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.

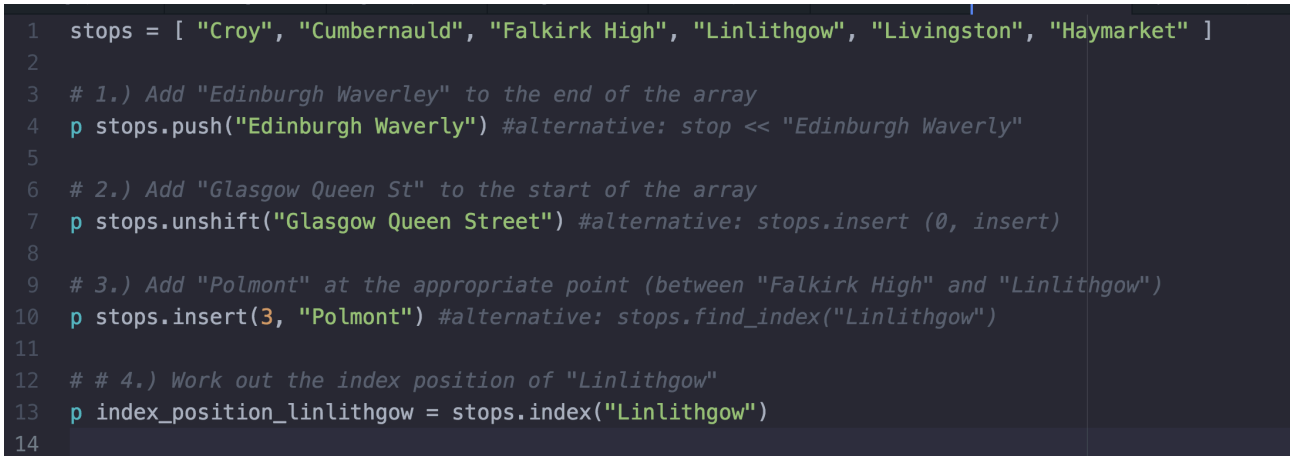
Please fill in each point with screenshot or diagram and description of what you are showing.

Week 2
Array & function using array

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
		Description:

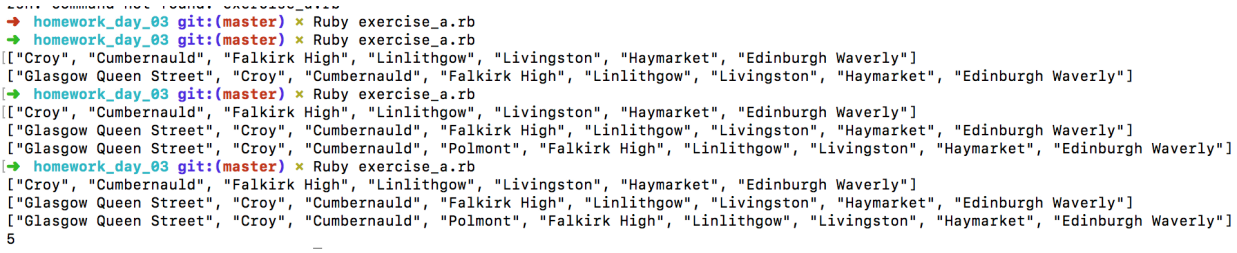
Paste Screenshot here

Image 1: Array_&function_using_array



```
stops = [ "Croy", "Cumbernauld", "Falkirk High", "Linlithgow", "Livingston", "Haymarket" ]

# 1.) Add "Edinburgh Waverley" to the end of the array
p stops.push("Edinburgh Waverly") #alternative: stop << "Edinburgh Waverly"

# 2.) Add "Glasgow Queen St" to the start of the array
p stops.unshift("Glasgow Queen Street") #alternative: stops.insert (0, insert)

# 3.) Add "Polmont" at the appropriate point (between "Falkirk High" and "Linlithgow")
p stops.insert(3, "Polmont") #alternative: stops.find_index("Linlithgow")

# # 4.) Work out the index position of "Linlithgow"
p index_position_linlithgow = stops.index("Linlithgow")

```

Image 2: Result of the function



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:

Paste Screenshot here

Image 3: Hash in program

```ruby
users = {
  "Jonathan" => {
    :twitter => "jonnyt",
    :lottery_numbers => [6, 12, 49, 33, 45, 20],
    :home_town => "Stirling",
    :pets => [
    {
      :name => "fluffy",
      :species => "cat"
    },
    {
      :name => "fido",
      :species => "dog"
    },
    {
      :name => "spike",
      :species => "dog"
    }
  ]
  },
  "Erik" => {
    :twitter => "eriksf",
    :lottery_numbers => [18, 34, 8, 11, 24],
    :home_town => "Linlithgow",
    :pets => [
    {
      :name => "nemo",
      :species => "fish"
    },
    {
```

Image 3: Function using hash

```ruby
# 1.) Get Jonathan's Twitter handle (i.e. the string "jonnyt")
p users ["Jonathan"][:twitter] # alternative: p users.fetch_values("Jonathan")

# 2.) Get Erik's hometown
p users ["Erik"][:home_town]

# 3.) Get the array of Erik's lottery numbers
p users ["Erik"] [:lottery_numbers]

# 4.) Get the type of Avril's pet Monty
p users ["Avril"] [:pets] [0] [:species] # '[0]' because array referencing index brings back pet

# 5.) Get the smallest of Erik's lottery numbers
p users ["Erik"] [:lottery_numbers].min #arrays have a smallest number method, i.e. the one used here

# 6.) Return an array of Avril's lottery numbers that are even
p users ["Avril"] [:lottery_numbers].select {|num| num.even? }
=begin
alternative a):
even numbers[]
for number in users ["Avril"] [lottery_numbers]
  even_numbers.push(number) if number even
end
=end
#alternative b): could have worked with '% = 0', i.e. remainder = 0, thus even
```

Image 4: Result of the function running

```
➜  homework_day_03 git:(master) × Ruby exercise_b.rb
"jonnyt"
"Linlithgow"
[18, 34, 8, 11, 24]
"snake"
8
[12, 14, 38]
[18, 34, 8, 11, 24, 7]
"Edinburgh"
```

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:

Paste Screenshot here

Image 5: Function that searches data

```
def films()
  sql = "SELECT films.* FROM films INNER JOIN tickets ON
  films.id = tickets.film_id WHERE tickets.customer_id =$1"
  values = [@id]
  films = SqlRunner.run(sql, values)
  return films.map {|result| Film.new(result)}
end
```

Image 6: Result of the function running

```
ccc=# SELECT films.* FROM films INNER JOIN tickets ON films.id = tickets.film_id WHERE tickets.customer_id = 12;
 id |       film_title       | film_price
----+------------------------+------------
 10 | Avengers: Infinity War |         10
(1 row)
```

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:

Paste Screenshot here

Image 7: Function that sorts data

```ruby
def sort()
  sql = "SELECT * FROM stars ORDER BY last_name
  DESC"
  values = [@id]
  stars = SqlRunner.run(sql, values)
  return stars.map {|star| Star.new(star)}
end
```

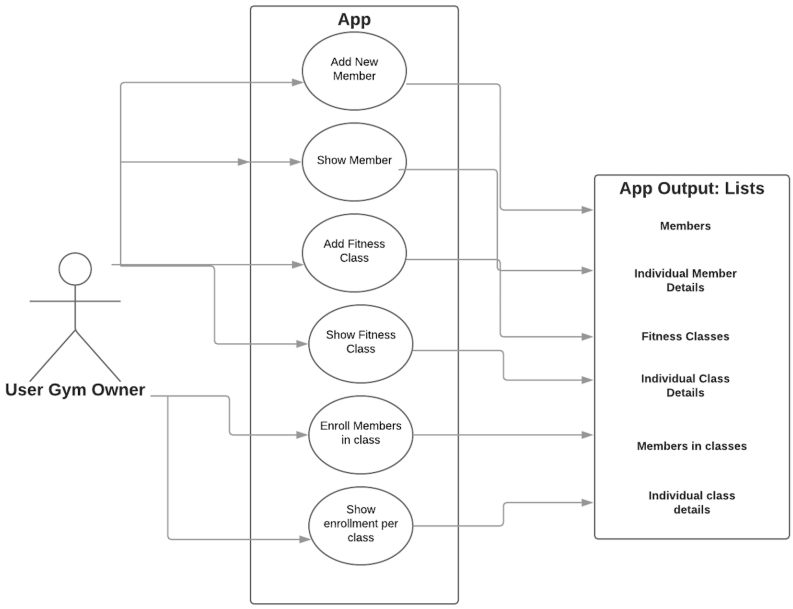
Image 8: Result of the function running

```
imdb=# SELECT * FROM stars ORDER BY last_name DESC;
 id | first_name | last_name
----+------------+-----------
 61 | John       | Wayne
 63 | Randolph   | Scott
 62 | Clint      | Eastwood
(3 rows)

imdb=# SELECT * FROM stars ORDER BY last_name DESC;
 id | first_name | last_name
----+------------+-----------
 61 | John       | Wayne
 63 | Randolph   | Scott
 62 | Clint      | Eastwood
(3 rows)

imdb=# SELECT * FROM stars ORDER BY last_name;
 id | first_name | last_name
----+------------+-----------
 62 | Clint      | Eastwood
 63 | Randolph   | Scott
 61 | John       | Wayne
(3 rows)
```

Week 5 and 6

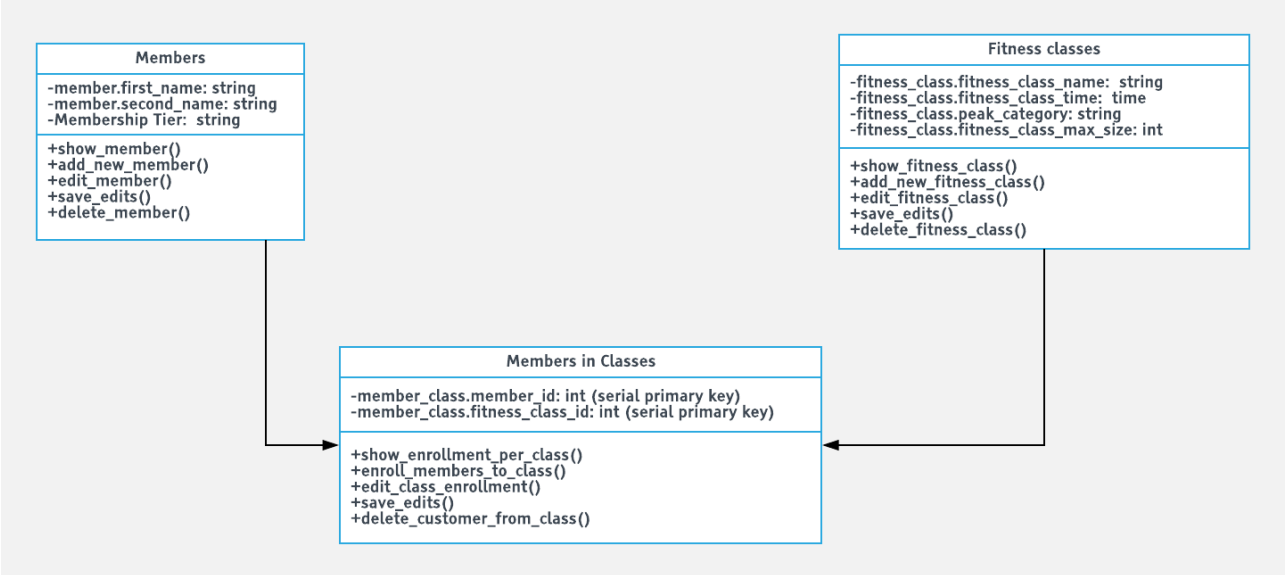
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram
		Description:

