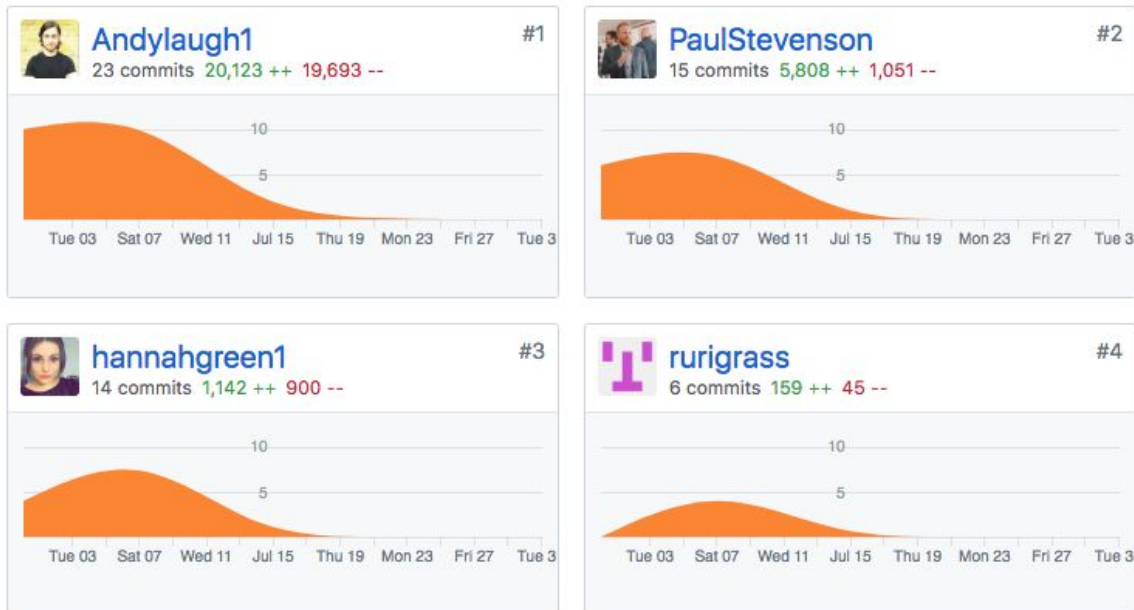


Evidence for Project Unit

Ruairidh Grass

E-21

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
- Explore the Solar System - navigate through planets and display information
- Interactive map of a historical event - e.g. World War 1, the travels of Christopher Columbus

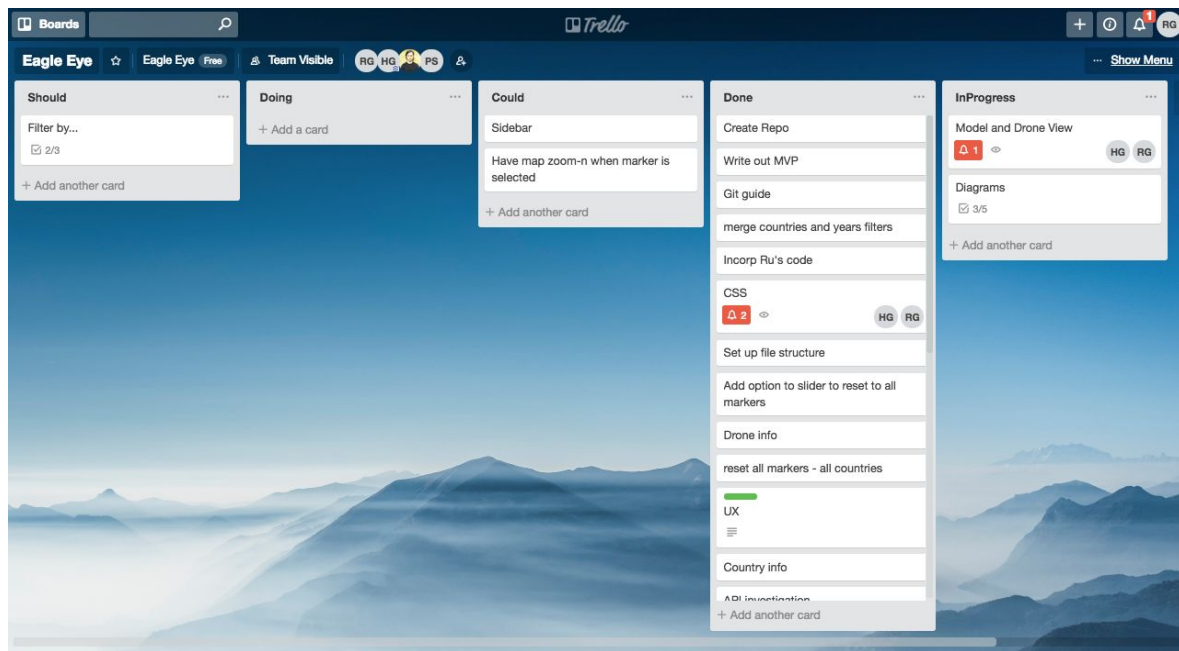
MVP

- Display some information about a particular topic in an interesting way
- Have some interactivity that enables a user to move through different sections of content

