SE 3XA3: Software Requirements Specification
Othello

Team #2, JMS Corporation
Mohammed Mirajkar, mirajkam
Jinesh Patel, patelj60
Sankar Renganathan, renganas

December 5, 2018

Contents

List of Tables

List of Figures

Date	**V**ersion	**N**otes
October 3	0.1	Updated Project Drivers and Issues and created requirements
October 5	0.2	Added in the Functional and Non function sections of the SRS.
November 28	1.0	Added additional features implemented

Table 1: **R**evision History

This document describes the requirements for JMS Corporation's development for the web-based implementation of Othello.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of the project is to revamp a python implementation of the game Othello and bring the game into a web-based implementation of itself. The web program will improve on any issues that currently exist with the python implementation and will add additional features to enhance the experience of the interface and the game itself.

1.2 The Stakeholders

1.2.1 The Client

The client of the project is the McMaster Software Department whom would like to see the redevelopment of the python implementation of Othello and see a step by step process of the development cycle and whom would review the final product.

1.2.2 The Customers

The customers of the project will be the users that play the game on the web platform. Users will be surveyed and will provide feedback on which specific aspects of the game need improvement and which specific aspects work well.

1.2.3 Other Stakeholders

Other stakeholders would be the software team tasked with maintaining the game after its initial release by making occasional updates as well as making bug fixes to improve the game throughout time. This team will be able to take feedback from users to modify the game to fit their preferences.

1.3 Mandated Constraints

1.4 Development Constraints

- The game will be built using HTML, CSS and JavaScript and the developers must have access to an IDE that supports these languages
- Game must be based off of the python implementation of Othello
- Functional requirements must not conflict with the in-game rules of Othello

1.5 Environmental Constraints

- Software can only be run on web browsers that can run JavaScript
- The game can only be accessed if the user has a device that is connected to the internet
- Users can only access the game on devices that support web browsers
- User interactions with the game require a device that allows mouse functionality or touch functionality

1.6 Time Constraints

- A working prototype that includes some of the functional requirements is to be finished by October 16th 2018
- The full game that includes all requirements will be built by November 27th

1.7 Naming Conventions and Terminology

- JavaScript: Scripting program that allows for web implementation
- HTML: Markup language for creating websites
- CSS: Programming language that helps style the HTML elements
- Program/Game/Software: Refers to Othello

- User: Players of the game
- Visual Studio: IDE used for programming
- IDE: Interface for writing code
- UI: User Interface
- Model: Part of the code that deals with functionality
- View: Part of the code that deals with the UI
- Controller: Part of the code that deals with user interaction

1.8 Symbolic Parameters

- AI_EXEC_TIME: 1 second
- RESP_TIME: 0.5 second
- ANIM_TIME: 0.5 second

1.9 Relevant Facts and Assumptions

The previous implementation of the game contains approximately 1000 lines of code not including the different libraries that were used. Othello is a very popular strategy board game that has its very own World Championships. This factor alone could peak the interest of multiple people online thus attracting them to try out the web program. Some assumptions made about the project is that there will be underlying bugs in the game upon initial release, thus a team will be dedicated to the maintenance of the game after its release. Another assumption is that all current web browsers run JavaScript and are capable of running Othello as well. The final assumption made is that the majority of users will have the tendency to be impatient thus the program will have a fast execution time between moves

2 Functional Requirements

2.1 The Scope of the Work and the Product

The scope of the work includes the deliverable of the core product being built, the Othello game. The scope of the product itself will depend on the given timeframe until Dec 6th, 2018. If time persists after the creation of the core features of the Othello game, advanced AI's will be built otherwise a simple AI will do for playing against. The product shall have an interactive UI to interface with along with animations during state changes of the game board. Milestones of the project include the creation of the virtual game board, interaction with the virtual game board, and lastly the creation of the AI opponent. The deliverables and their timeline will be the same as determined by the course instructor.

2.1.1 The Context of the Work

The context of the work is going to be a web application that is created using web technologies such as HTML/CSS/JavaScript and possibly other web libraries that will ease the developers in creating the product such as Reast and Pose, both of which help in creating the user interface.

