Checkers

Software Test Case Document

GROUP 4

|  |  |
| --- | --- |
| **Group Members** | Omair Bhore, Phuc Ngo, Nhan Nguyen, Bruce Zhao |
| **Faculty Advisor** | Dr. Filippos Vokolos, Ph. D. |
| **Teaching Assistant** | Mr. Sean Grimes |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Revision Information** | **Revision** |
| Omair Bhore  Nhan Nguyen  Phuc Ngo  Bruce Zhao | 08/15/2017 | First Draft - Created and planned out sections and subsections | 1.0 |
| Omair Bhore  Nhan Nguyen  Phuc Ngo  Bruce Zhao | 08/16/2017 | Second Draft - Filled all sections and subsections | 1.1 |
| Omair Bhore  Nhan Nguyen  Phuc Ngo  Bruce Zhao | 08/17/2017 | Final Draft - Revision all sections and subsections | 1.2 |
| Omair Bhore  Bruce Zhao | 08/28/2017 | Final Draft - Actual Test Results Updated | 2.0 |

Table of Contents

1. **Document Introduction**
2. Purpose
3. Definitions, Acronyms, Abbreviations
4. References
5. **Testing Environments**
6. Environment 1: Google Chrome macOS Sierra Version 10.12.6
7. Environment 2: Google Chrome Windows 10 Version 1703 Creators Update
8. **Setup Information and Prerequisites**
9. **Test Cases**
   1. Test Case 1: Initialize Game
      1. Description
      2. Pre-Conditions
      3. Scenario
   2. Test Case 2: Play game (legal and illegal moves)
10. Description
11. Pre-Conditions
12. Scenario
    1. Test Case 3: Ending game
13. Description
14. Pre-Conditions
15. Scenario
16. **Appendix**

5.1 Glossary

**1. Document Introduction**

**1.1 Purpose**

This document is to describe the testing approaches used to test the functionality and the performance of the Checkers game as described in the requirements document. Checkers game is a NodeJS application designed to be played between two players on different systems and allows for multiple game sessions to occur at the same time.

**1.2 Definitions, Acronyms, Abbreviations**

Please refer to the Glossary and the Definitions, Acronyms, Abbreviations sections in the Software Design and Requirements Specification documents respectively.

**1.3 References**

This document may contain terms and references which can be found throughout the Software Design and the Requirements Specification documents.

**2. Testing Environments**

The program and all of it’s test cases have been run on following test environments.

**2.1 Environment 1: Google Chrome Mac OS X**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Machine name | Macbook Air - Mid 2012 | DB Directory | N/A | | | |
| OS and Version | macOS Sierra Version 10.12.6 | Platform | Google Chrome  Version  60.0.3112.90 | | Client Server / Back-end | Localhost (NodeJS) |
| Tester Name | Nhan Nguyen, Phuc Ngo | | Test Date | 08-15-2017 | | |
| New Log | List the passing results and errors, if any, after tests have been run | | | | State | PASS |

**2.2 Environment 2: Google Chrome Windows 10**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Machine name | Razer Blade Late 2016 | DB Directory | N/A | | | |
| OS and Version | WIndows 10  Version 1703  Creators Update | Platform | Google Chrome  Version  60.0.3112.90 | | Client Server / Back-end | Localhost (NodeJS) |
| Tester Name | Omair Bhore, Bruce Zhao | | Test Date | 08-16-2017 | | |
| New Log | List the passing results and errors, if any, after tests have been run | | | | State | PASS |

**3. Setup Information and Prerequisites**

Before running the application, the follow prerequisites must be met.

* The front-end test environment and associated test cases using Mocha and Karma can be run through a terminal.
* The back-end test environment using Mocha and associated test cases can be run through a terminal
* The back-end can be launched as long as there is a working NodeJS environment is installed on the system.
* A Google Chrome browser that supports HTML5.

**4. Test Cases**

**4.1 Test case 1: Initialize Game**

**4.1.1 Description:**

The case consists of the steps required to host the application on a computer

**4.1.2 Pre-conditions for the test case:**

A NodeJS environment installed on the system, and all the modules that are mentioned in the package.js file.

