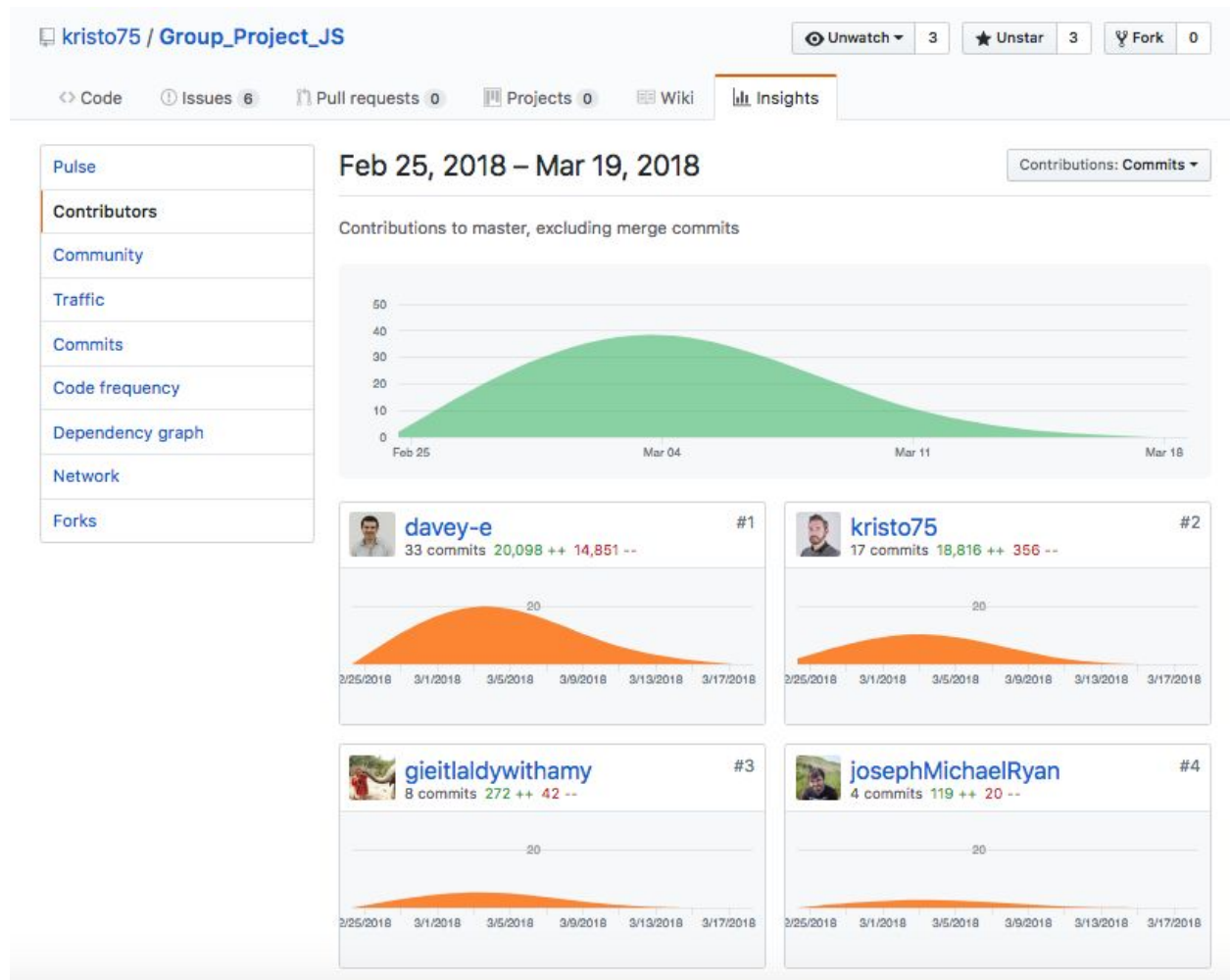


Project Unit
Amy Morrison
Cohort 18

P1



P2

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.

