

Project Unit

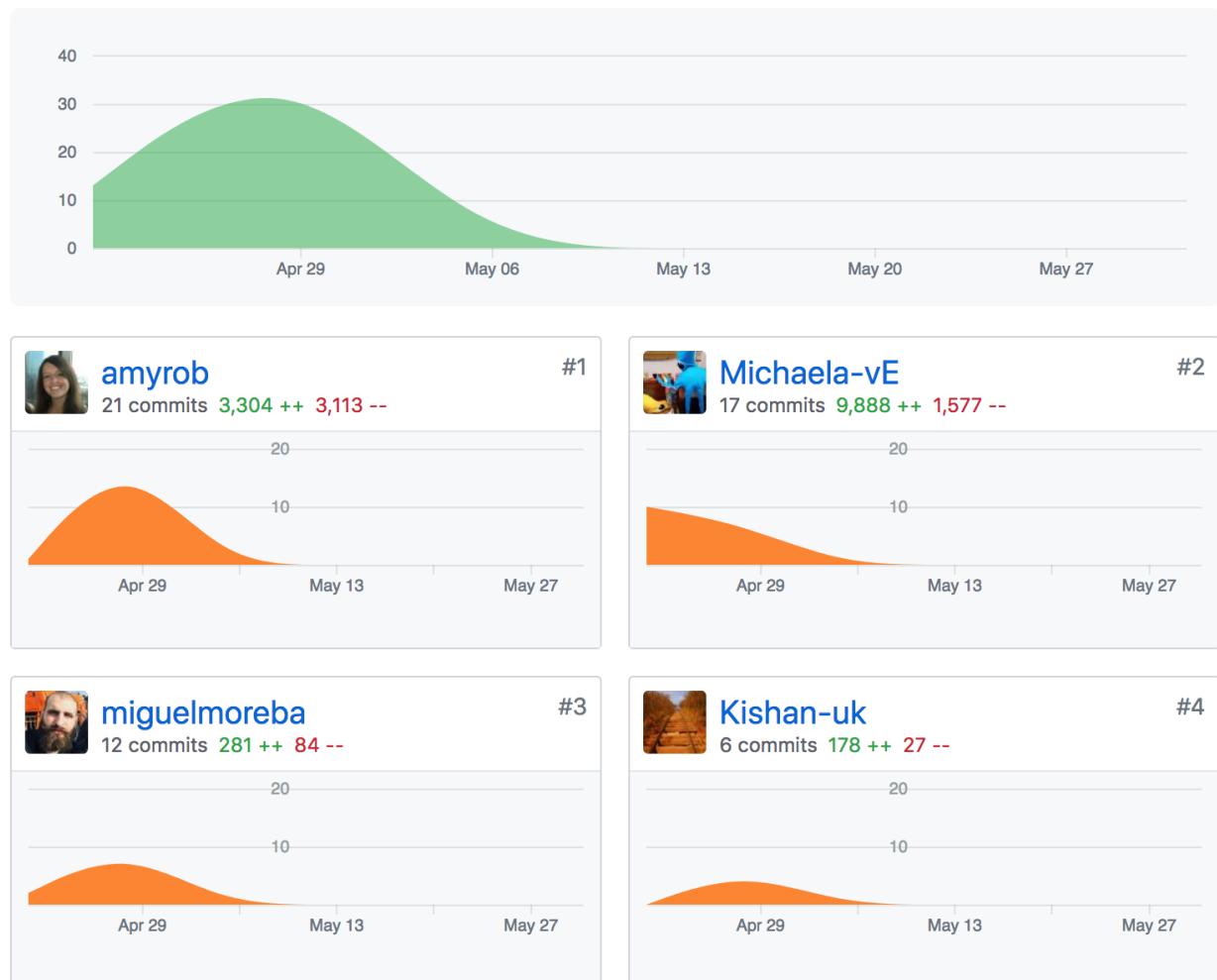
Kishan Bachoo
E19

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

Apr 22, 2018 – Jun 1, 2018

Contributions: Commits ▾

Contributions to develop, excluding merge commits



P 2 Take a screenshot of the project brief from your group project

Store Finder

Global pub giant Withoutaspoon, wants a pub finder that allows users to "Find your nearest store" for their website.