Examples of further features

- Bring in data using an API or create your own
- Use charts or maps to display your information

P. 3 Use of Trello

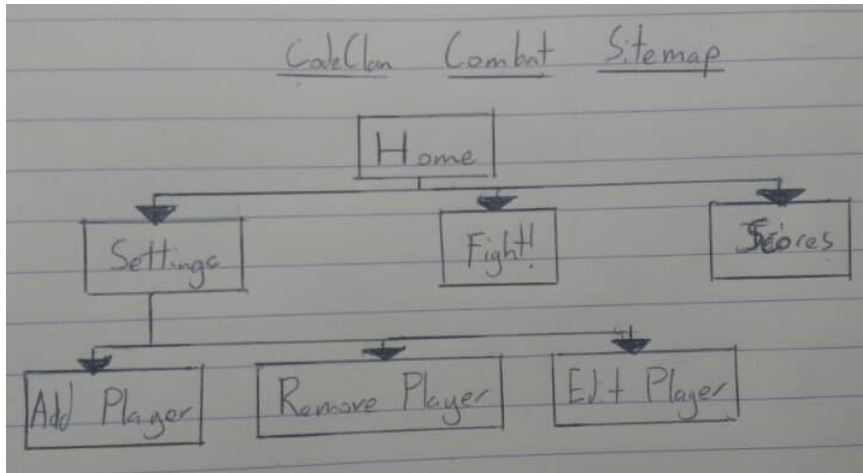


P. 4 Acceptance Criteria

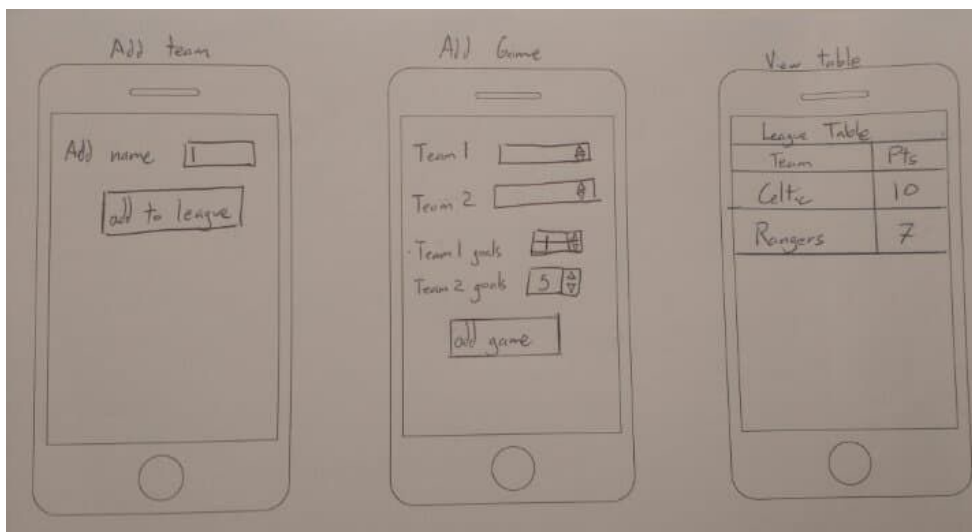
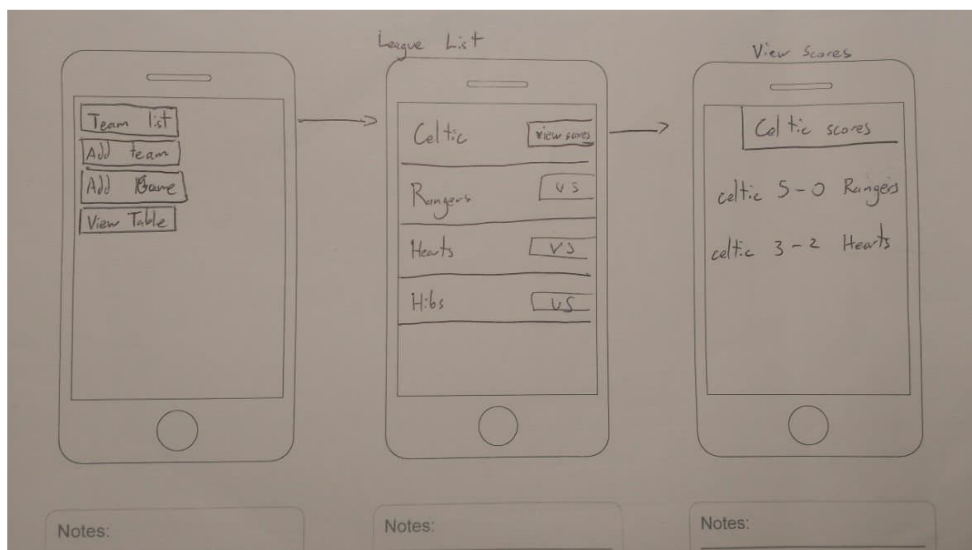
Acceptance Criteria	Expected Outcome	Pass/Fail
The user can click on a pin on the map.	The sidebar will display the information about the selected pin.	Pass
The user can select a specific country from the dropdown menu.	The map will zoom in on the selected country.	Pass
The user can move the slider and select a year.	Only the pins from the selected year will display on the map.	Pass
Reset map button can be clicked.	The map display will reload with all pins showing and the general information will show on the sidebar.	Pass

