**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

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

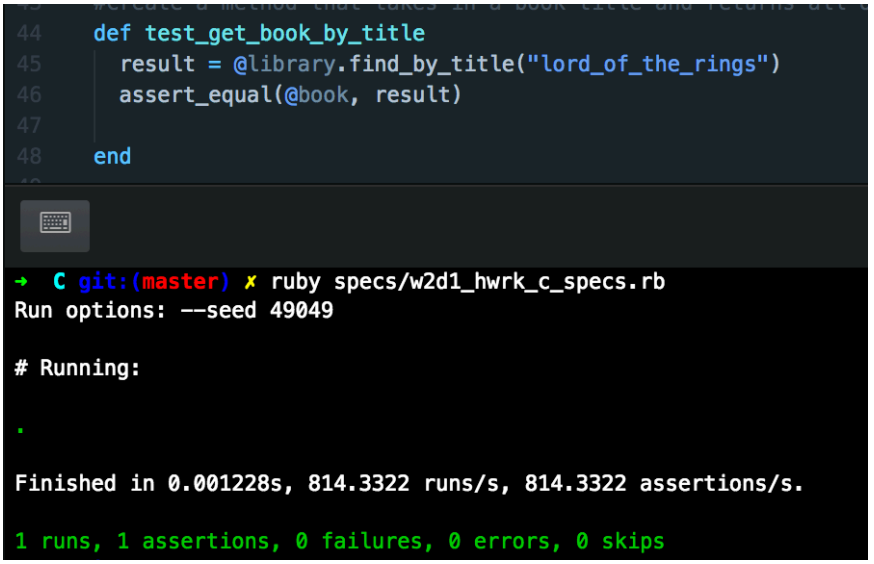
Example of an array in a program: Initialising an empty array as a 'book' object in the 'Library' class (Week 2, Day 1 homework)

```
class Library

  attr_accessor :books

  def initialize()
    @books = []
  end
```

Example of a function using the array: 'find_by_title' searches the array and returns the 'book' hash from the array.

```
def find_by_title(title)
@books.select {|book| book[:title] == title }[0]
end
```

Example of the function running within a MiniTest test environment, and the test passing.



```
44    def test_get_book_by_title
45      result = @library.find_by_title("lord_of_the_rings")
46      assert_equal(@book, result)
47
48    end
```

```
➜  C git:(master) ✗ ruby specs/w2d1_hwrk_c_specs.rb
Run options: --seed 49049

# Running:

.

Finished in 0.001228s, 814.3322 runs/s, 814.3322 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

| Unit | Ref | Evidence |
| --- | --- | --- |
| **I&T** | I.T.6 | Demonstrate the use of a hash in a program. Take screenshots of:<br>*A hash in a program<br>*A function that uses the hash<br>*The result of the function running |
| | | **Description:** |

Example of a hash in a program (Week 2 day 1 homework):

```
@book =
    {
      title: "lord_of_the_rings",
      rental_details: {
        student_name: "Jeff",
        date: "01/12/18"
      }
    }
```

A function, 'find_by_title' that uses the hash

```
def find_by_title(title)
@books.select {|book| book[:title] == title }[0]
end
```

The result of the function running, resulting in a successful test past in MiniTest.

```
53   def test_get_rental_details_by_title
54     result = @library.get_rental_details("lord_of_the_rings")
55     assert_equal(@book[:rental_details], result)
56   end
57
```

```
Finished in 0.001280s, 781.2500 runs/s, 781.2500 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
→  specs git:(master) ✗
```

specs/w2d1_hwrk_c_specs.rb ⓘ 0 ⚠ 0 ⓘ 0 88:1

# Week 3

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

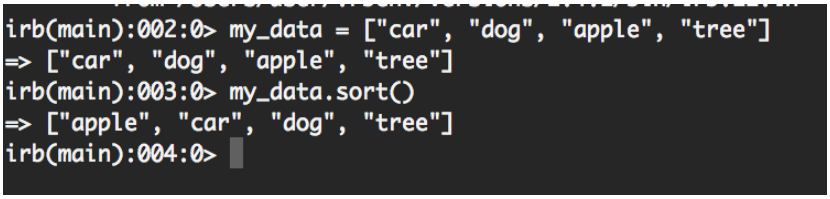
Example SQL query that searches a database and returns:

top: all results
bottom: a specific id instance.

```
imdb=# SELECT * FROM movies;
 id  |          title          | genre
-----+-------------------------+--------
 118 | The Blair Witch Project | Horror
 119 | Sideways                | Comedy
 120 | American Splendour      | Comedy
(3 rows)

