SE 3XA3: Test Report
Ultimate Tic Tac Toe

Team 3,
Kunal Shah - shahk24
Pareek Ravi - ravip2

December 7, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
December 5	0.0	Initial commit
December 6	0.1	list of all tests
December 6	0.2	Added figs to comparison section
December 7	0.3	Fixed test counter
December 7	0.4	Test trace to FR and MG
December 7	0.5	Finished comparison section
December 7	0.6	Added unit testing and changes to testing
December 7	2.0	Rev 1 Sumbission

Abstract

This document describes the results of all the testing done for Ultimate Tic Tac Toe.

1 Functional Requirements Evaluation

List of Functional Tests

This section provides an overview of the functional tests completed. Tests are summarized in Table 2.

Table 2: **Table of Tests**

ID	Test
in-test-id 1	Click should set random tile
in-test-id 2	Win inner board
in-test-id 3	Draw inner board (tie)
log-test-id 1	Win the full game
log-test-id 2	Draw the full game
log-test-id 3	Play Again
log-test-id 4	Return to Board
logistic-test-id 1	Click on a cell that is already occupied
logistic-test-id 2	Users are alternating turns

1.1 User Input

- in-test-id 1
  Testing if user's input being received

  Initial State: It is the start of a new game and the board is empty

  Input: A click on any random cell of any inner tic tac toe board

  Expected Output: That cell should have the user's character on it

  Actual Output: The cell contains that user's character

  Result: The test passed

- in-test-id 2
  Testing if user click on invalid inner board

  Initial State: Opponent has played on a inner board and it is the user's turn

  Input: Click on a tile that they are not designated to click based on the rules of the game

  Expected Output: The board should not change at all. The game data should not change and it is still the user's turn

  Actual Output: Nothing happens

  Result: The test passed

- in-test-id 3
  Testing if user's input on cell already clicked

  Initial State: Opponent has played on a inner board and it is the user's turn

  Input: Click on a tile that has already been clicked previously

  Expected Output: The board should not change at all. The game data should not change and it is still the user's turn

  Actual Output: The cell doesn't change

  Result: The test passed

1.2 Game Logic

- log-test-id 1
  Test if inner board gets completed

  Initial State: Inner board is almost completed by user. It is their turn to play in the inner board where they will complete it.

  Input: Clicks on the cell that will complete the inner board

  Expected Output: The inner board will be marked with the character representing the user

  Actual Output: The inner board is marked with the character representing the player who completed the board

  Result: The test passed

- log-test-id 2
  Test if inner board ends in draw

  Initial State: The board is in a state where an inner board only has 1 cell available to click and it will result in that board being a draw

  Input: Click on the only available cell

  Expected Output: The inner board will be marked with the character '-' meaning it is a draw

  Actual Output: The inner board is marked with '-'

  Result: The test passed

- log-test-id 3
  Test if next move can be made on any incomplete inner board

  Initial State: Game board is partially filled with one inner board completed

  Input: Click at a cell corresponding to a completed inner board

  Expected Output: All incomplete inner boards active

  Actual Output: All the incomplete inner boards changed to an active board color

  Result: The test passed

- log-test-id 4
  Test if user turns are alternating

  Initial State: Player with character O or X just made a move

  Input: Click at any available cell

  Expected Output: The character on the other cell is the opposite

  Actual Output: The character was the opposite

  Result: The test passed

1.3 Game Logistics

- logistic-test-id 1
  Test if game launches

  Initial State: User is in the file explorer

Input: User launches the html file in browser

Expected Output: The game launches in all browsers and shows the welcome dropdown

Actual Output: The game launches to the welcome dropdown in the user's preferred browser

Result: The test passed.

Note: though the game does not work in Firefox, it has no issues in launching

- logistic-test-id 2
  Test if user input shows in window

  Initial State: User has just opened game

  Input: User clicks on any cell

  Expected Output: Cell they clicked on should change appearance

  Actual Output: The cell on the game board shows the user's character

  Result: The test passed

2 Nonfunctional Requirements Evaluation

Nonfunctional Requirements were tested using two methods, manual testing during the development of the game, and usability survey (sur, 2016) after revision 0 demonstration. After receiving feedback from beta testers' usability survey, changes where made to the application accordingly. Refer to section 5 for details.

