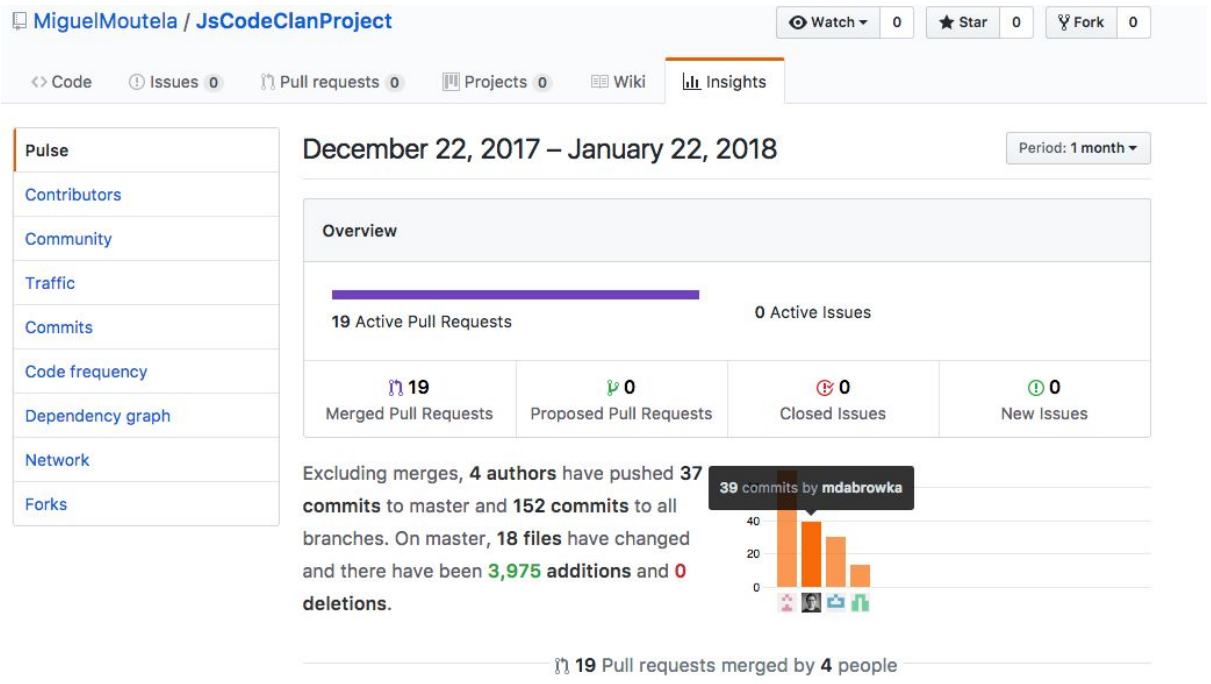
Evidence for Project Unit

Your name here Marta Dabrowka
Your Cohort E16
Date here: 25.01.2018

P- 1 Github Contributors page



P- 2 Project Brief

Users looking to attend events around them or in a city of their choice are to be able to view relevant events in a table. Use an existing API or create a new API to display information about times, location and content of events.

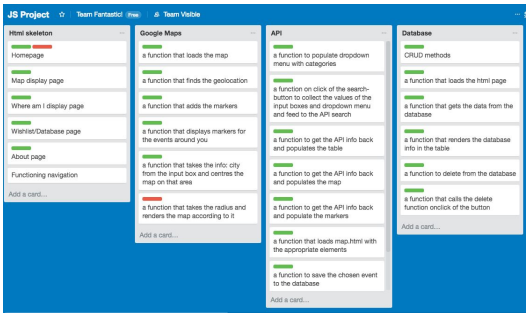
APIs used:

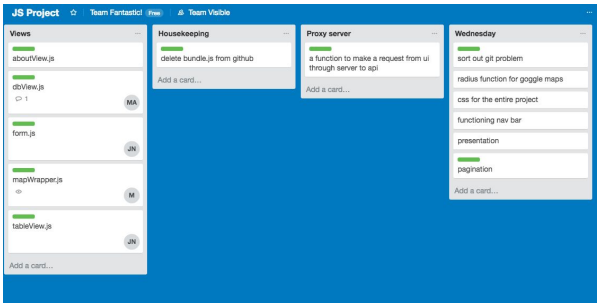
- Eventful API and Google Maps API

MVP user has to be able to:

- Display events on the map
- Display info about the events in a table and on pop-up windows
- Save chosen events to the database

