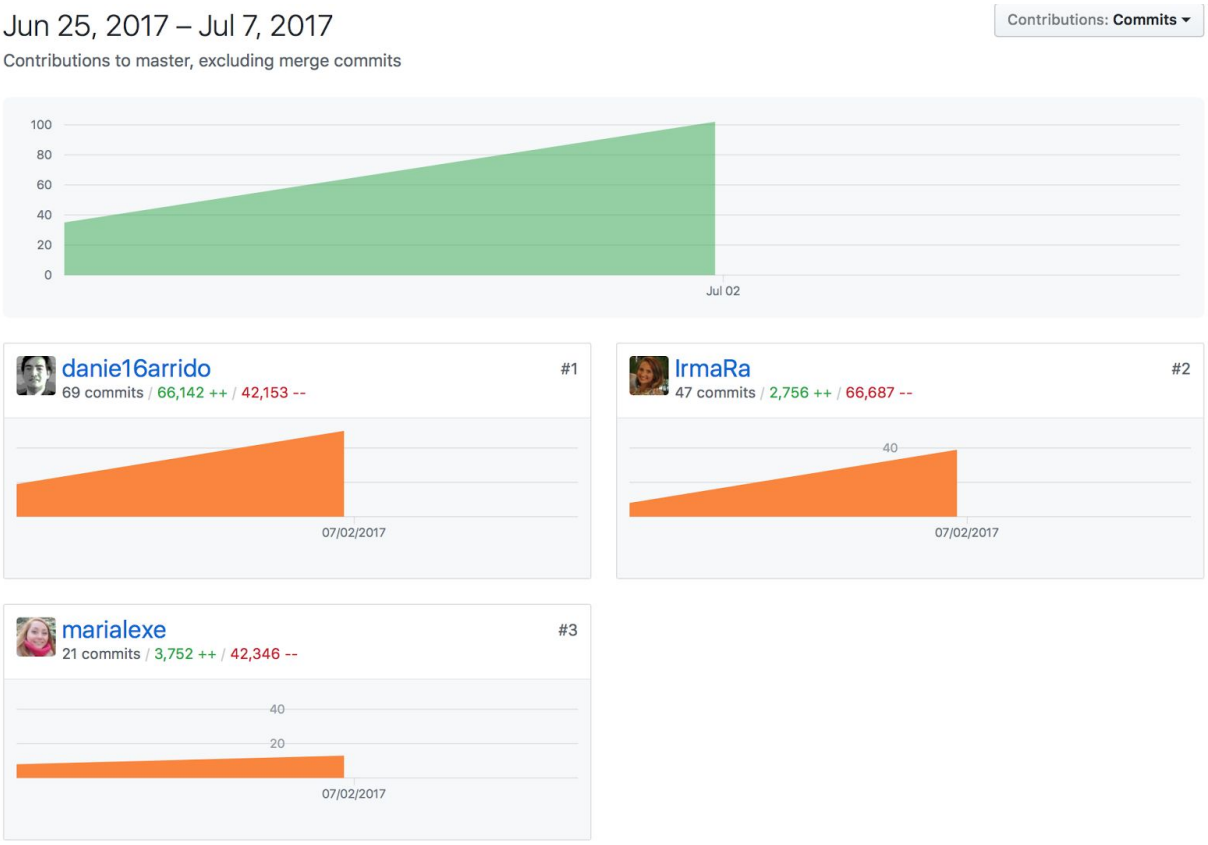
PDA Software Development: Project Unit part 1

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



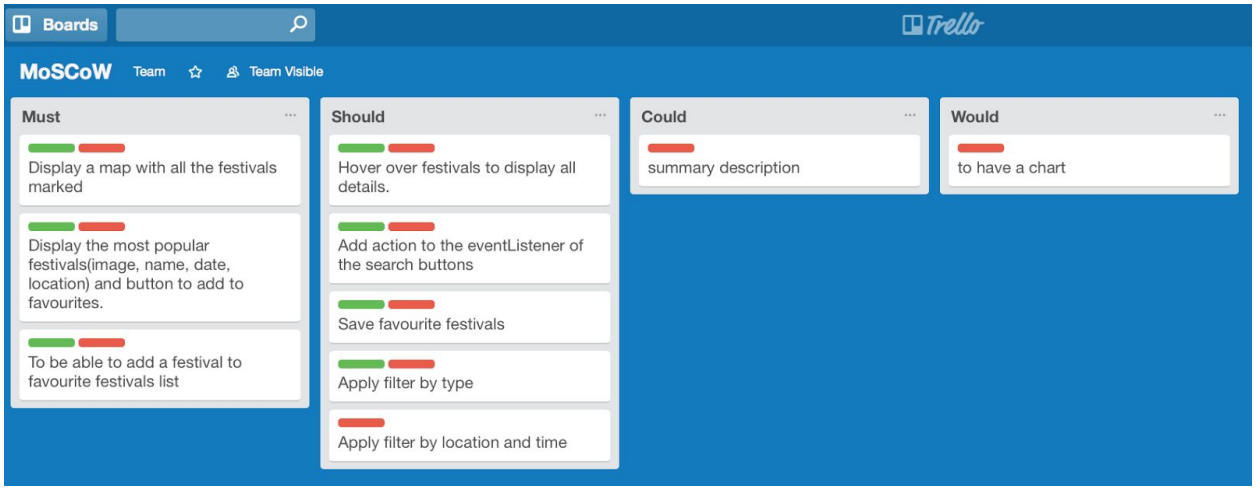
P2 Write a project brief which outlines the requirements of the program(project group)

