Testing Documentation

This document provides detailed information about the unit tests and integration tests for the `SessionManager`, `GameConfiguration`, and `TicTacToeWidget` classes. The tests ensure that the individual components and their interactions work correctly.

Unit Tests

1. SessionManager

Test Cases:

1. **testLogin**:
    - **Description:** Verifies that a user can log in successfully.
    - **Setup:** Initialize `SessionManager` instance.
    - **Execution:** Call `login` method with a test username.
    - **Verification:** Check if the current user is set to the test username.
2. **testLogout**:
    - **Description:** Verifies that a user can log out successfully.
    - **Setup:** Initialize `SessionManager` instance and log in a user.
    - **Execution:** Call `logout` method.
    - **Verification:** Check if the current user is cleared.
3. **testGetCurrentUser**:
    - **Description:** Verifies that the current user is returned correctly.
    - **Setup:** Initialize `SessionManager` instance and log in a user.
    - **Execution:** Call `getCurrentUser` method.
    - **Verification:** Check if the returned user is the same as the logged-in user.

```
PASS   : TestSessionManager::initTestCase()
PASS   : TestSessionManager::testLogin()
PASS   : TestSessionManager::testLogout()
PASS   : TestSessionManager::testGetCurrentUser()
PASS   : TestSessionManager::cleanupTestCase()
```

2. GameConfiguration

Test Cases:

1. **testSingleton**:
    - **Description:** Verifies that `GameConfiguration` follows the singleton pattern.
    - **Setup:** Get two instances of `GameConfiguration`.
    - **Execution:** Compare the two instances.
    - **Verification:** Ensure both instances are the same.
2. **testSetPlayer1Name**:
    - **Description:** Verifies that the player 1 name can be set correctly.
    - **Setup:** Get `GameConfiguration` instance.
    - **Execution:** Call `setplayer1name` method with a test name.
    - **Verification:** Check if `getplayer1name` returns the correct name.
3. **testSetPlayer2Name**:
    - **Description:** Verifies that the player 2 name can be set correctly.
    - **Setup:** Get `GameConfiguration` instance.
    - **Execution:** Call `setplayer2name` method with a test name.
    - **Verification:** Check if `getplayer2name` returns the correct name.
4. **testGetPlayer1Name**:
    - **Description:** Verifies that the player 1 name can be retrieved correctly.
    - **Setup:** Get `GameConfiguration` instance and set player 1 name.
    - **Execution:** Call `getplayer1name` method.
    - **Verification:** Check if the correct name is returned.
5. **testGetPlayer2Name**:
    - **Description:** Verifies that the player 2 name can be retrieved correctly.
    - **Setup:** Get `GameConfiguration` instance and set player 2 name.
    - **Execution:** Call `getplayer2name` method.
    - **Verification:** Check if the correct name is returned.

```
PASS   : TestGameConfiguration::initTestCase()
PASS   : TestGameConfiguration::testSingleton()
PASS   : TestGameConfiguration::testSetPlayer1Name()
PASS   : TestGameConfiguration::testSetPlayer2Name()
PASS   : TestGameConfiguration::testGetPlayer1Name()
PASS   : TestGameConfiguration::testGetPlayer2Name()
PASS   : TestGameConfiguration::cleanupTestCase()
```

3. TicTacToeWidget

Test Cases:

1. **testDetermineWinner**:
    - **Description:** Verifies that the winner determination logic works correctly.
    - **Setup:** Initialize `TicTacToeWidget` and create the board.
    - **Execution:** Simulate a sequence of moves that results in a win.
    - **Verification:** Check if the correct winner is determined.
2. **testHandleClicksOnBoard**:
    - **Description:** Verifies that clicks on the board are handled correctly.
    - **Setup:** Initialize `TicTacToeWidget` and create the board.
    - **Execution:** Simulate clicks on the board.
    - **Verification:** Check if the correct symbols are placed on the board.
3. **testResetBoard**:
    - **Description:** Verifies that the board can be reset correctly.
    - **Setup:** Initialize `TicTacToeWidget`, create the board, and simulate some moves.
    - **Execution:** Call `resetBoard` method.
    - **Verification:** Check if the board is cleared correctly.

```
PASS   : TestTicTacToeWidget::initTestCase()
PASS   : TestTicTacToeWidget::testInitialState()
PASS   : TestTicTacToeWidget::testDetermineWinner()
PASS   : TestTicTacToeWidget::testHandleClicksOnBoard()
PASS   : TestTicTacToeWidget::testResetBoard()
PASS   : TestTicTacToeWidget::testAiMoveSelection()
PASS   : TestTicTacToeWidget::cleanupTestCase()
```

Integration Tests

IntegrationTest

Test Cases:

1. **initTestCase**:
    - **Description:** Sets up the test environment before any test functions are executed.
    - **Setup:** Initialize `GameConfiguration`, `SessionManager`, and `TicTacToeWidget`.
2. **cleanupTestCase**:
    - **Description:** Cleans up the test environment after all test functions have been executed.
    - **Execution:** Delete the instances of `GameConfiguration`, `SessionManager`, and `TicTacToeWidget`.
3. **testFullGameFlow**:

- **Description:** Verifies the full game flow from configuration to playing the game.
- **Setup:** Initialize and configure the game using `GameConfiguration` and `SessionManager`.
- **Execution:**
  - Log in a user using `SessionManager`.
  - Set player names and game mode using `GameConfiguration`.
  - Simulate a complete game using `TicTacToeWidget`.
  - Determine the winner.
- **Verification:** Check if the game flow works correctly and the winner is determined as expected.

```
PASS   : IntegrationTest::initTestCase()
QDEBUG : IntegrationTest::testFullGameFlow() getwinner() called, winner is: Player 1
PASS   : IntegrationTest::testFullGameFlow()
PASS   : IntegrationTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 3ms
```

Conclusion

This testing documentation outlines the unit tests and integration tests for the `SessionManager`, `GameConfiguration`, and `TicTacToeWidget` classes. These tests ensure the correctness of individual components as well as their interactions in a full game flow.