imdb=# SELECT * FROM movies WHERE id = 118;;
 id  |          title          | genre
-----+-------------------------+--------
 118 | The Blair Witch Project | Horror
(1 row)
```

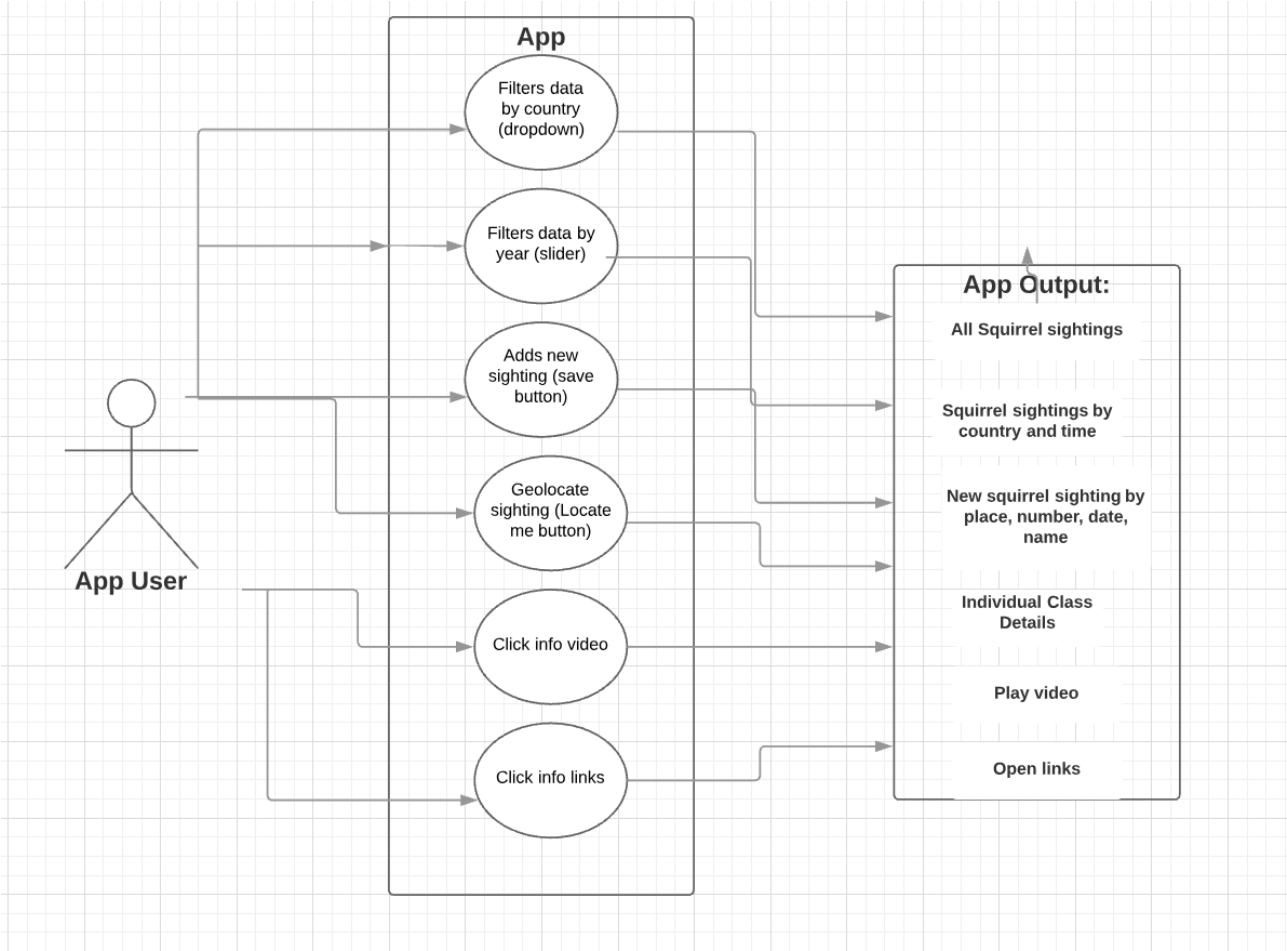
| Unit | Ref | Evidence |
| --- | --- | --- |
| **I&T** | I.T.4 | Demonstrate sorting data in a program. Take screenshots of:<br>*Function that sorts data<br>*The result of the function running |
| | | **Description:** |

Example of data sorting on an array to arrange the contents alphabetically.

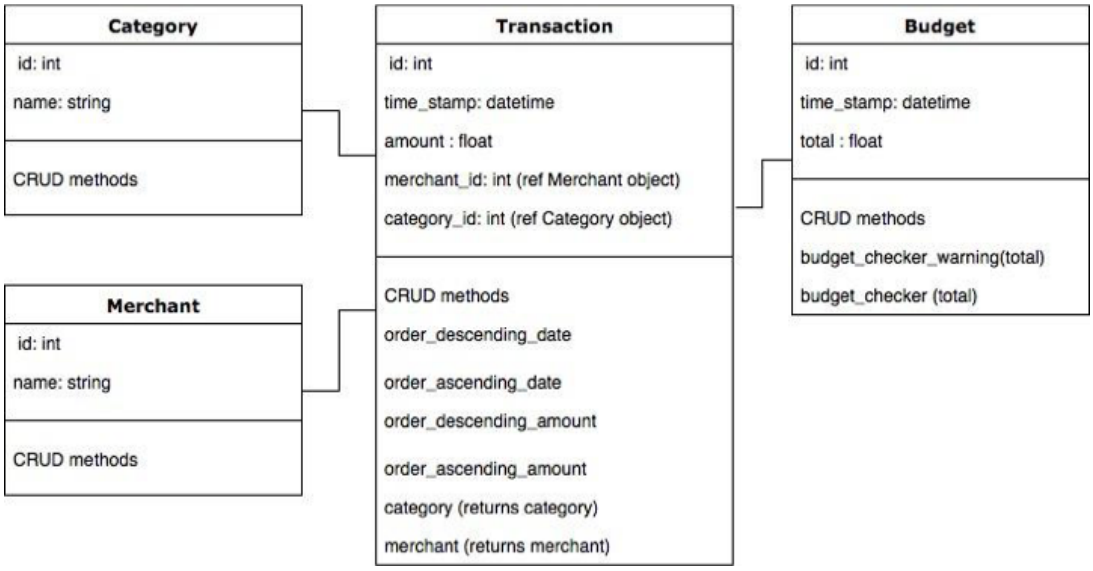


## Week 5 and 6

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



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

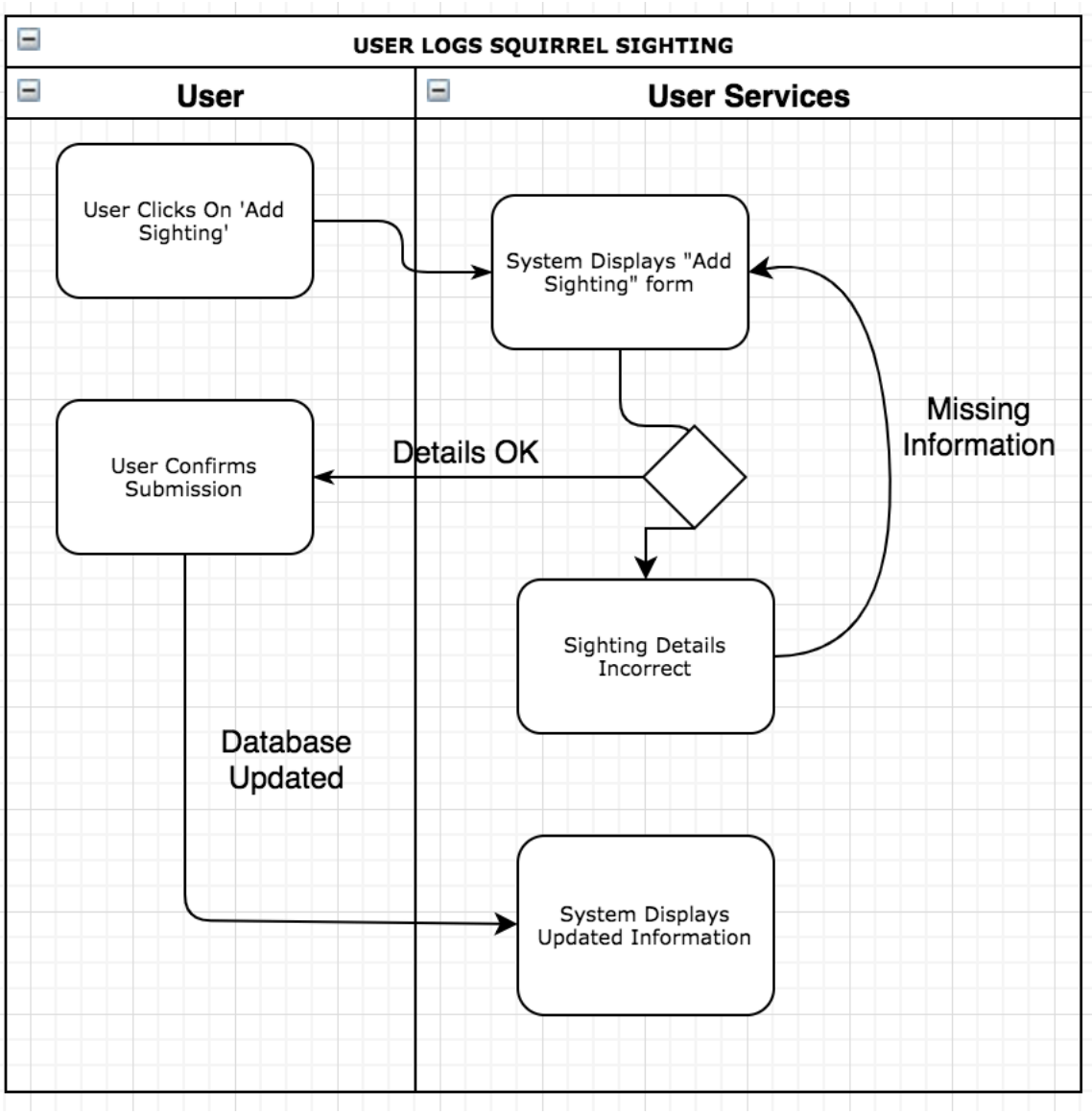


| Unit | Ref | Evidence |
| --- | --- | --- |
| **A&D** | A.D.3 | An Object Diagram |
| | | **Description:** |

| **Transaction: 4** | |
| --- | --- |
| time_stamp: | <DateTime: 2018-09-21T18:37:59+00:00> |
| amount | 17.70 |
| merchant_id | 3 |
| category_id | 3 |

_An example object diagram for a Transaction.

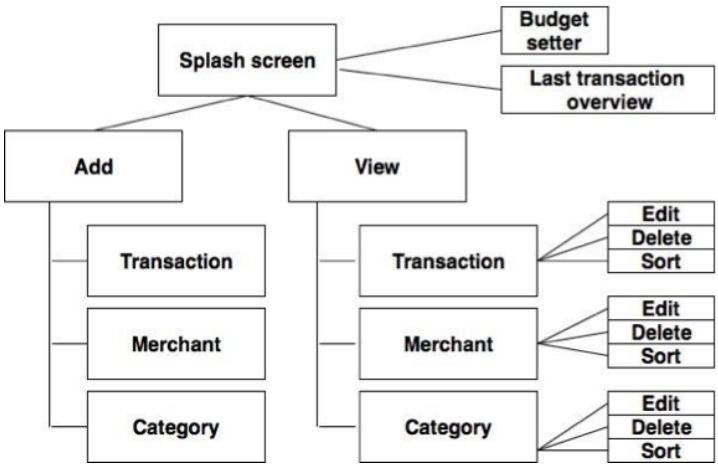
| Unit | Ref | Evidence |
|---|---|---|
| **A&D** | A.D.4 | An Activity Diagram |
| | | **Description: Activity diagram showing data flow of** |



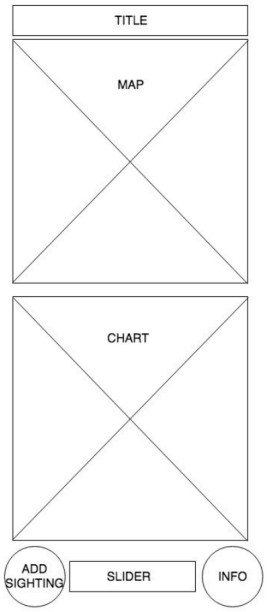
| Unit | Ref | Evidence |
|---|---|---|
| **A&D** | A.D.6 | Produce an Implementations Constraints plan detailing the following factors:<br>*Hardware and software platforms<br>*Performance requirements<br>*Persistent storage and transactions<br>*Usability<br>*Budgets<br>*Time |
| | | **Description:**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

