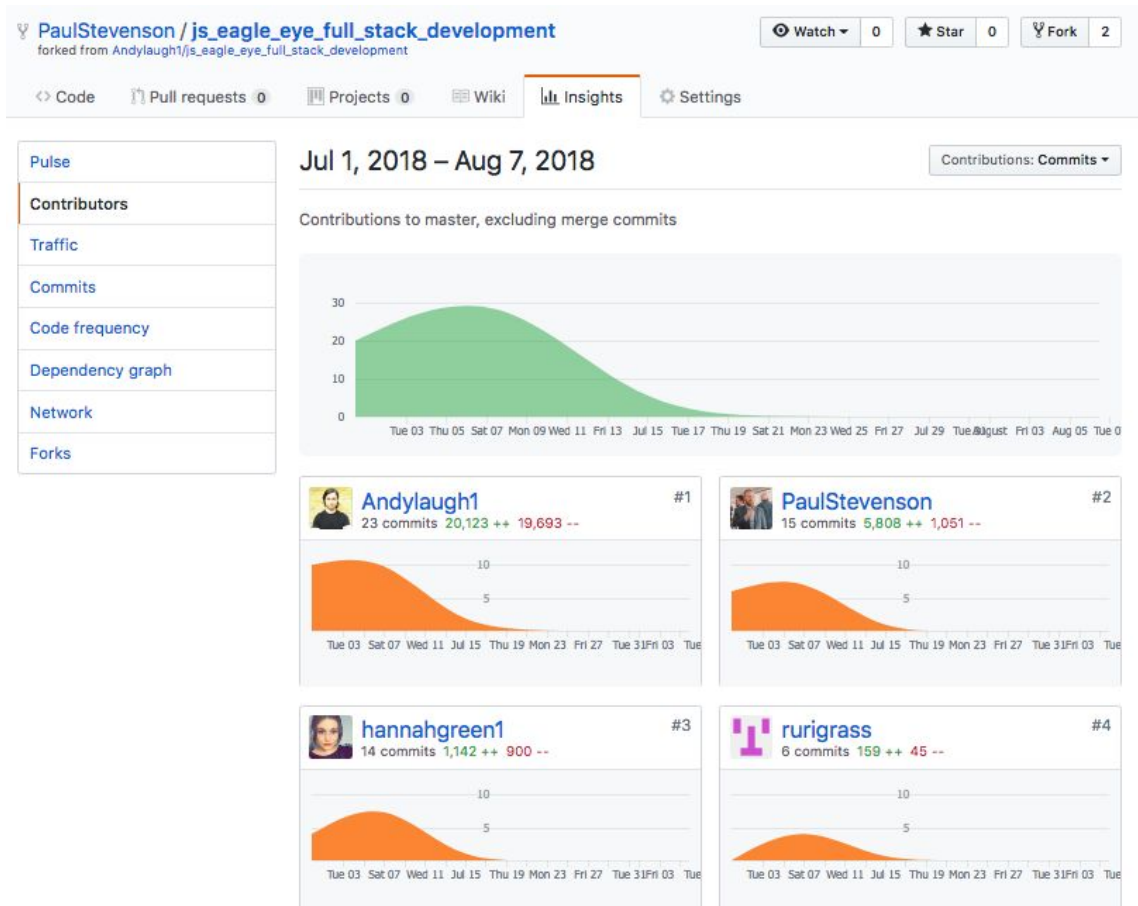


Project Unit

Paul McPhail Stevenson

Cohort E21

P 1 Github Contribution Page



P 2 Project Brief

📄 readme.txt

Educational App- Project Brief

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.

MVP- Designed by team

Build an app to allow users to view information regarding covert drone strikes since 2002.

Users should be able to click on map and find out the drone data from that area.

- * Build Full Stack JavaScript application
 - * NodeJS
 - * Webpack
 - * HTML
 - * CSS
- * Plan project using:
 - * UX
 - * Trello
- * Behaviour Driven Development
- * Make use of API's
 - * <https://api.dronestre.am/data>
- * Have some interactivity that enables a user to move through different sections of content

How to open and run this project:

From terminal run the following commands:

- Npm install
- Npm run Build
- Npm run server:dev
- mongod
- Open the project in editor
- <http://localhost:3000/>

Possible improvements in future or time permitting:

- Code refactor and clean up (movement of logic into models)
- Switch to REACT
- Add more chart features

P 3 Project Planning

Eagle Eye ☆ Eagle Eye **Free** Team Visible PS HG RG 2 ... **Show Menu**

Should ...

Filter by...

☑ 2/3

+ Add another card

Doing ...

+ Add a card

Could ...

Sidebar

Have map zoom-n when marker is selected

+ Add another card

Done ...

Create Repo

Write out MVP

Git guide

merge countries and years filters

Incorp Ru's code

CSS

HG RG

Set up file structure

Add option to slider to reset to all markers

Drone info

reset all markers - all countries

UX

+ Add another card

InProgress ...

