SE 3XA3: Test Plan
Othello

Team #2, Team Name: JSM Corporation
Mohammed Mirajkar, mirajkam
Jinesh Patel, patelj60
Sankar Renganathan, renganas

December 5, 2018

Contents

List of Tables

List of Figures

1 General Information

1.1 Purpose

The purpose of this document is to discuss the test plan to validate and verify if the program is able to meet specific requirements. In order to ensure that the game will provide full functionality, a variety of test cases will be used to check for different aspects of the game such as winning scenarios and validity of player and AI moves

1.2 Scope

The scope of this document includes the various kinds of test plans, such as non-functional, functional, unit testing, and the details of those plans as well as the expected outcome. The tests will be conducted based on this test plan document and the scope and test plans shall be revised as the timeline of the project progresses.

1.3 Acronyms, Abbreviations, and Symbols

No acronyms, abbreviations or symbols are used in this document.

1.4 Overview of Document

What this document will be touching on are the specific test scenarios that will be used to verify and validate the game as well as the different testing methods that are going to be used in the testing process. The plan will include the specific tools that will be used to create the test scenarios as well as discussing how the functional and non-functional requirements will be tested.

2 Plan

2.1 Software Description

The software is a recreation of the game Othello with different visuals and features that doesn't stray from the core rules of the game. The game will allow for the user to choose an AI difficulty level and will play against that AI.

The game will be able to show players which moves are valid and which moves are invalid as well as be able to keep track of the score of the game. When the board is filled, the game will declare a winner based on the scoreboard.

2.2 Test Team

The team of testers consists of Mohammed Mirakar, Jinesh Patel and Sankar Renganathan. The test cases will be evenly divided among these testers and the testers will be responsible for reporting the results what they tested.

2.3 Automated Testing Approach

The ~~Jasmine~~ Jest framework for JavaScript will be used to develop test cases based on specific inputs and its corresponding outputs. Once these test cases are created, ~~Jasmine~~ Jest will be used to run automated tests which will then be compiled into a report format that shows us which test cases have passed and which have failed. The results will be based on whether the test cases are able to produce the expected output based on the input. Based on this, the developer team can work on fixing the faults of specific portions of code. Code coverage metrics will also be taken by ~~using a third party library integrated with Jasmine called Karma with the necessary plugins that will be determined at a later time~~ Jest .

2.4 Testing Tools

The tools defined here represent concecptual or physical tools that help organize and execute the processes for different testing procedures.

- ~~Jasmine which will be used for automated testing as well as Karma for code coverage.~~
- Jest which will be used for automated testing and code coverage.
- Visual Studio Code will be used to write and execute test cases as well as for static checking for syntax errors.
- White Box Testing:

  - This procedural method will be used to check individual functionalities in the code implementation of othello such as checking if the move validity function works properly.

- Black Box Testing:
  - This process will be used to verify the program without judging the code. This procedure will be used to check the functionality of user interfaces and response times.

2.5 Testing Schedule

After every function is created, test cases will be created based on how the function is expected to work. Testing will be done weekly by the entire team. Before the demonstration of the final product, the team will once again test every test case that has been created during the entire testing process. The specific details of the schedule can be seen in the Gantt chart.

3 System Test Description

Test Types :
Functional: The test is checking the functionality of the program.
Dynamic: The program has to be run in order to be tested.
Automatic: The program acts as the tester.
Unit: indvidual function/component is being tested.
Manual: Test requires a person to act as at tester.

3.1 Tests for Functional Requirements

3.1.1 User Options

Save and Load

1. FT-SL-1

   Type: Functional, Dynamic, Manual

   Initial State: Board is in state of play and no previous save state exists.

   Input: The "Save" button is pressed.

Output: A message is displayed to the user indicating the game has been saved.

How test will be performed: The tester wil press the save button and check whether the corresponsing cookie exists in their respective browser. Afterwards the tester shall play the game to whatever extent he deems neceesary in order to check that the game is still functioning.

