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.

Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

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

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	
		I created a program simulating a karaoke bar. the program uses an array to store guests into a class. I used the array to check if the guests favourite song matches a song in the rooms playlist created. As well as charging the party(array of guests) their total amount for the room.	

Week 2

— ~/codeclan_work/week_2/day_5/caraoke

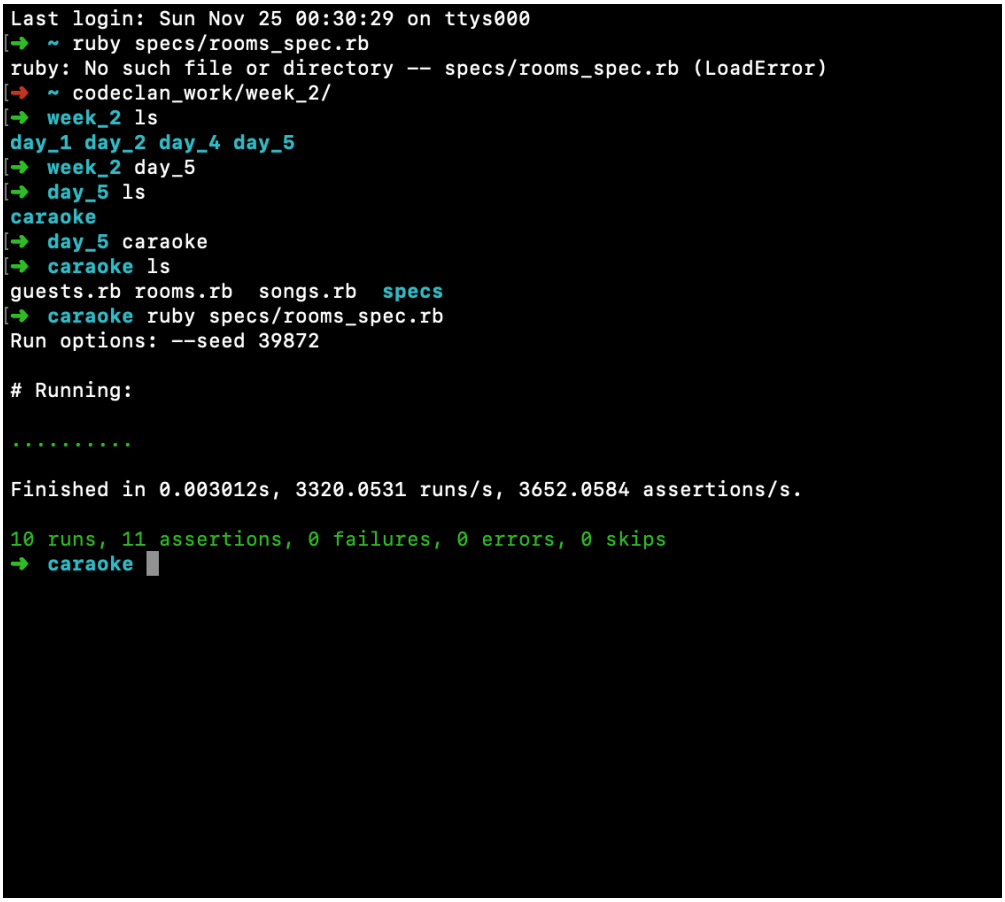
guests_spec.rb | rooms_spec.rb | song_spec.rb

```ruby
require('minitest/autorun')
require('minitest/rg')
require_relative('../rooms.rb')
require_relative('../songs.rb')
require_relative('../guests.rb')

class RoomsTest < MiniTest::Test

  def setup()
    @guest1 = Guests.new("John", 50, "Numb")
    @guest2 = Guests.new("Marie", 20, "Sandstorm")
    @guest3 = Guests.new("Paul", 40, "Tik Tok")
    @guest4 = Guests.new("Megan", 70, "Blow shit up")
    @guest5 = Guests.new("Mike", 10, "Gimme a Call")
    @guest6 = Guests.new("Nancy", 20, "Not a problem")
    @guest7 = Guests.new("Connor", 45, "Encore")
    @guest8 = Guests.new("Caitlin", 60, "Ambient 24hr rain")
    @guest9 = Guests.new("Peter", 60, "Forrest Sounds for Sleeping")
    @guest10 = Guests.new("Rebecca", 20, "Beastie Boys")

    @party1 = [@guest1, @guest2, @guest3]
    @party2 = [@guest4, @guest5, @guest6]
    @party3 = [@guest7, @guest8]
    @party4 = [@guest9, @guest10]

    @room1 = Rooms.new(2, ["Numb", "Jelly", "We are Human"], 25)
    @room2 = Rooms.new(6, ["Forrest Sounds for Sleeping", "Ambient 24
    rain"], 120)
    @room3 = Rooms.new(4, ["Not a Problem", "Encore"], 80)
    @room4 = Rooms.new(1, ["Tik Tok", "Gimme a Call"], 80)
```

rooms_spec.rb

guests.rb | rooms.rb | songs.rb

```ruby
    else
      return false
    end
  end

  def charge_for_room(party, room)
    total = 0
    for guest in party
      total += guest.money
    end
    total -= room.entry_fee
  end

  def guest_song(party, room)
    for guest in party
      guests_fav = guest.favourite_song
      for song in playlist
        rooms_song = song
        if guests_fav == rooms_song
        return "Whoo"
      end
      end
    end
    return false
  end


end

```

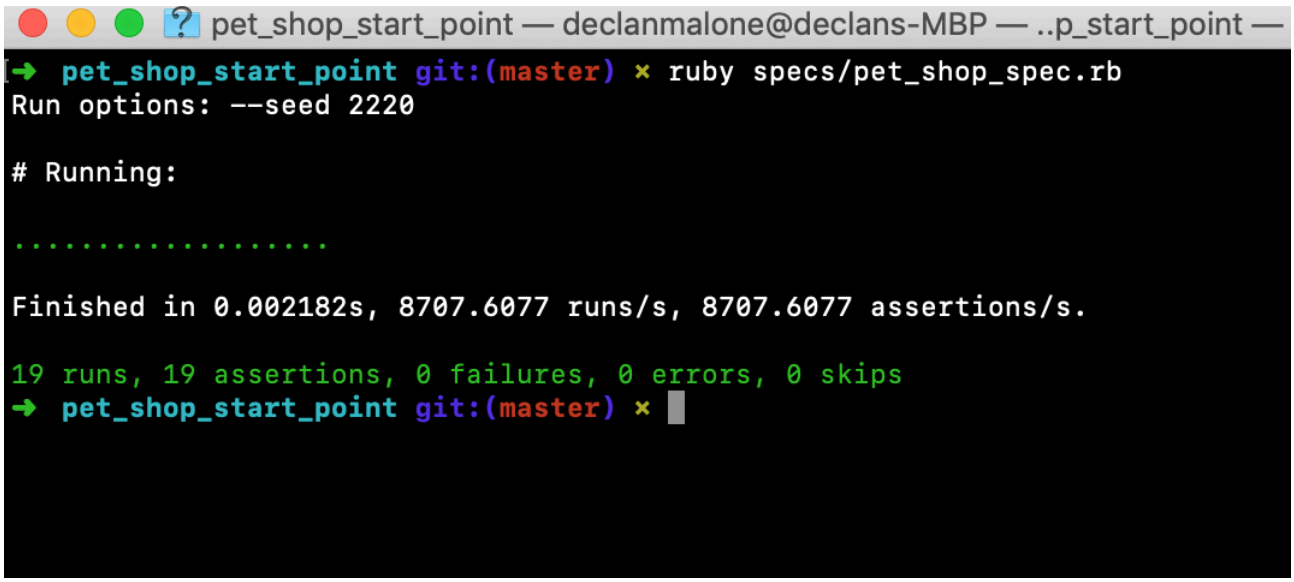


Unit	Ref	Evidence	
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running	
		Description: Created a Pet Shop with customer and pets as hashes within an array. We created functions for basic operations of a pet to function, like sell pet, cash total, search for pets.	

```ruby
require('minitest/autorun')
require('minitest/rg')
require_relative('../pet_shop')

class TestPetShop < Minitest::Test

  def setup

    @customers = [
      {
        name: "Alice",
        pets: [],
        cash: 1000
      },
      {
        name: "Bob",
        pets: [],
        cash: 50
      }
    ]

    @new_pet = {
      name: "Bors the Younger",
      pet_type: :cat,
      breed: "Cornish Rex",
      price: 100
    }
```

Paste Screenshot here

```ruby
def pet_shop_name(name)
return name[:name]
end

def total_cash(sum)
  return sum[:admin][:total_cash]
end

def add_or_remove_cash(pet_shop_cash, customer_cash)
pet_shop_cash[:admin][:total_cash] += customer_cash
end

def pets_sold(sold)
return sold[:admin][:pets_sold]
end

def increase_pets_sold(pets_amount, pets_sold)
pets_amount[:admin][:pets_sold] += pets_sold
end

def stock_count(pet_amount)
  pet_amount[:pets].count
end

def pets_by_breed(shop, breed)
  pets = []
  for find_pet in shop[:pets]
   if find_pet[:breed] ==  breed
    pets.push(find_pet[:name])
   end
```



Week 3

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	
		Description: I created a function within my PostgreSQL database to be able to display all of the hashes.	

