Checkers Application
Test Case Document

Group 4:

Christopher Fosythe
Samuel Platek
Robert Roche
Justin Roszko

Table of Contents

1. Introduction

1.1 Purpose of Document

The purpose of this document is to describe the procedures that will be used to test the Software Application for Checkers. These procedures will test functionality, performance, and any additional expectations set in the requirements document. The end goal is to have a fully functional, optimized Python application that allows users to play checkers on separate devices on their local network.

1.2 Definitions

1.2.1 Game State

Main Game Screen - Refers to the screen that both the client and server see that includes the game board, the pieces, and a button for either player to forfeit the match

Turn - Defines who is allowed to move a piece at any given moment during a game. If it is one players turn, only that player may move a piece, the other players pieces will be locked at this point in time

Win Conditions - The circumstances under which a game ends and one player wins. The win conditions are one player completely eliminating the other's pieces, in which case they win, or if one of the players forfeit, in which case the other player wins

1.2.2 Pieces

Normal - Normal pieces are the pieces that are put on the board at the beginning of a game. They are represented by circles of a certain color depending on the player that they belong to, either white or black, with the host being the white pieces and the guest being the black pieces. The white team will go first. The normal pieces will only be able to move diagonally towards the other side of the board, until that piece becomes a king.

Jump - A Jump is a move that takes an opponent's piece out of play. It is possible to perform multiple jumps with one piece in one turn, wherein the players piece captures multiple of the opponent's pieces. A jump is only possible when the player's piece is adjacent to one or many of the opponent's pieces, and there is a spot free of other pieces behind the desired piece to be captured. If there are one or many jump moves available, the player will be forced to pick among those piece(s) only. Likewise, if the selected piece can jump multiple opponents pieces

in one turn, then the piece is forced to continue jumping until it is no longer in a position to do so.

King - A king is just like a normal piece but the image used has a gold crown in the middle of the circle. The king has the ability to move diagonally towards the area that they originated as well as away.

1.2.3 Glossary

Move - Moving a piece to a diagonal empty space from the current position

Modal - A pop-up window that is not a screen in the UI. These will be used mostly for selections made by the user

Host - the player begins the game by selecting “Host Game” from the connection modal. The application then opens a socket for another player to join

Host Code - A code displayed to the Host, which a guest must enter to join the game

Guest - the other player that chooses “Join Game” from the connection modal. The application requires a host code to be entered by this player to connect to the correct Host

Forward - A synonym for “Away from the player”, where *forwards for white* == *backwards for black* and vice versa

Backwards - A synonym for “Towards the player”, where *backwards for white* == *forward for black* and vice versa

2. Testing Environment

2.1 Windows Environment - Windows 10 Education

Machine Name:	Windows PC	DB Directory:	N/A
OS and Version:	Windows10 Education; 8 GB RAM; 1 TB HDD	Interpreter Platform	Python3

Client/Server Backend	LAN	Tester Name:	Justin Roszko
Test Date	TBD	New Log	N/A
State	TBD		

2.2 Mac Environment - Mac OS 10.15 Catalina

Machine Name:	Macbook Pro	DB Directory:	N/A
OS and Version:	Mac OS 10.15; 12 GB RAM; 256 GB SSD	Interpreter Platform	Python3
Client/Server Backend	LAN	Tester Name:	Sam Platek
Test Date	08/26/19	New Log:	Does not render board's colors well
State	Passed		

3. Setup Information and Prerequisites

Prior to running the tests, the following conditions must be met.

- Many of the tests require the GUI in order to perform the tests.
- The Github repo locally.
- Internet is required for those tests which test the connection and data passed along it

4. Test Cases

4.1 Start the Game

4.1.1 Description

This case covers the steps required for a host and to start a game and have another player join the game.

4.1.2 Pre-Conditions

An internet connection and the application present onto the computer.