2.1.2 Work Partitioning

Event Number	**E**vent Name	**I**nput	**O**utput
1	State Creation	Developer Code	Internet Browser
2	Graphics Creation	Developer Code	Internet Browser
3	User Interation	Developer Code	Internet Browser
4	AI Creation	Developer Code	Internet Browser
5	Final Edits	Developer Code	Internet Browser

Table 2: **W**ork Partitioning

2.1.3 Individual Product Use Cases

Use Case #1.
Related Requirements: FR1-9

Initiating Actor: The player of the game.
Actor's Goal: Acheive the winning state.
Participating Actors: None.
Preconditions: The state of the game is not in the win/lose (ing) state.
Postconditions: The state of the game is in the win/lose (ing) state.
Flow of events for main success:
→ 1. The initiating actor places their game piece on the board.
← 2. The system displays the new state.
← 3. The system AI places their game piece on the board.
← 4. The system displays the new state.
→ 5. This continues until a win/lose state occurs.
Flow of events for Extensions:
→ 1. The initiating actor invalidly places their game piece.
← 2. The system displays an error state.

2.2 Functional Requirements

1. **Requirement #:** FR1.
   Description: The game shall have a main screen for standby.
   Rationale: To allow the user to start the game when they choose to.
   Fit Criterion: Execute game, if no action is produced by the game without user input in a certain timeframe, the game is is standby mode.
   Originator: Jinesh Patel. Oct 4th, 2018

2. **Requirement #:** FR2.
   Description: The game shall have a screen for game win/loss.
   Rationale: To allow the user to recieve feedback on their performance.
   Fit Criterion: Execute game until a win or loss has occured to display the win/loss screen.
   Originator: Jinesh Patel. Oct 4th, 2018

3. **Requirement #:** FR3.
   Description: After user input from the win/loss screen, the game will return to the main screen with the game reset to it's initial state.
   Rationale: To allow the user to exit the game or replay another round.
   Fit Criterion: Wait for user input at the win/loss screen and see if the game resets to the main screen.
   Originator: Jinesh Patel. Oct 4th, 2018

4. **Requirement #:** FR4.
   Description: The game AI will perform only valid moves based on the game board state.
   Rationale: If the AI does not perform only valid moves, it poses an anfair advantage.
   Fit Criterion: Let the AI only choose from a certain set of valid places it can make the move.
   Originator: Jinesh Patel. Oct 4th, 2018

5. **Requirement #:** FR5.
   Description: The game, during play, shall show the current state of the board through a UI.
   Rationale: The user needs some method of input and output to the state of the game.
   Fit Criterion: Check to see that the web application renders the game board correctly.
   Originator: Jinesh Patel. Oct 4th, 2018

6. **Requirement #:** FR6.
   Description: The game shall allow the user to see the possible valid moves.
   Rationale: The user needs some method to know where they can place their game piece.
   Fit Criterion: Check to see that the web application renders the valid moves boxes correctly.
   Originator: Jinesh Patel. Oct 4th, 2018

7. **Requirement #:** FR7.
   Description: The game shall provide feedback to the user if they attempt to perform an invalid move.
   Rationale: The user needs some method to know their attempt is invalid.
   Fit Criterion: Perform an invalid move to see if any feedback is shown.
   Originator: Jinesh Patel. Oct 4th, 2018

8. **Requirement #:** FR8.
   Description: The game shall provide a means to save the current state of the game.
   Rationale: The user should be able to stop playing and resume at

another time.
Fit Criterion: Test to see if writing and reading the state to memory is consistent and the game is able to continue from that state.
Originator: Jinesh Patel. Oct 4th, 2018

9. **Requirement #:** FR9.
Description: The game shall provide a means to reset the state of the game.
Rationale: The user should be able to quit and start a new game.
Fit Criterion: Test to see if the game state has been reset.
Originator: Jinesh Patel. Oct 4th, 2018

10. **Requirement #:** FR10.
Description: The game shall be able to return to previous states of the game.
Rationale: The user should be able to undo their moves.
Fit Criterion: Test to check if a previous version of the board is being used currently.
Originator: Sankar Renganathan. Nov 28th, 2018

