Checkers Game

Requirements Specification

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 | 07/24/2017 | First Draft - Created and planned out sections and subsections | 1.0 |
| Omair Bhore  Nhan Nguyen  Phuc Ngo  Bruce Zhao | 07/29/2017 | Second Draft - Added content to Section 1-3 | 1.1 |
| Omair Bhore  Nhan Nguyen  Phuc Ngo  Bruce Zhao | 07/30/2017 | Third Draft - Added content to Section 4-7 | 1.2 |
| Omair Bhore  Nhan Nguyen  Phuc Ngo  Bruce Zhao | 08/05/2017 | Fourth Draft - Added Section 5 - User Interface. | 1.3 |

Table of Contents

1. **Document Introduction**
2. Purpose
3. Scope
4. Overview
5. **Description**
6. Product Perspective
7. Product Functions
8. User Description
9. Assumptions and Dependencies
10. Requirements Apportioning
11. **Functional Requirements**
12. Server
13. Client
14. **Non-Functional Requirements**
    1. Network Performance.
    2. Web Browser Requirements.
    3. Game Rules
15. **User Interface**
    1. Front-end
16. **Use Cases**
    1. Setting a username
    2. Clicking the ‘Start’ button
    3. Entering a game
    4. Setting up a game
    5. Move a chess
    6. Ending a game
17. **Glossary**
18. **References**

**1. Document Introduction**

**1.1 Purpose**

This document provides all of the descriptions and requirements for a repository of laws being proposed . This document shall be used as a reference for current and future developers. It shall also provide users with a guide to how to use the program to its fullest.

**1.2 Scope**

The document shall provide developers with enough information to understand the technical side of this product and users with enough information to be able to play checkers.

**1.3 Overview**

This document shall contain both functional and nonfunctional requirements for the Checkers program along with use cases, diagrams, and UI mockups that can be used to elaborate on all the details.

This game contains a client interface component which shall be called the **front-end**, and a server component which shall be called the **back-end**. The requirements for each is provided and they also contain mock-ups for the user interface and their use cases.

**2. Description**

**2.1 Product Perspective**

The game of checkers is a one-on-one game designed for casual play with a minimal ruleset so people of all ages, creeds, and births can play. Each player is pitted against each the other, one side being red and the other being black, on an eight-by-eight board, not so different than a chess board. Once the game is started, the players take turns moving their pieces diagonally on the board, seeking to capture all the opponents pieces. The game is intended to reach as many people as possible on platforms connected to the internet by providing a web-based application. When the players open the game, they shall decide their username and shall then be matched with someone who has a vacant slot or if none are available, they shall begin a new session and wait for a player to join. Once they are matched, they shall be moved to a new page to play the game.

|  |
| --- |
|  |
| **Figure 1. Checkers board** |

The game is intended to be played on Google Chrome web browser on all computers. It shall be hosted on a server we control.

|  |  |
| --- | --- |
|  |  |
| **Figure 2. Sample top-down checkerboard view with sample move log on the side** | |

The game itself shall be played in a 2-D, top-down environment with the view of the checkerboard displayed in Figure 2. The checkerboard shall be an 8x8 board with alternating black and white squares. There are 12 checkers pieces on each side with one person as black and the other as red. They shall be positioned like the pieces in Figure 2 and there shall be a section for log which displays each move that has been made correctly labeled by who made the move similar to the log displayed in figure 2.

For the mobile platform, users will be able to access the game through an application developed for each specific mobile operating system. This would allow for cross platform play so the users would not be limited to one group of platforms. The mobile application would be a slimmed down version of the browser-based game. The main screen of the application would be focused on the board, allowing for zooming in for more precise moves. The move log would not appear by default but would still be contained in the application as a menu that the user could make appear.

**2.2 Product Functions**

**2.2.1 Back-end functionality:**

The back-end will have the following functionalities:

* Ability to put players into a matching queue for game matching
* Ability to match players who are in the matching queue
* Ability to generate a game session over socket for players
* Ability to support real-time communication in a game for players
* Ability to validate duplications of username.

**2.2.2 Front-end functionality:**

The front-end will have the following functionalities:

* Ability to render the website
* Ability to accept the user’s input
* Ability to connect to the back-end
* Ability to send data over socket to the back-end
* Ability to respond to events on the back-end by updating the UI elements
* Ability to validate possible moves of chess pieces

**2.3 User Descriptions**

The ideal users for Checker game are two people of any age who like to play checkers.

