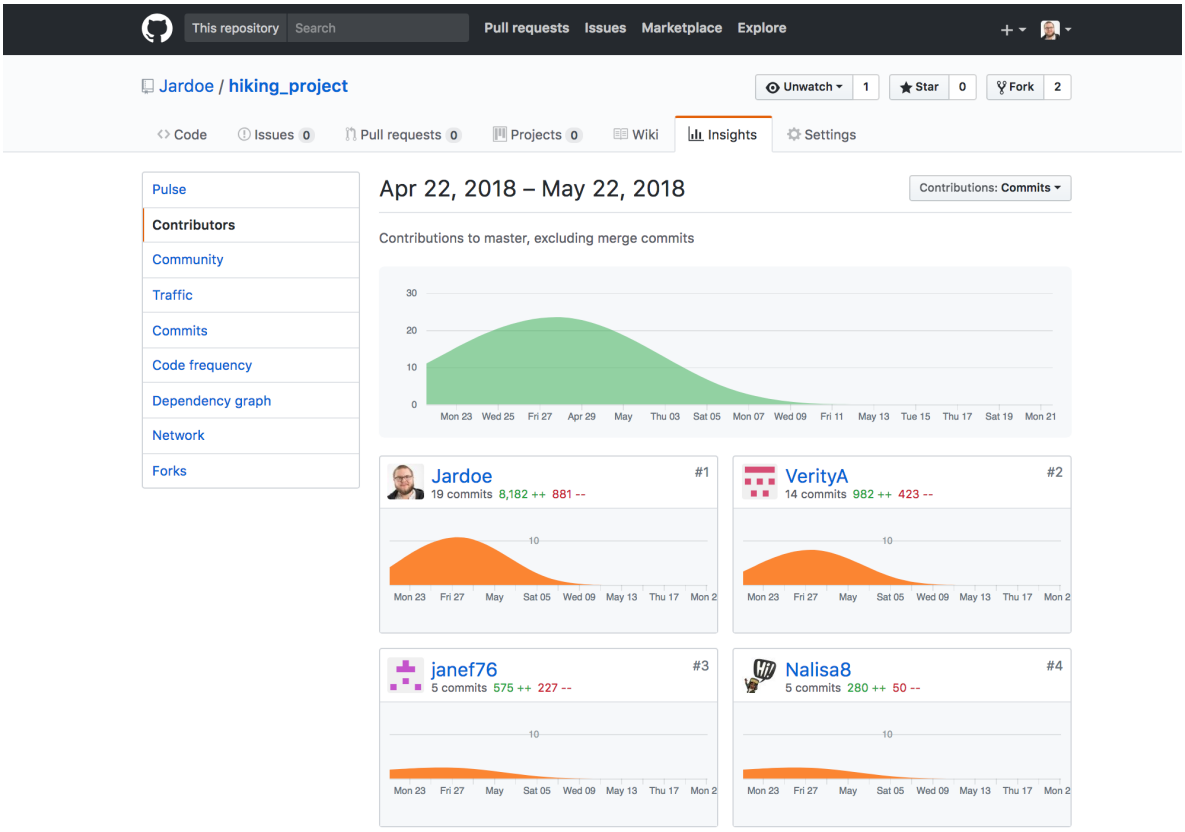
Evidence for Project Unit

Name: Jussi Pardoe
Cohort: e19

P.1 Github Contributors Page



P.2 Project Brief

Hiking Tracker Group Project

Visit Scotland are look for ways to encourage people to walk and cycle. Your task is to create an app that allows users to search for cycling and hiking routes, view routes on a map, save routes to a wishlist and mark a route done.

You could use GoogleMaps Directions API:

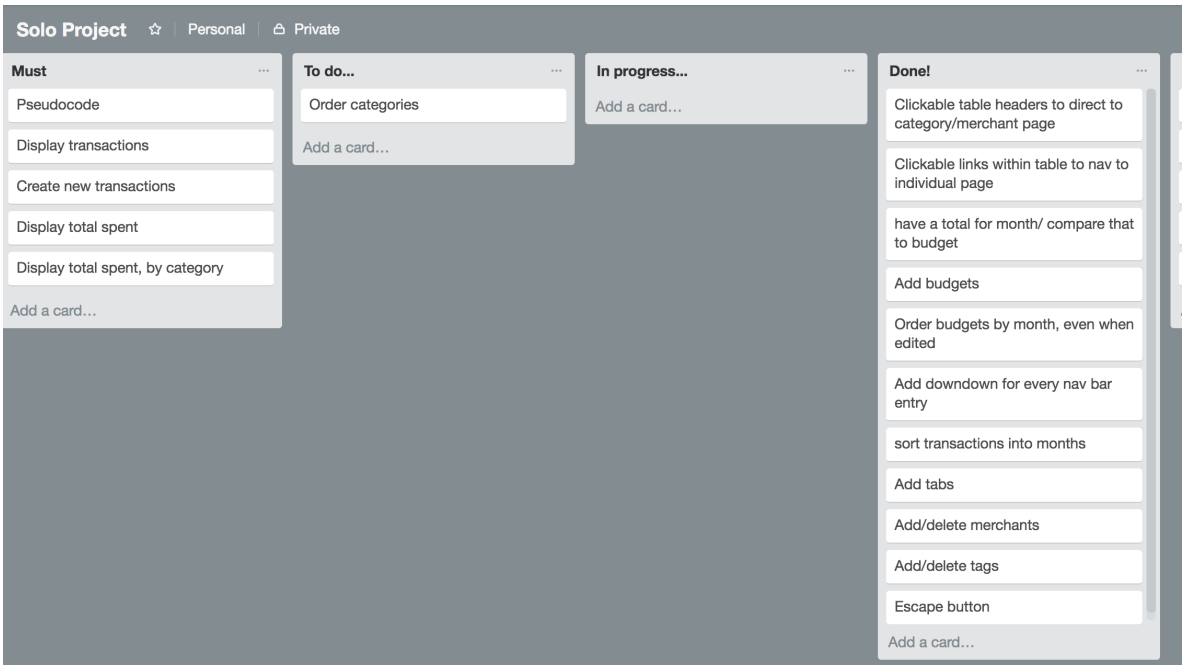
https://developers.google.com/maps/documentation/directions/

MVP

Users should be able to:

- Select start and finish locations for their route
- Save routes to a wishlist
- Mark completed routes as 'done'