Paste Screenshot here



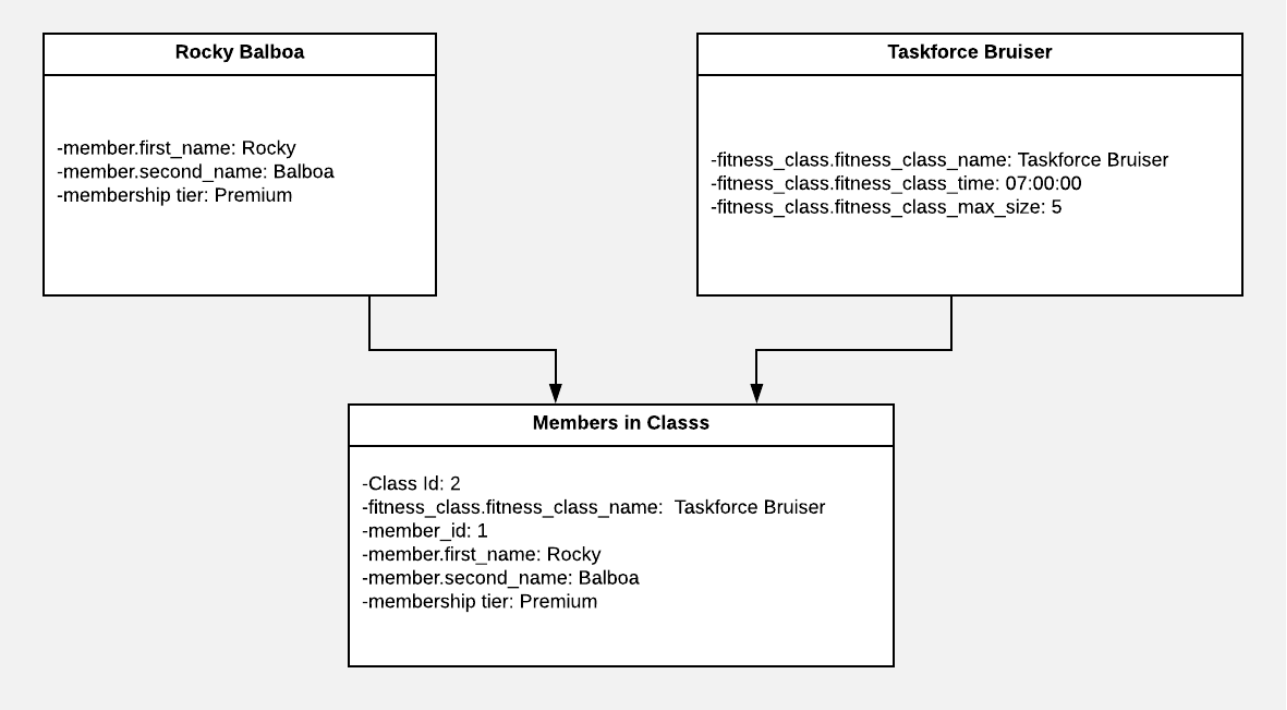
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		Description:

Paste Screenshot here



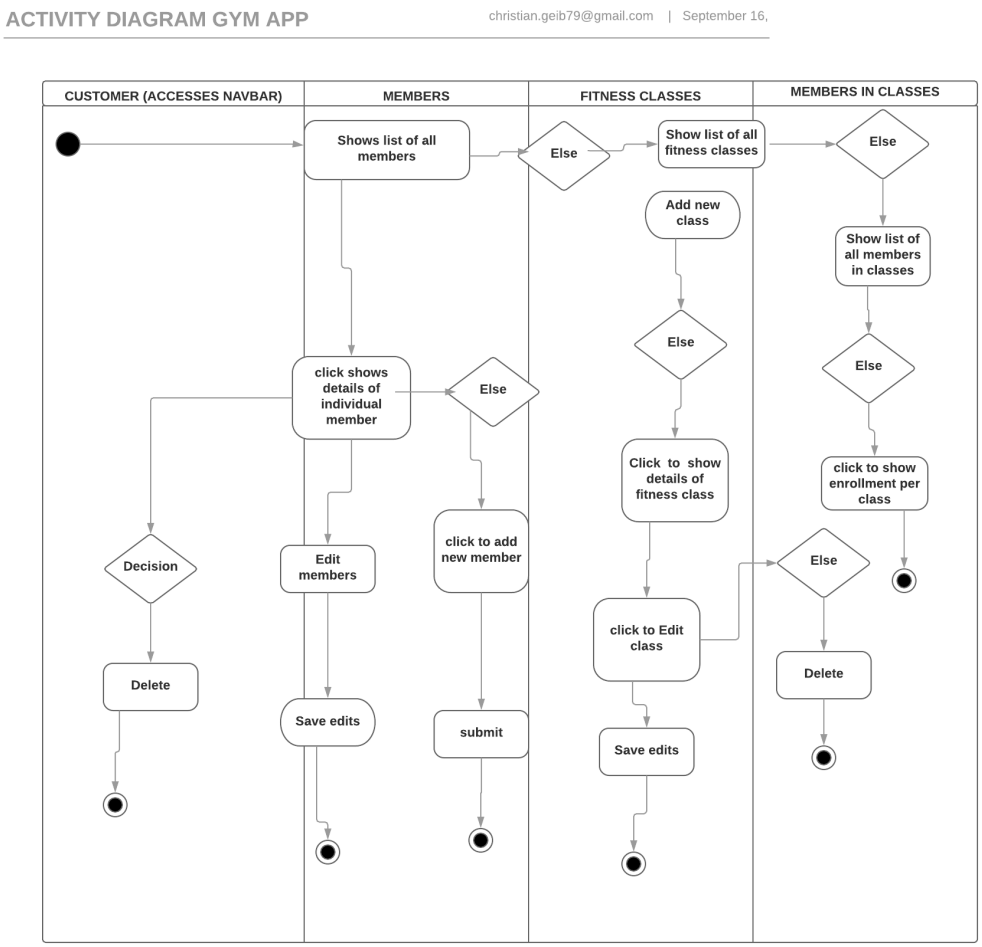
Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram
		Description:

Paste Screenshot here



Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram
		Description:

Paste Screenshot here



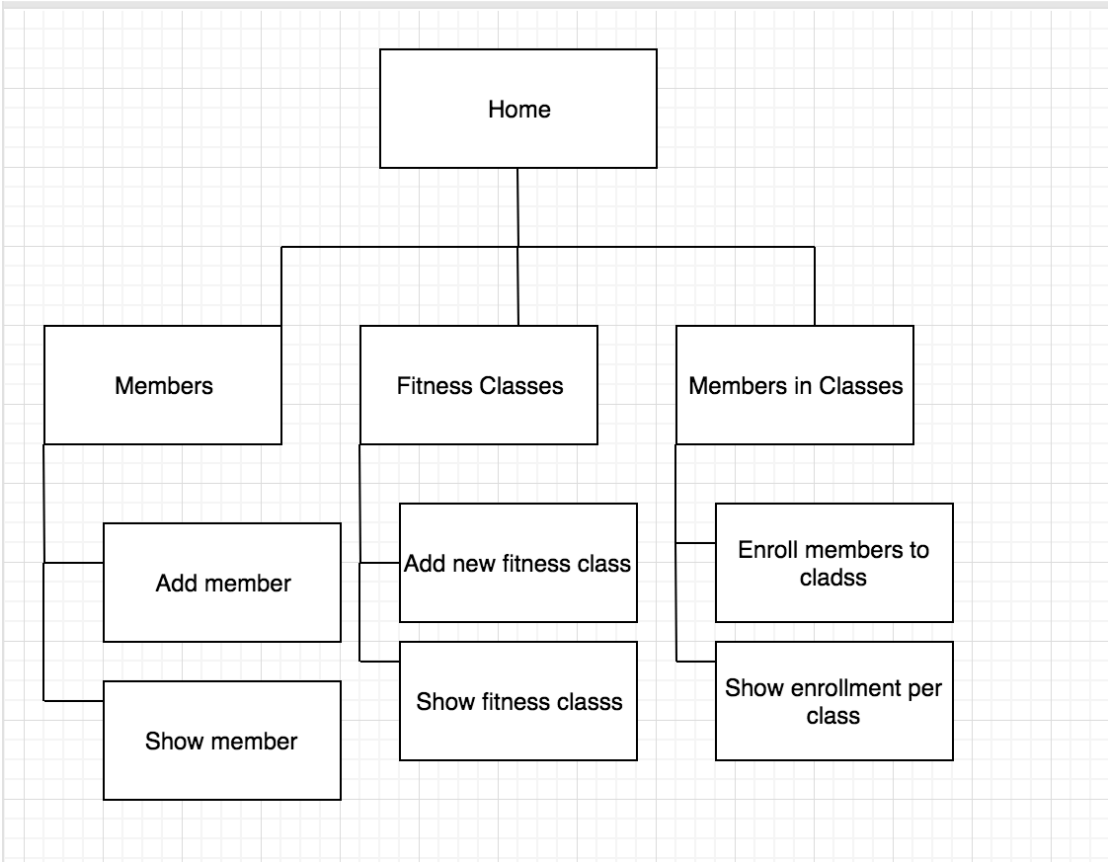
Unit	Ref	Evidence
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time
		Description:

Topic	Possible Effect of Constraint of Product	Solution
Hardware and software platforms	-system/programming language chosen may perform differently on different hardware/ software platforms	-select a programming language such as python that runs cross-platform on Mac and Windows alike
Performance requirements	-Latency/response time should be minimal for the tool to gain acceptance	-design code that it runs as few iterations as possible and try to use local storage instead of needing to connect to database for every query
Persistent storage and transactions	-choosing suboptimal data format and database system could create slow system performance and high costs for storage/server space	-chose data format that minimises data size and makes it as interoperation as possible
Usability	Complicated UI/UX might lead to fallacious input and or abandoned transactions	Carefully consider the user proto personas and plan the user journey in order to simplify the user journey/interaction as simple/seamless as possible
Budgets	-While some software solutions may be preferable from a technical perspective, they may be too expensive (e.g. licensing costs) -choosing suboptimal data format can lead to unnecessarily high costs	-if possible choose open source solutions
Time	Certain extensions/“nice to have” features might take too much time to implement within the time allocated to a project	Implement a MOSCOW structure and prioritise accordingly

