SOFTWARE REQUIREMENTS SPECIFICATION

for

Checkers

Version 1.0.4

Prepared by:
Adam Luong
Benny Mai
Dakota Wessel
Jacky Zheng
Tony Zhu

Group: 10

October 18, 2020

Contents

Revision History

Name	Date	Reason For Changes	Version
1.0.0	15–10–20	Initial Setup	Dakota Wessel
1.0.1	18–10–20	Intro. and Use Cases	Benny Mai
1.0.2	18–10–20	Functional Requirements	Jacky Zheng
1.0.3	18–10–20	Nonfunctional Requirements	Adam Luong
1.0.4	18–10–20	Overall Description & User Interface	Tony Zhu

1 Introduction

1.1 Purpose of Document

The goal of this document is to outline the requirement specifications of our web-based checkers game. This application will allow two users to connect and interact remotely, allowing them to play and chat. This document will cover the scope, objective, basic requirements, and goals for this application. After the application's high-level look, this document will dive deeper into topics like functional, non-functional, user interface, design, test cases, program usage, and references. This document will clearly explain to an engineer or end-user the overall implementation and goals of our application.

1.2 Project Scope

The main objective for documentation is to educate the reader about our Checkers application, its functionality, the technologies used, and outline the application requirements.

1.3 Overview of Document

The documentation will provide a clear explanation about which technologies we used, how we implemented them, and why we chose to use them. It will outline each component of our application. The flow starts with the functional requirements, non-functional requirements, user interface, and finished with lobbies, gameplay, and winning conditions, concluding with our references.

1.4 Background

1.4.1 History

Throughout history, the game Checkers has been around, so the exact date for Checkers' invention is unknown. One of the earliest records of the game dates back to 3000 B.C in what is present-day Iraq. Later in Egypt, in 1400 B.C, the game was played using a 5 x 5 board [2]. However, the version of Checkers that we know of today was established in the mid-1500s by an English mathematician. Now the board game of checkers is cemented as one of the most popular board games of all time.

1.4.2 Game Rules

The rules provided are from the American Checker Federation [1].

1. Red always makes the first move.
2. A player can forfeit at any time; as a result, the opponent wins.

1.4.3 Moves

1. A player may only move their own pieces.
2. Normal Piece

   A normal piece may only move toward the other player's side of the board.

   A normal piece may move diagonally to the left or right to a vacant square in front of it.

   A normal piece may capture on the diagonal if there exists a vacant square one more diagonal position ahead.

   A piece may move again if there exists another piece to capture after making a capture.
3. A King Piece moves the same as a normal piece but can move and capture backward.
4. A King Piece may capture forward or backward.
5. If a normal piece reaches the opposite edge of the board, it becomes a King Piece.

1.4.4 Win Condition

1. When one player has no more pieces to move, the other player is the winner.
2. If a player forfeits the match, the other player is conceded the winner.

1.5 Abstract

Our goal is to create a document that will help a team member create an application, both client-side and server-side, that allows remote users to play checkers. The construction of this application will host one game of checkers to two users.

2 Overall Description

2.1 Product Perspective

Checkers is a game meant to be played by two players on a 8x8 checkerboard with 12 dark pieces and 12 light pieces on opposing sides of the board. The pieces are placed on the dark checkered squares on the first 3 rows of each player's respective side. The objective of the game is to place the opposing player's pieces in a position where the opposing player can no longer make any moves.

The game is meant to be run on 2 computers/web browsers on a website(server). Each player will be able to connect to the server and have the board displayed to them on their web browser, each player's moves will be recorded by the server and then be sent to each player's browsers to accurately display the current board state.

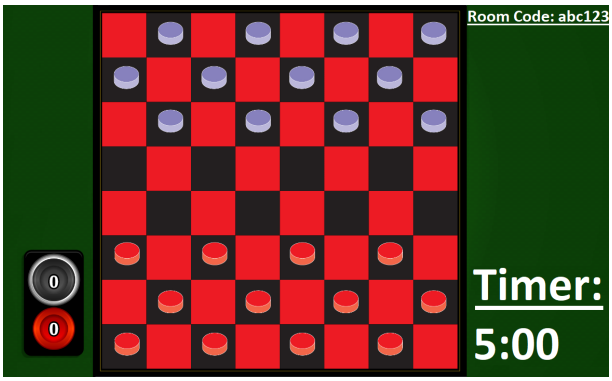

Figure 1: Mockup of the main game screen, board used from reference 3

The game will display a top down view of the 2-D board as well as a timer and room code as seen in Figure 1. The room code will be used to match players with one another. Likewise the players will interact with the pieces displayed on the board with their mouse to perform moves.