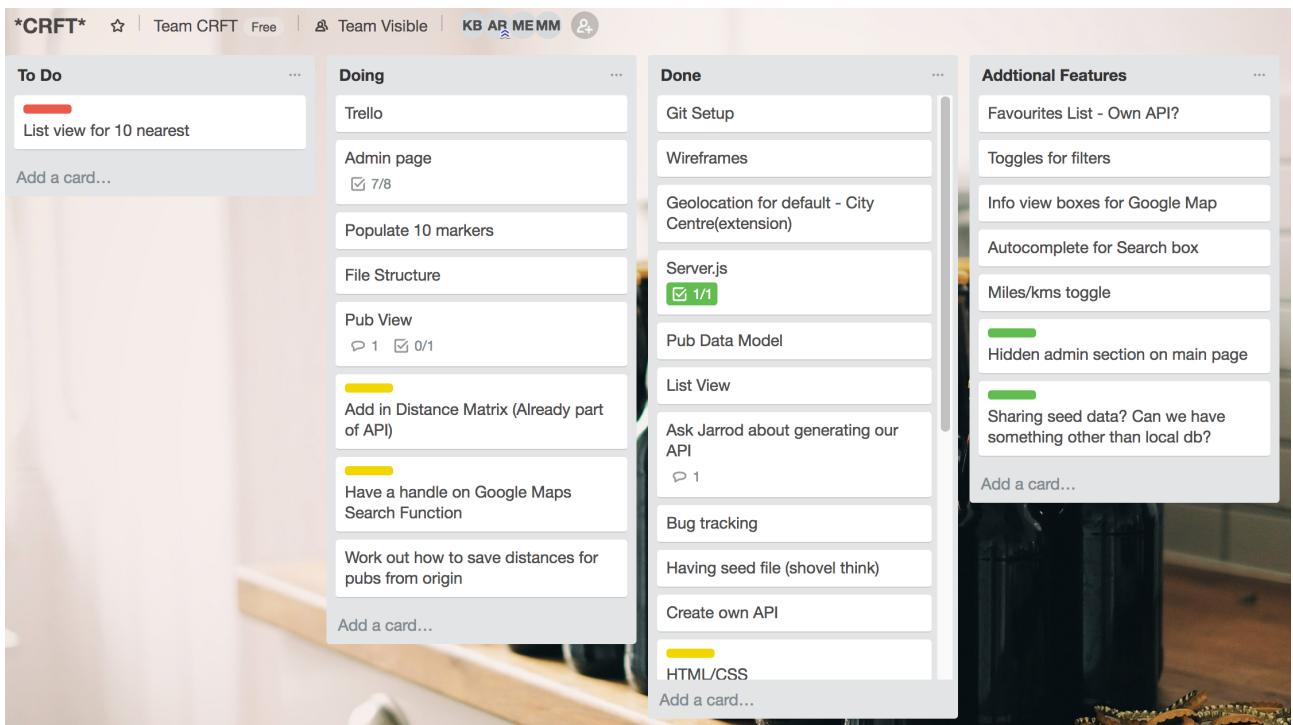
User should be able to enter their postcode or town or street name and get a list of the top ten nearest stores. Each store should have a distance, address, phone number, opening hours and a list of facilities and services (e.g. cash machine, car parking).

You'll have to build your own API to persist the store data and use an external api to find addresses from postcodes.

MVP

- Search by postcode or town or street name
- A list of stores including the details of the stores
- A map marking the stores
- Use Geolocation to allow users to "use current location" to find stores

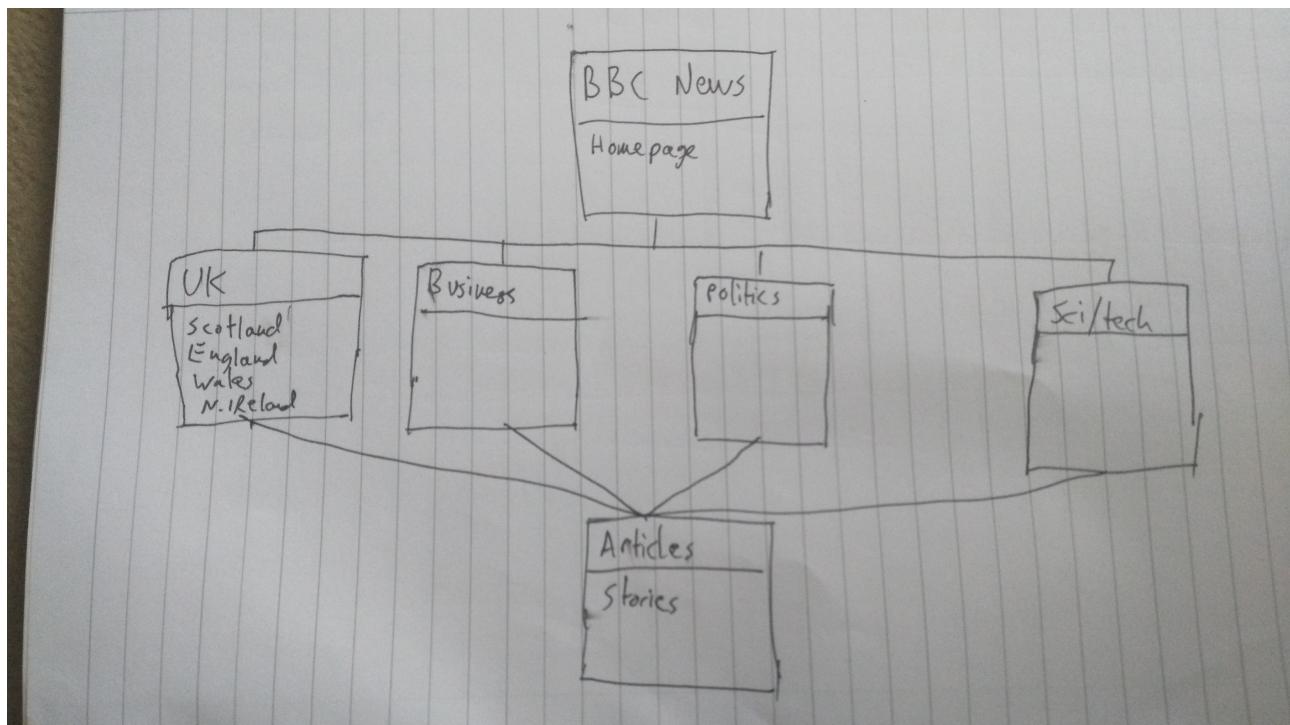
P 3 Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board



