Checkers

Requirements Specification

Version 1.2

March 04, 2018

Group Members	Chetan K Parakh, Cody Moser, Nicholas Nugent, Saugat Sthapit
Submitted To	Dr. Filippos Vokolos, Ph. D.

Table of Contents

1. Introduction

1.1 Purpose

Here, you will find the detailed specification on the product we are building. The game of Checkers will be multiplayer so this document will be useful in finding the detailed specification and our approach to successfully create this game. In this Software Requirement Specification (SRS) document, we will go through the scope, functional, non functional requirements and the use cases briefly.

1.2 Scope

This document will contain enough information to allow a developer to easily translate requirements within the document into code without any ambiguity.

1.3 Overview

This document will contain a description of our implementation of the Checkers game. It will contain the description of the product and the functional and nonfunctional requirements. It also contains diagrams and specifications of the UI along with how they are expected to work.

2. Description

2.1 Product Perspective

We will be creating a version of the 2-person board game checkers, which will allow people to play each other interactively from remote locations. We will be using the tournament rules from The American Checker Federation.

2.1.1 Client

The client will give the user an intuitive user interface to find a game against another player, and then play out that game interactively against their opponent. It will display a checkerboard of the current game state, show which

player's turn it is to move, and update the board as either as each player makes their moves.

2.1.2 Node.js Server

Our server will be written in NodeJS and will serve the role of letting users connect to the server, pairing up players waiting to play, and sending the moves in between players mid game.

2.2 Product Functions

2.2.1 Client Functionality

The client will have the following functionality:

- Allow user to request the server to start a game against another player
- Allow user to make moves and receive moves for the current game
- Validate moves attempted by the user
- Allow user to exit the game when they wish

2.2.2 Server Functionality

The Server will have the following functionality:

- Receive two connections from clients and start a game
- Apply moves sent from clients and send the new state to the clients
- Notify the clients if a user quits the game or drops connection

2.3 User Description

The ideal users for our checkers game would be 2 people playing each other while they are in separate locations.

2.4 Assumptions and Dependencies

2.4.1 Node.js 8.9.4

Our server will be written using Node.js. Node.js is an open source server framework that uses JavaScript on the server. It uses an event driven, non-blocking Input/Output model that makes it able to handle many requests at the same time. Our server will be a lightweight server that users can connect to and play the game through.

2.4.2 HTML5, JavaScript, and CSS

We will be using a basic front end using standard HTML, JavaScript, and CSS. This will be used for displaying everything to the user, allowing the to interact with the game board, and sending requests to the server.

2.4.3 Github

Our team will use Github for our version control.

2.4.4 Amazon Web Services

Our server will be hosted on an EC2 instance of Amazon Web Service (AWS).

2.4.5 Mocha

Mocha will be used as our JavaScript test automation framework. It will be used for writing our tests as well as calculating our test coverage.

