David Pears

Evidence Gathering Document
SQA Level 8 Professional Developer Award.

Week 2

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

array.rb

```ruby
def users
  users = {
    "jonathan" => {
      :twitter => "jonnyt",
      :lottery_numbers => [6, 12, 49, 33, 45, 20],
      :home_town => "Stirling",
      :pets => [
        {
          :name => "fluffy",
          :species => "cat"
        },
      ]
    },

    "Avril" => {
      :twitter => "bridpally",
      :lottery_numbers => [12, 13, 33, 38, 9, 25],
      :home_town => "Dunbar",
      :pets => [
        {
          :name => "monty",
          :species => "snake"
        }
      ]
    }
}
end

p users["Avril"][:lottery_numbers]
```

Simple program that shows two users in an array, each user has various attributes (their twitter, lottery numbers, home town and pets)

The function returns Avril's lottery numbers.

```
pda_examples — davidpears@Davids-MacBook-Pro — ../pda_examples — -zs...
➜  pda_examples ruby array.rb
[12, 13, 33, 38, 9, 25]
➜  pda_examples
```

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

PDAExample2.rb

```ruby
person1 = {
  age: 40,
  name: "David",
  quote: "I am studying at CodeClan",
  can_talk: true}

  def person_speaks person
    if person[:can_talk] == true
      return person[:quote]
    else
      return "I have nothing to say."
    end
  end

  puts person_speaks(person1)
```

```
➜  Wednesday git:(master) ✗ ruby PDAExample2.rb
I am studying at CodeClan
➜  Wednesday git:(master) ✗
```

An example of a hash in a program that returns a short sentence if a person has an ability to ‘talk’.

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

PDAExample3.rb

```ruby
holidays = [
  {destination: "Cuba", year: 2013, airline: "Air France"},
  {destination: "Sri Lanka", year: 2014, airline: "Qatar"},
  {destination: "Myanmar", year: 2015, airline: "Qatar"},
  {destination: "Madagascar", year: 2017, airline: "Turkish
  Airlines"}]

  def find_holiday_by_year(holidays, year)
    return holidays.find{ |holidays|
      holidays[:year] == year
    }
  end

puts find_holiday_by_year(holidays, 2015)

```

```
[➜  Wednesday git:(master) ✗ ruby PDAExample3.rb
{:destination=>"Myanmar", :year=>2015, :airline=>"Qatar"}
➜  Wednesday git:(master) ✗
```

Search function that finds all details of a holiday using the year of the trip.

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

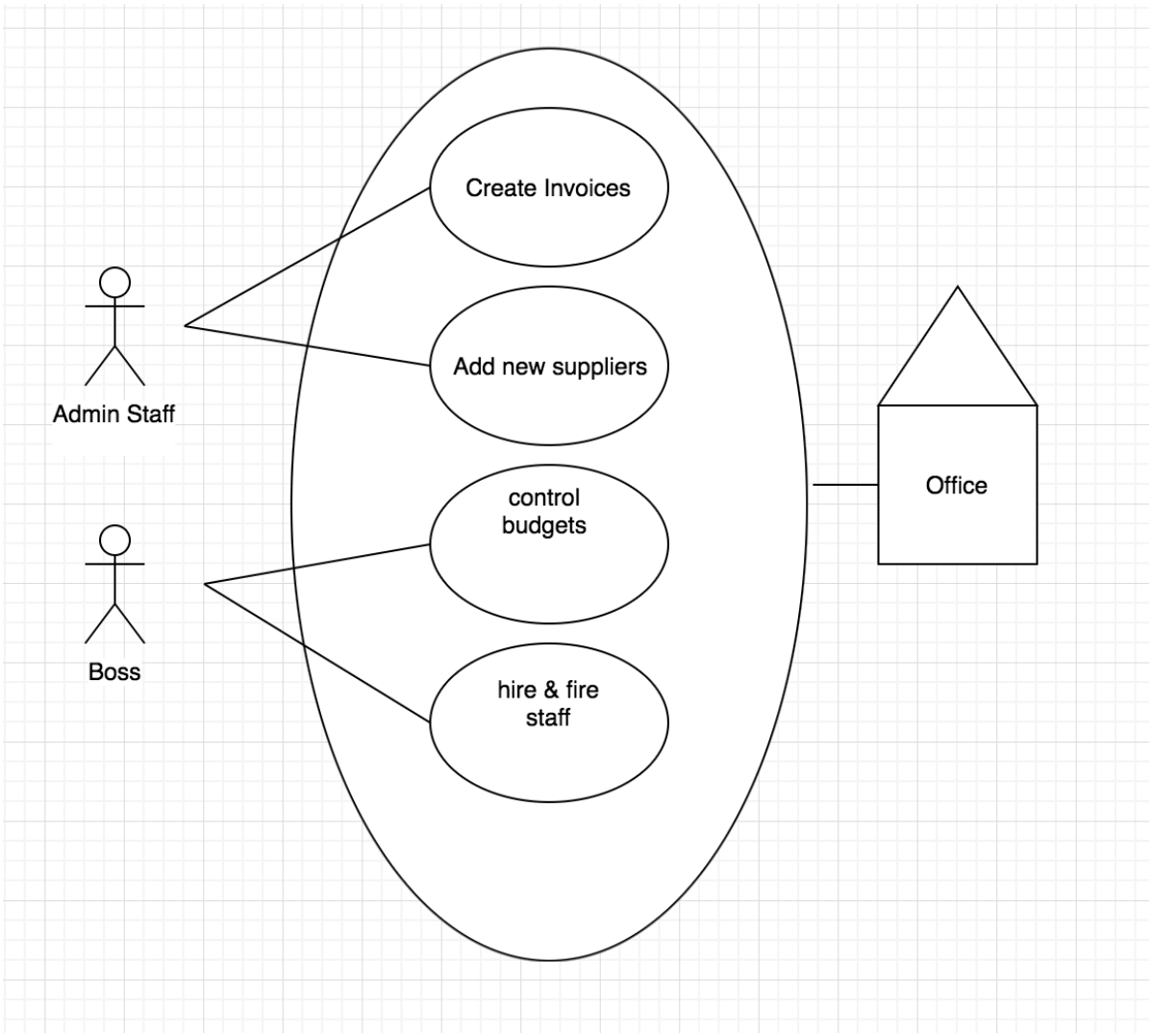
```ruby
class_mates = ["David", "Shaun", "Jesus", "Can", "Ben", "James A", "James H", "Gary", "Lee", "Iona", "Helen", "Gregor", "Stuart", "Ed", "Chris", "Daniel", "Magda", "Stephen", "Wil"]

def sort_class(array)
  array.sort
end

puts sort_class(class_mates)
```

```
➜  Wednesday git:(master) ✗ ruby PDAExample4.rb
Ben
Can
Chris
Daniel
David
Ed
Gary
Gregor
Helen
Iona
James A
James H
Jesus
Lee
Magda
Shaun
Stephen
Stuart
Wil
➜  Wednesday git:(master) ✗
```

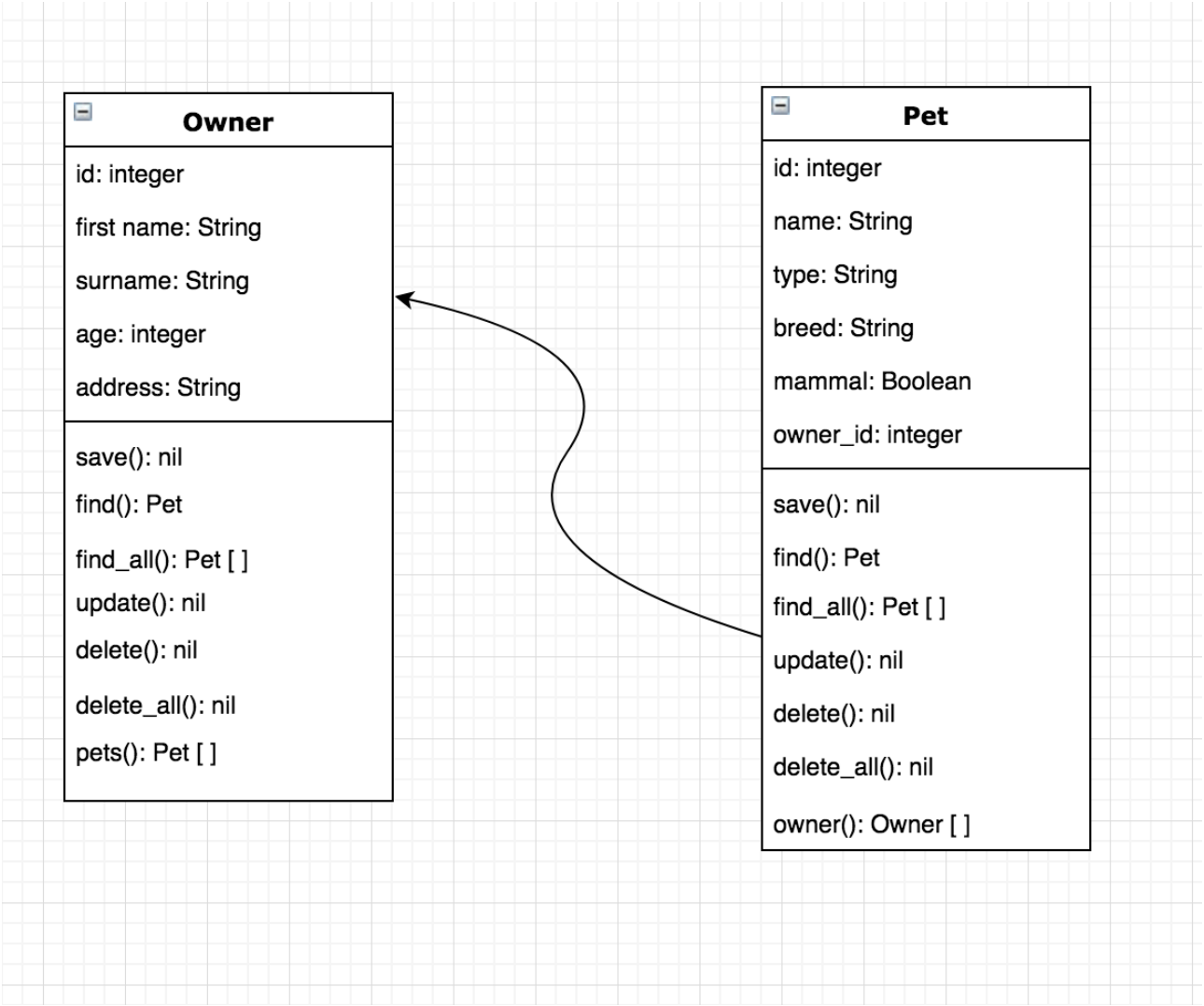
The above function sorts the people from the class in alphabetical order.