P-3 Use of Trello

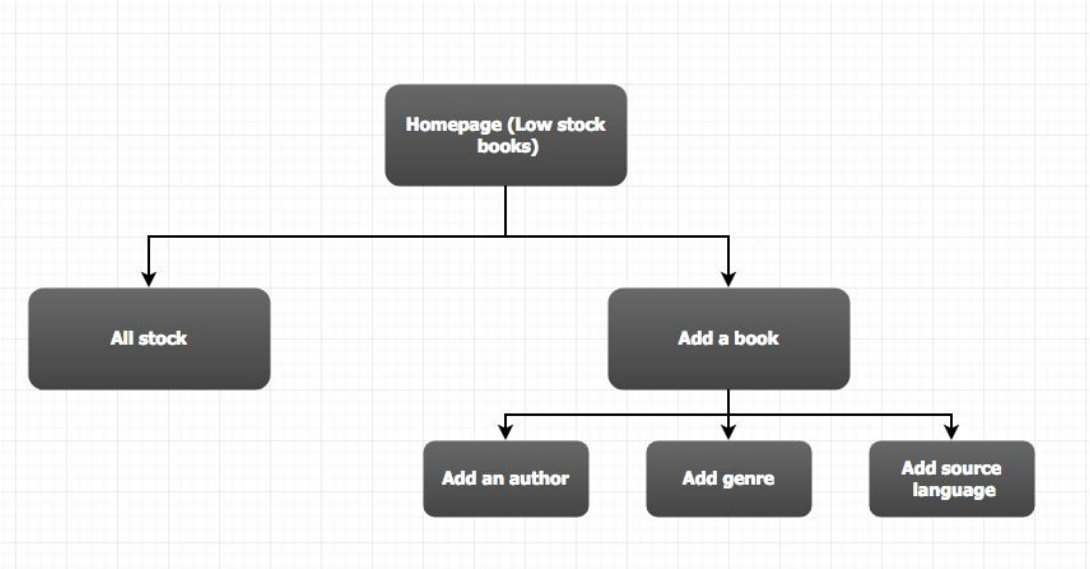




P-4 Acceptance Criteria

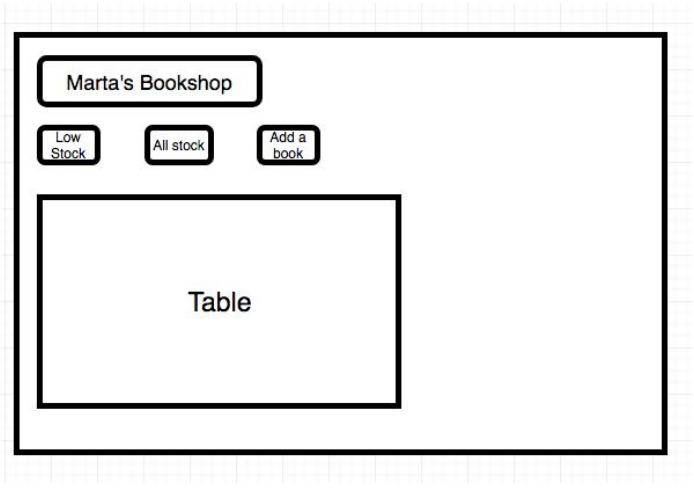
Acceptance Criteria	Expected Result/Output	Pass / Fail
A user is able to see the events around her	Events that match the specified criteria displayed in the table and marked on the map	PASS
A user is able to input the city and display the events in the input city	Events that match the specified criteria displayed in the table and marked on the map	PASS
A user is able to save a chosen event to the database	When the 'save' button is clicked, the chosen event is saved to the database and displayed on the Wishlist page	PASS
A user is able to delete the chosen event from the database	When the 'delete' button is clicked, the chosen event is deleted from the database and no longer displayed on the Wishlist page	PASS

