Fundamentals of Software Engineering

**[CSE 3310-004]**

**Group 12**

**We developed 36 different testing procedures for Project Testing. Of these 36 testing procedures, 12 were manual testing procedures, 14 were unit tests, 2 tested specific scenarios, and the rest (8) were junit tests created exclusively using JSON strings to automate the verification of specific requirements. Through testing, we were able to identify 37 critical defects across all our implemented classes.**

**Manual Tests**

|  |  |
| --- | --- |
| **Test Name** | **Description** |
| **Manual Test- two games** | **Tests the ability to run two games simultaneously without affecting the functionality of each other.** |
| **Manual test- Grid Countdown** | **Tests the presence of a game countdown and tests two players finding the word at nearly the same time.** |
| **Manual Test- 2 game chat** | **Tests chat for games running two games concurrently ensuring that chat is game-specific.** |
| **Manual Test - Server close** | **Tests the behavior of the webpage when the server is closed (the program is stopped while running).** |
| **Manual Test- words and grid** | **Ensures:**   * **That a grid is 20x20.** * **A word bank with all possible words is visible.** * **A valid word when selected shows up in the selectors color on the grid for all the players in the game.** * **The word is highlighted in green in the word bank to mark it as “found”.** * **Points are awarded to the player that finds the word.** |
| **Manual Test - player disconnect** | **Tests a scenario where a player disconnects mid game. Ensures that a player leaving mid game has no effect on the game.** |
| **Manual Test - leaderboard** | Ensures:   * That a grid is 20x20 * A word bank with all possible words is visible. * A valid word when selected shows up in the selectors color on the grid for all the players in the game. * The word is highlighted in green in the word bank to mark it as “found”. * Points are granted to the player that finds the word. * The game screen disappears upon the completion of the game. * **A leaderboard is displayed after the game screen disappears.** |
| **Manual Test - Grid, Leaving mid-game, 2 game chat.xlsx** | Tests:   * Using Login * Names of all players in the lobby being displayed. * A word when selected is marked on the grid with the players respective color * Ensure grid size of 20x20 * Ensures that a player leaving mid game has no effect on the current game. * Ensures that the player who has left can join a different game. * Ensures that chat between two games is not shared. |
| **Manual Test - Login Manager** | Ensures:   * People can join games. * Usernames cannot be <3 characters long. * No duplicate usernames. |
| **Manual Test - 3 Player Game + Chat** | Tests:   * Player registration. * Grid selection * Score Validation * Chat * Game end * Countdown * Leaderboard |

**JUNIT JSON String Tests**

|  |  |
| --- | --- |
| **Test Name** | **Description** |
| **ChatTest.java** | Verifies that messages can be sent and received correctly, and that the chat log is updated accordingly. |
| **GridTest.java** | Performs operations such as registering users, starting the game, clicking on grid cells, and verifying the correctness of the grid state after each operation. It also tests the behavior of the game when invalid selections are made. Additionally, it tests the handling of different types of words (horizontal, vertical upwards, vertical downwards, diagonal upwards, diagonal downwards) in the grid. |
| **LeaderboardTest.java** | Creates a new game, registers players, starts the game, performs clicks on the game grid, and then checks the leaderboard to ensure that the scores are calculated correctly. The test function uses JSON strings to provide inputs and compare the expected outputs. |
| **LoginManagerTest.java** | Creates a Game object, registers multiple users with different usernames, and then asserts that the registered users are stored correctly. It also includes additional assertions to check for duplicate usernames and ensures that usernames are not less than 3 characters in length. |
| **Scenario1Test.java** | Tests a scenario where players join a game, chat with each other, and then start the game. It verifies various conditions such as registering players, sending and receiving chat messages, and starting the game. |
| **Scenario2Test.java** | Creates a Game object, performs a series of actions on the game, and asserts the expected results. The test covers various scenarios such as registering users, starting the game, clicking on grid cells, checking word points, sending and receiving chat messages, and generating a leaderboard. |
| **WholeGameTest.java** | It performs a series of operations and assertions to verify the correctness of the game logic. It tests registration, starting the game, clicking on grid cells, checking word colors, updating the game state, handling chat messages, and generating the leaderboard. It tests the overall functionality of the game. |

**Unit tests are not listed here because they were completed as a part of implementation. They verify basic functionality of each of the classes.**