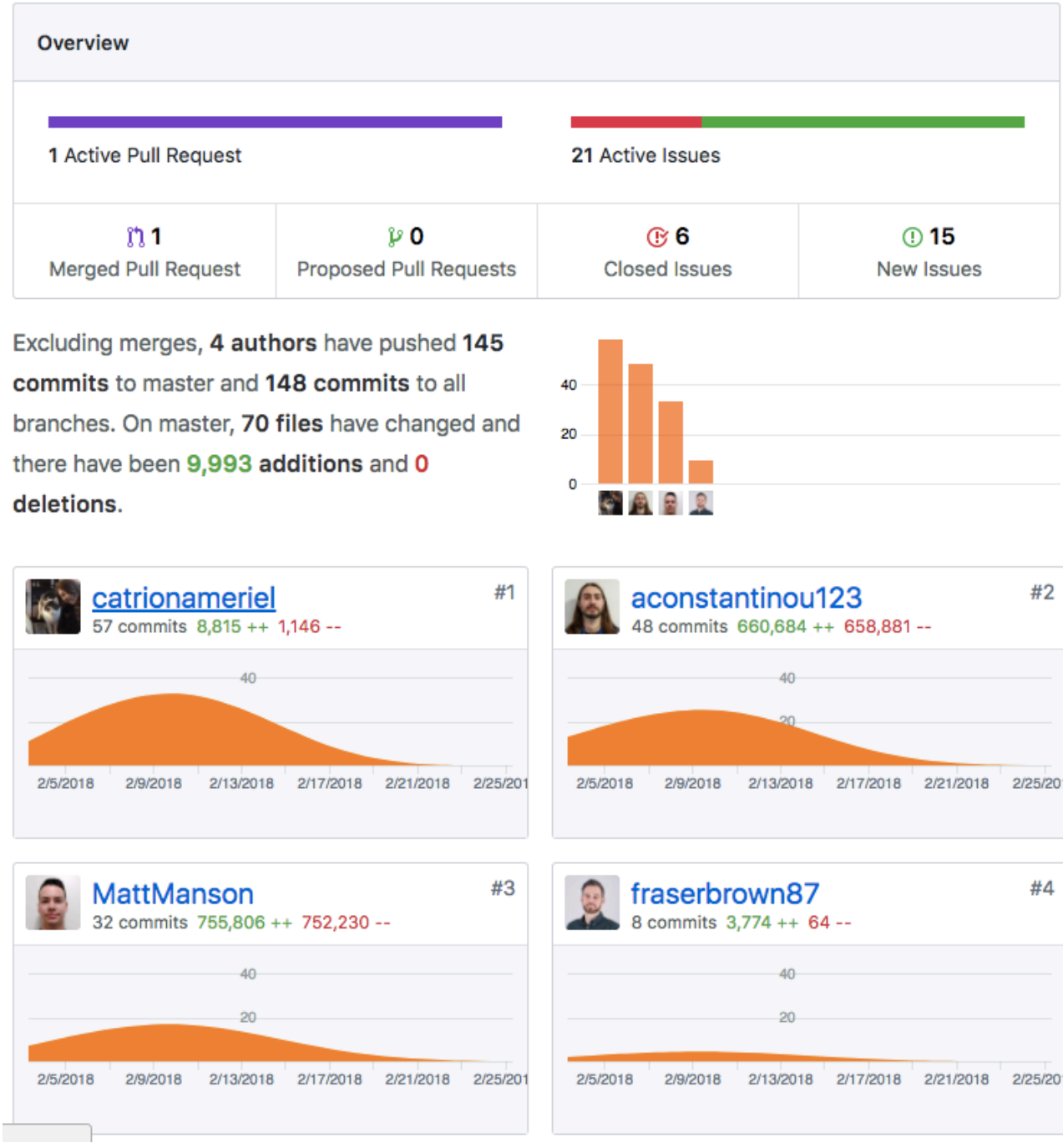
Evidence for Project Unit

Fraser Brown: Cohort E17

P. 1 Github Contributors page



P. 2 Project Brief

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

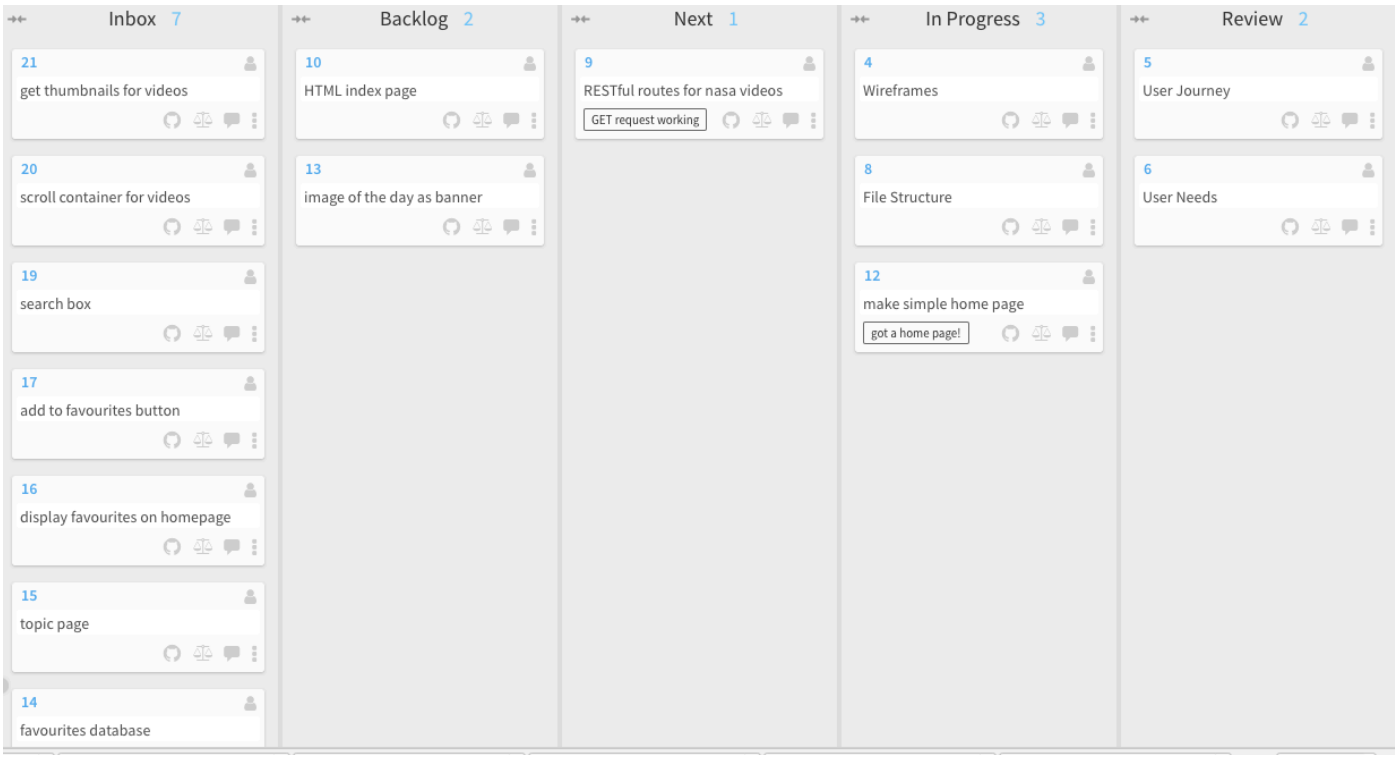
