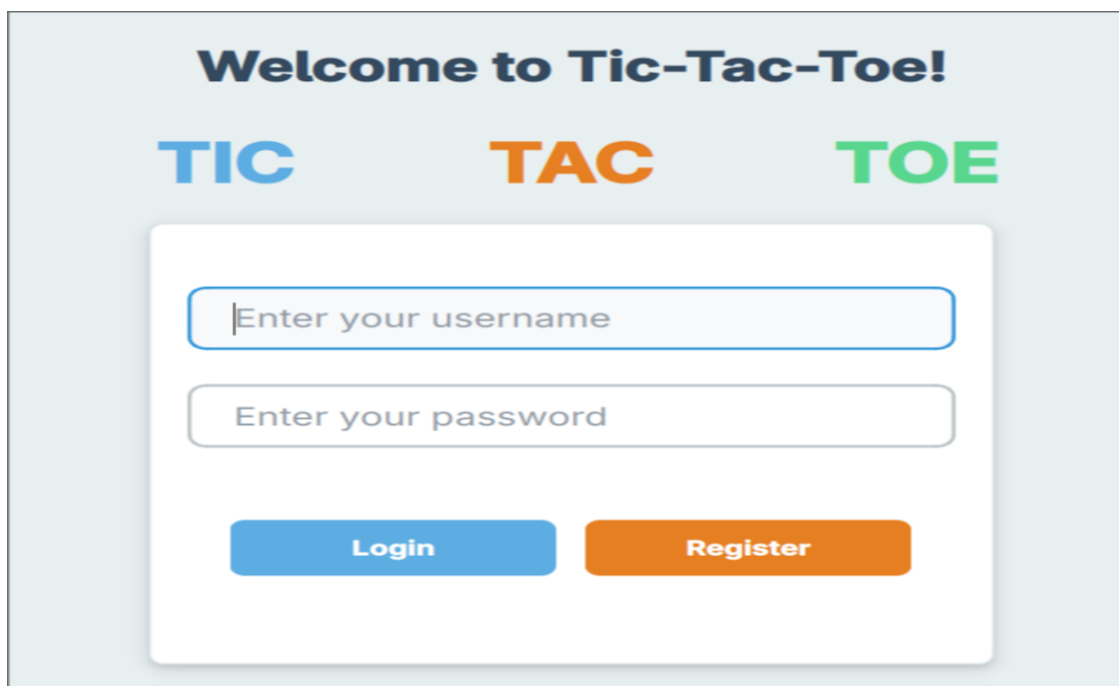
4.2 Security

- Password hashing using SHA-256

- SQL injection prevention via parameterized queries

- Error messages for incorrect credentials or sign-up validation

5. Interface Requirements

5.1 User Interfaces

The GUI includes a modern design with custom colors, fonts, rounded buttons and animated effects for moves and winning cells. The game window features Logout and View History buttons with distinctive styles. Game setup screens allow for mode and symbol selection. The GameHistoryGUI window displays a list of previous games, details, and a replay feature for each game.

Screen	Components
Login	QLineEdit, QPushButton
Game Board	QLabel grid, StatusBar
History	QTableView, QPushButton

Game History

Game	Date	Players	Result
3	2025-06-18 19:...	AI vs islam	AI Won
2	2025-06-18 19:...	islam vs ...	Draw
1	2025-06-18 19:...	islam vs AI	Draw

Game Information

Game ID:	1
Date:	2025-06-18 19:18:43
Player X:	islam
Player O:	AI
Winner:	Draw

Game Board

X	O	X
X	O	O
O	X	X

Replay Game

Moves

Move #	Player	Position
1	X	(0,0)
2	O	(1,1)
3	X	(2,2)
4	O	(0,1)

Game Board

View History   Logout

Game started – islam's turn (X)!

New Game