Week 5

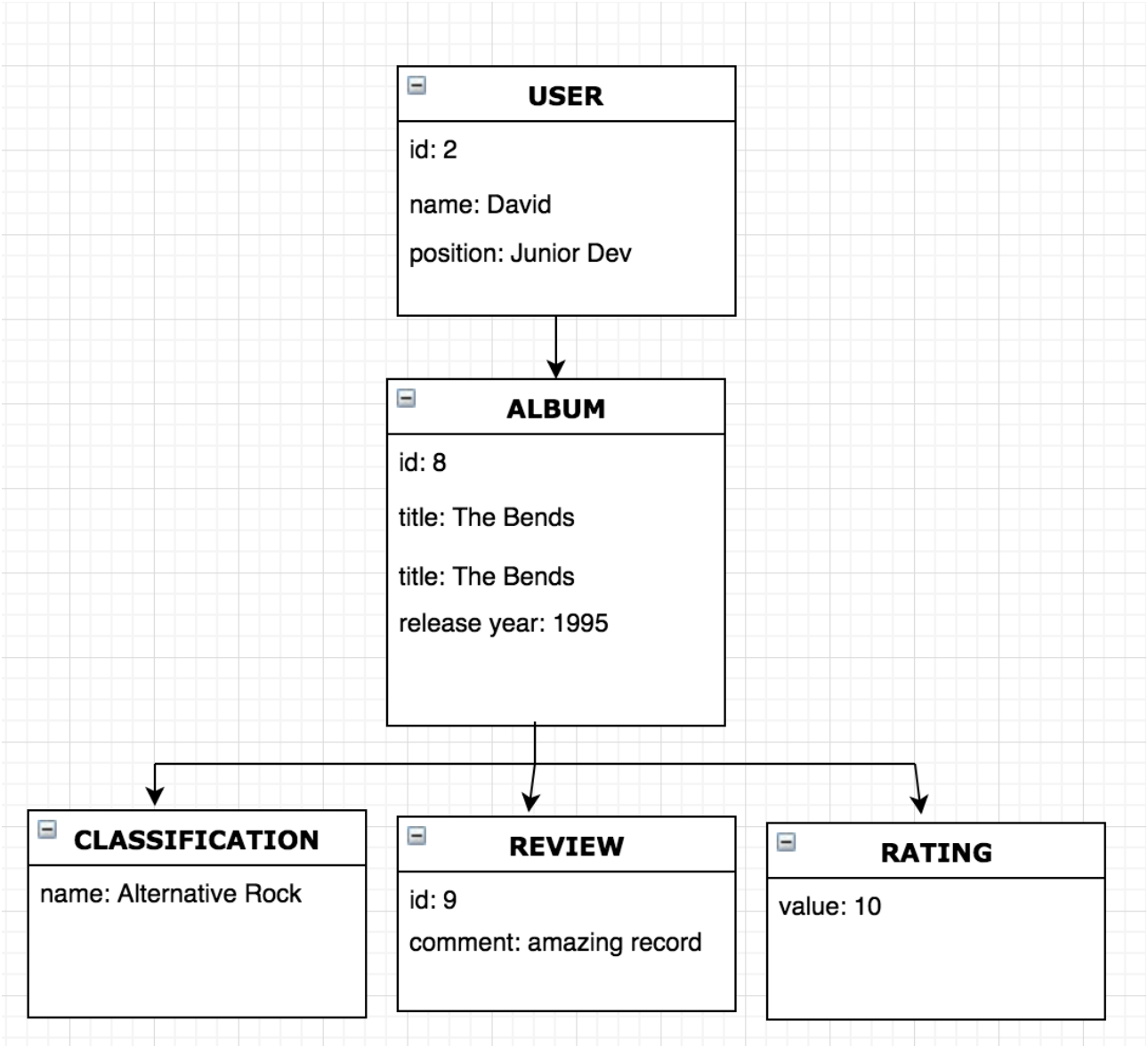
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram



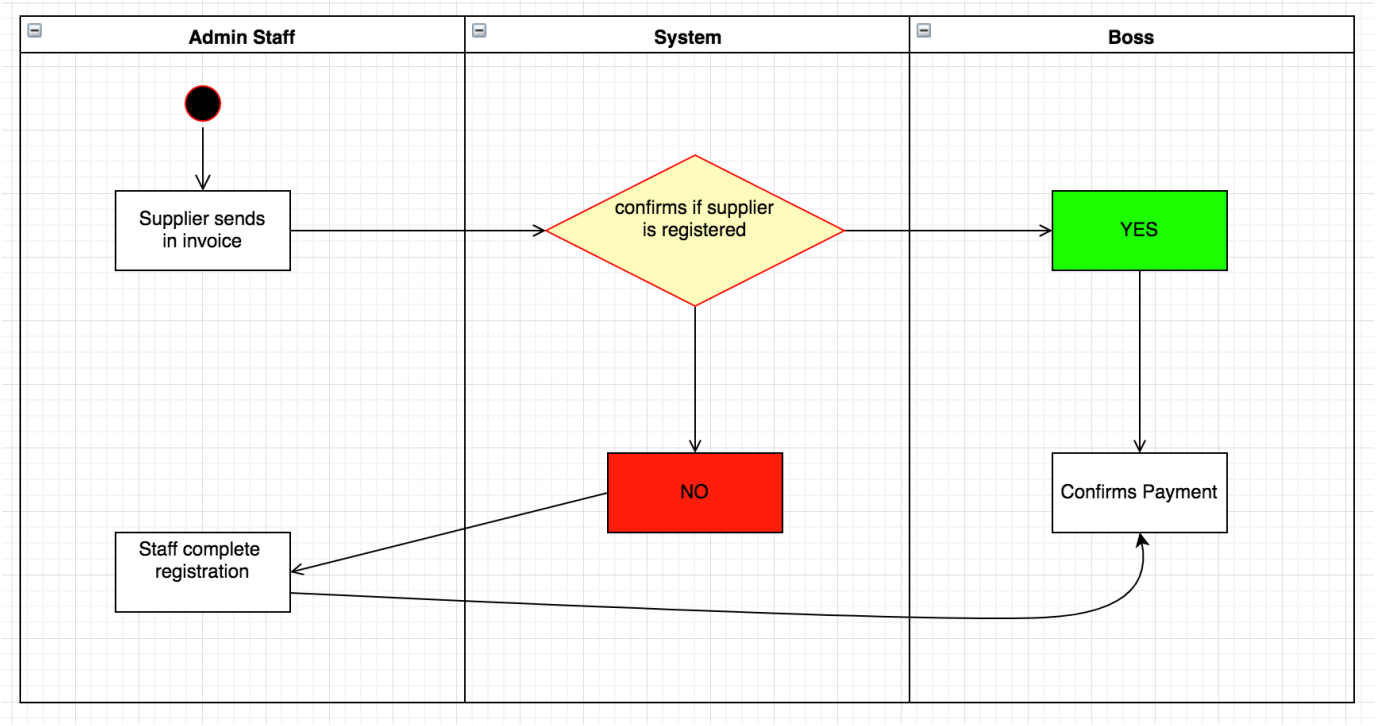
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram



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



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

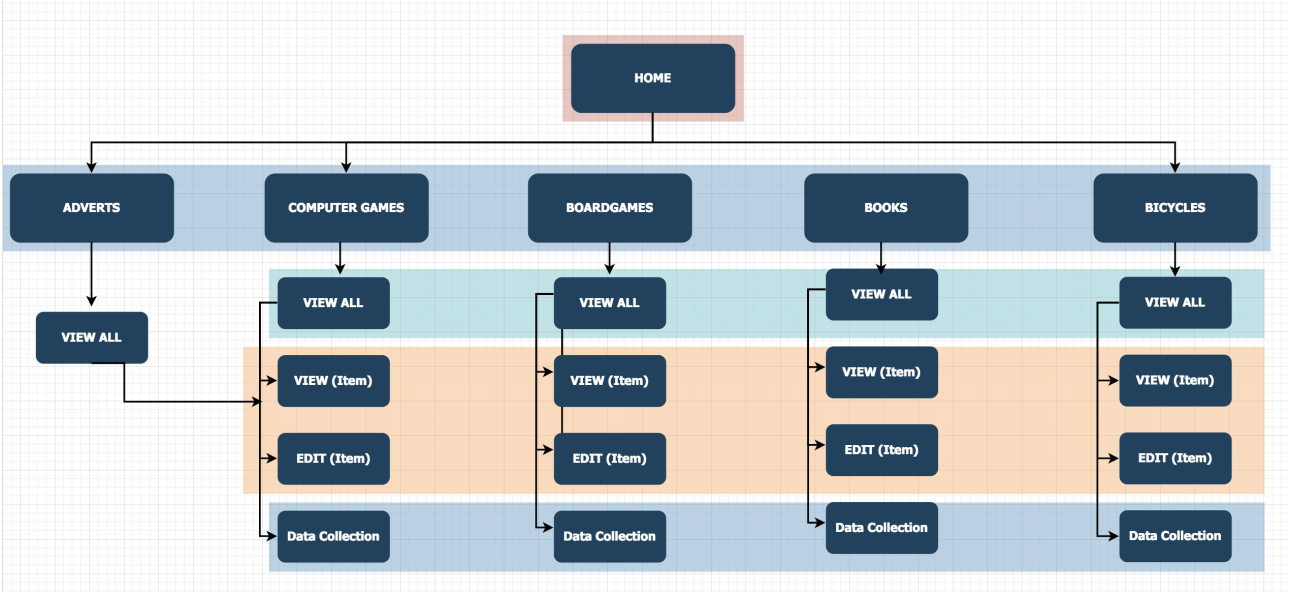


Unit	Ref	Evidence
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time

	Constraints & possible effect	Soloution
Hardware & software platforms	App/Project not resposive to mobile users and only tested in firefox. Other browsers may display differently	User a responsive design (flexbox or grid) to ensure mobile users can rcv the correct experience
Performance requirements	User device(s) may not be ideal. However optimisation for anything other than a desktop web-app not specified in the brief.	User-test to reveal any issues, then reformat for to work with a larger selection of devices
persistent storage and transactions	images added as url strings (i.e web-hosted images), which relies on those urls remaing active	Save file images to the database so they can be accessed permanently
Budgets	Limited budget designed to reach MVP	Once MVP reached, use proof of concept to aquire further budget to increase functionality and extend project
Time	timeframe (1 week) resulted in some extenstions/plans not being completed	Further extensions/functionality to be added at a later date

