SE 3XA3: Test Plan
Ultimate Tic Tac Toe

Team 3
Kunal Shah - shahk24
Pareek Ravi - ravip2

November 1, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
October 19	0.0	Initial setup
October 31	1.0	Finished Document

Abstract

This document describes the Testing Plan for the Ultimate Tic Tac Toe Project.

1 General Information

1.1 Purpose

The purpose of testing this project's is to confirm all requirements that where outlined in the Requirements Specifications document have been met and to build confidence that the software for the project was implemented correctly.

1.2 Scope

The Test Plan presents a basis for testing the functionality of the re- implementation of Ultimate Tic Tac Toe. It has the objective of proving that Ultimate Tic Tac Toe has met the requirements specified in the Requirements Document and to attach metrics to those requirements so that adherence to requirements is quantifiable and can be measured. The testing plan acts as a means to arrange testing activities. This document will present what is to be tested of the software. It will also act as an outline of testing methods and outline the tools that will be utilized.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
JS	JavaScript
UTTT	Ultimate Tic Tac Toe
npm	Node Package Manager
html	HyperText Markdown Language
POC	Proof of Concept

Table 3: **Table of Definitions**

Term	Definition
Main Board	Full game board, all clickable regions
Inner Board [ID]	One of 9 Tic Tac Toe boards within the Main Board
Cell [ID]	One of 9 Tic Tac Toe elements of an Inner Board
Active Board	Inner board that next player is able to play on
Complete	Result has been determined for an Inner board
In-complete	Result has yet to be determined for an Inner board

1.4 Overview of Document

The Ultimate project will re-implement the project Ultimate Tic Tac Toe. The software will allow user to play the the game based on the rules defined by mathwithbaddrawings.com (mathwithbaddrawings, 2016). All the software's requirements can be referenced in the Requirements Document. This document demonstrates how the game Ultimate Tic Tac Toe will be tested, the testing schedule and the tools used.

2 Plan

2.1 Software Description

The software is a JavaScript implementation of the game Ultimate Tic Tac Toe.

2.2 Test Team

The individuals responsible for testing are Kunal Shah and Pareek Ravi. For a detailed breakdown of responsibilities refer to the Gantt Chart

2.3 Automated Testing Approach

The primary testing approach will be to use Karma Unit Testing to automate unit tests. Karma will automatically play the game in a headless browser following pre-defined moves. The unit testing framework will compare the results to expected values

2.4 Testing Tools

The tool that will be utilized for this project is Karma unit testing with the Jasmine framework. It will be used to automate the unit testing. There will be a package.json file in the src directory to download all dependencies required for Karma Unit Testing.
There will also be a Google Form used to survey user's experience and provide and avenue for feedback and suggestions.

2.5 Testing Schedule

Refer to the Gantt Chart for details about Testing Schedule

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 User Input

1. in-test-id1
   Testing if user's input being received

   Type: Functional, Dynamic, Automatic

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

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

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

   How test will be performed: This test will be performed automatically with the use of the Karma Unit Test. In the test, an element will be clicked and then the inner html of the text will be checked to see if it is updated to the character that represents that person's turn. There will also be a check on the variable that stores the all game information to see that it is updated.

2. in-test-id2
   Testing if user click on invalid inner board

   Type: Functional, Dynamic, Automatic

   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

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

How test will be performed: This test will be done automatically with theuse of the Karma Unit Test. When the use clicks on a inner board that they cannot select, the javascript will prevent anything from changing in the game data

3. in-test-id3
   Testing if user's input on cell already clicked

   Type: Functional, Dynamic, Automatic

   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

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

   How test will be performed: This test will be performed automatically with the use of the Karma Unit Test. In the test, an element in the inner board they are meant to click is already selected previously will be clicked. There will be no change to the board as the cell was previously selected. The script will not change anything as that cell is not a null value, it has the character representing the player that selected it.

3.1.2 Game Logic

1. log-test-id1
   Test if inner board gets completed

   Type: Functional, Dynamic, Automatic

   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

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

How test will be performed: This test will be performed automatically with the use of the Karma Unit Test. The cell element will be clicked by the tester. The logic will check if there is any three in a row in that inner board that the player just played in. If there is a three in a row, the entire board is deemed as completed and marked as so. The check for a three in a row is to check all 8 possible win scenarios, i.e. three rows, three columns and two diagonals

2. log-test-id2
   Test if inner board ends in draw

   Type: Functional, Dynamic, Automatic

   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

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

   How test will be performed: This test will be performed automatically with the use of the Karma Unit Test. The cell element will be clicked by the tester. The logic will check all 8 possible cases for a completed three in a row. If non exist and the inner board is full it is classified as a draw.

