PDA: Software Development
Level 8
Student Evidence Checklist

Full name	**Kelsie Braidwood**
Cohort	**G5**

The evidence required can be taken from your assignments, homework that you have completed on your own or by creating a specific example for the PDA.

Unit	Ref.	Evidence	Done
I & T	I.T 5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running	Y

```ruby

#this is an array containing the different positions that players in a
#volleyball team does
#this is written in Ruby
volleyball_team_positions = ["Middle Blocker",
                             "Setter",
                             "Power",
                             "Swing",
                             "Off - Setter",
                             "Libero"]

```

```ruby
#this is a function that finds a volleyball team position by name
#it prints out the the position if it's found
#and prints nil if not
def find_position_by_name(positions_array, position_name)
  for position in positions_array
    return position if position_name == position
  end
  return nil
end
p find_position_by_name(volleyball_team_positions, "Middle Blocker")
p find_position_by_name(volleyball_team_positions, "Setter")
p find_position_by_name(volleyball_team_positions, "Coach")

```

Week 2	 This is the print method showing the results when you run the ruby file in the terminal.	

I & T	I.T 6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running	Y

```ruby
#this is an hash containing the different positions that players in a
#volleyball team does, along with the name of the players
#this is written in Ruby
 volleyball_team =  { Middle_Blocker: "Kelsie",
                      Setter: "Priya",
                      Power: "Livia",
                      Swing: "Mascha",
                      Off_Setter: "Ina",
                      Libero: "Jill"}

#This is a function that returns the setters name
def print_out_volleyball_setter_name(team_hash)
    return team_hash[:Setter]
end
p print_out_volleyball_setter_name(volleyball_team)

```

```
➜  wk_2 git:(master) ✗ ruby hashes_example.rb
"Priya"
➜  wk_2 git:(master) ✗
```

This is the results of the print function when you run the file in the terminal.

I & T		Static and Dynamic testing task A - on GitHub repo	Y

	Unit	Ref.	Evidence	Done
	I & T	I.T 3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	Y
Week 3				

```
def customers()
  sql = "SELECT customers.* FROM customers INNER JOIN tickets ON
  customers.id = tickets.customer_id WHERE tickets.film_id = $1"
  values = [@id]
  customers = SqlRunner.run(sql, values)
  return Customer.map_customers(customers)
end
```

Above is the function that searches the data, below if the data in the terminal when you run it with Pry.

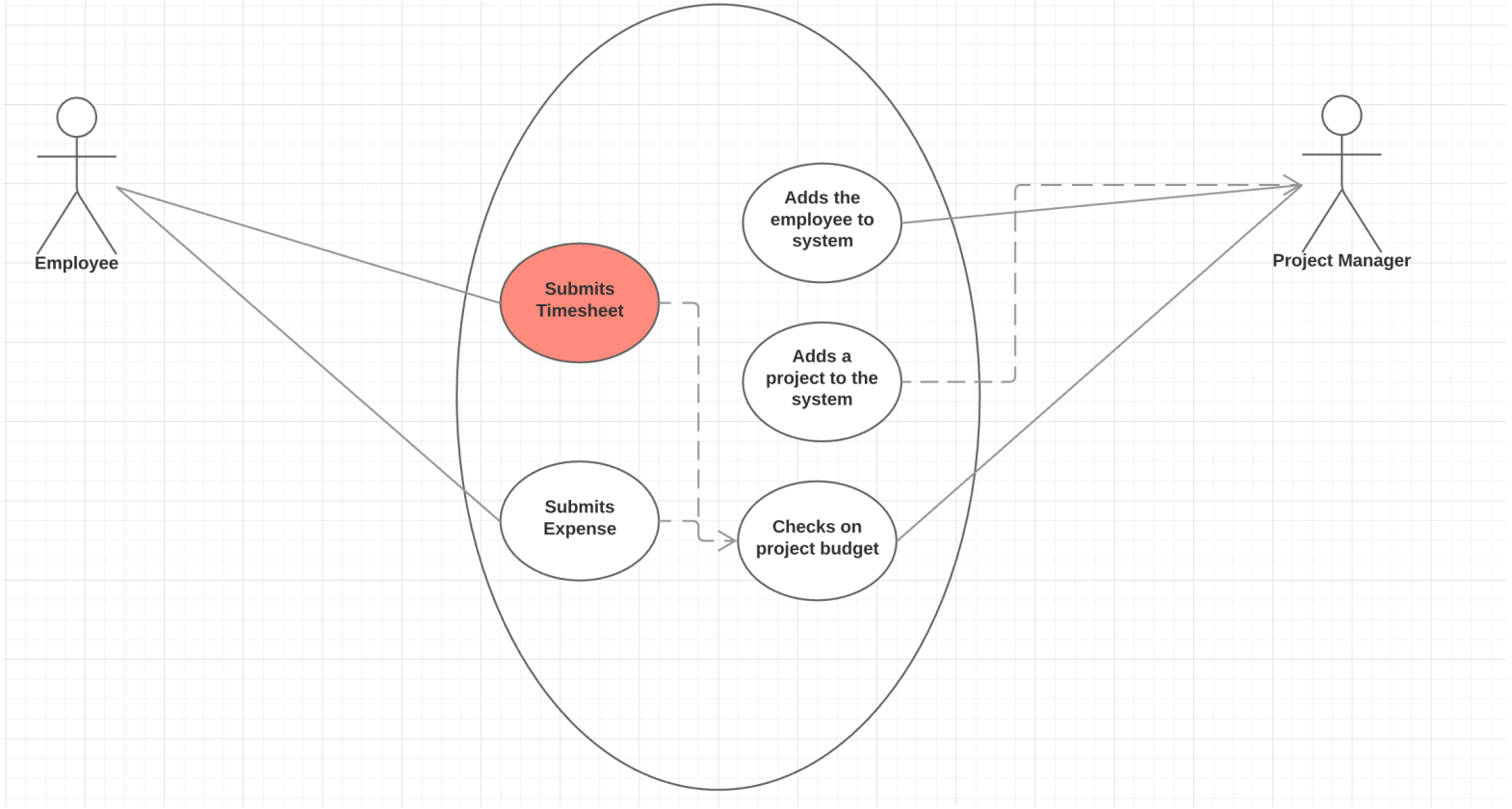
```
[1] pry(main)> film1.customers
=> [#<Customer:0x007fb0a89afd20 @funds=30, @id=17, @name="Kelsie">,
 #<Customer:0x007fb0a89afc30 @funds=30, @id=17, @name="Kelsie">,
 #<Customer:0x007fb0a89afb40 @funds=50, @id=20, @name="Fraser">,
 #<Customer:0x007fb0a89af960 @funds=20, @id=18, @name="Stewart">,
 #<Customer:0x007fb0a89af6e0 @funds=40, @id=19, @name="Claire">]
[2] pry(main)>
```

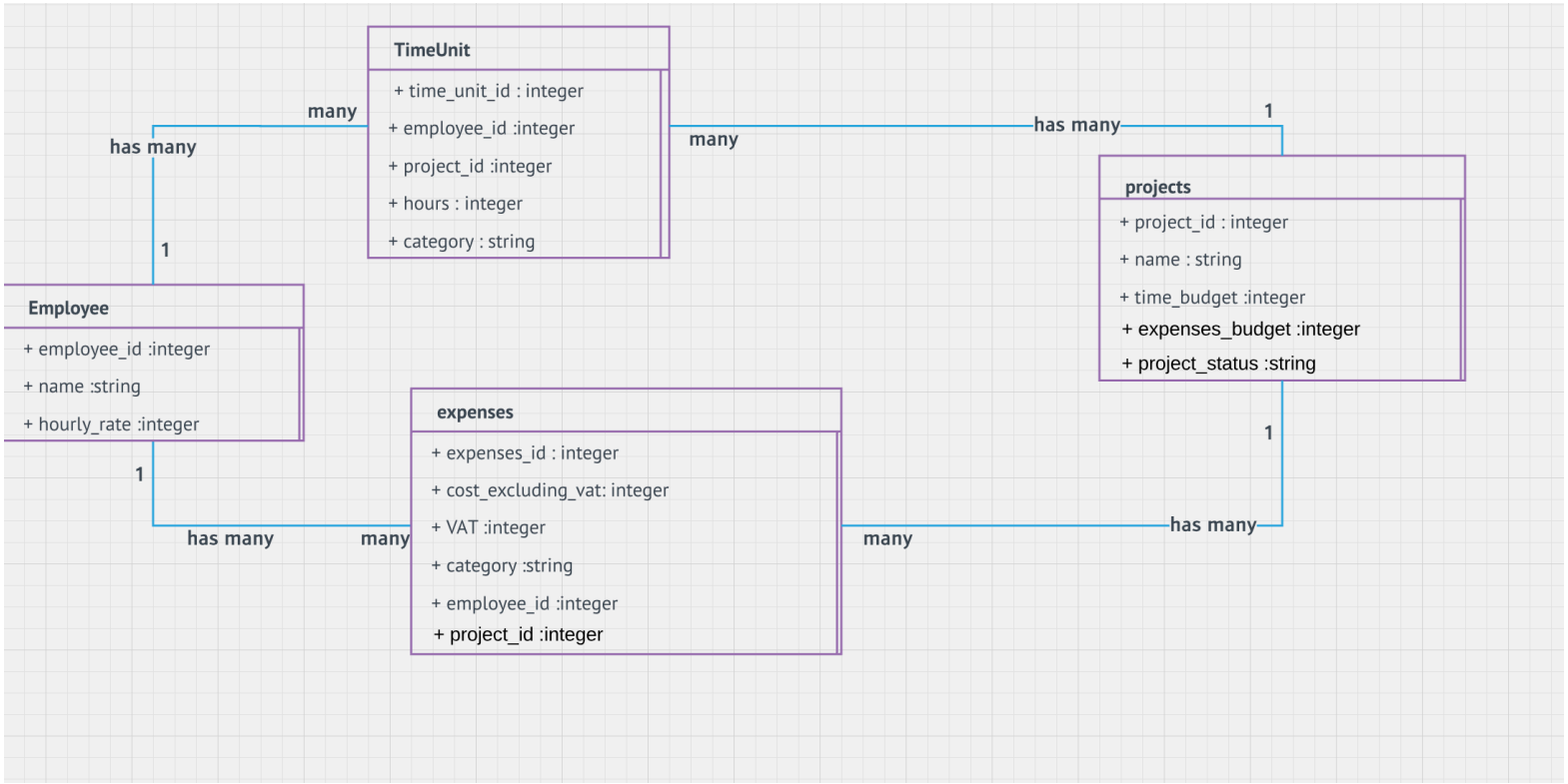
	I & T	I.T 4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	Y

```
def customers()
  sql = "SELECT customers.* FROM customers INNER JOIN tickets ON
  customers.id = tickets.customer_id WHERE tickets.film_id = $1 ORDER BY
  customers.name ASC;"
  values = [@id]
  customers = SqlRunner.run(sql, values)
  return Customer.map_customers(customers)
end
```