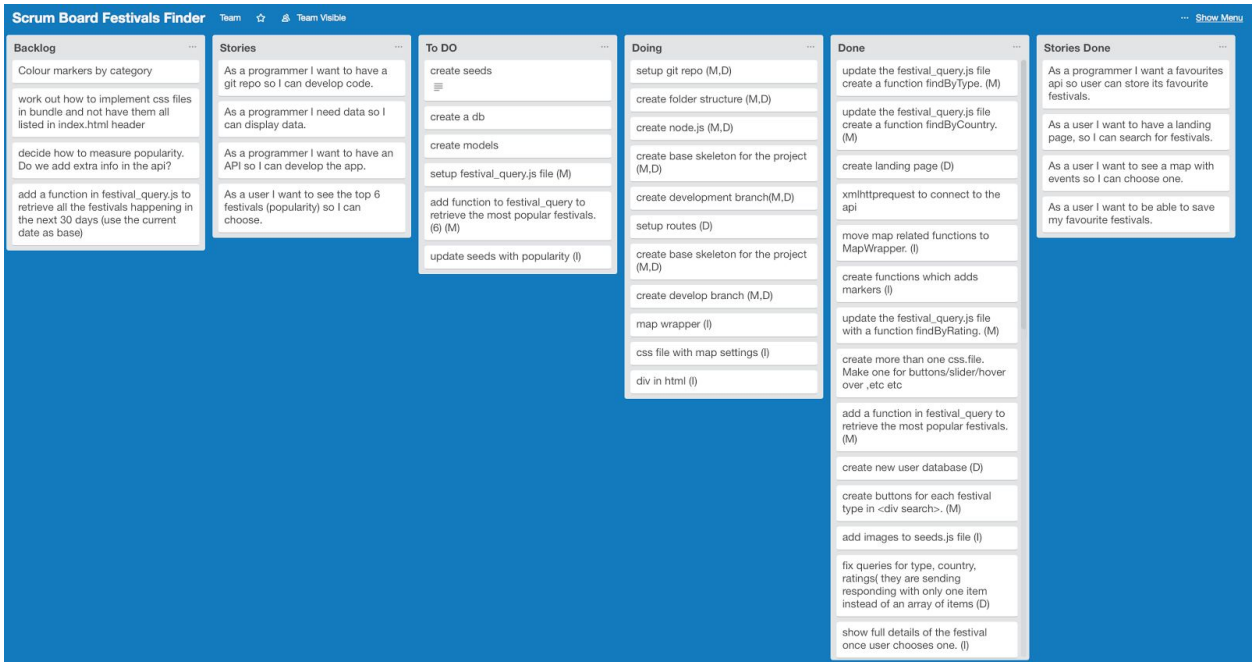
Brief:
Festivals fans want to be able to view relevant festivals events on a dashboard.
Use and create a new API to display information about festivals,
news and travel information for events.

MVP

Display upcoming events on a map
Display results and ranking of best/Festivals
Allow users to add festivals to a favourites list.

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

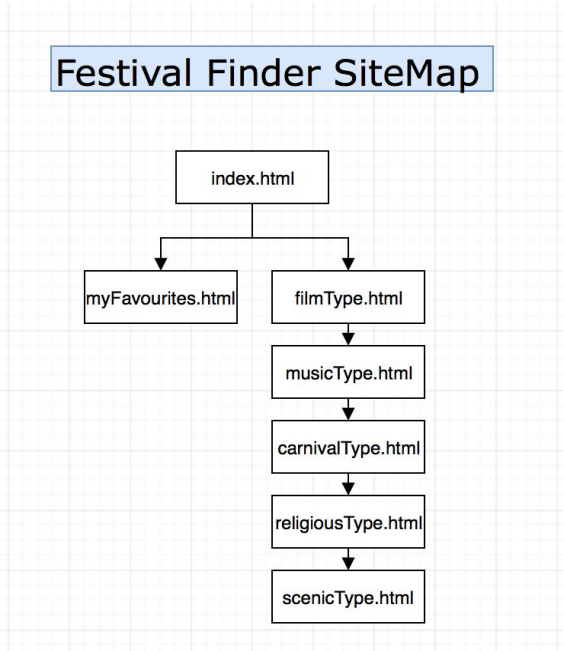




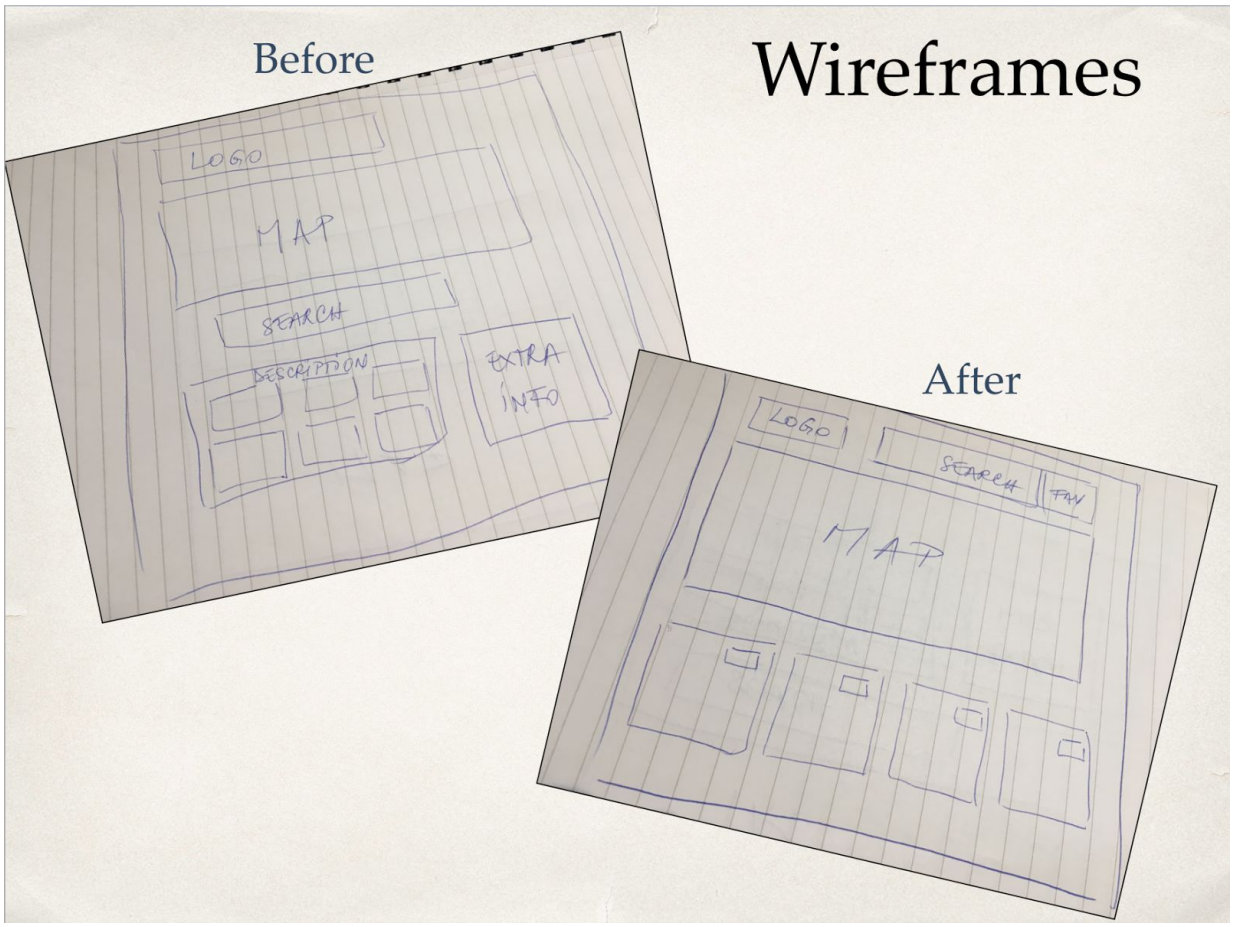
P4 Write an acceptance criteria and test plan.