3. log-test-id3
   Test if next move can be made on any incomplete inner board

   Type: Functional, Dynamic, Automatic

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

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

   Output: All incomplete inner boards active

   How test will be performed: When the click is made on the inner board, the background of all inner boards that are not completed is set to blue. Based on the array which contains a map of the board, a loop through all the inner board elements and check their background colors. Any inner board that is not complete will have a background style blue.

4. log-test-id4
   Test if user turns are alternating

   Type: Functional, Dynamic, Automatic

   Initial State: Player with character O just made a move

   Input: Click at any available cell

   Output: The character changes to X

   How test will be performed: When a click is made, the character should change and the innerHTML should represent the other one.

3.1.3 Game Logistics

1. logistic-test-id1
   Test if game launches

   Type: Functional, Dynamic, Manual

   Initial State: User is in the file explorer

   Input: User launches the html file in browser

   Output: The game launches in all browsers and shows an empty UTTT game board.

   How test will be performed: The user will launch the game from their file explorer. If they are able to see a blank UTTT board, the game has launched.

2. logistic-test-id2
   Test if user input shows in window

   Type: Functional, Dynamic, Manual

   Initial State: User has just openned game

   Input: User clicks on any cell

   Output: Cell they clicked on should change appearance

   How test will be performed: If the user's click was registered, it would indicate that to the user by changing the cell they clicked on to the character that represents them. This will be seen graphically.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel

1. laf-test-id1
   This will be tested by survey question 4.

   Pass: 4.5/5 average rating on this question

2. laf-test-id2
   This will be tested by survey question 10.

   Pass 4.5/5 average rating for this question. The details will provide more information on repairs needed

3. laf-test-id3
   This will be tested by survey question 11.

   Pass 4.5/5 average rating for this question. The details will provide more information on repairs needed

3.2.2 Ease of Use

1. eou-test-id1
   This will be tested by survey question 1.

   Pass: 4.5/5 average rating on this question

2. eou-test-id2
   This will be tested by survey question 2.

   Pass 4.5/5 average rating for this question.

3. eou-test-id3
   This will be tested by survey question 3.

   Pass 4.5/5 average rating for this question.

3.2.3 Environmental Requirements

1. er-test-id1
   This will be tested by survey question 6.

   Pass: 4.5/5 average rating on this question. The details could lead to optimization

3.2.4 Performace Requirements

1. pr-test-id1
   This will be tested by survey question 5.

   Pass: 4.5/5 average rating on this question.

2. pr-test-id2
   This will be tested by survey question 7.

   Pass: 4.5/5 average rating on this question. The details could lead to optimization

3.2.5 Safety Requirements

1. safreq-test-id1
   This will be tested by survey question 9.

   Pass: 4.5/5 average rating on this question.

3.2.6 Cultural Requirements

1. culreq-test-id1
   This will be tested by survey question 8.

   Pass: 4.5/5 average rating on this question. The details could lead to optimization

3.2.7 Legal Requirements

1. legreq-test-id1
   Description: This game should have a rating suitable for all ages to play according to ESRB How: ESRB will rate the game based on their standards which are accepted universally Pass: The game has a rating of E for everyone.

3.2.8 Security Requirements

1. secreq-test-id1
   Description: When the game is implemented on a server, no personal data should be transfered How: Check all the packages that are transfered over the server Pass: Only game data is passed, no personal data

4 Tests for Proof of Concept

4.1 User Input

User inputs from the same local machine

1. Test first click

   Type: Manual

   Initial State: On Load

   Input: click on Inner Board [B00] Cell [1]

   Output: Player1 symbol (either X or O) Figure:1

   How test will be performed: User clicks input on game page loaded on an browser. User watches for graphical response.

2. Set Active Board

   Type: Manual

   Initial State: On Load

   Input: User click Inner Board [B03] Cell [5]

   Output: Active Board set to Inner Board [B11] Figure:2

   How test will be performed: User clicks input on game page loaded on an browser. User watches for graphical response.

3. All incomplete Inner Boards active when player sets complete inner board active

   Type: Manual

   Initial State: One move till Player 1 completes inner board [B02]

   Input: User click Inner Board [B02] Cell [3] Figure:**??**

   Output: all Inner Boards excluding Inner Board [B02] show blue background colour

   How test will be performed: Users clicks the following sequence on game page loaded on an browser. User watches for graphical response.

(a) Player 1 clicks on Inner Board [B02] Cell [5]
(b) Player 2 clicks on Inner Board [B11] Cell [3]
(c) Player 1 clicks on Inner Board [B02] Cell [7]
(d) Player 2 clicks on Inner Board [B20] Cell [3]
(e) Player 1 clicks on Inner Board [B02] Cell [3]

4.2 Game Logic

4. Complete Inner Board

   Type: Manual

   Initial State: One move till Player 1 completes inner board [B01]

   Input: User click Inner Board [B01] Cell [3]

   Output: Inner Board [B01] displays Player 1 symbol Figure:**??**

   How test will be performed: Users clicks the following sequence on game page loaded on an browser. User watches for graphical response.