Model and Drone View

HG RG

Diagrams

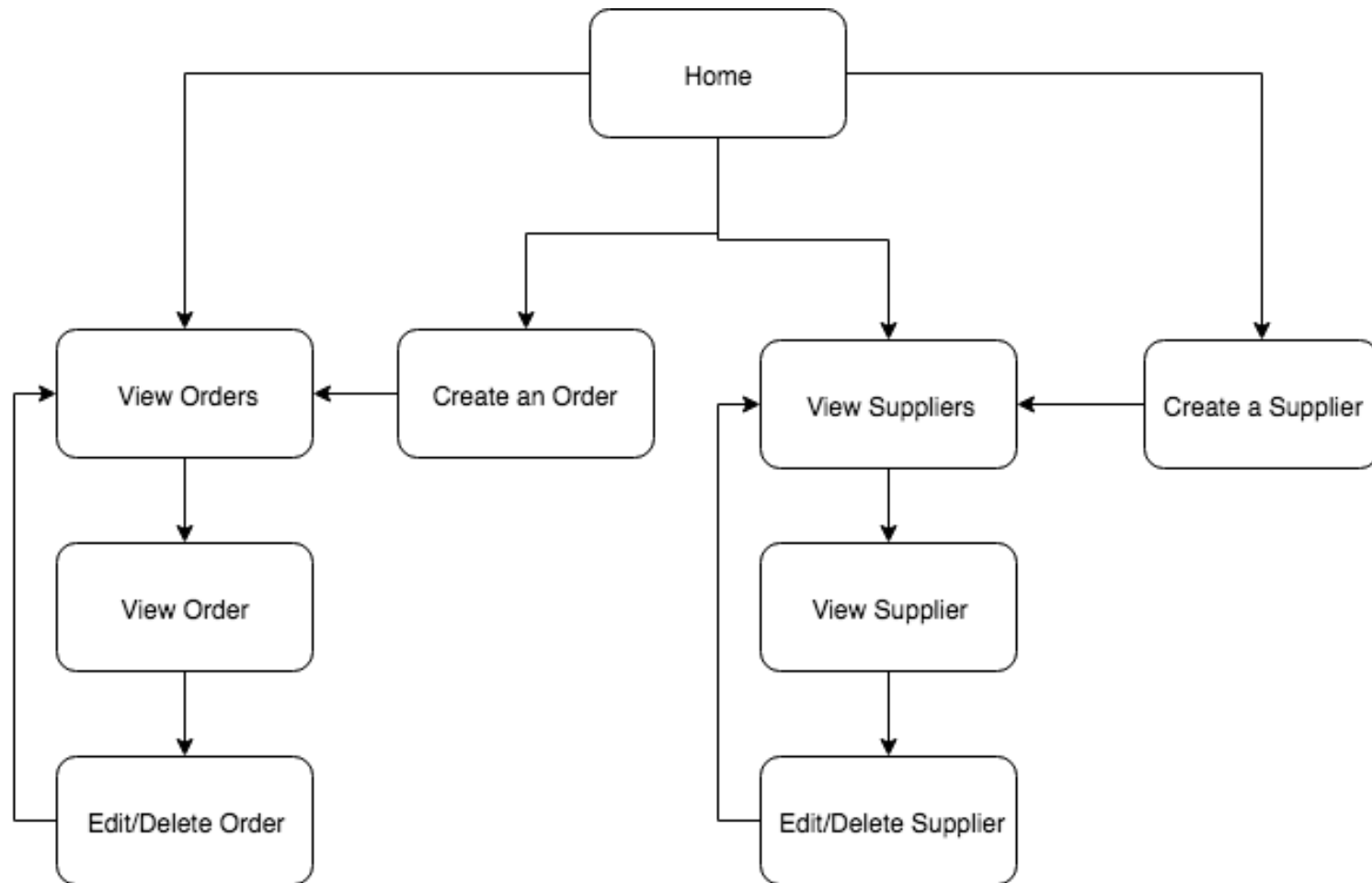
☑ 3/5

+ Add another card

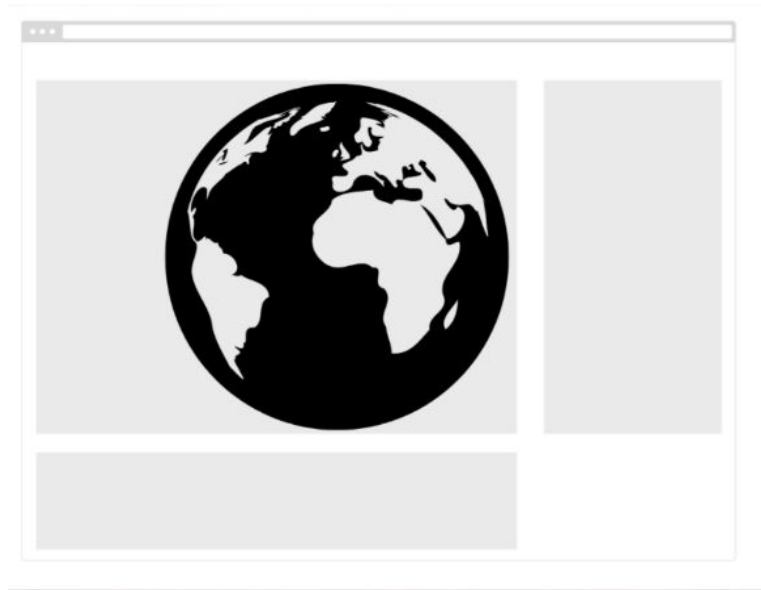
P 4 Acceptance Criteria and Test Plan

Acceptance Criteria	Expected Result	Pass
A user should be able to open and view the app	Webpage successfully loads with the map rendering the all the markers. Content should populate in the appropriate containers.	Pass
A user should be able to select a marker on the map, changing to the content displayed.	Clicking the marker populates the the 'Strike Info' container, and 'Conflict History' container specific to the data attached to the marker .	Pass