List of Survey Questions

This section provides an overview of the survey questions asked in the usability survey. Questions are summarized in Table 3.

Table 3: **Table of Questions**

ID	Question
survey-ques-id 1	The rules were very easy to find
survey-ques-id 2	The rules are easy to understand
survey-ques-id 3	The color pallet is visually appealing
survey-ques-id 10	The response time was satisfying.
survey-ques-id 4	It is clear which player's turn it is. (X or O)?
survey-ques-id 11	The game runs smoothly on your browser
survey-ques-id 5	Were you offended by anything in the game?
survey-ques-id 6	Did you encounter epileptic symptoms while playing the game?
survey-ques-id 7	Did anything unexpected appear on the screen while playing?
survey-ques-id 8	Did you have fun playing this game?
survey-ques-id 9	Any Additional Comments?

2.1 Usability

The following figures show that all of out non functional fit criteria(refer to SRS) where met though the usability survey beta test.

- survey-ques-id 1

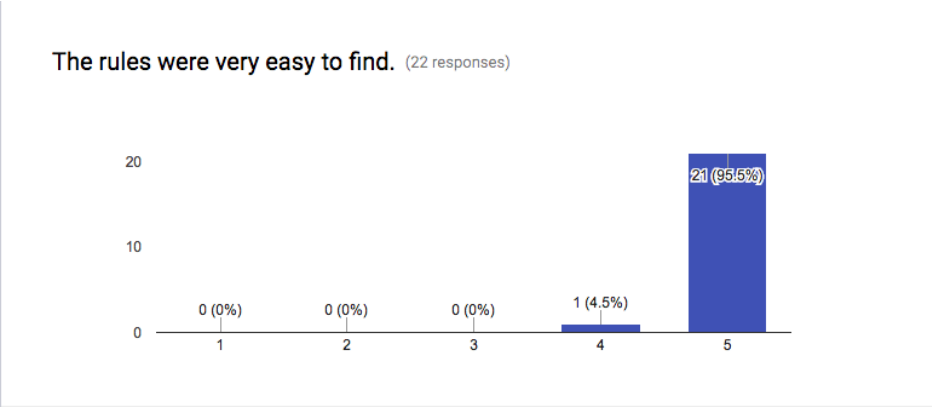


Figure 1: The rules were very easy to find

- survey-ques-id 2

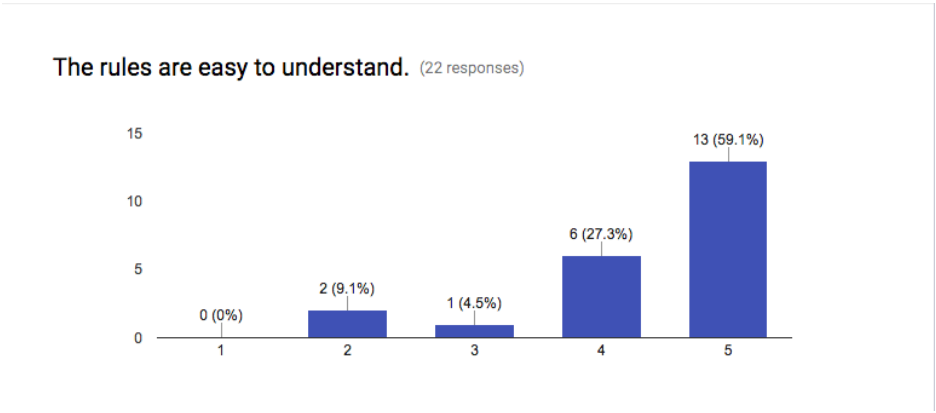


Figure 2: The rules are easy to understand

- survey-ques-id 3

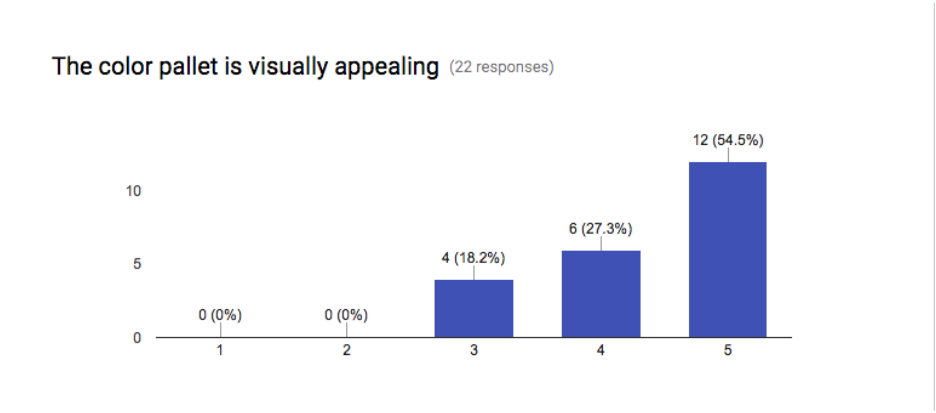


Figure 3: The color pallet is visually appealing

- survey-ques-id 4

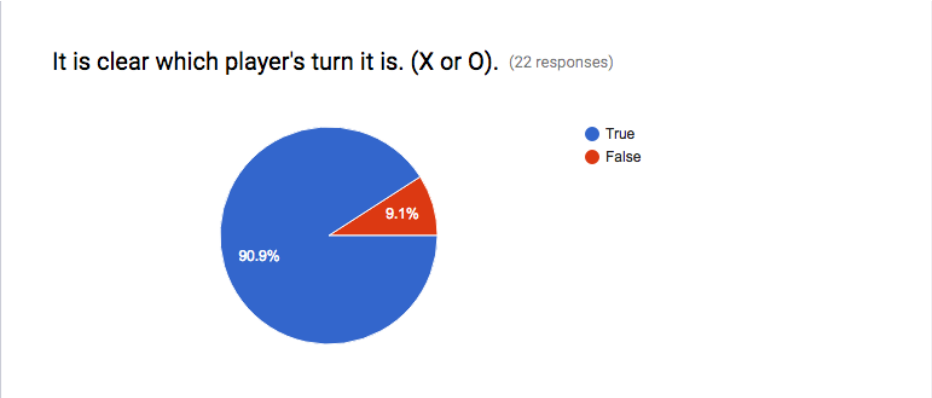


Figure 4: It is clear which player?s turn it is. (X or O)?

- survey-ques-id 5

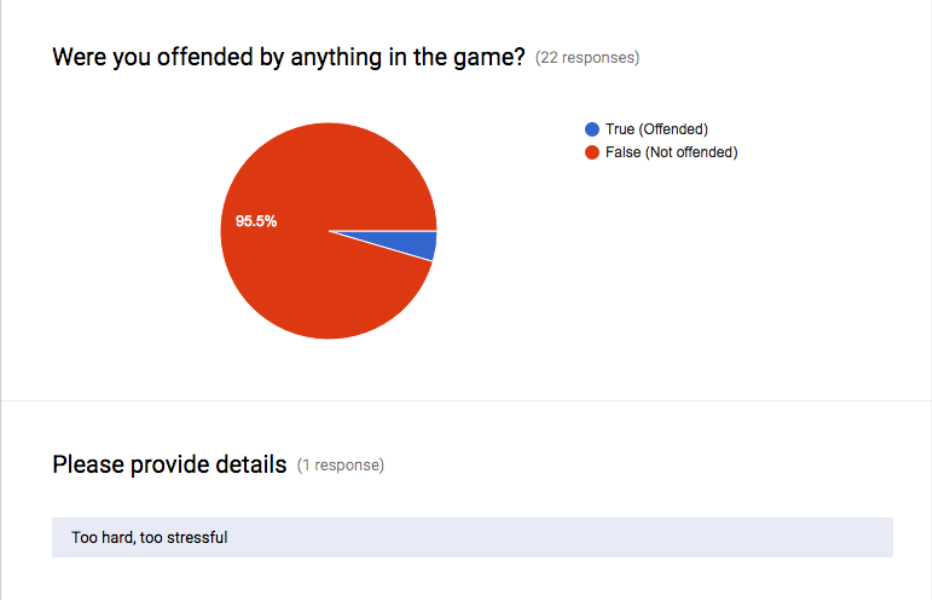


Figure 5: Were you offended by anything in the game?