P-5 User sitemap

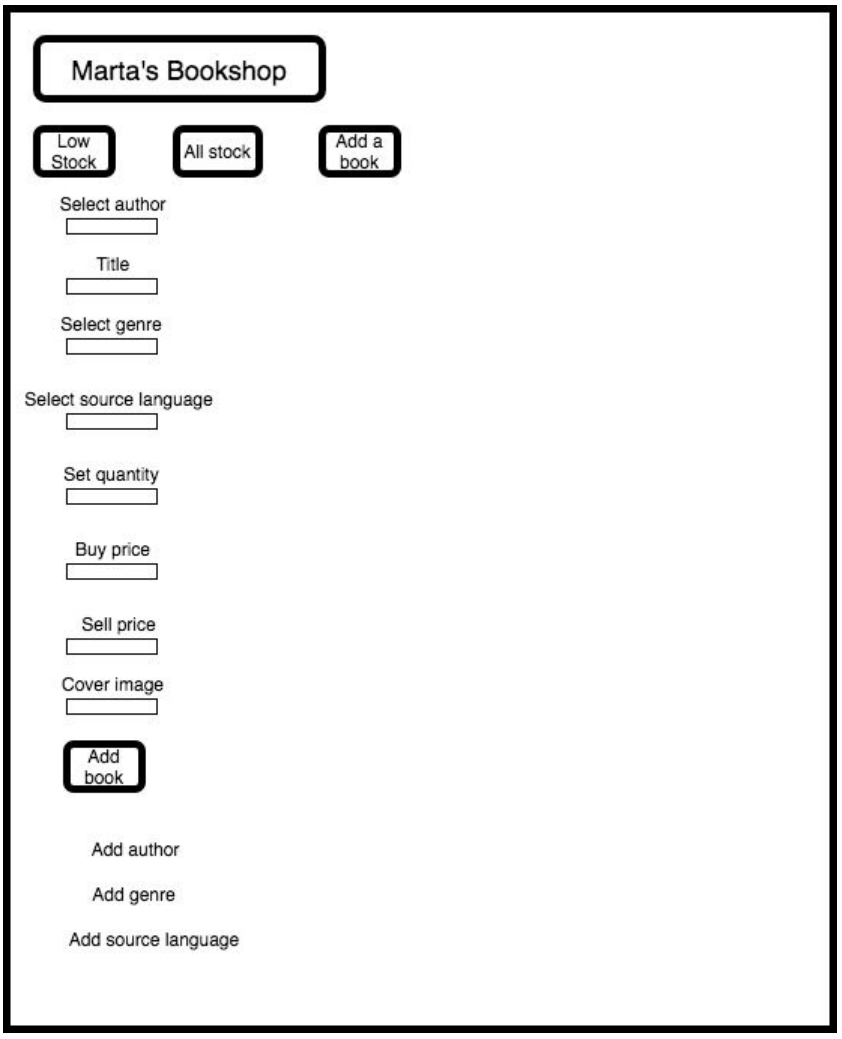


P-6 Wireframes designs

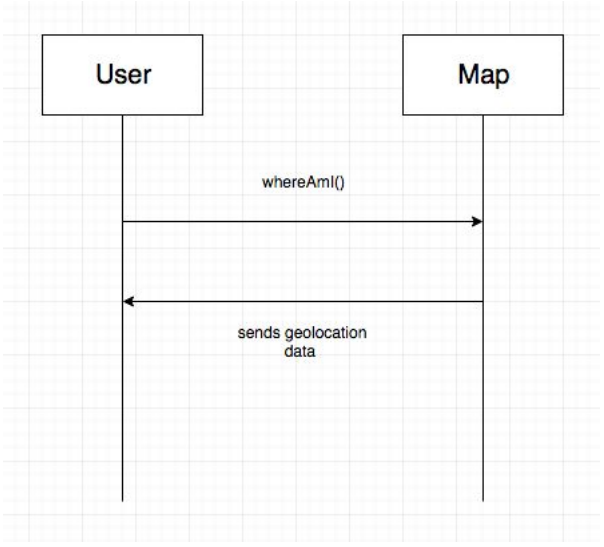
Wireframe 1:

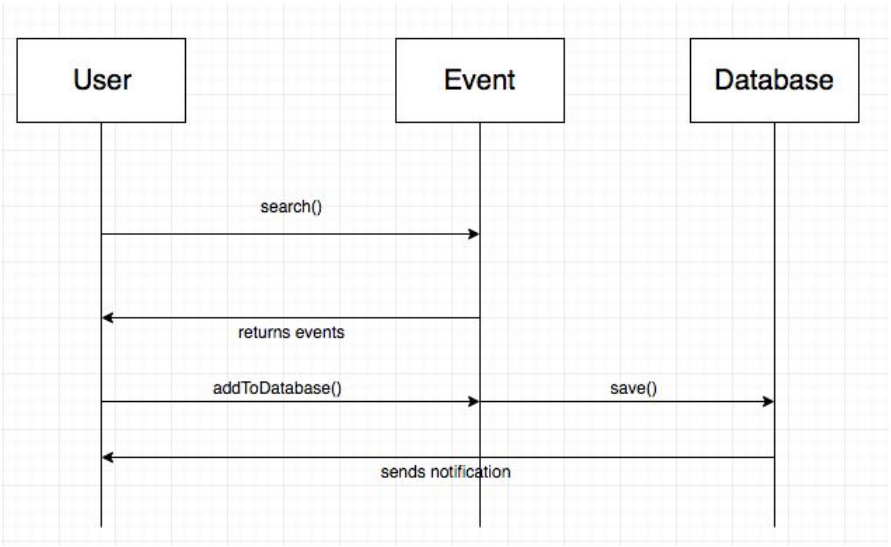


Wireframe 2:

