**Integration Testing**

UI (Web App) and Node Server

**Communication from UI (Web App) to Node Server**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Actions** | **Expected Result** | **Pass (Y/N)** |
| Access UI with Node Server running | Access url: localhost:3000 with web browser | Bring you to ‘home’ page of UI | Y |
| Communication from UI to Node server- attempt to log in | Go to localhost:3000  Attempt to log in using username ‘neil’ | Node Server should receive log in request.  Data should be a JSON object include: socket, username logged in with. | Y |
| Communication from UI to Node server- attempt to fold | Go to localhost:3000  Attempt to log in using username ‘neil’  Attempt to fold the hand | Node Server should receive action message.  Data received should be JSON object including: socket, username, round of play and action= ‘fold’.  The opponent model record for the user name should be updated in MongoDB | Y |
| Communication from UI to Node server- attempt to check | Go to localhost:3000  Attempt to log in using username ‘neil’  Attempt to check the hand | Node Server should receive action message.  Data should include: socket, username,cards, board cards, round of play and action= ‘check’.  The opponent model for user name should be retrieved from MongoDB through Mongoose | Y |
| Communication from UI to Node server- attempt to call | Go to localhost:3000  Attempt to log in using username ‘neil’  Attempt to call the hand | Node Server should receive action message.  Data should include: socket, username, cards, board cards,round of play and action= ‘call’.  The opponent model for user name should be retrieved from MongoDB through Mongoose | Y |
| Communication from UI to Node server- attempt to bet | Go to localhost:3000  Attempt to log in using username ‘neil’  Attempt to bet the hand with 300 chips | Node Server should receive action message.  Data should include: socket, username,cards, board cards, round of play, action= ‘bet’ and amountBet:300.  The opponent model for user name should be retrieved from MongoDB through Mongoose | Y |
| Communication from UI to Node server- attempt to raise | Go to localhost:3000  Attempt to log in using username ‘neil’  Attempt to raise the hand with 300 chips | Node Server should receive action message.  Data should include JSON object with: socket, username, cards, board cards, round of play, action= ‘raise’ and amountBet:300.  The opponent model for user name should be retrieved from MongoDB through Mongoose | Y |
| Communication from UI to Node Server - showdown | Go to localhost:3000  Attempt to log in using username ‘neil’  Go to showdown. | Node Server should receive JSON object to evaluate the hands:  Hole Cards, User Cards, Board Cards, Username, socket | Y |

**Communication from Node Server to UI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Actions** | **Expected Result** | **Pass (Y/N)** |
| Accepting log in | Send login accepted socketIO message to specific socket | UI moved on from login page to main gameplay screen | Y |
| Rejecting log in | Send login rejected socketIO message to specific socket | Message on UI appears informing that the log in has been rejected | Y |
| Sending AI action - fold | Send AI action ‘fold’ socketIO message to specific socket | The hand is folded by the AI. | Y |
| Sending AI action - check | Send AI action ‘check’ socketIO message to specific socket | The hand is checked by the AI. | Y |
| Sending AI action - fold | Send AI action ‘call’ socketIO message to specific socket | The hand is called by the AI. | Y |
| Sending AI action - bet | Send AI action ‘bet’ (bet1, bet2, bet3) socketIO message to specific socket | The hand is bet by the AI. The amount is determined as follows:   * Bet1 = 1\* ¾ current pot * Bet2 = 2 \* ¾ current pot * Bet2 = 3 \* ¾ current pot   OR if the AI stack size is smaller, the value of the AI stack | Y |
| Sending AI action - raise | Send AI action ‘raise’ (raise1, raise2, raise3) socketIO message to specific socket | The hand is bet by the AI. The amount is determined as follows:   * Raise1 = 1\* ¾ current pot * Raise2 = 2 \* ¾ current pot * Raise3 = 3 \* ¾ current pot   PLUS the amount to call the bet  OR if the AI stack size is smaller, the value of the AI stack | Y |
| Sending winner at showdown | Send socketIO message informing user if they have won at showdown to specific socket | Depending on the winner at showdown, the winner of the hand should be displayed at showdown. The winning hand type should also be displayed. | Y |

Java Server to Node Server

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Actions** | **Expected Result** | **Pass (Y/N)** |
| Receiving action response from Java Server | Java Server sends message with following info:  Username, Action | Node Server receives the message (prints to console). | Y |

Node Server to Java Server

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Actions** | **Expected Result** | **Pass (Y/N)** |
| Receiving inputs from Node Server | Node Server sends message with following inputs:  Username, cards, board cards, round, opponent model, previous action, amount bet | Java Server receives this (print to console) and creates new thread to deal with determining an action. | Y |
| Receiving malformed inputs | Node server sends malformed inputs | Java Server receives this (print to console), creates new thread and deals with it gracefully - sends error message back to to node server | N |

**Web Browser Testing**

The web app has been tested and validated in the following web browsers/versions.

|  |  |
| --- | --- |
| **Web Browser** | **Version** |
| Google Chrome | 58 |
| Mozilla Firefox | 49 |
| Microsoft Edge | 38 |
| Internet Explorer | 11 |