   (a) Player 1 clicks on Inner Board [B02] Cell [5]
   (b) Player 2 clicks on Inner Board [B11] Cell [3]
   (c) Player 1 clicks on Inner Board [B02] Cell [7]
   (d) Player 2 clicks on Inner Board [B20] Cell [3]
   (e) Player 1 clicks on Inner Board [B02] Cell [3]

5. Complete Inner Board with draw(or tie)

   Type: Manual

   Initial State: One move till Player completes inner board

   Input: User click to cause Inner Board to draw Figure:**??**

   Output: Inner Board displays dash to indicate draw Figure:5

   How test will be performed: User clicks input on game page loaded on an browser. User watches for graphical response.

6. Win Full Game

   Type: Manual

   Initial State: One move till Player 1 wins full game

   Input: User click cell to complete last inner board needed to win Main Board Figure:??

   Output: Browser alert with player that won (X or O) Figure:7

   How test will be performed: Users clicks the following sequence on game page loaded on an browser. User watches for graphical response.

5 Comparison to Existing Implementation

There are 8 tests that compare the program to the Existing Implementation of the program. Please refer to:

- test 1 to 6 in Tests for Proof of Concept

  Reference section 4 of the document

- test 1 to 3 in Look and Feel Nonfunctional Requirements

  Reference section 3.2.1 of Tests for Nonfunctional Requirements

6 Unit Testing Plan

The Karma and Jasmine framework will be used to implement unit testing for this project. This will require the installation of multiple dependencies that cam be found in a .json file. Using the npm framework these can all be easily installed.

6.1 Unit testing of internal functions

In order to unit test the internal functions of the game, the values of the arrays will be checked after the function is run. There are arrays which are used to record the user who controls each inner board, and also an array which has the full 81 cells of the board in a multi-dimensional array. After running a function which will update these arrays, compare the results to expected. The values in the arrays will need to match. For the functions that have a return, providing the proper inputs and comparing it to the expected output, test cases can be created. Knowing that all functions are not testable, the aim will be to have at least 75% of the functions tested using unit testing.

6.2 Unit testing of output files

Given that this project does not have an output file, the unit testing out the output will be the graphical portion. The unit testing will involve clicking on cells of the game board and matching the change in the html to the expected result. For example, a click on an available playable cell, should result in the innerHTML of that cell to have either an X or O. Similar tests will be run for numerous use cases such as:

- Winning an inner board
- Inner board results in a draw
- Next move is set to entire board because previous click resulted in selection of completed inner board
- Competition of inner board corresponds to same inner board resulting in next click to be made to entire board
- Winning the entire game
- Entire game results in a draw

References

Bradley Braithwaite. Getting started with karma for angularjs testing, May 2015. URL http://www.bradoncode.com/blog/2015/05/19/karma-angularjs-testing/.

mathwithbaddrawings. Ultimate tic tac toe rules, October 2016. URL https://mathwithbaddrawings.com/2013/06/16/ultimate-tic-tac-toe/.

7 Appendix

Karma JS Installation Tutorial and Example source code (Braithwaite, 2015)

