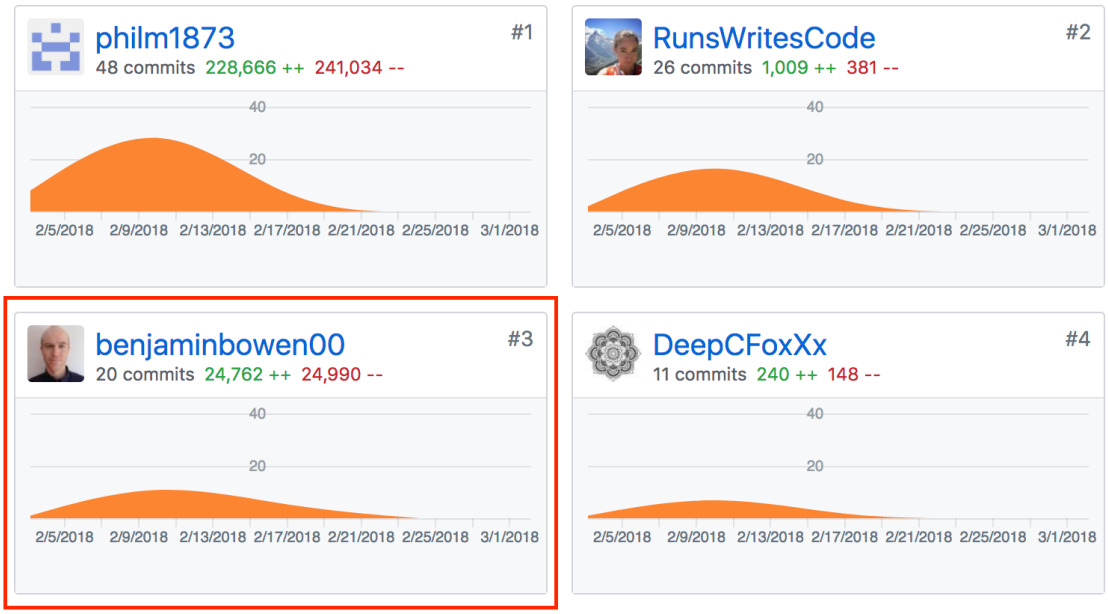
Evidence for the PDA in software development
P - Planning Unit

Benjamin Bowen - Cohort E17

P1
Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.



P2
Take a screenshot of the project brief from your group project.
The group used the given project brief:

Shares App

A local trader has come to you with a portfolio of shares. She wants to be able to analyse it more effectively. She has a small sample data set to give you and would like you to build a minimal viable product (MVP) that uses the data to display her portfolio in useful ways so that she can make better decisions.

MVP

- View total current value
- View individual and total performance trends
- Retrieve a list of share prices from an external API and allow the user to add shares to her portfolio
- Provide a chart of the current values in her portfolio

Examples of further features

Speculation based on trends and further financial modelling.

- Based on the last 6 weeks of data, (e.g. it reduced by 5%) predict the performance of a stock for the next 6 weeks. Based on that what would be the sum of the total on a certain day in the future.
- She wants to know what would happen to the value of her portfolio if the price of a share changed by 10% or if she reduced her quantity by 15%.
- What price does a stock have to get to in order to meet her investment target?
- To get her £1700 worth of RBS shares what price does it have to go to?
- Buy new shares
- Sell shares

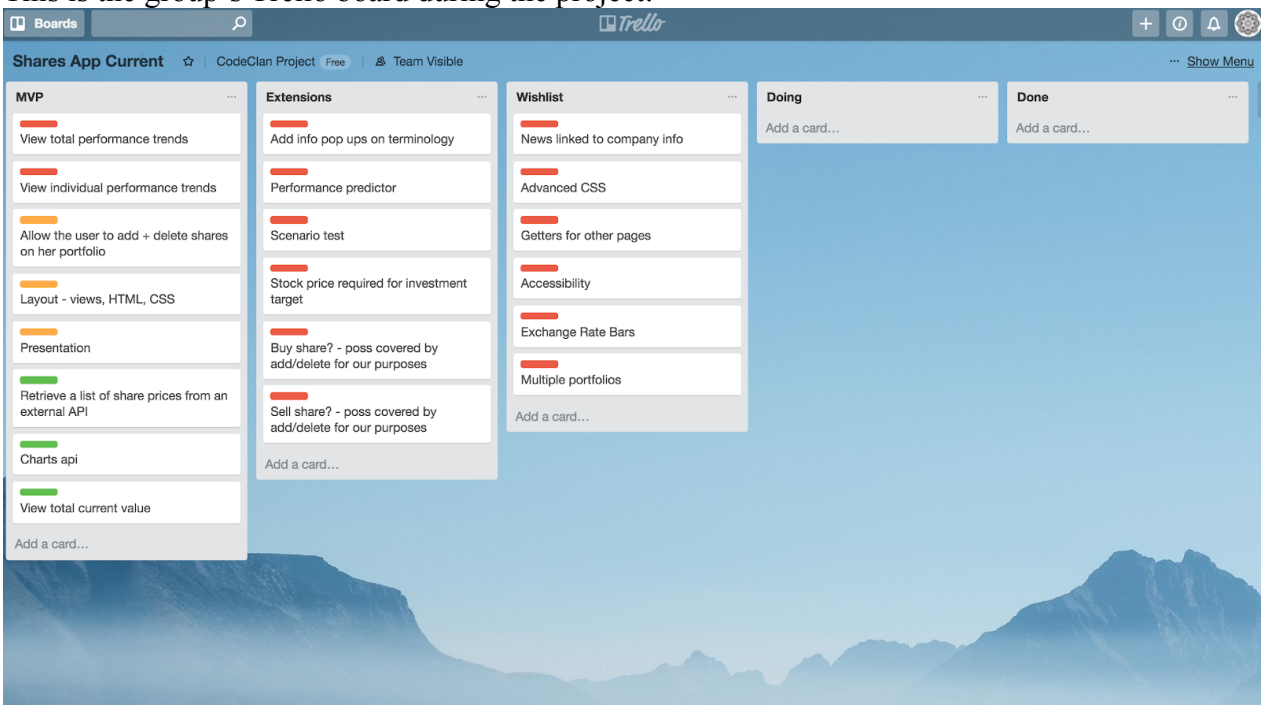
Example API

https://www.alphavantage.co/ (Requires sign up)

