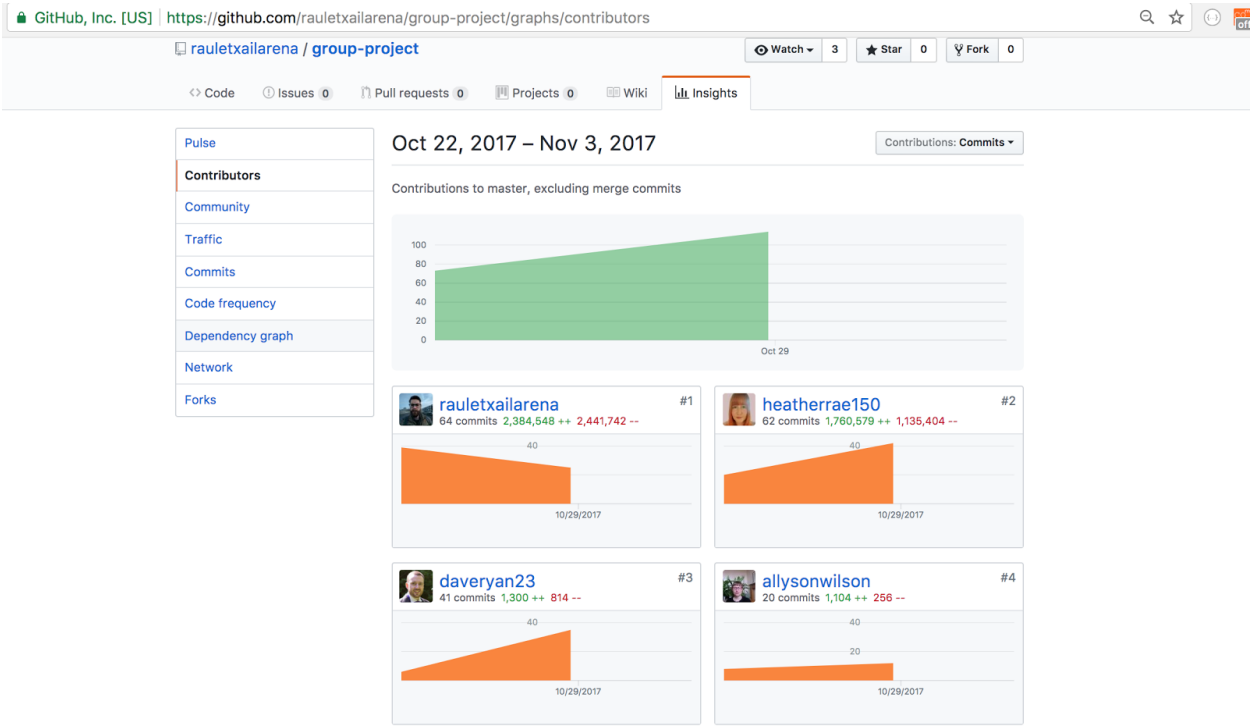
Allyson Wilson
Cohort 15
10/11/2017

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

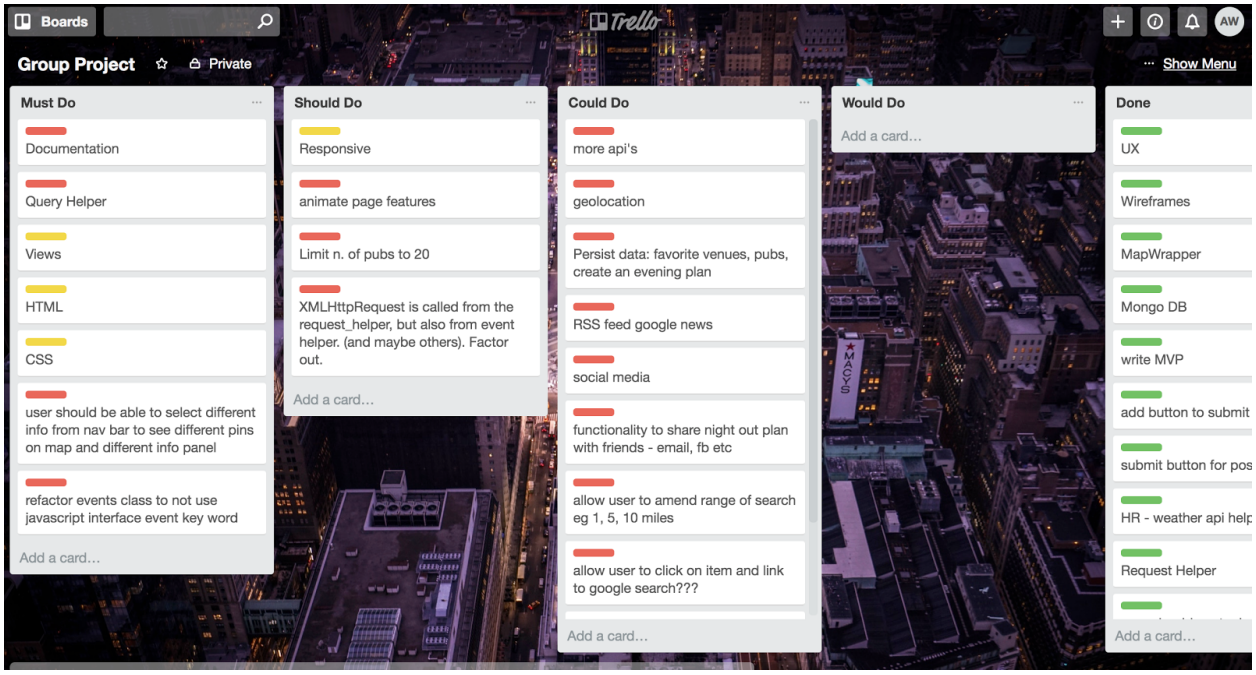
MVP

- 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

Some samples of existing apps for inspiration:

```
# Our MVP

Should be able to take in user input and display it in a map

Should display information about the area
  pubs
  weather
  events

Interactivity - user can click on map or input postcode or city and retrieve
accurate information according to input.

Extensions

restaurants
crime???
historical data???
bus routes
taxi
```

P3



P4 Acceptance Criteria Test Plan

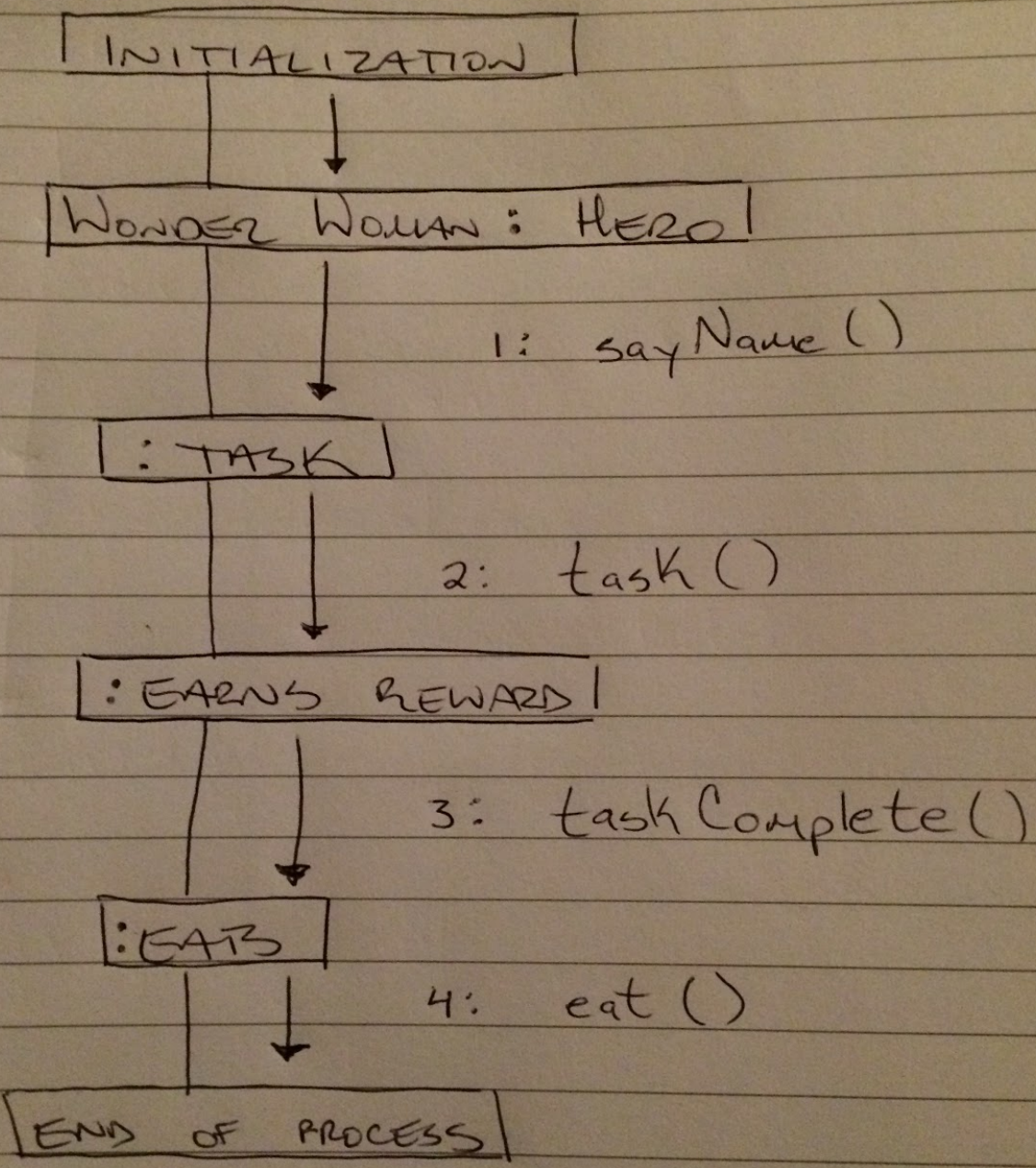
Acceptance Criteria	Expected Result/Output	Pass/Fail
User can bring up a map of an area by inputting a postal code	Users will obtain map information relative to post code entered	Pass
User can choose what kind of information relative to location they would like to see on map	Users will be able to click a button representing information type and see it represented on map	Pass
User will be able to get further information on selections made	User can click on map markers to see details	Pass
User can plan out their evening by saving selections to favourites	Users selections will be saved and data persisted	Pass
User can share plan for their evening out with friends using social media	User will be able to click on social media buttons to access their accounts and share their plan.	Fail
User can find out about events information for the area selected	Users will see a list of events taking place in the area in the next 24 hours	Pass