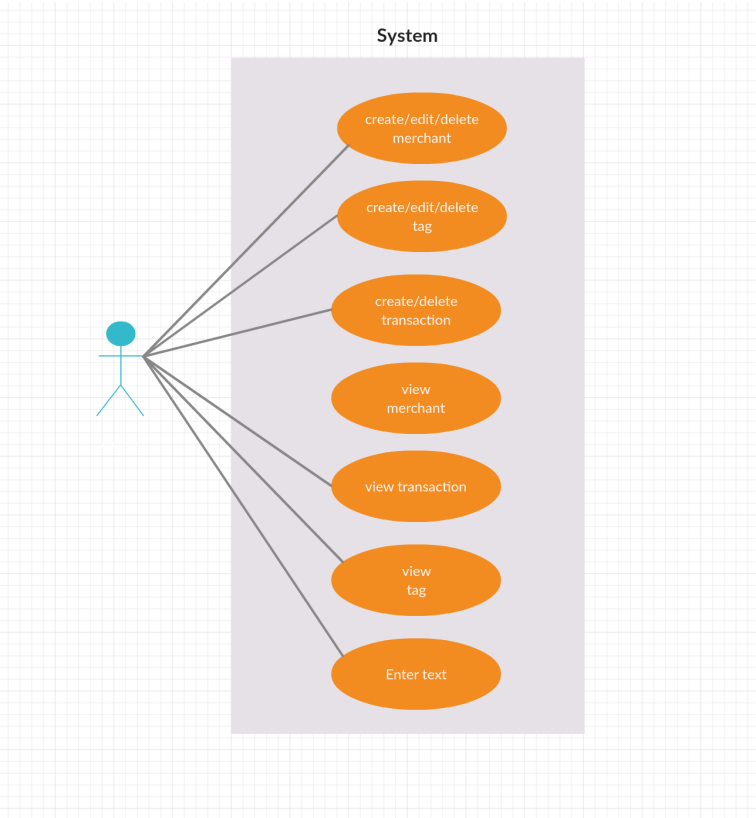
```
[4] pry(main)> merchant1.tags
=> [#<Tag:0x007fef033ba340 @id=8, @tag_name="Books">]
[5] pry(main)>
```

```ruby
def tags()
  sql = "SELECT t.* FROM tags t INNER JOIN transactions transaction ON
  transaction.tag_id = t.id WHERE transaction.merchant_id = $1;"
  values = [@id]
  results = SqlRunner.run(sql, values)
```

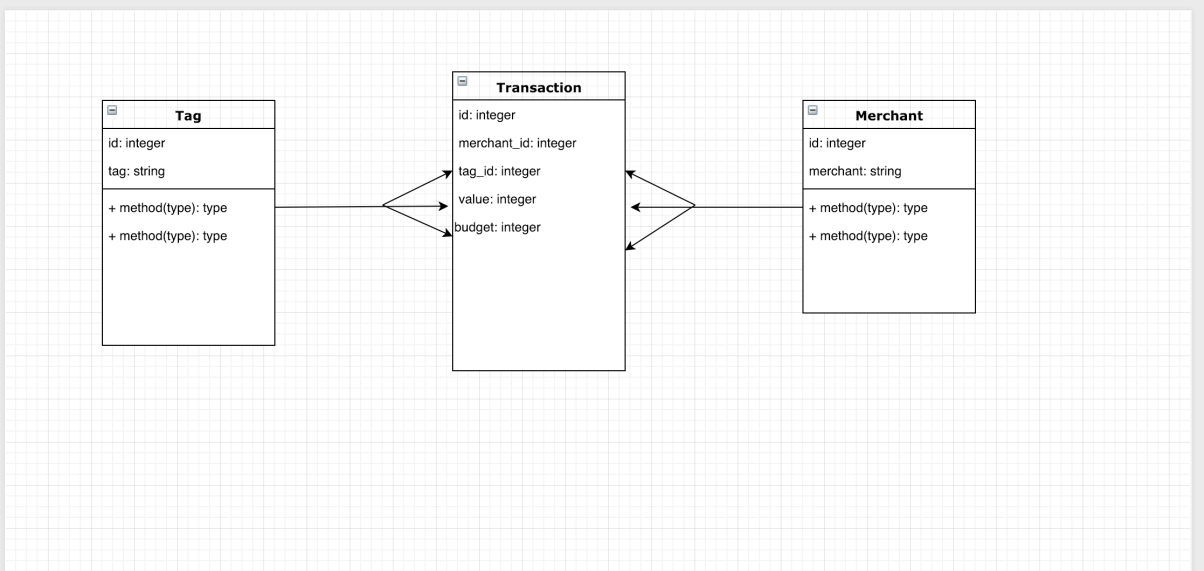
Unit	Ref	Evidence	
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	
		Description: Using an ORDER BY function to sort data in a psql table.	

```
declanmalone=# \c cinema
You are now connected to database "cinema" as user "declanmalone".
cinema=# SELECT * FROM movies
cinema-#
cinema-# ;
ERROR:  relation "movies" does not exist
LINE 1: SELECT * FROM movies
                      ^
cinema=# SELECT * FROM films;
 id  |      title       | price
-----+------------------+-------
 141 | Ex Machina       |    20
 142 | Venom            |    15
 143 | Independence Day |    10
 144 | Black Hawk Down  |    12
(4 rows)

cinema=# SELECT title FROM films ORDER BY title ASC
cinema-# ;
      title
-----------------
 Black Hawk Down
 Ex Machina
 Independence Day
 Venom
(4 rows)

cinema=#
```

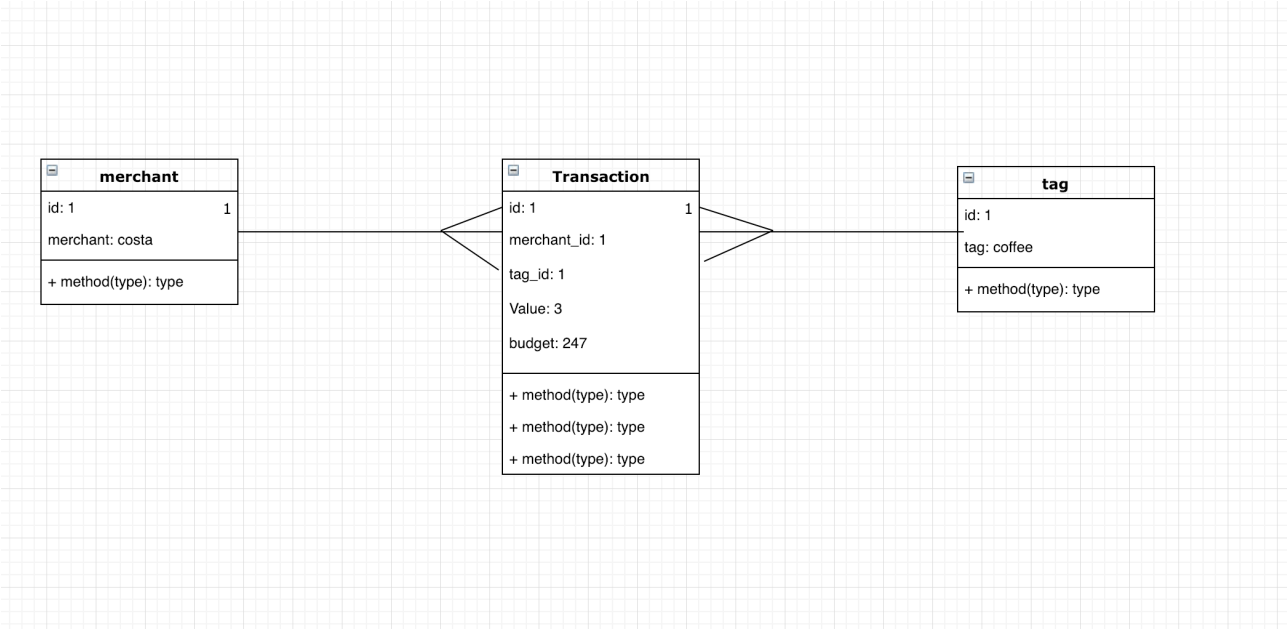
Unit	Ref	Evidence	
A&D	A.D.1	A Case Diagram	
		Description: a flow of how my website would work	



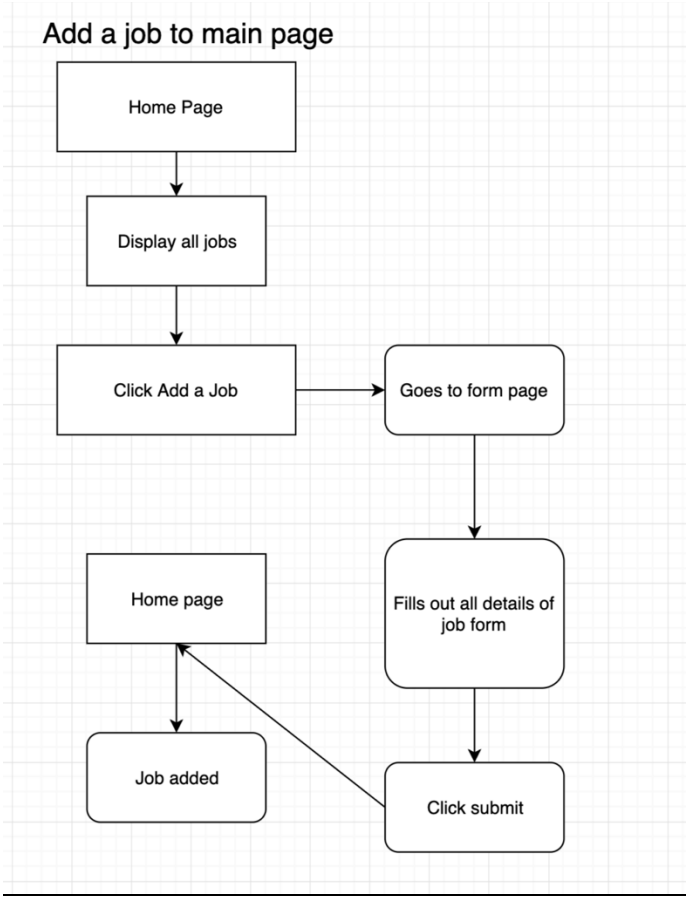
Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	
		Description: a diagram showing the objects classes and what is needed from the object. Objects from my spending tracker app in ruby.	



Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	
		Description: a diagram displaying the object properties of my spending tracker app.	



Unit	Ref	Evidence	
A&D	A.D.4	An Activity Diagram	
		Description: A map displaying the way a user will interact with the app.	

