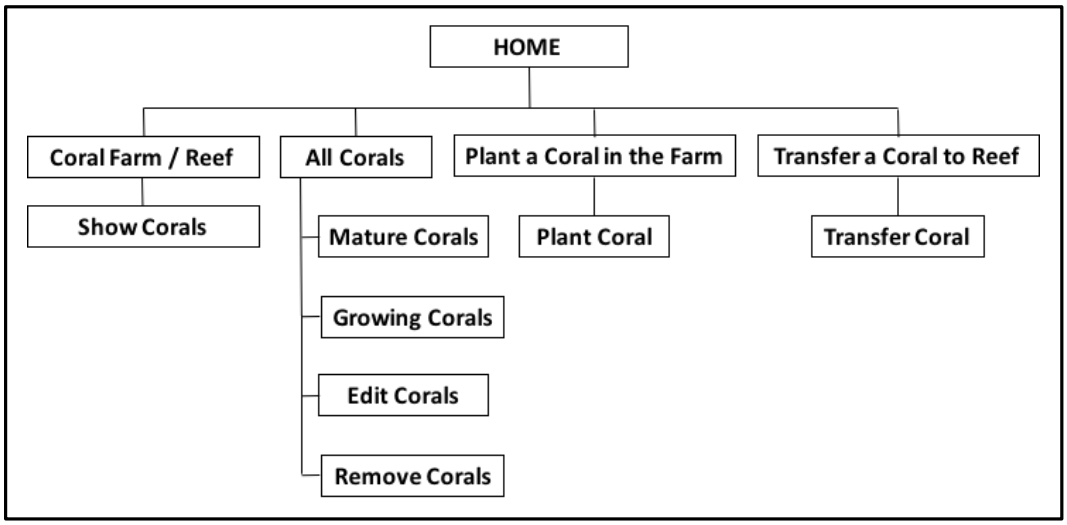
PDA: Software Development Level 8

Student: David Sanchez Rodriguez

EVIDENCE: UNIT P

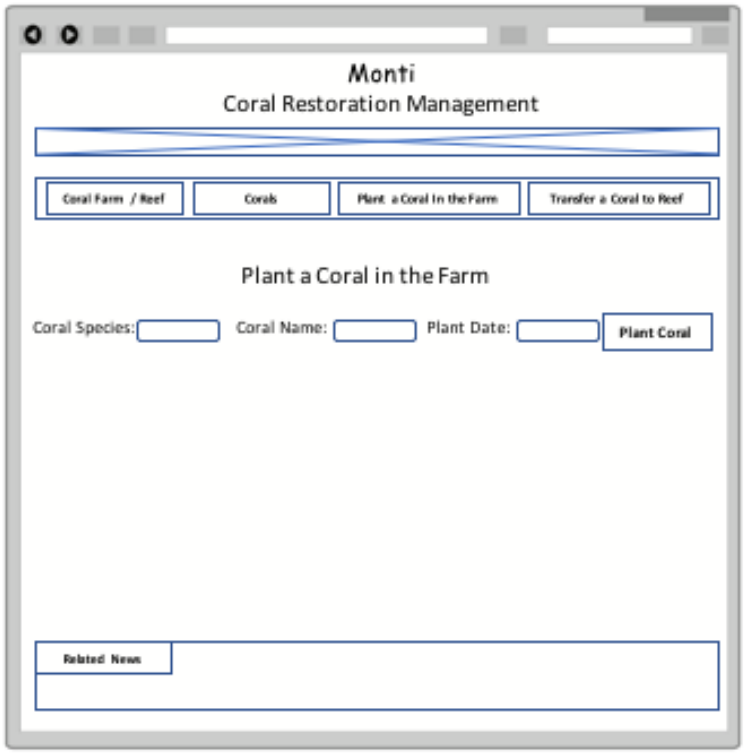
Ref: P5

- Create a user sitemap



Ref: P6

- Produce two wireframe designs



Ref: P10

- Example of pseudocode for a function

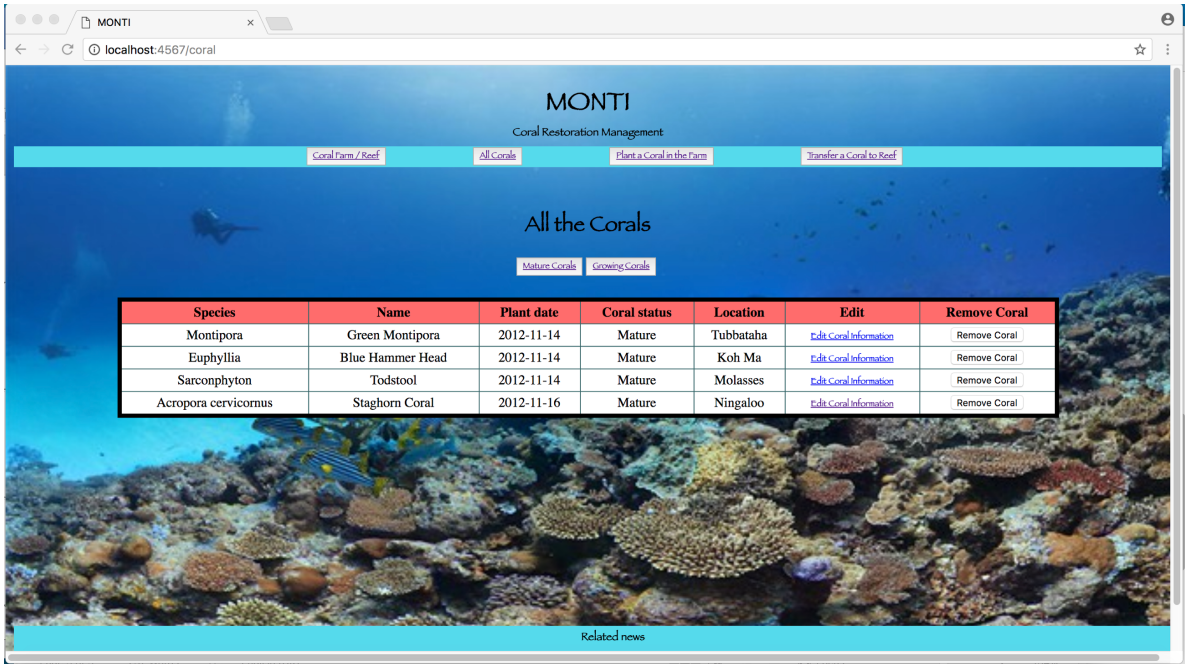
pda.rb

```
82  # Unit: P - P 10
83  #Pseudocode for a function
84  it("should show the name of the corals in the reef", function(){
85    #test the coral name
86    #test the reef name
87    #verify the save() method is working properly in the Database
88    #verify the update() method is working properly in the Database
89    #the coral-reef relationship is one to many, 'one reef can have many corals'
90    #verify each coral is assigned to one and only one reef
91    #the corals should be listed by transfer date
92  }
```