A user should be able to choose a specific country from the dropdown menu.	The map should zoom into the chosen country.	Pass
A user should be able to use the slider showing the chronological order of drone strikes.	The markers should render on the map in chronological order using the Date Key provided by the API using the slider.	Pass
A user should be able to reset the map to the default view	Clicking the reset button should recentre the map, and re-populate the markers.	Pass

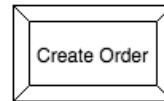
P 5 User Site Map



P 6 Two Wireframe Designs

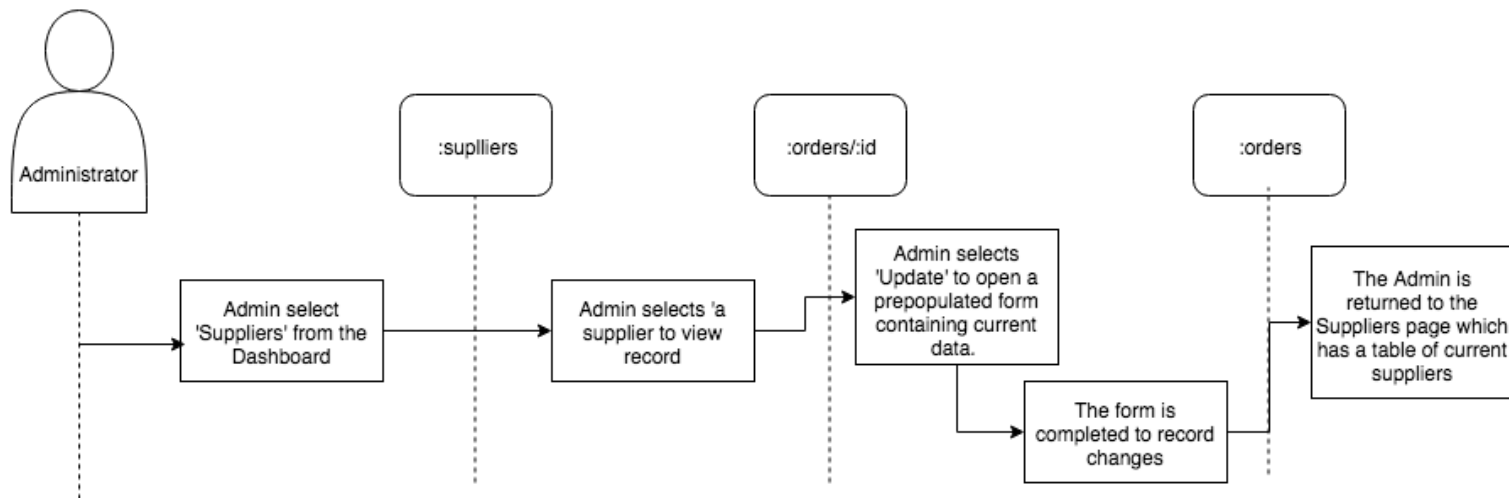
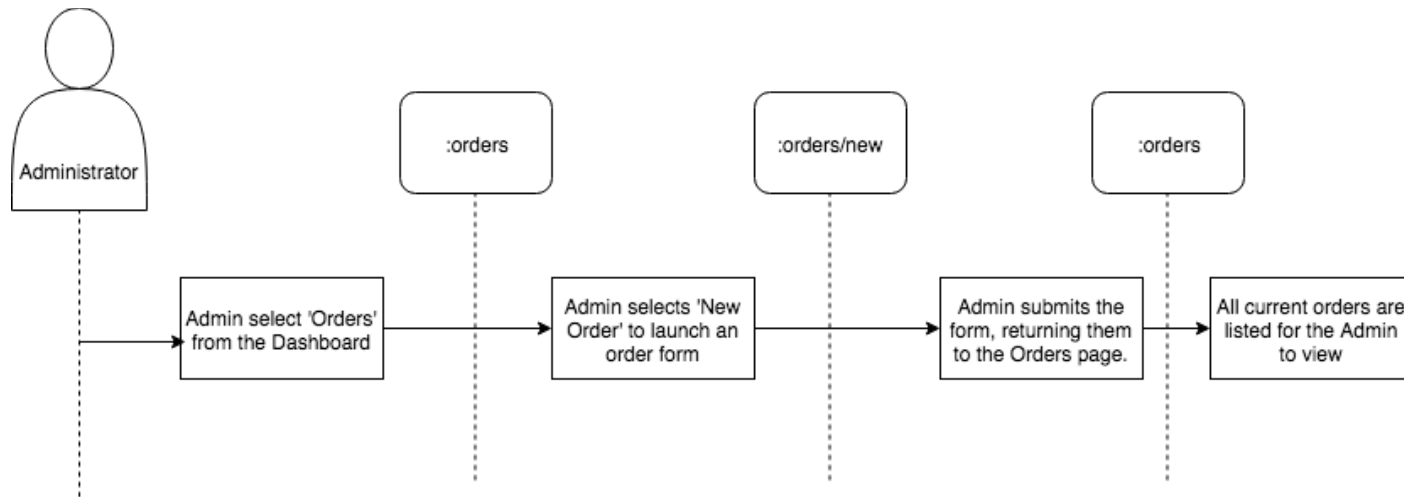