**2.4 Dependencies**

**2.4.1 NodeJS**

NodeJS is an open source cross-platform JavaScript environment which allows users to build network applications quickly and professionally. This project completely depends on NodeJS to build the web server. NodeJS is also the environment to set up socket networking for this project.

**2.4.2 Socket.io**

Socket.io is an open source JavaScript library for real-time web applications which enables real-time, bidirectional communication between web clients and servers. This will support the real-time UI updating when a player makes a move for the other player.

**2.4.3 ReactJS**

ReactJS is an open source JavaScript library for building user interfaces. The project will use this library to build the front-end.

**2.4.4 ReactDnD**

ReactDnD is an open source framework to support drag and drop functionality for ReactJS. It will be used to provide the chess piece drag and drop functionality in the project.

**2.4.5 Redux**

Redux is an open source JavaScript library to manage application state. This will be used to keep track of the UI state and the chess coordination and update when a player makes a move.

**2.5 Requirements Apportioning**

|  |  |
| --- | --- |
| **Priority Level** | **Description** |
| **1** | **Priority 1** rated items are required and planned to be in the final build to be production ready. |
| **2** | **Priority 2** rated items are not required to be in the final build but will be developed if time permits. The product shall be designed for the future implementations of Priority 2, so that it will be easily expanded upon. |
| **3** | **Priority 3** rated items are not required in the final build and is not a priority to be in the product itself. These are the functions that will be given priority upon the completion of Priority 1 and priority Priority 2. |

**3. Functional Requirements**

**3.1 Back-end**

**R1.1 Server**

The HTTP server is built on top of NodeJS. Express, which is a minimal and flexible web framework, is used to serve static content and set up routing. The server is ran on localhost, port 3000. Static content that are served include index.html, bundle.js, socket.io.js, socket.js, style.css. **Priority 1**.

**R1.2 Connection**

Socket.io is used to initiate socket connections between 2 players for real time data and state communications. Socket.io has built-in event handlers that listen to connection statuses (connected or disconnected), emit data, and receive emitted data. **Priority 1**.

**R1.3 Matchmaking**

The matchmaking shall place people in a matching queue that moves people up a list depending on when they entered the queue in a FIFO order. The back-end will continuously match people in the queue and if it only has one person and no one to match them with, they’ll be prioritized. **Priority 1**.

**R1.4 Username validation**

The server shall validate and notify

**R1.5 Server Flow**

|  |
| --- |
| **Screen Shot 2017-08-05 at 8.00.26 PM.png** |
| **Figure 4. Server flow** |

**3.2 Front-end**

**R2.1 Main UI layer:**

HTML, CSS, and JavaScript are used for the base of the GUI. ReactJS, a component-based JavaScript library is used to build an interactive user interface. In addition, ReactJS is compatible with NodeJS. The GUI includes a “lobby” web page, a page to play the game, and a pop-up modal for any post or pregame responses. The lobby is where the players will enter their nickname and start the matchmaking process. The gameplay page shall contain the checkerboard where the players can drag and drop the pieces to play the game. The page also has a log on the right where players can chat and find important game logs will be shown there as well. Lastly, the pop-up modal contains messages and alerts. It is used to ask the player whether to quit or continue with a new game. **Priority 1**.

**R2.2 Drag-and-Drop functionality:**

React DnD library, which is built on top of the HTML5 drag and drop API, is used to implement the drag-and-drop functionality of the checker pieces. **Priority 1**.

**R2.3 Move Validation:**

When a player drag a chess piece, the front-end shall force validation of the move the player can make. The possible locations where the chess pieces can move to shall be highlighted in transparent yellow color. The locations where the chess pieces are not allowed to move to shall be highlighted in transparent red color. **Priority 1**.

**4. Non-functional Requirement:**

**4.1 Network Performance**

**R.1 Connection Management**

The game depends on the connection of each player to the server and their latency will affect how our server reacts and responds to them. The server shall handle any timeout issues and idling users to allow for the people who are playing the game to be matched with people are actually playing the game. Priority 1.

**4.2 Web Browser Requirements**

The game is expected and tested to work on Google Chrome browser version 60.0.3112.78. Other browsers shall not be tested and are not expected to be fully functional. Priority 1.