11. **Requirement #:** FR11.
Description: The game shall be able to store state of game in local storage.
Rationale: The user should be able to save and load games.
Fit Criterion: Test to check if loaded game matches the exact state of the game when saved.
Originator: Sankar Renganathan. Nov 28th, 2018

12. **Requirement #:** FR12.
Description: The game shall be able to show all moves sequentially at the end of game.
Rationale: The user should be able to playback the events that unfolded throughout the game.
Fit Criterion: All states in the playback are equivalent to the history stack in that order.
Originator: Sankar Renganathan. Nov 28th, 2018

3 Non-functional Requirements

3.1 Look and Feel Requirements

- **Requirement** # NF1:
  The JSM team logo must be visible at every instance of the game
  Reasoning: The team logo shall be represented to build a rapport with the user
  Criterion: Check if the Logo of the team is visilbe during all stages of the game
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement** # NF2:
  The game graphics shall be generally aesthetically pleasing
  Reasoning: The user interface should be aesthetcally pleasing to the user to increase user enjoyment
  Criterion: Ask various people to rate if the user interface is aesthetically pleasing.
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement** # NF3
  The game pieces and the board shall constrast in color
  Reasoning: The user should be able to differentate their pieces from the users pieces
  Criterion: Ask various people to rate if the games pieces contrast in colour.
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement** # NF4:
  The games pieces shall be seperated into two constrast colors
  Reasoning: The user should be able to see their pieces on the board. The game pieces should not blend in with the background that is the game board
  Criterion:Check with various people to see if they can differentiate they can see their game piece on the game board
  Originator: Mohammed Mirajkar, Oct 4th, 2018

3.2 Usability and Humanity Requirements

- **Requirement** #NF5:
  The user interface for the product shall be familiar to users and shall follow existing rules consistent
  with the open source python implementation of othello on github.
  Reasoning: The python implementation of othello follows user interface rules that are used in mainstream programs. A familiar user interface is much more easier to use.
  Criterion: Check with various people if they can immediately use the user interface for the game
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement** #NF6:
  Children of atleast age 6 should be able to use the user interface to play the game
  Rationale: The user interface should be incredibly easy to learn, such that even someone of low intellect should be able to use the interface.
  Criterion: Check with various children of age 6 and above to see if they can use the user interface of the game
  Originator: Mohammed Mirajkar, Oct 4th, 2018

3.3 Performance Requirements

- **Requirement** #NF7:
  The time for the AI to make a move must be within AI_TIME
  Reasoning: Latency in the AI will diminish the user's enjoyment of the game.
  Criterion: Time the AI to check that a move is made within AI_TIME
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement**#NF8:
  When a user makes move, the move should be represented on the graphical othello board within $RESP\_TIME$
  Reasoning: Latency in recording the users actions will dimish the user's enjoyment of the game.
  Criterion: record the time the game needs to display the users move and compare with $RESP\_TIME$
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement** #NF9:
  When a user makes move, the animation for the move shall not take more than $ANIM\_TIME$
  Reasoning: Long game animations can also dimish the user's enjoyment of the game.
  Criterion: record the time the game takes to display an animation compare with $ANIM\_TIME$
  Originator: Mohammed Mirajkar, Oct 4th, 2018

3.4 Operational and Environmental Requirements

- **Requirement** #NF10:
  The project shall be used with a computer and a mouse or a mouse alternative.
  Reasoning: A user cannot play our game without the tools above
  Criterion: Check if the game can played on a different device such as a PS4 for example.
  Originator: Mohammed Mirajkar, Oct 4th, 2018

- **Requirement**#NF11:
  The game must be able to interface with any HTML browser
  Reasoning: The game should be able to run on any browser in order to be accesible to users
  Criterion: Check that the game runs on all valid HTML browsers
  Originator: Mohammed Mirajkar, Oct 4th, 2018

