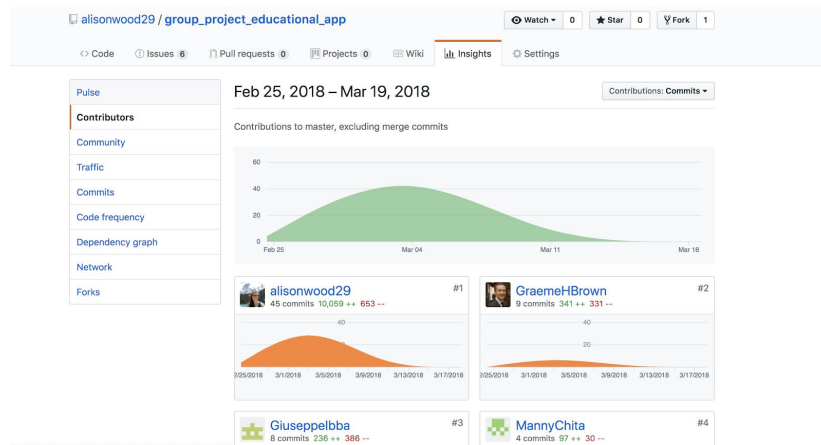


P.1

-group project contributors page



P.2

-group project brief

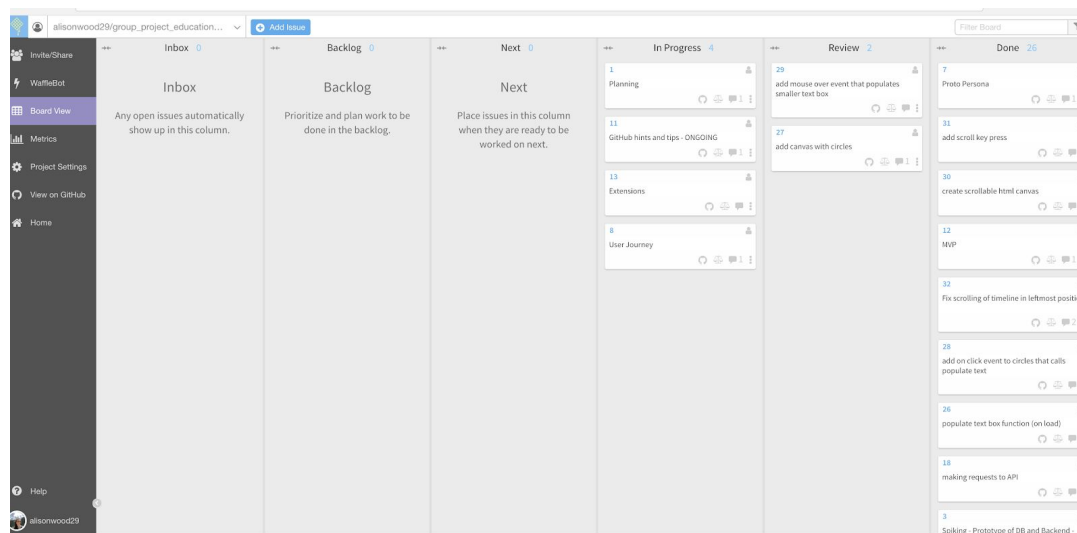
Educational App The BBC are looking to improve their online offering of educational content by developing some interactive apps that display information in a fun and interesting way.

Your task is to make an MVP 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. You might use an API to bring in content or a database to store facts. The topic of the app is your choice, but here are some suggestions you could look into:

Interactive timeline, e.g. of the history of computer programming
Interactive map of a historical event - e.g. World War 1, the travels of Christopher Columbus
MVP Display some information about a particular topic in an interesting way
Have some user interactivity using event listeners, e.g. to move through different sections of content