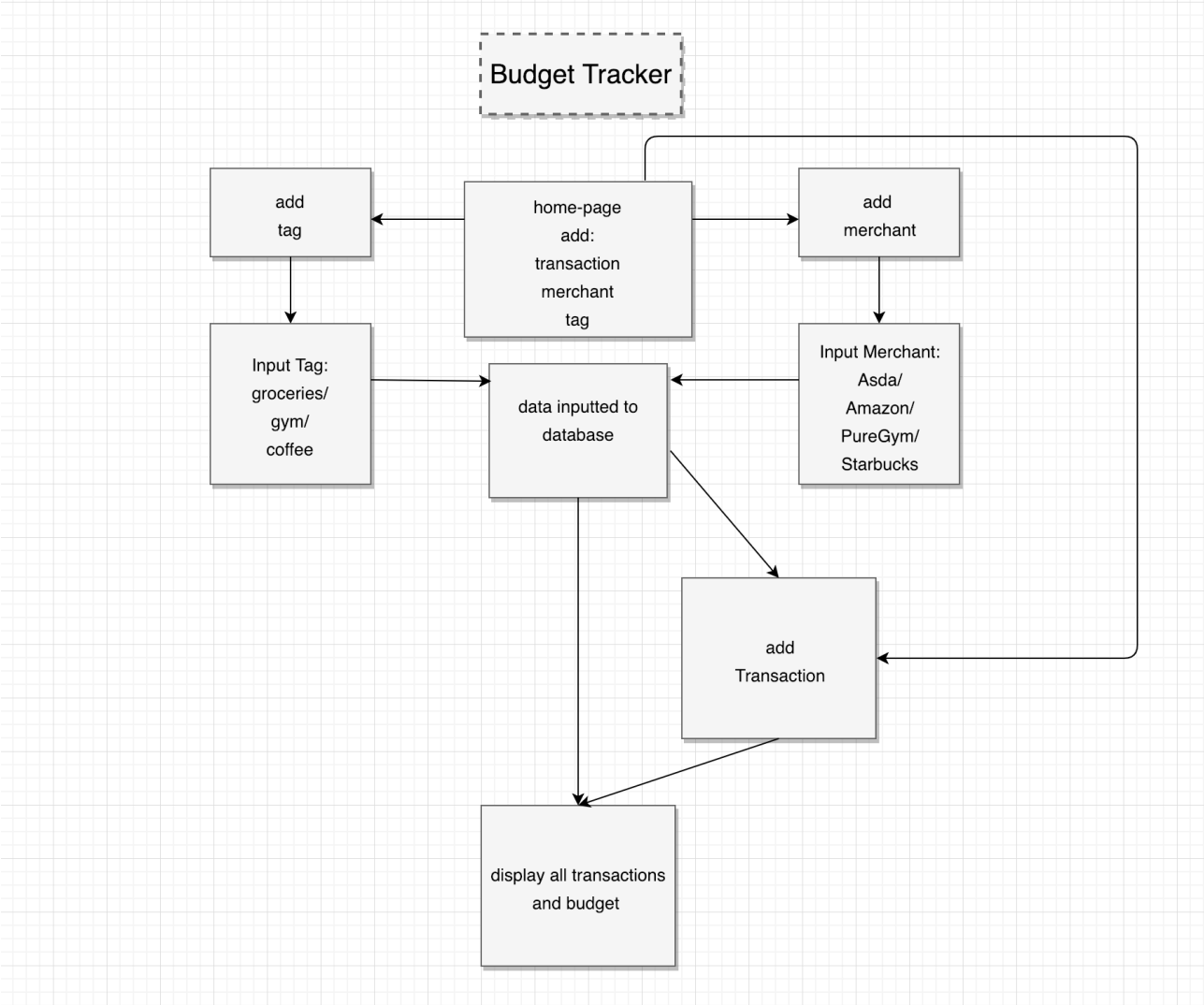


Unit	Ref	Evidence	
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	
		Description: implementation constraints on an app created for our JavaScript project, the recipe app.	

Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	The program was written on a mac and was not considered for Microsoft platforms. It was also only tested on Google Chrome and not checked If it could work on any other browser	Make sure the program can format correctly on a Microsoft stack as well as other browsers
Persistence Storage and Transactions	The database was only on one computer.	Look into buying data storage externally to be hosted
Usability	There was no consideration to people who would have vision disabilities. Could be difficult for a selection of people to see.	Consider this in the user design and stories process
Time	Created issues of not people able to finish the product on time.	Extend the deadline or plan better for time management.

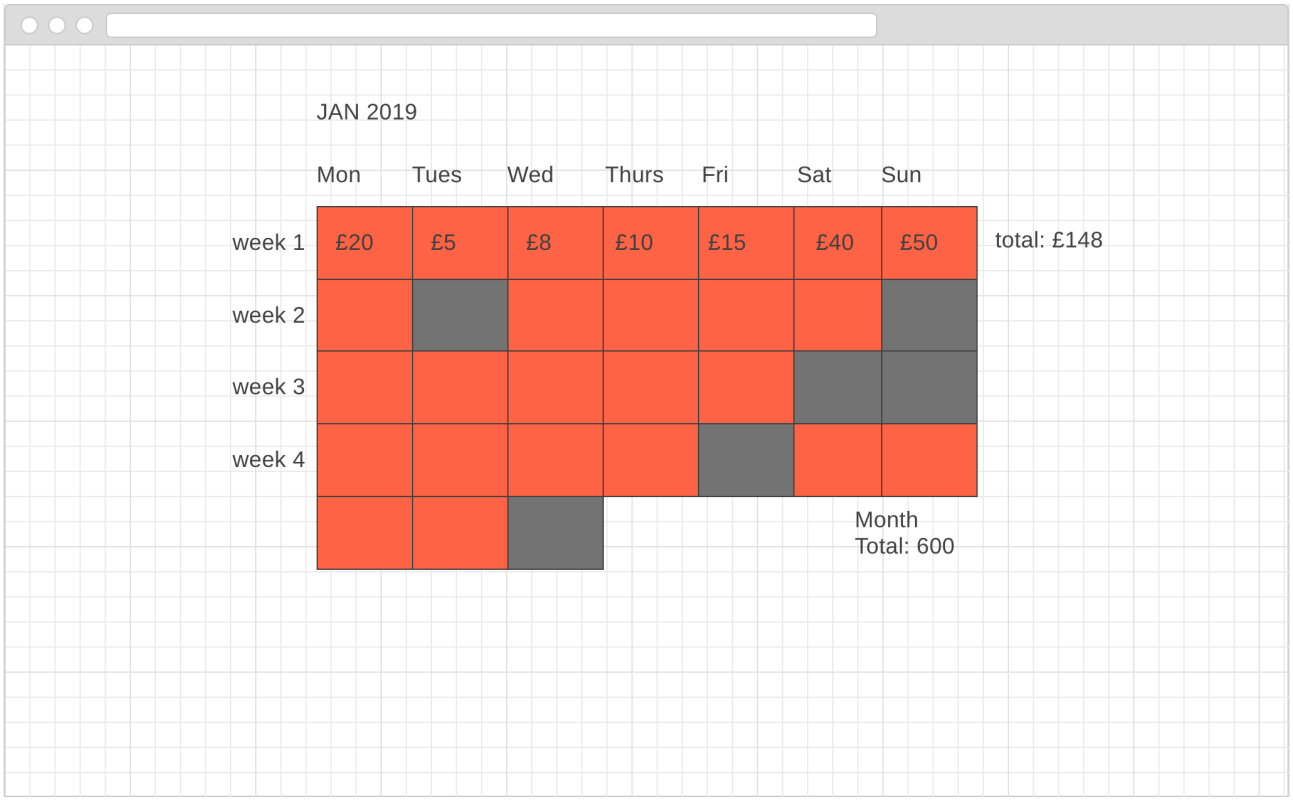
Unit	Ref	Evidence	
P	P.5	User Site Map	

Unit	Ref	Evidence	
		Description: A map to display the way the user will navigate the app.	



Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	
		Description: I made a spending tracker app for my first solo project. These are examples of what I my project could look like. I wanted to initially created my layout for the tracker to look like it did below but it didn't turn out this way. As it was too complicated for the time.	


HOME
MERCHANT
TAG
TRANSACTION
SPENDING TRACKER
Lorem ipsum dolor sit amet et delectus accommodare his consul copiosae legendos at vix ad putent delectus delicata usu. Vidit dissentiet eos cu eum an brute copiosae hendrerit. Eos erant dolorum an. Per facer affert ut. Mei iisque mentitum moderatius cu. Sit munere facilis accusam eu dicat falli consulatu at vis. Te facilisis mnesarchum qui posse omnium mediocritatem est cu. Modus argumentum ne qui tation efficiendi in eos. Ei mea falli legere efficiantur et tollit aliquip debitis mei. No deserunt mediocritatem mel. Lorem ipsum dolor sit amet et delectus accommodare his consul copiosae legendos at vix ad putent delectus delicata usu. Vidit dissentiet eos cu eum an brute copiosae hendrerit. Eos erant dolorum an. Per facer affert ut. Mei iisque mentitum moderatius cu. Sit munere facilis accusam eu dicat falli consulatu at vis. Te facilisis mnesarchum qui posse omnium mediocritatem est cu. Modus argumentum ne qui tation efficiendi in eos. Ei mea falli legere efficiantur et tollit aliquip debitis mei. No deserunt mediocritatem mel. Lorem ipsum dolor sit amet et delectus accommodare his consul copiosae legendos at vix ad putent delectus delicata

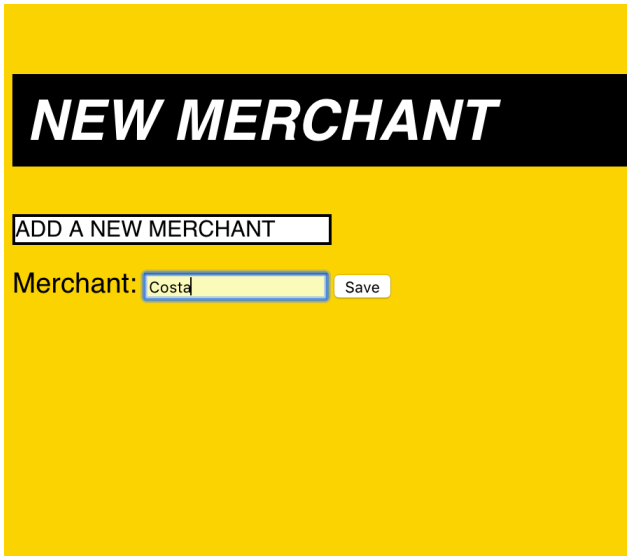

JAN 2019
Mon
Tues
Wed
Thurs
Fri
Sat
Sun
week 1
£20
£5
£8
£10
£15
£40
£50
total: £148
week 2
week 3
week 4
Month
Total: 600