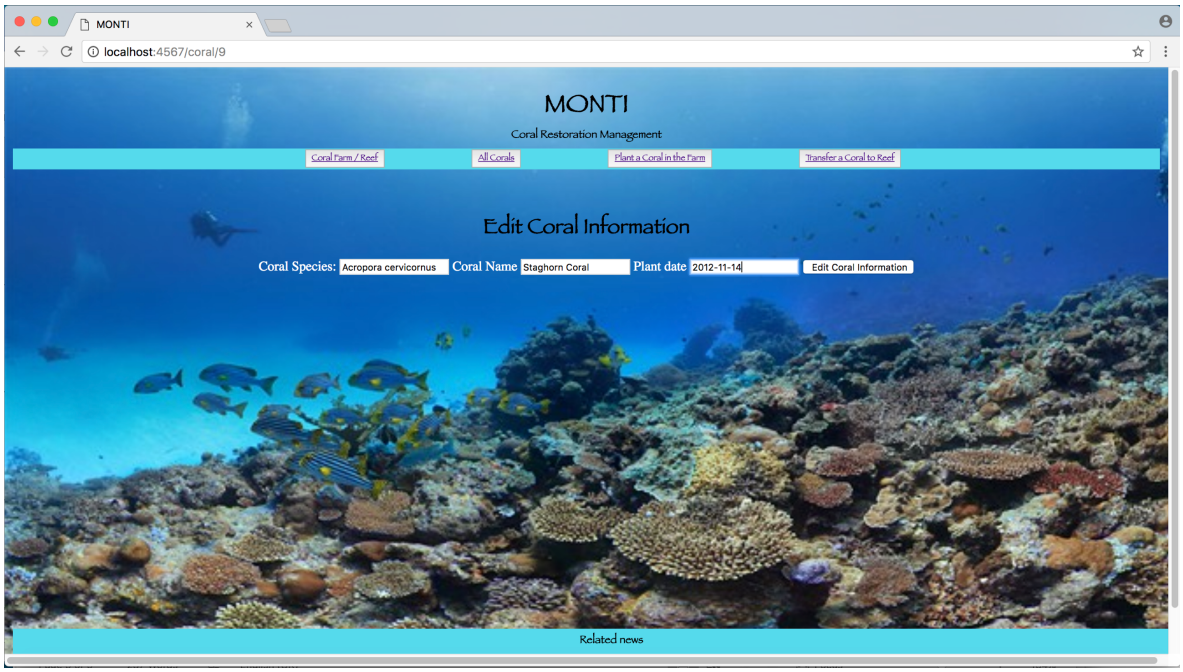
Ref: P13

- **The user inputting something into your program (Updating data)**

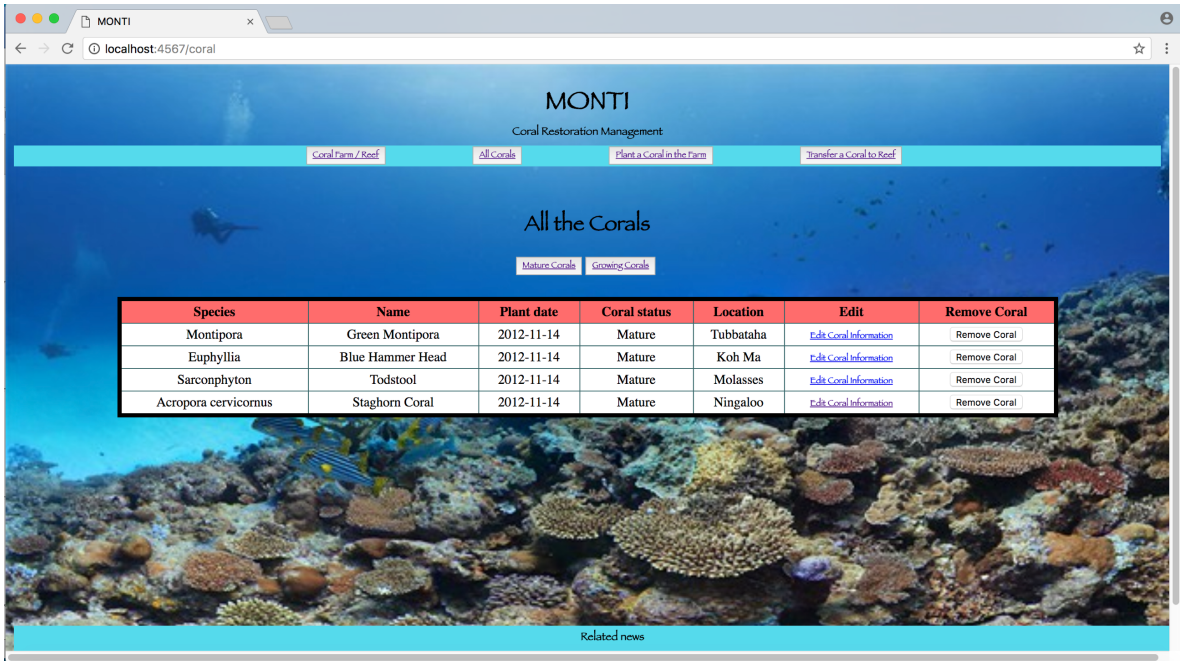
1) The user wants to change the plant date for the coral at the bottom of the table (Acropora cervicornus) from “2012 – 11 – 14” to “2012 – 11 – 16” by clicking the button Edit Coral Information



2) Once the Edit Coral Information is clicked the user is sent to the website view for editing coral information



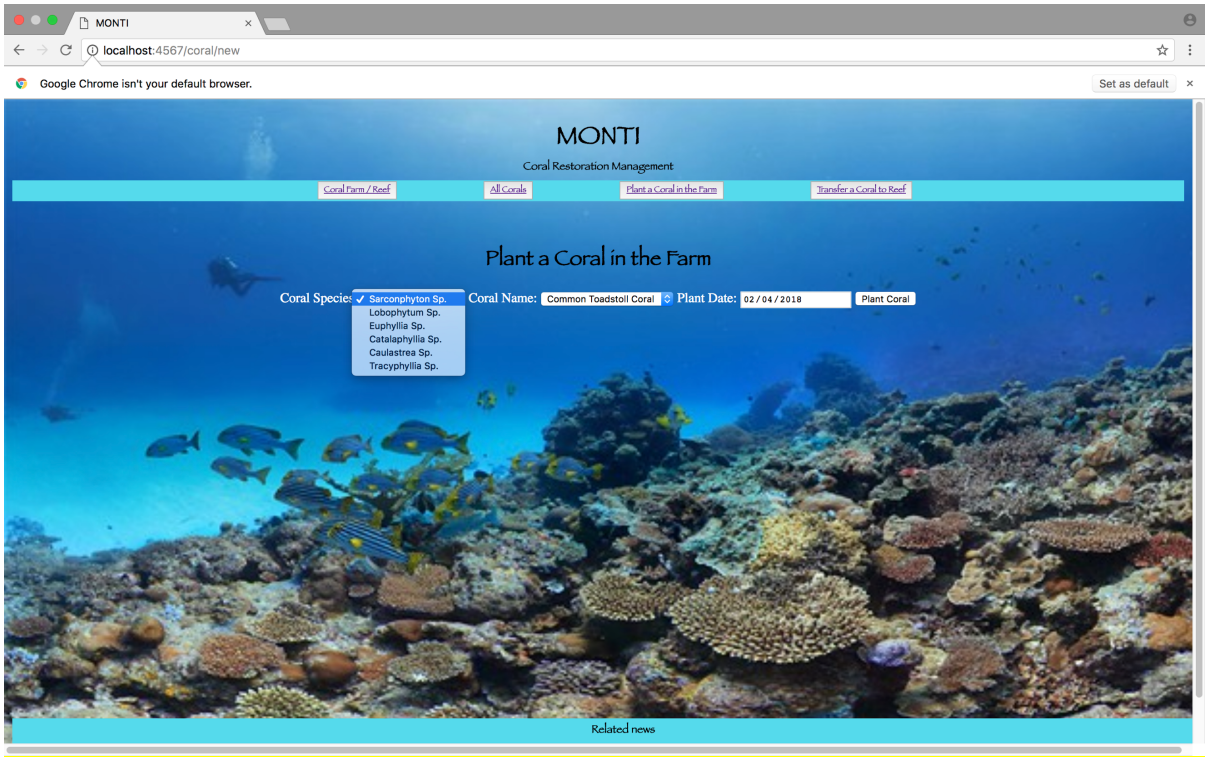
3) Once the information is edited, the coral at the bottom of the table appears with the new plant date.



