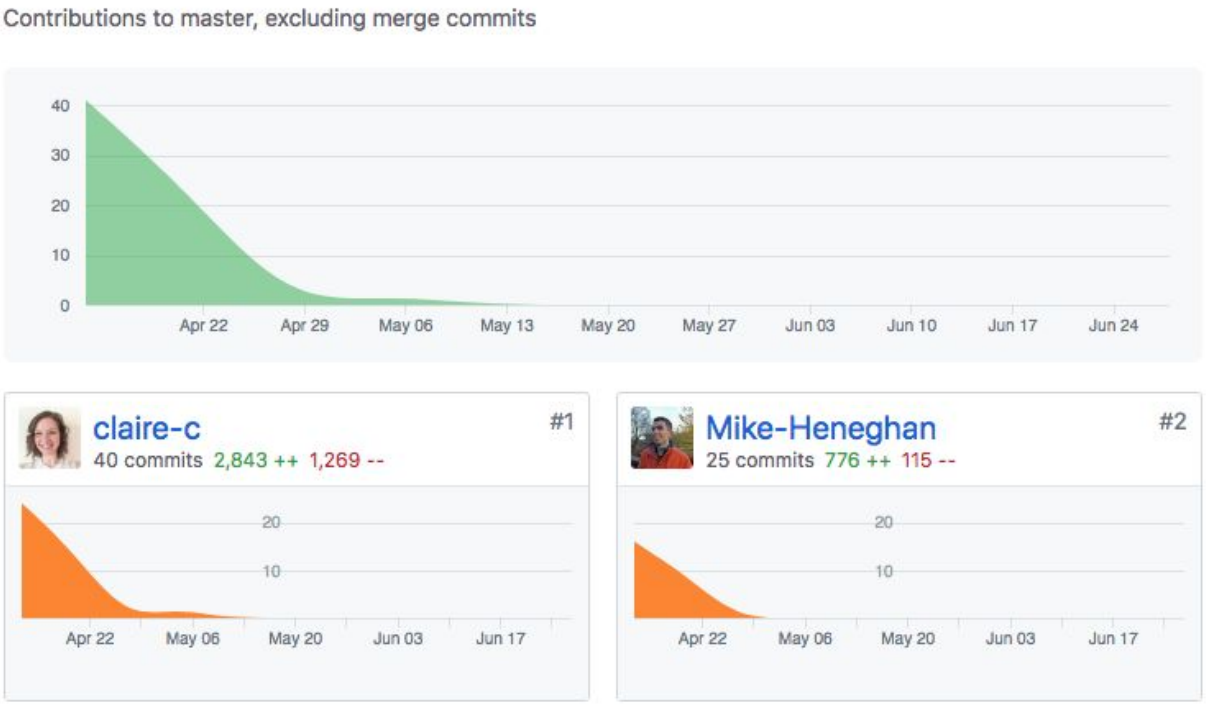
Claire Connachan
CodeClan Cohort - E20

Evidence for Project Unit

P1 - Take a screenshot of the contributor's page on GitHub from your group project to show the team you worked with.



P2 - Take a screenshot of the project brief from your group project.

Castle Finder!

A custom MVP to adapt the store finder app project to use maps to find the nearest castle for the user with an extension of adding education infromation.

MVP

- Search by town, postcode or street name.
- A list of castles and the details of the castles initial stored in a seeds file and loaded to a MongoDB database.
- A map marking the castles.

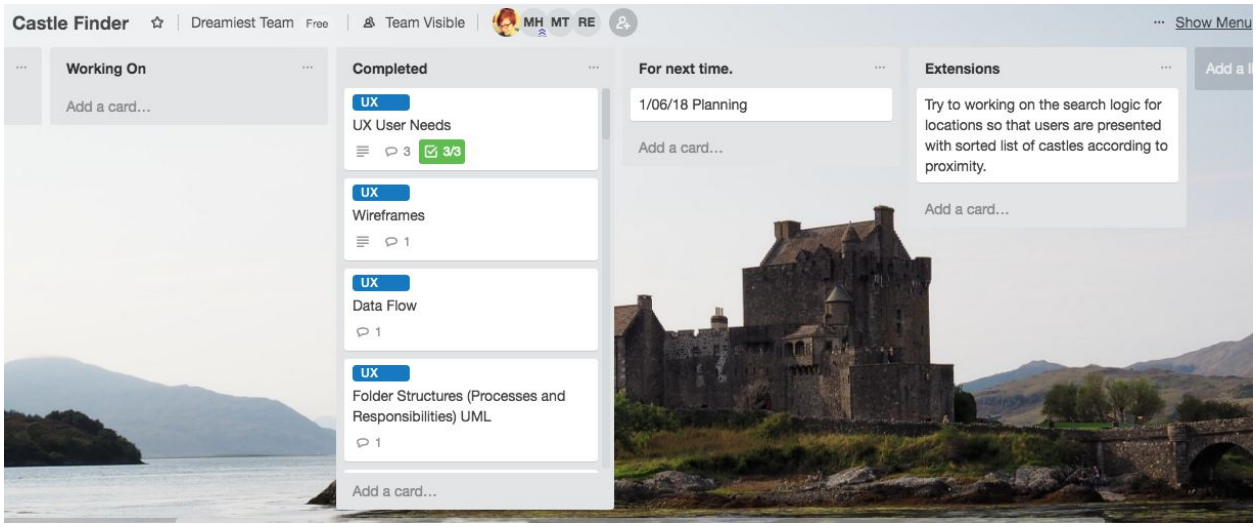
Extensions

- Addtional information for the castles
- Days by horse.
- Gamify the web app to include quizzes.

Useful Links

- Trello
- Leaflet JS

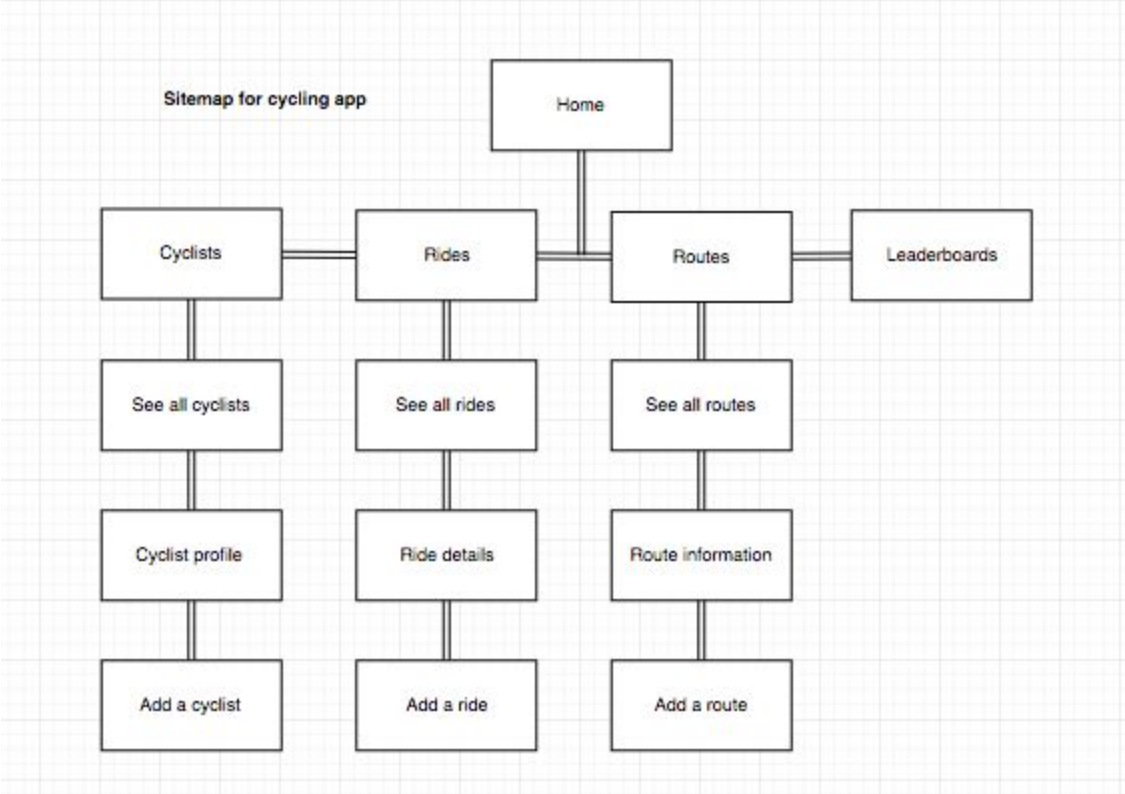
P3 - Provide a screenshot of the planning you completed during your group project, eg Trello MoSCoW board.