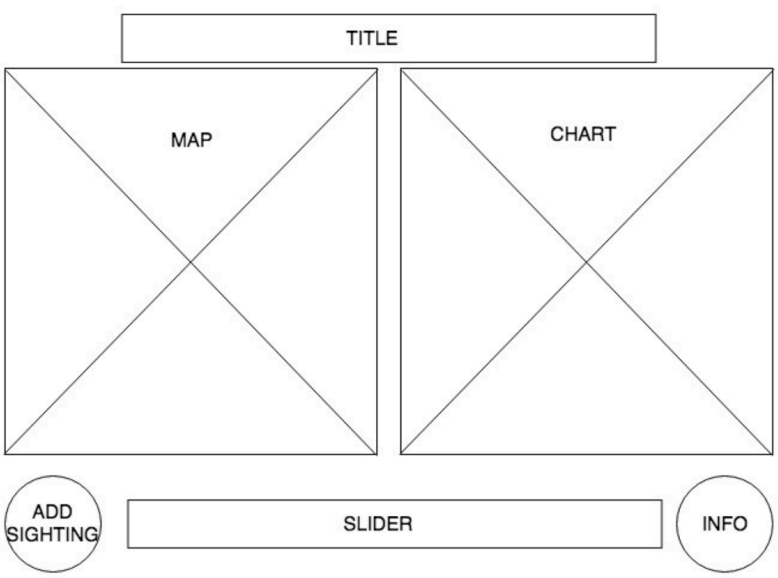
| Topic | Possible Effect On Constraint | Solution |
|---|---|---|
| Hardware and Software Platforms | Compatibility Issues, requirement for speciality or licensed software | Ensure cross-platform software is used, check compatibility on different browsers and operating systems prior to deployment. |
| Performance Requirements | Potential for database to slow with increasing number of entries | Check limits of MongoDB, determine number of potential user inputs. |
| Persistent Storage and transactions | Possibility to run out of local storage space | Ensure that seed data and trial data doesn't use up available space. |
| Usability | Potential for users to misunderstand requirements and enter incorrect information. | Watch users using the site to determine were ambiguity in functionality exists. |
| Budgets | Not a consideration | Not a consideration |
| Time | Possibility of reaching project deadline before MVP is reached. | Ensure proper planning and daily standups to make sure deliverables are on schedule. |

| Unit | Ref | Evidence |
| --- | --- | --- |
| **P** | P.5 | User Site Map |
| | | **Description:** |



| Unit | Ref | Evidence |
|---|---|---|
| **P** | P.6 | 2 Wireframe Diagrams |
| | | **Description:** |



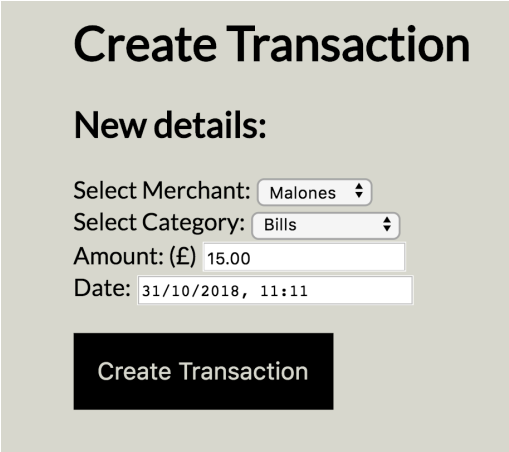

**Left:** Mobile design wireframe diagram
**Right:** Desktop design wireframe diagram

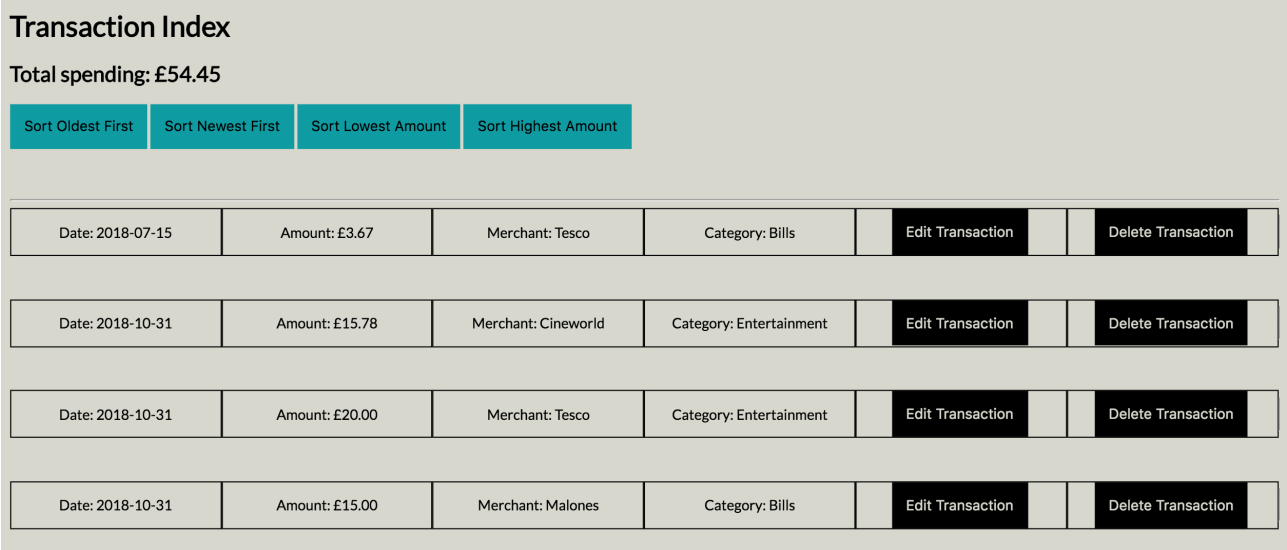
| Unit | Ref | Evidence |
| --- | --- | --- |
| **P** | P.10 | Example of Pseudocode used for a method |
| | | **Description:** |