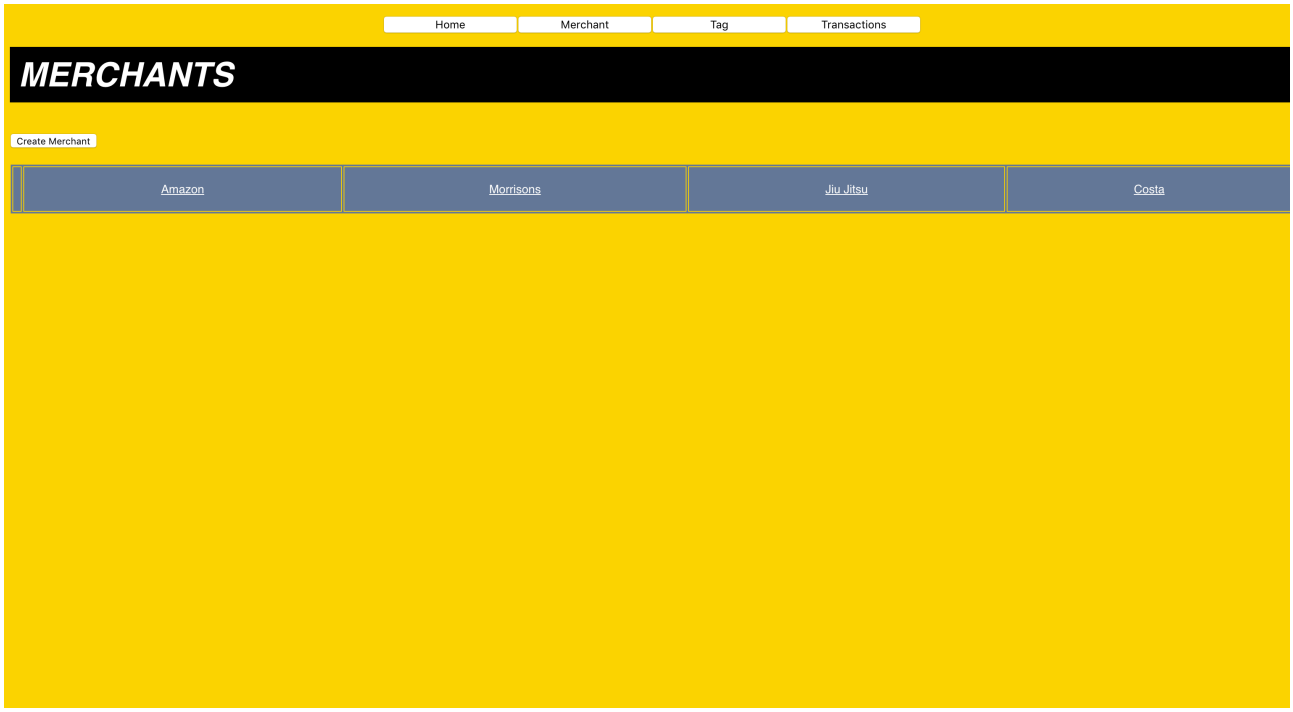
Unit	Ref	Evidence	
P	P.10	Example of Pseudocode used for a method	
		Description: I used an inbuilt random function within Javascript to return 6 random numbers between 1 and the length of the recipes array. I then put those numbers in an array matching the index of the recipe object.	

```
RecipeGridView.prototype.limitRecipes = function(recipes){
  //set up empty array
    const randomNumbers = [];
    //increment the index number up to 6
    for(let i=0; i < 6; i++){
      //randomnumber = a random 6 numbers up to the length of
      the recipes array.
      const randomNumber = Math.floor(Math.random() *
      recipes.length -1) + 1
      //push those 6 index numbers into array
      randomNumbers.push(randomNumber);
    };
    //set new arary for 6 recipes
    const limitedRecipes = [];
    //for each of the randomNumbers with there index
    randomNumbers.forEach((randomNumber) => {
      //the recipe index number is the same number as the random
      one
      const recipe = recipes[randomNumber];//[1,14,12,21,8]
      //push those recipes indexes into the new array
      limitedRecipes.push(recipe);
    });
    //pash the limitRecipes into render function
    return limitedRecipes;
}
```

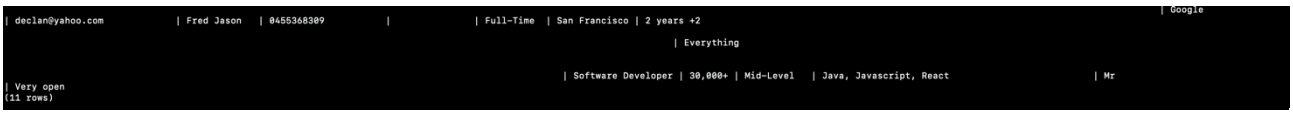
Paste Screenshot here

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	
		Description: Inputting a merchant in my spending tracker app which stores the results in the merchant tag page.	

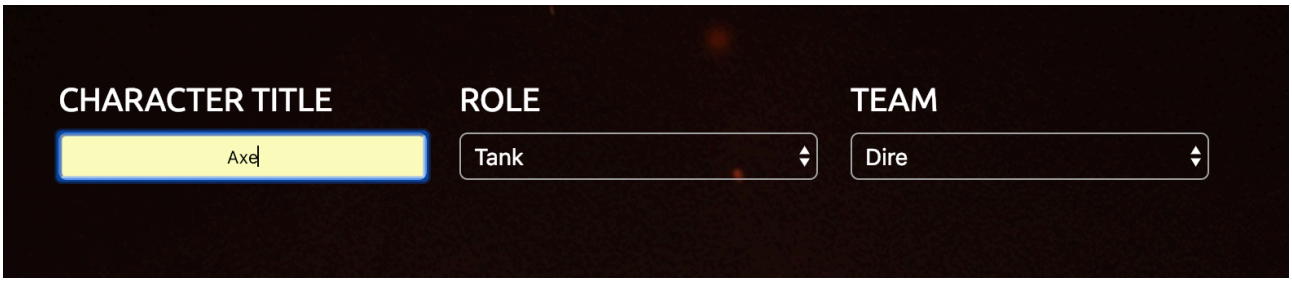




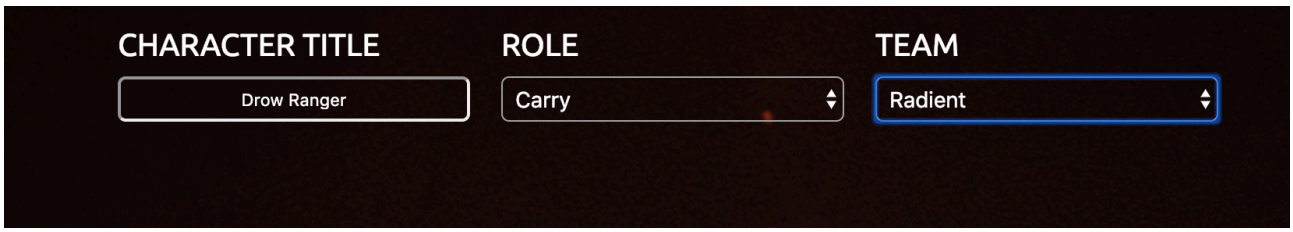
Unit	Ref	Evidence	
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved	
		Description: Data being stored in our recruitment app and then displayed on the page where the data should be posted to.	

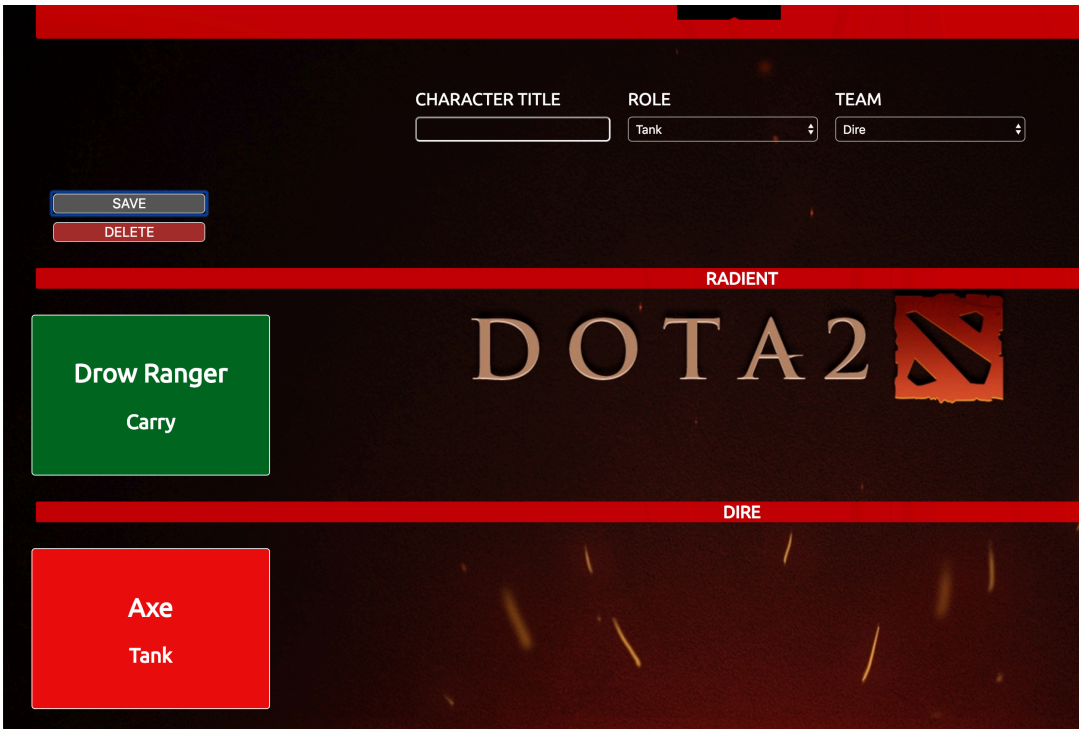


Unit	Ref	Evidence	
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program	
		Description: adding in character to specific team. Input character name specific the role, then the team which will place in a container depending on the result.	