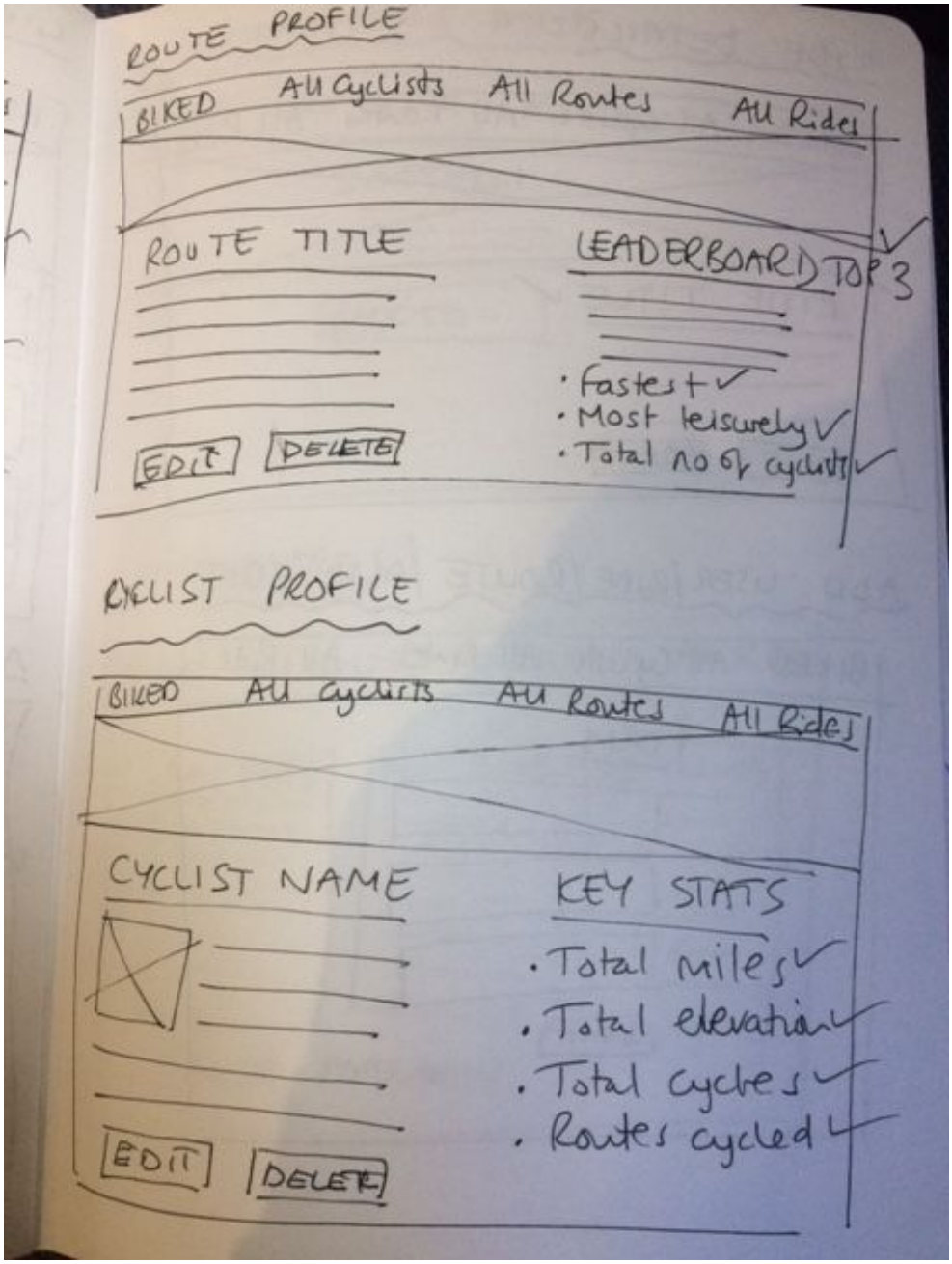
P4 - Write an acceptance criteria and test plan.

Acceptance Criteria	Expected Result/Output	Pass/Fail
Knight can use shield	Game class uses roll dice method on the Knight object, damage taken is 50% reduced due to use of shield.	Pass
Knight can collect treasure	Knight object calls addToInventory method and passes in treasure object. Treasure object is added to the knight's inventory property.	Pass
Knight can get value of treasure	Knight object calls getTotalTreasureValue method to return the total value of all treasure items in the inventory.	Pass
Knight can fight monsters.	Room object calls heroesFightMonster method, Knight object calls standandMove method on the monster. Either the Knight or monster will receive health point damage.	Pass
Knight can be healed	Cleric object calls beHealed method and passes in the Knight. Knight object's health points are increased.	Pass

P5 - Create a user sitemap.

