Checkers Application
Requirements Document

Group 4:

Christopher Fosythe
Samuel Platek
Robert Roche
Justin Roszko

Table of Contents

Table of Contents **2**

Introduction **4**

Purpose of Document 4

Scope of Document 4

Overview of Document 4

Description **4**

Product Perspective 4

Product Functions 4

User Description 5

Assumptions and Dependencies 5

Functional Requirements **5**

R.1.1 Core Game 5

Movement Detection 5

Capturing 5

Winning a Game 6

R.1.2 Client Side 6

R.1.3 Server Side 6

Non-Functional Requirements **6**

R.2.1 Network Performance 6

R.2.2 Host Operating System Requirements 6

R.2.3 Accessibility 6

Playtesting 6

User Interface **7**

Main Menu Screen 7

Main Game Screen 7

About Screen 8

Game End Screen 8

Use Cases **9**

Use Case Flow 9

Starting Game (Server) 9

Starting Game (Client) 9

Moving a Piece 9

Capturing a Piece 10
Piece Becoming a King 10
Exiting Game (Server and Client) 11
Activity Diagram (Core Game Loop) 12
Glossary **12**
References **12**

1. Introduction

a. Purpose of Document

This document will describe all of the requirements for our checkers application. It will serve as an outline for the development team and the professor to see what is planned to be implemented for the game.

b. Scope of Document

This document will allow developers to create an application based on the requirements outlined in this document and for the instructors to know what to expect from the final product.

c. Overview of Document

This document will contain all of the requirements for the checkers application. The game will be a python application that will be played by users on two different PCs using the python socket feature.. This document will also contain a description of the UI and use cases.

2. Description

a. Product Perspective

Checkers is a classic strategy game where two players face off to capture each others pieces. This application aims to emulate the classic board game on a PC using python.

b. Product Functions

Client

I. Front End (UI) Functions

- Allows users to track game progress
- Displays piece position and allows movement

II. Backend Functions

- Verify player moves
- Process captures and kings
- Track game progress and declare a winner

Server

- Host a game with two clients
- Launch and reset the application as needed

c. User Description

Upon launch the checkers application will support exactly two players on a python socket. These players are assumed to have enough knowledge of the game of checkers, as well as enough knowledge to run a python application on their own.

d. Assumptions and Dependencies

1. Python

   We will be utilizing the Python sockets feature to host the game, thus we must assume both clients have python functioning on their Devices.

2. Windows PC

   This application is developed for Windows PCs so it is assumed users have functioning PCs with basic graphical capabilities.

3. Functional Requirements

a. R.1.1 Core Game

i. Movement Detection

1. R.1.1.1 Not as King: A player moving as a non-king can move only diagonally away from the player on the board. For jumping rules please see the “Capture” section.
2. R.1.1.2 When a normal piece reaches the other side it is “kinged” and given more abilities.
3. R.1.1.3 As King: A king may move both towards and away the player diagonally.

ii. Capturing

1. R.1.1.4 Not as King: As a non-king one can capture a piece when an enemy piece is the forward-diagonal space directly next to the players piece and the space beyond that piece is empty.
2. R.1.1.5 If a jump is possible the player must take the jump. After completing a jump the player lands in the space beyond the enemy piece in the same direction.
3. R.1.1.6 If a jump is available after the initial jump, the player must jump again and continue jumping pieces until they no longer can.
4. R.1.1.7 As King: A king may jump backwards or forwards diagonally however, it can only jump one piece at a time.

iii. Winning a Game

1. R.1.1.8 The game is won when all of the opponents pieces are captured or they are unable to make any more moves.
2. R.1.1.9 Any player may forfeit if they feel that they are unable to continue playing.

b. R.1.2 Client Side

1. R.1.2.1 The ability to see the state of the game board at any time as defined as knowing the location of all pieces.
2. R.1.2.2 The ability to click and drag pieces to make a move
3. R.1.2.3 An indicator to determine who's turn it currently is.
4. R.1.2.4 A button to forfeit the match, which grants victory to the opponent.