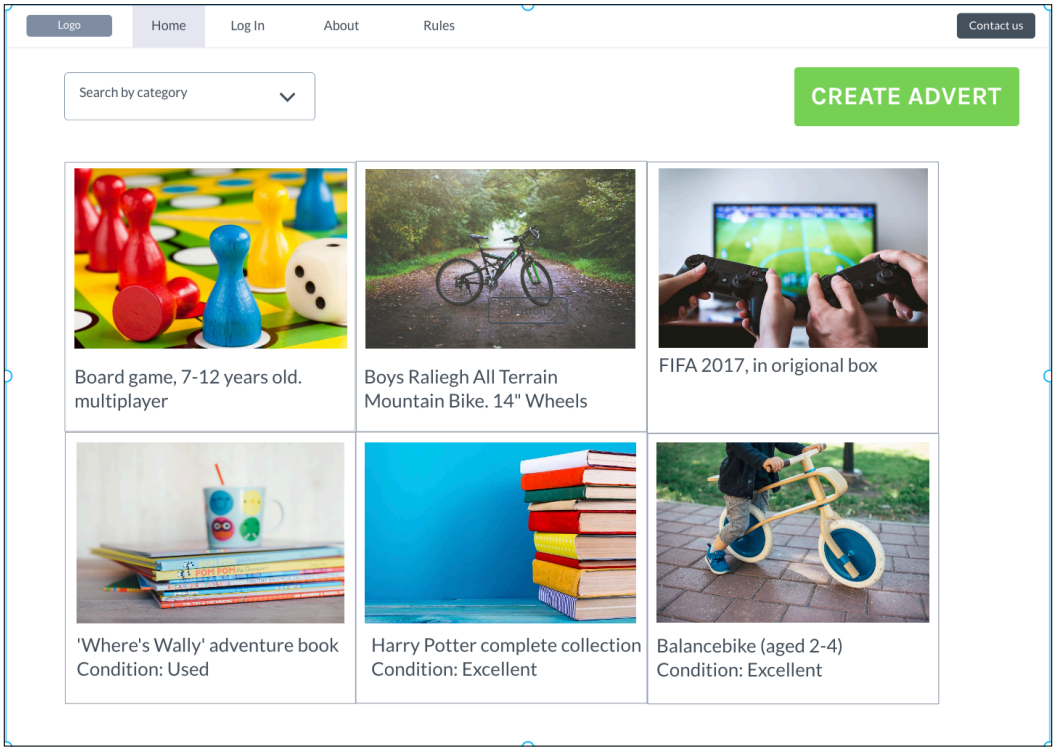
Unit	Ref	Evidence
P	P.5	User Site Map

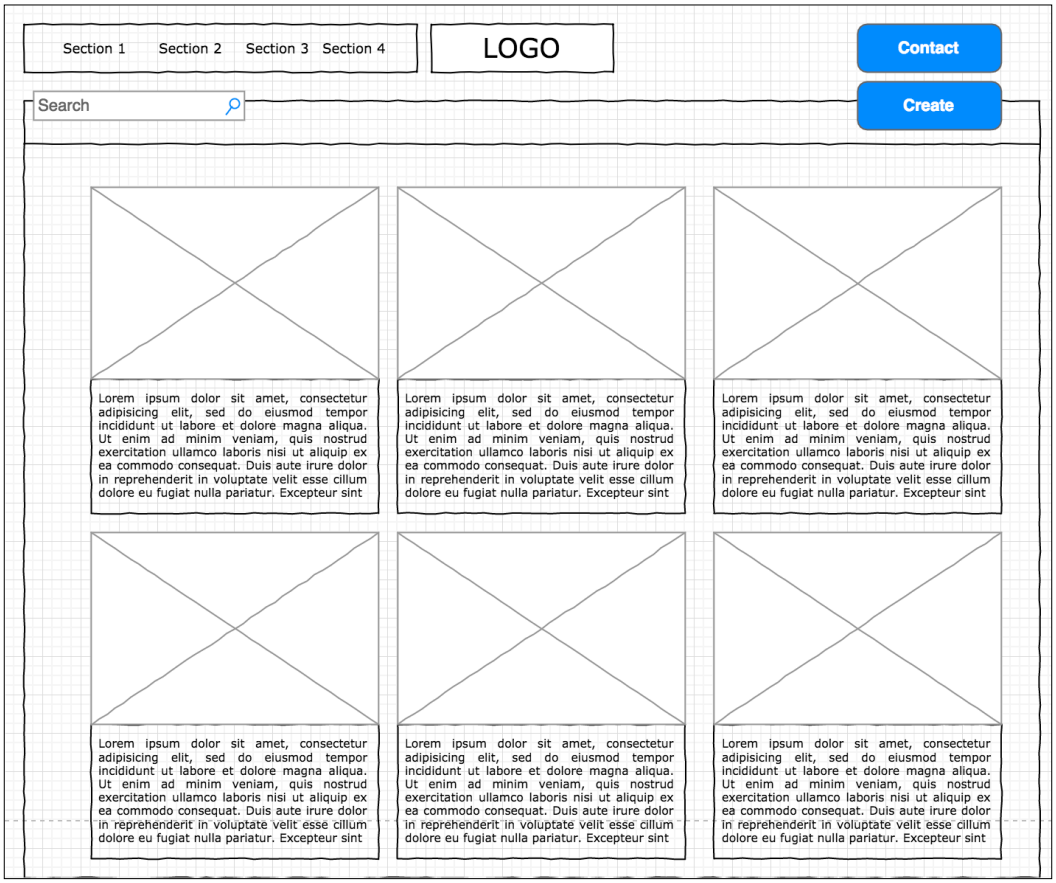
Site map for ‘Yodo’ Java project.



Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Taken from the ‘Yodo’ Java Project:





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

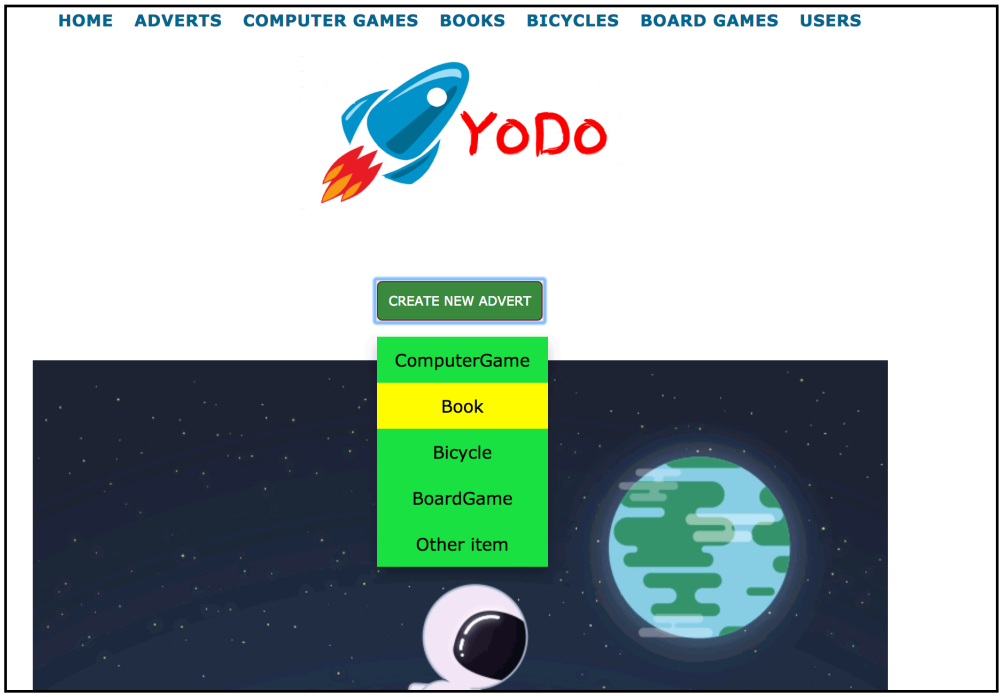
Pseudocode bookmarking a search bar:

```
it('should search for advert using keyword', function(){
  # a search bar that will return adverts
  # if the term entered appears in that advert
  # posssibly using the 'like' operator in java
}
```

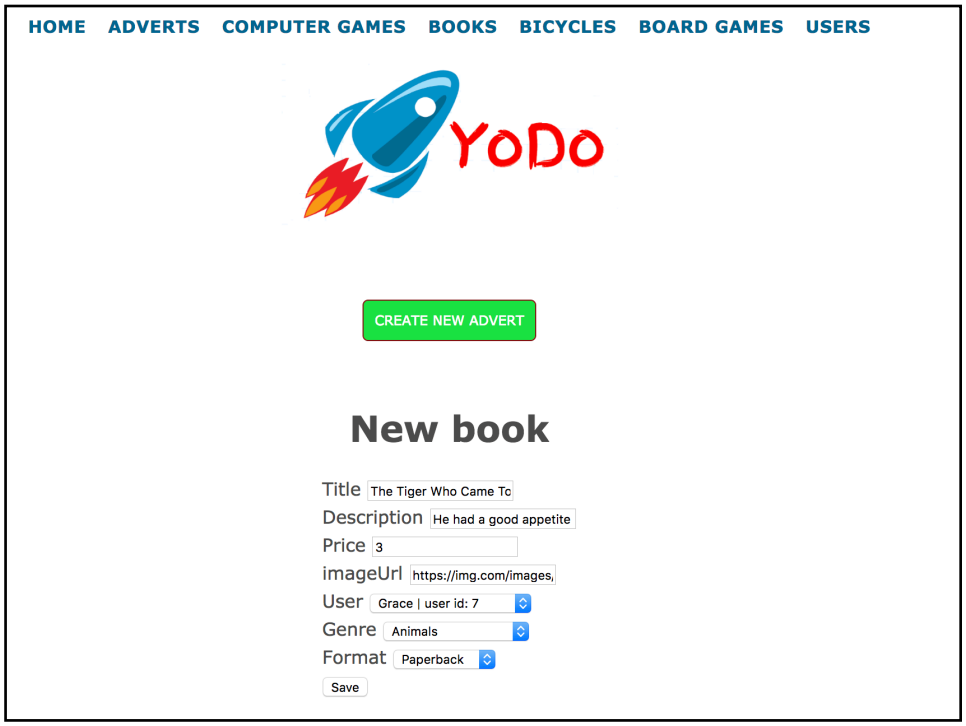
Unit	Ref	Evidence
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way

User adding ‘book’ to our online shop (‘YoDo’ project, Java)

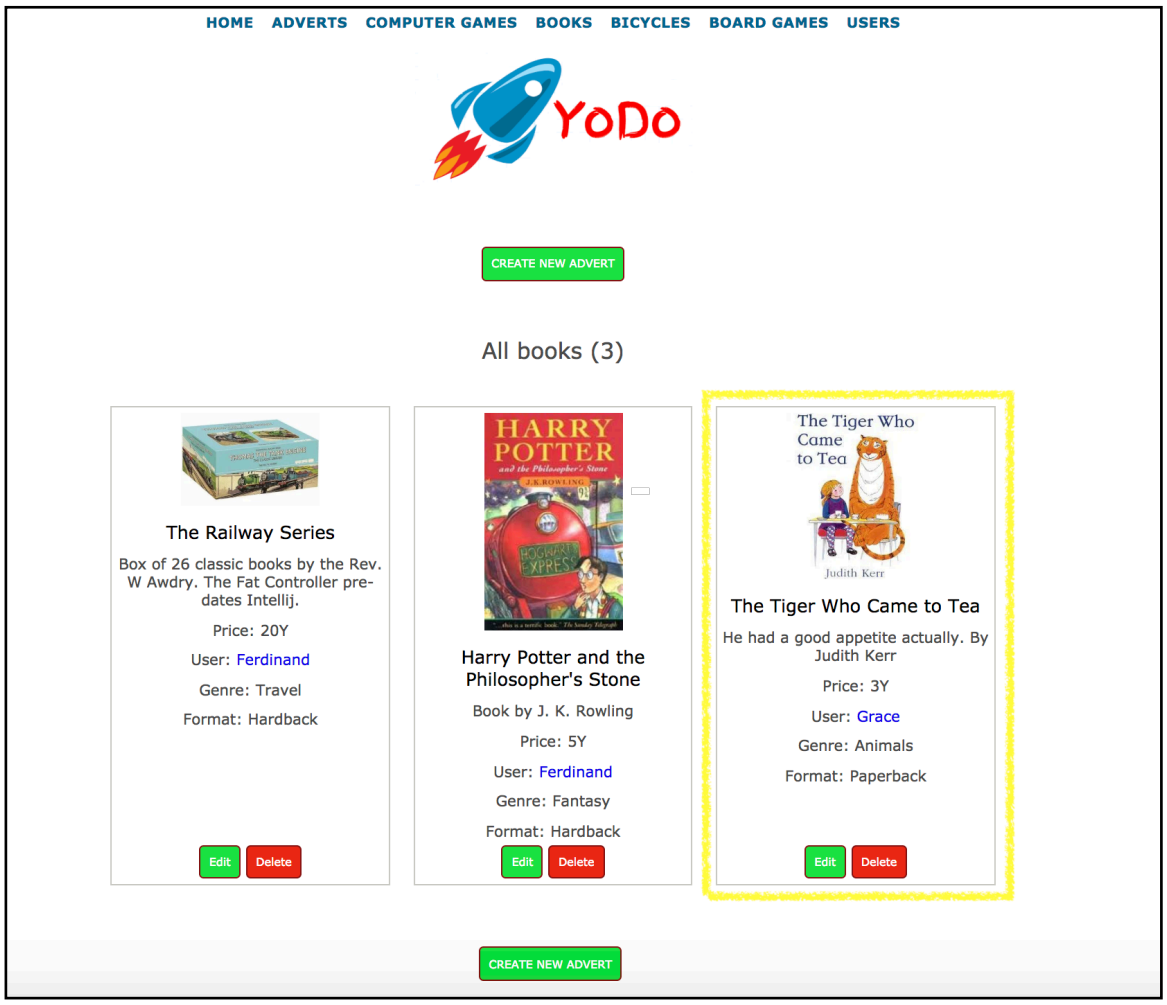
STEP ONE: User selects drop down category



STEP TWO: User adds the details:



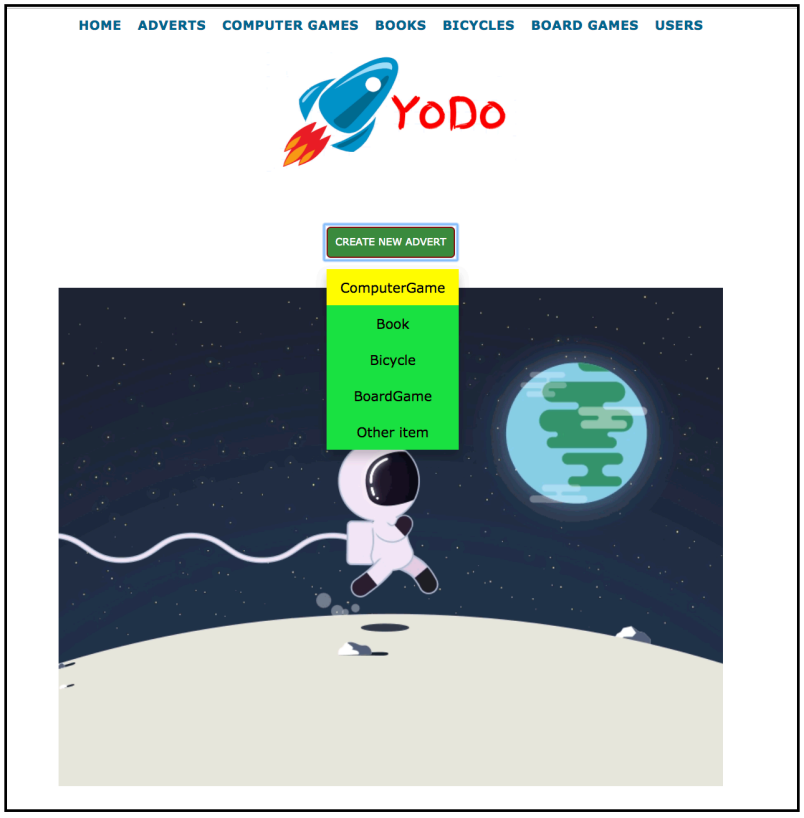
STEP THREE: New advert appears



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

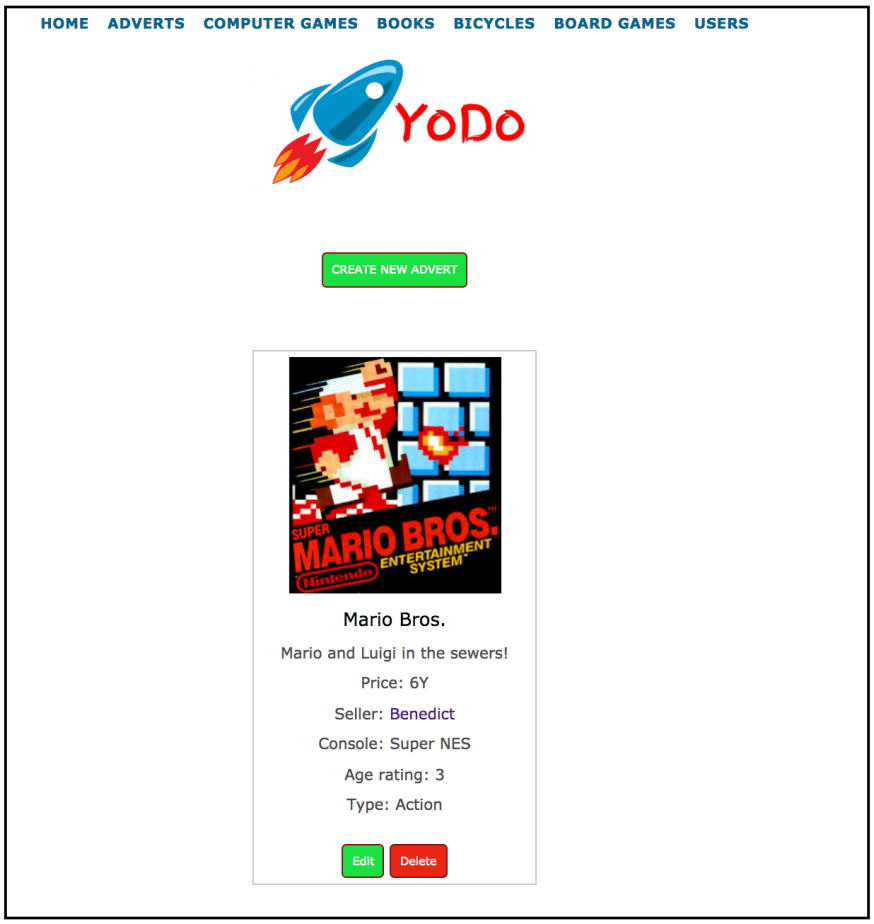
User adding ‘computer game’ to our online shop (‘YoDo’ project, Java)

STEP ONE: User selects drop down category



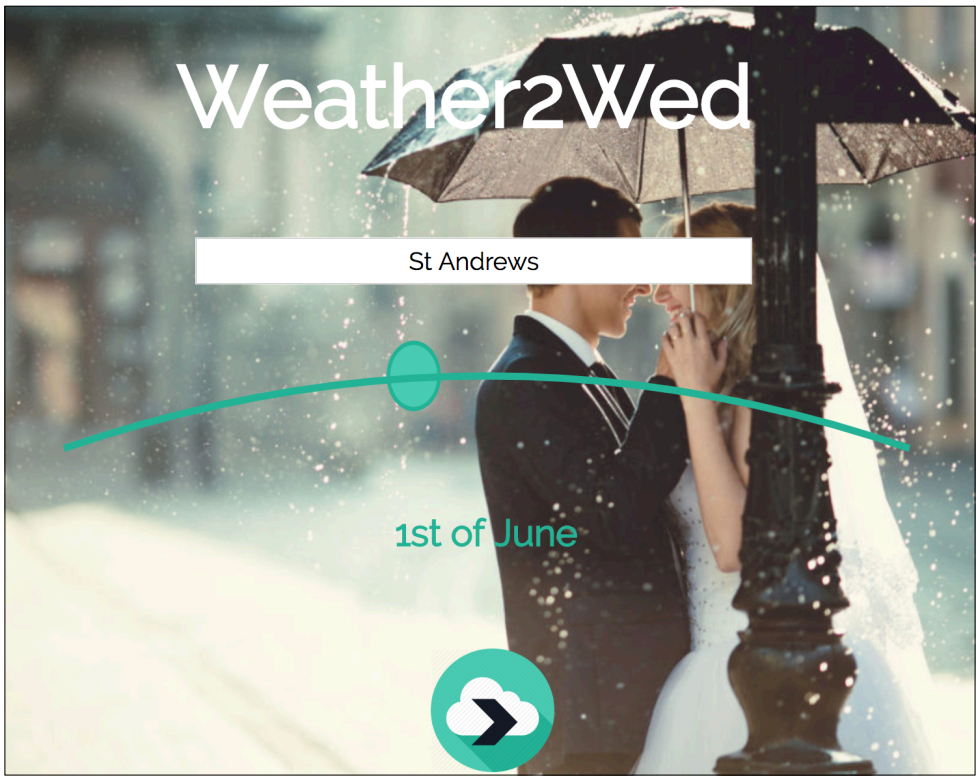
STEP TWO: User adds all relevant info:

STEP THREE: Advert is placed/displayed

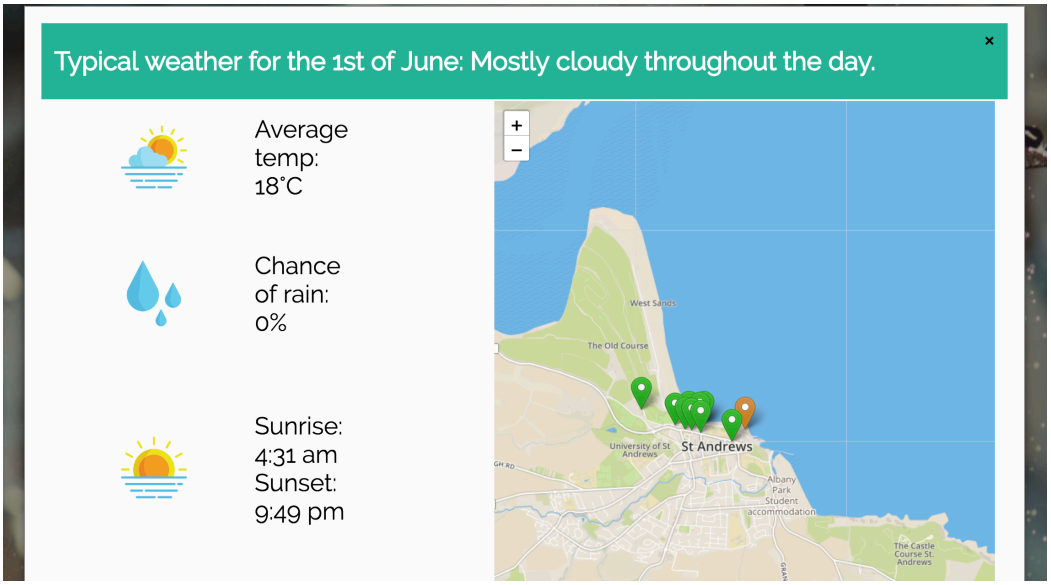


Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program

User enters a request for weather data based on ‘location’ and ‘date’



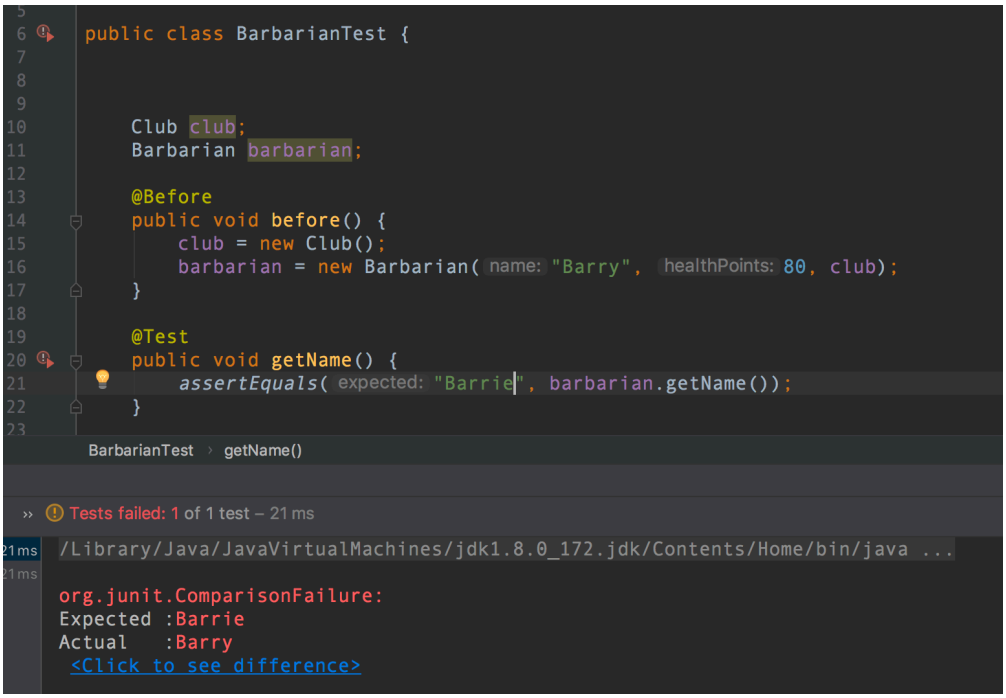
User is returned weather information for that date/location



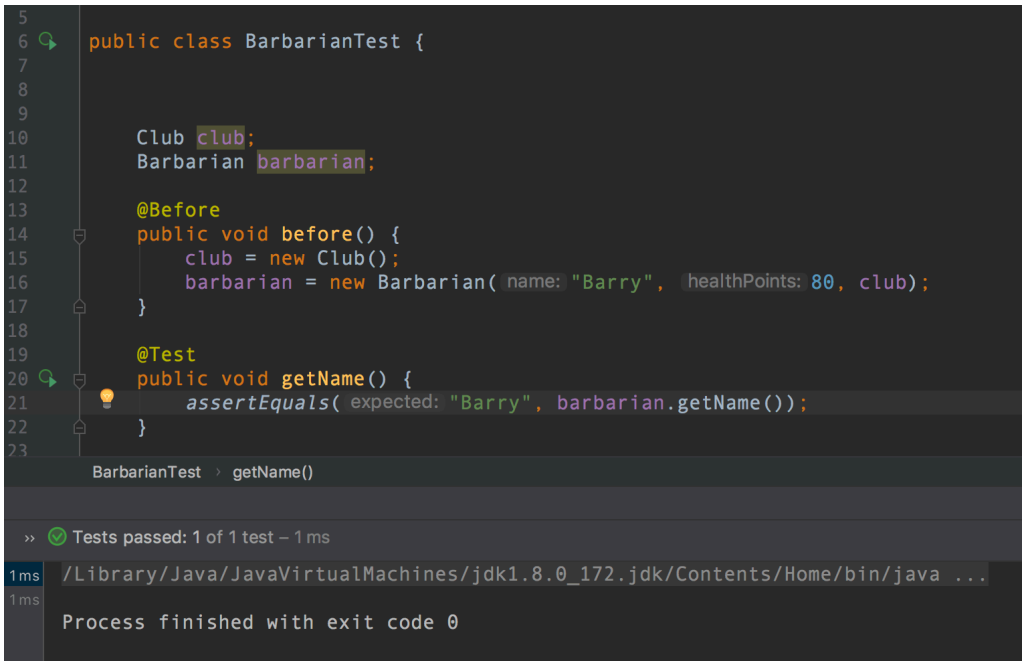
Unit	Ref	Evidence
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing

Testing for a Barbarians name in a game:

- test initially fails as his name is spelt incorrectly:



- test then passes as mistake is rectified:



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

Example of an 'ISell' Interface in a 'Music Shop' project.