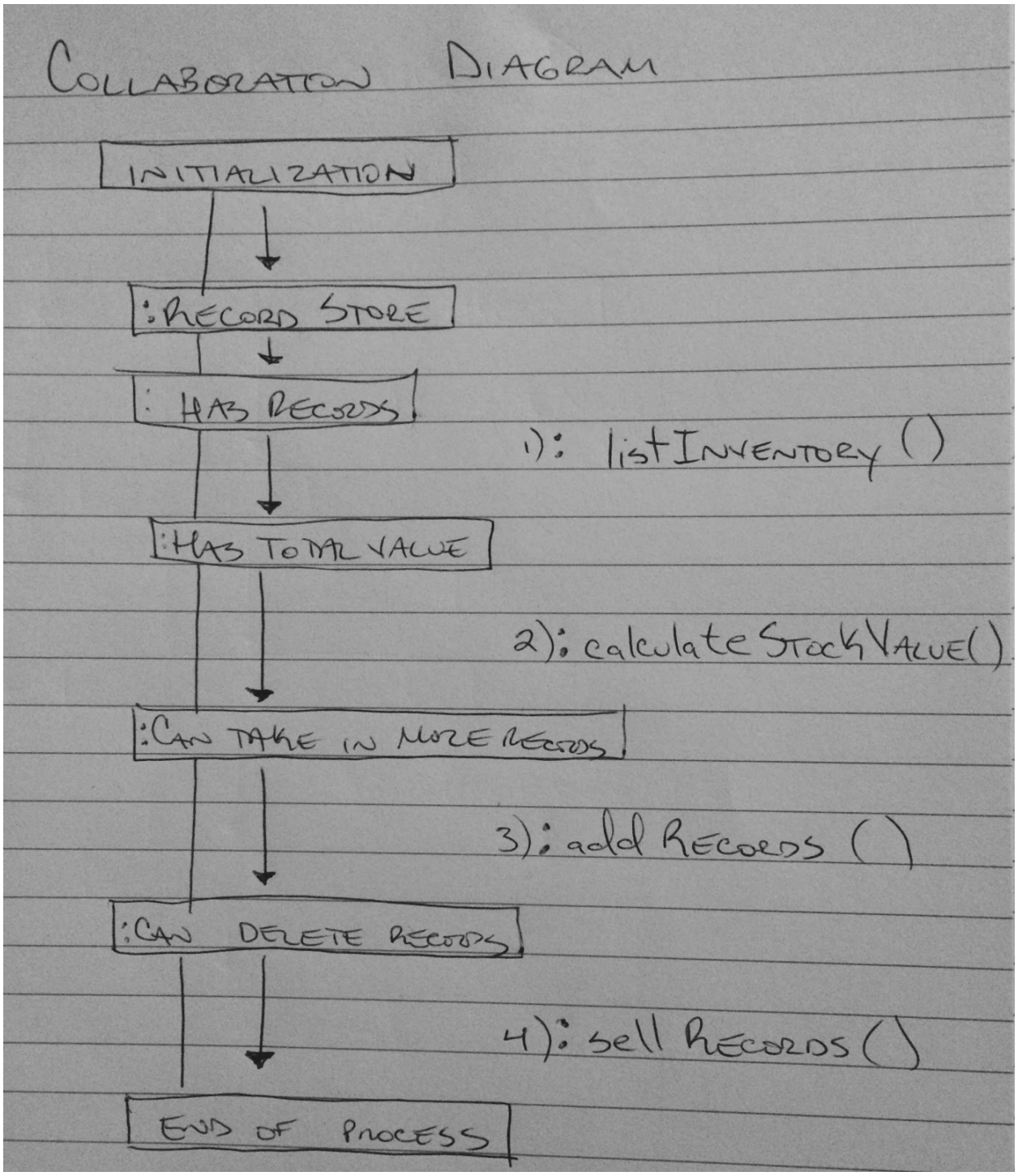
P5 User Sitemap

P6 Wireframes

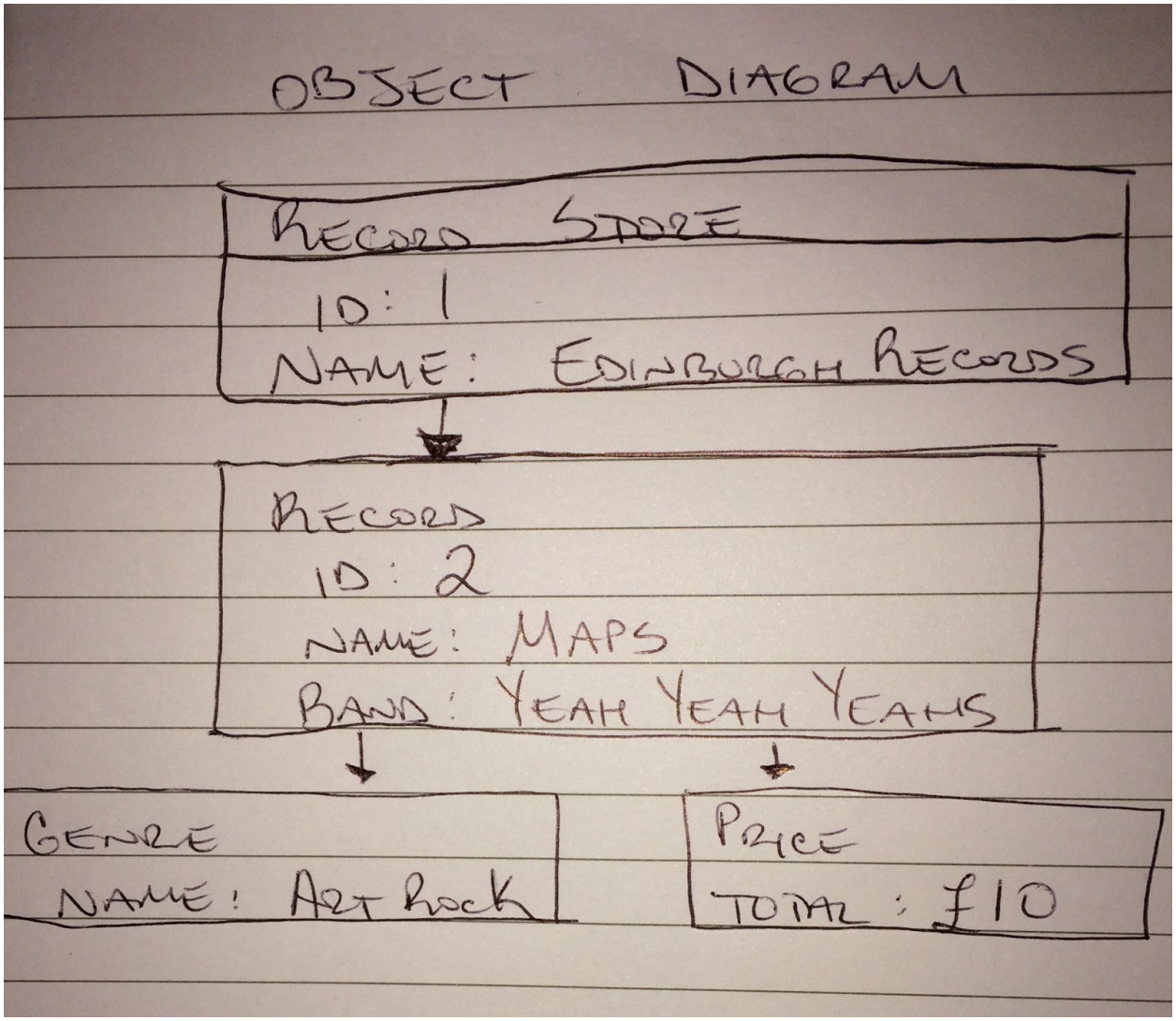
P7 System Interaction Diagrams

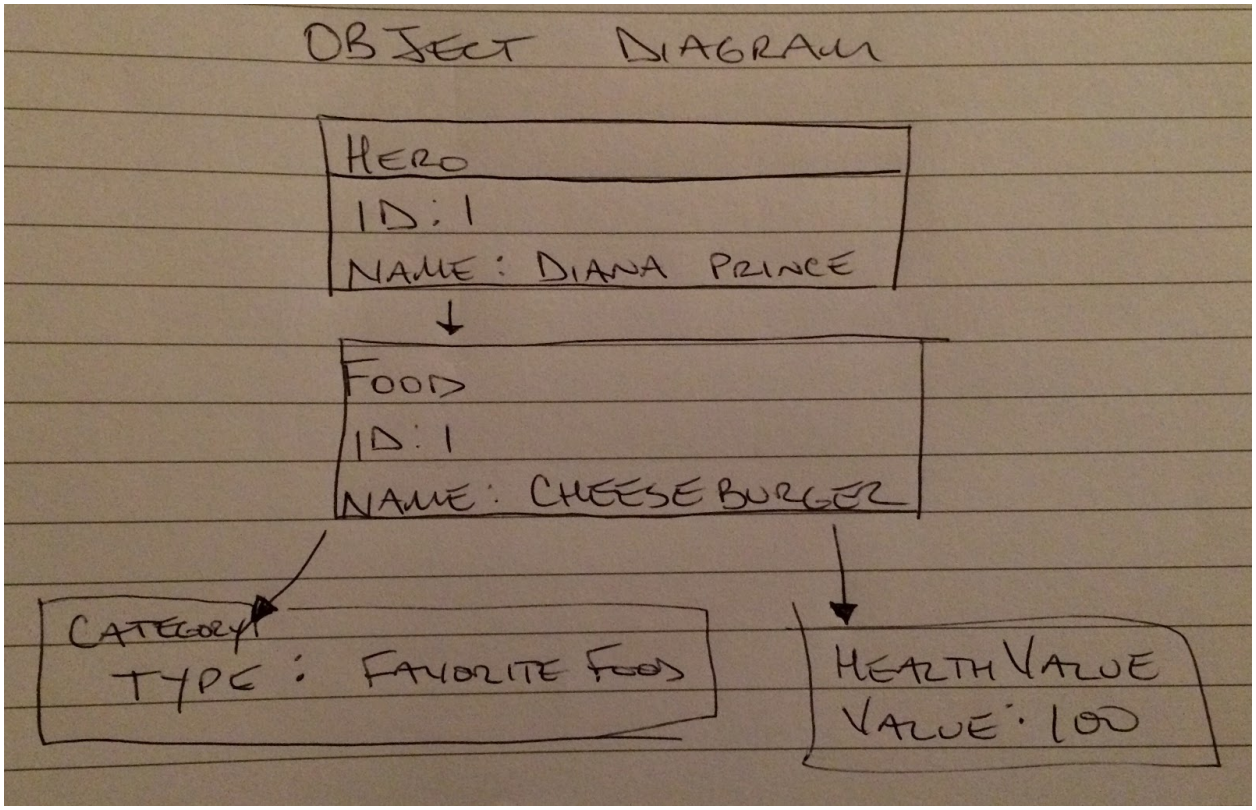
Collaboration Diagram





P8 Object diagrams


OBJECT DIAGRAM
RECORD STORE
ID: 1
NAME: EDINBURGH RECORDS
RECORD
ID: 2
NAME: MAPS
BAND: YEAH YEAH YEAHS
GENRE
NAME: ART ROCK
PRICE
TOTAL: £10


OBJECT DIAGRAM
HERO
ID: 1
NAME: DIANA PRINCE
FOOD
ID: 1
NAME: CHEESEBURGER
CATEGORY
TYPE: FAVORITE FOOD
HEALTH VALUE
VALUE: 100

P9

```java
