Testing - Appendix A

Test Plan

The following testing activities have been carried out as part of this project

- Unit Testing
- Integration Testing
- Acceptance Testing
- User Evaluation Testing

Unit Testing

Automated unit testing should be completed on every module/component developed as part of the project, unless given reasons as to why not. Unit tests should be run with a tool that generates a report detailing code coverage numbers (e.g. class coverage, method coverage, line coverage) to be used as an indicator whether the unit tests are helpful or not (e.g. to help avoid having two tests doing the exact same thing). Unit tests should test ideally test one specific thing or unit (e.g. method in a class) and use mock data where appropriate.
Where possible, the unit tests should have an error message detailing the reason as to why it would fail. These messages will serve as part of unit test documentation.

Integration Testing

The purpose of integration tests within this project is to verify and validate the communications between the different components of the project (i.e. web app, NodeJS server, Java server).

Acceptance Testing

Acceptance testing should detail every scenario a user could find themselves, as per the use cases defined in the functional specification and use cases thought of during the course of the development of the project.
An ideal way to document these test cases would be to take inspiration from the 'Gherkin' syntax (https://github.com/cucumber/cucumber/wiki/Gherkin) - Gherkin is used to help describe software's behaviour in a readable way. An example test case format could have (Gherkin syntax in brackets):

- Preconditions (Given)
- Action required (When)
- Expected Results (Then)

As part of this activity, it should also be noted as to what browsers it has been tested in, along with their version number.

User Evaluation

User evaluation should be completed in a way that can quantify the usability of the system.

A proposed way to carry this out would be to follow the ISO 9126-4 Approach.
This approach focuses on three main 'usability metrics' that quantify the usability of the system:

- Effectiveness
- Efficiency
- Satisfaction

Effectiveness
Firstly, a set of tasks that a user must complete must be defined. The tasks must be defined to correlate to the use cases of the systems - how a user would use the system.

Effectiveness can be measured by observing how many tasks a user completes successfully. A way to show the effectiveness could be represent as a percentage of tasks complete and could be calculated like this:
(Number of successfully completed tasks / Number of total tasks) * 100.
These numbers collected from multiple users could then be averaged to show a value for effectiveness of the system.

Efficiency
The same set of tasks could be used as was used to measure effectiveness, and can be completed concurrently.

Efficiency can be measured in terms of the time it takes to complete a task. It is important to take into account unsuccessfully completed tasks - specifically the time it took before a user gave up on the task. Efficiency per user for each task could be measured as follows:

Result of Task/Time Taken(in seconds).
where Result of Task equals 1 if task successfully complete, else 0.

These numbers can then be averaged across the corresponding tasks for all users. This will express a result of efficiency in terms of 'goals per second'. The higher the number, the better the efficiency.

Satisfaction
After users attempt to complete each task the task list, they should be given a questionnaire assessing the ease/difficulty of the task. A 'single ease question' which represents a 10 point scale(eg. 1=very easy, 10=very difficult) which will allow users to define how challenging they found the task.

There should also be a questionnaire given to each test user at the end of test session. The questionnaire can be used to measure the ease of use of the system. A suggestion would be use the 'System Usability Scale' (https://measuringu.com/sus/) to measure the overall satisfaction of the user.

These questionnaires will help pinpoint areas in need of improvement in terms of UI/UX. Users should also be able to describe any comments they have on the system, as they may bring up ideas on how to improve the overall UI/UX.

Unit Testing

AI_Agent

The main focus of unit tests were on the main AI_Agent module. The primary reason for this is to monitor and limit the impact had by any changes made within this module (e.g. supplying different parameters etc.). The tests were run using the JUnit framework.

All unit tests have an assert statement as part its test. The assert statement describes the reason for failure of a test. It also serves as documentation to the test.

