S1. Introduction

1.1 Purpose

The purpose of this document is to define the design specifications of the Tic Tac Toe game developed using C++ and Qt. It outlines the architecture, key modules, user interface, database interaction, and the design principles applied.

1.2 Scope

This software allows users to register/login, play Tic Tac Toe against AI or another player, view their game history, and securely store game data. The system supports user authentication and password security, and includes AI gameplay using Minimax with alpha-beta pruning.

2. Overall Description

2.1 System Architecture

The system consists of the following core components:

- **Register/Login Windows** – for authentication
- **Main Game Window** – displays the Tic Tac Toe board and controls
- **Game Logic** – handles rules, moves, and state
- **AI Logic** – provides competitive computer gameplay using Minimax + alpha-beta pruning
- **Database Module** – manages user data and saved games using SQLite

2.2 User Interface

The UI uses Qt (C++) and is styled with modern neon visuals using custom QSS stylesheets. It is structured with QMainWindow, QVBoxLayout, QGridLayout, QPushButton, QLabel, and QDialog for modals.

2.3 Database Structure

The backend is powered by SQLite. It uses two main tables:

- users (id INTEGER PRIMARY KEY, username TEXT, password TEXT)
- games (id INTEGER, user_id INTEGER, board TEXT, result TEXT, timestamp TEXT)

3. Design Details

3.1 Class Descriptions

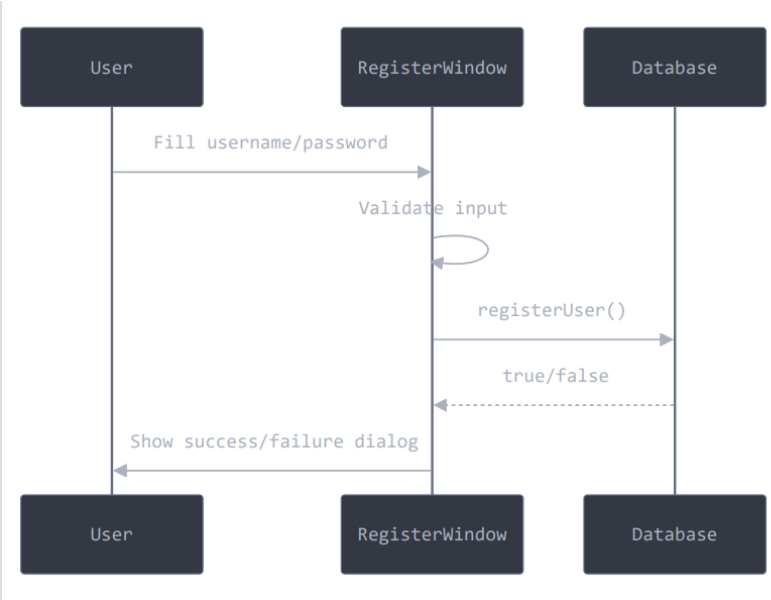
- **RegisterWindow**
  Handles user registration, including field validation, and real-time password strength checking.
- **LoginWindow**
  Verifies user credentials and opens the game window on success.
- **MainWindow**
  Hosts the main game logic, board display, buttons for control, and result tracking.
- **Game**
  Handles move logic, board state, win detection, and AI move generation.
- **Database**
  Connects to and interacts with SQLite to manage users and game records.

3.2 Important Functions

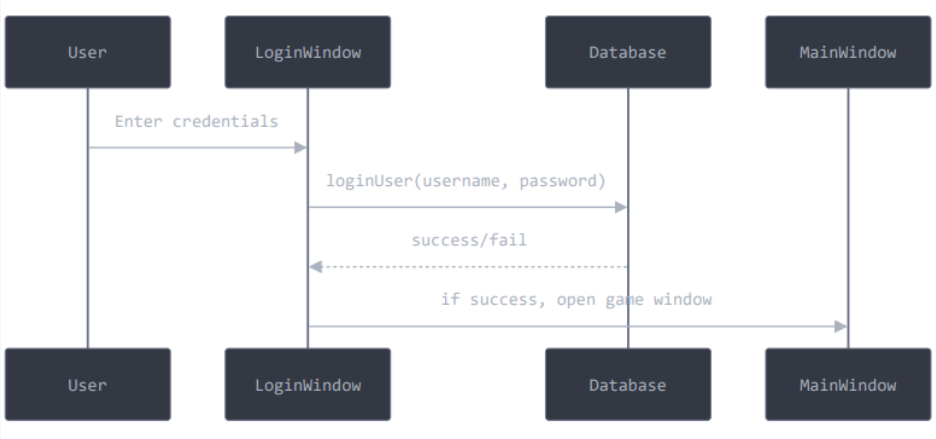
- attemptRegister() – Validates registration inputs and creates new users
- calculatePasswordStrength() – Grades password as Weak/Medium/Strong
- handleCellClick() – Responds to user's board interaction
- aiMove() – Triggers Minimax-based AI decision
- saveGame() – Records game data and results in the database

