Remote_Checkers Requirements Specification

Christopher Macco
Khang Nghe
Zakary Olyarnik
Dana Thompson

Revision History

Member	Date	Change
Olyarnik	1/22/17	Completed outline of all sections (r0.1.0)
Olyarnik	1/28/17	First draft of Use Cases and Glossary sections (r0.2.0)
Thompson	2/8/17	First draft of Introduction and Description sections (r0.3.0)
Nghe	2/8/17	First draft of Functional Requirements sections (r0.4.0)
Macco	2/9/17	First draft of Non-Functional Requirements and User Interface sections (r0.5.0)
Macco	2/10/17	Table of Contents update to renumber appropriately (r0.5.1)
Macco Olyarnk Thompson	2/10/17	Final revisions of Introduction and Description sections (r0.6.0)
Nghe	2/11/17	Final revision of Functional Requirements section (r0.7.0)
Macco Nghe Olyarnik	2/11/17	Completion of Submission Draft (r1.0.0)

Table of Contents

1. Introduction

1.1 Purpose of Document

This document will provide all of the necessary requirements for Remote_Checkers. It is to be used as a reference for the developers and end-users.

1.2 Scope of Document

This document contains enough information that a developer can understand and communicate the details of the system to other developers and end-users. This will allow the developers to translate the requirements into code without any uncertainty.

1.3 Overview of Document

This document contains the functional and non-functional requirements for the Remote_Checkers application, as well as use cases, diagrams, and user interface mockups. The game contains a host Server component and Player Client components which connect to the host Server.

2. Description

2.1 Product Perspective

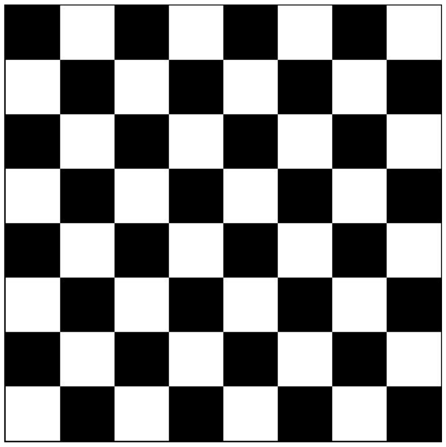

Figure 1: The checkerboard layout used by Remote_Checkers

Remote_Checkers is a game application for two Players to play Checkers from two different locations. The game requires a Server to be running, which can be started by either of the two Players or a third party. The game requires two Players to connect to the Server in order to create a Game session. The game is top-down and follows the standard rules and regulations of Checkers. Players will take turn making Moves and trying to capture all of their opponent's Pieces. When a winner is declared, Players can vote for a rematch, or go back to the main screen and connect to a new Game.

2.1.1 Server Component

The Server is a Java program that must be running before a Game between Players can begin. The Server can reside on either Player's computer or a third party computer. The Server machine's IP Address and Port must be known beforehand so that potential Clients can enter this data and connect to the Game. The Server sends informational Messages to the Client (to be displayed to Players) and handles game logic including Game creation, Move validation, Active Player switching, checking for end-game, and maintaining Board state, as is further defined in **Section 3.1 Server Requirements**. The Server application has no graphical user interface component; a simple command line output message will be displayed on successful launch.

2.1.2 Client Component

The Client is a Java program that Players will directly interface with. It is responsible for basic Move validation, sending Player Moves to the Server, drawing the gameboard including available Moves, and displaying Messages from the Server. The responsibilities of the Client are further defined in **Section 3.2 Client Requirements.** The Client application contains a main screen with options to connect to a Game, view and edit connection settings, and get help. Once a Game has started, the Client will display a graphical interface of the Board, from which Moves can be made.

2.2 Product Functions

After the development team's brainstorming session and a discussion with Zahra Samani (serving as the surrogate customer), the following functionality have been agreed upon:

2.2.1 Server Functions

The Server shall have the following functionality:

- Game logic
  - Game creation
  - Move validation
  - Active Player switching
  - Checking for end-game
  - Maintaining Board state
- Tracking Player scores
- Sending/receiving Messages
- In general, hosting and facilitating a Game session over a Wi-fi network between pairs of Player Clients

2.2.2 Client Functions

The Client shall have the following functionality:

- Connecting to a Server (and joining a Game)
- Sending Player Move Messages
- Basic Move validation
- Displaying the gameboard

- Displaying Messages
- Viewing and editing connection settings
- Getting gameplay help
- In general, providing an interface for Players to interact with the online Game.