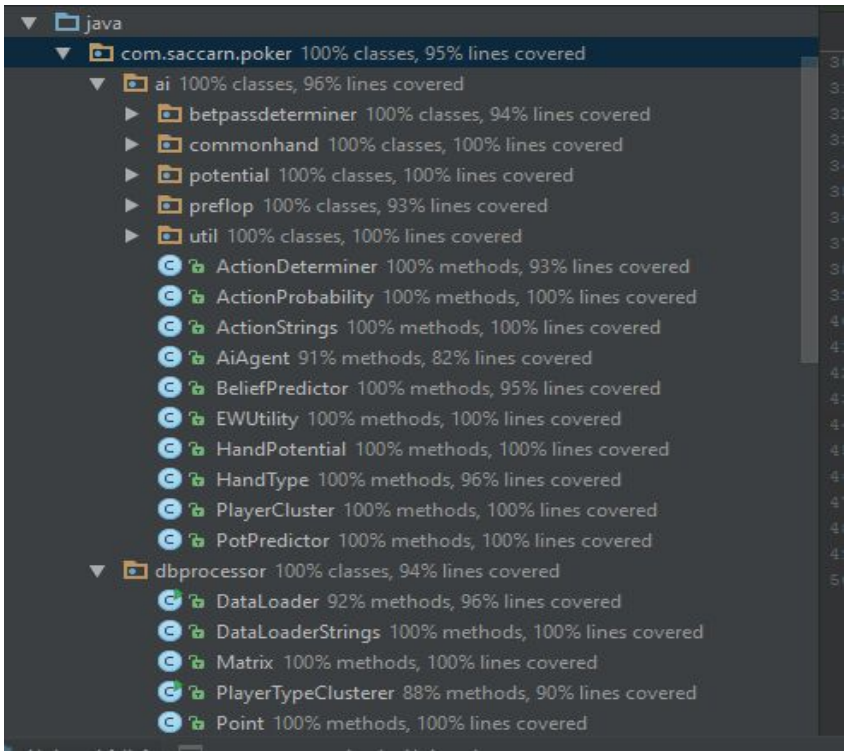
Code coverage was generated using Intellij surefire reporting tool. This tool also generates HTML reports documenting which individual lines were covered as part of the unit tests.

Results of Unit Tests

Tests were run using the Intellij JUnit test runner and Maven:

Num Tests: 147. **Num Tests Passed**: 147. **Percentage of num tests passed**: 100%.

Code Coverage Results



Unit Tests for UI
Currently, there are no unit tests for the UI. This has been attempted to been compensated by a high number of acceptance tests.

Unit Tests for PokerHandsProcessor
Currently there is a very low number of unit tests for this module. This is because of the high dependency of parsing files. This module should be refactored in order to use more generic objects (such as InputStream instead of File), in order to make it easier to mock test data for unit testing. There are currently 16 unit test for this module, all of which passing. It has an overall line coverage of 23%.

Integration Testing

The integration test cases are manual tests. As part future contributions to this project, these could be automated. The test cases describe the communications from one module to another

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	Y

		should be retrieved from MongoDB through Mongoose	
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	Java Server sends message with following	Node Server receives the message (prints to console).	Y

Server	info: Username, Action		

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

User Acceptance Tests

There is a high amount of user acceptance tests due to lack of UI unit tests. The test cases describe the functionality of the web app itself regarding how a user would use it.

Feature: Login

Scenario	Preconditions	Action	Result	Pass
User attempts to log in with valid username	User types valid username into username input box.	User clicks enter or presses enter on keyboard	User should be brought to a game screen where a game has started.	**Y**
User attempts to log in with no user name	- User has typed not typed a username into input box.	User clicks enter or presses enter on keyboard	A modal screen should appear informing the user that they have submitted an invalid username and prevent them from logging in.	**Y**

User attempts to log in with a username that is already logged in.	- User types valid user name that is already logged in to the web app.	User clicks enter or presses enter on keyboard	A modal screen should appear informing the user that they have submitted a username that is already logged in and prevent them from logging in.	**Y**
User attempts to log in with a user name with ':' in it	- User types user name containing the colon character	User clicks enter or presses enter on keyboard	A modal screen should appear informing the user that they have submitted a username that is invalid and prevent them from logging in.	

Feature: Initial Game State

Scenario	Preconditions	Action	Result	Pass
A player should have the dealer button	- User should be succesfully logged in. - User is on the main game screen	**N/A**	One of either the A.I. player or the user should have dealer button.	**Y**
Players should have equal initial stack sizes	- User should be succesfully logged in. - User is on the main game screen	**N/A**	Both the A.I player and the user must have the same number of chips in each of their stacks.	**Y**
User should be able to see their cards	- User should be succesfully logged in. - User is on the main game screen	**N/A**	The user should be able to see their cards clearly. The user should also only be able to see the back of the A.I.'s cards.	**Y**
User should not be able to see any	- User should be succesfully logged in.	**N/A**	The user should not be able to see any	**Y**

