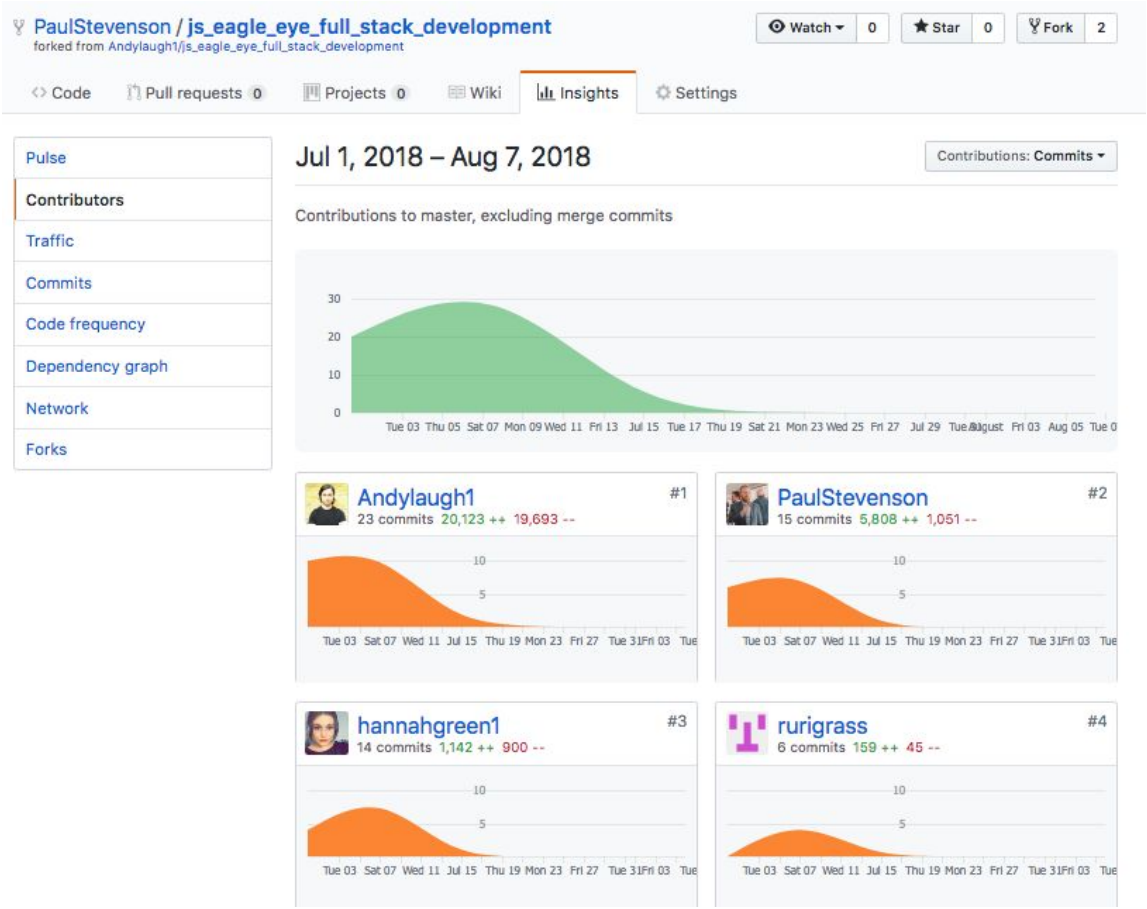
Project Unit

Paul McPhail Stevenson
Cohort E21

P 1 Github Contribution Page

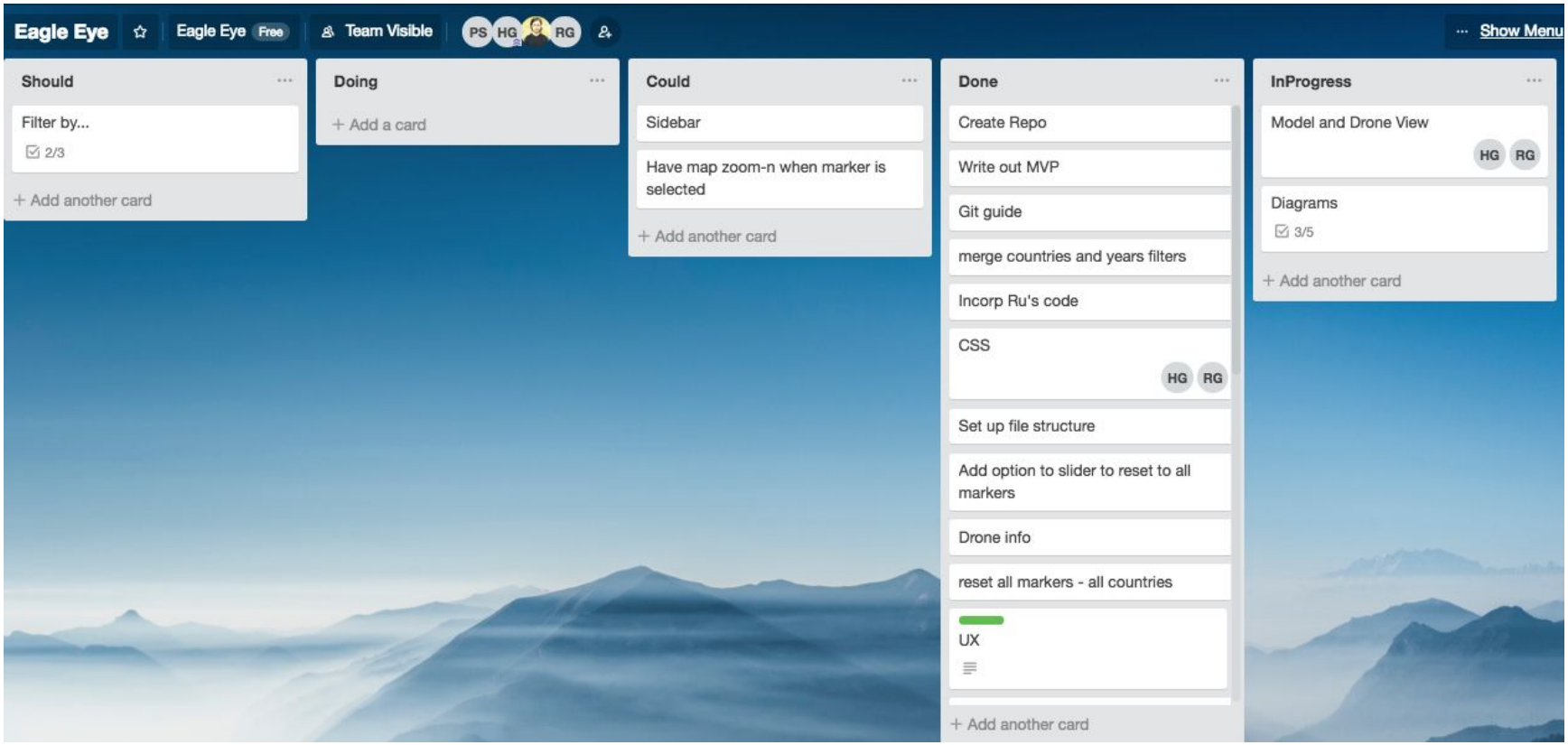


P 2 Project Brief