2.1.1 Web Browser/Computer Interface

The users will only be interfacing with the game through a web browser. The browser will contain a menu to either create a game or join a game through a code. Likewise, it will also allow the players to move the pieces and also display the current board and time remaining per turn for to the users.

2.2 Product Functions

2.2.1. Server Functionality:

1. The Server will perform the following:

- Remains running to allow users to connect to one another through web sessions
- The server will mediates gameplay, game sessions, and client interactions.

2.2.2. Web Browser/Client Functionality:

1. The Client will perform the following:
    - Ability to create new game session with unique code
    - Ability to connect to game sessions with unique code through server
    - Update board based on data from server

2.3 User Description

The ideal users for Checkers would be 2 players

2.4 Assumptions and Dependencies

2.4.1. SQLite

- The server will be using a SQLite database to hold all the data used to control the game of Checkers. Therefore, if SQLite were to cease support or no longer be available the team would have to restart development with a completely new database in order to proceed, which is not immpossible but will take significant amounts of time. The team assumes that SQLite will continue to work as intended.

2.4.2. Webserver

- The team will be hosting the game on a webserver, as such if said webserver were to crash or become unavailable, the team would have to rehost the game on a new webserver. The team assumes that the webserver will stay online as intended.

2.5 Requirements Apportioning

Priority Level	Description
1	**Priority 1** requirements are essential to the project and must be implemented and tested in the final version.
2	**Priority 2** requirements are not essential and can be delayed in the case that Priority 1 requirements have yet to be development. These will be added once the team finishes Priority 1 development. The team will ensure that during development that these requirements can be easily incorporated and tested.

3 Functional Requirements

3.1 Client

R1. Client - Server Interaction

- R1.1. Clients will be able to request a new game session from the server and be given a unique ID from the server. **Priority 1**
- R1.2. Clients will be able to connect to an existing game session by providing an unique ID to the server. **Priority 1**
- R1.3. Clients will not be able to connect to a game session with an incorrect unique ID. **Priority 1**
- R1.4. Clients will be able to send moves to the server for validation if it is their turn. **Priority 1**
- R1.5. Clients will be able to pause a game session given approval from both clients. **Priority 2**
- R1.6. Clients will be able to leave from the game session without consequence should the game be paused. **Priority 2**
- R1.7. Clients will be able to leave from the game session regardless of game state other than paused, but will immediately concede the game. **Priority 2**

3.1.1 Board State

R2. Board

- R2.1. The clients will have a copy of the board for rendering purposes. **Priority 1**
- R2.2. The board will update upon receiving a server update. **Priority 1**

3.2 Server

R3. Server status

R3.1. Server should be able to be run constantly without crashing. **Priority 1**

R3.2. Server will have a heartbeat function that will send an email to the developers if the server is down. **Priority 2**

R4. Server - Client Interaction

R4.1. Server will keep track of active game sessions and active client connections. **Priority 1**

R4.2. Server will keep track of time clients have spent on each move. **Priority 2**

Should a client go over specified time limit, initiate game loss state for that client.

R4.3. Server will validate moves before sending updated move to clients. **Priority 1**

On invalid move, signal client that move was invalid and to try again.

R4.4. Server will validate win conditions and notify clients with win condition. **Priority 2**

4 Non-Functional Requirements

4.1 Network Performance

N1. Lag Management

N1.1. Lag will be based off how stable the network connection of the computer the user is on. If network connection is table, there should be little to no lag. It should not negatively affect the game. Lag management will also be tested during the playtesting phase to ensure game quality.

4.2 Host Operating System Requirements

S1. Server

Node.js to allow the client and the server to communicate through endpoints

SQLite is used to store the generated key identifiers so that players can use that keycode to enter a room

S2. Client

Support Desktop Browsers

Google Chrome (latest stable version)

Firefox (latest stable version)

Microsoft Edge (latest stable version)

Microsoft Internet Explorer 11

4.3 Accessibility

N1. The application will be accessible through URL on the web browser

4.4 Playtesting

N1. After the prototype is completed, the game will then undergo playtesting with approximately 8 people of any age, 4 being familiar with Checkers and 4 with the purpose of attempting to break the game. This way, it will test all aspects of the rules of Checkers and see if the game will run smoothly and correctly. It will also see if any moves that are not in the regulations of Checkers are being permitted within the game. Also it will see if players face little difficulty with the controls and understanding of our prototype. The game will be playtested near the end of Drexel Fall Quarter of 2020. After each playtesting session, the participants will fill out a form that helps answer the findings that we are testing for. The forms will be reviewed by the team.