Here both drum sticks and drum skins can be 'sold' - i.e we can perform a single action in different ways.

ISell.java

```java
package Accessories;

public interface ISell {

    double calculateMarkup();

    double getBuyingPrice( );

    double getSellingPrice( );

}
```

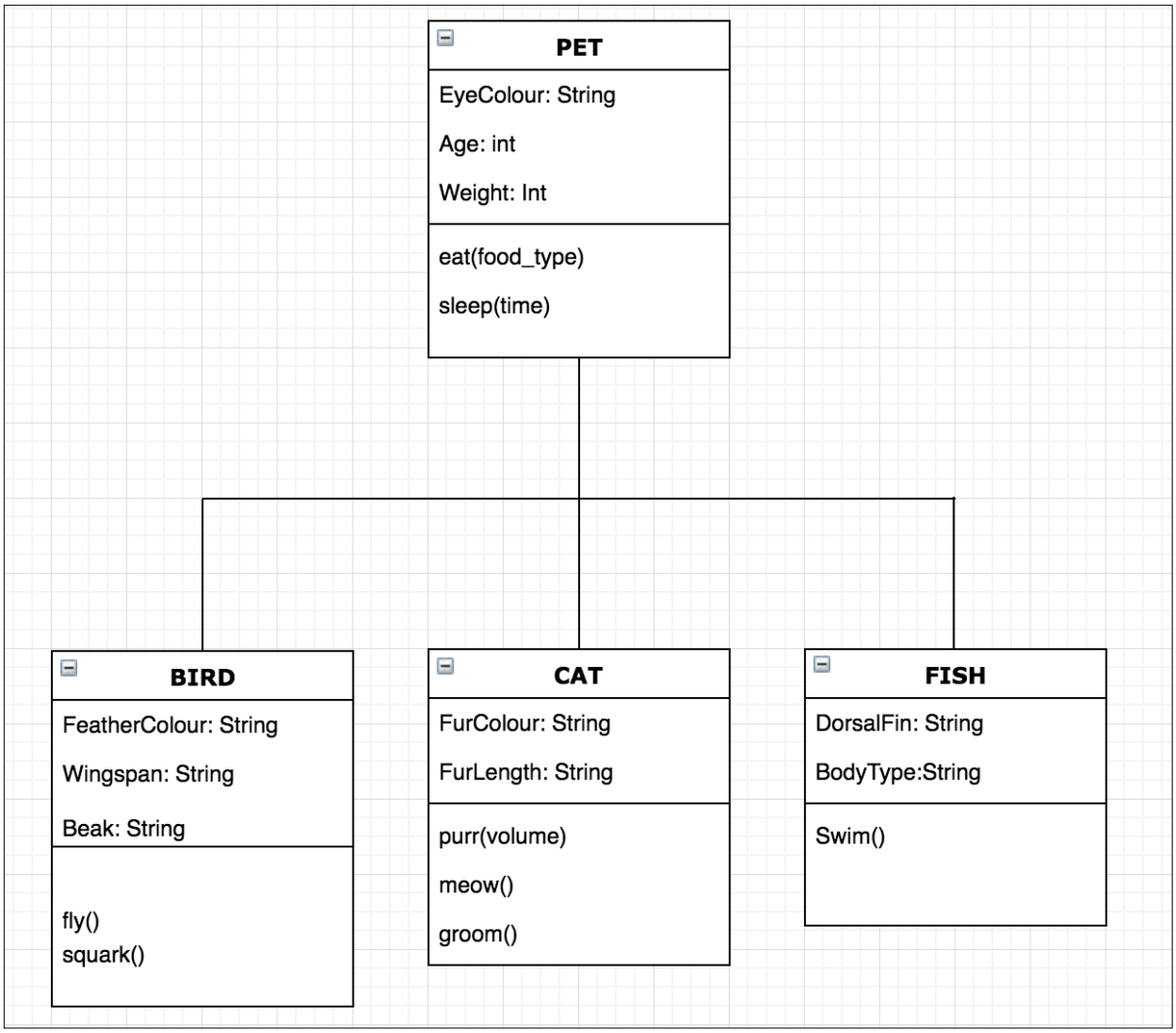
Drumsticks.java

```java
package Accessories;

public class Drumsticks implements ISell {

    private String type;
    private double buyingPrice;
    private double sellingPrice;

    public Drumsticks(String type, double buyingPrice, double sellingPrice) {
        this.type = type;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getType() { return type; }

    public double getBuyingPrice() { return buyingPrice; }

    public double getSellingPrice() { return sellingPrice; }

    @Override
    public double calculateMarkup() { return sellingPrice - buyingPrice; }

}
```

Drumskins.java

```java
package Accessories;

public class Drumskins implements ISell {

    private String type;
    private double buyingPrice;
    private double sellingPrice;

    public Drumskins(String type, double buyingPrice, double sellingPrice) {
        this.type = type;
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
    }

    public String getType() { return type; }

    public double getBuyingPrice() { return buyingPrice; }

    public double getSellingPrice() { return sellingPrice; }

    @Override
    public double calculateMarkup() { return sellingPrice - buyingPrice; }

}
```

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



Inheritance. Inheritance means that objects can inherit the traits and functions of other objects. Think of a cat. It can inherit characteristics of a ‘pet', but also has its own unique characteristics.

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

BankCustomer.java

```java
public class BankCustomer {
    private String first_name;
    private String surname;
    private String email;


    public BankCustomer(String first_name, String surname, String email) {
        this.first_name = first_name;
        this.surname = surname;
        this.email = email;
    }

    public String getFirst_name() {
        return this.first_name;
    }

    public String getSurname() {
        return this.surname;
    }

    public String getemail() {
        return this.email;
    }
}

```

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.

Example: Class 'Pirates' (Abstract class)

Pirate.java

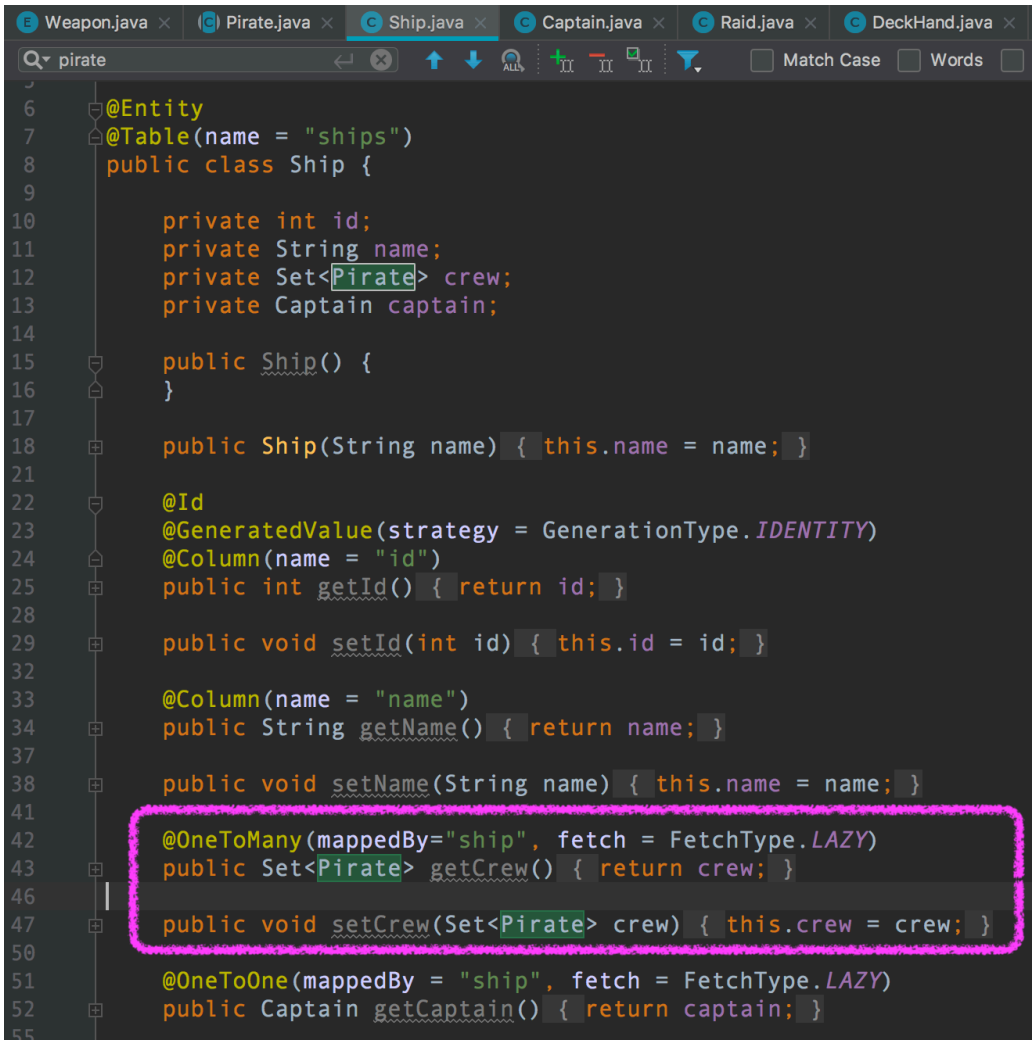
```java
package models;

import org.hibernate.annotations.Cascade;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name="pirates")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Pirate {

    private int id;
    private String firstName;
    private String lastName;
    private int age;
    private Weapon weapon;
    private Ship ship;
    private List<Raid> raids;

    public Pirate() {
    }

    public Pirate(String firstName, String lastName, int age, Weapon weapon, Ship ship) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        this.weapon = weapon;
        this.ship = ship;
        this.raids = new ArrayList<Raid>();
    }
```

The class 'Deckhand' ***inherits*** from 'Pirate', creates a 'deckhand' object. It has the 'super' characteristics from the abstract class, plus its own characteristics.

Pirate.java | Captain.java | DeckHand.java

```java
package models;

import javax.persistence.Entity;
import javax.persistence.Table;
import java.util.List;

@Entity
@Table(name = "deckhands")
public class DeckHand extends Pirate {

    public DeckHand() {
    }

    public DeckHand(String firstName, String lastName, int age, Weapon weapon, Ship ship) {
        super(firstName, lastName, age, weapon, ship);
    }
}
```

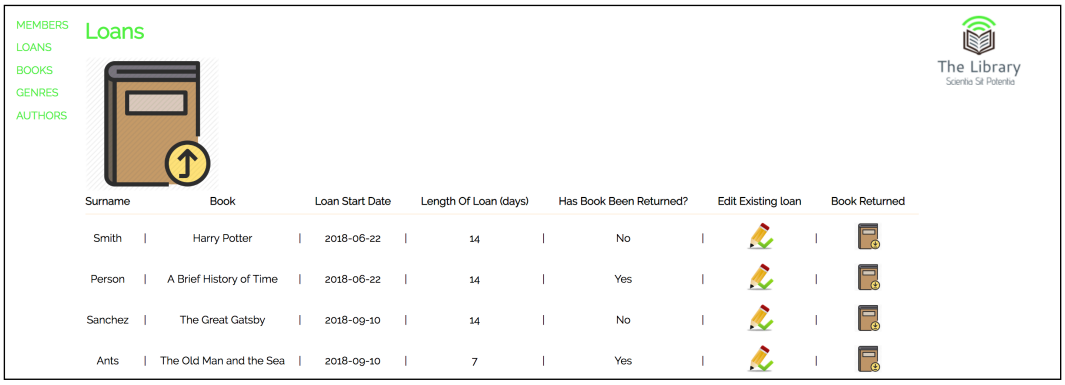
The method that uses the information (pirates) inherited from another class