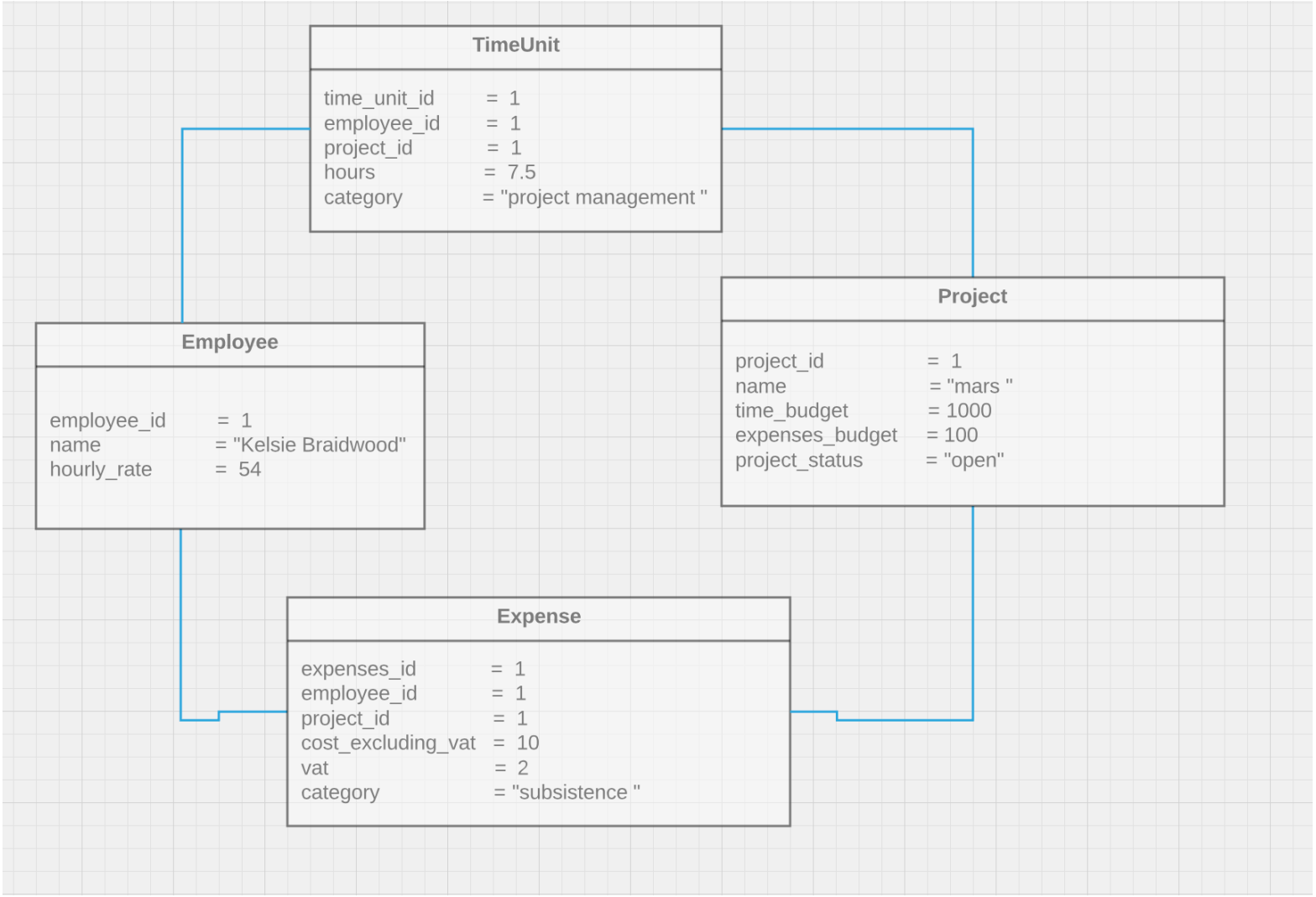
Above is a function that sorts data by the customers name alphabetically, below is the results when you query the data with pry.

```
[1] pry(main)> film1.customers
=> [#<Customer:0x007fcc46226c98
  @funds=40,
  @id=31,
  @name="Claire">,
 #<Customer:0x007fcc46226b30
  @funds=50,
  @id=32,
  @name="Fraser">,
 #<Customer:0x007fcc462269f0
  @funds=30,
  @id=29,
  @name="Kelsie">,
 #<Customer:0x007fcc462266d0
  @funds=30,
  @id=29,
  @name="Kelsie">,
 #<Customer:0x007fcc46226518
  @funds=20,
  @id=30,
```

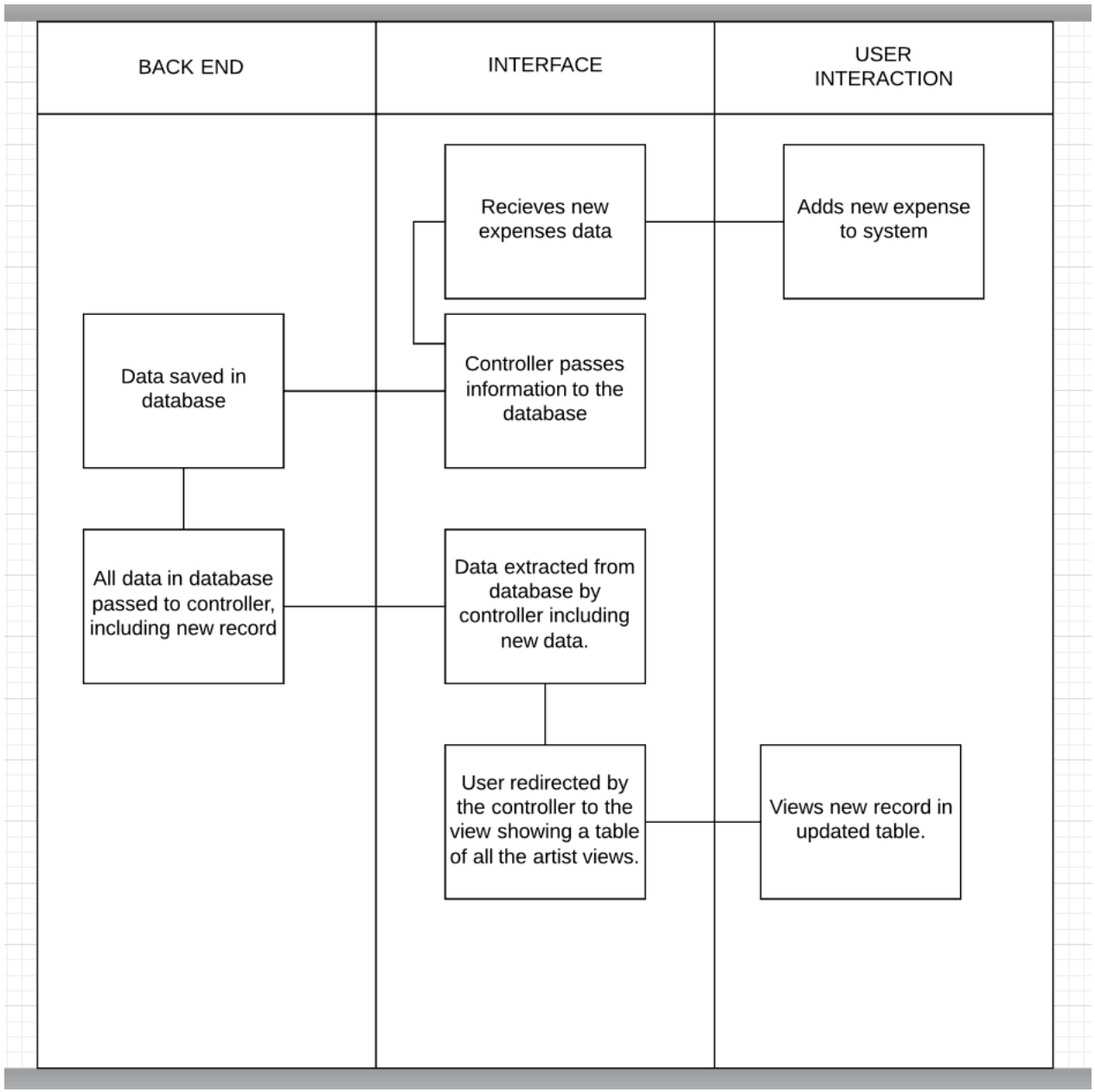
Unit	Ref.	Evidence	Done
A & D	A.D 1	A Use Case Diagram	Y
			
A & D	A.D 2	A Class diagram.	Y



A & D	A.D 3	An Object diagram.	Y

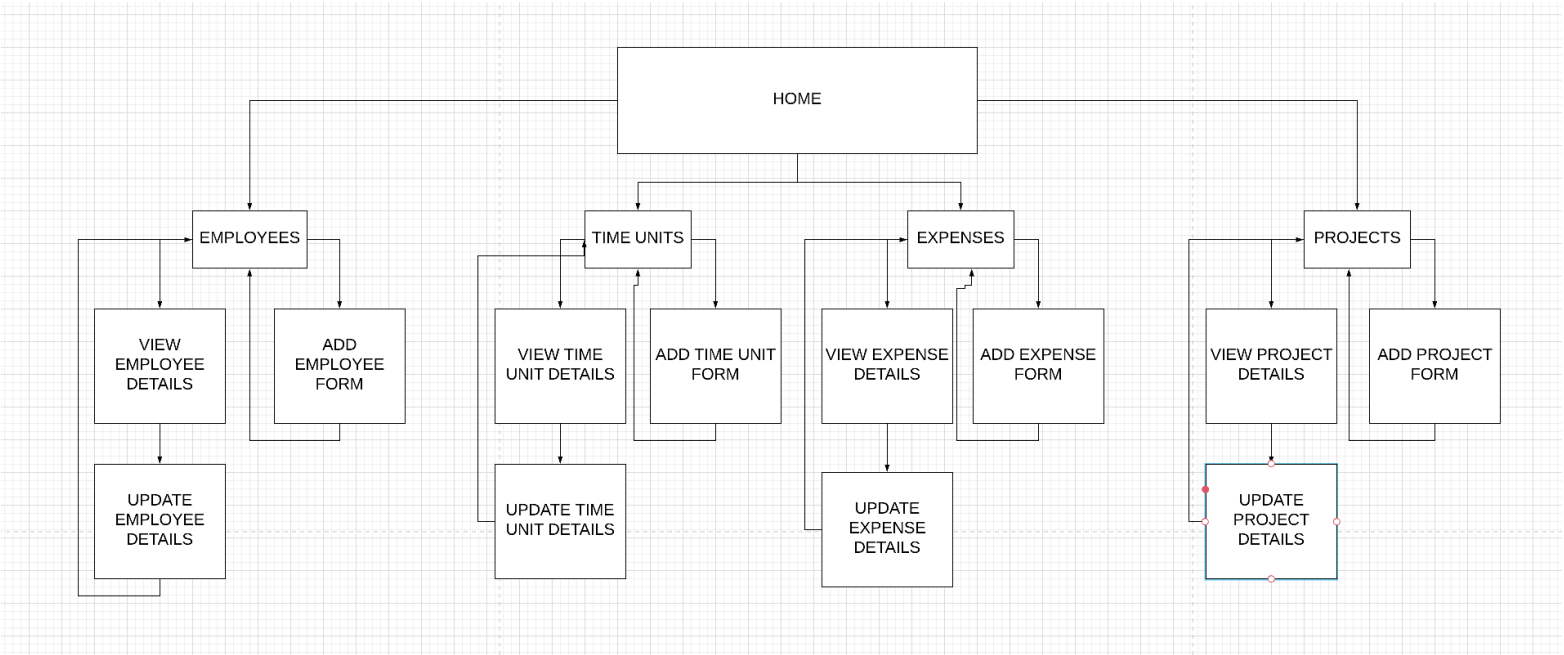


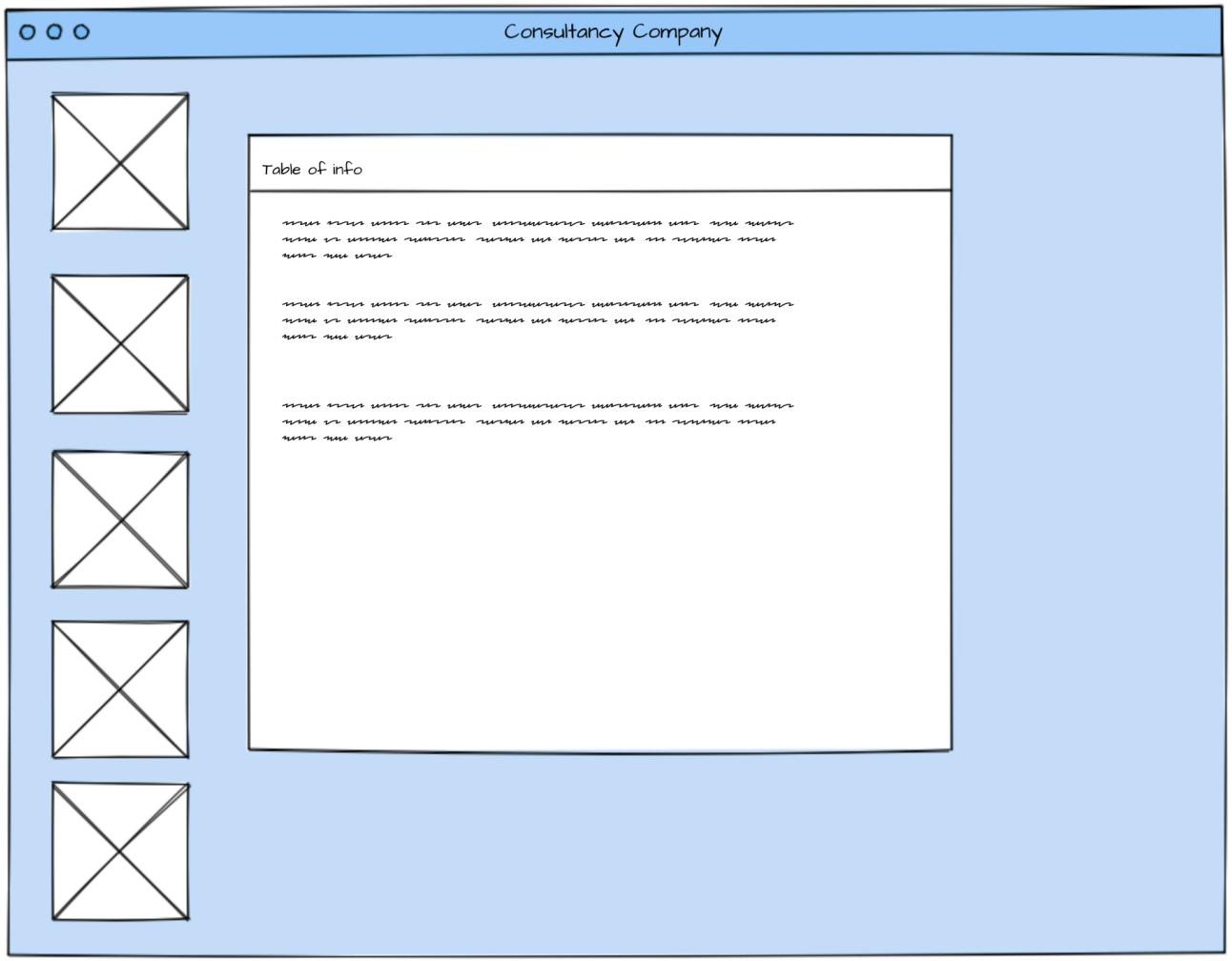
A & D	A.D 4	An Activity Diagram	Y


BACK END
INTERFACE
USER INTERACTION
Recieves new expenses data
Adds new expense to system
Data saved in database
Controller passes information to the database
All data in database passed to controller, including new record
Data extracted from database by controller including new data.
User redirected by the controller to the view showing a table of all the artist views.
Views new record in updated table.