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
```

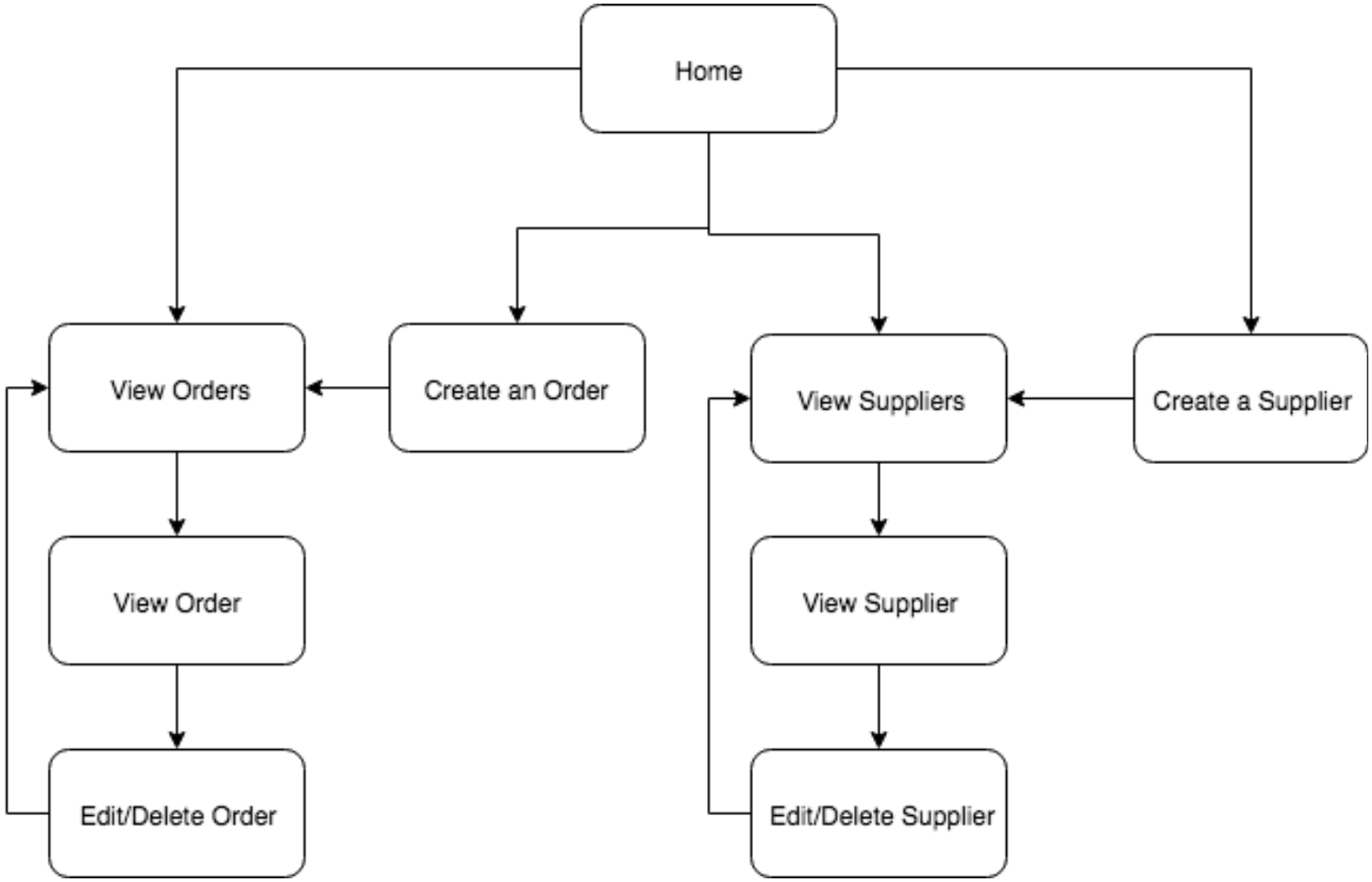
P 3 Project Planning



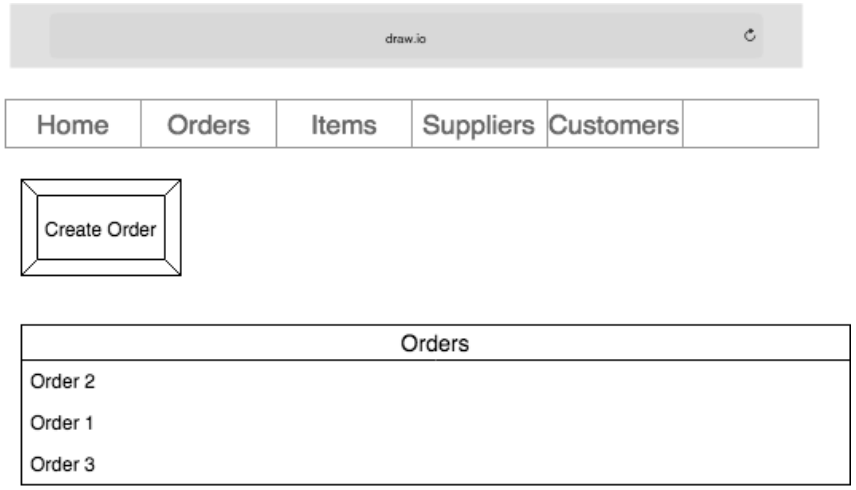
P 4 Acceptance Criteria and Test Plan