board cards	- User is on the main game screen		board/community cards.	
User bet options when user has the dealer button	- User should be succesfully logged in. - User is on the main game screen - User has the dealer button - User bet options have been enabled	**N/A**	The user should be given the option to: - Raise - Bet - Fold	
User bet options	- User should be succesfully logged in. - User is on the main game screen - User bet options should be unclickable and look disabled	User waits for a few seconds	User bet options should be clickable and look enabled.	
User bet options when A.I. has dealer button	- User should be succesfully logged in. - User is on the main game screen - User has the dealer button - User bet options have been enabled	**N/A**	The user should be given the option to: - Raise - Bet - Fold	
Blinds have been correctly placed when the user has the dealer button	- User should be succesfully logged in. - User is on the main game screen - User has the dealer button	**N/A**	The user should have placed half of the the big blind as contribution to the pot. The A.I should have placed all of the big blind as their contribution to the pot. The pot should be made up of the total contributions by both players	
Blinds have been	- User should be	**N/A**	The AI should have	

correctly placed when the A.I has the dealer button	succesfully logged in. - User is on the main game screen - A.I. has the dealer button		placed half of the the big blind as contribution to the pot. The user should have placed all of the big blind as their contribution to the pot. The pot should be made up of the total contributions by both players	

Feature: Preflop

Pre conditions for all Preflop scenarios:

- User should be succesfully logged in.
- User is on the main game screen

Scenario	Preconditions	Action	Result	Pass
No community cards at any point of pre flop stage			No community cards should appear at any point at this stage regardless action take by user	
User bet options when user has dealer button	- User has the dealer button - User bet options have been enabled	**N/A**	The user should be given the option to: - Raise - Bet - Fold	
User has dealer button and carries out first action of hand - call	- User has dealer button - User has enough chips to call	User clicks action 'Call'	The pot should be correctly calculated to account for the 'called' contribution towards the pot. The A.I should carry out one of the following actions: - Call - Raise	

			- Fold	
User has dealer button and carries out first action of hand - fold	- User has dealer button	User clicks action 'Fold'	The pot should be added to the A.I. chip stack. A new hand should be dealt out.	
User has dealer button and carries out first action of hand - raise	- User has dealer button - User has enough chips to raise	User inputs amount to raise by and clicks action 'Raise'	The raise contribution should be added to the user contribution and to the total pot. The A.I should carry out one of the following actions: - Call - Raise - Fold	
A.I has dealer button and carries out first action of hand - call	- A.I. has dealer button - A.I. has enough chips to call	A.I. carries out action 'Call'	The amount to call by should be added to the AI contribution and the total pot. The user should be given the option to - Bet - Check - Fold	
A.I has dealer button and carries out first action of hand - raise	- A.I has dealer button - A.I has enough to chips to raise	A.I. carries out action 'Raise'	The amount raised, plus the amount needed to call the bet should be added to A.I. contribution and to the total pot. The user should be given the option to - Call - Raise - Fold	
A.I has dealer button and carries out first action of hand - fold	- User has dealer button	A.I carries out action to 'Fold'	The pot should be added to the user chip stack. A new hand should be dealt out.	
AI checks, followed	The previous AI	The user checks	The hand should move	

by User checks	action is Check		on to the flop stage	
User checks, followed by AI checking	The previous user action is Check	The AI checks	The hand should move to the flop stage	
User raises, follwed by AI calling	The previous user action is raise	The AI calls	The hand should move to the flop stage	
AI raises, followed by user calling	The previous user action is raise	The AI call	The hand should move to the flop stage	
User bets, followed by AI checking	The previous user action is bet	The AI calls	The hand should move to the flop stage	
AI bets, followed by AI calls	The previous user action is bet	The AI calls	The hand should move to the flop stage	

Feature: Flop

Precondition for all Flop scenarios:

- Preflop completed

Scenario	Preconditions	Action	Result	Pass
Community cards			Should be three three community cards visible to the user	
User has the dealer button, no action completed	User has dealer button		User has option to - Check - Bet - Raise	
A.I has dealer button, no action completed	A.I. has dealer button		User does not have the option to carry out any action until A.I carries out one of following actions: - Check - Bet - Raise	
AI checks, followed	The previous AI	The user checks	The hand should move	

by User checks	action is Check		on to the turn stage	
User checks, followed by AI checking	The previous user action is Check	The AI checks	The hand should move to the turn stage	
User raises, follwed by AI calling	The previous user action is raise	The AI calls	The hand should move to the turn stage	
AI raises, followed by user calling	The previous user action is raise	The AI call	The hand should move to the turn stage	
User bets, followed by AI checking	The previous user action is bet	The AI calls	The hand should move to the turn stage	
AI bets, followed by AI calls	The previous user action is bet	The AI calls	The hand should move to the turn stage	
A.I carries out action - fold		A.I carries out action to 'Fold'	The pot should be added to the user chip stack. A new hand should be dealt out.	
User carries out action - fold		User carries out action to 'Fold'	The pot should be added to the A.I. chip stack. A new hand should be dealt out.	

Feature: Turn

Precondition for all Turn scenarios:
- Flop completed

Scenario	Preconditions	Action	Result	Pass
Community cards			Should be four community cards visible to the user	
User has the dealer button, no action completed	User has dealer button		User has option to - Check - Bet	

			- Raise	
A.I has dealer button, no action completed	A.I. has dealer button		User does not have the option to carry out any action until A.I carries out one of following actions: - Check - Bet - Raise	
AI checks, followed by User checks	The previous AI action is Check	The user checks	The hand should move on to the river stage.	
User checks, followed by AI checking	The previous user action is Check	The AI checks	The hand should move to the river stage.	
User raises, follwed by AI calling	The previous user action is raise	The AI calls	The hand should move to the river stage.	
AI raises, followed by user calling	The previous user action is raise	The AI call	The hand should move to the river stage.	
User bets, followed by AI checking	The previous user action is bet	The AI calls	The hand should move to the river stage.	
AI bets, followed by AI calls	The previous user action is bet	The AI calls	The hand should move to the river stage.	
A.I carries out action - fold		A.I carries out action to ‘Fold’	The pot should be added to the user chip stack. A new hand should be dealt out.	
User carries out action - fold		User carries out action to ‘Fold’	The pot should be added to the A.I. chip stack. A new hand should be dealt out.	

Feature: River

Precondition for all River scenarios:
- Turn completed

Scenario	Preconditions	Action	Result	Pass
Community cards			Should be five community cards visible to the user	
User has the dealer button, no action completed	User has dealer button		User has option to - Check - Bet - Raise	
A.I has dealer button, no action completed	A.I. has dealer button		User does not have the option to carry out any action until A.I carries out one of following actions: - Check - Bet - Raise	
AI checks, followed by User checks	The previous AI action is Check	The user checks	The hand should move on to the showdown stage.	
User checks, followed by AI checking	The previous user action is Check	The AI checks	The hand should move to the showdown stage.	
User raises, follwed by AI calling	The previous user action is raise	The AI calls	The hand should move to the showdown stage.	
AI raises, followed by user calling	The previous user action is raise	The AI call	The hand should move to the showdown stage.	
User bets, followed by AI checking	The previous user action is bet	The AI calls	The hand should move to the showdown stage.	
AI bets, followed by AI calls	The previous user action is bet	The AI calls	The hand should move to the showdown	

			stage.	
A.I carries out action - fold		A.I carries out action to 'Fold'	The pot should be added to the user chip stack. A new hand should be dealt out.	
User carries out action - fold		User carries out action to 'Fold'	The pot should be added to the A.I. chip stack. A new hand should be dealt out.	

Feature: Showdown

Precondition for all Showdown scenarios:
- River stage completed or all in called.

Scenario	Preconditions	Action	Result	Pass
User has better hand rank than AI at showdown (per rules of poker hand ranking)			The pot should be added to the User chip stack. Contributions and the pot should be reset to 0. A new hand should be dealt out.	
A.I. has better hand rank than user (as per rules of poker hand ranking)			The pot should be added to the AI chip stack. Contributions and the pot should be reset to 0. A new hand should be dealt out	