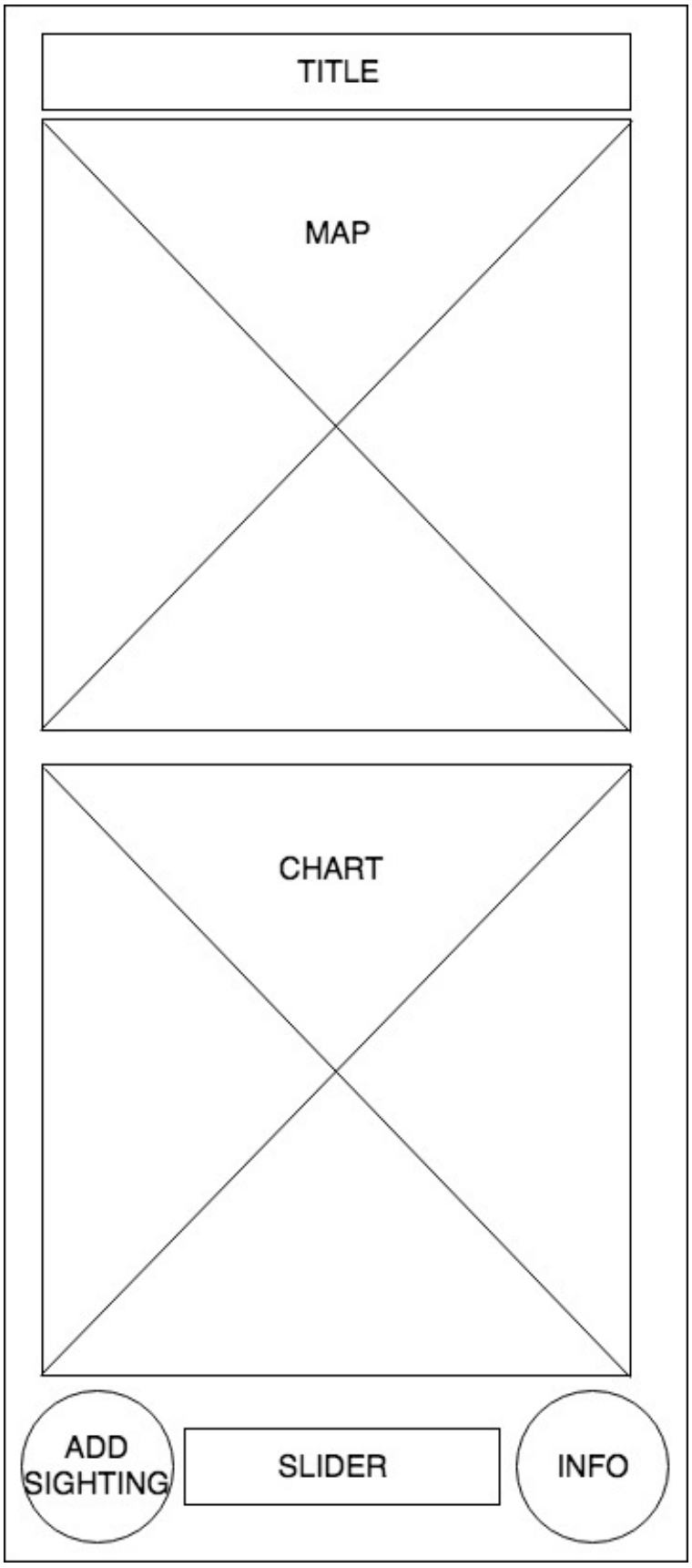
Paste Screenshot here

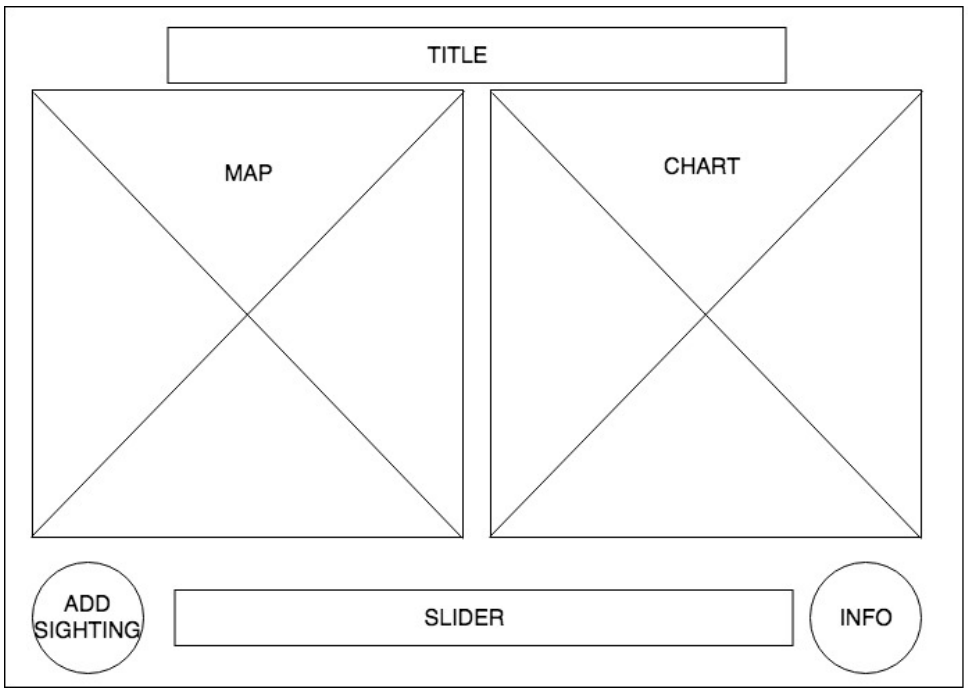
Unit	Ref	Evidence
P	P.5	User Site Map
		Description:

Paste Screenshot here



Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams
		Description:





CHART

Paste Screenshot here

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

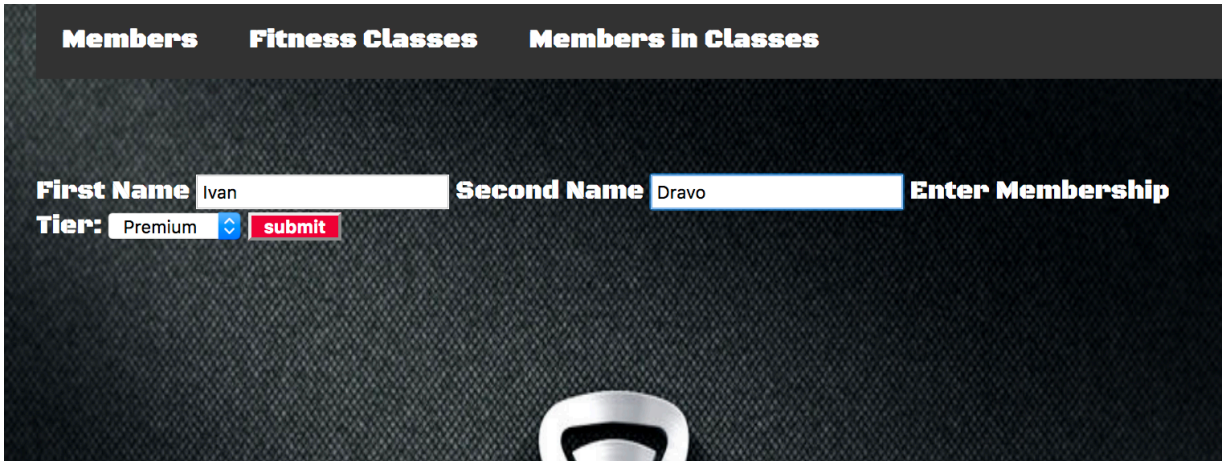
show.e... | new_m... | show_... | new_fit... | new.erb | create_... | create_... | create.... | edit_fit... | edit_m... | edit.erb | seeds.rb

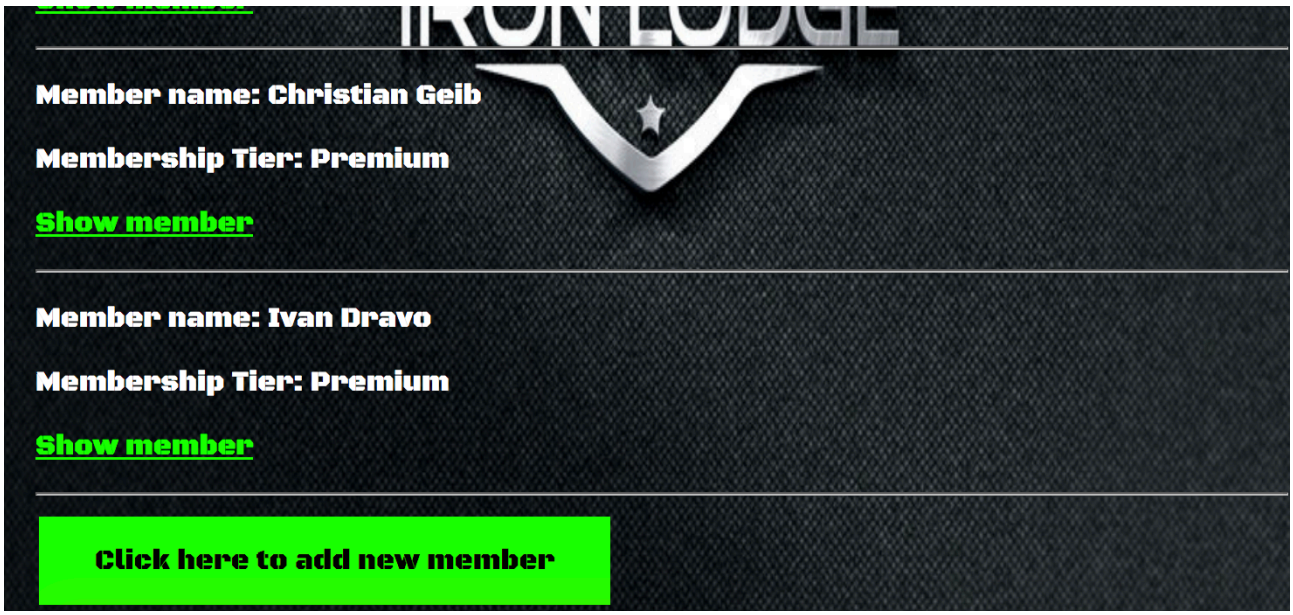
```
13  <select id="fitness-class-id" name="fitness_class_id">
14    <% for fitness_class in @classes %>
15        <option value= "<%=fitness_class.id%>"><%=fitness_class.fitness_class_name%></option>
16    <%end%>
17  </select>
18
19  <input type="submit"  value="submit">
20
21  </form>
22  </show>
23
24
25  #Pseudocode for adding member into fitness class, ensuring that maximum class-size not exceeded
26
27  #IF
28  #-the adding one member to the fitness class results in a class size larger than the maximum size
    of the class return string "This would exceed the maximum number of the class of #{}. Please add
    member to different class"
29
30  #ELSE
31
32  #-add member_id to list of member_ids already in the class
33
34  #-save the change
35
36  #-display the change
37
```

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:

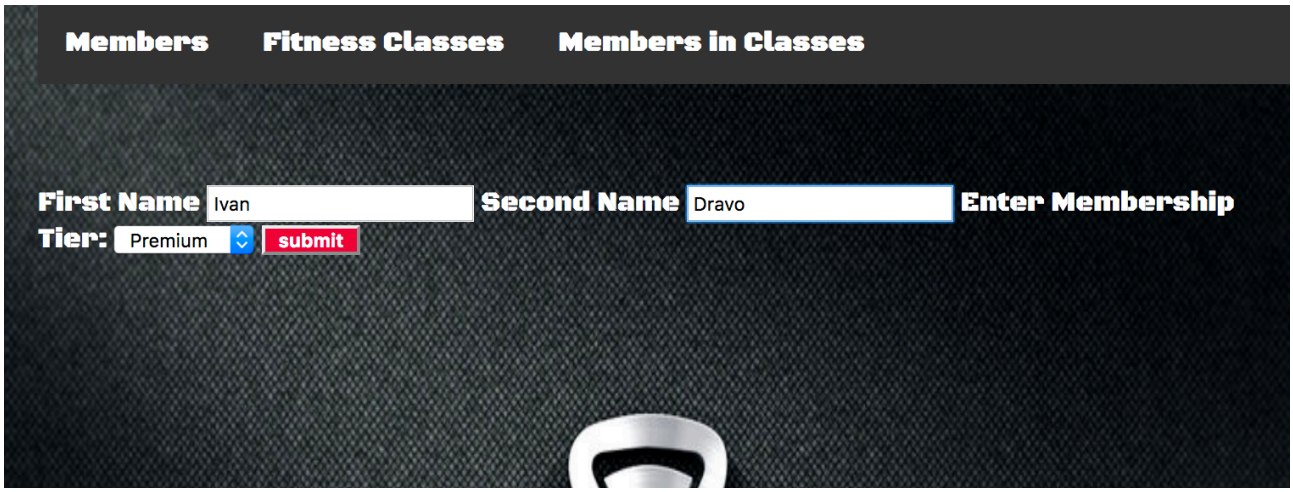
Paste Screenshot here

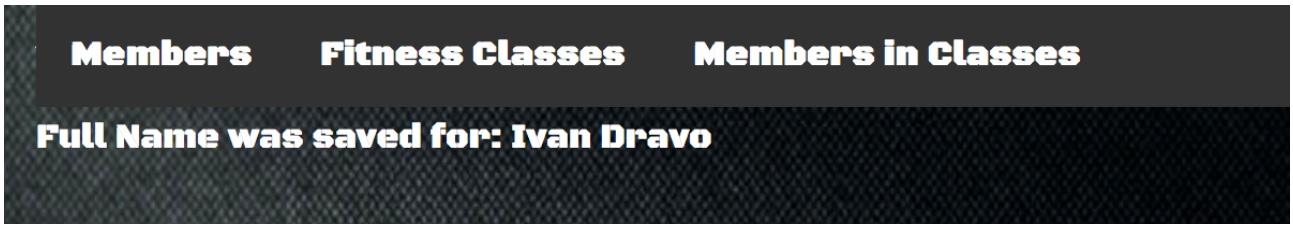




Unit	Ref	Evidence
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved
		Description:

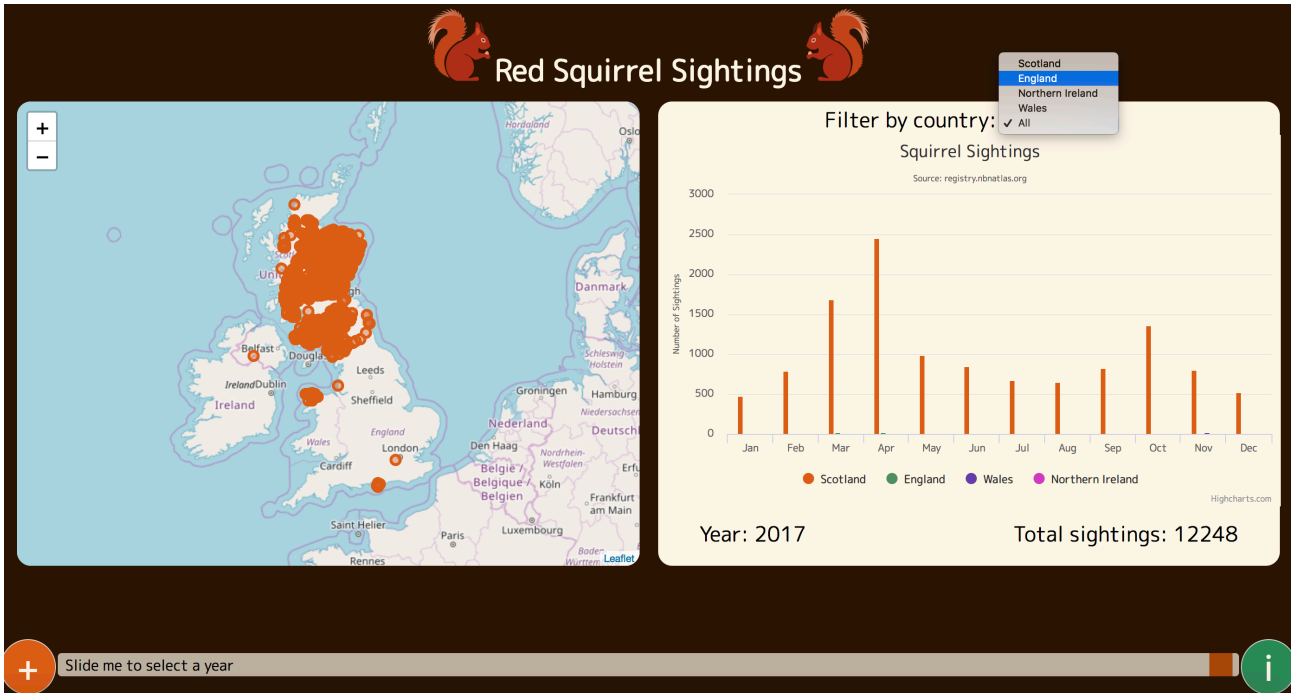
Paste Screenshot here

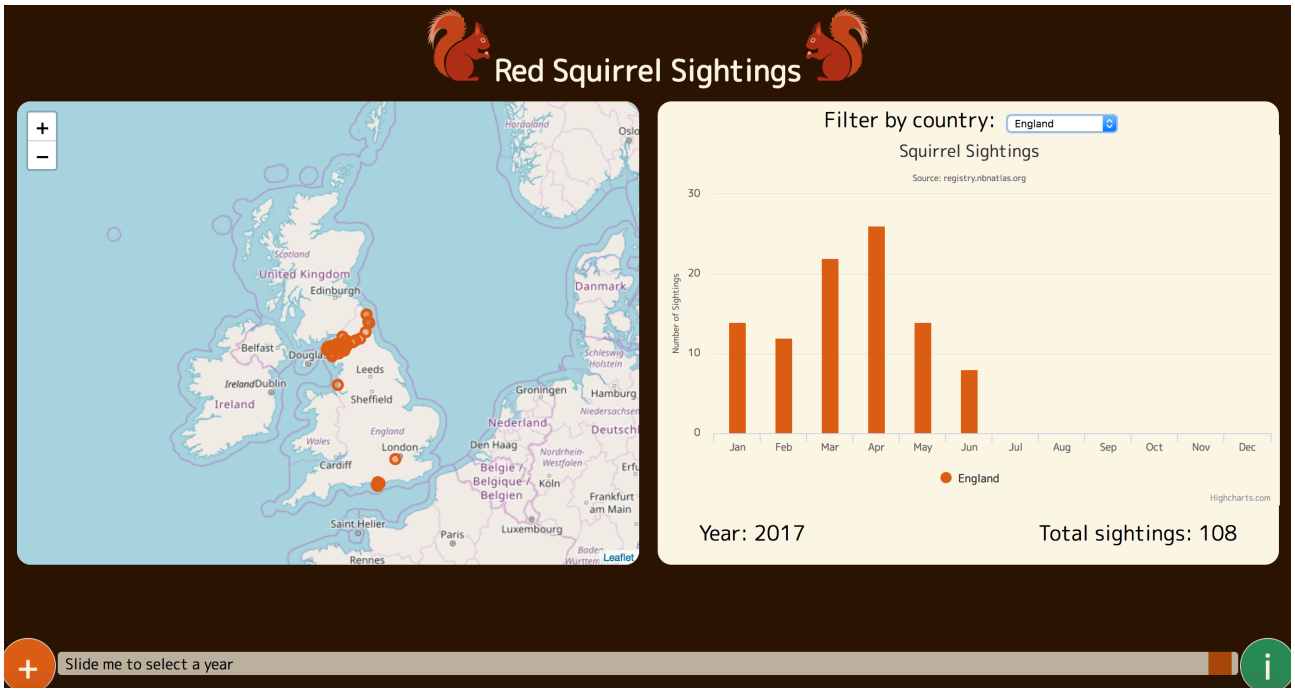




Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program
		Description:

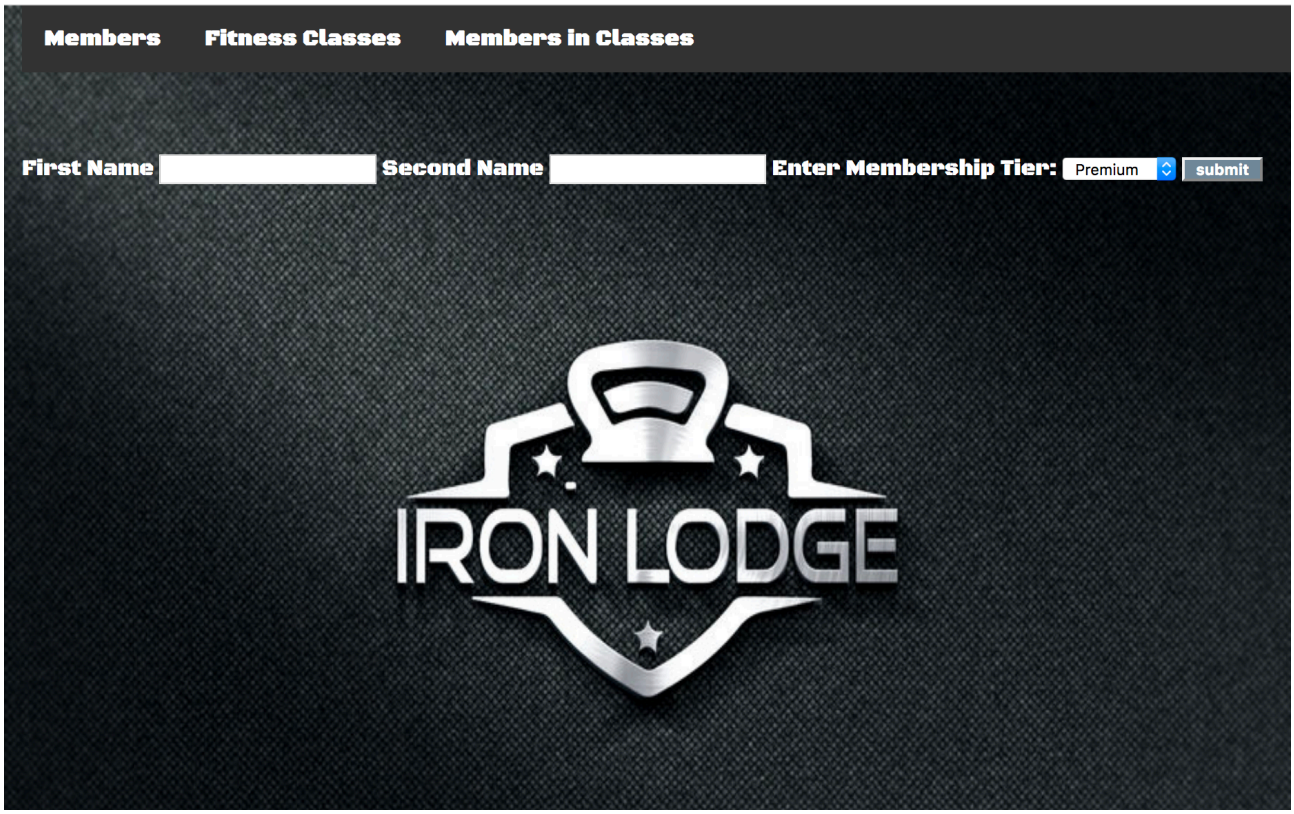
Paste Screenshot here


Red Squirrel Sightings
Scotland
England
Northern Ireland
Wales
All
Filter by country:
Squirrel Sightings
Source: registry.nbnatlas.org
3000
2500
2000
1500
1000
500
0
Number of Sightings
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Scotland
England
Wales
Northern Ireland
Highcharts.com
Year: 2017
Total sightings: 12248
Slide me to select a year


Red Squirrel Sightings
Filter by country:
England
Squirrel Sightings
Source: registry.nbnatlas.org
30
20
10
0
Number of Sightings
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
England
Highcharts.com
Year: 2017
Total sightings: 108
Slide me to select a year

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: This is a screenshot of the individual project in which I developed an app for a gym

Paste Screenshot here

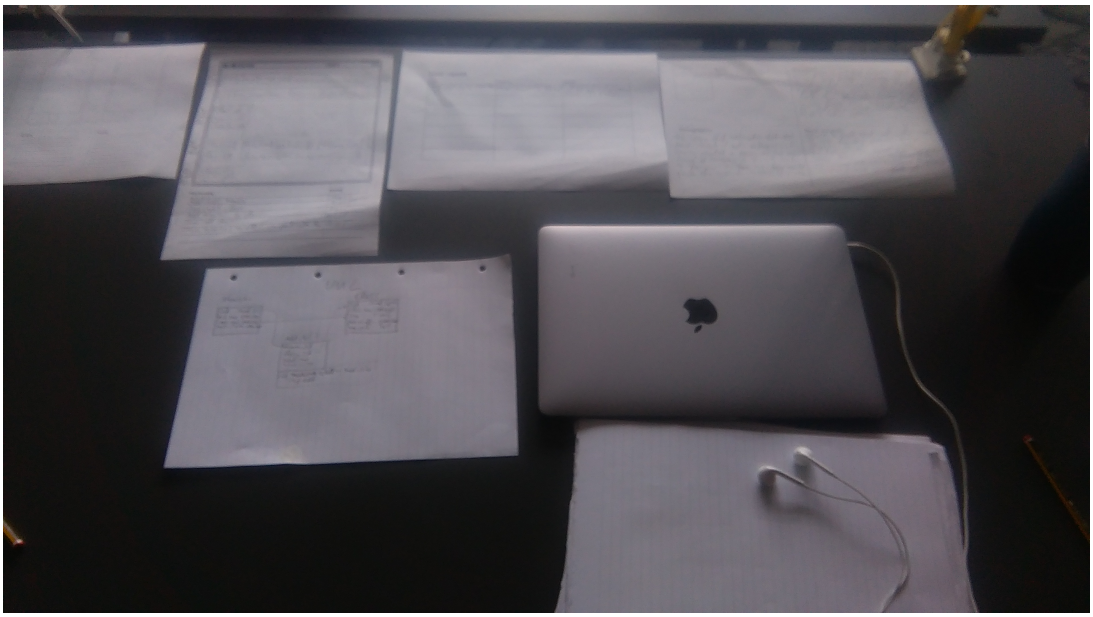


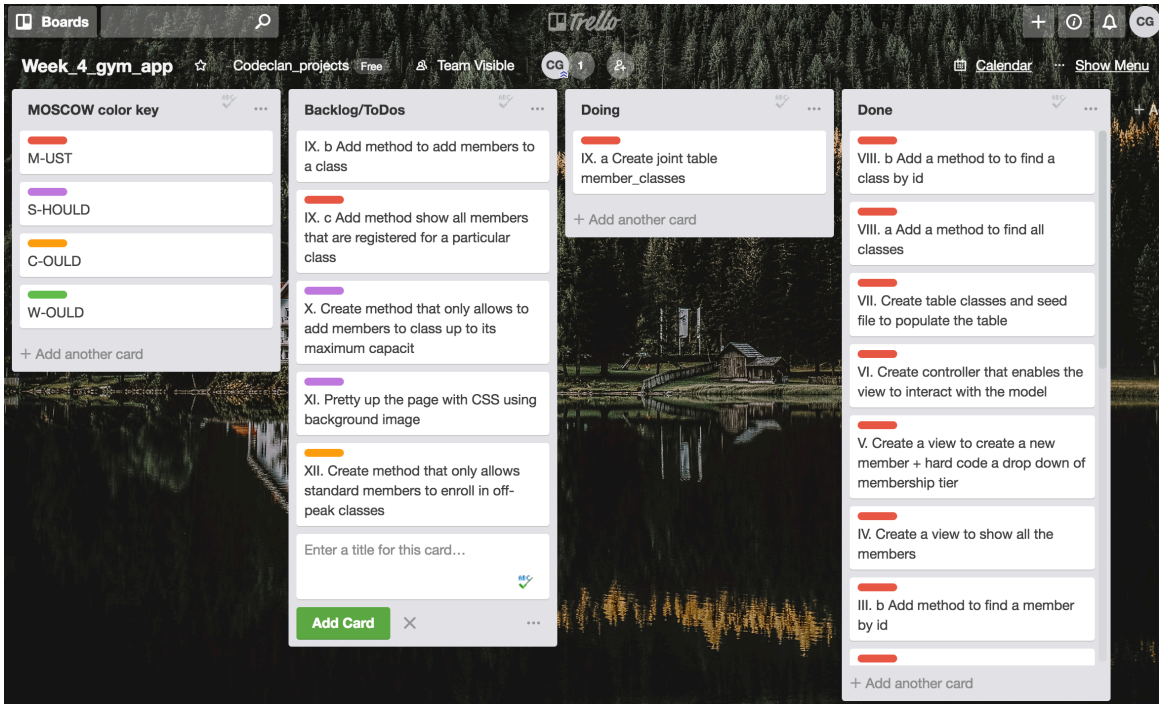
Github link:

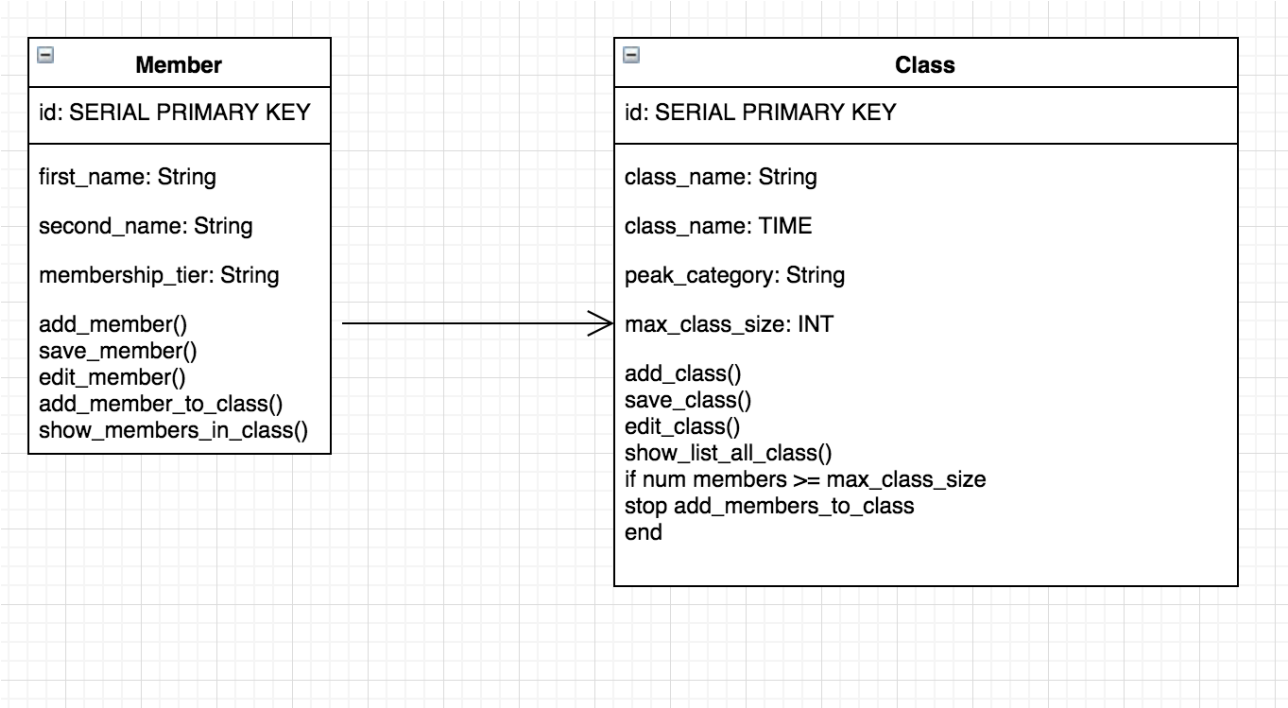
https://github.com/cgeib79/project_week_4_gym_app

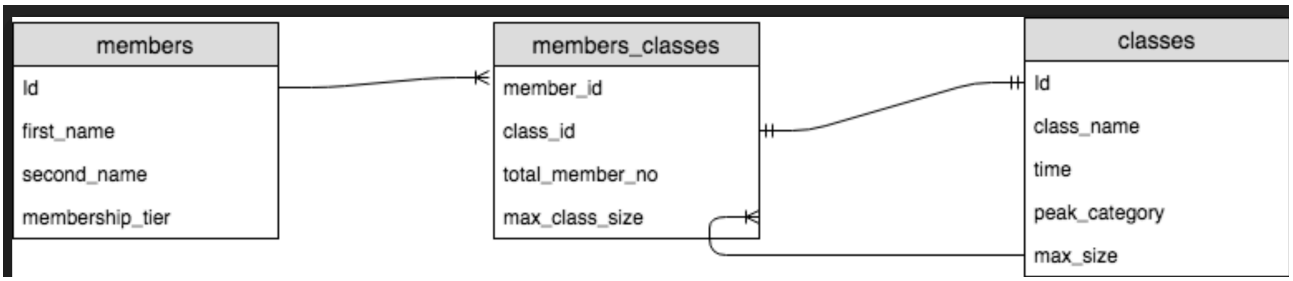
Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.
		Description:

Paste Screenshot here






Member
id: SERIAL PRIMARY KEY
first_name: String
second_name: String
membership_tier: String
add_member()
save_member()
edit_member()
add_member_to_class()
show_members_in_class()
Class
id: SERIAL PRIMARY KEY
class_name: String
class_name: TIME
peak_category: String
max_class_size: INT
add_class()
save_class()
edit_class()
show_list_all_class()
if num members >= max_class_size
stop add_members_to_class
end

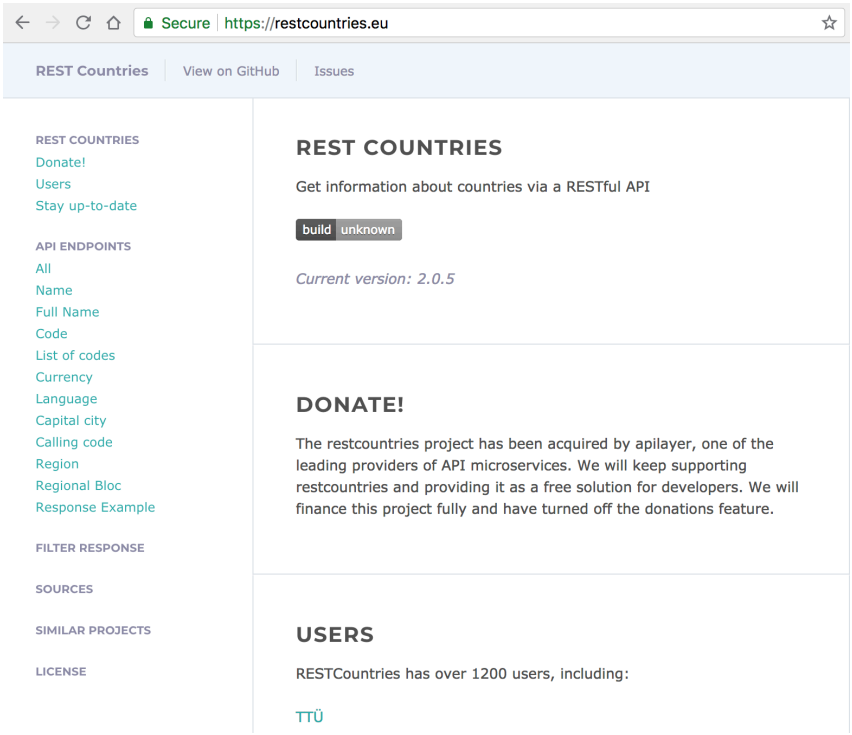

members
Id
first_name
second_name
membership_tier
members_classes
member_id
class_id
total_member_no
max_class_size
classes
Id
class_name
time
peak_category
max_size

Week 7

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: The following screenshots show the code for making requests to the API rest countries.eu

Paste Screenshot here

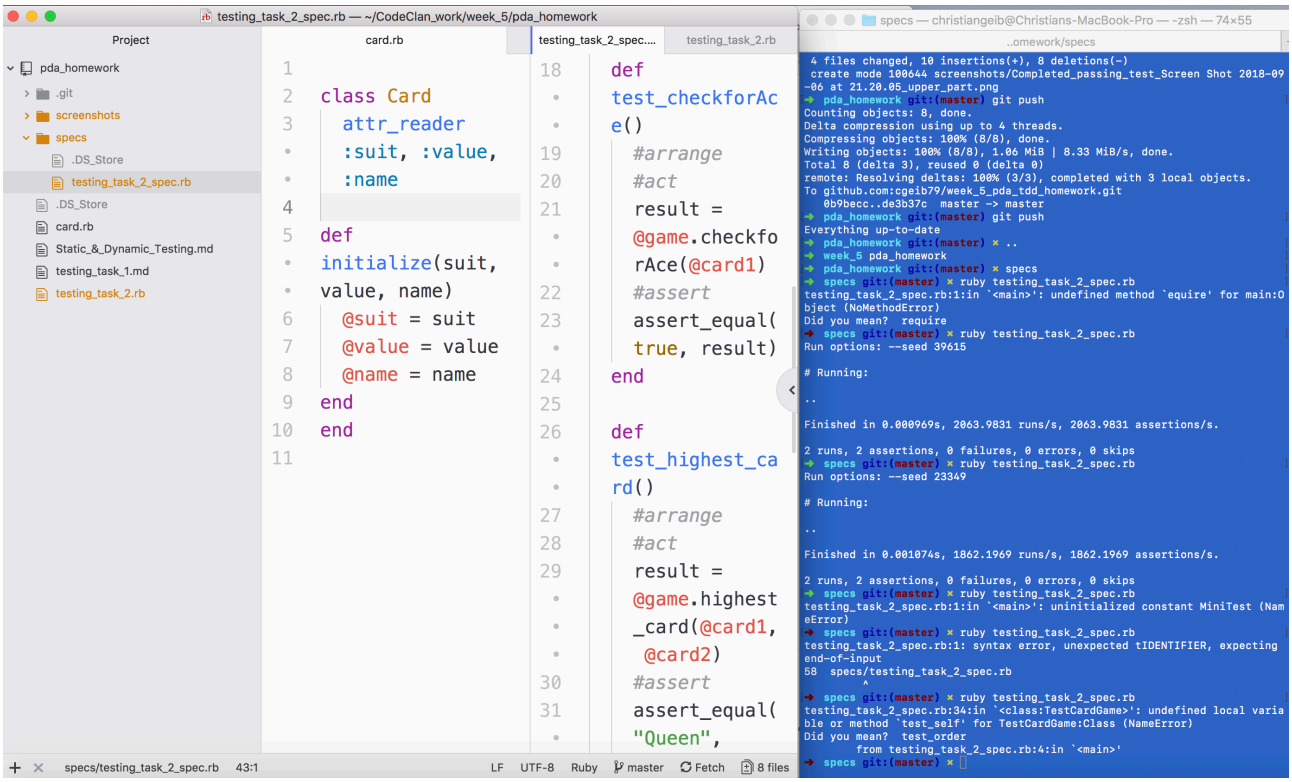
https://restcountries.eu/



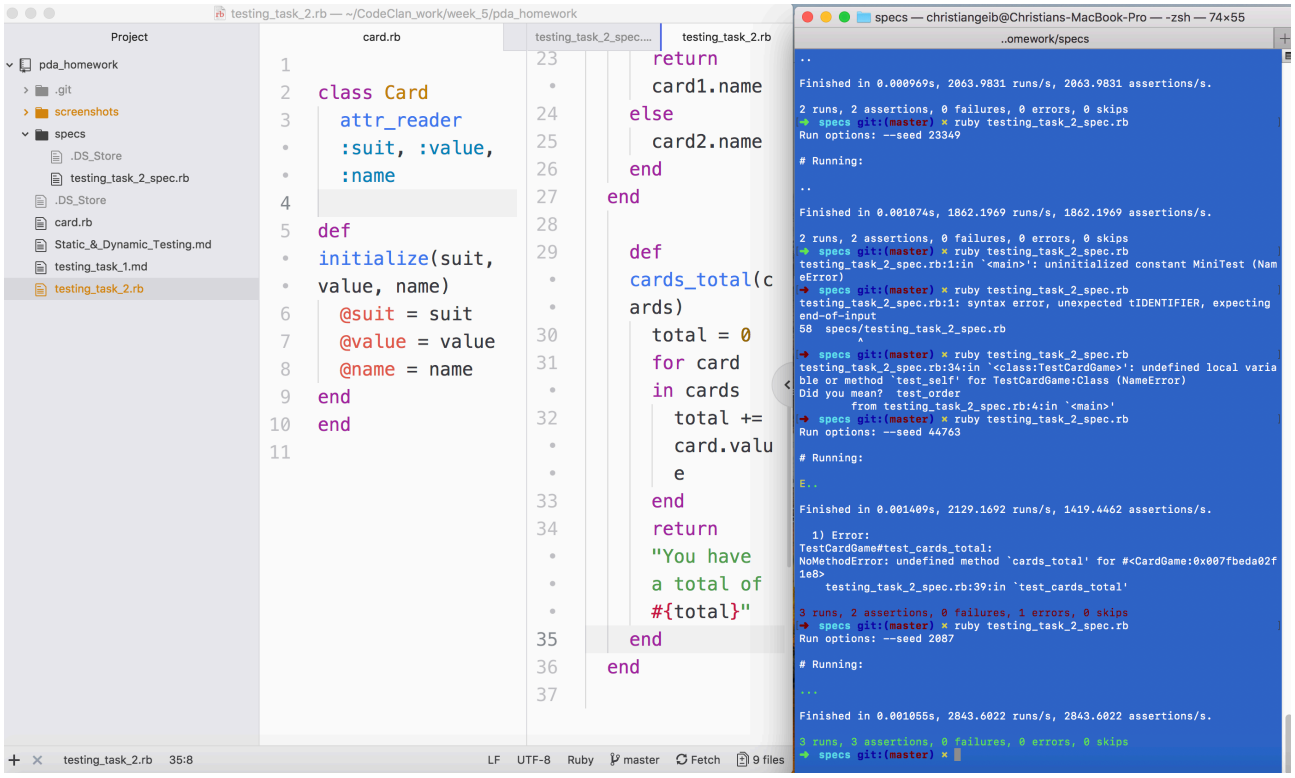
Unit	Ref	Evidence
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing
		Description: Descriptions below