public class Ward {
    public ArrayList<Patient> rooms;
    public int capacity;
    public Gender gender;


    public Ward(Gender gender){
        this.rooms = new ArrayList<>();
        this.capacity = 10;
        this.gender = gender;
    }

    public ArrayList<Patient> getRooms  () {
            return rooms;
    }
    public Gender getGender() { return gender; }

    public int getCapacity() { return capacity; }

    public void admitPatient(Patient patient) {
        if (rooms.size() < capacity) {
            rooms.add(patient);
        }

    }

    public void dischargePatient(Patient patient) { rooms.remove( patient ); }


    public void admitCorrectGender(Patient patient) {
        if (patient.gender == this.gender) {
            rooms.add( patient );
        }
    }

}
```

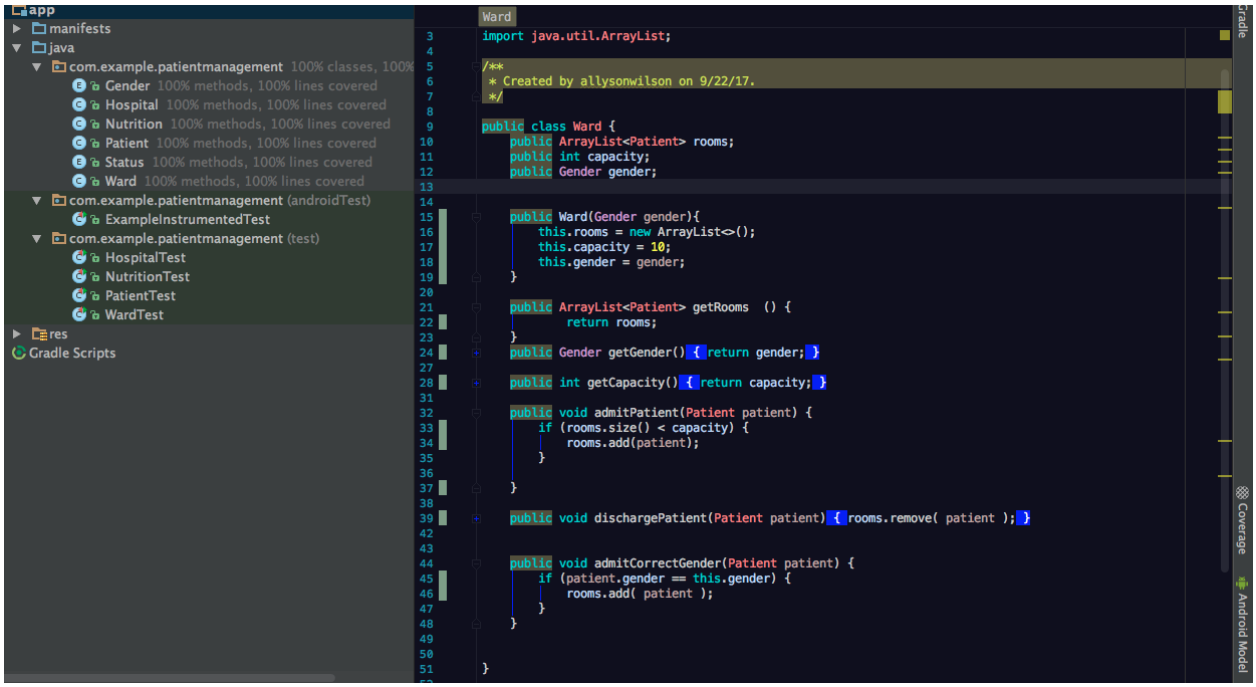
The admitPatient algorithm compares the patients currently admitted to a room, to the capacity of the room and will not allow additional patients to be admitted. This helped me reach my MVP for my Java project.