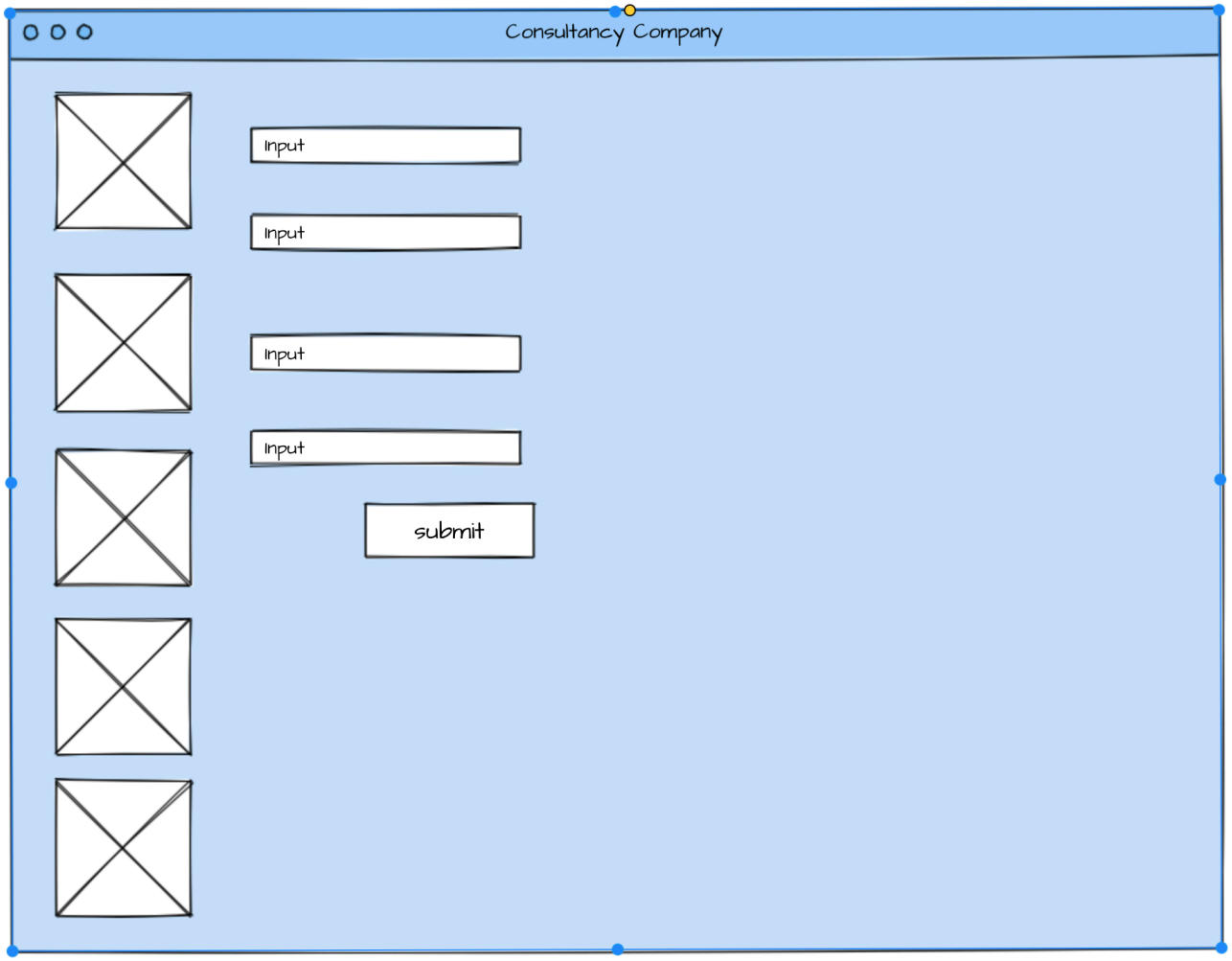
A & D	A.D 6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	Y

TOPIC	POSSIBLE EFFECT OF CONSTRAINT ON PRODUCT	SOLUTION
Hardware and software platforms	Product needs to be useable across a range of devises, such as a tablet, or mobile phone.	Webapp will be targeted at a laptop or desktop user, so not primary constraint. However test readability using the google chrome developer tool to view interface across a range of viewer platforms.
Performance requirements	User connection speeds to web app need have reasonable performance.	Use google chrome developer tools to test the web app, to ensure that waterfall charts remain low.
Persistent storage and transactions	Data loss from the web app following user interaction.	Test CRUD functions to ensure working. Use PostGRES database and Sinatra to host webapp.
Usability	Front end needs to be accessible to additional requirement users.	Use good html semantics to ensure readability for accessibility.
Budgets	No budget so requirement to use free software.	Utilise software including ATOM for code editing, Sinatra an open source web-framework and open source UX design tools at the planning stage.
Time limitations	Could lead to restriction on additional specs outside of the minimum viable product(MVP). Little room for error in weaker coding aspects, i.e. front end. Room room for data loss.	Use test driven development to focus on meeting the functionality requirements of the MVP within the allotted time. Focus design on simplicity to ensure viable product deliverable within week long time frame. Maintain good GIT practice including regular GIT commits and utilise GITHUB regularly.

P	P 5	Create a user sitemap.	Y
			
P	P 6	Produce two wireframe designs.	Y


Consultancy Company
Table of info

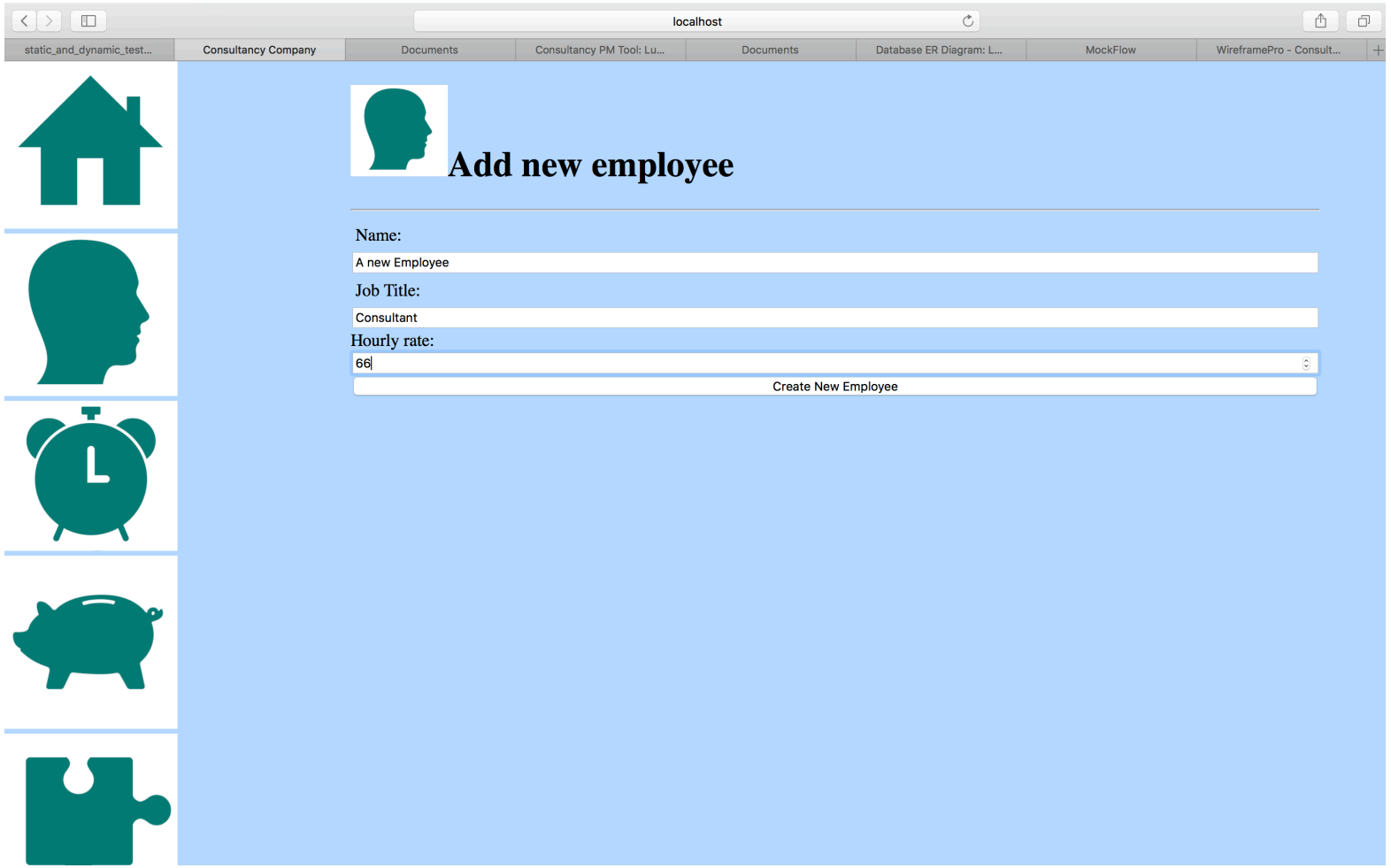
Week 5

			
P	P 10	Take a screenshot of an example of pseudocode for a function.	Y

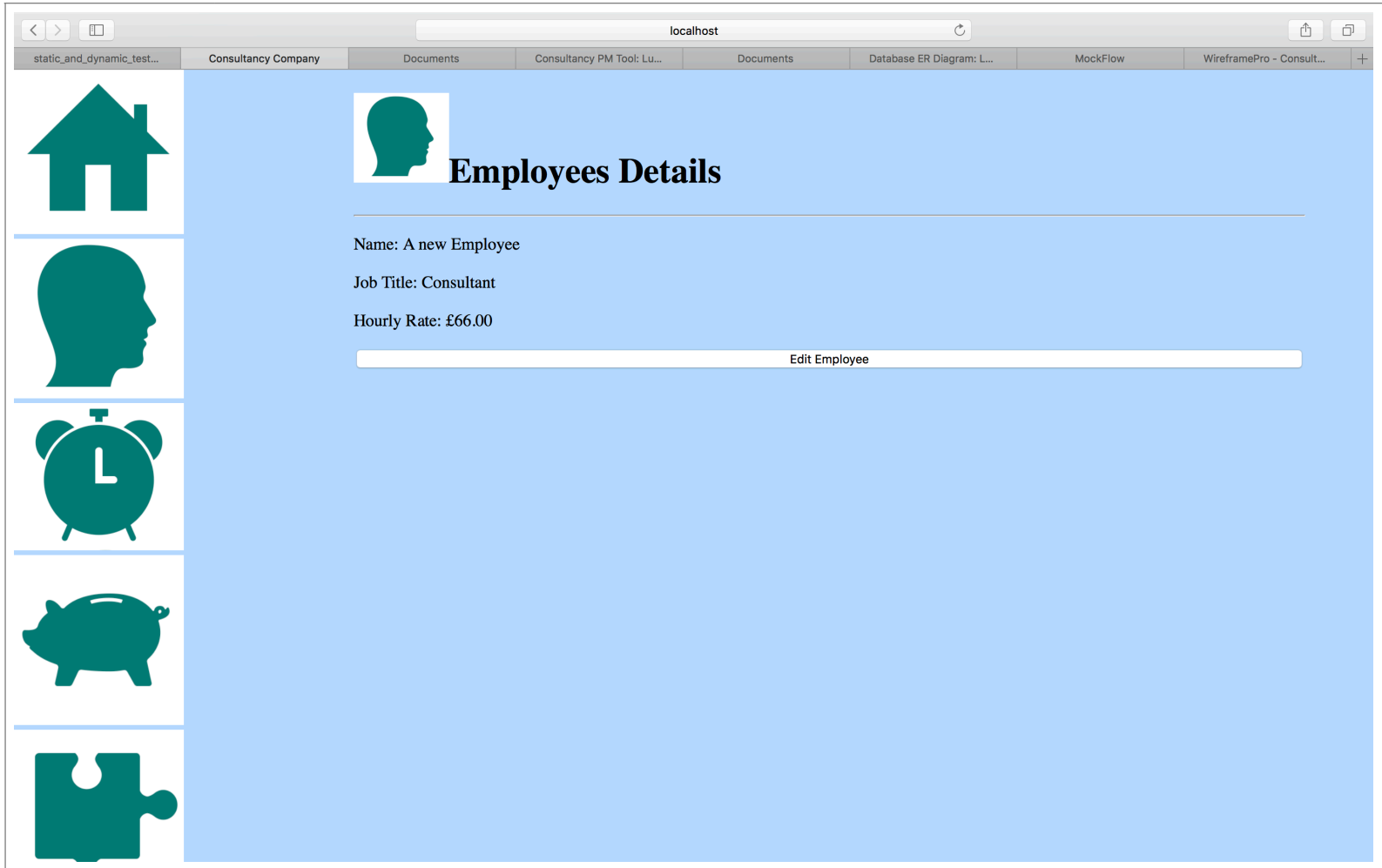
```ruby
#extract time_units from database where the project
id matches. Use the sql runner and convert the
database hash like object to a ruby object to extract
the first item in the list.

def find_time_by_id(id)
  sql = "SELECT * FROM time_units WHERE id = $1"
  values = [id]
  result = SqlRunner.run(sql, values)
  @id = result[0]["id"].to_i
  result_hash = result[0]  #convert here from
  database object to a ruby object
  return TimeUnit.new(result_hash)
end
```

The Psuedocode is greyed out and starts with a hash.