- survey-ques-id 6

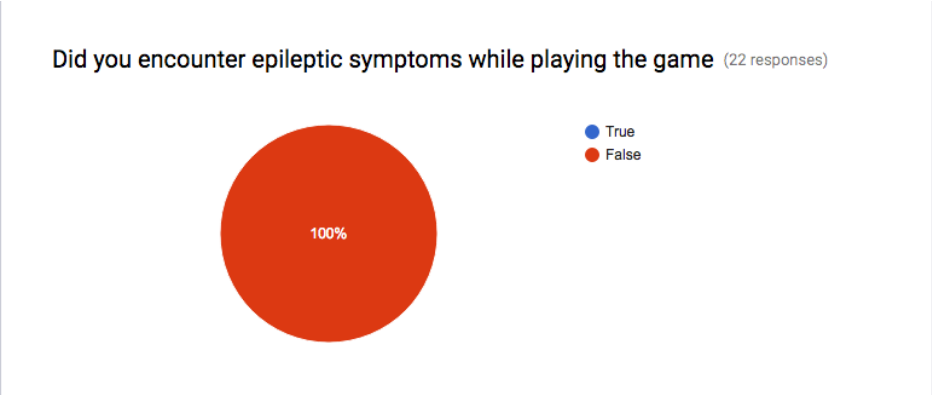


Figure 6: Did you encounter epileptic symptoms while playing the game?

- survey-ques-id 7

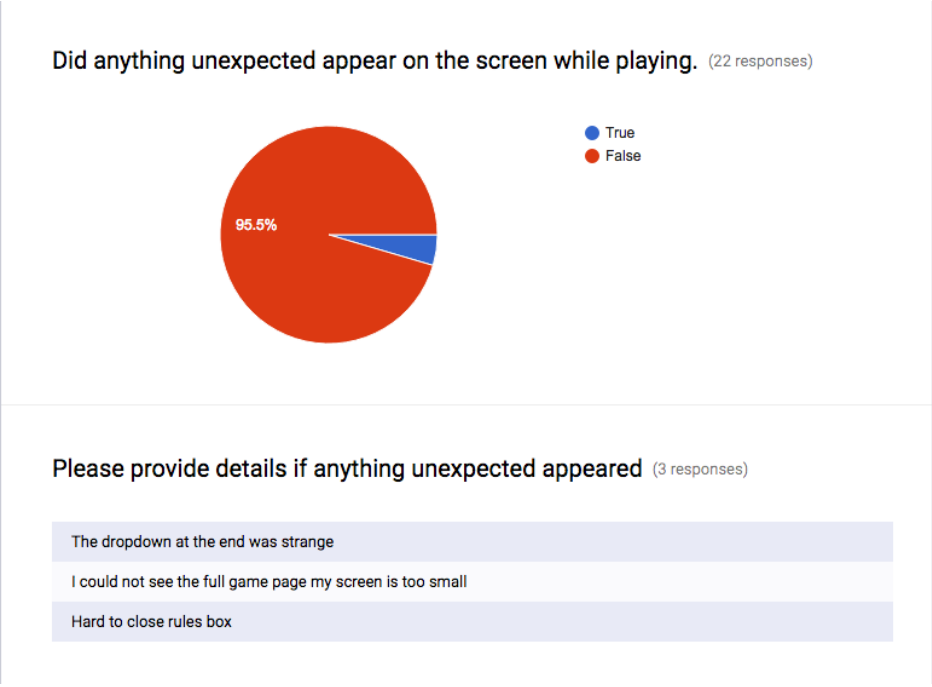


Figure 7: Did anything unexpected appear on the screen while playing?

