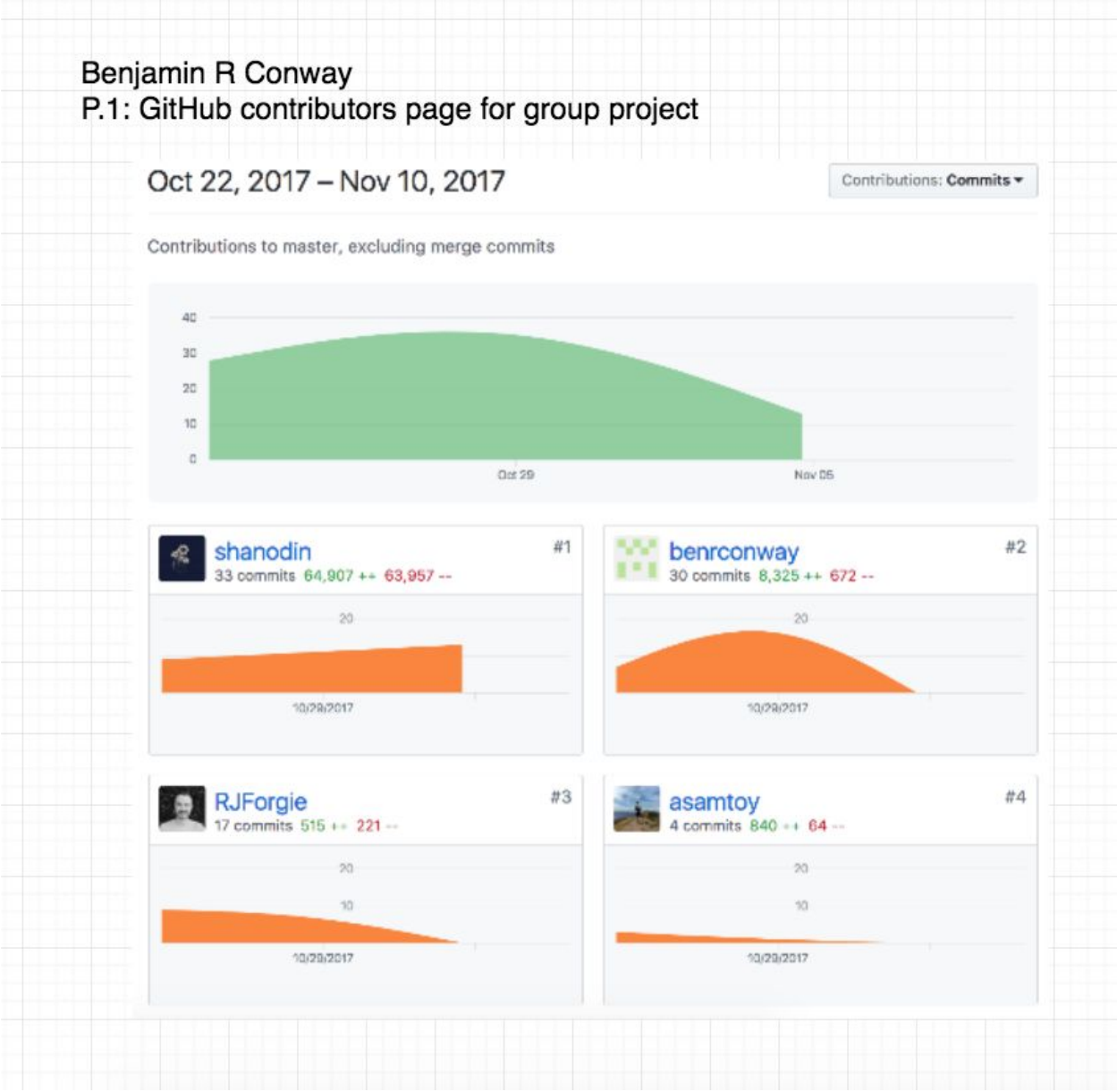
Evidence for Project Unit

Name: Benjamin R Conway
Cohort: E15
Date: 05/10/2017



Benjamin R Conway
P.2: Project Brief of Group Project

Route Planner

Visit Scotland are look for ways to encourage people to walk and cycle. Your task is to create an app that allows users to search for cycling and hiking routes, view routes on a map, save routes to a wishlist and mark a route done.

You could use GoogleMaps Directions API:

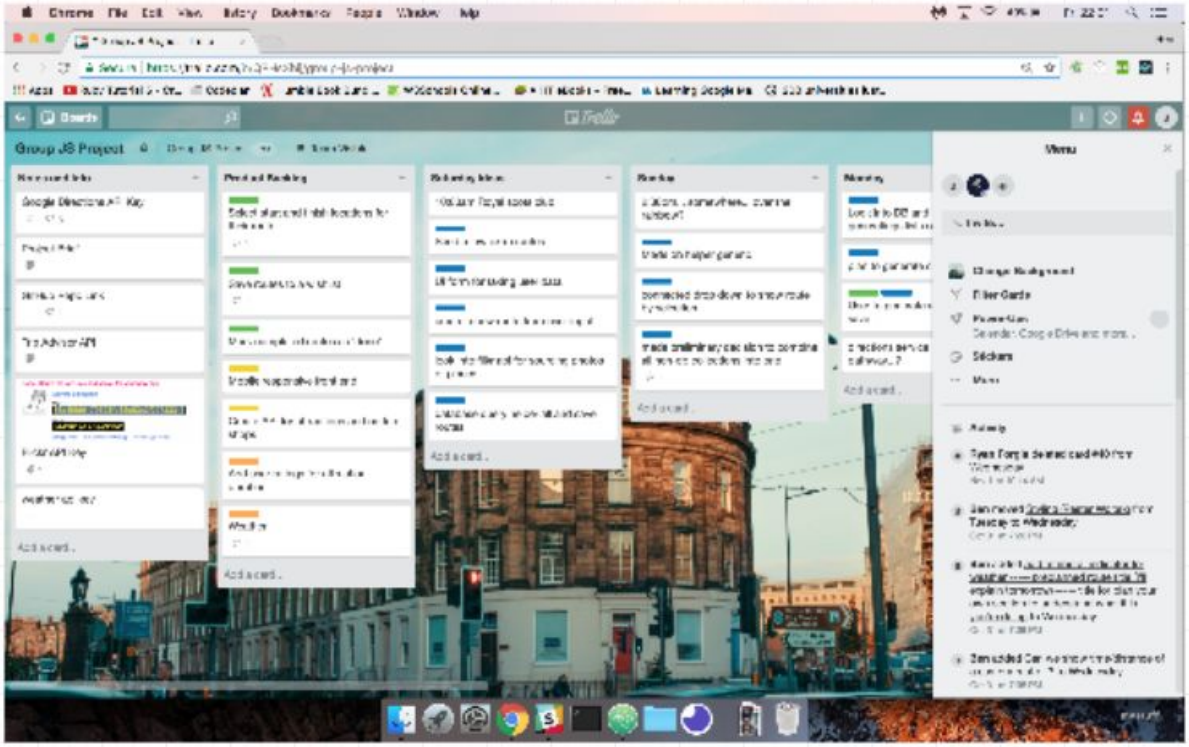
- https://developers.google.com/maps/documentation/directions/