P. 5 User sitemap

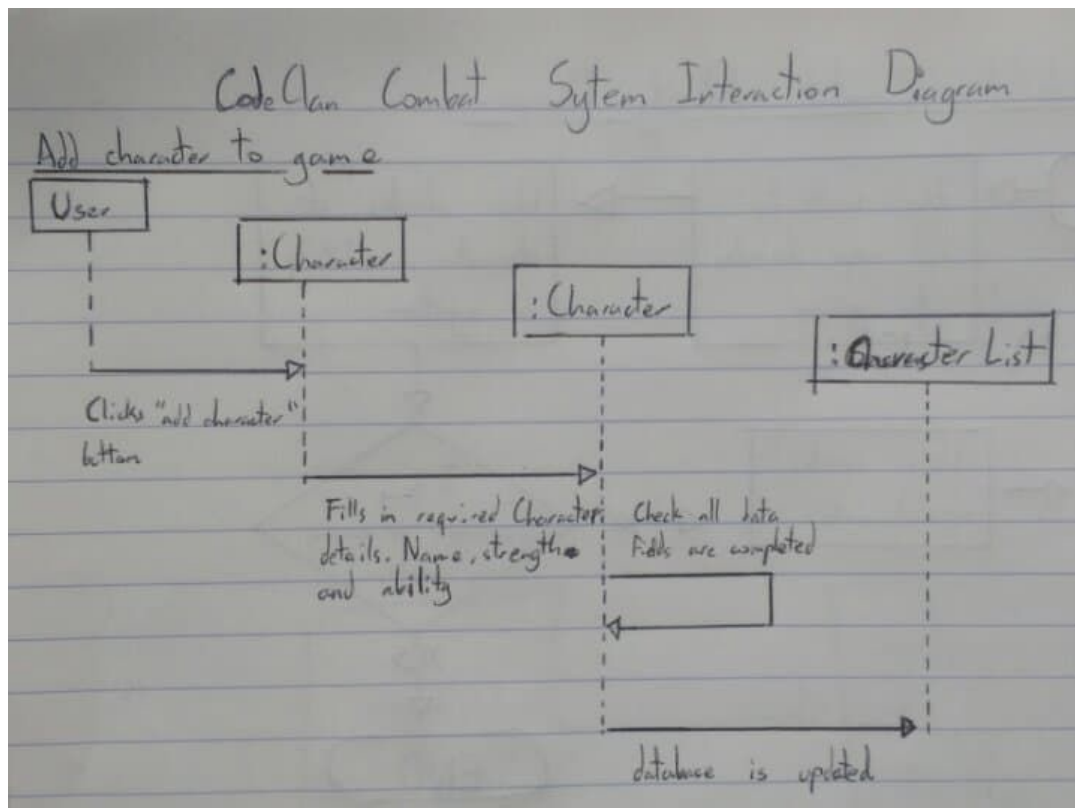
For CodeClan Combat App:



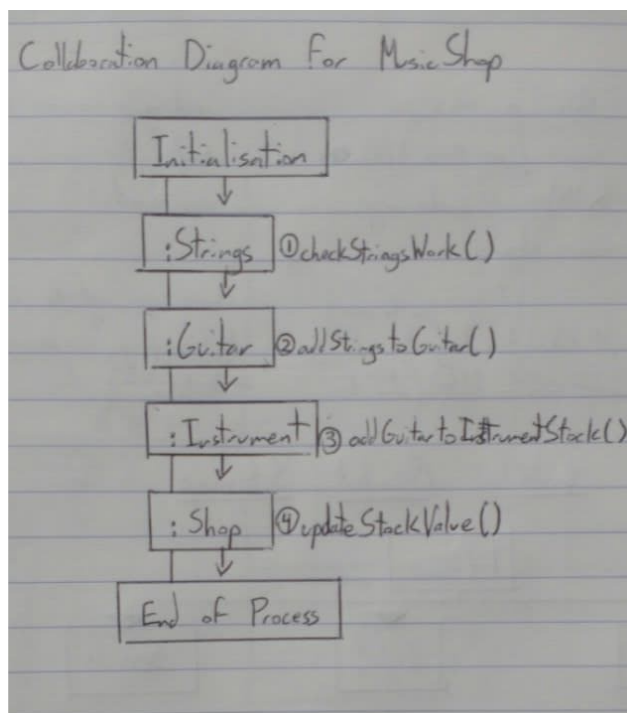
P. 6 Wireframes designs



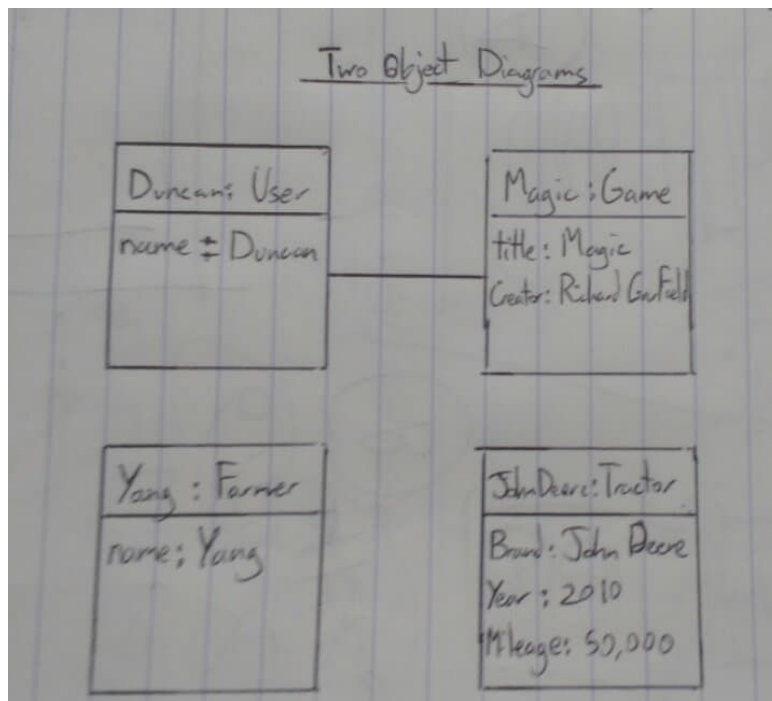
P. 7 System interactions diagrams



Collaboration diagram showing the interaction between the Strings, Guitar, Instrument and Shop classes. First we check if the Strings work, we then add them to the Guitar which is then added to the Instrument stock which then finally updates the total value of the Shop's assets.



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.)

```
49 //EDUCATION
50
51 educationData() {
52   return this.props.educationArray.filter(c => !isNaN(c) );
53 }
54
55 totalEducation() {
56   const eduData = this.educationData();
57   const result = eduData.reduce( (accumulator, currentValue) => accumulator + currentValue , 0);
58   return result;
59 }
60
61 averageEducation() {
62   const average = this.totalEducation() / this.educationData().length;
63   return average
64 }
```

1. EducationData() will loop through an array and count the objects within it that are numbers.
2. totalEducation() will then loop through this same array and count all the numbers within it on a accumulator.
3. averageEducation() will then take the number from totalEducation() and divide it into the number from totalEducation().

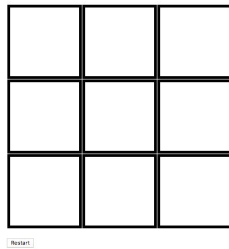
On this example please take a screenshot and write what it is doing and why you decided to use it.

P. 10 Example of Pseudocode

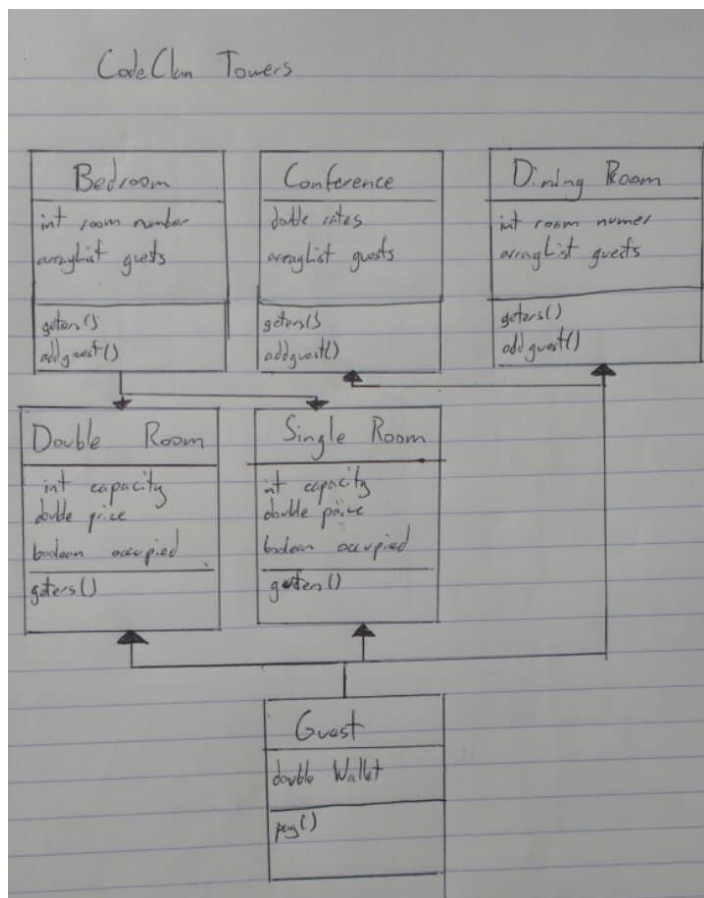
Create a function that will check the animal's hunger level int
If hunger level int is below ≤ 5 make animal return
a "I'm hungry!" string else return "I'm not hungry".