Acceptance Criteria	Expected Output/Result	Pass/Fail
User is able to see all festivals in a map	When opening the landing page a map must be displayed with markers to all the festivals happening from now on.	Pass
User is able to filter festivals by type	When user clicks on one of the menu categories the data display area lists all the festivals of the menu type	Pass
User is able to get more info of a festival	When user hovers over the festival component it automatically displays date about the data and more info of selected festival	Pass
User is able to save festivals to favourites	When user hovers over the festival component it automatically displays a button to save said festival being able to later click on my Favourites button and see all the saved festivals	Pass
User is able to delete festival from favourites	When user is his/her favourites ara he/she should be able see the saved festivals and click on the delete button and have that item deleted from favourites.	Pass

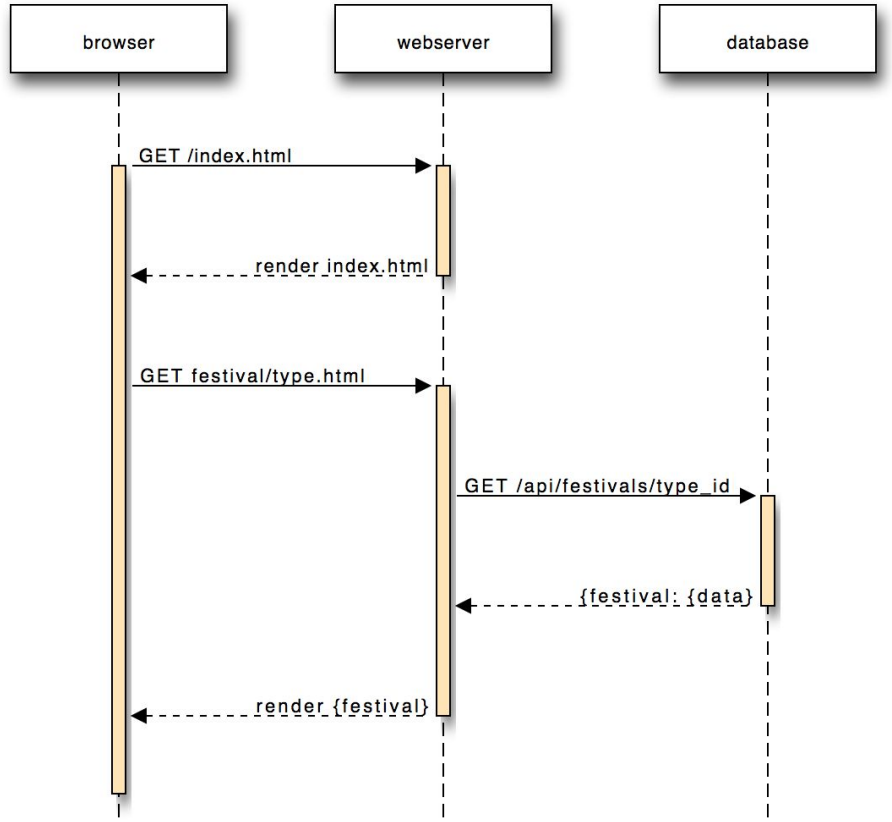
P 5 Create a sitemap for the project

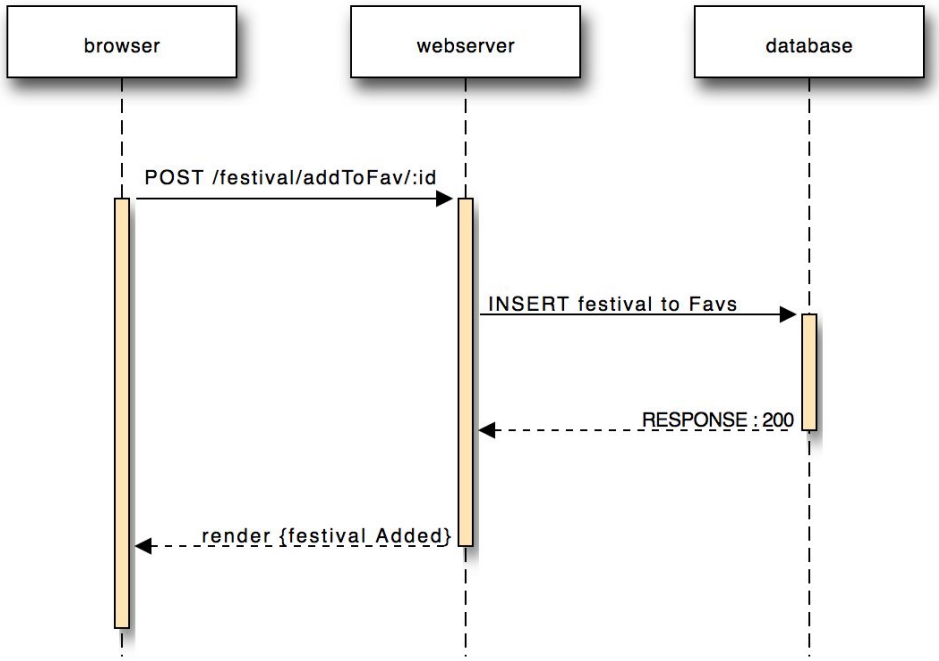


P 6 Produce two wire frame designs showing user interface and interactions, what you have hoped it would look like.