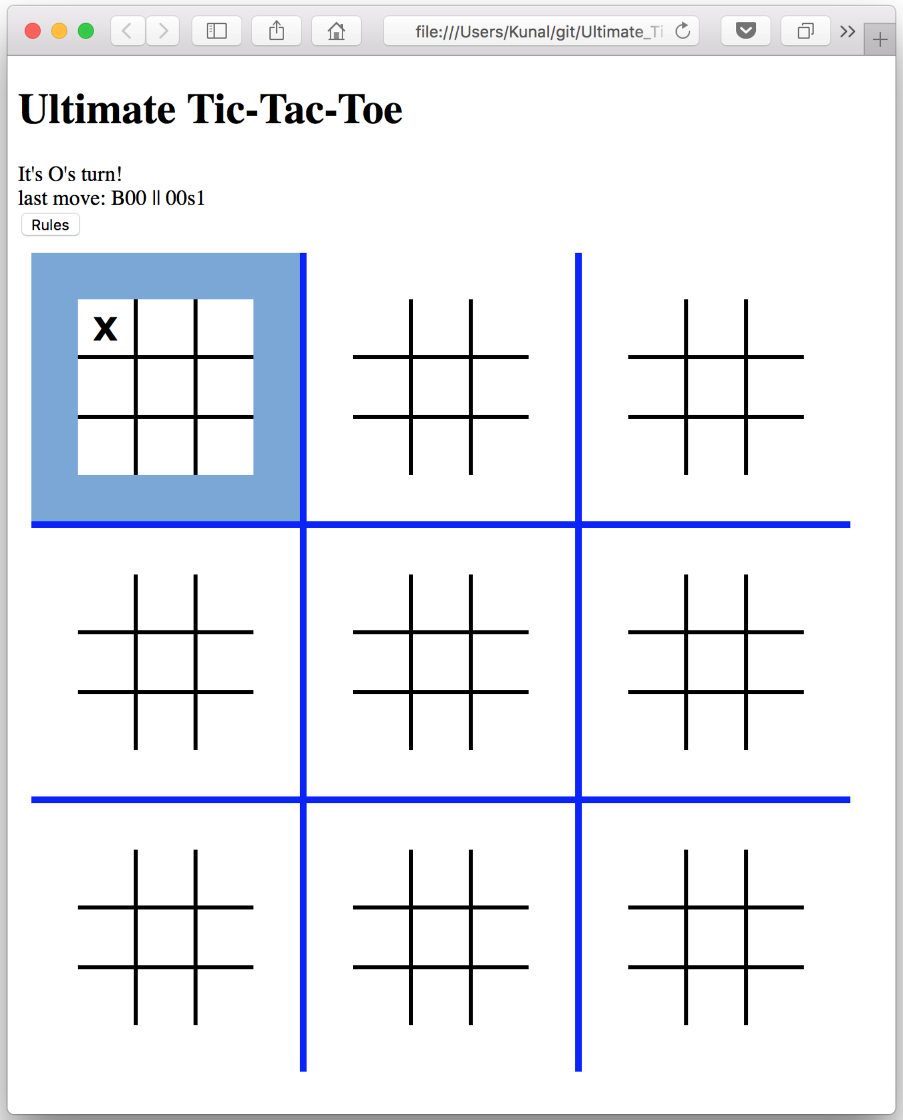
7.1 Symbolic Parameters

MAX_PLAYERS: Maximum number of concurrent players.

7.2 Usability Survey Questions?

Answers will be rated from 1 to 5, 5 being the highest

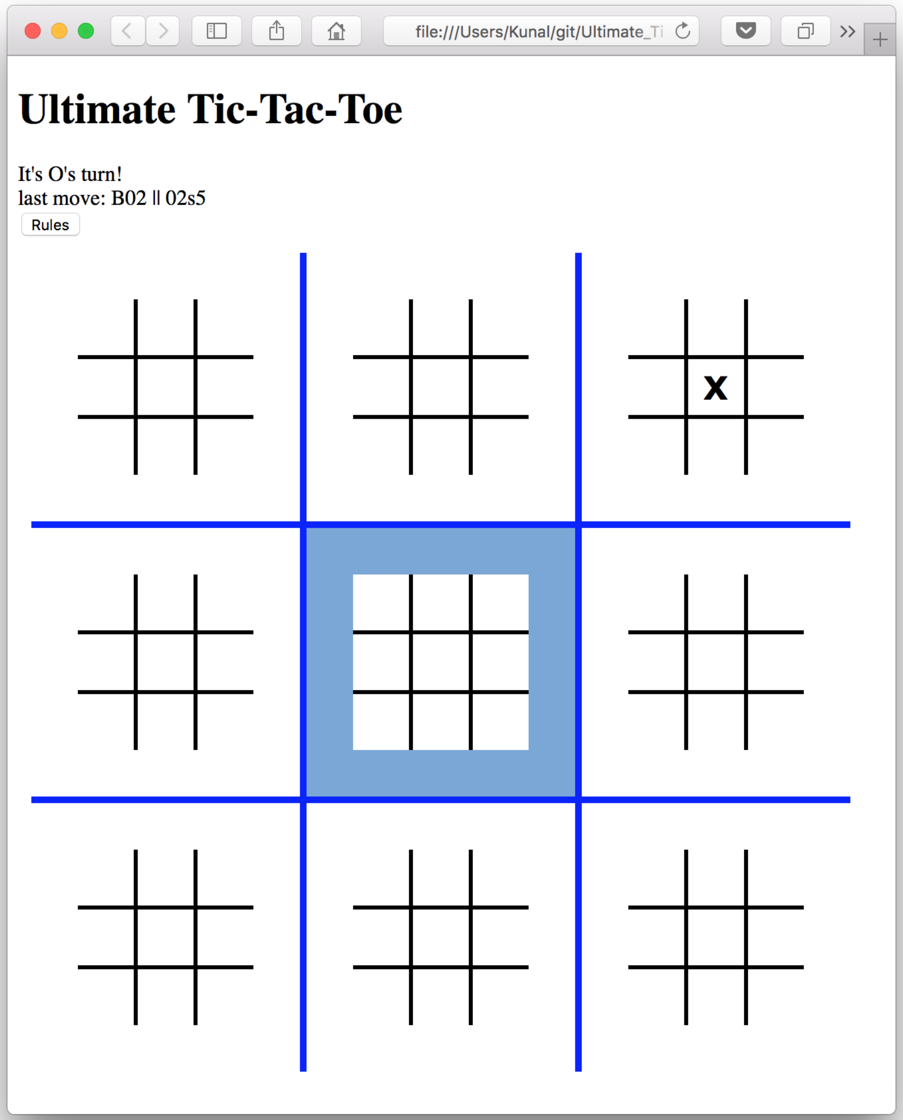
1. Is it easy to find the rules of the game?
2. Are the rules easy to understand?
3. Is it clear which player's turn it is? (X or O)
4. Is the color pallet visually appealing?
5. Is the response time satisfying?
6. Was the game able to run on your browser? Please indicate the browser you used
7. Was the game able to run smoothly on your device. Please indicate the computer used and the manufactured year
8. Were you offended by anything in the game. Please provide details
9. Did you encounter epileptic symptoms while playing the game
10. Did anything unexpected appear on the screen while playing. Please describe
11. Is the entire game board visible on the screen? Are there any aspects that are cut off? Please describe