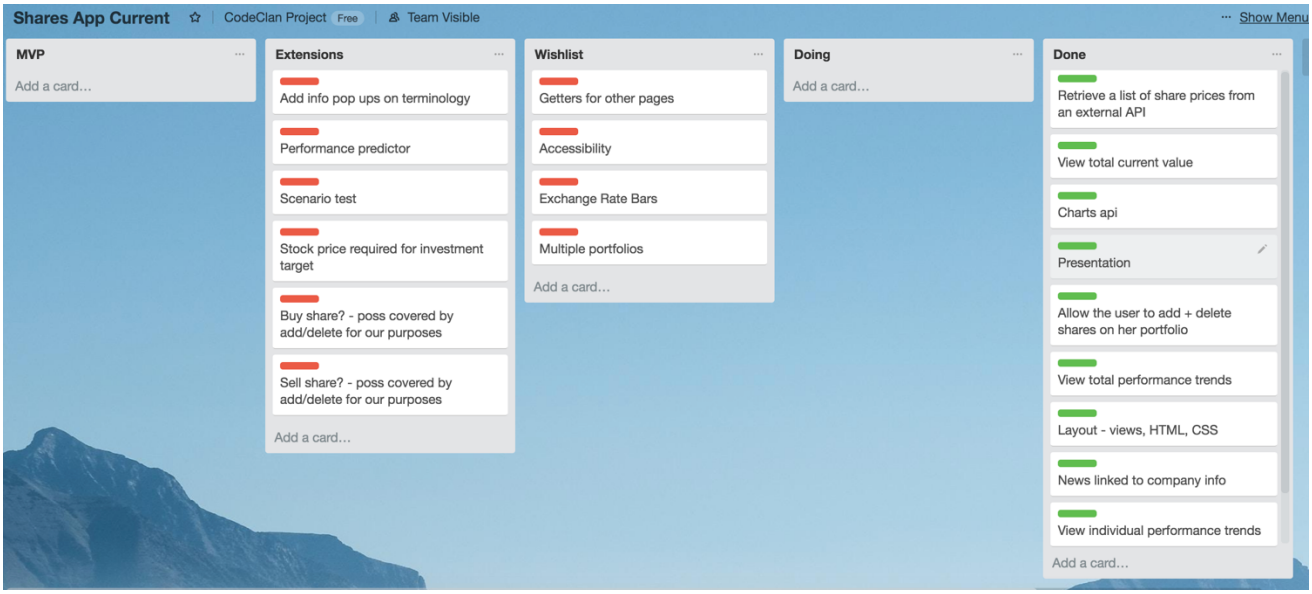
P3

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

This is the group's Trello board during the project:



The is the Trello board at the end of the project:



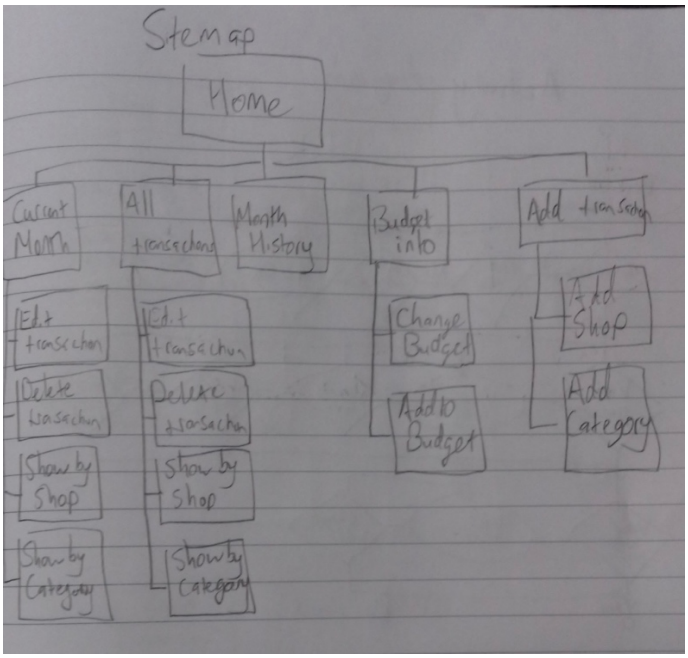
P4
Write an acceptance criteria and test plan.

This is for the group project 'Peak Portfolio'.

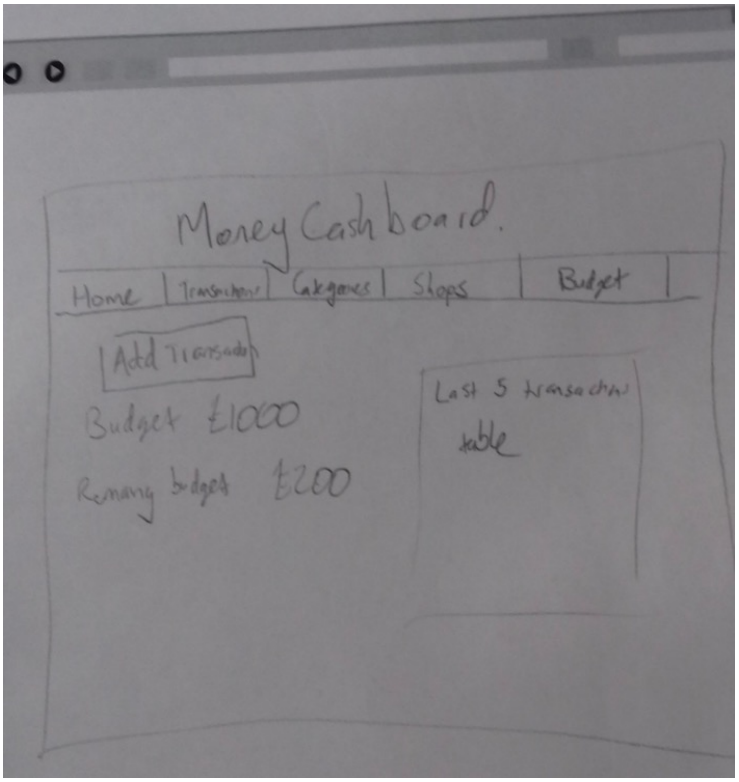
Acceptance Criteria	Expected Result/Output	Pass/Fail
A user is able to record transactions of shares (both buying and selling).	When the add shares button is clicked a modal box opens with a search box for company and text box for inputting the price of share	Pass
A user is able to view the total, up to date, portfolio value	When the app loads this is shown clearly at the top of the page. This displays with the prices of each share using the latest value on alphavantage	Pass

A user is able to view a chart showing the breakdown of the values of shares in the portfolio.	A pie chart is displayed showing the breakdown of the total portfolio value for each share.	Pass
A user is able to view the trend of the value of the whole portfolio	A line chart is displayed at the bottom of the page showing the historical value of the portfolio.	Pass
A user is able to find news information relating to an individual share	A user can click on the name of a share in the table showing all their shares and is taken to a new page showing links to news stories related to that share	Pass