**4.1.3 Scenario:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | | | | | |
| **ID** | **Req** | **Description** | **Execution Steps** | **Expected Result** | **Actual Result** |
| A1 | 3.1.1 | Launch App | 1. Open terminal and make sure the project folder is the current working directory  2. Type the following commands in order:  npm install  npm run build  npm start | Server is listening on port 3000 | Server is listening on port 3000 |
| A2 | 3.2.1 | Launch Website | 1. Open Google Chrome Version 60.0.3112.90  or higher.  2. Type in the browser <http://localhost:3000/> | The Lobby Page with Game title, input box, and start button display | The Lobby Page with Game title, input box, and start button display |
| A3 | 3.1.4 | Set username - Success | 1. Type into the input box to set username | Server checks if the username has already been taken and report success back to the user | Server checks if the username has already been taken and report success back to the user |
| A4 | 3.1.4 | Set username - Duplicate username | 1. Type into the input box to set a duplicate username | Server checks if the username has already been taken and report error back to the user | Server checks if the username has already been taken and report error back to the user |
| A4 | 3.1.3 | Start Matchmaking  (Front End) | 1. Type into the input box to set username  2. Click start to start the match making process | Waiting icon is displayed. Player waits while the server processes | Waiting icon is displayed. Player waits while the server processes |
| A5 | 3.1.3 | Start Matchmaking - No client to match | 1. Type into the input box to set username  2. Click start to start the match making process | Waiting icon is displayed. Player waits while the server processes. Server responds back with no other players available to match. Application informs users | Waiting icon is displayed. Player waits while the server processes. Server responds back with no other players available to match. Application informs users |
| A5 | 3.1.3 | Start Matchmaking - Success | 1. Type into the input box to set username  2. Click start to start the match making process | Server matches two players together and responds back with a socket connection ID. Application informs users a match | Server matches two players together and responds back with a socket connection ID. Application informs users a match |
| A6 | 3.1.3 | Redirect to the game page | 1. Type into the input box to set username  2. Click start to start the match making process  3. Client waits until get redirected | Client redirects to a game session and establish a socket connection with the socket ID from the server | Client redirects to a game session and establish a socket connection with the socket ID from the server |
| A7 | 3.1.2 | Join game - Success | 1. Client waits until socket initializes and the page fully render | Socket initializes and game page fully render | Socket initializes and game page fully render |
| A8 | 3.1.2 | Join game - Socket fails to connect | 1. Client waits until socket initializes and the page fully render | Socket fails to connect | Socket fails to connect |

**4.2 Test case 2: Play the game**

**4.2.1 Description:**

The case consists of the steps required to play a game

**4.2.2 Pre-conditions for the test case:**

Website launched and a successful game session has been launched after matchmaking. Make sure that all test cases from Test case 1 have passed.

**4.2.3 Scenario:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | | | | | |
| **ID** | **Req** | **Description** | **Execution Steps** | **Expected Result** | **Actual Result** |
| B1 |  | Play a turn | 1. Click and drag on a red or black game piece. This depends on which color is assigned to you.  2. When the appropriate color piece is being dragged, all valid squares are highlighted in transparent yellow.  3. Drag the clicked piece to a highlighted square and drop the piece. | Selected piece has moved to the desired highlighted square | Selected piece has moved to the desired highlighted square |
| B2 |  | Move a piece - Success | 1. Follow Test Scenario B1 to click a piece and drop it on a highlighted square.  2. Note that a regular piece can only move forward one diagonal square | Piece has moved to the desired highlighted square. | Piece has moved to the desired highlighted square. |
| B3 |  | Move a piece - Failure | 1. Follow Test Scenario B1 to click a piece and try dropping it on a non-highlighted square. | Piece does not move and still stays in current position.  User still has to make a valid move to finish his/her turn. | Piece does not move and still stays in current position.  User still has to make a valid move to finish his/her turn |
| B4 |  | Eliminate a piece - Success | 1. Follow Test Scenario B2 to check valid moves by observing the highlighted squares.  2. When there is an opportunity to eliminate an opponent piece(s), one or more squares will be highlighted red instead of yellow.  3. Following B2, the selected piece can be dropped on a red square, effectively resulting in an opponent piece being eliminated.  4. If there is an opportunity to kill more than one piece, the player must take the additional jumps to complete a double, triple or more jumps.  5. After making the first jump, the user will have to drag and drop to the next highlighted red square in order to complete his/her turn. | Squares are highlighted red.  Single jump:  After user drops piece on a red highlighted square, the opponent's piece is eliminated and the turn ends.  Multiple jumps:  After user drops the piece on the first red highlighted square, the opponent’s piece gets eliminated and the turn doesn’t end. The user then has to repeat the above step for the next red highlighted square till all elimination jumps have been completed. After all jumps have been made, the turn ends. | Squares are highlighted red.  Single jump:  After user drops piece on a red highlighted square, the opponent's piece is eliminated and the turn ends.  Multiple jumps:  After user drops the piece on the first red highlighted square, the opponent’s piece gets eliminated and the turn doesn’t end. The user then has to repeat the above step for the next red highlighted square till all elimination jumps have been completed. After all jumps have been made, the turn ends. |
| B5 |  | Become a King | 1. Following Test Case B2 repeatedly could result in a player’s piece reaching the opponent’s end of the board safely. | After reaching the other end of the board safely. The piece should look different as the King piece has a crown in the middle of the circle.  A King piece can move both forwards and backwards, so there should be more highlighted squares when clicked on. | After reaching the other end of the board safely. The piece should look different as the King piece has a crown in the middle of the circle.  A King piece can move both forwards and backwards, so there should be more highlighted squares when clicked on |
| B6 |  | Activity Log | 1. As players play their turns, the activity log will keep track of which player’s turn it is and what happens at the end of that turn  2. Follow Test Case B2 to make a non-elimination move, then observe the activity log.  3. Follow Test Case B4 to make a elimination move, then observe the activity log.  5. The activity log shall display the end of game when the game has ended. | At the start of the game, the activity log should display the start of game header and mention which player’s turn it is.  When a non-elimination turn has been played, the log should display the piece’s initial location and the location after the turn has ended.  An elimination turn shall result in the activity log displaying the piece’s initial location and the location after the turn has ended. The activity log shall also display the information of all the pieces which were eliminated at the end of that turn. The activity log shall also display the number of pieces left. After each elimination.  At the end of the game, the activity log should display the end of game footer and mention which player won. | At the start of the game, the activity log should display the start of game header and mention which player’s turn it is.  When a non-elimination turn has been played, the log should display the piece’s initial location and the location after the turn has ended.  An elimination turn shall result in the activity log displaying the piece’s initial location and the location after the turn has ended. The activity log shall also display the information of all the pieces which were eliminated at the end of that turn. The activity log shall also display the number of pieces left. After each elimination.  At the end of the game, the activity log should display the end of game footer and mention which player won. |