Home	Orders	Items	Suppliers	Customers	
------	--------	-------	-----------	-----------	--

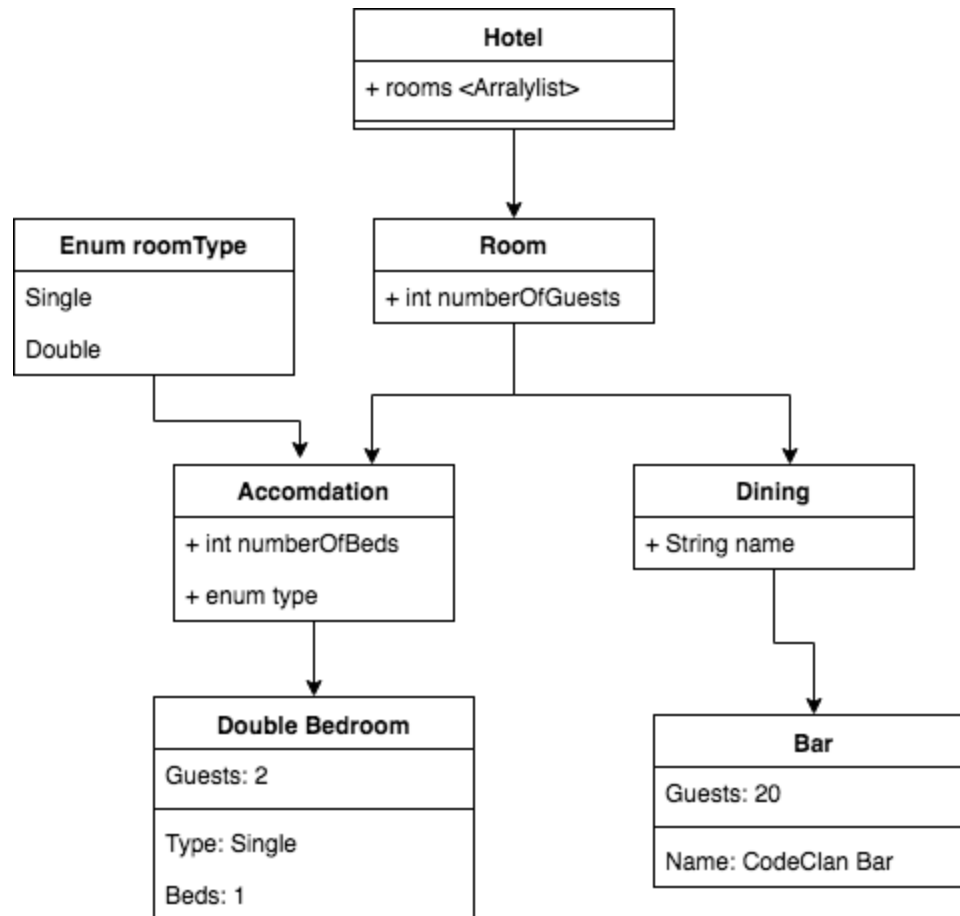


Orders
Order 2
Order 1
Order 3

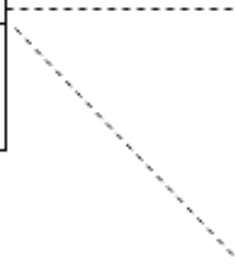
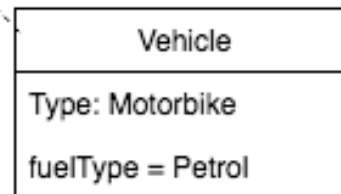
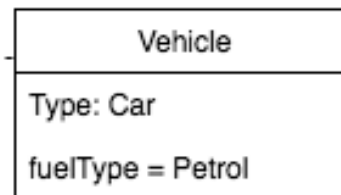
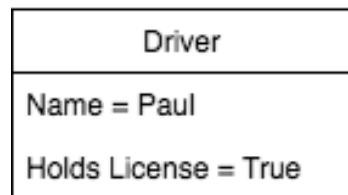
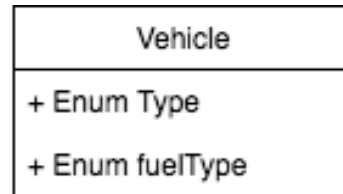
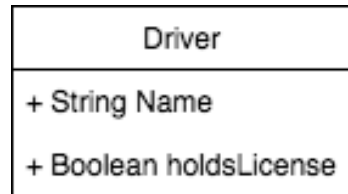
P 7 Two System Interaction Diagrams



P 8.1 Two Object Diagrams



P 8.2 Two Object Diagrams



P 9.1 Select Two Algorithms

To the right, the array `films` contains film objects which contain a title, genre, release date and runtime.