The admitCorrectGender algorithm uses an if statement to compare two enums, the patient's gender with the room's designated gender. This would allow only female patients to be admitted to the maternity ward. This was an extension I was able to do above and beyond the MVP for my Java project.

P 10 PSEUDOCODE

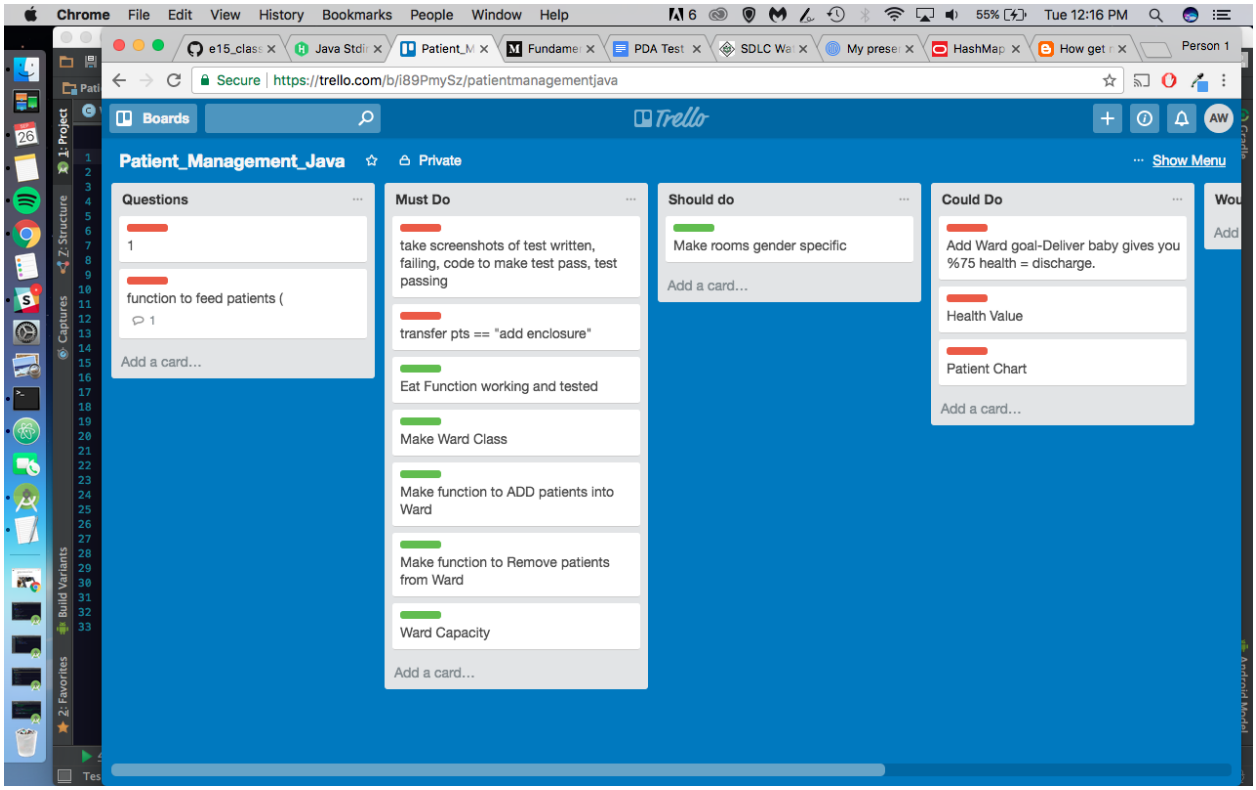
```
need to make adoptions/new page
2 drop down boxes for heroes and animals

once selections made
adoption button which will add new pair to adoptions page

will need animal id joined to hero id to create adoption function
```