Paste Screenshot here

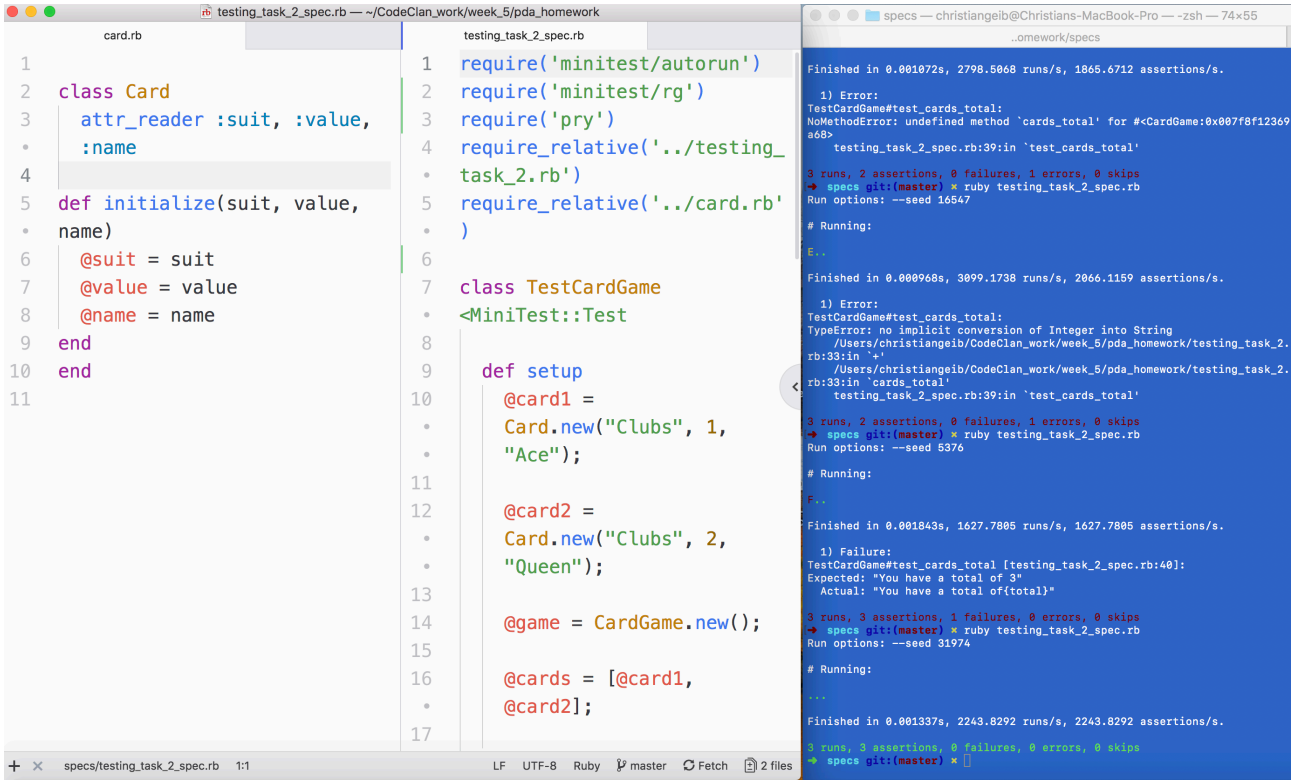
Screenshot: 5th iteration - name error



Screenshot: final fix in different file than testing file that made the test pass



Screenshot: final completed, passing test with previously failing tests

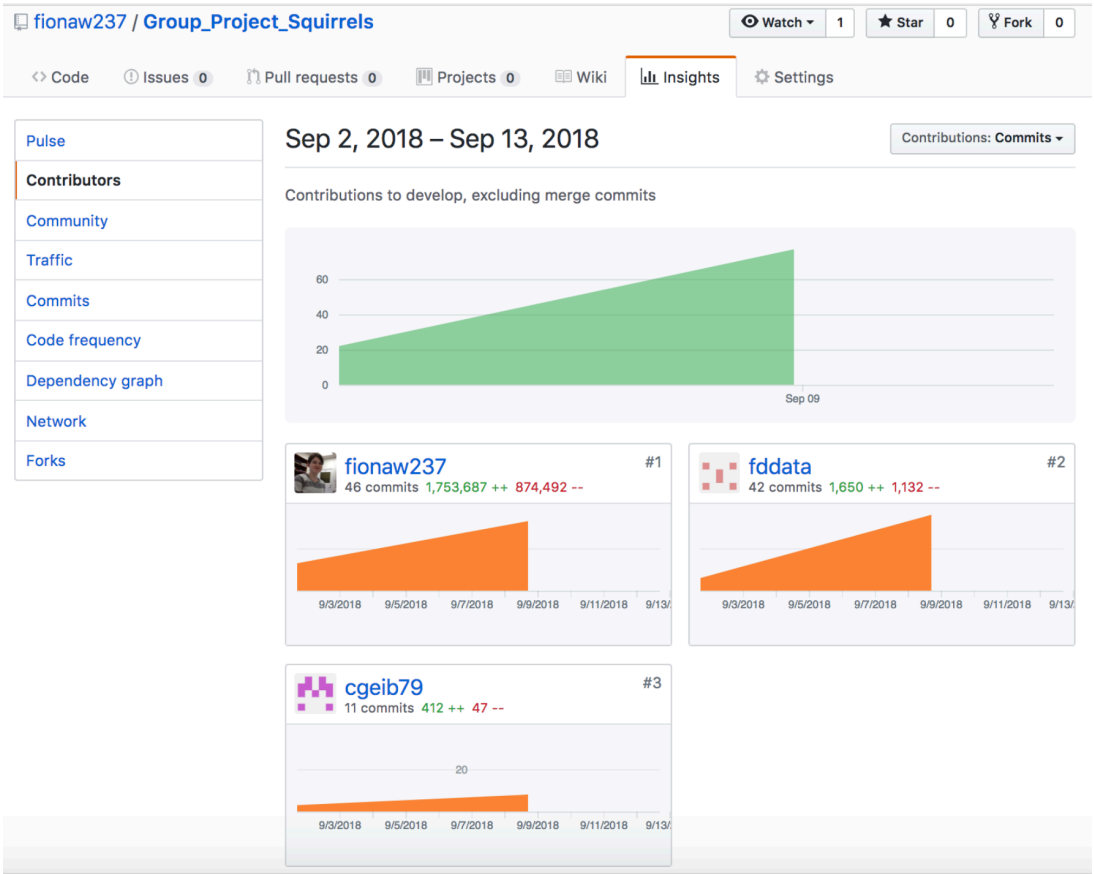


Week 9

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: I was part of a team of three developing an app for squirrel sightings. Github contributors page below.

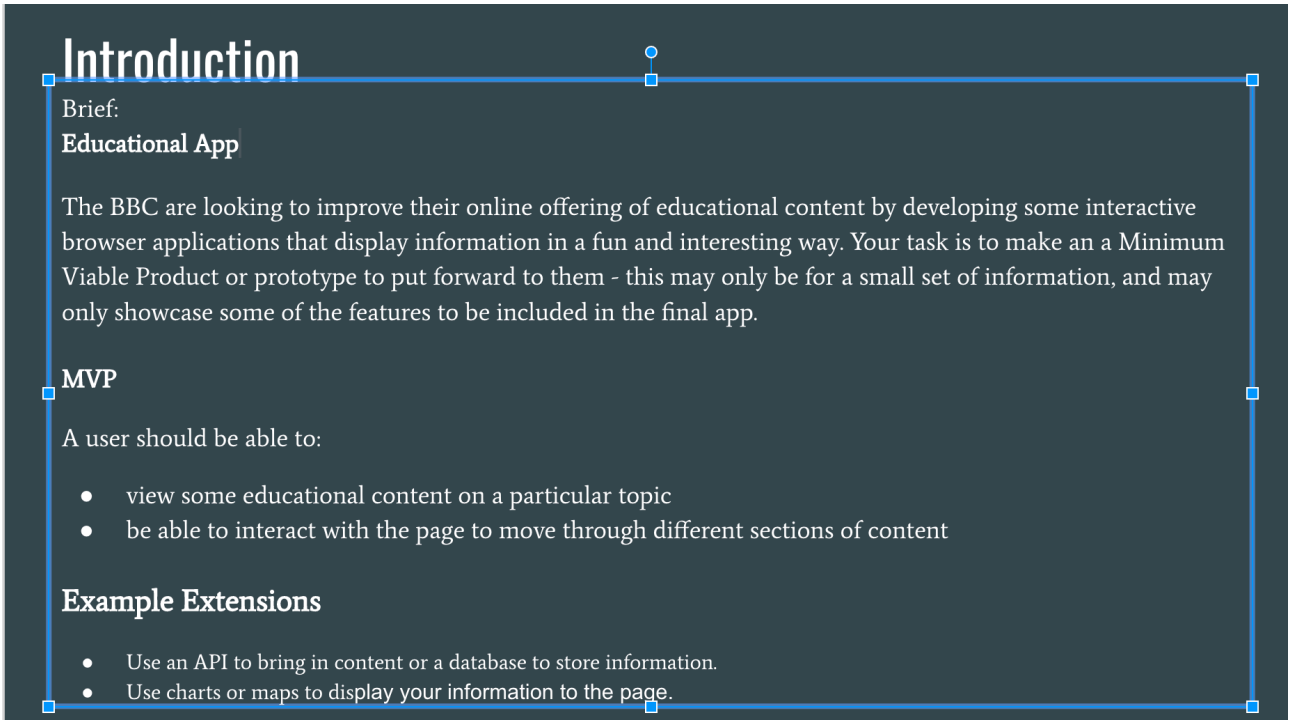
Paste Screenshot here

https://github.com/cgeib79/Group_Project_Squirrels



Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.
		Description: This is a screenshot of the project brief for the aforementioned squirrel app

Paste Screenshot here



Introduction

Brief:

Educational App

The BBC are looking to improve their online offering of educational content by developing some interactive browser applications that display information in a fun and interesting way. Your task is to make an a Minimum Viable Product or prototype to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app.

MVP

A user should be able to:

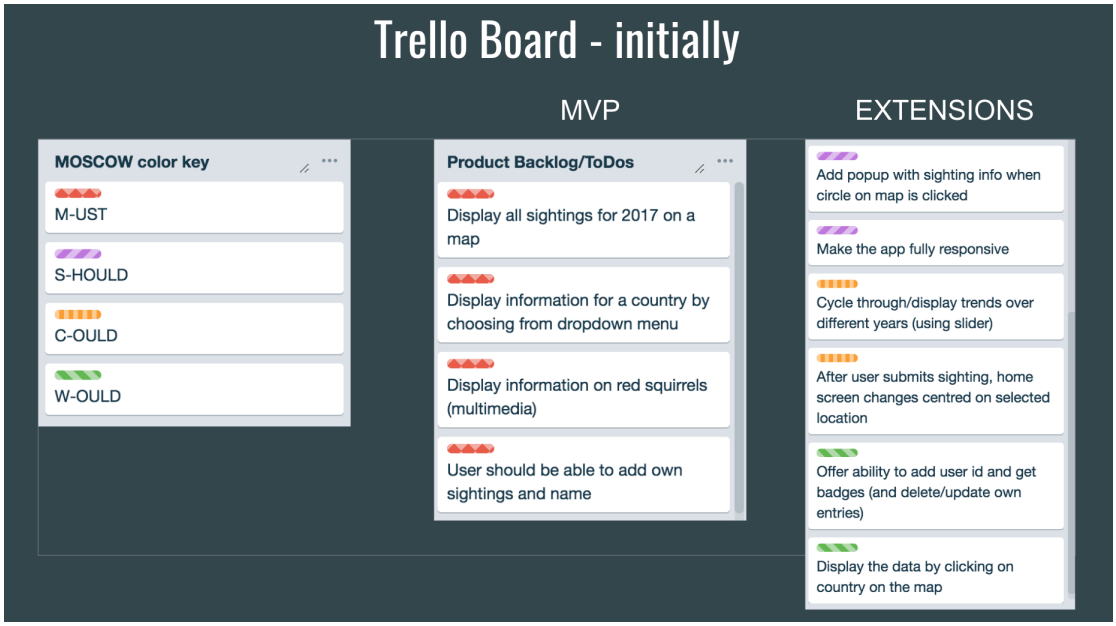
- view some educational content on a particular topic
- be able to interact with the page to move through different sections of content