The below algorithm loops through the array to pull the film titles. Instead of a `for` loop, enumeration was used with the `map()` method.

```
const Cinema = function (films) {  
  this.films = films;  
};  
  
Cinema.prototype.filmTitles = function () {  
  const titlesArray = this.films.map((film) => {  
    return film.title;  
  });  
  return titlesArray;  
};
```

```
beforeEach(function() {  
  moonlight = new Film('Moonlight', 'drama',  
    2016, 111);  
  bladeRunner = new Film('Blade Runner 2049',  
    'sci-fi', 2017, 164);  
  dunkirk = new Film('Dunkirk', 'history', 2017,  
    96);  
  blackPanther = new Film('Black Panther',  
    'action', 2018, 134);  
  trainspotting = new Film('T2 Trainspotting',  
    'drama', 2017, 117);  
  
  films = [moonlight, bladeRunner, dunkirk,  
    blackPanther, trainspotting];  
  cinema = new Cinema(films);  
});
```

P 9.2 Select Two Algorithms

The second algorithm uses the filter() method sort through the array of film objects based on a specific property. For example, the array could be looped through sorting the data by genre.

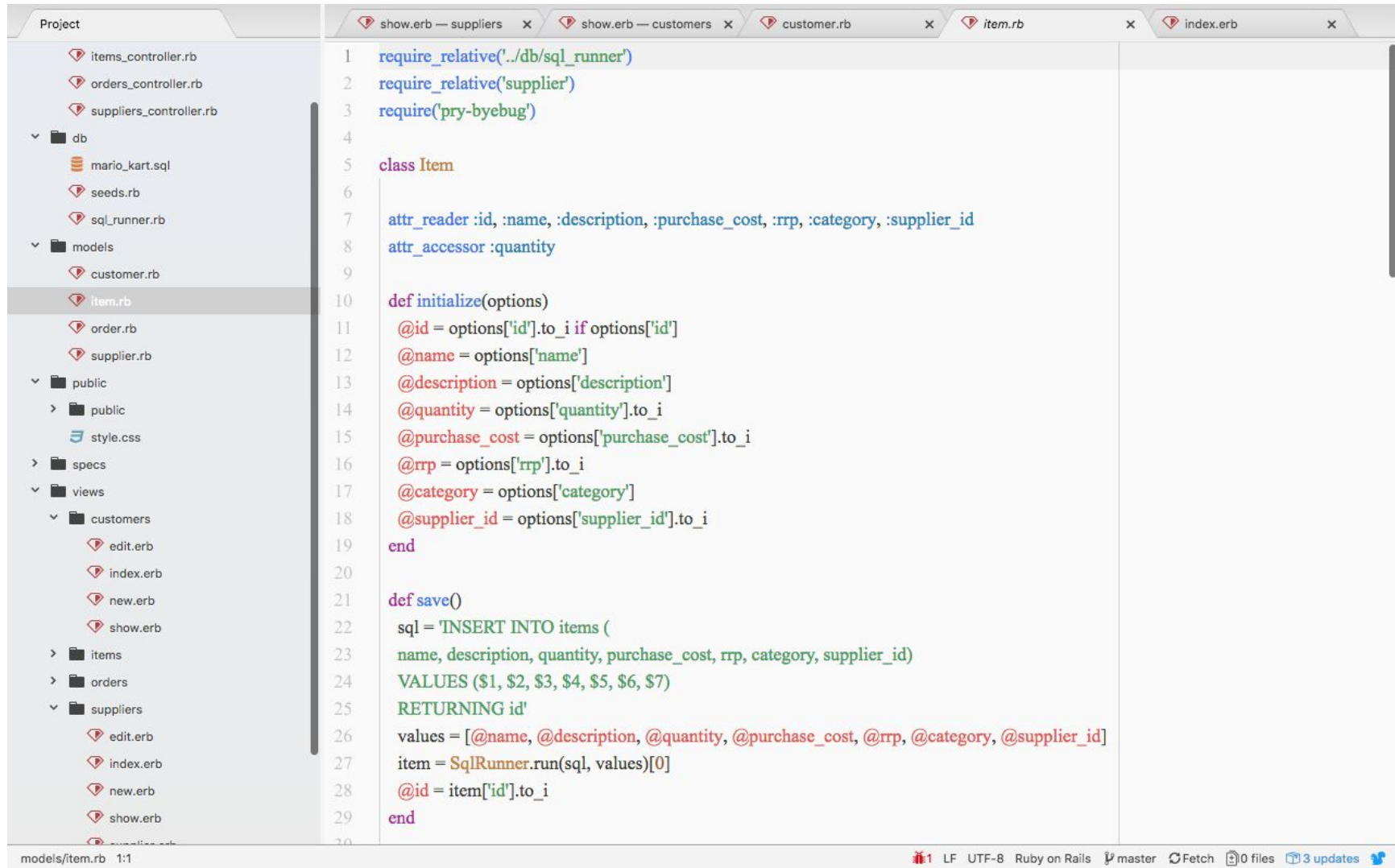
```
Cinema.prototype.filmsByProperty = function
  (property, value) {
    const objectArray = this.films.filter((film) => {
      return film[property] === value;
    });
    return objectArray;
  };
};
```