Feature: New hand

Scenario	Preconditions	Action	Result	Pass
User should not be able to see any board cards		New hand dealt	The user should not be able to see any board/community cards.	**Y**
The A.I should have the dealer button	In the previous hand, the user had the dealer button	New hand dealt	The A.I should have the dealer button for this hand	
The user should have the dealer button	In the previous hand, the A.I. had the dealer button	New hand dealt	The user should have the dealer button for this hand	
Blinds have been correctly placed when the user has the dealer button	- User has the dealer button	New hand dealt	The user should have placed half of the the big blind as contribution to the pot. The A.I should have placed all of the big blind as their contribution to the pot. The pot should be made up of the total contributions by both players	
Blinds have been correctly placed when the A.I has the dealer button	- User should be succesfully logged in. - User is on the main game screen - A.I. has the dealer button	New hand dealt	The AI should have placed half of the the big blind as contribution to the pot. The user should have placed all of the big blind as their contribution to the pot. The pot should be made up of the total contributions by both players	

Feature: Game Completion

Scenario	Preconditions	Action	Result	Pass
User does not have enough chips to place a big blind in a hand	User does not have chips to place a big blind in a new hand	New hand	Message appears informing user that they have lost, button inviting player to play a new game is shown	
A.I does not have enough chips to place a big blind in a new hand	A.I. does not have enough chips to place a big blind in a new hand	New hand	Message appears congratulating player on their win, button inviting player to play a new game is shown	

Game Actions

Feature: Bet

Scenario	Preconditions	Action	Result	Pass
User chooses to bet	User has option to bet	- User types amount greater than/equal the minimum bet and less than/equal their own chip stack - User clicks option to bet	Amount is added to both user contribution and the total pot A.I. turn to carry out an action(call, fold or raise).	
A.I chooses to bet	A.I has option to bet A.I has enough chips to bet.	A.I carries out action to bet.	Amount that the AI has bet by is added to AI contribution and also to the total pot.	
User chooses to bet - not enough chips	User has option to bet User has less chips than minimum bet	- User types amount greater than/equal the minimum bet and less than/equal their own chip stack	A modal appears to the user informing them that they must bet a valid amount - between the minimum	

		- User clicks option to bet	bet and their chip stack size.	
User chooses to bet - amount over their stack size	User has option to bet	- User types amount greater to their own chip stack - User clicks option to bet	A modal appears to the user informing them that they must bet a valid amount - between the minimum bet and their chip stack size.	
User chooses to bet - amount below minimum	User has option to bet	- User types amount below the minimum bet - User clicks option to bet	A modal appears to the user informing them that they must bet a valid amount - between the minimum bet and their chip stack size.	
User chooses to bet - invalid number	User has option to bet	- User types an amount that can not be parsed as a positive integer e.g) “test”, “-400”, “forty” - User clicks option to bet	A modal screen appears to the user informing them that: ‘The amount you have entered is not a valid integer. Please enter valid value.’	

Feature: Fold

Scenario	**Preconditions**	**Action**	**Result**	**Pass**
User chooses to fold	User has the option to fold	User clicks on ‘fold option’	The pot should be added to the A.I. chip stack. Contributions and the pot should be reset to 0. A new hand should be dealt out.	
A.I. chooses to fold	A.I. has the option to fold	User clicks on ‘fold’ option	The pot should be added to the user chip	

			stack. Contributions and the pot should be reset to 0. A new hand should be dealt out.	

Feature: Raise

Scenario	Preconditions	Action	Result	Pass
User chooses to raise	User has option to raise User has enough chips to both call and raise a bet.	- User types amount into raise by input greater than/equal the minimum bet and less than/equal their own chip stack - User clicks option to raise	Both the amount to call by and amount to raise by is added to both user contribution and the total pot A.I. turn to carry out an action(call, fold or raise).	
A.I chooses to raise	A.I has option to raise A.I has enough chips to both call and raise a bet.	A.I carries out action to raise.	Amount that the AI has raise by is added to AI contribution and also to the total pot. User turn to carry out action (call, fold, or raise)	
User chooses to raise- not enough chips	User has option to raise User has raise chips than minimum bet	- User types amount greater than/equal the minimum bet and less than/equal their own chip stack - User clicks option to raise	A modal appears to the user informing them that they must bet a valid amount - between the minimum bet and their chip stack size.	
User chooses to raise - amount over their stack size	User has option to raise	- User types amount greater to their own chip stack - User clicks option to raise	A modal appears to the user informing them that they must bet a valid amount - between the minimum bet and their chip stack size.	
User chooses to raise- amount	User has option to raise	- User types amount below the minimum	A modal appears to the user informing	

below minimum		bet - User clicks option to raise	them that they must bet a valid amount - between the minimum bet and their chip stack size.	
User chooses to raise - invalid number	User has option to raise	- User types an amount that can not be parsed as a positive integer e.g) “testraise”, “-400”, “forty” - User clicks option to raise	A modal screen appears to the user informing them that: ‘The amount you have entered is not a valid integer. Please enter valid value.’	