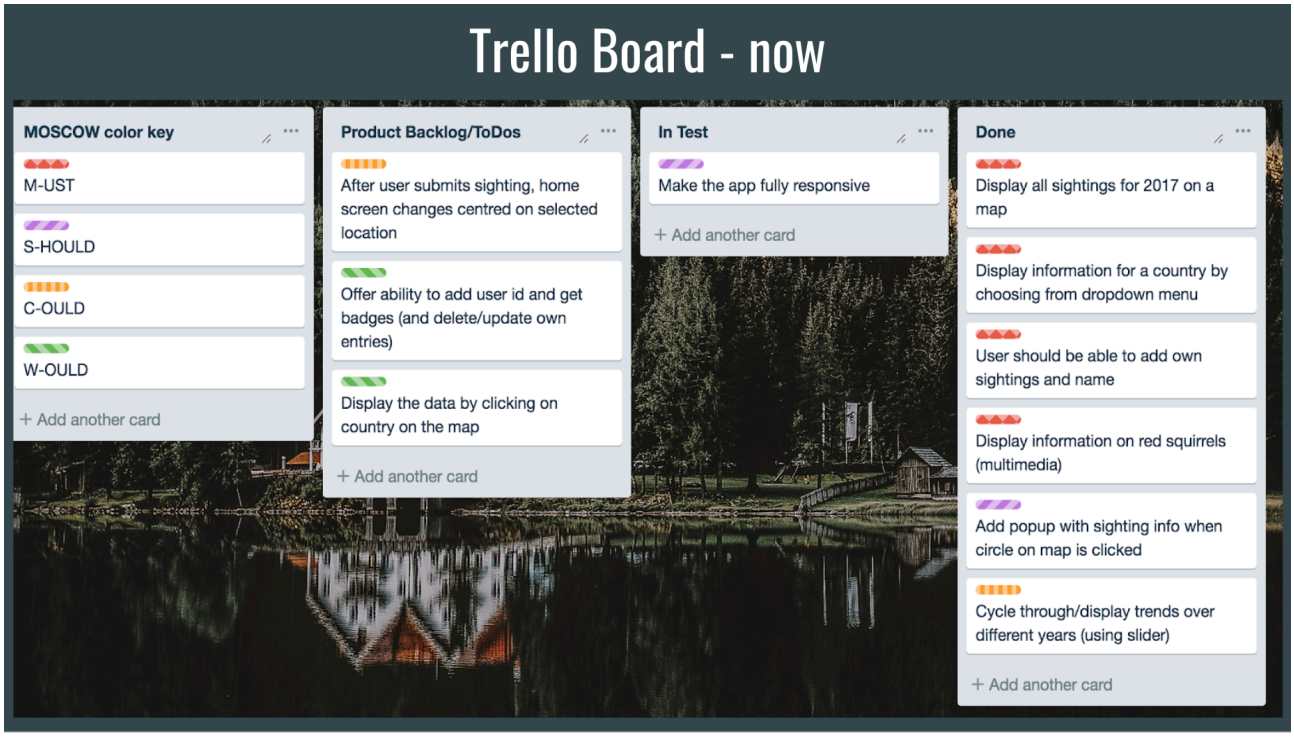
Example Extensions

- Use an API to bring in content or a database to store information.
- Use charts or maps to display your information to the page.

Unit	Ref	Evidence
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.
		Description: Screenshots of Trello board for the aforementioned Squirrel App

Paste Screenshot here





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

Paste Screenshot here

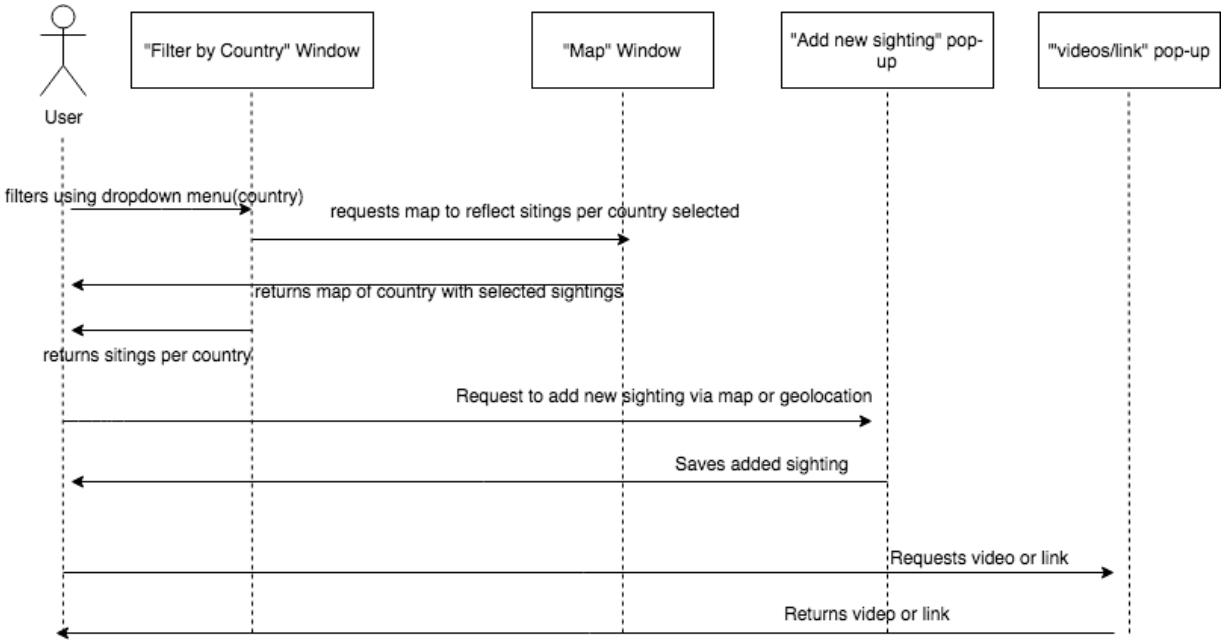
Acceptance Criteria/Acceptance Test Plan

Acceptance Criteria	Expected Result/Output	Pass/Fail
User is able to filter chart diagram by country	Charts change when dropdown menu button selected	Pass
Map synchronising with the chart	If, e.g. England selected on dropdown menu in the chart window, map on the left only display England sightings	Pass
Right ‘i’ button click opens video/ link pop-up	If right “i” button clicked, video/ link pop-up opens	Pass
Left “+” button click opens pop-up	If left “+” button clicked, opens the “add sighting” pop-up	Pass
User can click on map in “add sighting” pop-up	User can click on map in add sighting” pop-up and save that location	Pass
In “add sighting” pop-up user can click on ‘locate me’ button	User clicks on “locate me” button, position on map is returned and saved by clicking the “save” button	Pass

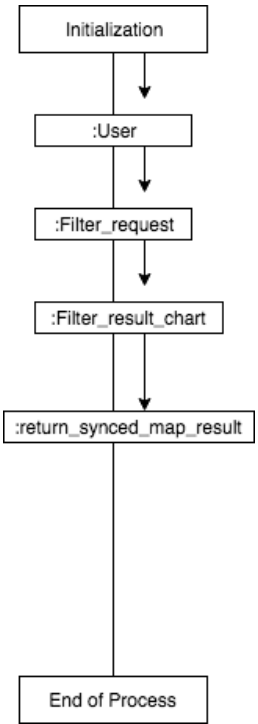
Unit	Ref	Evidence
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).
		Description: Below screenshots of sequence and collaboration diagrams of aforementioned squirrel app

Paste Screenshot here

Sequence Diagram

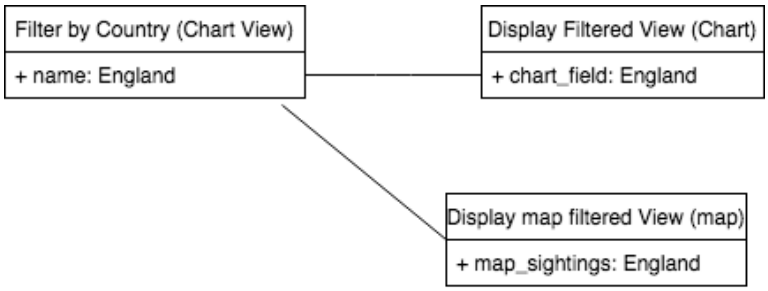


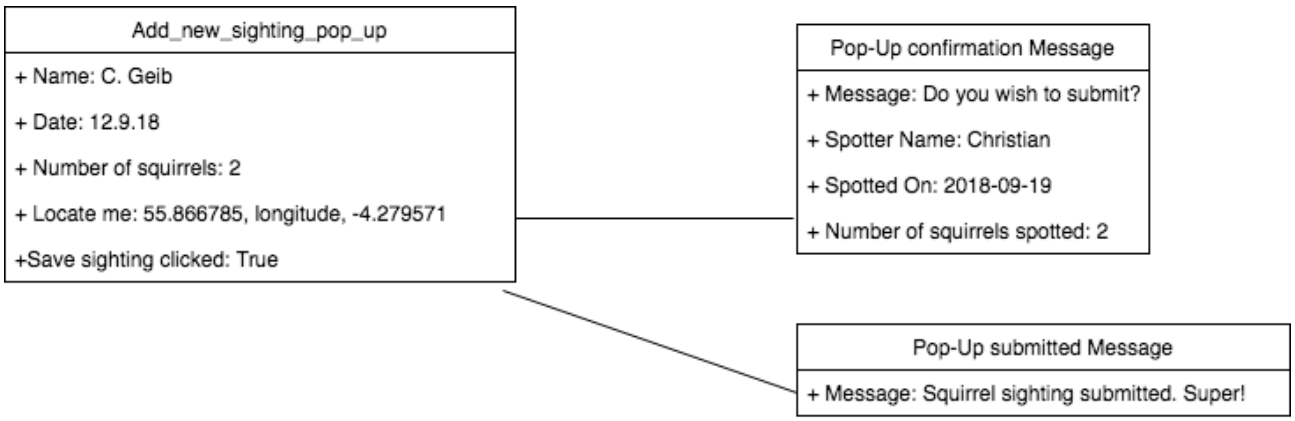
Collaboration Diagram



Unit	Ref	Evidence
P	P.8	Produce two object diagrams.
		Description:

Paste Screenshot here





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

Paste Screenshot here

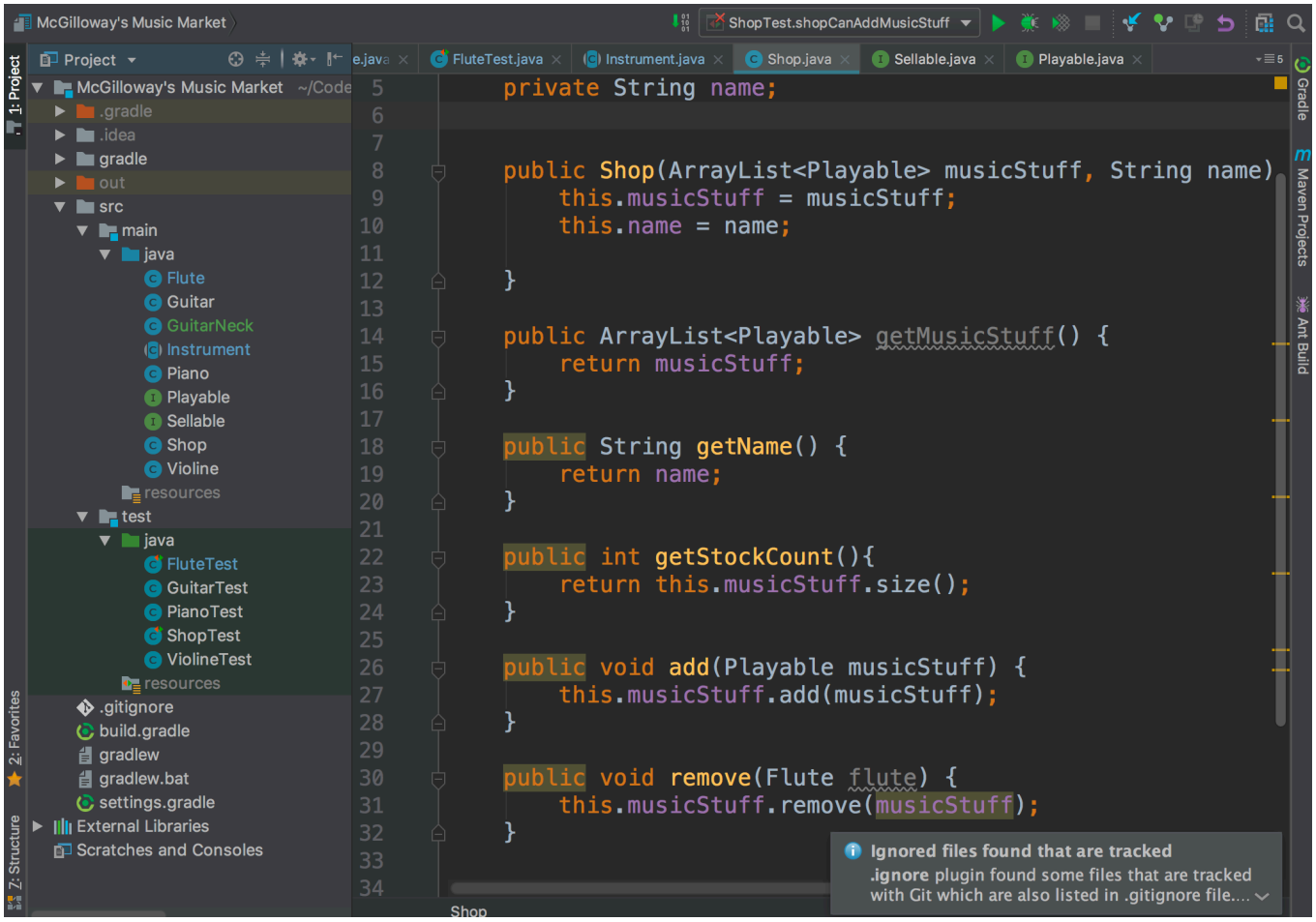
Requirements	Pass/Failed	Solution	Pass Failed
Add data from JSON file to database	Failed - problem with "insertMany" command in MongoDB	Used 'mongoimport' command in terminal	Pass
If a country is chosen from dropdown then the slider is used, all the data for the new year is shown instead of the data for the chosen country	Failed	Call getPlottingData function when either a country is chosen from the dropdown, or a new year is chosen from the slider	Pass
Map should render properly in pop up form	Failed - known problem with Leaflet	Use 'invalidatesize' method	Pass
Links on information page should change colour according to chosen format	Failed - problem with CSS	Currently unresolved	
Locate me button should find user's current location	Failed	Identify properly what 'this' refers to within each block	Pass