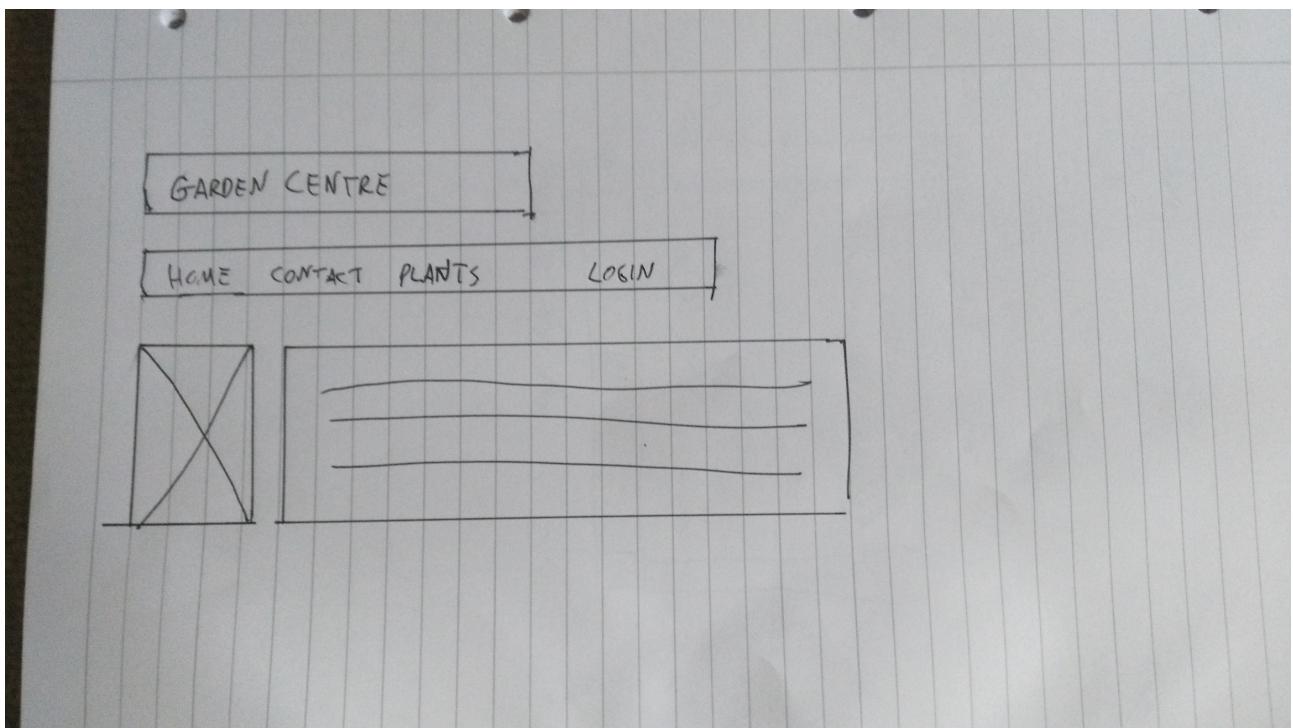
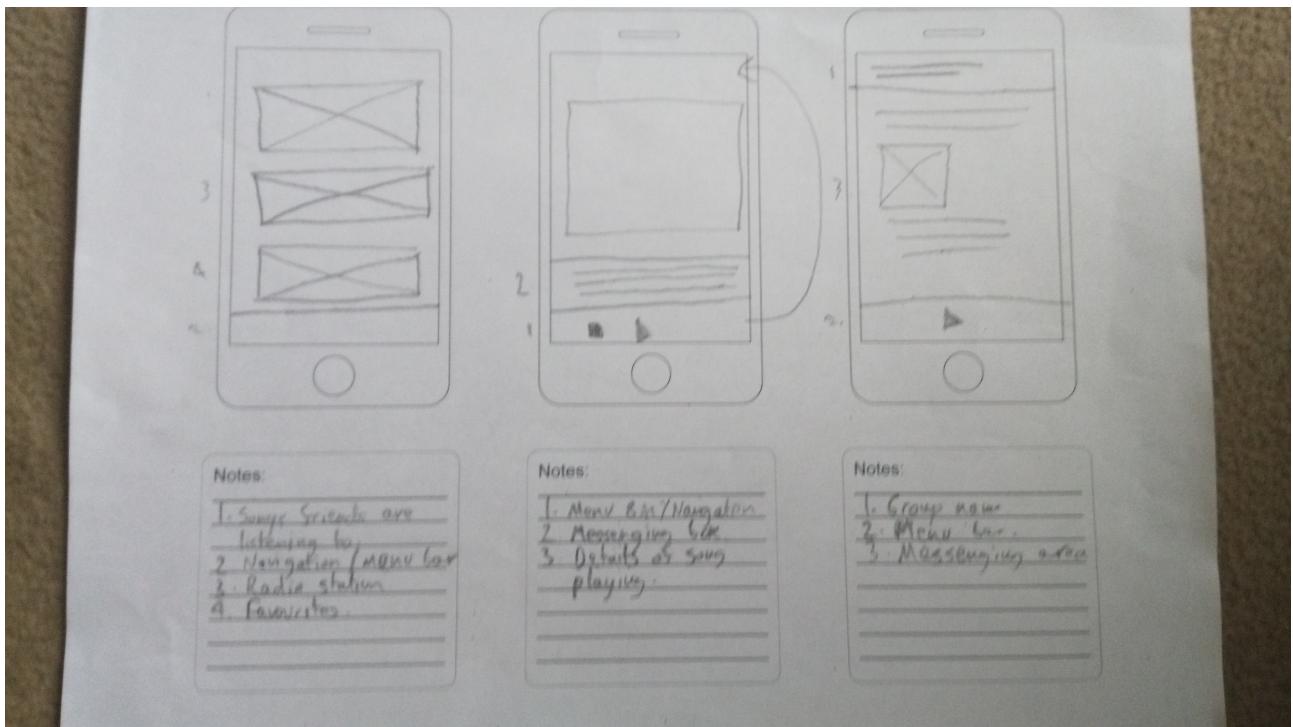
P 4 Write an acceptance criteria and test plan

Acceptance Criteria	Expected Result/Output	Pass/Fail
A user should be able to check move around the map	The user should be able to click and hold a button that allows them to pan around the map	PASS
The user should be able to select a pub and get details about it	The user should be able to click on a pub marker and bring up information about the pub in an infobox	PASS
The user should be able to get the 10 nearest pubs from their location	The map should display 10 markers of the nearest pubs from their location	PASS