- Interactive map of a historical event - e.g. World War 1, the travels of Christopher Columbus

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

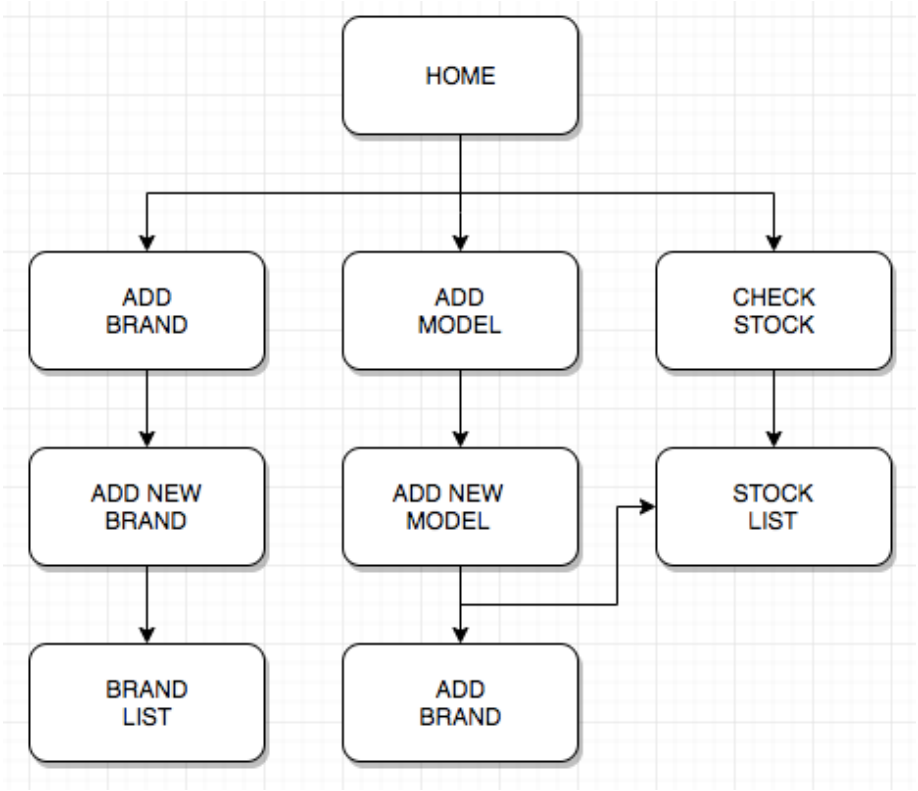
P. 3 Use of Trello (waffle.io)