**4.3 Game Rules**

**R.3 US Rules**

**R.3.1**

Checkers is played by two players. Each player begins the game with 12 colored discs. (Typically, one set of pieces is black and the other red.)

**R.3.2**

The board consists of 64 squares, alternating between 32 dark and 32 light squares. It is positioned so that each player has a light square on the right side corner closest to him or her.

**R.3.3**

Each player places his or her pieces on the 12 dark squares

closest to him or her.

**R.3.4**

Black moves first. Players then alternate moves.

**R.3.5**

Moves are allowed only on the dark squares, so pieces always move diagonally. Single pieces are always limited to forward moves (toward the opponent).

**R.3.6**

A piece making a non-capturing move (not involving a jump) may move only one square.

**R.3.7**

A piece making a capturing move (a jump) leaps over one of the opponent's pieces, landing in a straight diagonal line on the other side. Only one piece may be captured in a single jump; however, multiple jumps are allowed during a single turn.

**R.3.8**

When a piece is captured, it is removed from the board.

**R.3.9**

If a player is able to make a capture, there is no option -- the jump must be made. If more than one capture is available, the player is free to choose whichever he or she prefers.

**R.3.10**

When a piece reaches the furthest row from the player who controls that piece, it is crowned and becomes a king. One of the pieces which had been captured is placed on top of the king so that it is twice as high as a single piece.

**R.3.11**

Kings are limited to moving diagonally, but may move both forward and backward. (Remember that single pieces, i.e. non-kings, are always limited to forward moves.)

**R.3.12**

Kings may combine jumps in several directions -- forward and backward -- on the same turn. Single pieces may shift direction diagonally during a multiple capture turn, but must always jump forward (toward the opponent).

**R.3.13**

A player wins the game when the opponent cannot make a move. In most cases, this is because all of the opponent's pieces have been captured, but it could also be because all of his pieces are blocked in.

**4.4 Playtesting**

**R.4.1**

In this scenario two players should finish the game and a modal window should pop up giving the user two option, to “Play Again” or “End Session”.

**R.4.1.1**

When a player clicks “Play Again”, the user shall be redirected to the landing page and automatically added to the matching queue. The user shall then be matched with another player.

**R.4.1.2**

When a player clicks “End Session”, the user shall be redirected to the landing page. That user can then enter a name and start a new game or leave by closing the browsers tab or window.

**R.4.2**

In this scenario one of the users moves away from the computer for an extended period of time. When the program senses inactivity, a 60 second timer will begin.

**R.4.2.1**

If the user returns to the program before the end of the 60 second timer ends, then the game shall resume as normal.

**R.4.2.2**

If the user does not return before the timer ends then the inactive user will lose the game and shall be redirected to the landing page where he or she shall be subject to the 180 second connection timeout timer. If the user returns before the end of the 180 seconds then the user could start a new game or close the browser tab or window. On the other hand, the active user will win the game will get the end of game modal window from requirement R.4.1

**5. User Interface**

**5.1 Front-end**

|  |
| --- |
| **Screen Shot 2017-08-05 at 8.44.31 PM.png** |
| **Figure 3. Front-end flow** |

The front-end will contain a page with the menu for choosing a username, starting the game, and About page of the game. Also, it has Gameplay page to let the players play the checker game. A popup modal shall appear after a game ends for players to navigate. Figure 3 shows the navigating flows of the front-end. Below is the requirements for each screen.

5.1.1 Lobby page

The lobby page will have the game title, an ‘About’ link to navigate to the ‘About’ page, an input box to set the username and a ‘Start’ button to start the game.

R5.1 Render the game title. **Priority 1**.

R5.2 Game title is ‘Checkers Game’. **Priority 1**.

R5.2 Render the ‘About’ link. **Priority 2**.

R5.3 ‘About’ link shall navigate to ‘About’ page. **Priority 2**.

R5.3 Render the username input box. **Priority 1**.

R5.4 Input box shall accept user’s input. **Priority 1**.

R5.4 Render the ‘Start’ button. **Priority 1**.

R5.5 ‘Start’ button shall start a game. **Priority 1**.

5.1.2 About page

The ‘About’ page has version and patch notes, credits, game tutorial link, and ‘Back to lobby’ button.

R5.6 Render the version and patch notes section. **Priority 2**.