P.3 Use of Trello

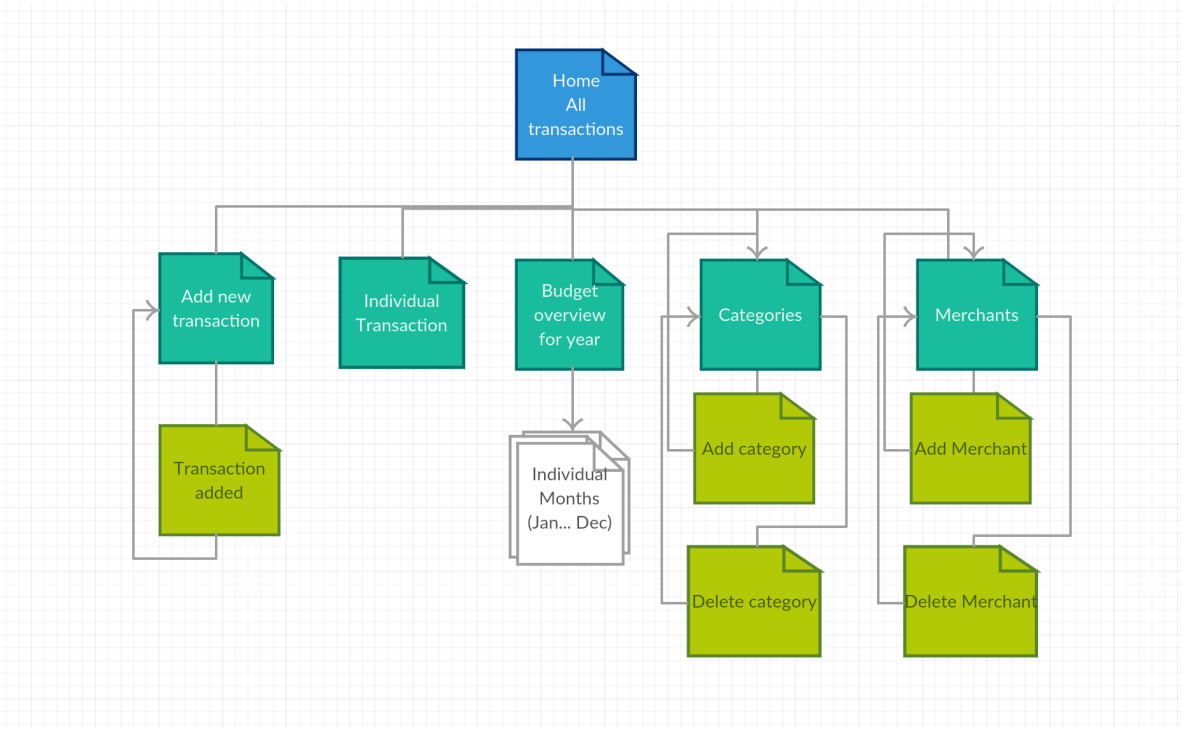


P.4 Acceptance Criteria

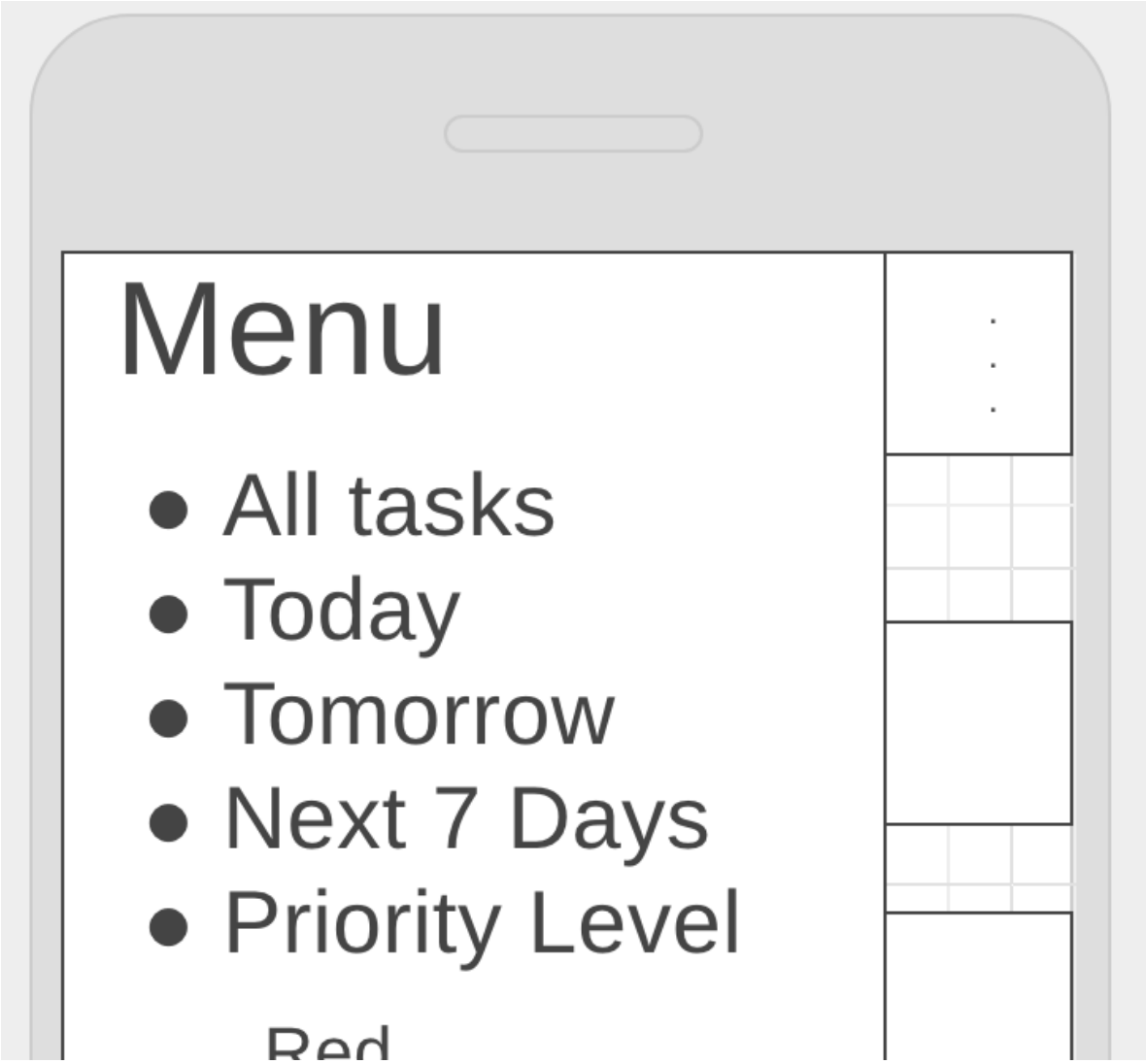
.