Ultimate Tic-Tac-Toe

It's O's turn!
last move: B00 || 00s1

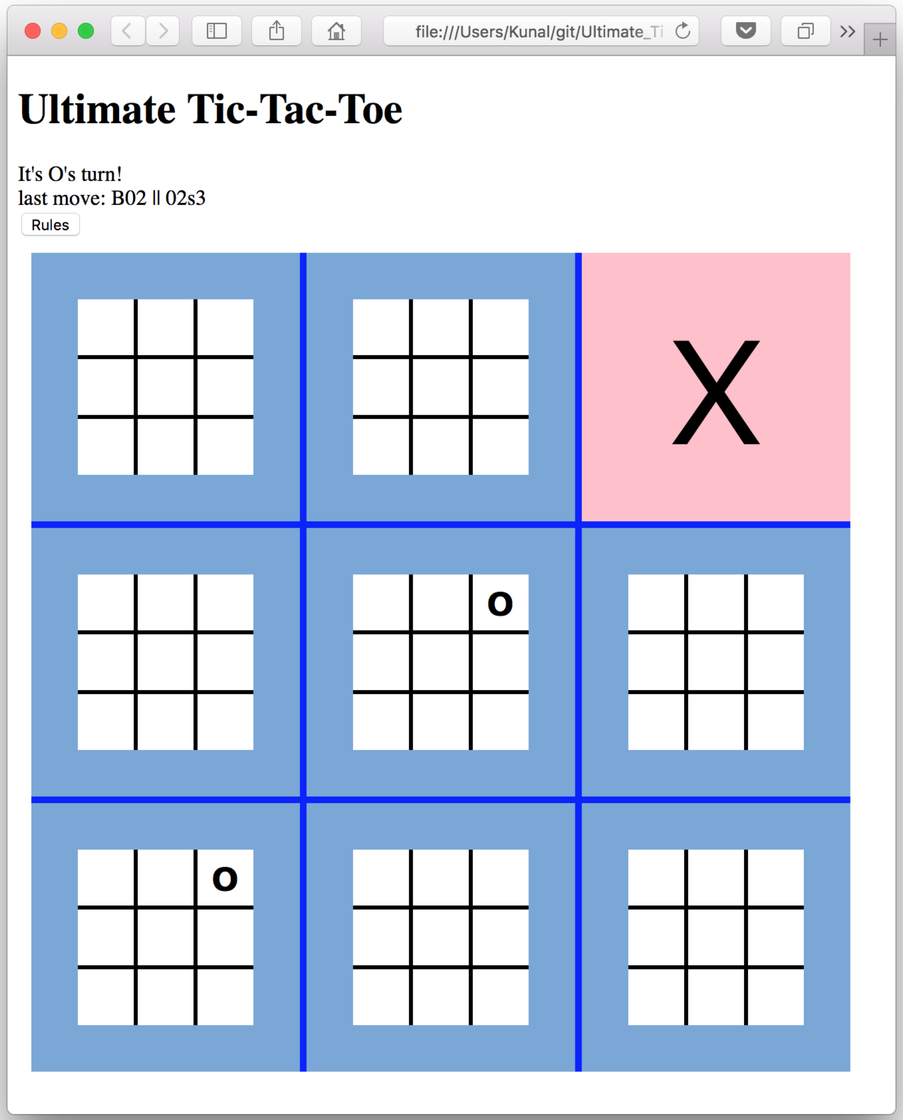
Figure 1: POC Test 1 Output



Ultimate Tic-Tac-Toe

It's O's turn!
last move: B02 || 02s5

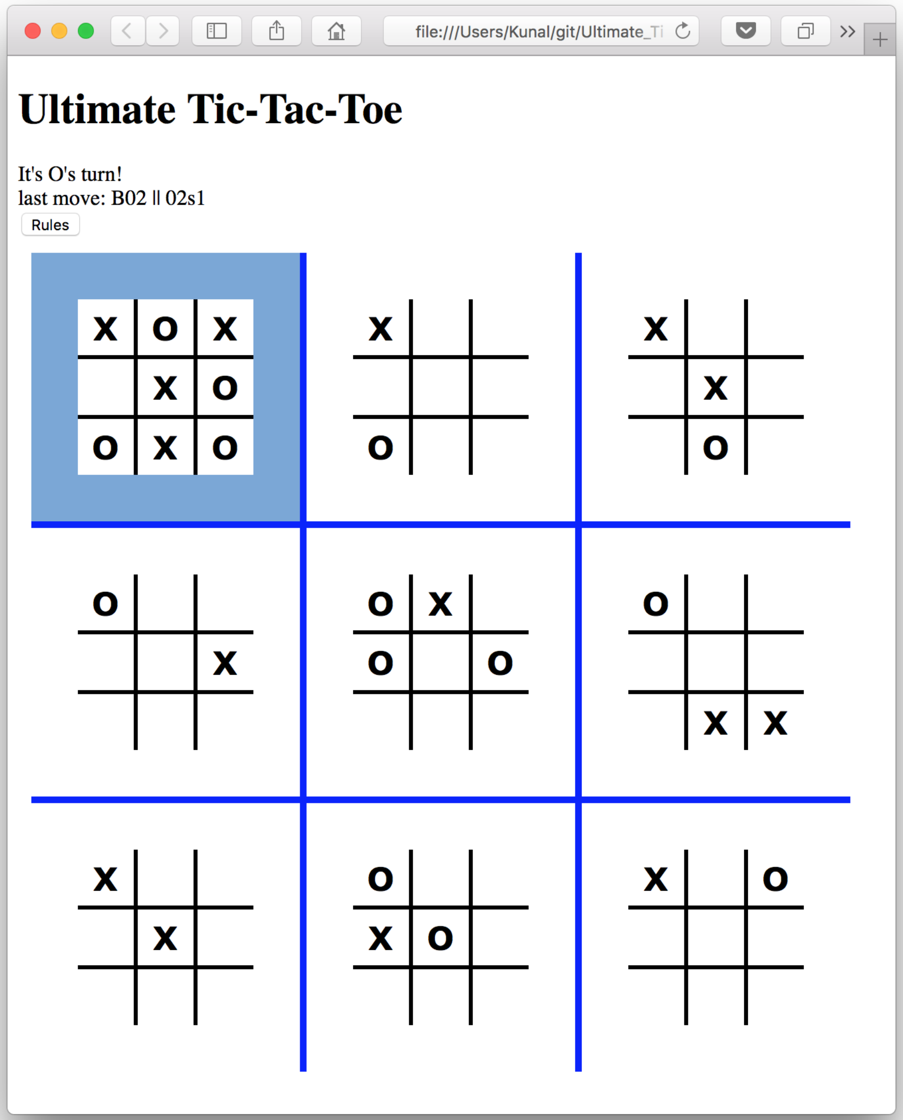
Figure 2: POC Test 2 Output