2. FT-SL-2

   Type: Functional, Dynamic, Manual

   Initial State: Board is currently in state of play and a previous save state exists.

   Input: The "Save" button is pressed.

   Output: A message is displayed to the user indicating the game has been saved.

   How test will be performed: The tester wil press the save button and check whether the corresponsing cookie exists in their respective browser. The tester will also check that the cookie correspoding to the previous save state. Afterwards the tester shall play the game to whatever extent he deems neceesary in order to check that the game is still functioning.

3. FT-SL-3

   Type: Functional, Dynamic, Automatic, Manual

   Initial State: The board currently is in some random state and a previous save state exists.

   Input:The load button is clicked.

   Output: A message is displayed in the console indicating that the load function has passed test FT-SL-3.

   How Test Will Be Performed: The current board state is saved using the save button and recorded by the tester using a camerea. Afterwards the load button is clicked and the tester checks that the current board state is equivalent to the previous board state stored in the camera.

4. FT-SL-4

   Type:Functional, Dynamic, Automatic, Manual

Initial State: The board is currently in some random state and no previous save state exists.

Input: No input.

Output: A message is displayed in the console indicating that the load function has passed test FT-SL-4.

How Test Will Be Performed: The load button is clicked and the testers checks that the load does not occur.

3.1.2 Options

Reset and Undo and Score

1. FT-RUS-1

   Type:Functional, Dynamic, Manual

   Initial State: The board is in state of play.

   Input: Clicking a valid cell (indicated by a green circle).

   Output: The scoreboard displays the current score of the board.

   How test will be peformed: The board object is created and is in some random state. The reset function is run on the board and subsequently the test checks that current state of the board is equivalent to the default state. The test also checks that the current score of the board for white and black is equal to 2.

2. FT-RUS-2

   Type:Functional, Dynamic, Automatic, Unit

   Initial State: The board is currently in some random state.

   Input: No input required

   Output: A message is displayed in the console indicating that the reset function has passed the test.

   How test will be peformed: The board object is created and is in some random state. The reset function is run on the board and the test subsequentlytt checks that current state of the board is equivalent to the default state. The test also checks that the current score of the board for white and black is equal to 2.

3. FT-RUS-3

   Type:Functional, Dynamic, Automatic, Unit

   Initial State: The board is currently in some random state.

   Input: No input required.

   Output: A message is displayed in the console indicating that the undo function has passed the test.

   How test will be peformed: The board object is created and is in some random state. The test records the current state into a variable and the score of black and white respectively into variables and the test changes the board state by making a random move. Afterwards the undo function on the board is run and the test asserts that the current board state, and current scores of black and white are equal to the value stored in their respective history variables.

4. FT-RUS-4

   Type:Functional, Dynamic, Manual

   Initial State: The board is in default state.

   Input: No input required.

   Output: The score board should change to reflect the number of white and black pieces in the current board state.

   How test will be peformed: The tester plays the game and systematically checks that the scoreboard for each piece correctly reflects the board. The tester performs this task until he is satisfied.

3.1.3 Gameplay

Click Events

1. FT-CE-1

   Type: Functional, Dynamic, Manual

   Initial State: The board is in state of play and it is the players turn.

   Input: The player clicks a cell with the green option button.

   Output: The correct board state should be displayed according to the rules of Othello.

How test will be performed: The tester will systematically play the game and click the cells that are valid. He will then check that the corresponding board state is correct based on the rules of othello. This test shall be performed concurrently with FT-CE-3.

2. FT-CE-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: The board is in state of play and it is the AI's turn.

   Input: No input required.

   Output: The corresponding clicked cell glows red, the board state should change to reflect AI's move.

   How test will be performed: The tester will systematically play the game and click the cells that are invalid. He will then check that the corresponding board state is correct based on the rules of othello.

3. FT-CE-3

   Type: Functional, Dynamic, Manual

   Initial State: The board is in state of play and it is the players turn.