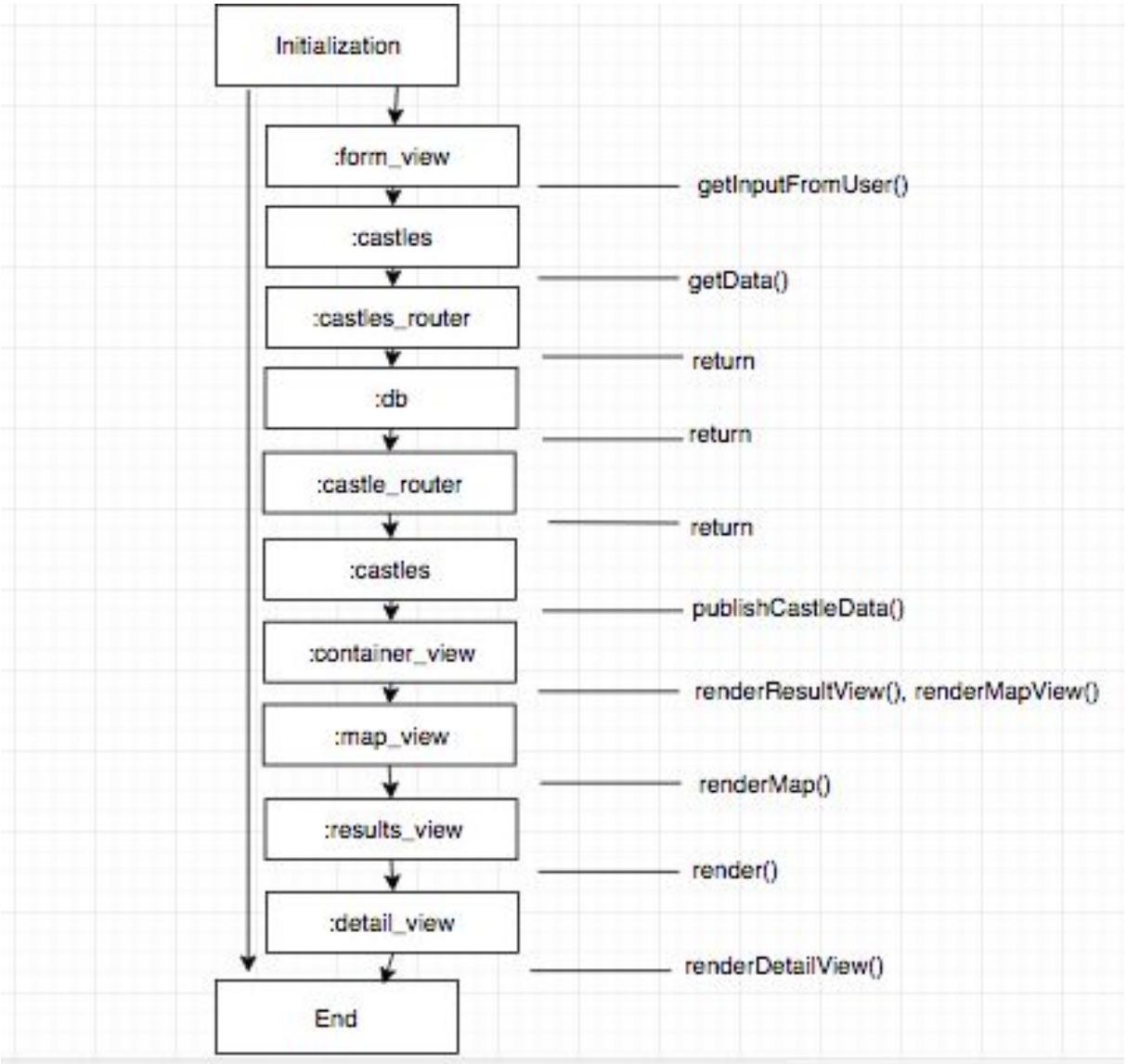


P6 - Produce at least two wireframe designs

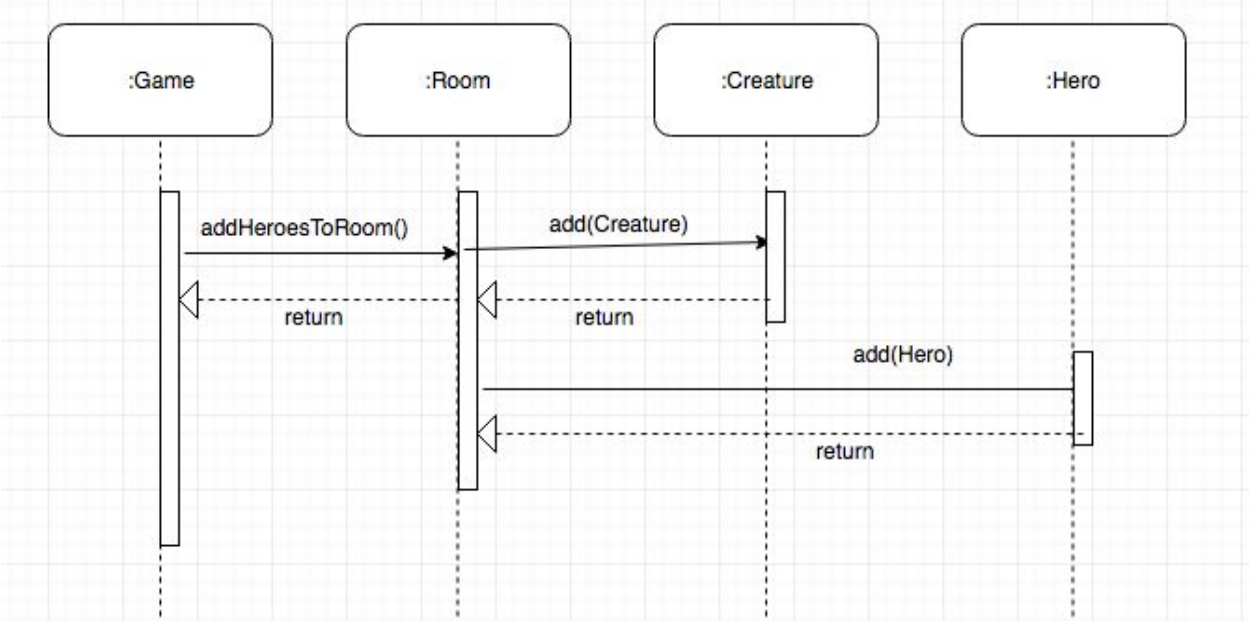


P7 - Week 14 Produce two system interaction diagrams (sequence and/or collaboration diagrams)

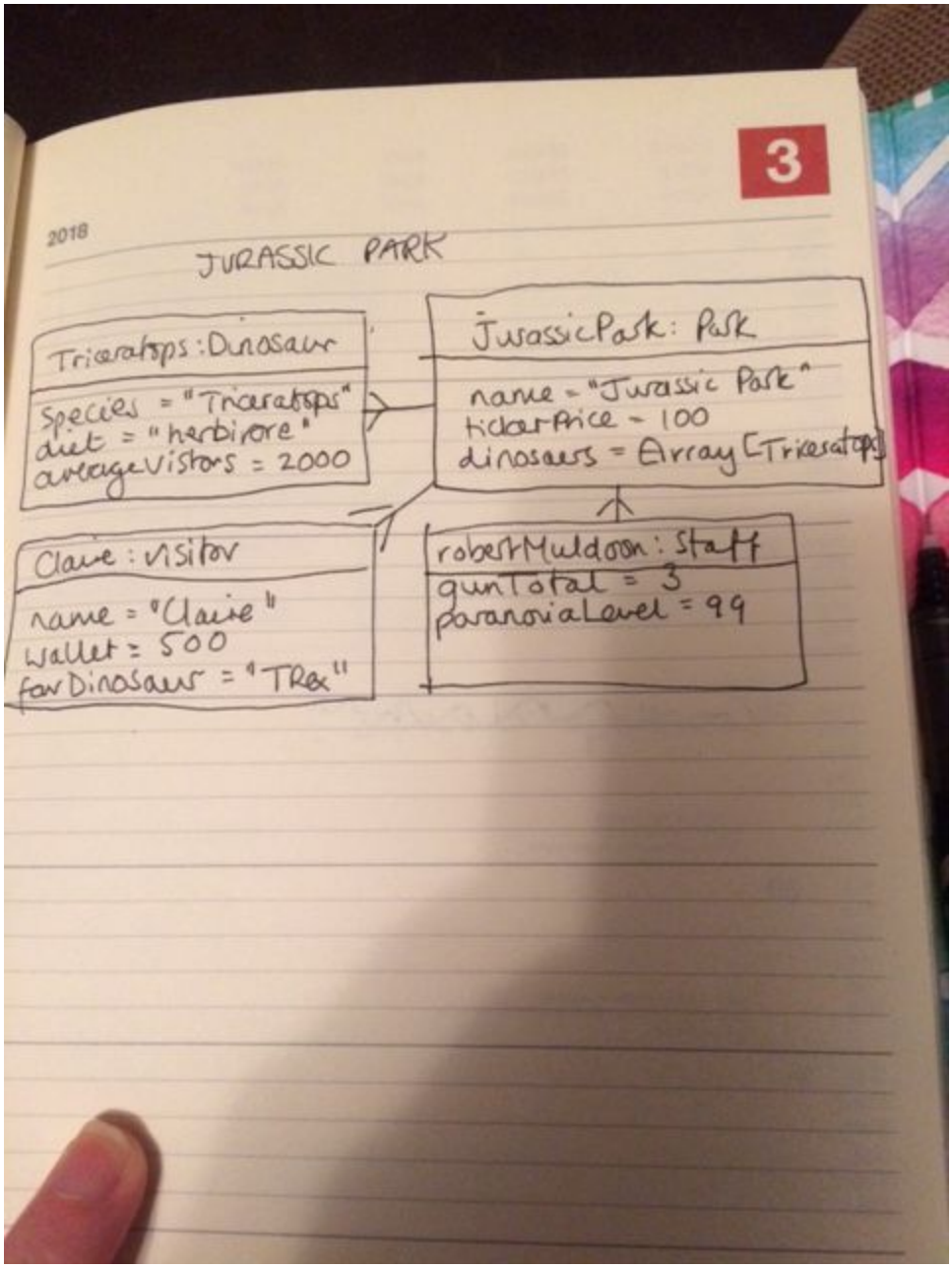
Collaboration diagram: display castles from db

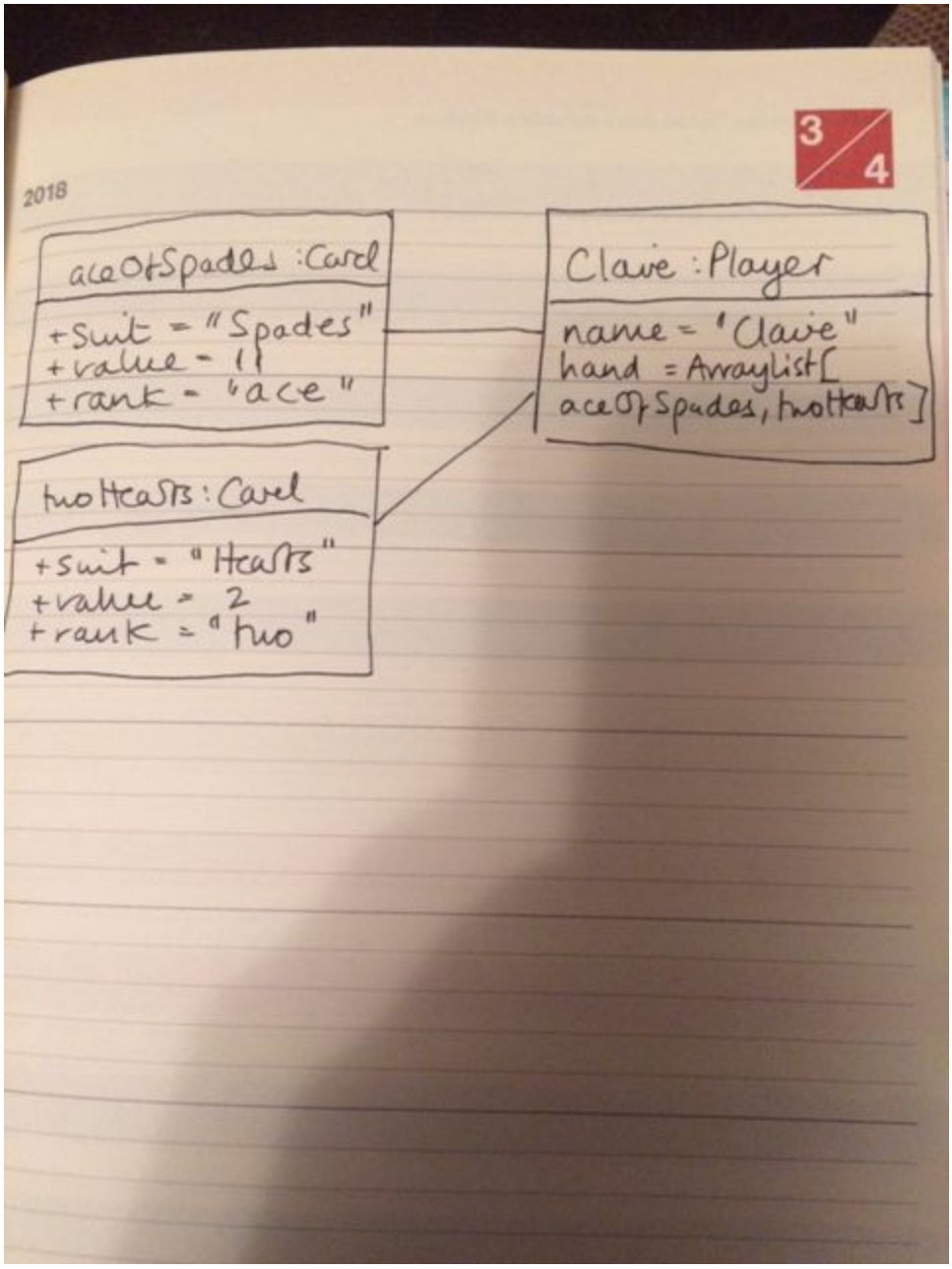


Interaction diagram: creating new room of character objects



P8 - Week 14 Produce two object diagrams




3/4
2018
aceOfSpades :Card
+suit = "Spades"
+value = 11
+rank = "ace"
twoHearts :Card
+suit = "Hearts"
+value = 2
+rank = "two"
Claire :Player
name = "Claire"
hand = ArrayList[
aceOfSpades, twoHearts]

P9 - Select two algorithms you have written (not group project). Take a screenshot of each and a short statement on why you have chosen to use those algorithms.