P	P 13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way	Y



Above the user is inputting the employee details.
Below shows the employee details in the system.


localhost
static_and_dynamic_test...
Consultancy Company
Documents
Consultancy PM Tool: Lu...
Documents
Database ER Diagram: L...
MockFlow
WireframePro - Consult...
Employees Details
Name: A new Employee
Job Title: Consultant
Hourly Rate: £66.00
Edit Employee

P	P 14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved	Y

```ruby
@project1 = Project.new(
  {
    "name" => "jupiter",
    "client" => "FDDC development",
    "time_budget" => 90000.00,
    "expenses_budget" => 9000.00,
    "project_status" => "Proposal Submitted"
  }
)
@project1.save
```

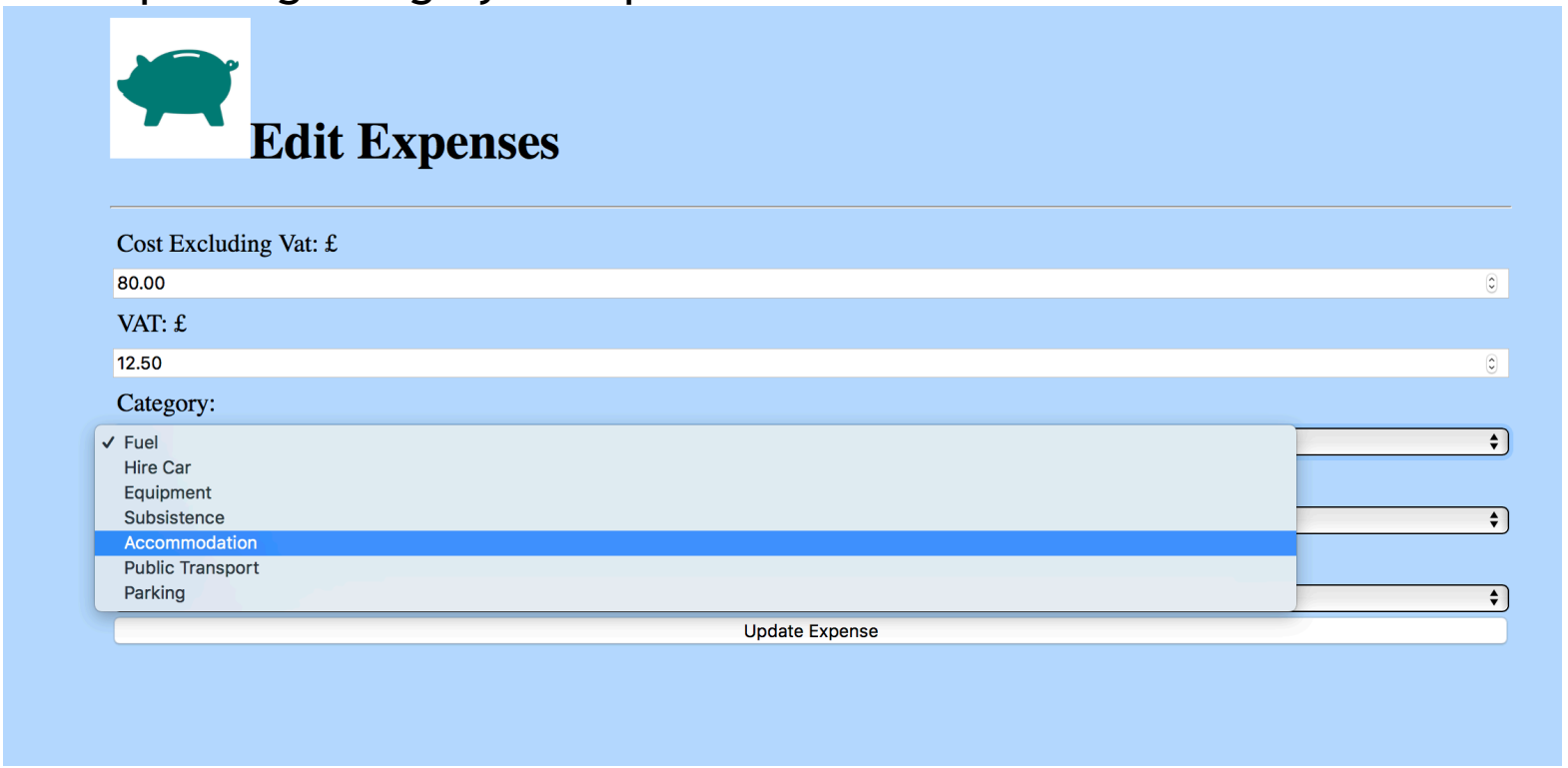
Seeding the database above - data in the database below.

ruby | psql

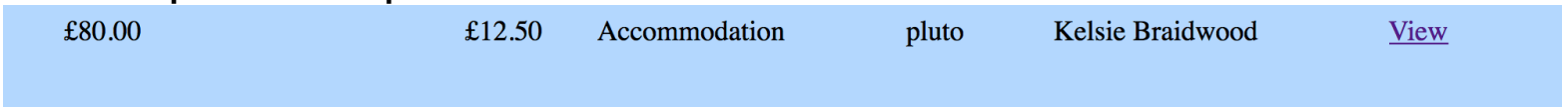
```
 id |   name   |            client             | time_budget | expenses_budge
t |   project_status
----+---------+-------------------------------+-------------+--------------
--+--------------------
  1 | jupiter | FDDC development              |       90000 |            900
0 | Proposal Submitted
  2 | mars    | Scottish Energy Systems       |      100000 |           1000
0 | Complete
  3 | pluto   | Finacial Services Company     |      450000 |           4500
0 | Proposal
  4 | earth   | Assest Management Systems     |       20000 |            200
0 | Open
  5 | saturn  | Hotel Management Services     |        1000 |             10
0 | On Hold
  6 | uranus  | Investment Services           |       79000 |            790
0 | Proposal
  7 | mercury | Scottish LLP                  |      190000 |            190
0 | Open
  8 | venus   | McDonald Construction England |      250000 |            250
```

P	P 15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program	Y

User updating category of expense.



Saved updated expense show.



	P	P 18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	Y

expense_spec.rb

```ruby
require('minitest/autorun')

require_relative("../expense.rb")


class TestExpense < MiniTest::Test
  def setup()
    @expense1 = Expense.new(
        {
          "cost_excluding_vat" => 10.50,
          "vat" => 1.20,
          "category" => "subsistence"
        }
      )
  end
  def test_get_cost_excluding_vat()
    assert_equal(11.50, @expense1.cost_excluding_vat)
  end
```

Test code.

```
➜  project_week1 git:(master) ✗ ruby models/specs/expense_spec.rb
Run options: --seed 62992

# Running:

.F.

Finished in 0.001406s, 2133.7127 runs/s, 2133.7127 assertions/s.

  1) Failure:
TestExpense#test_get_cost_excluding_vat [models/specs/expense_spec.rb:17]:
Expected: 11.5
  Actual: 10.5

3 runs, 3 assertions, 1 failures, 0 errors, 0 skips
➜  project_week1 git:(master) ✗
```

Test failing

```
➜  project_week1 git:(master) ✗ ruby models/specs/exp
ense_spec.rb
Run options: --seed 31289

# Running:

...

Finished in 0.001026s, 2923.9769 runs/s, 2923.9769 as
sertions/s.

3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
➜  project_week1 git:(master) ✗
```

Test passing

expense_spec.rb

```ruby
require('minitest/autorun')

require_relative("../expense.rb")


class TestExpense < MiniTest::Test
  def setup()
    @expense1 = Expense.new(
        {
          "cost_excluding_vat" => 10.50,
          "vat" => 1.20,
          "category" => "subsistence"
        }
      )
  end
  def test_get_cost_excluding_vat()
    assert_equal(10.50, @expense1.cost_excluding_vat)
  end
```

Code with test passing

	Unit	Ref.	Evidence	Done
	I & T	I.T 7	Demonstrate the use of Polymorphism in a program.	Y

Here is the parent class Employee, which is abstract.

Below, Manager extends Employee, but Employee is versatile and polymorphic as you could have a child class like Engineer which also extends Employee.

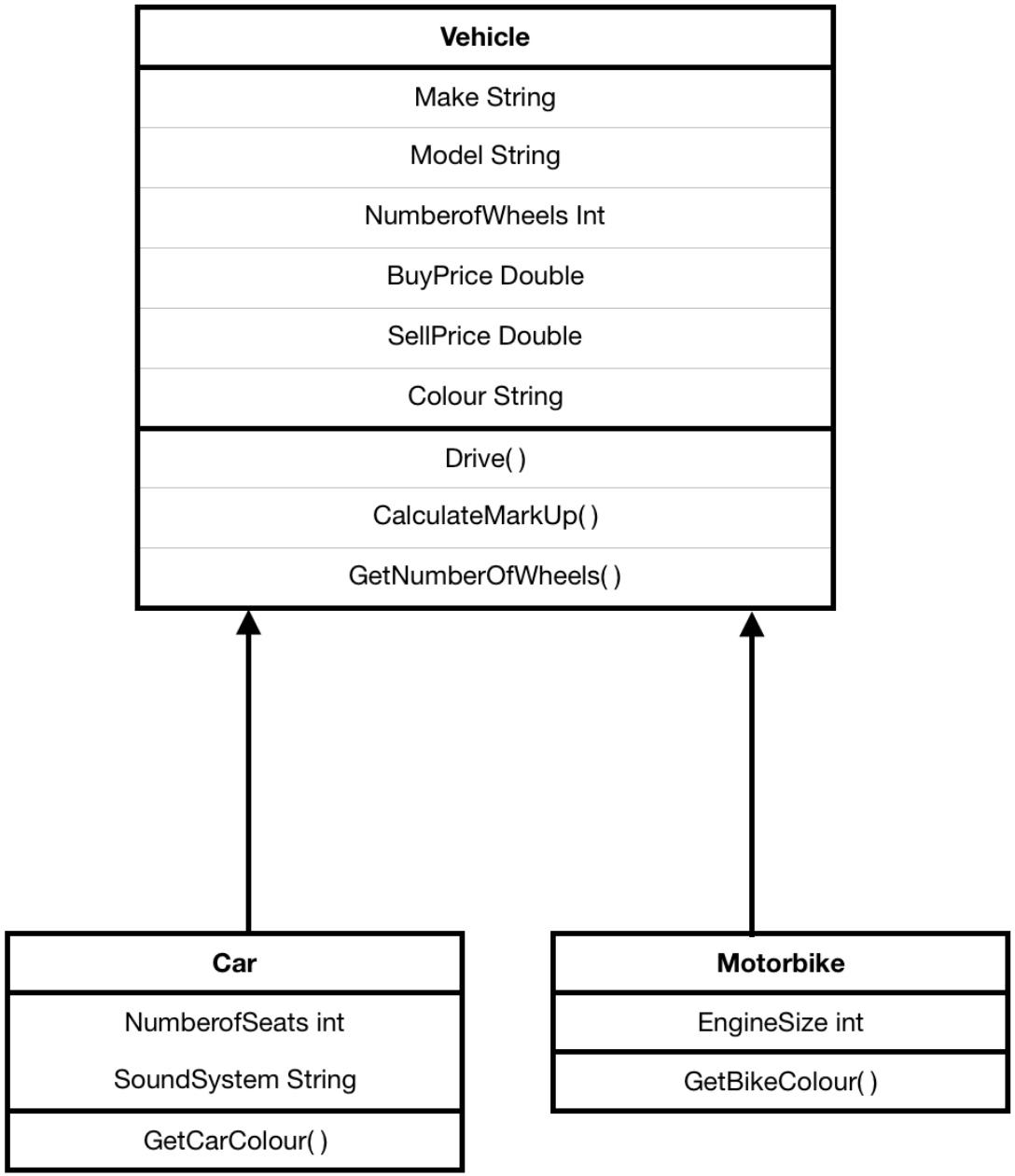
Employee.java

```java
package models;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Employee {

    private int id;
    private String firstName;
    private String lastName;
    private int salary;
    private Department department;

    public Employee() {
    }
```

```java
package models;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;


@Entity
```

A & D	A.D 5	An Inheritance Diagram	Y

Here Motorbike and Car inherit from Vehicle.