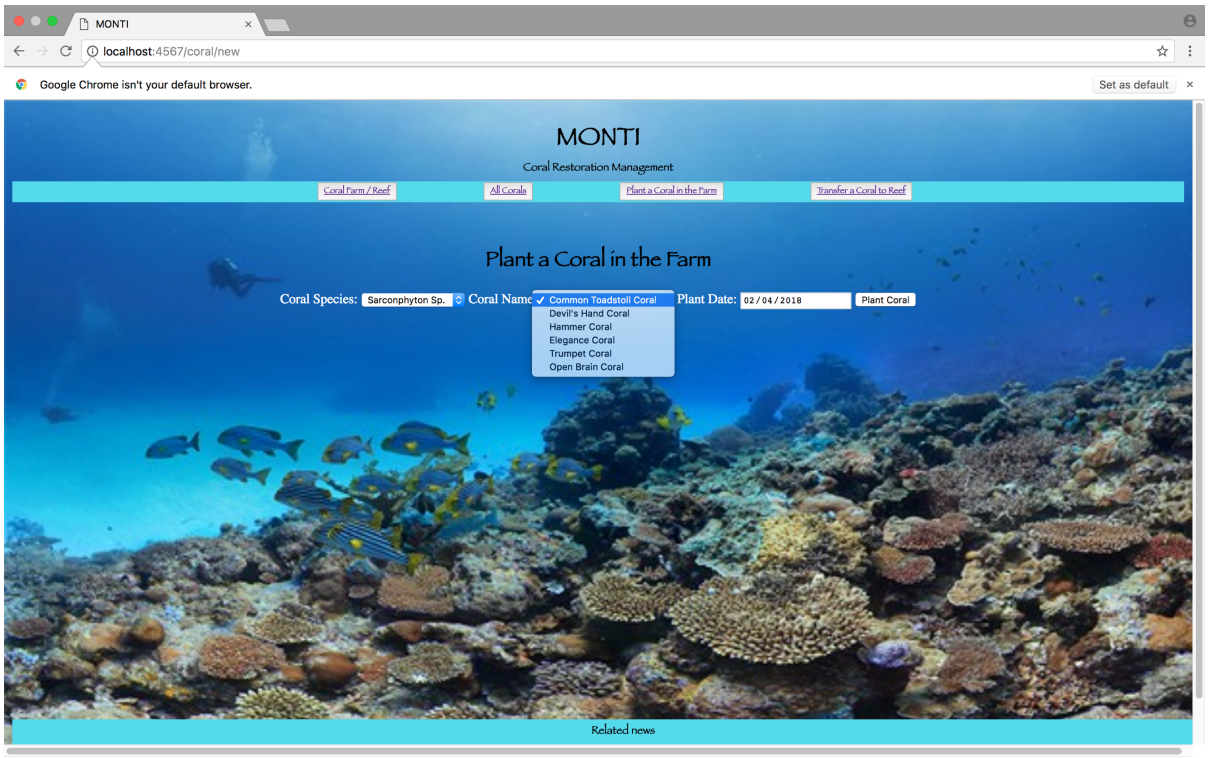
Ref: P14

- **Show an interaction with data persistence (Adding new data).**

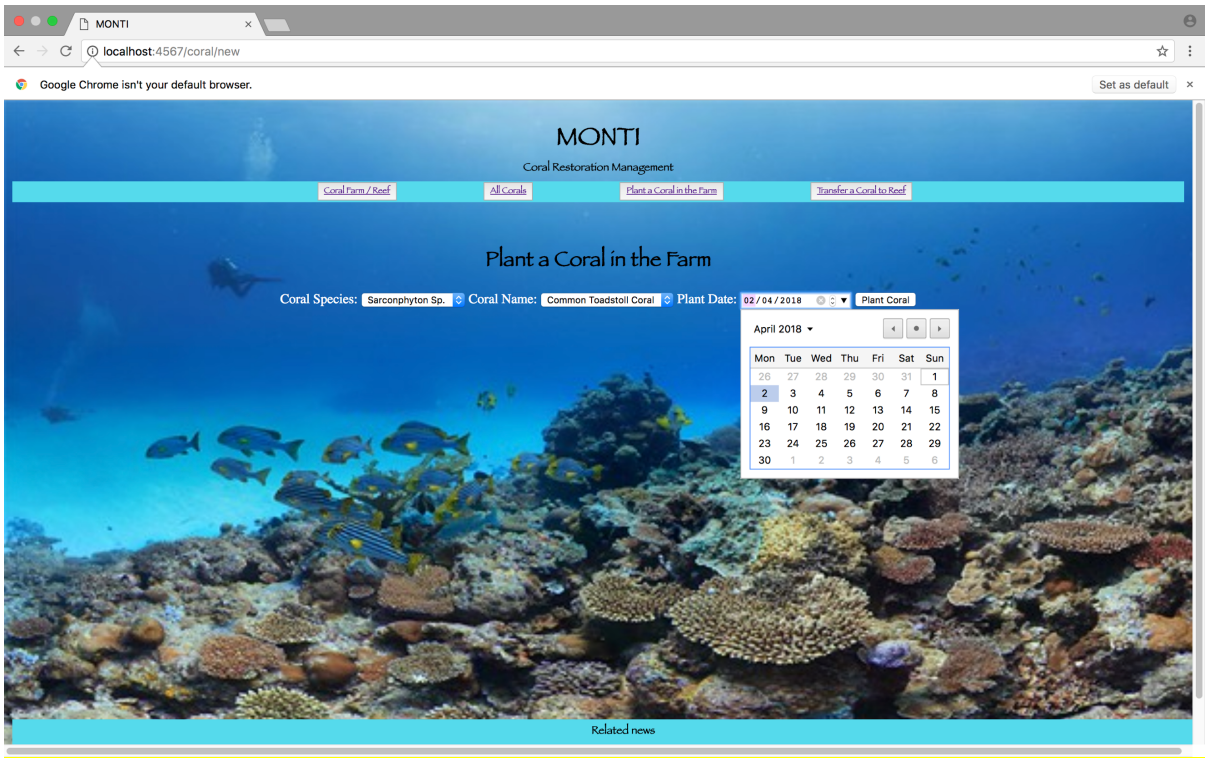
1) The user selecting the Coral Species using the drop-down menu.



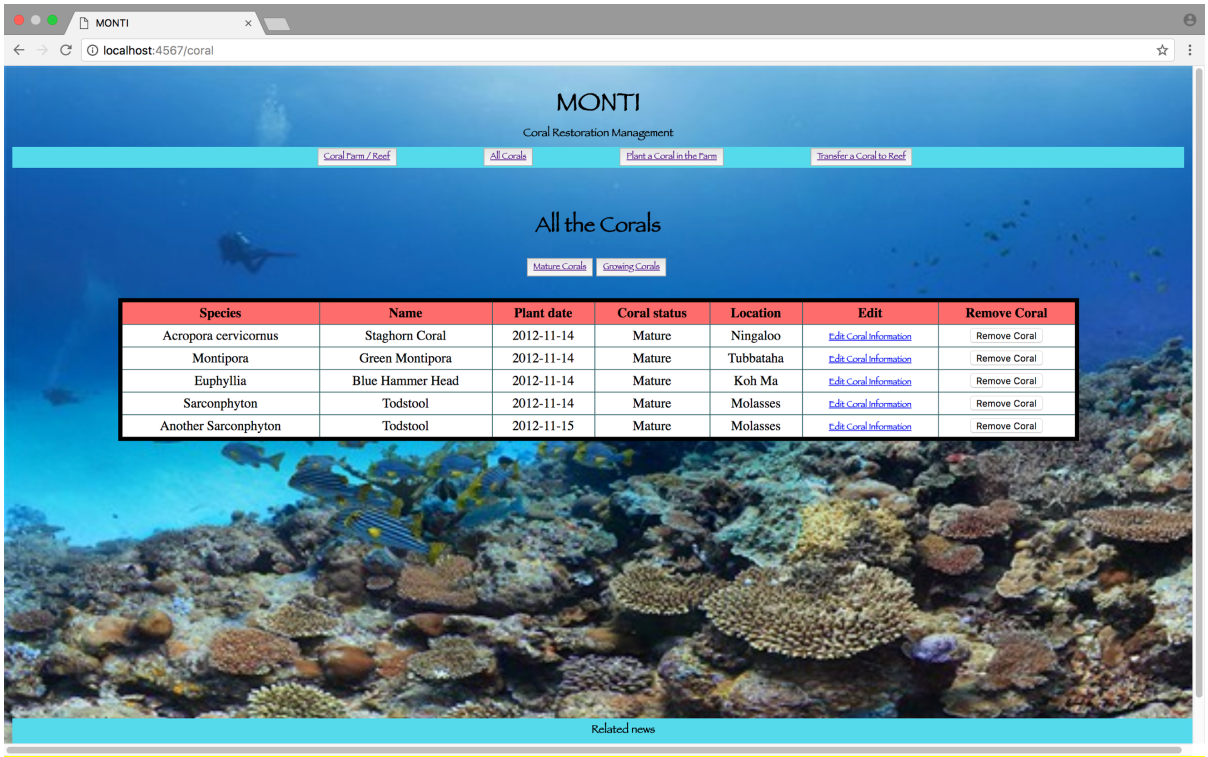
2) The user selecting the Coral Name using the drop-down menu.



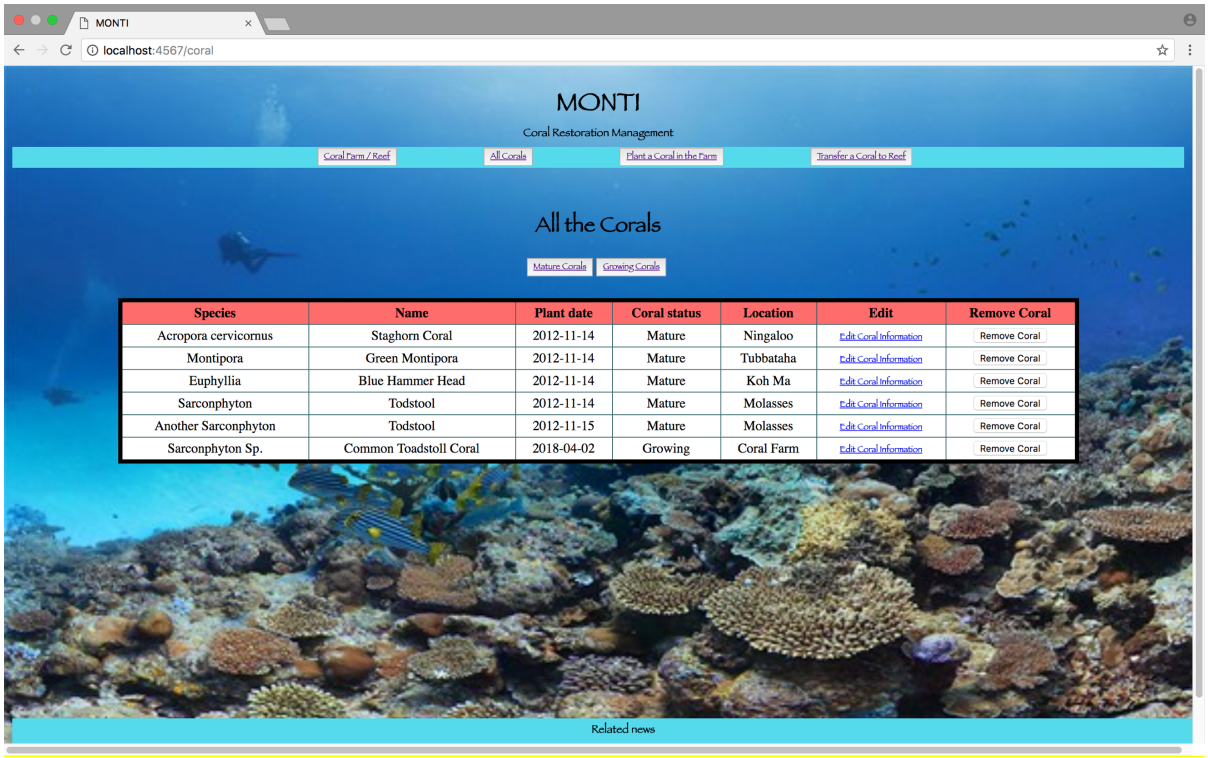
3) The user selecting the Date the coral was planted.



4) Before adding the coral, the “All the Corals” have 5 corals in it.



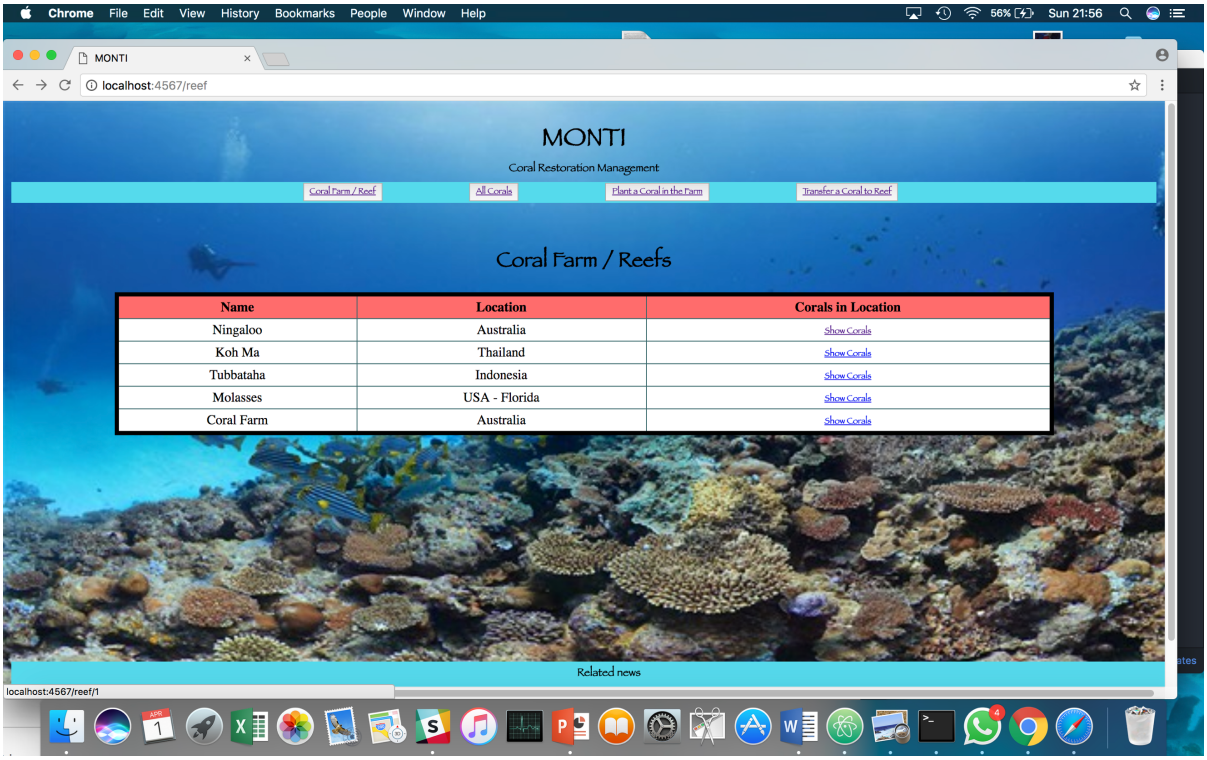
5) After adding the coral, the “All the Corals” have 6 corals in it.