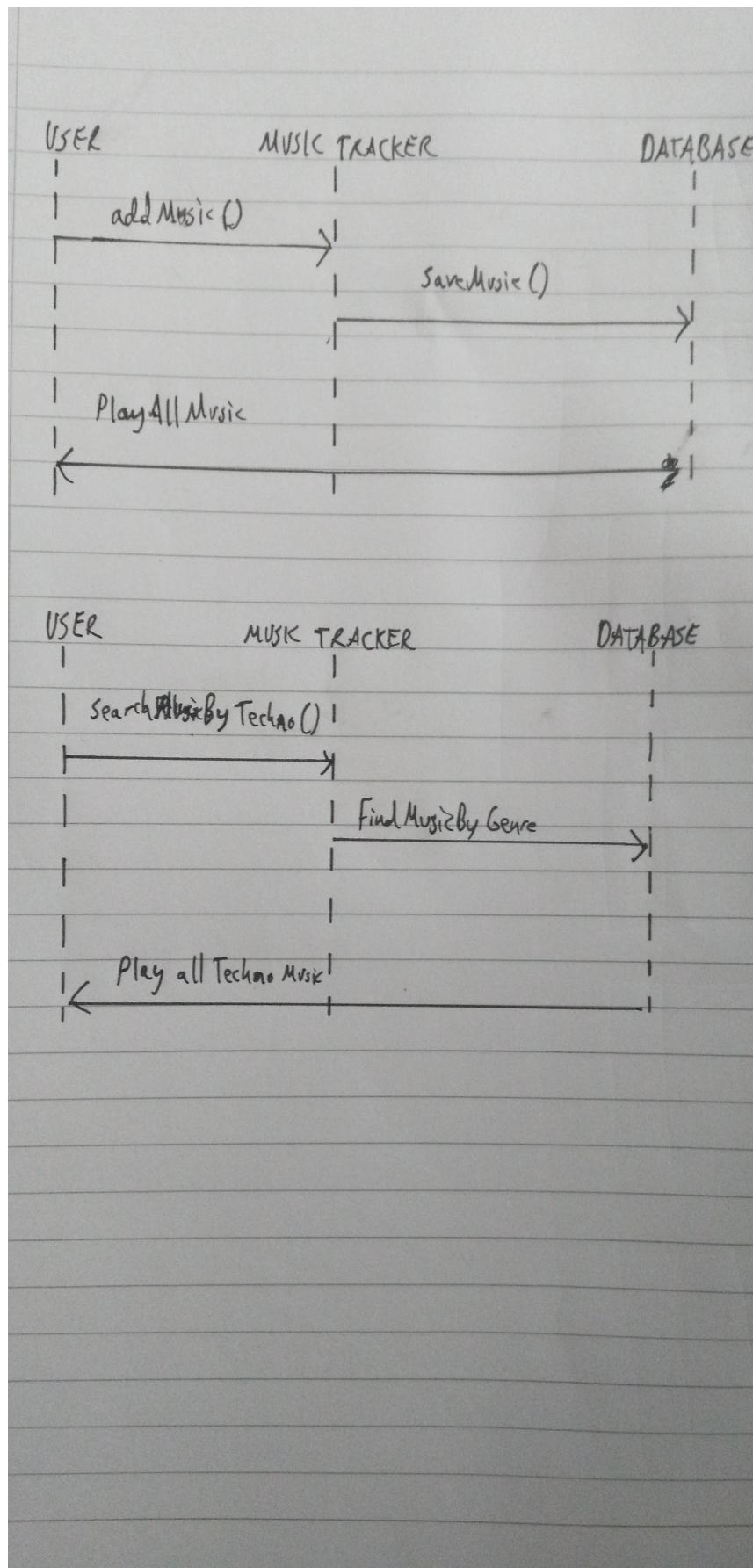
P 5 Create a user sitemap

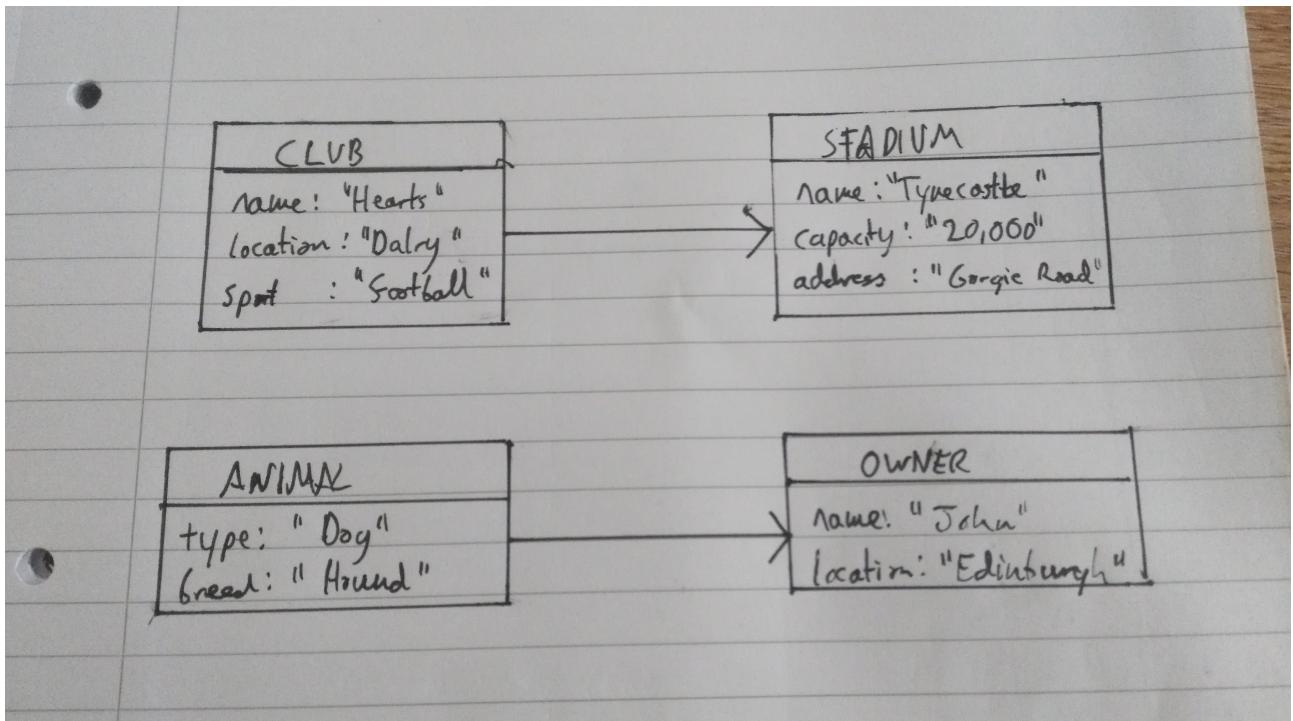


P 6 Produce two wireframe designs

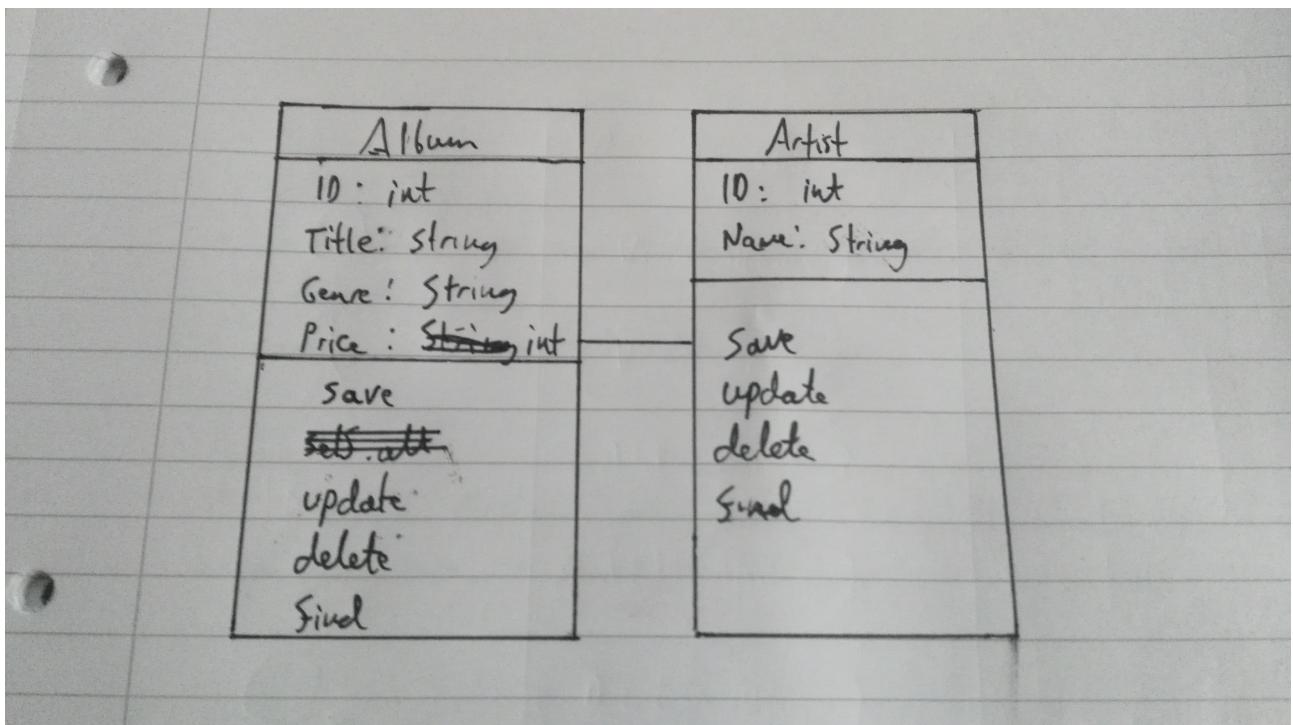


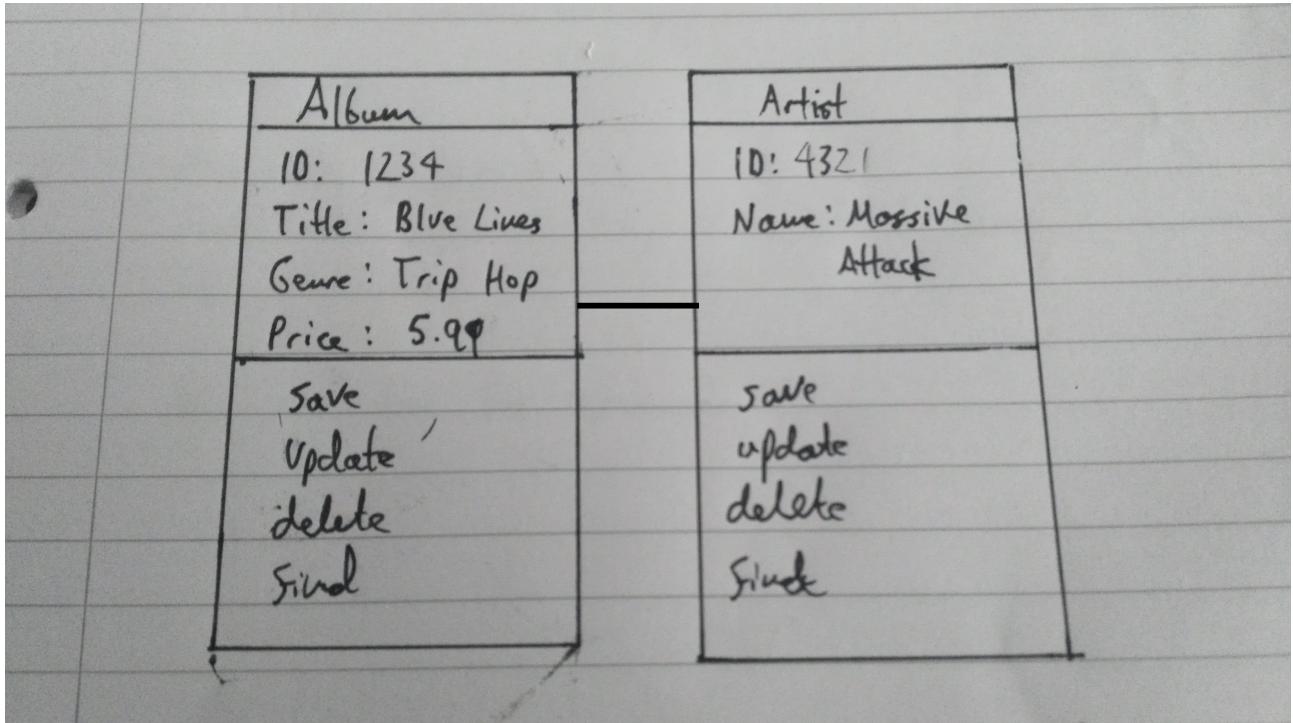
P 7 Produce two system interaction diagrams (sequence and/or collaboration diagrams)





P 8 Produce two object diagrams





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

```

public void selectPaymentType(String paymentMethod) {
    for(PaymentType paymentType: paymentList ) {
        if(paymentType.getPaymentMethod() == paymentMethod){
            payment = paymentType;
        }
    }
}

```

I chose this algorithm as it was the simplest way of selecting a payment method.

```

public void makeSaleToCustomer(Product product, Customer customer, int orderItemQuantity) {

    if (products.contains(product)) {
        if (product.getQuantity() > 0) {
            customer.buyProduct(product.getPrice(), orderItemQuantity);
            purchasedProducts.add(product);
            sales = sales + (product.getPrice() * orderItemQuantity);
            int newQuantity = product.getQuantity() - orderItemQuantity ;
            product.setQuantity(newQuantity);
        }
    }

    if (product.getQuantity() == 0) {
        products.remove(product);
    }
}

```

I chose the algorithm as it demonstrated the robustness of working out the sales price only when there are items that have been added to the product quantity.

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

```

1 // Create function that allows users to display all Albums
2 // Select all albums from the database
3 // Return albums via a Map
4 // End function

```

P 11 Take a screenst hot of one of your projects where you have worked alone and attach a Github link

Kishan-uk / record-store

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Record Store inventory project

Add topics

8 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Kishan-uk updated readme.md file			Latest commit 69aa7b1 on 27 Apr
controllers	updated readme.md file		a month ago
db	updated readme.md file		a month ago
models	updated readme.md file		a month ago
public	updated readme.md file		a month ago
views	updated readme.md file		a month ago
README.md	updated readme.md file		a month ago
app.rb	updated readme.md file		a month ago
README.md			