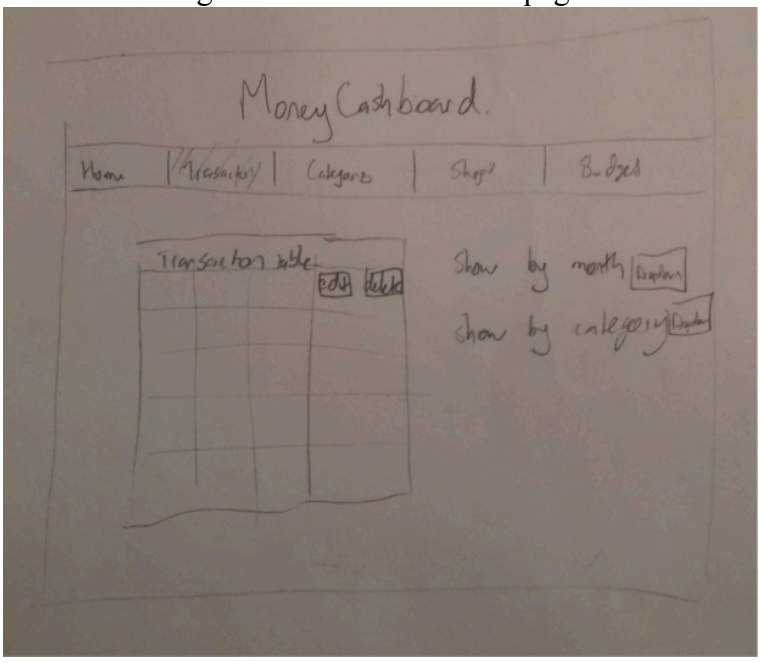
P5
Create a user site map



P6
Produce two wireframe designs
Wire frame diagram for my app’s homepage

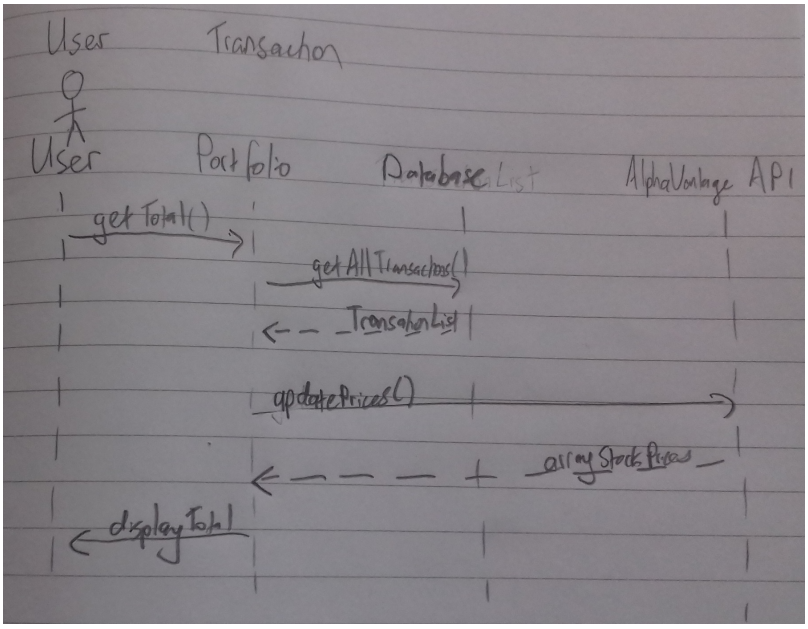


Wire frame diagram for the transactions page.

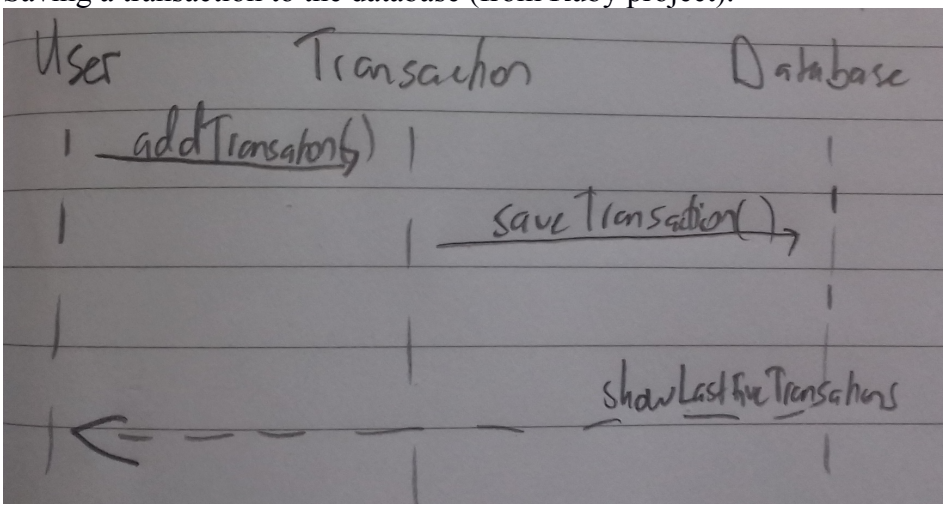


P7
Produce two system interaction diagrams (sequence and/or collaboration diagrams).

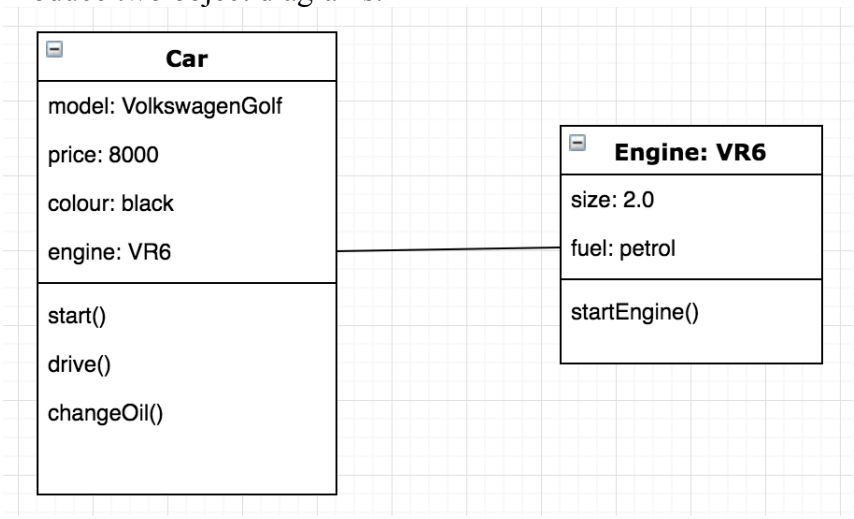
Viewing the total portfolio value (from JavaScript project)