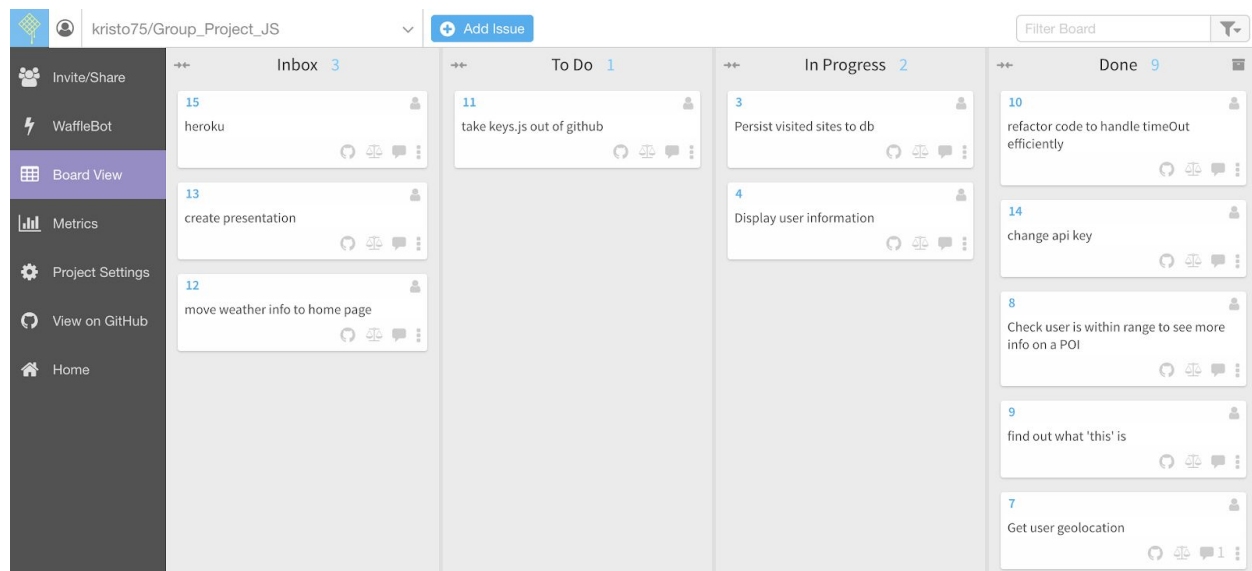
Your task is to make an MVP to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app. You might use an API to bring in content or a database to store facts. The topic of the app is your choice, but here are some 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