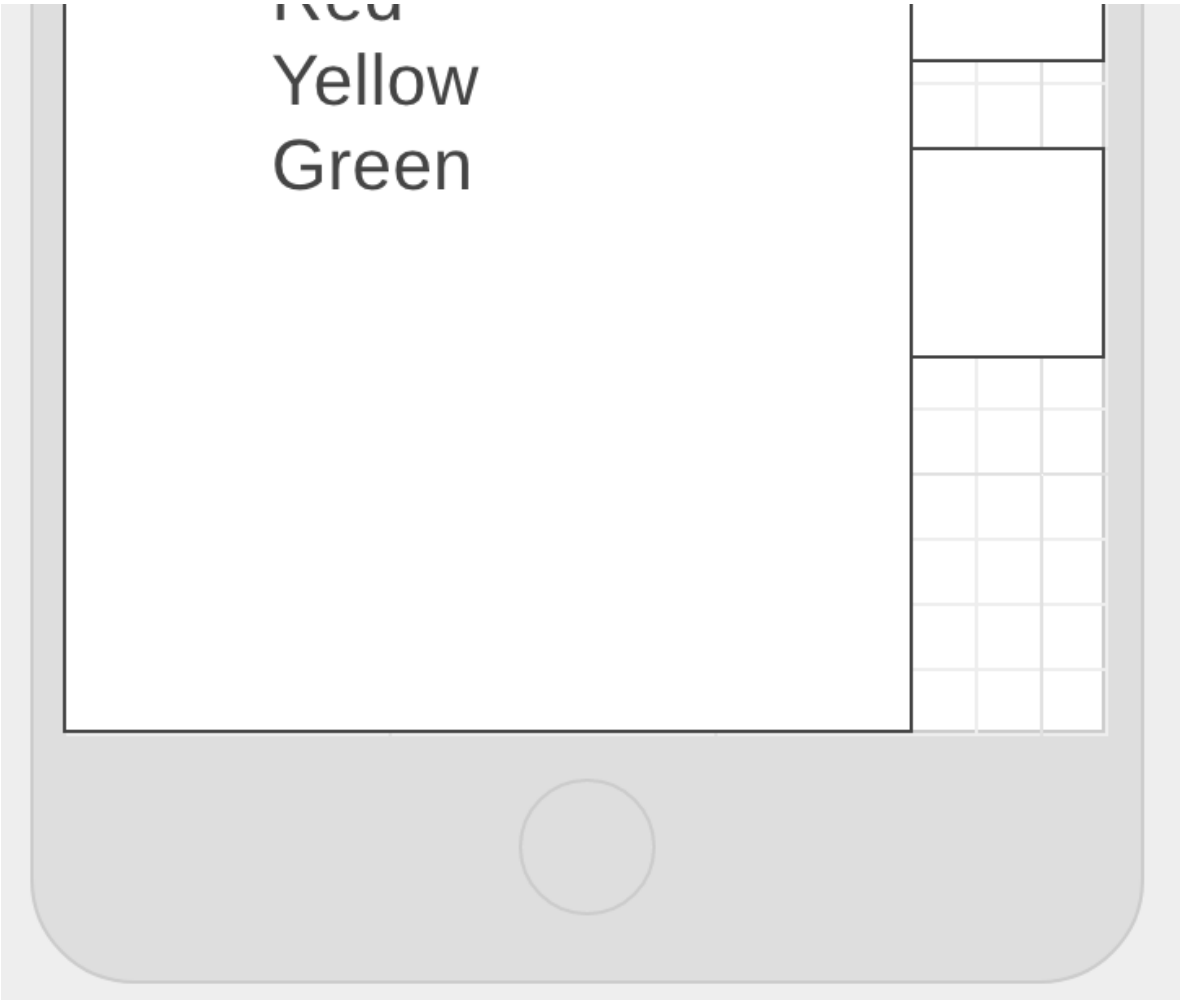
Acceptance Criteria	**Expected outcome**	**Pass/Fail**
User should be able to add start and end point of route	Each time map is clicked it drops a pin which then generates a route automatically.	Pass
User should be able to save their route to a database	Upon clicking the save button, the route information is saved and displayed	Pass
User should be able to mark a route as completed	Once completed button is clicked, the route object is saved to the completed collection and rendered in the completed list. The object is also deleted from the wish-list and so not rendered in that list	Pass