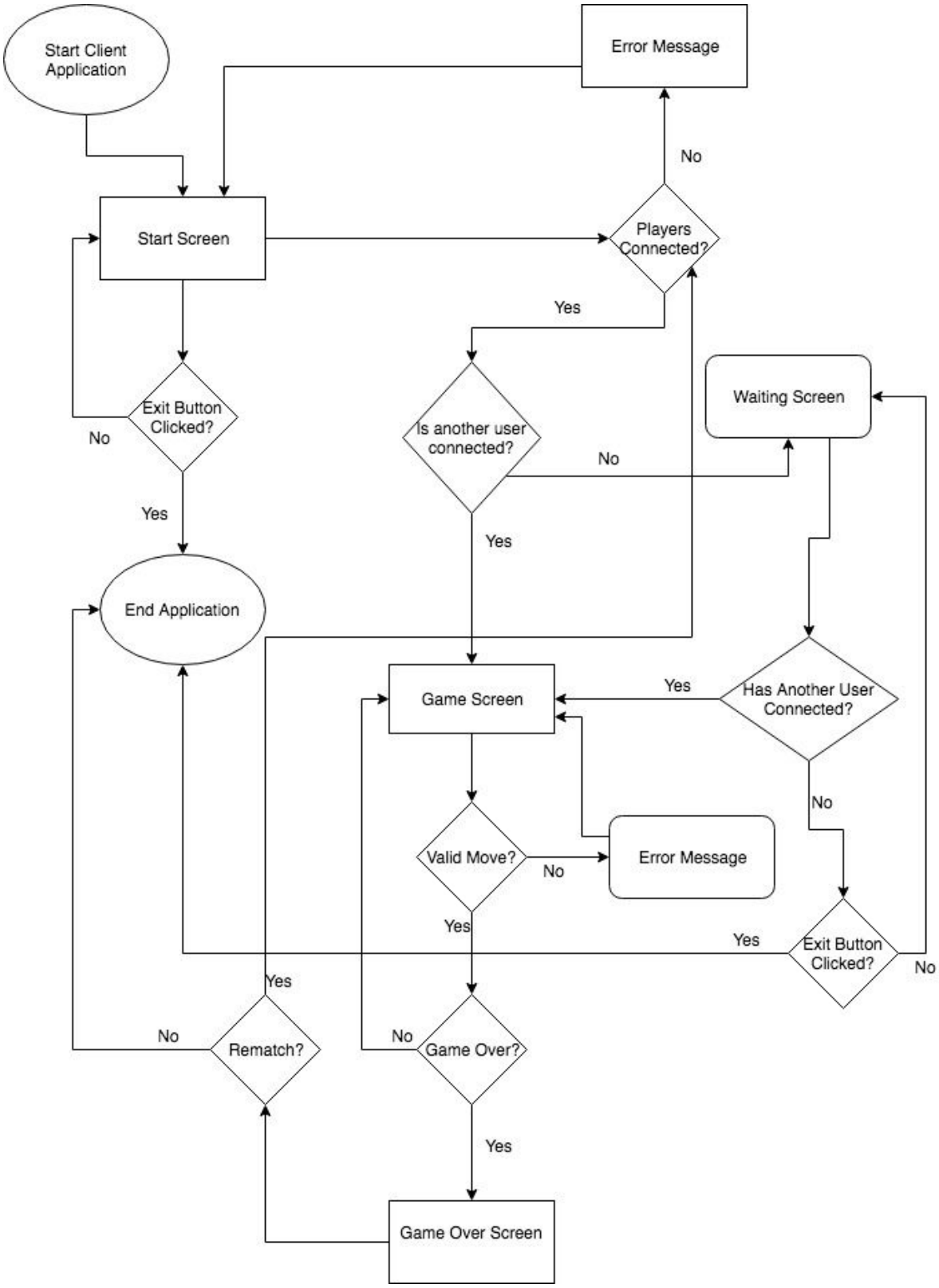
2.4.6 Sentry

Sentry will be used in both server side and client side for monitoring and fixing/ tracking errors in real time

2.5 Requirements Apportioning

Priority Level	Description
1	**Priority 1** requirements are essential to the product and must be in the final build. These requirements must be tested and verified to ensure proper functionality.
2	**Priority 2** requirements are not required for the final build, but will be provided if there is sufficient time. The system will be designed such that it is extendable to easily incorporate these requirements at a later time.
3	**Priority 3** requirements are not required, and will not be considered in the design of the system. If sufficient time remains the requirement will be incorporated.

3. User Interface Requirements



3.1. Start screen

It contains:
R1 Button to Connect **(Priority 1)**
R2 Exit Button **(Priority 1)**

3.2. Wait screen

It contains:
R3 Message showing the status of connection **(Priority 1)**
R4 Exit Button **(Priority 1)**

3.3. Game Screen

It contains:
R5 Graphical representation of board **(Priority 1)**
R6 Pieces for each player with different colors **(Priority 1)**
R7 A flag to indicate whose turn it is **(Priority 1)**
R8 Sequence of chosen moves highlighted on board **(Priority 2)**
R9 Submit move button **(Priority 1)**
R10 Clear move button **(Priority 1)**
R11 Timer **(Priority 2)**
R12 Piece counter **(Priority 2)**
R13 Resign game button **(Priority 2)**

3.4. Game Over screen

It contains:
R14 Message display **(Priority 1)**
R15 Rematch button **(Priority 2)**
R16 Exit button **(Priority 1)**

4. Functional Requirements

4.1. Client

4.1.1. Starting and connecting

R17 Users must connect to our server through a web browser.

