Software Design Specification

For

Advanced Tic Tac Toe

Prepared by TicTacTactics

Submitted to: Dr. Omar Nasr

Table of Contents

[1. Introduction 2](#_Toc170323392)

[1.1 Purpose 2](#_Toc170323393)

[1.2 Scope 2](#_Toc170323394)

[1.3 Definitions, Acronyms and Abbreviations. 2](#_Toc170323395)

[2. System Overview 2](#_Toc170323396)

[2.1 High-Level-Description 2](#_Toc170323397)

[2.2 Main Features 3](#_Toc170323398)

[3. System Architecture 3](#_Toc170323399)

[3.1 High-Level Architecture Diagram 3](#_Toc170323400)

[3.2 Class Diagram 3](#_Toc170323401)

[3.3 Module Descriptions 3](#_Toc170323402)

[4. Module Descriptions 4](#_Toc170323403)

[4.1 Class Descriptions 4](#_Toc170323404)

[4.2 Data structures 4](#_Toc170323405)

[4.3 Algorithms 4](#_Toc170323406)

[5. User Interface Design 5](#_Toc170323407)

[5.1 Screens and layouts 5](#_Toc170323408)

[5.2 User Interactions 5](#_Toc170323409)

[6. Data Flow and Control Flow 5](#_Toc170323410)

[6.1 Data Flow Diagrams 5](#_Toc170323411)

[6.2 Control Flow Diagrams 5](#_Toc170323412)

[7. External Interfaces 6](#_Toc170323413)

[8. Constraints and Assumptions 6](#_Toc170323414)

[9. Appendices 7](#_Toc170323415)

# Introduction

## Purpose

The purpose of this document is to provide a comprehensive software design specification for the development of a Tic Tac Toe game using Qt and C++. It outlines the system architecture, design decisions, user interface details, and implementation strategies.

## Scope

The scope includes implementing a graphical user interface (GUI) for user sign-up and login functionalities, game mode selection (Player vs Player, Player vs AI with three difficulty levels), game play mechanics (placing X or O), game logic (win, lose, draw conditions), and database integration for storing user profiles and game history.

## Definitions, Acronyms and Abbreviations.

* **GUI**: Graphical User Interface.
* **AI**: Artificial Intelligence.
* **SQLite**: A lightweight, embeddable database engine.
* **UML**: Unified Modeling Language.
* **Minimax Algorithm**: An algorithm used in decision-making and game theory.

# System Overview

## High-Level-Description

The Tic Tac Toe game allows users to interact through a visually appealing GUI. Upon launching the application, users can sign up for a new account or log in to an existing one. After authentication, they can choose from different game modes: Player vs Player for local multiplayer, Player vs AI with Easy, Medium, and Hard difficulty levels. The game tracks and stores each session's results in an SQLite database, enabling users to view their game history.

## Main Features

* **User Authentication**: Enables new users to create accounts securely with password hashing and existing users to log in.
* **Game Modes**: Offers multiple modes including local multiplayer and AI opponents with varying difficulty levels (Easy, Medium, Hard).
* **Game History**: Tracks and stores game outcomes (win, lose, draw) along with timestamps and unique IDs in an SQLite database.
* **Database Integration**: Utilizes SQLite for persistent storage of user profiles and game data.

# System Architecture

## High-Level Architecture Diagram

The architecture diagram illustrates the structure of the Tic Tac Toe game system, highlighting components such as:

* **Presentation Layer**: GUI components for user interaction.
* **Business Logic Layer**: Implements game rules, AI algorithms, and database operations.
* **Data Access Layer**: Manages interactions with the SQLite database

## Class Diagram

The UML class diagram depicts the key classes and their relationships within the system:

* **User**: Manages user profiles and authentication.
* **Game**: Represents the game session, including board state and game logic.
* **AI**: Implements AI strategies (Easy, Medium, Hard) using algorithms like minimax for decision-making.

