**Testing Documentation**

**Purpose**

The purpose of this documentation is to outline the testing strategies, detailed test cases, and results for the unit and integration tests implemented using Google Test for the MainWindow, Startgame, gameHistory and gameList classes. This documentation serves to ensure thorough testing coverage, identify any defects, and verify that the system meets its requirements.

**Test Strategy**

Our test strategy encompasses:

* Unit Tests: Testing individual components (methods) of the classes MainWindow, Startgame, gameHistory and gameList to ensure they function correctly in isolation.
* Integration Tests: Testing the interaction between different components to ensure they work together as intended.

**Test Coverage**

The following classes and functions are covered by the tests:

* MainWindow:
  + registerUser
  + checkValid
  + loadUsers
  + getUsers
* Startgame:
  + checkWin
  + checkTie
  + aiMove
  + minimax
  + saveGame
* gameList:
  + split
  + filterNames
* gameList:
  + isValidString
  + splitAndEnqueue

**Test Cases**

**MainWindow Tests**

1. **RegisterUser**
   * **Purpose:** To test the user registration functionality.
   * **Test Steps:**
     1. Register a new user.
     2. Attempt to register the same user again.
   * **Expected Results:**
     1. First registration should succeed (return 1).
     2. Second registration should fail (return 0).
2. **ValidateUserLogin**
   * **Purpose:** To test the user login validation functionality.
   * **Test Steps:**
     1. Register a user.
     2. Validate login with correct credentials.
     3. Validate login with incorrect password.
     4. Validate login again with correct credentials (should indicate already logged in).
   * **Expected Results:**
     1. Correct login returns 1.
     2. Incorrect password returns 0.
     3. Already logged in returns 2.
3. **LoadUsersFromFile**
   * **Purpose:** To test loading users from a file.
   * **Test Steps:**
     1. Create a temporary file with user data.
     2. Load users from the file.
   * **Expected Results:**
     1. The users vector should contain the expected number of users.
     2. The usernames and passwords should match the data in the file.

**Startgame Tests**

1. **CheckWin**
   * **Purpose:** To test the win-checking functionality.
   * **Test Steps:**
     1. Set a winning condition for Player X.
     2. Check for a win for Player X.
     3. Check for a win for Player O.
   * **Expected Results:**
     1. Player X should win (return true).
     2. Player O should not win (return false).
2. **CheckTie**
   * **Purpose:** To test the tie-checking functionality.
   * **Test Steps:**
     1. Set a tied game state.
     2. Check for a tie.
     3. Modify the state to not be a tie and check again.
   * **Expected Results:**
     1. The game should be recognized as a tie (return true).
     2. The modified state should not be a tie (return false).
3. **AIMove**
   * **Purpose:** To test the AI move functionality.
   * **Test Steps:**
     1. Set up a game state.
     2. Get AI move.
   * **Expected Results:**
     1. The AI should choose the correct move based on the game state.
4. **Minimax**
   * **Purpose:** To test the minimax algorithm functionality.
   * **Test Steps:**
     1. Set up a game state.
     2. Run the minimax algorithm.
   * **Expected Results:**
     1. The minimax algorithm should return the correct score.
5. **SaveGame**
   * **Purpose:** To test the game-saving functionality.
   * **Test Steps:**
     1. Simulate some game moves and set a winner.
     2. Save the game state to a file.
     3. Read the file and check its contents.
   * **Expected Results:**
     1. The file should contain the correct game data.

**gameList Tests**

1. **SplitFunctionTest**
   * **Purpose:** To test the string splitting functionality.
   * **Test Steps:**
     1. Split a string with multiple parts.
     2. Split a string with a single part.
     3. Split a string with a trailing delimiter.
   * **Expected Results:**
     1. The function should correctly split the string into parts.
2. **FilterNamesFunctionTest**
   * **Purpose:** To test filtering game history by player names.
   * **Test Steps:**
     1. Filter the history file for a specific player.
   * **Expected Results:**
     1. The function should return the correct filtered results.

**gameHistory Tests**

1. **IsValidStringTest**
   * **Purpose:** To test string validation functionality.
   * **Test Steps:**
     1. Validate various strings with valid and invalid characters.
   * **Expected Results:**
     1. The function should correctly identify valid and invalid strings.
2. **SplitAndEnqueueTest**
   * **Purpose:** To test the string splitting and queuing functionality.
   * **Test Steps:**
     1. Split and enqueue a string with multiple parts.
     2. Split and enqueue a string with a single part.
     3. Split and enqueue a string with a trailing delimiter.
   * **Expected Results:**
     1. The function should correctly split the string and enqueue the parts.

**Test Results**

All tests were executed using Google Test framework. The tests passed successfully, indicating that the functionalities of MainWindow, Startgame, gameHistory and gameList classes, as well as the utility functions, work as expected. The following is a summary of the test results:

| **Test Case** | **Result** |
| --- | --- |
| RegisterUser | Pass |
| ValidateUserLogin | Pass |
| LoadUsersFromFile | Pass |
| CheckWin | Pass |
| CheckTie | Pass |
| AIMove | Pass |
| Minimax | Pass |
| SaveGame | Pass |
| SplitFunctionTest | Pass |
| FilterNamesFunctionTest | Pass |
| IsValidStringTest | Pass |
| SplitAndEnqueueTest | Pass |

**Conclusion**

The comprehensive suite of unit and integration tests provided extensive coverage of the key functionalities within the MainWindow, Startgame, gameHistory and gameList classes, as well as related utility functions. All tests passed, demonstrating that the system meets its requirements and performs as expected under various conditions. This documentation serves as a reference for the testing strategy, test cases, and results, ensuring that any future modifications or extensions to the system can be similarly tested for reliability and correctness.