P. 11 Github link to one of your projects

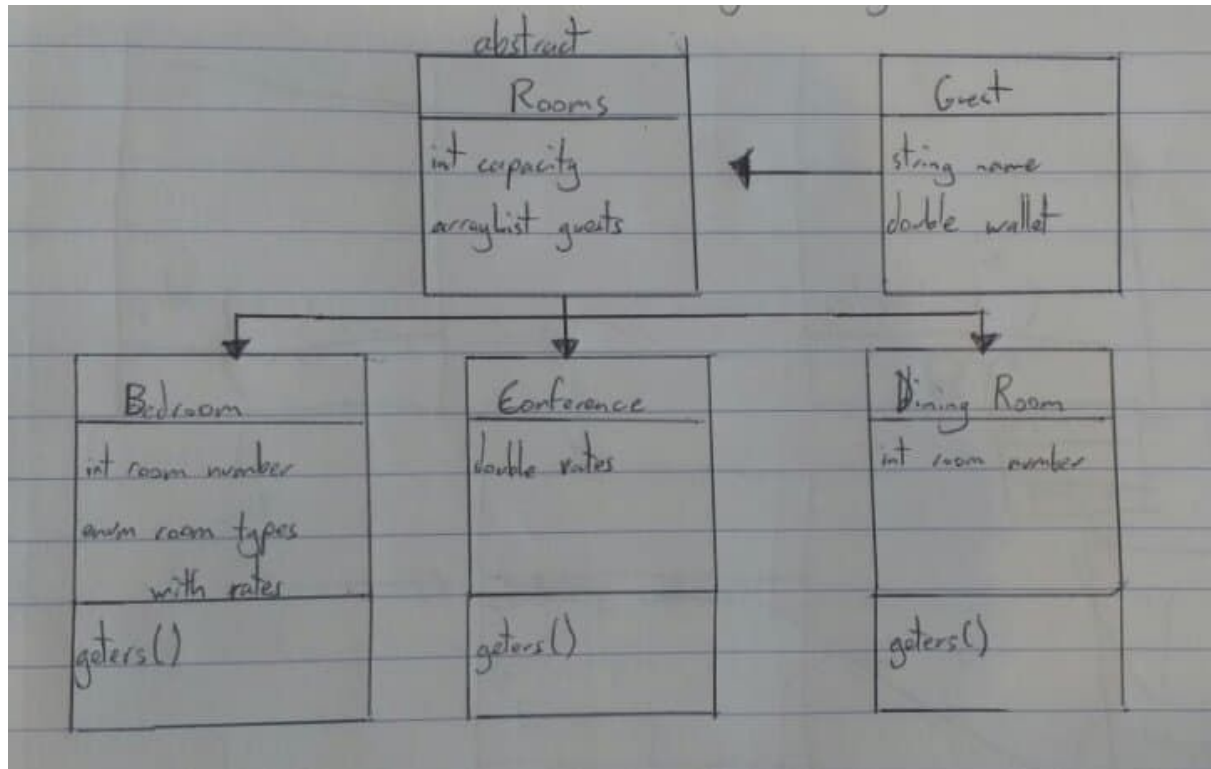
<https://github.com/rurigrass/RockPaperScissors>



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

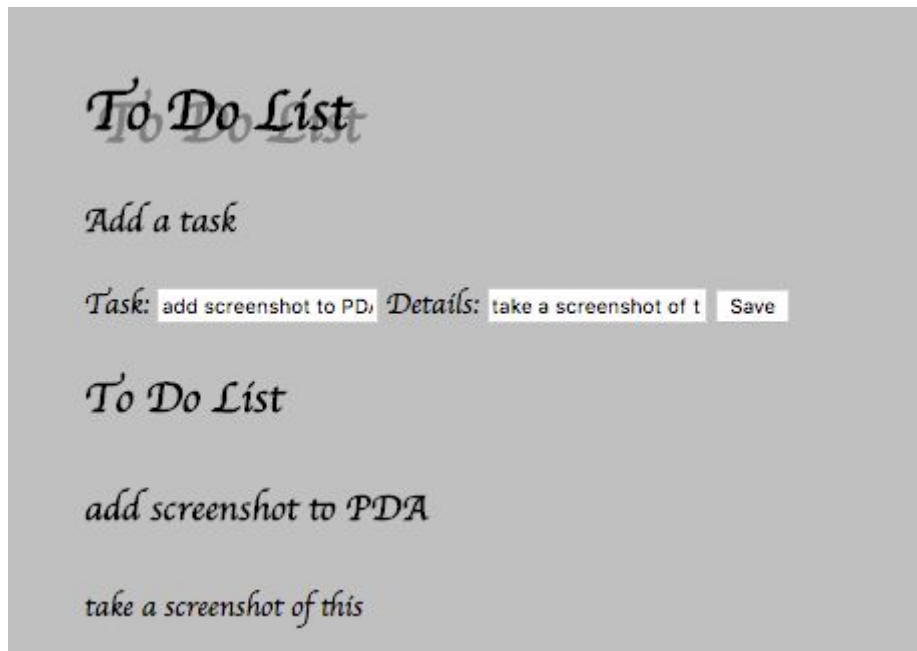


Initial planning stage was too messy and disorganised. I decided to put all 3 room types into a Room abstract parent class and to replace the single and double rooms with enums.



