CS 451: Software Engineering Term Project Test Document

Group Members

Phillip Ryan
Daniel Fitzick
Troy Santry
Karishma Changlani

Revision History

Name	Date	Reason for Change	Revision
Phillip Ryan Daniel Fitzick Troy Santry Karishma Changlani	08/10/16	Initial template	0.0

Table of Contents

1. Introduction

The purpose of this document is to describe the testing approaches used while evaluating the functionality and performance of Chekrz program as to meet the requirements outlined in the requirements document. Chekrz is a java application that allows two users to play a game of checkers from remote locations via connecting across the internet.

1.1. Definitions, Acronyms and Abbreviations

Please refer to the Appendix and Glossary sections for any definitions and abbreviations.

1.2. References

The document may feature terms and references which can be found in the preceding requirements and design documents related to Checkerz.

2. Test Environments

The program and associated test cases will be run with the following test environments.

2.1. Environment 1: Windows OS

Machine Name	Windows PC	Interpreter Platform	Java v7
OS and Version	Windows 10	Network	Private Network
Tester Name	Phillip Ryan	Test Date	8-21
New Log	If necessary list log	State	PASS

Machine Name	Windows PC	Interpreter Platform	Java v7
OS and Version	Windows 7	Network	Private Network
Tester Name	Phillip Ryan	Test Date	8-21
New Log	If necessary list log	State	PASS

2.2. Environment 2: Mac OSX El Capitan

Machine Name	Apple Computer	Interpreter Platform	Java v7
OS and Version	MAC OSX	Network	Private Network
Tester Name	Phillip Ryan	Test Date	8-21
New Log	If necessary list log	State	PASS

3. Setup information and Prerequisites

Prior to running the program the following prerequisites must be met.

- The program and associated test cases can be run by launching the program. The program features a GUI and is not meant for interaction through a console or terminal program.
- Program binaries specific to a platform can be launched on that platform as long as a working Java Runtime Environment installed on the system.
- An internet connection (broadband recommended) is required for optimum performance of the program.

4. Test Cases

The purpose of this document is to describe the testing approaches used while evaluating the functionality and performance of Checkerz as to meet the requirements outlined in the requirements document. Checkerz is a Java application that allows two users to play a game of draughts from remote locations via the internet.

4.1. Test Cases 1: Start a game

4.1.1. Description

This case covers the steps to begin a game, both for the user acting as the server and the user acting as a client.

4.1.2. Pre-conditions for this test case

LAN or WLAN connection for two computers on the same network. Java runtime environment installed on both systems.

4.1.3. Scenario

Test Cases					
ID	Req	Description	Execution Step	Expected Result	Actual Result
A1	2.2	Launch application	Click on the checkers.jar file	GUI showing options to host or connect to a game appears	PASS
A2	3.5.1	Begin hosting game	Select the option to host a game on computer α	GUI changes to a checkers board superimposed with a loading icon	PASS
A3	3.5.2	Navigate to the games list	Select the option to connect to a game on computer β	GUI changes to a list of hosts on the current network	PASS
A4	3.5.2	Connect to a game	Select the host α on the computer β	GUI changes to a checkers board, loading icon on computer α disappears	PASS

4.2. Test Cases 2: Game play

4.2.1. Description

This case covers the steps during the game, both for the user acting as the server and the user acting as a client.

4.2.2. Pre-conditions for this test case

The game should have already begun smoothly, i.e. Test Case 1 should have executed smoothly.

4.2.3. Scenario

Test Cases

ID	Req	Description	Execution Step	Expected Result	Actual Result
P1.	3.1.5	Highlight all valid spaces that a normal piece, (that cannot jump) can move to. (No mandatory jump moves are present)	1. Click a piece that can move diagonally one space.	All spaces that the clicked piece is able to move are highlighted.	PASS
P2.	3.1.5	Highlight all valid spaces that a normal piece that can jump can move to. (No mandatory jump moves are present)	1. Click a piece that can jump an opponent's piece.	Only the jump move(s) should be highlighted.	PASS
P3.	3.1.5	When there is a jump move available, don't highlight any non jump moves when a piece is clicked.	1. Click on a piece that can jump, but could have moved without a jump if there wasn't a jump move available.	No non jump moves should be highlighted when the piece is clicked.	PASS
P4.	3.1.2.2	When a normal piece reaches the first row of the opponent's side the piece should turn into a king piece	1. Click on a space to move a normal piece to the first row of the opponent's side.	The normal piece should change into a king piece.	PASS
P5	3.1.3.1	When a king piece, that cannot jump, is clicked highlight all spaces diagonally in front and in back of the piece that the piece can move to.(No mandatory jump moves are present)	1. Click on a king piece.	All spaces that the king piece can move to should be highlighted.	PASS
P6.	3.1.4.4	When a king piece that can jump is	2. Click on a king piece.	All spaces that the king	PASS