Example 1:

```ruby
#Returns integer with total climb from all logged
rides.
  def total_climb()
    sum = 0
    routes().each {|route| sum += route.elevation}
    return sum
  end
```

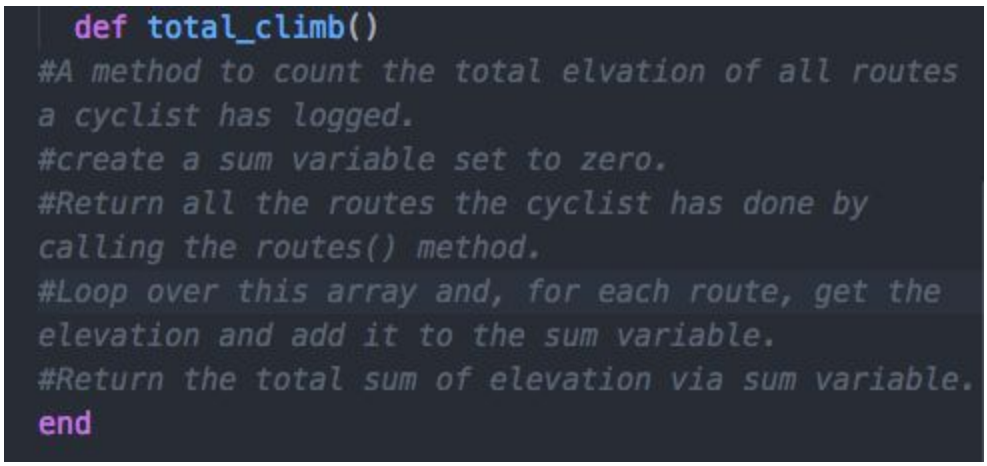
This Ruby function loops over all route objects in an array. For each route, it finds the elevation and adds the value to the sum. I chose this algorithm as it is efficient and human readable.

Example 2:

```java
public int getTotalTreasureValue(){
    int total = 0;
    for(Treasure item : inventory){
        total += item.getTreasureValue();
    }
    return total;
}
```

This Java function loops over an inventory array that contains treasure items. For each treasure item, it accesses its value using the getTreasureValue() method. This is then added to the total variable. I chose this algorithm as it is efficient and human readable.

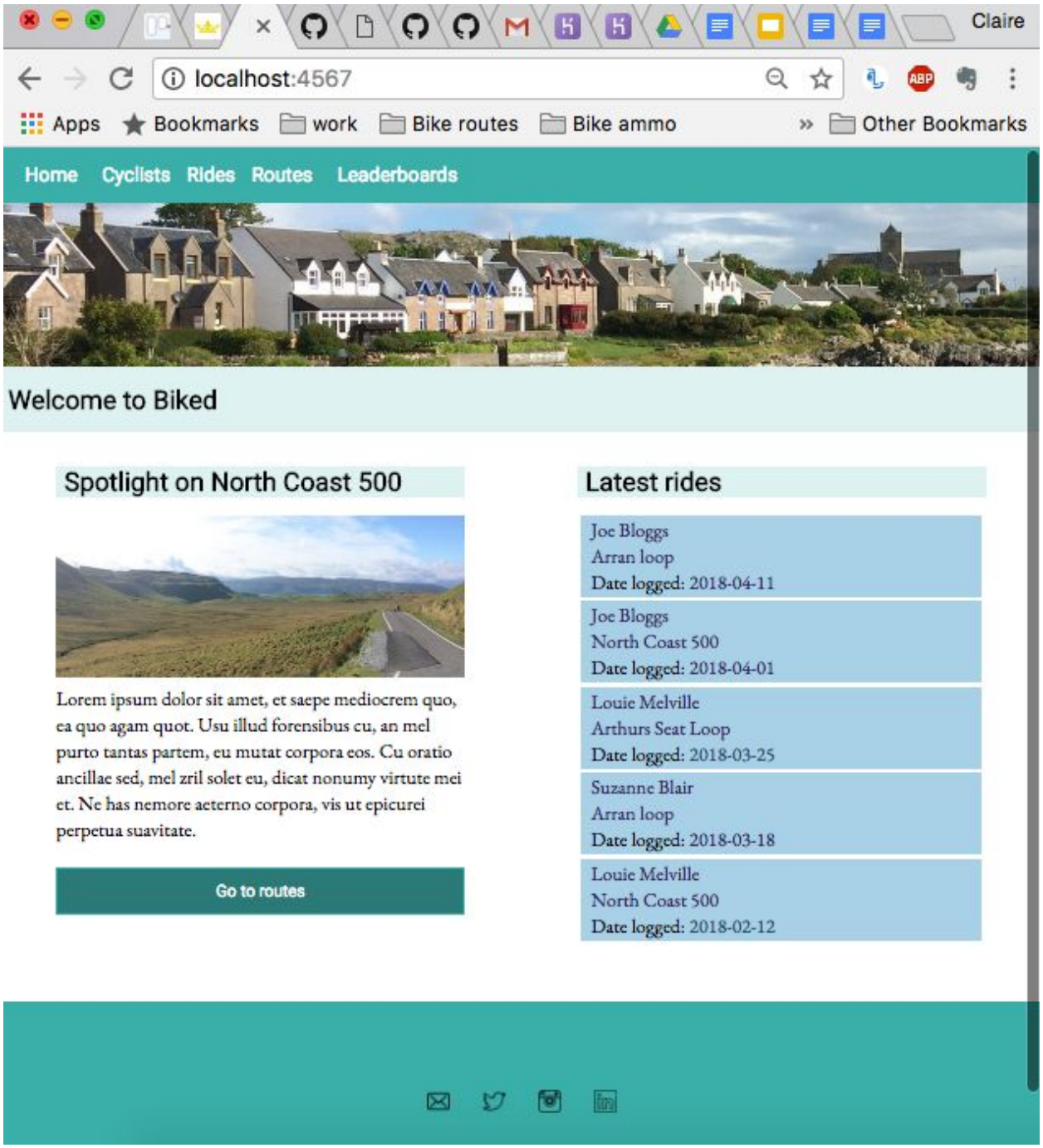
P10 - Take a screenshot of an example of pseudocode for a function.



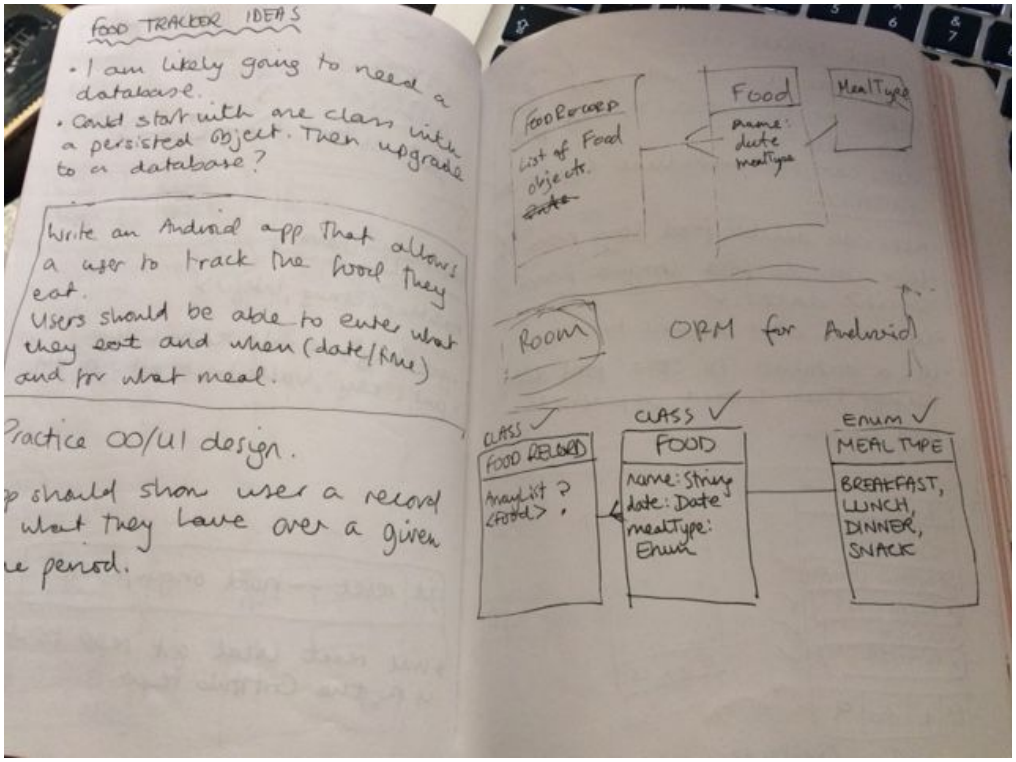
```
  def total_climb()
#A method to count the total elvation of all routes
a cyclist has logged.
#create a sum variable set to zero.
#Return all the routes the cyclist has done by
calling the routes() method.
#Loop over this array and, for each route, get the
elevation and add it to the sum variable.
#Return the total sum of elevation via sum variable.
end
```