3.5 Maintainability and Support Requirements

- **Requirement** #NF12:
  The code for this project must be available to the public on github.
  Reasoning: Adding the project on to github increases the ability to maintain and monitor the system.
  Criterion: Check that code for the project is availabe to the public on github
  Originator: Mohammed Mirajkar, Oct 4th, 2018

3.6 Security Requirements

- **Requirement**# NF13:
  The game shall not access any of the user's folders or files not related to the game save state.
  Reasoning: In case of a security breach, the user should not be at risk
  Criterion: Look at game source code to see whether the game access the users folders or files that do no correspond to the game save state file.
  Originator: Mohammed Mirajkar, Oct 4th, 2018

3.7 Cultural Requirements

- **Requirement** # NF14:
  The game shall not contain any cultural imagery or text at all.
  Reasoning: Containing cultural imagery or text may offend some people by potentially appropriating or mocking their culture.
  Criterion: Play the game and identify if any cultural icons or texts exist within the game.
  Originator: Mohammed Mirajkar, Oct 4th, 2018.

3.8 Legal Requirements

- **Requirement**# NF15:
  The game must be licensed under the MIT Open License.
  Reasoning: The license is under the MIT Open License for the open source project that we based our implementation on
  Criterion: Check that the game is licensed under the MIT Open License

3.9 Health and Safety Requirements

There are no Health and Safety Requirements for this game.

4 Project Issues

4.1 Open Issues

- The AI in the previous implementation takes about 2 seconds longer than the intended execution time

- The animations for when the disks change colour also takes longer than the intended execution time
- The user interface in the previous implementation doesnt explicitly say which colour belongs to your team
- The user interface in the previous implementation doesnt have an explicit indicator for when it is the users turn
- The previous implementation does not support 2 gameplay for human players

4.2 Off-the-Shelf Solutions

Current off-the-shelf solutions include web browsers and servers to run the program and laptops, smartphones and tablets to run the web browsers.

4.3 New Problems

Some problems that arise due to the switch to a web platform include converting certain python functionalities for the specific function to work in JavaScript. Also, though a web implementation is more accessible to the general public, those without a web browser cannot play the game. There is also potential for a user to lose all their progress during a game depending on if the browser crashes.

4.4 Tasks

1. The first task is to create a JavaScript model based on the python implementation that contains functions required to perform actions and check results for Othello
2. The second task is to create a view that contains the board of the game as well as the pieces and some menu options
3. The third task is to create a controller that links the view and the model and allows for user interaction with the game
4. Testing will be done throughout the development of the project and will be used to test for different winning scenarios, move validity and

output statements to ensure that the program will be able to meet all its functional requirements

*Tasks will be evenly split among the team members and multiple members can contribute to the same task

4.5 Migration to the New Product

The previous python implementation will be heavily referenced when building the model for the web-based implementation of Othello. The AI and move validity as well as the winning scenarios will be inspired from the code in the python implementation by converting the syntax between Python and JavaScript. Once the model is finished, the view/UI will be created from scratch with little basis from the python implementation. The controller aspect will also be inspired by the python implementation with a conversion between the syntax while keeping the semantics the same with minor improvements.

4.6 Risks

As of now, there are no risks that can affect the outcome of the project.

4.7 Costs

The budget for this project is set at zero dollars. All programming will be done on free IDEs and application itself will be free to play as well.

4.8 User Documentation and Training

The website will provide a link to a user manual or a have a built in user manual to explain the rules of Othello to the user. Outside of this, not much documentation or training is required for the user as the rules of the game along with its interactions are set to be fairly simple.

4.9 Waiting Room

Current features that could potentially be added but are not required is an in-game timer that acts as a stopwatch to see how fast he/she can beat

the AI in one game. Game music can also be potentially implemented into the game provided that there is no copyright infringement that could be claimed. Additional game modes such as rapid fire (each user will only have 10 seconds to make a move) can also be implemented to add a variety of gameplay methods for the user.

4.10 Ideas for Solutions

The goal is to use a model-view-controller scheme to implement Othello as a web-based implementation. May use certain libraries for specific animations. Visual Studio will be used to create and document the code for the project.

5 Appendix

The gantt chart has been updated for the week of October 1st.

5.1 Symbolic Parameters

N/A