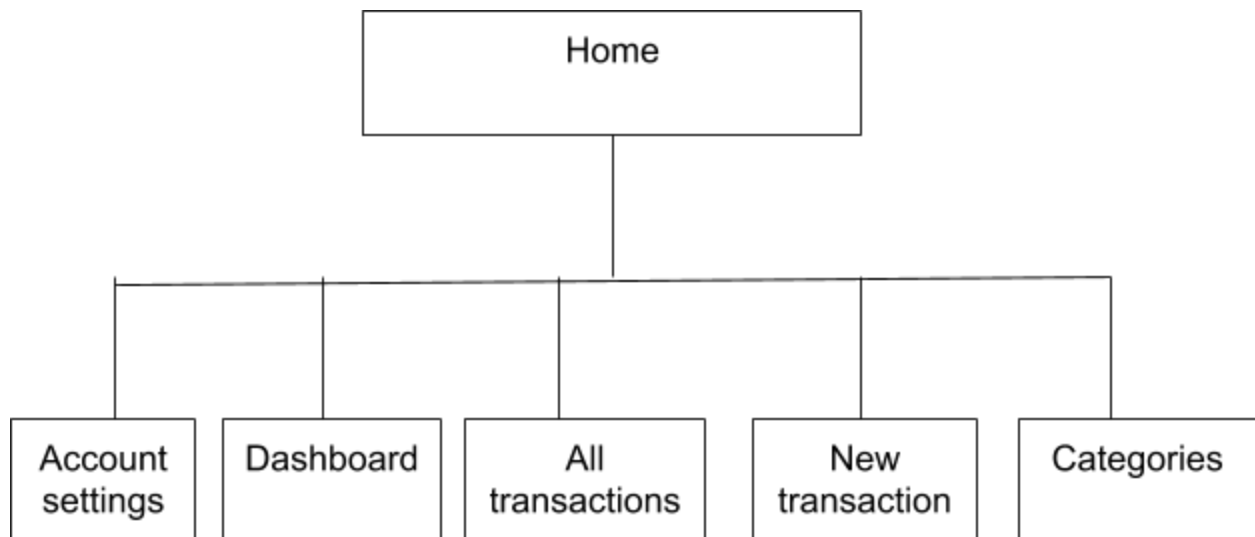
P3



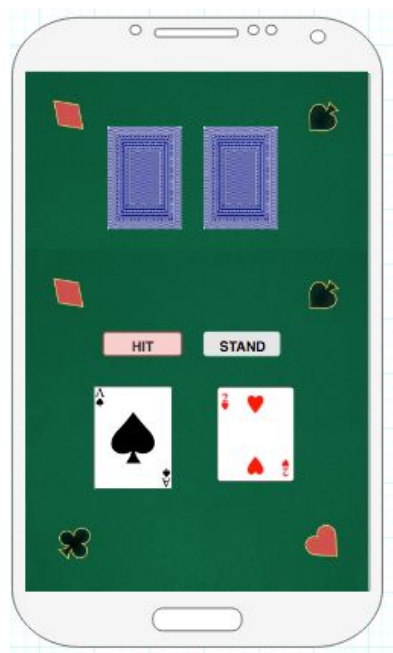
P4

Acceptance Criteria	Expected Result/Output	Pass/Fail
A user is able to click on a button(in the shape of a betting chip) to raise bet	User's current bet within betting section is raised by 5 pounds	Pass
A user is able to click on double down when they have two cards	User's bet is doubled and they receive another card face up	Fail
A user is able to click stand	User receives no more cards and dealer picks up cards until the sum of their hand is 17 when all of their cards are revealed and both player's hands are compared to reveal winner	Pass