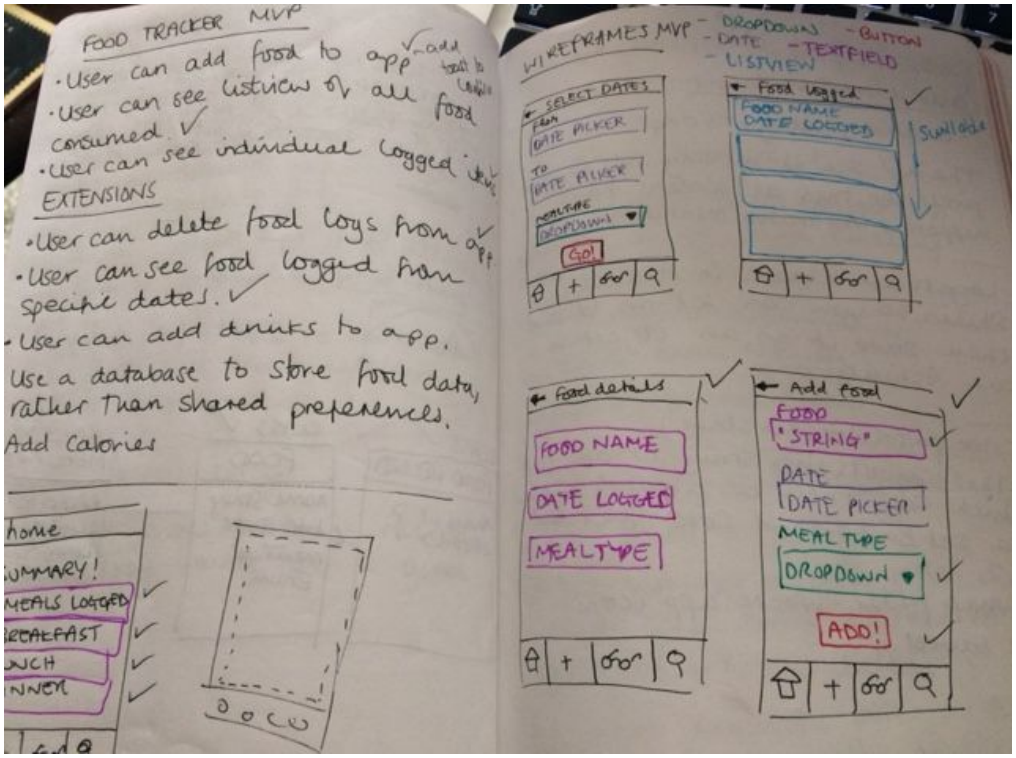
P11 - Take a screenshot of one of your projects where you worked alone and attach the GitHub link.

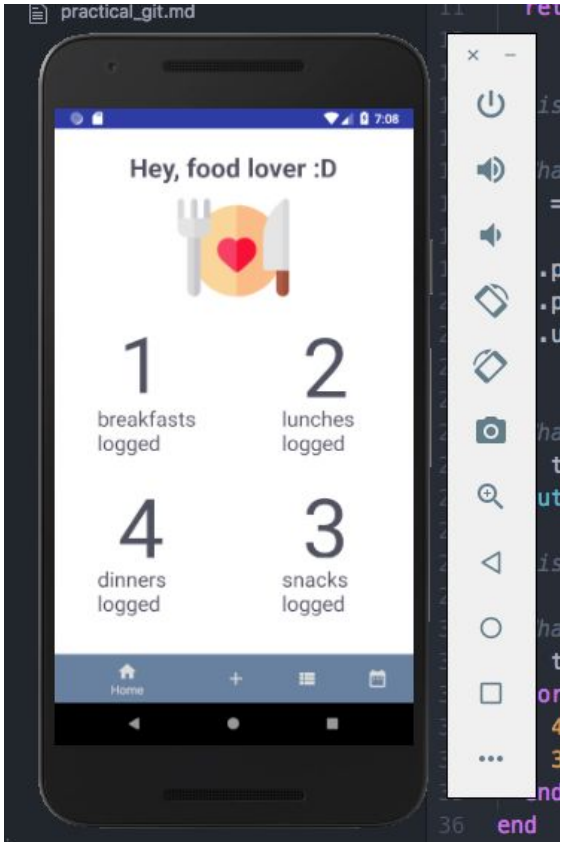
GitHub link: https://github.com/claire-c/cc_wk5_cycling_project

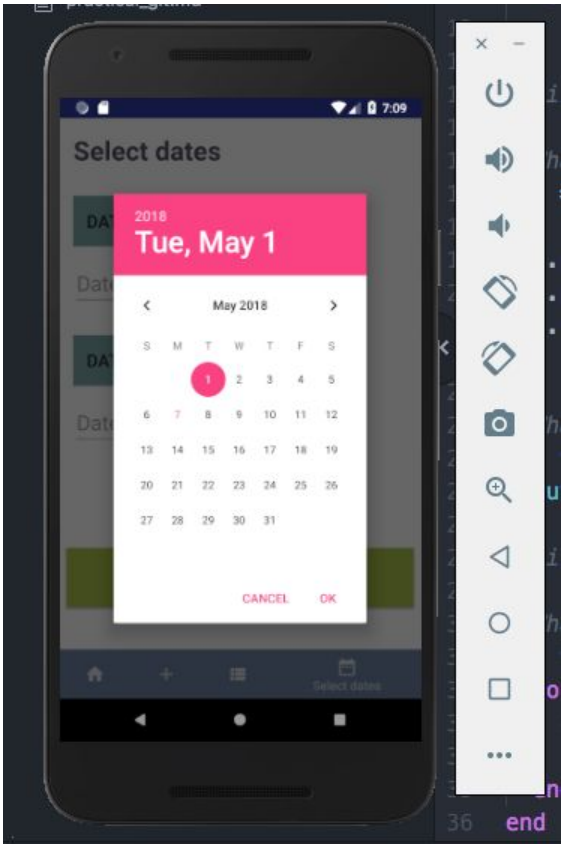


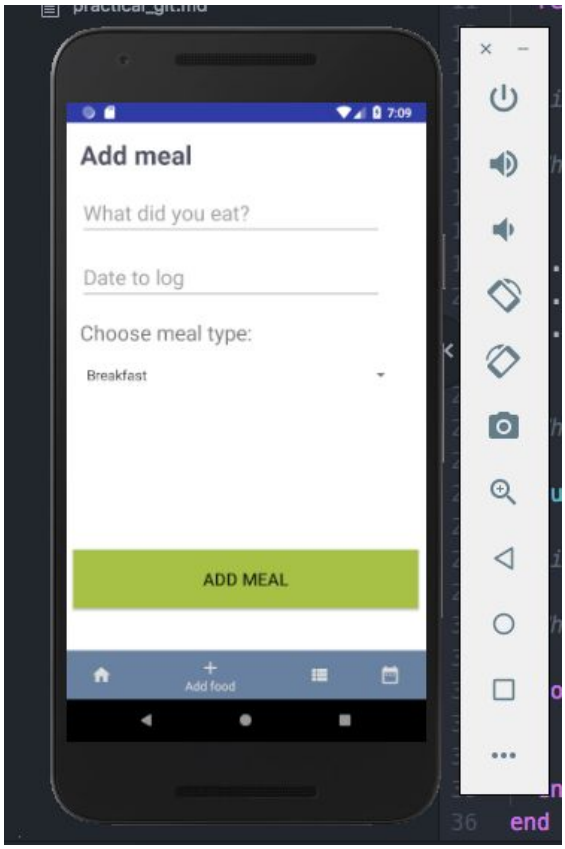
P12 - Take screenshots or photos of your planning and the different states of development to show changes.