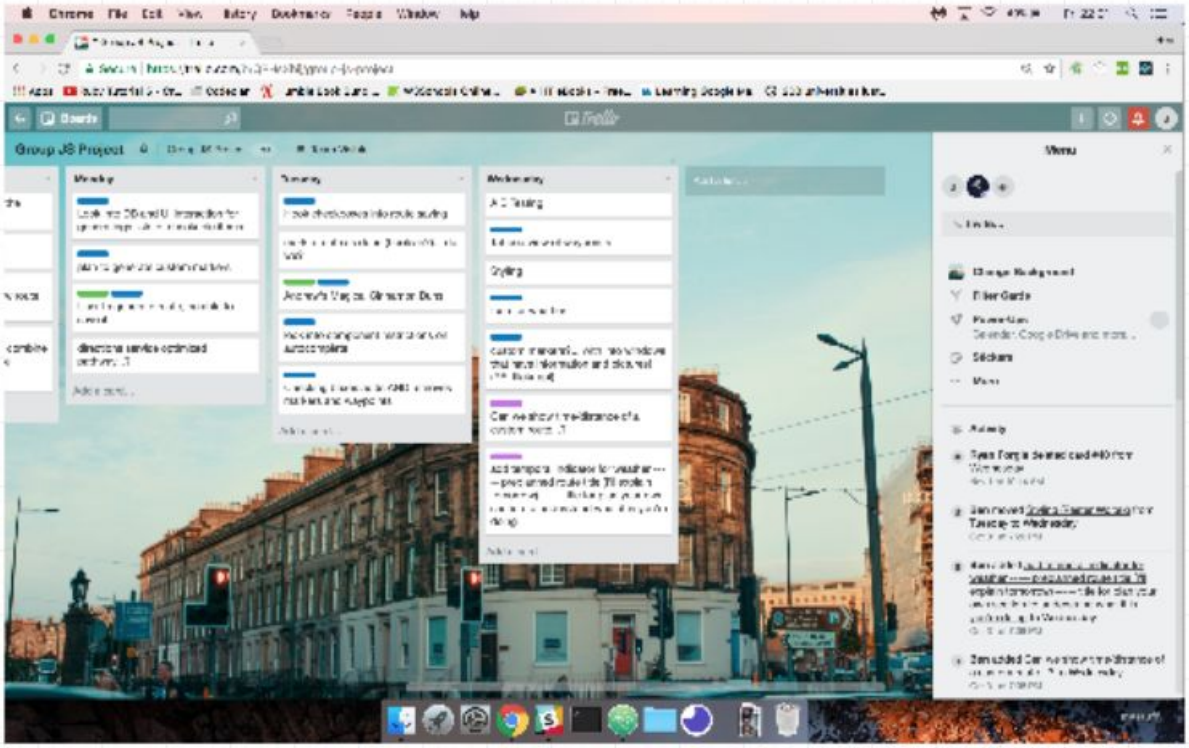
MVP

Users should be able to:

- Select start and finish locations for their route
- Save routes to a wishlist
- Mark completed routes as 'done'

Benjamin R Conway
P.3: Planning completed during group project.

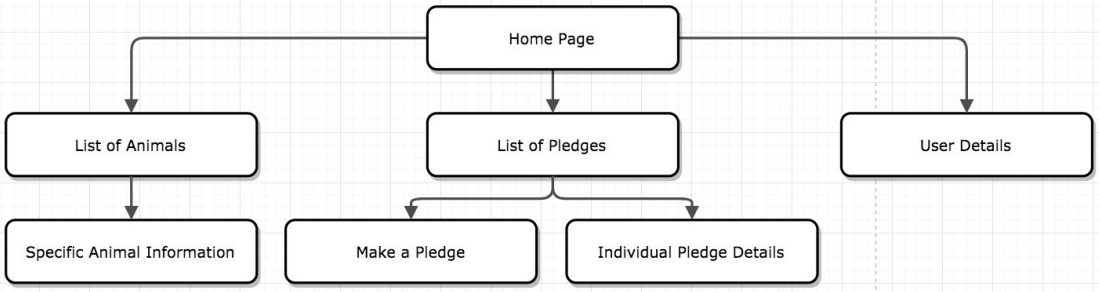




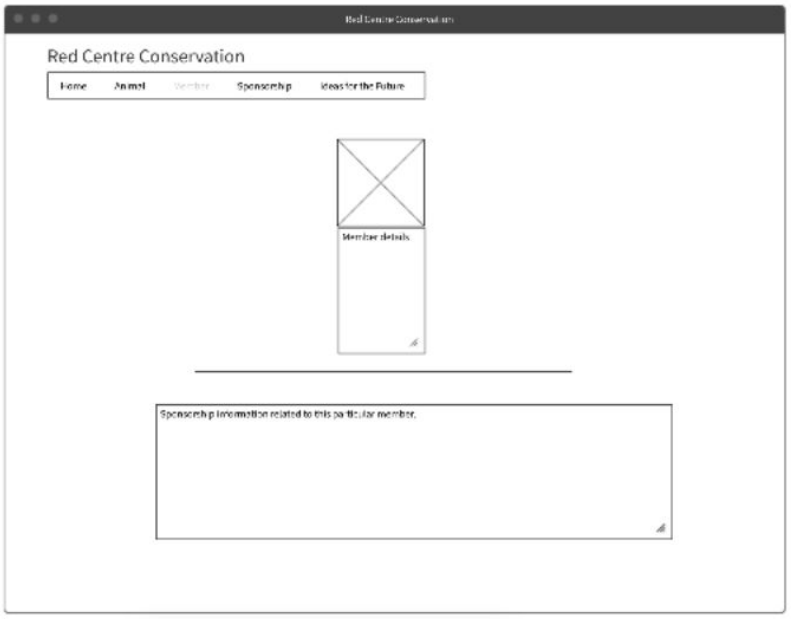
Benjamin R Conway
P.4 Acceptance Critera and Test Plan