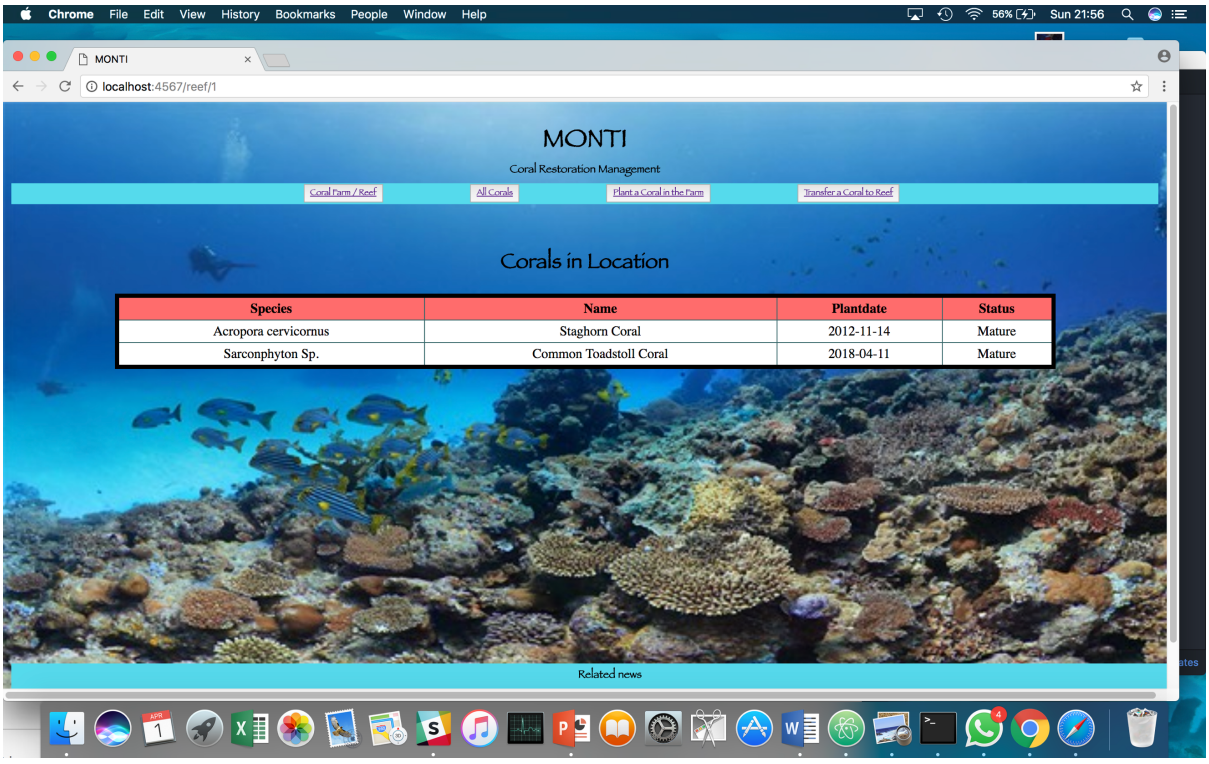
Ref: P15

- Show the correct output of results and feedback to user:

1) The user requesting information or an action to be performed:
   The request is to see all the corals in Ningaloo, Australia

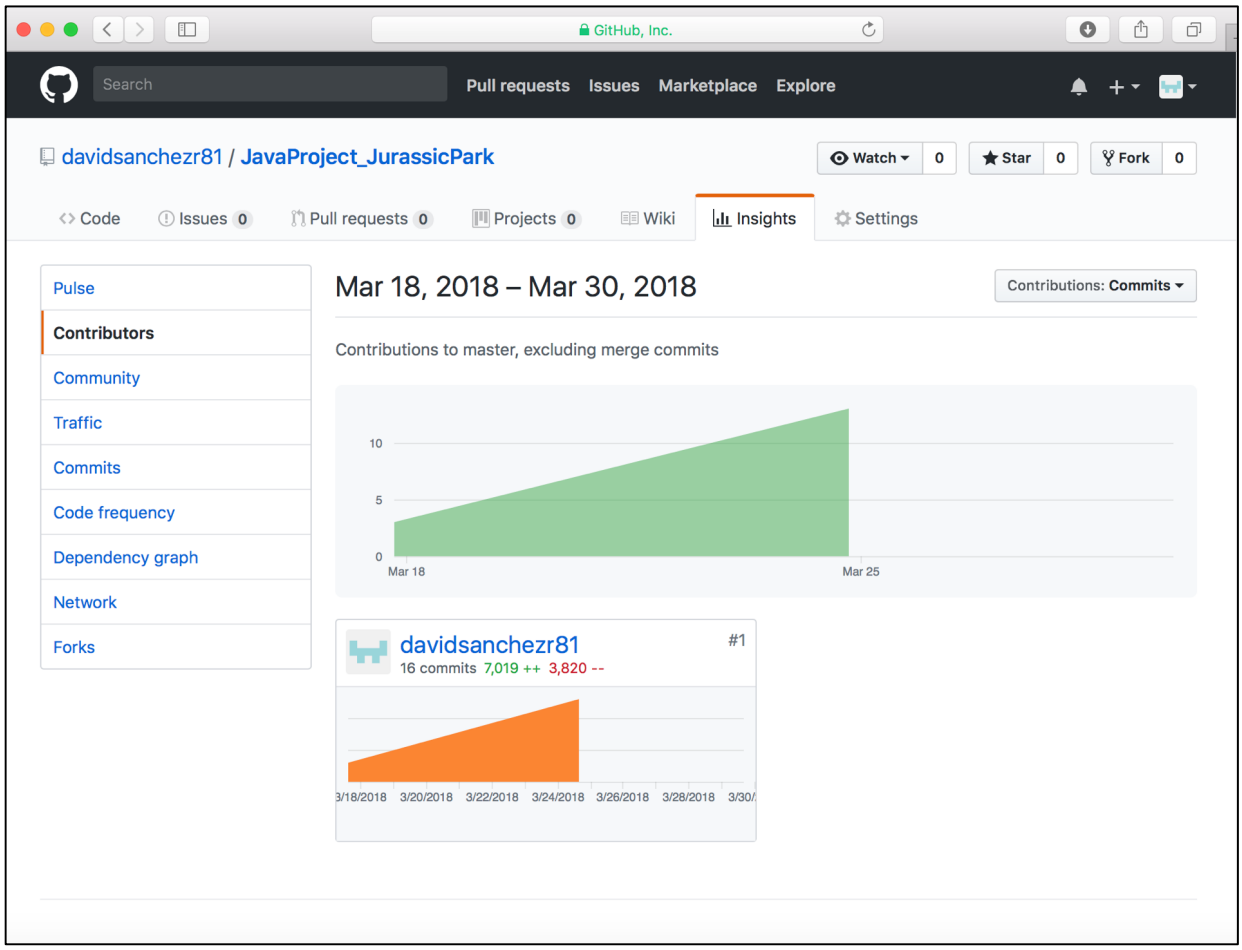


2) The user request being processed correctly and demonstrated in the program:
   The result of the request the 2 corals currently present in Ningaloo, Australia.



Ref: P11

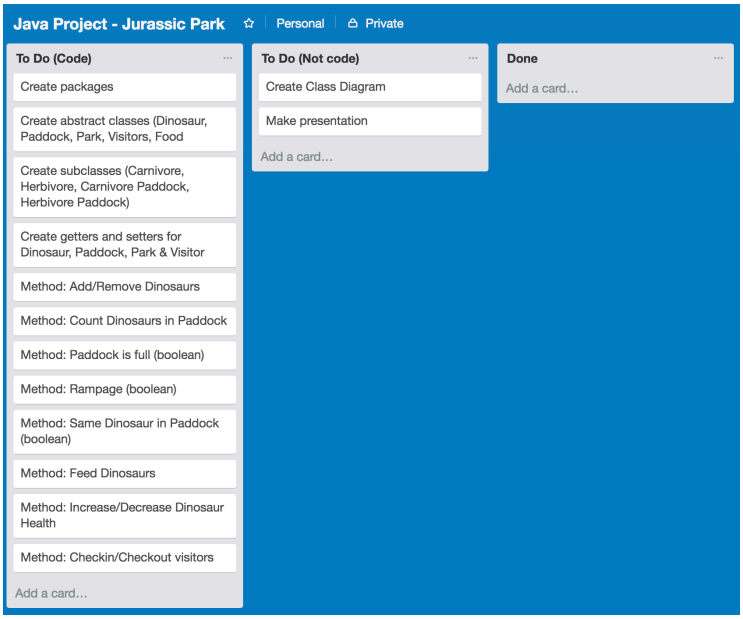
- Take a screenshot of one of your projects where you have worked alone and attach the Github link.