2.2.3 Control Flow Diagram

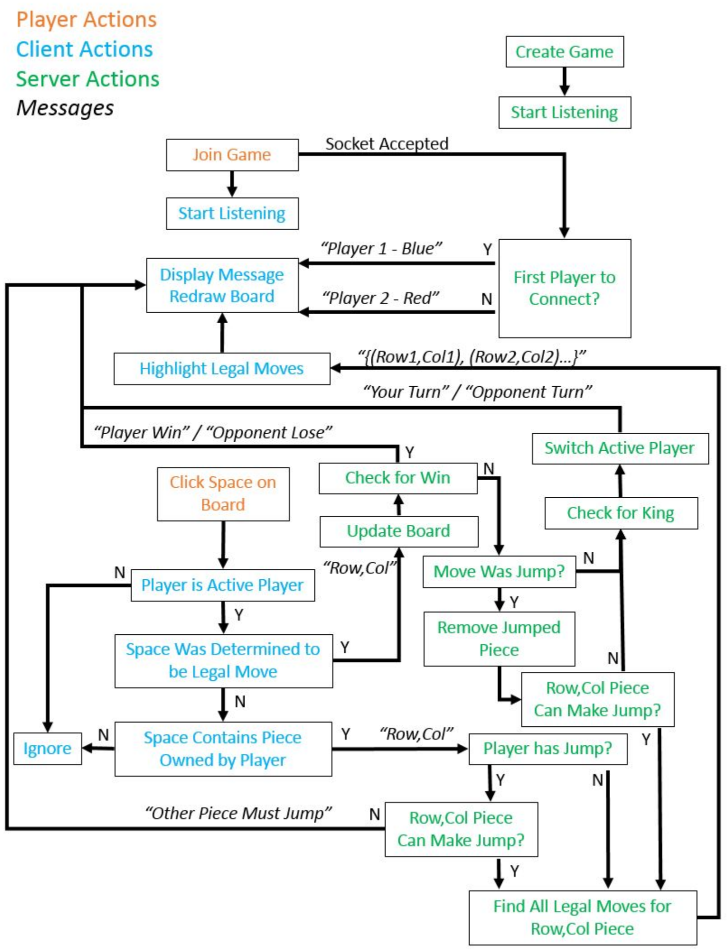

Figure 2: Control Flow Diagram

This diagram shows the general control flow of a game of Remote_Checkers. The user link-in points are as follows:

- Starting the Server to create a Game, which must be done before any other actions
- A Player joining a Game, which launches a Client and starts communication

- A Player clicking a Space on the Board, which will either be validated as a Piece they wish to move or a Space they want to move into or an invalid Move and ignored

2.3 User Description

The ideal users of this app are two people looking to play Checkers but who are in separate locations or do not have access to a physical board and Pieces. Users must be running Java on their computers and must meet the other requirements laid out in **Section 4.1.1 Client System Requirements**.

2.4 Assumptions and Dependencies

The following preconditions are necessary to successfully run Remote_Checkers:

- The Server must be up and running in order for Player Clients to connect and join a Game.
- The Server's IP Address and Port must be known by Players to make this connection.
- Players' computers must be running Java.
- The Wi-fi network connection must not fail.
- All other non-functional requirements listed in **Sections 4.1 System Requirements** and **4.2 Network Requirements** must be met.
- It is NOT assumed that Players know anything about the rules of Checkers. Basic rules are laid out on the help menu and the application does validation of all Moves and displays Messages to Players if they try something illegal.

3. Functional Requirements

3.1 Server Requirements

3.1.1 Rules of Checkers

The Server shall enforce the rules of Checkers and validate Moves from the Clients using those rules. The rules include:[1]

R1.1 Setup

R1.1.1 The checkerboard is an 8-by-8 grid of alternating black and white squares.

R1.1.2 Only the black squares, called Spaces, are used for gameplay.

R1.1.3 Each Player controls 12 Pieces of either blue or red.

R1.1.4 A Player is assigned either the top half or the bottom half of the Board.