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

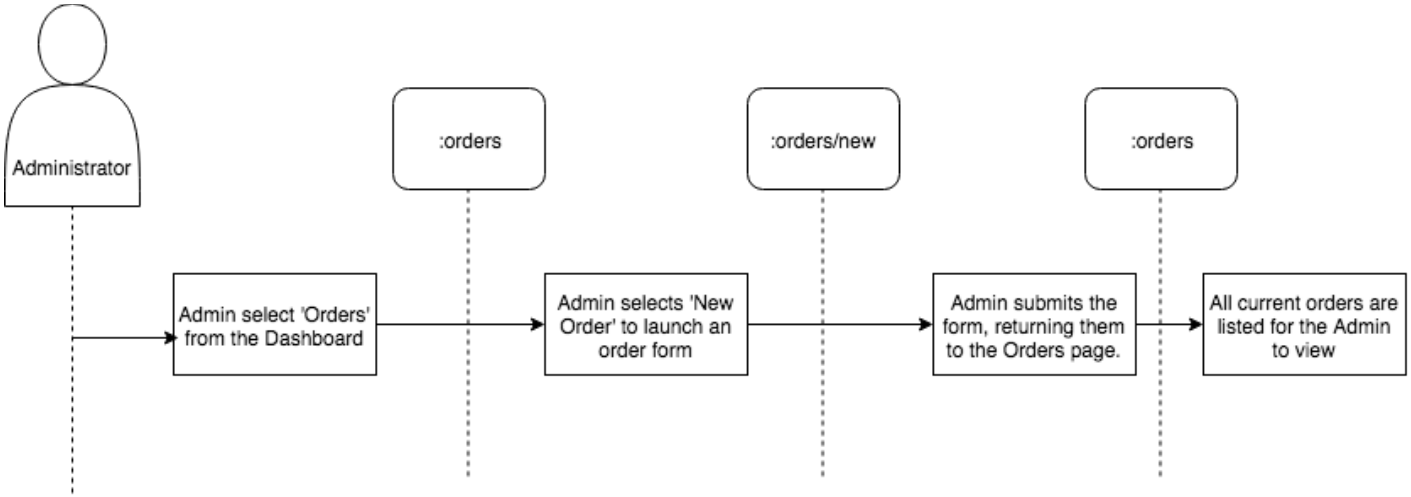
P 5 User Site Map

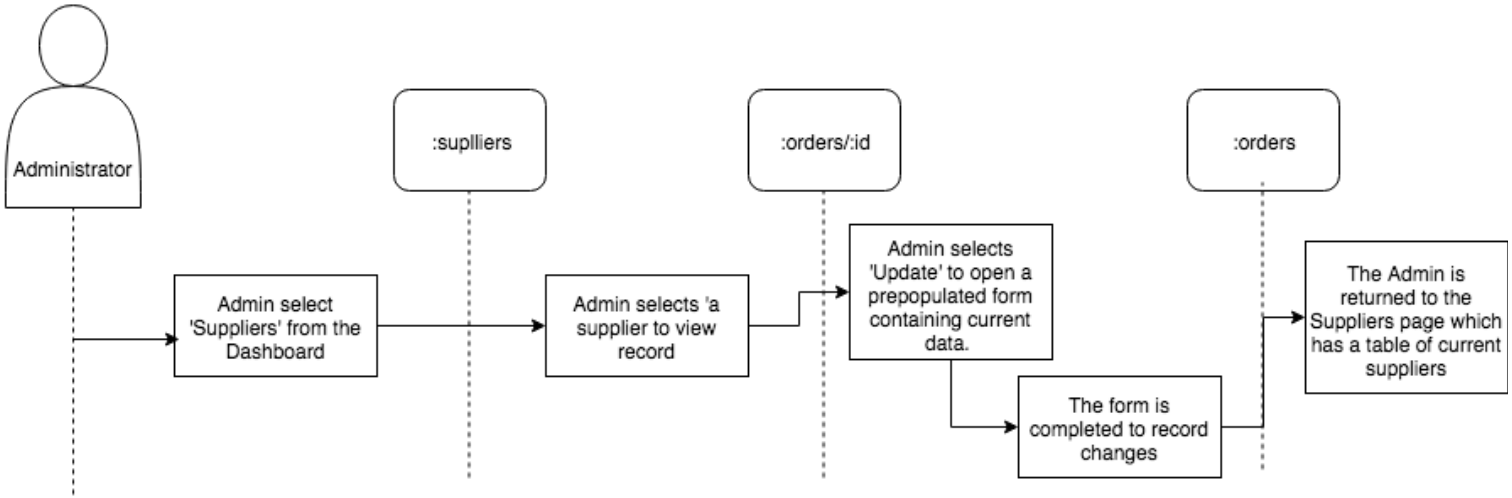


P 6 Two Wireframe Designs

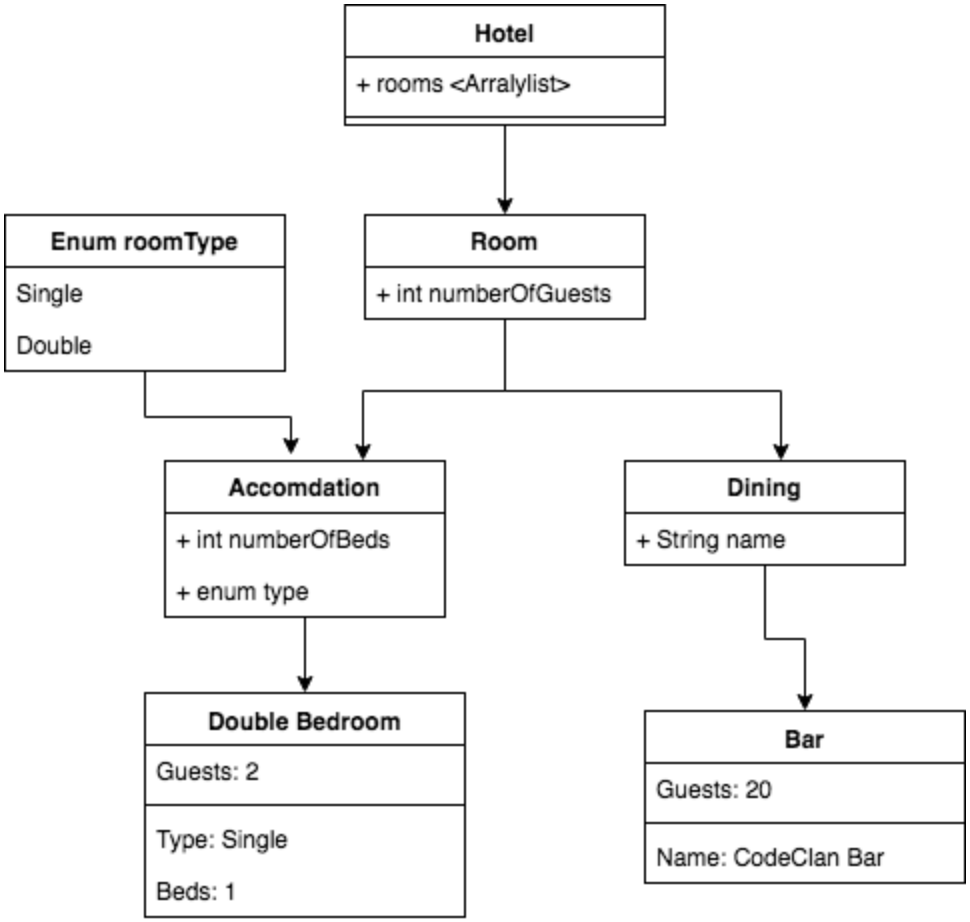