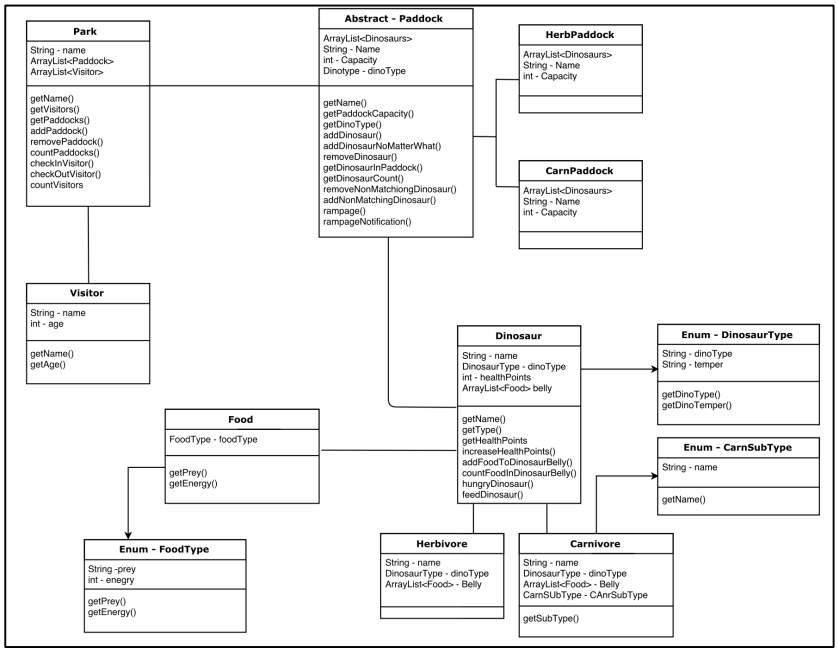
Ref: P12

- Take screenshots or photos of your planning and the different stages of development to show changes.

- *Starting the Java Project using Trello for planning*

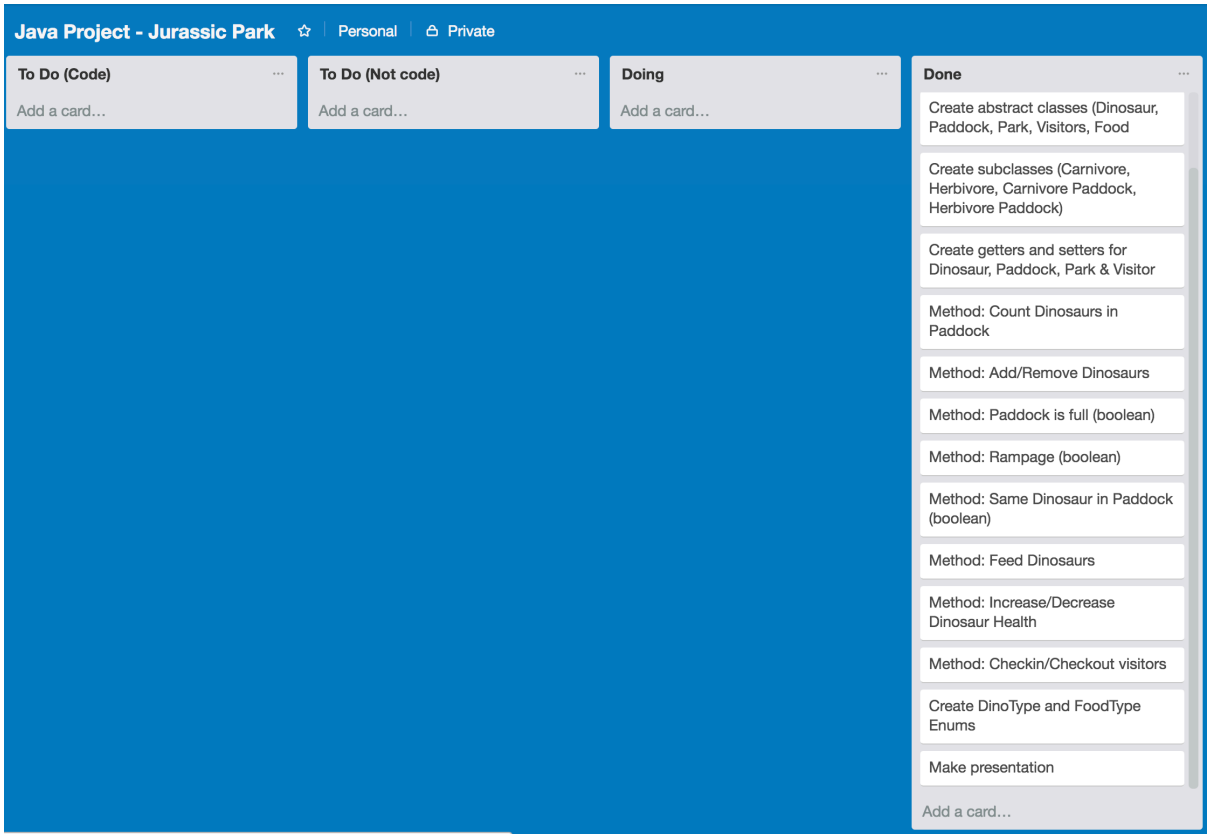


- *Class Diagram of Jurassic Park Project*



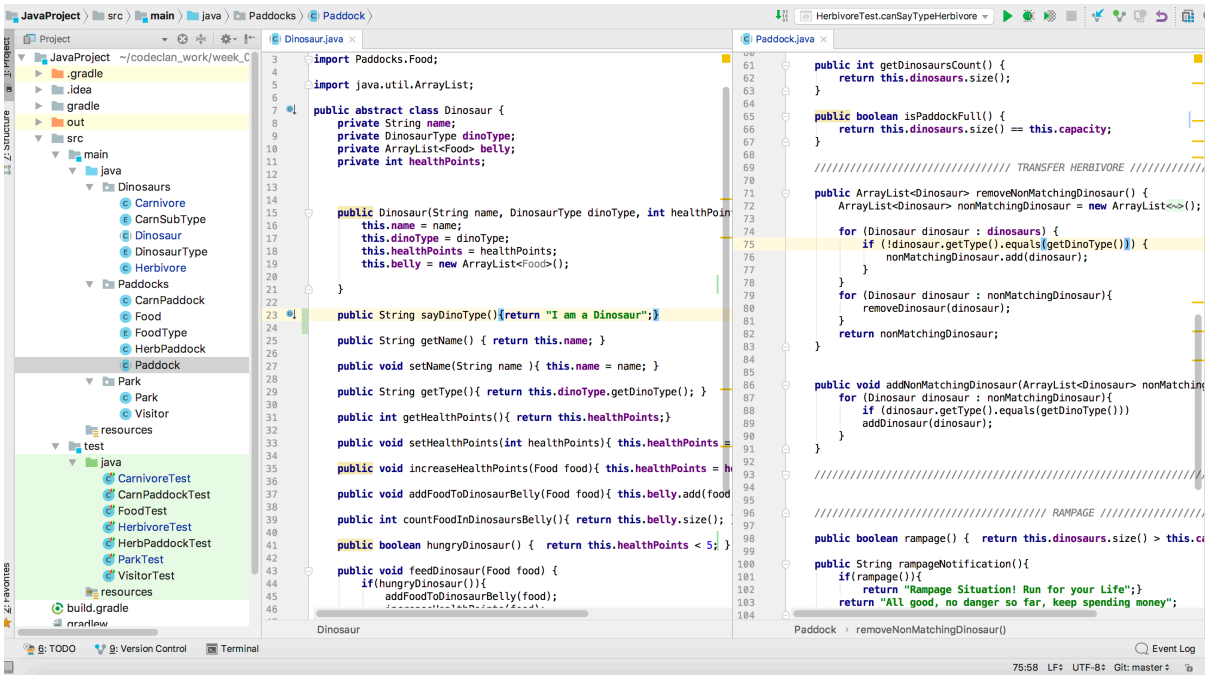
- *50% of the Java Project Completed*

- *100% of the Java Project Completed*



- *PowerPoint presentation*

- *Final Java project layout*



Ref: P18

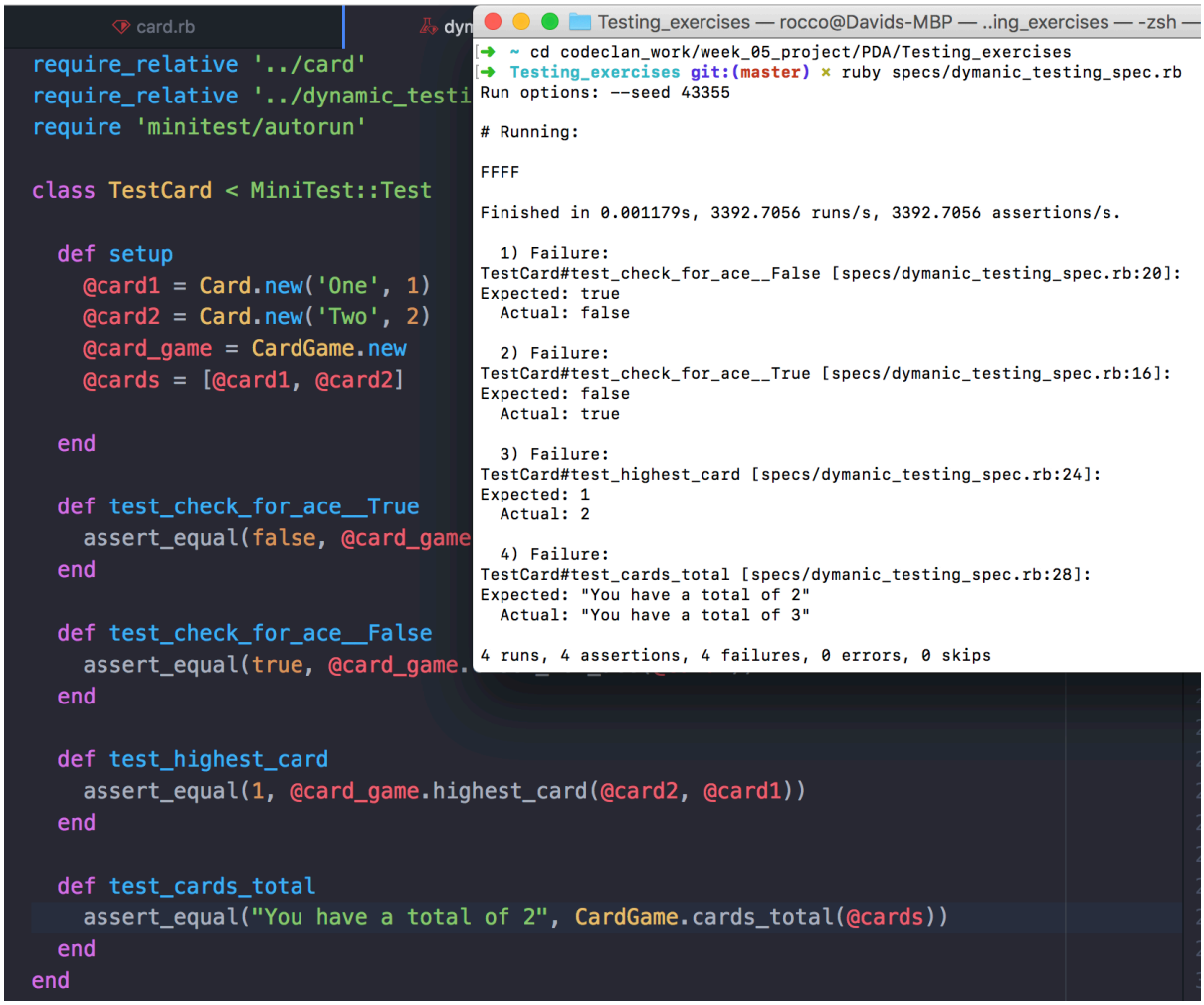
Demonstrate testing in your program. Take screenshots of:

- Example of test code

card.rb | dymanic_testing_spec.rb

```ruby
require_relative '../card'
require_relative '../dynamic_testing.rb'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]

  end

  def test_check_for_ace__True
    assert_equal(false, @card_game.check_for_ace(@card1))
  end

  def test_check_for_ace__False
    assert_equal(true, @card_game.check_for_ace(@card2))
  end

  def test_highest_card
    assert_equal(1, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 2", CardGame.cards_total(@cards))
  end
end
```

- The test code failing to pass



```
card.rb
require_relative '../card'
require_relative '../dynamic_testi
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]

  end

  def test_check_for_ace__True
    assert_equal(false, @card_game
  end

  def test_check_for_ace__False
    assert_equal(true, @card_game.
  end

  def test_highest_card
    assert_equal(1, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 2", CardGame.cards_total(@cards))
  end
end
```

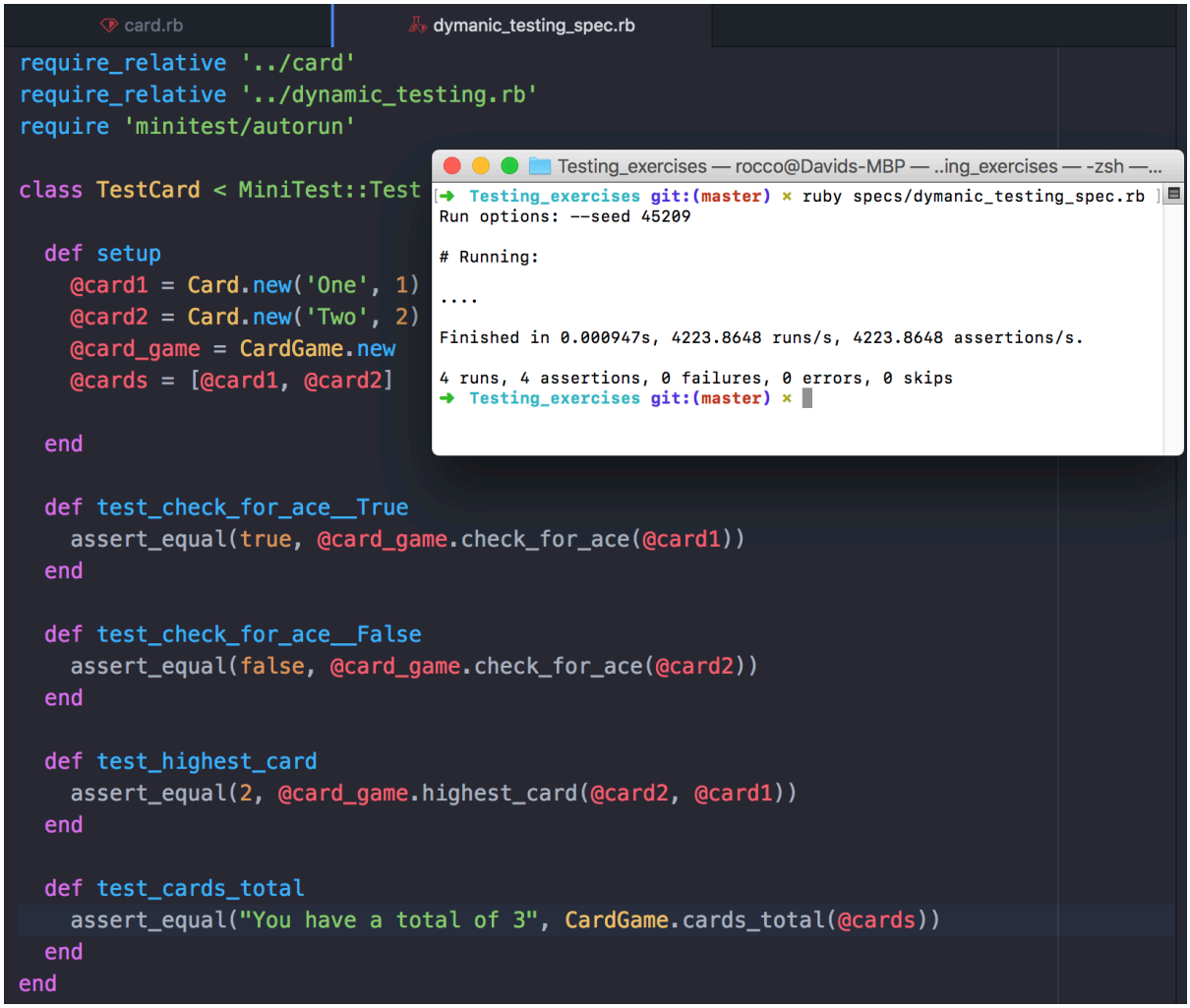
```
Testing_exercises — rocco@Davids-MBP — ..ing_exercises — -zsh —
➜  ~ cd codeclan_work/week_05_project/PDA/Testing_exercises
➜  Testing_exercises git:(master) ✗ ruby specs/dymanic_testing_spec.rb
Run options: --seed 43355

# Running:

FFFF

Finished in 0.001179s, 3392.7056 runs/s, 3392.7056 assertions/s.

  1) Failure:
TestCard#test_check_for_ace__False [specs/dymanic_testing_spec.rb:20]:
Expected: true
  Actual: false

  2) Failure:
TestCard#test_check_for_ace__True [specs/dymanic_testing_spec.rb:16]:
Expected: false
  Actual: true

  3) Failure:
TestCard#test_highest_card [specs/dymanic_testing_spec.rb:24]:
Expected: 1
  Actual: 2

  4) Failure:
TestCard#test_cards_total [specs/dymanic_testing_spec.rb:28]:
Expected: "You have a total of 2"
  Actual: "You have a total of 3"

4 runs, 4 assertions, 4 failures, 0 errors, 0 skips
```

- Example of the test code once errors have been corrected

card.rb | dymanic_testing_spec.rb

```ruby
require_relative '../card'
require_relative '../dynamic_testing.rb'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]

  end

  def test_check_for_ace__True
    assert_equal(true, @card_game.check_for_ace(@card1))
  end

  def test_check_for_ace__False
    assert_equal(false, @card_game.check_for_ace(@card2))
  end

  def test_highest_card
    assert_equal(2, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 3", CardGame.cards_total(@cards))
  end
end
```

- The test code passing



card.rb

dymanic_testing_spec.rb

```ruby
require_relative '../card'
require_relative '../dynamic_testing.rb'
require 'minitest/autorun'

class TestCard < MiniTest::Test

  def setup
    @card1 = Card.new('One', 1)
    @card2 = Card.new('Two', 2)
    @card_game = CardGame.new
    @cards = [@card1, @card2]

  end

  def test_check_for_ace__True
    assert_equal(true, @card_game.check_for_ace(@card1))
  end

  def test_check_for_ace__False
    assert_equal(false, @card_game.check_for_ace(@card2))
  end

  def test_highest_card
    assert_equal(2, @card_game.highest_card(@card2, @card1))
  end

  def test_cards_total
    assert_equal("You have a total of 3", CardGame.cards_total(@cards))
  end
end
```

Testing_exercises — rocco@Davids-MBP — ..ing_exercises — -zsh —...

```
[➜  Testing_exercises git:(master) ✗ ruby specs/dymanic_testing_spec.rb ]
Run options: --seed 45209

# Running:

....

Finished in 0.000947s, 4223.8648 runs/s, 4223.8648 assertions/s.

4 runs, 4 assertions, 0 failures, 0 errors, 0 skips
➜  Testing_exercises git:(master) ✗
```