R1.1.5 A Player's Pieces are placed on the 12 black Spaces closest to the edge of their assigned half of the Board.

R1.2 Movement

R1.2.1 Pieces can move onto empty Spaces located one square diagonally forward from their current position. This action is called a Step.

R1.2.2 If an opponent's Piece is located one square diagonally forward from a Player's Piece, and the next Space beyond the opponent's Piece on that same diagonal is empty, then the Player's Piece can Jump the opponent's Piece and land in the Space beyond. The opponent's Piece is captured and removed from the Board. This action is called a Jump.

R1.2.3 A normal Piece, called a Man, can only make Steps and Jumps.

R1.2.4 A Jump takes priority over a Step. If one of the Player's Pieces can Jump, it must take that Jump.

R1.2.5 If more than one Piece can Jump, the Player must choose one to take the Jump during that turn.

R1.2.6 After a Jump, if the Player's Piece that Jumped can make another Jump from the new position, it must do so. It must continue to Jump until it cannot Jump anymore.

R1.2.7 If one of the Player's men reaches a Space in the row at the opponent's end of the Board, it becomes a special Piece, called a King. This action is called Crowning.

R1.2.8 Kings shall be designated by displaying the letter "K" on their Piece image.

R1.2.9 A King can Step or Jump both forwards and backwards.

R1.2.10 Once one of the Player's Pieces has taken a Step or has finished Jumping, the Player's turn ends.

R1.3 End-game Conditions

R1.3.1 The Game ends with a win for one of the Players if their opponent has no available Moves on their turn.

R1.3.2 A Player shall have no available Moves if they have no Pieces left or if positioning prevents them from making a Move.

R1.3.3 A Player shall also win if their opponent forfeits or disconnects from the Server.

R1.3.4 The Game ends in a draw if 50 turns have happened without a capture or a Crowning, or if the same exact Board positioning has occurred three times.

3.1.2 Maintaining Board State

R2.1 Board State Storage:

R2.1.1 The Server shall store a representation of the checkerboard as a collection of 64 Board square states.

R2.1.2 The possible states are white Space, empty black Space, empty black Space that is a legal Move for the Active Player (also known as a green space), black Space with red Man, black Space with red King, black Space with blue Man, black Space with blue King.

R2.2 Board State Update:

R2.2.1 Whenever the Active Player selects a Piece they can legally move, the Board representation shall be updated to highlight the legal Moves for that Piece.

R2.2.2 Whenever a legal Move has been made, the Board representation shall be updated to reflect the results of the Move.

R2.2.3 At the start of a Game, the Server shall send the initial Board state to the Clients.

R2.2.4 The Client of the Active Player receives all Board updates that occur during their turn.

R2.2.5 At the end of each turn, the Server shall send the Board state to both Clients.

3.1.3 Post-Game

R3.1 Scoring

R3.1.1 The Server shall keep track of both Players' scores as numbers indicating the number of Games won against the same opponent.

R3.1.2 After a win, the Player who won shall have their score incremented by 1.

R3.2 Rematch

R3.2.1 Once a Game has ended, the Players shall be prompted for a rematch.

R3.2.2 If both Players choose to rematch, the Game will restart with Players' colors and turn order switched.

R3.2.3 If either Player declines the rematch, both Players shall be disconnected from the Server.

3.1.4 Networking

R4.1 Hosting the Game

R4.1.1 The Server shall be able to act as the host to Clients for the purpose of playing a Game of Checkers.

R4.1.2 The Server shall open a Game for every two Clients that are connected.

R4.1.3 If a Client disconnects during a Game, the Game that the Client was in ends. The other Client is then disconnected from the Server.

R4.2 Sending and Receiving Messages

R4.2.1 The Server shall be able to communicate with the Clients via Messages.

R4.2.2 The Server shall be able to interpret Messages from the Clients and act accordingly.

3.2 Client Requirements

3.2.1 Gameplay

R5.1 Game Knowledge

R5.1.1 The Player shall have knowledge of whether they are the Active Player.

R5.1.2 The Player shall have knowledge of the color of Pieces they control.

R5.1.3 The Player shall only be able to select and move their own Pieces when they are the Active Player.

R5.1.4 The Player shall have knowledge of their score and the opponent's score.