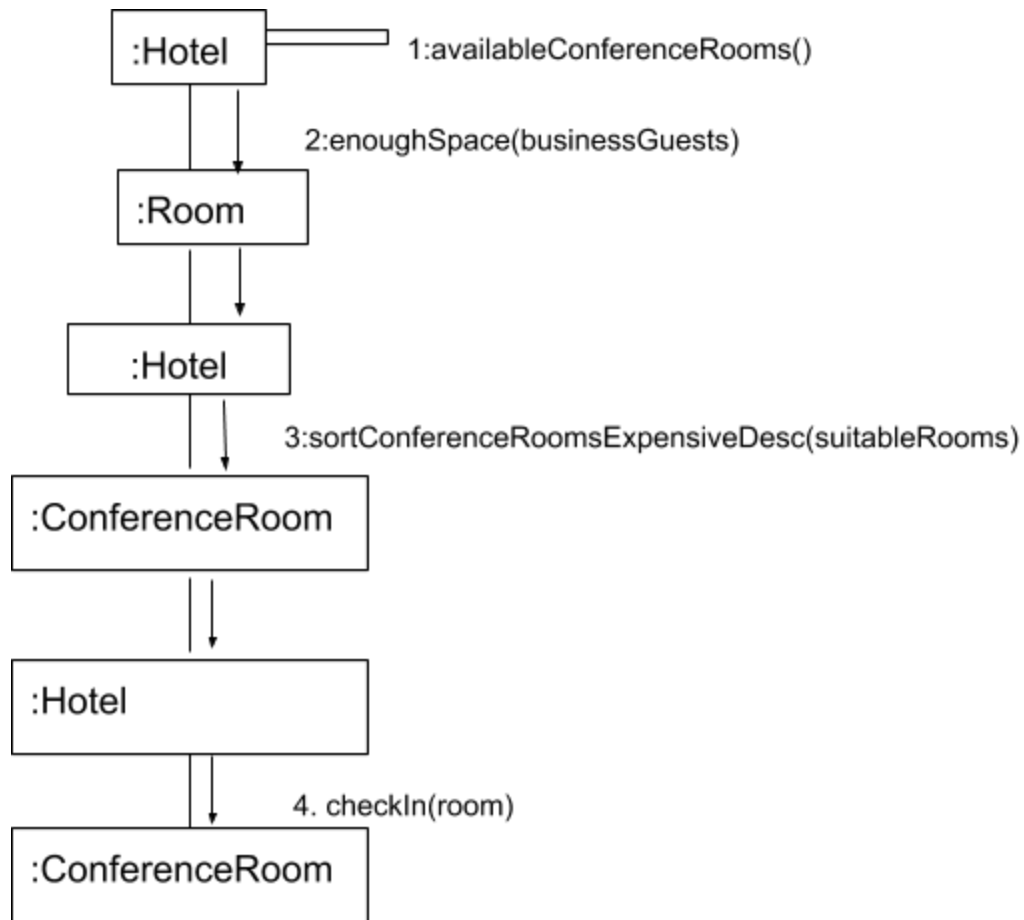
P5

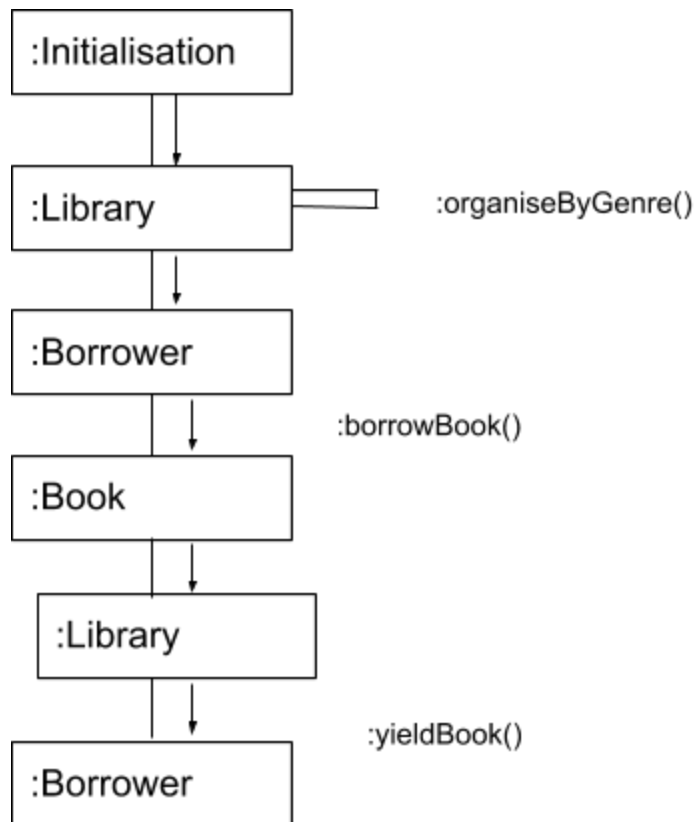


P6

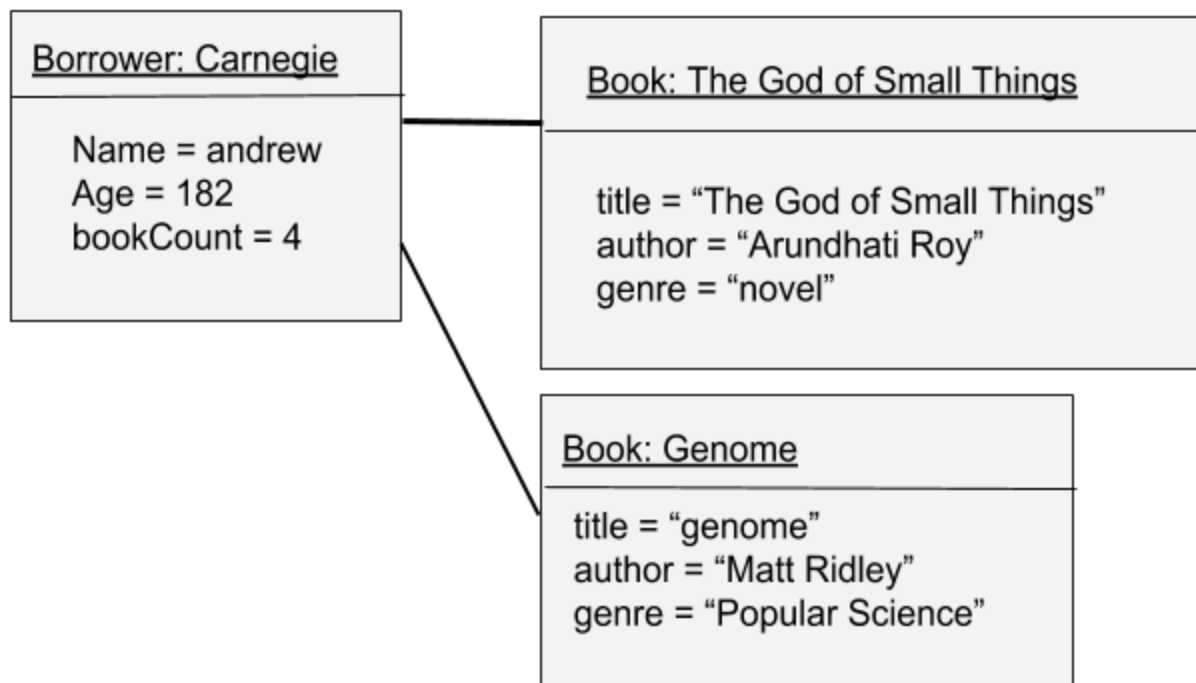


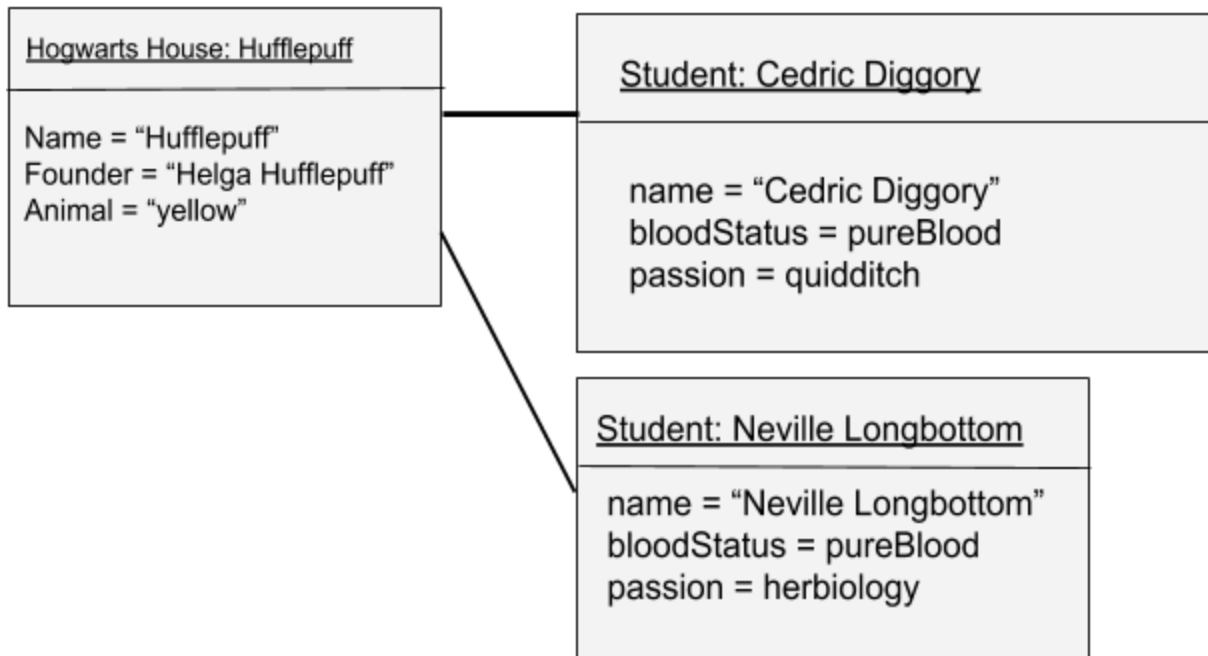
P7





P8





P9

```
Hero.prototype.eat = function(food){  
  let revilitisation;  
  if (this.favouriteFood === food){  
    revilitisation = food.replenishmentValue * 1.5;  
  } else {  
    revilitisation = food.replenishmentValue;  
  }  
  
  food.poisonous ? this.health -= revilitisation : this.health += revilitisation;  
}
```

This algorithm takes in a food parameter and updates the heroes health according to the revilitisation level of such food. This algorithm was needed because food has different effects on a hero according to whether it's poisonous, if it is the hero's favourite food and if both.

```

public boolean shouldHit() {
    return handValue() <= 16;
}

public void move() {
    turnOverHiddenCard();
    while (shouldHit()){
        drawCard(dealCard());
    }
}

```

This algorithm is called when a dealer is blackjack plays (i.e. after all players have played). This algorithm was chosen because a dealer should keep drawing cards until their hand value is 17 or more. A dealer also has to start by turning over their face down card. ??

```

public ConferenceRoom findExpensiveConferenceRoom(ArrayList<Guest> businessGuests) {
    ArrayList<ConferenceRoom> suitableRooms = availableConferenceRooms();
