Software Design Specification (SDS)

Project: Advanced Tic Tac Toe Game

1. Purpose

This document defines the architecture and design details for the Advanced Tic Tac Toe Game, ensuring modular, extensible, and maintainable software development. It includes class diagrams and sequence diagrams for key functionalities.

2. Software Architecture

Architecture Type: Layered Modular Architecture
Presentation Layer (Qt UI - GameWindow class)
Business Logic Layer (Game Mechanics - GameLogic class)
Data Access Layer (Authentication and History - UserAuth class using SQLite)

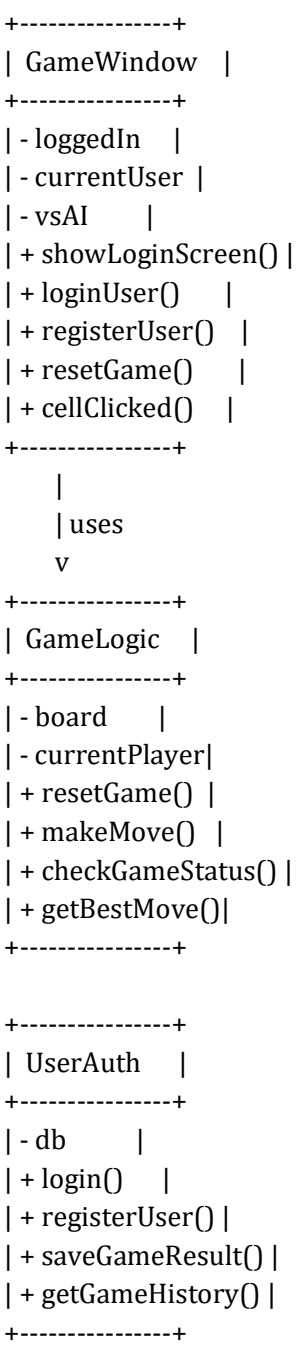
The application follows separation of concerns:
- UI handles display and user interactions.
- Game Logic handles moves, game status checks, AI decision-making.
- Database manages user accounts and game history.

3. Design Details

3.1 Class Overview

Class	Purpose
GameWindow	Handles the UI, game flow, user interactions (clicks, resets, login/logout).
GameLogic	Manages the game state, player moves, win/draw conditions, AI decision making.
UserAuth	Manages user registration, login, and game history storage/retrieval via SQLite.

3.2 Class Diagram (UML)



3.3 Key Class Descriptions

Descriptions of GameWindow, GameLogic, and UserAuth classes and their responsibilities.

3.4 Sequence Diagrams

3.4.1 User Login Sequence

User -> GameWindow: Enter username/password
GameWindow -> UserAuth: login(username, password)
UserAuth -> Database: Query user credentials
Database -> UserAuth: Return result
UserAuth -> GameWindow: Success/Failure
GameWindow -> User: Proceed to Game Screen or Show Error

3.4.2 Gameplay vs AI Sequence

User -> GameWindow: Click on a cell
GameWindow -> GameLogic: makeMove(cell)
GameLogic -> GameLogic: Update board, switch player
GameLogic -> GameLogic: checkGameStatus()
GameWindow -> GameLogic: getBestMove() (if vs AI and AI's turn)
GameLogic -> GameLogic: Run minimax algorithm
GameLogic -> GameLogic: Update board with AI move
GameLogic -> GameLogic: checkGameStatus()
GameWindow -> User: Update board, show winner if any

3.5 Database Design (Tables)

Table	Columns
users	username (TEXT PRIMARY KEY), password (TEXT)
game_history	id (INTEGER), username (TEXT), datetime (TEXT), result (TEXT), opponent (TEXT)

4. Technologies and Tools

Technology	Purpose
C++	Core language
Qt	GUI library (QMainWindow, QPushButton, QStackedWidget, etc.)
SQLite	Lightweight embedded database
Google Test (gtest)	Unit testing framework
QTestLib	Qt testing framework for GUI testing

5. Design Considerations

Performance: Minimax is optimized using alpha-beta pruning to prevent unnecessary calculations.
Security: Passwords are stored in plain SQLite (could be improved with hashing).
Scalability: Architecture is modular and can support online multiplayer, statistics tracking, and UI themes customization.