I & T	I.T 1	Take a screenshot of an example of encapsulation in a program.	Y
The below demonstrates encapsulation because the departments id, title and set of employees are all set to the key word Private. They are accessed from outside of the class by the use of a getter and setter method instead of accessing the properties of the class directly.			

```java
public class Department {


    private int id;
    private String title;
    private Set<Employee> employees;

    public Department() {}

    public Department(String title) {
        this.title = title;

    }

    @Column(name="title")
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
```

I & T	I.T 2	Take a screenshot of the use of Inheritance in a program. Take screenshots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.	Y
Here is the abstract Item class. Below, book inherits from Item.			

Week 10

Item.java

```java
package inventory;

import behaviours.ISell;

public abstract class Item implements ISell {
    private String name;
    private double stock_value;
    private double sell_price;

    public Item(String name, double stock_value, double sell_price) {
        this.name = name;
        this.stock_value = stock_value;
        this.sell_price = sell_price;
    }

    public String getName() {
        return name;
    }

    public double getStock_value() {
        return stock_value;
    }
```

Book.java

```java
package inventory.misc_items;

import behaviours.ISell;
import inventory.Item;

public class Book extends Item implements ISell{
    private String author;
    private String title;

    public Book(String name, double stock_value, double sell_price,
    String author, String title) {
        super(name, stock_value, sell_price);
        this.author = author;
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public String getTitle() {
        return title;
    }
}

```

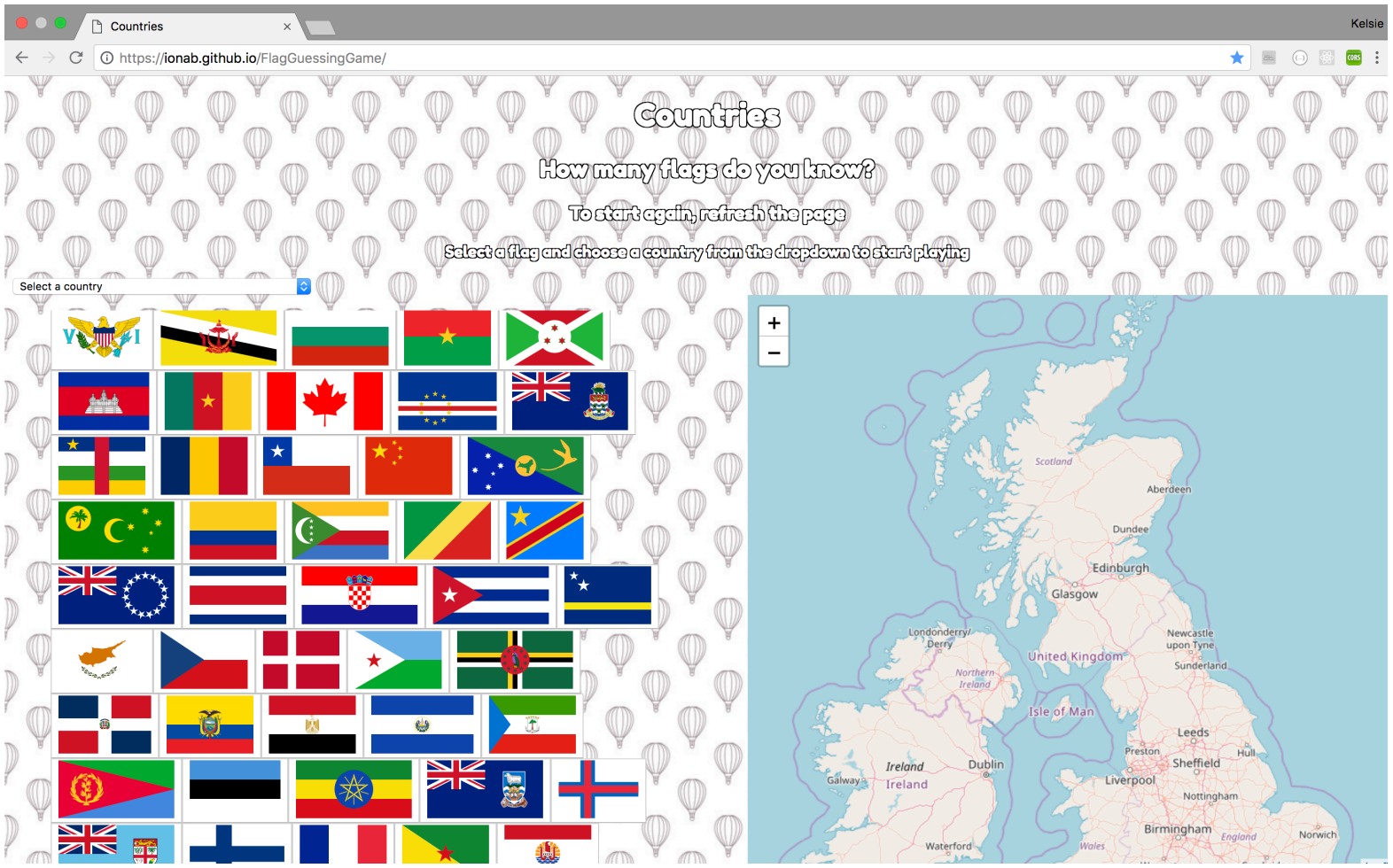
```
book = new Book("test book", 6.00, 7.00, "test author", "test
title");
```

A book object.

```
import behaviours.ISell;
import inventory.Item;

public class Book extends Item implements ISell{
    private String author;
    private String title;

    public Book(String name, double stock_value, double sell_price, String author, String title) {
        super(name, stock_value, sell_price);
        this.author = author;
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public String getTitle() {
        return title;
    }


    public Double calculateProfit(){
        return this.sell_price - this.stock_value;
    }
}
```

Here the calculateProfit method uses the sell_price and stock_value which are inherent properties from the parent class.

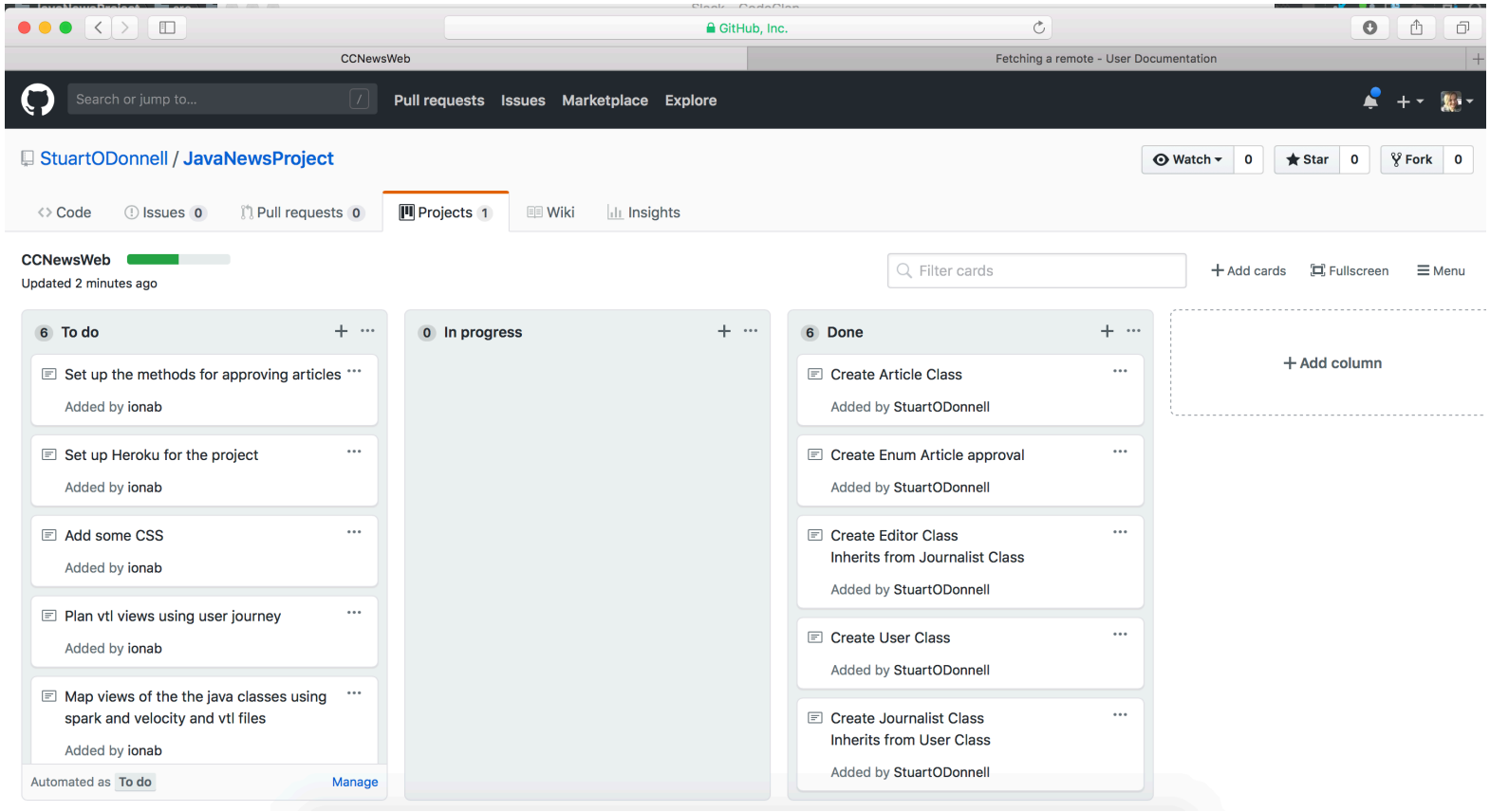
P	P 11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	Y

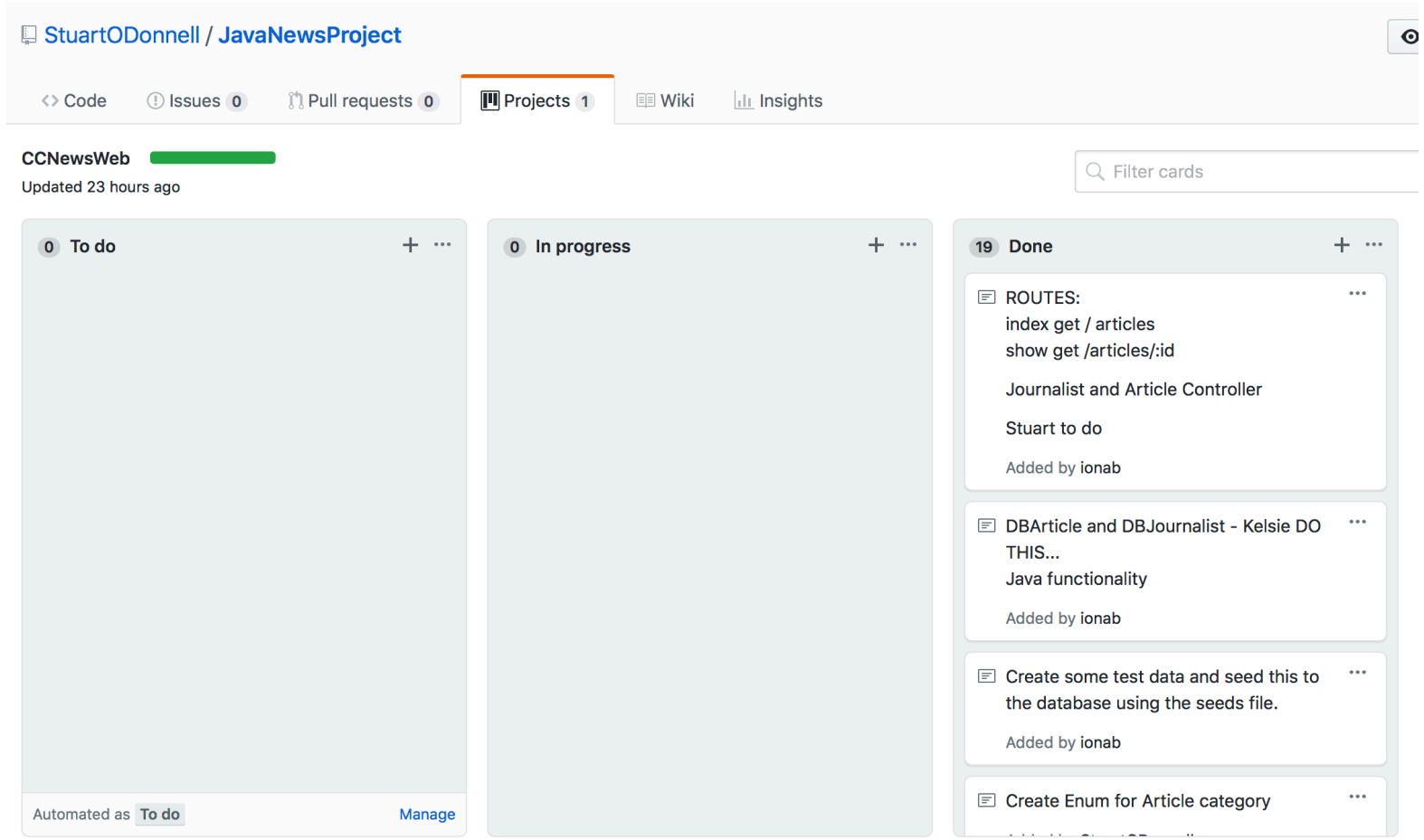


https://github.com/ionab/FlagGuessingGame

P	P 12	Take screenshots or photos of your planning and the different stages of development to show changes.	Y