5 User Interface

5.1 Menus

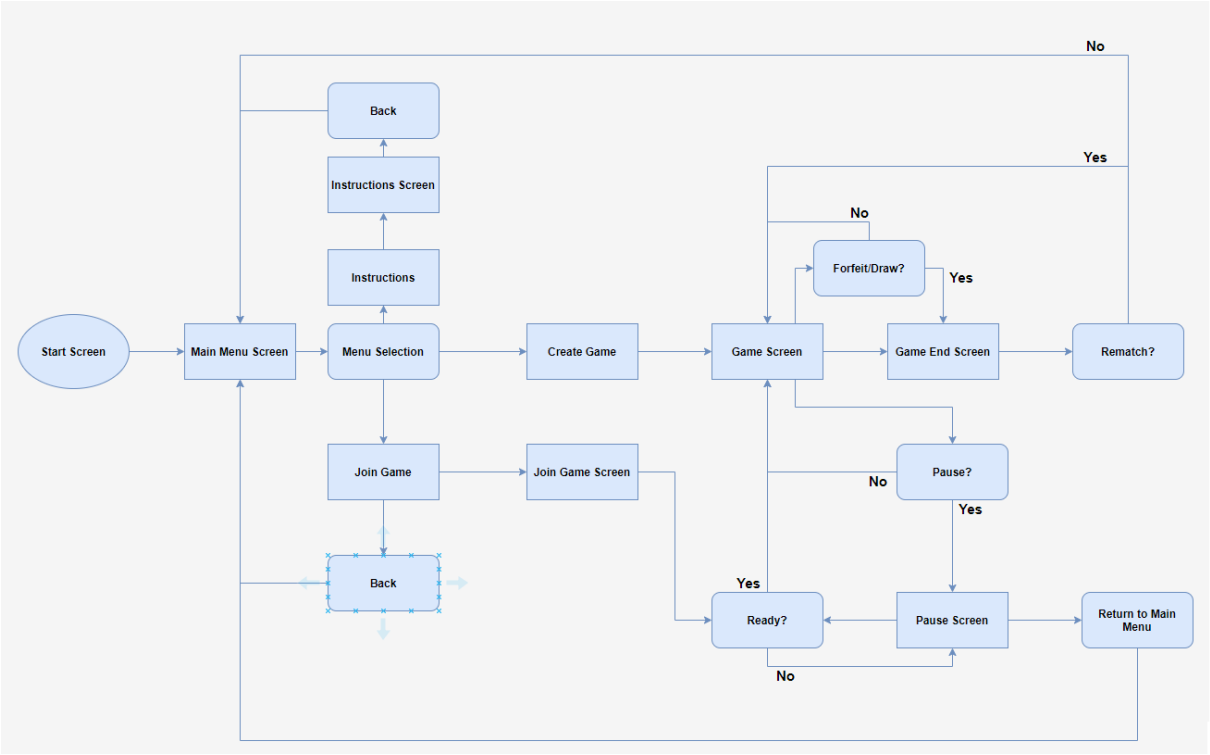

Figure 2: Navigating menu screens on the client

1. Main Menu Screen - Contains 3 Buttons: Instructions, Create Game, Join Game
    - Create Game button: Generates room code, creates game instance and displays Game Screen. **Priority 1**
    - Join Game button: Navigates to Join Game Screen. **Priority 1**
    - Instructions button: Navigates to Instructions screen. **Priority 2**
2. Game Screen - Contains Checker Board with pieces, a player turn indicator (displays whose turn it is), Room Code, and a Timer (See Figure 1) along with 3 buttons: Draw, Forfeit, and Pause
    - Room Code: Shows current room code - used by Player 2 to connect to game session. **Priority 1**

- Once game end condition is met (a player cannot make any more moves), Game End Screen will display **Priority 1**
- Players will control their pieces movements by interacting with the board using their mouse. **Priority 1**
- Turn indicator: Will display which players turn it is. Updates when a move on the board has been made by indicated player **Priority 1**
- Timer: Shows amount of time left for current player to make a move, if timer ends and no move is made then the current player will automatically lose and Game End Screen will display. **Priority 2**
- Draw button: Sends draw request to other player. **Priority 2**
- Forfeit button: Ends game and declares other player as winner. Navigates to Game End Screen. **Priority 2**
- Pause button: Sends pause request to other player. **Priority 2**