    suitableRooms.removeIf(conferenceRoom -> !conferenceRoom.enoughSpace(businessGuests));
    System.out.println(suitableRooms);
    suitableRooms = sortConferenceRooms(suitableRooms);
    System.out.println(suitableRooms);
    return suitableRooms.get(0);
}

```

This algorithm takes in an ArrayList of Guests and finds the most expensive room that can accommodate the group. This algorithm finds rooms that are not currently occupied and have enough space. This algorithm was chosen to ensure maximum profit was made on every conference room booking.

P10

```

//Return the monthly average spend on travel over a year

//initialise mean to be zero
//iterate through spending travel array
//for each entry add spending[travel] value / 12 to mean
//return mean

```

P11

Github link: <https://github.com/gieitlaldywithamy/BlackjackAndroid>


```
1 package com.codeclan.ammymorrison.toptrumps.controllers;
2
3 import ...
4
5 /**
6  * Created by ammymorrison on 29/01/2018.
7  */
8
9 public class DealerCardAdapter extends BaseAdapter {
10
11     private final Context mContext;
12     private ArrayList<Card> cards;
13
14     public DealerCardAdapter(Context context, ArrayList<Card> cards) {
15         mContext = context;
16         this.cards = cards;
17     }
18
19     @Override
20     public int getCount() { return this.cards.size(); }
21
22     @Override
23     public Object getItem(int position) { return this.cards.get(position); }
24
25     @Override
26     public long getItemId(int position) { return position; }
27
28     //a bit hacky...
29     public void refresh(ArrayList<Card> cards)
```

P12

To Do	Doing	Done
Build a deck of 52 cards which start off face down	+ Add a card	+ Add a card
Allow user to bet		
Game logic		
User to stand functionality		
Hit functionality		
Save user's current bet to local storage		
Allow user to double down (double bet) after dealer has turned over a card		
Grid layout for dealer cards		
Handle dealer's hole card		
Replace text views with toasts		
+ Add another card		

To Do

Save user's current bet to local storage

Allow user to double down (double bet) after dealer has turned over a card

Grid layout for dealer cards

Handle dealer's hole card

Replace text views with toasts

Handle Ace implementation

Add animation for bet button

+ Add another card

Doing

Allow user to bet

+ Add another card

Done

Build a deck of 52 cards which start off face down

Game logic

User to stand functionality

Hit functionality

+ Add another card

Blackjack Android ☆ Personal Private

To Do

Allow user to double down (double bet) after dealer has turned over a card

Replace text views with toasts

Handle Ace implementation

Add animation for bet button

Add cash out button

Enter a title for this card...