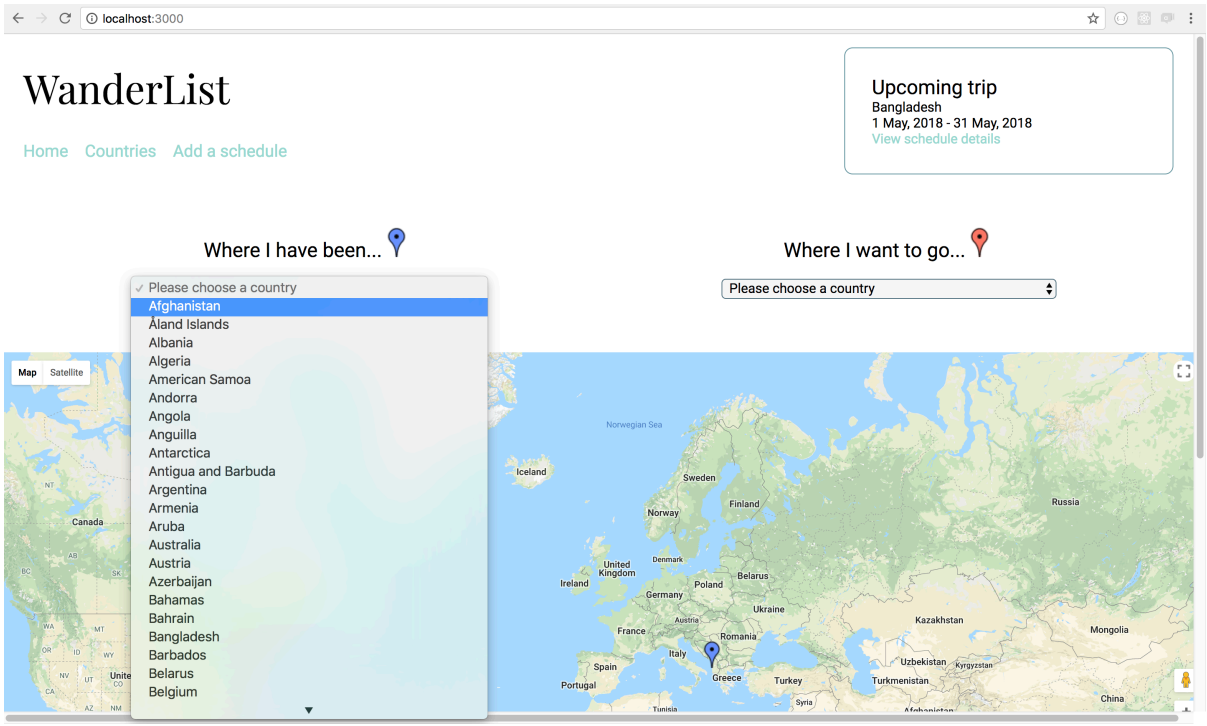
Ref: P16

Show an API being used within your program. Take a screenshot of:

- The code that uses or implements the API

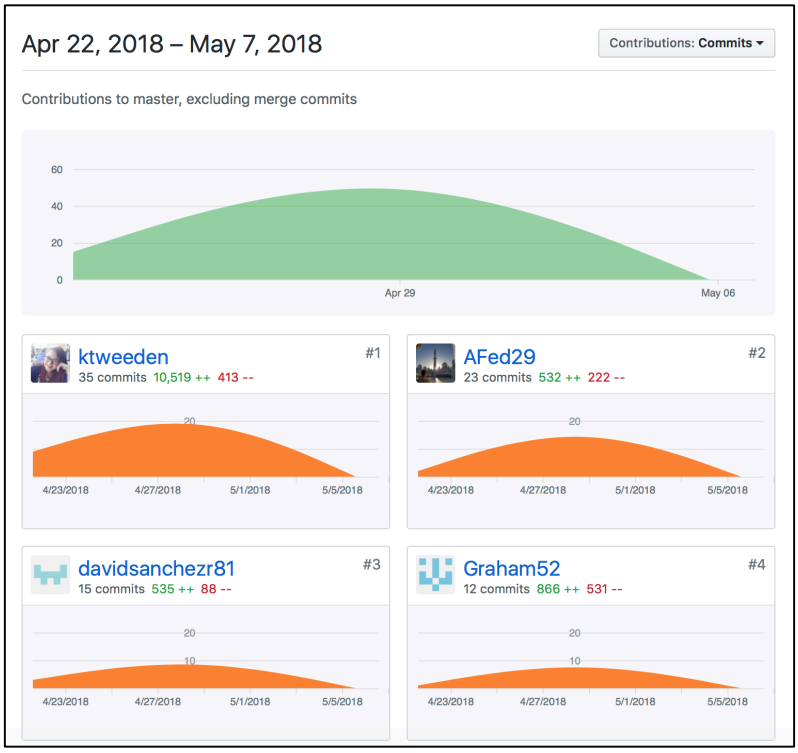
```
const Request = require('../helpers/request.js');

const Countries = function() {
  this.allCountries = [];
  this.visitedCountries = [];
  this.toVisitCountries = [];
  this.url = 'https://restcountries.eu/rest/v2/all';
}

Countries.prototype.getData = function(onComplete) {
  const request = new Request(this.url);
  request.get((response) => {
    this.allCountries = response;
    onComplete(response);
  });
};

Countries.prototype.findIfCountryAlreadyInVisited = function (inputCountry) {
  return this.visitedCountries.some((country) => {
    return country.name === inputCountry.name;
  });
};

Countries.prototype.findIfCountryAlreadyInToVisit = function (inputCountry) {
  return this.toVisitCountries.some((country) => {
    return country.name === inputCountry.name;
  });
};

module.exports = Countries;
```

- The API being used by the program whilst running



Ref: P1

Take a screenshot of the contributor's page on Github from your group project to show the team you worked with:



Ref: P2

Take a screenshot of the project brief from your grouo project

README.md

Travel Dashboard

Travel enthusiasts want to be able to track the countries they have visited and where they want to travel to in the future. Use an existing API, or create a new one, to display information about countries, travel plans, and other relevant information.

MVP

- Display countries for users to select from
- Allow users to add countries to a "countries I have been to" list
- Allow users to add countries to a "countries I would like to visit" list

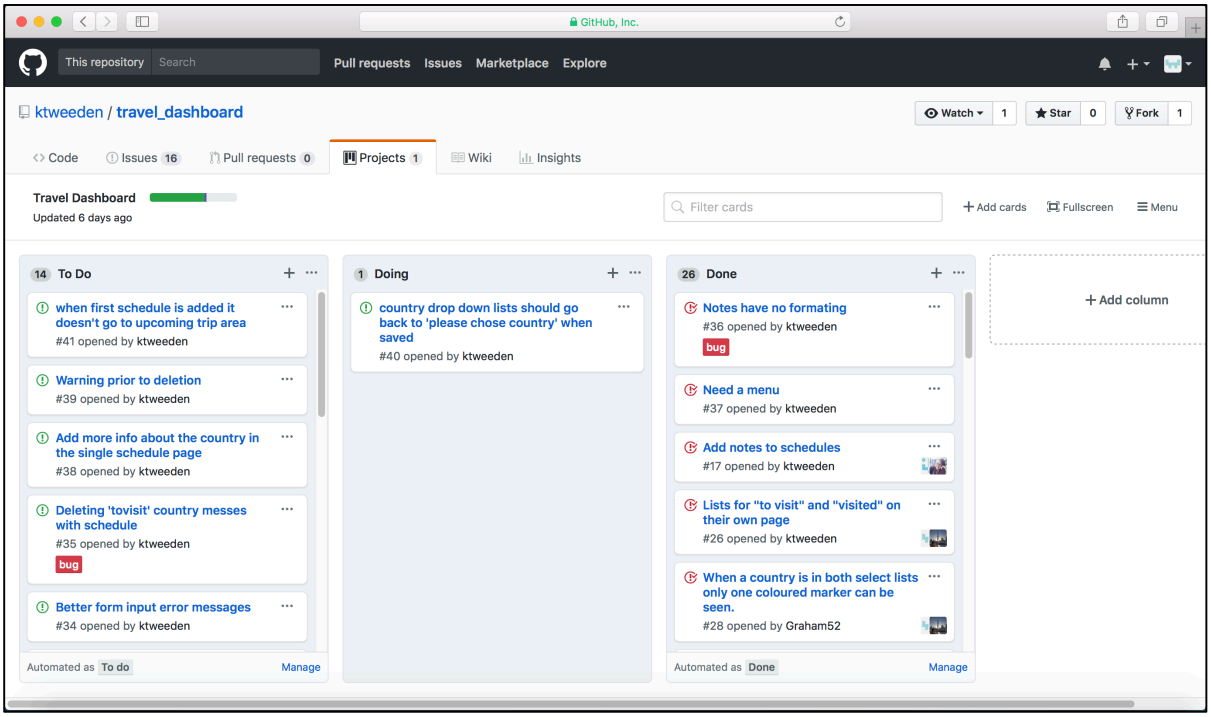
Possible extensions

- Display both lists on a map
- Display currency and languages for a selected country
- Allow users to create a schedule for an upcoming trip

A project made in week 14 of CodeClan by Alex, David, Graham and Kate.

