Software Test Case Document

Chess of Champions
Version 1.0

Created by:
Akshay Sharma
Jeremy Bennett
Austin Herring
Josh Weinstein

Revision History

Name	Date	Reason for Change	Version	Editor
Jeremy Bennett, Austin Herring, Akshay Sharma, Josh Weinstein	February 26, 2015	Initial Version	1.0	Jeremy Bennett, Austin Herring, Akshay Sharma, Josh Weinstein

Table of Contents

1. Introduction

The purpose of this document is to describe the testing approaches used while evaluating the functionality and performance of *Chess of Champions* program as to meet the requirements outlined in the requirements document. *Chess of Champions* is a Java application that allows two users to play a game of chess from remote locations via connecting to a web server.

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 *Chess of Champions* program.

2. Testing Environments

The program and associated test cases have been run within the following test environments.

1.1 Environment 1: Windows 8.1

Machine Name	Windows PC	DB Directory	N/A		
OS and Version	Windows 8.1; 8 GB RAM; 1 TB HDD	Interpreter Platform	Java v7 (update 76)	Client Server /Back-End	Tux (Java)
Tester Name	Akshay Sharma	Test Date	02-21-2015		
New Log	If necessary, list the new log after tests have been run.			State	**PASS**

1.2 Environment 2: Ubuntu 14.04 LTS

Machine Name	Linux VM	DB Directory	N/A		
OS and Version	Ubuntu 14.04 LTS; 8 GB RAM; 1 TB HDD	Interpreter Platform	Java v7 (update 76)	Client Server /Back-End	Tux (Java)
Tester Name	Akshay Sharma	Test Date	02-21-2015		
New Log	If necessary, list the new log after tests have been run.			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.

- Tux server along with the associated chess server program must be up and functioning properly, without any glitches.

4. Test Cases

The purpose of this document is to describe the testing approaches used while evaluating the functionality and performance of *Chess of Champions* program as to meet the requirements outlined in the requirements document. *Chess of Champions* is a Java application that allows two users to play a game of chess from remote locations via connecting to a web server.

4.1 Test Cases 1: Initialize Game

4.1.1 Description

The case consists of covering the steps required to host a game on a computer.

4.1.2 Pre-conditions for this test case

An internet connection, Java runtime environment installed on the system, Tux (server) is up and running.

4.1.3 Scenario

Test Case					
ID	**Req**	**Description**	**Execution Steps**	**Expected Result**	**Actual Result**
A1.	1.4.1	Launch App	1. Launch application by clicking .jar file.	GUI window launches showing plain chessboard	
A2.	1.2, 1.3	Connect Modal	1. Click ‘Connect’	Connect modal	

			button	window launches	
A3.	1.2.1	Host Game - Successful	1. Click ‘Connect’ button 2. Click ‘Host’ radio button 3. Click OK.	Application waits while a game ID is generated by the server; program then displays game ID	
A4.	3.2	Host Game - Too Many Waiting Clients	1. Click ‘Connect’ button 2. Click ‘Host’ radio button 3. Click OK	Application waits while server processes; server informs application too many users waiting and application informs user	
A5.	1.3.1, 1.3.3	Join Game - Successful	1. Click ‘Connect’ button 2. Click ‘Join’ radio button. 3. Enter game ID of game a host has created 4. Click OK	A connection is established and chessboard populated with pieces	
A6.	1.3.2	Join Game - Invalid ID	1. Click ‘Connect’ button 2. Click ‘Join’ radio button 3. Enter game ID of a game no host has created 4. Click OK	Application waits while server processes; server informs application ID is invalid and application informs user	

4.2 Test Cases 2: Play the Game

4.2.1 Description

The case consists of covering (testing) the steps involved in mutual play of the game by the host and the opponent. The test case essentially validates if the logic behind the moves has been correctly implemented and follows the rules of the game as set forth by *World Chess Organization*.

4.2.2 Pre-conditions for this test case

An internet connection, Java runtime environment installed on the system, Tux (server) is up and running, a game is already setup and active between the host and the opponent.