P 7 Two System Interaction Diagrams

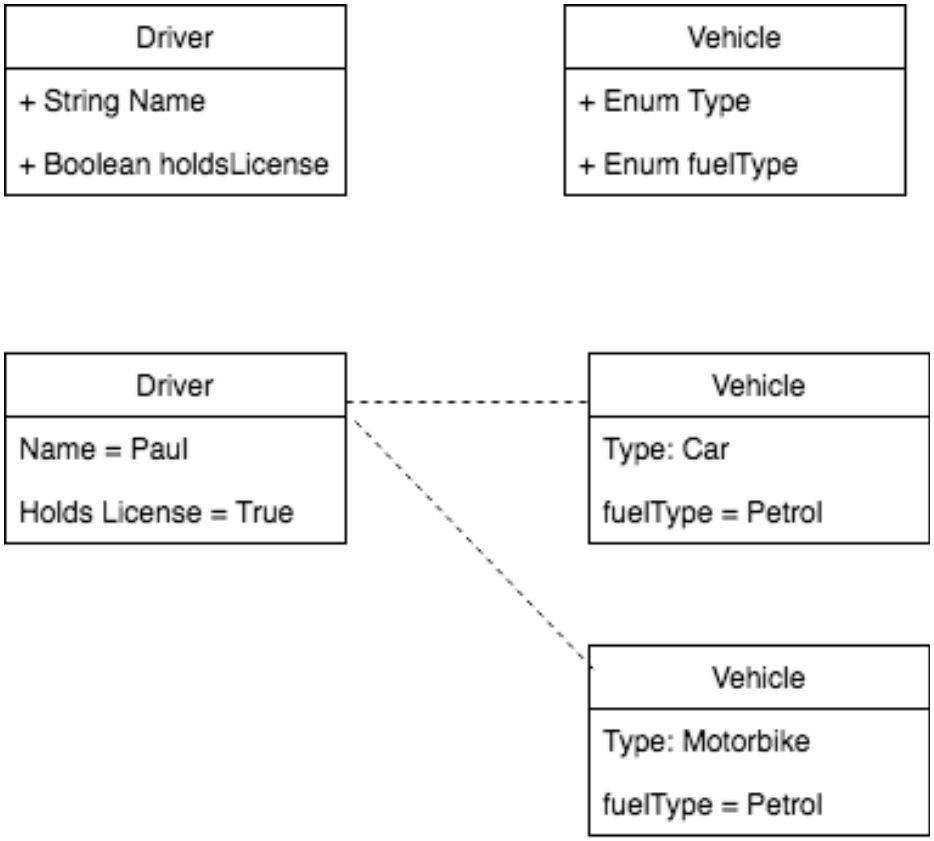




P 8.1 Two Object Diagrams



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.

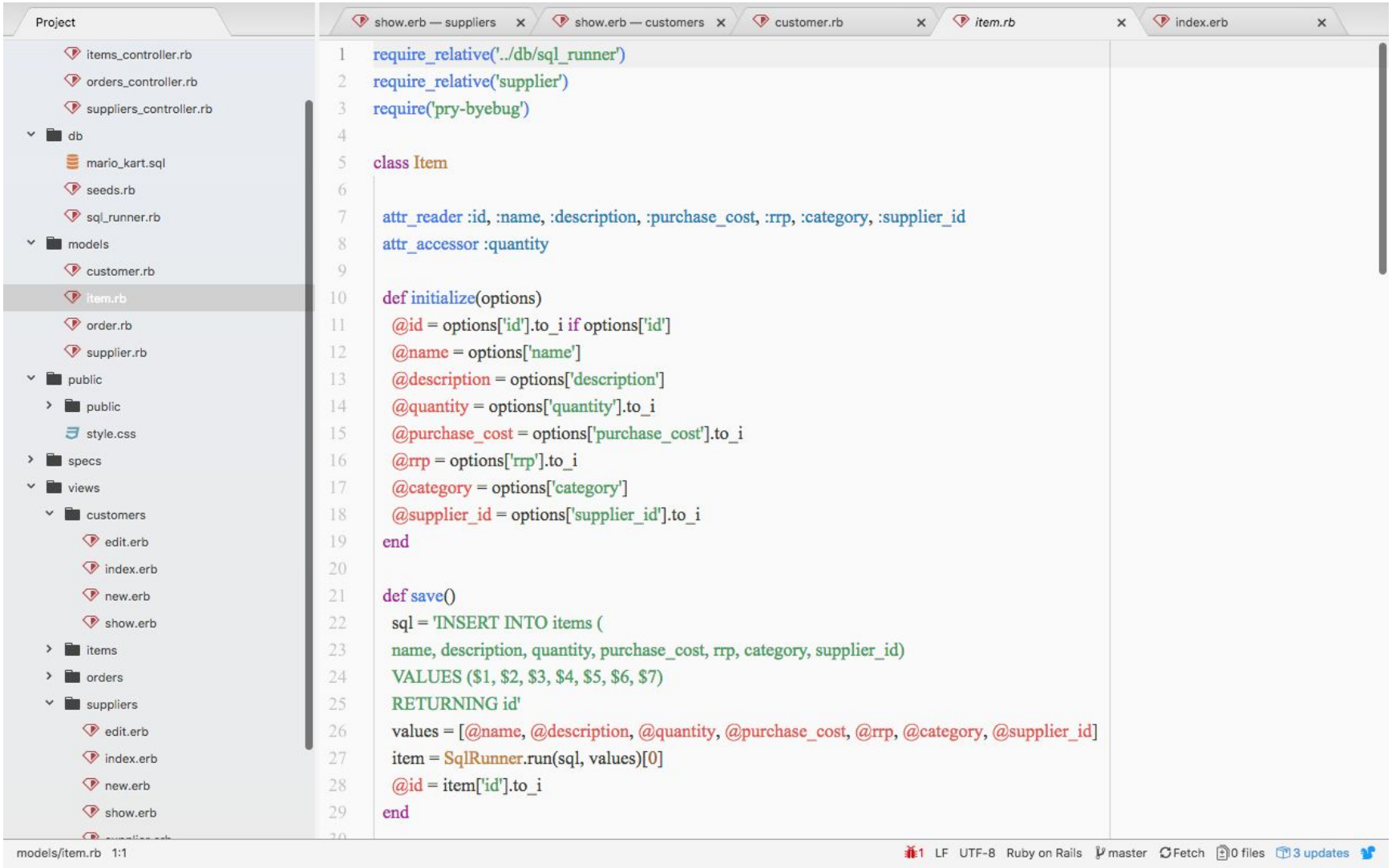
```
Cinema.prototype.filmsByProperty = function
  (property, value) {
 const objectArray = this.films.filter((film) => {
  return film[property] === value;
 });
 return objectArray;