Add Card

Doing

+ Add a card

Done

Build a deck of 52 cards which start off face down

Game logic

User to stand functionality

Hit functionality

Save user's current bet to local storage

Allow user to bet

Handle dealer's hole card

Grid layout for dealer cards

+ Add another card

P13

[illegible]

P14

New Category

Name

[Save](#)

Categorys


- rent
- electricity bill
- gas bill
- phone bill
- gym membership
- groceries
- eating out
- coffee
- alcohol
- socialising
- presents
- credit card bill
- shopping
- travel
- lazy travelling
- amazon spending
- subscription box

```
.piggybank=# select * from categories;
```

id	name	luxury
1	rent	f
2	electricity bill	f
3	gas bill	f
4	phone bill	f
5	gym membership	f
6	groceries	f
7	eating out	t
8	coffee	t
9	alcohol	t
10	socialising	t
11	presents	t
12	credit card bill	t
13	shopping	t
14	travel	t
15	lazy travelling	t
16	amazon spending	
17	subscription box	
18	stationary	

(18 rows)

P15



[Eleanor](#)
[Dashboard](#)
[Transactions](#)
[Add entry](#)
[Categories](#)
[Account settings](#)
[Log Out](#)

Eleanor's PiggyBank £2000.00 left

Description
 How much?
 Transaction date
 Merchant
 Category

Save



[Dashboard](#)
[Transactions](#)
[Add entry](#)
[Categories](#)
[Account settings](#)
[Log Out](#)

Eleanor's PiggyBank £1996.80 left

You earn £24000.00 a year thus your monthly budget is £2000.00

March: You have spent £3.20 and have £1996.80 left in your budget.

You have spent most on coffee

Check monthly spend for

Date	Description	Amount	Category	Merchant	
13/3/2018	Brewlab	3.20	coffee	Brewlab	Details

P16

```

BeerList.prototype = {
  populate: function() {
    var url = 'https://api.punkapi.com/v2/beers';
    var request = new XMLHttpRequest();
    request.open("GET", url);
    request.addEventListener('load', function () {
      if (request.status === 200) {
        var jsonString = request.responseText;
        var beers = JSON.parse(jsonString);
        this.beers = beers;
        this.onUpdate(beers);
      }
    }).bind(this);
    request.send();
  }
}

```

```
const dropdown = document.querySelector('#beer-list-dropdown');
beers.forEach(function(beer, currentIndex){
  const option = document.createElement('option');
  option.innerText = beer.name;
  option.value = currentIndex;
  dropdown.appendChild(option);
});
```



P17

User's marker on map should update according to user's location	FAIL	Rather than creating a new marker every time geolocate function returns a location, store the user marker on an object and update that marker to be attached to new latitude longitude	PASS
Weather icon should update according to user's location and respective weather	FAIL	Made a separate API call to get the current city closest to user's latitude and longitude coordinates and use that city in weather API call	PASS
Duplicate points of interest should only be displayed once	FAIL	When the API is called, a safeguard is built in so that multiple entries with the same point of interest id are eliminated and only 1 is kept	PASS
The site should be mobile responsive	FAIL	Used flexbox to display the map, and added breakpoints so that text and markers are larger on mobile.	PASS

