# Testing Documentation

## For Tic-Tac-Toe

## 1. Introduction

- This document provides a comprehensive testing plan for the Tic-Tac-Toe application. The application is a Qt-based C++ implementation featuring user authentication, player vs. player (PvP) and player vs. AI (PvAI) game modes, AI difficulty levels, game history with replay functionality, and settings management. The goal of testing is to ensure the application is functional, reliable, and user-friendly.

## 2. Test Objectives

- Verify that all features (authentication, game logic, AI, history, settings) work as intended.
- Ensure the application handles edge cases and invalid inputs gracefully.
- Validate the user interface for responsiveness and clarity.
- Confirm that game history is accurately saved and replayed.
- Test the AI's behavior across different difficulty levels.

## 3. Scope

**In-Scope**

- User authentication (login, registration, guest mode).
- Game modes: Player vs. Player and Player vs. AI.
- AI difficulty levels: Easy, Medium, Hard.
- Game board interactions (move placement, win/tie detection, reset).
- Game history saving, loading, and replay functionality.
- Settings window (switch account, history view, exit, back).
- UI responsiveness and error messaging.

**Out-of-Scope**

- Performance testing under high load.
- Security testing of password hashing (beyond basic functionality).
- Cross-platform compatibility (assumed to be Windows-based for this project).
- Localization testing for multiple languages.

## 4. Test Environment

- **Operating System**: Windows 11
- **IDE**: Qt Creator
- **Compiler**: MinGW
- **Libraries**: Qt 6.x
- **Hardware**: Standard laptop with 16GB RAM and 2.3GHz CPU
- **Test Data**: Sample usernames, passwords, and game sessions

## 5. Test Strategy

- **Manual Testing**: Primary method due to the project's scale and UI-driven nature.
- **Unit Testing**: Limited to critical components (game logic, AI move generation) using QTest if time permits.
- **Functional Testing**: Verify each feature against requirements.
- **Edge Case Testing**: Test invalid inputs, boundary conditions, and error scenarios.
- **Regression Testing**: Re-test features after fixing bugs.
- **Exploratory Testing**: Ad-hoc testing to identify unexpected issues.

## 6. Test Cases

### 6.1. User Authentication (MainWindow, RegisterWindow)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| AUTH-01 | Register new user | 1. Open Register window.<br>2. Enter username "testuser", password "Password123", confirm password "Password123".<br>3. Click Register. | Success message: "REGISTERED SUCCESSFULLY!". Window closes. | Pass: User registered and saved in registered_users.json. |

| | | | | |
|---|---|---|---|---|
| AUTH-02 | Register with weak password | 1. Open Register window.<br>2. Enter username "testweak", password "123", confirm password "123".<br>3. Click Register. | Warning: "Password is too weak. Use a stronger password." | Pass: Registration fails, no user added. |
| AUTH-03 | Register with medium password (proceed) | 1. Open Register window. 2. Enter username "testmed", password "pass123", confirm password "pass123<br>3. Click Register.<br>4. Confirm "Proceed anyway?" with Yes. | Success message: "REGISTERED SUCCESSFULLY!". Window closes. | Pass: User registered. |
| AUTH-04 | Register with existing username | 1. Register "testuser" (from AUTH-01).<br>2. Attempt to register "testuser" again with any password.<br>3. Click Register. | Warning: "USERNAME ALREADY EXISTS!" | Pass: User logged in, username passed to GameModeWindow. |
| AUTH-05 | Login with invalid password | 1. Enter username "testuser", password "Password123".<br>2. Click Login. | Success: "LOGGED IN SUCCESSFULLY!". GameModeWindow opens. | Pass: User logged in, username passed to GameModeWindow. |
| AUTH-06 | Login with invalid password | 1. Enter username "testuser", password "WrongPass".<br>2. Click Login. | Warning: "Invalid username or password." | Pass: Login fails, stays on MainWindow. |
| AUTH-07 | Login with non-existent user | 1. Enter username "nonuser", password "Password123".<br>2. Click Login. | Warning: "Invalid username or password." | Pass: Login fails. |
| AUTH-08 | Guest login | 1. Click Guest button. | GameModeWindow opens with empty username. | Pass: GameModeWindow opens, no username saved. |

| | | | | |
|---|---|---|---|---|
| AUTH-09 | Clear login fields | 1. Enter username and password.<br>2. Click Clear button. | Username and password fields are cleared. | Pass: Fields are empty. |