- survey-ques-id 8

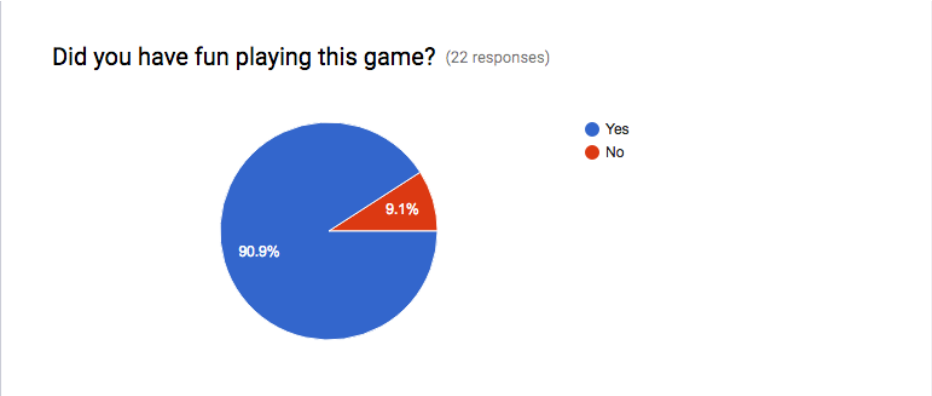


Figure 8: Did you have fun playing this game?

- survey-ques-id 9

Figure 9: Any Additional Comments?

2.2 Performance

- survey-ques-id 10

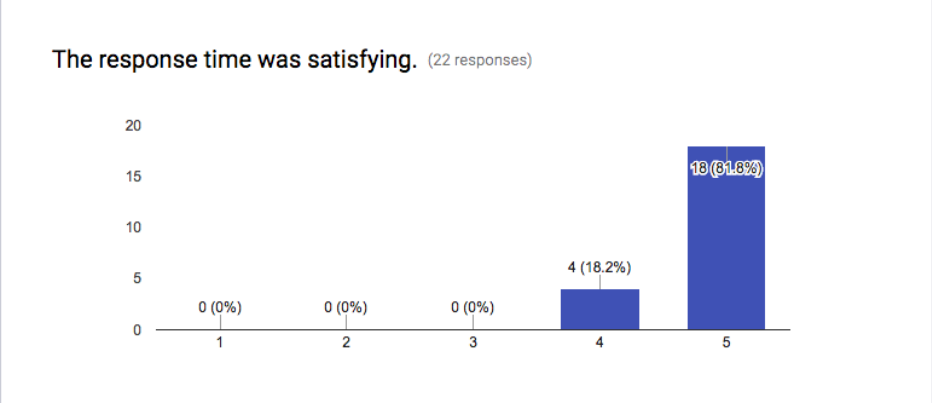


Figure 10: The response time was satisfying.