4.1.3 Scenario

ID	Req	Description	Execution Step	Expected Result	Actual Result
A1	R.1.4.1, R.1.4.2	Launch Application	1. Launch Application by Clicking the .py file	Application Launches and Main Menu Displays	Must be launched from command
A2	R.1.4.3, R.1.4.4	Host Creates a Game	1. Click ‘Host Game’ Button	Game is created with a unique host code	As Expected
A3	R.1.4.5	Connect	1. Click ‘Join Game’ button 2. Enter active host code	Game window opens and connection is attempted	As Expected
A4	R.1.4.5, R.1.4.6	Connection Failure (Too many clients)	1. Click ‘Join Game’ button 2. Enter host code in modal	Connection to host fails, an error message is returned to the user. User is returned to the main menu.	Not implemented. Program quits instead

				Game remains undisturbed.	
A5	R.1.4.5, R.1.4.6	Connection Failure (Invalid ID)	1. Click ‘Join Game’ Button 2. Enter a host code that is not for an active game	Connection fails, an error message is returned to the user, the user is returned to main menu	
A6	R.1.4.5	Connection Success	1. Click ‘Join Game’ Button 2. Enter active host code	Server validates the game has only 1 player. Client joins and a board populated with checkers appears.	As Expected

4.2 Playing the Game

4.2.1 Description

This case describes all the possible actions that a player could take, their expected response, and the steps required to enact those actions for playing the game. These steps will be repeated for both the host and the guest.

4.2.2 Pre-Conditions

An internet connection, the application has been downloaded onto the computer, and a game has to be in progress.

4.2.3 Scenario

ID	Req	Description	Execution Step	Expected	Actual

				Result	**Result**
B1	R.1.1.1, R.1.3.3	Non-King forward, diagonal Movement	1. Find a normal piece that has an empty square diagonally away from the player 2. Click the piece 3. Click a square diagonal to and forward that is empty	The piece moves from the source square and is now at the destination square. Switches active player to the other player.	As Expected
B2	R.1.1.1, R.1.3.3	Non-King Illegal Movement	1. Find a normal piece that has an empty square vertically and horizontally 2. Click the piece 3. Click an empty square horizontal to the piece 4. Click an empty square vertical to the piece	The normal piece does not change squares due to it being an illegal move. User is notified every time an illegal move is attempted.	As Expected
B3	R.1.1.1, R.1.3.3	Non-King Illegal, empty Backwards Movement	1. Find a normal piece that does have any empty square diagonally backwards 2. Click the piece 3. Click a square diagonal to and backward that is empty	The normal piece does not change squares due to it being an illegal move. User is notified every time an illegal move is attempted.	As Expected
B4	R.1.1.4, R.1.3.3	Non-King jumping	1. Find a normal piece that has an enemy piece 1 square forward diagonally from the square piece and also has an empty square in the 2nd square forward diagonally in	The normal piece is moved to the empty square. The enemy piece is removed from the board. If no other jump	As Expected

			the same direction 2. Click the piece 3. Click the empty square forward diagonally 2 squares from the player	is available, switches active player to the other player.	
B5	R.1.1.3, R.1.3.3	King Forward and Backward movement	1. Locate and click a king piece that can move 2. Click the empty square in the forward direction 3. Click the empty square in the backward direction	The king piece moves forward to the clicked space, and backward to the clicked space. Switches active player to the other player.	As Expected
B6	R.1.1.3, R.1.3.3	King illegal vertical and horizontal movement	1. Locate and click a movable king piece 2. Click the empty space directly in front of the piece 3. Click the empty space directly behind the piece 4. Click the empty space directly right of the piece 5. Click the empty space directly left of the piece	The king piece does not move. User is notified every time an illegal move is attempted.	As Expected
B7	R1.1.7, R.1.3.3	King jumping	1. Find a king piece that has an enemy piece 1 square forward diagonally from the square piece and also has an empty square in the 2nd square forward diagonally in the same direction 2. Click the piece 3. Click the empty square	The king should move from its previous square to the square beyond the enemy piece, the enemy piece should disappear. This should happen in	As Expected

