Evidence Gathering Document for SQA Level 8 Professional Developer Award.

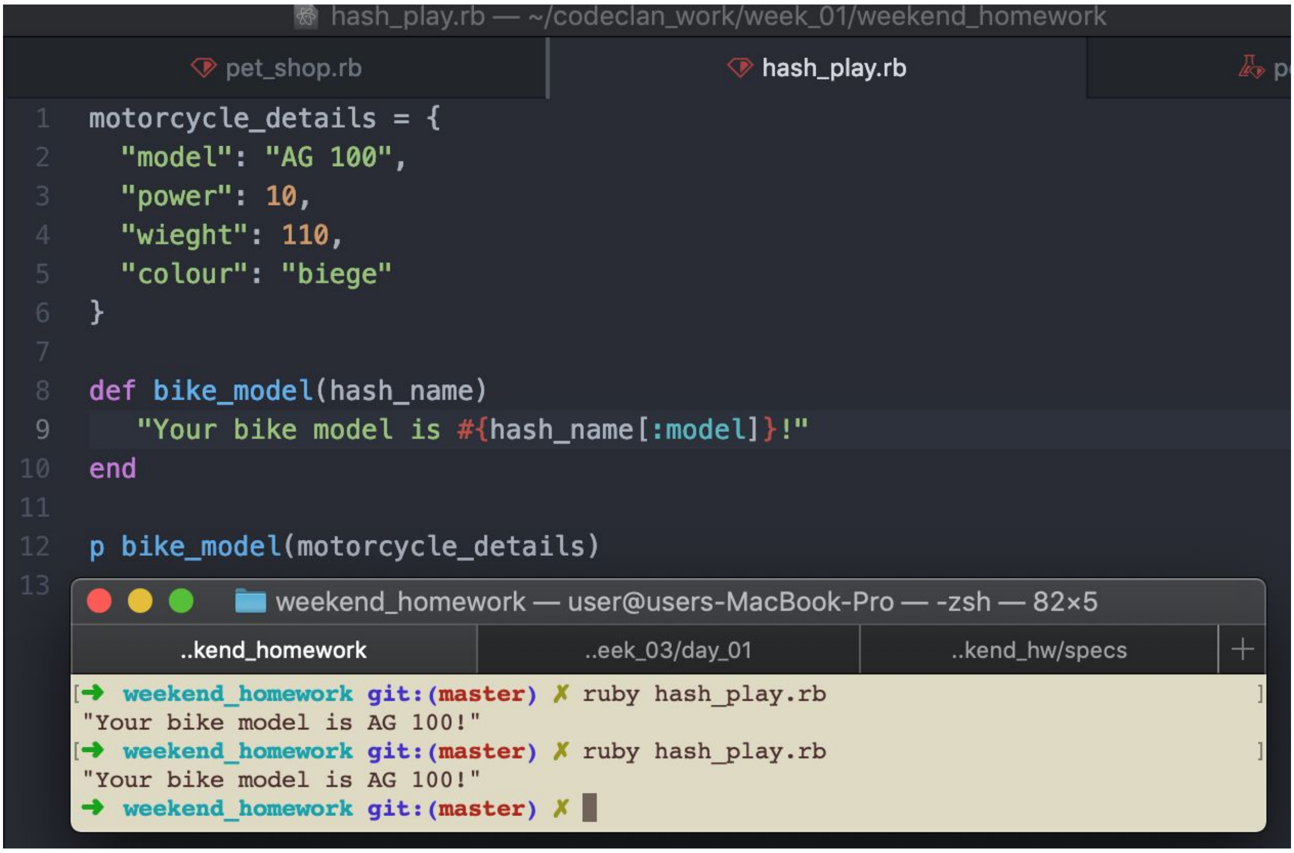
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

Fill in each point with screenshot or diagram and description of what you are showing.

Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

Week 1

Unit	Ref	Evidence	
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	


hash_play.rb — ~/codeclan_work/week_01/weekend_homework
pet_shop.rb
hash_play.rb
p
weekend_homework — user@users-MacBook-Pro — -zsh — 82×5
..kend_homework
..eek_03/day_01
..kend_hw/specs

```ruby
motorcycle_details = {
  "model": "AG 100",
  "power": 10,
  "wieght": 110,
  "colour": "biege"
}

def bike_model(hash_name)
   "Your bike model is #{hash_name[:model]}!"
end

p bike_model(motorcycle_details)
```