4.2.3 Scenario

Test Case					
ID	**Req**	**Description**	**Execution Steps**	**Expected Result**	**Actual Result**
B1.	1.5.4	Pawn forward movement	1. Find pawn with one empty space in front. 2. Move pawn forward one step to empty square. 3. Find pawn with no movement yet and no pieces in two spaces ahead. 4. Move pawn forward two spaces.	Pawn moves from source square and stays on the new destination square	

B2.	1.5.4	Pawn backward movement	1. Find pawn with space behind empty. 2. Move pawn backward.	Pawn does not move as it is an illegal move.	
B3.	1.5.4	Pawn horizontal movement	1. Find pawn with left space and/or right space clear. 2. Move pawn left/right.	Pawn does not move as it is an illegal move.	
B4.	1.5.4	Pawn capture	1. Move pawn such that it can capture opposing piece diagonally. 2. Move pawn diagonally forward on top of other piece	Pawn replaces the piece successfully	
B5.	1.5.4	Pawn forward movement over a piece	1. Move piece infront of a pawn that has not moved. 2. Move pawn over a piece in forward direction (straight)	Pawn does not move as it is an illegal move.	
B6.	1.5.4	Pawn multiple two jumps	1. Move pawn legally 2 forward spaces on opening move. 2. Attempt to move 2 spaces forward again.	Pawn does not move 2 spaces forward the second time as it's illegal.	
B7.	1.5.7	*En Passant* Capture	1. Move pawn in an "en passant" legal maneuver (see glossary)	Pawn moves successfully.	
B8.	1.5.7	Illegal En Passant - Timing	1. Put pawn in legal En Passant maneuver.	Pawn does not move as it is an illegal move.	

			2. Wait 1 additional turn before applying En Passant while pawns don't move.		
B9.	1.5.7	Illegal En Passant - Mid Board	1. Set up opposing and host's pawns in "En Passant" formation but in middle of the board. 2. Attempt "En Passant" style capture when opposing pawn has moved but is not opposing pawn's initial movement.	Pawn does not capture in "En Passant" style as it's an illegal move.	
B10.	1.5.6	Pawn Promotion	1. Clear path for pawn to move to opposite side of the board. 2. Have host's pawn advance to opposite side of the board	A screen appears that allows the host to select a new piece. This piece replaces the pawn.	
B11.	1.5.4	Illegal Knight forward move	1. Move knight forward one to 'n' steps	Does not move as it is an illegal move.	
B12.	1.5.4	Legal Knight move with vertical L shape	1. Move knight on an L path (2 squares horizontal; 1 square vertical or vice-versa)	Knight moves successfully and stays at the destination square.	
B13.	1.5.4	Legal Knight move with horizontal L shape	1. Move knight on an L path (2 squares horizontal; 1 square vertical or vice-versa) provided the square is	Knight moves successfully, replaces the existing piece and stays at the	

			pre-occupied by another piece	destination square.	
B14.	1.5.4	Legal Rook horizontal/vertical movement	1. Move rook forward/backward/left/right one to ‘n’ steps (either horizontally or vertically)	Rook moves successfully and stays at the destination square replacing any pre-existing pieces of the opposite color as rook.	
B15.	1.5.4	Illegal Rook diagonal move	1. Locate rook with unoccupied diagonal space. 2. Move rook diagonally in any direction	Does not move as it is an illegal move.	
B16.	1.5.4	Illegal Bishop horizontal/vertical movement	1. Locate bishop with unoccupied horizontal space. 2. Move bishop forward/backward/left/right one to ‘n’ steps (either horizontally or vertically)	Does not move as it is an illegal move.	
B17.	1.5.4	Legal Bishop movement	1. Locate Bishop with unoccupied diagonal space. 2. Move Bishop diagonally.	Bishop moves successfully and stays at the destination square replacing any pre-existing pieces of the opposite color.	
B18.	1.5.4	Legal Queen Movement	1. Locate Queen with unoccupied horizontal / vertical spaces.	Queen moves successfully and stays at the destination	

			2. Move Queen horizontally 3. Wait till next turn, move Queen vertically.	square replacing any pre-existing pieces of the opposite color as rook.	
B19.	1.5.4	Illegal Queen Jump	1. Locate Queen. 2. Attempt to move Queen by jumping over one or more multiple pieces to an unoccupied space or occupied by opposing player.	Does not move as it is illegal move.	
B20.	1.5.4	Illegal Queen Movement	1. Locate host's Queen. 2. Move Queen in movement that is not horizontal / vertical.	Does not move as it is illegal move.	
B21.	1.5.4	Legal King movement	1. Locate host's King with valid unoccupied adjacent space. 2. Move King in any direction 'one' step where King is not in check.	King moves successfully and stays at the destination square replacing any pre-existing pieces of the opposite color as rook.	
B22.	1.5.4	King Movement	1. Locate host's king. 2. Move King in any direction >1 steps	Does not move as it's an illegal move.	
B23.	1.5.5	Correct Castle Left	1. Move pieces between king and left rook so path is clear. 2. Move King two spaces to the left.	The king should now be two spaces to the left and the left rook to the immediate right of the rook.	