```
def return_terms_sorted_by_date
  from my_database select everything
  sorted by date (earliest first)
  then
  use this data to construct
  an array of new instances of a particular class
end
```

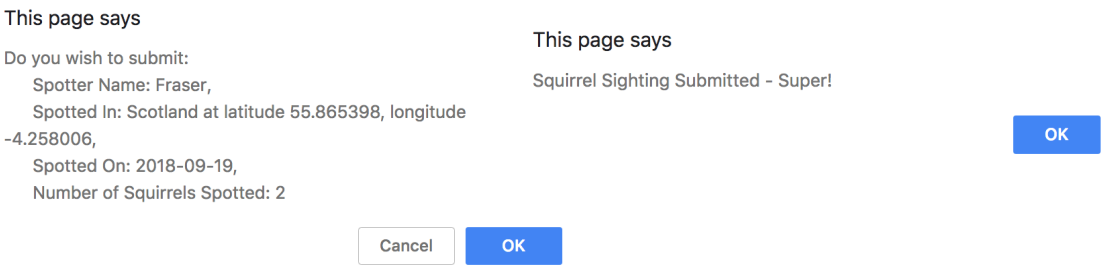
| Unit | Ref | Evidence |
|---|---|---|
| **P** | P.13 | Show user input being processed according to design requirements. Take a screenshot of:<br>* The user inputting something into your program<br>* The user input being saved or used in some way |
| | | **Description:** |

**Top screenshot:** User submitting transaction details into the app.
**Bottom screenshot:** Transaction showing in the app (bottom row)



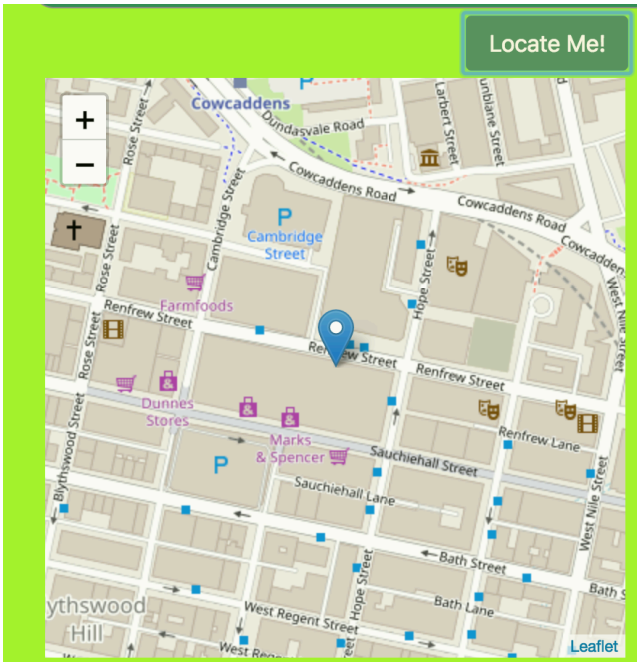
| Unit | Ref | Evidence |
|---|---|---|
| **P** | P.14 | Show an interaction with data persistence. Take a screenshot of:<br>* Data being inputted into your program<br>* Confirmation of the data being saved |
| | | **Description:** |




Left - inputted user data presented for confirmation
Right - confirmation of data being saved

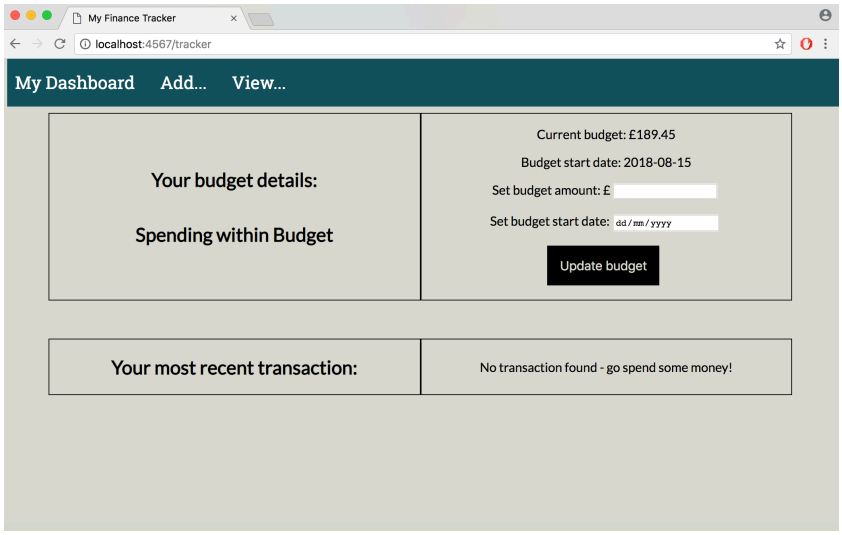
| Unit | Ref | Evidence |
|---|---|---|
| **P** | P.15 | Show the correct output of results and feedback to user. Take a screenshot of:<br>* The user requesting information or an action to be performed<br>* The user request being processed correctly and demonstrated in the program |
| | | **Description:**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

Left: user requesting location via ‘Locate Me!’ Button.
Right: user request being processed correctly to return the location of user

| 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/fddata/w5_ruby_project



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

**Below:** User stories from the planning stage

## User 1

### Name

- Joe

### Demographics

- Student, 22

### Behaviours

- Gets a student loan payment every month
- Wants to budget but doesn't really know how
- Doesn't usually carry cash

### Needs & Goals

- Wants a better way to keep track of his spending
- Doesn't like his bank's mobile app
- Would like to be able to stick to a budget

## User 2

### Name

- Jane

### Demographics

- Working mum with young child, 40

### Behaviours

- Runs the family finances

### Needs & Goals

- Would like a breakdown of categories
- Would like to be able to compare cost of shopping trip at different retailers to get the best deal

**Below:** User needs from the planning stage

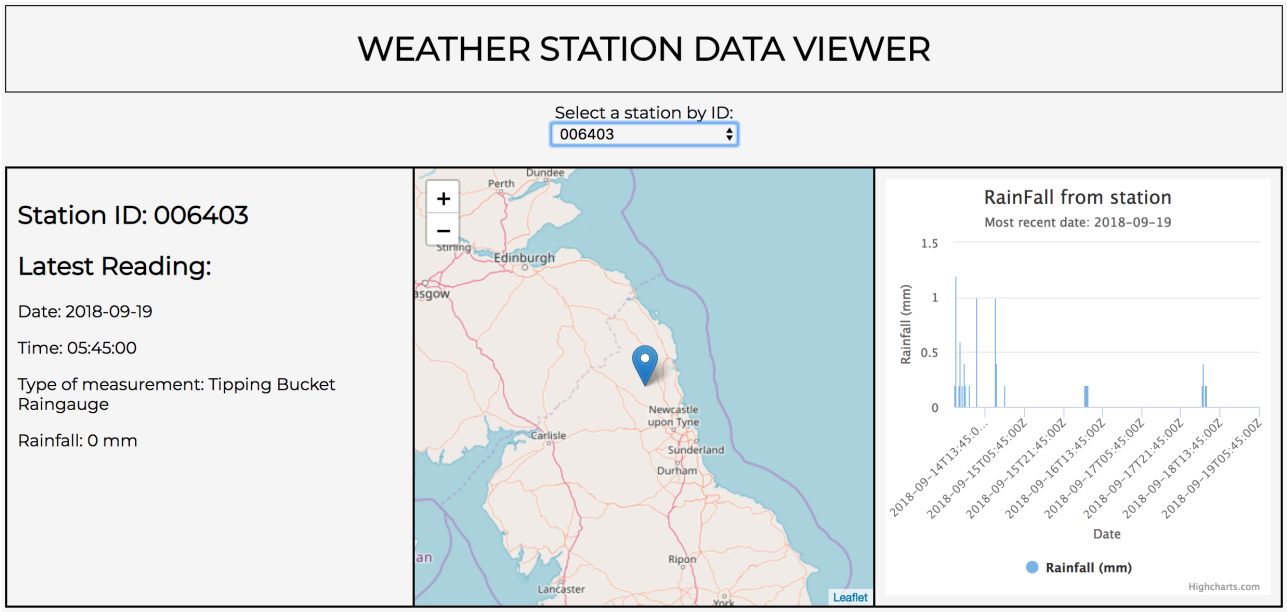
| | As a... | I want to... | So that... |
|---|---|---|---|
| Person 1 | Person on a budget | Know how much I've spent and how much I have left | I don't run out of money before pay day |
| Person 2 | Person who does one large shop a week | Know how much it costs per shop at different retailers | I can get the best value for money |
| Person 3 | Person who does not monitor their spending at all | Have a breakdown of where my money is going by category (groceries, taxis, restaurants etc) | I can identify and quantify bad (and good!) spending habits |

# Week 7

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