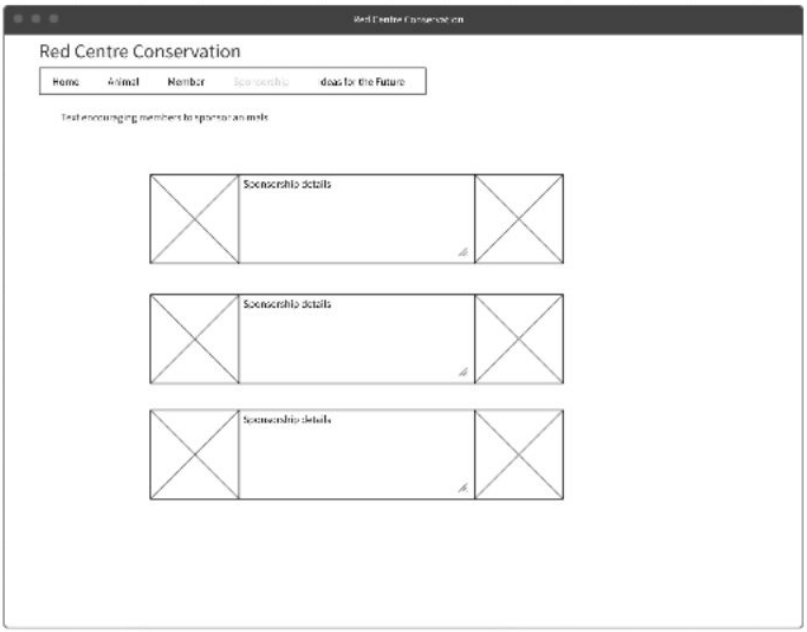
Acceptance Critera	Expected Result / Output	Pass / Fail
User can see a list of saved routes upon opening application	Select drop down populates with saved routes	Pass
User can select a route from routes saved and those given	When a route from the drop down is selected it immediately renders on the map window	Pass
User can set start point	Autocomplete box designates the origin of the route to be entered	Pass
User can nominate waypoints from a curated list of sites around Edinburgh	Middle field of the “Create a route” section allows for multiple waypoint selection	Pass
When a user selects waypoints, markers are added to the map to show how near or far they are from present location	Markers are added to the map window as user selections are made for their custom route	Pass
Users made routes are persisted and made available in the planned route list.	After mapping a route, it is persisted to the database and available through the pre-planned route drop down.	Pass
Routes will have clear markers to designate differing types of location.	Markers have programmatically altered images for different types of waypoints.	Pass

Benjamin R Conway
P 5, A User Sitemap



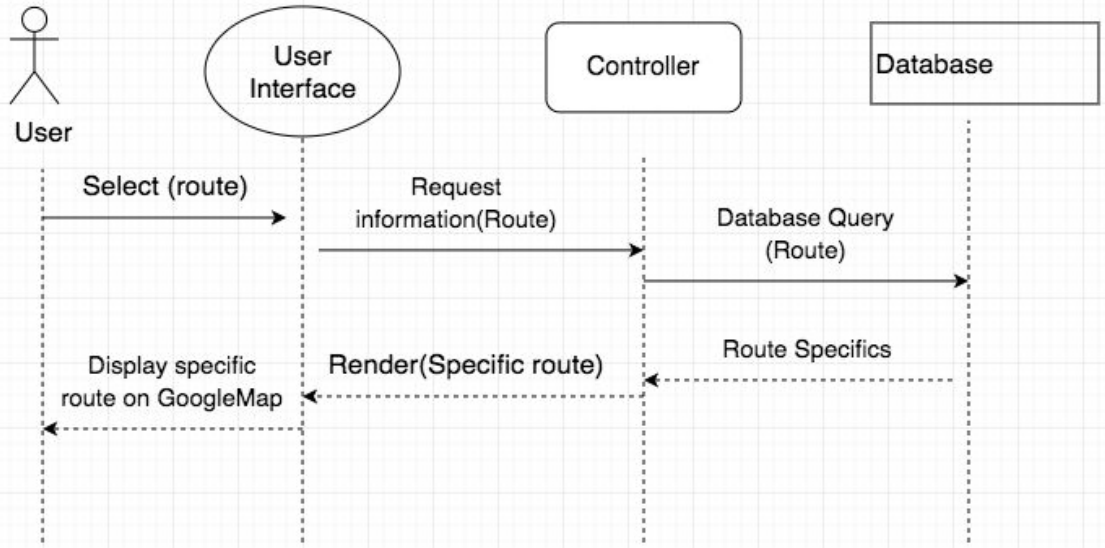
Benjamin R Conway
P. 6. Produce two wireframes