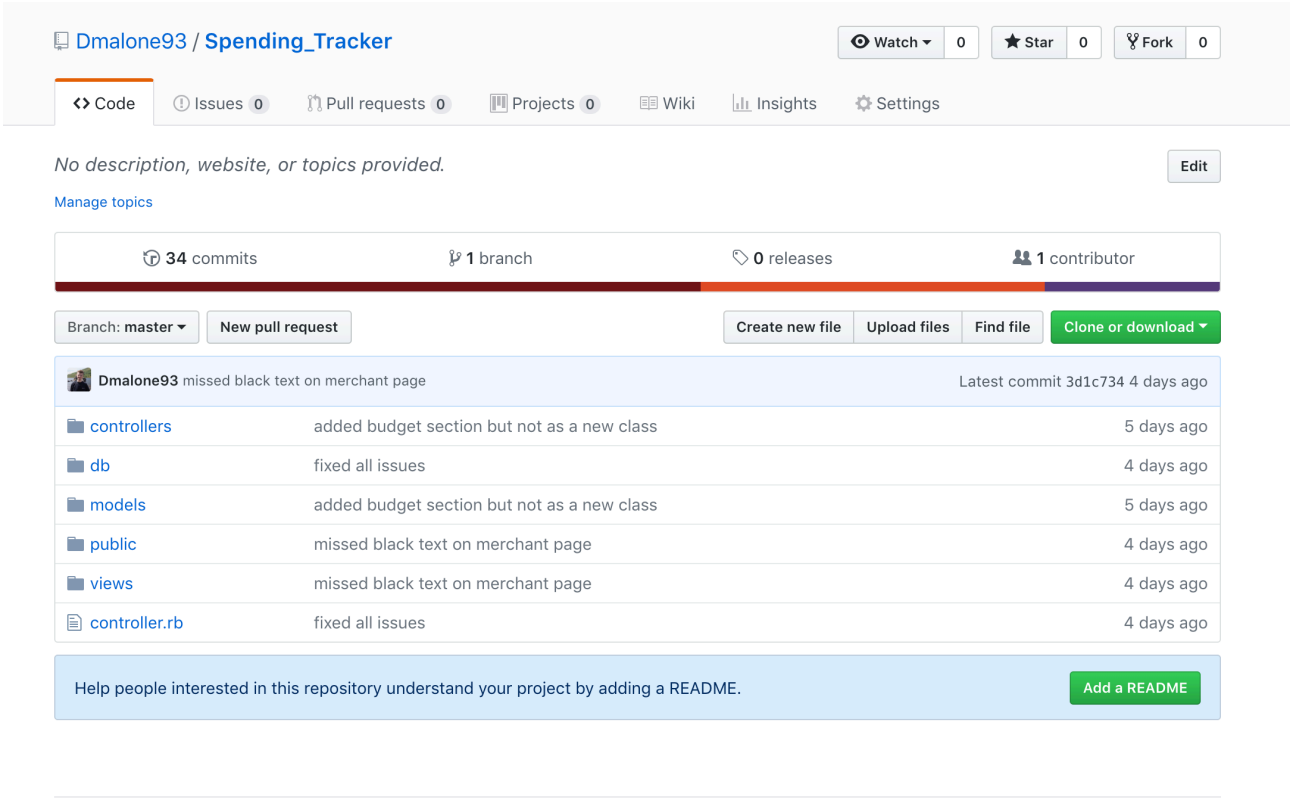


Paste Screenshot here





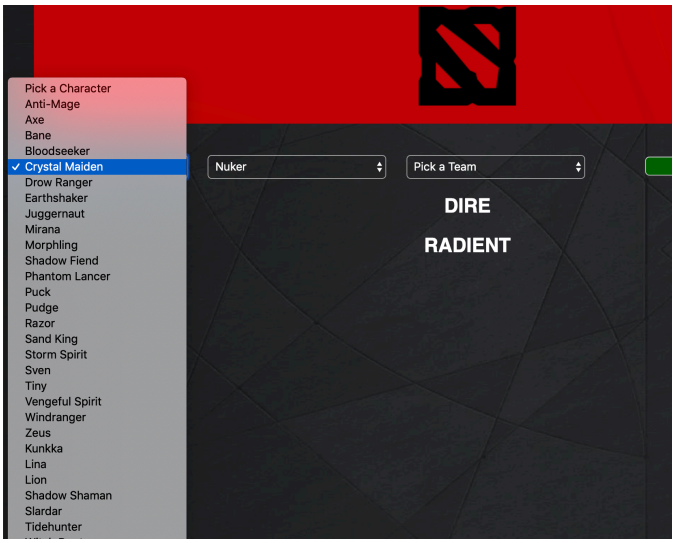
Unit	Ref	Evidence	
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	
		Description: https://github.com/Dmalone93/Spending_Tracker	



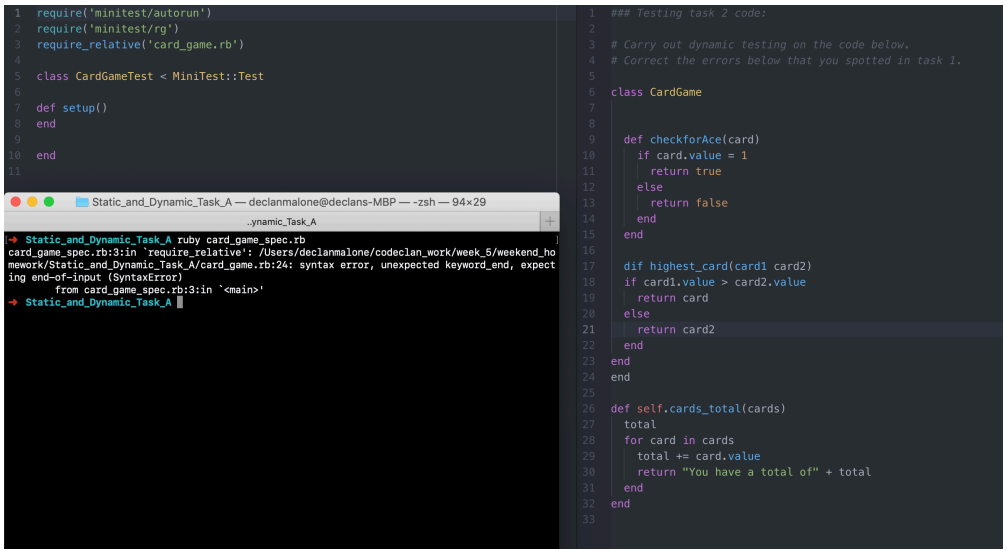
Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	
		Description: I used a DOTA 2 api which is based on an online game. I used the hero name keys to populate the first dropdown to select a character. Below follows my request to the API data and the web pack npm that is updating the changes	

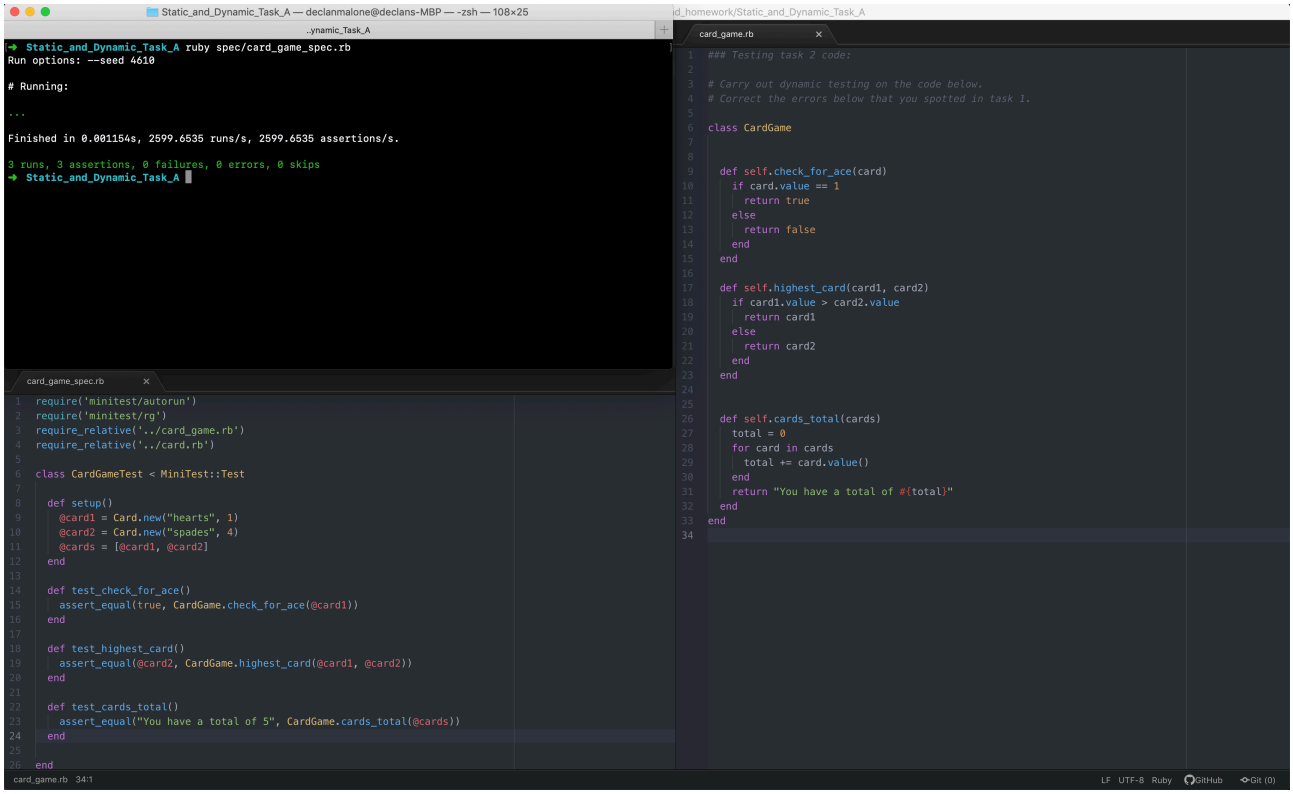
```
15
16  Dota.prototype.getData = function(){
17    const request = new Request('https://api.opendota.com/api/heroes')
18    request.get().then((data) => {
19      this.heroes = data;
20      PubSub.publish('Dota:all-ready', this.heroes)
21    });
22  };
23
24
```

```
webpack is watching the files…

Hash: da14d656cabc42fdbefb
Version: webpack 4.28.4
Time: 139ms
Built at: 14/01/2019 10:31:02
    Asset      Size  Chunks             Chunk Names
bundle.js  11.8 KiB    main  [emitted]  main
Entrypoint main = bundle.js
[./src/app.js] 1.38 KiB {main} [built]
[./src/helpers/pub_sub.js] 301 bytes {main} [built]
[./src/helpers/request.js] 208 bytes {main} [built]
[./src/models/dota.js] 619 bytes {main} [built]
[./src/views/dota_detail_view.js] 1.13 KiB {main} [built]
[./src/views/dota_list_view.js] 1.1 KiB {main} [built]
[./src/views/select_view.js] 805 bytes {main} [built]
```



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	
		Description: test code running in top screenshot and then test code passing.	