P 10 Pseudocode for a function

```
@Override
public int calculateTotalPurchaseCost() {
    return this.purchaseCost + this.additionalCost;
}
/* calculateTotalPurchaseCost() should take in the object properties (purchaseCost and additionalCost)
   It will add these two properties together
   The result will equal the total amount spent of purchasing the object. |
*/
```

P 11 Solo Project with Github Link

https://github.com/PaulStevenson/project_mariokartshop_ruby_sql_html_css



The screenshot shows a web application editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like db, models, public, specs, and views. The code editor displays the content of `models/item.rb`, which defines the `Item` class and its `initialize` and `save` methods.

```
1 require_relative('../db/sql_runner')
2 require_relative('supplier')
3 require('pry-bybug')
4
5 class Item
6
7   attr_reader :id, :name, :description, :purchase_cost, :rrp, :category, :supplier_id
8   attr_accessor :quantity
9
10  def initialize(options)
11    @id = options['id'].to_i if options['id']
12    @name = options['name']
13    @description = options['description']
14    @quantity = options['quantity'].to_i
15    @purchase_cost = options['purchase_cost'].to_i
16    @rrp = options['rrp'].to_i
17    @category = options['category']
18    @supplier_id = options['supplier_id'].to_i
19  end
20
21  def save()
22    sql = 'INSERT INTO items (
23      name, description, quantity, purchase_cost, rrp, category, supplier_id)
24    VALUES ($1, $2, $3, $4, $5, $6, $7)
25    RETURNING id'
26    values = [@name, @description, @quantity, @purchase_cost, @rrp, @category, @supplier_id]
27    item = SqlRunner.run(sql, values)[0]
28    @id = item['id'].to_i
29  end
30
```

The status bar at the bottom indicates the file is `models/item.rb` at line 1:1, with encoding `UTF-8`, Ruby on Rails, and 3 updates.

P 12 Project Planning with changes

Goal: Consolidate Learning
Practise CSS

Planning

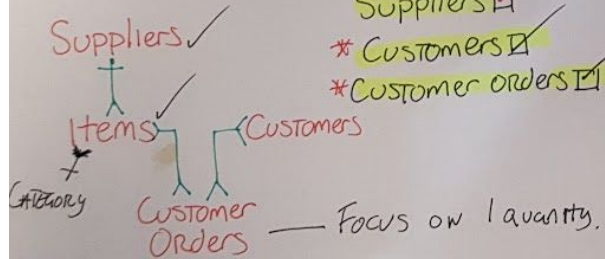
PROTO PERSONA ☒
USER NEEDS ☒
USER JOURNEY ☒
WIREFRAME ☒

CLASS DIAGRAM ☒
ACTIVITY DIAGRAM ☒

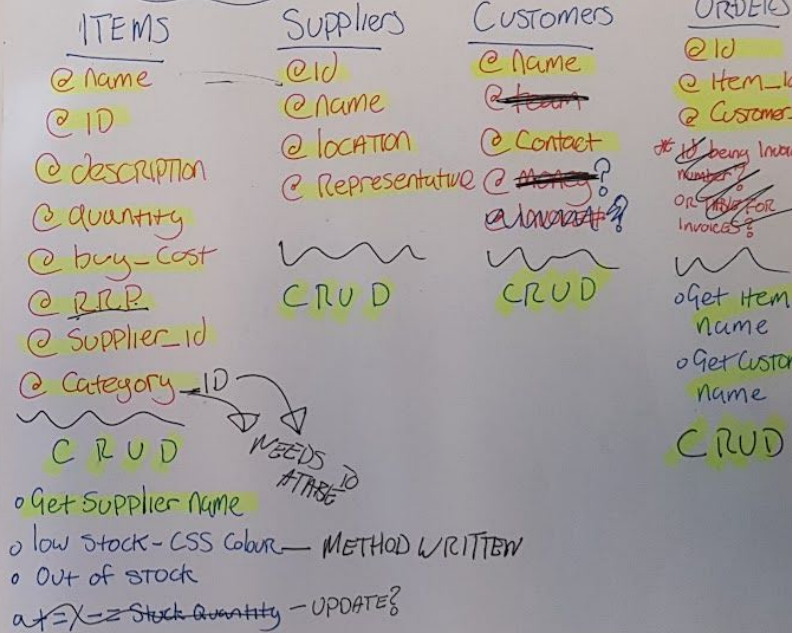
Setup:

DATABASE: maria_kart ☒
SQL: maria_kart ☒
Classes: Items ☒
Suppliers ☒
* Customers ☒

Tables: Items ☒
Suppliers ☒
* Customers ☒
* Customer Orders ☒



CLASS DIAGRAM



EXT +

- o Filter: Supplier & Category
- o Search function
- o Sort tables A-Z etc
- o Stock Order Function
- o ERROR MESSAGE Supplier Delete

Order Form

Select from Customers & Items

P 13 Processing User Input

DRONE STRIKE!!!

P 14 Data Persistence

DashboardOrdersItemsSuppliersRacers

Add Item

NameCodeClanDescriptionExample of data inputQuantity1Purchase Cost100RRP100Category

Defence ▾SupplierBanana Boy ▾Submit New Item

Item Profile

Item: CodeClan

Description: Example of data input

Quantity: 1

Purchase Cost: 100

RRP: 100

Category: Defence

Supplier: [Banana Boy](#)

[Update Item](#)

Delete Item

P 15 Show the correct output of results and feedback to user.