Week 12

Unit	Ref	Evidence
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.
		Description: Screenshots below show inheritance model in Java of a music shop. The Shop class contains an ArrayList of playable items of the type Playable. Playable is an interface which is implemented by every instrument present at this shop. The various instruments extend the public abstract class Instrument and thereby form subclasses of Instrument which is an example of polymorphism. The abstract class Instrument itself implements the interface Sellable.

e.g. having musical instruments with interfaces (playable, sellable) in the array list of stock

Paste Screenshot here



Shop class

```java
import java.util.ArrayList;

public class Shop {
    private ArrayList<Playable> musicStuff;
    private String name;


    public Shop(ArrayList<Playable> musicStuff, String name) {
        this.musicStuff = musicStuff;
        this.name = name;

    }

    public ArrayList<Playable> getMusicStuff() {
        return musicStuff;
    }

    public String getName() {
        return name;
    }

    public int getStockCount(){
        return this.musicStuff.size();
    }

    public void add(Playable musicStuff) {
        this.musicStuff.add(musicStuff);
    }
```

```java
public abstract class Instrument implements Sellable {
    private String modelName;
    private String color;
    private String material;
    private String body;
    private String keyboard;
    private double buyingPrice;
    private double sellingPrice;


    public Instrument(String modelName, String color, String material, String body, String ke
        this.modelName = modelName;
        this.color = color;
        this.material = material;
        this.body = body;
        this.keyboard = keyboard;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getModelName() {
        return modelName;
    }

    public String getColor() {
        return color;
    }
```

Public abstract class instrument implements interface Sellable
Public class Guitar extends public abstract class Instrument and extends interfaces Playable, Sellable

```
public class Guitar extends Instrument implements  Playable, Sellable  {

    GuitarNeck guitarNeck;

    public Guitar(String modelName, String color, String material, String body, String keyboa
        super(modelName, color, material, body, keyboard, buyingPrice, sellingPrice);
        this.guitarNeck = guitarNeck;
    }

    public GuitarNeck getGuitarNeck() {
        return guitarNeck;
    }

    public String play() {
        return "bing";
    }


    public double markUp() {
        return markUp();
    }
}
```

Public class Flute extends abstract class Instrument and implements Interface Playable

```
import com.sun.xml.internal.ws.api.pipe.Tube;

public class Flute extends Instrument implements Playable {

    Tube tube;


    public Flute(String modelName, String color, String material, String body, String keyboar
        super(modelName, color, material, body, keyboard, buyingPrice, sellingPrice);
        this.tube = tube;
    }

    public Tube getTube() {
        return tube;
    }


    public String play() {
        return "bing";
    }

}
```

Public Interface Playable

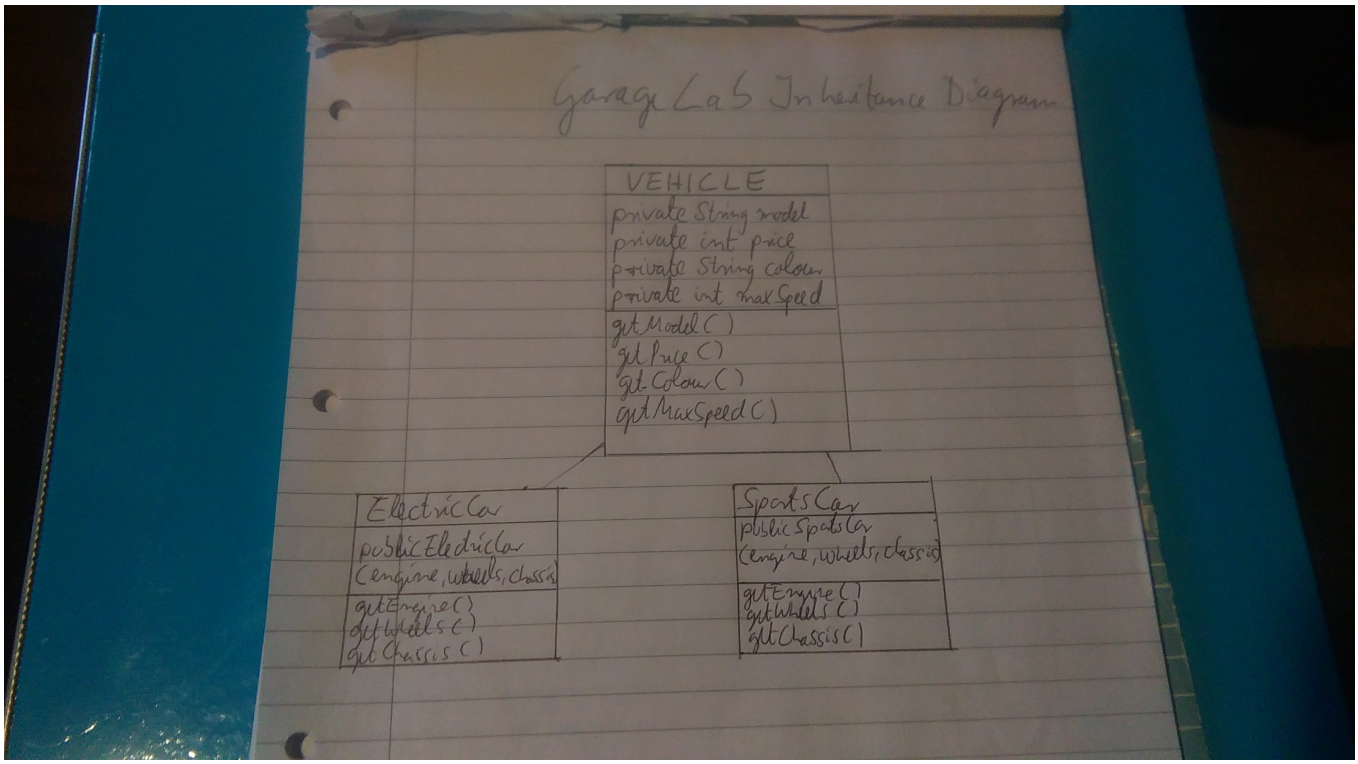
```
public interface Playable {
    String play();
}
```

Public Interface Sellable

```
public interface Sellable {
    double markUp ();
}
```

Unit	Ref	Evidence
A&D	A.D.5	An Inheritance Diagram
		Description: This inheritance diagram shows the two subclasses ElectricCar and SportsCar inheriting from the abstract class Vehicle

<u>Paste Screenshot here</u>



Unit	Ref	Evidence
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.
		Description: Encapsulation in Java means the wrapping data (variables) and code acting on data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. It is therefore also known as data hiding. **In order to achieve encapsulation in Java a) the variables of a class need to be declared as private and b) provide public setter and getter methods to modify and view the variables values**

Paste Screenshot here

Making a class private

```
public abstract class Instrument implements Sellable {
    private String modelName;
    private String color;
    private String material;
    private String body;
    private String keyboard;
    private double buyingPrice;
    private double sellingPrice;


    public Instrument(String modelName, String color, String material, String body, String k
        this.modelName = modelName;
        this.color = color;
        this.material = material;
        this.body = body;
        this.keyboard = keyboard;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getModelName() {
        return modelName;
    }

    public String getColor() {
        return color;
    }
```

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: As shown under Ref A.D.5 the screenshots below show the subclasses SportsCar and Electric vehicle inheriting from the abstract class Vehicle

Paste Screenshot here

```
package vehicles;

import behaviours.Sellabe;

public abstract class Vehicle implements Sellabe {

    private String model;
    private int price;
    private String colour;
    private int maxSpeed;

    public Vehicle(String model, int price, String colour, int maxSpeed){
        this.model = model;
        this.price = price;
        this.colour = colour;
        this.maxSpeed = maxSpeed;
    }

    public String getModel() {
        return model;
    }

    public int price() {
        return price;
    }

    public String getColour() {
        return colour;
    }

    public int getMaxSpeed() {
```

```java
public class SportsCar extends Vehicle implements Drivable {

    Engine engine;
    Wheels wheels;
    Chassis chassis;

    public SportsCar(String model, int price, String colour, int maxSpeed, Engine engine,
        super(model, price, colour, maxSpeed);
        this.engine = engine;
        this.wheels = wheels;
        this.chassis = chassis;
    }

    public String drive(){
        return "Vroom Vroom! Super fast";
    }

    public String getEngine() {
        return engine.getEngineType();
    }

    public String getWheels() {
        return wheels.getType();
    }

    public String getChassis() {
        return chassis.getMaterial();
    }
}
```

```java
public class ElectricCar extends Vehicle implements Drivable {

    Engine engine;
    Wheels wheels;
    Chassis chassis;

    public ElectricCar(String model, int price, String colour, int maxSpeed, Engine engine
        super(model, price, colour, maxSpeed);
        this.engine = engine;
        this.wheels = wheels;
        this.chassis = chassis;
    }

    public String drive(){
        return "Vroom.. slow and steady";
    }

    public String getEngine() {
        return engine.getEngineType();
    }

    public String getWheels() {
        return wheels.getType();
    }

    public String getChassis() {
        return chassis.getMaterial();
    }
}
```

```java
public class ElectricCarTest {

    ElectricCar electricCar;
    Engine engine;
    Wheels wheels;
    Chassis chassis;

    @Before
    public void before(){
        engine = new Engine( engineType: "Electric");
        wheels = new Wheels( type: "Biodegradable");
        chassis = new Chassis( material: "Leaves");
        electricCar = new ElectricCar( model: "Prius",  price: 20000,  colour: "green",  maxSpeed:
    }

    @Test
    public void hasModel(){
        assertEquals( expected: "Prius", electricCar.getModel());
    }

    @Test
    public void hasPrice(){
        assertEquals( expected: 20000, electricCar.price());
    }
```

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:

Paste Screenshot here

This algorithm of the card game lab (Blackjack) checks if a player holds a card value higher than 21. If that is the case, a loss is returned. Similarly, the algorithm checks if a player has won which is determined by him holding a value of 0 cards. In order to do this a for loop is used, looping over the array of all players. The final if condition checks if the player's holds a HandValue greater than the HandValue required for winning AND has not lost in order to be able to be declared a winner. I chose this algorithm as it is simpler and shorter than use else or elseif conditions (the latter ones tend to be enumerative in nature).

```
    private boolean checkLoss(Player player){
        if (player.getHandValue() > 21){
            player.setLoss(true);
        }
        return player.hasLoss();
    }



    public Player checkWin(){
       Player win = this.players.get(0);
        for (Player player : this.players){
            if (player.getHandValue() > win.getHandValue() && !player.hasLoss()){
                win = player;
            }
        }
        return win;
    }
}
```

The second algorithm as well is for a card game and checks for a) an Ace and b) which card is higher.

```
# Correct the errors below that you spotted in task 1.

require_relative('card.rb')
class CardGame
  def initialize()

  end

  def checkforAce(card1)
    if card1.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card1.name
    else
      card2.name
    end
  end
```