R18 Upon successful connection to the server the client will present the user with the start page. (**Priority 1)**

R19 If there are no players currently connected the start page will display a Start Game button. (**Priority 2)**

R18 If there is a player connected and waiting the start page will display a Join Game button. (**Priority 2)**

R19 If the user clicks the Start button the client will redirect the user to a waiting page. (**Priority 1)**

R20 If the user clicks the Join button the client will redirect both users to the game page. (**Priority 1)**

R21 If two users are already playing on the server the start page will inform a third user they must wait to play. (**Priority 1)**

4.1.2 Waiting for opponent

R22 Users will wait on the waiting page until a second player is ready to play. (**Priority 1)**

R23 If a user clicks the cancel button they will be removed from waiting. (**Priority 2)**

4.1.3 Playing the game

R24 The game page will display which users turn it is. (**Priority 1)**

R25 If the user clicks the forfeit button they will lose the game and be shown the game over screen. (**Priority 2)**

R26 If it is not the users turn all buttons except the forfeit button will be disabled. (**Priority 1)**

R27 If it is the users turn:

- The user can select a piece they control to move. (**Priority 1)**
- The user can select a space to move this piece to. (**Priority 1)**
- If the space the user selected is not a legal move they will be presented with an error indication. (**Priority 1)**
- If the user selects an illegal move the move will not be executed. **(Priority 1)**
- If the user selects a legal move the piece will be moved. (**Priority 1)**
- The user can finalize their move and send it to the server by clicking the end turn button. (**Priority 1)**
- If the user has not made a legal move they will not be able to select the end turn button. (**Priority 1)**
- If the user is eligible for a second move because of a capture they will not be able to select the end turn button. (**Priority 1)**

R28 If it is not the users turn the game page will be updated once their opponent has made a move. (**Priority 1)**

4.1.4 Ending the game

R29 Once the client detects a game ending move both clients will be redirected to a game over page. (**Priority 1)**

R30 The game over page will display if the game ended in a draw, a win, or a loss for each player. (**Priority 1)**

4.2. Server

4.2.1 Accepting connections

R31 The server will accept connections from clients that query its endpoint. **(Priority 1)**

R32 The server will send the status of the player queue, empty, 1 player, full, to the client. (**Priority 1)**

R33 The server will track connected players during a session with an identifier. **(Priority 1)**

R34 The server will only allow two players to join a game at once. (**Priority 1)**

4.2.2 Propagating game data

The server will track the status of the game:

R35 The server will track which players turn it is. (**Priority 1)**

R36 The server will track if the game is waiting, in progress, or over. (**Priority 1)**

R37 The server will accept moves from players whose turn it is. (**Priority 1)**

R38 The server will either send or allow to be queried:

- The players move. (**Priority 1)**
- Which players turn it is. (**Priority 1)**
- If the game is waiting, in progress, or over. (**Priority 1)**
- The player queue. (**Priority 1)**

5. Non-Functional Requirements

5.1. Network Performance

5.1.1. Lag Management

R39 The move made by one player must be immediately visible to both the players. The lag should not affect other functionality of the game including time remaining to make a move. (**Priority 1)**

5.1.2. Server Disconnect

R40 In case of player's machine disconnecting from the server. The game will end and application will declare the other player as the winner. (**Priority 1)**

5.2. Browser Support

R41 The web app is expected to work on the latest version of Google Chrome, Firefox, Safari, Edge and Internet Explorer as on 01/01/2018. (**Priority 1)**

5.3. Deployment/ Hosting

R42 The server side application and the client side web app will be hosted on EC2 instance of Amazon Web Service (AWS). (**Priority 1)**

5.4. Play Testing

R43 After addition of each new feature to the project, the code will be tested. The game will be tested by 4 player trying to play at the same time from 4 different browsers. This test will let us ensure the quality and performance of the game and also to find bugs if present. After the testing of the game, the players would be asked to fill out a feedback form to help us find issues and to fix it. (**Priority 2)**