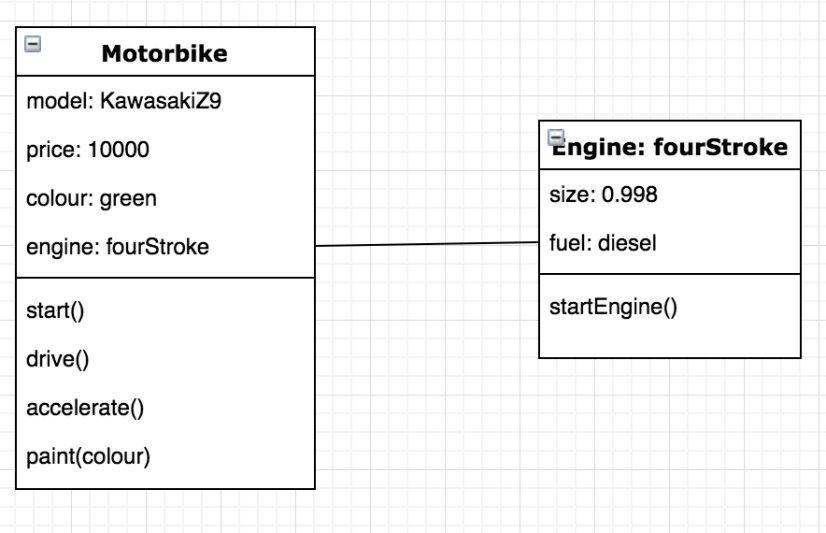


Saving a transaction to the database (from Ruby project).



P8
Produce two object diagrams.





P9
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.

This algorithm written in javascript calculates the number of dinosaurs there will be in the park for a given number of years in the future. I used this algorithm as each dinosaur in the park needs to be considered as they have different offspring rates. If a dinosaur has an offspring rate of 4, after a year the expected number of dinosaurs is the four offspring plus the original dinosaur. This is raised to the power of the number of years as each of the offspring will themselves produce offspring.

```
Park.prototype.calculateDinosaurs = function(years){
  let total = 0;
  for(let dino of this.enclosure){
    dinoTotal = (dino.offspringRate + 1) ** years;
    total += dinoTotal;
  }
  return total;
}
```

This algorithm removes all the dinosaurs of a given type by filtering all dinosaurs so only those not equal to the given type are put in an array. The dinosaurs you don't want have therefore been removed. The enclosure content is then set to this new array.

```
Park.prototype.removeAllDinosByType = function(type){
  let dinoTypeArray = this.enclosure.filter(dino =>
  dino.type !== type);
  this.enclosure = dinoTypeArray;
}
```

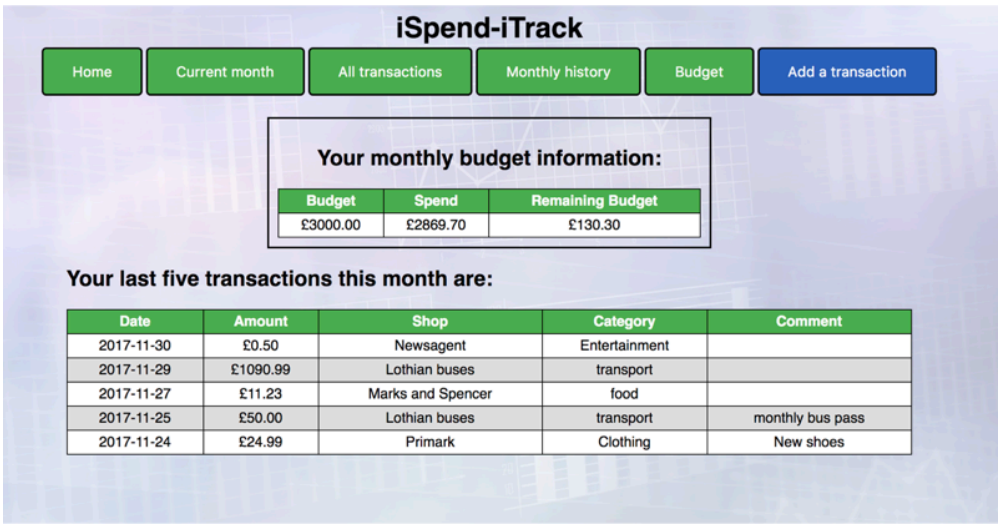
P10
Take a screenshot of an example of pseudocode for a function.
This is the pseudocode and java code for the move forward one step method of a robot. The robot is on a two dimensional (x,y) grid, and has an orientation, the direction it is facing, in addition to the initial position.

```
public void moveF(){
    // Move forward method
    // get the orientation of the robot
    //do a switch statement
    //if facing north increase the y coord by 1
    //if facing east increase the x coord by 1
    // if facing south decrease the y coord by 1
    //if facing west decrease the x cooord by 1


    switch(this.getOrientation()){
        case "N":{Integer newY = this.getPosition().get(1) +1 ;
            this.position.set(1, newY); }
        break;
        case "E": {Integer newX = this.getPosition().get(0) +1 ;
            this.position.set(0, newX); }
        break;
        case"S": {Integer newY = this.getPosition().get(1) -1 ;
            this.position.set(1, newY); }
        break;
        case "W": {Integer newX = this.getPosition().get(0) -1 ;
            this.position.set(0, newX); }
        break;
    }
}
```

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

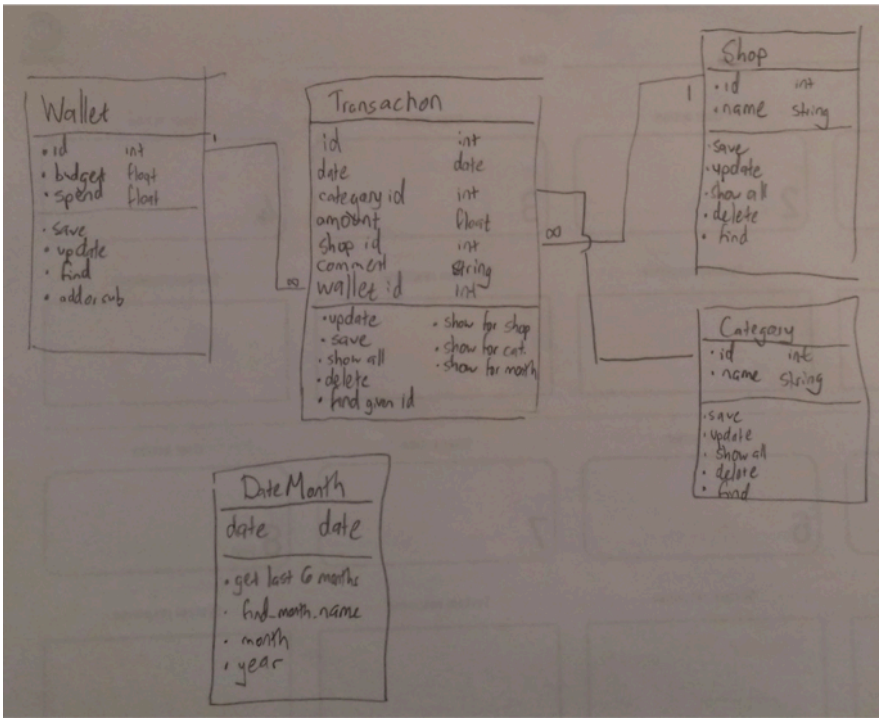
This is a ruby, sinatra, sql, html and css project. I built a money tracking app:

