CHECKERS ONLINE

Requirements Specification

Group Members	Zacharia Thottakara Yansen Tjandra Lucas Vitalos Andrew Yaros
Faculty Adviser	Filippos Vokolos, Ph.D

REVISION HISTORY

Contributors	Date	Change Summary	Revision Number
Zacharia Thottakara	2018-07-25	First revision -- set up requirements outline.	0.5
Zacharia Thottakara Yansen Tjandra Lucas Vitalos	2018-07-29	Flesh out major points, add a game window mockup.	0.8
Zacharia Thottakara Yansen Tjandra Lucas Vitalos	2018-07-30	Streamline further, disambiguate some points, add UI flowchart.	0.9
Zacharia Thottakara Yansen Tjandra Lucas Vitalos Andrew Yaros	2018-07-31	Document cleanup, formatting; **final version**.	1.0

TABLE OF CONTENTS

1. Introduction

1.1. Purpose of document

This document has been made to provide all of the requirements necessary to create, operate, and maintain an online checkers game. It will serve as a reference to anyone involved with the game, user, developer or otherwise.

1.2. Scope of document

This document will contain all of the information necessary for the developer to create an online checkers program with little to no ambiguity.

1.3. Overview of the document

This will contain both functional and non-functional requirements for the game Online Checkers. It will also contain use cases, diagrams if needed, and user interface mockups.

2. Description

2.1. Product

Online Checkers is designed to be a two-player game, playable from anywhere with an internet connection. The users will launch the game via browser. Once in the game, **players** will play a traditional game of checkers to **game completion**, or **game termination**.

The game itself will be a 2D top down view of a traditional 8x8 checkers board. Pieces are displayed as circles with the player's chosen color. At the start all pieces will be placed in starting positions. As the game progresses, King pieces will be given special symbol (crown) denoting the piece. A running total of pieces taken, and pieces taken from you, will be totaled below the board (in each players view).