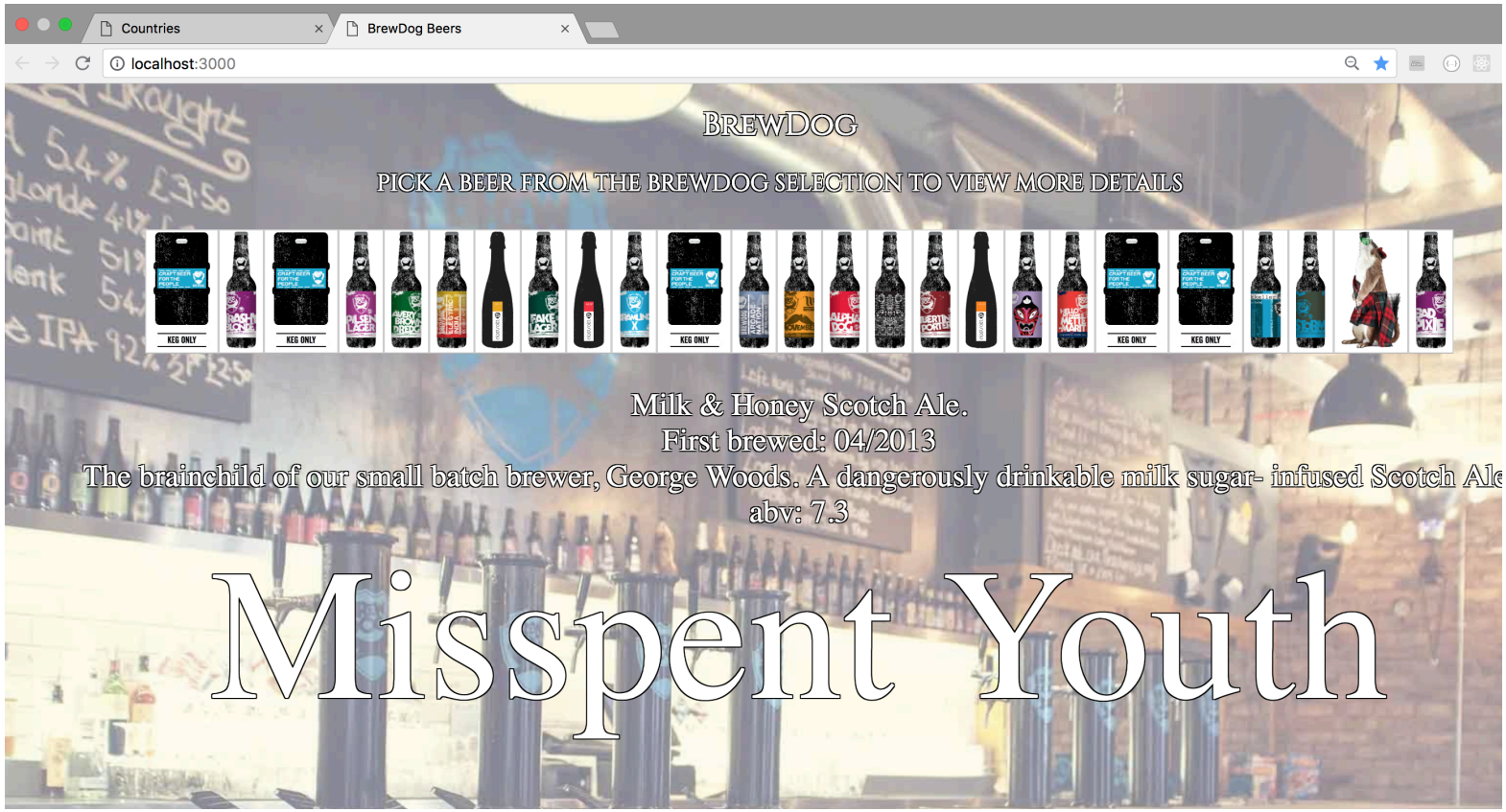
GitHub Project Page during project.





Project complete.

	Unit	Ref.	Evidence	Done

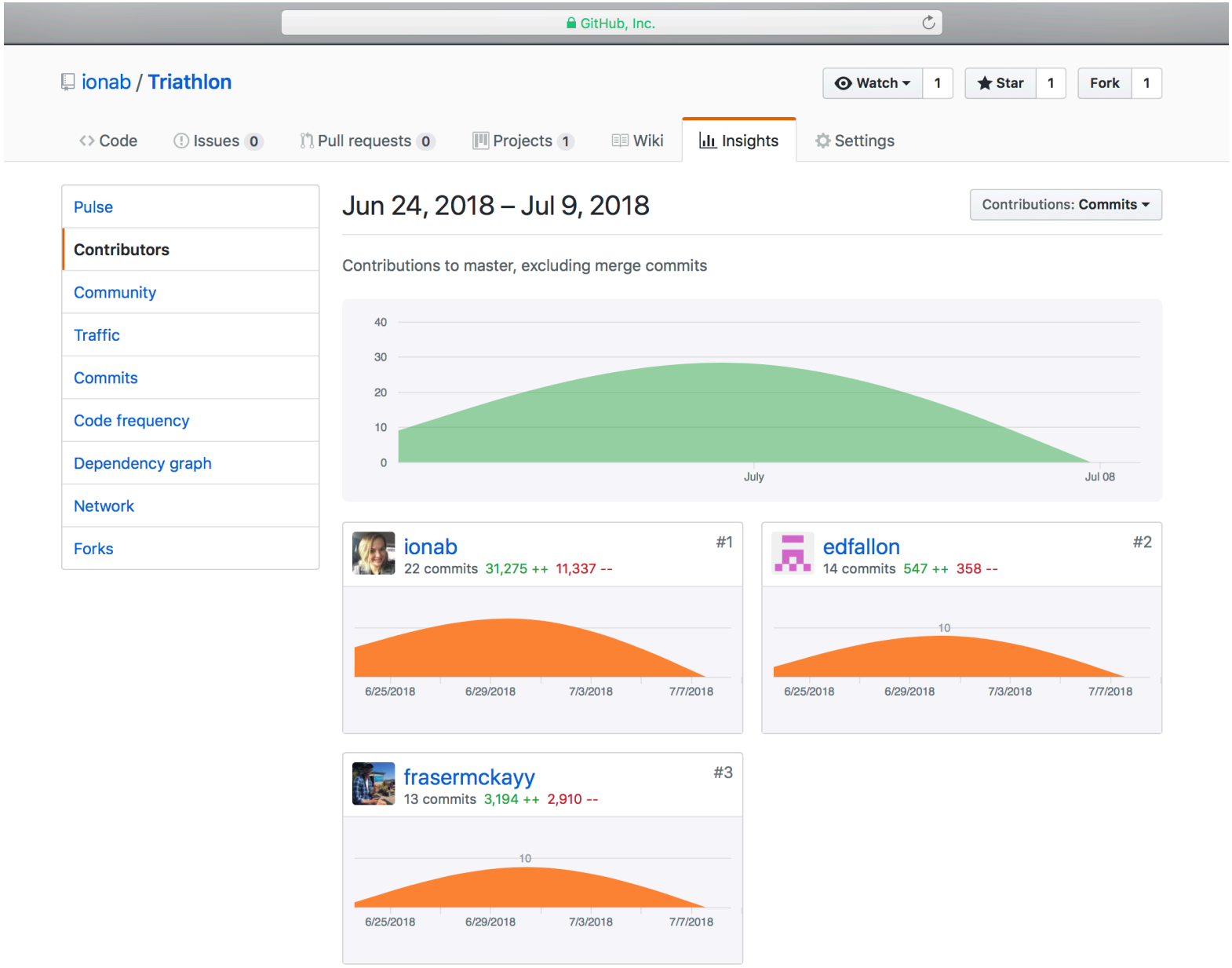
Week 12	I & T		Unit, integration and acceptance testing task B	Y
	P	P 16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	Y
	 BrewDog API in use in app.		

index.js

```
const app = function(){
 const url = " https://api.punkapi.com/v2/beers";
 makeRequest(url, requestComplete);
}

const makeRequest = function(url, callback){
 // back in the day was XML but now returns JSON, still called X
 const request = new XMLHttpRequest();
 request.open("GET", url);
 request.addEventListener("load", callback);
 request.send();
}

const requestComplete = function(){
   if(this.status !== 200) return;
   const beers = JSON.parse(this.response);
   console.log(this);
   populateList(beers);


}
```

This is the code making the request to the api and populating the drop down with the JSON data returned.

	Unit	Ref.	Evidence	Done
	P	P 1	Take a screenshot of the contributor’s page on Github from your group project to show the team you worked with.	Y



P	P 2	Take a screenshot of the project brief from your group project.	Y

Sports Dashboard

Sports fans want to be able to view relevant sporting events on a dashboard. With a sport of your choice, use an existing API or create a new API to display information about fixtures, news and travel information for events.

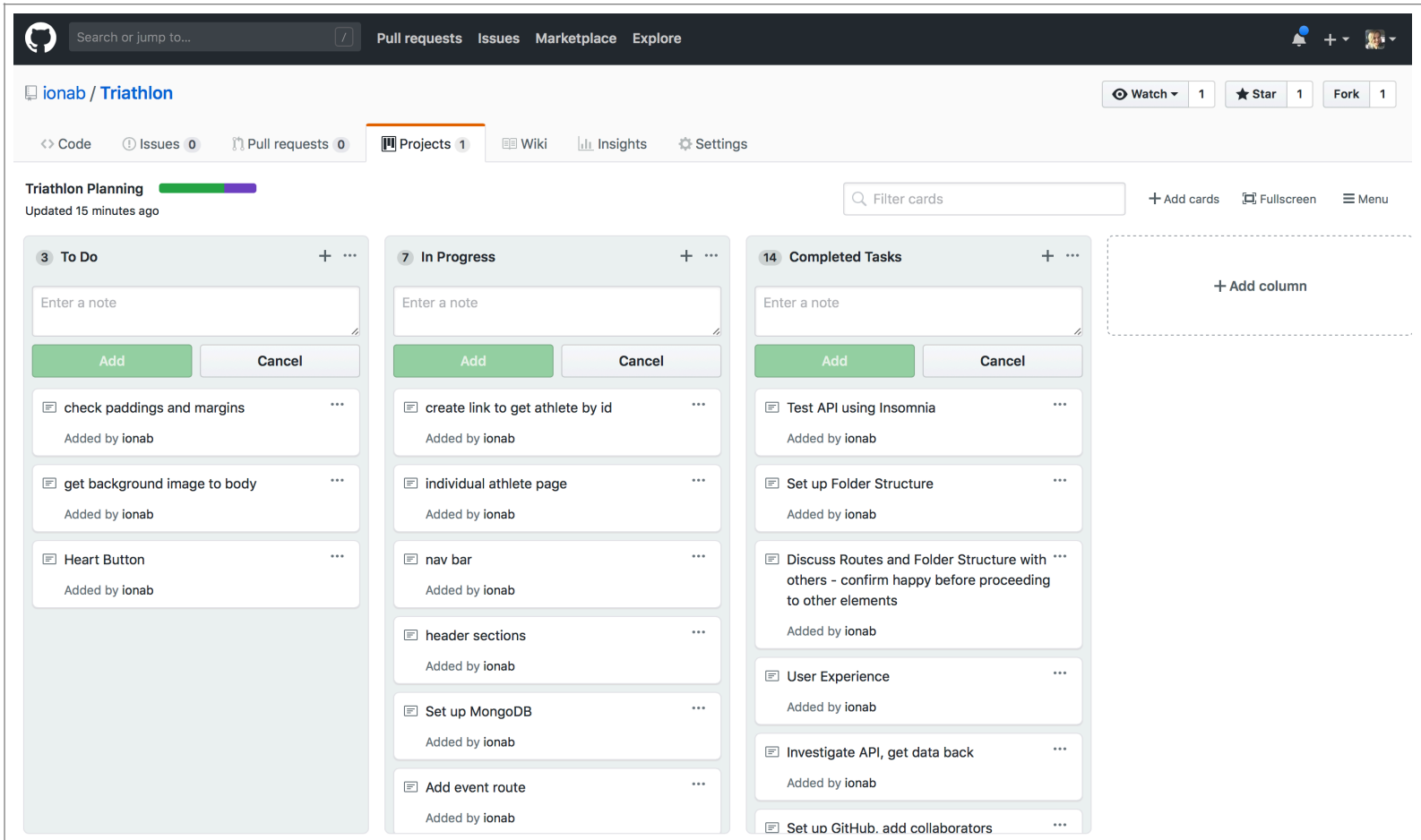
Possible APIs to use:

- UFC: http://ufc-data-api.ufc.com/api/v1/us
- Football: http://api.football-data.org/index
- Triathalon: https://developers.triathlon.org/docs

MVP

- Display upcoming events on a map
- Display results and ranking of players/teams
- Allow users to add events to a favourites list

P	P 3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	Y



MOSCOW board above, user journey below.