Ultimate Tic-Tac-Toe

It's O's turn!
last move: B02 || 02s3

Figure 3: POC Test 3 and Test4 Output



Ultimate Tic-Tac-Toe

It's O's turn!
last move: B02 || 02s1

X	O	X
	X	O
O	X	O

X		
O		

X		
	X	
	O	

O		
		X

O	X	
O		O

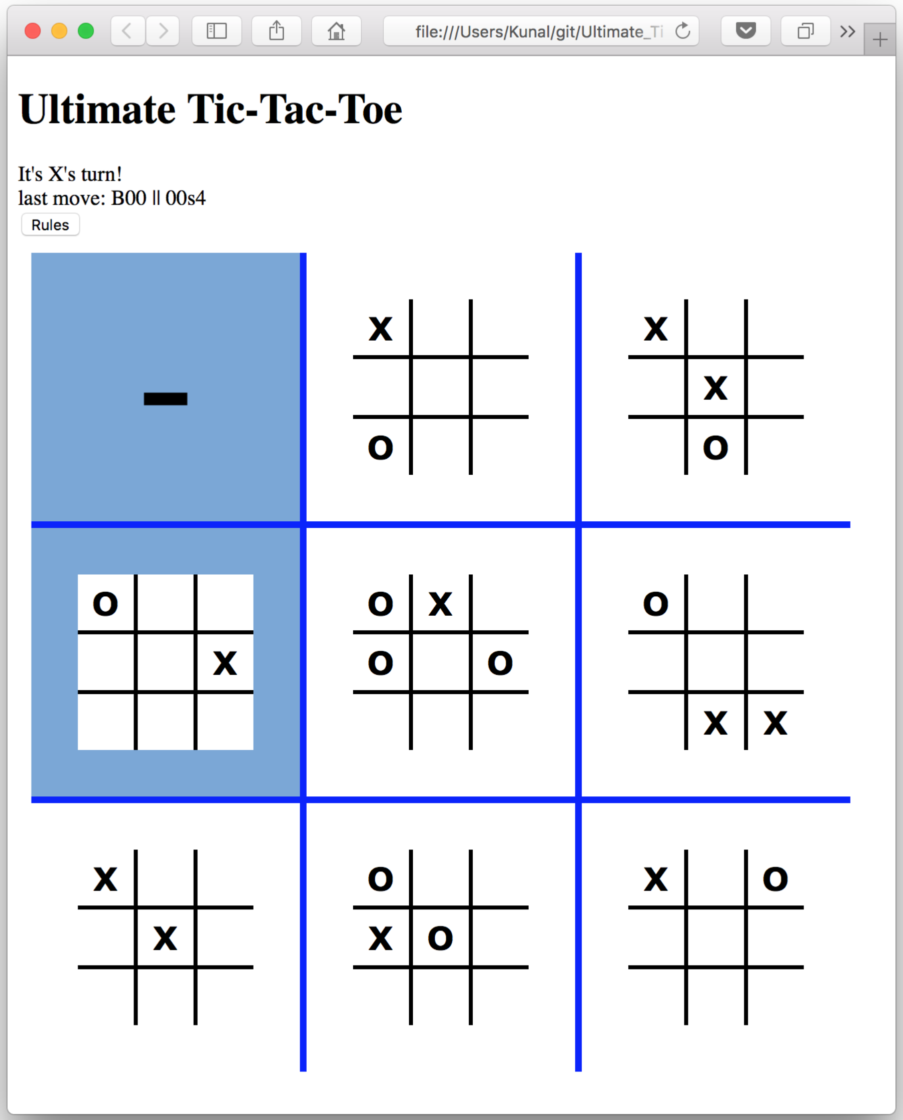
