Software Test Document

WebGL Card Game Platform

Version 1.6

3/15/19

Team WebGL Game

https://docs.google.com/document/d/1_rBI6LyQNQanQNhyFx34lwRcTmN-1m3b9Sm8l6Z7bDg/edit

Computer Science / Bellevue College

Sara Farag

Revisions

Version	Primary Author	Description of Version	Date completed
1.0	Jeffrey Talada	Created primary version of document	11/21/18
1.1	Jeffrey Talada, Sean Hardin	Finalized for Milestone 1	12/4/18
1.2	Anthony Klobas, Sean Hardin, Jeffrey Talada	Added more test cases Added Test Log	1/24/19
1.3	Anthony Klobas, Sean Hardin, Jeffrey Talada	Added test cases and more test logs	2/8/19
1.4	Sean Hardin, Jeffrey Talada	Added test cases and more test logs	2/20/19
1.5	Sean Hardin, Jeffrey Talada, Anthony Klobas	Added test cases and more test logs	3/8/19
1.6	Jeffrey Talada	Updated Testing Schedule	3/15/19

Table of Contents

Contents

1. Introduction

1.1 Purpose

The purpose of this document is to lay out how testing the card gaming platform will function and progress. It is intended for us as developers, any future developers, and our teacher, Sara Farag.

1.2 Scope

Testing will ensure designed functionalities work as intended. Security testing shall ensure that moves and bets can't be forged and that players can't know more information about game objects than allowed. Load testing shall ensure the system can support one to eight users per game without server move and bet verification exceeding 100ms. Multi-game performance is dependent on the server being used and is out of scope. Any third party libraries are assumed to be working properly and will only be tested for existence during integration and system testing.

1.3 Test Approach

Unit testing will be performed on each class before being uploaded to the repository for integration testing. Once the full system is complete, it will go through system testing and finally acceptance testing, both alpha and beta assuming testers can be found.

1.4 References

Mocha: mochajs.org, 'Mocha', 2018. [Online]. Available: https://mochajs.org/. [Accessed: 3- Dec- 2018].
Software Testing Levels: tutorialspoint.com, 'Software Testing Overview'. [Online]. Available: https://www.tutorialspoint.com/software_engineering/software_testing_overview.htm. [Accessed: 3- Dec- 2018].

2. Test Plan

2.1 Features to be tested

- Renderer
    - Request scene
    - Display to user
- Scene
    - Encode scene
- View
    - Send inputs to controller
    - Update object
        - Location
        - Existence
    - Request render
    - Objects render
- Controller
    - Send commands to model
- Gamerules - Basic Blackjack
    - Request card
    - Assign card to player
    - Add to bet
    - Player wins if they beat Dealer’s score and stay under or equal to 21
    - Player loses if they are lower than dealer’s score or go over 21
    - Game pays out when player wins
    - Reset game
- Player
    - Make bet
- Deck
    - Draw card

2.2 Features not to be tested

- Frame rate
- Displays properly in browser suite
- Responsiveness to user actions
- Displays properly on computers that only have WebGL emulation on CPU
- Pixel perfect movement
- 3rd party libraries
    - Node.js
    - Three.js

2.3 Testing Tools and Environment

- Mocha - Javascript Unit Testing
- Azure CI/CD - Integration Testing

3. Test Cases

3.1 Renderer

3.1.1 Instantiate Empty Renderer

3.1.1.1 Purpose

Make sure Renderer can be instantiated

3.1.1.2 Input

var renderer = new Renderer();

3.1.1.3 Expected Output

No exceptions thrown

3.1.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.1.3 Render Scene

3.1.2.1 Purpose

Test whether a scene can be rendered

3.1.2.2 Input

Renderer renderer = new Renderer();
render(test_scene, test_camera);

3.1.2.3 Expected Output

Test scene is displayed

3.1.2.4 Pass/Fail Criteria

Passes if the scene is drawn and no exceptions are thrown
Fails if either isn't the case

3.2 Scene

3.2.1 Instantiate Scene

3.2.1.1 Purpose

Make sure a scene object can be instantiated

3.2.1.2 Input

var scene = new Scene();

3.2.1.3 Expected Output

No exceptions thrown

3.2.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.2.2 Configure Scene

3.2.2.1 Purpose

Make sure the scene object can be configured

3.2.2.2 Input

scene.setConfiguration(test_config);
scene.getConfiguration();

3.2.2.3 Expected Output

Scene displays with given configuration

3.2.2.4 Pass/Fail Criteria

Passes if the configuration is correct
Fails otherwise

3.2.3 Add Object to Scene

3.2.3.1 Purpose

Make sure objects can be added to the scene

3.2.3.2 Input

```
scene.addObject(test_object1);
scene.listObjects();
```

3.2.3.3 Expected Output

Object visible in scene

3.2.3.4 Pass/Fail Criteria

Passes if id 0 is returned

3.2.4 Delete Object from Scene

3.2.4.1 Purpose

Make sure objects can be deleted from the scene

3.2.4.2 Input

```
scene.addObject(test_object1);
scene.deleteObject(test_object1);
scene.listObjects();
```

3.2.4.3 Expected Output

[]

3.2.4.4 Pass/Fail Criteria

Passes if no objects are returned

3.2.5 Delete Correct Object from Scene

3.2.5.1 Purpose

Make sure objects can be deleted from the scene

3.2.5.2 Input

```
scene.addObject(test_object1);
scene.addObject(test_object2);
scene.deleteObject(test_object1);
scene.listObjects();
```

3.2.5.3 Expected Output

[test_object2]

3.2.5.4 Pass/Fail Criteria

Passes if only test_object2 is returned

3.3 WebGL_View

3.3.1 Instantiate WebGLView

3.3.1.1 Purpose

Make sure a WebGLView object can be instantiated

3.3.1.2 Input

var view = new WebGLView();

3.3.1.3 Expected Output

Make sure a view object can be instantiated

3.3.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.3.2 Set Layout

3.3.2.1 Purpose

Make sure view object can load a layout

3.3.2.2 Input

var view = new WebGLView();
view.setLayout(test_layout);

3.3.2.3 Expected Output

Correctly displays the test_layout

3.3.2.4 Pass/Fail Criteria

Passes if the test_layout is displayed correctly

3.3.3 Set Location

3.3.3.1 Purpose

Make sure view object can be displayed from a given location

3.3.3.2 Input

var view = new WebGLView();
view.set_location(test_coordinates);

3.3.3.3 Expected Output

Correctly displays the view from the test_coordinates

3.3.3.4 Pass/Fail Criteria

Passes if the view is displayed correctly from the perspective of the test_coordinates

3.3.4 Add Object

3.3.4.1 Purpose

Make sure view object can add game objects to its mapper

3.3.4.2 Input

var view = new WebGLView();
view.add(test_object1, 0);
view.scene.listObjects();

3.3.4.3 Expected Output

[Test_object]

3.3.4.4 Pass/Fail Criteria

Passes if the list of objects contains the test_object and the mapper contains the coordinates of the object when searched with key 0.

3.3.5 Remove Object

3.3.5.1 Purpose

Make sure view object can remove game objects from its mapper

3.3.5.2 Input

```
var view = new WebGLView(test_scene);
view.add(test_object1, 0);
view.remove(test_object1, 0);
view.list();
view.mapper.lookup(0);
```

3.3.5.3 Expected Output

```
[ ]
[ ]
```

3.3.5.4 Pass/Fail Criteria

Passes if the list returned is empty and test_object1 is removed from the mapper

3.3.5 Remove Correct Object

3.3.5.1 Purpose

Make sure view object can remove specific game objects from its mapper

3.3.5.2 Input

```
var view = new WebGLView(test_scene);
view.add(test_object1, 0);
view.add(test_object2, 1);
view.remove(test_object1, 0);
view.mapper.lookup(0);
view.mapper.lookup(1);
```

3.3.5.3 Expected Output

```
[ ]
[test_object2]
```

3.3.5.4 Pass/Fail Criteria

Passes if the list returned includes an id of 1 and test_object2

3.4 ObjectMapper

3.4.1 Instantiate ObjectMapper

3.4.1.1 Purpose

Make sure an ObjectMapper can be instantiated

3.4.1.2 Input

var objectMapper = new ObjectMapper();

3.4.1.3 Expected Output

No exceptions thrown

3.4.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.4.2 Set Configuration

3.4.2.1 Purpose

Make sure an ObjectMapper can accept a configuration

3.4.2.2 Input

var objectMapper = new ObjectMapper();
objectMapper.setConfiguration(test_configuration)

3.4.2.3 Expected Output

No exceptions thrown

3.4.2.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.5 GLPosition

3.5.1 Instantiate GLPosition

3.5.1.1 Purpose

Make sure a GLPosition can be instantiated

3.5.1.2 Input

var glPosition = new GLPosition();

3.5.1.3 Expected Output

No exceptions thrown

3.5.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.5.2 Set Location of GLPosition

3.5.2.1 Purpose

Make sure a GLPosition can have its location set

3.5.2.2 Input

```
var glPosition = new GLPosition();
glPosition.setLocation(testCoordinates);
glPosition.getLocation();
```

3.5.2.3 Expected Output

testCoordinates

3.5.2.4 Pass/Fail Criteria

Passes if the testCoordinates are returned

3.6 GLObject

3.6.1 Instantiate GLObject

3.6.1.1 Purpose

Make sure a GLObject can be instantiated

3.6.1.2 Input

var glObject = new GLObject();

3.6.1.3 Expected Output

No exceptions thrown

3.6.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.6.2 Set Location of GLObject

3.6.2.1 Purpose

Make sure a GLObject can have its location set

3.6.2.2 Input

```
var glObject = new GLObject();
glObject.setLocation(testCoordinates);
glObject.getLocation();
```

3.6.2.3 Expected Output

testCoordinates

3.6.2.4 Pass/Fail Criteria

Passes if the testCoordinates are returned

3.7 Collection

3.7.1 Instantiate Collection

3.7.1.1 Purpose

Make sure a Collection can be instantiated

3.7.1.2 Input

```
var collection = new Collection();
```

3.7.1.3 Expected Output

No exceptions thrown

3.7.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.7.2 Add to Collection

3.7.2.1 Purpose

Make sure a Collection can have objects added to it

3.7.2.2 Input

```
var collection = new Collection();
collection.add(test_gl_object1);
collection.list();
```

3.7.2.3 Expected Output

[test_gl_object1]

3.7.2.4 Pass/Fail Criteria

Passes if test_gl_object is returned

3.7.3 Remove Object

3.7.3.1 Purpose

Make sure view object can remove game objects from its mapper

3.7.3.2 Input

```
var collection = new Collection();
collection.add(test_gl_object1);
collection.remove(test_gl_object1);
collection.list();
```

3.7.3.3 Expected Output

[]

3.7.3.4 Pass/Fail Criteria

Passes if the list returned is empty

3.7.4 Remove Correct Object

3.7.4.1 Purpose

Make sure view object can remove specific game objects from its mapper

3.7.4.2 Input

```
var collection = new Collection();
collection.add(test_gl_object1);
collection.add(test_gl_object2);
collection.remove(test_gl_object1);
collection.list();
```

3.7.4.3 Expected Output

```
[test_gl_object2]
```

3.7.4.4 Pass/Fail Criteria

Passes if the returned list consists only of test_gl_object2

3.7.5 Remove Doesn't Break Add Object

3.7.5.1 Purpose

Make sure remove doesn't break adding objects to the mapper

3.7.5.2 Input

```
var collection = new Collection();
collection.add(test_gl_object1);
collection.add(test_gl_object2);
collection.remove(test_gl_object1);
collection.add(test_gl_object1);
collection.list();
```

3.7.5.3 Expected Output

```
[[test_gl_object2],
[test_gl_object1]]
```

3.7.5.4 Pass/Fail Criteria

Passes if the list returned is equal to the expected output
Fails if they are in the wrong order or one or both don't appear

3.7.6 Sort Collection

3.7.6.1 Purpose

Make sure the collection is sorted properly

3.7.6.2 Input

```
var collection = new Collection();
collection.add(test_gl_object2);
collection.add(test_gl_object1);
collection.sort();
collection.list();
```

3.7.6.3 Expected Output

```
[[test_gl_object1],
[test_gl_object2]]
```

3.7.6.4 Pass/Fail Criteria

Passes if the list returned is equal to the expected output
Fails if they are in the wrong order or one or both don't appear

3.8 Bezier

3.8.1 Instantiate Bezier

3.8.1.1 Purpose

Make sure a Bezier can be instantiated

3.8.1.2 Input

```
var bezier = new Bezier();
```

3.8.1.3 Expected Output

No exceptions thrown

3.8.1.4 Pass/Fail Criteria

Passes if no exceptions are thrown

3.8.2 Get Position of a Bezier

3.8.2.1 Purpose

Make sure a Bezier can return its position

3.8.2.2 Input

```
var bezier = new Bezier();
```

bezier.setPosition(test_vec);
bezier.getPosition();

3.8.2.3 Expected Output

test_vec

3.8.2.4 Pass/Fail Criteria

Passes if test_vec is returned

3.8.3 Get Bezier Normal

3.8.3.1 Purpose

Make sure a Bezier can return its normal

3.8.3.2 Input

var bezier = new Bezier();
bezier.setPosition(test_vec);
bezier.getnormal();

3.8.3.3 Expected Output

normal

3.8.3.4 Pass/Fail Criteria

Passes if a vec3 with the correct normal is returned

3.9 Index.mjs, ServerMainController, and SingleGameServer

3.9.1 Instantiate Server

3.9.1.1 Purpose

Make sure the server starts

3.9.1.2 Input

node --experimental-modules index.js

3.9.1.3 Expected Output

Server is running

3.9.1.4 Pass/Fail Criteria

Pass if server runs
Fails if there are any exceptions

3.9.2 Make Connection

3.9.2.1 Purpose

Establish a TCP session with a client

3.9.2.2 Input

GET / HTTP1.1

3.9.2.3 Expected Output

Client receives the website

3.9.2.4 Pass/Fail Criteria

Passes if the client receives the site
Fails otherwise

3.9.3 Instantiate Game

3.9.3.1 Purpose

Establish a game

3.9.3.2 Input

GameInstance game = new GameInstance(‘blackjack’);

3.9.3.3 Expected Output

Game of Blackjack is constructed

3.9.3.4 Pass/Fail Criteria

Passes if a game of Blackjack is constructed

3.9.4 Add Client to Game

3.9.4.1 Purpose

Be able to add players to a game instance

3.9.4.2 Input

game.addClient(client);

3.9.4.3 Expected Output

Client is in the turn queue

3.9.4.4 Pass/Fail Criteria

Passes if the client is added to the game instance
Fails otherwise

3.10 Blackjack

3.10.1 Shuffle Deck

3.10.1.1 Purpose

Have a shuffled deck

3.10.1.2 Input

deck.shuffle();

3.10.1.3 Expected Output

Cards are in random positions in the deck

3.10.1.4 Pass/Fail Criteria

Pass if cards are randomized in the deck
Fails if the cards are in the same position as before shuffling or any exceptions are thrown

3.10.2 Call player turn

3.10.2.1 Purpose

Give the player a turn

3.10.2.2 Input

client.yourTurn();

3.10.2.3 Expected Output

‘Player 1 turn’

3.10.2.4 Pass/Fail Criteria

Pass if the expected output is printed
Fails if the player never gets a turn

3.10.3 Hit

3.10.3.1 Purpose

Player can hit

3.10.3.2 Input

client.yourTurn();
return ‘hit’;

3.10.3.3 Expected Output

client.getHand() shows three cards

3.10.3.4 Pass/Fail Criteria

Pass if the expected output is printed
Fails otherwise

3.10.4 Stay

3.10.4.1 Purpose

Player can stay

3.10.4.2 Input

client.yourTurn();
return ‘stay’;

3.10.4.3 Expected Output

client.getHand() shows two cards

3.10.4.4 Pass/Fail Criteria

Pass if the expected output is printed
Fails otherwise

3.10.5 Bet

3.10.5.1 Purpose

Player can submit a bet

3.10.5.2 Input

client.yourTurn();
return (‘bet’, amount);

3.10.5.3 Expected Output

client.getBet() shows bet

3.10.5.4 Pass/Fail Criteria

Pass if the expected output is correct
Fails otherwise

3.10.6 start

3.10.6.1 Purpose

Player can start game

3.10.6.2 Input

client.yourTurn();
return ‘start’;

3.10.6.3 Expected Output

Game starts

3.10.6.4 Pass/Fail Criteria

Pass if the game starts properly
Fails otherwise

3.10.7 Double Down

3.10.7.1 Purpose

Player can double down

3.10.7.2 Input

client.yourTurn();
return ‘double_down’;

3.10.7.3 Expected Output

client.getBet() shows double the initial bet on win

3.10.7.4 Pass/Fail Criteria

Pass if the expected output is printed
Fails otherwise

3.10.8 Split

3.10.8.1 Purpose

Player can split their initial hand

3.10.8.2 Input

client.yourTurn();
return ‘split’;

3.10.8.3 Expected Output

Client now has another hand

3.10.8.4 Pass/Fail Criteria

Pass if the player has additional hand
Fails otherwise

3.10.9 Surrender

3.10.9.1 Purpose

Player can surrender on initial hand

3.10.9.2 Input

client.yourTurn();
return ‘surrender’;

3.10.9.3 Expected Output

Client surrendered and received half their bet back

3.10.9.4 Pass/Fail Criteria

Pass if the bet returned as expected
Fails otherwise

3.11 Index.html

3.11.1 Load Dynamic Buttons

3.11.1.1 Purpose

Be able to load all buttons for a given game on the page with just a json file

3.11.1.2 Input

loadButtons(jsonName);

3.11.1.3 Expected Output

Html button area is populated with all relevant buttons for game

3.11.1.4 Pass/Fail Criteria

Passes if all buttons added
Fails otherwise

3.11.2 Update Player Summary

3.11.2.1 Purpose

Be able to see the gamestate of all players in the same game

3.11.2.2 Input

updatePlayerSummary(gameStateObject);

3.11.2.3 Expected Output

HTML player summary area populated with all card/bet info of the server's current gamestate

3.11.2.4 Pass/Fail Criteria

Passes if the summary is accurate
Fails otherwise

3.11.3 Change Game Settings

3.11.2.1 Purpose

Ensure that game settings can be changed

3.11.2.2 Input

HTML form populated from gameRules.json
User changes fields
Send gameRules.json to server

3.11.2.3 Expected Output

Server receives updated gameRules.json

3.11.2.4 Pass/Fail Criteria

Passes if the updated gameRules.json is different from the original, valid, and what the user input
Fails otherwise

3.11.4 Choose game

3.11.2.1 Purpose

Ensure that player can choose which active game to join

3.11.2.2 Input

HTML form populated from gameRules.json
User changes fields or selects an existing set of settings
Send settings to server to make or join

3.11.2.3 Expected Output

Server receives settings and makes the game or adds the player to the existing game

3.11.2.4 Pass/Fail Criteria

Passes if the server receives the settings from the user to use
Fails otherwise

3.12 SingleSessionController

3.12.1 Set configuration

3.12.1.1 Purpose

Be able to set the configuration for that game session

3.12.1.2 Input

setConfig(jsonName);

3.12.1.3 Expected Output

Sessions specific settings are set according to passed json

3.12.1.4 Pass/Fail Criteria

Passes if settings are properly set
Fails otherwise

3.12.2 Verify Move

3.12.2.1 Purpose

Be able to verify that the player's move is valid

3.12.2.2 Input

verify(playerNumber, moveName);

3.12.2.3 Expected Output

If the model's verify function returns true, should run the model's selected() function updateGameState() function.
If the model's verify function returns false, should respond to the player denying their move.

3.12.2.4 Pass/Fail Criteria

Passes if either expected output occurs
Fails otherwise

3.12.3 Add Player

3.12.3.1 Purpose

Be able to add a player to self and the attached model after initializing them with a unique id.

3.12.3.2 Input

add(player);

3.12.3.3 Expected Output

Player successfully added to session and model with unique id, with their socket listening for a ‘verify’ call to run the verify() function.

3.12.3.4 Pass/Fail Criteria

Passes if added with no errors
Fails otherwise

3.12.4 Check Room

3.12.4.1 Purpose

Be able to check if the current session has room for another player

3.12.4.2 Input

checkRoom();

3.12.4.3 Expected Output

If there is room in the sessions still, return true;
If not, return false;

3.12.4.4 Pass/Fail Criteria

Passes if output matches the conditions listed under expected output.
Fails otherwise

3.12.5 Remove Player

3.12.5.1 Purpose

Be able to remove a player from the session

3.12.5.2 Input

remove(player);

3.12.5.3 Expected Output

Removes the passed player from the session
If the session is empty afterwards, removes the session too.

If the session is not empty, calls dropPlayer() function.

3.12.5.4 Pass/Fail Criteria

Passes if player completely removed as expected
Fails otherwise

3.12.6 Drop Player

3.12.6.1 Purpose

Clean up the session after removing a player

3.12.6.2 Input

dropPlayer(playerNumber);

3.12.6.3 Expected Output

Calls model's dropPlayer() function.
Reassigns player numbers to remove the gap

3.12.6.4 Pass/Fail Criteria

Passes if session updated correctly
Fails otherwise

3.12.7 Update Gamestate

3.12.7.1 Purpose

Send the new gamestate to every connected player.

3.12.7.2 Input

updateGameState();

3.12.7.3 Expected Output

Calls model's getGameState() function.
Sends the received gamestate to all players in session

3.12.7.4 Pass/Fail Criteria

Passes if correct gamestate is sent out.
Fails otherwise

3.12.8 Construct Session

3.12.8.1 Purpose

Create a new session with the given model

3.12.8.2 Input

serverSingleSessionController sssc = new serverSingleSessionController(model);

3.12.8.3 Expected Output

Attaches passed model to self
Initializes an array of players

3.12.8.4 Pass/Fail Criteria

Passes if correct both expected outputs occur
Fails otherwise

3.13 Client.mjs

3.13.1 Construct Client

3.13.1.1 Purpose

Be able to construct a client with a socket and mainServer

3.13.1.2 Input

Client c = new Client(socket, server);

3.13.1.3 Expected Output

Client created successfully
Socket listens for login and join calls.

3.13.1.4 Pass/Fail Criteria

Passes if client created and listening.
Fails otherwise

3.13.2 Set Session Server

3.13.2.1 Purpose

Be able to know which session the client is connected to

3.13.2.2 Input

setServer(server);

3.13.2.3 Expected Output

Server attached to client.

3.13.2.4 Pass/Fail Criteria

Passes if client is attached to server
Fails otherwise

3.13.3 Your Turn

3.13.3.1 Purpose

Be able to send message to client indicating their turn

3.13.3.2 Input

yourTurn();

3.13.3.3 Expected Output

Emits a ‘your turn’ call to the client

3.13.3.4 Pass/Fail Criteria

Passes if call is sent
Fails otherwise

3.13.4 Initialize Player

3.13.4.1 Purpose

Be able to identify a player by a unique value

3.13.4.2 Input

init(number);

3.13.4.3 Expected Output

Emits an ‘id’ call to the client with their number attached

3.13.4.4 Pass/Fail Criteria

Passes if call is sent

Fails otherwise

3.13.5 Deny Move

3.13.5.1 Purpose

Be able to tell a player that their move was denied

3.13.5.2 Input

deny();

3.13.5.3 Expected Output

Emits a ‘deny’ call to the client

3.13.5.4 Pass/Fail Criteria

Passes if call is sent
Fails otherwise

3.13.6 Drop Player

3.13.6.1 Purpose

Be able to tell a player that another player was dropped

3.13.6.2 Input

dropPlayer(num);

3.13.6.3 Expected Output

Emits a ‘drop’ call to the client with a player number attached

3.13.6.4 Pass/Fail Criteria

Passes if call is sent
Fails otherwise

3.13.7 Update

3.13.7.1 Purpose

Be able to tell a player the new gamestate

3.13.7.2 Input

update(gamestate);

3.13.7.3 Expected Output

Emits an ‘update’ call to the client with a gamestate attached

3.13.7.4 Pass/Fail Criteria

Passes if call is sent
Fails otherwise

3.13.8 Remove

3.13.8.1 Purpose

Be able to remove a player from their session

3.13.8.2 Input

remove();

3.13.8.3 Expected Output

Calls the server’s remove() function

3.13.8.4 Pass/Fail Criteria

Passes if function is called successfully
Fails otherwise

3.13.9 Login

3.13.9.1 Purpose

Be able to log in

3.13.9.2 Input

login(user, pass);

3.13.9.3 Expected Output

Calls the mainServer’s databaseModel/s login function

3.13.9.4 Pass/Fail Criteria

Passes if function is called successfully

Fails otherwise

3.13.10 Authenticate

3.13.10.1 Purpose

Actually log the player in

3.13.10.2 Input

authenticate(data);

3.13.10.3 Expected Output

Attach the data to the client.
Send ‘authenticated’ call with data attached.

3.13.10.4 Pass/Fail Criteria

Passes if data attached and message sent successfully
Fails otherwise

3.13.11 Join

3.13.11.1 Purpose

Let a player join a chosen game

3.13.11.2 Input

join(game);

3.13.11.3 Expected Output

mainServer runs add() function on player, game.

3.13.11.4 Pass/Fail Criteria

Passes if function called successfully
Fails otherwise

3.14 Blackjack AIs

3.14.1 On update dealer receives correct hand

3.14.1.1 Purpose

Make sure the dealer sees the correct cards

3.14.1.2 Input

```
update = [[1,10],[2,5]];
AI.update(update);
AI.getHand();
```

3.14.1.3 Expected Output

[1,10]

3.14.1.4 Pass/Fail Criteria

Passes if the dealer hand is returned
Fails if anything else is returned

3.14.2 Dealer Calculates Correct Hard Total

3.14.2.1 Purpose

Make sure the dealer calculates correct hard total

3.14.2.2 Input

```
update = [[2,10],[2,5]];
AI.update(update);
AI.getTotal();
```

3.14.2.3 Expected Output

12

3.14.2.4 Pass/Fail Criteria

Passes if the returned value is 12
Fails if anything else is returned

3.14.3 Dealer Calculates the Correct Soft Total

3.14.3.1 Purpose

Make sure the dealer calculates correct soft total

3.14.3.2 Input

```
update = [[1,1],[2,5]];
AI.update(update);
AI.getTotal();
```

3.14.3.3 Expected Output

12

3.14.3.4 Pass/Fail Criteria

Passes if the returned value is 12
Fails if anything else is returned

3.14.4 Hit on Soft Flag Works Correctly

3.14.4.1 Purpose

Make sure the dealer hits on soft 17 when flag is true and not when false

3.14.4.2 Input

```
AI = new dealerAI(true);
update = [[1,6],[2,5]];
AI.update(update);
Var move = AI.yourTurn();
console.log(move);
AI = new dealerAI(false);
update = [[1,6],[2,5]];
AI.update(update);
Move = AI.yourTurn();
console.log(move);
```

3.14.4.3 Expected Output

‘Hit’
‘Stay’

3.14.4.4 Pass/Fail Criteria

Passes if ‘hit’ then ‘stay’ are returned

Fails if anything else is returned

3.14.5 PlayerAI Initializes with Correct Values

3.14.5.1 Purpose

Ensure the player AI initializes properly

3.14.5.2 Input

```
AI = new PlayerAI(1, 10, true, true, false, true, true);
console.log(AI.hand);
console.log(AI.dealerHand);
console.log(AI.bank);
console.log(AI.softLimit);
console.log(AI.doubleAfterSplitAllowed);
console.log(AI.basicStrategy[0][9][0]);
AI = new PlayerAI(1);
console.log(AI.basicStrategy[0][9][0]);
```

3.14.5.3 Expected Output

```
[]
[]
1000
20
False
‘Rs’
‘S’
```

3.14.5.4 Pass/Fail Criteria

Passes if the expected output is returned
Fails if anything else is returned

3.14.6 PlayerAI Splits

3.14.6.1 Purpose

Ensure the player AI returns ‘split’ for correct hand values

3.14.6.2 Input

```
AI = new PlayerAI(1, 10, true, true);
AI.update([[-1,4],[7,7]]);
```

```
console.log(AI.yourTurn());
```

3.14.6.3 Expected Output

‘split’

3.14.6.4 Pass/Fail Criteria

Passes if ‘split’ is returned
Fails if anything else is returned

3.14.7 PlayerAI Doubles Down

3.14.7.1 Purpose

Ensure the player AI returns ‘double’ for correct hand values

3.14.7.2 Input

```
AI = new PlayerAI(1, 10, true, true, true);
AI.update([[-1,4],[4,5]]);
console.log(AI.yourTurn());
```

3.14.7.3 Expected Output

‘double’

3.14.7.4 Pass/Fail Criteria

Passes if ‘double’ is returned
Fails if anything else is returned

3.14.8 PlayerAI Hits When Double is Disabled

3.14.8.1 Purpose

Ensure the player AI returns ‘hit’ for a hit if you can’t double hand

3.14.8.2 Input

```
AI = new PlayerAI(1, 10, true, true);
AI.update([[-1,4],[4,5]]);
console.log(AI.yourTurn());
```

3.14.8.3 Expected Output

‘hit’

3.14.8.4 Pass/Fail Criteria

Passes if ‘hit’ is returned
Fails if anything else is returned

3.14.9 PlayerAI Surrenders

3.14.9.1 Purpose

Ensure the player AI returns ‘surrender’ for a surrender situation

3.14.9.2 Input

```
AI = new PlayerAI(1, 10, true, true, true, true, true);
AI.update([[-1,1],[10,5]]);
console.log(AI.yourTurn());
```

3.14.9.3 Expected Output

‘surrender’

3.14.9.4 Pass/Fail Criteria

Passes if ‘surrender’ is returned
Fails if anything else is returned

3.14.10 PlayerAI Stays When Surrender is Disabled

3.14.10.1 Purpose

Ensure the player AI returns ‘stay’ for a ‘stay’ if you can’t surrender hand

3.14.10.2 Input

```
AI = new PlayerAI(1, 10, true);
AI.update([[-1,4],[4,5]]);
console.log(AI.yourTurn());
```

3.14.10.3 Expected Output

‘stay’

3.14.10.4 Pass/Fail Criteria

Passes if ‘stay’ is returned
Fails if anything else is returned

4. Requirements Traceability Matrix

Requirement #	Requirement	Module Design Component	Data Design Component	Test-Case #
3.2.1	Make connection to server	3.2	n/a	n/a
3.2.2	Login	4.4	client	n/a
3.2.3	Start game	4.3	n/a	n/a
3.2.4	Send moves/bets	3.1	Playerhand, playerbet	n/a
3.2.5	Send chat	3.2	n/a	n/a
3.2.6	Send gamestate	4.1	n/a	n/a
3.2.7	Update chat messages	4.1	n/a	n/a
3.2.8	Join game	4.2	n/a	n/a
3.2.9	Handle dropped players	4.2	playerhand	n/a
3.2.10	Modify game settings	3.1	n/a	n/a
3.2.11	Determine bet	2.1	n/a	n/a
3.2.12	Display rules	2.2	n/a	n/a
3.2.13	Display game	2.2	n/a	3.1-8
3.2.14	Display win/loss and payout	2.2	n/a	n/a
3.2.15	Read mouse input	2.1	n/a	n/a
3.2.16	Query database	4.5	all	n/a
3.2.17	Validate bets/moves	4.3/1.1	n/a	n/a
3.2.18	Shuffle deck	1.3	n/a	n/a
3.2.19	Display opponent visible cards	2.2	n/a	n/a
3.2.20	Play against AI	4.3	n/a	n/a
3.2.21	Load buttons dynamically	2.1	n/a	n/a
3.2.22	Select game	3.1	n/a	n/a

5. Responsibilities

5.1 Unit Testing

Everyone is responsible for writing unit tests and testing the classes they write.

5.2 Integration Testing

Integration tests will be written by the writer of the class using another.

5.3 System Testing

System tests will be divided up, written, and performed by all team members.

5.4 User Testing

User tests will be solicited as needed by the team member writing a particular piece of functionality.

6. Staffing and training needs

All team members will become familiar with:

- Mocha (Javascript Unit Testing)
- Azure CI testing

7. Schedule

	Fall Quarter			Winter Quarter				Spring Quarter
Week/Sprint	Oct	Nov	Dec	Sprint 5	Sprint 6	Sprint 7	Sprint 8	
Development and Test Environment Setup								
Fall Sprints								
Unit Testing								
Integration Testing								
User Testing								
Sprint Review								
Winter Sprints								
Unit Testing								
Integration Testing								
User Testing								
Sprint Review								
Spring Sprints								
Unit Testing								
Integration Testing								
User Testing								
Sprint Review								

8. Test Log

Test Case ID	3.1.1
Test Case Title	Instantiate Empty Renderer
Inputs	Start App
Expected Results	Renderer is made
Actual Results	Renderer was made
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.1.2
Test Case Title	Render Scene
Inputs	Start app, place bet
Expected Results	A scene is rendered
Actual Results	A scene was rendered
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window, click on bet button
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.2.1
Test Case Title	Instantiate scene
Inputs	Start App
Expected Results	Scene is made
Actual Results	Scene was made
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.2.2
Test Case Title	Configure Scene
Inputs	GLBlackJack.json
Expected Results	Scene is laid out with 2 hands of cards and a deck
Actual Results	Scene wan laid out with 2 hands of cards and a deck
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.2.3
Test Case Title	Add object to scene
Inputs	addObject called from blackjack model

Expected Results	New object visible in scene
Actual Results	A new object was visible in the scene
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.2.8
Test Case Title	Join Game
Inputs	Click on blackjack
Expected Results	Enter a game of blackjack
Actual Results	Entered a game of blackjack
Anomalies	none
Date and Time	2/6/19
Procedure Step	1. Navigate to site 2. Click on blackjack
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.2.16
Test Case Title	Query Database
Inputs	login(client, username, password)
Expected Results	If username and password a valid client data Else nothing
Actual Results	With correct username and password the client was updated
Anomalies	none

Date and Time	2/5/19
Procedure Step	1. Navigate to site 2. Type in username and password 3. Press enter
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.3.1
Test Case Title	Instantiate webgl_view
Inputs	Start app
Expected Results	A view is made
Actual Results	A view was made
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.3.2
Test Case Title	Set layout
Inputs	GLBlackJack.json
Expected Results	View is laid out with 2 hands of cards and a deck
Actual Results	View was laid out with 2 hands of cards and a deck
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window

Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.3.4
Test Case Title	Add object to view
Inputs	addObject called from blackjack model
Expected Results	View now has a glObject containing a model object
Actual Results	View had a glObject containing a model object
Anomalies	none
Date and Time	1/24/19
Procedure Step	Open window
Attempts to Repeat	N/A
Testers	Anthony Klobas(manual)

Test Case ID	3.9.1
Test Case Title	Instantiate Server
Inputs	node --experimental-modules index.js
Expected Results	Server is running
Actual Results	Server is running
Anomalies	none
Date and Time	1/24/19
Procedure Step	Enter input in command line while in folder with index.js
Attempts to Repeat	N/A
Testers	Jeff Talada(manual)

Test Case ID	3.9.2

Test Case Title	Make Connection
Inputs	GET / HTTP1.1
Expected Results	Client receives index.html
Actual Results	Client received index.html
Anomalies	none
Date and Time	1/24/19
Procedure Step	1. Start server 2. Enter localhost:1337 as the url
Attempts to Repeat	N/A
Testers	Jeff Talada(manual)

Test Case ID	3.9.3
Test Case Title	Instantiate Game
Inputs	GameInstance game = new GameInstance('blackjack');
Expected Results	Game of Blackjack is constructed
Actual Results	Log outputted that a game was instantiated of the type Blackjack
Anomalies	none
Date and Time	1/24/19
Procedure Step	Call the gameInstance constructor with gametype of 'blackjack'
Attempts to Repeat	N/A
Testers	Jeff Talada(manual)

Test Case ID	3.9.4
Test Case Title	Add Client to Game
Inputs	game.addClient(client);
Expected Results	Client is added to the turn queue

Actual Results	Client is entered into the game
Anomalies	N/A
Date and Time	1/31/19
Procedure Step	1. Create a game 2. Create a client 3. Add the client to the game
Attempts to Repeat	N/A
Testers	Sean Hardin, Anthony Klobas (manual)

Test Case ID	3.10.1
Test Case Title	Shuffle Deck
Inputs	deck.shuffle();
Expected Results	Cards are in random positions in the deck
Actual Results	Cards are in random positions in the deck
Anomalies	none
Date and Time	1/24/19
Procedure Step	Call deck.shuffle()
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.2
Test Case Title	Call player turn
Inputs	client.yourTurn();
Expected Results	‘Player 1 turn’
Actual Results	‘Player 1 turn’
Anomalies	none
Date and Time	1/24/19

Procedure Step	1. Add client to the game 2. Run the game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.3
Test Case Title	Hit
Inputs	client.yourTurn(); return ‘hit’
Expected Results	client.getHand() shows three cards
Actual Results	client.getHand() shows three cards
Anomalies	none
Date and Time	1/24/19
Procedure Step	1. Add client to the game 2. Run the game 3. Have player hit
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.4
Test Case Title	Stay
Inputs	client.yourTurn(); return ‘stay’
Expected Results	client.getHand() shows two cards
Actual Results	client.getHand() shows two cards
Anomalies	none
Date and Time	1/24/19
Procedure Step	1. Add player to game 2. Run game 3. Have player return stay

Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.11.1
Test Case Title	Load Dynamic Buttons
Inputs	loadButtons(jsonName);
Expected Results	All buttons in the given json are created
Actual Results	All buttons in the given json were created
Anomalies	N/A
Date and Time	2/4/19
Procedure Step	1. Load page
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.11.2
Test Case Title	Update Player Summary
Inputs	updatePlayerSummary(gameStateObject);
Expected Results	Player summary updates with new information as moves are made
Actual Results	Player summary updates with the information after every move
Anomalies	N/A
Date and Time	2/4/19
Procedure Step	1. Start a game 2. Select hit to draw card
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.11.3
Test Case Title	Change settings
Inputs	Clicking the accept button on settings form
Expected Results	Fields are loaded into an object and sent to server
Actual Results	As expected
Anomalies	N/A
Date and Time	3/6/19
Procedure Step	1. Start a game 2. Change settings 3. Click accept
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.11.4
Test Case Title	Choose game
Inputs	Clicking accept button on settings form after changing game shown in dropdown menu
Expected Results	Player added to the game that they had selected
Actual Results	As expected
Anomalies	N/A
Date and Time	3/6/19
Procedure Step	1. Start a game 2. Select a game from dropdown on settings form 3. Click accept
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.1
Test Case Title	Set Configuration
Inputs	setConfig(json);
Expected Results	Settings are set according to json
Actual Results	Settings were set correctly
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Start game after choosing settings
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.2
Test Case Title	Verify Move
Inputs	verify(num, move);
Expected Results	If the model's verify function returns true, should run the model's selected() function updateGameState() function. If the model's verify function returns false, should respond to the player denying their move.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player clicks any of the buttons after choosing and starting their game.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.3
Test Case Title	Add Player
Inputs	add(player);
Expected Results	Player successfully added to session and model with unique id, with their socket listening for a ‘verify’ call to run the verify() function.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player chooses a game and joins it.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.4
Test Case Title	Check Room
Inputs	checkRoom()
Expected Results	If there is room in the sessions still, return true; If not, return false;
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player chooses a game and joins it, server called this function.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.5

Test Case Title	Remove Player
Inputs	remove(player);
Expected Results	Removes the passed player from the session If the session is empty afterwards, removes the session too. If the session is not empty, calls dropPlayer() function.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player closes the window after joining a game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.6
Test Case Title	Drop Player
Inputs	dropPlayer(num);
Expected Results	Calls model's dropPlayer() function. Reassigns player numbers to remove the gap
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player closes the window after joining a game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.7
Test Case Title	Update Gamestate

Inputs	updateGameState();
Expected Results	Calls model's getGameState() function. Sends the received gamestate to all players in session
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player joins a game then makes any move that the server accepts.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.12.8
Test Case Title	Construct Session
Inputs	serverSingleSessionController sssc = new serverSingleSessionController(model);
Expected Results	Attaches passed model to self Initializes an array of players
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player attempts to join any game.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.1
Test Case Title	Construct Client

Inputs	Client c = new Client(socket, server);
Expected Results	Client created successfully Socket listens for login and join calls.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player joins any game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.2
Test Case Title	Set Session Server
Inputs	setServer(server);
Expected Results	Server attached to client.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player joins any game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.3
Test Case Title	Your Turn
Inputs	yourTurn();
Expected Results	Emits a ‘your turn’ call to the client

Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. The previous player in turnOrder finishes their move.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.4
Test Case Title	Initialize Player
Inputs	init(num);
Expected Results	Emits an ‘id’ call to the client with their number attached
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player joins any game.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.5
Test Case Title	Deny Move
Inputs	deny();
Expected Results	Emits a ‘deny’ call to the client
Actual Results	Match expected results.
Anomalies	N/A

Date and Time	2/8/19
Procedure Step	1. Player attempts to make any move they are not allowed to make.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.6
Test Case Title	Drop Player
Inputs	dropPlayer(num);
Expected Results	Emits a ‘drop’ call to the client with a player number attached
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player closes tab after joining a game.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.7
Test Case Title	Update
Inputs	update(gamestate);
Expected Results	Emits an ‘update’ call to the client with a gamestate attached
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Any player makes any valid move
Attempts to Repeat	N/A

Testers	Sean Hardin(manual)

Test Case ID	3.13.8
Test Case Title	Remove
Inputs	remove();
Expected Results	Calls the server's remove() function
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player closes tab after joining game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.9
Test Case Title	Login
Inputs	login(user,pass);
Expected Results	Calls the mainServer's databaseModel/s login function
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player enters their info and hits enter.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.10
Test Case Title	Authenticate
Inputs	authenticate(data);
Expected Results	Attach the data to the client. Send ‘authenticated’ call with data attached.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player enters their info and hits enter.
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.13.11
Test Case Title	Join
Inputs	join(game);
Expected Results	mainServer runs add() function on player, game.
Actual Results	Match expected results.
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	1. Player chooses and joins any game
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.14.1
Test Case Title	On update dealer receives correct hand

Inputs	update = [[1,10],[2,5]]; AI.update(update); AI.getHand();
Expected Results	[1,10]
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.2
Test Case Title	Dealer Calculates Correct Hard Total
Inputs	update = [[2,10],[2,5]]; AI.update(update); AI.getTotal();
Expected Results	12
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.3
Test Case Title	Dealer Calculates the Correct Soft Total
Inputs	update = [[1,1],[2,5]]; AI.update(update);

	AI.getTotal();
Expected Results	12
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.4
Test Case Title	Hit on Soft Flag Works Correctly

Inputs	AI = new dealerAI(true);
	update = [[1,6],[2,5]];
	AI.update(update);
	Var move = AI.yourTurn();
	console.log(move);
	AI = new dealerAI(false);
	update = [[1,6],[2,5]];
	AI.update(update);
	Move = AI.yourTurn();
	console.log(move);

Expected Results	‘Hit’ ‘Stay’
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.5
Test Case Title	PlayerAI Initializes with Correct Values

Inputs	AI = new PlayerAI(1, 10, true, true, false, true, true);
	console.log(AI.hand);
	console.log(AI.dealerHand);
	console.log(AI.bank);
	console.log(AI.softLimit);
	console.log(AI.doubleAfterSplitAllowed);
	console.log(AI.basicStrategy[0][9][0]);
	AI = new PlayerAI(1);
	console.log(AI.basicStrategy[0][9][0]);

Expected Results	[] [] 1000 20 False 'Rs' 'S'
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.6
Test Case Title	PlayerAI Splits
Inputs	AI = new PlayerAI(1, 10, true, true); AI.update([[-1,4],[7,7]]); console.log(AI.yourTurn());
Expected Results	'split'
Actual Results	Matched expected results

Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.7
Test Case Title	PlayerAI Doubles Down
Inputs	AI = new PlayerAI(1, 10, true, true, true); AI.update([[-1,4],[4,5]]); console.log(AI.yourTurn());
Expected Results	‘double’
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.8
Test Case Title	PlayerAI Hits When Double is Disabled
Inputs	AI = new PlayerAI(1, 10, true, true); AI.update([[-1,4],[4,5]]); console.log(AI.yourTurn());
Expected Results	‘Hit’
Actual Results	Matched expected results
Anomalies	N/A

Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.9
Test Case Title	PlayerAI Surrenders
Inputs	AI = new PlayerAI(1, 10, true, true, true, true, true); AI.update([[-1,1],[10,5]]); console.log(AI.yourTurn());
Expected Results	‘surrender’
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19
Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.14.10
Test Case Title	PlayerAI Stays When Surrender is Disabled
Inputs	AI = new PlayerAI(1, 10, true); AI.update([[-1,4],[4,5]]); console.log(AI.yourTurn());
Expected Results	‘stay’
Actual Results	Matched expected results
Anomalies	N/A
Date and Time	2/8/19

Procedure Step	N/A
Attempts to Repeat	N/A
Testers	Jeffrey Talada(manual)

Test Case ID	3.10.5
Test Case Title	Bet
Inputs	client.yourTurn(); return ‘bet’
Expected Results	Client.bet shows given bet
Actual Results	Client.bet shows given bet
Anomalies	none
Date and Time	2/20/19
Procedure Step	1. Add player to game 2. Run game 3. Have player return bet
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.6
Test Case Title	Start
Inputs	client.yourTurn(); return ‘start’
Expected Results	Game starts
Actual Results	Game starts
Anomalies	none
Date and Time	2/20/19
Procedure Step	1. Add player to game

	2. Run game 3. Have player return start
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.7
Test Case Title	split
Inputs	client.yourTurn(); return ‘split’
Expected Results	client.getHand() shows two hands
Actual Results	client.getHand() shows two hands
Anomalies	none
Date and Time	2/20/19
Procedure Step	1. Add player to game 2. Run game 3. Have player return split
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.8
Test Case Title	Double down
Inputs	client.yourTurn(); return ‘double_down’
Expected Results	client.getBet() shows double bet
Actual Results	client.getBet() shows double bet
Anomalies	none
Date and Time	2/20/19

Procedure Step	1. Add player to game 2. Run game 3. Have player return double down
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)

Test Case ID	3.10.9
Test Case Title	surrender
Inputs	client.yourTurn(); return ‘surrender’
Expected Results	client.getBet() shows half the bet
Actual Results	client.getBet() shows half the bet
Anomalies	none
Date and Time	2/20/19
Procedure Step	1. Add player to game 2. Run game 3. Have player return surrender
Attempts to Repeat	N/A
Testers	Sean Hardin(manual)