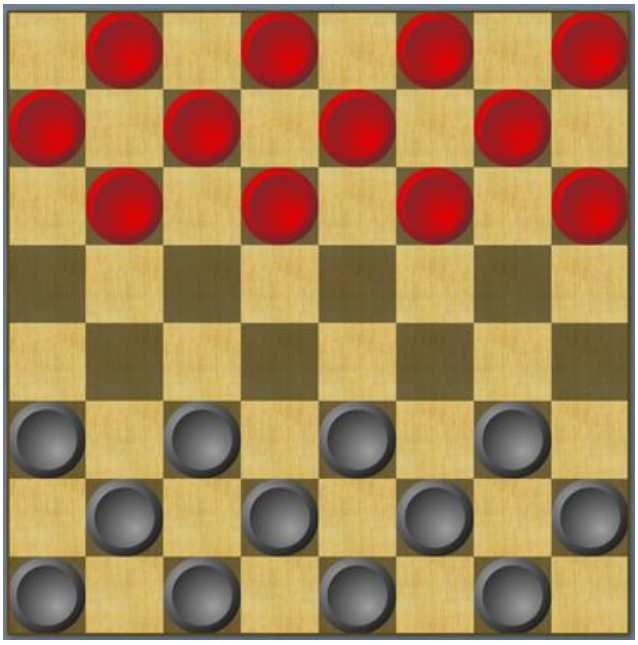
6. Rules [1]

6.1. The DraughtBoard and Its Arrangement

- The draughtboard is composed of 64 squares, alternately light and dark arranged in a square array of 8 rows and 8 columns and bounded by a neutral border.
- The official draughtboard for use in all major events shall be of Green and White (or off white/cream) colours for the dark and light squares.
- At the commencement of play the draughtboard is placed between the players in such a way that a green square is to be found on the player's near left side, and a white square to their right side. The playing squares to the near left side of the

draughtboard is referred to as a player's "Single Corner", while the playing squares on the near right side is referred to as a player's "Double Corner" side.

- The 32 green squares used for play on the draughtboard shall be assigned numbers 1-32 for descriptive purposes. These numbers are the official reference system for notations and recording games.



6.2. Order of Play

One of the two colors (red/ white) is randomly assigned to one of the two players. The first move in each game is made by the player with the Red men

6.3. Moves

There are fundamentally 4 types of move: the ordinary move of a man, the ordinary move of a king, the capturing move of a man and the capturing move of a king.

6.4. Ordinary Move Of A Man

An ordinary move of a man is its transfer diagonally forward left or right from one square to an immediately neighbouring vacant square. When a man reaches the

farthest row forward (known as the “king-row” or “crown-head”) it becomes a king, and this completes the turn of play. The man can be crowned by either player ([i]) by placing a man of the same colour on top of it before the next move is made. (It may be necessary to borrow from another set if no captured man is available for the purpose).

6.5. Ordinary Move Of A King

An ordinary move of a king (crowned man) is from one square diagonally forward or backward, left or right, to an immediately neighbouring vacant square.

6.6. Capturing Move Of A Man

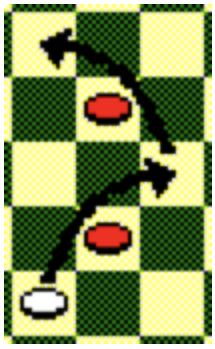
A capturing move of a man is its transfer from one square over a diagonally adjacent and forward square occupied by an opponent's piece (man or king) and on to a vacant square immediately beyond it. (A capturing move is called a "jump"). On completion of the jump the captured piece is removed from the board.

6.7. Capturing In General

If a jump creates an immediate further capturing opportunity, then the capturing move of the piece (man or king) is continued until all the jumps are completed. The only exception is that if a man reaches the king-row by means of a capturing move it then becomes a king but may not make any further jumps until their opponent has moved. At the end of the capturing sequence, all captured pieces are removed from the board.

All capturing moves are compulsory, whether offered actively or passively. If there are two or more ways to jump, a player may select any one that they wish, not necessarily that which gains the most pieces. Once started, a multiple jump must be

carried through to completion. A man can only be jumped once during a multiple jumping sequence



6.8. Capturing Move Of A King

A capturing move of a king is similar to that of a man, but may be in a forward or backward direction.

6.9. Result of the Game

There are only two possible states to define: the win and the draw.

6.10. Definition of a Win

The game is won by the player who can make the last move; that is, no move is available to the opponent on their turn to play, either because all their pieces have been captured or their remaining pieces are all blocked. A player also wins if their opponent:

- Resigns at any point.
- Forfeits the game.
- Fails to reach the time control when using clocks

6.11. Definition of Draw

- Neither player has advanced an uncrowned man towards the king-row during their own previous 40 moves.

- No pieces have been removed from the board during their own previous 40 moves.

7. Use Cases

7.1 Use Case Flow

7.1.1. Starting the game

Player will visit the URL and click on Start Game.