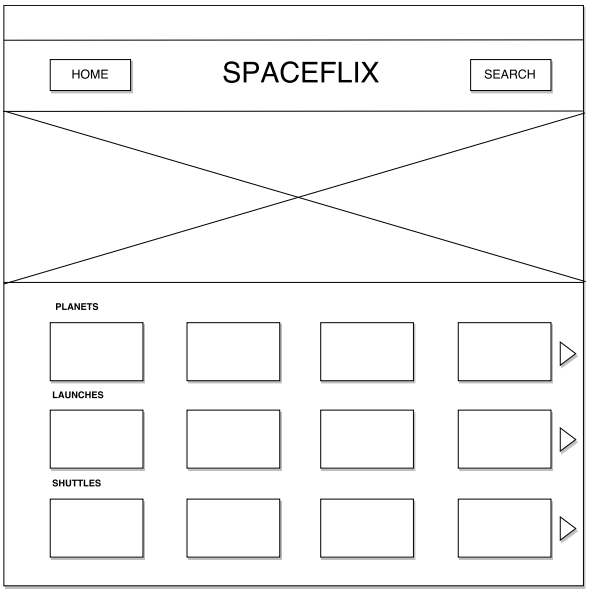
P. 4 Acceptance Criteria

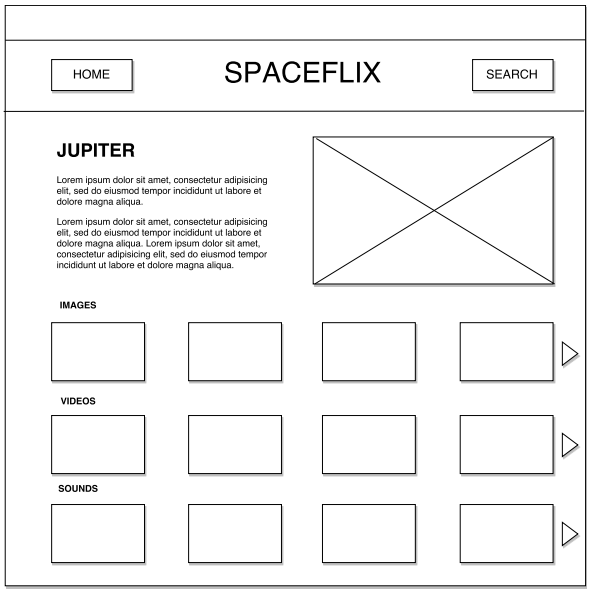
Acceptance Criteria	Expected Result/ Output	Pass/Fail
A user is able to select a planet.	A button that indicates which planet is provided and when clicked takes the user to that planet info and videos	PASS
A user can search for space related words	Key strokes that are accurate and word searched for correctly	PASS
A user can hear the nasa sound of the day.	A speaker button is available, and when clicked the sound is played by a NASA API	PASS
User is able to add to favourites.	User can click on “add to favourites” where videos are stored.	PASS