P 11 Screenshot of solo project

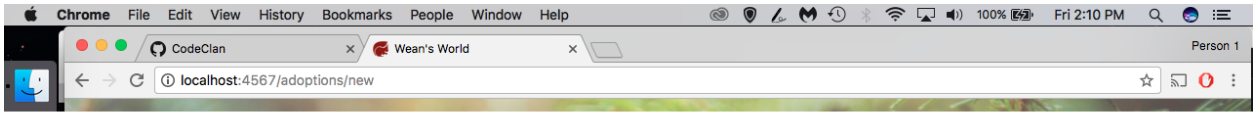


https://github.com/allysonwilson/Patient_Management_Java_Project

P 12 photo of planning

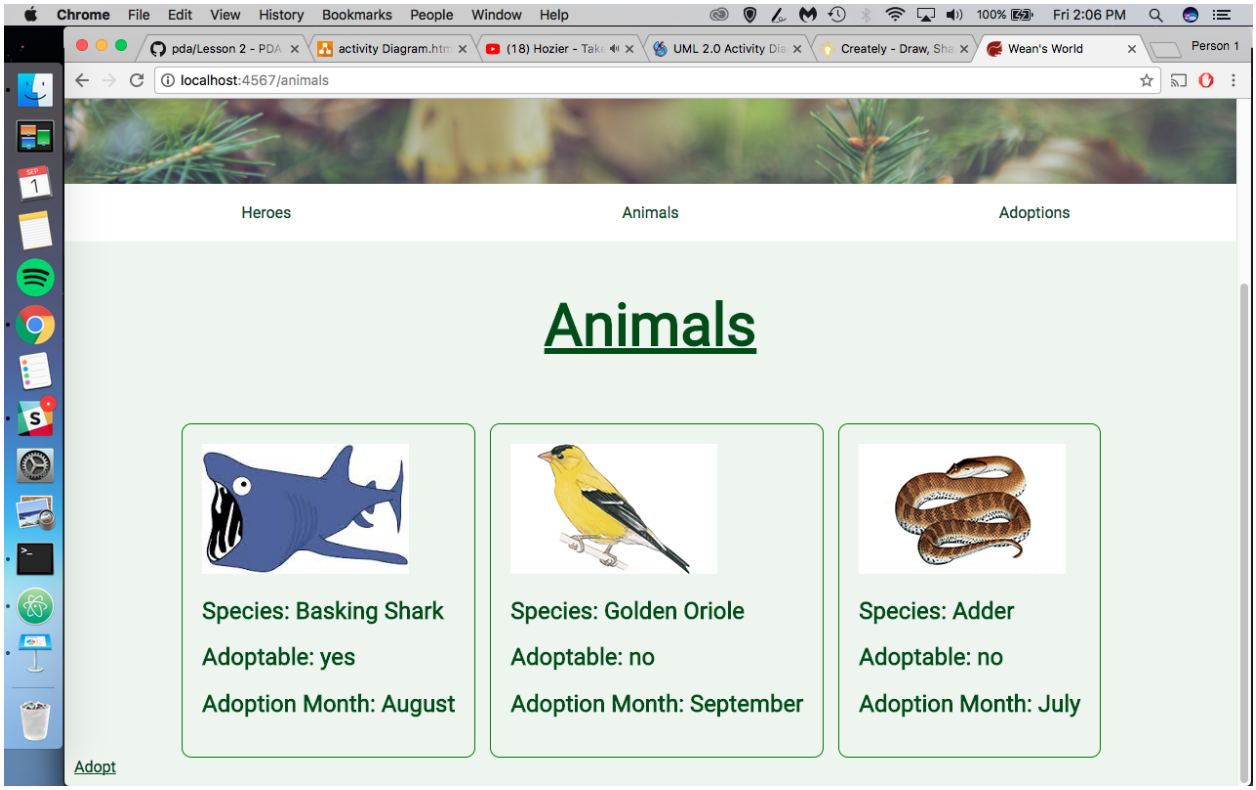


P 13 USER INPUT



P 14 DATA PERSISTENCE

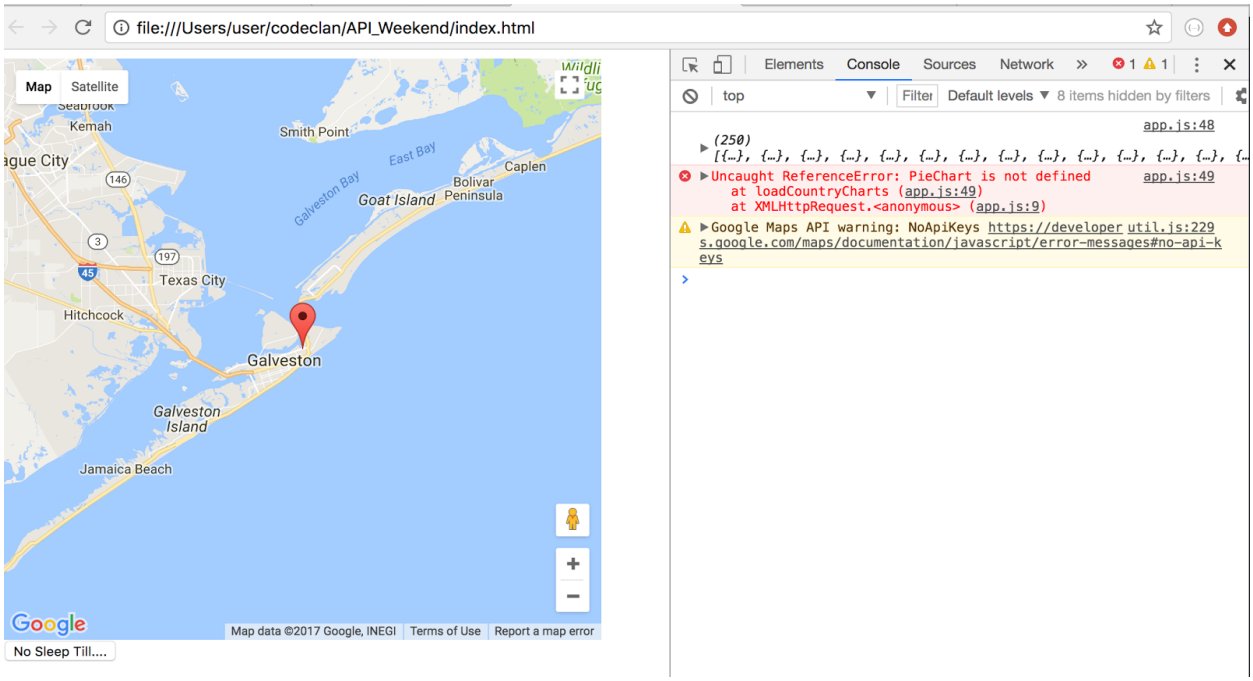
P 15 RESULTS/ FEEDBACK TO USER



P 16 Code that implements API

```
53
54  var initialize = function(){
55    var mapDiv = document.getElementById('main-map');
56    var center = { lat: 29.313  , lng: -94.7766 };
57    var mainMap = new MapWrapper(mapDiv, center, 10);
58    mainMap.addMarker(center,"Galveston,TX my birthplace.");
59    // mainMap.addClickEvent();
60
61    var noSleepTillBrooklyn = function () {
62      var brooklyn = { lat:40.665535 , lng: -73.96974};
63      mainMap.googleMap.setCenter(brooklyn);
64    }
65
66    var brooklynButton = document.getElementById('brooklyn');
67    brooklynButton.addEventListener('click', noSleepTillBrooklyn);
68  }
69
70  window.addEventListener('load', initialize);
```

P 16 API being used whilst program is running.



P17 Bug Tracking Report

User must be able to see local weather forecast	Failed	Used another api that allowed post code to be a search query rather than latitude and longitude	Passed
Event has a location and a description	Failed	Had to access the location by using different keys of JSON object	Passed
User must be able to save an event, pub, or restaurant to favourites	Failed	Had to create model objects in MongoDB	Passed
Events must be shown for next 24 hours	Failed	Had to find correct way to query API and structure the URL	Passed
User must be able to delete from favourites	Failed	Create a remove button with a function to remove item from MongoDB	Passed

P 18 Testing

Test code

```
var assert = require("assert");
var Hero = require("../hero.js");
var Task = require("../task.js");
var Food = require("../food.js");

beforeEach(function () {
  wonderWoman = new Hero("Diana Prince", 100, "cheese burger");
  flyAirplane = new Task(5,10, 100);
  eatCheeseBurger = new Task(-5 , 100, 5);
});

describe("Hero", function() {

  it("should have a name", function() {
    assert.strictEqual(wonderWoman.name, "Diana Prince");
  });

  it("should have a favouriteFood", function() {
    assert.strictEqual(wonderWoman.favouriteFood, "cheese burger");
  });

  it("should have a health value", function() {
    assert.strictEqual(wonderWoman.healthValue, 100);
  });

  it("should be able to say name", function() {