Tom wants to find out about an upcoming event details and athletes in Australia.

Start Step 1 Loads up the webpage.	Step 2 Clicks on the events tab in the navigation bar.	Step 3 Events page loads with a picture of the map. Tom, views the map.	Step 4 Tom, selects the region he wants to find out more information about events on(Australia).
Gets the access to the localhost:3000 And the page loads on the landing page.	Interaction with navigation button. Link to events page loads with the new route.	Page loads with the map displaying. The map will display and be interactive.	Button interaction when link is clicked. Map re located and the fly to method operates.
Step 5 Clicks on one of the events to find out more information. He then wants to find out which athletes will be competing.	Step 6 Clicks on the event on the map. Reads the information on the athletes competing.	Step 6 Tom finds his information and leaves the site. End	
Button interaction with map. Get request of data from the database to find all the information on the athletes of that event.	Button interaction occurs. Post request to display all the relevant information.	Page is fully loaded and at final stage without any more requests to the database or server.	

- Tom
- 33
- Born in Scotland.
- Lives in Glasgow.

Behaviours and Actions

- Likes to go for runs in the morning.
- Puts in long hours in the office.
- Uses social media apps frequently.
- Comfortable speaker and engages well with others.
- Likes to plan everything out.
- Very organised.

Demographic

- FullTime Job.
- Lives with girlfriend.
- Regular Income.
- Has a dog.
- Enthusiastic Triathlete.

Needs and Requirements

- Easy to navigate system.
- Information on his favourite teams and athletes.
- Where the current races are going on and the times.
- Results and current standings of all event participants.

User profiling.

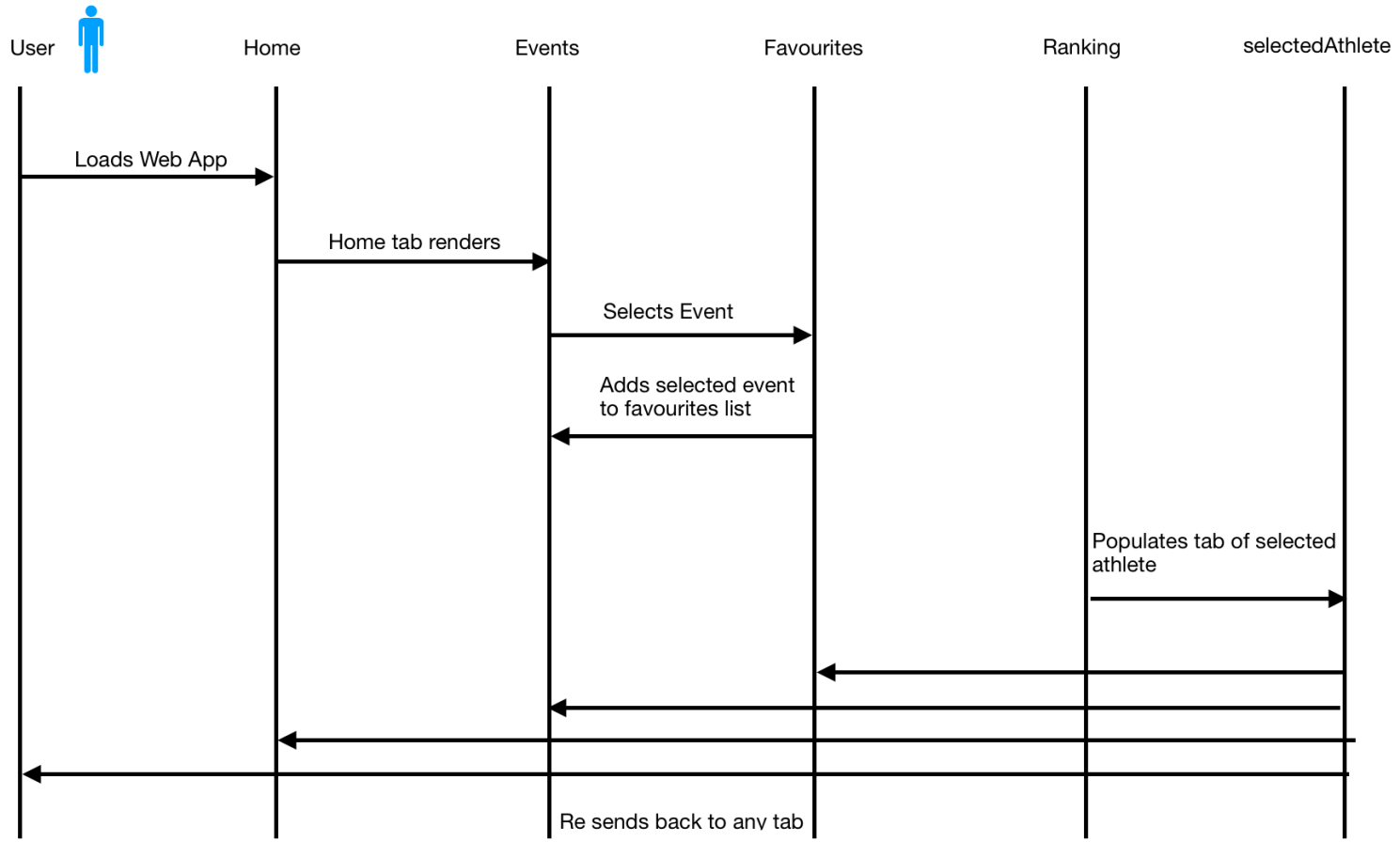
P	P 4	Write an acceptance criteria and test plan.	Y

Week 14				

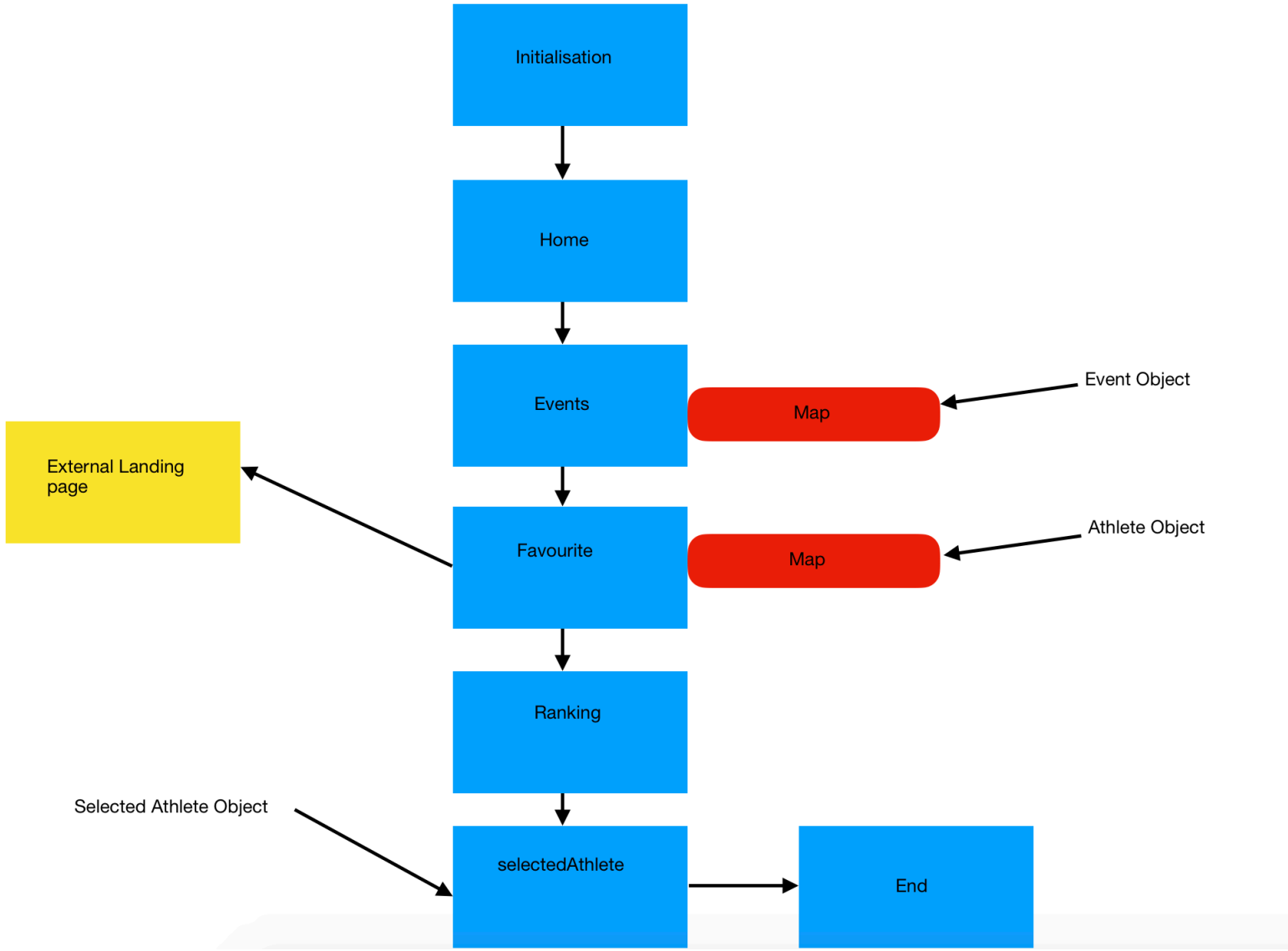
Acceptance Criteria	Expected Result/output	Pass/fail
A user is able to access a list of events.	List of events is created with the load of the events page.	Pass
A user is able to navigate freely to all the routes	New route loads up when navigation is clicked	Pass
A user will be able to add a favourite event to a favourites list.	Once an event is selected it will add it to a favourites list	Pass
A user will be able to see a specific events rankings and placements.	When the events page is loaded a list of the results will display	Fail
A user will be able to see a list of athletes rankings for a specific event	When the specific event is loaded the rankings for all the participating athlete's will load.	Pass