B24.	1.5.5	Correct Castle Right	1. Move pieces between king and right rook so path is clear. 2. Move King two spaces to the right.	The king should now be two spaces to the right and the right rook to the immediate right of the rook.	
B25.	1.5.5	Incorrect Castle Left, Rook movement	1. Move pieces between king and left rook so path is clear. 2. Move left rook and then move back to original position. 3. Move King two spaces to the left.	Should not permit the player to castle and inform invalid move.	
B26.	1.5.5	Incorrect Castle Right, Rook movement	1. Move pieces between king and right rook so path is clear. 2. Move right rook and then move back to original position. 3. Move King two spaces to the right.	Should not permit the player to castle and inform invalid move.	
B27.	1.5.5	Incorrect Castle, king has moved	1. Move pieces between king and rook so path is clear. 2. Move king once and then back to original position. Attempt to castle with rook.	Should not permit the King to castle as the king has moved and inform the user of invalid move.	
B28.	1.5.5	Incorrect Castle, king moves through check	1. Move pieces between King and rook so path is clear. 2. Have opposing rook in column that	Should not permit player to castle as you cannot move the king through check.	

			the king moves through in castling. Attempt castle.		
B29.	1.5.5	Incorrect Castle, blocking path	1. Locate unmoved king and rook for host. 2. Attempt to castle with pieces between rook and King.	Should not permit player to castle as pieces are in between castle/rook.	
B30.	1.5.4	Move piece blocking King from being in Check.	1. Move opposing piece to put king in check if a barrier piece does not exist. 2. Remove a piece serving as a barrier between king and opponent piece causing a “possible” check	Does not move as to prevent check/checkmate.	
B31.	1.5.4	Move King into check	1. Locate host King. 2. Attempt to move King into check.	Does not move as it’s an illegal move.	
B32.	1.5.4	King in Check	1. Have player 2 put player 1’s King in check. 2. Attempt to move a piece that keeps King in check.	Does not allow any movement that keeps King in check.	
B33.	1.5.4	Put Opponent in Check	1. Move a piece as to put the opponent’s King in check.	Opponent should be notified of check.	
B34.	1.5.2	Move own piece on top of another Piece	1. Move any piece over any other same colored piece (with an intent to replace it)	Does not move as it is an illegal move.	

B35.	1.6.1 , 1.6.2	Chat Log	1. Enter message into chat window. 2. Hit send	Message should appear on host and opponent's chat window.	
B36.	1.7.1	Log Move, Normal Move	1. Move host's piece to any square. 2. Repeat for one piece of each type.	In move log it should list piece and where piece moved from/to	
B37.	1.7.2	Castle Log	1. Move pieces in between King and rook. 2. Perform legal Castle	In move log two moves should be listed, one for King and one for Rook.	
B38	1.7.3	Pawn Promotion Log	1. Move pawn to opposite side of board. 2. Promote pawn to any listed piece	In the move log it should note that the pawn has been promoted.	

4.3 Test Case 3: Ending the game

4.3.1 Description

The case consists of covering (testing) the steps involved in resigning, drawing, or winning a game from an active gaming session.

4.3.2 Pre-conditions for this test case

An internet connection, Java runtime environment installed on the system, Tux (server) is up and running, a game is already setup and active between the host and the opponent.

4.3.3 Scenario

Test Cases

ID	Req	Description	Execution Steps	Expected Result	Actual Result
C1.	1.1.1	Resign game	1. Host a session with host player and receive game ID. 2. Opponent joins session of client 1. 3. Host hits the resign button.	A “resigned” message is displayed to the user and game ends	
C2.	1.4.6	Win game	1. Host session between two players. 2. Play game and move client host’s King into checkmate.	A “You win” message should display for opponent and host should have a “You lose” game message. No more moves are permitted on either side.	
C3.	1.5.3	Lose by Timeout	1. Host session between two players with time limit of 30 minutes. 2. Allow host’s time to run to 0.	A “You Lose by Timeout” should appear for host and the opponent should be notified of his victory.	
C4.	1.4.7	Stalemate	1. Host a session with host player and opponent. 2. Play game and place client 1’s King in situation where is not in check but has	A “Stalemate - Draw” message should appear for both host and opponent players and no more moves should be permitted for either side.	

			no legal moves.		

5. Appendix

5.1 Glossary

En Passant - A special case pawn move where a pawn can capture an opposing pawn on it's opening move. See http://en.wikipedia.org/wiki/En_passant

Guest/Opponent - The *other* player willing to join the game using the ID code provided by the Host player.

Host - The player hosting the game. The player who launches chess application first and receives the unique game ID code from the application.

Modal - A pop-up window that is not part of the main screen interface. The user must complete interaction with the modal before he can go back to interaction with the active screen.

Tux - Servers available to computer science students at Drexel University. Tux servers will be used for running game sessions.

World Chess Organization - Organization that sets the rules of chess for international play Refer to their website (http://www.fide.com/component/handbook/?id=124&view=article) for the complete set of rules which are the same rules this application should follow.