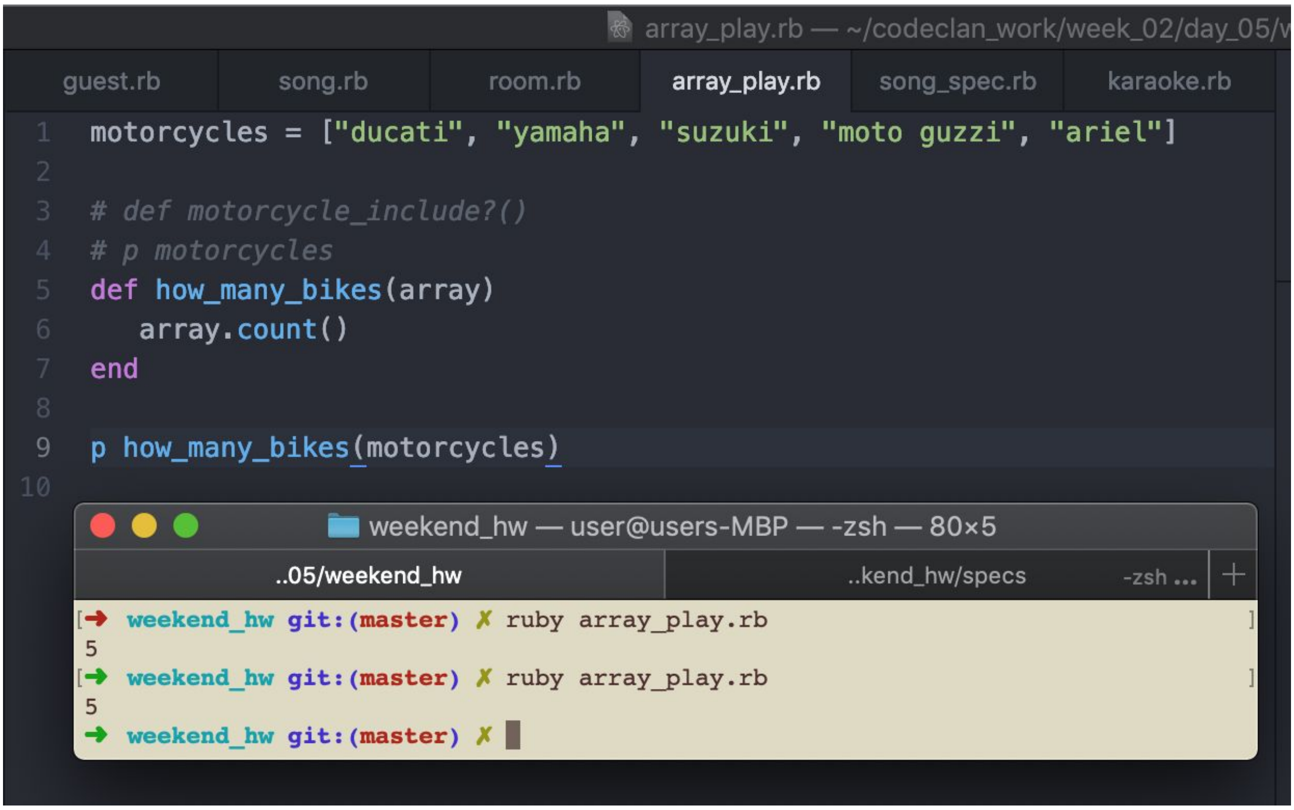
```
[➜  weekend_homework git:(master) ✗ ruby hash_play.rb
"Your bike model is AG 100!"
[➜  weekend_homework git:(master) ✗ ruby hash_play.rb
"Your bike model is AG 100!"
➜  weekend_homework git:(master) ✗ 
```

Here is an example of a hash being used to store the details of a motorcycle. The function is bike_model() is then called to print what is the model name.

Week 2

Unit	Ref	Evidence	
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	

Here is an array of motorcycles, and the function how_many_bikes() returns the number of bikes



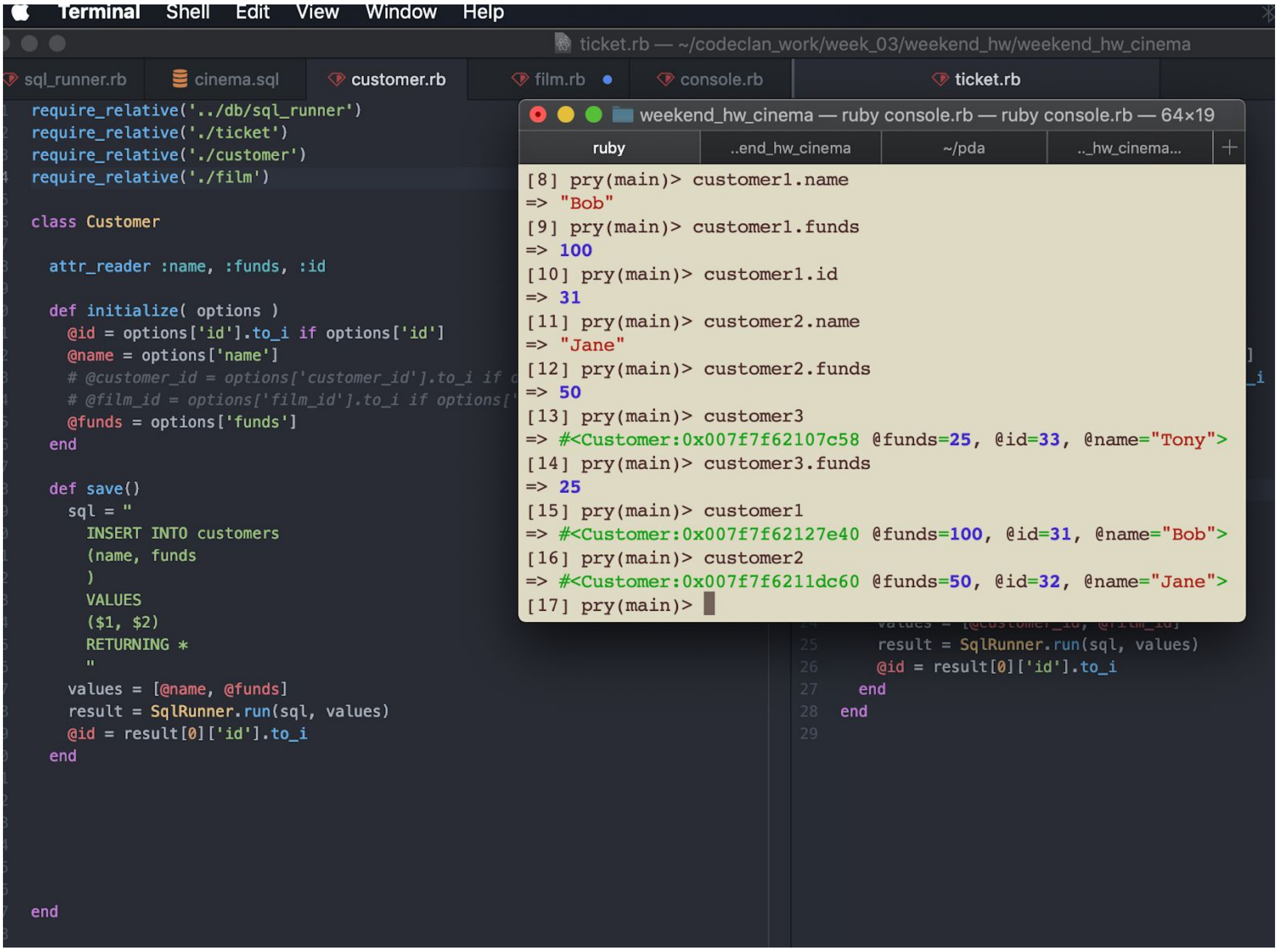
```ruby
motorcycles = ["ducati", "yamaha", "suzuki", "moto guzzi", "ariel"]

# def motorcycle_include?()
# p motorcycles
def how_many_bikes(array)
   array.count()
end

p how_many_bikes(motorcycles)