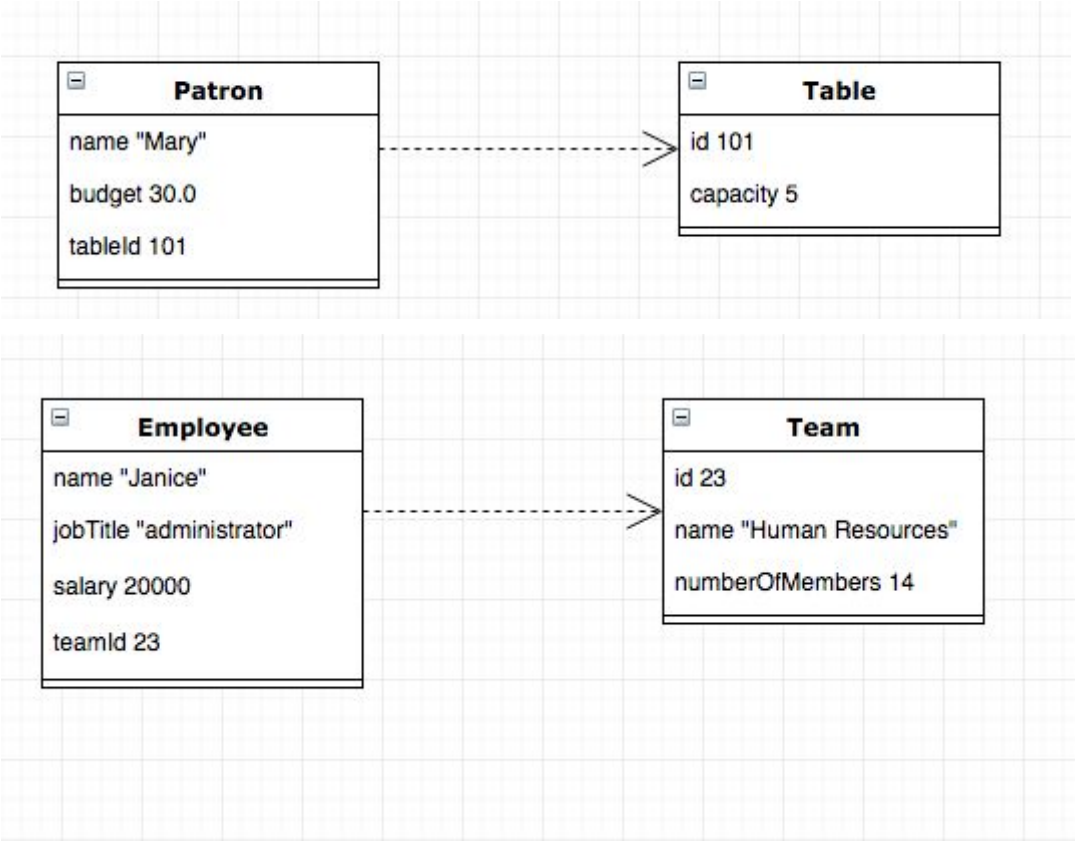


P-7 System interactions diagrams





P-8 Two Object Diagrams



P- 9 D.T.- a Choice of two algorithms (find the algorithms on a program you might have written, show the code you have used.)

A - Search Algorithm- For a bookstore management system project I needed a way to return all the books by a certain author. I used a search algorithm, where each book had an author with a unique ID. I had passed the author's ID into the sql query and mapped the results into an array of books with the author ID I was after.

```
def find_books
  sql = "SELECT * FROM books WHERE author_id = $1"
  values = [@id]
  results = SqlRunner.run(sql, values)
  books_array = results.map{|book| Book.new(book)}
  return books_array
end
```

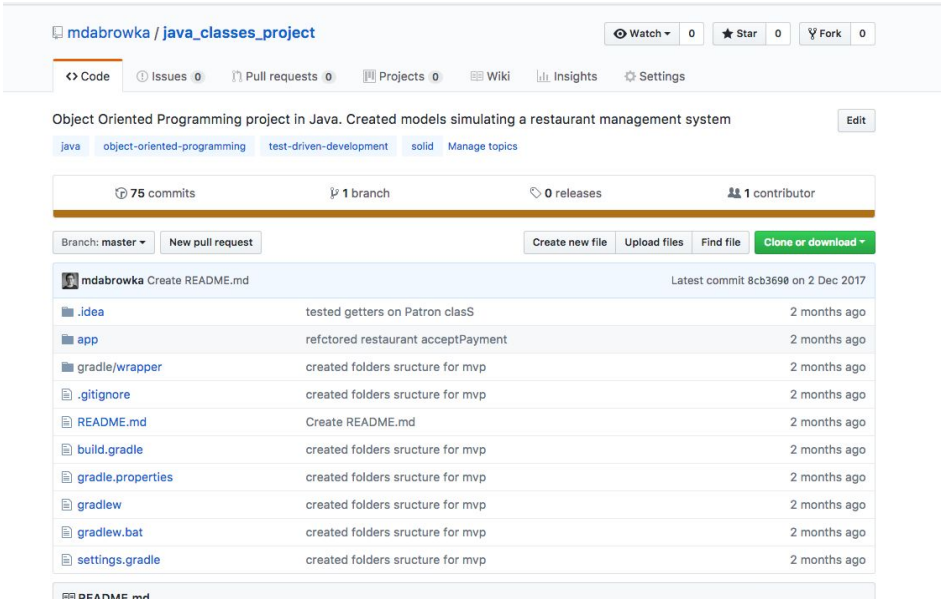
B- Update Algorithm - In the same project I needed to be able update the book's details, e.g. the stock levels, when sales have been made. The update algorithm allows me to find a book by its unique ID and update the fields in the database as required. The updates are saved to the database.

```
def update
  sql = "UPDATE books SET (title, author_id, quantity, genre_id,
  source_language_id, buy_price, sell_price, cover_image) =
  ($1, $2, $3, $4, $5, $6, $7, $8 ) WHERE id = $9"
  values = [@title, @author_id, @quantity, @genre_id,
    @source_language_id, @buy_price, @sell_price, @cover_image, @id]
  SqlRunner.run(sql, values)
end
```