P.3

-group project planning



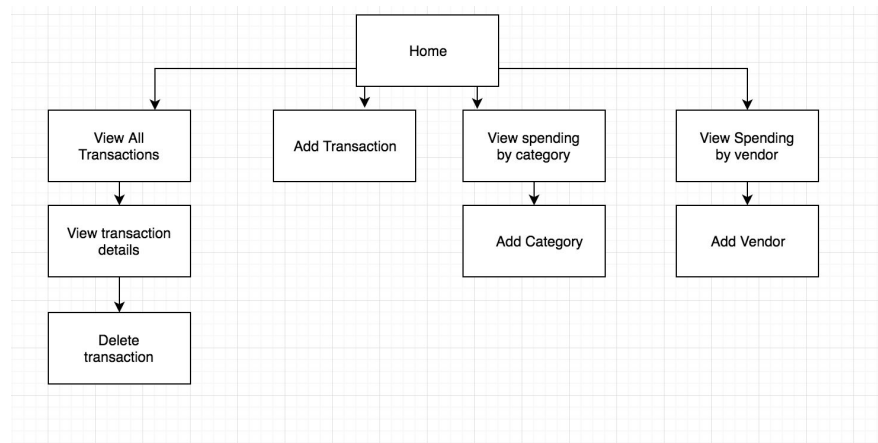
P.4

-An acceptance criteria and test plan

Acceptance Criteria for Budget Tracker	Expected Result/Output	Pass/Fail
A user can view all transactions	List of all transactions is displayed when link in nav is clicked	Pass
A user can add a transaction	User can fill out and submit a form with transaction details which adds to db	Pass
A user can add a new category field for transactions	User can add a new category via a form. Added category now appears in the dropdown menu when adding a new transaction	Pass
A user can view spending by category	Table of spending per category is displayed when user clicks on link	Pass
A user can view spending by date	Table of spending on a selected date is displayed when a user selects a date from a dropdown calendar	Pass

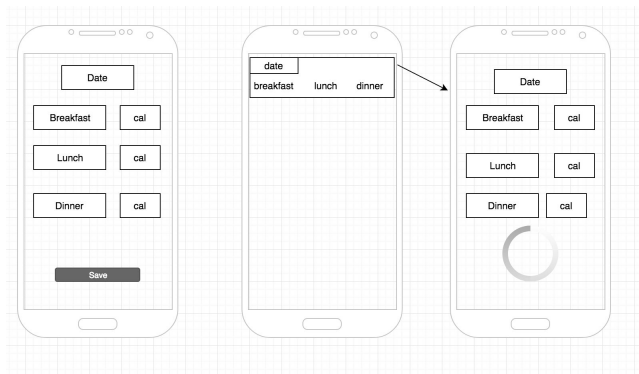
P.5

-create a user sitemap



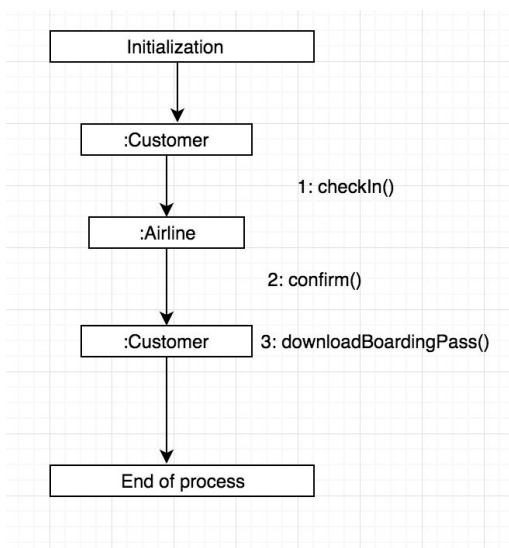
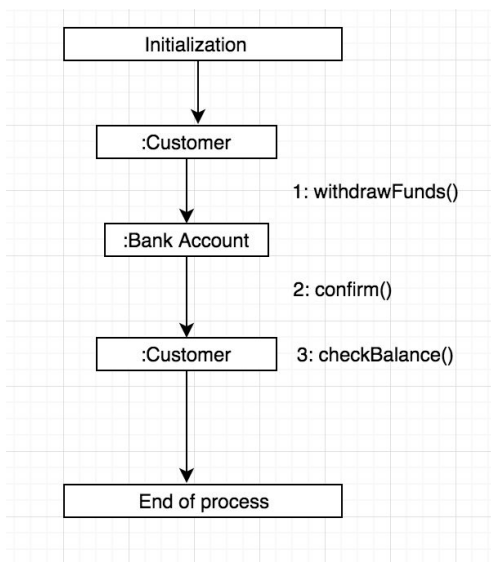
P.6