```

```
[➜ weekend_hw git:(master) ✗ ruby array_play.rb                    ]
5
[➜ weekend_hw git:(master) ✗ ruby array_play.rb                    ]
5
➜ weekend_hw git:(master) ✗ █
```

Week 3

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

Here is a function running that can return data when called. Here we can find the customer name, id and funds individually by calling .name, .funds or .id respectively. Alternatively all the customers data can be displayed by simply calling customer1 etc.

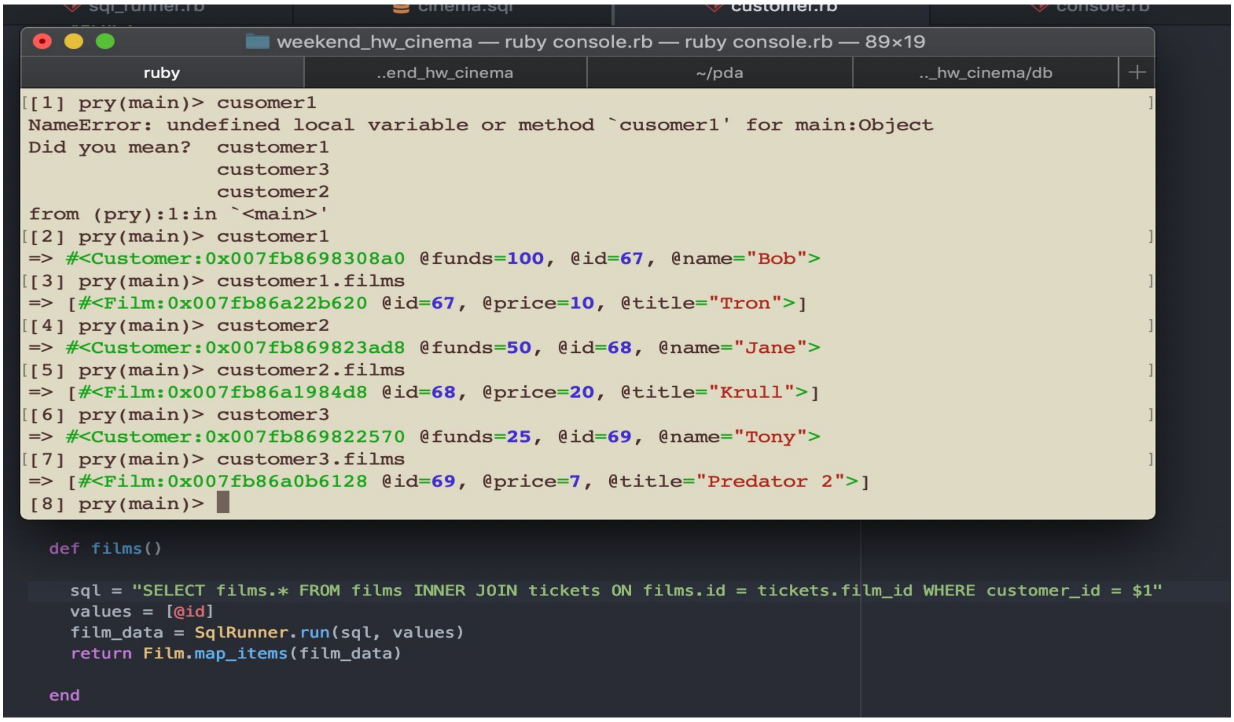


```ruby
require_relative('../db/sql_runner')
require_relative('./ticket')
require_relative('./customer')
require_relative('./film')

class Customer

  attr_reader :name, :funds, :id

  def initialize( options )
    @id = options['id'].to_i if options['id']
    @name = options['name']
    # @customer_id = options['customer_id'].to_i if c
    # @film_id = options['film_id'].to_i if options['
    @funds = options['funds']
  end

  def save()
    sql = "
      INSERT INTO customers
      (name, funds
      )
      VALUES
      ($1, $2)
      RETURNING *
      "
    values = [@name, @funds]
    result = SqlRunner.run(sql, values)
    @id = result[0]['id'].to_i
  end





end
```

```
[8] pry(main)> customer1.name
=> "Bob"
[9] pry(main)> customer1.funds
=> 100
[10] pry(main)> customer1.id
=> 31
[11] pry(main)> customer2.name
=> "Jane"
[12] pry(main)> customer2.funds
=> 50
[13] pry(main)> customer3
=> #<Customer:0x007f7f62107c58 @funds=25, @id=33, @name="Tony">
[14] pry(main)> customer3.funds
=> 25
[15] pry(main)> customer1
=> #<Customer:0x007f7f62127e40 @funds=100, @id=31, @name="Bob">
[16] pry(main)> customer2
=> #<Customer:0x007f7f6211dc60 @funds=50, @id=32, @name="Jane">
[17] pry(main)>
```

```ruby
      result = SqlRunner.run(sql, values)
      @id = result[0]['id'].to_i
    end
  end