c. R.1.3 Server Side

1. R.1.3.1 Maintains a consistent state between the two players in order to have the same information present for both players
2. R.1.3.2 Updates the shared State of the game when a player makes a move
3. R.1.3.3 Determines if a player's move is valid as determined by the Requirements R.1.1.1 through R.1.1.7

d. R.1.4 Hosting,Joining, and Quitting a Game

1. R.1.4.1 Launch the application by clicking the .py file.
2. R.1.4.2 Load main menu on launch.
3. R.1.4.3 User hosts a game when the 'Host Game' button is clicked.
4. R.1.4.4 Game is initialized with a unique ID.
5. R.1.4.5 Other user joins a game when the 'Join Game' button is clicked and the game ID is entered.
6. R.1.4.6 Server checks that there is room for the player in the game.
7. R.1.4.7 Ability to end the game when the quit button is clicked.
8. R.1.4.8 Ability for both players to select to rematch when rematch buttons are clicked

4. Non-Functional Requirements

a. R.2.1 Network Performance

1. R.2.1.1 The delay between when a player makes a move and when it's updated on both player's screens should not have a delay greater than 2000 milliseconds

b. R.2.2 Host Operating System Requirements

1. R.2.2.1 This product will guarantee support on Windows 10 computers only. Other operating systems may work, but is NOT a part of the requirements.
2. R.2.2.2 This product will support python3 only
3. R.2.2.3 The computer in use will be required to have a network card to communicate with the other player

c. R.2.3 Accessibility

1. R.2.3.1 Players will be given a link to download all required files to run the game

d. Playtesting

Playtesting will be done by the development team as well as other college students throughout the development process. First and foremost playtesting will ensure we minimize bugs, as defined as a player doing something against the rules of checkers (see Requirements R.1.1.1 through R.1.1.9). If a player runs into a bug they will be asked to record the condition in which the bug was found. In addition to bug detection, playtesting will allow us to receive feedback on the aesthetics of the game. Since checkers has established rules and balancing is not needed, most of the critical feedback will be based on the visuals of the game. This feedback will allow us to adjust color schemes, and add or delete UI elements based on testers feedback. With the bug and aesthetic feedback we will ensure our game not only runs as well as possible, but is also pleasing to look at.

5. User Interface

a. Main Menu Screen

The main menu screen will contain two buttons

i. Play game
   - This will open a module with two buttons saying “Host” and “Join”. The Host button will replace the content of the module with a code that the host gives to the client to connect to the same game. The join button replaces the module with a text box where they can enter a code from a host that allows them to join their game.

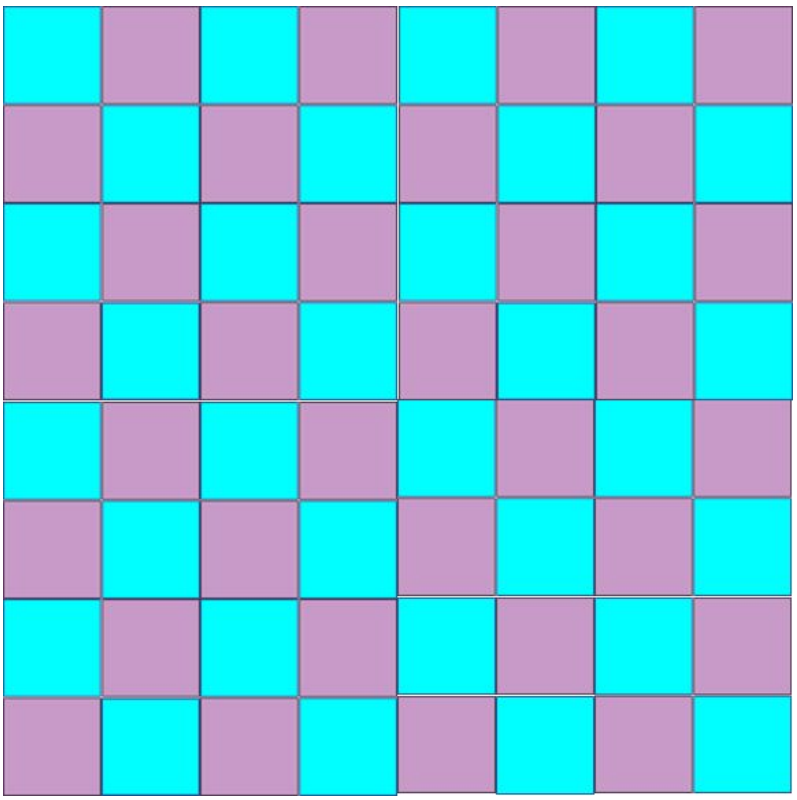
ii. About
   - Redirects to About Screen to give information about the game and the changelog of the application.