P. 5 User sitemap

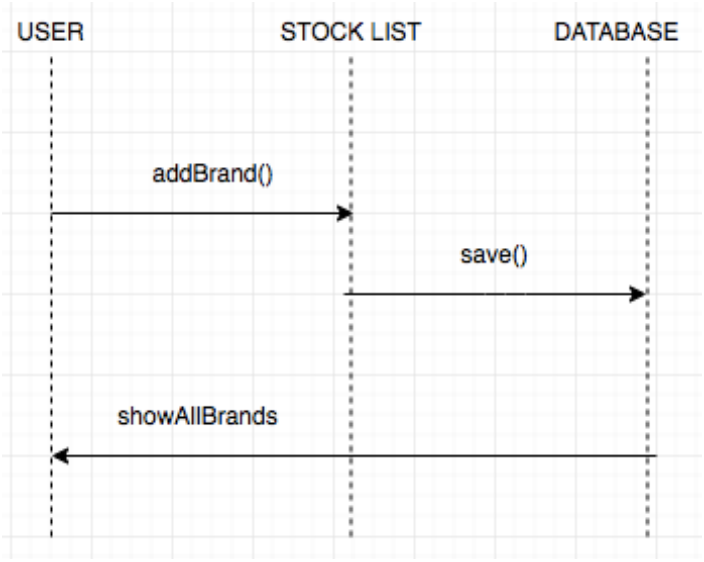


P. 6 Wireframes designs

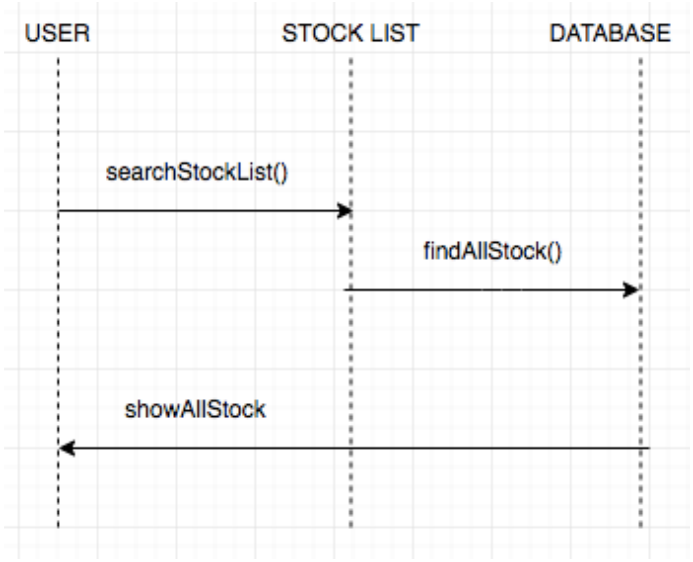




P. 7 System interactions diagrams

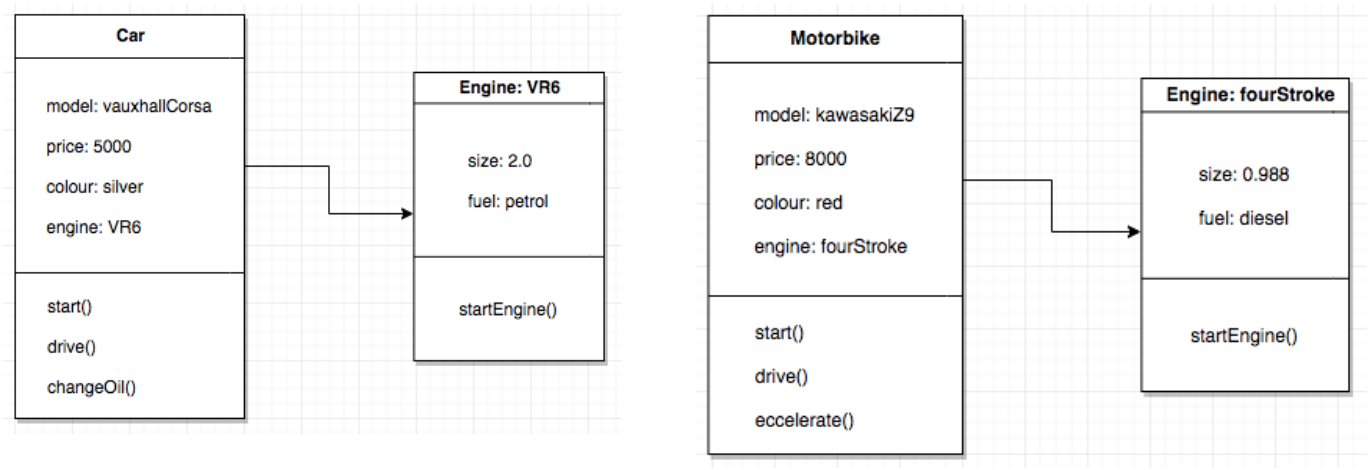


Adding a brand to the stock list.



Searching all stock list

P. 8 Two Object Diagrams



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

```
Bank.prototype.findAccount = function(searchName){
  for (account of this.accounts) {
    if (searchName === account.name){
      return account;
    }
  }
};
```

A search function that returns the back account, from an array or accounts, associated with the searchName provided.

```
Bank.prototype.totalValueByType = function(findType){
  var totalAccountValue = 0;
  for (account of this.accounts) {
    if (findType === account.type){
      totalAccountValue += account.balance;
    }
  }
  return totalAccountValue;
};
```

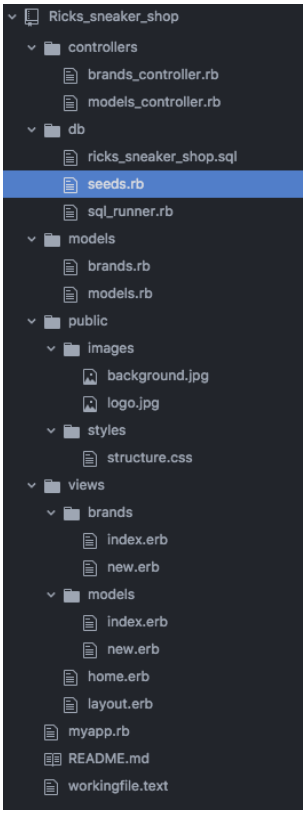
A function that loops through the accounts array and returns an accumulative total for all bank accounts.