https://github.com/benjaminbowen00/ruby_sql_sinatra_project



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

Class diagrams. Initially the date was saved only as string, but this was later changed to a DateMonth object so that only transactions from the last six months could be shown:

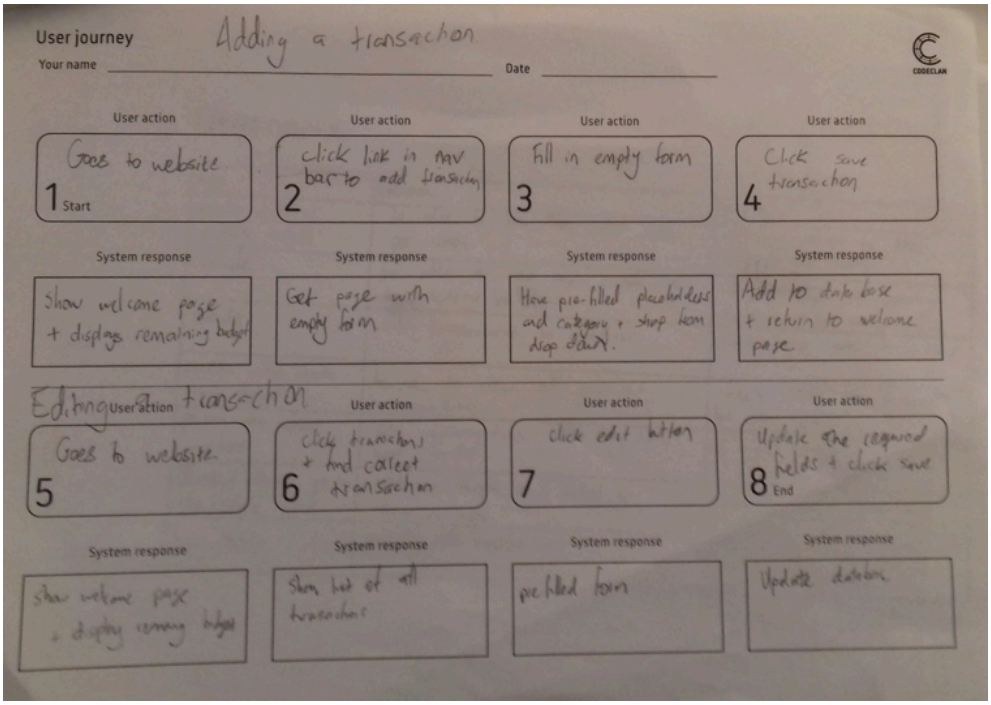


Planning the user needs for the app:

User needs

As a...	I want to...	So that...
person with a monthly budget	be able to see the total spent per month	manage money for the rest of the month.
spender	be able to see the amount spent by category for month	know if I am spending too much on a category
spender	add transactions to database	transactions are recorded
person who likes to track spending	be able to view all previous transactions	check where my budget has been spent.
person who makes mistakes	edit transactions after they have been saved	I can correct errors
doesn't always go to the same shop	add new shops to be linked to transactions	correctly record information in the database
busy person	quickly add transactions	I don't waste time just recording a transaction

An example of a user journey from the planning of the project:

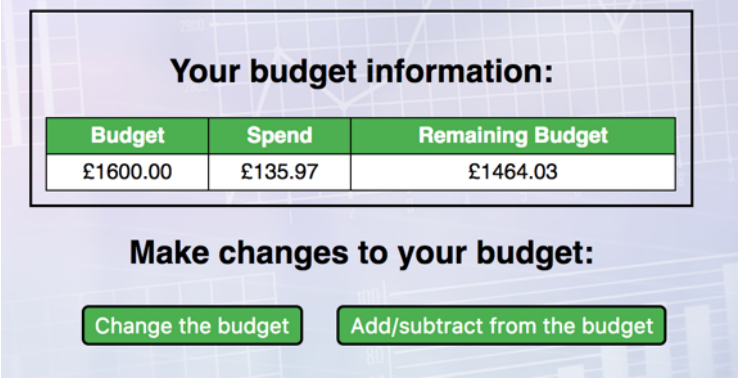


P13

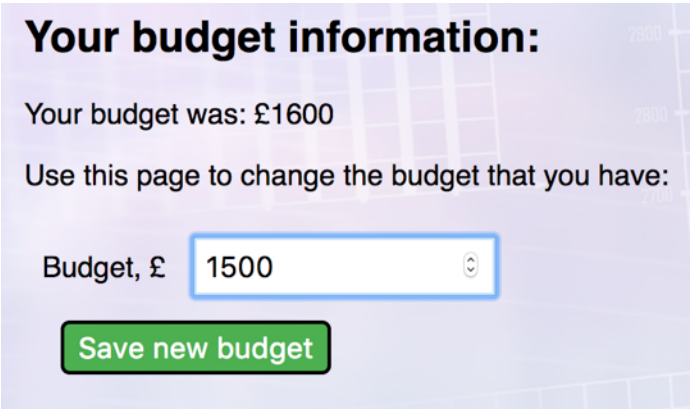
Show user input being processed according to design requirements. Take a screenshot of:

- The user inputting something into your program
- The user input being saved or used in some way

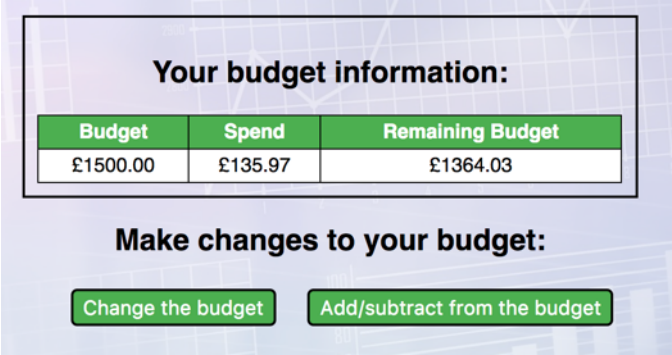
The monthly budget was initially £1600:



The user inputs the new budget of £1500:



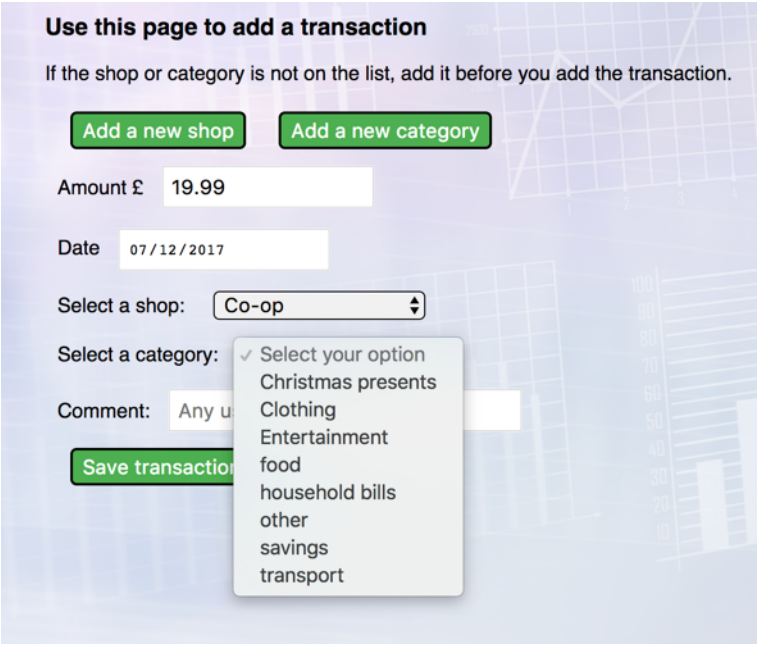
This new budget is then saved and displayed on the budget information page. The remaining budget is now £100 lower than before:

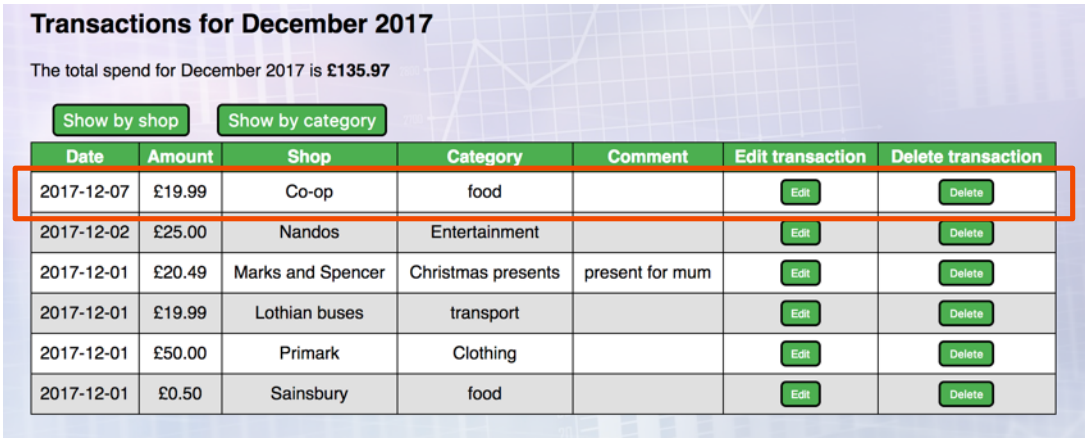


P14
Show an interaction with data persistence. Take a screenshot of:

- Data being inputted into your program
- Confirmation of the data being saved

A new transaction is added through completing the form which saves the record to the database:





Transactions for December 2017

The total spend for December 2017 is **£135.97**

Show by shop | Show by category

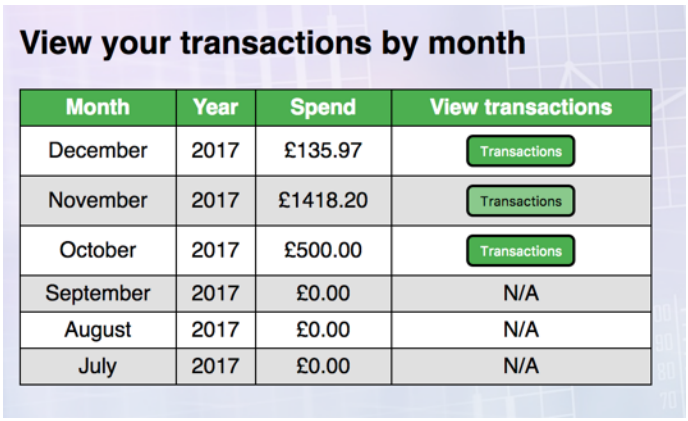
Date	Amount	Shop	Category	Comment	Edit transaction	Delete transaction
2017-12-07	£19.99	Co-op	food		Edit	Delete
2017-12-02	£25.00	Nandos	Entertainment		Edit	Delete
2017-12-01	£20.49	Marks and Spencer	Christmas presents	present for mum	Edit	Delete
2017-12-01	£19.99	Lothian buses	transport		Edit	Delete
2017-12-01	£50.00	Primark	Clothing		Edit	Delete
2017-12-01	£0.50	Sainsbury	food		Edit	Delete

This is then visible on the monthly transactions page:

P15
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

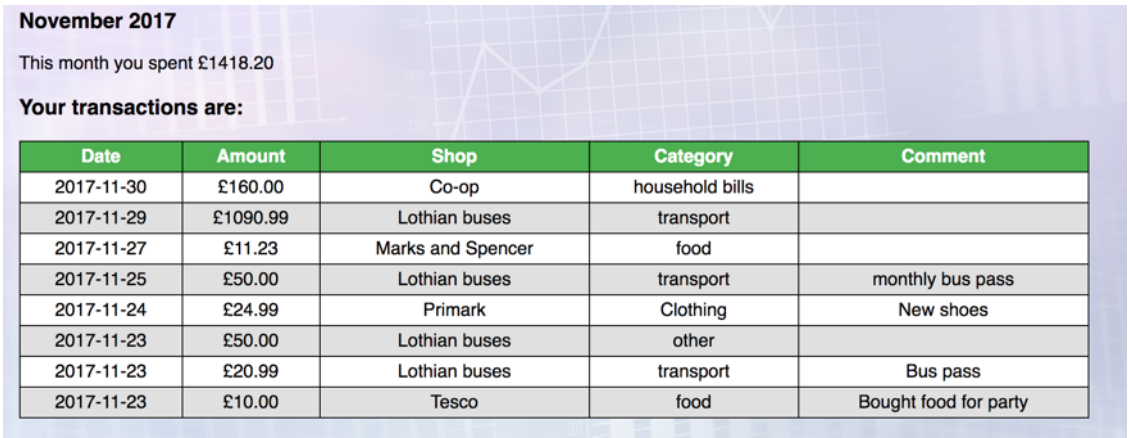
The user selects to view the transactions for November 2017 - the button changes colour when the mouse hovers over it.