```
const selectedStationid = selectedStation.stationReference;
const selectedDataURL =
`http://environment.data.gov.uk/flood-monitoring/id/stations/${selectedStationid}/readings?_sorted`;

const dataRequest = new Request(selectedDataURL);
dataRequest.get()
    .then((data) => {
      PubSub.publish('Stations:selected-station-all-data', data);
      });


});
```

Example of code to 'get' request information on weather stations from UK Government environment website API




Program using API to populate drop down list, return date time reading, latitude and longitude and populate graph.

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

Example of test code:

```
require_relative('card.rb')
class CardGame

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

  dif highest_card(card1 card2)
  if card1.value > card2.value
    return card.name
  else
    card2
  end
end
end

def self.cards_total(cards)
  total
  for card in cards
    total += card.value
    return "You have a total of" + total
  end
end
```

Test code failing to pass:

```
  1) Error:
CardGameTest#test_check_for_ace_false:
NoMethodError: undefined method `value=' for #<Card:0x007f8b1b07db60 @suit=1, @value=2>
Did you mean?  value
    /Users/user/codeclan_work/week_05/day_05/testing_hwk/test_2.rb:11:in `checkforAce'
    specs/cards_spec.rb:22:in `test_check_for_ace_false'

  2) Error:
CardGameTest#test_check_for_ace_true:
NoMethodError: undefined method `value=' for #<Card:0x007f8b1b06eb38 @suit=1, @value=1>
Did you mean?  value
    /Users/user/codeclan_work/week_05/day_05/testing_hwk/test_2.rb:11:in `checkforAce'
    specs/cards_spec.rb:17:in `test_check_for_ace_true'