Ref: P3

Provide a screenshot of the planning you completed during your group project:



Ref: P4

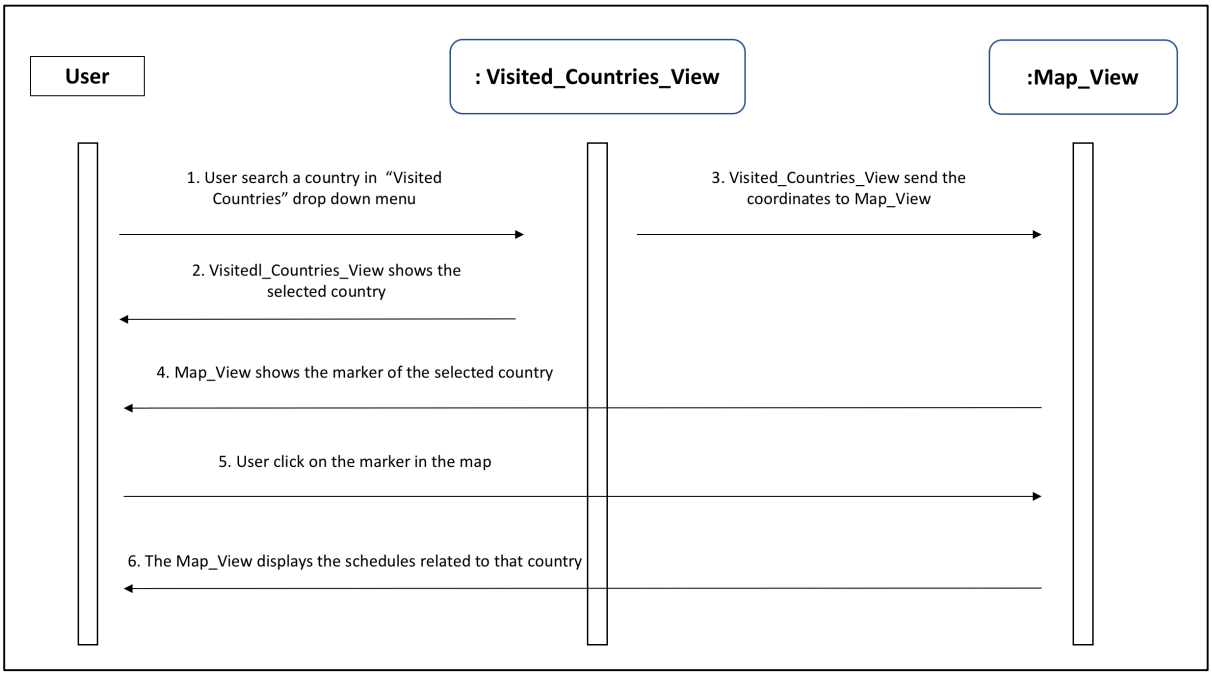
Write an acceptance criteria and test plan.

Acceptance Criteria	Expected Result / Output	Pass / Fail
The user is able to see the countries he previously visited	The list of the countries visited by the user is displayed	Pass
The user is able to see the marker in the map over every country he visited	The markers appears in the map once the user selects a country from the dropdown menu	Pass
The user is able to see the schedules on "trip date order"	The schedules are displayed on "trip date order"	Pass
The user is able to see photos/notes from the schedule	The user should be able to see photos and notes from the previous trips	Fail

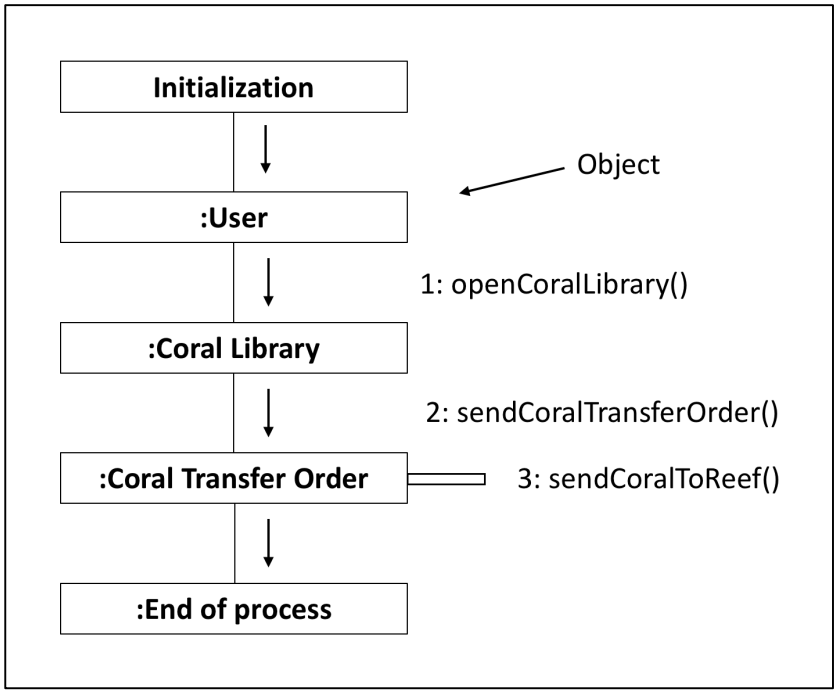
Ref: P7

Produce two system interaction diagrams:

- Sequence diagram

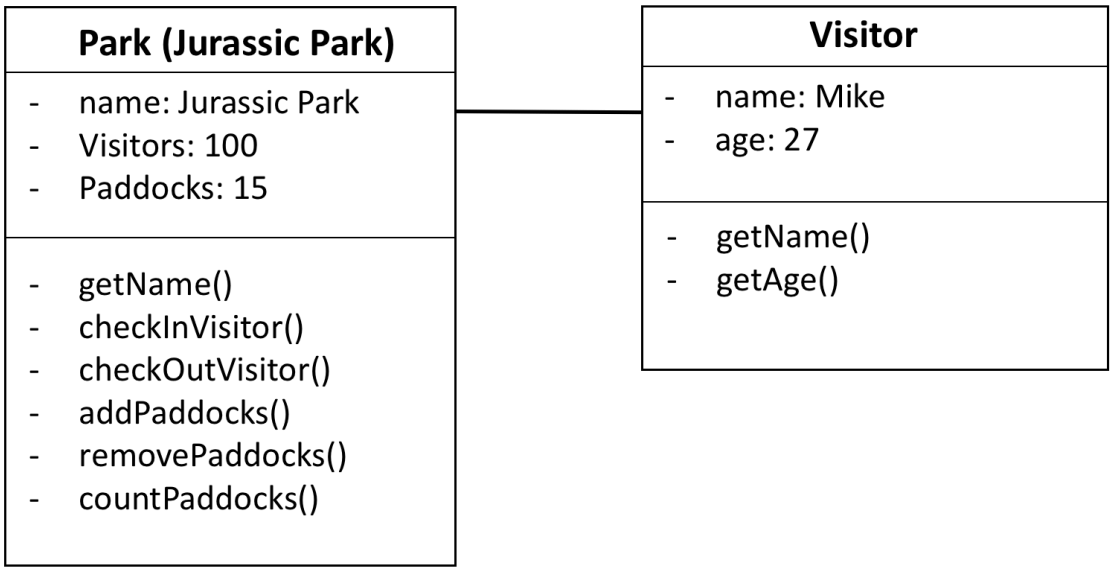


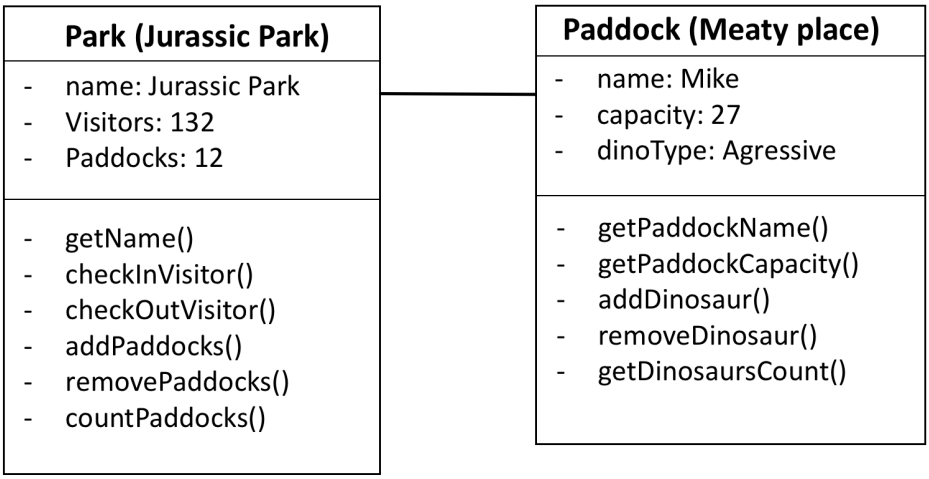
- Collaboration diagram



Ref: P8

Produce two object diagrams.





Ref: P9

Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.

1) **feedDinosaur()**

The reason this method was chosen is because the dinosaurs needed to be fed. However, this method is composed by 3 different methods (hungryDinosaur(), addFoodToDinosaurBelly(), increaseHealthPoints()).

The first method *hungryDinosaur()* would check if the animal is hungry giving a true or false result based on the current health level of the dinosaur. Therefore, if the animal is hungry the second method *addFoodToDinosaurBelly()* would run and the number of units of food eaten would be added to the belly. Finally, the method *increaseHealthPoints()* would increase the health level of the animal. Each food eaten would have a different health value.

```
public void feedDinosaur(Food food) {
    if(hungryDinosaur()){
        addFoodToDinosaurBelly(food);
        increaseHealthPoints(food);
    }
}
```

2) **removeNonMatchingDinosaur()**

The reason this method was chosen is because it was necessary to remove those dinosaurs of a different type from the paddock under evaluation.

First of all, an array for those dinosaurs to be removed from the paddock is created, *nonMatchingDinosaur*.

The first for loop would take those different dinosaurs from the paddock and would add them to the array previously created.

The second for loop would remove form the paddock the dinosaurs added to the array.

This method will return the array.

```
public ArrayList<Dinosaur> removeNonMatchingDinosaur() {
    ArrayList<Dinosaur> nonMatchingDinosaur = new ArrayList<~>();

    for (Dinosaur dinosaur : dinosaurs) {
        if (!dinosaur.getType().equals(getDinoType())) {
            nonMatchingDinosaur.add(dinosaur);
        }
    }
    for (Dinosaur dinosaur : nonMatchingDinosaur){
        removeDinosaur(dinosaur);
    }
    return nonMatchingDinosaur;
}
```

Ref: P17

Produce a bug tracking report

Expected result	1st Testing Attempt		2nd Testing Attempt
The user must be able to see all the countries in the list			Passed
The schedules should be shown based in trip date			Passed
The marker for "Visited" countries should have a different color than the "To visit" countries	Failed	In the "map_wrapper.js" file two different icons were created and linked to google icon .png images (Blue Marker for Visited countries ; Red Marker for To Visit countries)	Passed
The schedules should appear in a different page	Failed	One "budle.js" file for each new page was created	Passed
Schedules cannot overlap	Failed	"If statements" added to avoid adding new schedules in dates that other schedules already exist	Passed