<https://github.com/Kishan-uk/record-store>

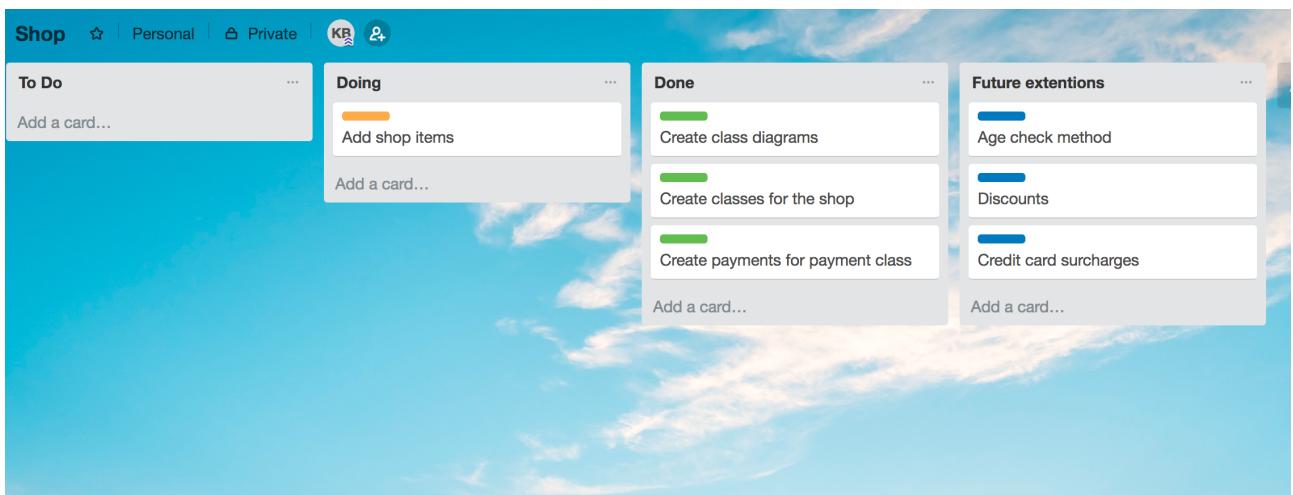
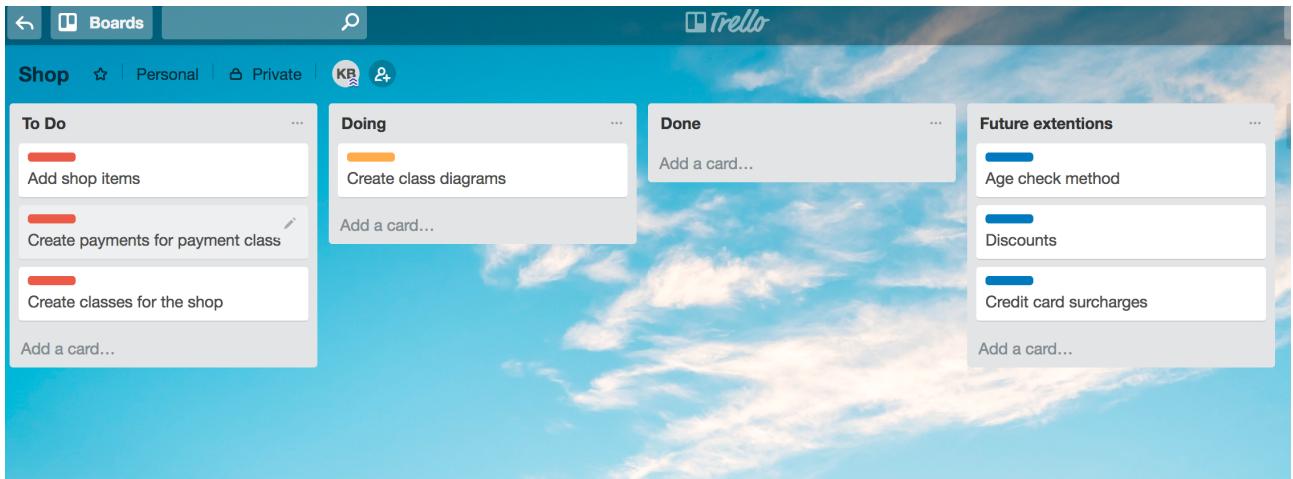
The screenshot shows a code editor with several tabs open. On the left, the project structure is displayed:

- Project
- record_store
- .git
- controllers
 - album_controller.rb
 - album_list.rb
 - artist_controller.rb
- db
 - record_store.sql
 - seeds.rb
 - sql_runner.rb
- models
 - album.rb
 - artist.rb
- public
- views
 - album
 - create.erb
 - new.erb
 - album_list
 - index.erb
 - artist
 - create.erb
 - new.erb
 - album_list.erb
 - index.erb
 - layout.erb

The main tab is 'album.rb', showing the following code:

```
67     end
68
69   def mark_up
70     result = @sell_price - @buy_price
71     return sprintf("%.2f", result)
72   end
73
74   def self.find(id)
75     sql = "SELECT * FROM albums WHERE id = $1"
76     values = [id]
77     result = SqlRunner.run(sql, values).first
78     return Album.new(result)
79   end
80
81   #added for RDA
82   def self.sort_by_quantity()
83     sql = "SELECT * FROM albums ORDER BY quantity"
84     values = [id]
85     result = SqlRunner.run(sql, values).first
86     return Album.new(result)
87   end
88
89   def self.delete_all()
90     sql = "DELETE FROM albums"
91     SqlRunner.run(sql)
92   end
93
94   def self.all()
95     sql = "SELECT * FROM albums"
96     albums = SqlRunner.run(sql)
97     return albums.map{|album| Album.new(album)}
98   end
99
```

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



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

- Data being inputted into your program

Amazing Records

Home Album list Add new album Add new artist

Albums

Artist	Title	Genre	Stock Level	Buy Price £:	Sell Price £:	Mark up: £:	Quantity:	Stock level:		
Air	Moon Safari	Electronica	high	3.99	6.99	3.00	57	high	<button>Update album</button>	<button>Delete album</button>
Daft Punk	Homework	French House	medium	4.99	8.99	4.00	31	medium	<button>Update album</button>	<button>Delete album</button>
Vitalic	OK Cowboy	Techno	low	4.49	6.5	2.01	20	low	<button>Update album</button>	<button>Delete album</button>

Powered by Amazing Software Systems - © 2018 All rights reserved.

Amazing Records

Home Album list Add new album Add new artist

Artist: Add artist

Powered by Amazing Software Systems - © 2018 All rights reserved.

- The user input being saved

Amazing Records

[Home](#) [Album list](#) [Add new album](#) [Add new artist](#)

Artist added!

[Return to the homepage](#)

Powered by Amazing Software Systems - © 2018 All rights reserved.

Amazing Records

[Home](#) [Album list](#) [Add new album](#) [Add new artist](#)

Artist: Title: Genre: Quantity: Buy price: Sell price:

Add a new artist

Daft Punk

Vitalic

Massive Attack

Powered by Amazing Software Systems - © 2018 All rights reserved.

P14 Show an interaction with data persistence. Take a screenshot of:

- Data being inputted into your program

Amazing Records

Home Album list Add new album Add new artist

Artist: Massive Attack Title: Blue Lines Genre: Trip hop Quantity: 50 Buy price: 4 Sell price: 2

Add album

Powered by Amazing Software Systems - © 2018 All rights reserved.

- Confirmation of the data being saved

Amazing Records

Home Album list Add new album Add new artist

Album added!

[Return to the homepage](#)

Powered by Amazing Software Systems - © 2018 All rights reserved.