   Input: The user clicks an invalid cell (any cell without the green option button).

   Output: The corresponding clicked cell glows red and the board state stays the same.

   How test will be performed: The tester will play the game and systematically attempt to make invalid moves until the game has been finished. This test shall be performed concurrently with FT-CE-1.

4. FT-CE-4

   Type: Functional, Dynamic, Manual

   Initial State: The board is in state of play and it is the AI's turn.

   Input: The user clicks a valid cell (indicated by a green circle).

   Output: The corresponding clicked cell glows red, the board state should change correctly to reflect AI's move.

   How test will be performed: The tester will systematically play the game and during the ai's turn the tester shall attempt to make a valid

move . He will then check that his move did not affect the board and that the move made by the ai is a valid move . This test shall be performed concurrently with FT-CE-5.

5. FT-CE-5

   Type: Functional, Dynamic, Manual

   Initial State: The board is in state of play and it is the AI's turn.

   Input: The user clicks an invalid cell (any cell without a green circle).

   Output: The corresponding clicked cell glows red, the board state should change to reflect AI's move.

   How test will be performed: The tester will systematically play the game and during the ai's turn he shall attempt click the cells that are invalid . He will then check that the corresponding board state follows the rules of othello. This test shall be performed concurrently with FT-CE-4.

Game End

1. FT-GE-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: The board is full (does not matter whos turn it is).

   Input: No input.

   Output: A message is displayed in the console indicating that the gameEnd function has passed test FT-GE-1.

   How test will be performed: The board is created and the state is set to full. The gameEnd function is then run and the test checks that the gameEnd functions returns true(indicating game has ended).

2. FT-GE-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: The board contains no valid moves for the player and it is the players turn.

   Input: No input.

Output: A message is displayed in the console indicating that the gameEnd function has passed test FT-GE-2.

How test will be performed: The board is created in its state to have no moves for the player and the turn is set to be the players turn . The gameEnd function is then run and the test checks that the gameEnd functions returns true(indicating game has ended).

3. FT-GE-3

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: The board contains no valid moves for the AI and it is the AI's turn.

   Input: No input.

   Output: A message is displayed in the console indicating that the score funciton has passed test FT-GE-3.

   How test will be performed: The board is created in its state to have no moves for the ai and the turn is set to be the ai's turn . The gameEnd function is then run and the test checks that the gameEnd functions returns true(indicating game has ended).

3.2 Tests for Nonfunctional Requirements

3.2.1 Performance

State Change

1. NFT-SC-1

   Type: Functional, Dynamic, Automated

   Initial State: The game has been loaded into an HTML browser.

   Input: The input is the user playing the game.

   Output: The result of this test shall determine if all individual animations occur within `ANIM_TIME`

   How test will be performed: There shall be individual test cases for each animation that is releveant to game play (not main screen, end screen, and buttons), where the animation will be triggered and after a timer of `ANIM_TIME` is triggered, the test will check the state of the UI element to see if the animation has completed.

2. NFT-SC-2

   Type: Functional, Dynamic, Automated

   Initial State: The game has been started and the board state of Othello is in any valid state according to the rules and it is the player's turn.

   Input: The input is the user clicking on a cell that provides a valid move according to Othello rules.

   Output: The result shall be the state of the board after the move is made. The state change shall be shown to the user in less than or equal to `RESP_TIME`.

   How test will be performed: The test shall be performed through automation where the player is virtual and makes a move. A timer is started that will trigger after `RESP_TIME`, and check if the user interface elements have updated to reflect the new board state. The value of `RESP_TIME` inclues the time it takes to animate the board state, that is `RESP_TIME` $\geq$ `ANIM_TIME`.

3. NFT-SC-3

   Type: Functional, Dynamic, Automated

   Initial State: The game has been started and the board state of Othello is in any valid state according to the rules and it is the AI's turn.

   Input: There is no input from the player.

   Output: The result shall be the state of the board after the move is made. The AI must decide and make a move within `AI_TIME`.