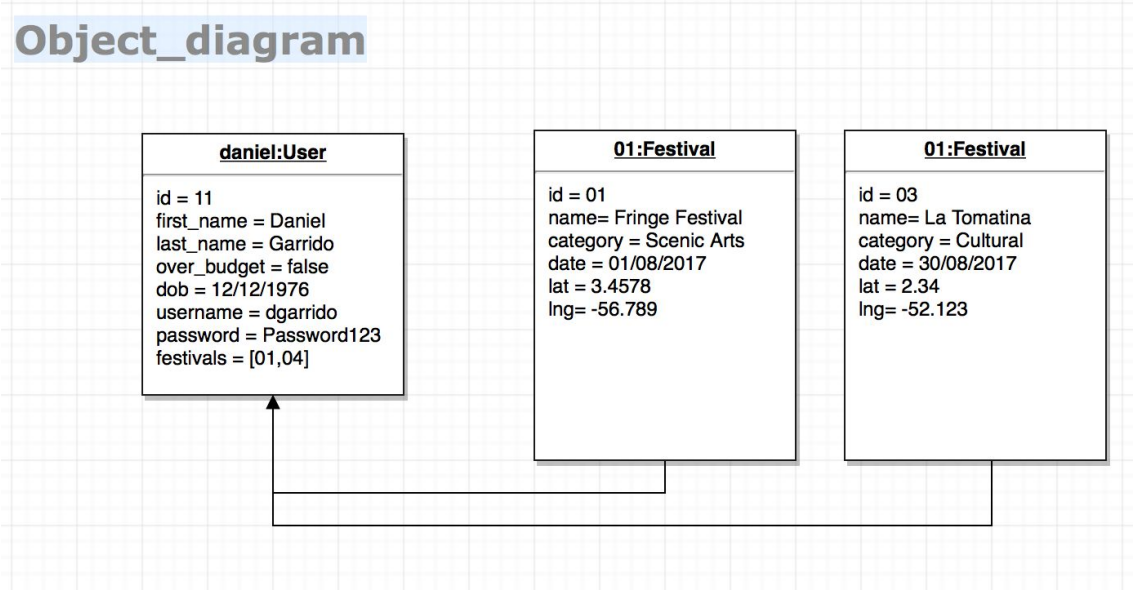


P 6 Produce two system interaction diagrams(sequence and/or collaboration diagrams)





P 8 Produce object diagrams



P 9 Select two algorithms you have written. Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.

Game.java

```
47            pHand.addIcard(this.deck.next());
48          }
49        }
50      }
51
52      public int handPlayerSum(Player player){
53        ArrayList<ICard> pHand = player.getHand().getCards();
54        int sum = 0;
55        int cardRank;
56        int aces = 0;
57        for (ICard card: pHand){
58          cardRank = card.getRank();
59          if (cardRank == 1){ //Ace card
60            aces++;
61            sum += 11;
62          } else if (cardRank > 10){ //face card
63            sum += 10;
64          } else {
65            sum += cardRank;
66          }
67        }
68        while (sum > 21 && aces > 0){
69          sum -= 10;
70          aces--;
71        }
72        return sum;
73      }
74
75      private int checkBlackJackState(Player player){
76        int playerHandSum = handPlayerSum(player);
77        if( playerHandSum == 21){
```

HandPlayerSum() is a function that describes an algorithm that sums the cards in the player's hand by initially adding the value of the card and then checking if the hand has aces and the total value is more than 21 then and only then the total hand's value will be subtracted 10 for each existing ace.

```
        this.addMarker( loc, crime.category );
        loc = "";
      }
      this.updateCategories()
    },

    circlePath: function(center, radius, points){
      var newPolygon=[],p=360/points,d=0;
      for(var i=0;i<points;++i,d+=p){
          newPolygon.push(google.maps.geometry.spherical.computeOffset(center,radius,d));
      }
      return newPolygon;
    },

    createRequestAdress: function( polygon ) {
      var requestAddress = "https://data.police.uk/api/crimes-street/"+ this.selectedCategory + "?";
      requestAddress += "poly="
```

CirclePath() is a function that helps to generate a polygon that simulate a circle based on a center(lat, lng in googlemaps API), radius(m) and the number of points needed.
I needed it to query my API with a polygon instead of an actual circle.

P 10 Take a screenshot of an example of pseudocode for a function.

```
1   //Blackjack game logic
2   //ToDo
3   Create a deck of cards
4   Shuffle a deck of cards
5   Create user's Hand
6   Create dealer's Hand
7   Deal two cards to each hand
8   Check if dealer has got blackjack (if so, game ends)
9   Check if user has got blackjack (if so, game ends)
10  User draws cards (if user goes over 21, game ends)
11  Dealer draws cards
12  Check for winner
```

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

CrimeSpottinghttps

An app to track crimes within a choosen area and filter it by data and type of crime.