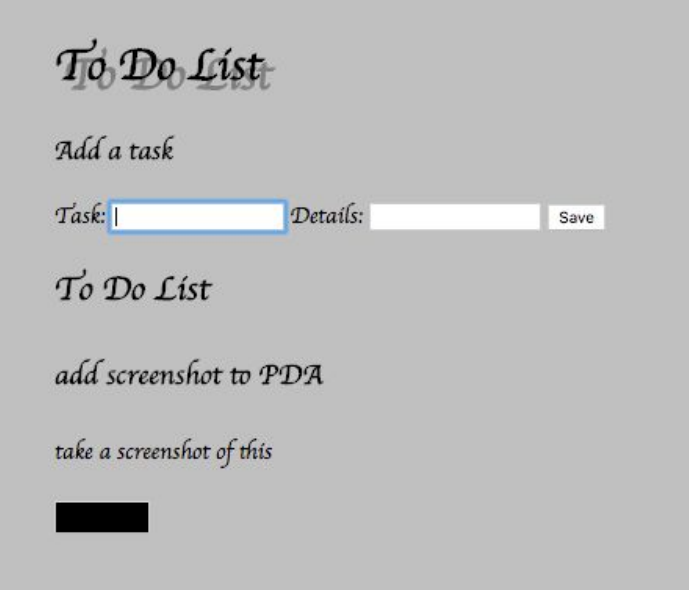
Final Plan showing final class relationships.

P. 13 User input



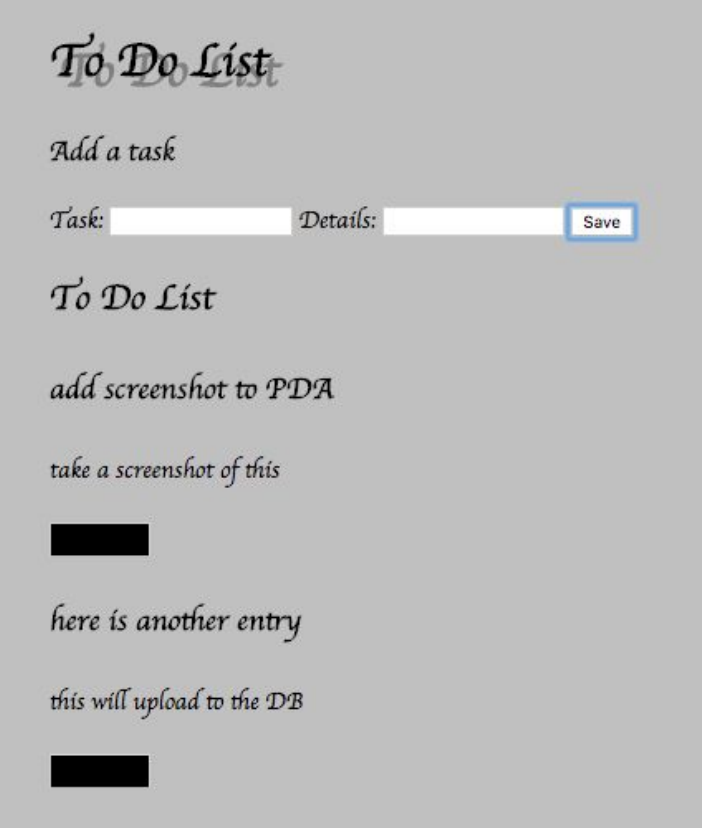
P. 14 Interaction with data persistence

User inputs new Task to ToDo list.



The screenshot shows the 'To Do List' application interface. At the top, the title 'To Do List' is displayed in a stylized font. Below the title, there is a section labeled 'Add a task'. This section contains a form with two input fields: 'Task:' and 'Details:'. The 'Task:' field is currently active, indicated by a blue border. To the right of the 'Details:' field is a 'Save' button. Below the form, the title 'To Do List' is repeated. Underneath, the text 'add screenshot to PDA' is shown, followed by 'take a screenshot of this' and a black rectangular placeholder.