```

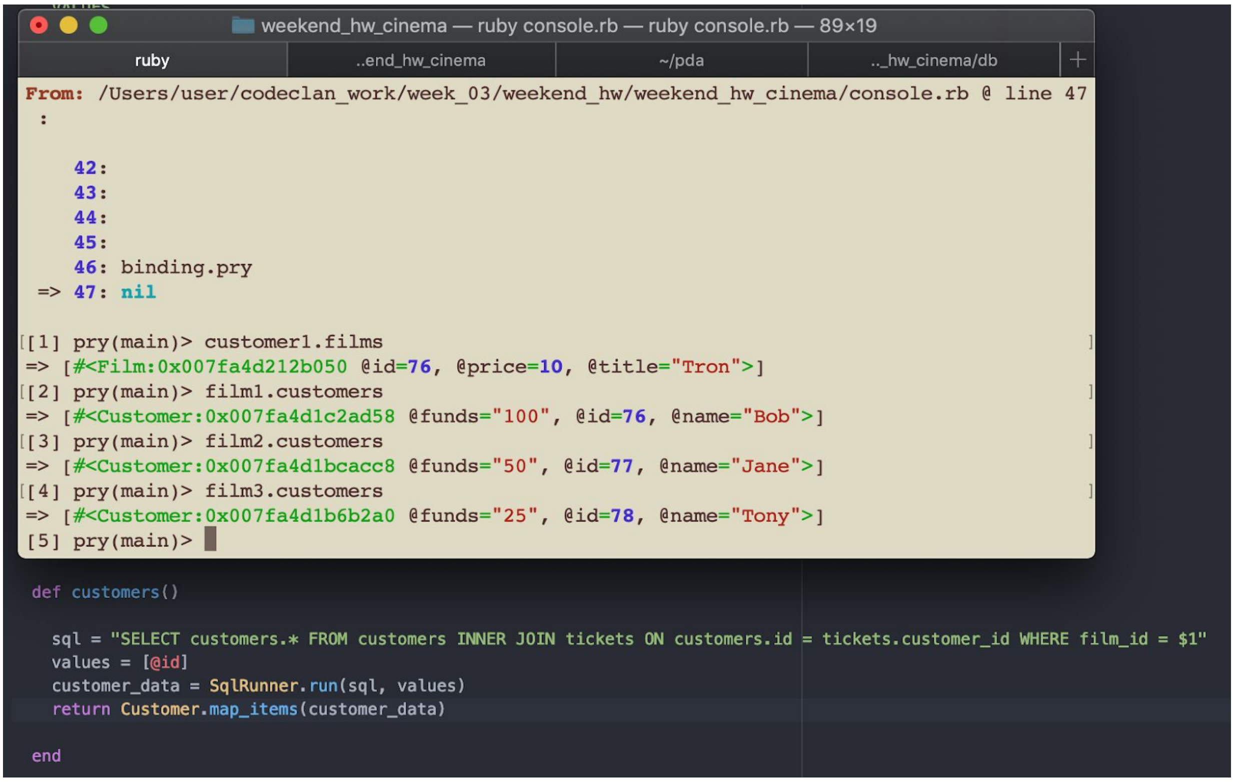
Unit	Ref	Evidence	
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	



```
[1] pry(main)> cusomer1
NameError: undefined local variable or method `cusomer1' for main:Object
Did you mean?  customer1
               customer3
               customer2
from (pry):1:in `<main>'
[2] pry(main)> customer1
=> #<Customer:0x007fb8698308a0 @funds=100, @id=67, @name="Bob">
[3] pry(main)> customer1.films
=> [#<Film:0x007fb86a22b620 @id=67, @price=10, @title="Tron">]
[4] pry(main)> customer2
=> #<Customer:0x007fb869823ad8 @funds=50, @id=68, @name="Jane">
[5] pry(main)> customer2.films
=> [#<Film:0x007fb86a1984d8 @id=68, @price=20, @title="Krull">]
[6] pry(main)> customer3
=> #<Customer:0x007fb869822570 @funds=25, @id=69, @name="Tony">
[7] pry(main)> customer3.films
=> [#<Film:0x007fb86a0b6128 @id=69, @price=7, @title="Predator 2">]
[8] pry(main)>
```

```
def films()

  sql = "SELECT films.* FROM films INNER JOIN tickets ON films.id = tickets.film_id WHERE customer_id = $1"
  values = [@id]
  film_data = SqlRunner.run(sql, values)
  return Film.map_items(film_data)

end
```