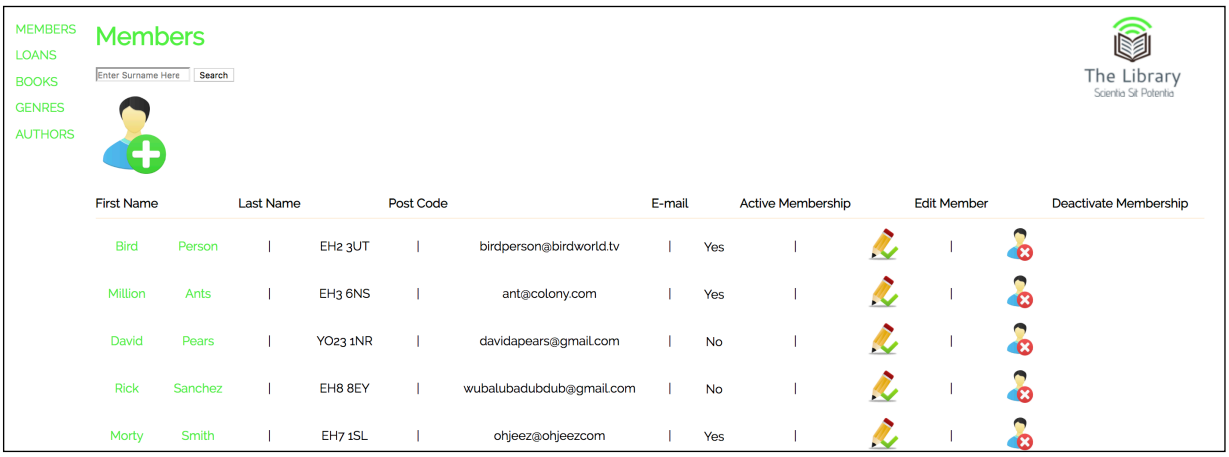


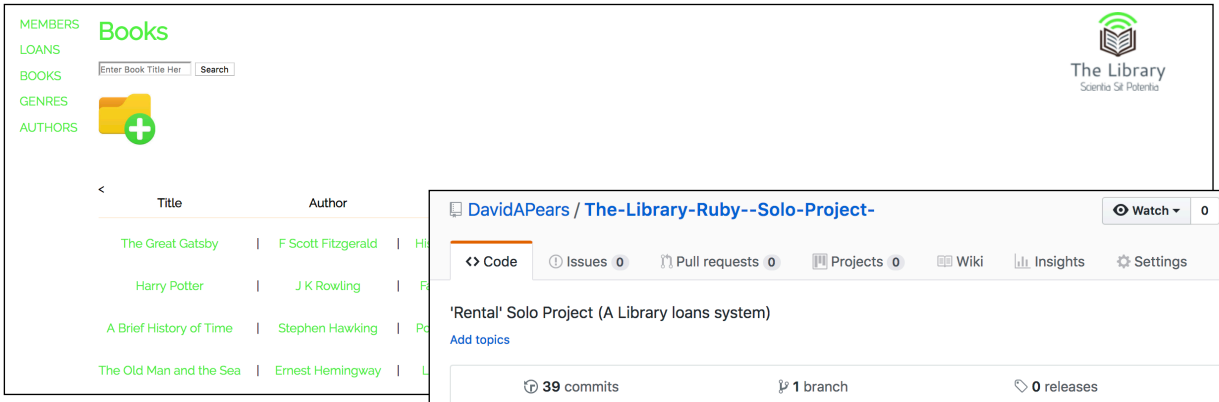
Unit	Ref	Evidence
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.

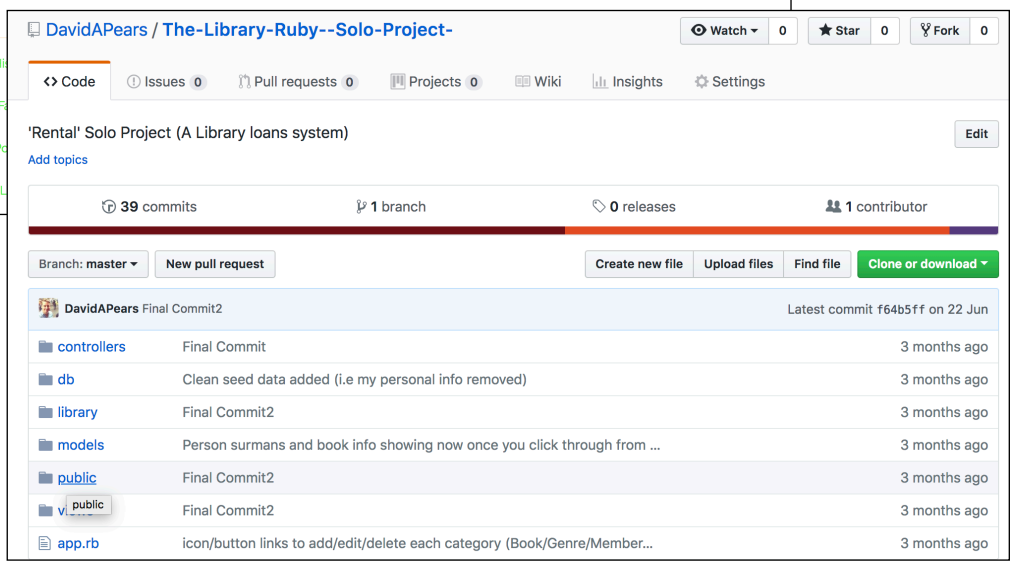
The library Project:

Github Link: https://github.com/DavidAPears/The-Library-Ruby--Solo-Project-





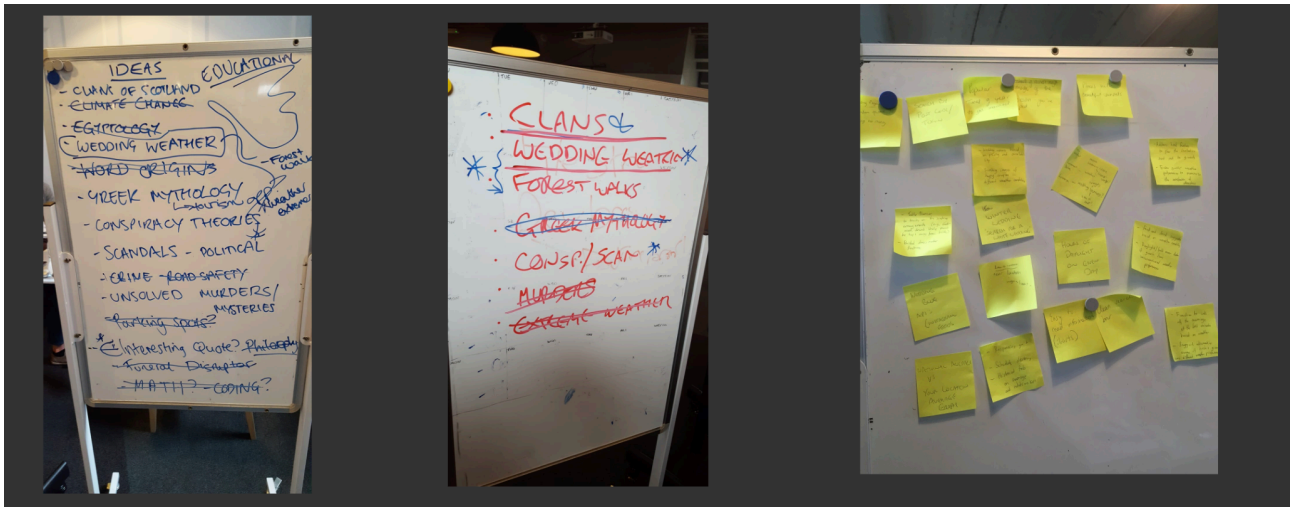




Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.

Planning for ‘Weather2Wed’ app
A JavaScript project

The below shows the concept being narrowed down from a huge list of initial ideas to a single idea, and then brain storming on features that were to appear on the project.



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.

sort.java

```
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {
                    //swap temp and arr[i]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
    }

//    Prints the array
    void printArray(int arr[]){
    int n = arr.length;
    for (int i=0; i<n; ++1)
        System.out.println(arr[i] + " ");
    System.out.println();
    }

//    Driver method to test above
    public static void main(String args[]){
        BubbleSort ob = new BubbleSort();
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
        ob.bubbleSort(arr);
        System.out.println("sorted array");
        ob.printArray(arr);
    }
}

```

The examples below are of a ‘bubble sort’ and an ‘insert sort’ both are used to sort arrays. These are both fairly easy to implement and work well. Neither of these were part of projects per se, just

```
Sorted array
11 12 22 25 34 64 90
```

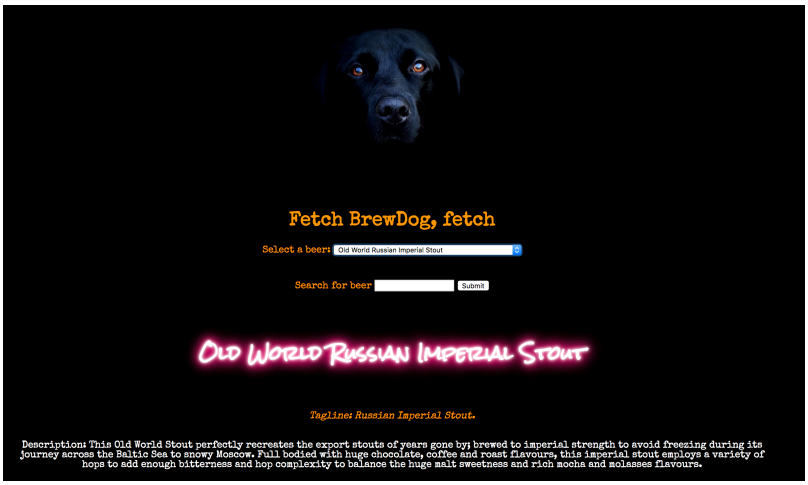
something I studied to understand the concept.

sort.java InsertionSort.java

```


public class InsertionSort {

    void sort(int arr[]) {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    static void printArray(int arr[]) {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    public static void main(String args[]) {
        int arr[] = {12, 11, 13, 5, 6};

        InsertionSort ob = new InsertionSort();
        ob.sort(arr);

        printArray(arr);
    }
}
```

Output: 5, 6, 11, 12, 13

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



The ‘Brewdog’ API being called to return info on their beer selection (limited to 80 pages per page).

beer_detail_view.js | beer_form_view.js | beer_list_view.js | select_beer_view.js | beers.js

```
const Request = require('../helpers/request_helper.js');
const PubSub = require('../helpers/pub_sub.js');

const Beers = function () {
  this.data = null;
}

// OPTION A 'GETTER': Get request, returns all data from API (all beers)
Beers.prototype.getData = function () {
 const url = `https://api.punkapi.com/v2/beers?page=2&per_page=80`;
 const request = new Request(url);
 request.get()
   .then((data) => {
     this.data = data
     PubSub.publish('Beers:beers-ready', this.data)
   })
   .catch((message) => {
     console.error(message);
   });
}
```