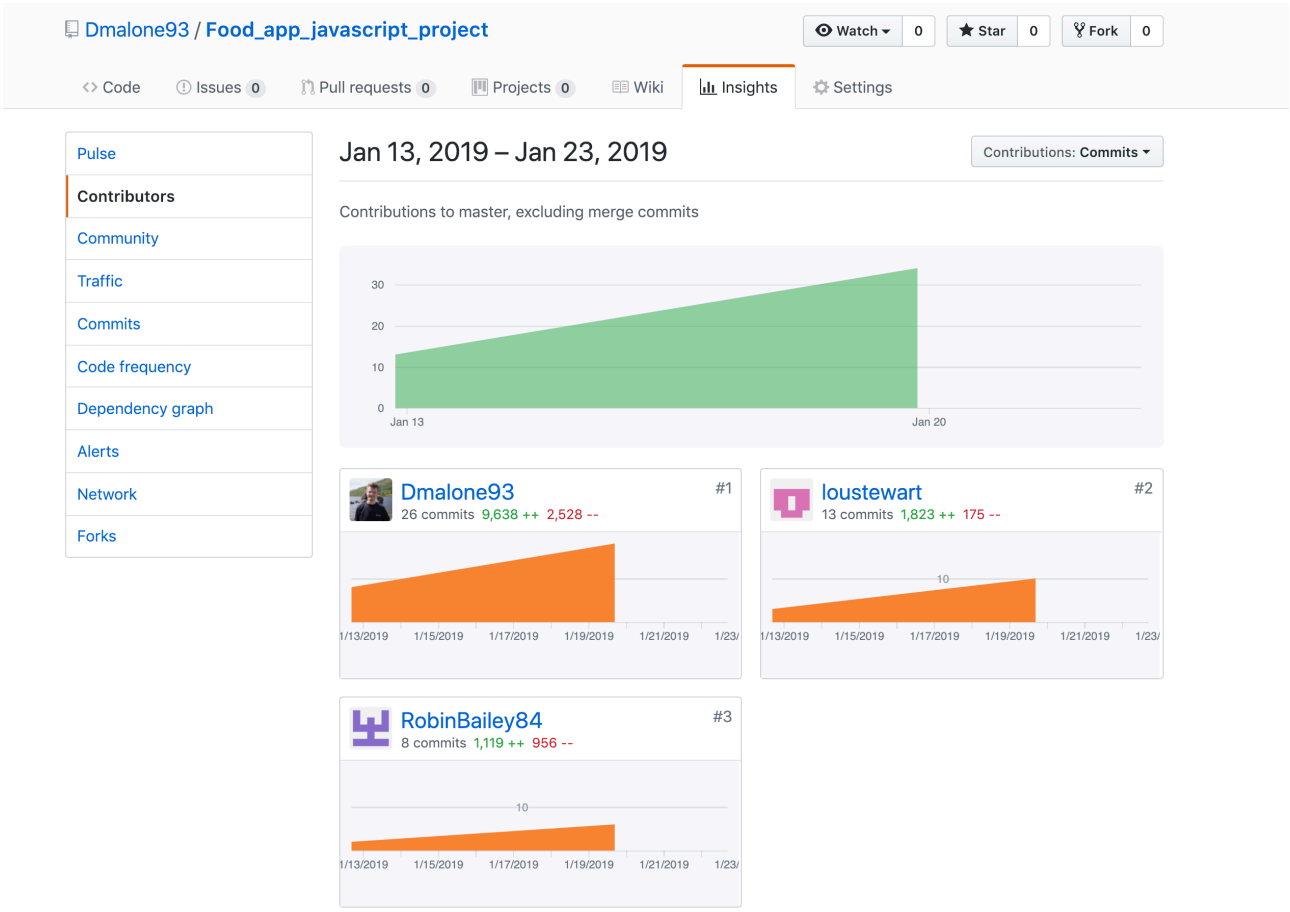




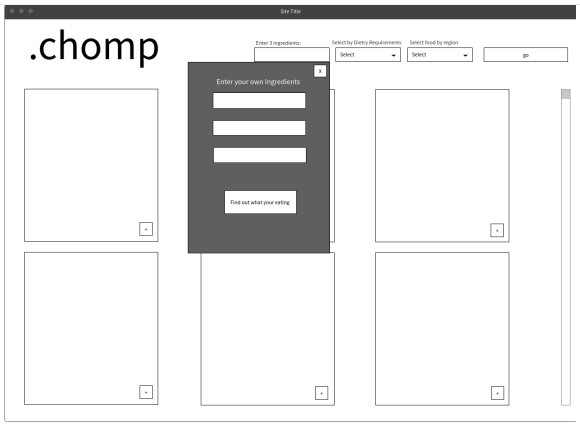
Week 9

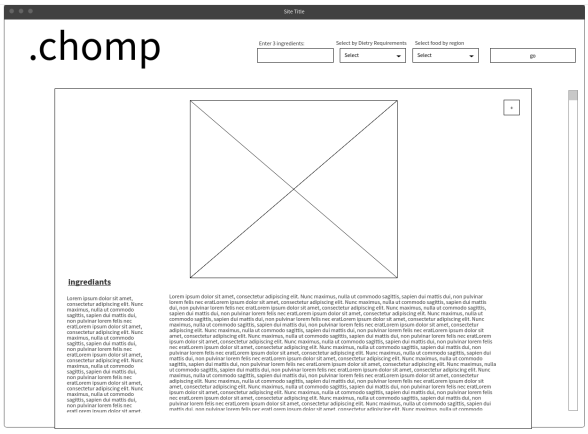
Unit	Ref	Evidence	
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.	
		Description: GitHub contributors team and their activity	

Paste Screenshot here



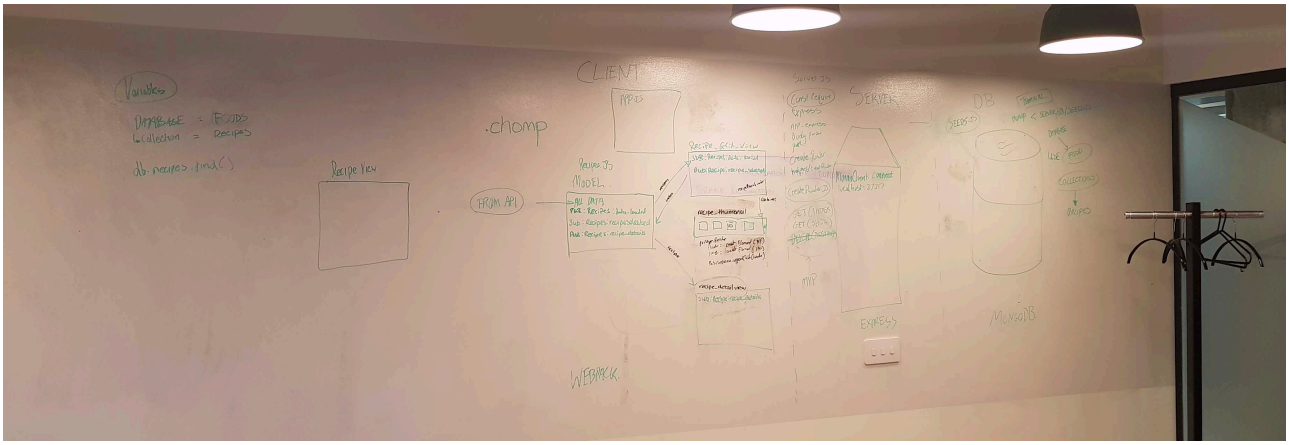
Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	
		Description: wireframe layout of our web app. Then moved onto data flow/interaction between client	





Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	
		Description: group project in javascript module. This involved 2 other members from the cohort.	

<u>Paste Screenshot here</u>



README.md

Food_app_javascript_project

Recipe Finder App

MVP

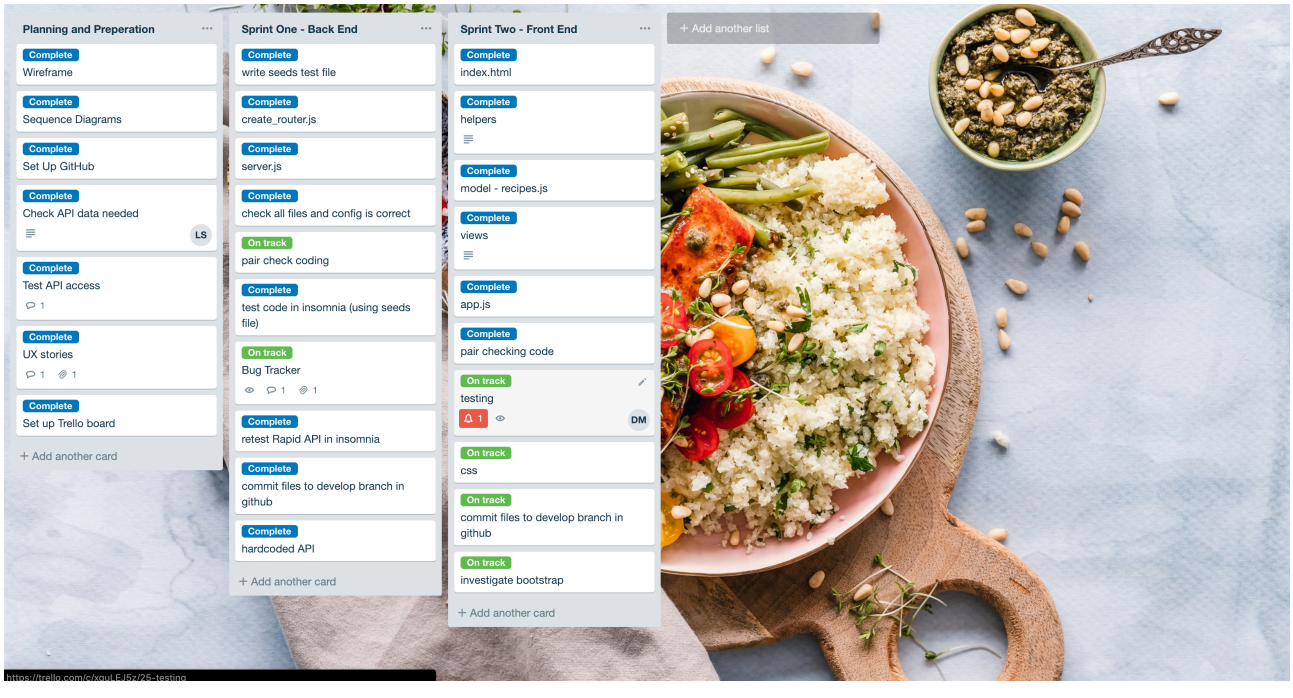
Different options available for generating recipes including a random function. App will display search results with images and recipe titles. Clicking on thumbnails will render a page with the full recipe details and links to the original recipe. Users can save their favourite recipes to their own 'recipe book'. User can have more than one recipe book e.g. mains, vegan etc. User can add their own recipes to their 'recipe book'.