## 6.2. Game Mode Selection (GameModeWindow)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| MODE-01 | Select Player vs. Player | 1. Click "Player vs Player" button. | PlayerNameWindow opens. | Pass: PlayerNameWindow displayed. |
| MODE-02 | Select Player vs. AI | 1. Click "Player vs AI" button. | AIDifficultyWindow opens, GameModeWindow hides. | Pass: AIDifficultyWindow displayed. |
| MODE-03 | Open Settings | 1. Click Settings button. | SettingsWindow opens, GameModeWindow hides. | Pass: SettingsWindow displayed. |

## 6.3. Player Names (PlayerNameWindow)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| NAME-01 | Enter valid player names | 1. Open PlayerNameWindow.<br>2. Enter "Alice" for Player 1, "Bob" for Player 2.<br>3. Click OK. | GameBoard opens with names "Alice" and "Bob". | Pass: GameBoard displayed, names set correctly. |
| NAME-02 | Empty player names | 1. Leave both name fields empty.<br>2. Click OK. | No action, debug log: "Player names cannot be empty". | Pass: GameBoard not opened. |
| NAME-03 | Clear player names | 1. Enter names in both fields.<br>2. Click Clear. | Both name fields are cleared. | Pass: Fields are empty. |

## 6.4. AI Difficulty and Symbol Selection (AIDifficultyWindow, SymbolWindow)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| AI-01 | Select Easy difficulty | 1. Open AIDifficultyWindow.<br>2. Click Easy button. | SymbolWindow opens. | Pass: SymbolWindow displayed. |

| | | | | |
|---|---|---|---|---|
| AI-02 | Select Medium difficulty | 1. Click Medium button. | SymbolWindow opens. | Pass: SymbolWindow displayed. |
| AI-03 | Select Hard difficulty | 1. Click Hard button. | SymbolWindow opens. | Pass: SymbolWindow displayed. |
| AI-04 | Back to GameMode | 1. Click Back button. | GameModeWindow reappears. | Pass: GameModeWindow displayed. |
| AI-05 | Select X symbol | 1. Open SymbolWindow.<br>2. Select "X".<br>3. Click OK. | GameBoard opens, player symbol is 'X', AI is 'O'. | Pass: GameBoard configured correctly. |
| AI-06 | Select O symbol | 1. Select "O".<br>2. Click OK. | GameBoard opens, player symbol is 'O', AI is 'X'. | Pass: GameBoard configured correctly. |

## 6.5. Game Board (GameBoard)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| GAME-01 | Player vs. Player: Valid move | 1. Start PvP game.<br>2. Click button (0,0). | Button (0,0) shows "X" in red, status updates to Player 2 ("O"). | Pass: Move placed, status updated. |
| GAME-02 | Player vs. Player: Win condition | 1. Place X in (0,0), (0,1), (0,2). | Message: "Player 1 wins!", score updates, board resets. | Pass: Win detected, history saved. |
| GAME-03 | Player vs. Player: Tie | 1. Fill board without three-in-a-row (e.g., X,O,X,O,O,X,O,X,O). | Message: "It's a tie!", tie score increments, board resets. | Pass: Tie detected, history saved. |
| GAME-04 | Player vs. AI: AI move | 1. Start PvAI game (Easy).<br>2. Make player move.<br>3. Wait for AI move. | AI places move, button shows AI symbol (e.g., 'O' in blue). | Pass: AI responds, board updated. |
| GAME-05 | Player vs. AI: Hard difficulty | 1. Start PvAI (Hard).<br>2. Play game. | AI makes strategic moves (e.g., blocks player wins). | Pass: AI plays optimally. |
| GAME-06 | Reset board | 1. Make several moves.<br>2. Click Reset button. | Board clears, status resets to Player 1. | Pass: Board and status reset. |
| GAME-07 | Return to GameMode | 1. Click Return button. | GameModeWindow opens, GameBoard hides. | Pass: GameModeWindow displayed. |

| | | | | |
|---|---|---|---|---|
| GAME-08 | Invalid move (filled cell) | 1. Click a button with 'X'.<br>2. Click same button again. | No action, debug log: "Invalid button click or already filled". | Pass: Move ignored. |