2 runs, 0 assertions, 0 failures, 2 errors, 0 skips
```

Example of the test code once errors have been corrected:

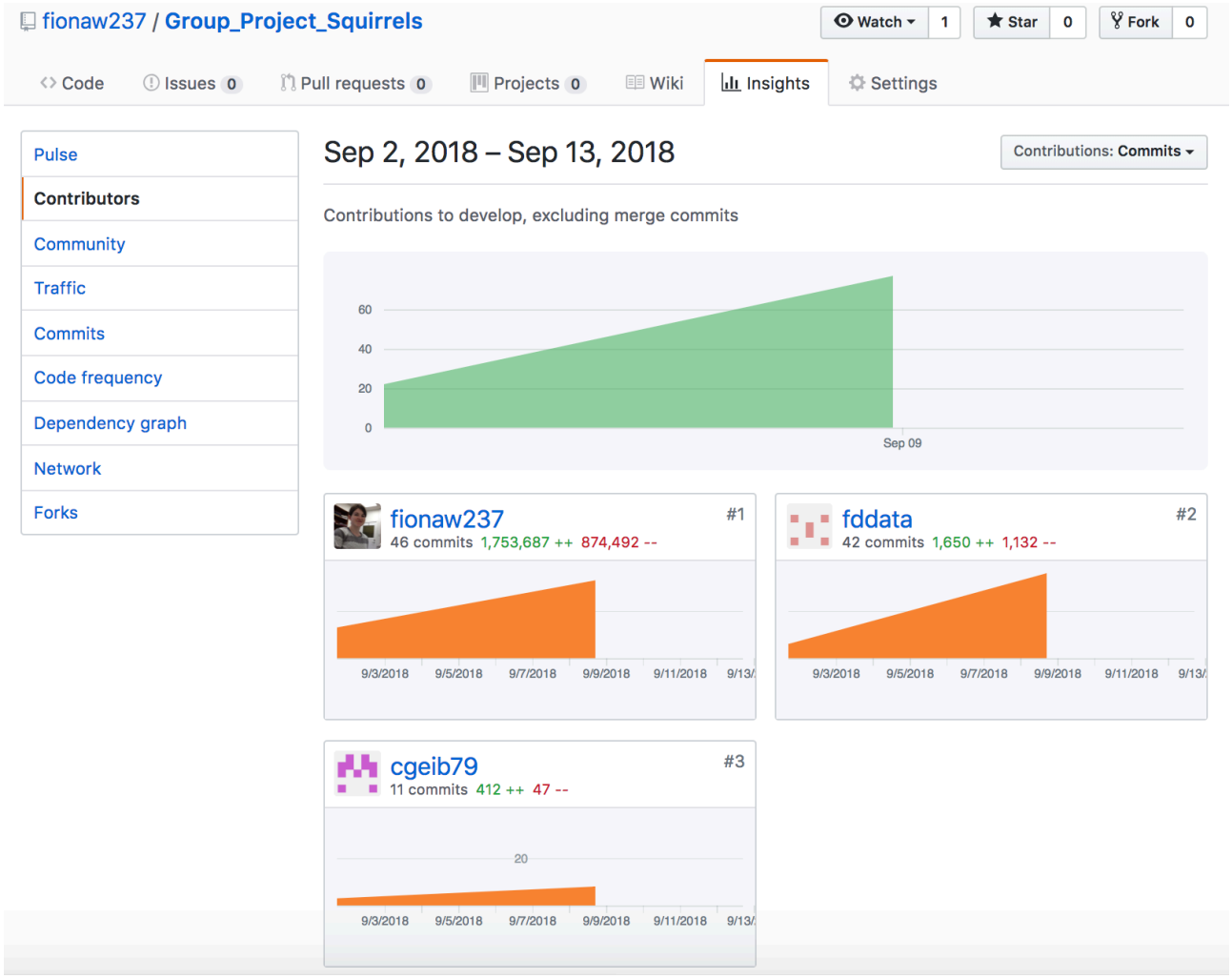
```ruby
require_relative('card.rb')
class CardGame


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

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


  def self.cards_total(cards)
    total = 0
    for card in cards
      total += card.value
    end
    return "You have a total of #{total}"
  end

end
```

Example of the test code passing:

```
# Running:

....

Finished in 0.001388s, 2881.8444 runs/s, 2881.8444 assertions/s.

4 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

# 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:** |



| Unit | Ref | Evidence |
|---|---|---|
| **P** | P.2 | Take a screenshot of the project brief from your group project. |
| | | **Description:** |

# Educational App - Project Brief

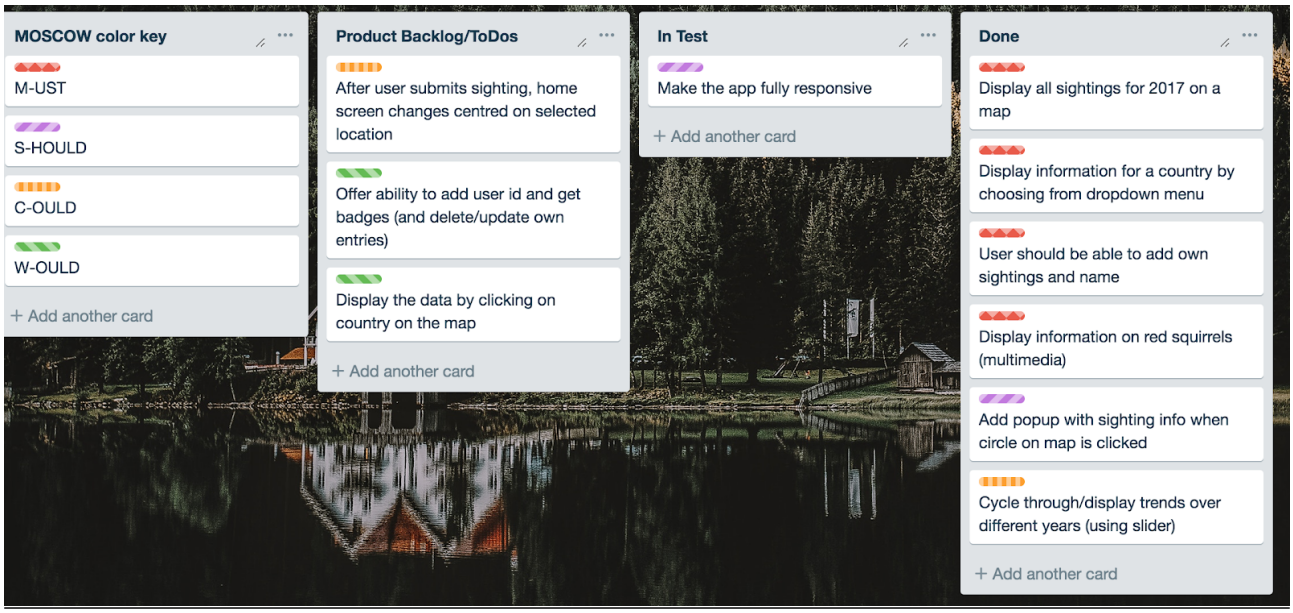
MVP

- Display all sightings for 2017 on a map
- Display information for a country by choosing from dropdown menu
- User should be able to add own sightings and name
- Display information on red squirrels (multimedia)

POSSIBLE EXTENSIONS

- Add popup with sighting info when circle on map is clicked
- Make the app fully responsive
- Cycle through/display trends over different years (using slider)
- After user submits sighting, home screen changes centred on selected location
- Offer ability to add user id and get badges (and delete/update own entries)
- Display the data by clicking on country on the map