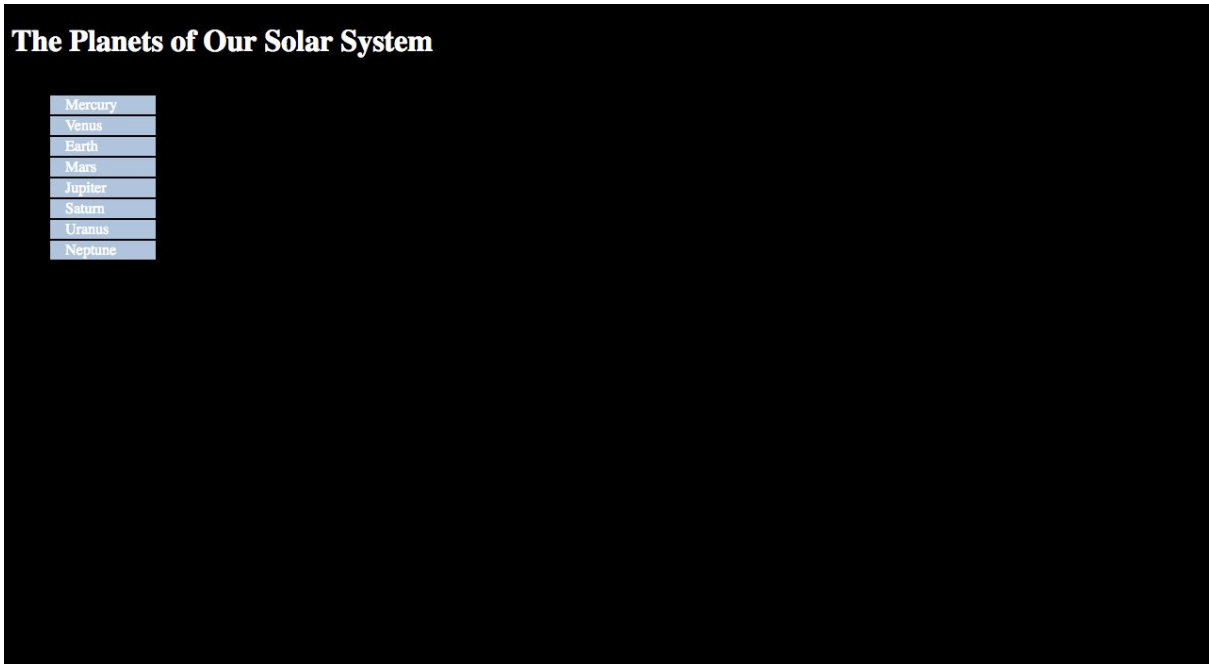
New entry added to list. This will remain here even if the page is reloaded as it has been uploaded to the DB.



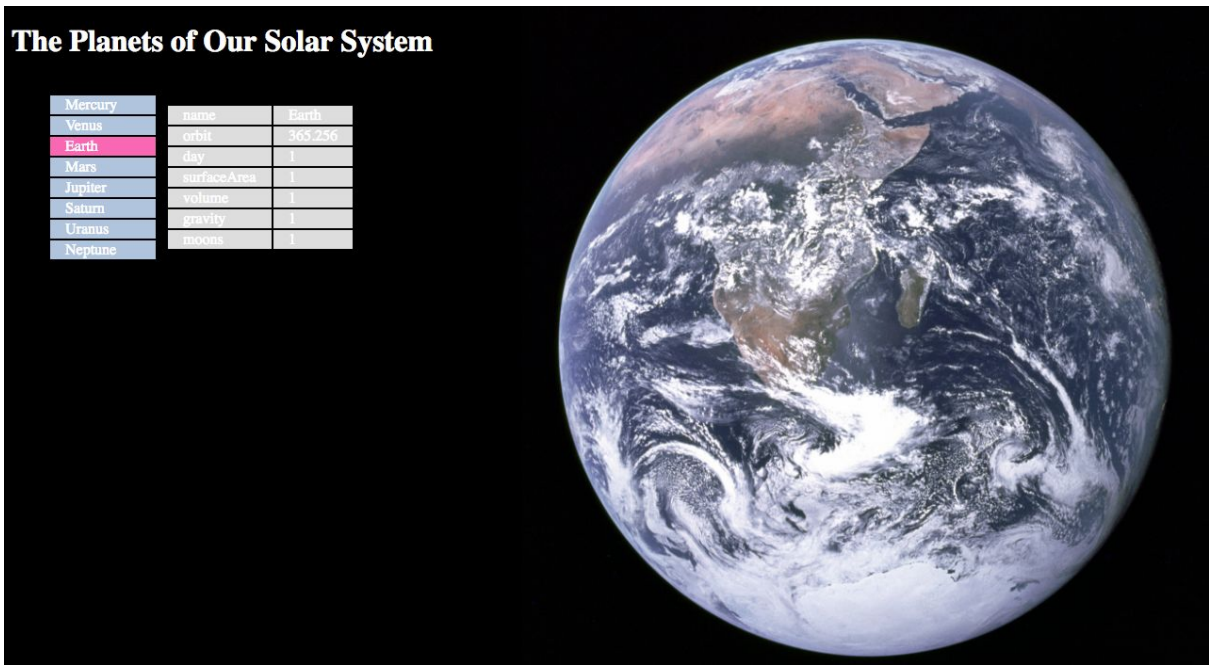
The screenshot shows the 'To Do List' application interface after a new entry has been added. The title 'To Do List' is at the top. Below it is the 'Add a task' section with the 'Task:' and 'Details:' input fields and a 'Save' button. The 'Save' button is now highlighted with a blue border. Below the form, the title 'To Do List' is repeated. Underneath, the text 'add screenshot to PDA' is shown, followed by 'take a screenshot of this' and a black rectangular placeholder. Below this, the text 'here is another entry' is shown, followed by 'this will upload to the DB' and another black rectangular placeholder.

P. 15 User output result

The user Selects a Planet from the planet list



User clicks on 'Earth' button. Request is processed and information is presented in program.