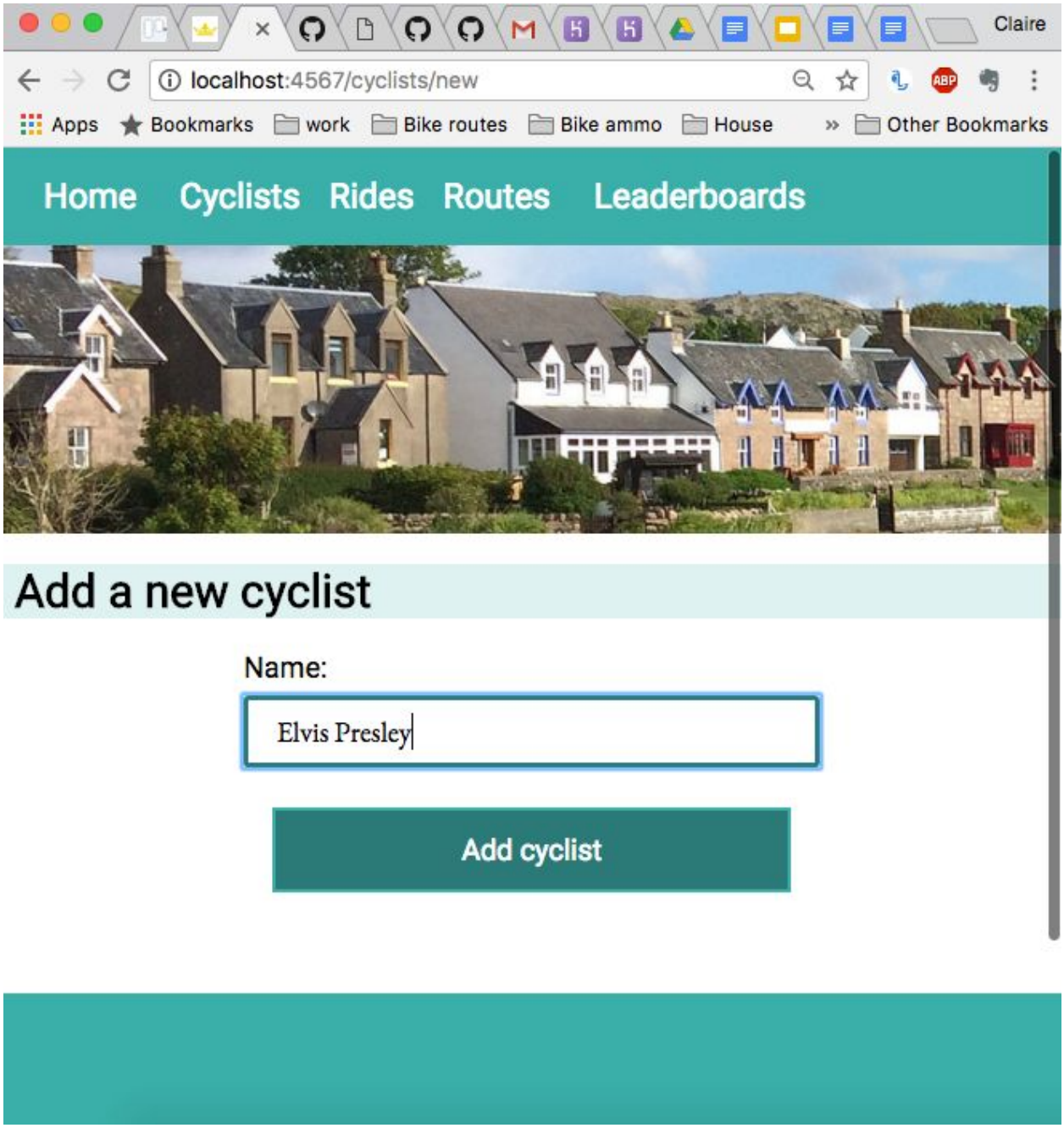

practical_git.md
7:08
Hey, food lover :D
1
breakfasts logged
2
lunches logged
4
dinners logged
3
snacks logged
Home
36
end


7:09
Select dates
2018
Tue, May 1
May 2018
CANCEL
OK
36
end

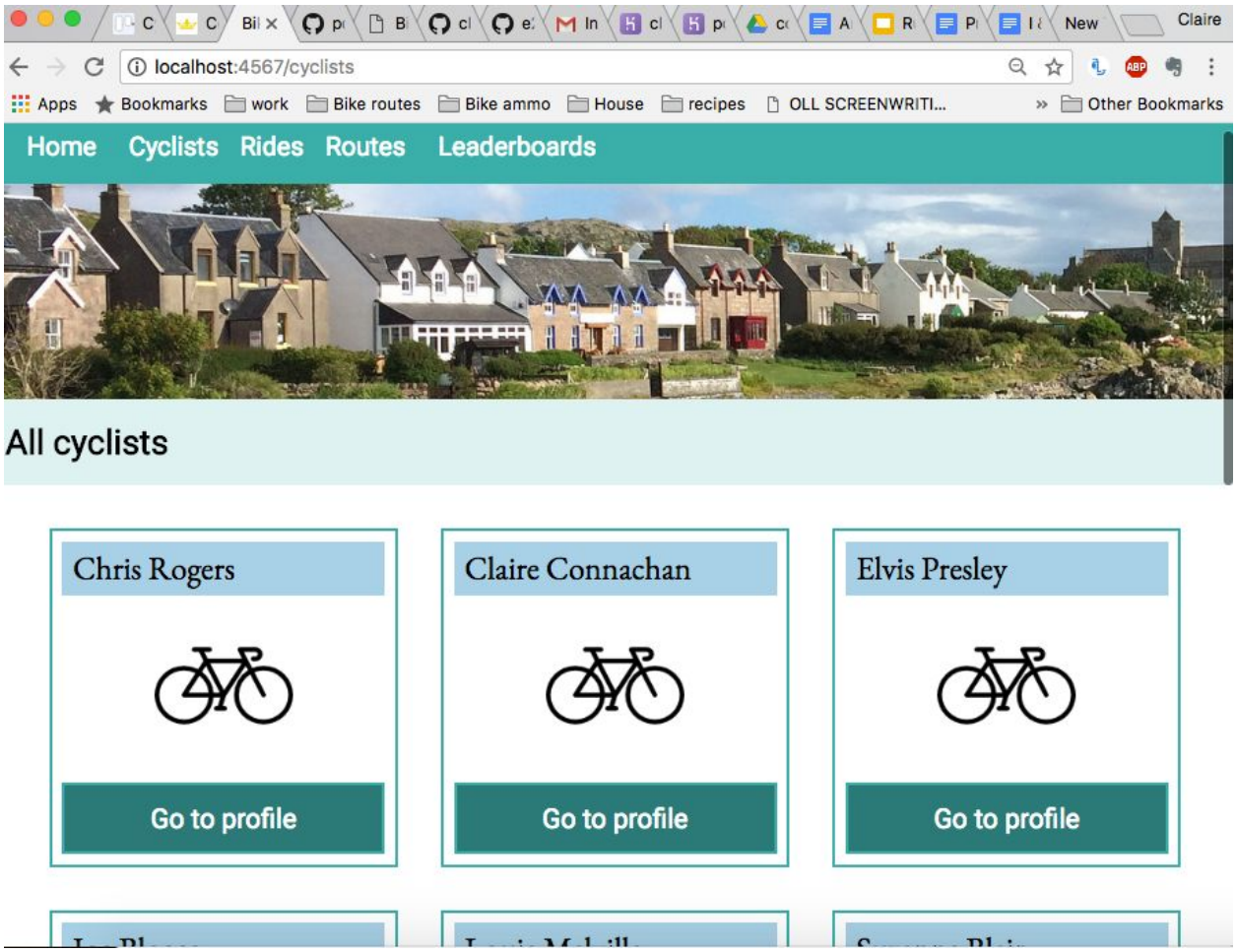

7:09
Add meal
What did you eat?
Date to log
Choose meal type:
Breakfast
ADD MEAL
Add food
36
end

P13 - Show user input being processed to design requirements.

User inputs new cyclist record into the database:

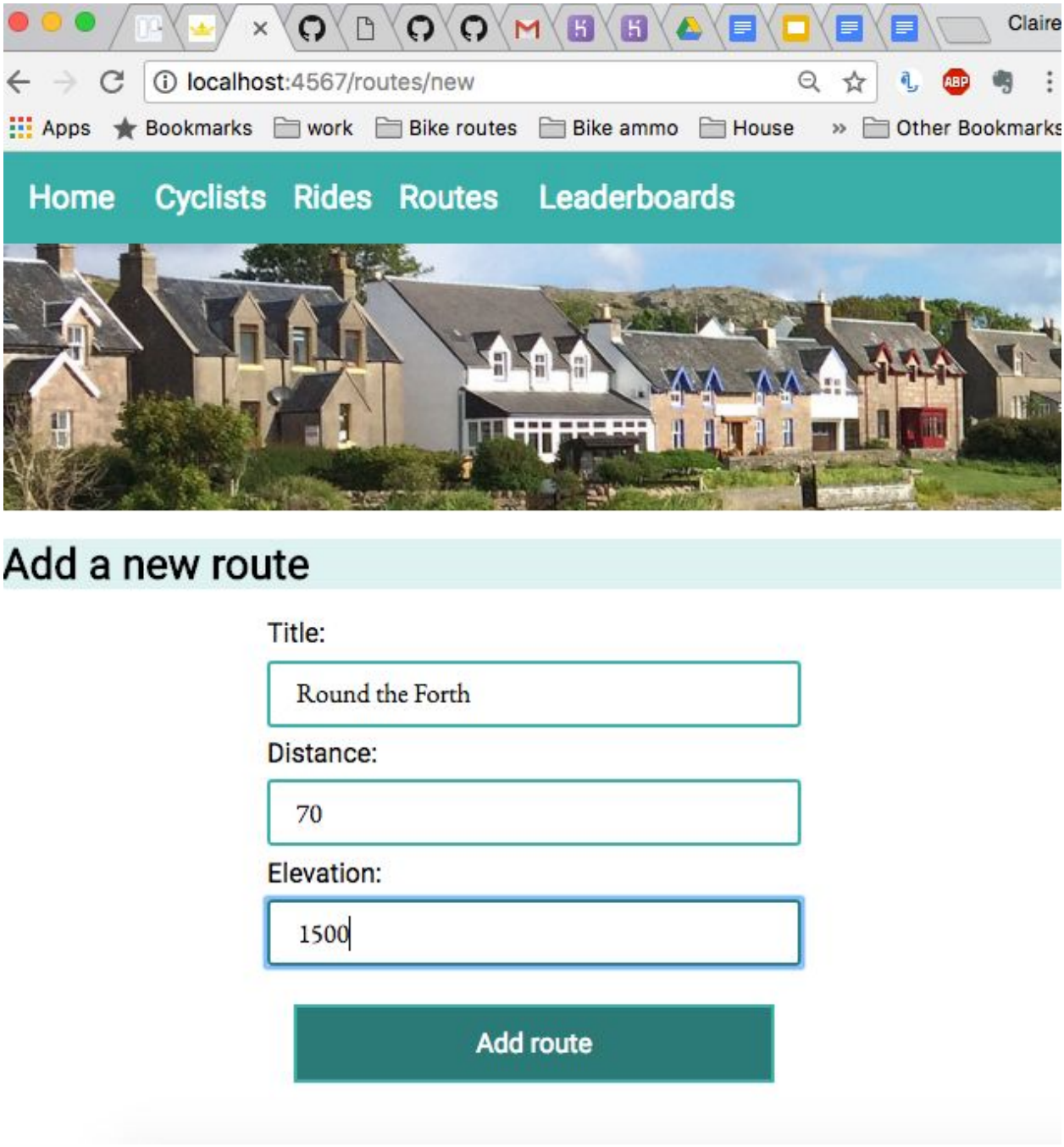


New cyclist record is saved and added to the list of all cyclists. It can now be accessed and rides logged against it:

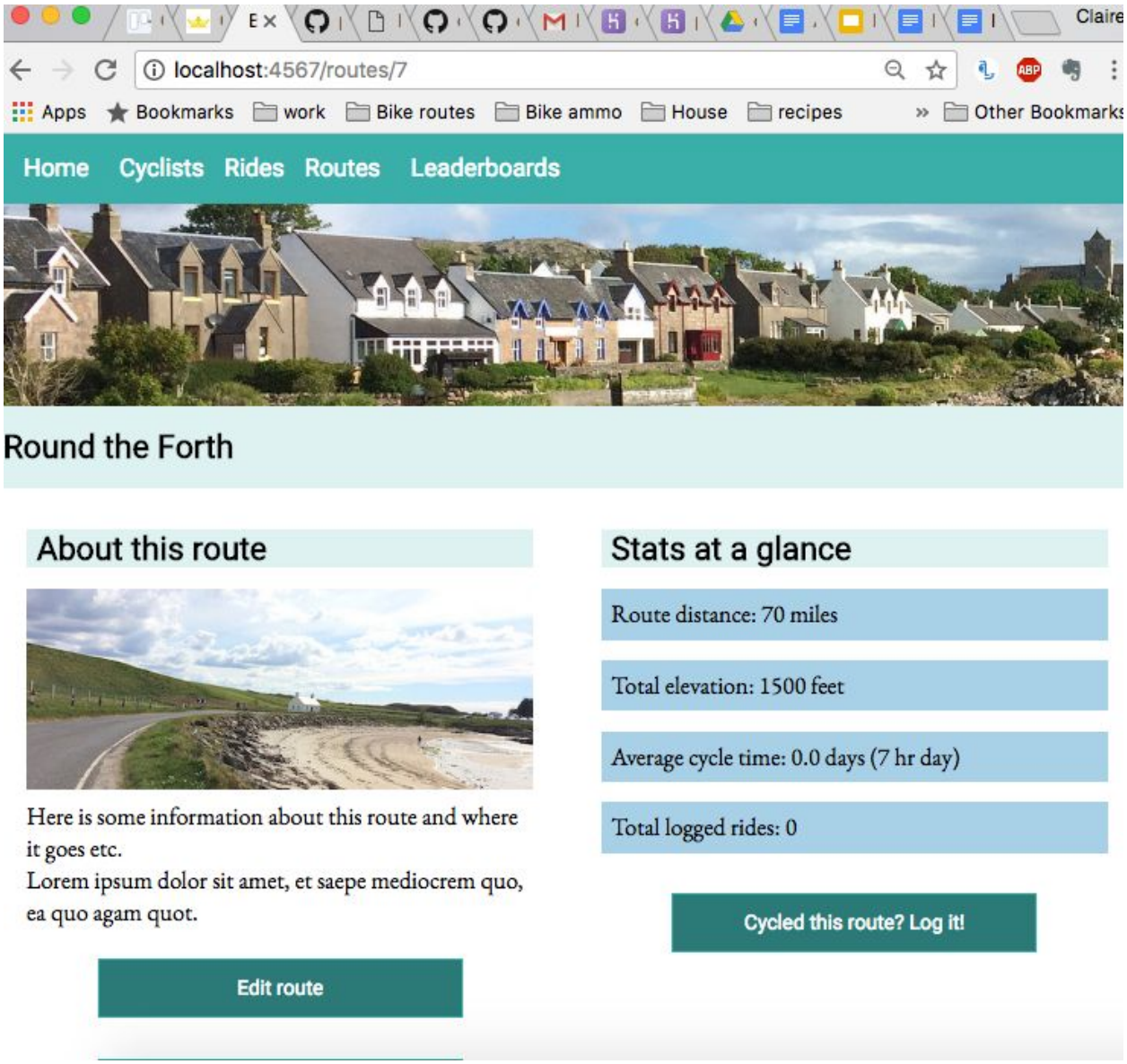


P14 - Show an interaction with data persistence

User inputs new route to database:

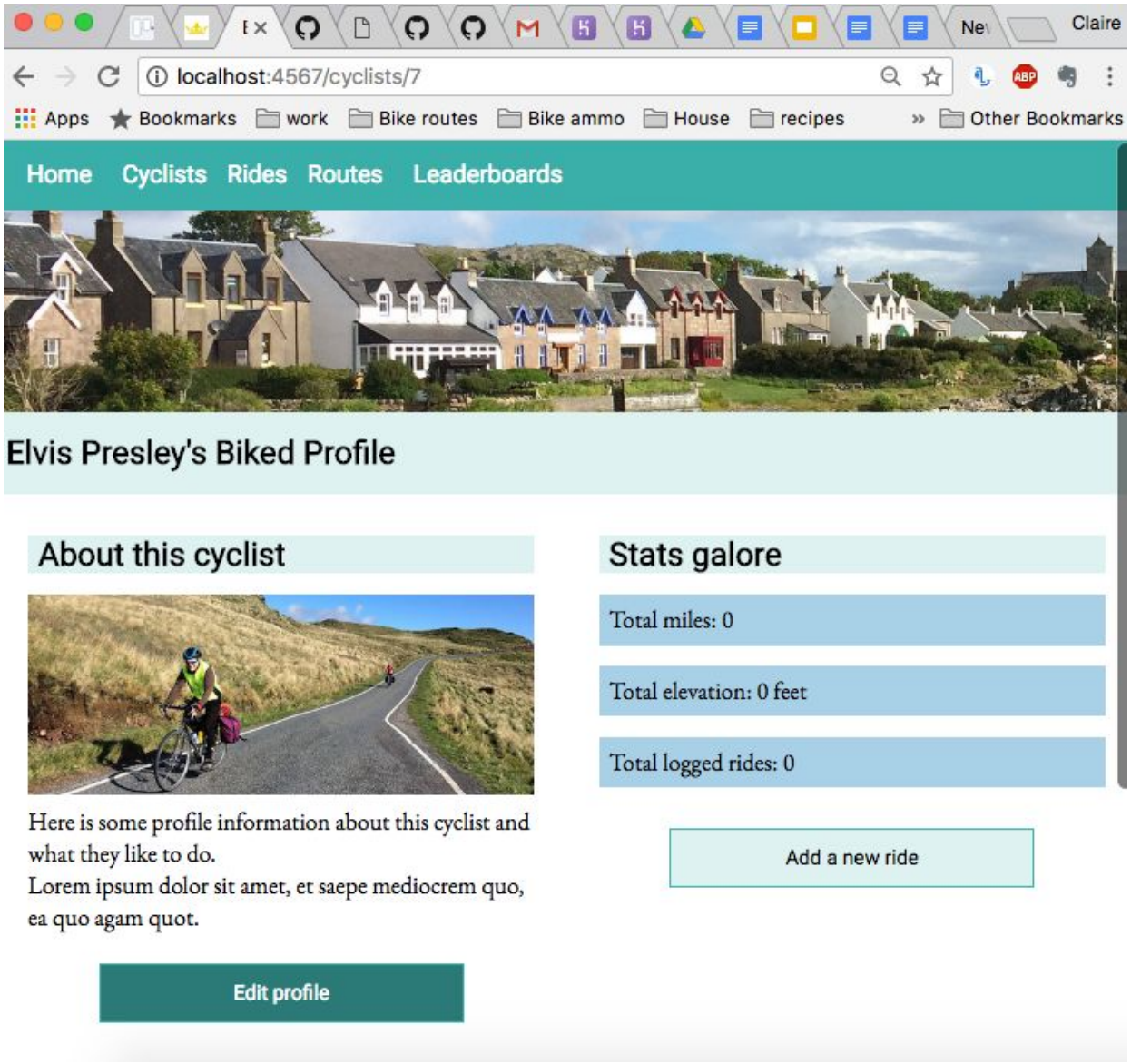


Route saved to database and is confirmed saved by accessing via website:

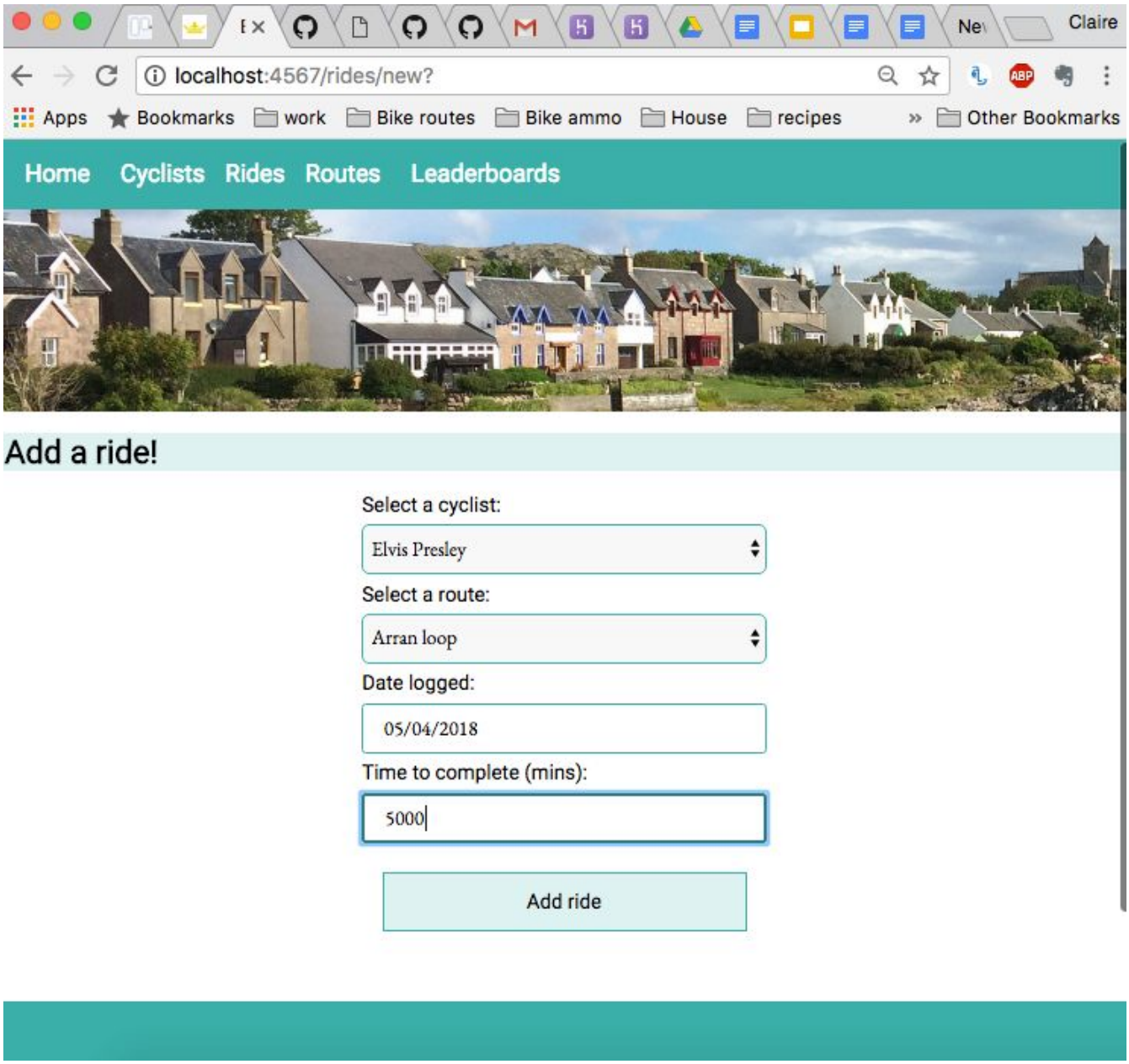


P15 - Show the correct output of results and feedback to user.

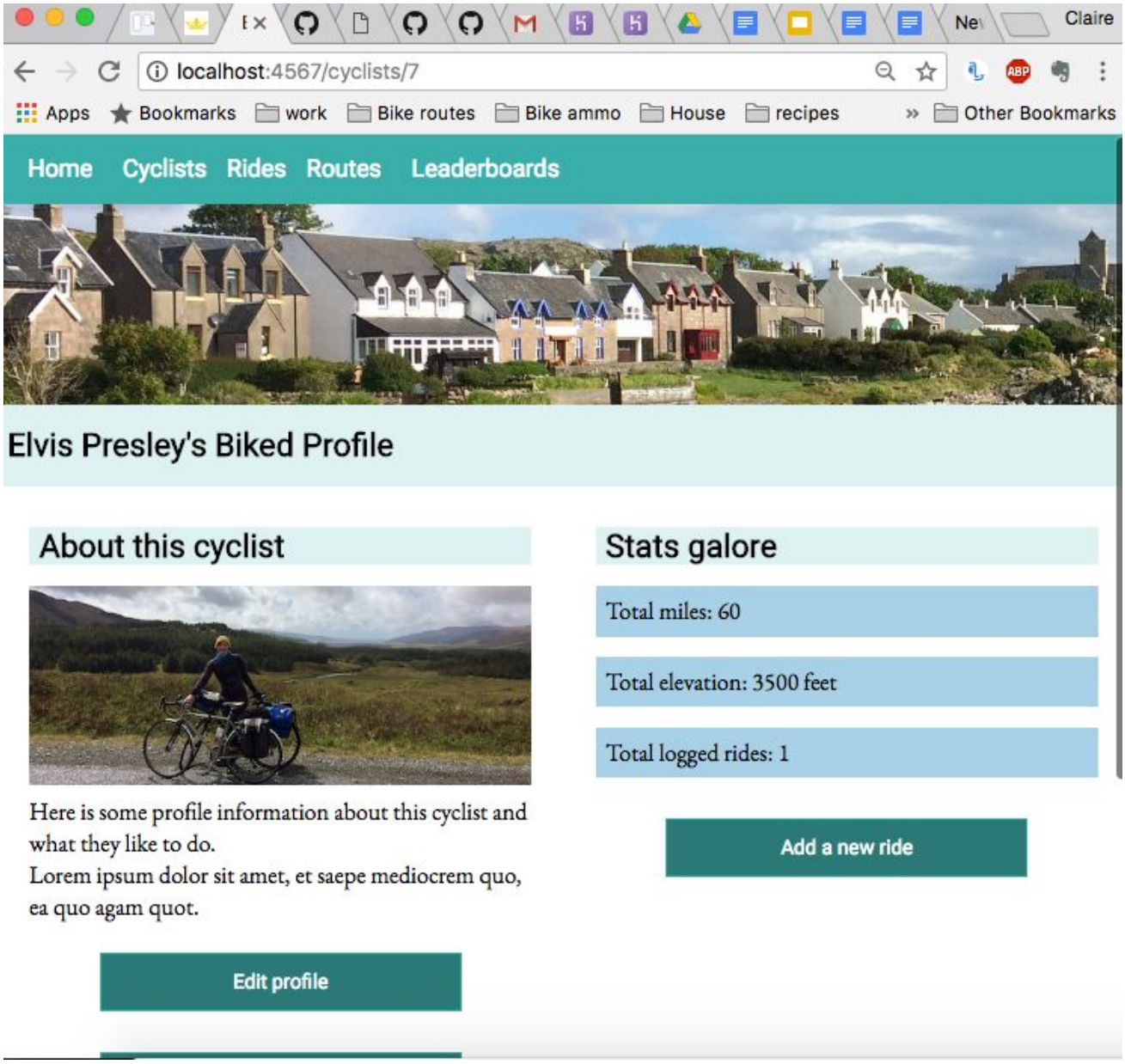
1. User requests to add a new ride to cyclist profile:



2. User taken to “add new ride page”, where they input details of ride they wish to log:



3. Logged ride is added to the database and cyclist profile updated to show ride data and associated statistics:



P16 Week 12 - Show an API being used within your program.

API request:

```
FilmData.prototype.loadFilms = function () {
  //returns an array of Studio Ghibli film objects.
  const request = new Request("https://ghibliapi.herokuapp.com/films");

  request.get((data) => {
    this.films = data;
    PubSub.publish("FilmData:films-loaded", this.films);
  });
};
```

Rendered in browser:

P17 Produce a bug tracking report.

BUG TRACKING REPORT:

Knight's shield reduces damage by half	**Pass**	**Knight can use signature move**	**Pass**
Knight can add treasure to their inventory	**Pass**	**Knight can get value of treasure in inventory**	**Pass**
Knight can take damage	**Pass**		

P18 Week 11 - Demonstrate testing in your program.

(see separate sheet with testing in the PDA repo - it's under 'static and dynamic testing' directory)