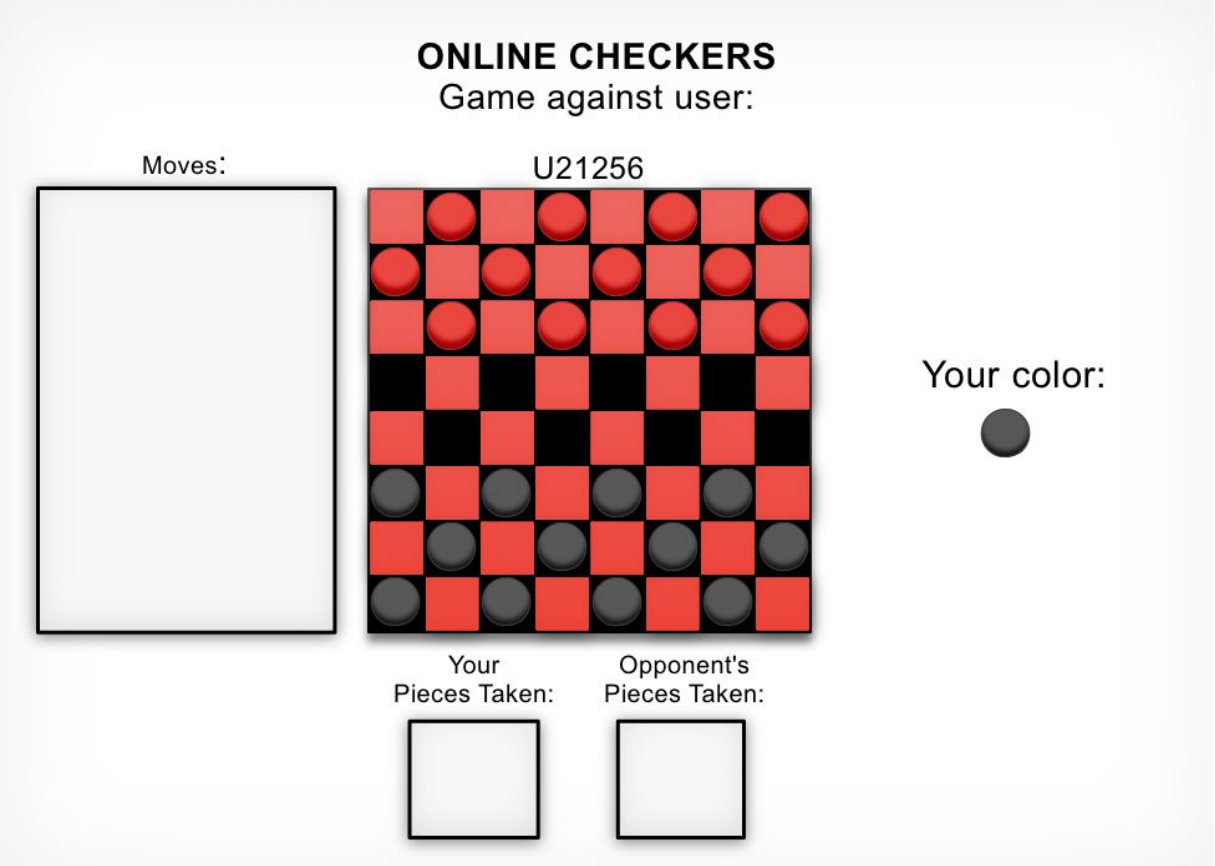

Fig. 01: Mockup of the game window

2.2. Game rules

Online Checkers will use English draughts (American checkers) rules.

2.2.1. Board

The game is played on an 8x8 board tiled alternately with black and red* squares. It shall be turned such that the near righthand corner of the board is a red square for both players.

Players shall take positions on opposite sides of the board.

* They need not actually be black and red, but they should be visually distinct colors. They shall be referred to as black and red hereafter.

2.2.2. Pieces

Pieces are divided into two sets: 12 black and 12 white*. Black pieces have priority (play first).
Assignment of priority is decided by a simulated coin flip.

Pieces are placed only on the black squares. Each player's pieces will be placed on the twelve black squares nearest to them.

Players may only move pieces of their own color.

There are two types of pieces: **normal** (uncrowned) and "**King**" (crowned).

* Again, they need not actually be black and white, but for simplicity they shall be referred to in this manner. One set must be mutually decided upon as the "black" set.

2.2.3. Moves

Players may **move** normal pieces in the direction opposite them ("forwards") to diagonally adjacent black spaces, if they are vacant. Kings may be moved either forwards or backwards.

If a normal piece reaches the opposite edge of the board, that piece is "**crowned**" and becomes a King. The act of crowning ends the player's turn.
If a piece is diagonally adjacent to an opposing player's piece, and there is a vacant spot on the other side of the opposing player's piece in the same direction, then that piece must "**capture**" the opposing player's piece. The captured piece is removed from the board, and the capturing piece takes the vacant spot opposite the captured piece.

Should the capturing piece then be in a position to make another capture, it must do so and continue in this way until it cannot make another capture or -- if it is uncrowned -- it reaches the opposite end of the board.
If the player must choose between more than one capture on a given turn, they may make that choice freely.

2.2.4. Progression of play

A **turn** is a maximal consecutive sequence of moves performed by one player.

Control moves to the other player at the end of each turn.

There are no time limits imposed by the game itself upon the length of a turn.

2.2.5. Game completion

The game is over when:

- all of one player's pieces have been captured;
- a player has no legal moves;
- or neither player is able to force a win. Such a determination is made if no captures have occurred in the last 40 turns.

2.3. Server functionality

The server will be responsible for:

- pairing players with opponents ("**matchmaking**");
- maintaining connections to players;
- maintaining the game state;
- final move validation;
- and keeping a history of the last three games played by a given user.

2.4. Client functionality

The client will be responsible for:

- displaying the game board and UI;
- basic move validation (giving move options for a selected piece);
- sending moves to the server;
- and receiving updates from the server.

2.5. Target audience

This game is meant for players of any age or skill level.

2.6. Scope of work

The main priority for this system is to create an environment where two people are able to play checkers game against each other.

Priority 1 - Essential requirement that will be added to the final build. These requirements will be completed and tested for correctness.

- Create a client for the game that will work on recent versions of mainstream browsers (Chrome, Firefox, Edge, Safari)
- Create a server that will be able to host the game
- Create a main menu where players can see the list of available games and join the game.
- Create and develop the GUI for the checkerboard and pieces that can be moved around the board by the players.
- Implement the rules of checkers for move validation and game completion.

Priority 2 - Requirements that will not necessarily be added to the final build, but will be implemented and fully tested if time permits.

- Add option to play a single player mode against a **CPU opponent** (very simple opponent logic)
- Maintain a textual history of games played by a user during the current session.