R5.2 The Player shall be able to forfeit their Game at any time. This will end the Game and offer both Players a rematch.
R5.3 The Player shall be able to quit their Game at any time. This will end the Game and disconnect the Client from the Server.
R5.4 The Player shall have access to a help screen detailing the basic rules of Checkers at all times.

3.2.2 Graphical Checkerboard

R6.1 Each Client shall be able to receive information about the Board state from the Server.
R6.2 Each Client shall display a graphical representation of the Board to the Player upon receiving the Server's update.
R6.3 The Player shall be able to interact with the Client's graphical representation of the Board state to select and move their Pieces.

3.2.3 Networking

R7.1 Joining the Game

R7.1.1 The Client shall be able to connect to the Server with the purpose of starting a Game of Checkers.

R7.1.2 The Client shall know whether their attempt at connection was successful.

R7.1.3 The Client shall know whether they have been disconnected from the Server.

R7.1.4 Once a successful connection has been made, the Client shall be prompted to wait for an opponent to join.

R7.1.5 Once two Clients have made successful connections, the Game will start.

R7.2 Sending and Receiving Messages

R7.2.1 The Client shall be able to communicate with the Server via Messages.

R7.2.2 The Client shall be able to interpret Messages from the Server and act accordingly.

R7.2.3 The Client shall be able to convert Messages from the Server into a readable format to the Player and display them on the screen if necessary.

4. Non-Functional Requirements

4.1 System Requirements

This section describes the different requirements needed for the system to run.

4.1.1 Client System Requirements

R8.1 The Client should run on any operating system that can run Java; specifically, the Client needs to be able to run Java 1.8.
R8.2 Clients should have a keyboard, mouse, and monitor to enter input into the program.
R8.3 Client systems need to connect to the Internet or local networks; this can be done any way possible.

4.1.2 Server System Requirements

R9.1 The Server should run on any operating system that can run Java; specifically, the Server needs to be able to run Java 1.8.
R9.2 The Server needs to have a way to connect to the Internet or local network.
R9.3 The Server can connect to the network in any way.

4.2 Network Requirements

R10.1 Both the Server and Client need to keep their connection to each other while the system is running.
R10.2 The system should not terminate any connection.

4.2.1 Network Performance

R10.3.1 The network should perform efficiently, and should not have lag over one second.
R10.3.2 Users should not experience wait times from the system for more than one second when they are playing.

4.3 Security

R11.1 The system should be secure enough to sustain a connection between a Server and Client without outside interruptions.
R11.2 Both Server and Client should be secure so that extra code cannot be executed.
R11.3 The system programs should be the only program that can be controlled by the Client-Server connection.

4.4 Support

R12.1 Java system support should come from FAQ's and online forums.
R12.2 The system shall have directions in the help menu, accessible from the main screen and game screen.

4.5 Accessibility

This section describes various ways to download and run the system.

4.5.1 Downloads

R13.1 Client and Server downloads should be provided separately.
R13.2 Both systems should be able to downloaded and run independently on different machines.
R13.3 One Client and the Server should also be able to be downloaded on the same machine.

4.5.2 Running Systems

R14.1 Clients and Servers should be able to be run on the same machine.
R14.2 Clients and Server should be able to be run on different machines.
R14.3 Both systems should run independently.

5. User Interface

5.1 Server

The Server shall not have a graphical user interface, and shall run from the console/terminal of the computer. A simple message shall be displayed on successful launch.

5.2 Client

This section describes different user interfaces on the Clients.

5.2.1 Main Screen

The main screen shall consist of a “Start Game” button which shall bring the user to the Board screen, a “Settings” button which shall bring the user to the settings screen, a “Help” button which shall take the user to a help screen, and a “Quit” button which exits the application.

5.2.2 Board

The Board shall consist of an image of a checkerboard starting with all 24 Pieces placed in the correct locations on only black squares. Each Player will be in control of the 12 Pieces of their assigned color, either blue or red. The Client shall display the Player’s Pieces on the side “closest” to the Player, which is the part of the Board closest to the bottom of the screen.