P	P 7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	Y

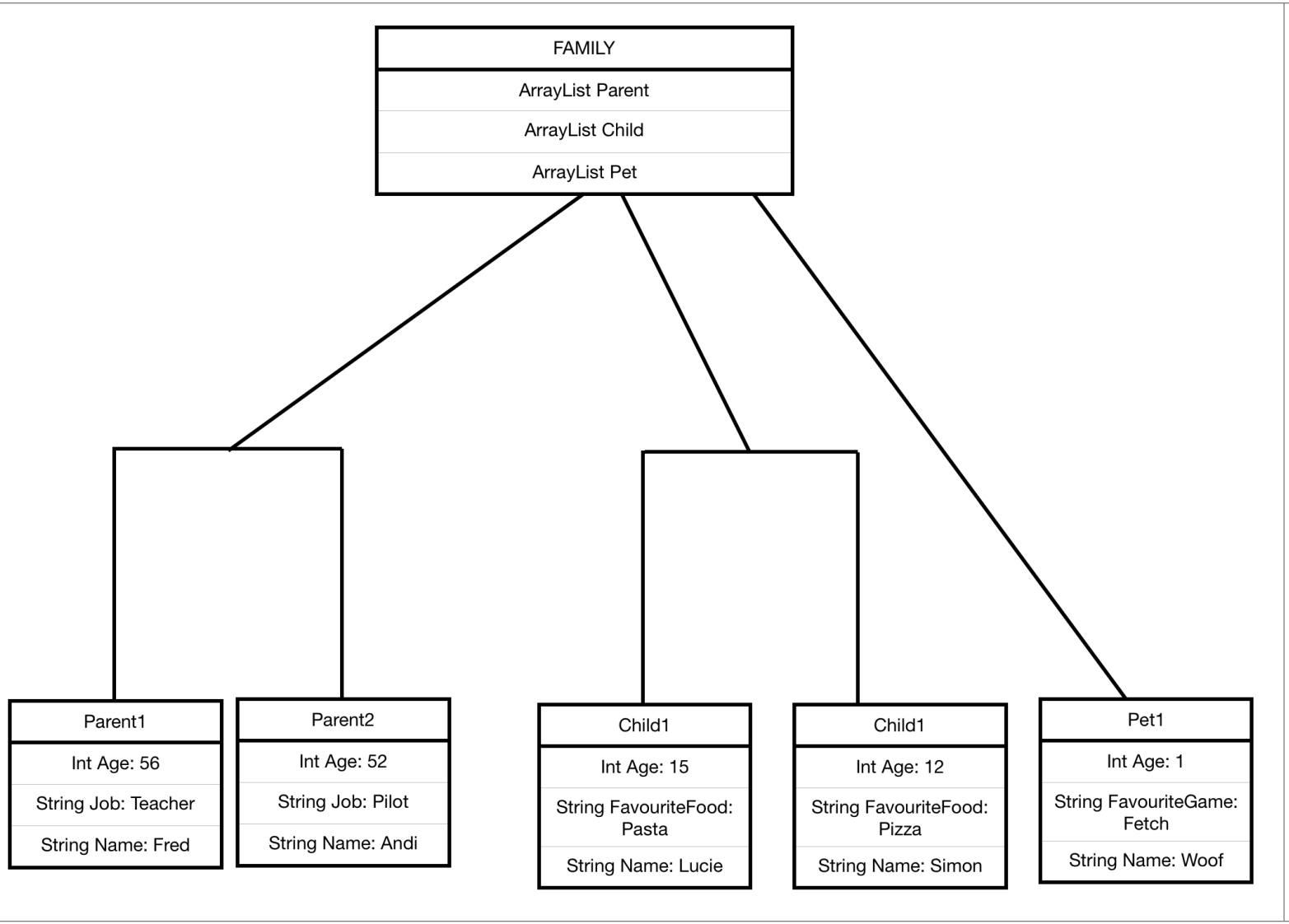
System interaction diagram.

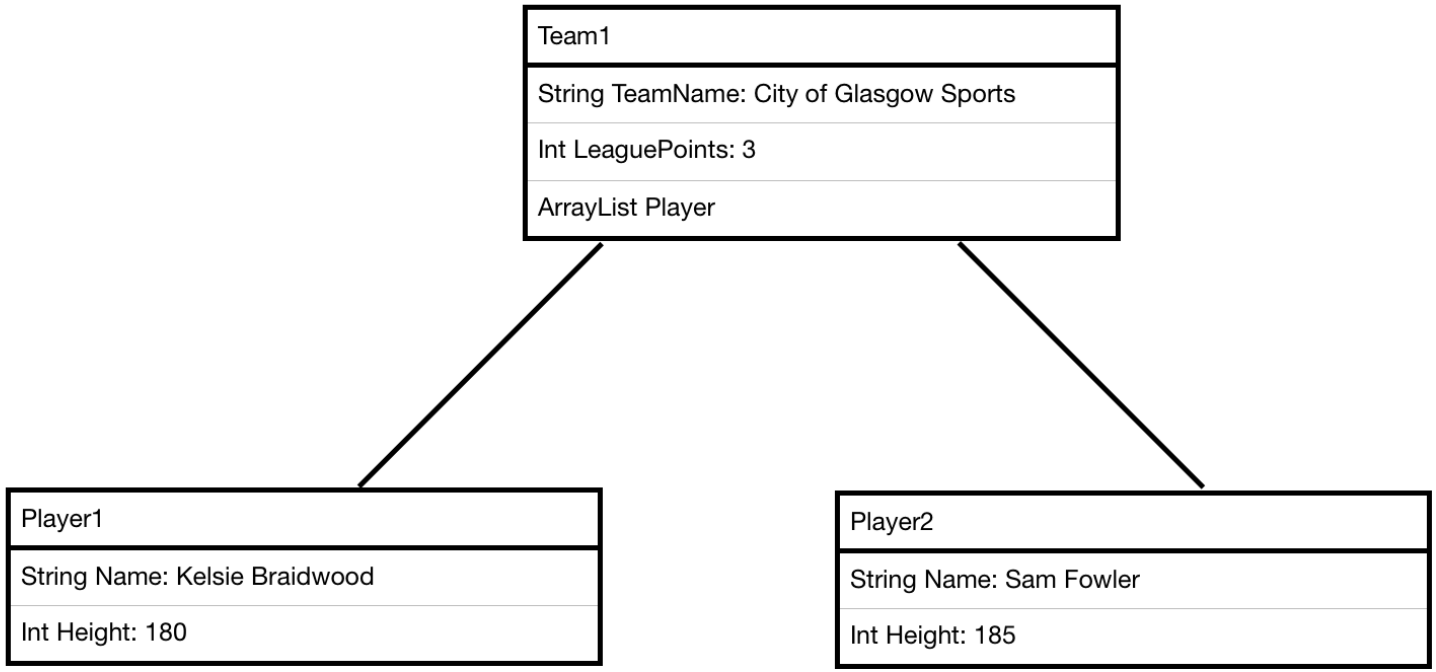


Collaboration diagram.

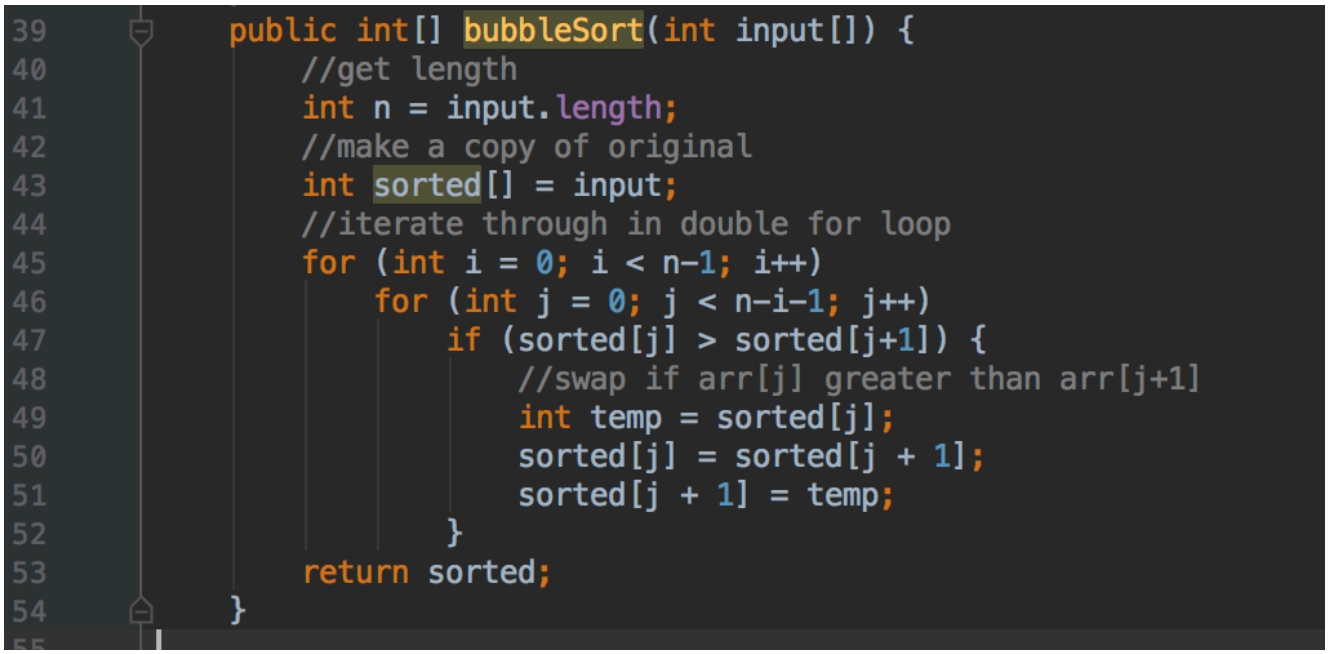
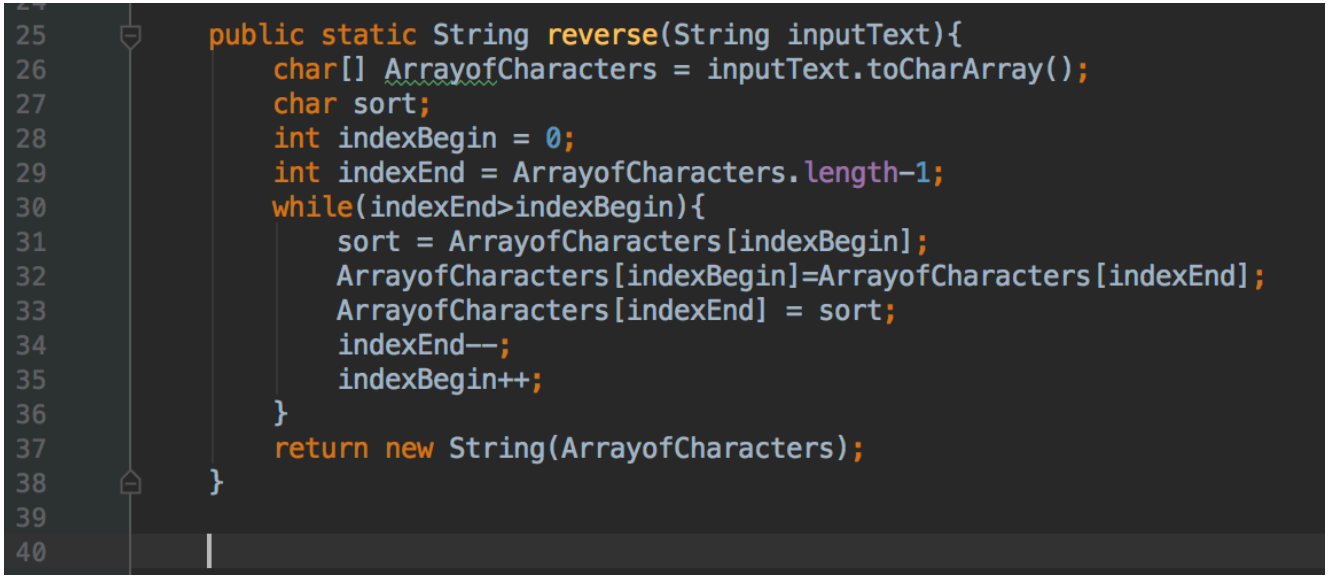


P	P 8	Produce two object diagrams.	Y


FAMILY
ArrayList Parent
ArrayList Child
ArrayList Pet
Parent1
Int Age: 56
String Job: Teacher
String Name: Fred
Parent2
Int Age: 52
String Job: Pilot
String Name: Andi
Child1
Int Age: 15
String FavouriteFood: Pasta
String Name: Lucie
Child1
Int Age: 12
String FavouriteFood: Pizza
String Name: Simon
Pet1
Int Age: 1
String FavouriteGame: Fetch
String Name: Woof



P	P 9	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.	Y

		 Bubble sort chosen to sort an array of integers into the smallest first.	
		 Algorithm takes a string and out puts the string in reverse order.	
P	P 17	Produce a bug tracking report	Y

```
39      public int[] bubbleSort(int input[]) {
40          //get length
41          int n = input.length;
42          //make a copy of original
43          int sorted[] = input;
44          //iterate through in double for loop
45          for (int i = 0; i < n-1; i++)
46              for (int j = 0; j < n-i-1; j++)
47                  if (sorted[j] > sorted[j+1]) {
48                      //swap if arr[j] greater than arr[j+1]
49                      int temp = sorted[j];
50                      sorted[j] = sorted[j + 1];
51                      sorted[j + 1] = temp;
52                  }
53          return sorted;
54      }
```

```
25      public static String reverse(String inputText){
26          char[] ArrayofCharacters = inputText.toCharArray();
27          char sort;
28          int indexBegin = 0;
29          int indexEnd = ArrayofCharacters.length-1;
30          while(indexEnd>indexBegin){
31              sort = ArrayofCharacters[indexBegin];
32              ArrayofCharacters[indexBegin]=ArrayofCharacters[indexEnd];
33              ArrayofCharacters[indexEnd] = sort;
34              indexEnd--;
35              indexBegin++;
36          }
37          return new String(ArrayofCharacters);
38      }
39
40
```

Getting data back from api	Fail	Added api key	Pass
Render map to page	Fail	Map given height and width	Pass
Set map over Europe	Fail	Add zoom level of 3.5	Pass
Marker shown on map	Fail	Added correct path to lat and long coordinates	Pass
Event flag displays in popup	Fail	Correct api path added	Pass