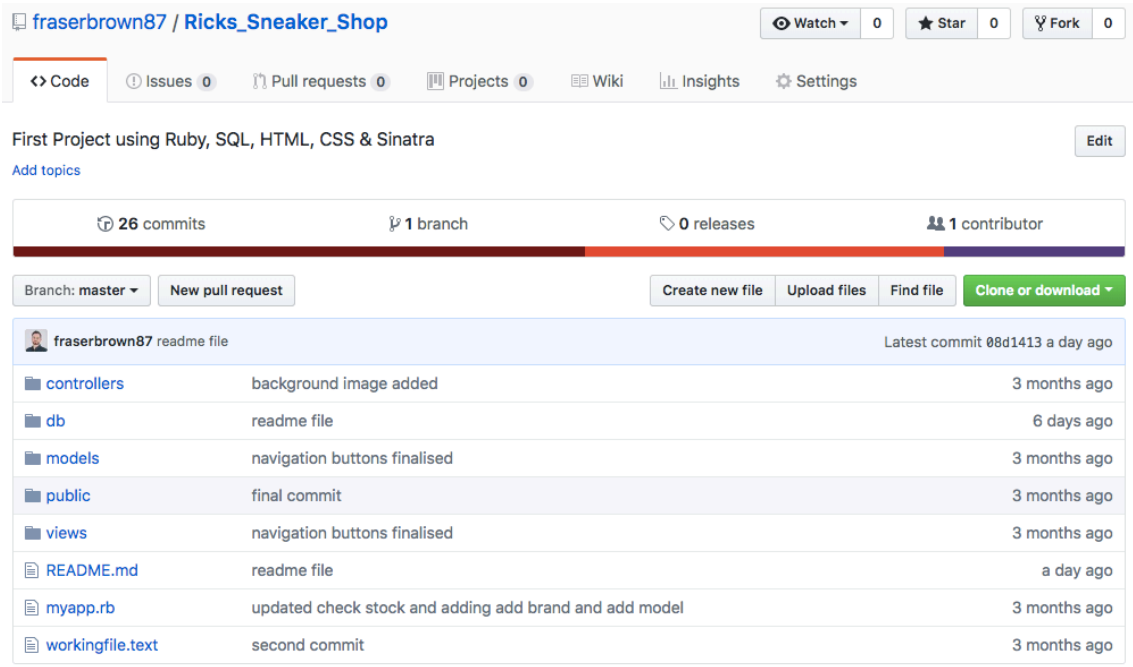
P. 10 Example of Pseudocode

```
componentDidMount(){
  const url = 'https://rss.itunes.apple.com/api/v1/gb/books/top-paid/all/25/explicit.json';
  // console.log('hello componentDidMount');
  //create a new XMLHttpRequest
  const xhr = new XMLHttpRequest();
  //open the request
  xhr.open('GET', url);
  //set an eventListener to the onLoad event
  xhr.addEventListener('load', () => {
    if(xhr.status !== 200) return;

    const jsonString = xhr.responseText;
    const data = JSON.parse(jsonString);
    console.log(data);
    //setState of countries to be the countries from the API
    this.setState({
      books: data
    })
  })

  xhr.send()
}
```

P. 11 Github link to one of your projects

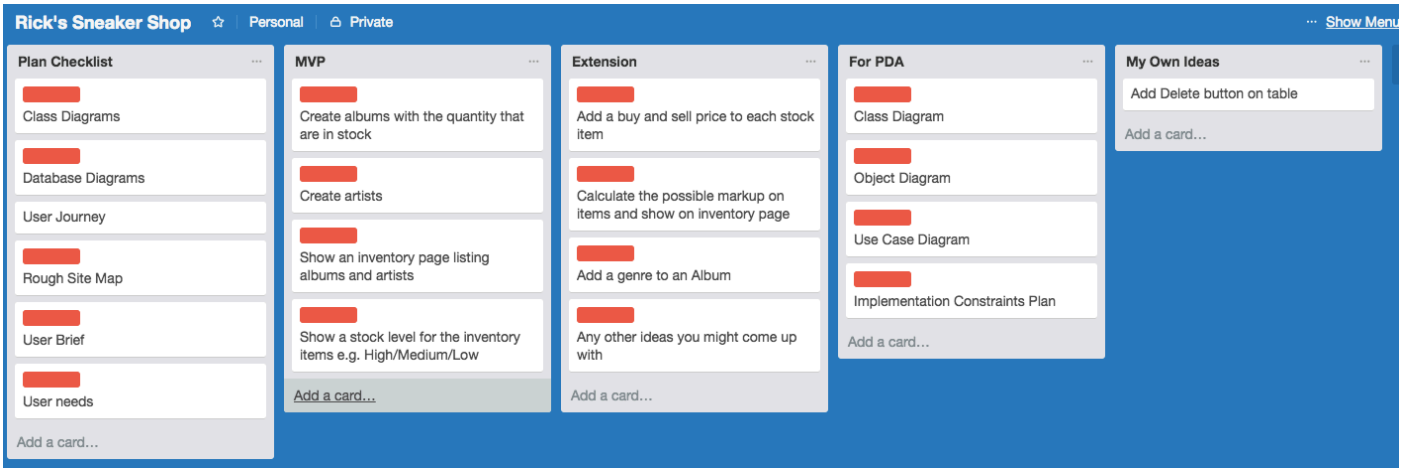
https://github.com/fraserbrown87/Ricks_Sneaker_Shop