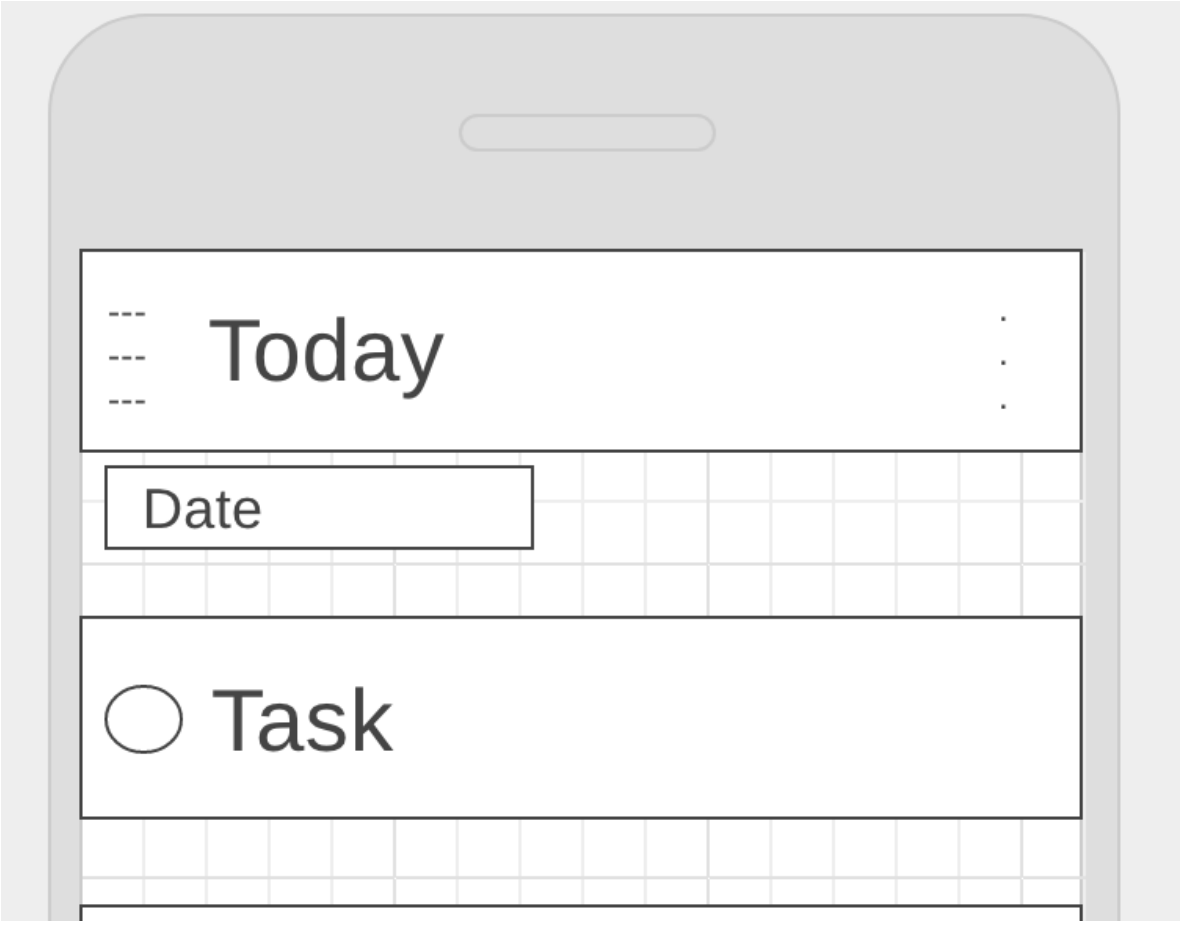
P.5 User sitemap

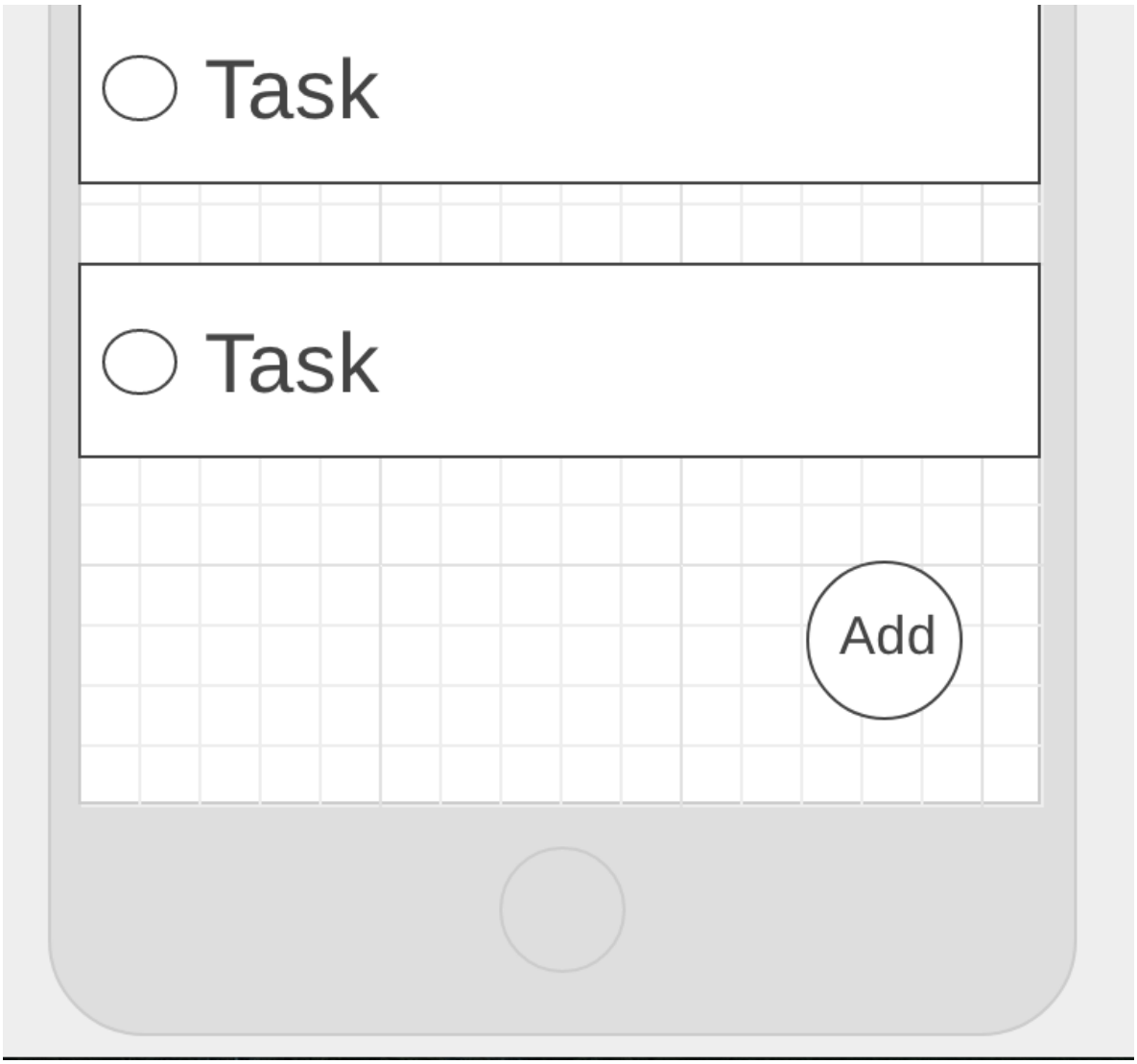


P.6 Wireframe designs



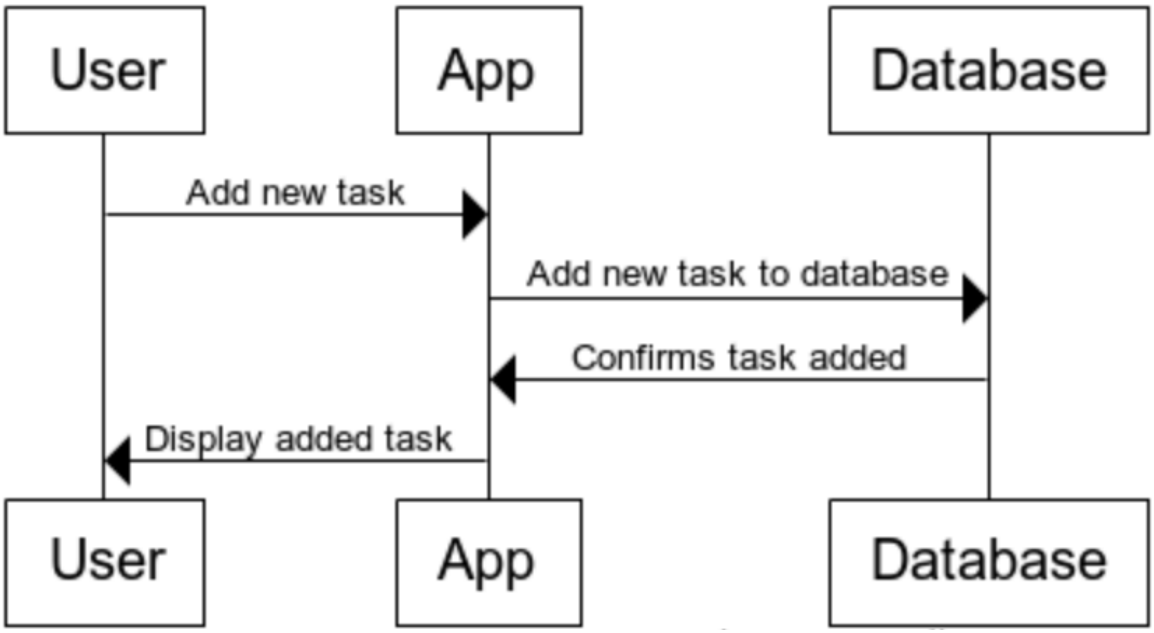

Red
Yellow
Green

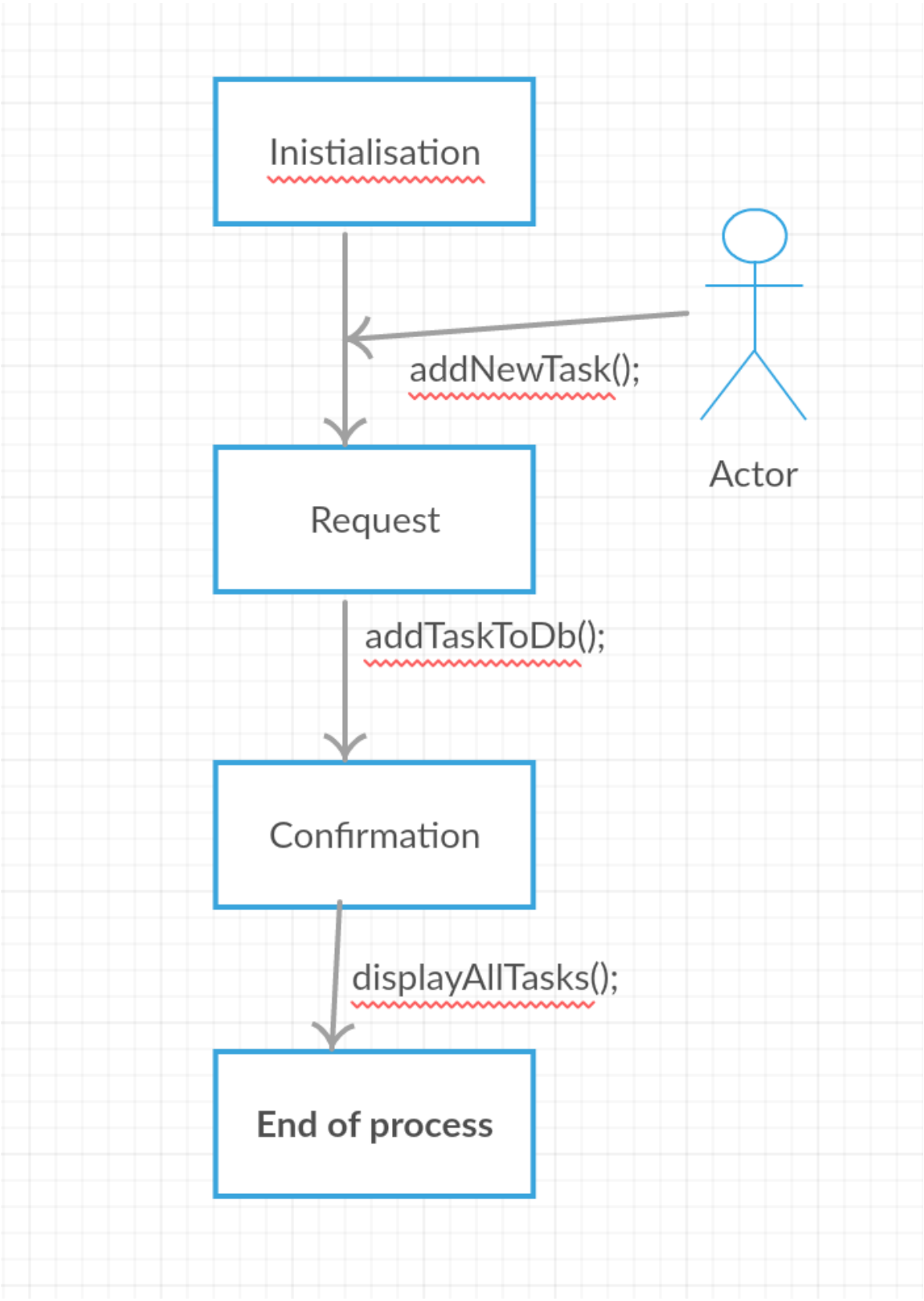

Today
Date
Task



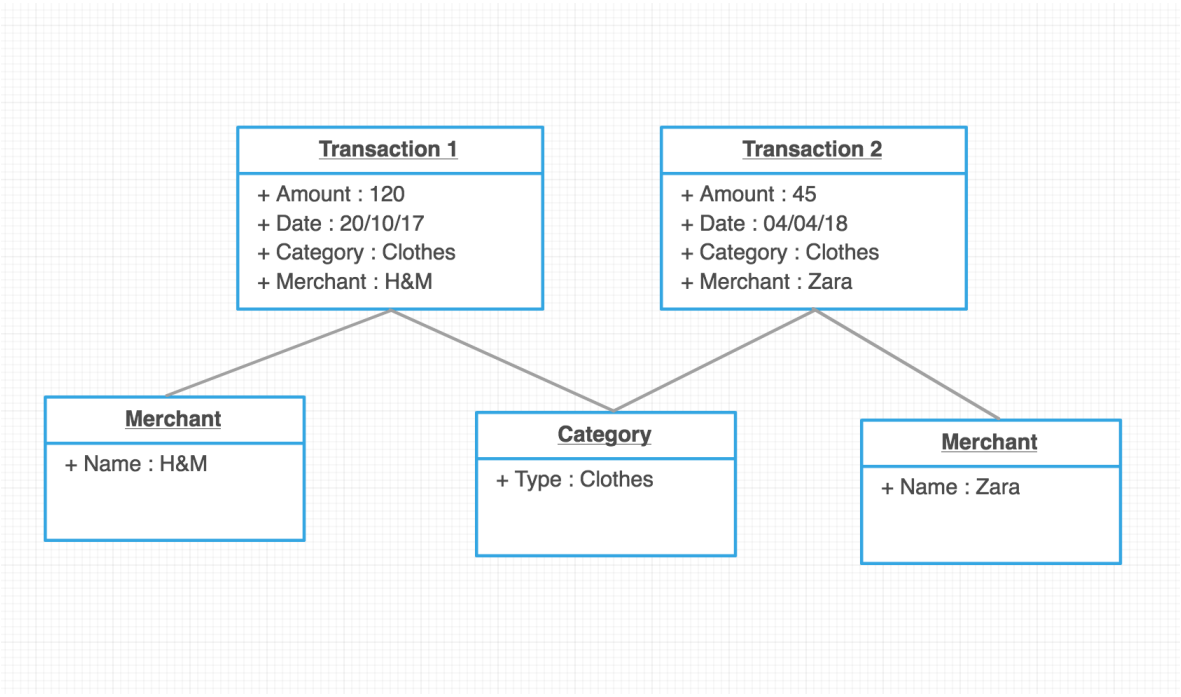
P.7 System Interaction diagrams

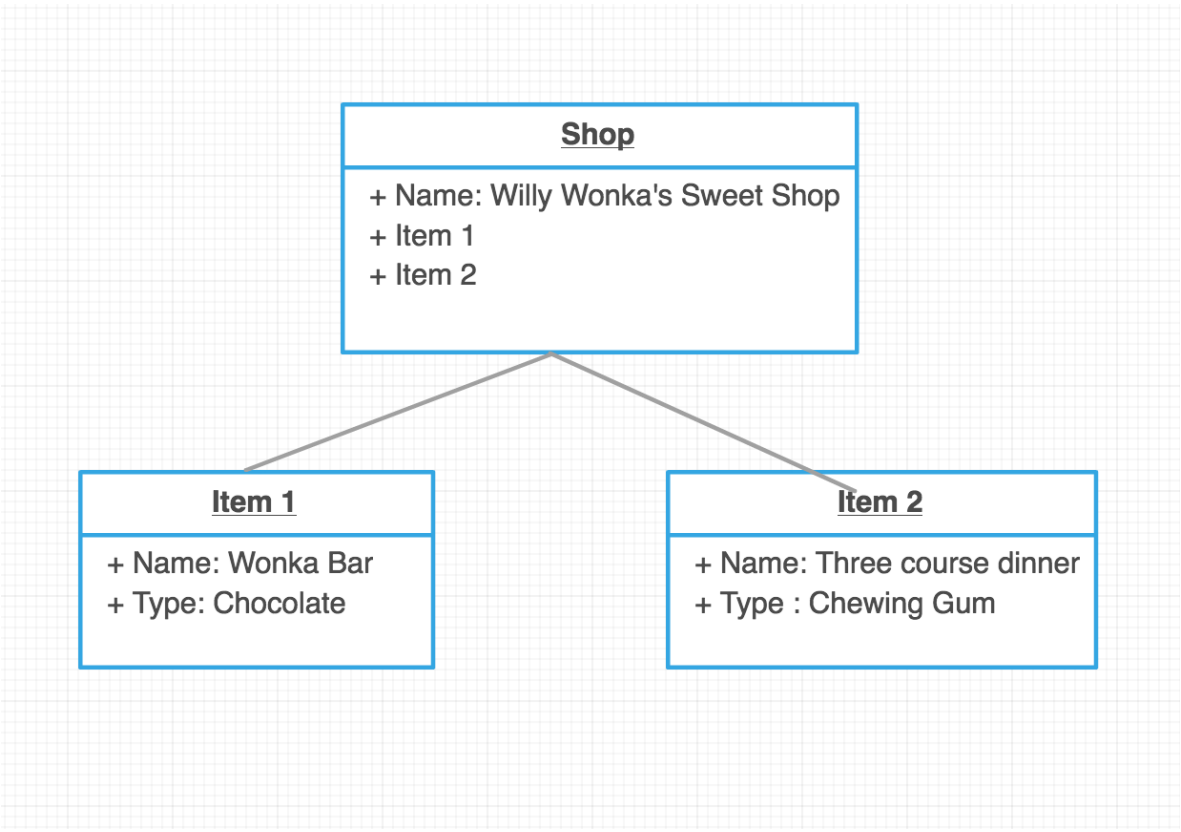
User adding task





P.8 Two Object Diagrams





P.9 Choice of Two algorithms

Algorithm 1 - This algorithm takes in a number, and an array. The number is how many unique random objects the user wants from the given array. It only functions if the number of objects requested is less than the number of objects in the array. It decrements the number, and gets a random object from the array, and then adds the object to a taken array. This prevents an object from being selected multiple times. The object is then inserted into the result array, which is returned after n objects have been inserted.

```
getRandomItems(n, array) {
  var result = new Array(n),
  len = this.array.length,
  taken = new Array(len);
  if (n > len)
    throw new RangeError("getRandom: more elements taken than available");
    while (n--) {
      var x = Math.floor(Math.random() * len);
      result[n] = this.array[x in taken ? taken[x] : x];
      taken[x] = --len in taken ? taken[len] : len;
    }
  return result;
}
```

Algorithm 2 - This loops through a word object array, and for each word it creates a new object, with a hidden visibility. This array is passed to the client-side in the browser where the visibility property plays a role.

```
makeWordObjects() {
  this.wordsForGame.forEach((word) => {
    const wordObject = {word: word, visablility: 'hidden'}
    this.wordObjectArray.push(wordObject);
  })
}
```

P.10 Example of Pseudocode

```
def total
  #Select all from transactions table
  #Make inner join categories table, where categoriesID = transactions.category_id
  #Sum the amount spent in all transactions on selected categories.
  #return the sum.
end
```

P.11 Github link to one of your projects

https://github.com/Jardoe/codenames

Jardoe / codenames

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

No description, website, or topics provided.