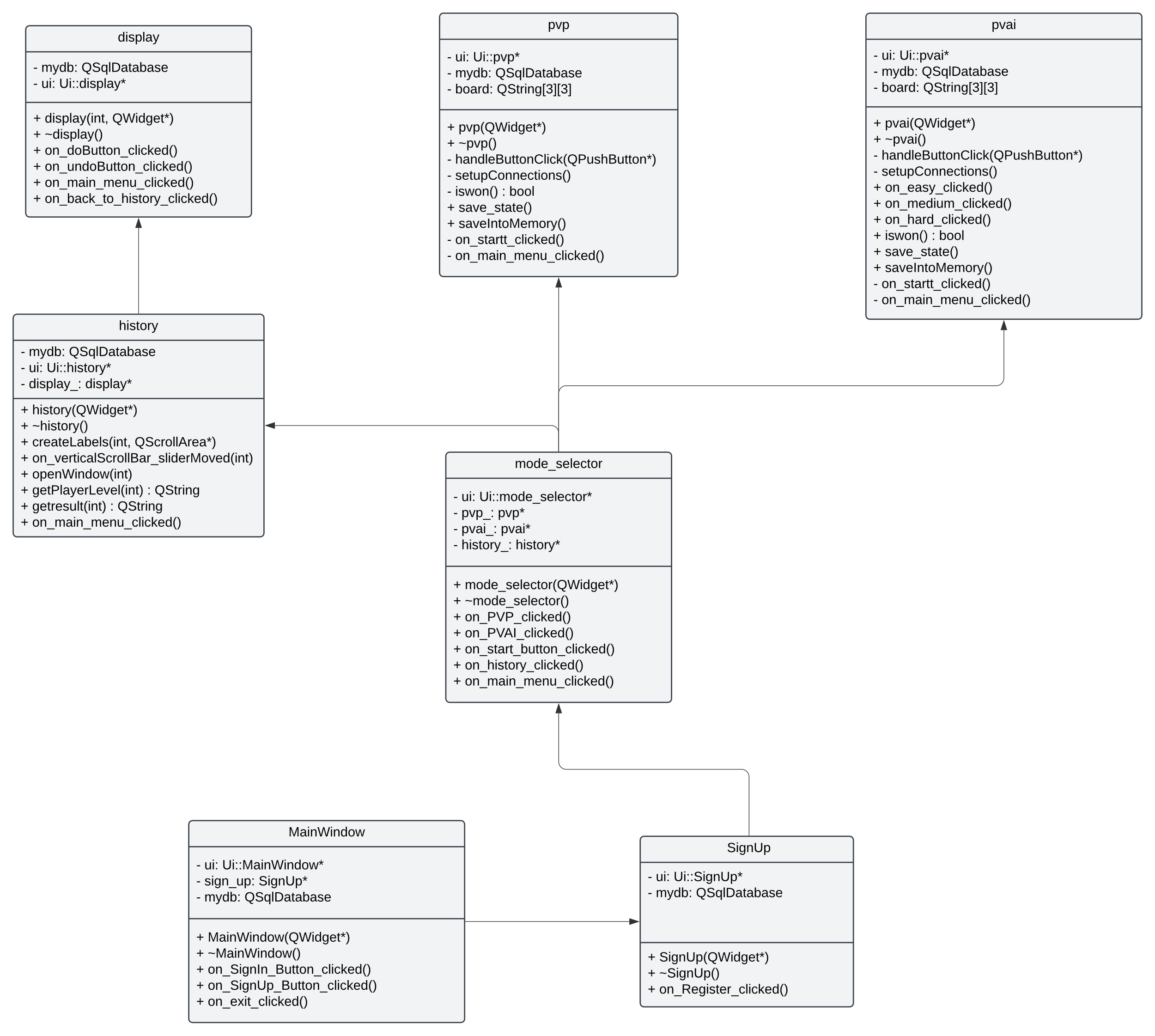


Figure 1: Class Diagram

## Module Descriptions

* **User Authentication Module**: Handles user registration, login, and password hashing.
* **Game Logic Module**: Implements Tic Tac Toe game rules, including win conditions and turn management.
* **Database Module**: Manages SQLite database interactions for storing user data and game history.

# Module Descriptions

## Class Descriptions

* **User Class**:
* Attributes: username, hashed\_password, win\_count, draw\_count, lose\_count.
* Methods: hash(passwordData,QCryptographicHash::Sha256),saveIntoMemory(),save\_state()
* **Game Class**:
* Attributes: board\_state, current\_player, game\_status.
* Methods: update(), iswon(), isdraw().
* **AI Class**:
* Attributes: difficulty\_level (Easy, Medium, Hard).
* Methods: computer\_turn\_easy(), computer\_turn\_medium(),

minimax(int depth, bool isMax, int alpha, int beta, findBestMove().

## Data structures

* **Game Board Representation**: 2D array (3x3) to represent the Tic Tac Toe board.
* **Database Schema**:
* **Users Table**: username, hashed\_password, win\_count, draw\_count, lose\_count
* **Games Table**: game\_id, player1, player2, game\_mode, game\_result, timestamp and game moves with its order.

## Algorithms

**AI Algorithms**:

* **Minimax Algorithm**: Used for implementing the hard level AI logic.
* **Blocking and Winning Tactics**: Used for the medium level AI.
* **Random Move Algorithm**: Used for easy level AI.

# User Interface Design

## Screens and layouts

* **Login/Sign up Window**: Allows users to sign up for a new account or log in with existing credentials securely.
* **Mode Selection Window**: Offers options for Player vs Player, Player vs AI (Easy, Medium, Hard), and access to the History Window.
* **Game Board Display**: Visual representation of the Tic Tac Toe board with interactive buttons for player moves and turn indication.

## User Interactions

* **Game Play Interaction**: Users click on empty spaces on the board to place their X or O. Error messages display if a space is already occupied.
* **End Game Interaction**: After a game concludes (win, lose, draw), users can choose to play again, switch game modes, or view their game history.