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

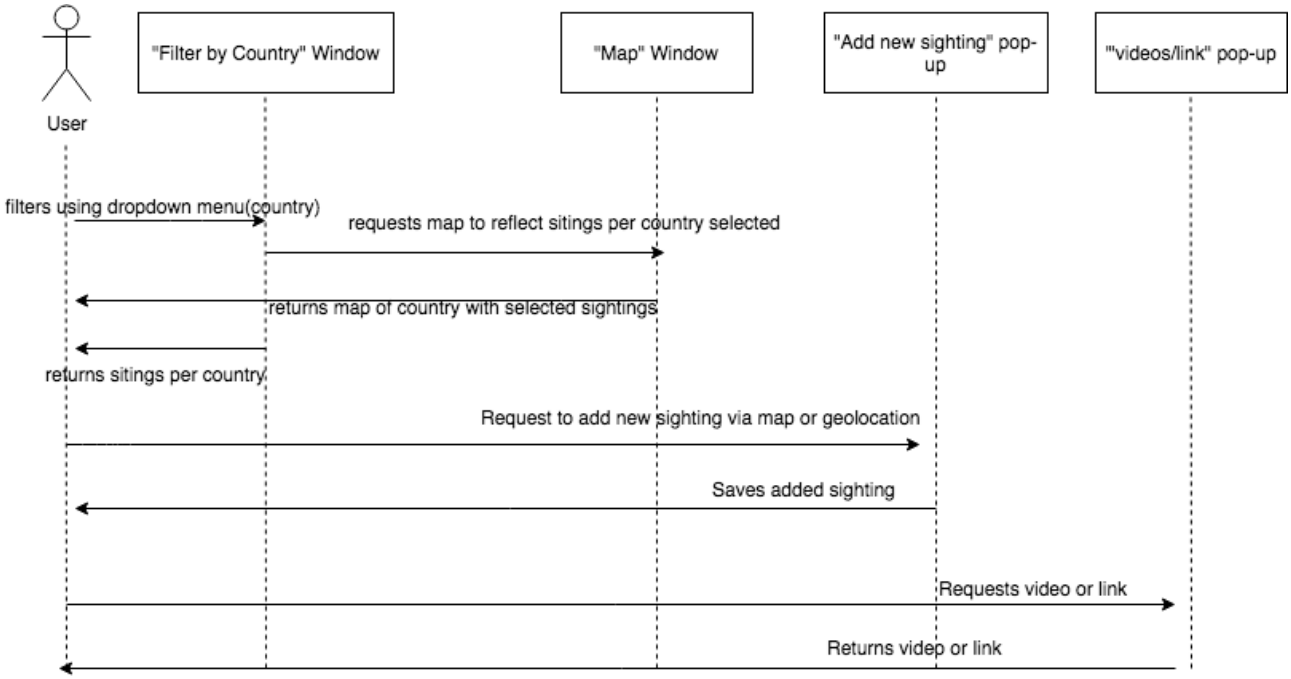


| Unit | Ref | Evidence |
| --- | --- | --- |
| **P** | P.4 | Write an acceptance criteria and test plan. |
| | | |

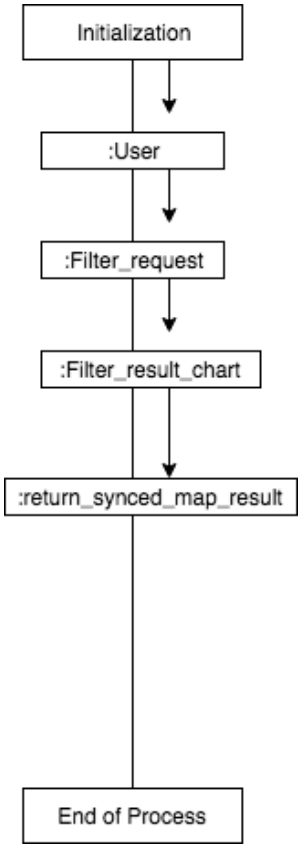
| **Acceptance Criteria** | **Test** |
| --- | --- |
| The user should be able to view all transactions | Seed database with a known number of transactions, confirm they are visible upon clicking on ‘View All Transactions’ |
| The user should be able to sort transactions by date | Seed database with known transactions, confirm that ‘Sort By Date’ (ascending/descending) buttons are working in the ‘View All Transactions’ page. |
| The user should be able to sort transactions by amount | Seed database with known transactions, confirm that ‘Sort By Amount’ (ascending/descending) buttons are working in the ‘View All Transactions’ page. |
| The user should be able to delete a transaction. | Confirm that transactions can be deleted using the ‘Delete Transaction’ button. |

| Unit | Ref | Evidence |
|---|---|---|
| **P** | P.7 | Produce two system interaction diagrams (sequence and/or collaboration diagrams). |
| | | **Description:** |

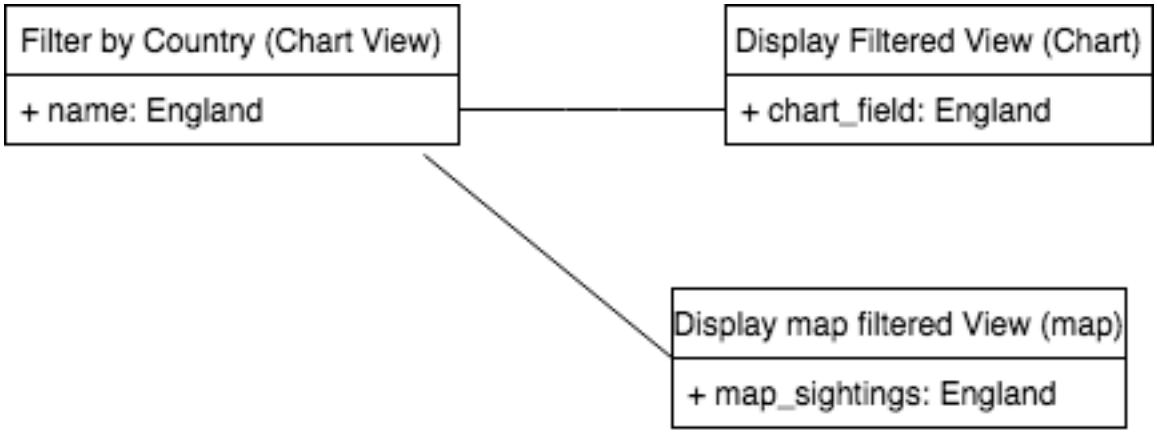
**Sequence diagram:**

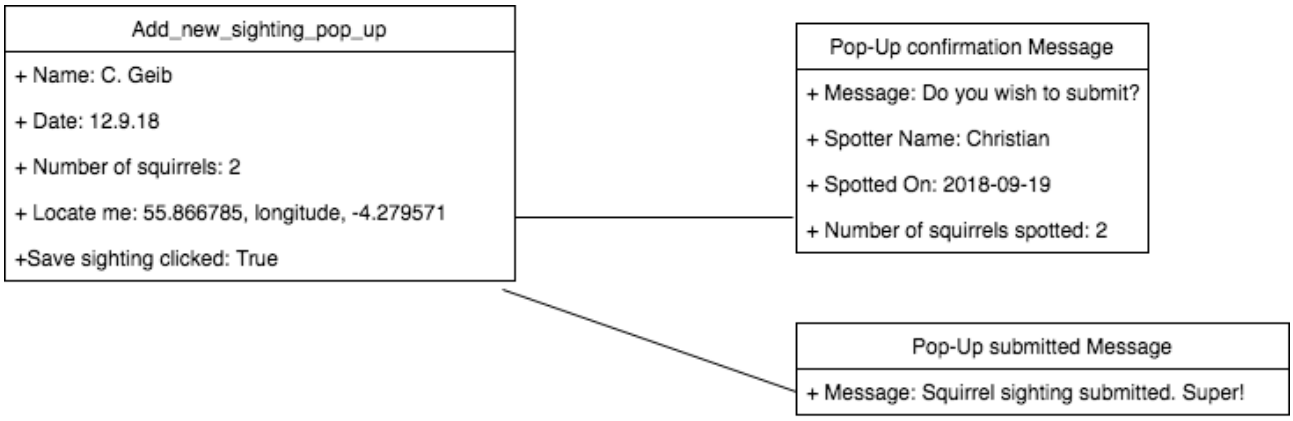



**Collaboration diagram:**



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



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

## Example bug tracking report:

| 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**: Polymorphism occurs when an object can take on many forms. In the example below, we see that an instance of the abstract 'Instrument' class implements the ISell and IPlay interfaces, meaning that it has to feature a 'play' method and 'calculateMarkup'. |

**Below:** Example of polymorphism: an instance of the Instrument class is also considered to be an ISell and an IPlay object too.