4. Sequence Diagrams

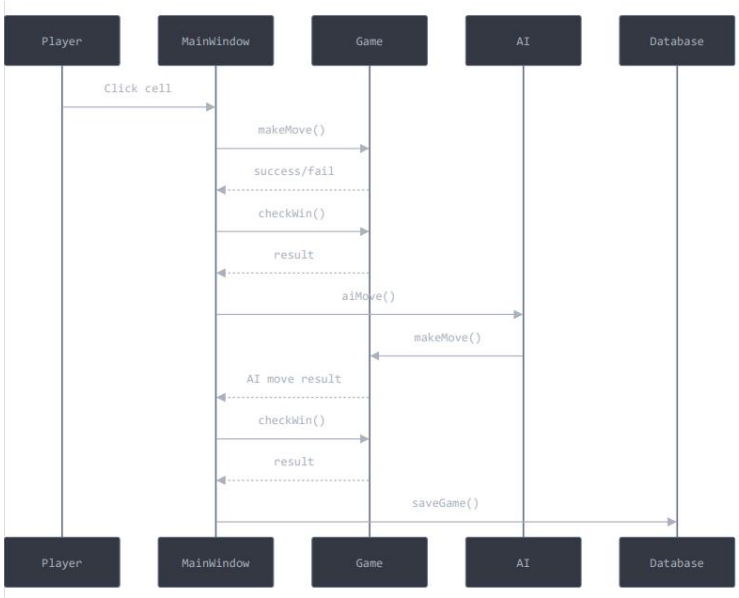
- **4.1 Register Sequence Diagram**



- **4.2 Login Sequence Diagram**

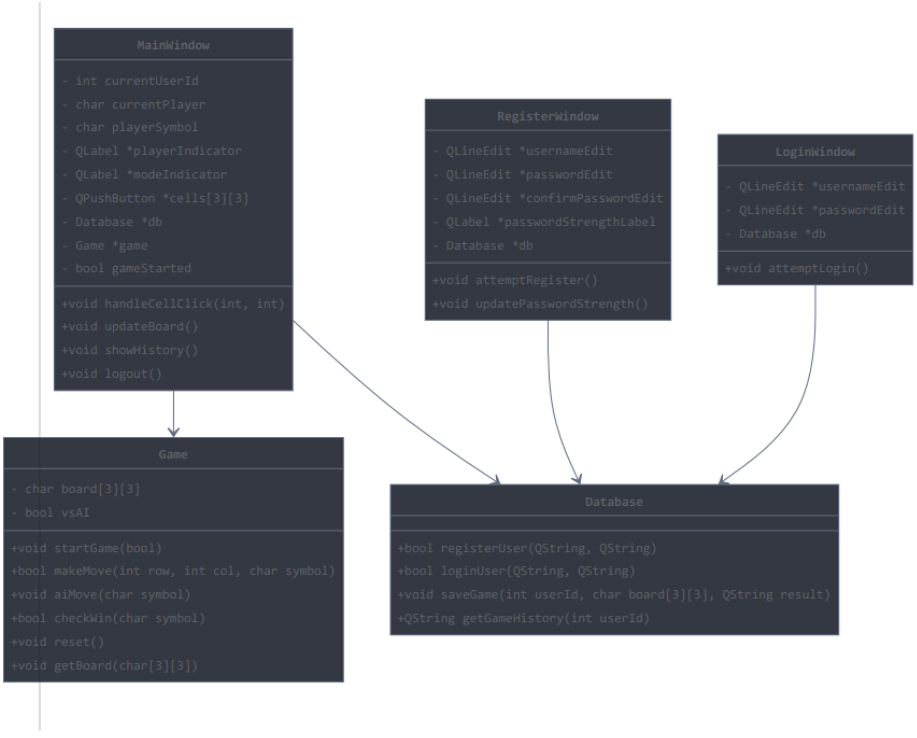


- **4.3 Gameplay Sequence Diagram**



5. Class Diagram

- **5.1 Class Diagram for System Structure**



6. Design Considerations

- GUI uses layouts (QVBoxLayout, QGridLayout) for flexible resizing
- Signal-slot connections are used for modular communication

- Passwords are validated locally for format and strength
- AI uses depth-limited Minimax with pruning for performance
- SQLite database is lightweight and persistent across sessions

7. Future Enhancements

- Add sound effects for moves and game results
- Implement online multiplayer via TCP/UDP sockets
- Provide a player statistics page with win rates, average game time, etc.
- Offer customizable board themes and animations