   How test will be performed: The test shall be performed through automation where the AI makes the move. A timer is started that will trigger after `AI_TIME`, and check if the AI has made a move and the board state has been updated to reflect the move.

3.2.2 Look and Feel

User Interface

1. NFT-UI-1

   Type: Functional, Dynamic, Manual

Initial State: Any state during the process of the game including the main screen and end screen.

Input: The input is having a person play the game.

Output: The output would be the reflection of the user interface by the person on a scale of 1-10.

How test will be performed: The test will be performed by having multiple people to play the game and comment on the user interface to get a general consensus of the appealability of the user interface. (multiple = 5 people)

2. NFT-UI-2

   Type: Functional, Dynamic, Manual

   Initial State: The game has been started and there are both the player's and opponent's game pieces on the board.

   Input: The input is having a person play the game.

   Output: The output would be whether the player is able to clearly understand which board pieces belong to them.

   How test will be performed: The test will be performed by having multiple people to play the game and observe whether they are able to quickly identify which game pieces belong to them (quickly is less than a minute, multiple = 5 people).

3. NFT-UI-3

   Type: Structural, Dynamic, Manual

   Initial State: The game has been started and there are both the player's and opponent's game pieces on the board.

   Input: The input is having the tester play the game.

   Output: The output would be whether the implementation is able to handle and process these clicks in quick succession and maintain the correctness of the game state and structual integrity of the user interface.

   How test will be performed: The test will be performed by having the tester click as fast as physically possible to stress test the implementation.

3.2.3 Usability

1. NFT-U-1

   Type: Functional, Dynamic, Manual

   Initial State: The game has been started and the board is in the start state.

   Input: The input is having a person play the game.

   Output: The output would be whether the player is able to quickly figure out how to interact with the game. (quickly is less than a minute).

   How test will be performed: The test will be performed by having multiple people to play the game and observe whether they are able to quickly figure out how to interact with the game (quickly is less than a minute, multiple = 5 people).

3.2.4 Portability

Render

1. NFT-R-1

   Type: Functional, Dynamic, Manual

   Initial State: The game software has been loaded into the default browser of the device.

   Input: Tester interaction through default method for device (e.g. touch for mobile device).

   Output: Game state updates and displays new state to the tester.

   How test will be performed: The software shall be tested on multiple devices with varying screen sizes including desktops, laptops, tablets, and mobile devices on several types of platforms to check whether the game renders and provides correct user interface updates upon user interaction. The test will be performed manually on the major device platforms of the market (iOS, Android, Windows, macOS). as well as on the correspoonding OS browsers (IE Edge, Chrome, Safari).

3.2.5 Security

File Manipulation and Access

1. NFT-FMA-1

   Type: Functional, Static, Manual

   Initial State: No initial state necessary for this test.

   Input: Tester will review the source code.

   Output: Tester will determine if any direct file access occurs in the source code.

   How test will be performed: The testers shall perform a code review and determine if any part of the source code directly attempts to access files for reading.

2. NFT-FMA-2

   Type: Functional, Static, Manual

   Initial State: No initial state necessary for this test.

   Input: Tester will review the source code.

   Output: Tester will determine if any direct file manipulation occurs in the source code.

   How test will be performed: The testers shall perform a code review and determine if any part of the source code directly attempts to manipulate files for storage.

4 Tests for Proof of Concept

The main demonstration for the proof of concept is that the valid moves are determined correctly as these make up the majority of the logic of the Othello game, and that the move's selected by the player and AI correctly change the state board. Since these are the two main functions, they are performed as unit tests for their respective functions in the source code. The rest of the program consists of the integration of the game board with the system controlling the user interface and is thus done as a manual integration test.

4.1 State

Validity

1. UT-V-1

   Type: Functional, Dynamic, Automated, Unit

   Initial State: There is an instance of a valid board state (predetermined by tester) loaded into the testing system.