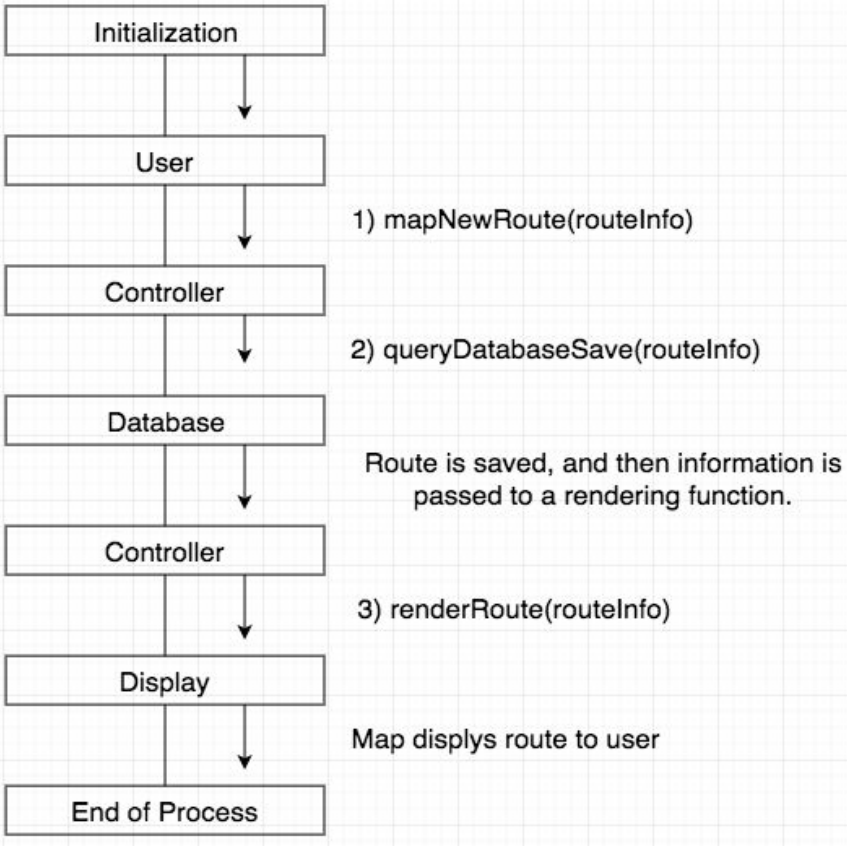


Benjamin R Conway
P.7. Produce two system integration diagrams

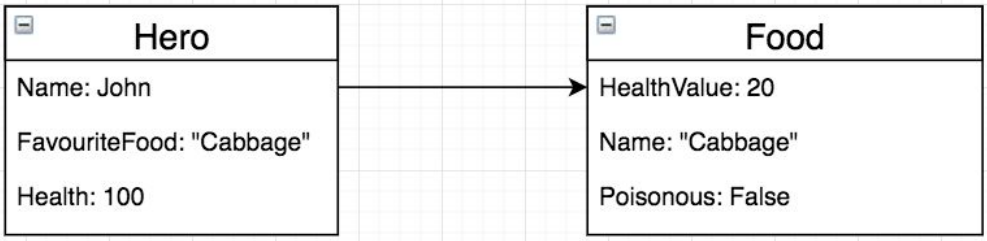
Sequence diagram of choosing a new route



Collaboration diagram of inputting a new route



Benjamin R Conway
P.8 Produce two object diagrams



Benjamin R Conway
P.9 Two Algorithms I have used.

Algorithm 1:

```
Hero.prototype.checkFood = function(food) {
  var healthGain = 0;
  if(!food.poisonous && this.checkIfFavourite(food)) {
      healthGain = food.replenishmentValue * 1.5;
    }
  if(!food.poisonous && !this.checkIfFavourite(food)) {
      healthGain = food.replenishmentValue;
    }
  if(food.poisonous){
      healthGain -= food.replenishmentValue / 2;
  }
  return healthGain;
};
```

I wrote and used this algorithm to meet a requirement that if a hero was to eat food they would gain health. If they ate their favourite, they would get an increased benefit and if it were poisoned they would have lesser returns.

The algorithm is on the Hero class and takes in a food object. The food object has a properties of type(a string) and poisonous (a boolean). The algorithm then checks the state of the food against a property on the Hero class that tells whether or not it is their favourite and renders the appropriate health gain.

Algorithm 2:

```
private boolean areAnimalsCompatible(ArrayList<Animal> animalsToBeChecked){
    boolean areCompatible = false;
    ArrayList<Animal> carnivores = new ArrayList<>();
    ArrayList<Animal> others = new ArrayList<>();
    for (Animal animal: animalsToBeChecked){
        if (animal instanceof Carnivore){
            carnivores.add(animal);
        }else{ others.add(animal);}
    }
    if((others.size() == 0) || (carnivores.size() == 0)){
        areCompatible = true;
    }
    return areCompatible;
}
```

The algorithm above is used as part of my program to check and block putting carnivores into enclosures along side herbivores or omnivores.

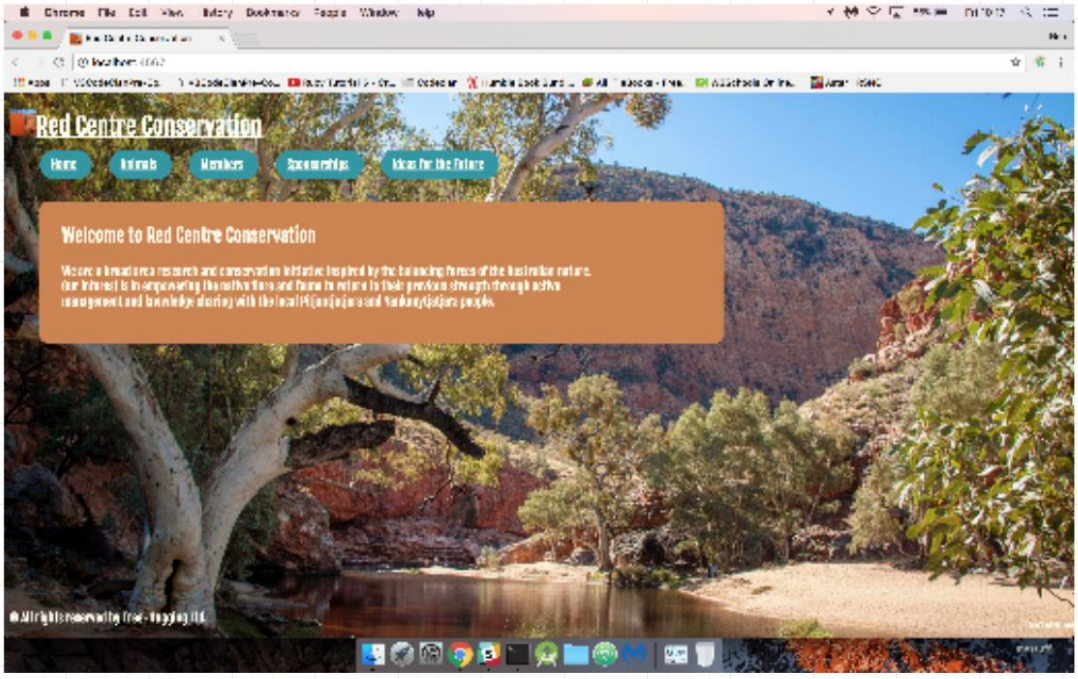
The algorithm takes in an ArrayList of animals, composed of the animals present in the enclosure and those you wish to add. It then iterates through and puts them into the sub-arraylists of carnivores and other(being herbivore and omnivore). It will then return a boolean of compatibility based on presence of non-compatible animals.