			4. Find a king piece that has an enemy piece 1 square backward diagonally from the square piece and also has an empty square in the 2nd square backward diagonally in the same direction 5. Click the piece 6. Click the empty square	both directions. If no jump available at this point, switches active player to the other player.	
B8	R.1.1.5, R.1.3.3	Force player to Jump if available	1. Player attempts a legal non-jumping move when a jump is available	Move does not execute, player is notified a jump move is available	Not implemented
B9	R.1.1.6, R.1.3.3	Force player to jump again if available after a jump	1. Player jumps an enemy 2. Player lands in another jumpable position 3. Player attempts a non-jump action	Move does not execute, turn does not change. Player is notified a jump is available	Not Implemented
B10	R.1.1.2, R.1.3.3	Normal Piece reaches the other side and is “Kinged”	1. Player selects a normal piece one row away from the opposing players side of the board 2. Player moves piece forward diagonal reaching the last row	Normal piece is converted to a king. Has new movement abilities and a new graphic is used for said piece	As Expected
B11	R.1.2.1	Game State of the board is displayed	1. Observe application GUI	A checkerboard and pieces should be displayed	As Expected
B12	R.1.2.2	Player can	1. Player clicks a piece	Piece should	As Expected

		click to move pieces	2. Player clicks empty square	move if legal move	
B13	R.1.2.3	Display Active Player	1. Observe application GUI	A box should be displayed listing the active player	A Popup is displayed when it is the start of their turn
B14	R.1.3.1	Game state is the same for host code and guest	1. Observe application GUI on Host 2. Observe application GUI on guest	The checkerboard, pieces, and active player should be the same across both systems	As Expected
B15	R.1.3.2	Update Game state when either player makes a move	1. Player makes a move that ends their turn 2. Other player observes application GUI 3. Other player makes their moves until they make a move that ends their turn 4. Original player observes GUI	The checkerboard should update after each turn, and the actions should be reflected by the GUI	As Expected
B16	R.2.1.1	Measure Delay between move and update on screen	1. Player makes a move 2. Player begins stopwatch upon clicking 3. Player ends stopwatch after GUI updates	The screen should updates in less than two seconds	Not Implemented

4.3 Ending the Game

4.3.1 Description

This case covers the steps involved in ending an active game through either forfeit or by a player winning, as well as the resulting options to start a new game or quit to the main menu.

4.3.2 Pre-Conditions

An internet connection, the application must be downloaded, a game is active between the host and the guest.

4.3.3 Scenario

ID	Req	Description	Execution Step	Expected Result	Actual Result
C1	R1.1.9, R1.2.4	Surrender Game	1. Host and guest are connected and playing a game 2. Host or guest selects surrender button	The game immediately finishes and both players are sent to the End Game Screen, with the screen specifying that the user who did not forfeit has won the game.	As Expected
C2	R.1.1.8	Win Game	1. Host and guest are connected and playing a game 2. Host or guest captures the last piece of the opponent	The game immediately finishes and both players are sent to the End Game Screen; The End Game Screen specifies that the player that still had pieces at the end of the game has won.	Works as specified, except it just exits the process and closes the board
C3	R.1.4.7	Quitting the Game	1. Game has been	The game is exited completely,	As Expected

			surrendered or won and the End Game Screen is displaying for both players 2. Either player selects to quit game	dropping the connection between the players. Both players are sent to the Main Menu Screen.	
C4	R.1.4.8	Rematch	1. Game has been surrendered or won and the End Game Screen is displaying for both players 2. Both Players select the rematch button	The game board is reset and a new game is initialized	Not Implemented

5. Appendix

https://docs.google.com/document/d/1tlQl__QgNoCikX2eYpTMEk8zDbBSKc3feafCkaj7tC0/edit?usp=sharing

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

https://www.draw.io/

https://kivy.org/#home