Extensions A plus button in the thumbnail images will allow a quick add function to add to the recipe book. Users can add comments to their recipes.

Different options available for generating recipes including a random function. App will display search results with images and recipe titles. Clicking on thumbnails will render a page with the full recipe details and links to the original recipe. Users can save their favourite recipes to their own 'recipe book'. User can have more than one recipe book e.g. mains, vegan etc. User can add their own recipes to their 'recipe book'.

Extensions A plus button in the thumbnail images will allow a quick add function to add to the recipe book. Users can add comments to their recipes.

Unit	Ref	Evidence	
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	
		Description: our trellio board for our JavaScript recipe project. Done over 3 sprints	

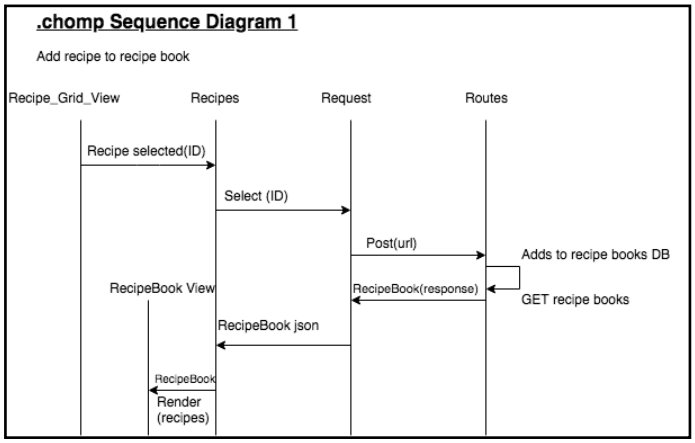


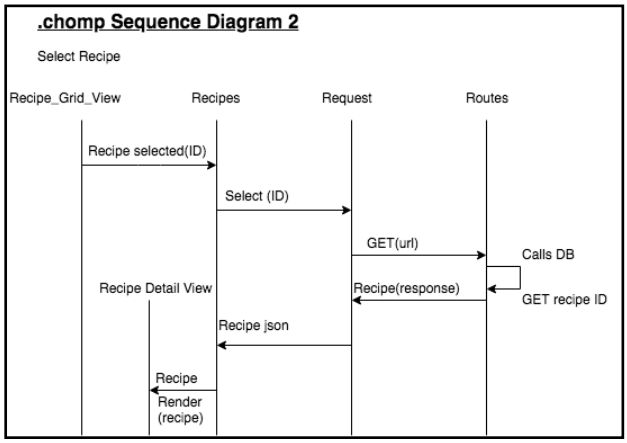
Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	
		Acceptance Criteria for our JavaScript recipe add project. This was made at the start of the project.	

Acceptance Criteria	Expected Result/Output	Pass/Fail
A user is able to view a random selection of recipes from the site.	The home page renders a random selection of six recipe each time the user lands on the home page.	Pass
A user can select a recipe they want more details for.	Taken to the recipe page giving details of the ingredients recipe etc.	Pass
A user can search the site for recipes within a defined diet type.	The site renders all the recipes within that diet type.	Pass
A user can search the site for recipes using an ingredient.	The site renders all recipes which include the specified ingredient.	Fail
The user can save recipes to their own recipe book to refer to later.	The site renders a page showing only recipes the user has saved.	Pass
The user can add their own personal recipes to their recipe book.	The site adds the recipes to the recipe book and show the recipes the user has saved from the site and those added.	Fail

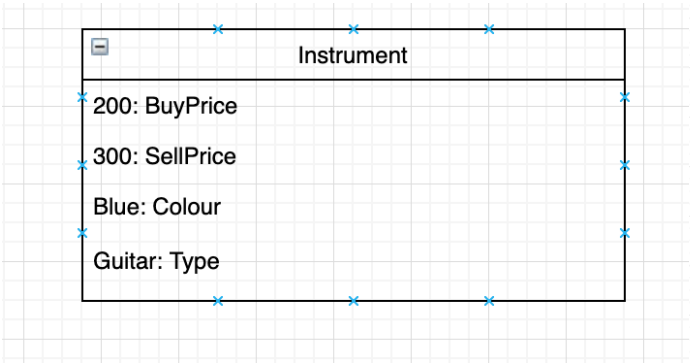
Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	

Unit	Ref	Evidence	
		Description: Systems diagrams displaying how one recipe and multiple recipes will be retrieved.	





Unit	Ref	Evidence	
P	P.8	Produce two object diagrams.	
		Description: one object diagram of an instrument and the other of a job	



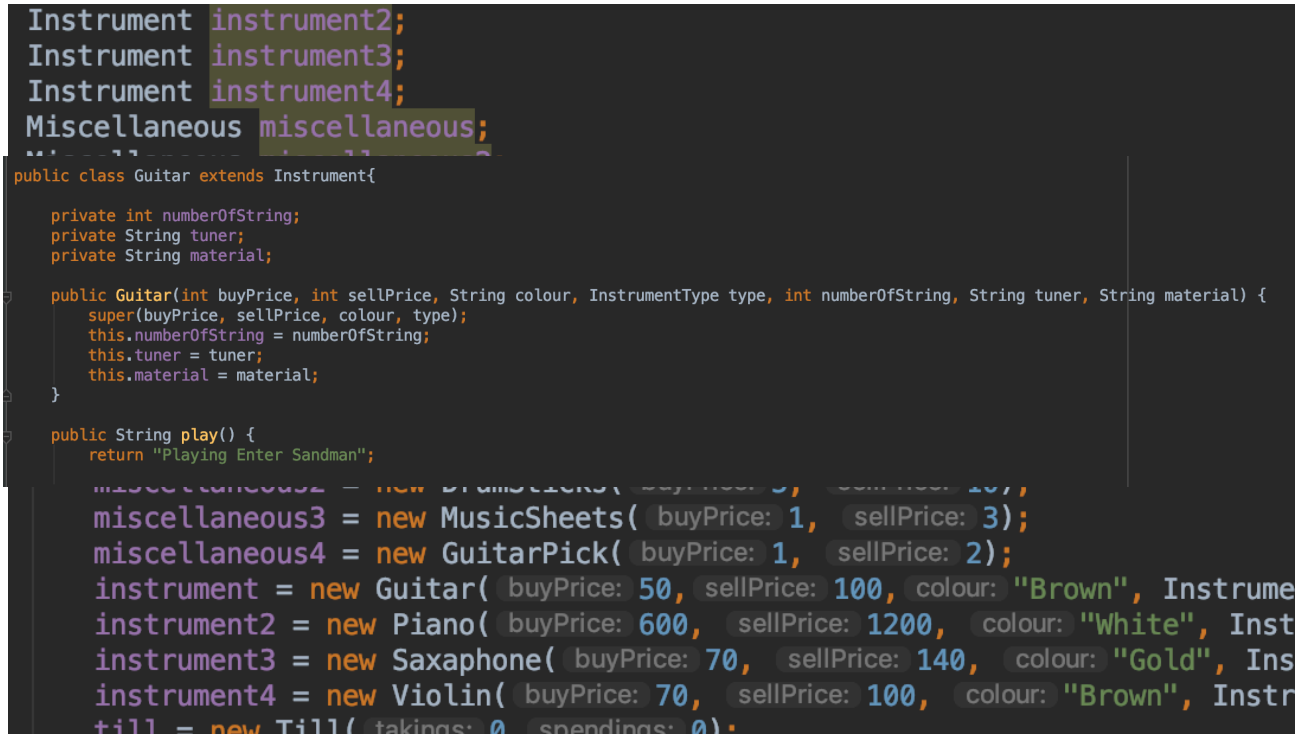
Unit	Ref	Evidence	
P	P.17	Produce a bug tracking report	
		Description: A bug tracking report that was used to track what issues.	