3. Game End Screen - Declares winner and a Rematch text that has 2 buttons under it: Yes or No
    - Winner will be declared on this screen **Priority 1**
    - Yes button: Will display under "Rematch" text and navigate to Game Screen with new room code if selected. **Priority 2**
    - No button: Will display under "Rematch" text and navigate to Main Menu Screen if selected. **Priority 2**
4. Instructions Screen - Contains game instructions and a back button
    - Displays instructions on how to connect or create game. As well as rules for the game Checkers. **Priority 2**
    - Back button: When clicked will return user back to Main Menu Screen. **Priority 2**
5. Join Game Screen - Contains text box input that accepts a room code and a ready button
    - Room code text box will display and allow for alphanumeric input **Priority 1**
    - Ready button: When clicked, will validate inputted room code and navigate to Game Screen if the code is valid **Priority 1**
6. Pause Screen: Will contain 3 buttons: Ready Player 1, Ready Player 2, Return to Main Menu
    - Ready Player 1 button: When clicked current player will be set as Player 1 and button unselectable from other players screen. When other player has clicked Ready Player 2 button both screens will navigate to the Game Screen. **Priority 2**

- Ready Player 2 button: When clicked current player will be set as Player 2 and button unselectable from other players screen. When other player has clicked Ready Player 1 button both screens will navigate to the Game Screen. **Priority 2**
- Return to Main Menu Butoon: When clicked the player will be taken to the Main Menu Screen. **Priority 2**

6 Standard Components

- Buttons: used for menu interations and navigation.
- Pieces: The pieces will be the basic playing pieces within gameplay.
- King Pieces: These pieces will behave identically to the regular pieces, but with additional movement options according to the rules of Checkers.
- Checkers Board: an 8x8 grid of alternating black and red tiles on which all pieces will be displayed in the gameplay.

6.1 Program Usage

6.1.1 Gameplay

G1. The match begins with an 8x8 grid with each tile alternating between black and red in color.

G2. One player is assigned the black pieces and the other red.

G3. Each player's pieces start on opposite ends of the board, occupying every other space within the first three rows (for a total of 12 pieces for each player).

See attached image for an example checkers setup:

G4. The black player starts the match by taking their turn.

G5. During each turn, the active player selects a piece of their own to move according to the following rules:

- If the piece is bordering an enemy piece and there is a free space on the other side of that enemy piece, the piece must "jump" to the empty space, removing the enemy piece it moved over from the board

  Multiple jumps can be made in a single turn if the piece is in position to jump an additional enemy piece after completing a jump
- If the piece cannot jump, it may move diagonally one space to an unoccupied space

  If the piece is a non-king, it must move forward

  If the piece is a king, it can move in any direction

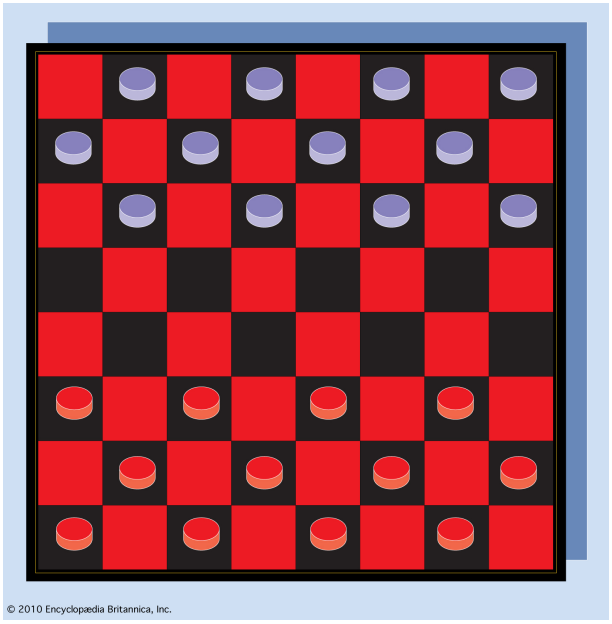

Figure 3: Example board, see reference 3

- A piece cannot jump over pieces of the same color as itself (friendly pieces)
- Two pieces cannot occupy the same space, regardless of color

G6. When a non-king piece has reached the edge of the board opposite its color's starting side, that piece will be crowned and turned into a king, allowing it to move in any direction.

6.1.2 Win Conditions

1. If player A has no more pieces, player B is the winner and vice versa.
2. If player A disconnects, player B is the winner and vice versa.

References

[1] The American Checker Foundation, *USA Checkers*, https://www.usacheckers.com/, 2019.

[2] W.J. Rayment, *History of Checkers or Draughts*, http://www.indepthinfo.com/checkers/history.shtml, 2004.

[3] Encyclopædia Britannica, *Checkers*, https://www.britannica.com/topic/checkers, 2018