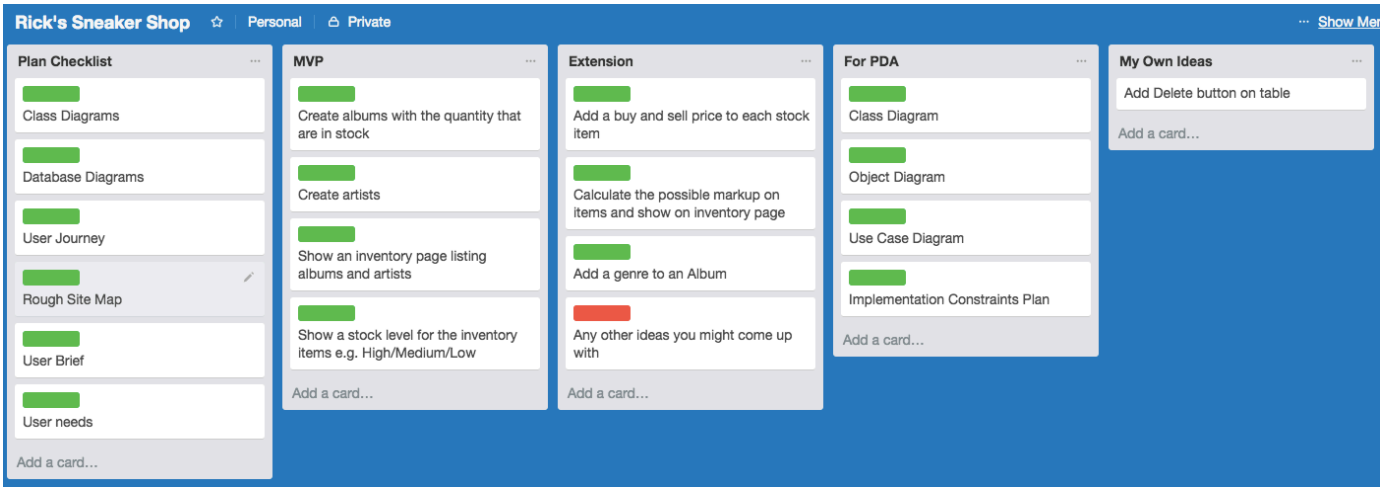


P. 12 Screenshot of your planning and the different stages of development to show changes.

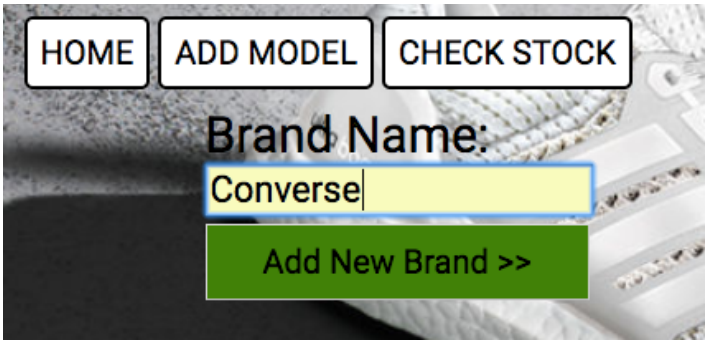
1.