- GoogleMaps API
- data.police.uk API

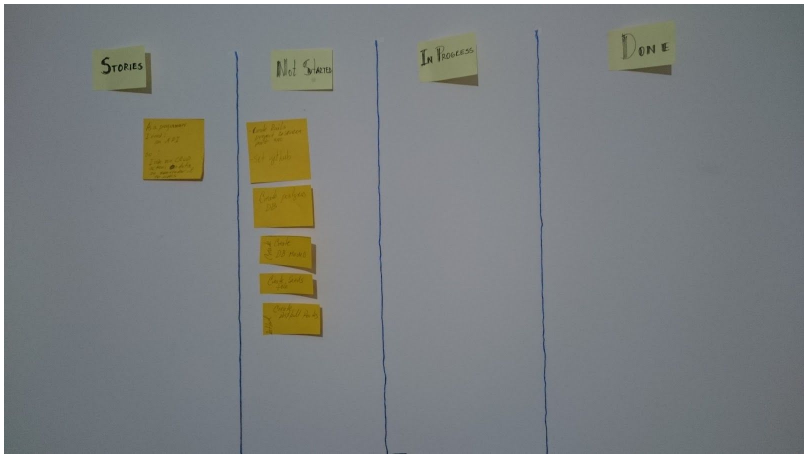
Key learning to practise and/or revise:

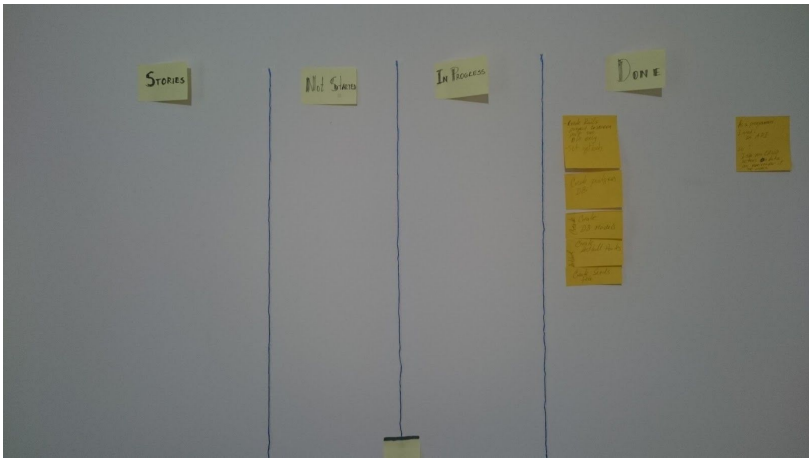
- Manipulating the DOM - selecting/creating/appending elements
- Using event listeners to make our apps interactive
- Making an HTTP request to retrieve some data and then using it

https://github.com/danie16arrido/CrimeSpotting

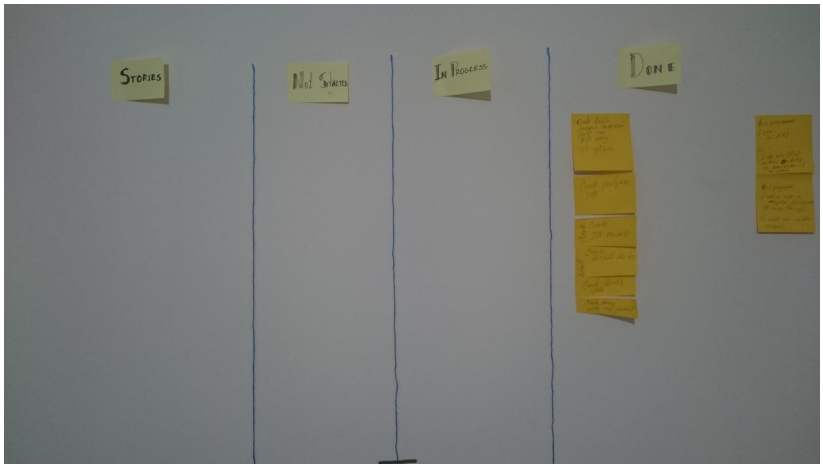
P12 Take screenshots or photos of your planning and the different stages of development to show changes

Agile(Scrum)




Stories
Not Started
In Progress
Done

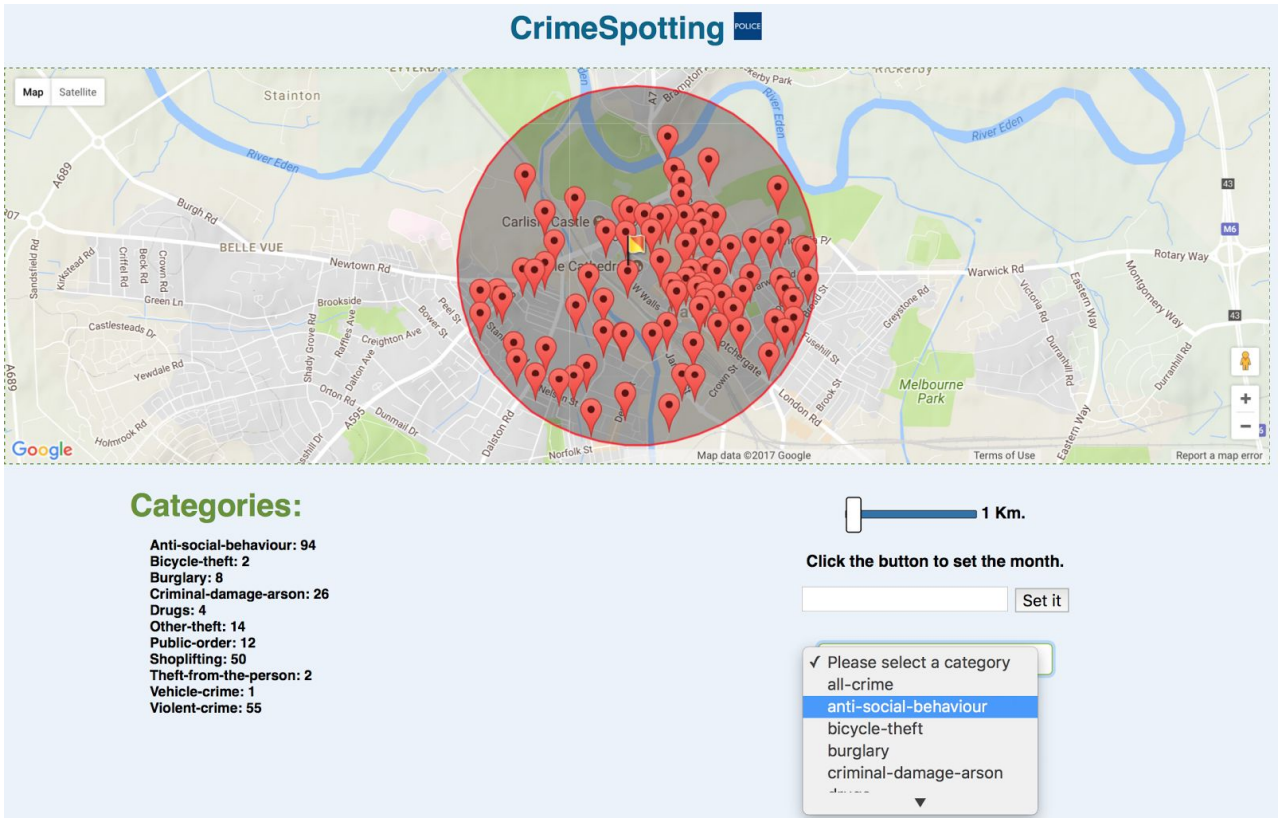

Stories
Not Started
In Progress
Done


Stories
Not Started
In Progress
Done

P 13 Show user input being processed according to design requirements. Take a screenshot of:

- **The user inputting something into your program**

User selects a filter(bicycle-theft)



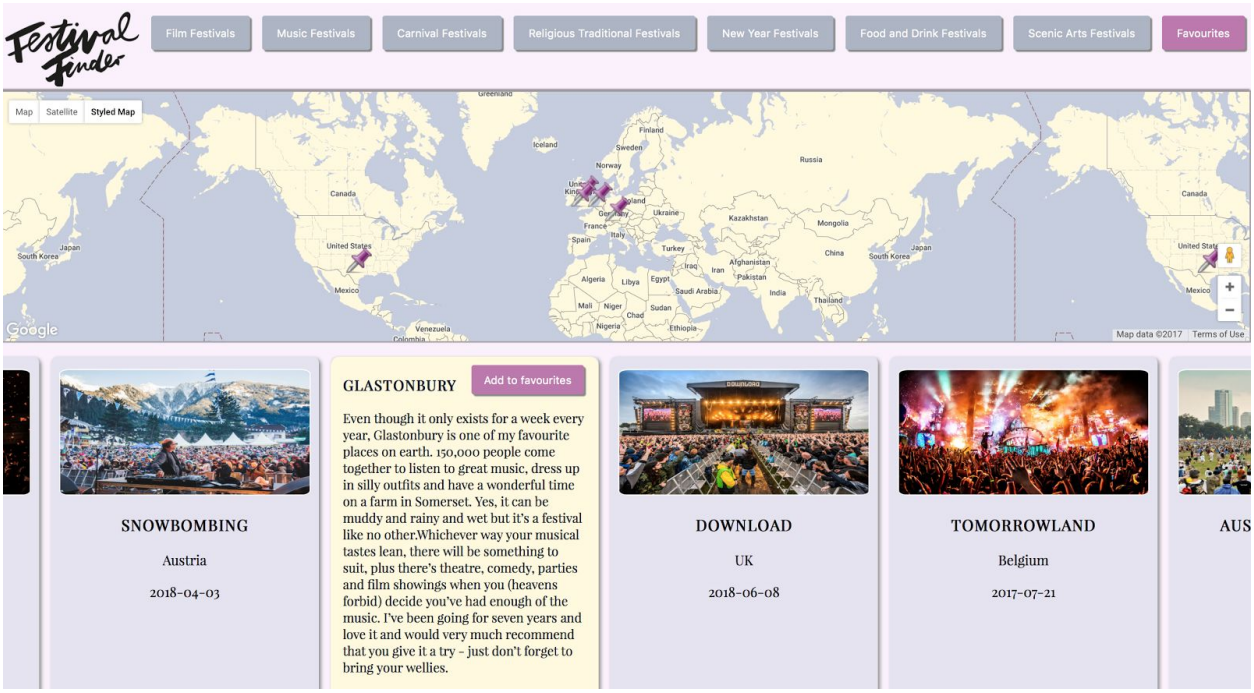
- **The user input being saved or used in some way**

User selection saved and query send to the API which populates the map with the data.

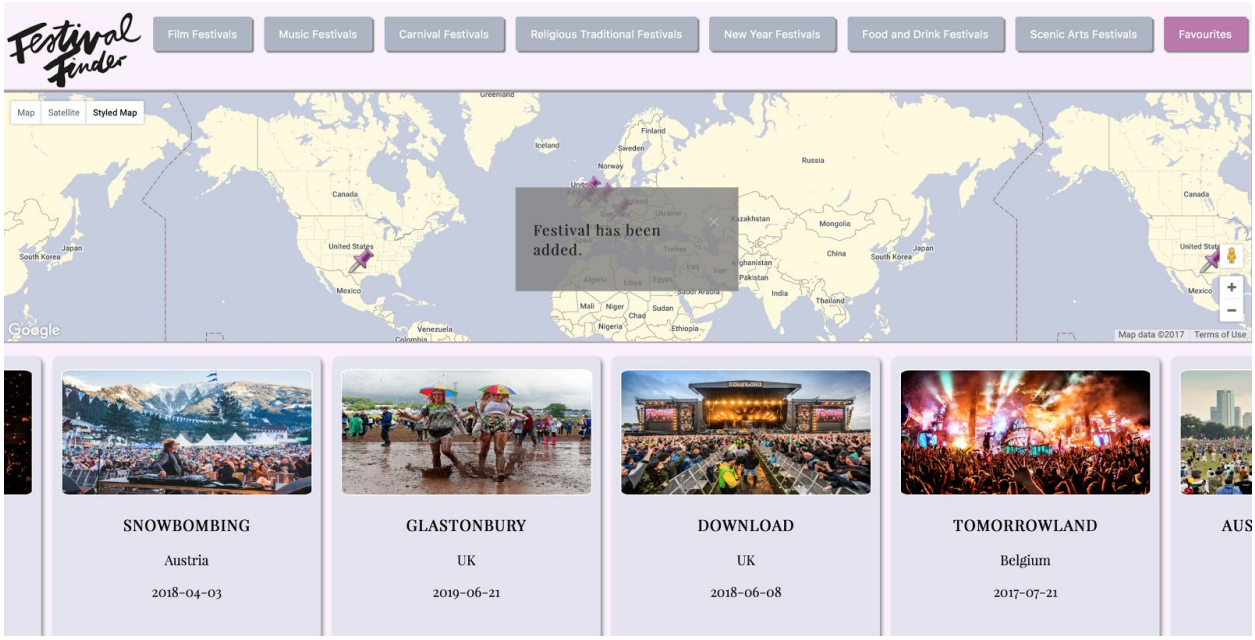
P 14 Show an interaction with data persistence. Take a screenshot of:

- **Data being inputted into your program**

User saving a festival

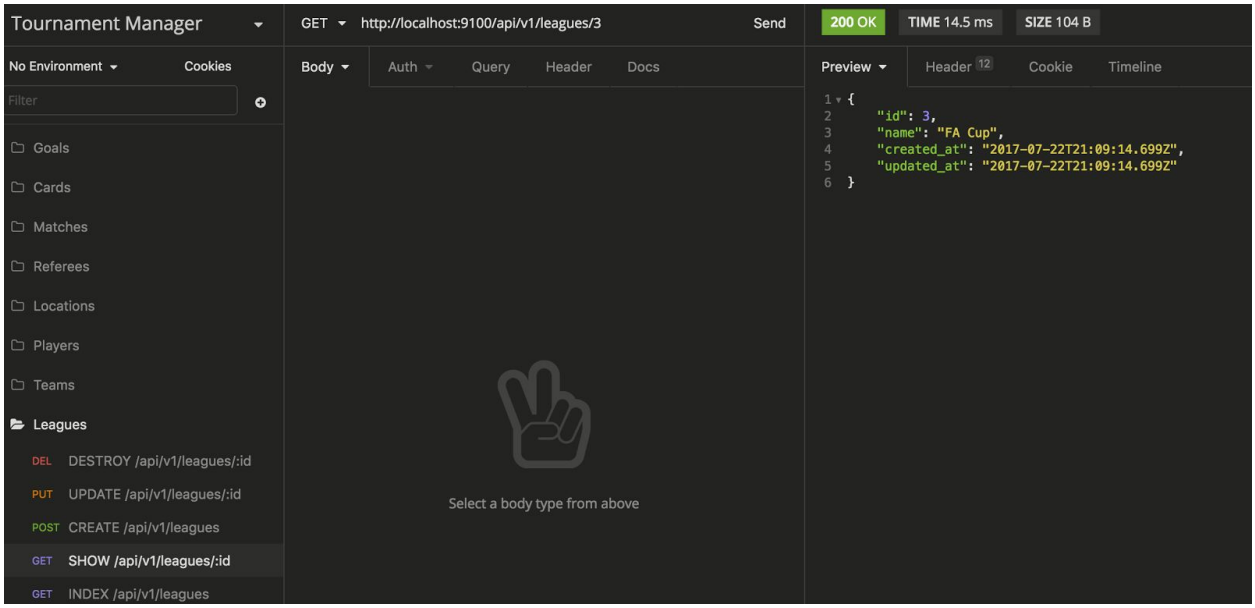


- **Confirmation of the data being saved**



P 15 Show the correct output of results and feedback to user. Take a screenshot of:

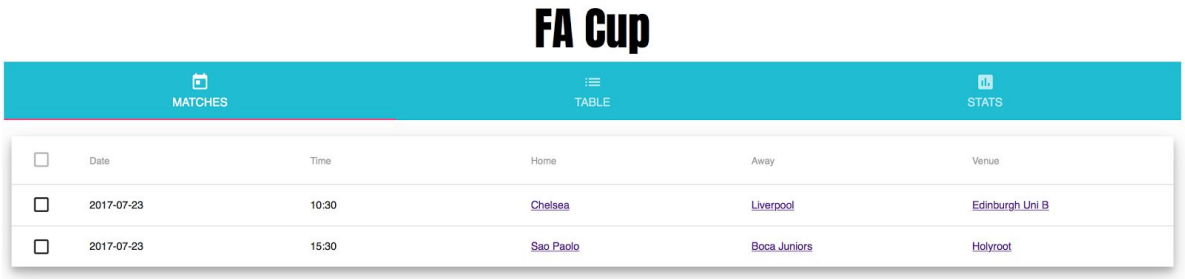
- **The user requesting information or an action to be performed**
  User requesting data about league with ID=3
- **The user request being processed correctly and demonstrated in the program**
  API responding with the data for league with ID=3



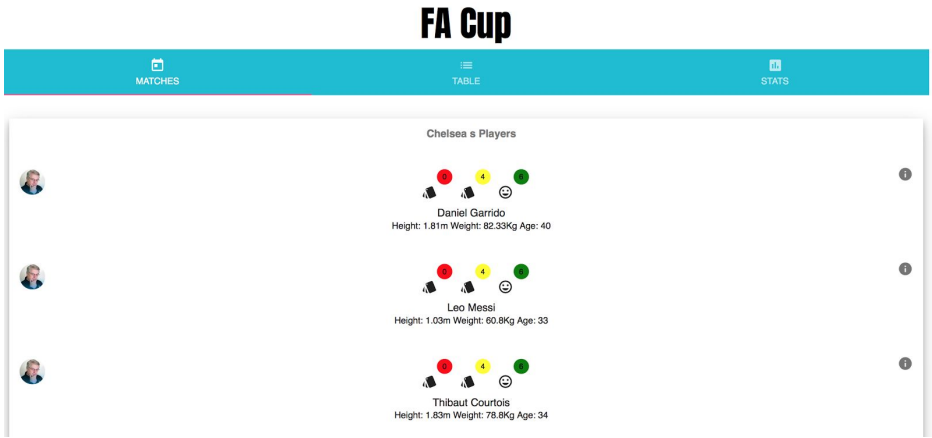
P 16 Show an API being used within your program. Take a screenshot of:● The code that uses or implements the API● The API being used by the program whilst running

```
import axios from 'axios'
const apiServer = "http://localhost:9100/api/v1"
const leagueID = 11
const getApiLeagueName = leagueID => {
  return dispatch => {
    return axios.get( apiServer + "/leagues/" + leagueID)
      .then( response => {
        dispatch(
          {
          type: "API_LEAGUE_NAME",
          league_name: response.data.name
          }
        )
      }
    )
  }
}
```

User requesting details for Chelsea team



User getting results for Chelsea team after the front end server querying the API server and rendering data.