Amazing Records

Home Album list Add new album Add new artist

Albums

Artist	Title	Genre	Stock Level	Buy Price £	Sell Price £	Mark up: £	Quantity:	Stock level:	Update album	Delete album
Air	Moon Safari	Electronica	high	3.99	6.99	3.00	57	high	<button>Update album</button>	<button>Delete album</button>
Daft Punk	Homework	French House	medium	4.99	8.99	4.00	31	medium	<button>Update album</button>	<button>Delete album</button>
Vitalic	OK Cowboy	Techno	low	4.49	6.5	2.01	20	low	<button>Update album</button>	<button>Delete album</button>
Massive Attack	Blue Lines	Trip hop	medium	4.0	2.0	-2.00	50	medium	<button>Update album</button>	<button>Delete album</button>

Powered by Amazing Software Systems - © 2018 All rights reserved.

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

Albums

Artist	Title	Genre	Stock Level	Buy Price £:	Sell Price £:	Mark up: £:	Quantity:	Stock level:		
Daft Punk	Homework	French House	medium	4.99	8.99	4.00	31	medium	<button>Update album</button>	<button>Delete album</button>
Vitalic	OK Cowboy	Techno	low	4.49	6.5	2.01	20	low	<button>Update album</button>	<button>Delete album</button>
Massive Attack	Blue Lines	Trip hop	medium	4.0	2.0	-2.00	50	medium	<button>Update album</button>	<button>Delete album</button>

(albums to be deleted through the *delete album* button)

- The user request being processed correctly and demonstrated in the program

Albums

Artist	Title	Genre	Stock Level	Buy Price £:	Sell Price £:	Mark up: £:	Quantity:	Stock level:		
Vitalic	OK Cowboy	Techno	low	4.49	6.5	2.01	20	low	<button>Update album</button>	<button>Delete album</button>

Powered by Amazing Software Systems - © 2018 All rights reserved.

P 16 Show an API being used within your program. Take a screenshot of:

- The code that uses or implements the API

```
const Request = require('../helpers/request.js');

const CountryData = function() {
    this.url = 'https://restcountries.eu/rest/v2/all';
    this.data = null;
}

CountryData.prototype.getData = function (onComplete) {
    const request = new Request(this.url);
    request.get((data) => {
        console.log(data);
        this.data = data;
        onComplete(data);
    })
};

module.exports = CountryData;
```

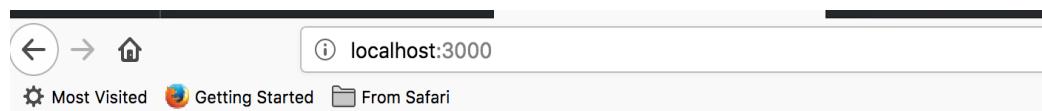
```
const CountryView = function(selectElement, countryContainer) {
  this.selectElement = selectElement;
  this.countryContainer = countryContainer;
}

CountryView.prototype.renderSelect = function (countryData) {
  countryData.forEach((country, index) => {
    const countryOption = document.createElement('option');
    countryOption.textContent = country.name;
    countryOption.value = index;
    this.selectElement.appendChild(countryOption);
  });
};

CountryView.prototype.renderDetail = function (country) {
  const countryName = document.createElement('p');
  countryName.textContent = country.name;
  this.countryContainer.appendChild(countryName);
}

module.exports = CountryView;
```

- This API being used by the program whilst running



Countries

Angola

Angola

P 17 Produce a bug tracking report

User must be able to connect to DB	Failed	Placement of server listen within Mongo client. No root url being passed in app.js server();	Passed
User must be able to see map	Failed	Reran build	Passed

P 18 Demonstrate testing in your program. Take screenshots of:

- Example of test code

```
@Test
public void makeSaleToCustomerTest() {
    shop.makeSaleToCustomer(product, customer, product.getQuantity());
    assertEquals( expected: 3, product.getQuantity());
    assertEquals( expected: 107.96, shop.getShopSales(), delta: 0.01);
    assertEquals( expected: 192.04, customer.getBalanceFromPaymentType(), delta: 0.01);
}
```

- The test code failing to pass

```

37
38
39     @Test
40     public void makeSaleToCustomerTest() {
41         shop.makeSaleToCustomer(product, customer, product.getQuantity());
42         assertEquals( expected: 3, product.getQuantity());
43         assertEquals( expected: 107.96, shop.getShopSales(), delta: 0.01);
44         assertEquals( expected: 192.04, customer.getBalanceFromPaymentType(), delta: 0.01);
45     }

```

ShopTest > makeSaleToCustomerTest()

13 tests done: 1 failed - 48ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java ...

java.lang.AssertionError:
Expected :3
Actual :4
[<Click to see difference>](#)

+ <1 internal call>
+ at org.junit.Assert.failNotEquals([Assert.java:834](#)) <2 internal calls>
+ at ShopTest.makeSaleToCustomerTest([ShopTest.java:41](#)) <31 internal calls>

- Example of the test code once errors have been corrected

```

37
38
39     @Test
40     public void makeSaleToCustomerTest() {
41         shop.makeSaleToCustomer(product, customer, product.getQuantity());
42         assertEquals( expected: 4, product.getQuantity());
43         assertEquals( expected: 107.96, shop.getShopSales(), delta: 0.01);
44         assertEquals( expected: 192.04, customer.getBalanceFromPaymentType(), delta: 0.01);
45     }

```

ShopTest > makeSaleToCustomerTest()

All 13 tests passed - 22ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

- The test code passing

```

37
38
39     @Test
40     public void makeSaleToCustomerTest() {
41         shop.makeSaleToCustomer(product, customer, product.getQuantity());
42         assertEquals( expected: 4, product.getQuantity());
43         assertEquals( expected: 107.96, shop.getShopSales(), delta: 0.01);
44         assertEquals( expected: 192.04, customer.getBalanceFromPaymentType(), delta: 0.01);
45     }

```