The delete button removes a racers profile from the table 'Existing Racers'

Racer Profile

[New Order](#)

Name: Paul

Contact: [0776](#)

[Update Racer](#)

Delete Racer

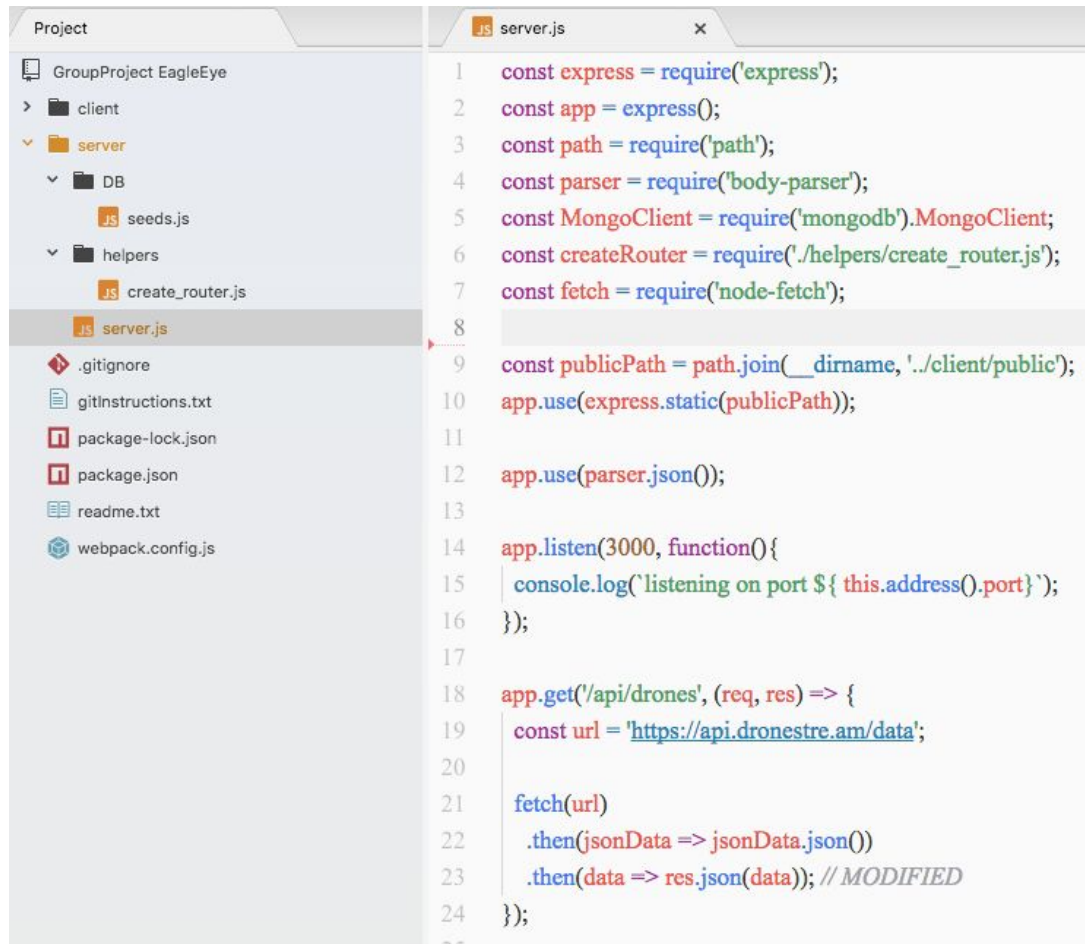
Exisiting Racers

[New Racer](#)

Name	Show Racer
Mario	Show Racer
Bowser	Show Racer

P 16 Use of an API

- The code that uses or implements the API



The image shows a code editor with a project structure on the left and a code file on the right. The project structure is for 'GroupProject EagleEye' and includes a 'server' folder with 'DB', 'seeds.js', 'helpers', and 'create_router.js'. The 'server.js' file is selected and its code is displayed on the right. The code is a Node.js server using Express, body-parser, and node-fetch. It sets up a static route for public files and a GET route for '/api/drones' that fetches data from 'https://api.dronestore.am/data' and returns it as JSON.

```
Project
└─ GroupProject EagleEye
   └─ client
      └─ server
         └─ DB
            └─ seeds.js
            └─ helpers
               └─ create_router.js
               └─ server.js
         └─ .gitignore
         └─ gitInstructions.txt
         └─ package-lock.json
         └─ package.json
         └─ readme.txt
         └─ webpack.config.js

server.js
1  const express = require('express');
2  const app = express();
3  const path = require('path');
4  const parser = require('body-parser');
5  const MongoClient = require('mongodb').MongoClient;
6  const createRouter = require('./helpers/create_router.js');
7  const fetch = require('node-fetch');
8
9  const publicPath = path.join(__dirname, '../client/public');
10 app.use(express.static(publicPath));
11
12 app.use(parser.json());
13
14 app.listen(3000, function() {
15   console.log('listening on port ${ this.address().port}');
16 });
17
18 app.get('/api/drones', (req, res) => {
19   const url = 'https://api.dronestore.am/data';
20
21   fetch(url)
22     .then(jsonData => jsonData.json())
23     .then(data => res.json(data)); // MODIFIED
24 });
```