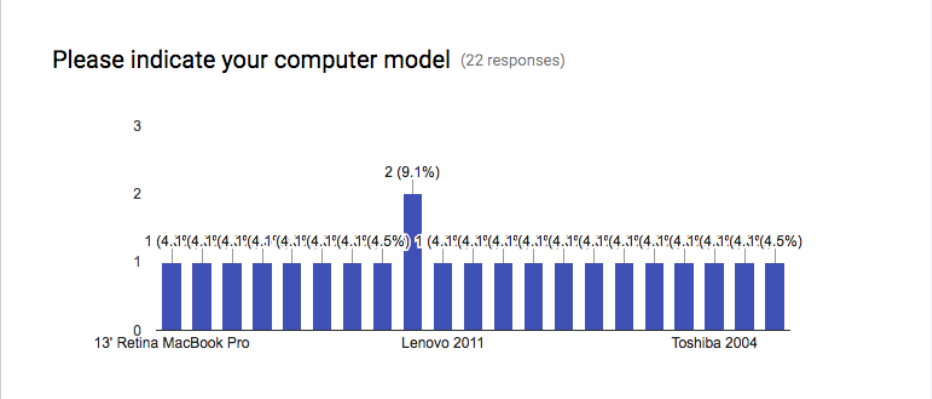


Figure 11: Model of computer

- survey-ques-id 11

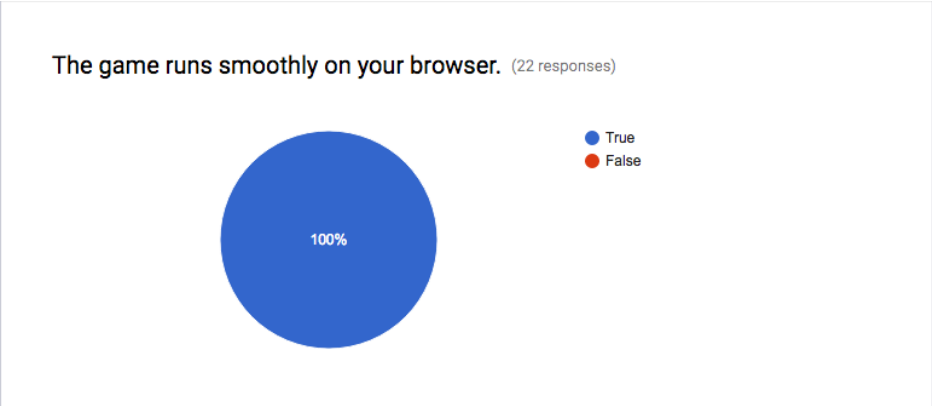


Figure 12: The game runs smoothly on your browser

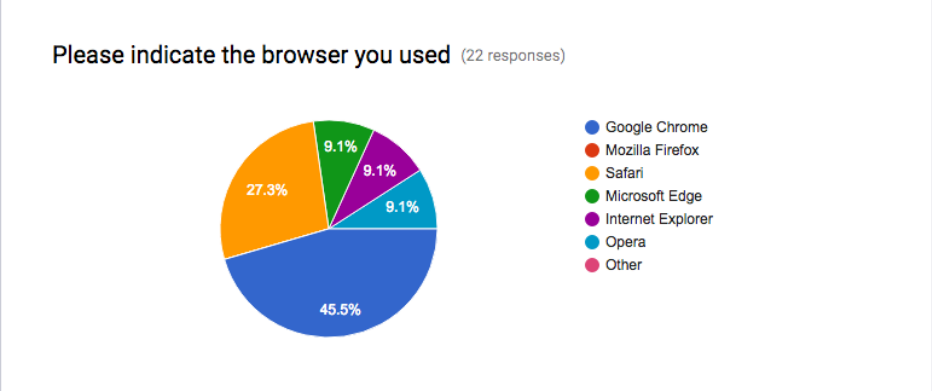


Figure 13: Browser Used

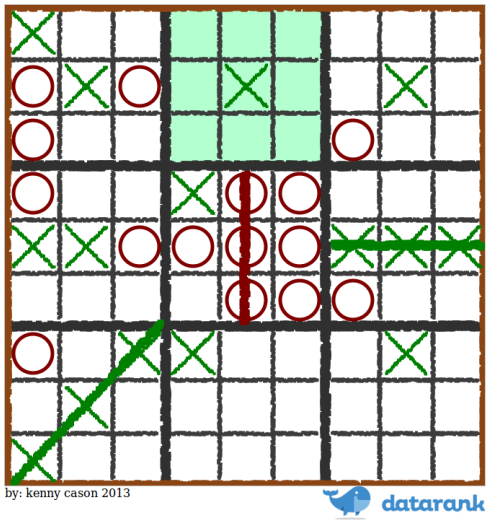
3 Comparison to Existing Implementation

This project is a reimplementation of the original Ultimate tic tac toe game (kennycason, 2010). Prior to starting this project we investigated the existing implementation. There were 3 main elements we wanted to change for our reimplementation of Ultimate Tic Tac Toe which can been seen in Figure 14.

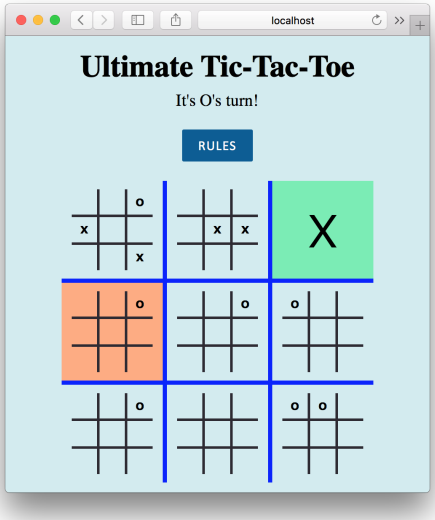
1. Rules explaining how to play the game
2. User interface (colours and ascetic)
3. Ability to repay the game