Edit

Add topics

3 commits | 1 branch | 0 releases | 1 contributor

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

Jardoe Update README.md — Latest commit 80328fe just now

client	first commit	a day ago
server	first commit	a day ago
README.md	Update README.md	just now

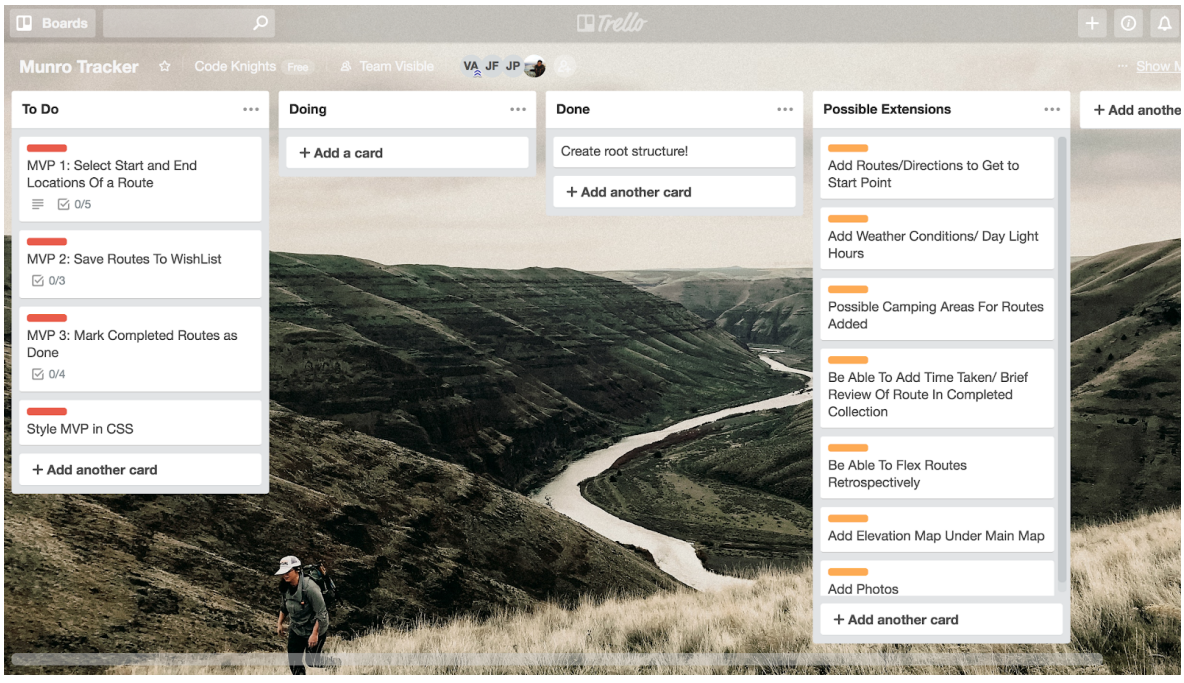
README.md

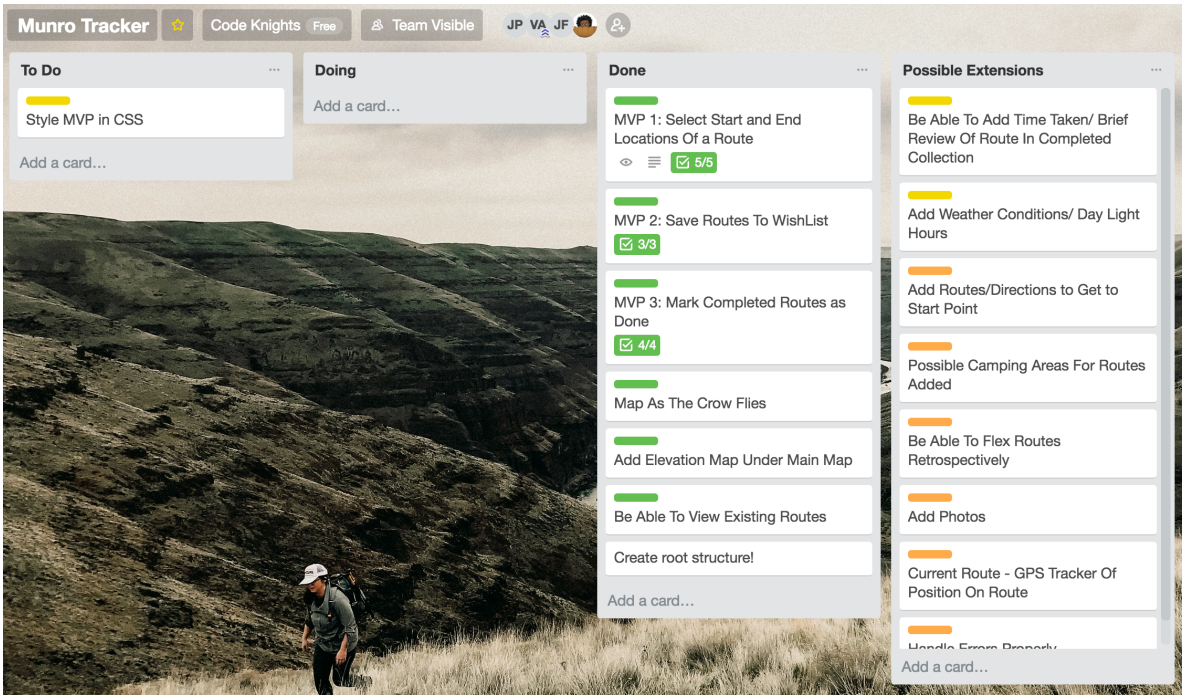
Codenames

About

This was done as the final project at CodeClan. It is a board game, and I took on the challenge of turning it into a digital product. It was done using JavaScript for the backend logic, and React for the frontend.