3. Functional Requirements

3.1. Server

3.1.1. Matchmaking **(Priority 1)**

After opting to enter a match, players will be paired with other players. This pairing occurs in one of two ways:

- directly, when two players enter each other's **UUID** (unique user identifier)
- randomly, when players opt to be matched with any available player

Once paired, the server will maintain a connection to both players (to the extent that this is possible). Paired players are unavailable for new matches for the duration of their current match.

Players may also opt to join a match with a CPU opponent. It will play as a normal match, but without the need to find another player (still requires a connection to the server). **(Priority 2)**

3.1.1.1. UUID assignment **(Priority 1)**

Players will be assigned a UUID upon connection to the server. This assignment will be in effect for as long as they are connected to the server, but it does not persist across connections.

3.1.2. Game state **(Priority 1)**

The server will track the following elements of the game state for each match:

- Color of each player
- King pieces
- Number of moves made
- Pieces taken

3.1.3. Move validation **(Priority 1)**

The server will validate every move made by each player.
After every move the server will check:

- if the game has been won;
- if the game is a tie;
- and if the there are any possible moves for the player in control.

3.1.4. History **(Priority 2)**

The server will maintain a textual history of last 3 games played -- for each game, a list of the moves from start to finish.

3.2. Client

3.2.1. Main Screen **(Priority 1)**

3.2.1.1. Join by UUID

- Present a text box for UUID input to the left of the button
    - Text validation on button press
    - If input is valid, join a game with the person whose UUID was entered (or wait for the entered user to accept your request for a game by entering your UUID at the same screen).

3.2.1.2. Join Random Game

- Join a random game when other players are logged on.
    - If no other players online, “No Players Found” will be displayed, then the player will be returned to the main menu.

3.2.1.3. Play against a CPU **(Priority 2)**

- Initiate a request to the server for a game with a CPU opponent.

3.2.1.4. Info

- Bring user to Info Menu.

3.2.2. Info Menu **(Priority 1)**

- (Top Left) Return: take user to Main Menu
- Display user’s UUID
- Display Win/Loss/Tie total **(Priority 2)**

3.2.2.1. History button **(Priority 2)**

- Textual history of games played
    - Can select a game to display a list of the game's moves from beginning to end

3.2.2.2. About button **(Priority 2)**

- Display description of game and how it works

3.2.3. Game Window **(Priority 1)**

- The top displays the game name (“Checkers Online”) and the UUID of the opponent.
- The game board is displayed below.
    - The game board is an 8x8 checkerboard.
    - Pieces will be displayed on their board positions.
    - Below the board, counts of pieces taken will be displayed.
    - To the left of the board, a list of recent moves will be displayed.
    - To the right of the board, the player's color will be displayed.

3.2.3.1. Making moves

- A player will click on a piece and potential end locations for moves will be highlighted.
- Players may then click on a highlighted spot to move the selected piece to that spot.

4. Non-Functional Requirements

4.1. Network performance and stability

Because of the nature of the game there is a minute amount of data that needs to be sent between devices, so network performance is not going to be considered heavily in the design of the application. The server will do all it can to maintain connections, but match stability largely depends on the ability of the user to maintain a stable connection to the server. Should a game be lost due to network difficulties, it will not be preserved for later reentry.

4.2. Server platform requirements

The server will be built for Ubuntu 17.04. There are no plans to support other operating systems at this time.

4.3. Client platform requirements

The will be no operating system requirements other than those for the latest web browser that we plan to support: Chrome 67, Firefox 61, Edge 42, or Safari 11. We will test the client in Safari on macOS 10, in Edge on Windows 10, and in Chrome and Firefox on both macOS 10 and Windows 10.

4.4. Accessibility

Free downloads for all software will be made available and setup instructions included with downloads in a README file.

4.5. Playtesting

At the end of each prototype, the game will be playtested by 4 people in 2 pairs.

5. User Interface

5.1. Server

The server will not have a user interface. It will be configurable through command line and configuration files. Once running the server should only need to be restarted when there are configuration changes.

5.1.1. Configurables

All configurables are set via command line option or configuration file, the former taking precedence when these two modes are in conflict.

5.1.1.1. Port

By default, the server will listen for requests on port 443.