P.10.

```
Pseudocode for a function

def sponsorship_searching_function(sponsorship_to_be_searched)
  search the sponsorship for member id.
  member id will inform an SQL query.
  SQL query will return a hash of member details.
  the hash will be inserted into a new member object.
  the new member object will be returned to where this function is called.
end
```

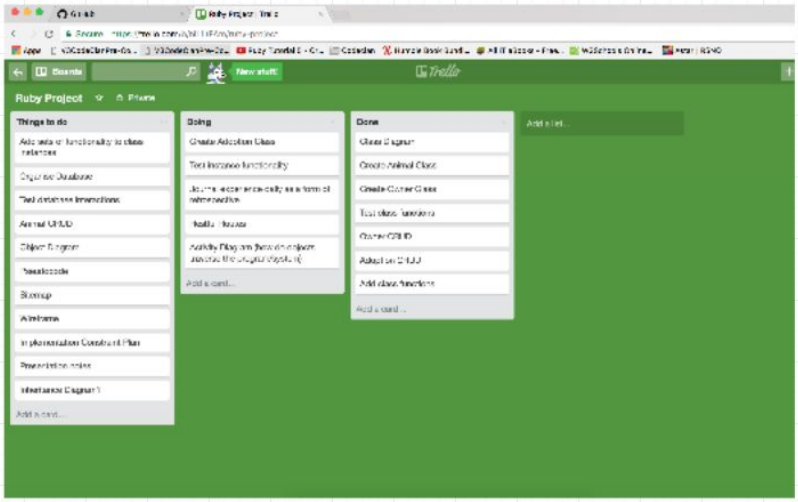
Benjamin R Conway
P. 11 Screenshot and Github link of a project I worked on alone.

https://github.com/benrconway/Conservation_Website_Ruby_Project

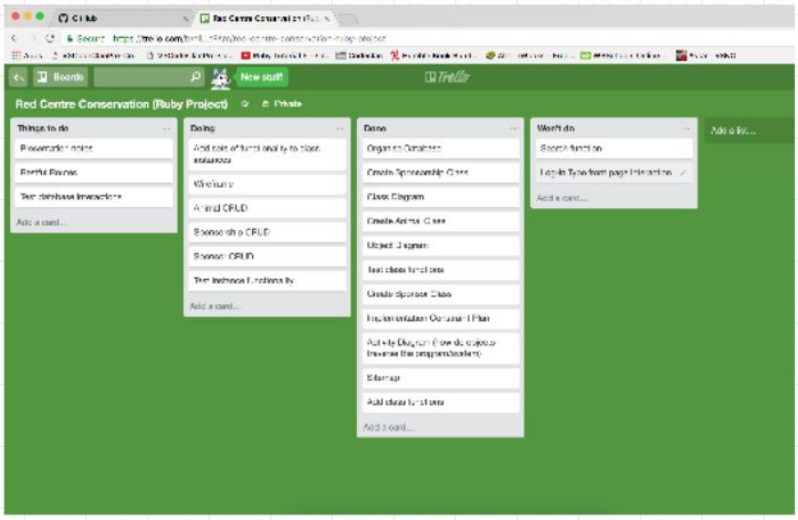


Benjamin R Conway
P. 12 Screenshots of Planning to show changes.

Trello Board of my project in the early stages of implementation.

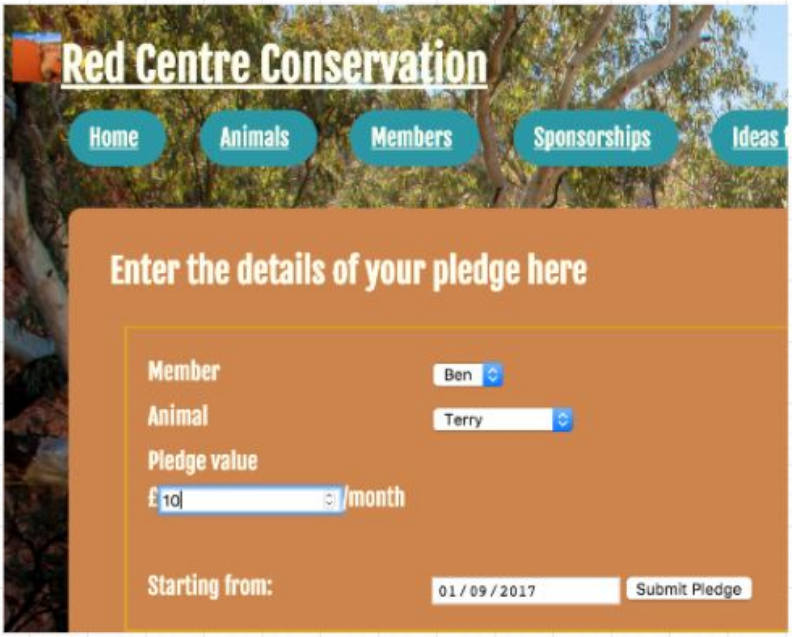


Approximately between 60-70% through my project, changes have been made to what I have done, and will/will not do.



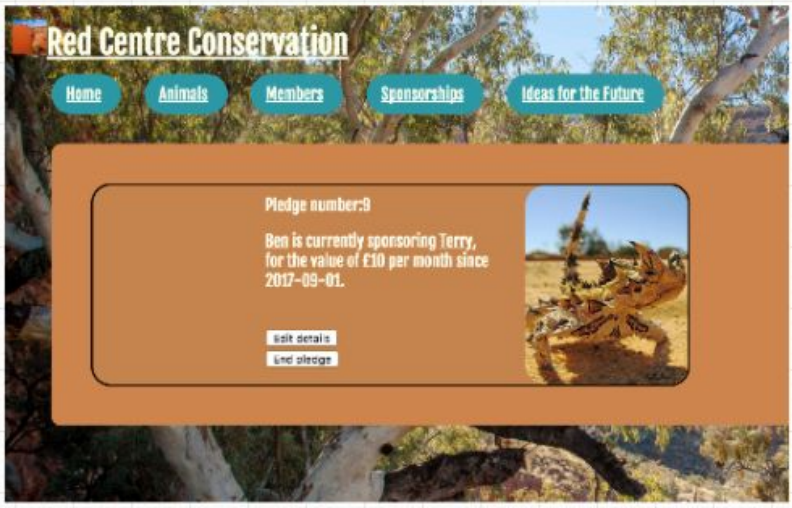
Benjamin R Conway
P. 13 User input being processed according to design requirements

The design brief required that users would be able to make a pledge for specific animal profiles for an amount of their choosing.



In the first image, the member named Ben has input the profile they wish to sponsor and set an amount per month.