P.12 Different stages of development

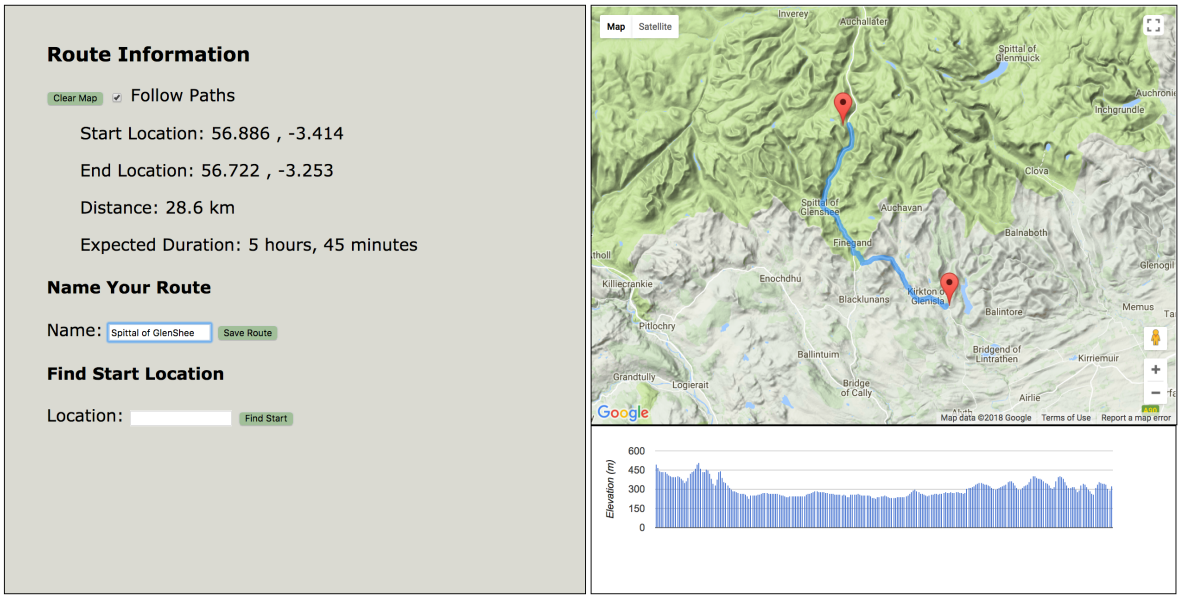




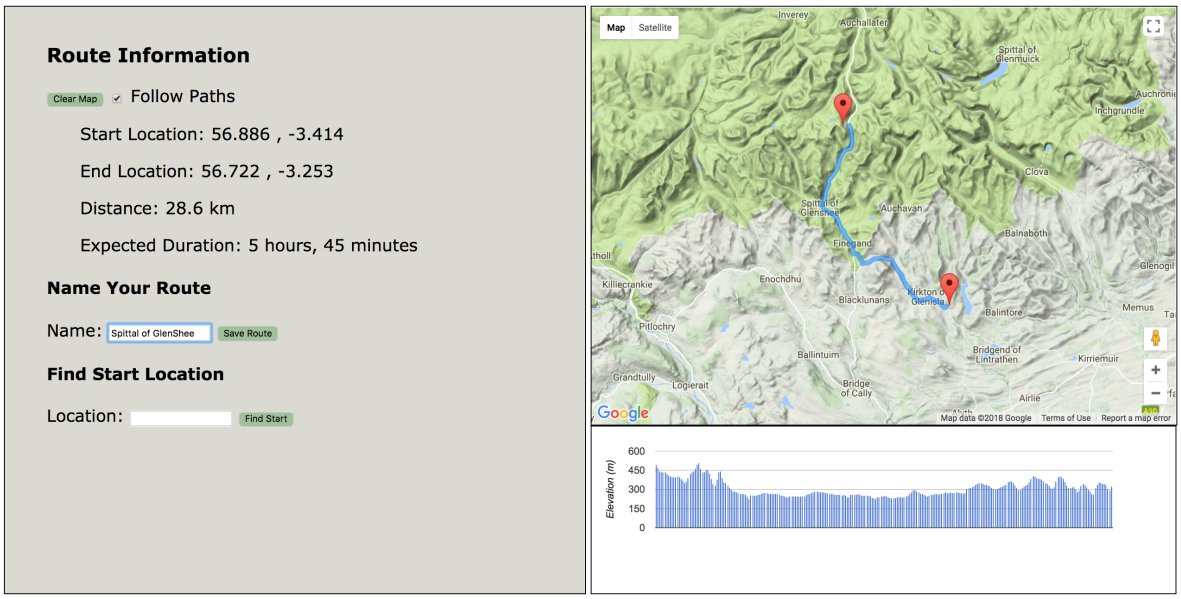
P.13 User input

User clicks on map to add a marker. The first marker placed acts as the start location. Whichever marker is placed last acts as the end location. Once more than two markers are placed, then the system responds by generating route, as well as elevation chart.

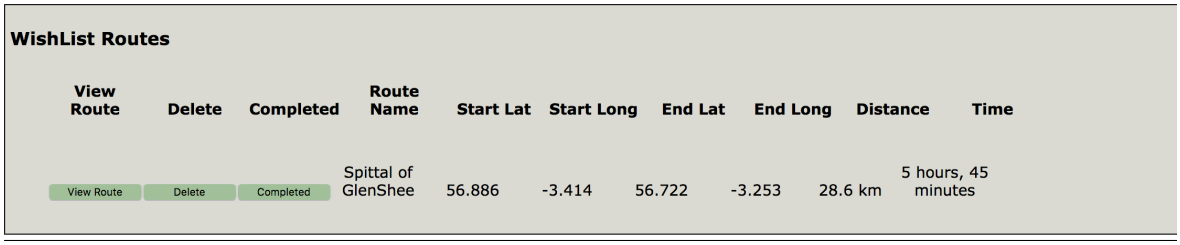
The user can then name the route and save it to a database.



P.14 Interaction with Data Persistence



The user has input the data, and then clicked the ‘save route’ button. This saves the route information to the database

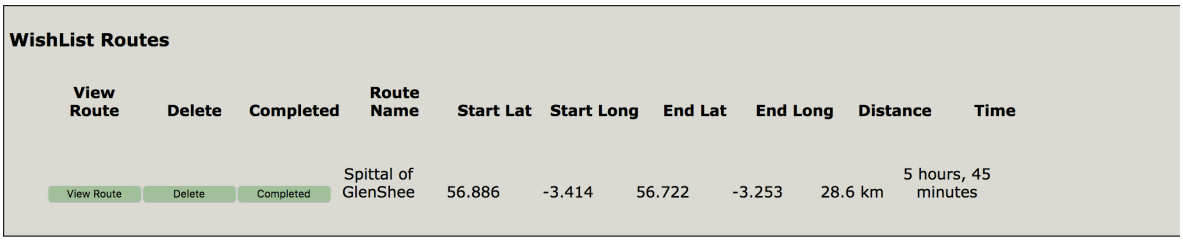


WishList Routes

View Route	Delete	Completed	Route Name	Start Lat	Start Long	End Lat	End Long	Distance	Time
View Route	Delete	Completed	Spittal of GlenShee	56.886	-3.414	56.722	-3.253	28.6 km	5 hours, 45 minutes

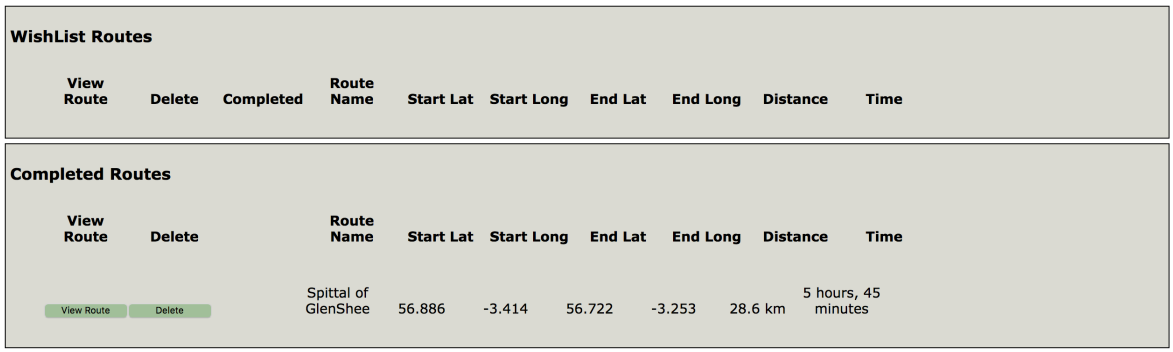
P.15 User output result

The user clicked the completed button. This moved the data from the wishlist to the completed routes section.