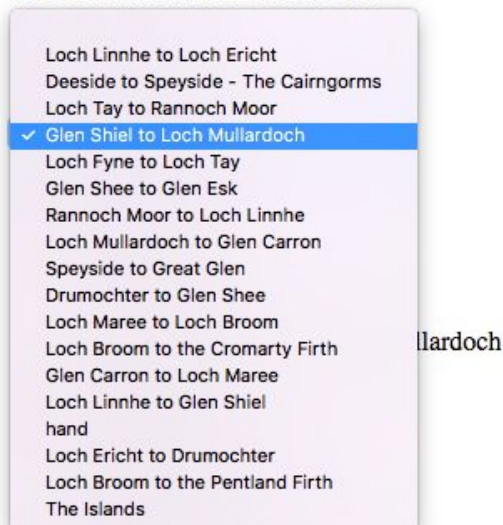
P. 16 Show an API being used in your program

App.js has an event listener that requests the data from the API when the app is loaded

```
13
14 Munros.prototype.getData = function () {
15   const request = new RequestHelper(this.url);
16   const handleRequestComplete = (responseData) => {
17     this.munros = responseData;
18     const uniqueRegions = this.getUniqueRegions();
19     PubSub.publish('Munros:munros-data-ready', this.munros);
20     PubSub.publish('Munros:munros-regions-ready', uniqueRegions);
21   };
22   request.get()
23     .then(handleRequestComplete)
24     .catch(error => console.error(error));
25 }
```

Here you can see the Munro API data is displayed when a Munro is selected

Scottish Munros



- Region: Glen Shiel to Loch Mullardoch

Sgurr nan Ceathreamhnan

- Meaning: Peak of the quarters
- Height: 1151
- Region: Glen Shiel to Loch Mullardoch

A' Chralaig

- Meaning: The basket or creel
- Height: 1120
- Region: Glen Shiel to Loch Mullardoch

Tom a' Choinich

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

Activity	Status	Action Taken	Pass/Fail
Character must have name	Failed	New character needed to be assigned a unique ID	Passed
Character's strength and ability must be greater or equal to 0 and below 11	Failed	Changed minimum and maximum ability and strength to be between 0 and 10	Passed
Character must have name	Failed	I made this field not Null in the SQL table	Passed
Must ask for confirmation before deleting content	Failed	Set delete all to require a confirmation checkbox	Passed

P. 18 Testing your program

Test code:

```
1  const assert = require('assert');
2  const Dinosaur = require('../models/dinosaur.js');
3
4  describe('Dinosaur', function() {
5
6      let dinosaur;
7
8      beforeEach(function () {
9          dinosaur = new Dinosaur('t-rex', 'carnivore', 50);
10     });
11
12     it('should have a species', function () {
13         const actual = dinosaur.species;
14         assert.strictEqual(actual, 't-rex');
15     });
16
17     it('should have a diet', function () {
18         const actual = dinosaur.diet;
19         assert.strictEqual(actual, 'carnivore');
20     });
21
22     it('should have an average number of visitors it attracts per day', function () {
23         const actual = dinosaur.guestsAttractedPerDay;
24         assert.strictEqual(actual, 50);
25     });
26
27 });
```

Failed test:

```
[→ hw_tdd_jurassic_park git:(master) ✖ npm run test ]

> hw_tdd_jurassic_park@1.0.0 test /Users/ruairidhgrass/codeclan_work/week_11/day_2/hw_tdd_jurassic_park
> mocha specs

Dinosaur
  1) should have a species
  2) should have a diet
  ✓ should have an average number of visitors it attracts per day

Park
  ✓ should have a name
  ✓ should have a ticket price
  ✓ should have a collection of dinosaurs
  ✓ should be able to add a dinosaur to its collection
  ✓ should be able to remove a dinosaur from its collection
  - should be able to find the dinosaur that attracts the most visitors
  - should be able to remove all dinosaurs of a particular species

6 passing (15ms)
2 pending
2 failing

1) Dinosaur
   should have a species:

    AssertionError [ERR_ASSERTION]: 't-rex' === 't-rox'
    + expected - actual

    -t-rex
    +t-rox

    at Context.<anonymous> (specs/dinosaur_spec.js:14:12)

2) Dinosaur
   should have a diet:

    TypeError: assert.Equal is not a function
    at Context.<anonymous> (specs/dinosaur_spec.js:19:12)
```

Passing Test:

```
[→ hw_tdd_jurassic_park git:(master) ✖ npm run test ]

> hw_tdd_jurassic_park@1.0.0 test /Users/ruairidhgrass/codeclan_work/week_11/day_2/hw_tdd_jurassic_park
> mocha specs

Dinosaur
  ✓ should have a species
  ✓ should have a diet
  ✓ should have an average number of visitors it attracts per day

Park
  ✓ should have a name
  ✓ should have a ticket price
  ✓ should have a collection of dinosaurs
  ✓ should be able to add a dinosaur to its collection
  ✓ should be able to remove a dinosaur from its collection
  - should be able to find the dinosaur that attracts the most visitors
  - should be able to remove all dinosaurs of a particular species

8 passing (13ms)
2 pending
```

```
npm update check failed
Try running with sudo or get access
to the local update config store via
sudo chown -R $USER:$USER /Users/ruairidhgrass/.config
```