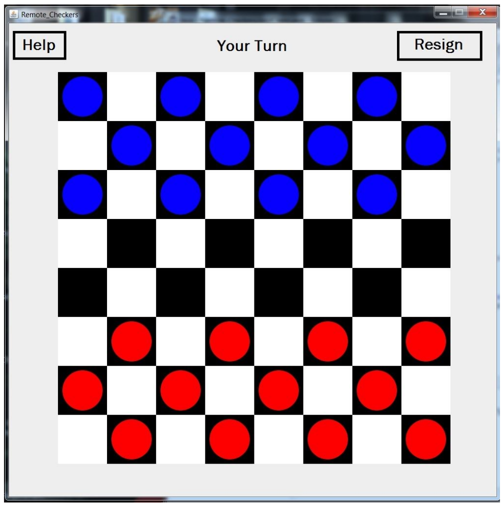

Figure 3: Mock Board Screen

5.2.2.1 Pieces

Each Piece functions as a button. On click, the Board shall highlight available legal Moves for that Piece. Clicking a highlighted Space shall move the Piece to that location, while clicking a different Piece shall cancel the original selection and display Moves for the new Piece.

5.2.2.2 Resign Button

When clicked, this button shall allow the Player to resign the Game and return to the main menu.

5.2.2.3 Help Button

When clicked, this button shall launch the help screen. See **Section 5.2.4 Help Screen**.

5.2.2.4 Message Display

This area is designated for displaying Messages from the Server, including specifying which Player's turn it is and specifying errors of invalid Moves or necessary Jumps.

5.2.3 Settings Screen

This screen allows configuration of the Server connections. This is where Players can enter the IP Address and Port number of the Server they want their Client to connect to when clicking the “Start Game” button.

5.2.4 Help Screen

This screen displays instructions to the Player on the basic rules of Checkers and how to use the program.

6. Use Cases

6.1 Invalid Actions

6.1.1 Attempting to Join a Game without Server Connection

Precondition: Server is not running or connection cannot be made.
Action: Player clicks the “Start Game” button from the main screen.
Postcondition: Message is displayed informing Player that a connection could not be made.

6.1.2 Selecting an Empty Space

Precondition: None.
Action: Player clicks on a white Space or an empty black Space.
Postcondition: Action is ignored.

6.1.3 Selecting an Opponent’s Piece

Precondition: None.
Action: Player clicks on a Space containing an opponent’s Piece.
Postcondition: Action is ignored.

6.1.4 Selecting a Space on Opponent’s Turn

Precondition: None.
Action: Player clicks on any Space while not the Active Player.
Postcondition: Action is ignored.

6.1.5 Selecting a Piece When Another Can Jump

Precondition: Player is Active Player and controls a Piece that can make a Jump.
Action: Player clicks on a Piece they control that cannot make a Jump.
Postcondition: Message is displayed telling Player they must make the Jump.

6.2 Valid Actions

6.2.1 Joining a Game

Precondition: Server is running, application is launched.
Action: Player clicks the “Start Game” button from the main screen.

Postcondition: Player joins a Game. Board is displayed. The Message is displayed informing Player if it is their turn or opponent’s turn.

6.2.2 Selecting a Piece to Move

Precondition: Player is Active Player. If this Piece cannot Jump, Player does not control any which can.

Action: Player clicks on a Piece they control.

Postcondition: The Board is updated with legal Move Spaces highlighted.

6.2.3 Selecting a Space to Move Into

Precondition: Player is Active Player, legal Move Spaces are highlighted.

Action: Player clicks on a highlighted Space.

Postcondition: The Board is updated with the Piece moved to the clicked Space.

6.2.4 Selecting a Space to Jump Into

Precondition: Player is Active Player, legal Move Spaces are highlighted.

Action: Player clicks on a highlighted Space involving a Jump.

Postcondition: The Board is updated with the Piece moved to the clicked Space, the Jumped opponent’s Piece removed, and any additional Jump Spaces highlighted.

6.2.5 Losing a Piece

Precondition: Opponent makes a Jump.

Action: Player’s Piece which was Jumped gets removed from play.

Postcondition: Board is updated with Jumped Piece removed.

6.2.6 Getting Crowned

Precondition: Player moves a Piece into the row farthest from them.

Action: Player’s Piece becomes a King Piece.

Postcondition: Board is updated with the moved Piece as a King. That Piece can now move backward.