View your transactions by month

Month	Year	Spend	View transactions
December	2017	£135.97	Transactions
November	2017	£1418.20	Transactions
October	2017	£500.00	Transactions
September	2017	£0.00	N/A
August	2017	£0.00	N/A
July	2017	£0.00	N/A

The page showing the transactions for November 2017 is returned:



November 2017

This month you spent £1418.20

Your transactions are:

Date	Amount	Shop	Category	Comment
2017-11-30	£160.00	Co-op	household bills	
2017-11-29	£1090.99	Lothian buses	transport	
2017-11-27	£11.23	Marks and Spencer	food	
2017-11-25	£50.00	Lothian buses	transport	monthly bus pass
2017-11-24	£24.99	Primark	Clothing	New shoes
2017-11-23	£50.00	Lothian buses	other	
2017-11-23	£20.99	Lothian buses	transport	Bus pass
2017-11-23	£10.00	Tesco	food	Bought food for party

P16
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

An HttpXML request is made and the result is set in the state as 'records'. The records are sent to the ChartList as props:

```
componentDidMount(){
  const url = 'https://itunes.apple.com/gb/rss/topsongs/limit=20/json';
  var xhr = new XMLHttpRequest();
  xhr.open('GET', url);
  xhr.addEventListener('load', ()=> {
    if(xhr.status !==200) {return};
    const jsonString = xhr.responseText;
    const data = JSON.parse(jsonString);
    this.setState({
      records: data.feed.entry
    })
  })
  xhr.send();
}

render(){
  return (
    <ChartList records={this.state.records}/>
  )
}
```

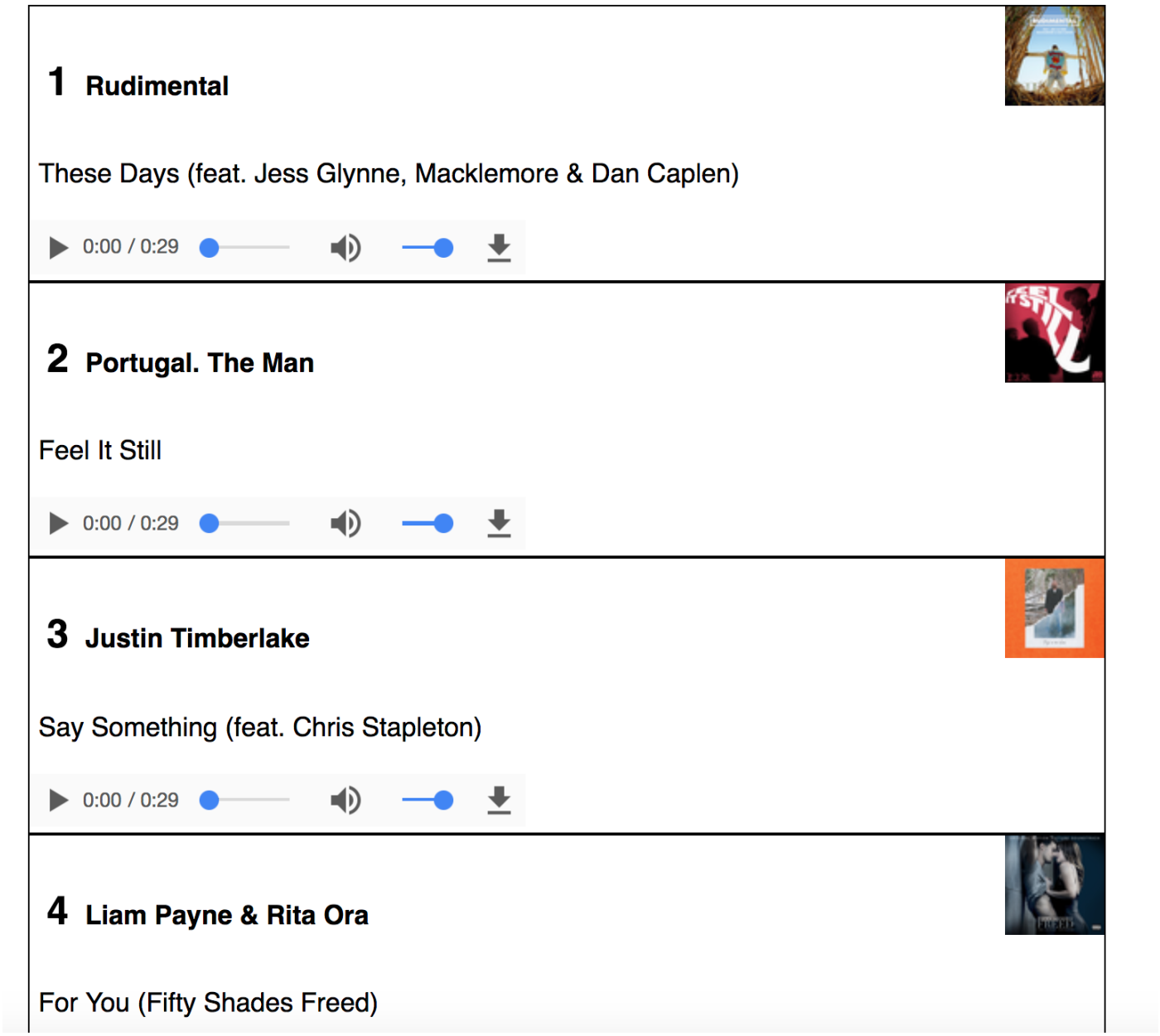
The ChartList component renders a div of records with props being sent to each Record:

```
25  class ChartList extends Component {
26
27    render(){
28      const recordNodes = this.props.records.map((record, index) => {
29        return (
30          <Record key={index} songTitle={record["im:name"]["label"]} artist={record["im:artist"]["label"]}
    •     image={record["im:image"][1]["label"]} audio={record["link"][1]["attributes"]["href"]}
    •     position={index + 1}>
31          </Record>
32        )
33      })
34      console.log(recordNodes);
35      return (
36        <div>
37          {recordNodes}
38        </div>
39      )
40    }
41  }
42
43  export default ChartList;
```