- The API being used by the program whilst running
DRONE APP OPEN

P 17 Bug Tracking Report

<u>Test</u>	<u>Fail</u>	<u>Solution</u>	<u>Pass/Fail</u>
Should be able to render markers on the map			Pass
Clicking a marker should populate the 'Drone Strike' info view			Pass
The slider should show the the markers rendering in chronological order	Failed	Formatted the Date key from the API so that it could be used more effectively	Pass
Selecting a country from the dropdown menu should zoom the map into that country	Failed	Seperated the map and marker rendering methods	Pass
The reset button should reset the map to default	Failed	Set the event to re-render the map zoom to centre	Pass

P 18 Demonstrate Testing In A Programme

The screenshot displays an IDE with two versions of the `ItemTest.java` file side-by-side. The left pane shows the original code, and the right pane shows a modified version with a new test method.

Original Code (Left Pane):

```
1 import org.junit.Before;
2 import org.junit.Test;
3
4 import static org.junit.Assert.assertEquals;
5
6 public class ItemTest {
7     Item item;
8
9     @Before
10    public void before(){
11        item = new Item( name: "item1", price: 10, quantity: 1, bogof: false);
12    }
13
14    @Test
15    public void canGetName(){
16        assertEquals( expected: "item1", item.getName());
17    }
18
19    @Test
20    public void canGetPrice(){
21        assertEquals( expected: 10, item.getPrice(), delta: 0.01);
22    }
23
24    @Test
25    public void canGetQuantity(){
26        assertEquals( expected: 1, item.getQuantity());
27    }
28
29    @Test
30    public void canSetQuantity(){
31        assertEquals( expected: 2, item.setQuantity(2));
32    }
33 }
```

Modified Code (Right Pane):

```
2 import org.junit.Test;
3
4 import static org.junit.Assert.assertEquals;
5
6 public class ItemTest {
7     Item item;
8
9     @Before
10    public void before(){
11        item = new Item( name: "item1", price: 10, quantity: 1, bogof: false);
12    }
13
14    @Test
15    public void canGetName(){
16        assertEquals( expected: "item1", item.getName());
17    }
18
19    @Test
20    public void canGetPrice(){
21        assertEquals( expected: 10, item.getPrice(), delta: 0.01);
22    }
23
24    @Test
25    public void canGetQuantity(){
26        assertEquals( expected: 1, item.getQuantity());
27    }
28
29    @Test
30    public void canSetQuantity(){
31        assertEquals( expected: 2, item.setQuantity(2));
32    }
33 }
```

Run Panel (Bottom):

Run ItemTest.canGetPrice

1 test passed - 6ms

ItemTest 6ms

canGetPrice 6ms

Process finished with exit code 0

CheckoutTest.applyPromotionalDiscountTrue: 0 classes reloaded // Stop debug session (moments ago)

23:24 LF UTF-8 Git: master

shoppingBasketTest > src > test > java > ItemTest

ItemTest.java x BasketTest.java x CheckoutTest.java x DiscountTest.java x ItemTest.java x

```
1 import org.junit.Before;
2 import org.junit.Test;
3
4 import static org.junit.Assert.assertEquals;
5
6 public class ItemTest {
7
8     Item item;
9
10    @Before
11    public void before(){
12        item = new Item( name: "item1", price: 10, quantity: 1, bogof: false);
13    }
14
15    @Test
16    public void canGetName(){
17        assertEquals( expected: "item1", item.getName());
18    }
19
20    @Test
21    public void canGetPrice(){
22        assertEquals( expected: 11, item.getPrice(), delta: 0.01);
23    }
24
25    @Test
26    public void canGetQuantity(){
27        assertEquals( expected: 1, item.getQuantity());
28    }
29
30    @Test
31    public void canSetQuantity(){
32        assertEquals( expected: 2, item.setQuantity(2));
33    }
34}
```

ItemTest

```
1 import org.junit.Test;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class ItemTest {
6
7     Item item;
8
9
10    @Before
11    public void before(){
12        item = new Item( name: "item1", price: 10, quantity: 1, bogof: false);
13    }
14
15    @Test
16    public void canGetName(){
17        assertEquals( expected: "item1", item.getName());
18    }
19
20    @Test
21    public void canGetPrice(){
22        assertEquals( expected: 11, item.getPrice(), delta: 0.01);
23    }
24
25    @Test
26    public void canGetQuantity(){
27        assertEquals( expected: 1, item.getQuantity());
28    }
29
30    @Test
31    public void canSetQuantity(){
32        assertEquals( expected: 2, item.setQuantity(2));
33    }
34}
```

ItemTest > canGetPrice()

Run ItemTest.canGetPrice

1 test failed - 21ms

ItemTest 21ms

canGetPrice 21ms

```
java.lang.AssertionError:
Expected :11.0
Actual   :10.0
<Click to see difference>

<1 internal call>
at org.junit.Assert.failNotEquals(Assert.java:834) <2 internal calls>
at ItemTest.canGetPrice(ItemTest.java:23) <23 internal calls>
```

CheckoutTest.applyPromotionalDiscountTrue: 0 classes reloaded // Stop debug session (moments ago)

5:15 LF UTF-8 Git: master