## 6.6. AI Logic (AIPlayer)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| AI-LOGIC-01 | Easy AI: Random move | 1. Start PvAI (Easy).<br>2. Make player move. | AI places move in random empty cell. | Pass: Move is valid and random. |
| AI-LOGIC-02 | Medium AI: Strategic move | 1. Start PvAI (Medium).<br>2. Place two player symbols in a row.<br>3. Wait for AI move. | AI blocks player's winning move or sets up own win. | Pass: Move is strategic. |
| AI-LOGIC-03 | Hard AI: Optimal move | 1. Start PvAI (Hard).<br>2. Play multiple games. | AI consistently blocks wins and seeks own wins. | Pass: AI is unbeatable or nearly so. |
| AI-LOGIC-04 | AI evaluates win | 1. Simulate board with AI win (three AI symbols in row). | evaluateBoard returns 10. | Pass: Correct score returned. |
| AI-LOGIC-05 | AI evaluates loss | 1. Simulate board with player win. | evaluateBoard returns -10. | Pass: Correct score returned. |

## 6.7. Game History (GameHistory, HistoryWindow)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| HIST-01 | Save game history | 1. Play PvP game to completion.<br>2. Check history/username_game_history.json. | Game details (players, mode, outcome, moves, timestamp) saved. | Pass: JSON file contains correct data. |
| HIST-02 | Load game history | 1. Open HistoryWindow.<br>2. View history list. | List shows all saved games with correct details. | Pass: Games displayed accurately. |
| HIST-03 | Replay game | 1. Select a game in HistoryWindow.<br>2. Click Replay button. | Moves replayed in order, 1 second per move, with | Pass: Replay matches saved moves. |

| | | | | |
|---|---|---|---|---|
| | | | correct symbols/colors. | |
| HIST-04 | Clear history | 1. Click Clear History button.<br>2. Check history list and JSON file. | List is empty, JSON file contains empty array. | Pass: History cleared. |
| HIST-05 | No username history | 1. Play as guest.<br>2. Check history saving. | Debug log: "No username provided, skipping game history save". | Pass: No history saved. |

### 6.8. Settings (SettingsWindow)

| Test Case ID | Description | Steps | Expected Result | Pass/Fail Criteria |
|---|---|---|---|---|
| SET-01 | Switch account | 1. Click Switch Account button. | MainWindow opens, SettingsWindow closes. | Pass: MainWindow displayed. |
| SET-02 | View history | 1. Click History button. | HistoryWindow opens. | Pass: HistoryWindow displayed. |
| SET-03 | Exit application | 1. Click Exit button. | Application closes. | Pass: Application terminates. |
| SET-04 | Back to GameMode | 1. Click Back button. | GameModeWindow opens, SettingsWindow hides. | Pass: GameModeWindow displayed. |

## 7. Test Execution

- **Execution Process**: Each test case will be executed manually by following the steps outlined. Results will be recorded in a test log (e.g., spreadsheet or document) noting Pass/Fail status and any defects.
- **Defect Reporting**: Issues will be logged with:
    .1. Test Case ID
    .2. Description of failure
    .3. Steps to reproduce
    .4. Screenshot (if applicable)
    .5. Severity (Critical, Major, Minor)

- **Retesting**: Fixed defects will be retested to confirm resolution.

## 8. Test Deliverables

- Test documentation (this document).
- Test log with results for each test case.
- Defect reports for any issues found.
- Screenshots or recordings of critical test scenarios ( AI behavior, replay).

## 9. Risks and Assumptions

- **Risks**:
  - .1. Limited time for exhaustive testing may miss edge cases.
  - .2. Manual testing is prone to human error.
  - .3. Qt UI issues (e.g., missing widgets) may require code fixes.
- **Assumptions**:
  - .1. Qt environment is properly configured.
  - .2. Test environment mirrors the development environment.
  - .3. JSON files (registered_users.json, game history) are writable.

## 10. Approval

| **Role** | **Name** | **Signature** |
|---|---|---|
| Tester | [Your Name] | |
| Project Advisor | [Advisor Name] | |