SOFTWARE REQUIREMENTS SPECIFICATION

for

Checkers

Version 1.0.0

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 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 Functions

1. Provide an application to host a checkers game with two user over the local network or internet.
2. Provide a server that mediates gameplay, game sessions, and client interactions.

2.2 Assumptions and Dependencies

1. A connection to a local network or internet connection.
2. A computer with a graphical environment for the client
3. A Unix or Windows based server.
4. Knowledge of the rules of checkers.
5. Client know how to launch a python file.

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.
- R1.2. Clients will be able to connect to an existing game session by providing an unique ID to the server.
- R1.3. Clients will not be able to connect to a game session with an incorrect unique ID.
- R1.4. Clients will be able to send moves to the server for validation if it is their turn.
- R1.5. Clients will be able to pause a game session given approval from both clients.
- R1.6. Clients will be able to leave from the game session without consequence should the game be paused.
- 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.

3.1.1 Board State

R2. Board

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

3.2 Server

R3. Server status

- R3.1. Server should be able to be run constantly without crashing.
- R3.2. Server will have a heartbeat function that will send an email to the developers if the server is down.

R4. Server - Client Interaction

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

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

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.

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

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

4 Other Requirements

4.1 System Requirements

S1. Server and Client

Python3

S2. Client

Windowing display environment:

Windows

MacOS

4.2 Network Requirements

N1. Client and Server

N1.1. An active internet connection

N1.2. port forwarding configured properly on their local network

N1.3. Client must be connected to Drexel's network

N1.4. Response time to the server must be less than 120ms

N2. Server

N2.1. Server must be hosted on tux.cci.drexel.edu

N2.2. Server will be running on one dedicated box

5 User Interface

5.1 Framework

The project shall use Python3? to create the user interface.

5.2 Menus

- Main Menu

  Play Game button: connects player to the server

  Quit Game: closes the game

- Game Lobby

  Show players in the lobby

  Ready? button: start game once both players click

  Main Menu button: return to the main menu

- Game

  Checkers match screen, showing a checkers board with both players' pieces

  Player's "side" of the board is always on the bottom of the screen

  Top of screen shows whose turn it is: Black or Red

  During the player's turn, they click on a piece to select it

  Piece becomes highlighted, as does all valid spaces the player could move to

  Player clicks on a highlighted space to move the piece there or clicks on a new piece to select it

  After a move has been made, the game checks if there is a winner. Move to Winner Display if a winner is found. Otherwise, start next player's turn.

- Winner Display

  Show the player who won the game: Black or Red

  Rematch? No button: return both players to the main menu

  Rematch? Yes button: register this player as wanting a rematch

  If both players click Yes, then return both players to the Game Lobby screen

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

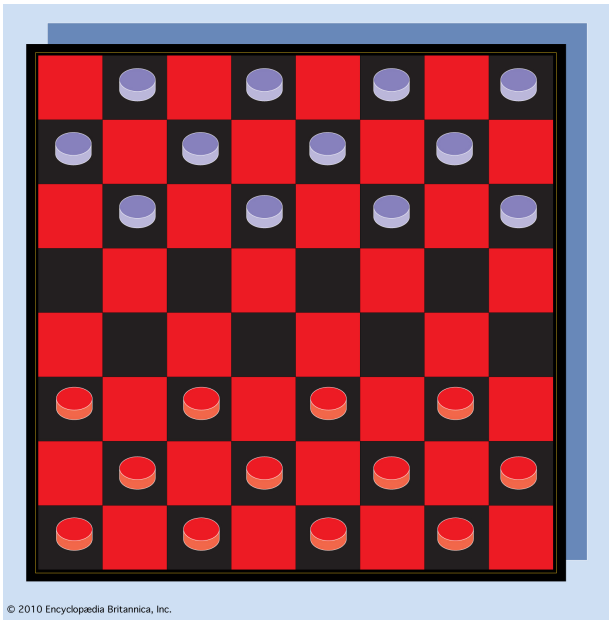
6.1.1 Gameplay

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

- 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.