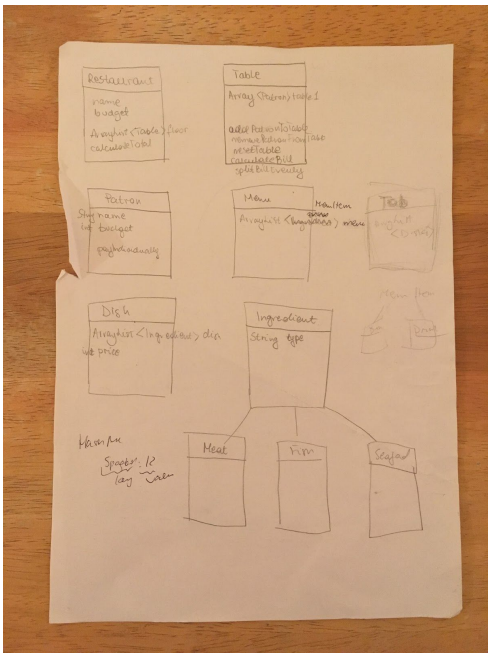
P - 10 Example of Pseudocode

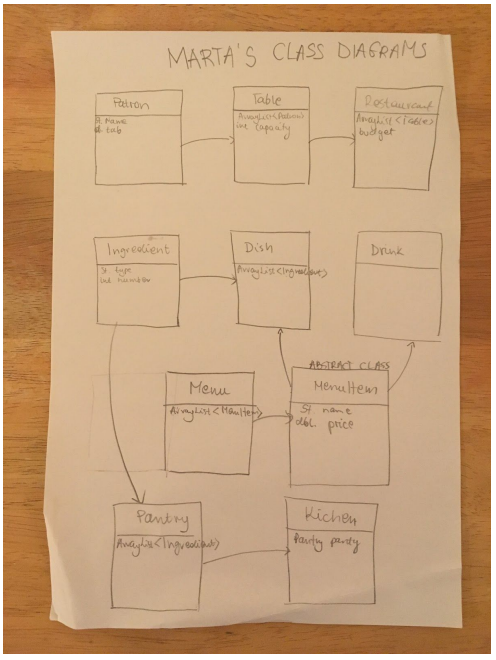
```
//public void function to accept payment from a table (takes in a table as a parameter) {
// takes the value of the bill from the table that is passed in
// adds the cash to the restaurant budget
// resets the table bill to 0
//}
```

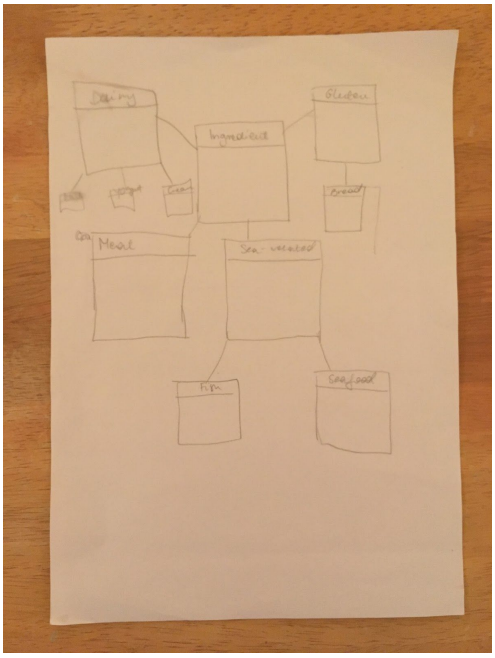
P - 11 Github link to one of your projects link:
https://github.com/mdabrowka/java_classes_project

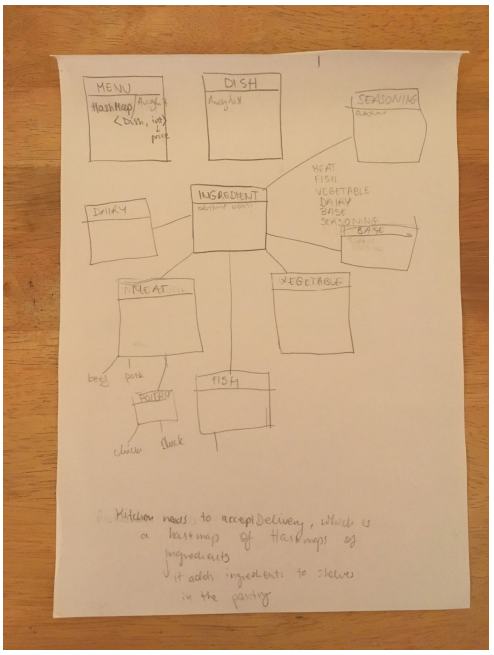


P - 12 Screenshot of your planning and the different stages of development to show changes.

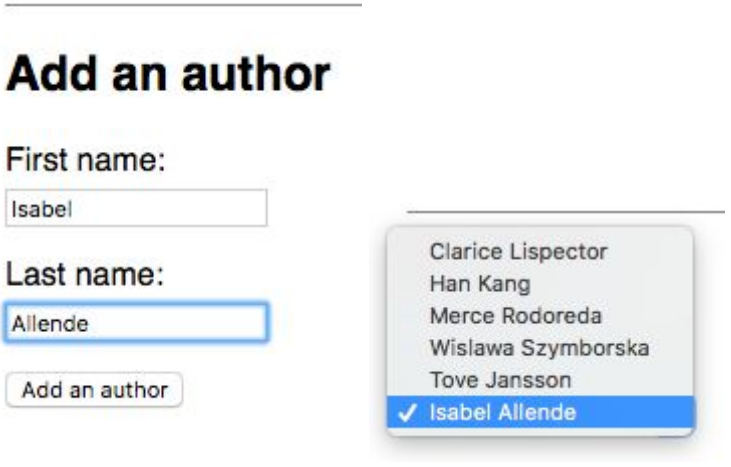








P - 13 User input



P - 14 Interaction with data persistence

Add a book

Select author:

Han Kang

Title:

The White Book

Select genre:

autobiography

Select source language:

Korean

Set quantity:

5

Buy price (£):

5

Sell price (£):

10

Cover image:

Add a book

Marta's Bookshop

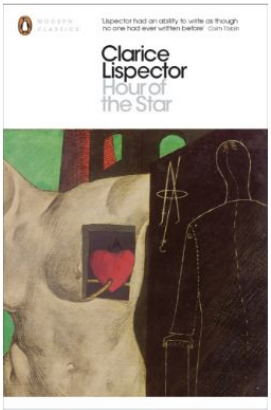
Low Stock Books | All Stock | Add a book

You have added a new book

P - 15 User output result

User updates the stock of Clarice Lispector's "The Hour of the Star" from 12 to 2 copies. The quantity is saved and the stock level goes from 'medium' to 'low'.

The Hour of a Star by Clarice Lispector



- Genre: novel
- Source language: Brazilian Portuguese

Stock	In stock	Buy price	Sell price	Markup
Medium	12	£6	£13	217.0

Edit Book

Edit The Hour of a Star by Clarice Lispector

Select author:

Clarice Lispector

Title:

The Hour of a Star

Select genre:

novel

Source language:

Brazilian Portuguese

Set quantity:

12

Buy price:

6

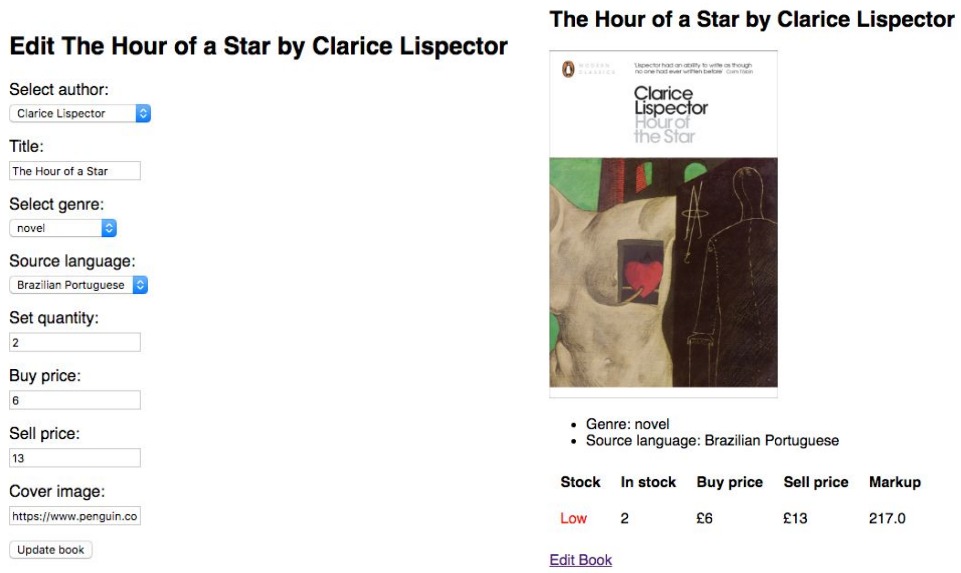
Sell price:

13

Cover image:

https://www.penguin.co

Update book

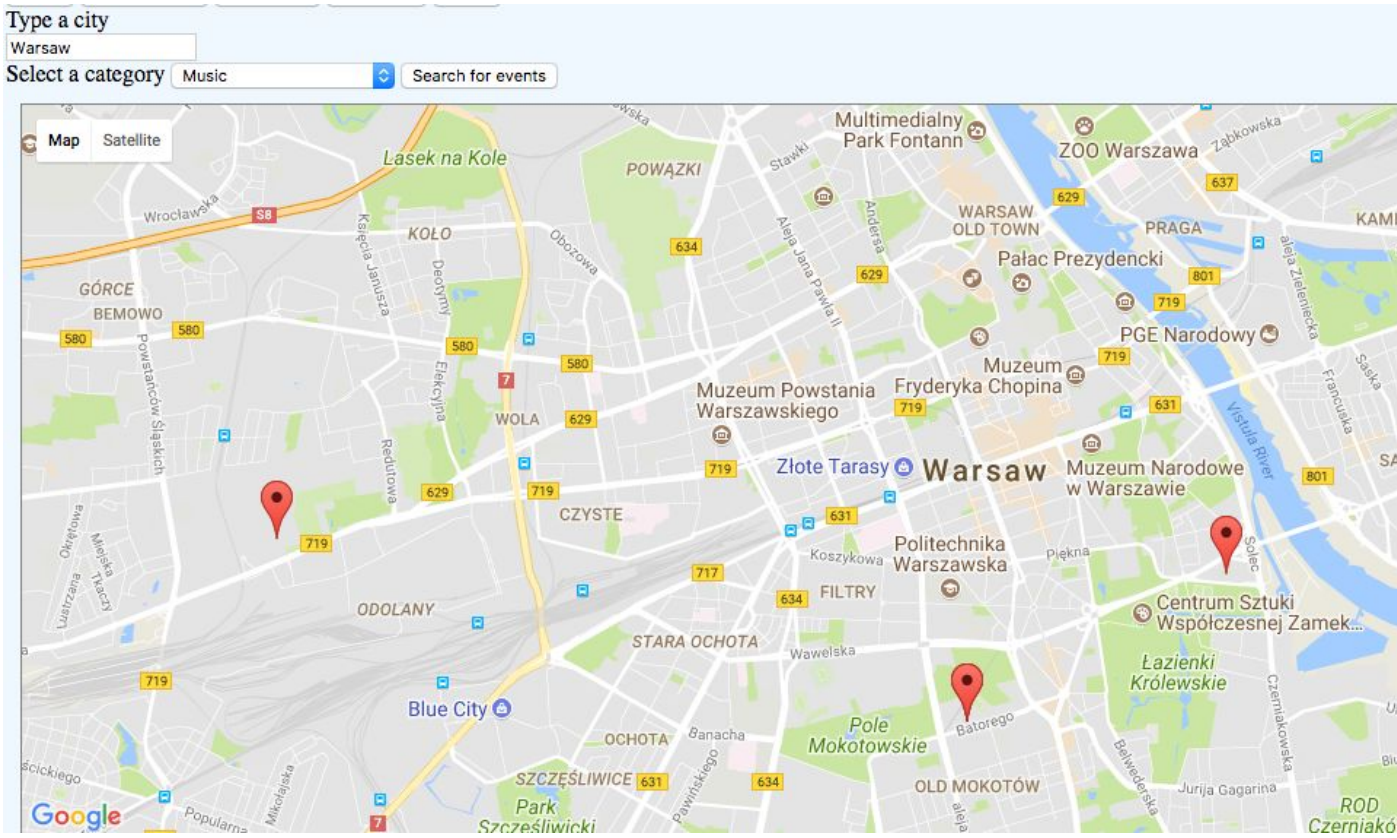


P-16 Show an API being used within your program

A screenshot of the code that uses API

```
MapWrapper.prototype.centerOnInputCity = function(city, map){
  const  geocoder = new google.maps.Geocoder();
  geocoder.geocode({'address': city}, function(results, status) {
    if (status === 'OK') {
      const result = results[0].geometry.location;
      const lat = result.lat();
      const lng = result.lng();
      const cityLocation = {
        lat,
        lng
      };
      this.map.setCenter(cityLocation);
      this.map.setZoom(15);
    };
  }.bind(this));
}
```

A screenshot of the API being used by the program whilst running



P - 17 Bug tracking report showing the errors diagnosed and corrected.

User is to find events in the area around her	**Failed**	**Added a geolocation functionality provided by the Google Maps API**	**Passed**
User is to save a chosen event to the database by clicking 'save' button			**Passed**
User is to see her location on the map	**Failed**	**Added a Google Maps marker with geolocation coordinates**	**Passed**
User is to be able to select events	**Failed**	**Added a dropdown menu with listed categories to choose**	**Passed**

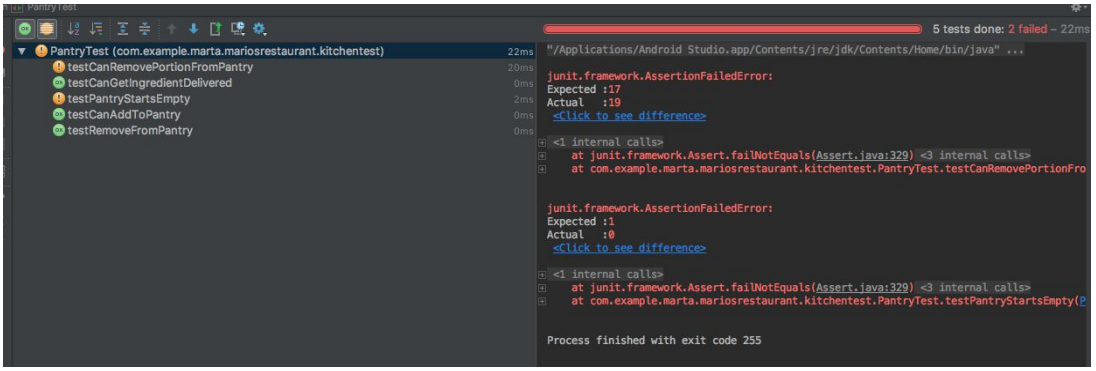
based on categories		from as provided by the API	