Here data can be sorted by finding the films a particular customer has been to see. Below is the opposite sort, here we can see all customers who have seen a particular film

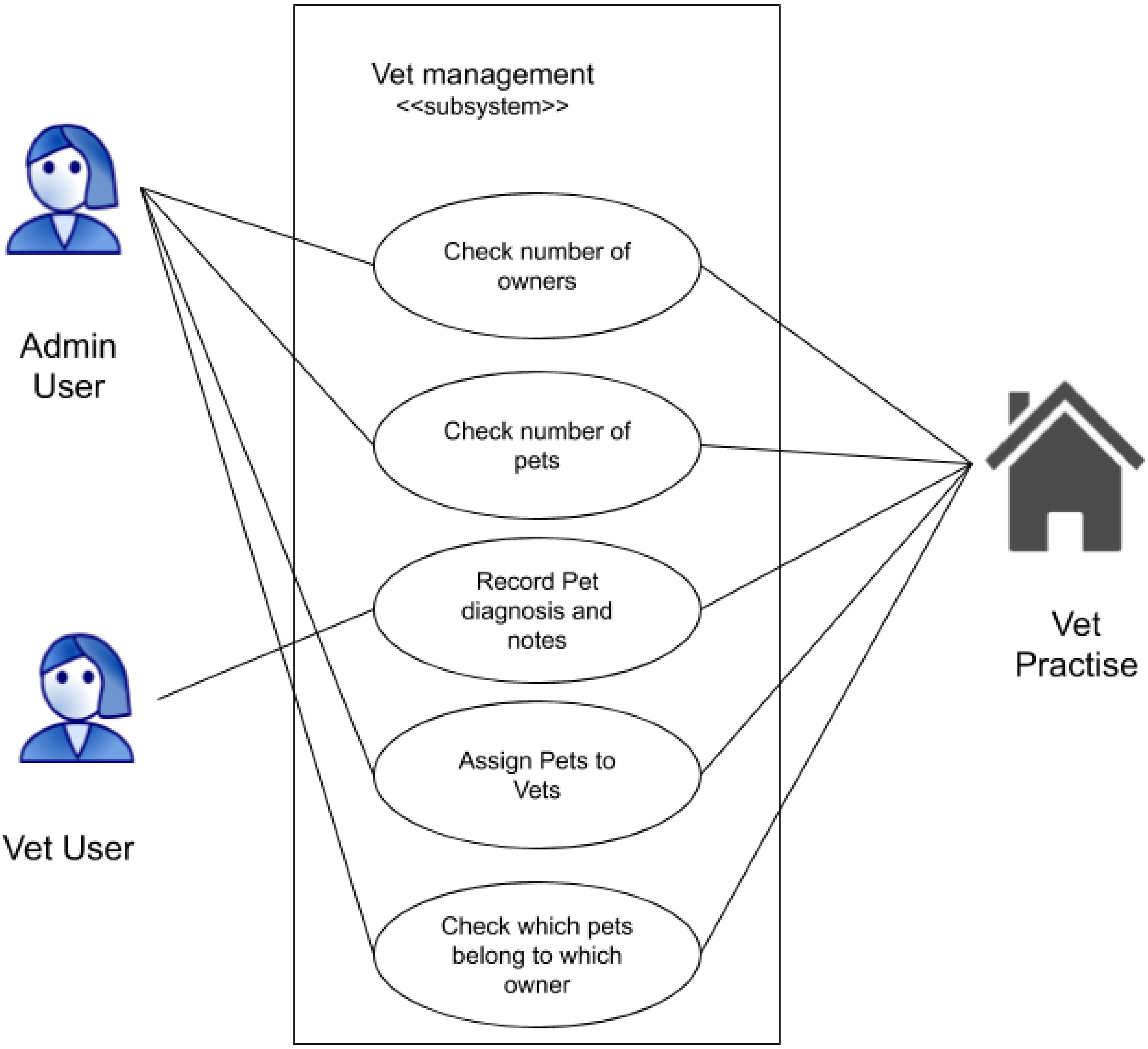

weekend_hw_cinema — ruby console.rb — ruby console.rb — 89×19
ruby
..end_hw_cinema
~/pda
.._hw_cinema/db

```
From: /Users/user/codeclan_work/week_03/weekend_hw/weekend_hw_cinema/console.rb @ line 47
 :

    42:
    43:
    44:
    45:
    46: binding.pry
 => 47: nil

[1] pry(main)> customer1.films
=> [#<Film:0x007fa4d212b050 @id=76, @price=10, @title="Tron">]
[2] pry(main)> film1.customers
=> [#<Customer:0x007fa4d1c2ad58 @funds="100", @id=76, @name="Bob">]
[3] pry(main)> film2.customers
=> [#<Customer:0x007fa4d1bcacc8 @funds="50", @id=77, @name="Jane">]
[4] pry(main)> film3.customers
=> [#<Customer:0x007fa4d1b6b2a0 @funds="25", @id=78, @name="Tony">]
[5] pry(main)>
```

