Evidence Gathering Document for SQA Level 8 Professional Developer Award.

This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

Fill in each point with screenshot or diagram and description of what you are showing.

Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

Evidence Key: A&D - Analysis and Design Unit
I&T - Implementation and Testing Unit
P - Project Unit

Week 1

Unit	Ref	Evidence
I&T	I.T.6	Demonstrate the use of an object literal in a program. Take screenshots of: *An object literal in a program *A function that uses the object *The result of the function running

Programme code in ATOM

```
const person = {
  name: 'David',
  age: 32,
  hobby: 'Coding',
};

function introduceYourself(){
  return `Hello, my name is ${person.name}. I am ${person.age} years old and
          really enjoy ${person.hobby}!`
};

console.log(introduceYourself());

```

Code running in Terminal:

```
Hello, my name is David. I am 32 years old and
          really enjoy coding!
```

Above is a simple programme I wrote that takes the 'person' object and uses the key: value pairs to output an introduction. You can see the output in the terminal above

Unit	Ref	Evidence
I&T	I.T.5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running

Programme code in ATOM

```
const shoppingList = [
  {item: 'Milk' , price: 1.50},
  {item: 'Bread' , price: 1.00},
  {item: 'Coffee' , price: 4.00},
  {item: 'Washing Powder' , price: 5.60},
  {item: 'Cereal' , price: 2.00}
];
 const getPrice = shoppingList.map((item)=>{
   return item.price
 })

 const totalPrice = getPrice.reduce((total, number)=>{
   return total + number
 })
console.log(`The total price of your shopping is ${totalPrice}`);
```

Code running in Terminal

```
➜  Calculator node introduction.js
The total price of your shopping is 14.1
```

The above code uses an array of shopping items and their price. It uses the map method to create an array of just the price of each item (getPrice) and then uses the reduce method to loop through each element in the array and return a total (totalPrice)

Week 2

Unit	Ref	Evidence
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing

Error in test code

```
24    highestCard(card1 card2){
25      if(card1.value > card2.value){
26        return card.name
27      }
28      else{
29        card2
30      }
31    }
32  }
```

```
      23 |
    > 24 |     highestCard(card1 card2)
{
         |                          ^
      25 |       if(card1.value > card2
.value){
      26 |         return card.name
      27 |       }
```

Code amended and passing

```
13    highestCard(card1, card2){
14      if(card1.value > card2.value){
15        return card1
16      }
17      else{
18        card2
19      }
20    }
```

```
PASS specs/
  cardgame
    ✓ Can check for Ace (2ms)
    ✓ Can identify the highest card (1
ms)
    ✓ Can add total value of cards

            1 passed, 1 total
            3 passed, 3 total
```

Test Code

```
11    beforeEach(()=>{
12      cardGame = new CardGame()
13      card1 = new Card('Spades',10)
14      card2 = new Card('Hearts',1)
15      cards = [card1, card2]
16    })
17
18    test ('Can identify the highest card', ()=>{
19      cardGame.highestCard(card1, card2)
20      expect(cardGame.highestCard(card1, card2)).toBe(card1)
21    })
```

Above is an example of code that I had to test. The code was poorly written and as a result would not run in the testing suite. I had to correct the code and and write a test to ensure the function performed as expected. The above screenshots include before and after views of the source code and test outcome and finally the testing script I wrote

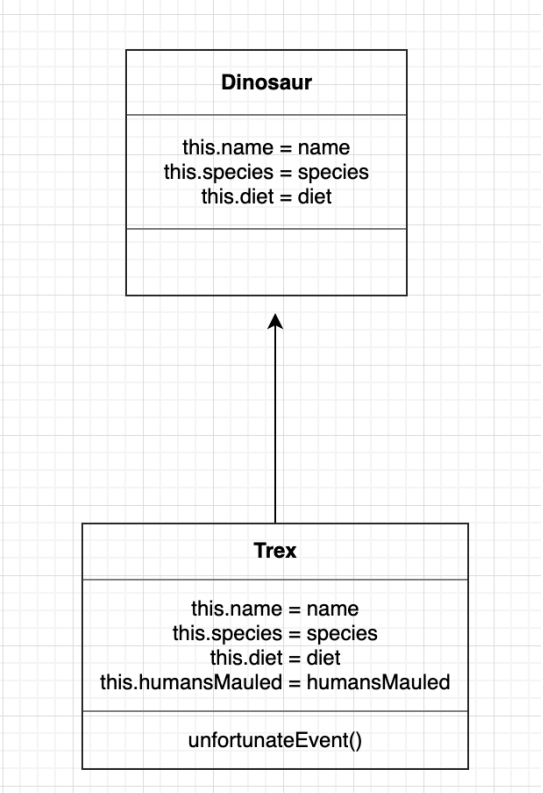
Unit	Ref	Evidence
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.

```
class Runner{

  constructor(){

    let distanceRan = 6; //variable only defined within the constructor

    this.getDistanceRan = function(){
      return distanceRan
    };

    this.setDistanceRan = function(extraLap){
      distanceRan += extraLap;
    };
  }
}

var myDistance = new Runner();
console.log(myDistance.getDistanceRan());
myDistance.setDistanceRan(14)
console.log(myDistance.getDistanceRan());
console.log(myDistance.distanceRan);
```

```
➜  encapsulation node runner.js
6
20
undefined
```

Unit	Ref	Evidence
A&D	A.D.5	An Inheritance Diagram

This is an inheritance diagram I created which illustrates properties being passed from the dinosaur class into the Trex class which are then used in a method



Unit	Ref	Evidence
I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screenshots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.

Dinosaur Class with name, species and diet properties

```
class Dinosaur{

  constructor(name, species, diet) {
    this.name = name;
    this.species = species;
    this.diet = diet;
  }
}
module.exports = Dinosaur;
```

Trex class which inherits properties from Dinosaur class.

```
const Dinosaur = require ('./dinosaur');

class Trex extends Dinosaur{
  constructor(name, species, diet, humansMauled){
    super (name, species, diet);
    this.humansMauled = humansMauled
 }

 unfortunateEvent(){
   console.log(`${this.name} is a ${this.species} which makes it a ${this.diet}.We had to close Jurassic Park because
   it mauled ${this.humansMauled} humans`);
 }
}

module.exports = Trex
```

New instance of Trex has been created using properties inherited from Dinosaur Class and Trex class.

```
➜  class_inherit node newsReport.js
Mark is a T-Rex which makes it a Carnivore.We had to close Jurassic Park
 because it mauled 7 humans
```

```
const mark = new Trex('Mark', 'T-Rex', 'Carnivore',7);

console.log(mark.unfortunateEvent())
```

Week 3

Unit	Ref	Evidence
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.

```
dealCards(){
  for (const card of this.cards){
    this.players[0].addCard(card);
    this.rotatePlayers();
  }
}
```

This algorithm was used to deal cards in a top trumps card game. The algorithm Loops through a deck of cards and deals a card (by calling on the addCard function I wrote) to each player. Cards are dealt to both players by calling the rotatePlayers function on each iteration.

```
checkForWinner(){
  if (this.players[0].handEmpty()) this.winner = this.players[1];
  if (this.players[1].handEmpty()) this.winner = this.players[0];
}
```

This algorithm will check for the winner of the game by calling the handEmpty function. This function checks if the player has any cards left in their hand(if hand.length === 0 then this is true). If this is true for either player then the other player wins.

Week 4

Unit	Ref	Evidence
I&T	I.T.3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running

```
//DELETE guitars
router.delete('/:id', function(req, res){
  SqlRunner.run('DELETE FROM guitars WHERE id = $1',[req.params.id]).then((result)=>{
    SqlRunner.run('SELECT * FROM guitars ORDER BY brand ASC').then((result) =>{
    res.status(201).json(result.rows);
  })
})
});
```

id	brand	type	strings
21	Fender	Electric	6
22	Gibson	Electric	6
23	Ibanez	Electric	7

```
DELETE FROM guitars WHERE id = 21
```

Load Query... Save Query...

d	brand	type	strings
22	Gibson	Electric	6
23	Ibanez	Electric	7

The function I have created to delete a guitar searches for a particular guitar by its ID and deletes it from the database. We can see this in action and the result of this in Posterico to the left. In this case we have searched for an item with the ID '21' and removed it

Unit	Ref	Evidence
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running

```
// GET guitar by Brand
router.get('/', function(req, res) {
  SqlRunner.run('SELECT * FROM guitars ORDER BY brand ASC').then((result) => {
    res.status(200).json(result.rows)
  });
});
```

The above function will sort the guitars by brand alphabetically. Below is the programme running.

id	brand	type	strings
19	Fender	Electro/Acoustic	6
14	Gibson	Electric	2
17	Ibanez	Acoustic	12

Unit	Ref	Evidence
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running

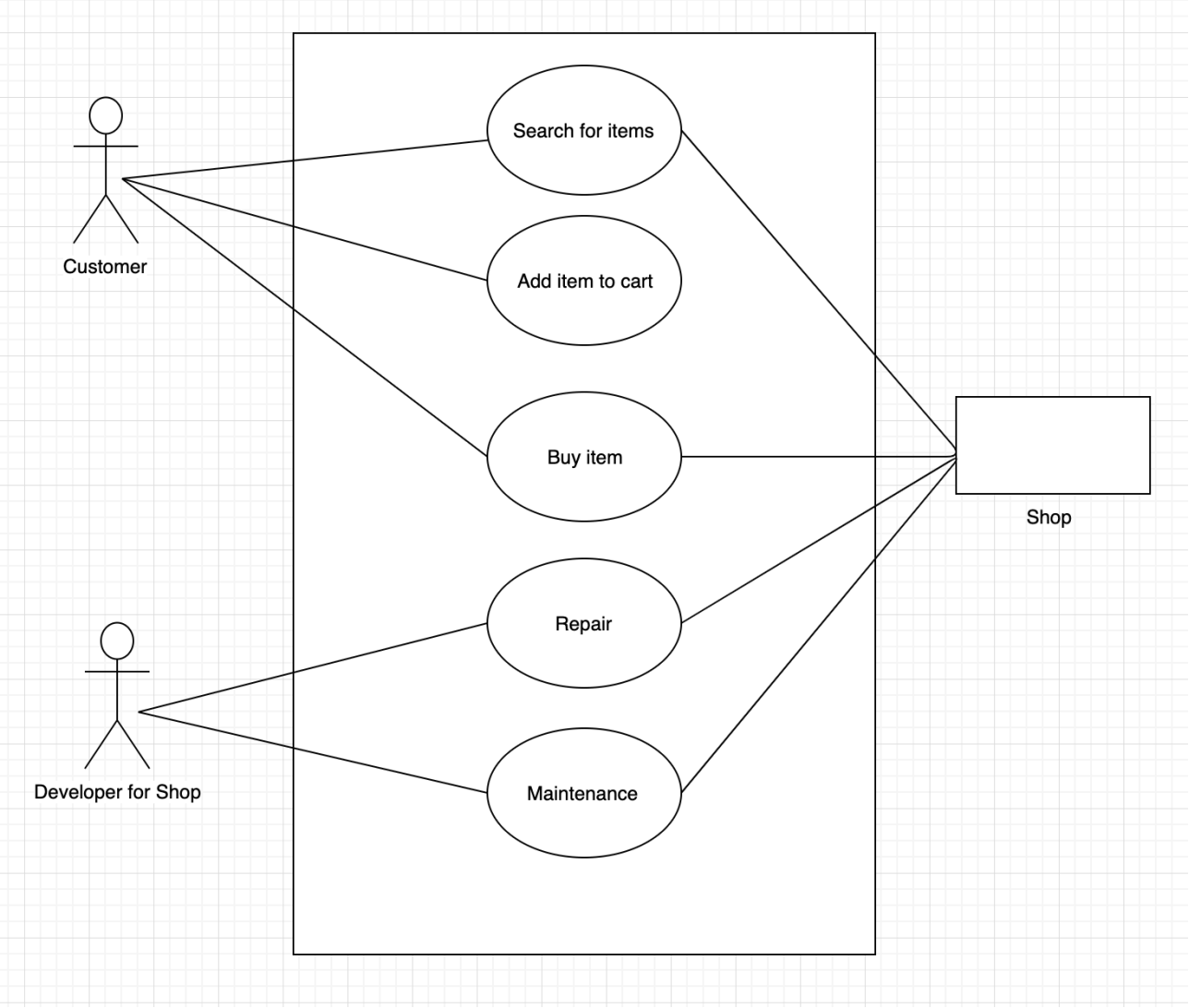
```
const mapDispatchToProps = (dispatch) =>({
    getCountriesData(){
      dispatch(()=>{
        fetch('https://restcountries.eu/rest/v2/all?fields=name;flag')
        .then (res => res.json())
        .then(countriesData =>{
          countriesData.map((country,index)=>{
            country.visited=false;
            country.id=index;
          })
          dispatch({
            type: 'GET_COUNTRIES',
            countriesData,
          })
        })
      })
    },
    selectedCountry(country){
      dispatch({
        type: 'ADD_TO_BUCKETLIST',
        bucketList: country,
      })
    }
})

export default connect(mapStateToProps, mapDispatchToProps)(Form)
```

Above is an API call collecting data from the restcountries API and returns the name and flag of each country.

My application then uses the data from the api and when a country is selected from the dropdown menu (top left) a stamp of each country will appear on the passport

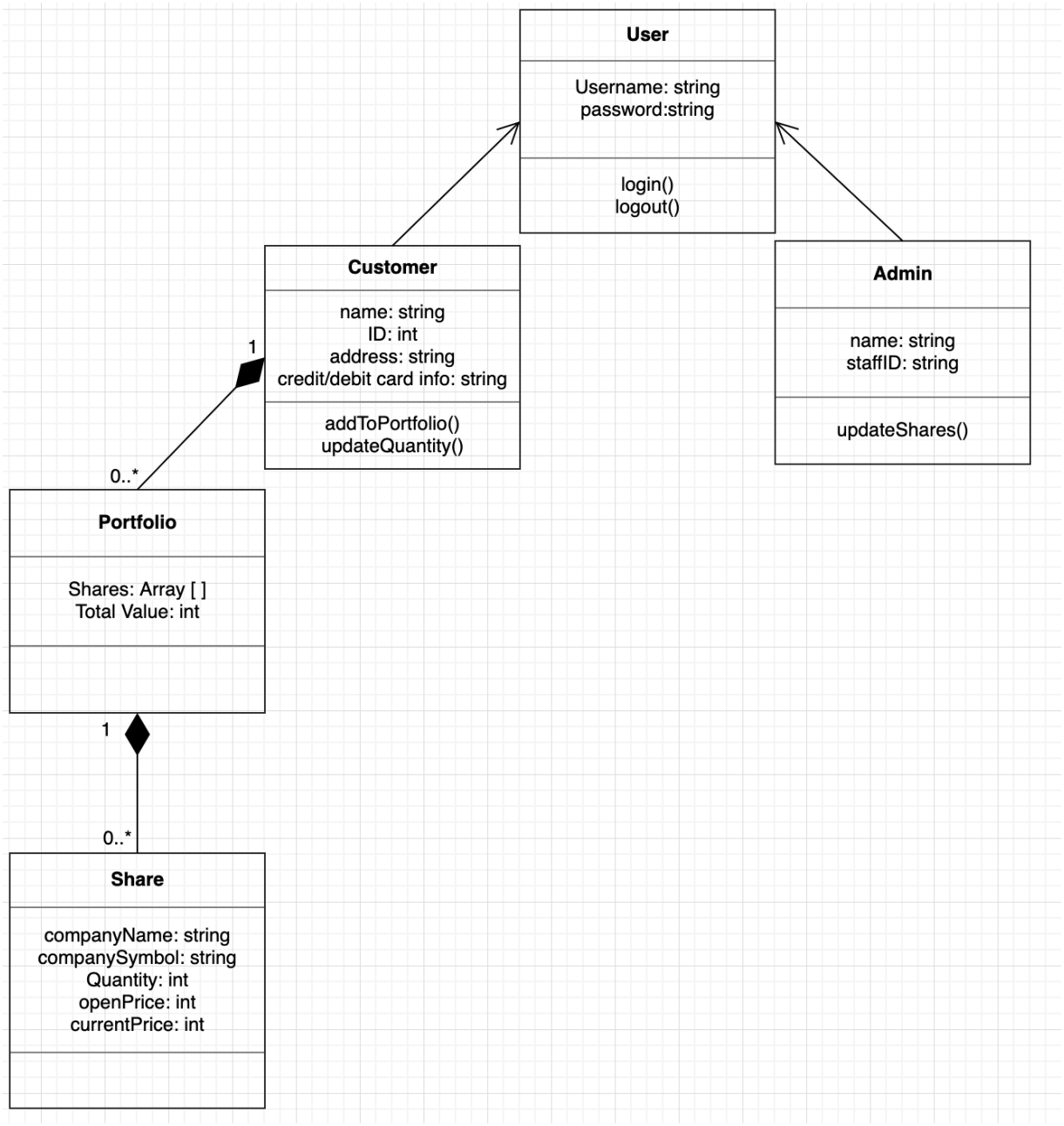
Unit	Ref	Evidence
A&D	A.D.1	Produce a Use Case Diagram

The following is a Use Case diagram for an online shop. The diagram highlights a customer, developer for the shop and the shop itself as the actors and their usage requirements



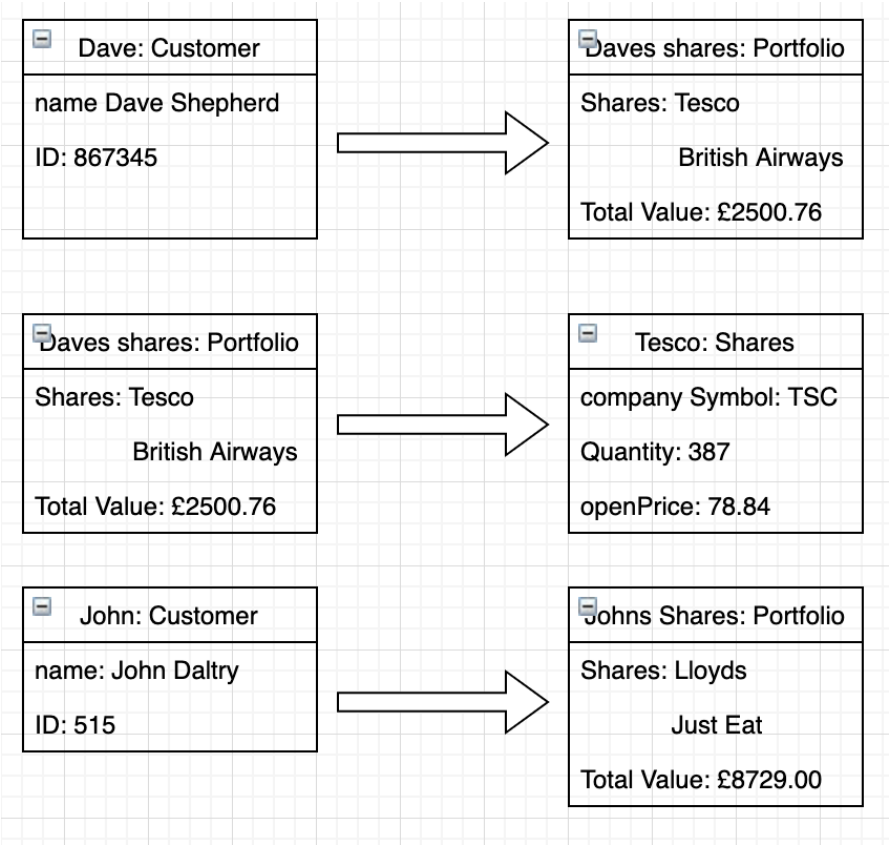
Unit	Ref	Evidence
A&D	A.D.2	Produce a Class Diagram

The following is a class diagram for my ShareTracker App highlighting the structure of the system and relationship between each class



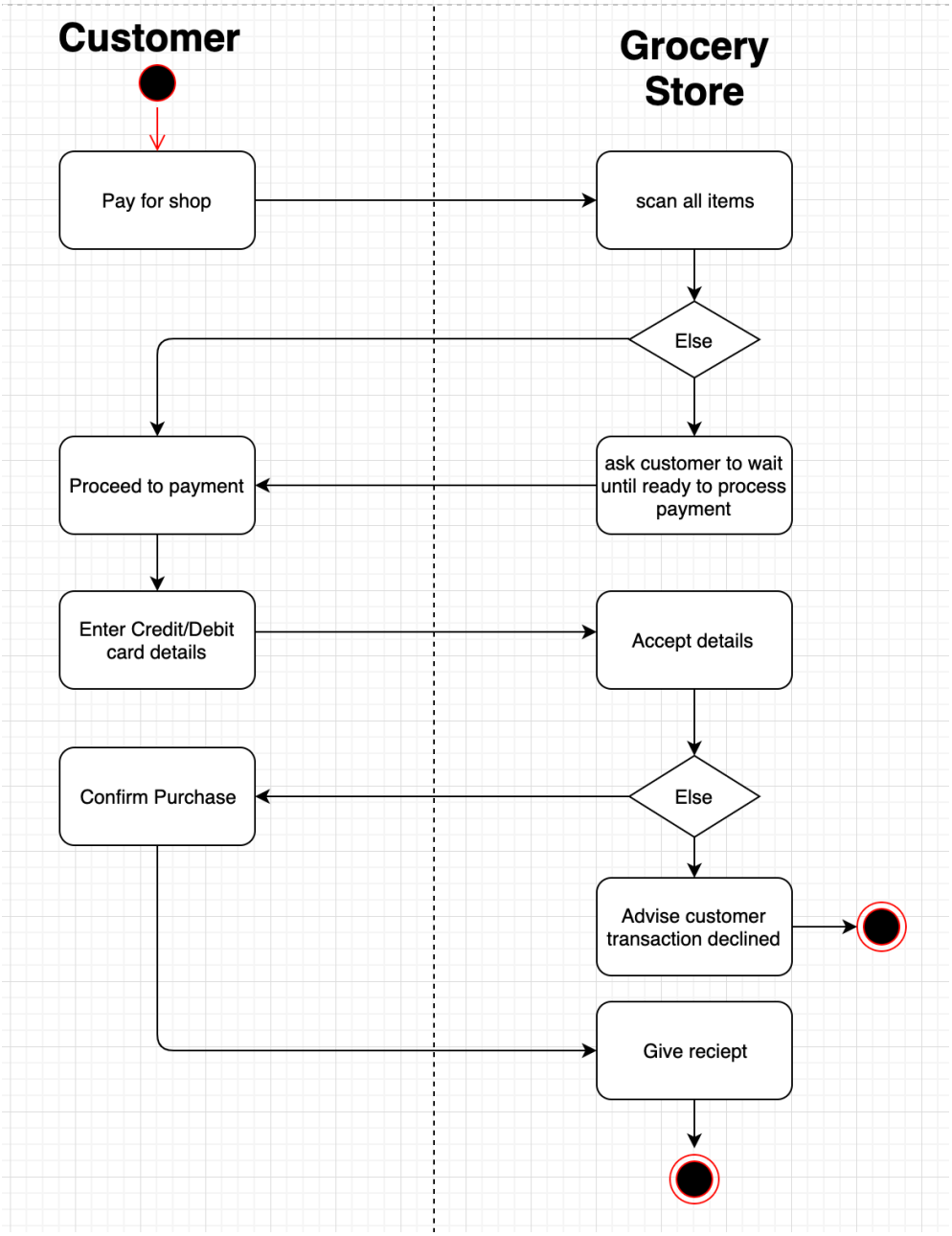
Unit	Ref	Evidence
A&D P	A.D.3 P.8	Produce three Object Diagrams

The following are object diagrams of various instances of classes based on my class diagram above.



Unit	Ref	Evidence
A&D	A.D.4	Produce an Activity Diagram

The following is an example of an activity diagram highlighting the process of a customer completing an online shop



Unit	Ref	Evidence
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time

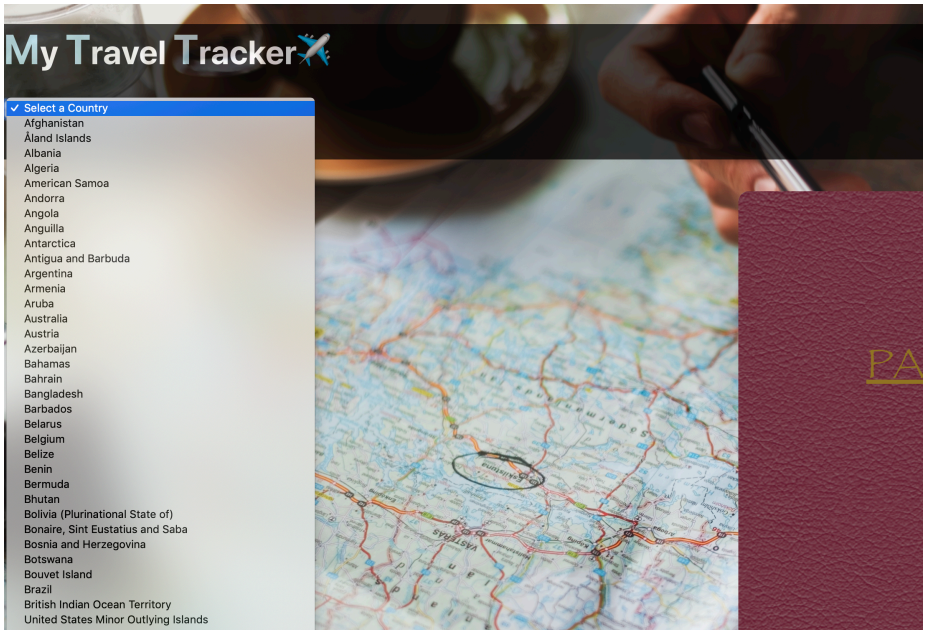
Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	Backend uses SQL which can be quite slow and clunky to run meaning certain calls could run slowly. This is an issue because it could drastically impact the user experience	Update this to using MongoDB instead
Performance Requirements	Limitations to the amount of times that the user can call from the API in an hour. This could stop the app from working	As an individual user, I would not anticipate that a user would refresh this data once every 2 minutes for an hour
Persistent Storage transactions	Using an API that has already been established includes data that is not required for my app. This could impact performance as we are handling more data than is required	Create own API that only includes the data that is necessary for my application
Usability	Considering some of the users might be visually impaired. If page is not easy for users to use/ read from then they will not use it	I have designed the page very simple and made the writing and buttons large so it is easy to read and navigate through.
Budgets	There are budget limitations that have been set which this project will need to abide by	Sufficient planning needs to be carried out in the early stages of the project to ensure spending phases are agreed and documented
Time	Timescales have been established and agreed that the project must deliver within	Planning and agreeing activities (including MVP) at intial stages will allow us to track progress and ensure timescales are kept to

Week 5

Unit	Ref	Evidence
P	P.10	Example of Pseudocode used for a method

```
// if the customer clicks on the remove button on the country card
// remove the country from the virtual passport array
```

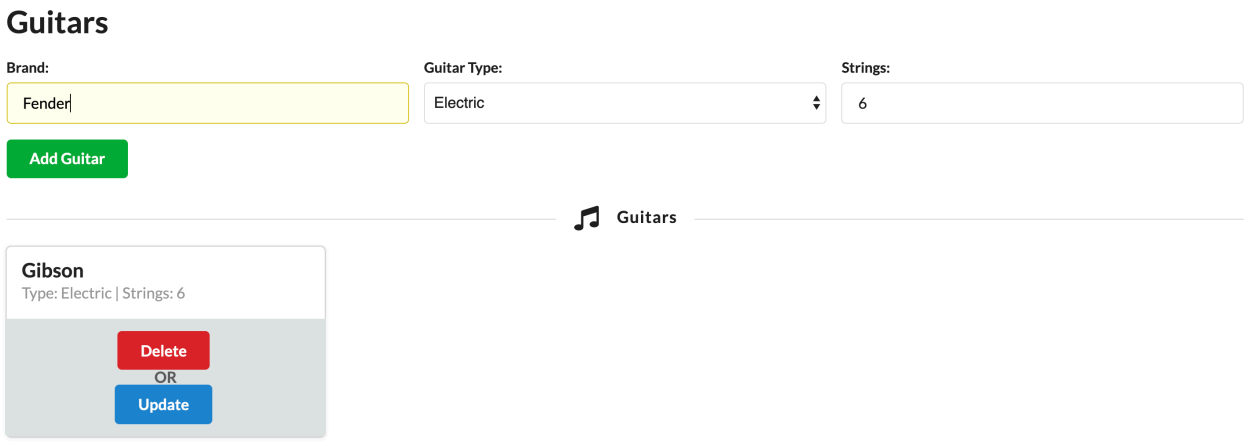
Unit	Ref	Evidence
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way

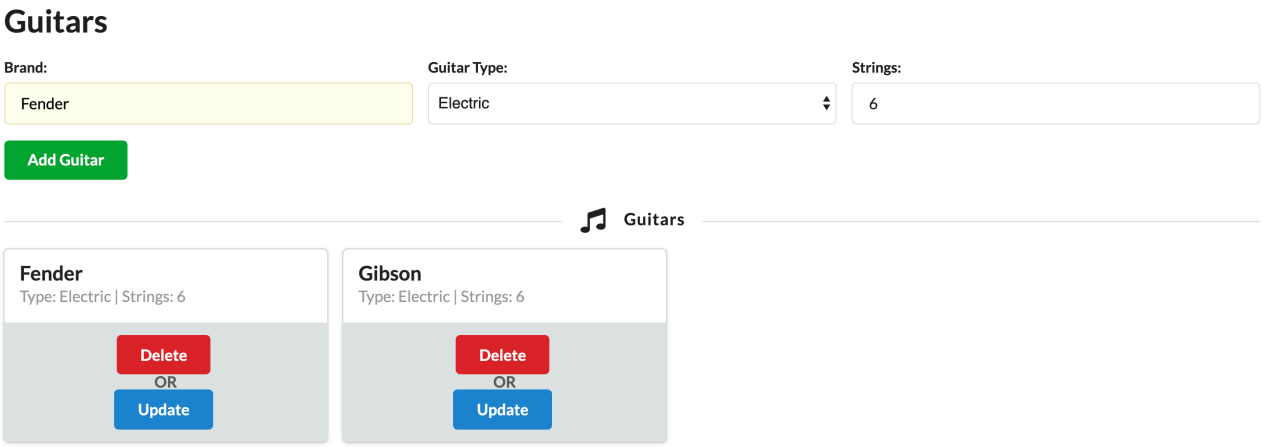


The above is an example of my travel tracker/bucket list app where the user can select a country they want to visit from a dropdown list and will be saved in their passport.

Unit	Ref	Evidence
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved

```
postGuitars(guit) {
  const url = `http://localhost:3000/guitars`;
  const request = new RequestHelper(url);
  request.post(guit)
    .then((guitars) => {
      PubSub.publish('Guitars:guitar-data-loaded', guitars);
    })
    .catch(console.error);
```





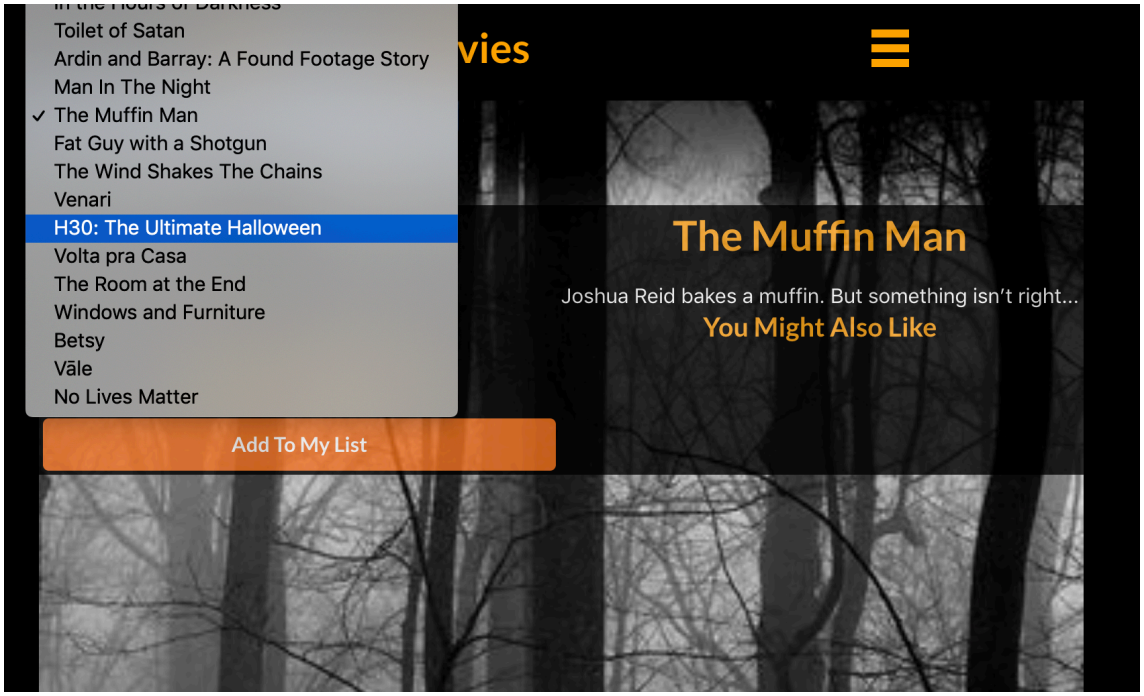

```
POST /guitars 201 8.997 ms - 115
```

The above code shows a user completing a form to add a guitar on a webpage. The post request to handle this is also highlighted above. Once the user clicks on the Add Guitar button the new entry is saved.

Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program

```
render(){
  console.log(this.state.selectedIndex);
  return(
    <>
      <div className = "fields">
      <select value = {this.state.selectedIndex} onChange = {this.handleChange}>
      {this.createList()}
      </select>
      </div>
    </>
  )
}
```

This is the code that handles the user input when the select a move from a dropdown list



The user selects a movie from the dropdown menu

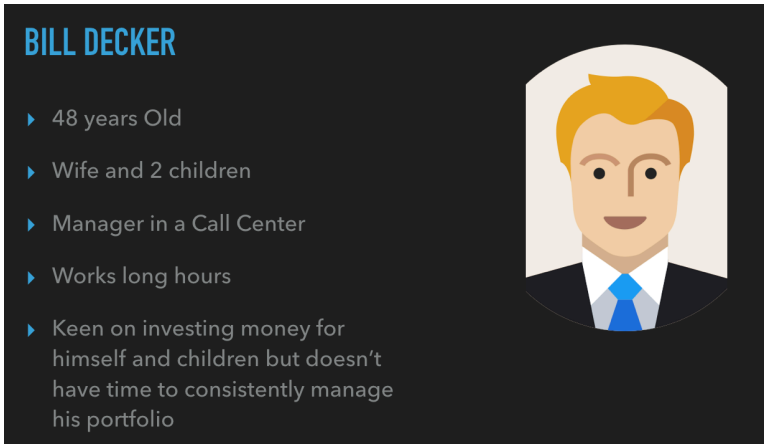
Once selected the expected movie will be displayed on screen with the relevant information

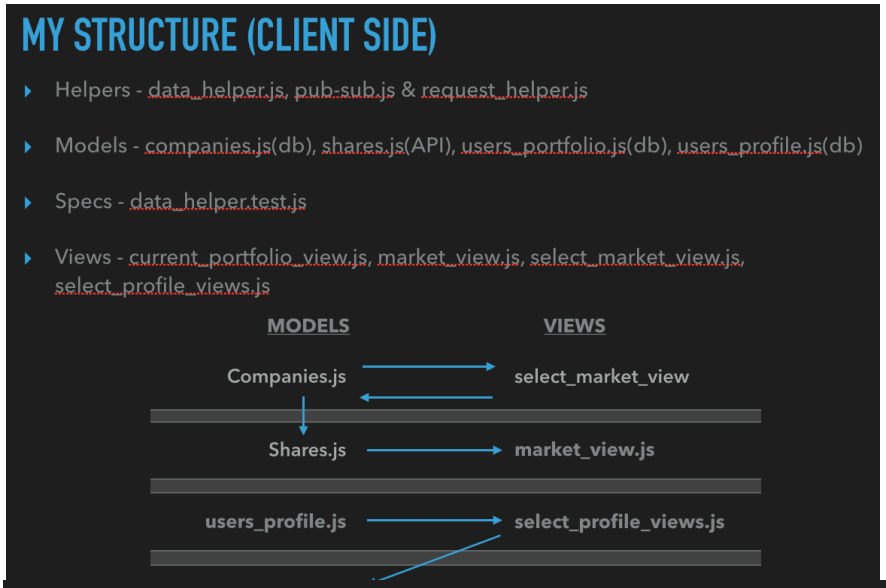
Unit	Ref	Evidence
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.

https://github.com/KBroughCode/My_Horror_Movies

Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.

The following screenshots are where I defined my customer, established my frontend structure and the relationship between files and my server side structure which includes information on the API I used and database files





Week 7

Unit	Ref	Evidence
P	P.17	Produce a bug tracking report

Bug/Error	Solution	Date
Could not get deal card function to stop dealing when user clicks snap button	Use the clearInterval() function in order to stop the setInterval function used to execute dealcards()	3/06/2019
Could not pass dealCards( ) function to component file from container file in react	Needed to add ‘this.’ Before the function as we were passing this from a class	3/06/2019
Information is being mutated in the reducer.	Added spread (‘…’) before state so we were creating a new array and not manipulating the original	1/06/2019
Could not get page to render to screen	Pass Blackjack into app.js render function	1/06/2019
Cards dealing too fast in snap game	Adjust the setInterval value from 1000ms to 1500ms	3/06/2019

Week 9

Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.

Project Brief

Create an online card game application that will allow a user to play one or more card games against the computer. This should include a

MVP

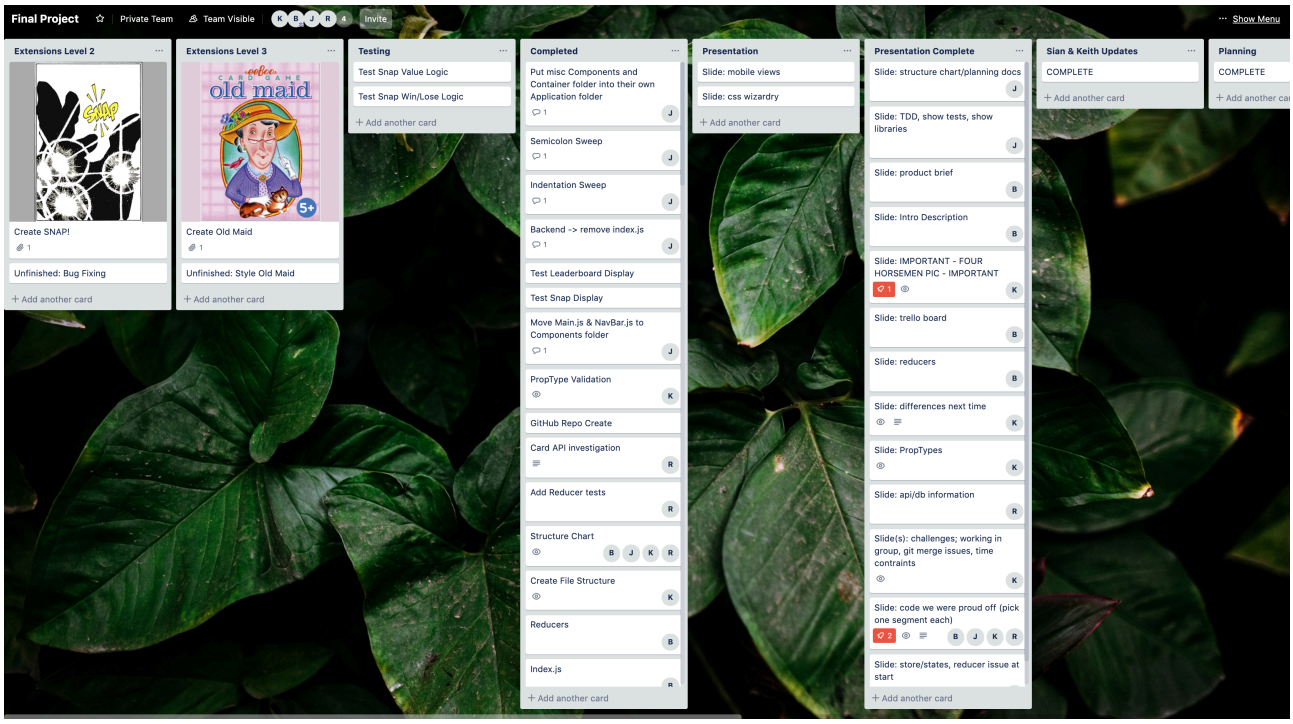
There should be at least one fully functional game and the option for a user to register their high score.

Expectations:

- All members of the group contributing to the planning, development and presentation of the project
- Members supporting each other to make sure everyone can get the most they can from the week
- Some application of Agile concepts e.g. a morning standup, sprints, a kanban board (Trello)

Technical Requirements:

- TDD - unit testing of models and Redux reducers where appropriate
- Frontend testing using react-testing-library
- API calls
- React, Redux and MongoDB
- Regular Git commits and use of branches and PRs. We are looking to see at least 100 commits.
- Validating props with prop-types

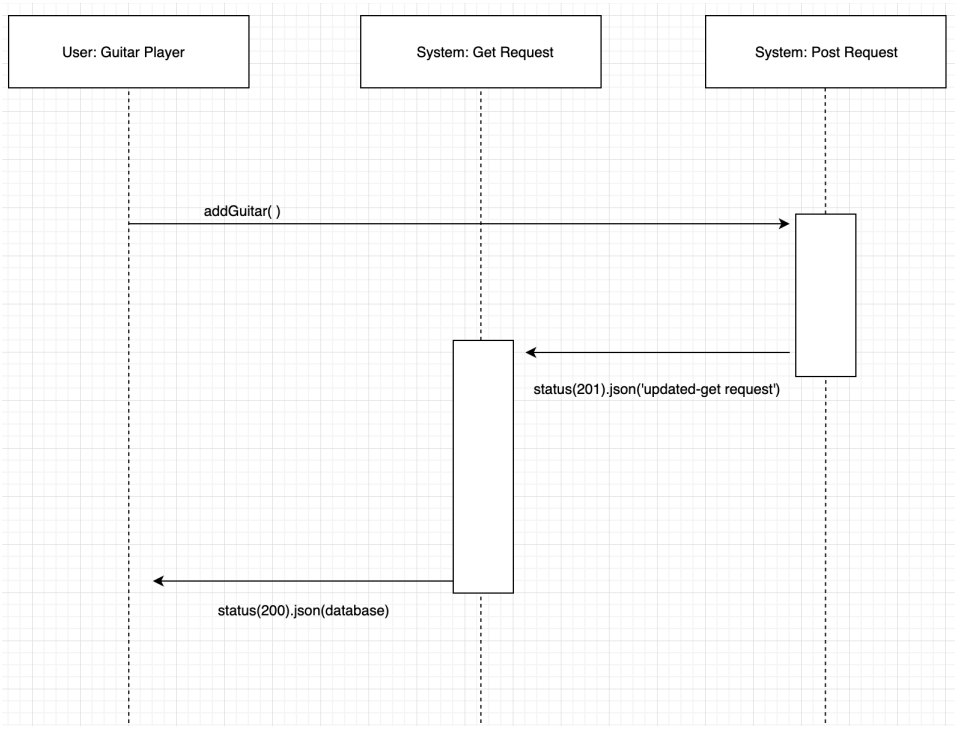


Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

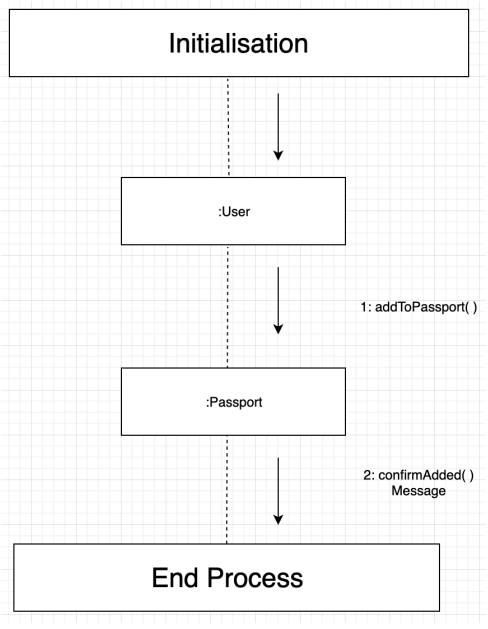
Acceptance criteria	Expected result	Pass or Fail
A user is able to select to play 1 of 3 games	The user should be able to access 1 of 3 games from the home page by hovering over a playing card which reveals a link to the game on hovering over it	Pass
A user is able to pick up a card in the blackjack game	When playing blackjack, the user can pick up a card from the deck by clicking on the “twist button”. This should increase their hand by 1 card	Pass
A user should be able to submit their hand of cards if they are happy with it	the user can click the stick button which will commit them to their current hand and the game will compare this to what the dealer has to determine who wins	Pass
A user should be able to submit their score once they are finished playing	Once the user is done playing, the should be able to submit their current score/coin count on the home page in an input form.	Pass
A user should be able to pause the game in snap	When the user is playing snap they should be able to pause the game by clicking the snap button	Pass
A user should be able to resume the game after pausing in snap	When the user is playing snap they should be able to resume the game after pausing by clicking the resume button	Pass

Unit	Ref	Evidence
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).

The following is a sequence diagram illustrating the interaction when the user is adding a guitar via a form submit. The post request is made updating the database and a get request is then carried out so the new guitar is now included in the data

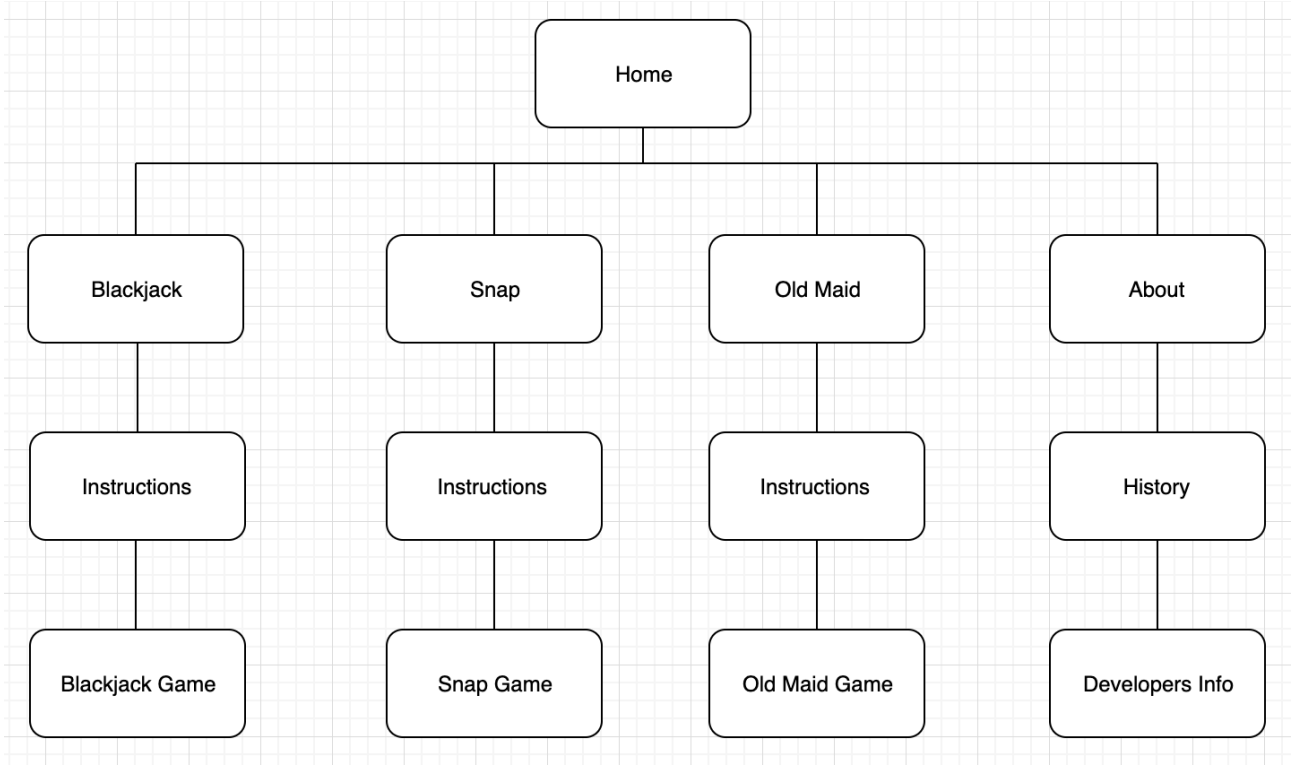


The following is a collaboration diagram highlighting adding a country from a dropdown list to a virtual passport



Week 10

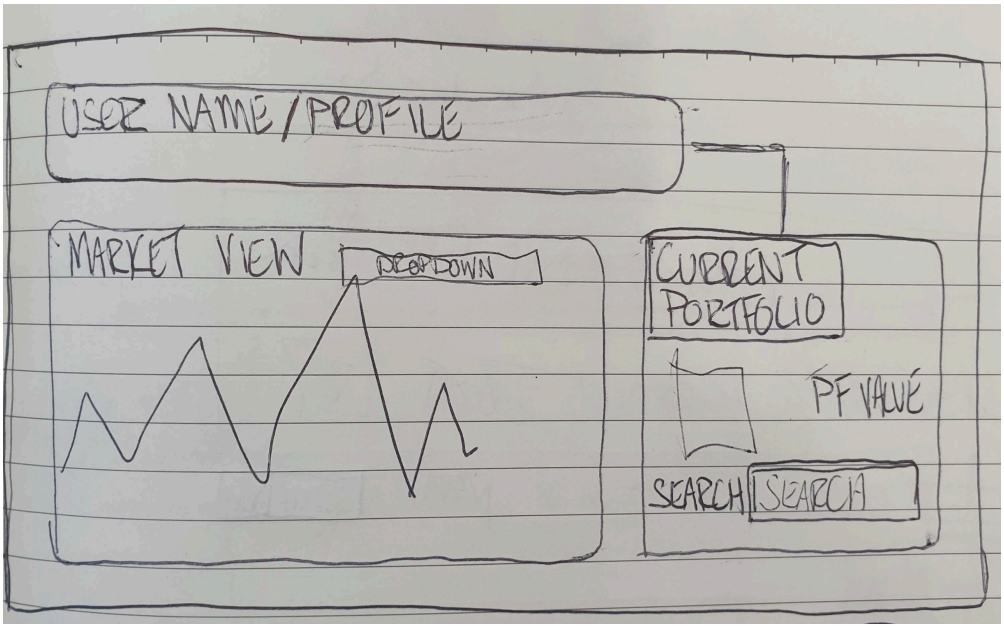
Unit	Ref	Evidence
P	P.5	User Site Map



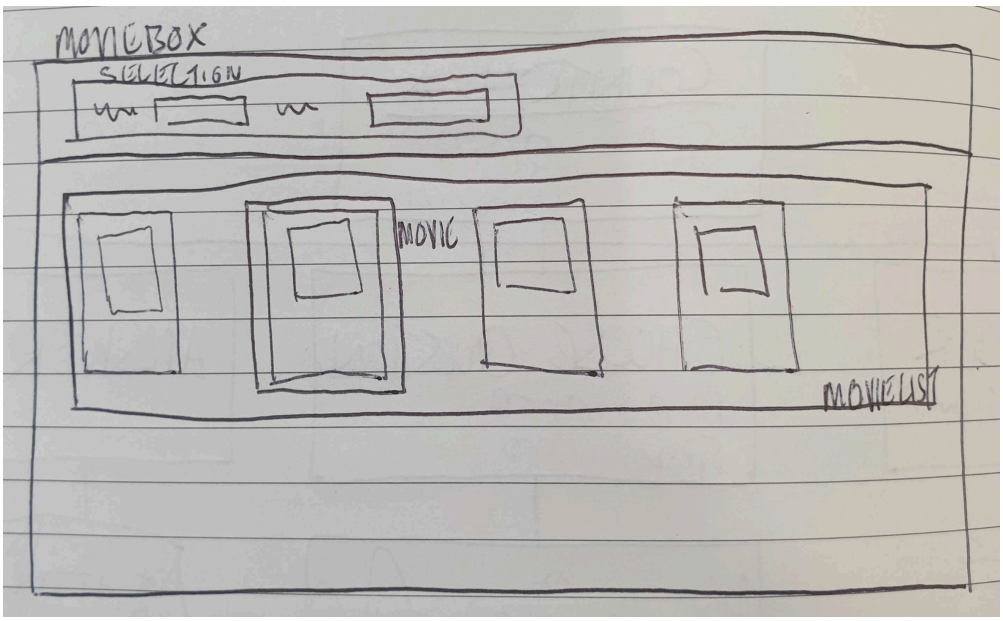
Above is a User Site Map for the online casino app developed for my final group project. The structure of the app is fairly straight forward in it's layout which I feel is reflected in the above

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

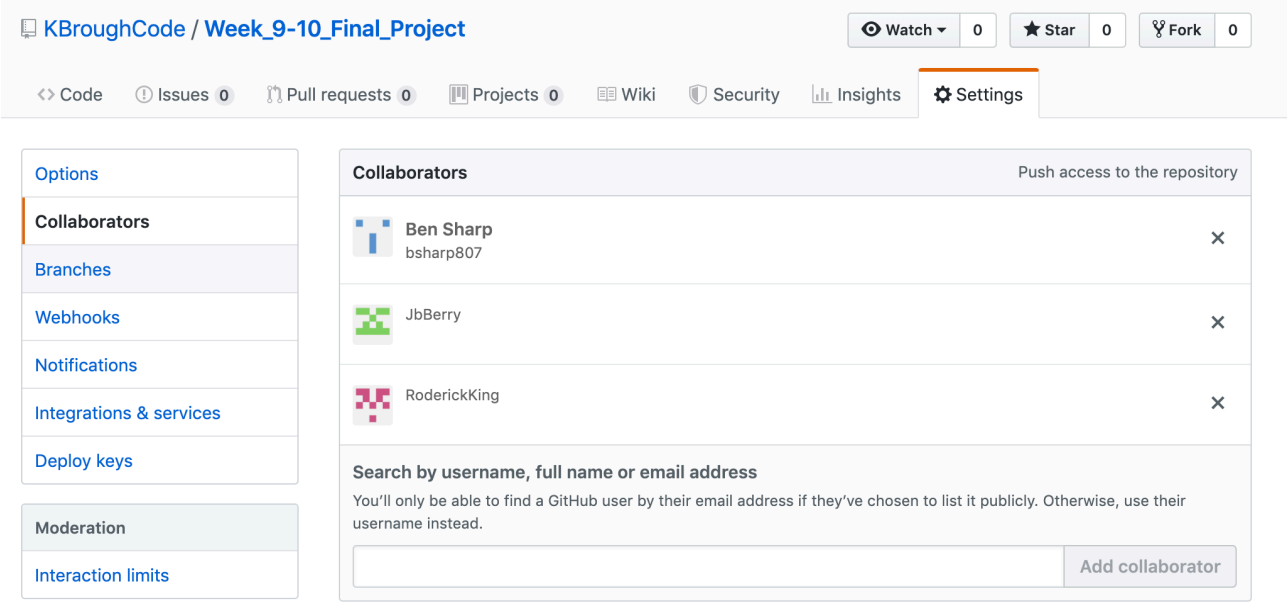
The first wireframe is for my shares management application that I created



The second wireframe is an initial wireframe created when building MyHorrorMovies application



Unit	Ref	Evidence
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.



Unit	Ref	Evidence
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.

This is my IPlay interface that will take two instruments (in this case bass and guitar) and will morph them into an IPlay element which will make up the my instruments array in the shop class.

```
public interface IPlay{
  public String makeNoise();
}
```

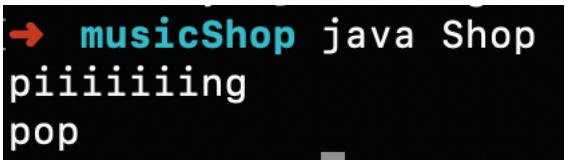
Below are the guitar and bass classes

```java
public class Guitar implements IPlay{

  private String sound;

  public Guitar(String sound){
    this.sound = sound;
  }

  public String makeNoise(){
    return "piiiiiiing";
  }
}
```

```java
public class Bass implements IPlay{

  private String sound;

  public Bass(String sound){
    this.sound = sound;
  }

  public String makeNoise(){
    return "pop";
  }
}
```

Below is the Shop class where the the myInstruments array that uses polymorphism is defined

```java
import java.util.*;

public class Shop{

  private ArrayList<IPlay> myInstruments;
  private String name;

  //constructor

  public Shop(String name){
    this.name = name;
    this.myInstruments = new ArrayList<IPlay>();

  }

  public void addInstrument(IPlay instrument) {
    this.myInstruments.add(instrument);
  }

  public String getName(){
    return this.name;
  }

  public ArrayList<IPlay> getInstruments(){
    return this.myInstruments;
  }

  public static void main(String[] args) {
    Shop pedrosShop = new Shop("Perdos Strings");
    Guitar guitar = new Guitar("Gibson SG");
    Bass bass = new Bass("Fender  Fretless Jazz");

    pedrosShop.addInstrument(guitar);
    pedrosShop.addInstrument(bass);


    for(int i=0; i<pedrosShop.getInstruments().size(); i++){
      System.out.println(pedrosShop.getInstruments().get(i).makeNoise());
    }
  }
```



Finally, this is the output of my polymorphism program in the terminal. It has taken information from the sound variables of both