O		
	X	X

X		
	X	

O		
X	O	

X		O

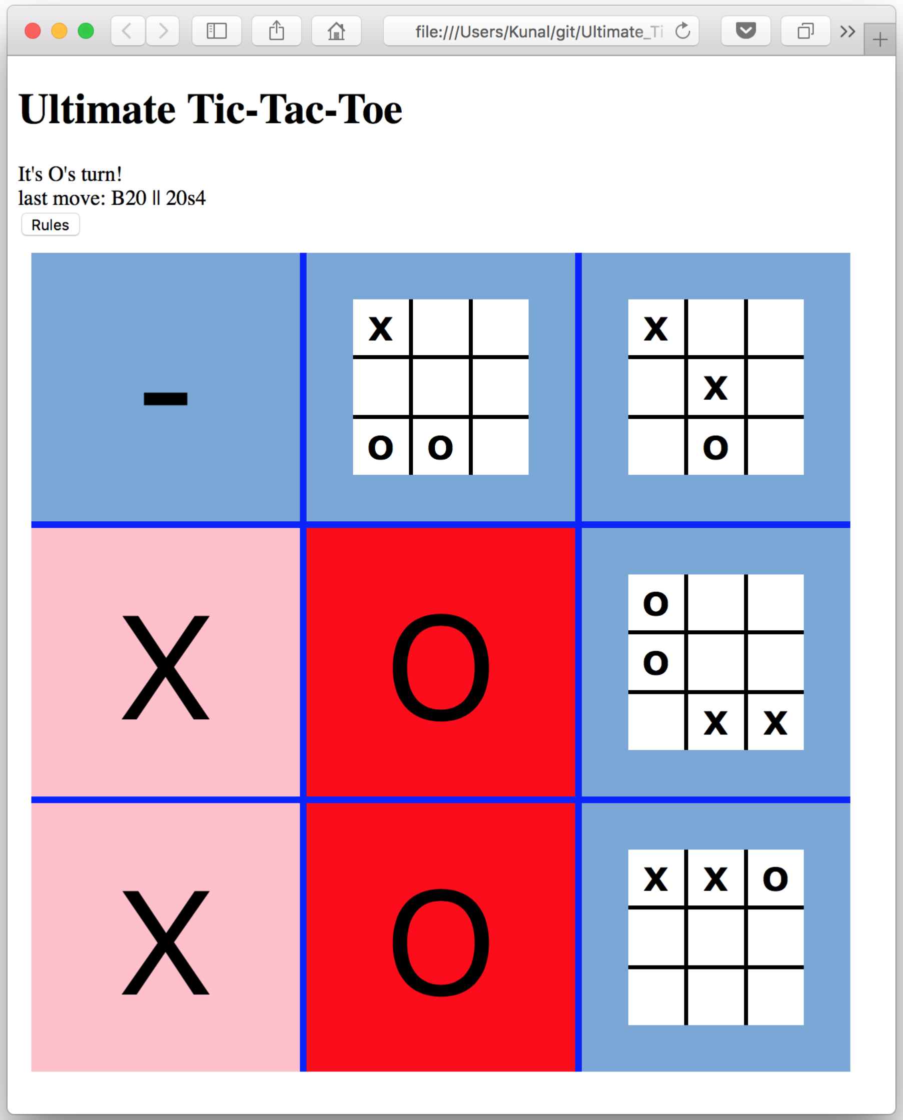
Figure 4: POC Test 5 Example input


file:///Users/Kunal/git/Ultimate_Ti

Ultimate Tic-Tac-Toe

It's X's turn!
last move: B00 || 00s4

Figure 5: POC Test 5 Output



Ultimate Tic-Tac-Toe