Bug Tracking Report

Issue	Solution	Pass/Fail
Installing all npm packages, webpack and getting the server to run	app.js file was outside the src directory.	
random function would bring the recipes length with is 22 but the indexes only goes up to 21. So there would be an undefined object returned every couple of refreshes.	reduce the recipe.length's number by 1.	
returning list of ingredients. Would bring back every ingredient as a variable and not an instance of each item.	appended the ingredientsItem in the ingredientsList inside the for loop.	
Creating text inside the div		
random function returns duplicates of recipes.	added in functionality to delete multiple recipes of the same id.	
our search by ingredients takes you to recipe book.	included an ID into the html node, to be able to get clearer access to element.	
recipe book was storing objects with only id'd and no key/values		

Week 12

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	
		Description: Turning an instrument class into a Guitar object	



```
Instrument instrument2;
Instrument instrument3;
Instrument instrument4;
Miscellaneous miscellaneous;
```

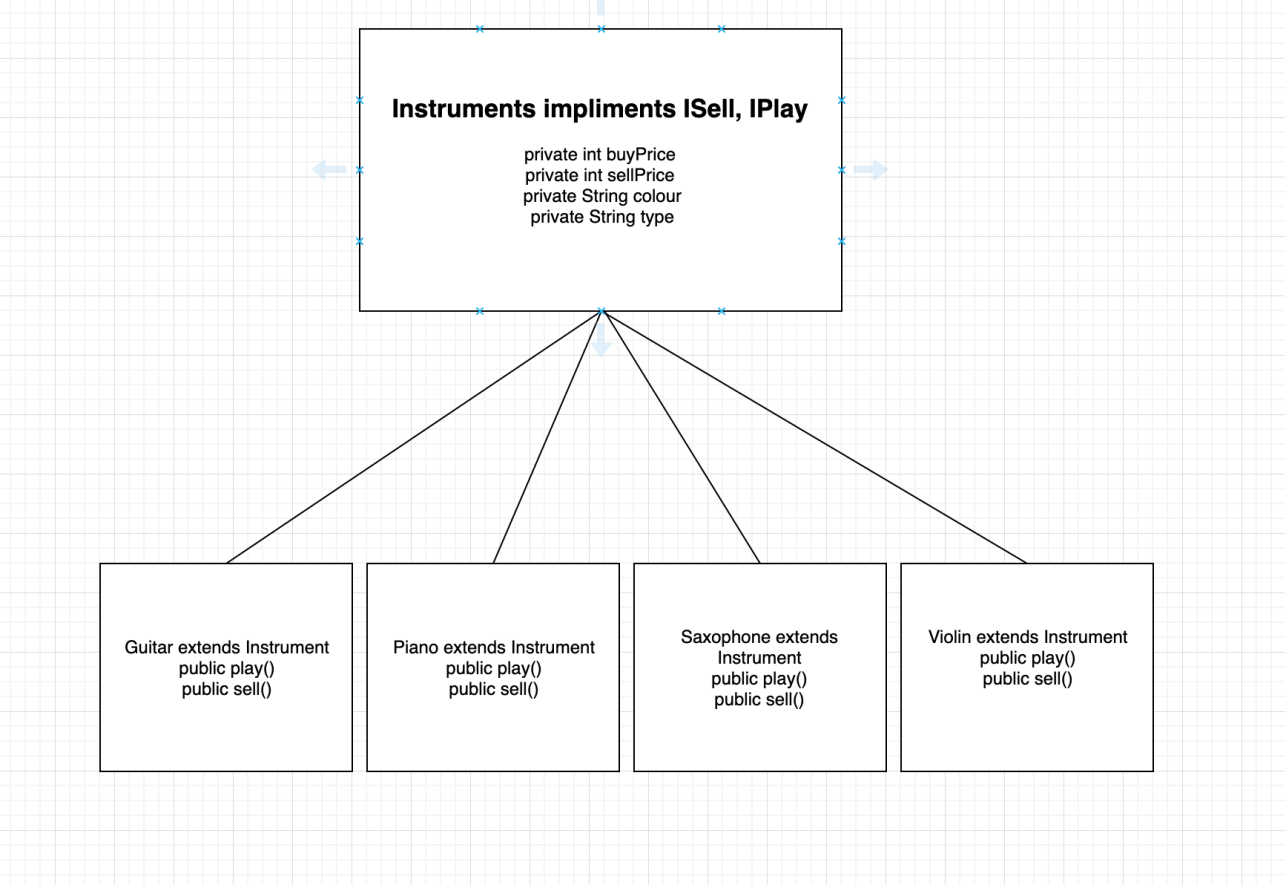
```
public class Guitar extends Instrument{

    private int numberOfString;
    private String tuner;
    private String material;

    public Guitar(int buyPrice, int sellPrice, String colour, InstrumentType type, int numberOfString, String tuner, String material) {
        super(buyPrice, sellPrice, colour, type);
        this.numberOfString = numberOfString;
        this.tuner = tuner;
        this.material = material;
    }

    public String play() {
        return "Playing Enter Sandman";
```

```
        miscellaneous3 = new MusicSheets( buyPrice: 1,  sellPrice: 3);
        miscellaneous4 = new GuitarPick( buyPrice: 1,  sellPrice: 2);
        instrument = new Guitar( buyPrice: 50,  sellPrice: 100,  colour: "Brown", Instrume
        instrument2 = new Piano( buyPrice: 600,  sellPrice: 1200,  colour: "White", Inst
        instrument3 = new Saxaphone( buyPrice: 70,  sellPrice: 140,  colour: "Gold", Ins
        instrument4 = new Violin( buyPrice: 70,  sellPrice: 100,  colour: "Brown", Instr
```

Paste Screenshot here

Unit	Ref	Evidence	
A&D	A.D.5	An Inheritance Diagram	
		Description: The parent class 'instrument' is inherited by the 4 different child classes.	

```java
public abstract class Instrument implements IPlay, ISell {

    private int buyPrice;
    private int sellPrice;
    private String colour;
    private InstrumentType type;

    public Instrument(int buyPrice, int sellPrice, String colour, InstrumentType type){
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
        this.colour = colour;
        this.type = type;
    }

    public int getBuyPrice() {
        return buyPrice;
    }

    public void setBuyPrice(int buyPrice) {
        this.buyPrice = buyPrice;
    }
```



Unit	Ref	Evidence	
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.	
		Description: the buyItem function encapsulates a function fron the person class and from the instrument class.	

```
public void buyItem(ISell item){
    this.items.add(item);
    setMoney(getMoney() - item.getSellPrice());
}
```

```
public class Person {

    private int money;
    private String name;

    public Person(int money, String name) {
        this.money = money;
        this.name = name;
    }

    public int getMoney() {
        return this.money;
    }
```

```
public Instrument(int buyPrice, int sellPrice, String colour, InstrumentType type){
    this.buyPrice = buyPrice;
    this.sellPrice = sellPrice;
    this.colour = colour;
    this.type = type;
}

public int getBuyPrice() {
    return buyPrice;
}

public void setBuyPrice(int buyPrice) {
    this.buyPrice = buyPrice;
}

public int getSellPrice() {
    return sellPrice;
}

public void setSellPrice(int sellPrice) {
    this.sellPrice = sellPrice;
}
```

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.	
		Description:A display of a guitar inheriting from an instrument and using a function from the parent class.	

```
public abstract class Instrument implements IPlay, ISell {

    private int buyPrice;
    private int sellPrice;
    private String colour;
    private InstrumentType type;

    public Instrument(int buyPrice, int sellPrice, String colour, InstrumentType type){
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
        this.colour = colour;
        this.type = type;
    }

    public int getBuyPrice() {
        return buyPrice;
    }

    public void setBuyPrice(int buyPrice) {
        this.buyPrice = buyPrice;
    }

    public int getSellPrice() {
        return sellPrice;
    }

    public void setSellPrice(int sellPrice) {
        this.sellPrice = sellPrice;
    }

    public String getColour() {
        return colour;
    }

    public void setColour(String colour) {
        this.colour = colour;
    }

    public InstrumentType getType() {
        return type;
    }

    public void setType(InstrumentType type) {
        this.type = type;
    }
```

```
Instrument instrument;
Instrument instrument2;
Instrument instrument3;
Instrument instrument4;
Guitar guitar;

@Before
public void setUp(){
    instrument = new Guitar( buyPrice: 50, sellPrice: 100, colour: "Brown",InstrumentType.STRING,  numberOfString: 6,  tuner: "D",  material: "Wood");
    guitar = new Guitar( buyPrice: 50, sellPrice: 100, colour: "Brown",InstrumentType.STRING,  numberOfString: 6,  tuner: "D",  material: "Wood");
    instrument2 = new Piano( buyPrice: 600,  sellPrice: 1200,  colour: "White", InstrumentType.KEYBOARD);
    instrument3 = new Saxaphone( buyPrice: 70,  sellPrice: 140,  colour: "Gold", InstrumentType.WOODWIND);
    instrument4 = new Violin( buyPrice: 70,  sellPrice: 100,  colour: "Brown", InstrumentType.STRING);
}
```

```
public class Guitar extends Instrument{

    private int numberOfString;
    private String tuner;
    private String material;

    public Guitar(int buyPrice, int sellPrice, String colour, InstrumentType type, int numberOfString, String tuner, String material) {
        super(buyPrice, sellPrice, colour, type);
        this.numberOfString = numberOfString;
        this.tuner = tuner;
        this.material = material;
    }

    public String play() {
        return "Playing Enter Sandman";

    }

    public int getNumberOfString() {
        return numberOfString;
    }

    public void setNumberOfString(int numberOfString) {
        this.numberOfString = numberOfString;
    }

    public String getTuner() {
        return tuner;
    }

    public void setTuner(String tuner) {
        this.tuner = tuner;
    }

    public String getMaterial() {
        return material;
    }

    public void setMaterial(String material) {
        this.material = material;
    }
}
```

```
@Test
public void hasBuyPrice() {
    assertEquals( expected: 50, instrument.getBuyPrice());
}
```

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.	
		Description: the first algorithm randomises seat numbers for a plane. The other takes random recipes and displays them at the front page each time it is refreshed.	

Paste Screenshot here

```java
public int getPlaneCapacity(){
    return flight.getPlaneCapacity();
}

public int randomSeatNumber(){


    int max = getPlaneCapacity();
    int min = 1;
    int range = max - min + 1;
    int rand = 0;
    for (int i = 0; i < max; i++){
        rand = (int)(Math.random() * range) + min;
    }

    return rand;
}
```

```
RecipeGridView.prototype.limitRecipes = function(recipes){
  //set up empty array
    const randomNumbers = [];
    //increment the index number up to 6
    for(let i=0; i < 6; i++){
      //randomnumber = a random 6 numbers up to the length of
      the recipes array.
      const randomNumber = Math.floor(Math.random() *
      recipes.length -1) + 1
      //push those 6 index numbers into array
      randomNumbers.push(randomNumber);
    };
    //set new arary for 6 recipes
    const limitedRecipes = [];
    //for each of the randomNumbers with there index
    randomNumbers.forEach((randomNumber) => {
      //the recipe index number is the same number as the random
      one
      const recipe = recipes[randomNumber];//[1,14,12,21,8]
      //push those recipes indexes into the new array
      limitedRecipes.push(recipe);
    });
    //pash the limitRecipes into render function
    return limitedRecipes;
}
```