-wireframe design



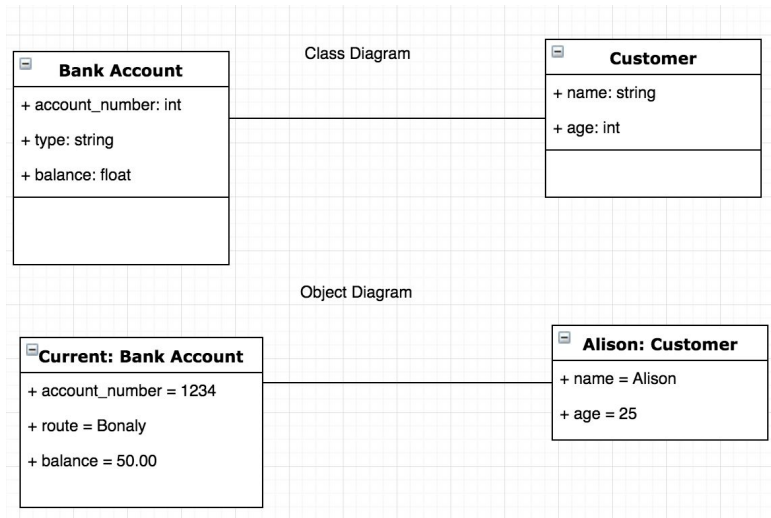
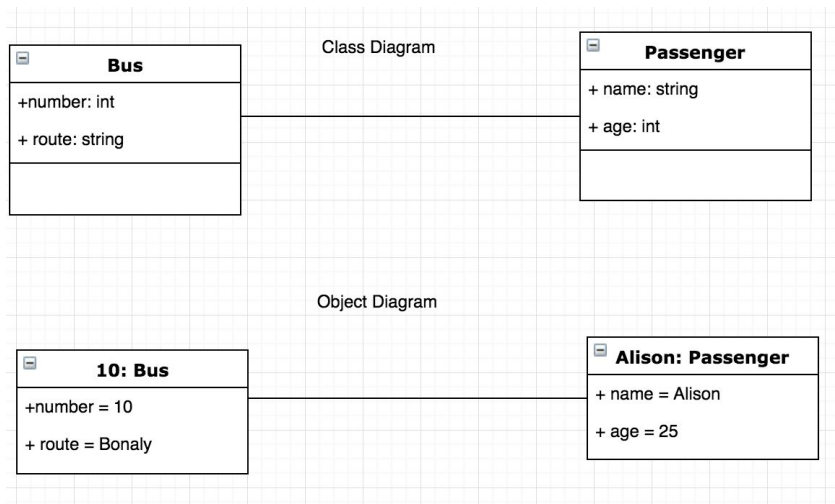
P.7

-2 system interaction diagrams



P.8

-2 object diagrams



P.9

-2 algorithms

```
public void addToStock(ISellable item) {
    if(item.getBuyPrice() <= getBudget()) {
        this.stock.add(item);
        this.budget -= item.getBuyPrice();
    }
}
```

This algorithm is for adding an item to the stock of a music shop. An item has a price. If the price of the item is less than or equal to the budget then the shop can add the item to the stock and update the budget by the price of the item. I chose to use this algorithm as you don't want to be able to buy an item if you don't have enough money.

```
public double calculateMarkUp(){
    return this.sellPrice - this.buyPrice;
}
```

```
public double potentialProfit() {
    double profit = 0;
    for(ISellable item: this.stock){
        profit += item.calculateMarkUp();
    }
    return profit;
}
```

This algorithm calculates the potential profit of a shop. The profit starts at 0 then for every item in the shops stock it adds the mark up price to the profit. The mark up is the difference between the sell price and buy price of an item. Once it has looped through every item in the stock array the potential profit will be returned. I used this algorithm because a shop will want to know how much money it can potentially make at any one time.