There where 7 tests performed to ensure that core functionality was maintained after the reimplementation of the original source. Please refer to the following tests:

1. in-test-id 1
2. in-test-id 3
3. in-test-id 3
4. log-test-id 1
5. log-test-id 2
6. logistic-test-id 1
7. logistic-test-id 2



(a) Original Game



(b) Reimplemented Game

Figure 14: Implementation Comparison

4 Unit Testing

4.1 How to Test

Unit testing was done using Karma and Jasmine frameworks. There is a folder called test located in the src directory which contains the JavaScript file. In order to execute the testing framework, npm is required and all required packages are in a json file. Once all packages are installed, executing 'npm test' will run all the tests that are written in the test.js file.

4.2 Tests

The unit testing was done on the game logic which includes testing click functionality and some additional functionality such as the dropdown menus.

5 Changes Due to Testing

After performing unit testing on multiple browsers i.e.(Google Chrome, Safari, Firefox, Opera), it was apparent that the game would not work on Firefox.This was the only browser that was having issues with this. Unfortunately, the reason for the incompatibility with Firefox is due to a click listener thus it could not be fixed. From surveys, it was made clear that there was a scaling issue when playing on smaller displays, as such changes were made to accommodate smaller devices screen sizes. The game board would adjust accordingly to the size of the display. The most common suggestion was to make it clear to the player whose turn it was. Some people were unsure even though it said it at the top. As a result of this, we added a feature to show the character that would populate the game board if they were to click on that cell in a light grey color. This grey character is based on where the user's mouse is positioned.

6 Automated Testing

The automated testing was implemented with the Karma framework. This involved forcing the game to various states and verifying if the results matched what was expected. Examples of the various states were a game ending in a win and ending in a draw. The automated testing was also done to make sure the smaller core functionality of the game worked. These include ensuring the players alternated turns, a player could only click on a cell that had not yet ben populated.

7 Trace to Requirements

This section shows two traceability matrices. These are between the survey questions and the functional (Table 4) and non-functional (Table 5) requirements in SRS respectively.

Table 4: **Trace to Functional Requirements**

Test ID	Functional Requirements ID
in-test-id 1	FR1 , FR2,
in-test-id 2	FR3
in-test-id 3	FR4
log-test-id 1	FR5
log-test-id 2	FR5
log-test-id 3	FR1
log-test-id 4	FR1
logistic-test-id 1	FR2, FR6
logistic-test-id 2	FR6

Table 5: **Trace to Non-Functional Requirements**

Test ID	Non-Functional Requirements ID
survey-ques-id 1	UH3
survey-ques-id 2	UH1
survey-ques-id 3	LF1, LF2
survey-ques-id 10	PR1, PR2
survey-ques-id 4	UH3
survey-ques-id 11	OE1, OE2, PR1, PR2, PR3
survey-ques-id 5	CU1, CU2
survey-ques-id 6	HS1
survey-ques-id 7	LF2

8 Trace to Modules

This section shows a traceability matrix between the functional tests and modules in Module Guide.

Table 6: **Trace to Modules**

Test ID	Non-Functional Requirements ID
in-test-id 1	M1, M2, M3
in-test-id 2	M4, M5, M6
in-test-id 3	M6
log-test-id 1	M6
log-test-id 2	M6
log-test-id 3	M1, M2, M3 ,M4, M6
log-test-id 4	M1
logistic-test-id 1	M2, M4
logistic-test-id 2	M4

9 Code Coverage Metrics

In the Test Plan document, we strove to achieve 80% coverage with our automated, unit testing and manual tests . While not every unit test possible was executed, most were, as all of our functional and non functional metrics where validated as well as edge cases. Due to the nature of Ultimate Tic Tac Toe it was easy to validate functionality though a usability survey (refer to section 2) where beta testers confirmed real world performance and user experience. Because the game was so heavily tested in real world and automated tests we feel that our 80% coverage metric was reached.

References

Usability survey, December 2016. URL https://goo.gl/forms/AlxYBxGe3iYOKp2F2.

kennycason. Ultimate tic tac toe, July 2010. URL https://github.com/kennycason/ultimate_tictactoe/.