WishList Routes

View Route	Delete	Completed	Route Name	Start Lat	Start Long	End Lat	End Long	Distance	Time
View Route	Delete	Completed	Spittal of GlenShee	56.886	-3.414	56.722	-3.253	28.6 km	5 hours, 45 minutes



WishList Routes

View Route	Delete	Completed	Route Name	Start Lat	Start Long	End Lat	End Long	Distance	Time

Completed Routes

View Route	Delete	Route Name	Start Lat	Start Long	End Lat	End Long	Distance	Time
View Route	Delete	Spittal of GlenShee	56.886	-3.414	56.722	-3.253	28.6 km	5 hours, 45 minutes

P.16 API used in program.

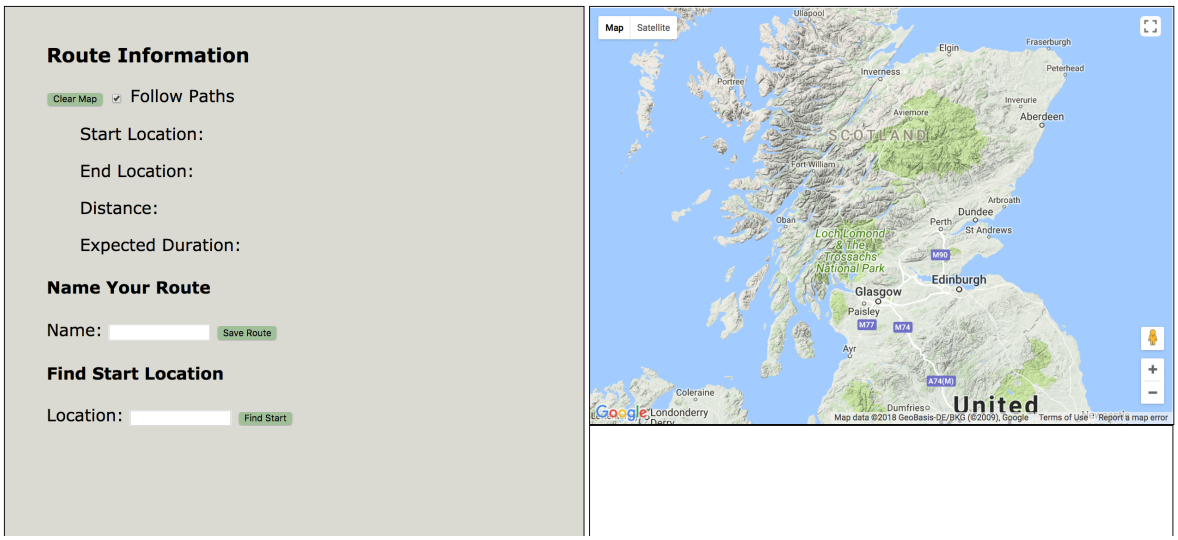
Using the google maps API with additional services.

```
const mapOptions = {
  zoom: 7,
  center: {lat: 56.4907, lng: -4.2026},
  mapTypeId: 'terrain'
};

const mapView = new MapView(mapContainer, elevationContainer, mapOptions);
mapView.render();
```

```
const GoogleMaps = require('google-maps');
const StatsView = require('./stats_view.js');
const GoogleCharts = require('google-charts').GoogleCharts;
```

```
MapView.prototype.render = function () {
  GoogleMaps.load((google) => {
    this.markers = [];
    this.waypoints = [];
    this.route = null;
    this.tempArray = [];
    this.google = google;
    this.renderElevationService();
    this.googleMap = new this.google.maps.Map(this.container, this.options);
    this.directionsService = new this.google.maps.DirectionsService();
    this.directionsRenderer = new
    this.google.maps.DirectionsRenderer({suppressMarkers: true});
    this.addMarkerOnClick();
    this.directionsRenderer.setMap(this.googleMap);
    this.geocoder = new this.google.maps.Geocoder();
    this.geodesicPoly = new google.maps.Polyline({
      strokeColor: '#CC0099',
      strokeOpacity: 1.0,
      strokeWeight: 3,
      geodesic: true,
      map: this.googleMap
    });
  });
```



P.17 Bug tracking

Map loads on Index	Failed	CSS width of map was zero, now desired width	Passed
Elevation chart renders	Failed	Imported correct module	Passed
Data can be deleted from DB	Failed	Changed from deprecated mongodb function to current function.	Passed
Map clears once data saved to DB	Failed	Re-render map on save button clicked	Passed

P.18 Testing your program

Failing -

```
const assert = require('assert');

describe('Park', function(){

  let park;
  let dino1, dino2, dino3;

  beforeEach(function(){
    park = new Park();
    dino1 = new Dinosaur("TRex", 1);
    dino2 = new Dinosaur("Steg", 5);
    dino3 = new Dinosaur("Steg", 5);
  });

  it('enclosure should start empty', function(){
    const result = park.dinosaurs;
    assert.deepStrictEqual(result, [1]);
  });


  it('should be able to return number of dinosaurs', function(){
    const result = park.numberOfDinos();
    assert.strictEqual(result, 1);
  })

  it('should be able to add dinosaur', function(){
    park.addDino(dino1);
    park.addDino(dino2);
    const result = park.numberOfDinos();
    assert.strictEqual(result, 1);
  });
```

```
Dinosaur
  ✓ should have type
  ✓ should have a number of offspring

Park
  1) enclosure should start empty
  2) should be able to return number of dinosaurs
  3) should be able to add dinosaur
  ✓ should get all dinosaurs with offspring of more than 2
  Extension
    ✓ dinos should not reproduce if left by themselves
    ✓ dinos should not reproduce by themselves, even if left after two years
    ✓ dinos can reproduce if there are more than two of the same type, for one
year
    ✓ 2 dinos (plus their babies) can reproduce in two years


7 passing (15ms)
3 failing
```

Passing -

```
const Park = require('../park.js');
const Dinosaur = require('../dinosaur.js')
const assert = require('assert');

describe('Park', function(){

  let park;
  let dino1, dino2, dino3;

  beforeEach(function(){
    park = new Park();
    dino1 = new Dinosaur("TRex", 1);
    dino2 = new Dinosaur("Steg", 5);
    dino3 = new Dinosaur("Steg", 5);
  });

  it('enclosure should start empty', function(){
    const result = park.dinosaurs;
    assert.deepStrictEqual(result, []);
  });


  it('should be able to return number of dinosaurs', function(){
    const result = park.numberOfDinos();
    assert.strictEqual(result, 0);
  })

  it('should be able to add dinosaur', function(){
    park.addDino(dino1);
    park.addDino(dino2);
    const result = park.numberOfDinos();
    assert.strictEqual(result, 2);
  });
```

```
Dinosaur
  ✓ should have type
  ✓ should have a number of offspring

Park
  ✓ enclosure should start empty
  ✓ should be able to return number of dinosaurs
  ✓ should be able to add dinosaur
  ✓ should be able to remove all dinosaurs of type
  ✓ should get all dinosaurs with offspring of more than 2
  Extension
    ✓ dinos should not reproduce if left by themselves
    ✓ dinos should not reproduce by themselves, even if left after two years
    ✓ dinos can reproduce if there are more than two of the same type, for one
year
    ✓ 2 dinos (plus their babies) can reproduce in two years


11 passing (10ms)
```