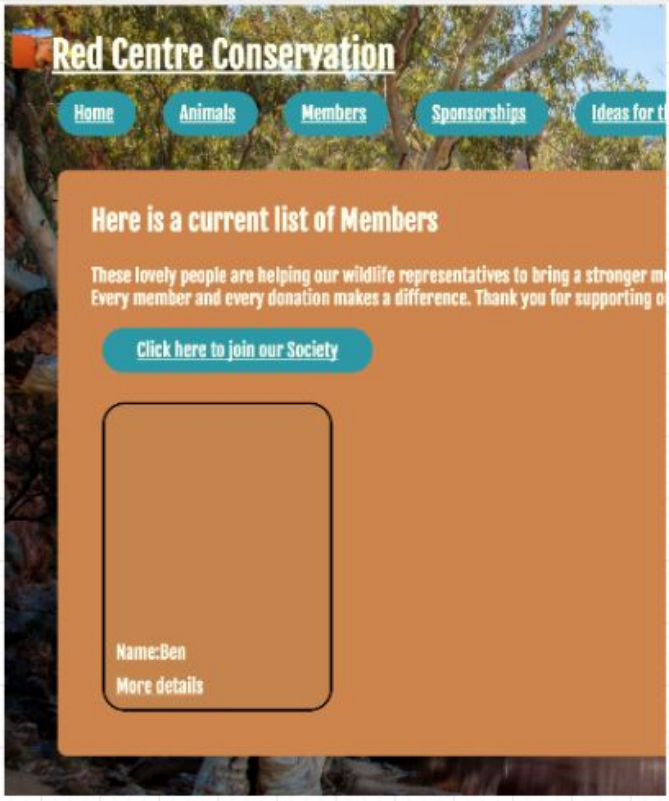
This information has then been saved to a database and is available to be reviewed, edited or deleted.



Benjamin R Conway
P. 14 Show an interaction with data persistence.

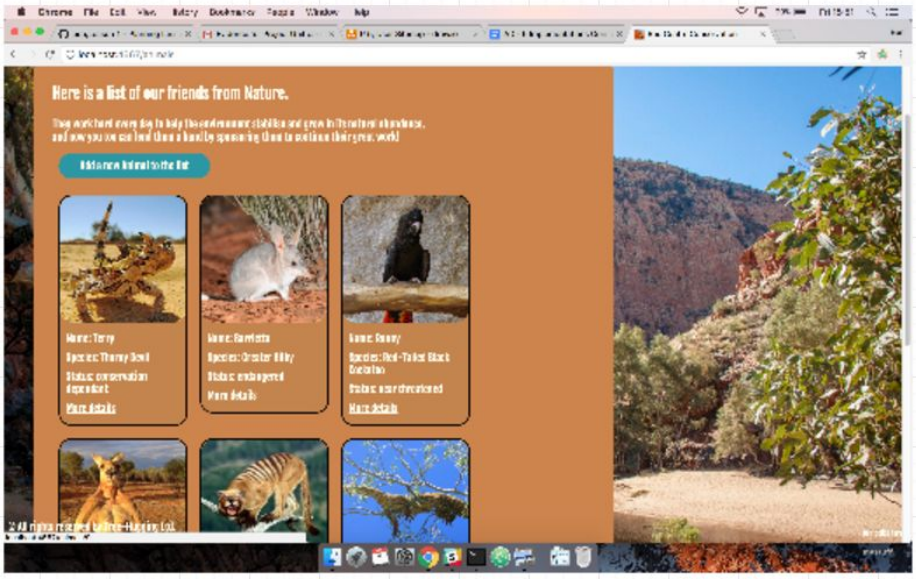
User inputs their name.

The information has now been persisted to a database

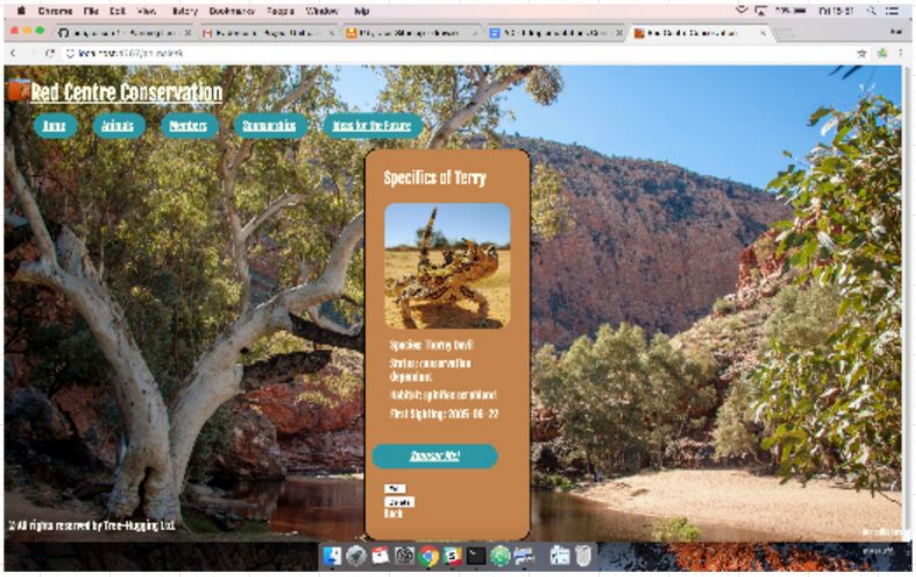


Benjamin Conway
P15 Evidence

Step 1
Click on
"more details"



Result



Benjamin R Conway
P.16: Show an API being used with your program

When the country select from this code is used, it samples the latitude and longitude of a particular country.

```
<script type"text/javascript" src="public/weatherInfo.js"></script>
```

```
var countryRequest = function (countryName, map) {
  var queryUrl = "https://restcountries.eu/rest/v2/name/" + countryName.toLowerCase() + "?fullText=true"
  var request = new XMLHttpRequest();
  request.open("GET", queryUrl);

  request.addEventListener("load", function() {
    var country = JSON.parse(this.responseText);
    displayCountryDetails(country[0]);

    var lat = country[0].latlng[0];
    var long = country[0].latlng[1];
    var area = country[0].area;
    var coords = {lat: lat, lng: long};
    map.addMarker(coords, country[0].name)
    changePosition(countryName, map);
    // map.adjustZoom(area)
    // map.googleMap.setCenter(coords)
    weatherRequest(lat, long);
    save(country[0])
    bordering(country[0].borders)
  })
  request.send();
}
```