R5.7 Version and patch notes section displays game version number and latest patch changelog. **Priority 2**.

R5.8 Render the credits section. **Priority 2**.

R5.9 Credits section displays the team members and their contribution. **Priority 2**.

R5.10 Display game tutorial link. **Priority 3**.

R5.11 Render ‘Back to lobby’ button. **Priority 2**.

R5.12 ‘Back to lobby’ button navigate back to Lobby Page. **Priority 2**.

5.1.3 Gameplay page

The Gameplay page has the chessboard with chess pieces, a move log section, a ‘Give up’ button, and a popup modal after a game ends.

R5.13 Render the chessboard with chess pieces. **Priority 1**.

R5.14 Chess piece shall have drag and drop functionality**. Priority 1**.

R5.15 When the chess piece is being dragged, the places that the chess piece can move to shall be highlighted in yellow transparent color. **Priority 1**.

R5.16 When the chess piece is being dragged, the places that the chess piece cannot move to, in all moves that it can make, shall be highlighted in red transparent color. **Priority 1**.

R5.17 Players shall move the chess pieces using the drag and drop functionality. **Priority 1**.

R5.18 Render move log section. **Priority 1**.

R5.19 Move log shall write the turns, the move the players make. **Priority 1**.

R5.20 A chess piece shall disappear when it gets discard by an opponent chess piece. **Priority 1**.

R5.21 A chess piece shall change its appearance when it becomes a king. **Priority 1**.

R5.22 Render the ‘Give up’ button. **Priority 2**.

R5.23 The ‘Give up’ button should end the game. **Priority 2**.

R5.23 After a game ends, a popup modal appear to navigate the players back to the Lobby page. **Priority 2**.

5.1.4 Popup Modal:

The popup modal has two buttons: ‘Return to lobby’ and ‘Find a new match’.

R5.24 Render the ‘Return to lobby’ button. **Priority 2**.

R5.25 ‘Return to lobby’ shall navigate the players to ‘Lobby page’. **Priority 2**.

R5.26 Render the ‘Find a new match’ button. **Priority 2**.

R5.27 ‘Find a new match’ shall put the players into the matching queue and navigate the players to ‘Lobby page’. Priority 2.

**6. Use Cases**

**6.1 Setting a username:**

Player enters a name to set as username.

**Precondition:** The username has not been taken

**Action:** Player enters a name

**Postcondition:** The username will become unavailable

**6.2 Clicking the ‘Start’ button:**

Player clicks the ‘Start’ button to start the match making process

**Precondition:** A valid username has been entered

**Action:** Player clicks the ‘Start’ button

**Postcondition:** The UI will update and ask the player to wait. The player will be put in the matching queue

**6.3 Entering a game:**

Player waits until get matched with another player.

**Precondition:** The player is in the matching queue

**Action:** The player get redirect to the game room page

**Postcondition:** The player is in the game room page

**6.4 Setting up a game:**

Players wait when all players have successfully connected to the game room

**Precondition:** 2 players have been redirected to the room

**Action:** Wait for all players to successfully connect to the back-end

**Postcondition:** Randomly select which players to go first

**6.5 Move a chess:**

Player drag a chess piece to a new location

**Precondition:** There is a valid place for a chess to move

**Action:** The player drag the chess to the desirable place

**Postcondition:** The UI update the new location of the chess on both players side.

**6.6 Ending a game:**

A player loses, idle or disconnect over the 60 seconds timeout time.

**Precondition:** 2 players are playing a game

**Action:** One loses or idle or disconnect

**Precondition:** A modal appears prompting each players to go back to lobby or go into matching queue

**7. Glossary**

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

**CSS -** Cascading Style Sheet, which describes how the associated HTML document(s) look and feel, abbreviation

**FIFO -** First in First out, abbreviation

**GUI -** Graphical User Interface, abbreviation

**HTML**- HyperText Markup Language, standard language for creating web pages, abbreviation

**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 which 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.

**ReactDnD:** React Drag and Drop abbreviation

**8. References**

<http://www.kidskubby.com/wp-content/uploads/2012/09/free-online-checkers-game.jpg>

<https://www.doublegames.com/images/screenshots/checkers_3_big.jpg>

<https://www.thespruce.com/play-checkers-using-standard-rules-409287>

<http://pokemonshowdown.com/>