		clicked highlight only the jump spaces.		piece can jump to should be highlighted.	
P7.	3.1.2.1	Move normal piece diagonally forward.	1. Click a piece with an empty space diagonally in front of it. 2. Then click on the highlighted empty space diagonally in front of it.	The piece will move there and stay there until moved again or captured. On the opponent's computer the piece will move to the same space.	PASS
P8.	3.1.4.1	Jump an opponent's piece. (single jump)	1. Click a piece that can jump an opponent's piece. 2. Then click on the empty space where the piece would end up after the jump.	The piece will move there and stay there. The opponent's piece will disappear from the board. On the opponent's computer the board will be the same.	PASS
P9.	3.1.4.2	Do a double jump or more.	1. Click a piece that can make a double or triple jump. 2. Click on the highlighted space to complete the (first) jump. 3. Repeat step 2 until you cannot jump any more.	After each jump move the piece will move to the new space and another highlighted space will appear if available. Each jump is replicated on the opponent's computer	PASS

				after we move to each new space. After we jump each piece the opponent's piece will disappear.	

4.3. Test Cases 3: End a Game

4.3.1. Description

This case tests resigning, drawing, and winning a game.

4.3.2. Pre-conditions for this test case

A game between two computers has been started.

4.3.3. Scenario

Test Cases					
ID	Req	Description	Execution Step	Expected Result	Actual Result
C1	3.2.2.2	Resign Button Pressed	Click the resign button.	Game loss is displayed (win for other player)	PASS
C2	3.2.3	Draw	● Play until only two kings are left (one for each player) ● Move each king back and forth without capture	A tie is detected automatically. Both players are alerted of this and the game ends	PASS
C3	3.2.3	Draw button Pressed	Click the Draw button.	The opponent is asked if they wish to call a draw as well. There is a 30 second time out. If the opponent clicks Yes both players are	PASS

				alerted and the game ends If the opponent clicks No the original draw caller is alerted and the game continues If the opponent doesn't decide until time out the game should continue and the original draw called is alerted of the time out.	
C4	3.2.2.1	Win	Capture all of the opponent's pieces.	A win is detected automatically. Game win is displayed (loss for the other player)	PASS

5. Glossary

5.1.1. Checkers game:

The game that is being made.

5.1.2. Piece:

Objects that belong to a player that are moved around the board.

5.1.3. King Piece:

A piece that has moved across the board and reached the opposing team's first row.

5.1.4. Board:

The layout of all the pieces for the players to view and perform actions on.

5.1.5. Move:

An action made by the player to move a piece on the Board.

5.1.6. Jump Move:

A move that includes “capturing” a piece by “jumping”. It steps over an enemy piece and the enemy piece is removed from the Board

5.1.7. Turn:

A sequence of moves made by one player, ending when no further moves are legal.

5.1.8. Draw:

When neither of the players have won and the game reaches a tie state and the game ends.

5.1.9. Resign:

When a player decides that they want to voluntarily end the game and declare themselves as lost

5.1.10. User Datagram Protocol (UDP):

A protocol used for transmitting data that is known for it’s speed but is not very reliable for data loss.

5.1.11. Transmission Control Protocol (TCP):

A protocol used for transmitting data that is known for it’s reliability but lacks speed.

5.1.12. Java Runtime Environment (JRE):

Development environment which will be used for development.

5.1.13. Networking:

Tasks to enable communication between a client and server.

6. References

6.1. CS 451: Software Engineering Term Project Requirements Specification.
6.2. CS 451: Software Engineering Term Project Design Document
6.3. http://cs.uccs.edu/~cs522/pp/pp.html