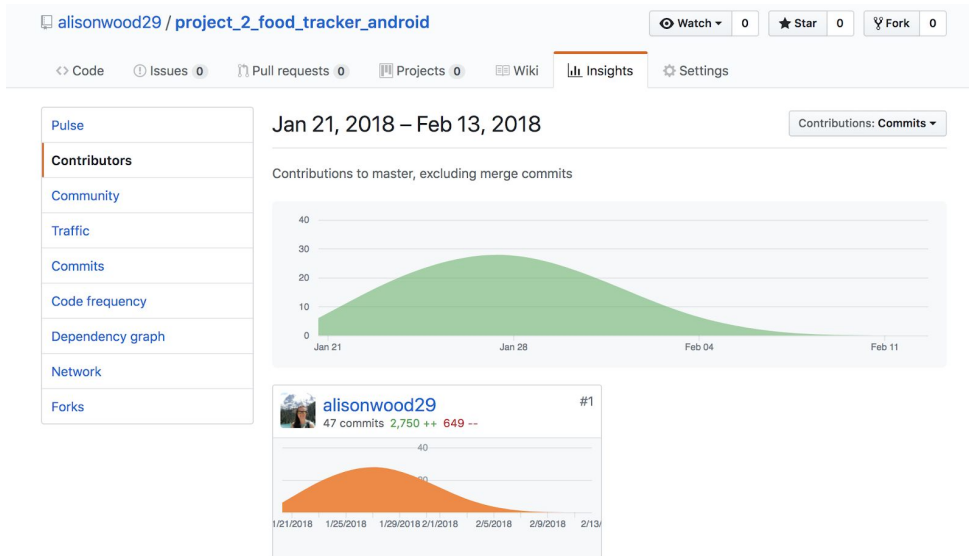
P.10

-pseudo code

```
// get objects out of db
// add to array
// sort array by date
// use array to display in browser
```

P.11

-project where i have worked alone



P.12

-planning at different stages

FoodTrackerApp ☆ Personal Private

Android Studio

- add edit text fields for meals
- add edit text field for date
- add list view for dailyfoods array
- add shared pref to save data
- add menu bar
- add an entry for max calories on launcher page

Add a card...

Classes

- create Breakfast class
- create Dinner class
- create Lunch class
- create DailyFood class
- create FoodTracker class
- create Meal super class

Add a card...

Extensions

- add drink water page
- add progress bar

Add a card...

FoodTrackerApp ☆ Personal Private Show Menu

Android Studio

- add room db
- add a new page that shows more details when you click on an item in the listview
- add edit text fields for meals
- add edit text field for date
- add list view for dailyfoods array
- add shared pref to save data
- add menu bar
- add a navigation bar

Add a card...

Classes

- create Breakfast class
- create Dinner class
- create Lunch class
- create DailyFood class
- create FoodTracker class
- create Meal super class

Add a card...

Extensions

- add drink water page
- add progress bar for calories
- add an entry for max calories on launcher page

Add a card...

Bugs to fix

- pie chart if calories is over max
- non-unique dates in table
- crashes if entries left empty

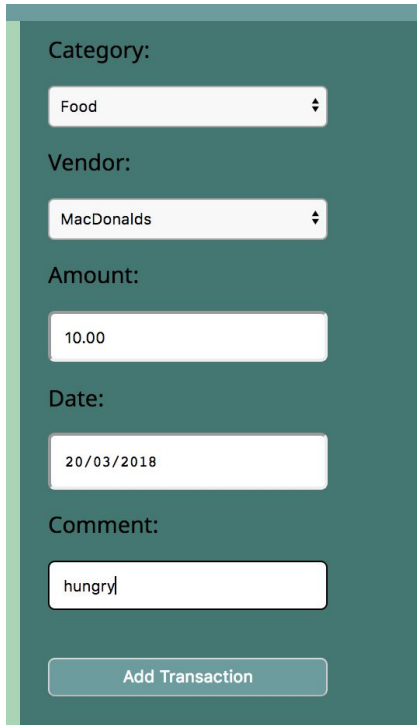
Add a list...

Save X

Add X

P.13

-user inputting into a program



A screenshot of a mobile application interface for adding a transaction. The form is set against a dark teal background with a light green vertical bar on the left. It contains several input fields: a dropdown menu for 'Category' with 'Food' selected, a dropdown menu for 'Vendor' with 'MacDonalds' selected, a text input for 'Amount' containing '10.00', a date input for 'Date' containing '20/03/2018', and a text input for 'Comment' containing 'hungry'. At the bottom is a light blue button labeled 'Add Transaction'.

Category:
Food

Vendor:
MacDonalds

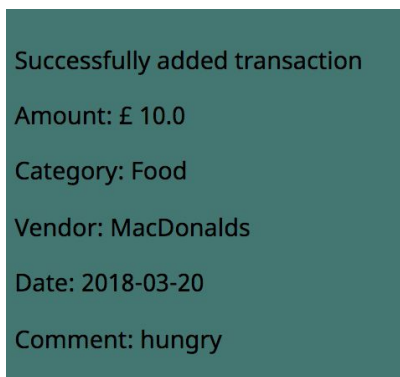
Amount:
10.00

Date:
20/03/2018

Comment:
hungry

Add Transaction

-user input being saved/used



A screenshot of a confirmation screen in the same application. It displays the transaction details in white text on the dark teal background: 'Successfully added transaction', 'Amount: £ 10.0', 'Category: Food', 'Vendor: MacDonalds', 'Date: 2018-03-20', and 'Comment: hungry'.