# Data Flow and Control Flow

## Data Flow Diagrams

Illustrate the flow of data within the system:

* **User Input**: Sign up/login credentials, move selection.
* **Game Logic**: Updates game state, checks win conditions.
* **Database Interaction**: Saves user profiles, game history.

## Control Flow Diagrams

Show the sequence of operations during gameplay:

* **Player Turn Management**: Switches between players, updates game board.
* **AI Decision Making**: Determines optimal moves based on difficulty level.
* **Game Outcome Check**: Verifies win, lose, or draw conditions.

# External Interfaces

* **SQLite Database**: Handles persistent storage for user profiles (sign up, login) and game history.
* **GUI Framework (Qt)**: Provides graphical components for the user interface and event handling.

# Constraints and Assumptions

* **Hardware Requirements**: Runs on standard desktop and laptop computers.
* **Security Assumptions**: Passwords are hashed for storage in the database, and sensitive information (like passwords) is not displayed in plain text.

# Appendices

* + Detailed flowchart illustrating the sequence of actions from sign up/login to gameplay and database interaction

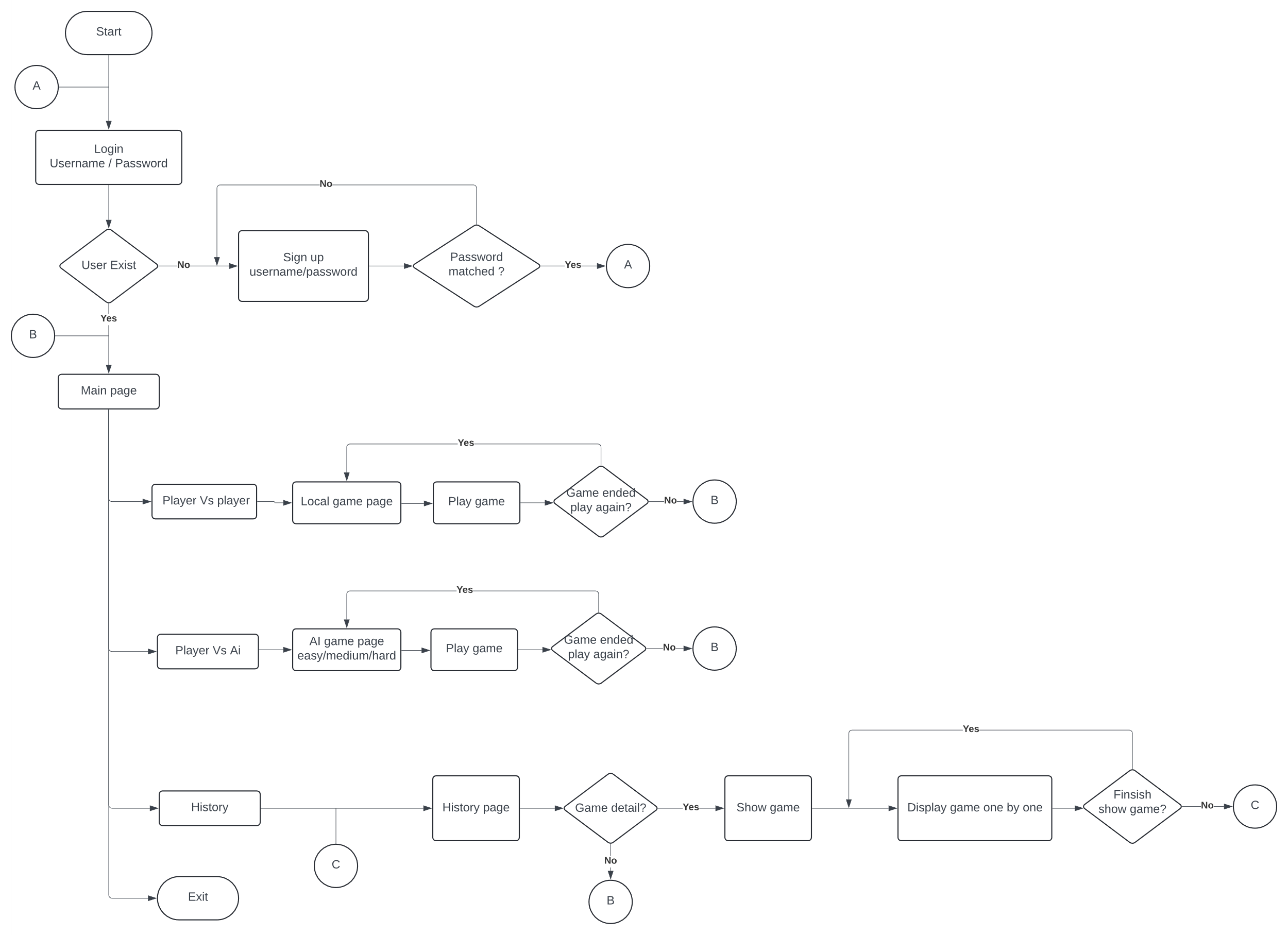


Figure 2: System Flow

**Additional Notes**: Any supplementary information such as third-party libraries used, development environment setup instructions.