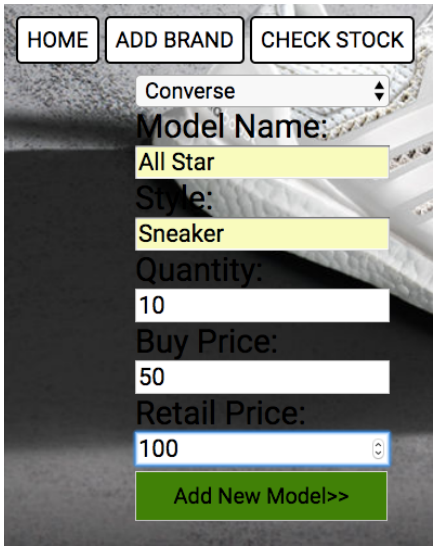


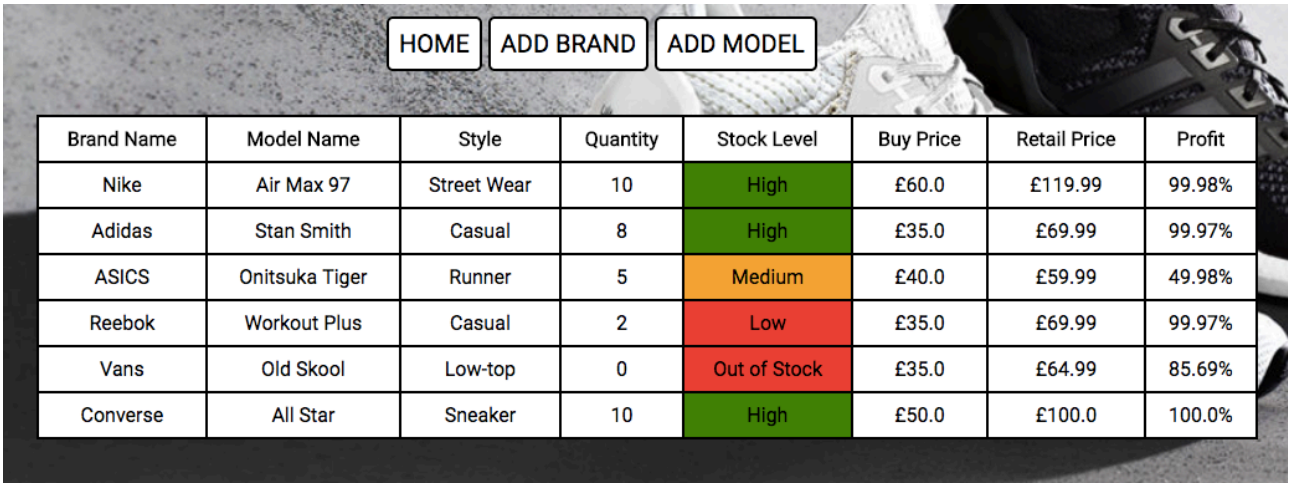
2.



P. 13 User input

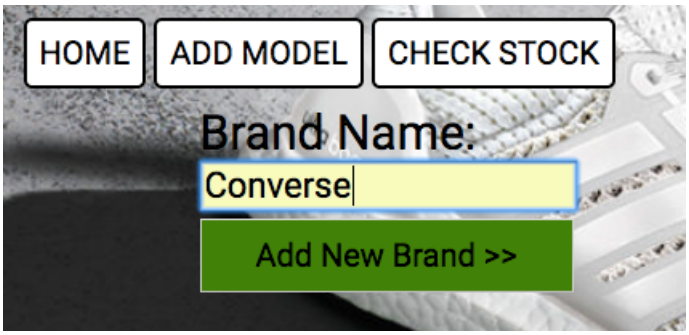


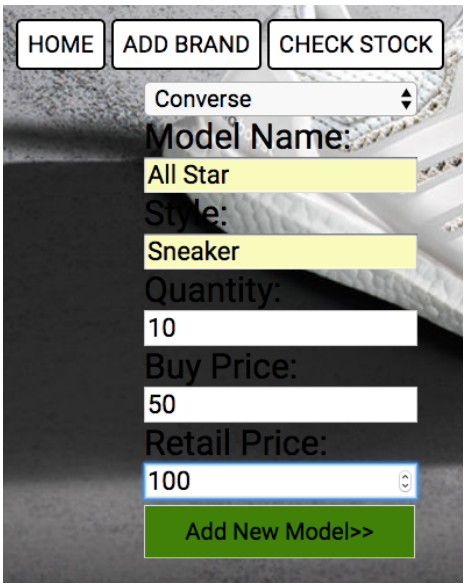




Brand Name	Model Name	Style	Quantity	Stock Level	Buy Price	Retail Price	Profit
Nike	Air Max 97	Street Wear	10	High	£60.0	£119.99	99.98%
Adidas	Stan Smith	Casual	8	High	£35.0	£69.99	99.97%
ASICS	Onitsuka Tiger	Runner	5	Medium	£40.0	£59.99	49.98%
Reebok	Workout Plus	Casual	2	Low	£35.0	£69.99	99.97%
Vans	Old Skool	Low-top	0	Out of Stock	£35.0	£64.99	85.69%
Converse	All Star	Sneaker	10	High	£50.0	£100.0	100.0%