Feature: Check

Scenario	Preconditions	Action	Result	Pass
User chooses to check	User has option to check	- User clicks option to check	No contribution is added to either user contribution or the total pot A.I. turn to carry out an action.	
A.I. chooses to check	A.I. has option to check	A.I. carries out option to check	No contribution is added to either A.I. contribution or the total pot User turn to carry out action	

Feature: Call

Scenario	Preconditions	Action	Result	Pass
User chooses to call	User has option to call	- User clicks option to call	The amount called is the contribution is	

	User has enough to chips to call		added to both user contribution and the total pot A.I. turn to carry out an action.	
A.I. chooses to call	A.I. has option to check A.I has enough chips to call	A.I. carries out option to call	The amount called is the contribution is added to both user contribution and the total pot User turn to carry out action	
User chooses to call - not enough chips	User has option to call User does not have enough to chips to call	- User clicks option to call	The game ends with winner being: - A.I	
A.I. chooses to call - not enough chips	A.I. has option to call A.I has not enough chips to call	A.I. carries out option to call	The game ends with winner being: - User	

Feature: All In

All In - extends bet/raise features

Scenario	Preconditions	Action	Result	Pass
User clicks on all in - AI and User have equal amount of chips	AI and user have equal amount of chips	User clicks option to go All In.	The amount that the user bet all in should be removed from the user stack (i.e. stack should be zero). The contribution should be added to the total pot.	
User clicks on all in - User has greater amount of chips than AI	User has more chips than the AI	User clicks option to go All In.	The amount that the AI has should be removed from the user stack. The contribution should be added to the pot.	

User clicks on all in - AI has greater amount of chips than User	AI has more chips than the User	User clicks option to go All In.	The amount that the User has should be removed from the user stack (i.e. user stack should be zero). The contribution should be added to the pot.	

Game Objects

Feature: Cards

Scenario	Preconditions	Action	Result	Pass
Valid Card - value			Valid card should one value of the following: - 2,3,4,5,6,7,8,9, 10,J,Q,K,A	
Valid Card - suit			Valid card should have one suit of one of the following: - Hearts, Spades, Clubs, Diamonds	
Valid Cards in a hand			No two cards equivalent in both suit and value may appear in one hand.	

Feature: Stacks

Scenario	Preconditions	Action	Result	Pass
Valid Stack			Stack should never have a negative value. (always >= 0)	

Feature: Minimum Bet

Scenario	Preconditions	Action	Result	Pass
Minimum should increase every 'X' hands		'X' hands played	The minimum bet should double	
The number of hands should decrease for every hand that passes, apart from '0 hands left' case		A hand is played	The number of hands left until the min. bet doubles should decrement by one.	
When there are 0 hands left, the number of hands should then go back to a higher number(e.g. 10/15/20)	1 hand left until minimum bet doubles	A hand is played	The number should reset to a higher number.	

Web Browser Testing

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

Web Browser	Version No.
Google Chrome	58
Mozilla Firefox	49
Microsoft Edge	38

User Evaluation

The tasks were designed to attempt to test all use cases/ functionality of the system. They were written in a manner to set up a context for the test user. The tasks for user evaluation were defined as follows:

User Task Evaluation - Tasks

Task # 1
You want to play some poker online. Log in to the system.

Task # 2
The AI has checked into you. You wish to continue playing into the next stage of play but do not wish to add any chips to the pot.

Task # 3
You have a hand of medium strength. You wish to increase the stakes by adding 400 chips to the pot.

Task # 4
The AI has bet into you. You do not wish to continue playing this hand. End this current hand.

Task # 5
You have just checked into the AI. You want to identify what action they perform in response.

Task # 6
You have a strong hand. You wish to bet the entirety of your chip stack into the pot.

Task # 7
You and the AI have gone to showdown. You wish to identify who has won the hands and what cards the AI had.

Effectiveness

Efficiency

Satisfaction
The following questionnaire was given out to measure satisfaction. It uses slightly adapted questions from the 'System Usability Scale' suggested in the test plan.

Questions were accompanied with scale marked from 1 - 5.

1. If I wanted to play poker online, I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

System usability test scores: