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

Paste Screenshot here

Object Definition and calls to update the object

```
// Initial Data
var trip_Details = {
  start_location: "Edinburgh",
  end_location: "Inverness",
  start_time: "10:00",
  end_time: "16:00",
  travel_method: "Bus"
};

console.log("Original Data");
console.log(trip_Details);

// Now apply updates cumulatively

amended_Trip = update_Trip(trip_Details, "travel_method", "Train");
end_Itinerary = update_Trip(amended_Trip, "end_time", "13:00");

console.log("Modified Data");
console.log(end_Itinerary);
```

Function using the object

```
function update_Trip(Itinerary, what_to_change, new_value) {
  // Only update the affected field
  new_itinerary = Itinerary;

  switch (what_to_change) {
    case "start_location":
      new_itinerary.start_location = new_value;
      break;
    case "end_location":
      new_itinerary.end_location = new_value;
      break;
    case "start_time":
      new_itinerary.start_time = new_value;
      break;
    case "end_time":
      new_itinerary.end_time = new_value;
      break;
    case "travel_method":
      new_itinerary.travel_method = new_value;
      break;
    default:
  }

  return new_itinerary;
}
```

Function Running

```
➜  object git:(master) ✗ node object.js
Original Data
{ start_location: 'Edinburgh',
  end_location: 'Inverness',
  start_time: '10:00',
  end_time: '16:00',
  travel_method: 'Bus' }
Modified Data
{ start_location: 'Edinburgh',
  end_location: 'Inverness',
  start_time: '10:00',
  end_time: '13:00',
  travel_method: 'Train' }
➜  object git:(master) ✗
```

Description here

Objects are always created with the braces and a list of key pairs name and data. This distinguishes them from 'conventional' variables. It's the same begin and end notation as a function in Javascript. This is shown below.

```
var trip_Details = {
  start_location: "Edinburgh",
  end_location: "Inverness",
  start_time: "10:00",
  end_time: "16:00",
  travel_method: "Bus"
};
```

They are addressed by using a '.' . For example the start_time attribute is addressed by trip_details.start_location

Functions using the objects just accept them passed over as parameters with whatever attributes are present.

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

Paste Screenshot here

untitled array.js

```
function calculate_average(input_array) {
  let total = 0;
  let divisor = input_array.length;
  for (var i = 0; i < input_array.length; i++) {
    total += input_array[i];
  }
  return total / divisor;
}

list_of_numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

console.log("Average of 1 to 10 : ", calculate_average(list_of_numbers));
```

When running

```
➜  array git:(master) ✗ node array.js
Average of 1 to 10 :  5.5
➜  array git:(master) ✗
```

Description here

In the above snippet, the array structure holds a list of 10 numbers 1..10. In Javascript there is no requirement to state the length of the array before use and thus it can be considered a dynamic data structure. This is in marked contrast to other programming languages. in this particular instance the type of data stored is solely numeric. The array could hold Strings / Boolean values / Object or another even array. This makes them very flexible data structures

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

Paste Screenshot here

Sample Start Test Code **:**

start_code.js

```
var card = require('./card');

class CardGame{

  checkforAce(card){
    if(card = 1){
      return true;
    }else{
      return false;
    }
  }

  highestCard(card1 card2){
    if(card1.value > card2.value){
      return card.name
    }
    else{
      card2
    }
   }
  }

  static.cardsTotal(cards){
    return total;
    for(let card of cards){
      total += card.value;
      return "You have a total of " + total;
    }
  }
}
```

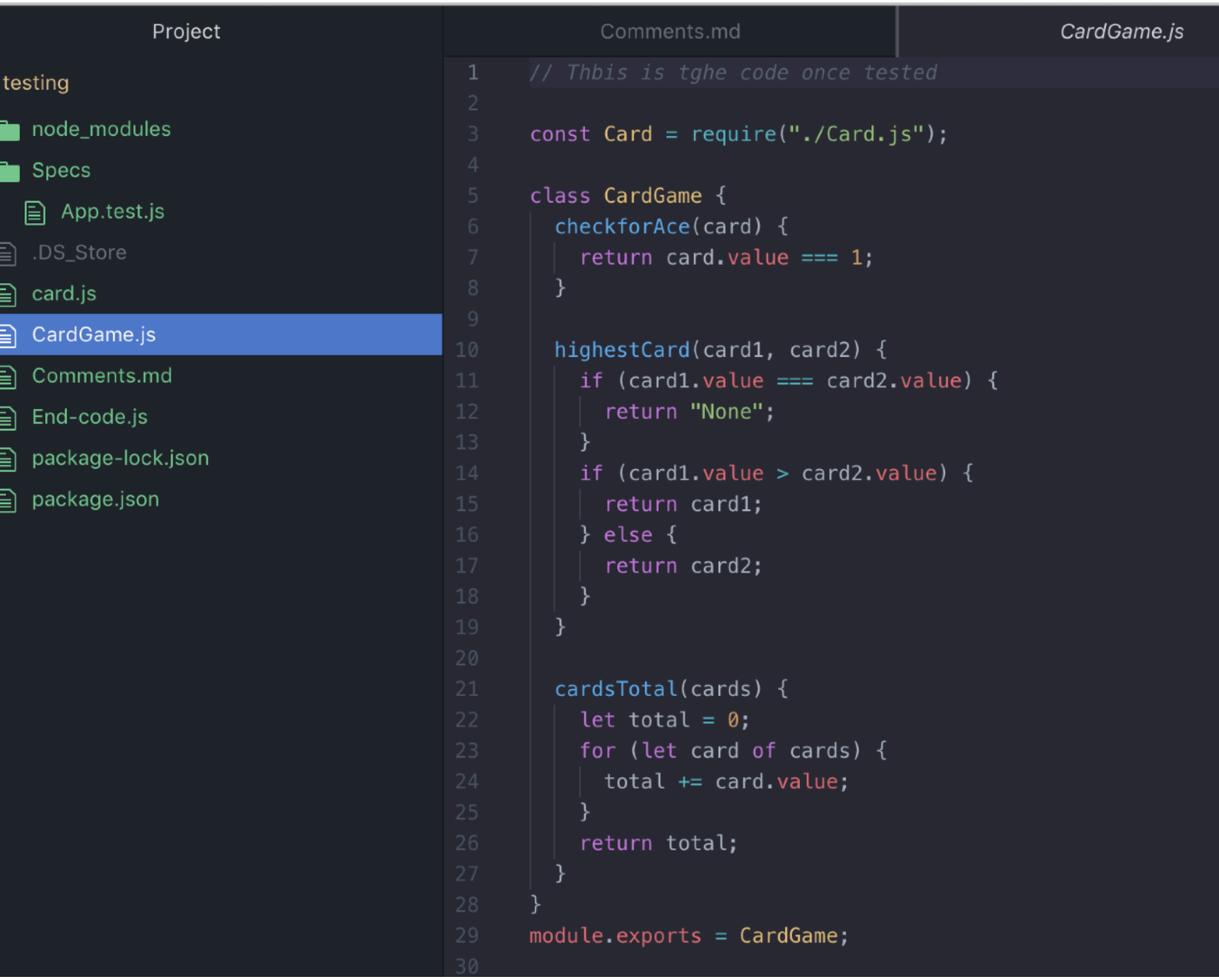
Highest Card function failing to run.

```
FAIL Specs/App.test.js
  • Test suite failed to run

    SyntaxError: /Users/a014790/Desktop/codeclan_work/PDA/testing/CardGame.
js: Unexpected token, expected "," (15:20)

      13 |   }
      14 |
    > 15 |   highestCard(card1 card2){
         |                     ^
      16 |     if(card1.value > card2.value){
      17 |       return card.name
      18 |     }

      at Parser.raise (node_modules/@babel/parser/lib/index.js:6322:17)
      at Parser.unexpected (node_modules/@babel/parser/lib/index.js:7638:16
`
```

Cards Total Function failing to run

```
    Details:

    SyntaxError: /Users/a014790/Desktop/codeclan_work/PDA/testing/CardGame.
js: Unexpected token (25:8)

      23 |
      24 |
    > 25 |   static.cardsTotal(cards){
         |         ^
      26 |     return total;
      27 |     for(let card of cards){
      28 |       total += card.value;

      at Parser.raise (node_modules/@babel/parser/lib/index.js:6322:17)
      at Parser.unexpected (node_modules/@babel/parser/lib/index.js:7638:16
)
```

End Code


Project
testing
node_modules
Specs
App.test.js
.DS_Store
card.js
CardGame.js
Comments.md
End-code.js
package-lock.json
package.json
Comments.md
CardGame.js
// Thbis is tghe code once tested
const Card = require("./Card.js");
class CardGame {
checkforAce(card) {
return card.value === 1;
}
highestCard(card1, card2) {
if (card1.value === card2.value) {
return "None";
}
if (card1.value > card2.value) {
return card1;
} else {
return card2;
}
}
cardsTotal(cards) {
let total = 0;
for (let card of cards) {
total += card.value;
}
return total;
}
}
module.exports = CardGame;

Sample Test scripts

Comments.md | *App.test.js*

```
const Card = require("../card.js");
const CardGame = require("../CardGame.js");

describe("card ", () => {
  let card1;
  let card2;
  let card3;
  let card4;
  let card5;

  let allcards;
  let total;
  cardgame = new CardGame();
  beforeEach(() => {
    card1 = new Card("Diamonds", 2);
    card2 = new Card("Spades", 1);
    card3 = new Card("Hearts", 3);
    card4 = new Card("Clubs", 7);
    card5 = new Card("Spades", 7);

    allcards = [card1, card2, card3, card4, card5];

    total = 0;
  });

  test("Card1 is not ace ", () => {
    expect(cardgame.checkforAce(card1)).toBe(false);
  });

  test("Card 1 should be greater than card 2 ", () => {
    expect(cardgame.highestCard(card1, card2)).toEqual(card1);
  });

  test("Card 3 should be  greater than card 2 ", () => {
    expect(cardgame.highestCard(card2, card3)).toEqual(card3);
  });
  test("Card 4 should be  the same as  card 5 ", () => {
    expect(cardgame.highestCard(card4, card5)).toBe("None");
  });

  test("grand total should be should be 20", () => {
    total = cardgame.cardsTotal(allcards);
    expect(total).toBe(20);
  });
});

```

Proof of test scripts passing after changes

```
> week_05@1.0.0 test /Users/a014790/Desktop/codeclan_work/PDA/testing
> jest

 PASS  Specs/App.test.js
  card
    ✓ Card1 is not ace  (3ms)
    ✓ Card 1 should be greater than card 2  (1ms)
    ✓ Card 3 should be  greater than card 2
    ✓ Card 4 should be  the same as  card 5
    ✓ grand total should be should be 20 (1ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.989s, estimated 1s
Ran all test suites.
```

Description here

Comments on above test code before running

checkforAce(card){ if(card = 1){ Comments 1,2,3 return true; }else{ return false; } }

Comments on checkforAce(card)

Comment 1 : card =1 is an assignment operator changing the value - not testing it's value.
Comment 2 : card is an object. Card.value should be tested as this is a object being passed in.
Comment 3: For simplicity - the returned value should be derived as return this.card.value === 1; } as it's a boolean value which is required.

highestCard(card1 card2){
Comment 1 if(card1.value > card2.value){ return card.name Comment 2 } else{ card2 Comment 3 } } } Comment 4, Comment 5

Comments for highestCard(card1 card2)
Comment 1 : A comma is missing between the card1 and card2 input parameters.
Comment 2 : card.name is undefined in the Card Class. Should read as per line below. return card1;
Comment 3 : This is doing nothing and is not valid Javascript syntax. It should be return card2;
Comment 4 : This is an extra }. There are 3 open { and there must be 3 closed - here there are 4.
Comment 5 : The chance of the same value for the cards being passed in must be explicitly catered for.

static.cardsTotal(cards){ return total; >> Comment 1 for(let card of cards){ total += card.value; Comment 2 return "You have a total of " + total; Comment 3 } } }

Comments for static.cardsTotal(cards)

This will only return undefined and leave. No other processing will be done.
Total is undefined - bad practice. card.value is undefined and processing will stop.
The return would be executed the first time and thus no total will be returned.
Extra Comments : Static won't wqrk as it's in a class defintion. There's also no module.exports

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

Paste Screenshot here

```
class Bus {

  constructor() {
    let passengers = 0;

    this.addPassengers = function(number_at_stop){
      passengers += number_at_stop;
    },

    this.getPassengers = function(){
      return passengers;
    }
  }
}

const newBus= new Bus();
newBus.addPassengers(20);
console.log('Initial Passengers : ' ,newBus.getPassengers()); // should be 20
newBus.addPassengers(15);
console.log('After adding 15 to Initial Passengers : ' ,newBus.getPassengers());
console.log('Direct addressing of passenger: ',newBus.passengers);
```

When running

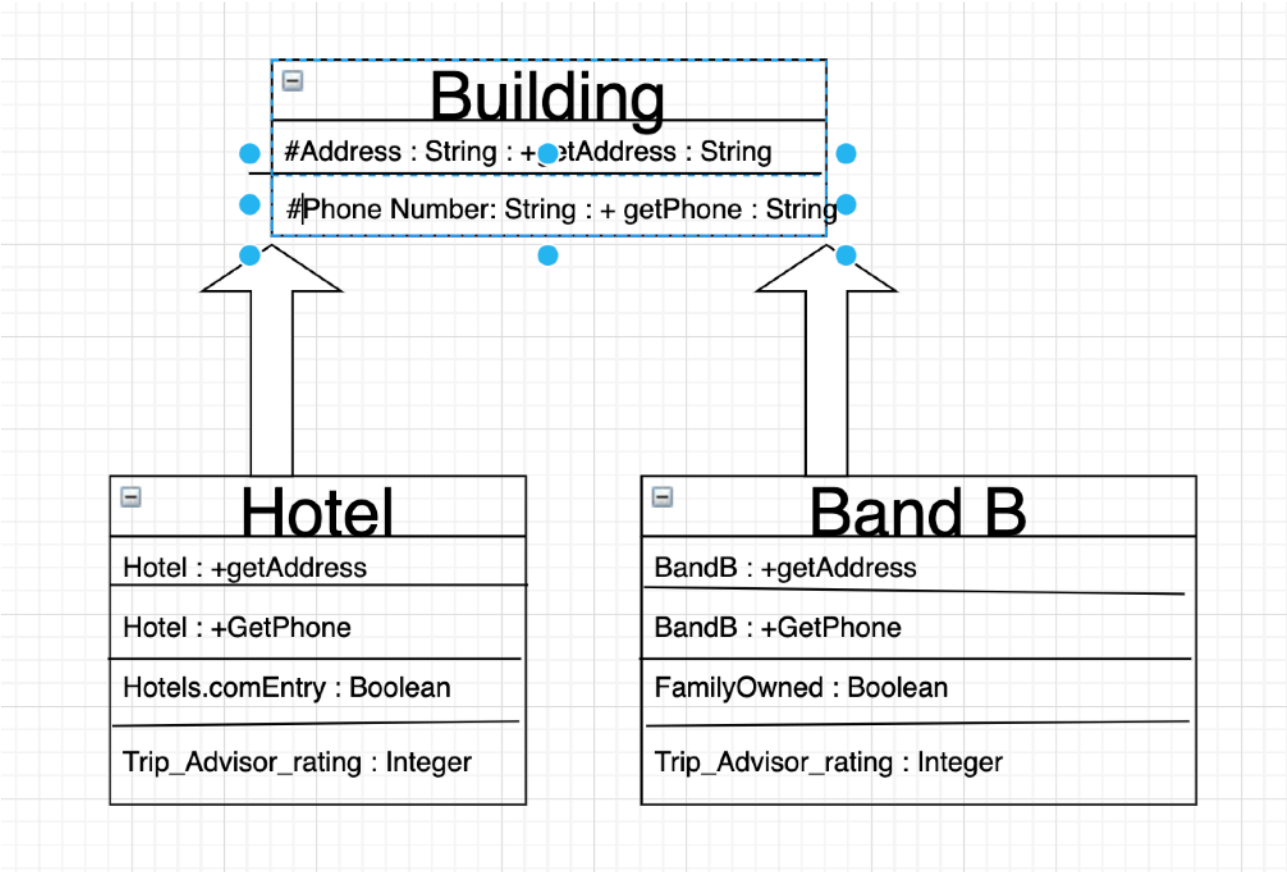
```
➜  encapsulate git:(master) ✗ node encaps.js
Initial Passengers :  20
After adding 15 to Initial Passengers :  35
Direct addressing of passenger:  undefined
➜  encapsulate git:(master) ✗
```

Description here

The above example is showing encapsulation in Bus Class.
What it means it that the variable passengers is only able to be updated or retrieved via a function. This makes it completely private and internal to the class

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

Paste Screenshot here



Description here

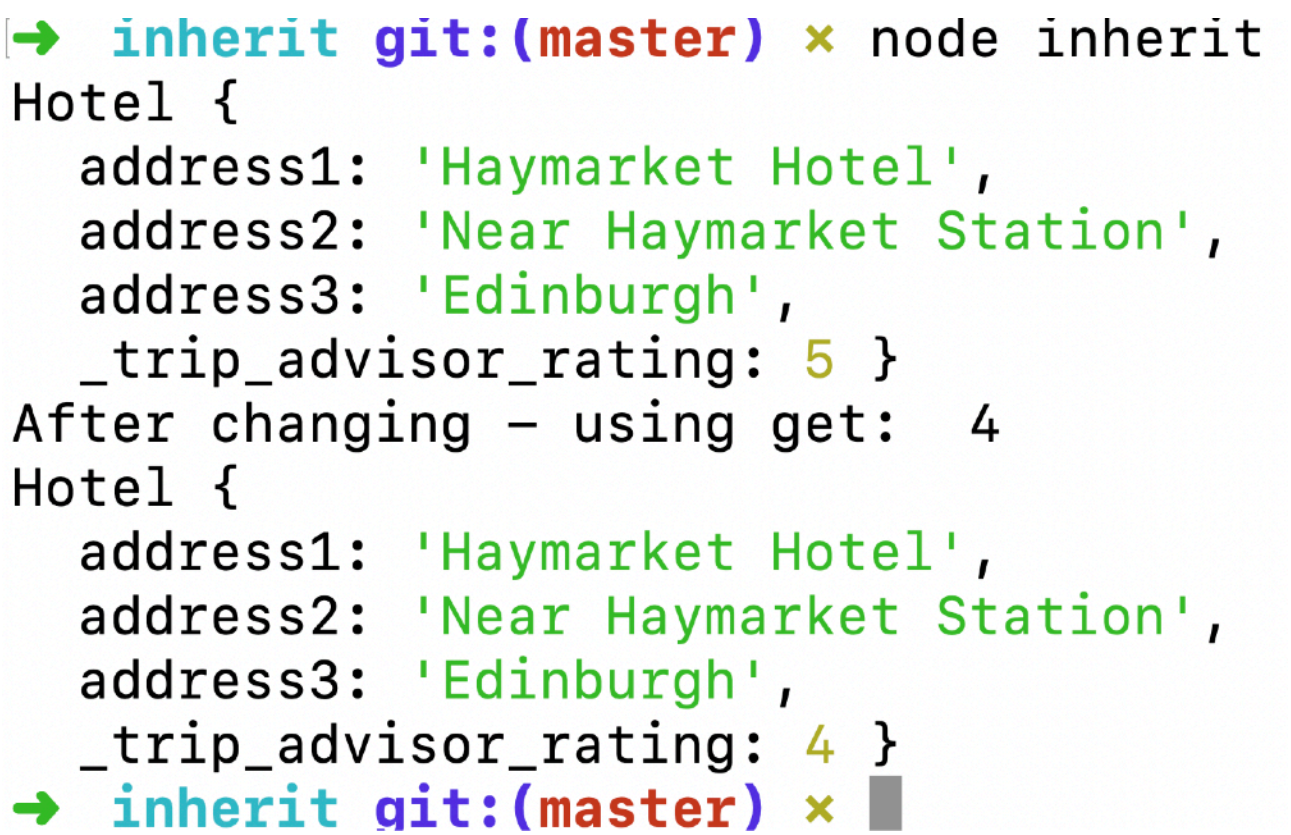
The above diagram shows that both the Hotel and BandB classes inherit the Address and Phone number attributes as well as a method to retrieve them. It's also important to note that the Hotel and BandB classes differ in the other attributes which are totally separate and only relate to the class in which it has been specified.

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.

Paste Screenshot here

untitled inherit.js

```
class Building {
  constructor(address1, address2, address3) {
    this.address1 = address1;
    this.address2 = address2;
    this.address3 = address3;
  }
}

class Hotel extends Building {
  constructor(address1, address2, address3, trip_advisor_rating) {
    super(address1, address2, address3);

    // trip advisor rating is specific to Hotel
    this.trip_advisor_rating = trip_advisor_rating;
  }

  get trip_advisor_rating() {
    return this._trip_advisor_rating;
  }

  set trip_advisor_rating(newTripadvisor) {
    this._trip_advisor_rating = newTripadvisor;
  }
}
let Haymarket_Hotel = new Hotel(
  "Haymarket Hotel",
  "Near Haymarket Station",
  "Edinburgh",
  5
);

console.log(Haymarket_Hotel);
Haymarket_Hotel.trip_advisor_rating = 4;
console.log("After changing - using get: ", Haymarket_Hotel.trip_advisor_rating);
console.log(Haymarket_Hotel);
```



Description here

The Hotel object extends the Building class by adding “Trip_advisor_rating”. That is also indicated by the underscore in front of the trip advisor rating output.

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.

Paste Screenshot here

1st Algorithm

```
Algorithm for displaying the initial front page view of the Photo Application.

Ensure that the map is displayed at 50% of the screen size.
Call Google maps API asynchronously using the private key.
Read all of the data in the locations SQL database table asynchronously and store the data
 in an array in JSON format.
When the data is ready indicated via a publish / subscribe cycle:
Read the entire array using an index
    Add name to the visible part of dropdown list.
    Add the index to the key part of the dropdown list
    Increment index by 1.

Indicate that the screen is ready for refresh

End of front Page initial view.
```

2nd Algorithm

```
calculate_average function  - parameter is an array
  initialise the running total to 0
  for every occurence in the array
      accumulate a running total
  The derived average is returned as the running total / length of the input parameter
```

Description here

1st Algorithm:

The initial view is the main user interface. The map size and it's constraints are crucial as there won't be enough room for the other components otherwise.
The data is held in a SQL database and calls to retrieve the data are asynchronous, meaning that the rest of the script will continue and not wait. This is why there is a publish / subscribe cycle to flag when it's ready.
For the dropdown box entries it's important that the index is also stored as it gives easy and efficient future access for display purposes.
The screen must use refreshed to take account of all of the above changes.

2nd Algorithm

The objective of the function is to produce an average for a given set of numbers in an array.
A total of all of the elements is accumulated.
The number of elements in the array is derived .
The returned value is determined by dividing the accumulated total by the number of elements

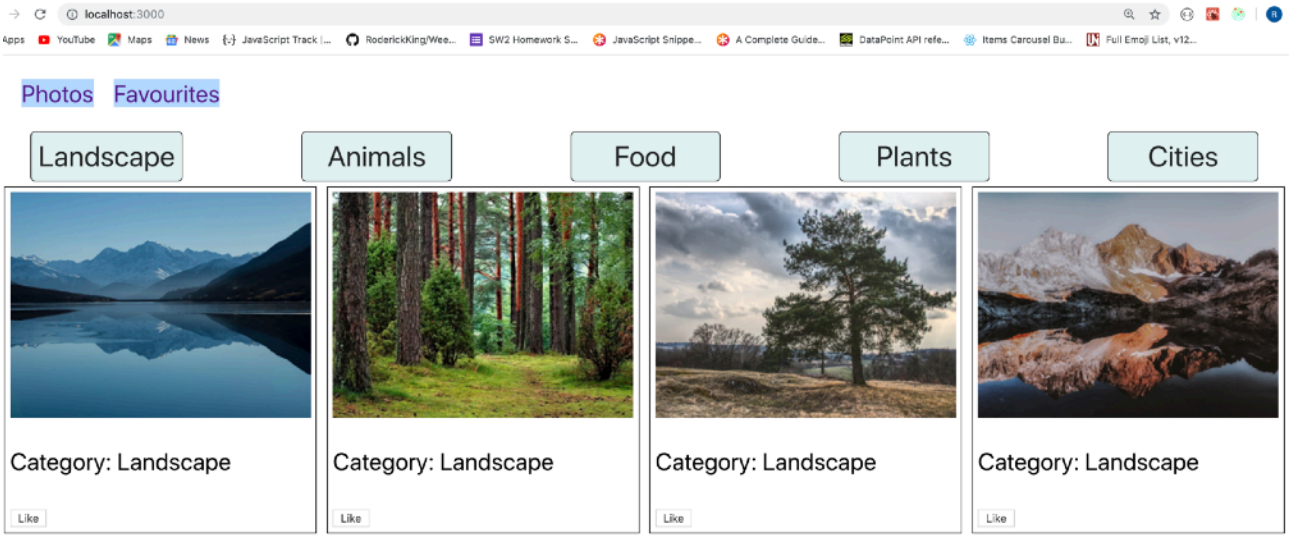
Week 4

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

Paste Screenshot here

```
const PhotosList = (props) => {

  const checkPhotos = () => {
    if (props.category === null) {
      return props.photos
    } else {
      return props.photos.filter(photo => {
        return photo.category === props.category;
      })
    }
  };
```

Result of running



Description here

When the “Landscape” button is clicked for example, the only photographs displayed are those which have a category of “Landscape”.
The function Photoslist applies a filter which searches over the entire list of photographs held in ‘props’ and returns only those which match the button pressed (in this case Landscape).
The filter is an in-built method in Javascript ES2017 and saves manually looping through an array.

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

Paste Screenshot here

```
➜  sort_method git:(master) ✗ node sort_method.js
Original List :  [ 'Zante',
  'Glasgow',
  'Delhi',
  'York',
  'Aberdeen',
  'Berlin',
  'Edinburgh' ]
Sorted List :  [ 'Aberdeen',
  'Berlin',
  'Delhi',
  'Edinburgh',
  'Glasgow',
  'York',
  'Zante' ]
➜  sort_method git:(master) ✗
```

Description here

The bubble sort method is shown above. Since the comparison is done with the nearest occurrence in the array, it needs to scanned again so that all possible comparisons can be done. A fresh copy of the input array was taken at the start of the routine to maintain clean code and to avoid unwanted side-effects.

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

Code Below

```
componentDidMount() {
  // This is invoked when tyhe page display initially
  // wait for asynchronous return  for the restcountries API before calling Countries Select.
  // This API has no requirement for a personal key and return a list of all countires and
  // associated data.
  // call Countires select to populate drop down list
  // Note that this is using Reacr Javascript
  fetch("https://restcountries.eu/rest/v2/all")
    .then(res => res.json())
    .then(data => {
      this.setState({ countriesData: data });
    });
}
render() {
  return (
    <>
      <h1>Countries of the World</h1>
      <CountriesSelect
        countriesData={this.state.countriesData}
        handleSelectedCountry={this.handleSelectedCountry}
      />
      {this.createCountry()}
    </>
  );
}
```

```
createCountry() {
  // call the display routine  Country using the selected item with subscripted data
  if (this.state.selectedIndex) {
    return <Country country={this.state.countriesData[this.state.selectedIndex]} />
  } else {
    return <h4>Select a country</h4>
  }
}
```

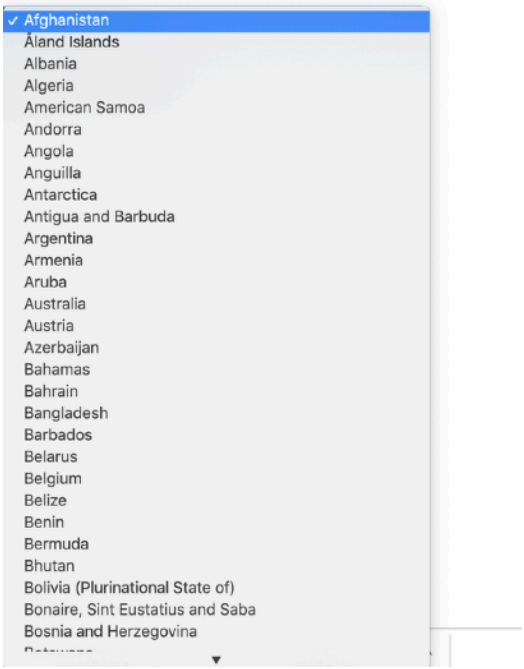
```
import React from 'react';

const Country = ({country}) => {

  return (
    <div>
      <h2>{country.name}</h2>
      <h3>Region: {country.region}</h3>
      <h3>Population: {country.population}</h3>
      <img src ={country.flag} width={200} height={300} />
    </div>
  )

};

export default Country;
```

Running

- Home
- About

Home

Countries of the World



- Home
- About

Home

Countries of the World

Australia

Australia

Region: Oceania

Population: 24117360

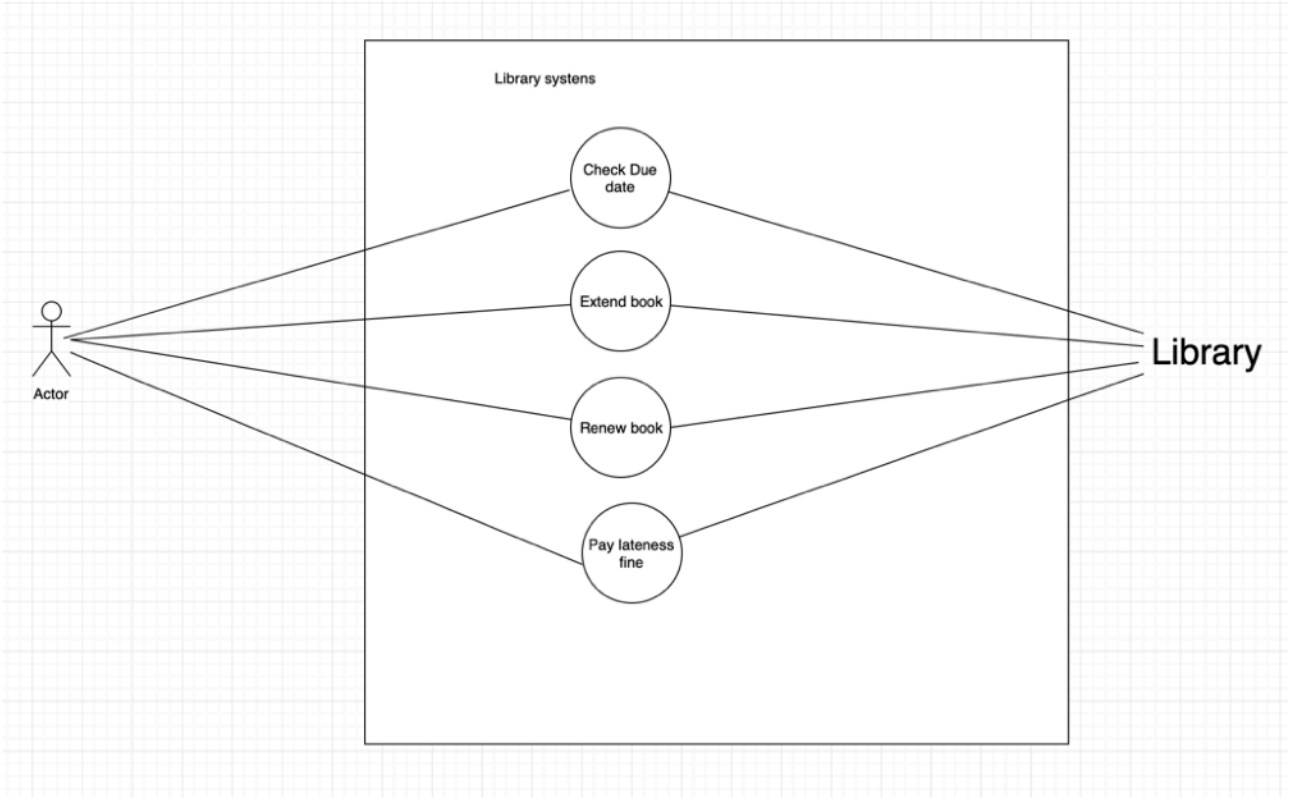
Description here

The API “https://restcountries.eu/rest/v2/all" returns a list of countries and it does not require a special API key. The data is returned in a standard JSON format. It’s a commonly used API.

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

Paste Screenshot here

Description here

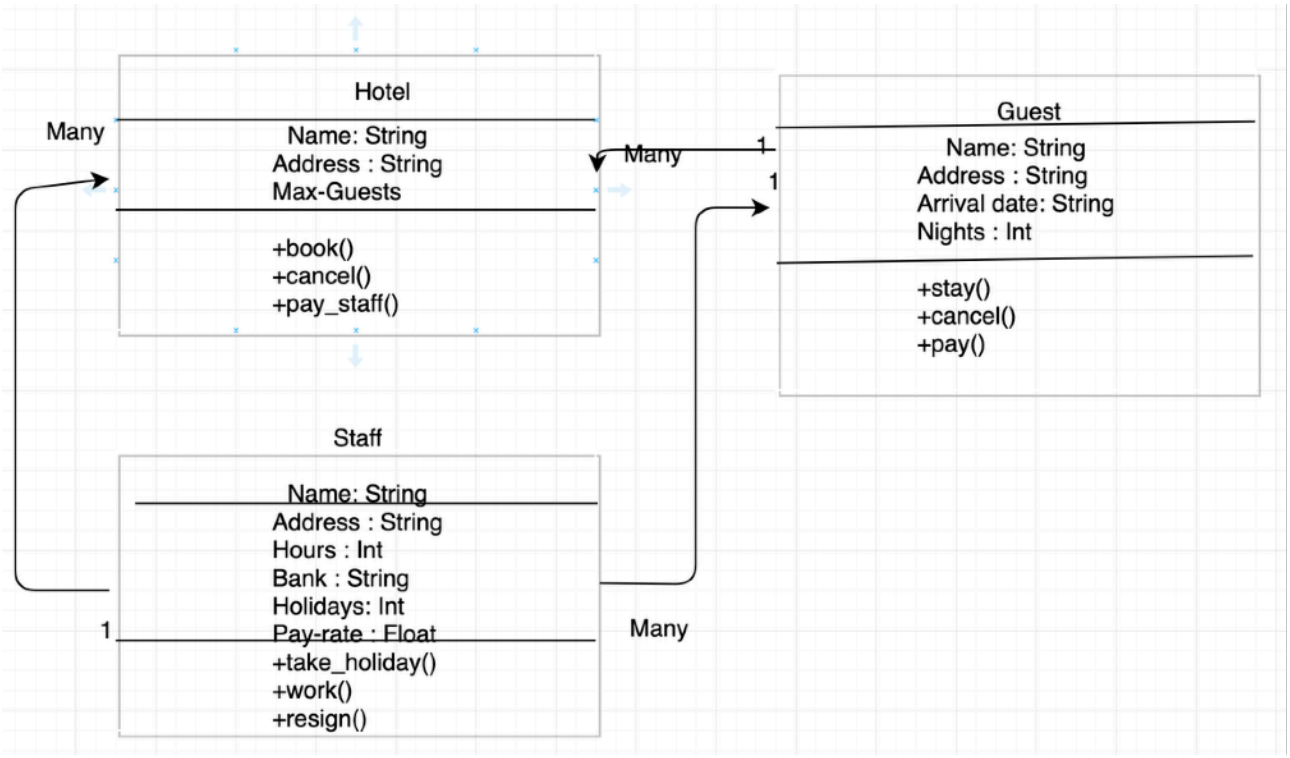


Description here

The above use diagrams shows that there are 4 actions which can be carried out by the Actor.
These are Checking Due date, Extend a book, Renew Book and Pay Lateness fine.
The Library performs the requests.

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

Paste Screenshot here

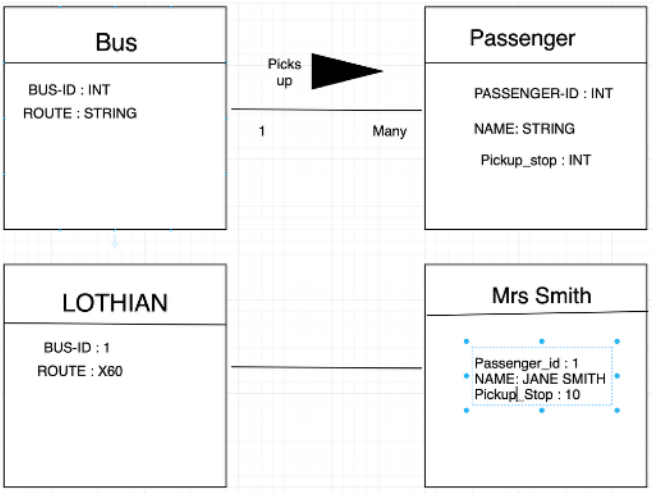


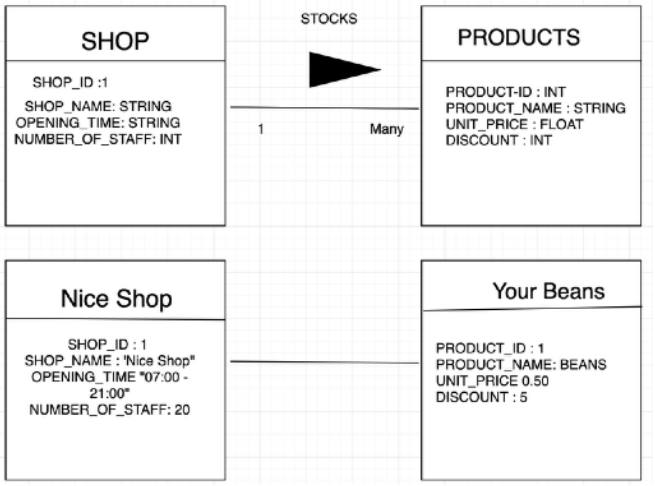
Description here

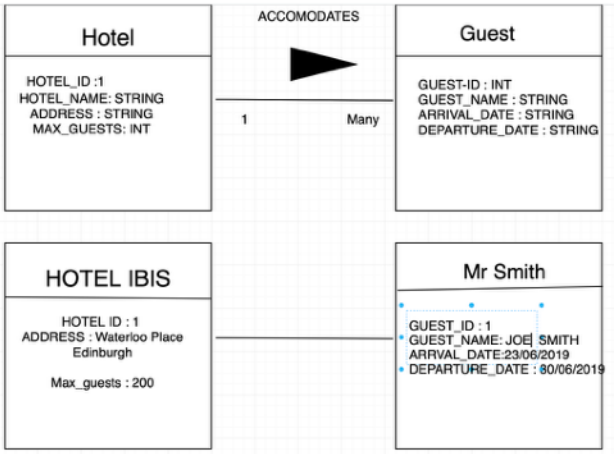
The above class diagram show the relationship between Hotel , Guests and Staff.
Hotel can have many guests and many staff.
Likewise there are many staff to each guest.
The methods (indicted by + on the front) show methods which can be used by each class.
For example, the hotel can pay staff, cancel / book.
Guests can stay, as well as cancel / pay.
The relationship may look circular, but that's exactly what it s.

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

Paste Screenshot here







Description here:

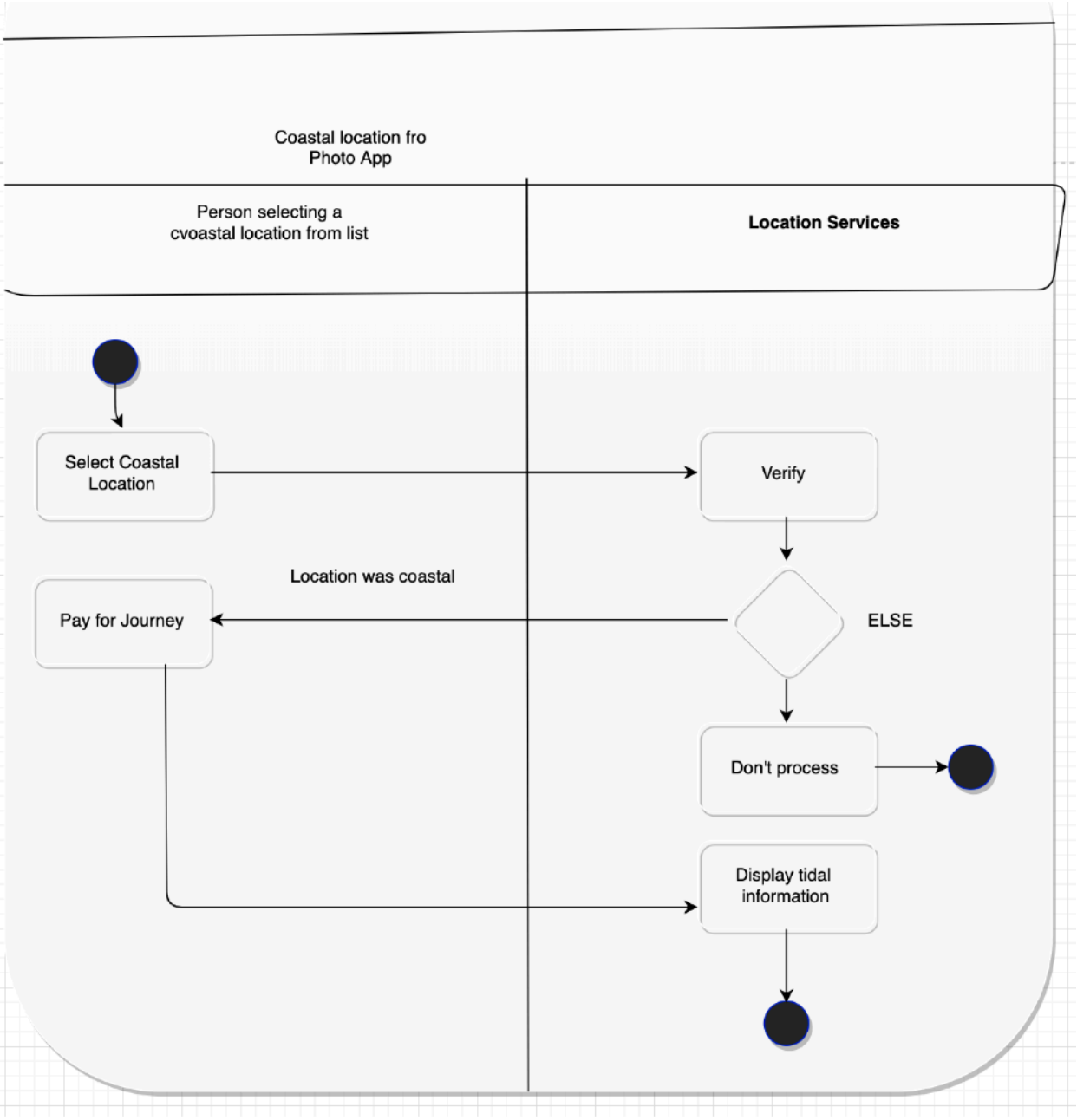
For the Bus :
Each Bus has many passengers. The attributes of the Bus are : bus_id and the route. Each passenger has an ID, a name and a pickup stop.

For the Hotel :
Each Hotel has many guests. The attributes of the Hotel are : Hotel_id, name, address and max_guests. Each Guest has an id, name, arrival and departure date.

For the Shop :
Each Shop sells many products The attributes of the Shop are :Shop_id, name, opening time and number of staff. Each individual product has an id, name, price and a discount.

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

Paste Screenshot here



Description here

The above Activity diagram shows the process of a finding out tidal information in the Photo App.
If it's not leave the process.
Otherwise the tide times can be derived by using a standard API / web site scrape..

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

Paste Screenshot here

Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	The Photo App means to be able to run on a wide variety of devices as are possible. Access to the internet is required to get the MAP data. A 6 inch screen is the minimum size of view and navigate the map.	By building a web application, a large number hardware / software platforms can use the Photo App. There's no work around if the Internet is not available
Performance Requirements	The photo App should be a response application and have no delays on startup. The map must also be displayed quickly and it's interactively	Use Google Maps API asynchronously for maximum performance
Persistent Storage and Transactions	Need to ensure that enough storage is available and that database activities are fast.	Use a well proven industrial strength SQL database such as POSTGRESS
Usability	The Photo App needs to fully comply with DDA. The map controls / interface must not confuse people.	Only use standard control and objects such as Google Maps / dropdown boxes so that the users have very little issue with familiarity
Budgets	There's only enough budget for 5 person days in total..	Come up with a realistic Minimum Value Product to ensure the timelines are met. Also ensure that a skilled developer is used to minimise the risk of overrun.
Time Limitations	Need to meet the publicity announced presentation date for the app.	Concentrate only on the scope of the Minimum Viable Product ensuring it has a realistic chance of being met. Review and re-scope MVP if the delivery fall behind schedule. Also ensure that a skilled developer is used to minimise the risk of overrun.

Description here

For Hardware and Software Platforms:
It's about not restricting our product to a small subset of people so constructing versions for Google Play, Apple's App Store and a web based version will maximise the poteb till user base.

Performance Requirements:
It's imperative that there are no adverse comments about the speed of the App. The map refreshes / interaction should not lag behind.

Persistent Storage and transactions:
This is about ensuring the user experience is a good one with fast storing and retrieving data. POSTGRESS SQL database was chosen as it's a fast SQL database.

Usability:
It's important to use industry standard control so there's learning curve or negative comments about an unusual interface. Therefore for the Photo App Google Maps was the chosen interface as it's widely used and a defacto standard.

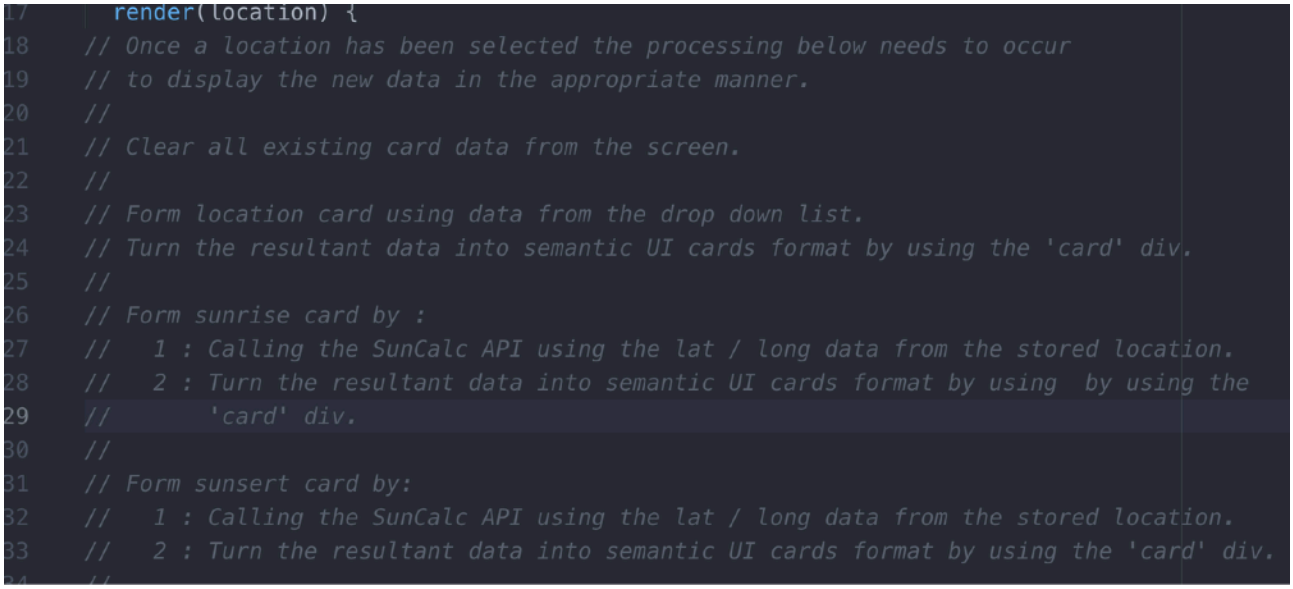
Budgets:
There's only enough budget for 5 person efforts effort in total. It's important therefore to concentrate on the contents of the MVP to ensure it's do-able. Also a skilled developer is almost prerequisite for this short time frame.

Tine Limitations:
There'a only 5 days to develop the Photo App in readiness for the presentation. It's important therefore to concentrate on the contents of the MVP to ensure it's do-able. Also a skilled developer is almost prerequisite for this short time frame.

Week 5

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

Paste Screenshot here



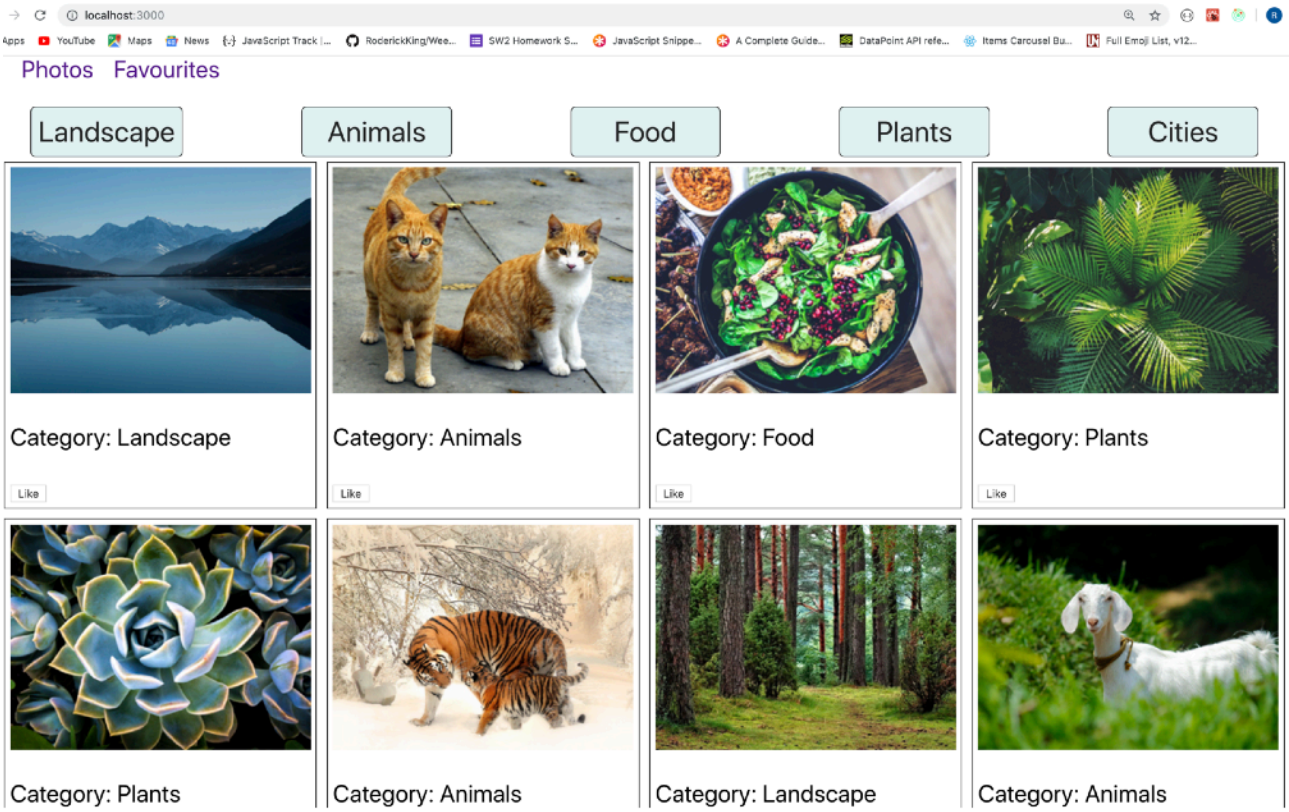
```
    render(location) {
// Once a location has been selected the processing below needs to occur
// to display the new data in the appropriate manner.
//
// Clear all existing card data from the screen.
//
// Form location card using data from the drop down list.
// Turn the resultant data into semantic UI cards format by using the 'card' div.
//
// Form sunrise card by :
//   1 : Calling the SunCalc API using the lat / long data from the stored location.
//   2 : Turn the resultant data into semantic UI cards format by using  by using the
//       'card' div.
//
// Form sunsert card by:
//   1 : Calling the SunCalc API using the lat / long data from the stored location.
//   2 : Turn the resultant data into semantic UI cards format by using the 'card' div.
//
```

Description here

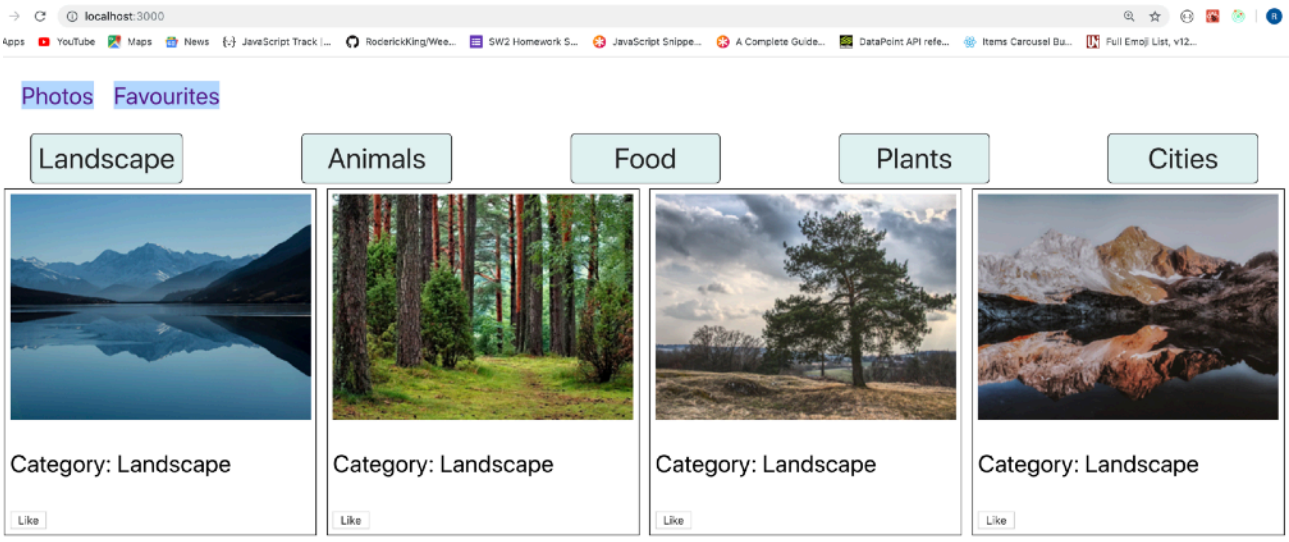
The above screenshot shows how to generate the Locations , Sunrise and Sunset cards.
This gives a high level breakdown of both the processing and order in which it is required.

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

Paste Screenshot here



When the Landscape button is clicked only Landscape photos will be shown as below



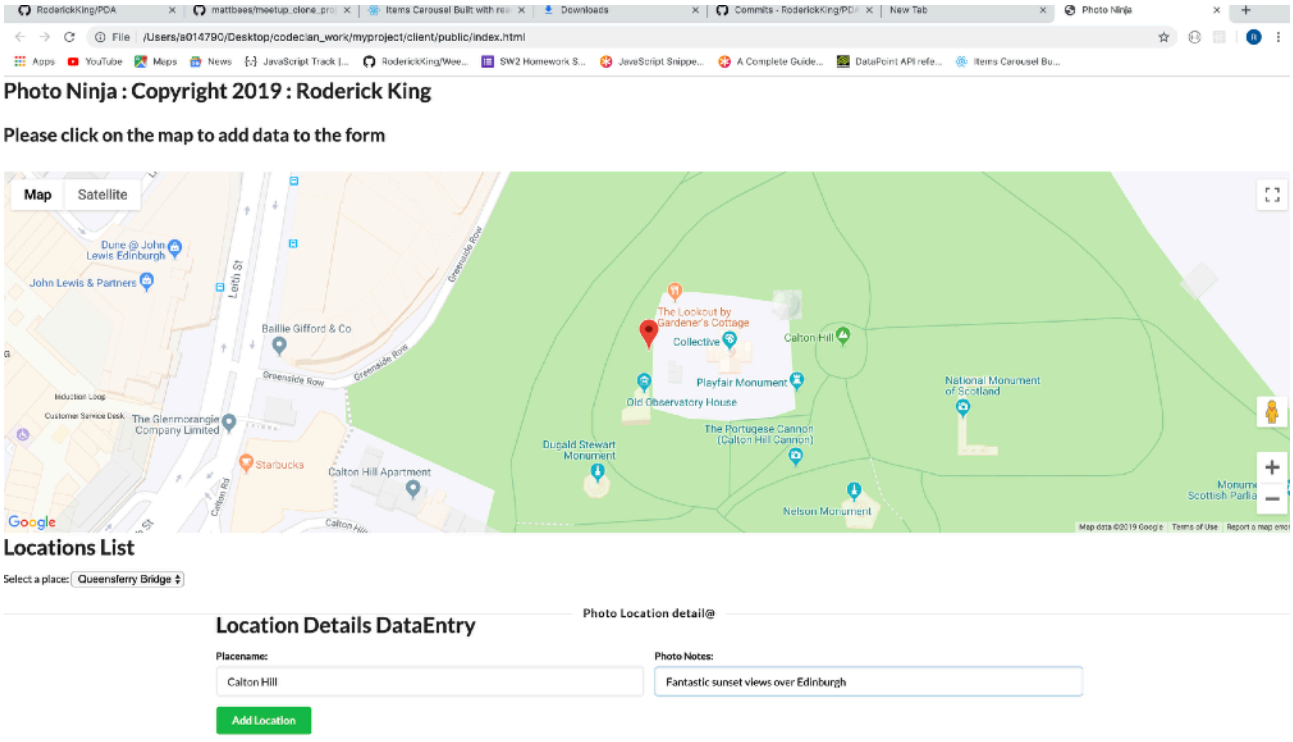
Description here

A Javascript Filter method is applied to the photograph's data using the code below.
This loops through all of the available photographs and only selects the matching categories.

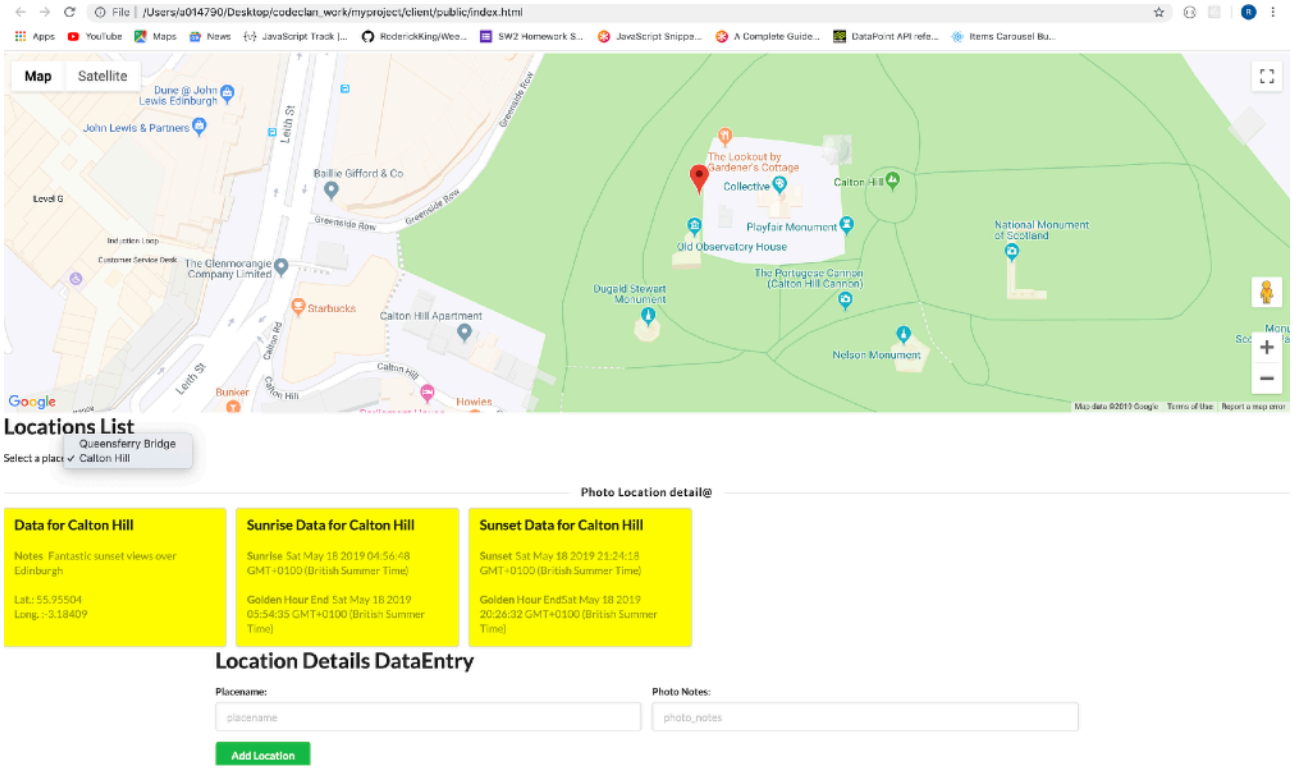
```
const PhotosList = (props) => {

  const checkPhotos = () => {
    if (props.category === null) {
      return props.photos
    } else {
      return props.photos.filter(photo => {
        return photo.category === props.category;
      })
    }
  };
```

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

Paste Screenshot here



Showing the retention of the location

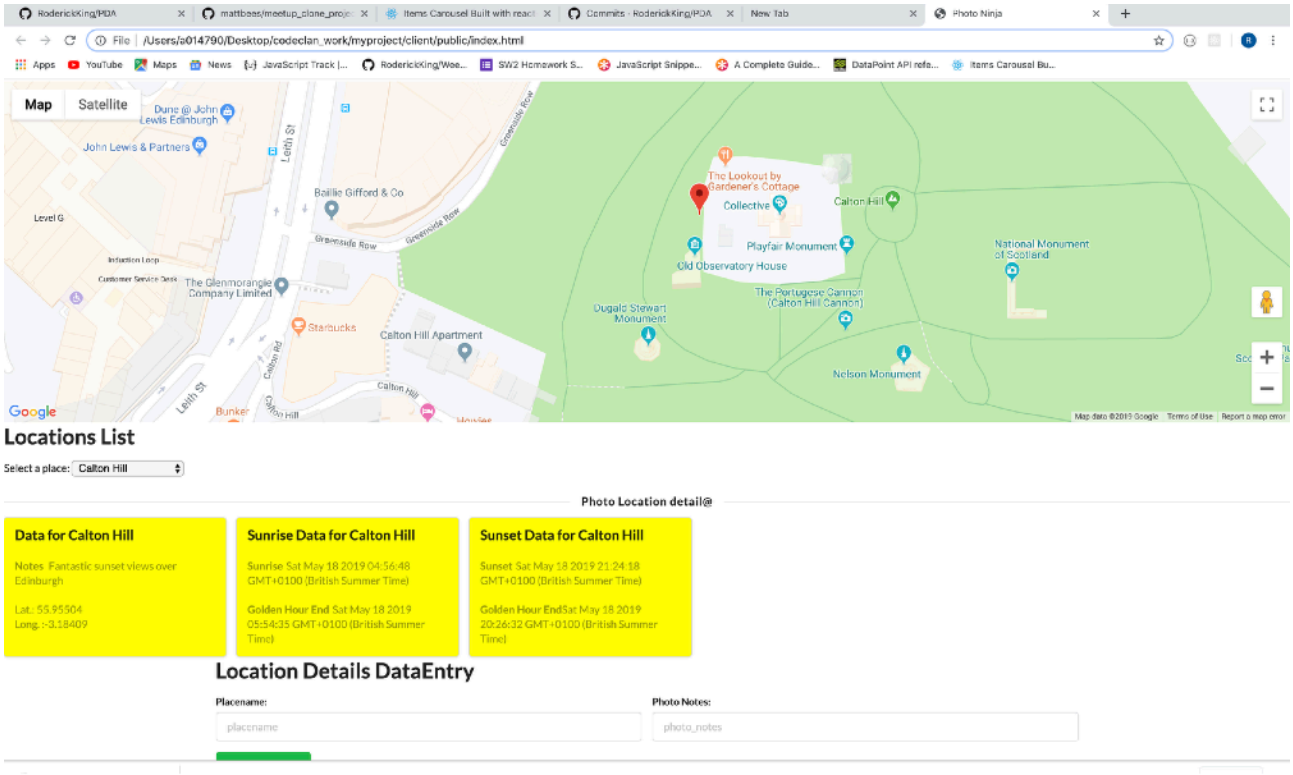


Description here

The dropdown box by the left side of the screen is a list of locations which are added to when the ‘pin’ is added onto the Google Map. They are dynamically added to an SQL database for persistence and this list is refreshed upon data entry as demonstrated.

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

Paste Screenshot here

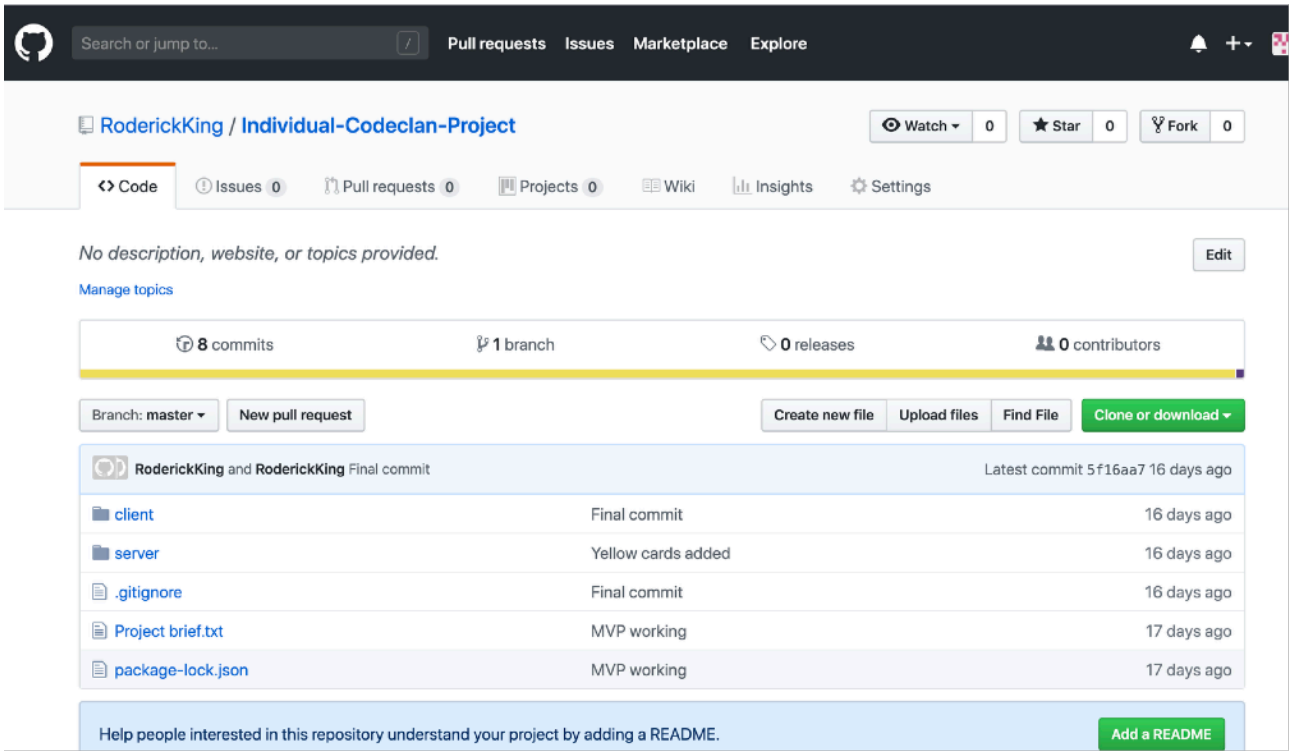


Description here

Once the location (Calton Hill) has been selected from the dropdown box extra items of information about it are obtained from the SunCalc API. The includes the saved latitude and longitude data as well as the original description are they are also displayed from the database.

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

Paste Screenshot here



Description here

Here is the hyperlink to the GIT repository. Detail of commit can be determined from this,

https://github.com/RoderickKing/Individual-Codeclan-Project

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

Paste Screenshot here

Original Brief.

```
Photographic Planning Aid

The objective of this is to help to make it easier to plan successful trips in a simple
intuitive manner.

As a user I want to be able to:

- Use a fully interactive map to select potential places to visit
- Click on a location to find out sunrise / sunset times
- Click on a coastal location to find out tide times
- Store the location details along with the above details
- Maintain the acquired information from above

MVP:

Create and maintain a list of locations and attributes using a database.
This will be achieved by utilising Google maps or Leaflet API to present locations in a
standard user friendly manner.
The existing list of places should be highlighted on the map in the usual manner.
For coastal locations, display the tide times.
Sunrise / Sunset times should be displayed - using the Suncalc api.
https://github.com/mourner/suncalc
// format sunrise time from the Date object
var sunriseStr = times.sunrise.getHours() + ':' + times.sunrise.getMinutes();

// get position of the sun (azimuth and altitude) at today's sunrise
var sunrisePos = SunCalc.getPosition(times.sunrise, 51.5, -0.1);

// get sunrise azimuth in degrees
var sunriseAzimuth = sunrisePos.azimuth * 180 / Math.PI;
SunCalc is also available as an NPM package:

$ npm install suncalc
var SunCalc = require('suncalc');
Reference
Sunlight times
SunCalc.getTimes(/*Date*/ date, /*Number*/ latitude, /*Number*/ longitude)
Returns an object with the following properties (each is a Date object):

Property  Description
sunrise sunrise (top edge of the sun appears on the horizon)
sunriseEnd  sunrise ends (bottom edge of the sun touches the horizon)
goldenHourEnd morning golden hour (soft light, best time for photography) ends
solarNoon solar noon (sun is in the highest position)
```

After MVP achieved a slight Re-Plan

```
Extensions after MVP completed..
==================================
After investigations, it was determined that the links and drawing parts
could not be achieved.
They all ended up either being mathematically challenging
( drawing lines for sunrise and sunset ) or
 very difficult public API's such as Traveline .
For another table with friends data / addresses  there really was not time to
achieve that satisfactorily.

================================
For locations, where applicable create extra table for friends / accomodation.
Draw lines showing the transit of the sun during the day.
Link into Late Rooms / Booking for arranging accomodation.
Link to Traveline to find Public transport to the location. API looks very complex.
```

Description here

The original project brief to display the Google map, and the other interfaces was quite cut down compared to my initial thoughts to make the MVP achievable in the very short timespan available. However, this was still a very challenging project for short project duration .

The extensions beyond MVP were tricky but could have been achieved if more time had been available to do the project.

In particular I think the database extension element could have been achieved without too many technical issues.

However, showing the line for where the sunrise / sunset was ended up being very mathematically challenging as the API did not explain the results very clearly. The results were in different units to that expected and a small amount of online revealed this could easily become a maths project in itself…

Week 7

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

Paste Screenshot here

Bug / Error	Solution	Date
Google Map not displaying	Change the HTML TO accommodate Google Maps asynchronous nature.	29th April 2019
Unable to utilise the standard Drop Pin in Google map.	The event handle for the map click needed to be turned into an arrow function.	29t April 2019
The newly added data was not added to the database	The Pub / Sub literal needed to be changed to Post the data.	30 April 2019
The newly entered data via the form is not displaying	When the 'post' request is issued to write the data, the data needs to be completely re-read.	30 April 2019
Data did not clear of the input form after the 'Add' button was pressed.	A form reset was introduced	1st May 2019

Description here

The above errors occurred chronologically during the development of my individual Code Clan project.

1st Error
The async tag was required to be introduced INDEX.HTML.

2nd Error:
The ability to 'drop' pins via clicking the mouse is a key requirement and standard behaviour of any Google Map application. A lot of research was required to get this resolved.

3rd Error:
This error was quick difficult to track, but with the Publish / Subscribe method, it soon became obvious that the literal was not correct.

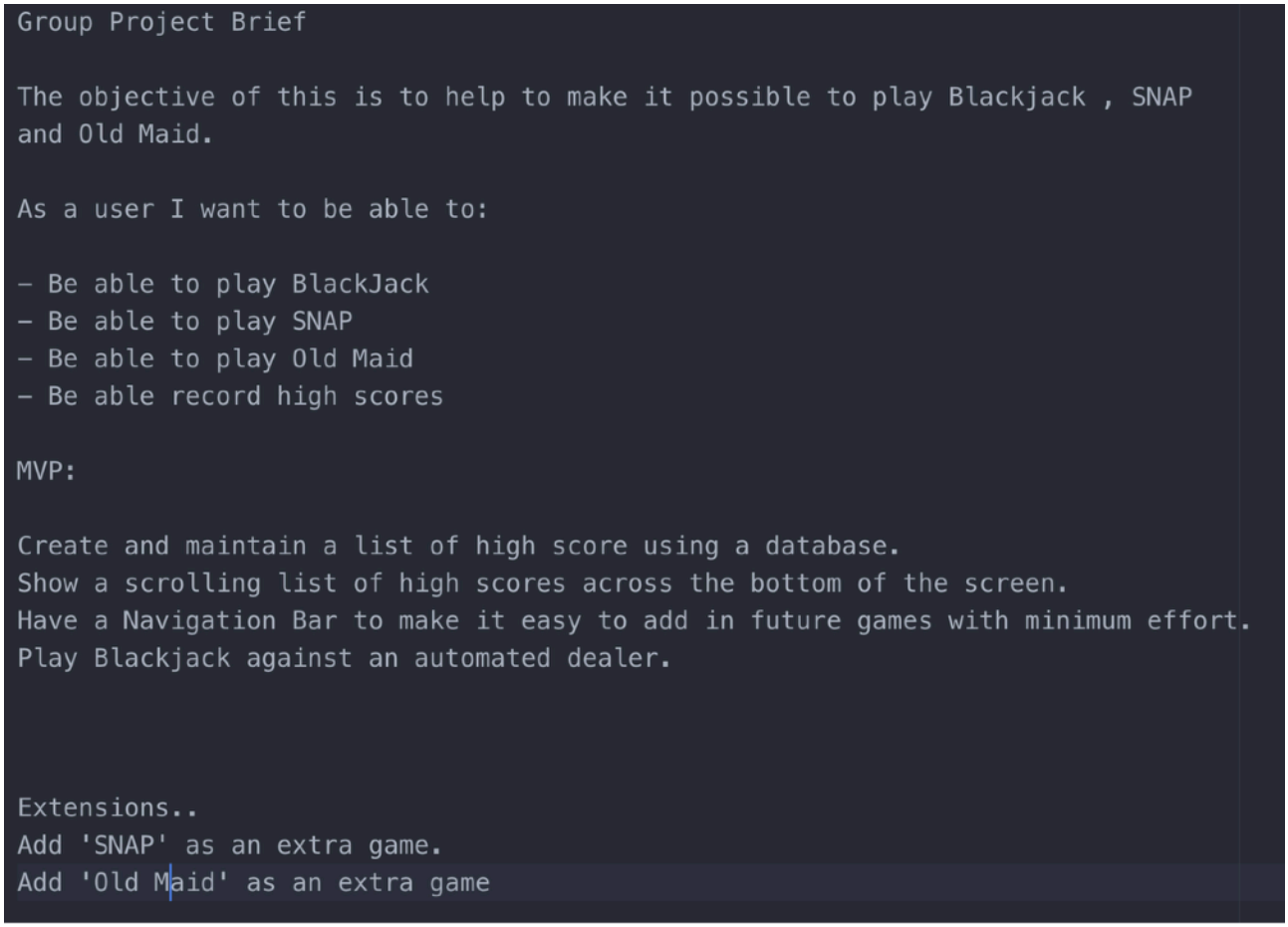
4th Error:
After it was proved that the data was added got the database via Postico it was easy to determine that an extra read of the data was always required after the write.

5th Error:
This was a comparatively easy error to track and to resolve.
It marked the last real error in the application.

Week 9

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

Paste Screenshot here

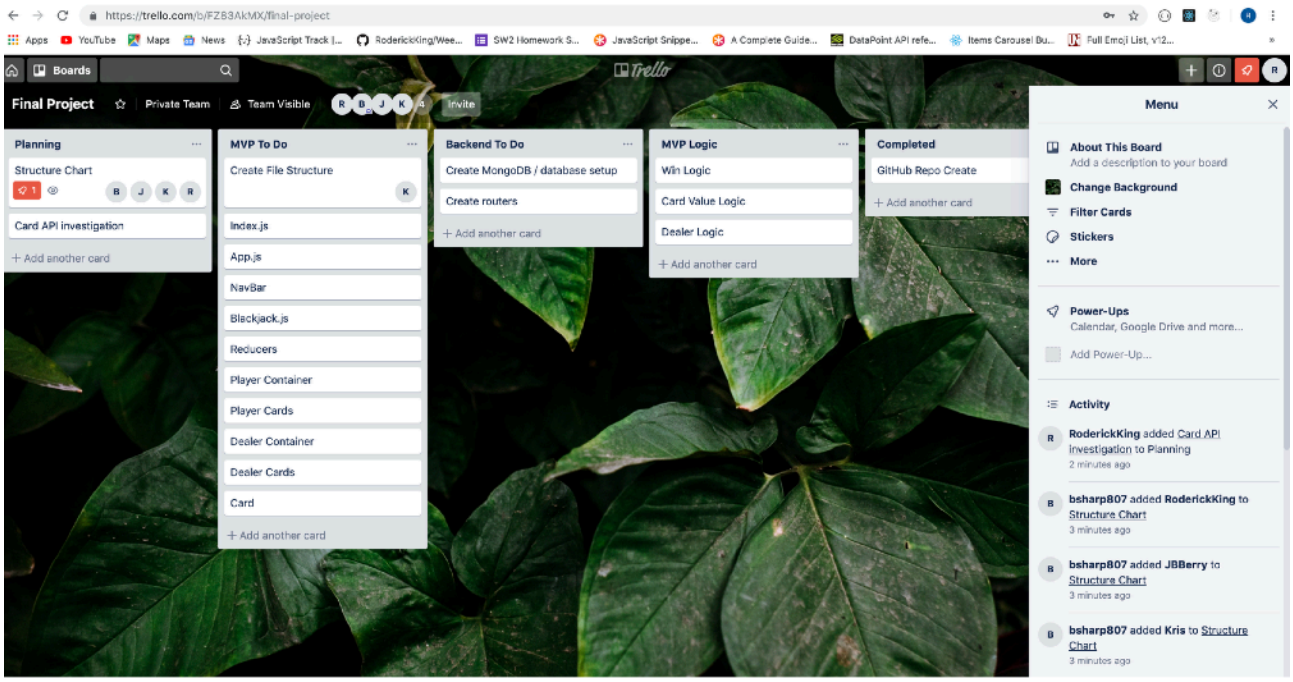


Description here

The objective of the group brief is to highlight the features required.
It's also important to realise what the Minimum Viable Product (MVP) is given the time constraints.
This is the smallest acceptable delivery to the customer which makes it worthwhile and must be achievable within the time scales requested.
The MVP also incorporates elements which make it easy to build upon , should time allow.
The extra games were added incase we managed to achieve MVP early and are to a large extent aspirational and challenging given the timescales,

Unit	Ref	Evidence
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.

Paste Screenshot here



Description here

The above show the initial planning stage for the group project. It helped to show that no features and major components would be missed from the project and that the work could be sensibly allocated. By adding the Navigation bar element at this early stage it would ensure it was easy to add in extra games with the least amount of extra effort.

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

Paste Screenshot here

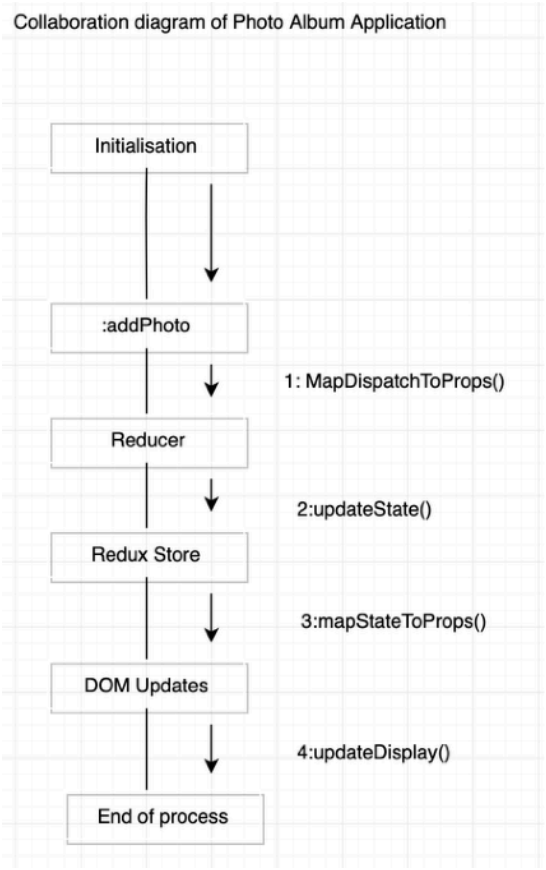
Acceptance Criteria	Expected Result	Pass/Fail
A user is able to see all of the photographs in the gallery	The photographs can be seen	Pass
A user should see the'like' buton should be displayed for each photograph	The like' button can be seen when the screen has loaded	Pass
A user is able to update the Like button.	The like button is clickable after the photographs have displayed	Pass
A user is able to see the Animals tag	Animals tag visible	Pass
A user is able to see the Landscape tag	Landscape tag visible	Pass
A user is able to see the Search bar button for the 2 categories.	The 2 search bar buttons are visible when the photographs have displayed	Pass
A user is able to see only Landscape photographs.	The search for Landscape returns Only landscape photographs when clicked	Pass
A user is able to see only Animal photographs.	The search for Animals returns only Animal photograph when clicked.	Pass

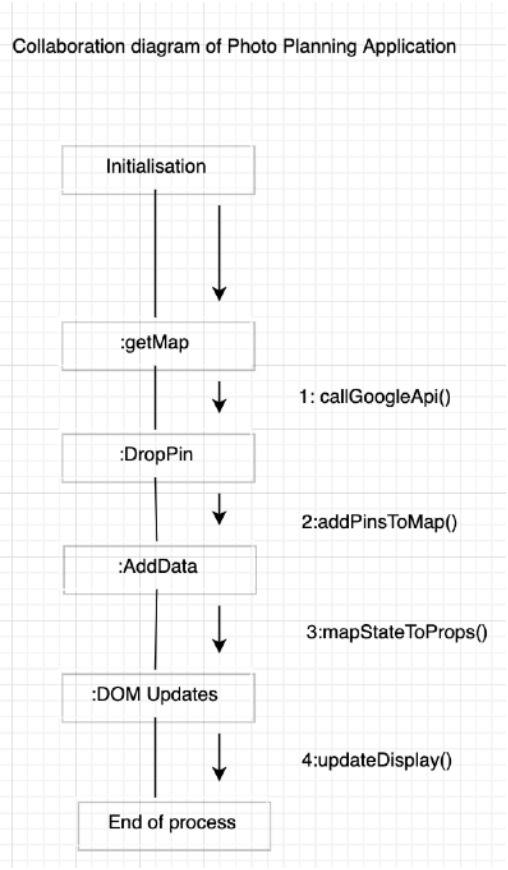
Description here

The application which is being tested shows photographs, with a 'like' button , 2 category descriptions and the ability to use a search bar to display only those photographs with those categories..
The acceptance criteria above test all of those features from a high level exactly as a user would do.

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

Paste Screenshot here





Description here

For the Photo Album Application :

The data is read and placed into the Redux Store via Props , re-read and finally re-displayed. This is a standard well tried and tested methodology when using React / Redux with most applications mapping out this way.

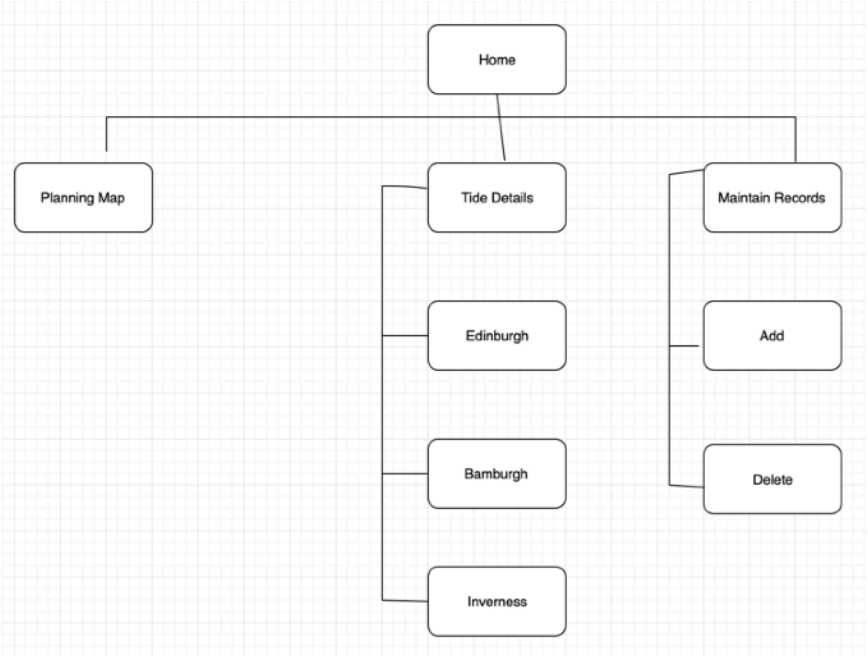
For the Photo Planning application:

The Google map's asynchronous map routines are called and that enables the ability to click on the map in a standard manner.

Week 10

Unit	Ref	Evidence
P	P.5	User Site Map

Paste Screenshot here

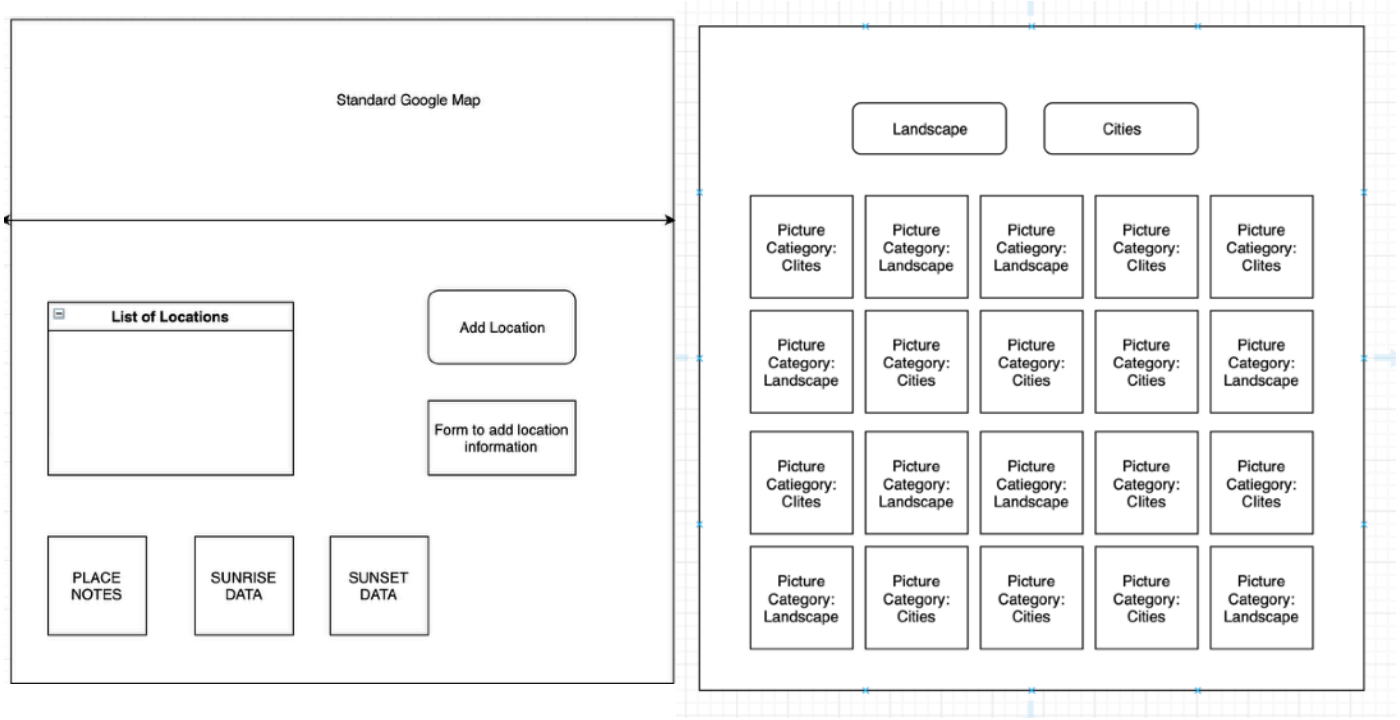


Description here

The user site map show that from the Home Page there are 3 sub menu items. These are Planning Map , Tide Details and Maintain Record. Under both the Tide details and maintain records menu items, there are sub levels of Edinburgh, Bamburgh and Inverness.

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Paste Screenshot here



Description here

Wireframe 1 (Left hand side)

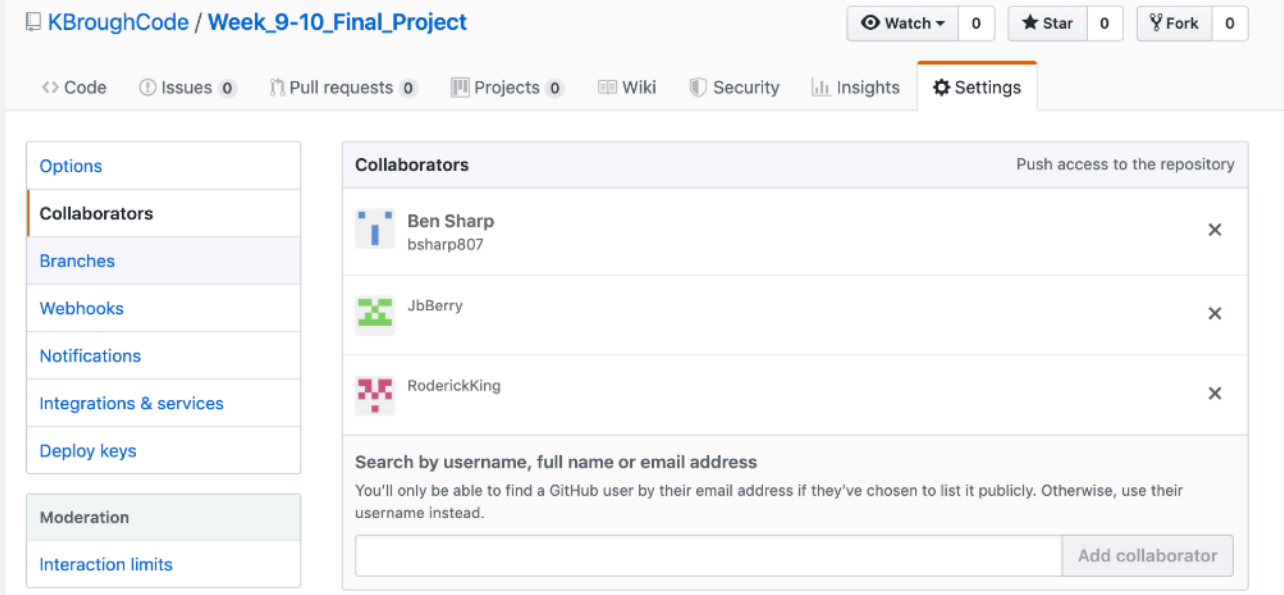
The layout shows that the user-user-interface with the standard Google map facility occupying the top 1/3 of the screen. The add location button is used in conjunction with the add location button / form.
The newly created diagram are added to the list of locations.
The Place Notes / Sunrise. / Sunrise cards are displayed upon drop down list on the List of Locations.

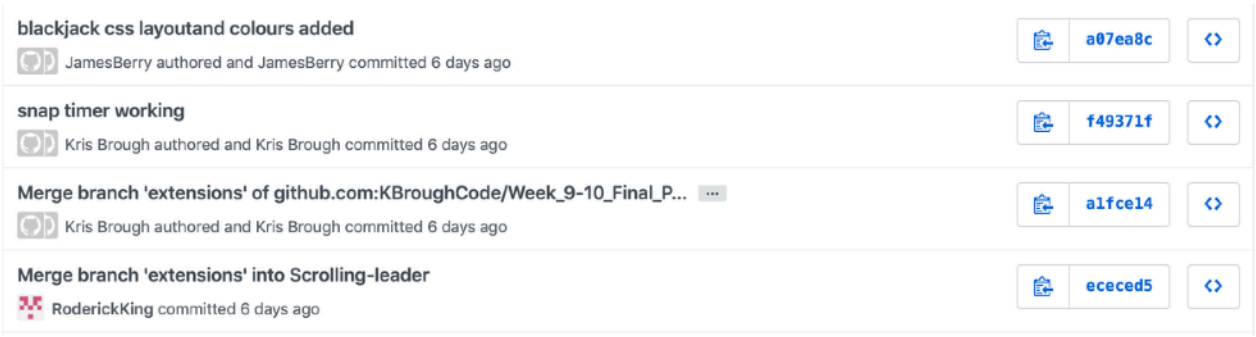
Wireframe 2 (Right hand side)

The layout show the initial view of loading the photo gallery application.
The pictures along with the category tags are displayed in a grid like structure.
When the ‘Landscape’ or ‘ Cities’ search bar is clicked , only the appropriate photographs with that tag are displayed.

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.

Paste Screenshot here



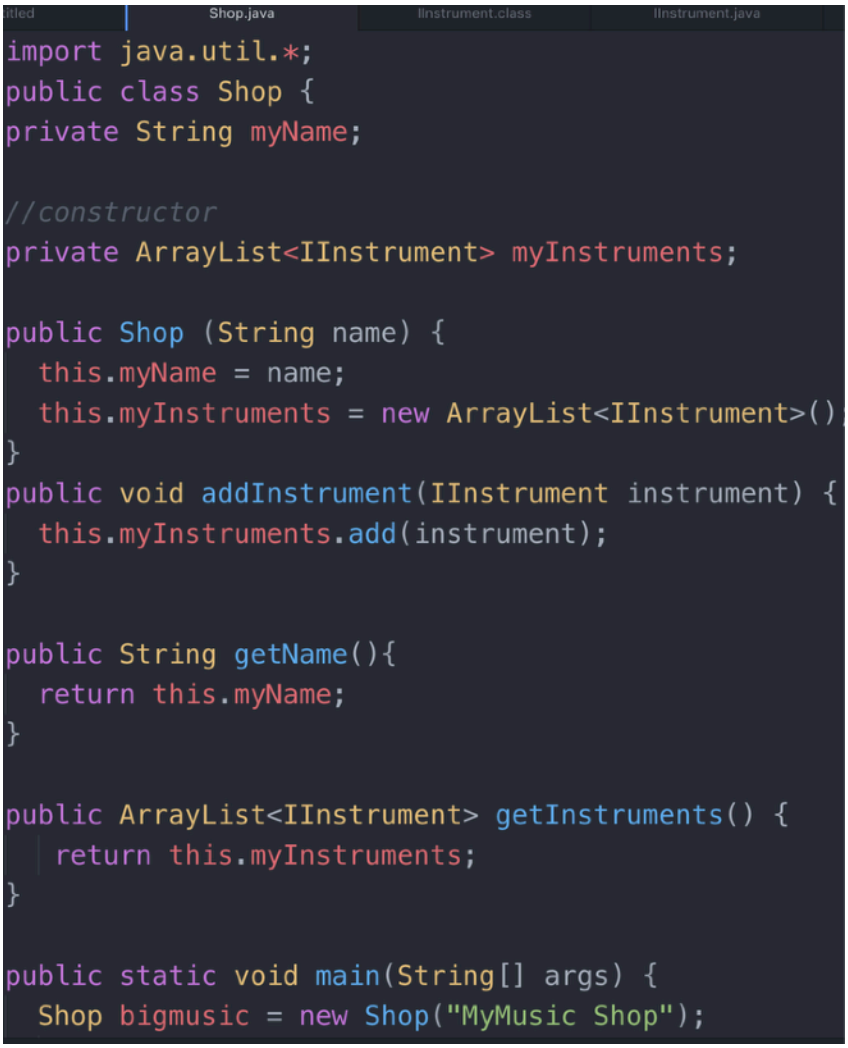


Description here

The first screen shot shows all of the team who made the team project.
The last screen shot shows that I have been working with multiple people successfully using GitHub along with Commit keys generated.

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

Paste Screenshot here

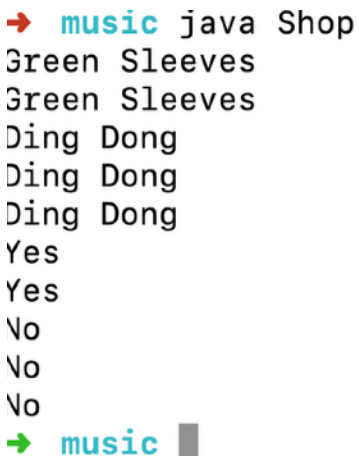


```
    Harpsicord variety1= new Harpsicord(200,"Smiths");
    Harpsicord variety2= new Harpsicord(100,"Yamaha");
    Bells tubular = new Bells(33);
    Bells notgreat = new Bells(3);
    Bells saintgiles = new Bells(899);

    bigmusic.addInstrument(variety1);
    bigmusic.addInstrument(variety2);
    bigmusic.addInstrument(tubular);
    bigmusic.addInstrument(notgreat);
    bigmusic.addInstrument(saintgiles);

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

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

```
public class Bells implements IInstrument {


  private int price;

  public Bells (int price) {
    this.price = price;
  }

  public String makeNoise(){
    return ("Ding Dong");
  }

  public String isAntique(){
    return ("No");
  }

}
```

```
public class Harpsicord implements IInstrument {

  private int price;
  private String maker;

public Harpsicord (int price , String maker){
  this.price = price;
  this.maker = maker;
}

public String makeNoise(){
  return ("Green Sleeves");
}

public String isAntique(){
  return ("Yes");
}

}
```

The output of running the shop is :



Description here

There are 2 classes of instrument , Harpsicord and Bells.
They utilise the Interface IInstrument to grab common methods, Makenoise and isAntique.

The instrument objects are polymorphic (e.g. being both a IInstrument and an Instrument at the same time).