Chromecast Checkers

Testing Plan

Authors: Zach Almon, Matt Dunbar, Omid Omidi

Unit/Function Testing:

Test Case	**Input**	**Expected Output**
New Game	Someone initiates the Checkers Game	Game starts and waits for another player to connect
New Game	Board is created	8 x 8 board is created with top and bottom 3 rows filled with 12 pieces each, correctly spaced and colored
Player 1 Turn	Game waits for player 1 input to be sent from Android	Upon correct choice the piece moves to players desired spot
Player 1 Incorrect Input	Game waits for player 1 input to be sent from Android	Upon an incorrect choice from player 1, a message is sent back to android and game waits for another choice
Player 2 Turn	Game waits for player 2 input to be sent from Android	Upon correct choice the piece moves to plays desired spot
Player 2 Incorrect Input	Game waits for player 2 input to be sent from Android	Upon an incorrect choice from player 1, a message is sent back to android and game waits for another choice
Player Jumps a piece	Player's piece jumps another piece	The jumping piece moves to its final destination, and the jumped piece(s) are removed from the board
Player Double Jump	If the player can Double Jump and chooses to Double jump [Includes multiple jumps]	The piece moves to players desired spot, and jumped pieces are removed from the board
Piece reaches other side	ONLY When a player's piece makes it to the other side it becomes king [Function to replace piece with King]	The piece moves to players desired spot, and The piece becomes a "King"
King Piece is chosen to move	Kings are allowed more directions to move than regular pieces. [Function to check if piece is King]	Kings will be allowed to move in more directions than regular pieces
End of Game	End of game is detected	The game is ended appropriately

Unit/Function Testing (continued):

The Unit/Function Testing is the JavaScript phase of testing.

To complete to Unit/Function testing we implemented a method to play the game without input from the android applications. We created a small set of clickable arrows as buttons that will act as the "controllers" for the purposes of testing the JavaScript checkers code and functions. This testing feature will be taken out and unusable in the final product. We tested the program as a single JavaScript program to make sure that the game works as intended. We have made sure that the clickable arrow buttons are easily translated to the Android JSON messages that call the same functions to make movements. There is also the test function to make sure the input is valid, otherwise a message is sent that the input was wrong and to try again.

We each played multiple games under this testing version with various people to eliminate any bugs. We have tested the functions that check jumps, this includes double or multi-jumps, that checks when a piece reaches the other side and is changed to a King, and that checks if a piece chosen to move is a King than it is allowed to make a wider range of moves.

The last portion we tested was how the game ends. There is a function that after each player’s turn checks the state of the game. If one of the two players has no pieces remaining the game ends, with the one who has pieces winning. Also, if a player cannot make any moves they lose. There is a function that runs before every turn, if a player cannot move any piece the end of game is detected and they lose.

After we tested this JavaScript portion, we took the clickable arrows out and made them hidden from user view. We added back in the functions to communicate with the android applications. The messages received from the applications will be parsed out and the function to check if the input was correct or legal was checked. If the input was not legal we send back an error code JSON message and then wait for another input. Once we get legal input, the appropriate tested functions will be called.

System/Integration Testing:

Test Case	**Input**	**Expected Output**
Player 1 Disconnects	Android Phones disconnect	Game waits until player reconnects or the player forfeits after a set time
Player 2 Disconnects	Android Phones disconnect	Game waits until player reconnects or the player forfeits after a set time
Player 1 Reconnects	Android Phone Reconnects	Game resumes where it was left off at
Player 2 Reconnects	Android Phone Reconnects	Game resumes where it was left off at
Player 1's Turn	Player 2 Cannot Input	Player 2's android screen is greyed out and will not accept input
Player 2's Turn	Player 1 Cannot Input	Player 1's android screen is greyed out and will not accept input
Player Clicks on Any Button	A Button is pressed	A Correctly Formed JSON Message for that button is formed
Player Clicks on the Left button	Left button is pressed	1. JSON Message is sent to Chromecast 2. The JSON Message is parsed by the Chromecast 3. If the player cannot move there a response message is sent back with Error 4. Player will then have to choose again and the process repeats 5. If the choice is good the Chromecast sends back a message that it is no longer that players turn
Player Clicks on the Right button	Right button is pressed	
Player Clicks on the Select button	Select button is pressed	
Player Clicks on the Up button	Up button is pressed	
Player Clicks on the Down button	Down button is pressed	
Player Clicks on the Back button	Back button is pressed	
Player Incorrect Input	JavaScript sends back an Error JSON Message	When an Error JSON Message is received back the application repeats for new user input while also displaying an Error message that the previous input was not valid

System/Integration Testing (continued):

The System/Integration Testing is the Android phase of testing.

Unfortunately for the Android Application there was not a simple thing we could do to test the different individual functions. Fortunately, the Android Application is pretty simple. The simple UI only has a few buttons for input, which send JSON messages to the server, and text. Along with the buttons functions there are also functions for when it is not that player's turn and a function to handle return JSON messages.

The first thing we tested when we connected the Android Applications to the JavaScript server was to test what happens upon disconnect. The applications attempt to reconnect. If there is no reconnection after a set time limit that player forfeits. If both players disconnect at the same time and do not reconnect the JavaScript program will display that both players have forfeited and exit the game. The JavaScript program is always listening for the applications to reconnect, while the applications are programmed to always try to connect.

After a player gives a valid input the JavaScript will send back a success message. When the application receives this message it will grey out the UI so that no input will be accepted. When this happens the JavaScript will also send a message to the opposite player unlocking their screen so that they will be able to input for their turn. This then repeats as players input valid moves.

When buttons are pressed a function switches based on which buttons were pressed. This function then creates a JSON message based on what button was pressed and what the player wants to do and sends it back to the JavaScript program. The application will be listening for a reply message, if this reply is an error message than the application will repeat itself waiting for new user input. An error message/box will be displayed letting the player know that the previous input was incorrect.

User Testing:

Test Case	**Input**	**Expected Output**
Game Plays Smoothly	Input from android to move pieces	Game smoothly and quickly processes moves and displays the moves on the board
Everything is Readable from a distance	Any words on the Chromecast screen is readable in its size and font	Readable Text
Messages are Readable on the Android Devices	Any messages that need to be displayed on the screen are readable and stay long enough to be read	Readable Messages
Text is Readable on the Android Devices	Text and Button Text are readable on the Android screen	Readable text on the Android screen, users have no problems differentiating buttons
Android App works on Android 4.0.4 and above	App runs on devices 4.0.4 and above	App runs and UI is the same

User Testing is the last phase of testing. This testing is to ensure that the UI on both the JavaScript/Chromecast display and the Android Application are readable and everything is played smoothly. An example would be, a new game starts without error, there are no display glitches, the Android Applications connect and stay connected, the input communications happen timely, the board is updated quickly and smoothly, the application UI is locked and unlocked appropriately, and the JavaScript detects the end of the game correctly and displays the winner before finally closing out.