Successfully added transaction

Amount: £ 10.0

Category: Food

Vendor: MacDonalds

Date: 2018-03-20

Comment: hungry

P.14

-data being input into program

Category:

Vendor:

Amount:

Date:

Comment:

-data being saved

Category: Food

Vendor: MacDonalds

Amount: £ 10.0

Date: 2018-03-20

Comment: hungry

[View Transaction](#)

P.15

-user requesting information

Select Date to View Transaction:

-request being processed

Spending on 2018-03-20

You spent: £20.00

	Category	Vendor	Amount	Comment
Food		MacDonalds	10.0	hungry
Food		MacDonalds	10.0	hungry

P.16

-Code that uses or implements API

```
const urlArray = [];  
const baseUrl = 'http://collection.sciencemuseum.org.uk/objects/';  
const fixedComputerObjects = ["co62748", "co64128", "co62427", "co8359400", "co503  
"co8035886", "co8430789", "co8184137", "co8361071"];  
const additionalObjects = ["co8362946", "co8361832",  
"co8361046", "co63204", "co8194710", "co8094235", "co8401258", "co8361038", "co8015289"  
8408693", "co62349", "co60113", "co60390", "co60127", "co62748", "co64128", "co62427",  
"co8401352", "co8035886", "co8430789", "co8184137", "co8361071"];  
  
additionalObjects.forEach(function(computer){  
    const requestUrl = baseUrl + computer;  
    urlArray.push(requestUrl);  
});  
  
urlArray.forEach(function (url) {  
    const requestUrl = new Request (url);  
    requestUrl.get(computerAPIRequestComplete);  
});
```

```
const Request = function (url) {  
    this.url = url;  
}  
  
Request.prototype.get = function (callback) {  
    const request = new XMLHttpRequest();  
    request.open('GET', this.url);  
    request.setRequestHeader('accept', 'application/json');  
    request.addEventListener('load', function () {  
        if(this.status !== 200) return;  
  
        const responseBody = JSON.parse(this.responseText);  
        callback(responseBody);  
    })  
    request.send();  
}
```

-API being used in a program

Date:


1976

Name:

Personal Computer, model Apple I

Description:

Personal computer by Apple Computers, model Apple I, 1976-79. This was the first computer made by Apple Computers Inc, which became one of the fastest growing companies in history, launching a number of innovative and influential computer hardware and software products. Most home computer users in the 1970s were hobbyists who designed and assembled their own machines. The Apple I, devised in the family garage by Steve Wozniak, Steven Jobs and Ron Wayne, was a basic circuit board to which enthusiasts would add display units and keyboards.



P.17

-bug tracking report

User can add a transaction	Fail	Save a transaction to the db assigning a unique ID	Pass
User can add a category	Fail	Save a category to db assigning a unique ID	Pass
Transaction has a date			Pass
User can search a transaction by date	Fail	Search db for dates that match input date	Pass
User can search for a date that has no transactions	Fail	Add error handling to redirect page to a message that says no transactions for certain date	Pass

P.18

-test code

```
describe('ComputerDates', function () {

  beforeEach(function () {
  });

  it('given a specific id a date is returned', function () {

  });
});
```

-test code failing

```

ComputerDates
  1) given a specific id a date is returned
     ✓ given another id a second date is returned

ComputerObject
  ✓ should have an id
  ✓ should have a date
  ✓ should have a name
  ✓ should have a first description
  ✓ should have a second description
  ✓ should have an image
  ✓ should have a type

8 passing (9ms)
1 failing

1) ComputerDates
   given a specific id a date is returned:
     ReferenceError: actual is not defined
       at Context.<anonymous> (specs/computer_dates_spec.js:11:28)

```

```

npm ERR! Test failed. See above for more details.

```

-updated test code

```

describe('ComputerDates', function () {

  beforeEach(function () {

  });

  it('given a specific id a date is returned', function () {
    const actual = ComputerDates['co62427'];
    assert.strictEqual(actual, 1940);
  });
});

```

-test code passing

```

ComputerDates
  ✓ given a specific id a date is returned
  ✓ given another id a second date is returned

ComputerObject
  ✓ should have an id
  ✓ should have a date
  ✓ should have a name
  ✓ should have a first description
  ✓ should have a second description
  ✓ should have an image
  ✓ should have a type

9 passing (7ms)

```