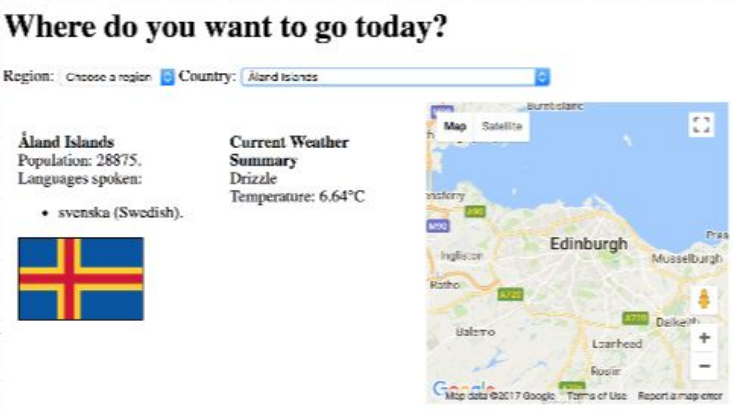
It sends those two figures into this request to Dark Sky weather API for current weather at the location specified.

```
//Weather API- Dark Sky
var displayWeather = function(weather) {
  var parentDiv = document.getElementById("p-secondary");
  while(parentDiv.firstChild){parentDiv.removeChild(parentDiv.firstChild)}
  var details = document.createElement("p");
  details.innerHTML = "<b>Current Weather Summary</b><br>" + weather.currently.summary
  + "<br>Temperature: " + weather.currently.temperature + '&#176;' + "C";
  parentDiv.appendChild(details);
}


var weatherRequest = function(lat,long){
  var url = "https://api.darksky.net/forecast/a048e42340dcb2a4065d8076283123c1/"
  + lat +"," + long + "?&units=si";
  var request = new XMLHttpRequest();

  request.open("GET", url);

  request.addEventListener("load", function() {
    var weather = JSON.parse(this.responseText);
    displayWeather(weather);
  })

  request.send()

}
```

The information is then rendered into a div and placed programmatically into the HTML of the application.

```
<div id="primary" class="macro-container">
  <div id="p-info" class="info-container"></div> <!-- information on primary country -->
  <div id="p-secondary" class="info-container"></div><!-- weather  -->
```

With the weather being displayed between the country selected and a googlemap.

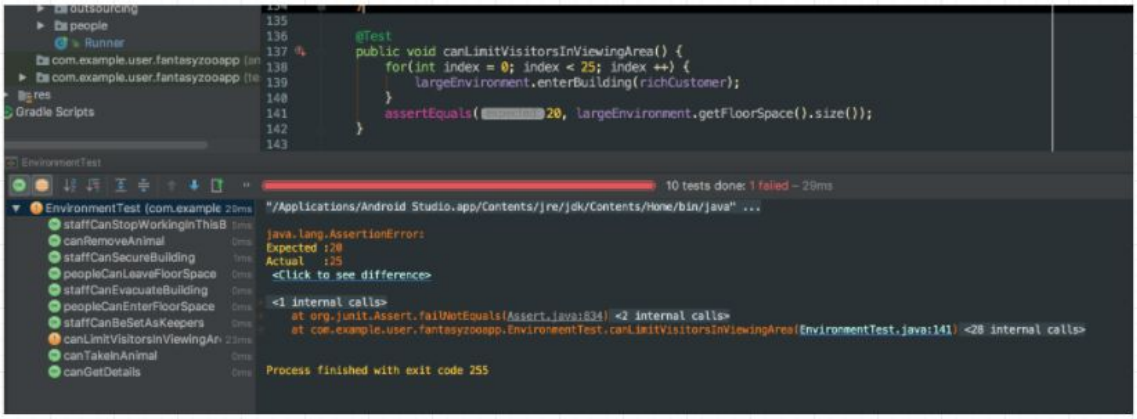


Benjamin R Conway
P.17 Bug Tracking Report

User can select pre-planned route	Failed	Refactored scope of render function	Passed
User can select waypoints and see markers on map	Failed	Add extra data for waypoints in database to define waypoint location	Passed
Waypoint Markers are correctly located	Failed	Correct database information with more accurate latitude/longitude	Passed
Origin and destination can be added by autocomplete	Failed	Implement proper request to Google API	Passed
Markers are rendered for pre-planned route waypoints	Failed	Functionality for adding markers is added to rendering routes	Passed
Icons are specific to waypoint type	Failed	Method to differentiate marker icon added to placement function	Passed
Photo from Flickr loads in waypoint InfoWindow	Failed	Flickr request made on InfoWindow open rather than marker placement	Passed

Benjamin R Conway
P. 18 Testing in a Program

Example 1 of testing in a program:
Limiting Customer Objects within an Environment



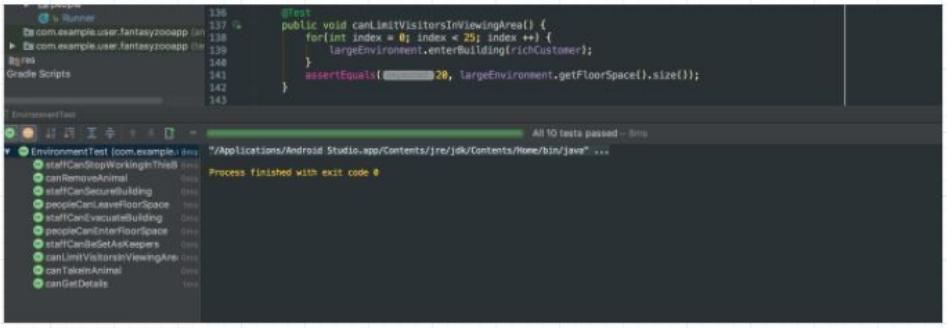
Code used to add customer objects to an environment

```
public void enterBuilding(Customer person) {
        floorSpace.add(person);

}
```