};
```

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.

 */
```

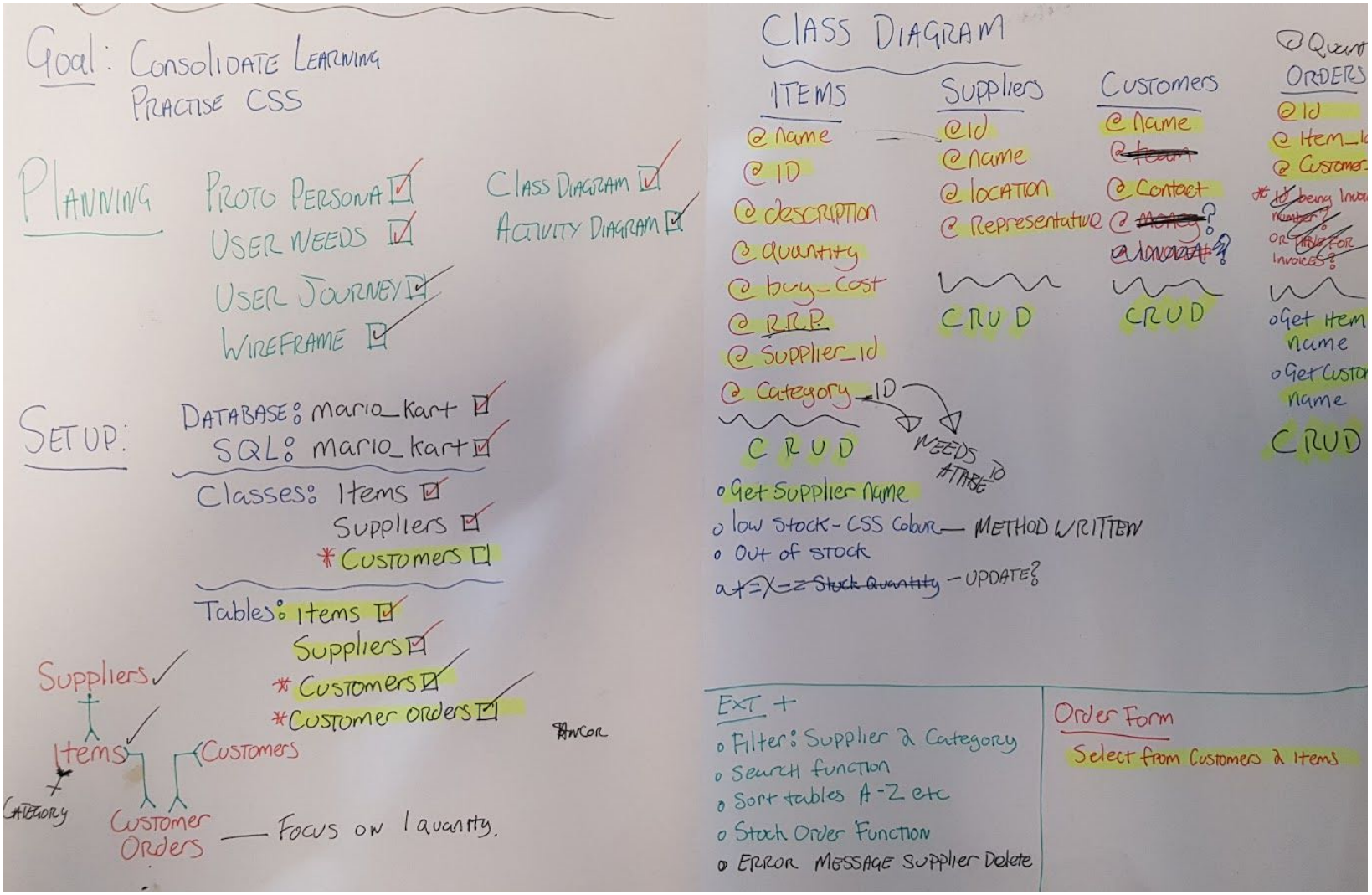
P 11 Solo Project with Github Link

https://github.com/PaulStevenson/project_mariokartshop_ruby_sql_html_css




```
require_relative('../db/sql_runner')
require_relative('supplier')
require('pry-byebug')

class Item

  attr_reader :id, :name, :description, :purchase_cost, :rrp, :category, :supplier_id
  attr_accessor :quantity

  def initialize(options)
    @id = options['id'].to_i if options['id']
    @name = options['name']
    @description = options['description']
    @quantity = options['quantity'].to_i
    @purchase_cost = options['purchase_cost'].to_i
    @rrp = options['rrp'].to_i
    @category = options['category']
    @supplier_id = options['supplier_id'].to_i
  end

  def save()
    sql = 'INSERT INTO items (
    name, description, quantity, purchase_cost, rrp, category, supplier_id)
    VALUES ($1, $2, $3, $4, $5, $6, $7)
    RETURNING id'
    values = [@name, @description, @quantity, @purchase_cost, @rrp, @category, @supplier_id]
    item = SqlRunner.run(sql, values)[0]
    @id = item['id'].to_i
  end
```

P 12 Project Planning with changes



P 13 Processing User Input

DRONE STRIKE!!!

P 14 Data Persistence

Dashboard Orders Items Suppliers Racers

Add Item

Name CodeClan Description Example of data input Quantity 1 Purchase Cost 100 RRP 100 Category Defence Supplier Banana Boy Submit New Item

