# Test Documentaion

## Purpose of Testing

The purpose of this testing effort is to verify that the core functionality of the Tic Tac Toe application meets design expectations. It includes validating user interactions, gameplay logic (PvP and AI), input validation, and persistence mechanisms.

## Scope

This testing process covers the following modules:

* GameWindow (main game logic including PvP and AI)
* DifficultyWindow (AI level selection)
* UserManager (handling user data and history)
* Security (input validation and user verification)
* DifficultyWindow (AI level selection)
* HistoryWindow (displaying past results)
* RegisterWindow (user registration)

**Tests were conducted using the Qt Test framework (QtTest) in Qt 6.9.1 on Windows 11.**

## Types of Testing

* Unit Testing
* Functional/UI Testing
* AI Behavior Testing
* Boundary Testing
* Negative Testing (invalid input, empty fields, etc.)

## Tools Used

* Qt 6.9.1
* Qt Creator IDE
* Qt Test Framework (QTestLib)
* Mock classes (e.g., MockGameWindow)

## Test Environment

* OS: Windows 11 64-bit
* Compiler: MinGW 64-bit
* Qt Version: 6.9.1
* Target Platform: Desktop GUI

## Summary of test classes

### RegisterwindowTest

| **Test Case** | **Description** | **Expected Result** | **Result** |
| --- | --- | --- | --- |
| Empty fields warning | Submitting form with empty username/password | Warning message shown | Pass |
| Duplicate username | Submitting an already registered user | Registration rejected | Pass |
| Successful registration | Valid input, new user | Goes to HomeWindow | Pass |
| Back button | Clicks "Back" from RegisterWindow | MainWindow is shown | Pass |

### GameWindowTest(PvP ,PvAI)

| **Test Case** | **Description** | **Expected Result** | **Result** |
| --- | --- | --- | --- |
| Button marks X | Clicking empty button | Shows X | Pass |
| Alternate turns | Players take alternating turns | Alternates between X and O | Pass |
| Prevent double click | Click same button twice | No change after first click | Pass |
| Board reset | Restarting game | Board cleared | Pass |
| Win detection (rows) | Fill a row with same symbol | Win detected | Pass |
| Draw detection | Full board, no win | Game ends in draw | Pass |
| AI Hard blocks player | Player about to win | AI blocking the winning cell | Pass |

### DifficultyWindowTest

| **Test Case** | **Description** | **Expected Result** | **Result** |
| --- | --- | --- | --- |
| Select Easy | Click "Easy" button | GameWindow opens with AI difficulty = 1 | Pass |
| Select Medium | Click "Medium" button | GameWindow opens with AI difficulty = 2 | Pass |
| Select Hard | Click "Hard" button | GameWindow opens with AI difficulty = 3 | Pass |
| Back Button | Click "Back" | Previous window (PlayMode) is shown | Pass |

### SecurityTest

| **Test Case** | **Description** | **Expected Result** | **Result** |
| --- | --- | --- | --- |
| Accept Valid Username/Password | Meets length and format requirements | Returns true | Pass |
| Reject Empty Username | Blank username | Returns false | Pass |
| Reject Short Password | Too short | Returns false | Pass |
| Reject Password Without Numbers | No digits | Returns false | Pass |

### UserManagerTest

| **Test Case** | **Description** | **Expected Result** | **Result** |
| --- | --- | --- | --- |
| Register New User | Add user to JSON | User is persisted and exists | Pass |
| Prevent Duplicate User | Register same username twice | Registration fails | Pass |
| Authenticate Correctly | Valid credentials | Returns true | Pass |
| Reject Invalid Login | Incorrect password | Returns false | Pass |
| Save Game | Save a dummy game | Entry appears in history | Pass |
| Get History | Retrieve game history | Returns QJsonArray | Pass |

### HistoryWindowTest

| **Test Case** | **Description** | **Expected Result** | **Result** |
| --- | --- | --- | --- |
| Load Empty History | User with no games | "No history found" or empty list | Pass |
| Load Saved Games | User has saved games | Game list displayed in table | Pass |
| Back Button | Click "Back" | HomeWindow is shown | Pass |

## Known limitations or Issues

* MessageBox UI may pop up during automated tests unless mocked.
* AI sometimes needs forced calls in tests due to reliance on internal game state (e.g., player1Turn).
* Currently uses manual mocks; dependency injection could improve testability.

## Conclusion

All core gameplay and registration features have passed their defined test cases. The AI behavior performs as expected across easy, medium, and hard difficulties. The application is stable and performs correctly under normal and edge input conditions.