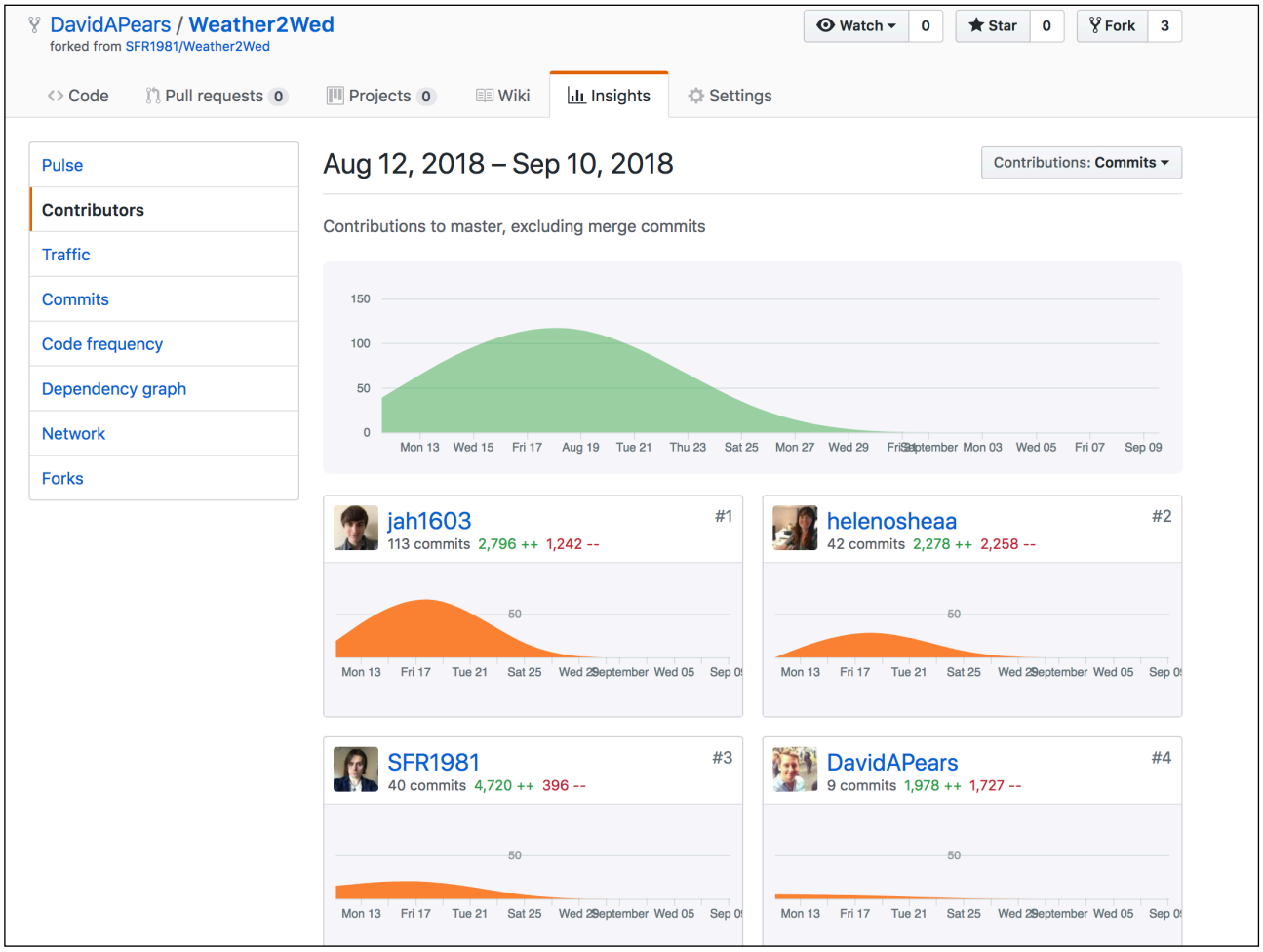
The code shows the API being used to populate the dropdown search box and returning info on that particular beer once one has been selected (see screen shot).

app.js | beers.js | index.html | request_helpe | beer_detail_vie | bee

```
// SELECT_BEER_VIEW = BEER DROPDOWN BOX

const PubSub = require('../helpers/pub_sub.js');

const SelectBeerView = function (selectElement) {
  this.element = selectElement;
};

SelectBeerView.prototype.bindEvents = function () {
  PubSub.subscribe('Beers:beers-ready', (evt) => {
    this.populate(evt.detail)
  });

  this.element.addEventListener('change', (evt) => {
    const selectedIndex = evt.target.value;
    PubSub.publish('SelectView:change', selectedIndex);
  })
};

SelectBeerView.prototype.populate = function (beers) {
  beers.forEach((beer, index) => {
    const beerOption = this.createOption(beer.name, index);
    this.element.appendChild(beerOption);
  });
};

SelectBeerView.prototype.createOption = function (name, index) {
  const option = document.createElement('option');
  option.textContent = name;
  option.value = name;
  return option;
};

module.exports = SelectBeerView;
```

Week 15

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



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

The brief

You have been approached by a wedding magazine looking to improve their online offering of wedding content by developing some interactive apps that display information in a fun and interesting way.

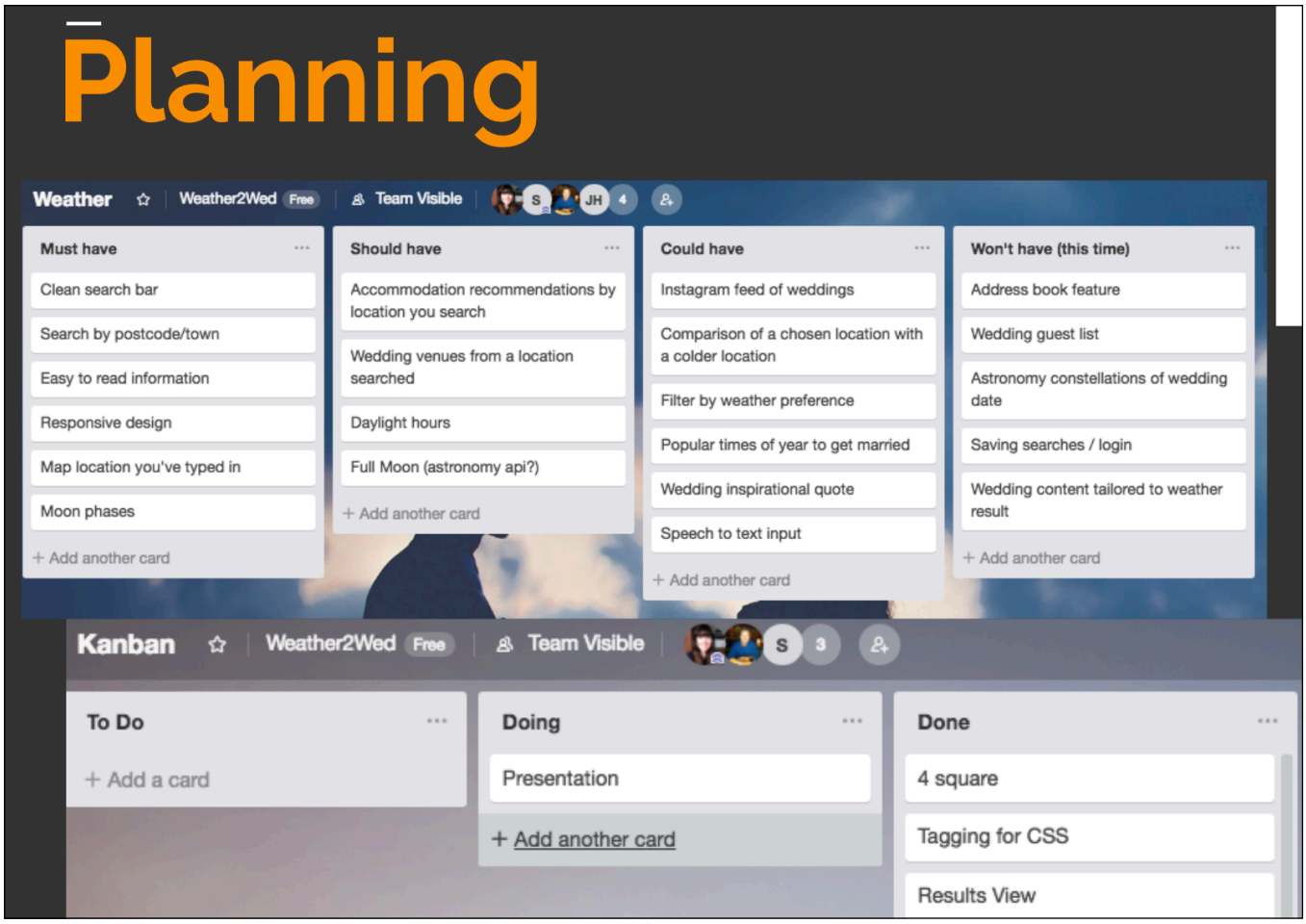
MVP

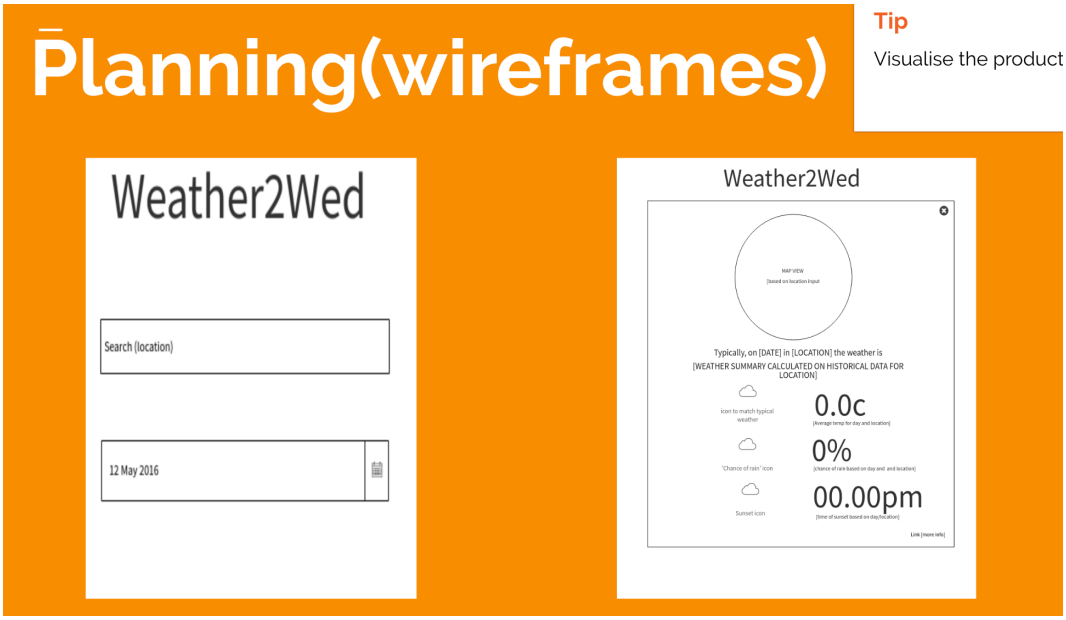
Display some information about a particular topic in an interesting way.

Have some interactivity that enables a user to move through different sections of content.

Examples of further features
Bring in data using an API or create your own.

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

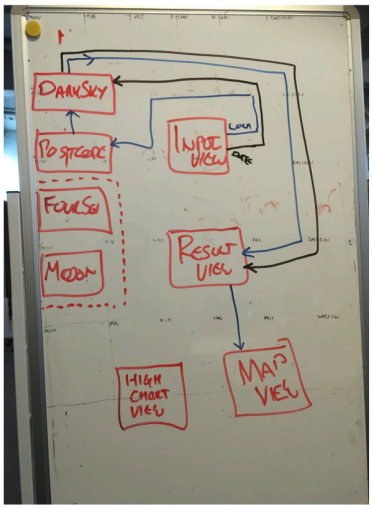


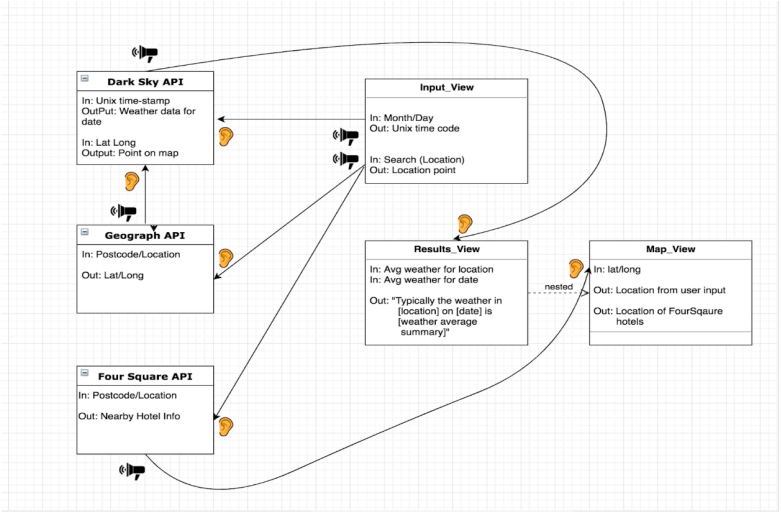


Planning(models)