7.1.2. Making move

Precondition: It is players x turn

Action: Player x selects a piece to move then selects the box to place the piece

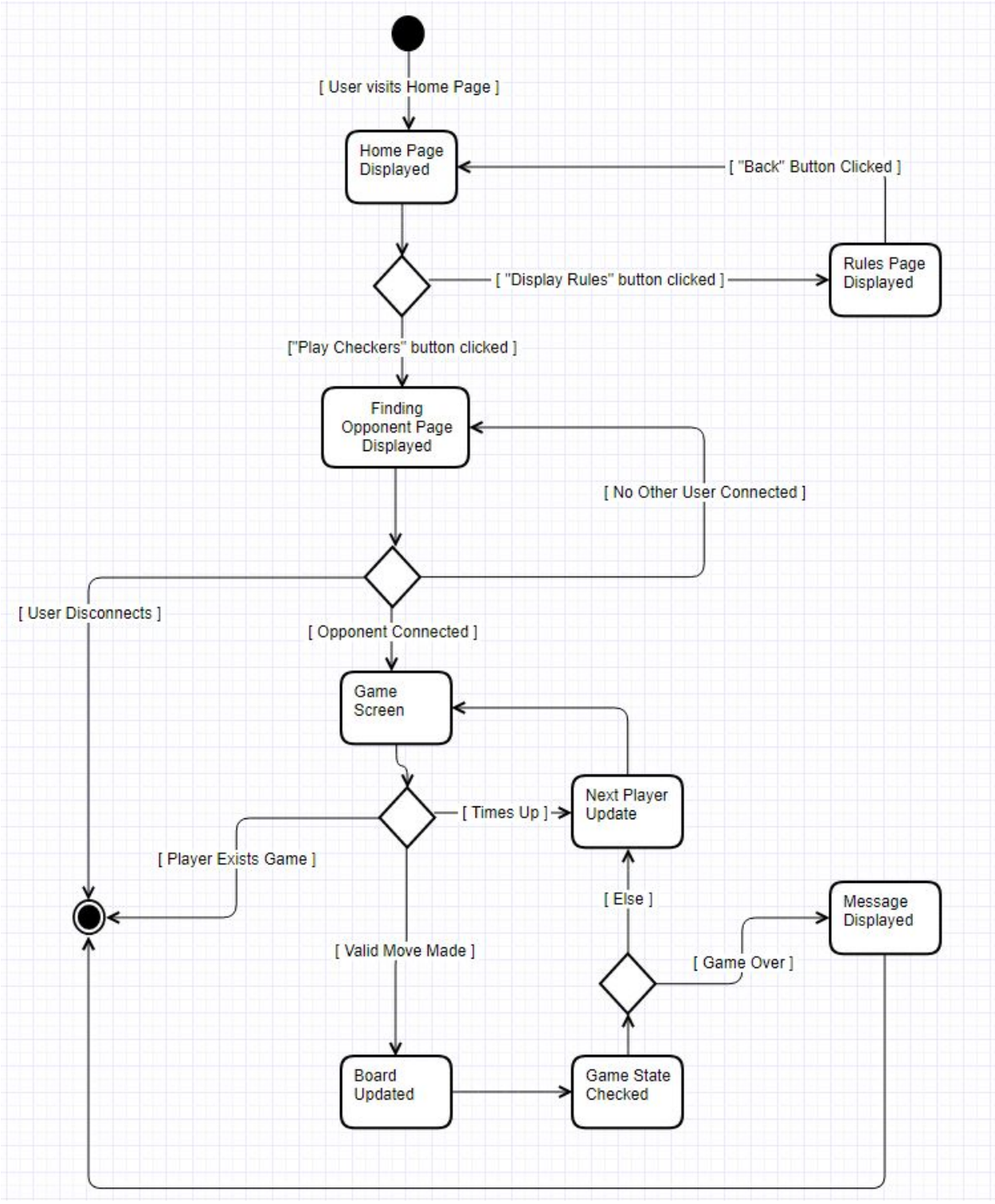
Postcondition: Piece is hidden from the old box and appears in the new box.

7.1.3. Ending the game

The game ends if any one or more of the following criteria is met:

- One of the player wins.
- The game comes to a draw.
- Any or both of the player gets disconnected.
- Player leaves the game.

7.2. Activity Diagram



The diagram shows activity case flow of the web application. The final state of the diagram refers to the termination of client side program.

8. References

[1] The American Checker Federation