**4.3 Test case 3: End the game**

**4.3.1 Description:**

The case consists of the steps required for resigning, timing out, stalemates, and winning a game from an active game session.

**4.3.2 Pre-conditions for the test case:**

Game session was successfully created for 2 players and is active. One of the player loses all of the checker pieces. Make sure that all test cases from Test case 1 and 2 have passed.

**4.3.3 Scenario:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | | | | | |
| **ID** | **Req** | **Description** | **Execution Steps** | **Expected Result** | **Actual Result** |
| C1 |  | Give up | 1. Host a session with two players.  2. One player clicks the ‘Give up’ button. | The player who gave up will get a ‘You lose’ message in a pop-up modal and the player clicks the 'Back to lobby' button on the modal to go back to the lobby page.  The player who stayed will get a ‘You Win’ message in a pop-up modal and the player clicks the 'Back To Lobby' button on the modal to go back to the lobby page. | The player who gave up will get a ‘You lose’ message in a pop-up modal and the player clicks the 'Back to lobby' button on the modal to go back to the lobby page.  The player who stayed will get a ‘You Win’ message in a pop-up modal and the player clicks the 'Back To Lobby' button on the modal to go back to the lobby page. |
| C2 |  | Winning | 1 .Host a session between two players.  2.The game is played and one player successfully captures all the opponent’s pieces. | The player who lost all the pieces will get a ‘You lose’ message in a pop-up modal and the player clicks the 'Back to lobby' button on the modal to go back to the lobby page.  The player who captured all the pieces will get a ‘You Win’ message in a pop-up modal and the player clicks the 'Back To Lobby' button on the modal to go back to the lobby page. | The player who lost all the pieces will get a ‘You lose’ message in a pop-up modal and the player clicks the 'Back to lobby' button on the modal to go back to the lobby page.  The player who captured all the pieces will get a ‘You Win’ message in a pop-up modal and the player clicks the 'Back To Lobby' button on the modal to go back to the lobby page. |
| C3 |  | Timing out | 1. Host a session between two players.  2. If one of the players is unresponsive for 2 minutes and the match is terminated. | The player that is still active will get a ‘You win’ message in a pop-up modal and the player clicks the 'Back To Lobby' button on the modal to go back to the lobby page. The player who timed out will be given a ‘You have timed out’ message and is re-directed to the lobby. | The player that is still active will get a ‘You win’ message in a pop-up modal and the player clicks the 'Back To Lobby' button on the modal to go back to the lobby page. The player who timed out will be given a ‘You have timed out’ message and is re-directed to the lobby. |
| C4 |  | Stalemate | 1. Host a session between two players.  2. Both players are in a position where they can make no winning moves due to being able to move back and forth with a king. | Both players will get a ‘Stalemate’ message in a pop-up modal and the players click the 'Back To Lobby' button on the modal to go back to the lobby page. | Both players will get a ‘Stalemate’ message in a pop-up modal and the players click the 'Back To Lobby' button on the modal to go back to the lobby page. |
| C5 |  | Go back to Lobby Page | 1. Host a session between two players.  2. Both players play until the match is over.  3. Click the ‘Back To Lobby’ in the popup modal at the end of the game  4. Get redirected to Lobby Page | The client gets redirected to the Lobby Page | The client gets redirected to the Lobby Page |

**5. Appendix**

**5.1 Glossary**

**Browser -** A program including but not limited to Chrome, Internet Explorer, Firefox, Opera, and safari, that can be used to access websites.

**JS -** JavaScript, a web programming language. Abbreviation.

**Matching queue -** A list of every user waiting to get connected to a game Session. The queue will follow a “first come first serve” model.

**Modal -** A window that pops up on top of the current window, effectively taking control and attention of the user. User must make a selection on the modal window before gaining access to the initial window.

**Page -** Shorthand web page.

**Platform -** Any such things that can be classified as a mobile or desktop operating system including but not limited to Android, Apple, Windows, Ubuntu, and OSX.

**Player -** A user who has been connected to a game session.

**Socket -** A socket is an end point of a two-way connection between two programs running on a network.

**UI -** User Interface, abbreviation. This is the layer of a computer program that lets the user and the computer interact.