It's O's turn!
last move: B20 || 20s4

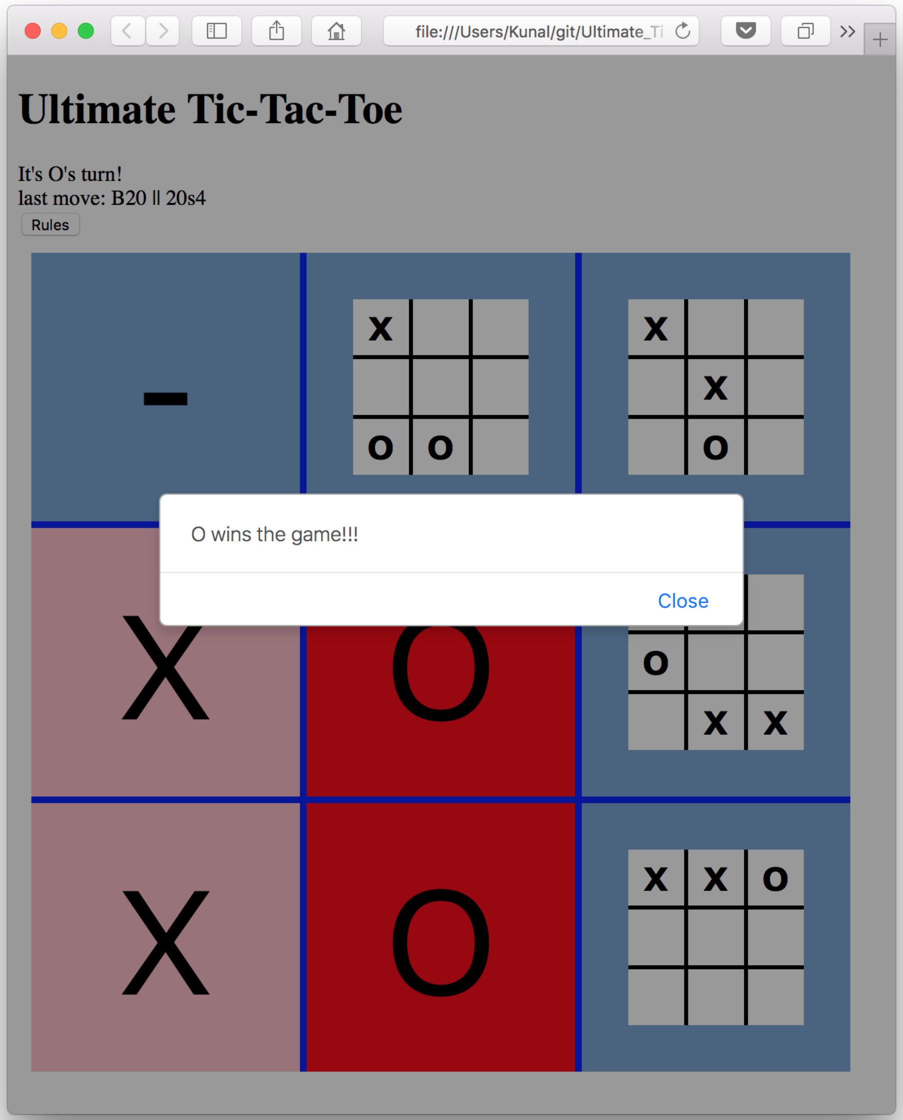
X		
O	O	

X		
	X	
	O	

O		
O		
	X	X

X	X	O

Figure 6: POC Test 6 Example input



Ultimate Tic-Tac-Toe

It's O's turn!
last move: B20 || 20s4

O wins the game!!!

Figure 7: POC Test 6 Output