b. Main Game Screen

The main game screen will mostly consist of a cyan and pink checkerboard (shown below) and will also have a border around it with a button at the bottom that says “Surrender”, as well as a text box to the right specifying the user’s color. The user will be unable to move pieces until they are connected with another game.

The checkerboard will be used to play the game, allowing one player to move a piece at a time, according to the rules specified in the use cases.

The Surrender button will forfeit the game for the user that clicks on it and will force both users back to the End Game Screen



c. About Screen

i. About the Team

There will be a section in a text area that gives a very short bio on the members of the team.

ii. Changelog

There will be a scrollable text area underneath the about the team section with a changelog, detailing what changes are made each time code is

altered. The log will have the most recent changes at the top and the oldest changes at the bottom.

iii. Return to Main Menu Button

This button will be at the bottom center of the screen and will allow the user to return to the main menu.

d. Game End Screen

The game end screen will only display when the game is ended through gameplay, when all of either team's pieces have been removed from the board.

The game end screen will consist of the words "Game Over" in large text at the top middle of the page and beneath this, also centered, will be text saying "[Color] team has won!".

Beneath both of these, also centered will be two buttons next to each other with the text "Rematch" on the left button and "Return to Main Menu" on the right button. Rematch will start a new game with the same players if both players select it. If either player selects Return to Main Menu, both players will be directed back to the Main Menu Screen.

6. Use Cases

a. Use Case Flow

i. Starting Game (Server)

The game is started and the user opts to host the game

Precondition: A user selects Play Game Button from the Main Menu Screen
Action: The user selects "Host" from the popout modal that shows after Play Game is selected
Postcondition: The modal changes its content to display a string that a client may enter to join that game

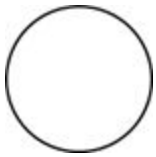
ii. Starting Game (Client)

A user attempts to join a host's game

Precondition: A user selects Play Game Button from the Main Menu Screen
Action: The user selects “Join” from the popout modal that shows after Play Game is selected
Postcondition: The modal content is replaced with a textbox where the user may enter a code. When a valid open code is entered, the server and client users will both be directed to the Main Game Screen with a newly-initialized game and both users will be assigned a color

iii. Moving a Piece

Precondition: It is the player’s turn, there are valid moves, and there are no captures that can be made
Action: Player moves one (1) piece in a valid move (one space diagonally towards the direction of the other player’s side of the board
Postcondition: Game board is updated for both players



Regular pieces will be filled with the color of the player

iv. Capturing a Piece

A player’s piece jumps over one of the pieces of the other player, removing that piece from play

Precondition: At least one move which captures at least one of the opponent’s pieces are available (A piece that belongs to the opponent is in a position that the player’s piece can normally move and has an open space one more space in the same direction) and it is the player’s turn
Action: Player is forced to complete all captures available from selected piece, continuing with more captures that may exist from the spot that the player lands after the prior capture
Postcondition: Captured pieces are removed from the board and the capturing piece is in its final position after all captures are made

v. Piece Becoming a King

If a piece reaches the furthest row from its player’s side of the board, it becomes a King and gains the following abilities:

1. Immunity from the Catholic Church
2. A funny hat

3. The ability to walk backwards

Precondition: A player's piece must be able to move a non-kinged piece piece into the furthest row from them
Action: The player moves this piece into the final row
Postcondition: The piece moved into the final row will get the king piece icon and be able to move towards the player that owns it, as well as away

King piece will be filled with the color of the player

vi. Exiting Game (Server and Client)