P18


```

const assert = require('assert')
const arrayTasks = require('./array_tasks.js')

describe('Array tasks', function () {

  it('should concatenate two arrays, returning a new array', function () {
    const arr1 = [1, 2, 3]
    const arr2 = [4, 5, 6]
    const expectation = [1, 2, 3, 4, 5, 6]
    assert.deepStrictEqual(arrayTasks.concat(arr1, arr2), expectation)
  })

  it('should insert an item in an array at any index position', function () {
    const arr = [1, 2, 4]
    assert.deepStrictEqual(arrayTasks.insertAt(arr, 3, 2), [1, 2, 3, 4])
  })

  it('should square all values in an array, returning a new array', function () {
    const arr = [1, 2, 3, 4, 5]
    assert.deepStrictEqual(arrayTasks.square(arr), [1, 4, 9, 16, 25])
  })

  it('should calculate the sum of all values in an array', function () {
    const arr = [1, 2, 3, 4, 5]
    assert.strictEqual(arrayTasks.sum(arr), 15)
  })

  it('should remove all instances of a value from an array, returning a new array', function () {
    const arr = [1, 2, 3, 1, 4, 5, 1]
    assert.deepStrictEqual(arrayTasks.removeAndClone(arr, 1), [2, 3, 4, 5])
  })
})

```

Array tasks

- ✓ should concatenate two arrays, returning a new array
- ✓ should insert an item in an array at any index position
- ✓ should square all values in an array, returning a new array
- 1) should calculate the sum of all values in an array
- ✓ should remove all instances of a value from an array, returning a new array
- ✓ should find all occurrences of a value, returning an array of index positions
- ✓ should calculate the sum of all of even numbers in an array squared
- ✓ EXTENSION - should find duplicate values in an array, returning a new array of the duplicates

7 passing (30ms)

1 failing

1) Array tasks

should calculate the sum of all values in an array:

```

AssertionError: 120 === 15
+ expected - actual

```

```

-120
+15

```

at Context.<anonymous> (tests.js:25:10)

```

const arrayTasks = {

  concat: function (arr1, arr2) {
    return arr1.concat(arr2);
  },

  insertAt: function (arr, itemToAdd, index) {
    arr.splice(index, 0, itemToAdd);
    return arr;
  },

  square: function (arr) {
    return arr.map(function(element){
      return element*element;
    });
  },

  sum: function (arr) {
    return arr.reduce(function(runningTotal, value){
      return runningTotal * value;
    });
  },

  removeAndClone: function (arr, valueToRemove) {
    const arrClone = arr.filter(function(value){
      return value !== valueToRemove;
    });
    return arrClone;
  },
};

```

Array tasks

- ✓ should concatenate two arrays, returning a new array
- ✓ should insert an item in an array at any index position
- ✓ should square all values in an array, returning a new array
- 1) should calculate the sum of all values in an array
- ✓ should remove all instances of a value from an array, returning a new array
- ✓ should find all occurrences of a value, returning an array of index positions
- ✓ should calculate the sum of all of even numbers in an array squared
- ✓ EXTENSION - should find duplicate values in an array, returning a new array of the duplicates

7 passing (30ms)

1 failing

1) Array tasks

should calculate the sum of all values in an array:

AssertionError: 120 === 15
+ expected - actual

-120
+15

at Context.<anonymous> (tests.js:25:10)


```
sum: function (arr) {  
  return arr.reduce(function(runningTotal, value){  
    return runningTotal * value;  
  });  
},
```

```
sum: function (arr) {  
  return arr.reduce(function(runningTotal, value){  
    return runningTotal + value;  
  });  
},
```

> mocha tests

Array tasks

- ✓ should concatenate two arrays, returning a new array
- ✓ should insert an item in an array at any index position
- ✓ should square all values in an array, returning a new array
- ✓ should calculate the sum of all values in an array
- ✓ should remove all instances of a value from an array, returning a new array
- ✓ should find all occurrences of a value, returning an array of index positions
- ✓ should calculate the sum of all of even numbers in an array squared
- ✓ EXTENSION – should find duplicate values in an array, returning a new array of the duplicates

8 passing (21ms)