Item Profile

Item: CodeClan

Description: Example of data input

Quantity: 1

Purchase Cost: 100

RRP: 100

Category: Defence

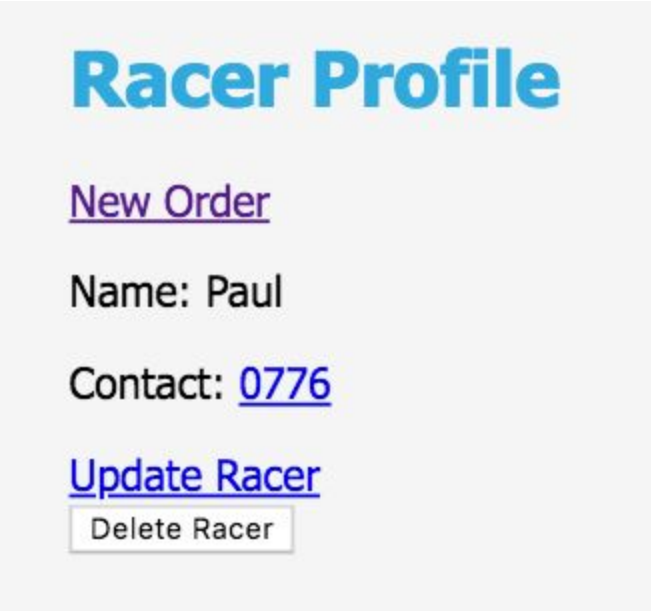
Supplier: Banana Boy

Update Item

Delete Item

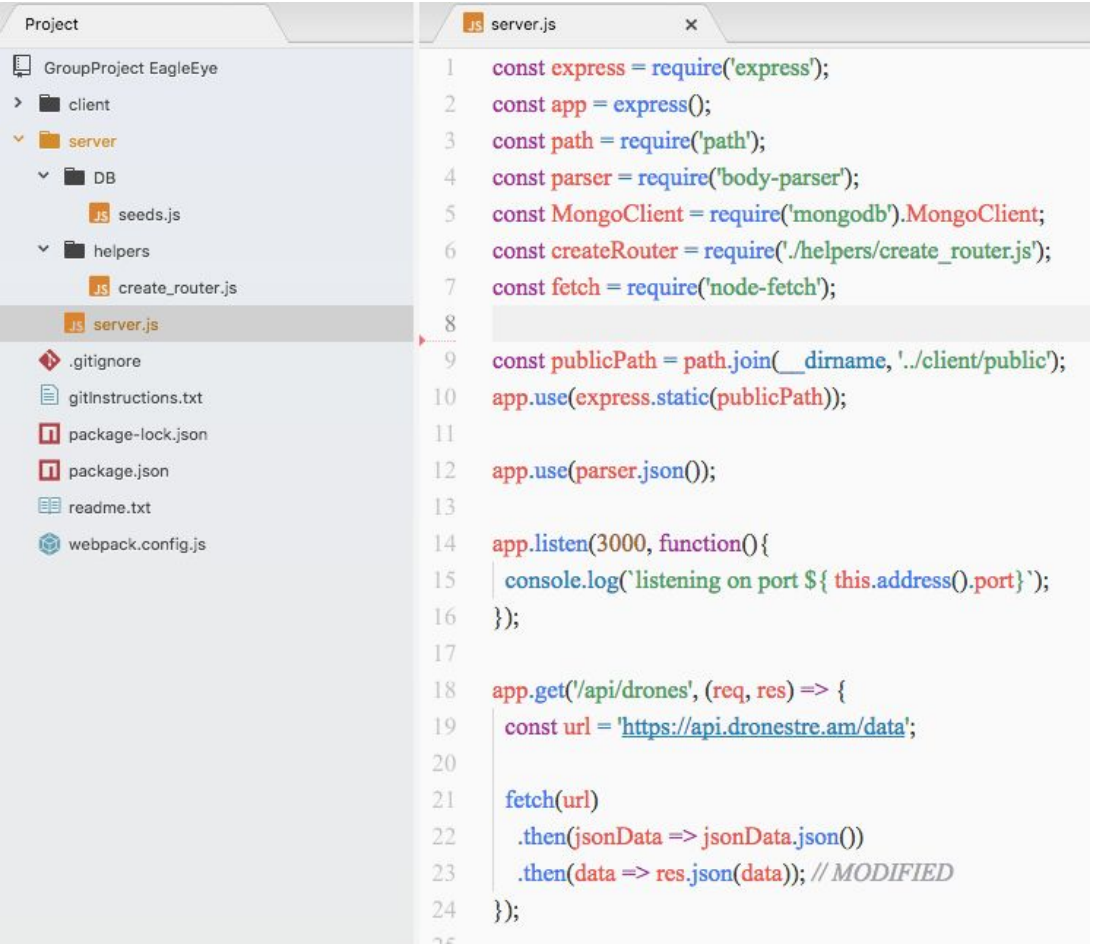
P 15 Show the correct output of results and feedback to user.

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



P 16 Use of an API

- The code that uses or implements the API



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

P 17 Bug Tracking Report

Test	**Fail**	**Solution**	**Pass/Fail**
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

shoppingBasketTest 〉 src 〉 test 〉 java 〉 ItemTest 〉

ItemTest.canGetPrice

ItemTest.java | BasketTest.java | CheckoutTest.java | DiscountTest.java

```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ItemTest {

    Item item;

    @Before
    public void before(){
        item = new Item( name: "item1",  price: 10,  quantity: 1,  bogof: false);

    }

    @Test
    public void canGetName(){
        assertEquals( expected: "item1", item.getName());
    }

    @Test
    public void canGetPrice(){
        assertEquals( expected: 10, item.getPrice(),  delta: 0.01);
    }

    @Test
    public void canGetQuantity(){
        assertEquals( expected: 1, item.getQuantity());
    }

    @Test
    public void canSetQuantity(){
        assertEquals( expected: 2, item.setQuantity(2));
```

ItemTest

ItemTest.java

```java
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ItemTest {

    Item item;

    @Before
    public void before(){
        item = new Item( name: "item1",  price: 10,  quantity: 1,  bogof: false);

    }

    @Test
    public void canGetName(){
        assertEquals( expected: "item1", item.getName());
    }

    @Test
    public void canGetPrice(){
        assertEquals( expected: 10, item.getPrice(),  delta: 0.01);
    }

    @Test
    public void canGetQuantity(){
        assertEquals( expected: 1, item.getQuantity());