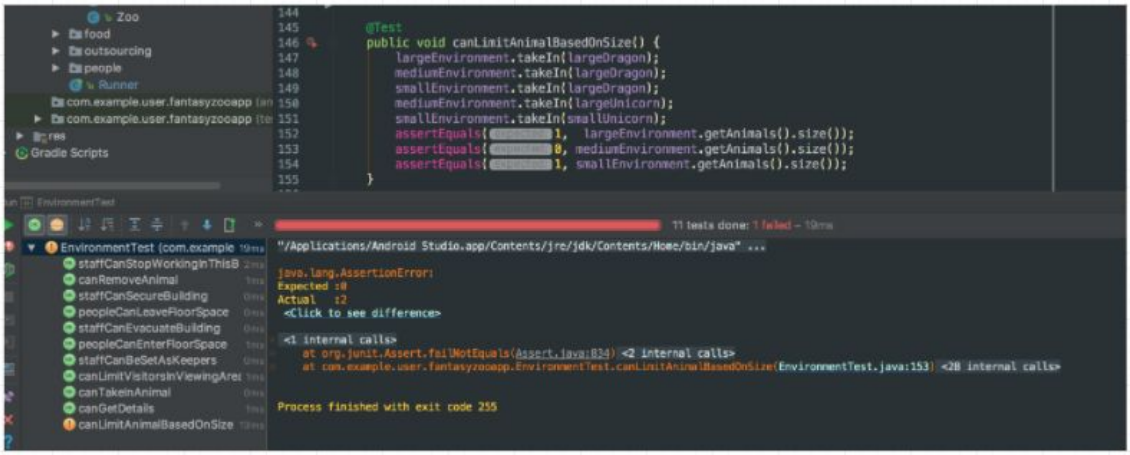
Code adjusted to include limitations

```
public void enterBuilding(Customer person) {
    if(doorsOpen && !buildingIsFull()) {
        floorSpace.add(person);
    }
}
```

With altered code, the test now passes



Example 2 of Testing in a program:
Limiting animal objects capable of being placed in an environment based on size



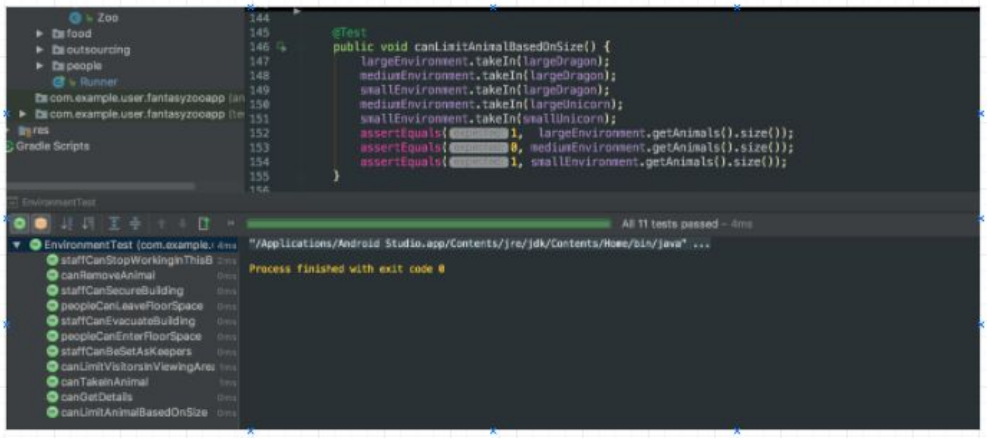
Code to accept Animals into enclosures while failing to pass the test

```
public void takeIn(Animal animal){
        animals.add(animal);
    }
```

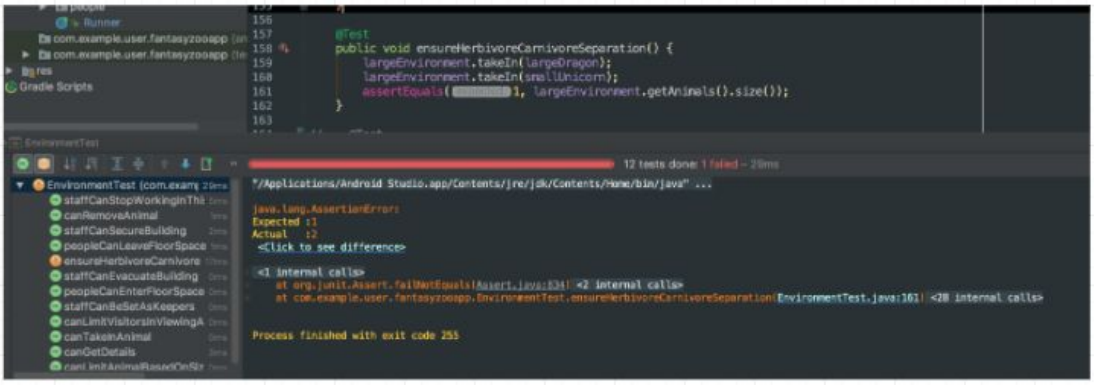
Adjustment made to code to limit size of Animal in particular environment

```
public void takeIn(Animal animal){
        if (isEnvironmentCorrectSize(animal.getSize())) {
    animals.add(animal);
    }
}

private boolean isEnvironmentCorrectSize(Enum<Size> animalSize){
    boolean isBigEnough = false;
        if(animalSize.ordinal() <= size.ordinal()){
            isBigEnough = true;
        }
    return isBigEnough;
}
```

Test running and passing with the new code



Example 3 of Testing in a program:
Ensuring Carnivore and Herbivore/Omnivore objects do not inhabit the same environment



Code that fails the new test

```
public void takeIn(Animal animal){
          if (isEnvironmentCorrectSize(animal.getSize())) {
    animals.add(animal);
    }
}

private boolean isEnvironmentCorrectSize(Enum<Size> animalSize){
    boolean isBigEnough = false;
        if(animalSize.ordinal() <= size.ordinal()){
            isBigEnough = true;
        }
    return isBigEnough;
}
```

Please see next page for changes made to the code and the test passing.

Alterations made:

```
public void takeIn(Animal animal){
    ArrayList<Animal> animalsToCompare = collectAnimalsForComparison(animal);
    if((areAnimalsCompatible(animalsToCompare))  &&
            (isEnvironmentCorrectSize(animal.getSize()))) {
    animals.add(animal);
    }
}
```

```
private ArrayList<Animal> collectAnimalsForComparison(Animal animal){
    ArrayList<Animal> animalsForComparison = new ArrayList<>();
    for(Animal animalPresent: animals){
        animalsForComparison.add(animalPresent);
    }
    animalsForComparison.add(animal);
    return animalsForComparison;
}

private boolean areAnimalsCompatible(ArrayList<Animal> animalsToBeChecked){
    boolean areCompatible = false;
    ArrayList<Animal> carnivores = new ArrayList<>();
    ArrayList<Animal> others = new ArrayList<>();
    for (Animal animal: animalsToBeChecked){
        if (animal instanceof Carnivore){
            carnivores.add(animal);
        }else{ others.add(animal);}
    }
    if((others.size() == 0) || (carnivores.size() == 0)){
        areCompatible = true;
    }
    return areCompatible;
}
```

Altered code passing the new test