P -18 Testing your program

Code test:

```
public class PantryTest {
    Pantry pantry;
    Ingredient beef, ham, chicken, pumpkin, garlic, mustard, rice, pasta;

    @Before
    public void before() {
        pantry = new Pantry();
        beef = new Ingredient( type: "beef",  number: 15);
        ham = new Ingredient( type: "ham",  number: 20);
        chicken = new Ingredient( type: "chicken",  number: 20);
        pumpkin = new Ingredient( type: "pumpkin",  number: 7);
        garlic = new Ingredient( type: "garlic",  number: 25);
        mustard = new Ingredient( type: "mustard",  number: 10);
        rice = new Ingredient( type: "rice",  number: 50);
        pasta = new Ingredient( type: "pasta",  number: 50);
    }

    @Test
    public void testPantryStartsEmpty() { assertEquals( expected: 1, pantry.pantrySize()); }

    @Test
    public void testCanAddToPantry() {
        pantry.addToPantry(chicken);
        assertEquals( expected: 1, pantry.pantrySize());
    }

    @Test
    public void testRemoveFromPantry() {
        pantry.addToPantry(chicken);
        pantry.addToPantry(ham);
        pantry.removeFromPantry(chicken);
        pantry.removeFromPantry(chicken);
        assertEquals( expected: 1, pantry.pantrySize());
    }

    @Test
    public void testCanRemovePortionFromPantry() {
        pantry.addToPantry(chicken);
        pantry.removePortionFromPantry(chicken);
        assertEquals( expected: 17, pantry.checkIngredientLevel(chicken));
    }
```

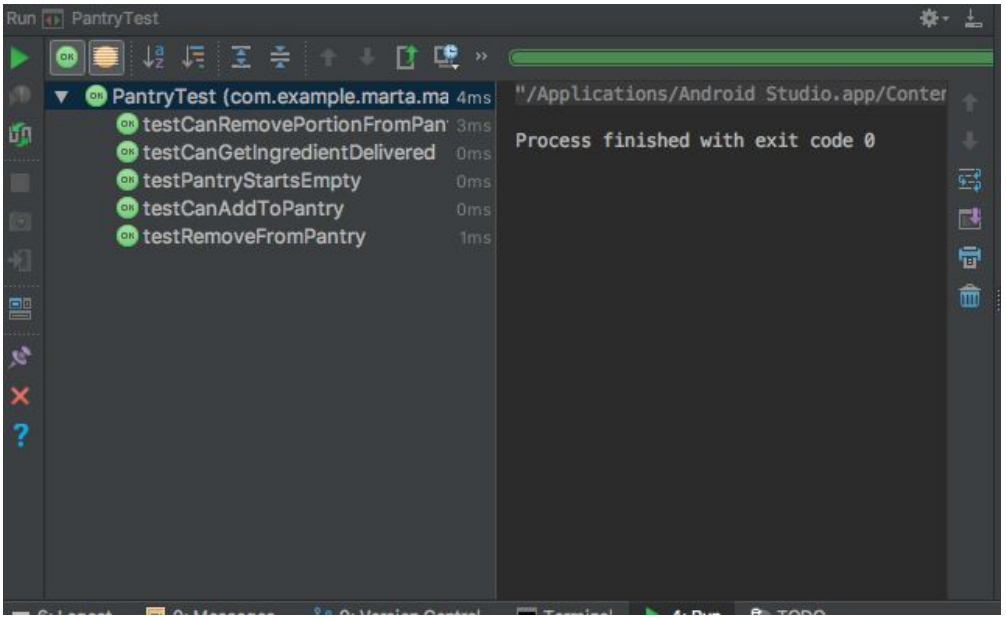
Test failing:



Test code fixed:

```java
public class PantryTest {
    Pantry pantry;
    Ingredient beef, ham, chicken, pumpkin, garlic, mustard, rice, pasta;

    @Before
    public void before() {
        pantry = new Pantry();
        beef = new Ingredient( type: "beef",  number: 15);
        ham = new Ingredient( type: "ham",  number: 20);
        chicken = new Ingredient( type: "chicken",  number: 20);
        pumpkin = new Ingredient( type: "pumpkin",  number: 7);
        garlic = new Ingredient( type: "garlic",  number: 25);
        mustard = new Ingredient( type: "mustard",  number: 10);
        rice = new Ingredient( type: "rice",  number: 50);
        pasta = new Ingredient( type: "pasta",  number: 50);
    }

    @Test
    public void testPantryStartsEmpty() { assertEquals( expected: 0, pantry.pantrySize()); }

    @Test
    public void testCanAddToPantry() {
        pantry.addToPantry(chicken);
        assertEquals( expected: 1, pantry.pantrySize());
    }

    @Test
    public void testRemoveFromPantry() {
        pantry.addToPantry(chicken);
        pantry.addToPantry(ham);
        pantry.removeFromPantry(chicken);
        assertEquals( expected: 1, pantry.pantrySize());
    }

    @Test
    public void testCanRemovePortionFromPantry() {
        pantry.addToPantry(chicken);
        pantry.removePortionFromPantry(chicken);
        assertEquals( expected: 19, pantry.checkIngredientLevel(chicken));
    }
```

Tests passing:


Run
PantryTest
PantryTest (com.example.marta.ma 4ms
testCanRemovePortionFromPan 3ms
testCanGetIngredientDelivered 0ms
testPantryStartsEmpty 0ms
testCanAddToPantry 0ms
testRemoveFromPantry 1ms
"/Applications/Android Studio.app/Conter
Process finished with exit code 0