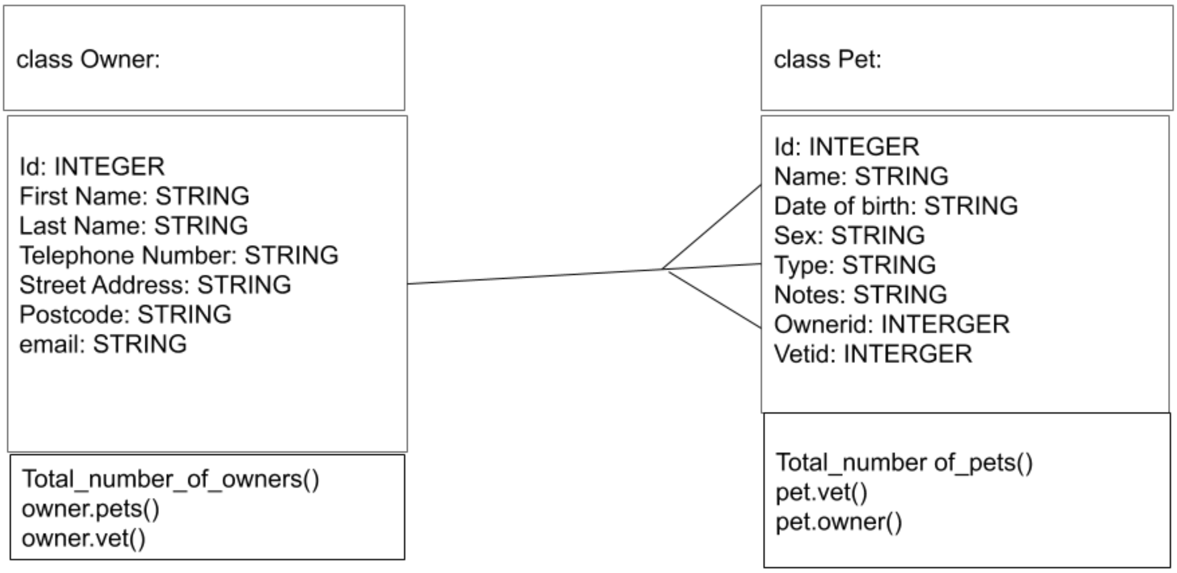
```
def customers()

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

end
```

Week 4

Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	



Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	



class Owner:

Id: INTEGER
First Name: STRING
Last Name: STRING
Telephone Number: STRING
Street Address: STRING
Postcode: STRING
email: STRING

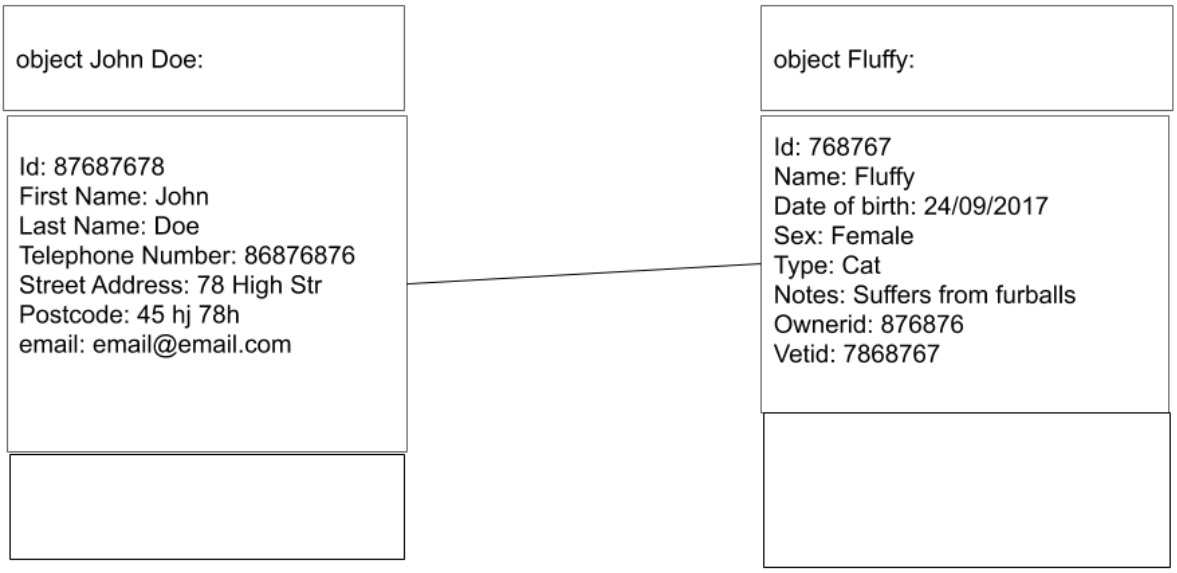
Total_number_of_owners()
owner.pets()
owner.vet()

class Pet:

Id: INTEGER
Name: STRING
Date of birth: STRING
Sex: STRING
Type: STRING
Notes: STRING
Ownerid: INTERGER
Vetid: INTERGER

Total_number of_pets()
pet.vet()
pet.owner()

Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	



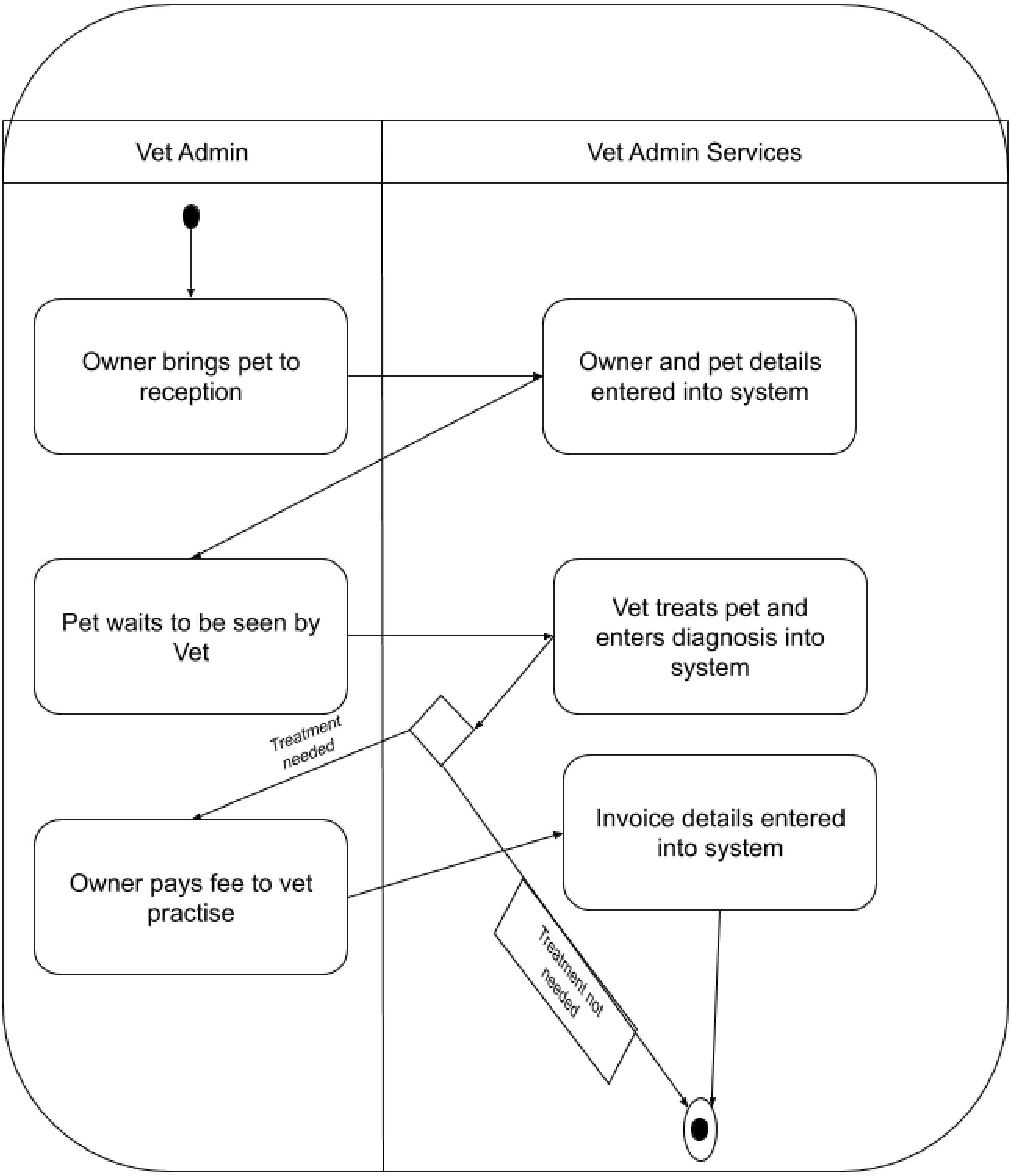
object John Doe:

Id: 87687678
First Name: John
Last Name: Doe
Telephone Number: 86876876
Street Address: 78 High Str
Postcode: 45 hj 78h
email: email@email.com

object Fluffy:

Id: 768767
Name: Fluffy
Date of birth: 24/09/2017
Sex: Female
Type: Cat
Notes: Suffers from furballs
Ownerid: 876876
Vetid: 7868767

Unit	Ref	Evidence	
A&D	A.D.4	An Activity Diagram	

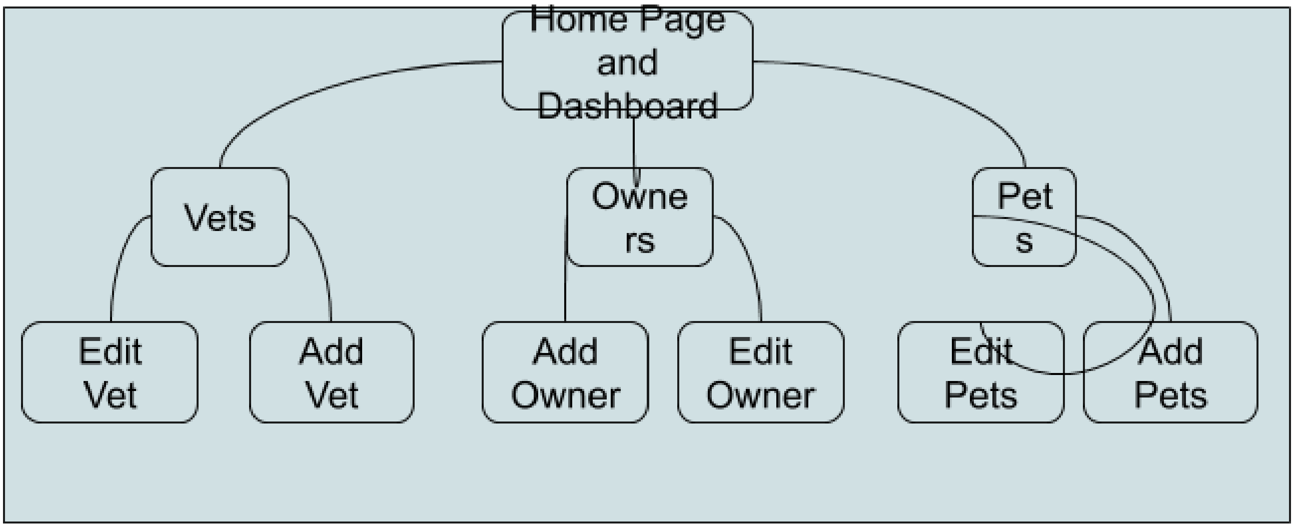

Vet Admin
Vet Admin Services
Owner brings pet to reception
Owner and pet details entered into system
Pet waits to be seen by Vet
Vet treats pet and enters diagnosis into system
Treatment needed
Owner pays fee to vet practise
Invoice details entered into system
Treatment not needed