P 17 Produce a bug tracking report

Description	Status	Solution	Status
User can not get results when filtering by “New Year” category	FAILED	In the database field name changed from “newyear” to “new_year” for easy parsing.	PASSED
Map not being updated with the selected category	FAILED	Added a method that updates the map with new results when rendering.	PASSED
User can not save festivals	FAILED	Added user ID to the API query.	PASSED
Data of festival not being displayed on hover over.	FAILED	#festival-container class edited in main.css file to be able to handle hover over.	PASSED
Logo image link not sending back user to home page	FAILED	Added link to the homepage to the logo image.	PASSED

P18 Demonstrate testing in your program . Take screenshots of:

- **Example of test code**

Welcome | record_store_spec.js

```
var assert = require('assert');
var Record = require('../record.js');
var RecordStore = require('../record_store.js');

describe("RecordStore", function(){
  var record;
  var recordStore;

  beforeEach(function(){
    record = new Record("test", "x&y", "chillout", 9.99);
    record1 = new Record("test2", "x&y&z", "chillout-more", 19.99);
    recordStore = new RecordStore("dans record store", "Gotham city");
  });

  it("should have a name", function(){
    assert.strictEqual(recordStore.name, "dans record store");
  });

  it("should have a city", function () {
    assert.strictEqual(recordStore.city, "Gotham city")
  });

  it("should have an inventory", function(){
    assert.deepEqual(recordStore.inventory, []);
    assert.strictEqual(recordStore.inventory.length, 0);

  });

  it("should have a balance 0 at init", function(){
    assert.strictEqual(recordStore.balance, 0 );
  });

  it("should be able to add records to inventory", function(){
    recordStore.add(record);
    assert.strictEqual(recordStore.inventory.length, 1);
  });

  it("should be able to list the inventory", function(){
    recordStore.add(record);
    var result = "1. test, x&y, chillout, £9.99"
    assert.strictEqual(recordStore.listInventory(), result);
  });

  it("should list multipley items ", function(){
    recordStore.add(record);
    recordStore.add(record1);
    var result = "1. test, x&y, chillout, £9.99\n2. test2, x&y&z, chillout-more, £19.99";
    assert.strictEqual(recordStore.listInventory(), result);
  });

  it("should be able to remove a record from inventory", function(){
    recordStore.add(record);
    recordStore.remove(record);
    assert.strictEqual(recordStore.inventory.length, 0);
  })

  it("should be able to modify balance", function(){
    recordStore.modifyBalance(10);
    assert.strictEqual(recordStore.balance, 10);
  });

  it("should be able to sell a record", function(){
    recordStore.add(record);
    recordStore.sell(record);
    assert.strictEqual(recordStore.balance, record.price);
    assert.strictEqual(recordStore.inventory.length, 0);
  });

  it("should be able to sell a record that exists", function(){
    assert.strictEqual( false ,recordStore.sell(record));
  });

  it("should be able to print finance data", function(){
    var result = "Balance: £0 Inventory: £0";
    assert.strictEqual(recordStore.financeData(), result);
  });

  it("should get the inventory value ", function(){
    recordStore.add(record);
    assert.strictEqual(recordStore.inventoryValue(), record.price);
  });

  it("should get the inventory value with multiple items", function(){
    recordStore.add(record);
    recordStore.add(record1);
    assert.strictEqual(recordStore.inventoryValue(), 29.98);
  });
})
```

- **The test code failing to pass**

```
28
29  it("should have a balance 0 at init", function(){
30    assert.strictEqual(recordStore.balance, 0 );
31  });
32
33  it("should be able to add records to inventory", function(){
34    assert.strictEqual(recordStore.inventory.length, 1);
35  });
36
37  it("should be able to list the inventory", function(){
38    recordStore.add(record);
39    var result = "1. test, x&y, chillout, £9.99"
40    assert.strictEqual(recordStore.listInventory(), result);
41  });
42
43  it("should list multipley items ", function(){
44    recordStore.add(record);
45    recordStore.add(record1);
46    var result = "1. test, x&y, chillout, £9.99\n2. test2, x&y&z, chillout-more, £19.99";
47    assert.strictEqual(recordStore.listInventory(), result);
48  });
49
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                    1: z

  18 passing (20ms)
  1 failing

  1) RecordStore should be able to add records to inventory:

      AssertionError: 0 === 1
      + expected - actual

      -0
      +1

      at Context.<anonymous> (specs/record_store_spec.js:34:12)


npm ERR! Test failed.  See above for more details.
➜  recordStorejs git:(master) ✗
```

- **Example of the test code once errors have been corrected**

```
28
29  it("should have a balance 0 at init", function(){
30    assert.strictEqual(recordStore.balance, 0 );
31  });
32
33  it("should be able to add records to inventory", function(){
34    recordStore.add(record);
35    assert.strictEqual(recordStore.inventory.length, 1);
36  });
37
38  it("should be able to list the inventory", function(){
39    recordStore.add(record);
40    var result = "1. test, x&y, chillout, £9.99"
41    assert.strictEqual(recordStore.listInventory(), result);
42  });
```

- **The test code passing**

```
32
33    it("should be able to add records to inventory", function(){
34      recordStore.add(record);
35      assert.strictEqual(recordStore.inventory.length, 1);
36    });
37
38    it("should be able to list the inventory", function(){
39      recordStore.add(record);
40      var result = "1. test, x&y, chillout, £9.99"
41      assert.strictEqual(recordStore.listInventory(), result);
42    });
43
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

> mocha specs/


  record
    ✓ should have an artist
    ✓ should hava a title
    ✓ should have a genre
    ✓ should hava a price
    ✓ should have print a record summary

  RecordStore
    ✓ should have a name
    ✓ should have a city
    ✓ should have an inventory
    ✓ should have a balance 0 at init
    ✓ should be able to add records to inventory
    ✓ should be able to list the inventory
    ✓ should list multipley items
    ✓ should be able to remove a record from inventory
    ✓ should be able to modify balance
    ✓ should be able to sell a record
    ✓ should be able to sell a record that exists
    ✓ should be able to print finance data
    ✓ should get the inventory value
    ✓ should get the inventory value with multiple items


  19 passing (15ms)

➜  recordStorejs git:(master) ✗
```