   Input: The input to the 'valid' function is the board state and a cell to check for valid moves for a particular player (player or AI), (predetermined by tester).

   Output: The output of the valid moves will be a `true/false` represented as 0 for `false` and any other number as `true` value.

   How test will be performed: The test will be performed by comparing the value produced by the 'valid' function with a manually computed `true/false` value for that particular state.

Moves

1. UT-M-1

   Type: Functional, Dynamic, Automated, Unit

   Initial State: There is an instance of a valid board state (predetermined by tester) loaded into the testing system.

   Input: A virtual player will make a valid move (cell is predetermined by tester).

   Output: The 'move' function will mutate the board state.

   How test will be performed: The test will be performed by comparing the new board state produced by the 'move' function with a manually computed board state depending on the move.

4.2 Game

Integration

1. UT-I-1

   Type: Functional, Dynamic, Manual

   Initial State: The game is loaded into an HTML browser.

   Input: The tester plays the game and tests the integration of the controls with the game board.

   Output: The tester shall comment on any bugs found out during the testing period.

   How test will be performed: The tester shall run through the program attempting to find any bugs with the integration for a set amount of time.

5 Comparison to Existing Implementation

The existing implementation of the Othello game does not include a means for automated testing of the program. Thus the only testing method is manual testing where the tester plays the game and validates the integrated functionality of the game.

6 Unit Testing Plan

6.1 Unit testing of internal functions

Internal functions will be tested based on predetermined inputs and will be validated based on if it produces the corresponding outputs. The white box testing procedure will be used to verify individual functions. The functions will be split into functions that change the state of the game and functions that determine the validity of the game. The state changing functions will have stubs that pretend the game is still valid, and the testing for these functions will be considered a success if the function is able to modify the game board to match the expected output for the game board. For example, if a user places a piece on the top left square, the test will check if the new game board contains the existing pieces plus an additional white piece on the top left square. The validity functions will be called by drivers that represent a certain game board state. The official rules of Othello will help determine if the game state contradicts the rules, which will determine if the test passes

or fails. Our goal for test coverage is to aim to test as much relevant code as possible based on the functions being tested. Our team is aiming for at least an 85% coverage of testing.

6.2 Unit testing of output files

Currently, the only output file that will be generated will contain all elements that describe the state of the game and will be stored in local storage. Integration testing will be used to test if the file is able to be recovered after exiting the game and unit tests will be used to check if the contents in the file are consistent to when the file was saved.

7 Appendix

7.1 Symbolic Parameters

- `ANIM_TIME`: The maximum time it takes to complete all animations regarding actual game play (does not include controls or main/end screen animations).
- `RESP_TIME`: The maximum time it takes to display the new board state to the screen for visual display. This parameter includes the animation timeing and thus `RESP_TIME` $\geq$ `ANIM_TIME`.
- `AI_TIME`: The maximum time it takes for the AI to decide and make a valid move according to Othello rules.

7.2 Usability Survey Questions?

- How familiar is the user interface compared to other similar board games such as Chess or Checkers?
- Is the interaction with the game relatively easy to learn on average compared to other games in a similar category?
- How would you rate the usability of the interface on a scale of 1-10?
- Did you have any struggles with the user interface, if so, what are they?
- Do you think the user interface follows the general style of the modern day online world?
- What is your win percantage against each AI?
- How often do you use the playback feature?

Table 1: **Revision History**

Date	**Version**	**Notes**
2018-10-21	0.1	Worked on Functional Tests (Rough Draft)
2018-10-23	0.2	Completed section 5 of document
2018-10-23	0.3	Completed non-functional tests of document
2018-10-23	0.3.1	Added symbolic paramaters
2018-10-23	0.3.2	Added usability survey questions
2018-10-23	0.4	Added scope and updated section 1.3
2018-10-23	0.5	Completed section 6
2018-10-23	0.5.1	Updated section 2
2018-10-23	0.6	Completed PoC testing.
2018-11-28	1.0	Revised output files