    assert.strictEqual(wonderWoman.sayName(), "My name is Diana Prince!");
  });

  it("should be able add task", function() {
    wonderWoman.addTask(flyAirplane);
    assert.strictEqual(wonderWoman.toDo.length, 1);
  });

  it("should be able remove task", function() {
    wonderWoman.addTask(flyAirplane);
    wonderWoman.addTask(flyAirplane);
    wonderWoman.addTask(eatCheeseBurger);
    wonderWoman.removeTask(0, 1);
    assert.strictEqual(wonderWoman.toDo.length, 2);
  });

```

Tests Failing

```
➜  hero_rat git:(master) ✗ mocha specs --reporter nyan
 8   -_-_-_-_-_-_-_-_,------,
 3   -_-_-_-_-_-_-_-_|  /\_/\
 0   -_-_-_-_-_-_-_-~|_( x .x)
     -_-_-_-_-_-_-_-  ""   ""

  8 passing (17ms)
  3 failing

  1) Hero
       should be able to say name:
     TypeError: wonderWoman.sayName is not a function
      at Context.<anonymous> (specs/hero_spec.js:27:36)

  2) Hero
       should be able add task:
     TypeError: wonderWoman.addTask is not a function
      at Context.<anonymous> (specs/hero_spec.js:31:17)

  3) Hero
       should be able remove task:
     TypeError: wonderWoman.addTask is not a function
      at Context.<anonymous> (specs/hero_spec.js:36:17)
```

Corrected Code

hero.js | hero_spec.js

```
var Hero = function (name, health, favouriteFood) {
  this.name = name;
  this.healthValue = 100;
  // shouldn't I be able to set this = to healthValue?
  this.favouriteFood = favouriteFood;
  this.toDo = [];
  // this.wallet = [];
};

Hero.prototype.sayName = function () {
  return ("My name is " + this.name + "!");
};

Hero.prototype.addTask = function (task) {
  this.toDo.push(task);
};

Hero.prototype.removeTask = function (index , deleteCount) {
  this.toDo.splice(index , deleteCount);
};

Hero.prototype.eat = function () {
  var newHealthValue = food.replenishmentValue += this.healthValue;
  return newHealthValue;
};
```

The test code passing

```
    devDependencies : {
      "mocha": "^4.0.1"
    },
    "dependencies": {},
    "repository": {
      "type": "git",
      "url": "git+https://github.com/allysonwilson/hero_rat.git"
    },
    "bugs": {
      "url": "https://github.com/allysonwilson/hero_rat/issues"
    },
    "homepage": "https://github.com/allysonwilson/hero_rat#readme",
    "description": ""
}


Is this ok? (yes)
➜  hero_rat git:(master) ✗ mocha specs --reporter nyan
 11  -_-_-_-_-_-_-_,------,
 0   -_-_-_-_-_-_-_|  /\_/\
 0   -_-_-_-_-_-_-~|_( ^ .^)
     -_-_-_-_-_-_-  ""  ""

  11 passing (14ms)

➜  hero_rat git:(master) ✗
```