The Record component renders the information to be displayed in the app.

```jsx
import React from 'react';

const Record = (props) => {

  return (<div className="record-div">
    <div className="image-div"><img src={props.image} alt="company logo"/></div>
    <h4 className="inline-block position">{props.position}</h4>
    <p className="inline-block artist">{props.artist}</p>
    <p className="title">{props.songTitle}</p>
      <br></br>
    <audio controls>
      <source src={props.audio} type="audio/mpeg" />
      Your browser does not support the audio tag.
    </audio>
    <br></br>
  </div>
)
}

export default Record;

```

The resulting app looks like this:



Here is the start of the API being used:


```
{
  "feed": {
    "author": {
      "name": {
        "label": "iTunes Store"
      },
      "uri": {
        "label": "http://www.apple.com/uk/itunes/"
      }
    },
    "entry": [
      {
        "im:name": {
          "label": "These Days (feat. Jess Glynne, Macklemore & Dan Caplen)"
        },
        "im:image": [
          {
            "label": "http://is1.mzstatic.com/image/thumb/Music128/v4/dd/41/51/dd4151d2-2758-4f81-9adf-2585c096b453/190295686109.jpg/55x55bb-85.png",
            "attributes": {
              "height": "55"
            }
          },
          {
            "label": "http://is2.mzstatic.com/image/thumb/Music128/v4/dd/41/51/dd4151d2-2758-4f81-9adf-2585c096b453/190295686109.jpg/60x60bb-85.png",
            "attributes": {
              "height": "60"
            }
          },
          {
            "label": "http://is1.mzstatic.com/image/thumb/Music128/v4/dd/41/51/dd4151d2-2758-4f81-9adf-2585c096b453/190295686109.jpg/170x170bb-85.png",
            "attributes": {
              "height": "170"
            }
          }
```

P17
Produce a bug tracking report:
For my android food tracking app:

User can record the date for when they ate the food.	Fail	Use android's datePickerDialog to display a calendar the user can select from	Pass
The meal for a food must be of only pre-set values	Fail	Use a spinner that is filled with values from an enum rather than text input.	Pass
An empty page is not displayed if no results for food by date/meal type/search	Fail	If the array formed from the database search is of size zero then display a message to say there were no results found for the criteria	Pass.
Whole database of foods can be cleared	Fail	Create a new activity with a button linked to a Dao that does a 'DELETE FROM foods' on the database	Pass
User can view foods for a particular date	Fail	Create a new activity with a datePickerDialog to select the date and then run a Dao which does a 'SELECT * FROM foods WHERE date = :dateString' where dateString is taken from the datePickerDialog.	Pass

P18
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

A food object has an attribute of poisoned that is initially set to false:

```
const Food = function(foodName, replenishmentValue){
  this.foodName = foodName;
  this.replenishmentValue = replenishmentValue;
  this.poisoned = false;
}

Food.prototype.equals = function(food) {
  return this.foodName.toLowerCase() === food.foodName.toLowerCase();
}

module.exports = Food;
```

A test is written in mocha so that a Rat object can change the poisoned attribute of a Food object to true:

```
const assert = require("assert");
const Rat = require("../rat.js");
const Food = require("../food.js");


describe("Rat test", function(){

  let rat1;
  let food1;

  beforeEach(function(){
    rat1 = new Rat("Ratty")
    food1 = new Food("Cheese sandwich", 10);
  })

  it.only('rat can make food poisoned', function(){
    rat1.touchFood(food1);
    assert.strictEqual(food1.poisoned, true);
  })

})
```

The test is run but fails because the method has not been written yet:

```
[➜  homework_heroes_rats git:(master) ✗ npm test

> homework_heroes_rats@1.0.0 test /Users/benjaminbowen/codeclan_work/week_11/homework_heroes_rats
> mocha specs

true
false


  Rat test
    1) rat can make food poisoned


  0 passing (5ms)
  1 failing

  1) Rat test
       rat can make food poisoned:
     TypeError: rat1.touchFood is not a function
      at Context.<anonymous> (specs/rat_spec.js:17:10)


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

The 'touchFood' is added as a Rat method.

```
const Rat = function(name){
  this.name;
}

Rat.prototype.touchFood = function (food) {
  food.poisoned = true;
};

module.exports = Rat;

```

The test is now re-run and passes as food1's poisoned attribute has been changed to true:

```
➜  homework_heroes_rats git:(master) ✗ npm test

> homework_heroes_rats@1.0.0 test /Users/benjaminbowen/codeclan_work/week_11/homework_heroes_rats
> mocha specs

true
false


  Rat test
    ✓ rat can make food poisoned


  1 passing (4ms)
```

Here are tests for Food being compared to each other based on having the same name:

```
describe("Food", function(){

  let food1;
  let food2;
  let food3;

beforeEach(function(){
  food1 = new Food("Cheese sandwich", 10);
  food2 = new Food("Ham sandwich", 5);
  food3 = new Food("Cheese sandwich", 7);
})

it.only("can compare food when the same", function(){
  let foodBoolean = food1.equals(food3);
  assert.strictEqual(foodBoolean, true )
})

it.only("can compare foods when not the same", function(){
  let foodBoolean = food1.equals(food2);
  assert.strictEqual(foodBoolean, false )
})

})

```

Initially they fail:

```
➜ homework_heroes_rats git:(master) ✗ npm test

> homework_heroes_rats@1.0.0 test /Users/benjaminbowen/codeclan_work/week_11/homework_heroes_rats
> mocha specs



  Food
    1) can compare food when the same
    2) can compare foods when not the same


  0 passing (8ms)
  2 failing

  1) Food
       can compare food when the same:
     TypeError: food1.equals is not a function
      at Context.<anonymous> (specs/food_spec.js:18:27)

  2) Food
       can compare foods when not the same:
     TypeError: food1.equals is not a function
      at Context.<anonymous> (specs/food_spec.js:23:27)


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

The equals() method is added to the Food prototype, which converts the food name to lower case and then compares them:

```
1  const Food = function(foodName, replenishmentValue){
2    this.foodName = foodName;
3    this.replenishmentValue = replenishmentValue;
4    this.poisoned = false;
5  }
6
7  Food.prototype.equals = function(food) {
8    return this.foodName.toLowerCase() === food.foodName.toLowerCase();
9  }
10
11 module.exports = Food;
```

The tests now pass:

```
➜ homework_heroes_rats git:(master) ✗ npm test

> homework_heroes_rats@1.0.0 test /Users/benjaminbowen/codeclan_work/week_11/homework_heroes_rats
> mocha specs



  Food
    ✓ can compare food when the same
    ✓ can compare foods when not the same


  2 passing (5ms)
```