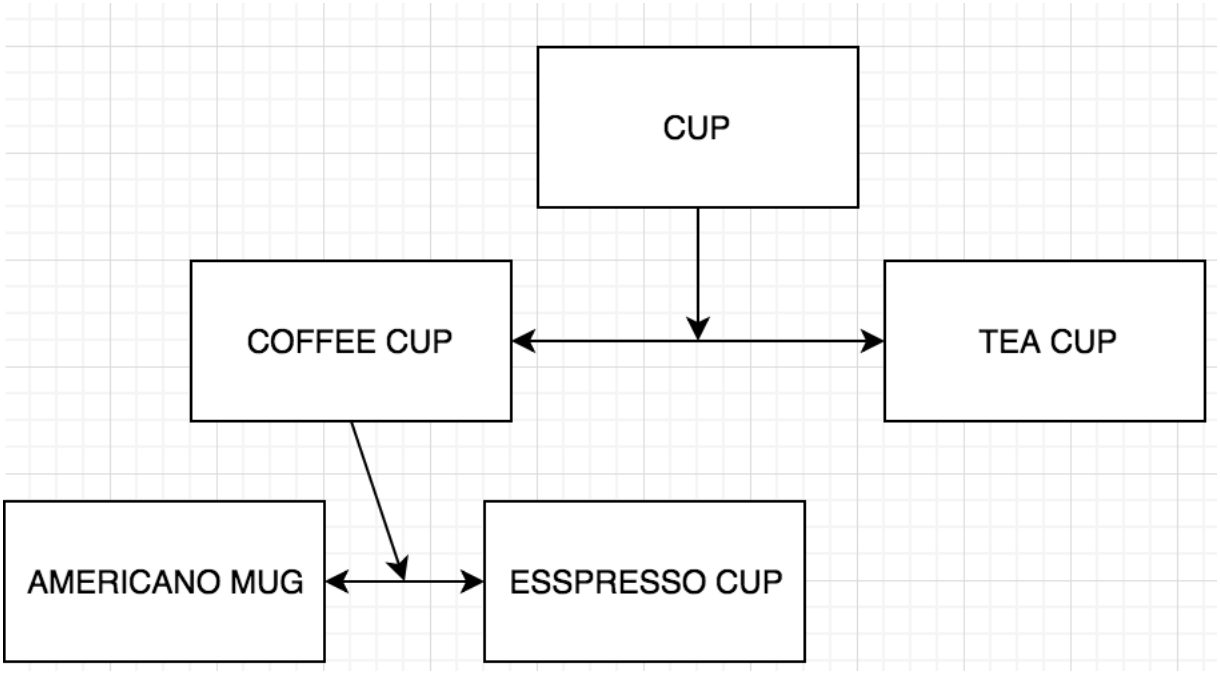
```
public abstract class Instrument implements ISell, IPlay {
    private InstrumentColour colour;
    private String name;
    private String brand;
    private double wholesalePrice;
    private double retailPrice;

    public Instrument(InstrumentColour colour, String name,
        this.colour = colour;
        this.name = name;
        this.brand = brand;
        this.wholesalePrice = wholesalePrice;
        this.retailPrice = retailPrice;
    }
```

```
public interface IPlay {

    String play();
}
```

```
public interface ISell {

    double calculateMarkup();
}
```

| Unit | Ref | Evidence |
|---|---|---|
| **A&D** | A.D.5 | An Inheritance Diagram |
| | | **Description:** |

**Below:** Diagram showing inheritance from parent classes, showing how specificity increases with each generation.



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

**Below:** 'Encapsulation' in object oriented programming refers to making the properties of a class hidden to other classes, and only making available the information that we wish. For example, in the screenshot below we can see that the name, niNumber and salary properties of the Employee class are private by default, and we can use the public getName method to allow access to the private name property.

```
public abstract class Employee {

    private String name;
    private String niNumber;
    private double salary;

    public Employee(String name, String niNumber, double salary) {
        this.name = name;
        this.niNumber = niNumber;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }
```

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

**Below:** An example of the abstract 'employee' class in a staff management system

```
public abstract class Employee {

    private String name;
    private String niNumber;
    private double salary;

    public Employee(String name, String niNumber, double salary) {
        this.name = name;
        this.niNumber = niNumber;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        if (name != null && name != "") {
            this.name = name;
        }
    }

    public String getNiNumber() {...}

    public double getSalary() {...}

    public void raiseSalary(double increment) {
        if (increment > 0) {
            this.salary *= increment;
        }
    }

        public double payBonus () {
            return this.salary * 0.01;
        }
    }
```

**Below:** An example of the 'Manager' class inheriting from the 'Employee' class and extending to include a deptName property.

```
public class Manager extends Employee {

    private String deptName;

    public Manager(String name, String niNumber, double salary, String deptName) {
        super(name, niNumber, salary);
        this.deptName = deptName;
    }

    public String getDeptName() {
        return deptName;
    }
}
```

**Below:** An object in the inherited class - a new instance of the 'Manager' class, showing the inherited Employee parameters plus the new 'deptName' parameter.

```
public void setUp() throws Exception {
    manager = new Manager( name: "Bob",  niNumber: "ABC123",  salary: 30000,  deptName: "IT");
}
```

**Below:** A method using the information inherited from the other class: example shows the Manager class using the Employee getName method.

```
@Test
public void getName() {
    assertEquals( expected: "Bob", manager.getName());
}
```

| 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:** |

**ALGORITHM 1**

```
def multiple_of_3_and_5():
        for n in xrange(1000):
            if not n % 3 or not n % 5:
                yield n

print sum(multiple_of_3_and_5())
```

```
➜  Desktop python multiples.py
233168
```

**Top:** Algorithm written in python to sum the multiples of 3 and 5 for all numbers under 1000. I chose to use this algorithm to practice logic and mathematical programming.
**Bottom**: result of running the algorithm, returning 233,168.

**ALGORITHM 2**

```
def fibonacci(limit):
    nums = [0,1]
    while len(nums) < limit:
        nums.append(nums[-1] + nums[-2])
    return nums

print fibonacci(10)
```

```
➜  Desktop python fibonacci.py
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

**Top:** Algorithm written in python to print out the Fibonacci sequence. I chose to use this algorithm to practice iterating and also to use as a basis for building more complicated algorithms.
**Bottom**: result of running the algorithm, showing the first 10 numbers of the fibonacci sequence.