    }

    @Test
    public void canSetQuantity(){
        assertEquals( expected: 2, item.setQuantity(2));
    }
```

ItemTest 〉 canGetPrice()

Run ItemTest.canGetPrice

1 test passed - 6ms

ItemTest 6ms
canGetPrice 6ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java ...

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.canGetPrice

ItemTest.java | BasketTest.java | CheckoutTest.java | DiscountTest.java

```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ItemTest {

    Item item;

    @Before
    public void before(){
        item = new Item( name: "item1",  price: 10,  quantity: 1,  bogof: false);

    }

    @Test
    public void canGetName(){
        assertEquals( expected: "item1", item.getName());
    }

    @Test
    public void canGetPrice(){
        assertEquals( expected: 11, item.getPrice(),  delta: 0.01);
    }

    @Test
    public void canGetQuantity(){
        assertEquals( expected: 1, item.getQuantity());
    }

    @Test
    public void canSetQuantity(){
        assertEquals( expected: 2, item.setQuantity(2));
```

ItemTest

ItemTest.java

```java
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class ItemTest {

    Item item;

    @Before
    public void before(){
        item = new Item( name: "item1",  price: 10,  quantity: 1,  bogof: false);

    }

    @Test
    public void canGetName(){
        assertEquals( expected: "item1", item.getName());
    }

    @Test
    public void canGetPrice(){
        assertEquals( expected: 11, item.getPrice(),  delta: 0.01);
    }

    @Test
    public void canGetQuantity(){
        assertEquals( expected: 1, item.getQuantity());
    }

    @Test
    public void canSetQuantity(){
        assertEquals( expected: 2, item.setQuantity(2));
    }
```

ItemTest > canGetPrice()

Run ItemTest.canGetPrice

1 test failed - 21ms

ItemTest 21ms
canGetPrice 21ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java ...

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