Tip

Understand your data flow

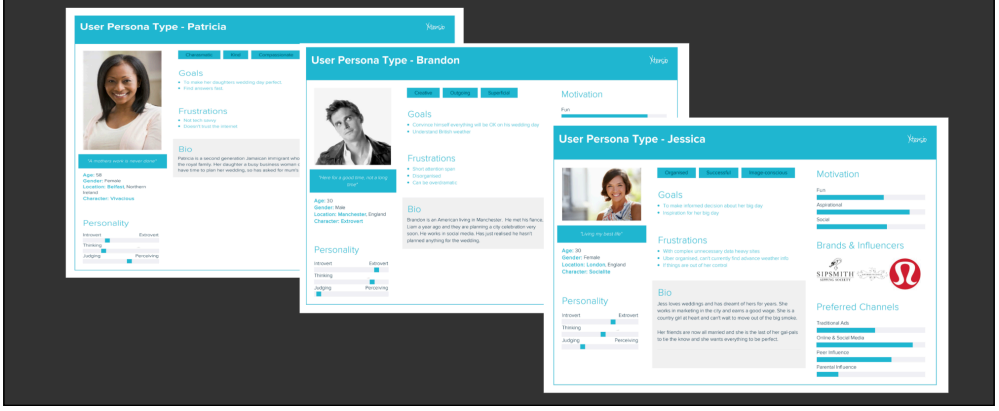




Planning(UX)

Tip

User tests on friends & family



A sample of the planning phase of the group project (taken from our group presentation)

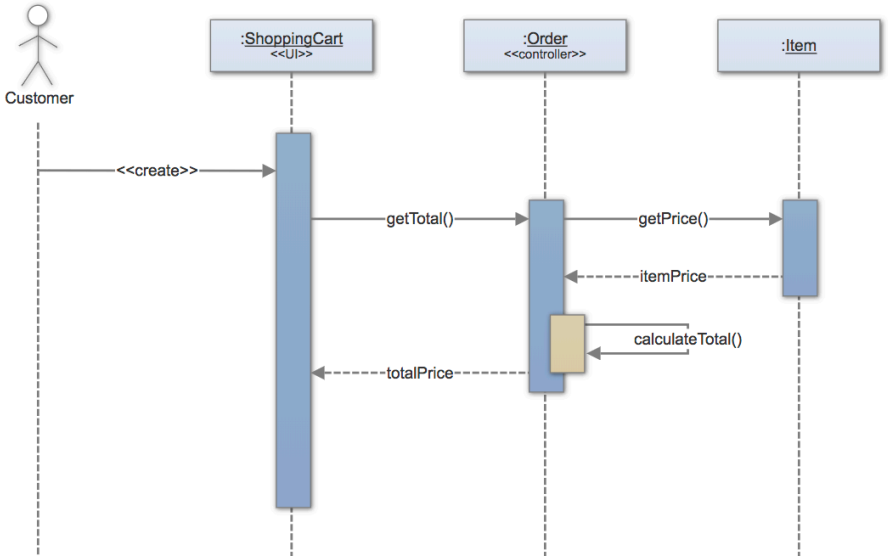
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

Criteria for ‘Weather2Wed’ app

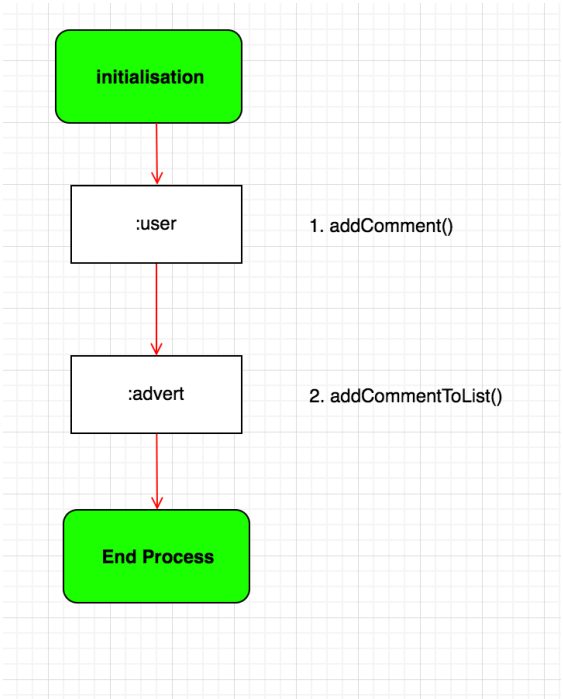
Acceptence Criteria	Expected Result	Pass / Fail ?
Can user enter date and location of future wedding	User should be able to enter the day (not year) and location of wedding	Pass
Can the user search by location	Weather data is returned using the location entered	Pass
Can the user search by specified date	The 'typical' weather (taken from 40 years of historical data) for given date is returned	Pass
Can user understand weather data	Weather data is displayed in familiar icon format (i.e icons used by othert common weather apps)	Pass

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

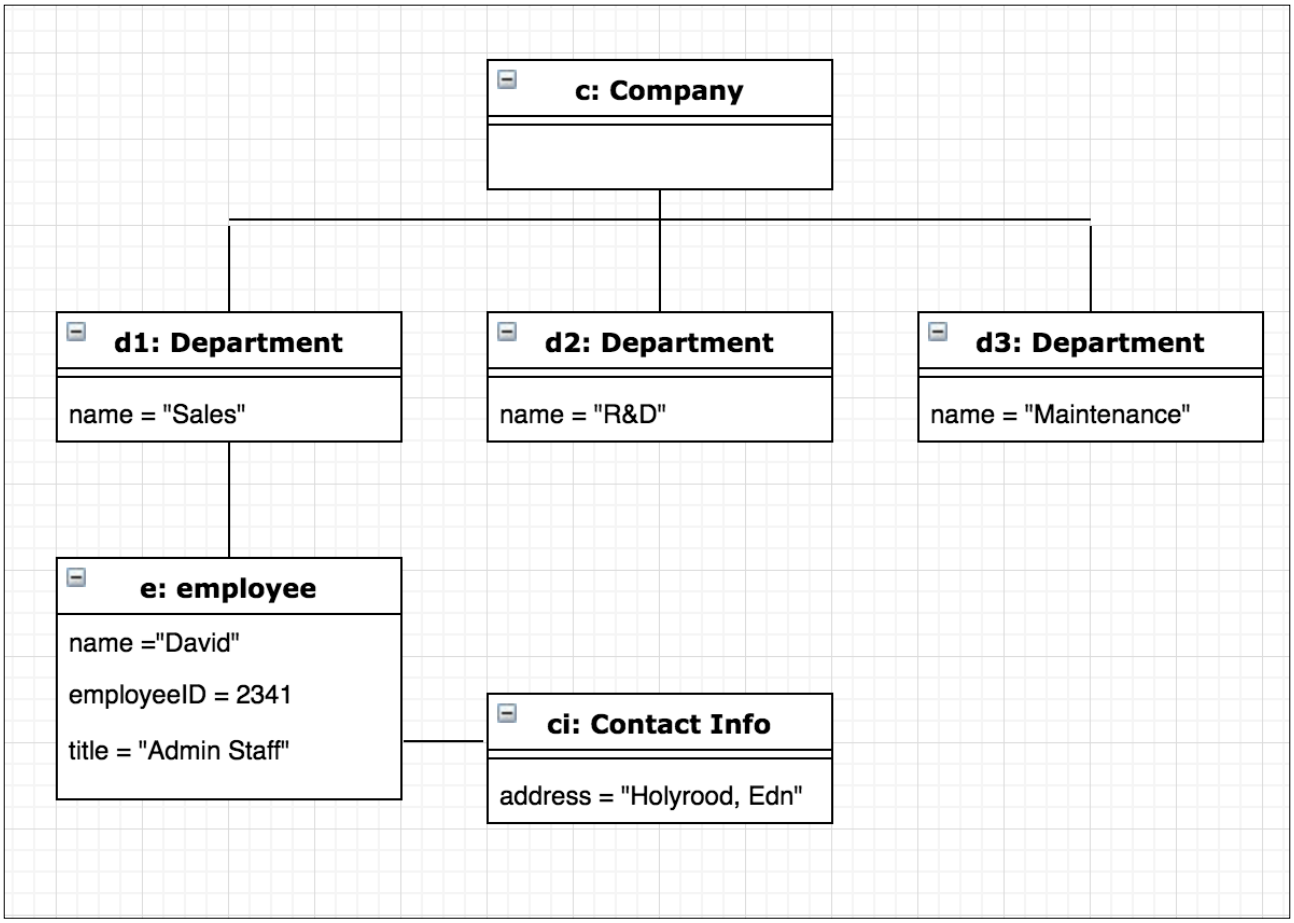
Sequence Diagram for a shopping cart:

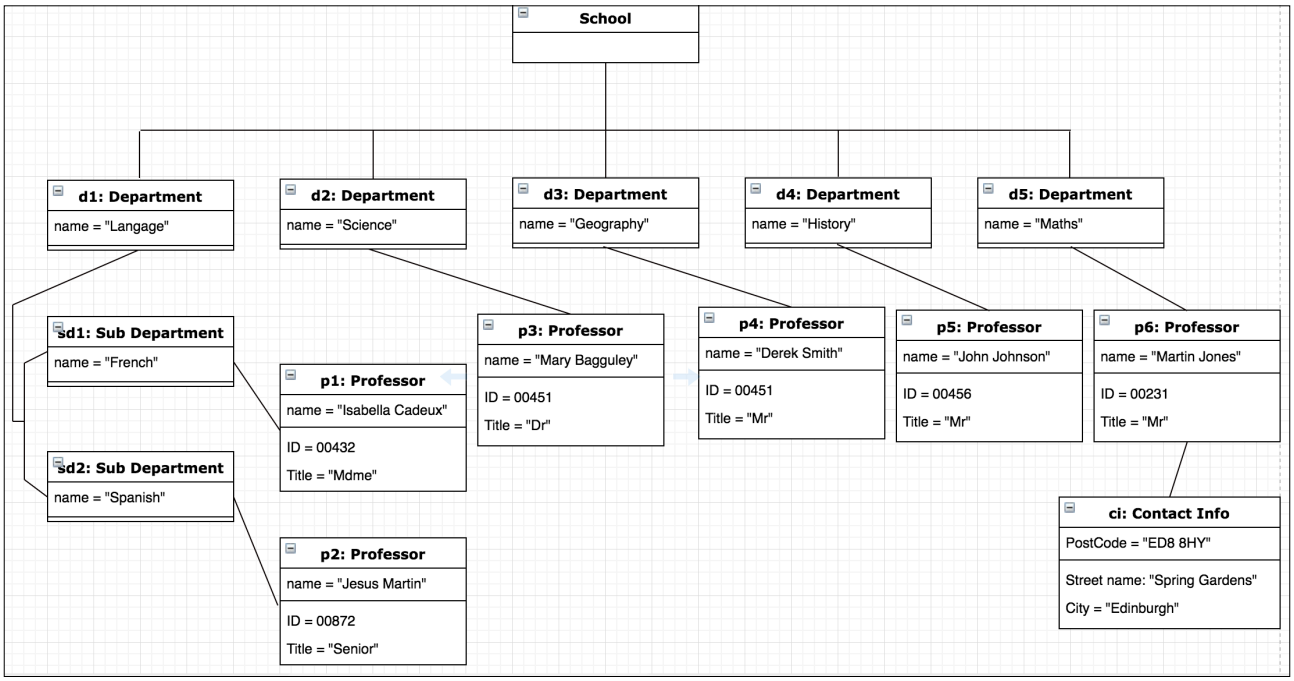


Collaboration Diagram showing a user interacting with an advert (i.e leaving a comment on it)



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





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

Sample bug tracking report (as pertaining to the ‘Weather2Wed’ js app)

BUG		RESOLOUTION	
Weather prediction cannot be given (unless postcode has been entered). I.e search to restrictive	Failed	Fuzzy search added which allows user to enter postcode, town name, landmark or even partial placename	Passed
Day-of-year selector on slide bar/arc does not reset after each submit	Failed	A 'reset' added to ensure that on refresh the day-selector does not show historic selection	Passed
Map will not initialise / open	Failed	Debugged code, it had been looping so opening multiple maps at once.	Passed
Respone method in results (view) runs several times, but only called once	Failed	n/a - ongoing	n/a
API max's out at 100 calls per day	Failed	Changed API from 'geocode.xyz' to 'geograph'	passed