6.2.7 Winning the Game

Precondition: Opponent does not control any Pieces or has no legal Moves, opponent forfeits, or opponent disconnects.

Action: Game ends, Player wins.

Postcondition: “You win!” Message is displayed. “Play again?” Message is displayed.

6.2.8 Losing the Game

Precondition: Player does not control any Pieces or has no legal Moves.

Action: Game ends, Player loses.

Postcondition: “You lose!” Message is displayed. “Play again?” Message is displayed.

6.2.9 Drawing the Game

Precondition: 50 turns have passed with no Jumps or Crownings, or the same Board position has been repeated three times.

Action: Game ends, draw occurs.

Postcondition: “Draw!” Message is displayed. “Play again?” Message is displayed.

6.2.10 Forfeiting

Precondition: None.

Action: Player clicks “Resign” button and confirms by clicking “Yes” button.

Postcondition: Opponent wins, Player is returned to the main screen.

6.2.11 Choosing to Play Again

Precondition: Current Game has ended, “Play again?” Message is displayed. The opponent has already confirmed they want to play again.

Action: Player clicks “Yes” to play again.

Postcondition: Game is reset, opponent remains same.

6.2.12 Choosing to Not Play Again

Precondition: Current Game has ended, “Play again?” Message is displayed.

Action: Player clicks “No”.

Postcondition: The main screen is displayed

6.2.13 Getting Help

Precondition: Player is on the main screen.

Action: Player clicks the “Help” button.

Postcondition: Help screen is displayed.

6.2.14 Viewing Connection Settings

Precondition: Player is on the main screen.
Action: Player clicks the “Settings” button.
Postcondition: Server connection settings are displayed.

6.2.15 Editing Connection Settings

Precondition: Player is viewing connection settings from the main screen.
Action: Player types in the fields to edit the Server IP Address and Port, then clicks “Save” button.
Postcondition: Connection settings are updated in the config file. The main screen is displayed.

6.2.16 Quitting the Application

Precondition: Player is on the main screen.
Action: Player clicks the “Quit” button.
Postcondition: Application closes.

7. Glossary

Active Player - The Player whose turn it is.

Board - Either the Server's abstract model representation of the current Game's checkerboard, or the Client's graphical representation of the same.

Checkers - The game being emulated. Players move their Pieces diagonally across the board to try and capture all of their opponent's Pieces.[1]

Client - Program which the Player interacts with. Displays a graphical representation of the Server's gameboard, which Players can click to make Moves. Player actions will be sent in Message form to the Server for validation, at which point the Board will be updated.

Crowning - The act of a Player moving one of their Pieces all the way to the opposite end of the Board, at which time it becomes promoted to a King.

Game - A single instance of a two-Player session of Checkers, created, stored, and updated by the Server. Holds the state of the Board and implements the rules.

Internet Protocol (IP) Address and Port - Information that a Client needs about a Server in order to make a network connection. The IP Address uniquely identifies the machine running Server code, and the Port specifies a channel where the Server is listening to accept potential connections.

Java - The coding language used for development of Remote_Checkers.

Jump - A Checkers Move where one Player "jumps" their Piece over an opponent's Piece in the Space directly diagonal. The opponent's Piece is captured. If a Player can make a Jump on their turn, they must do so.

King - A promoted Piece which can move backwards.

Man - A normal Checkers Piece.

Message - Text sent from Server to Client or Client to Server. The receiver parses the text and decides how to act on it.

Move - The act of a Player choosing a Piece and then a Space to move that Piece into. Either a Step or a Jump.

Piece - A single Checker, either a Man or a King, which a Player controls and moves around the board.

Player - A physical human playing the game, done by interacting with a Client instance unique to them.

Remote_Checkers - The game being designed, allowing two Players to play a game of Checkers from separate locations over a network connection.

Server - Program which handles main Game facilitation. Starts a Game and allows Clients to connect to it to play. Manages any running Games.

Space - A single one of the 64 squares making up the Board. A Space can either be white, empty black, black with a blue Man, black with a red Man, black with a blue King, or black with a red King. Remote_Checkers also defines green Spaces as Spaces marked as valid Moves for the Active Player.

Step - The basic Move of moving a Piece into an empty forward-diagonal Space.

8. References

[1] https://www.itsyourturn.com/t_helptopic2030.html#helpitem1197