5.2. Client

The client interface will be as follows:

1. The user launches the application and sees the game logo and 3 options:
    a. Join with UUID
    b. Join random game
    c. Open info menu
2. After joining a game the user will see the interface below.

a. The top it the name of the game
b. Next down is the username of the opponent
c. Next down is the current game board with all pieces
d. To the right of that is the players current color
e. Below the board are 3 boxes, counting:
    i. Pieces the opponent has taken
    ii. Pieces the player has taken
    iii. Total number of moves

3. After the game is completed, the "game end" screen will display either:
    a. Winner
    b. Try again
    c. Tie

    The game end screen will have the following options:
    d. Exit: return to main menu
    e. Rematch: prompt opponent for rematch or join rematch
    f. Join new Random game
4. The info menu will display a profile page, which contains:
    a. Users UUID
    b. Win/Loss/Tie total
    c. History button
        i. The history button will take the user to a list of files containing the textual history of a game. Users will have the option to read through the moves
    d. About button
        i. The about button will list a simple description of the application and how it works.

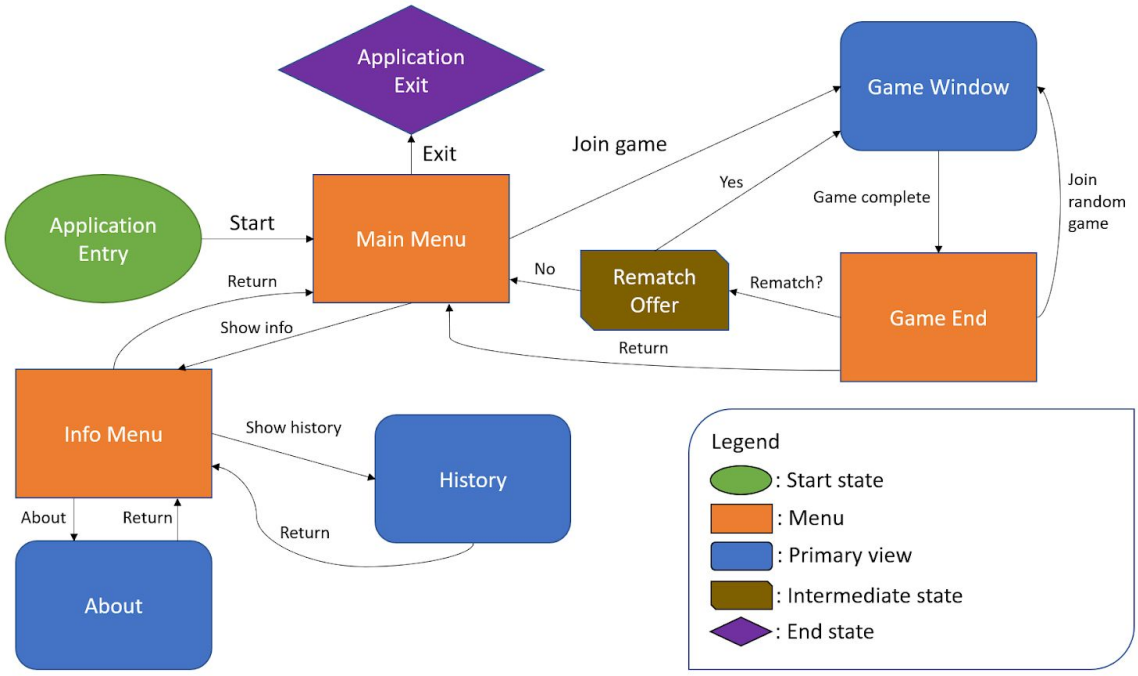

Fig. 02: States and transitions for the Client UI

6. Glossary

6.1. Players

Any of the two users currently in game.

6.2. Game completion

A situation in which one of the players wins or the game ends in a tie.

6.3. Game termination

If one or both of the players disconnects before game completion; considered an error.

6.4. Move

The transfer of a piece on the board from one space to another. Pieces may only move diagonally and onto black spaces (all start on black spaces).

6.5. Normal piece

An unadorned piece which may only move forwards (towards the opponent's side of the board). This is the starting state for all pieces in the game.

6.6. Crowning

The act of making a normal piece into a King. Occurs when a normal piece makes it to the opponent's edge of the board.

6.7. King piece

A piece which has been crowned. It will have a visual marker to indicate that it is not a normal piece. King pieces may move either forwards or backwards on any given turn.

6.8. Capture

A special type of move in which a piece "leaps" over an opponent's piece and lands on the spot opposite it. The opponent's piece is then "captured" and removed from the board.

6.9. Turn

A maximal consecutive sequence of moves performed by one player.

6.10. Matchmaking

The process of pairing players with other players as opponents. This must occur in some manner before games can begin. In this game, the modes of matchmaking are "direct" (unmanaged), or "random" (managed).