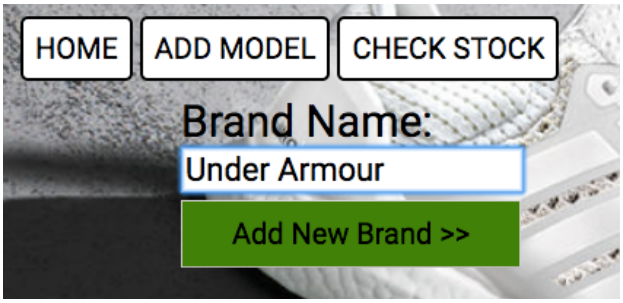
P. 14 Interaction with data persistence

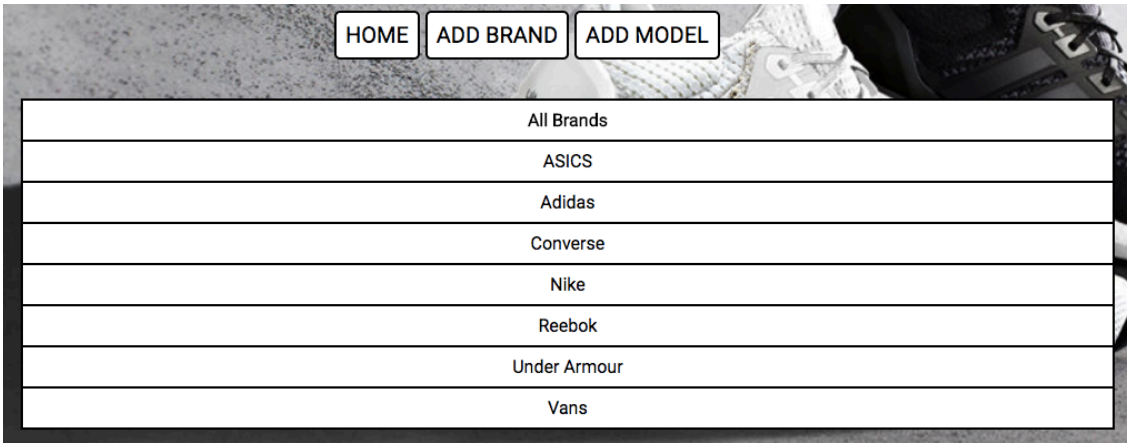




id	model_name	brand_id	quantity	style	buy_price	sell_price
1	Air Max 97	1	10	Street Wear	60	119.99
2	Stan Smith	2	8	Casual	35	69.99
3	Onitsuka Tiger	3	5	Runner	40	59.99
4	Workout Plus	4	2	Casual	35	69.99
5	Old Skool	5	0	Low-top	35	64.99
6	All Star	6	10	Sneaker	50	100

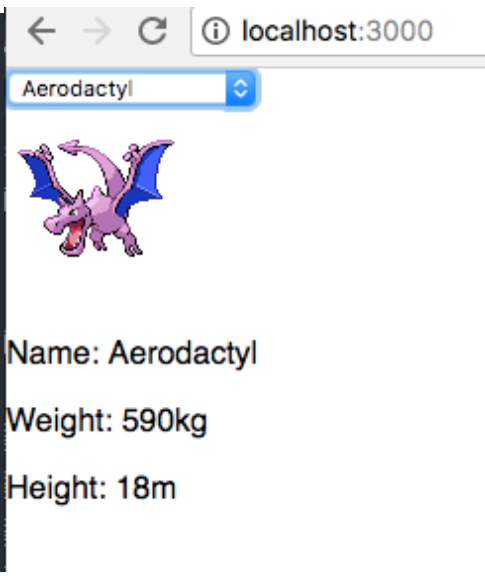
P. 15 User output result





P. 16 API being used within a program.

```
onPokemonSelect(index) {
  index++
  let url = 'https://pokeapi.co/api/v2/pokemon/' + index;
  let request = new XMLHttpRequest();
  request.open('GET', url);
  request.addEventListener('load', () => {
    if(request.status !==200) return;
    let pokemonDeets = JSON.parse(request.responseText);
    this.setState({
      selectedPokemon: pokemonDeets
    })
  })
  request.send();
}
```



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

Limited number of videos and images displayed in search.	FAILED	Added a limit to request so that only a certain number are returned.	PASSED
User must be able to hit the enter key on the search box.	FAILED	Enter key code from the event so that could trigger the search as well.	PASSED
User is able to hear space sounds.	FAILED	Added NASA sound API support and sound was able to play	PASSED
User must be able to store video in favourites	FAILED	Created a database to be able to have data persistence.	PASSED

P. 18 Testing your program.

Test not passing:

```
import ...


public class BedroomTest {

    Bedroom bedroom;

    @Before
    public void before() { bedroom = new Bedroom( capacity: 2, roomNumber: 23, RoomType.DOUBLE, RoomValue.MEDIUM); }

    @Test
    public void getRoomNumber() { assertEquals( expected: 23, bedroom.getRoomNumber()); }

    @Test
    public void getRoomType() { assertEquals(RoomType.DOUBLE, bedroom.getRoomType()); }

    @Test
    public void setRoomType(){
        bedroom.setRoomType(RoomType.SINGLE);
        assertEquals(RoomType.SINGLE, bedroom.getRoomType());
    }

    @Test
    public void doubleHasValue200() { assertEquals( expected: 200, bedroom.getValueFromEnum()); }

}
```

```
BedroomTest › getRoomNumber()

4 tests done: 1 failed - 25ms
```

Test Passing:

```java
import ...


public class BedroomTest {

    Bedroom bedroom;

    @Before
    public void before() { bedroom = new Bedroom( capacity: 2, roomNumber: 23, RoomType.DOUBLE, RoomValue.MEDIUM); }

    @Test
    public void getRoomNumber() { assertEquals( expected: 22, bedroom.getRoomNumber()); }

    @Test
    public void getRoomType() { assertEquals(RoomType.DOUBLE, bedroom.getRoomType()); }

    @Test
    public void setRoomType(){
        bedroom.setRoomType(RoomType.SINGLE);
        assertEquals(RoomType.SINGLE, bedroom.getRoomType());
    }

    @Test
    public void doubleHasValue200() { assertEquals( expected: 200, bedroom.getValueFromEnum()); }

}
```