Either player selects to forfeit the game or all of either player's pieces are removed, causing the game to be over, in this case either player may quit out of the game

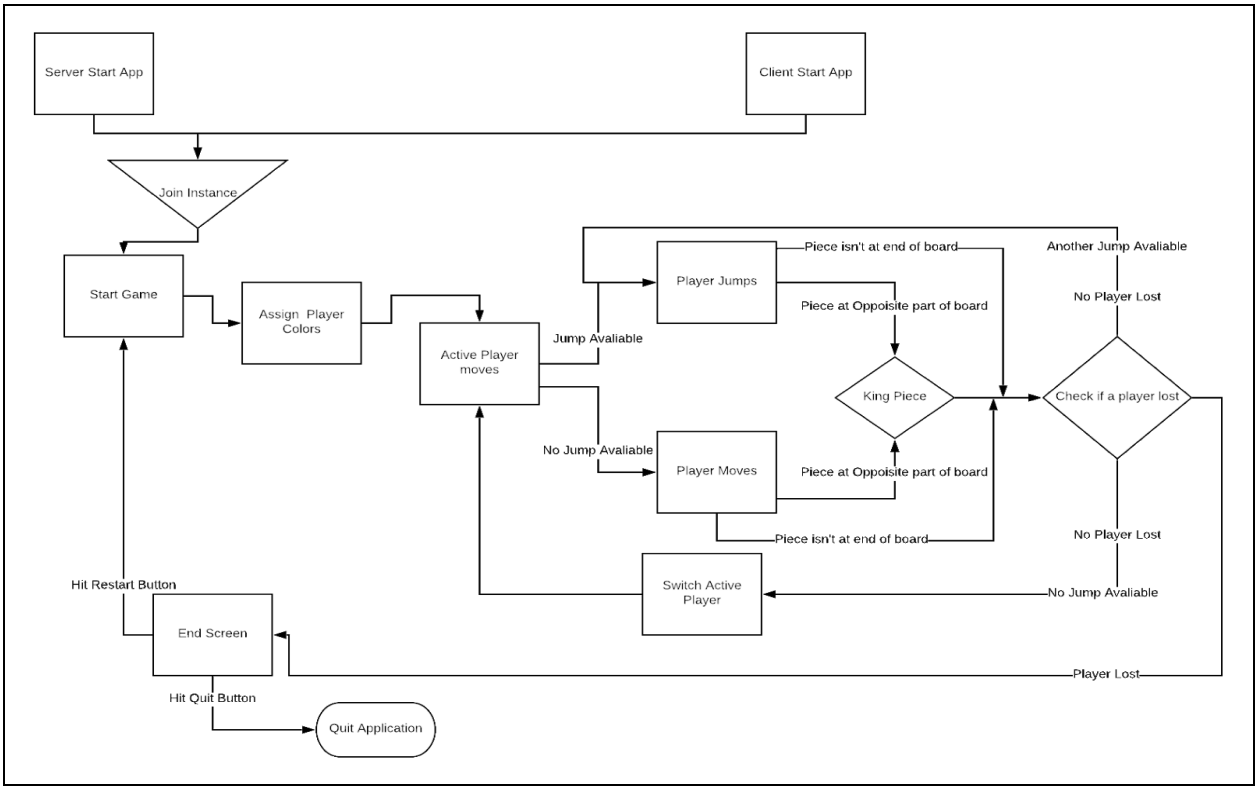
Precondition: The last of either player's pieces have been removed or either player hits the forfeit button
Action: Either player selects the Return to Main Menu button on the End Game Screen
Postcondition: Both players are sent to the main menu

vii. Restarting Game (Server and Client)

A rematch is chosen by both players after a game is ended through forfeit or win/loss

Precondition: The last of either player's pieces have been removed or either player hits the forfeit button
Action: Both players select the rematch button while on the End Game Screen
Postcondition: Both players are sent back to the Main Game Screen and given a newly-initialized game

b. Activity Diagram (Core Game Loop)



7. Glossary

a. Python - Python3

8. References

a. https://www.fgbradleys.com/rules/Checkers.pdf