Unit	Ref	Evidence	
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	

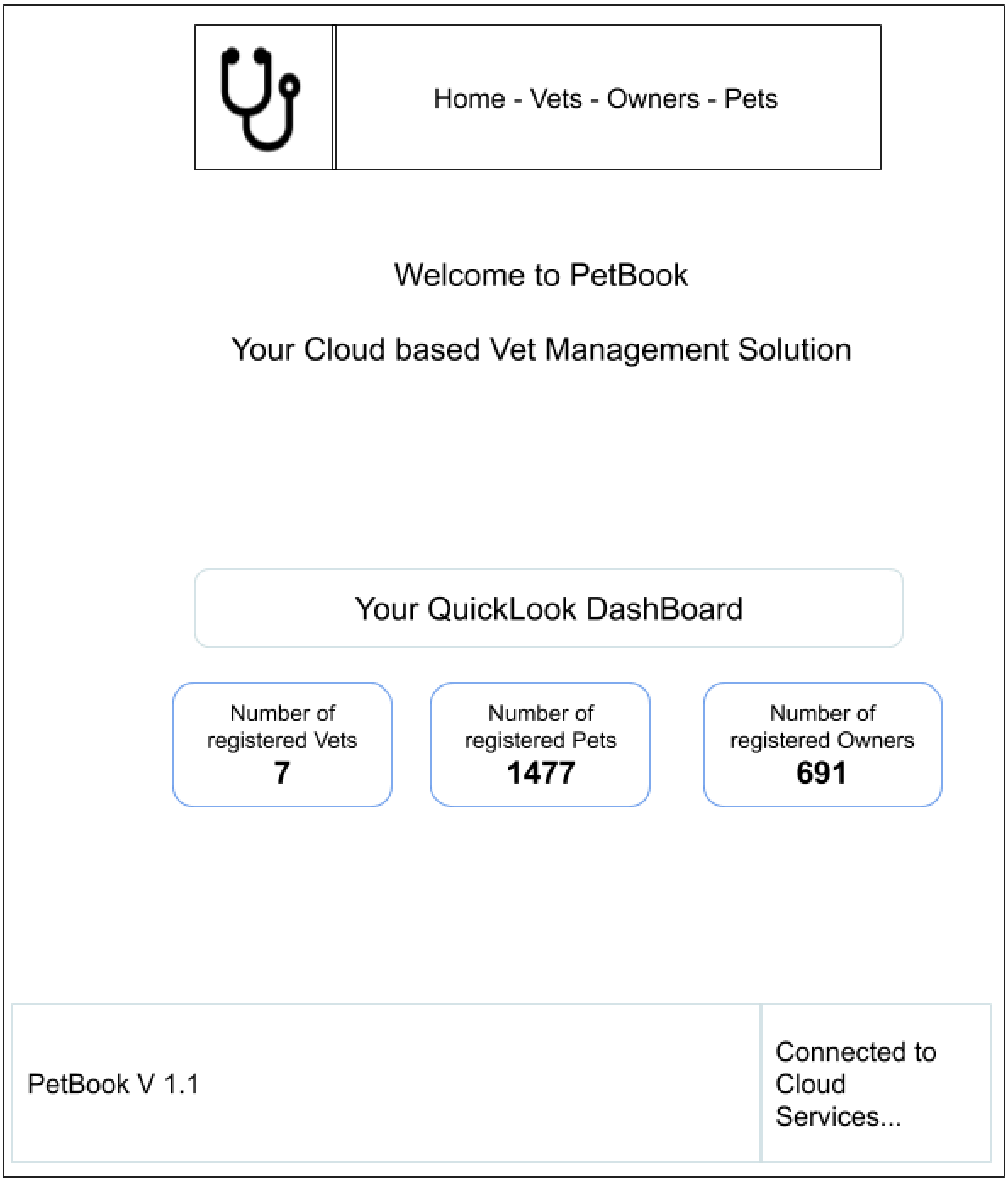
Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	Not enough memory to run the desired software which could make the application behave in unpredictable ways ie hanging or crashing	Increase the available ram
Performance Requirements	Fast enough to server x number of concurrent users. Users may experience dissatisfaction if the system fails to meet expectations. Users may leave the application before finishing the user journey	Purchase extra system resources
Persistent Storage and Transactions	The current size and type of the storage solution. Without sufficient storage space, enough relevant data may not be retained. Accessing persistent data is one of the core pieces of functionality of the application.	Increase the size and upgrade the type of storage solution ie Cloud Based
Usability	Difficult for user to efficiently use the system. The easier it is for the user to interact with the system, the more successful the system will be, allowing the user to meet their expectations and improve productivity.	Streamline the UX to improve workflow
Budgets	Limited funding prevents additional features and functionality being added. Only the features for which there is funding can be implemented. The budget must be realistic for the	Secure additional funding resources
Time Limitations	The product has to be delivered to the client in a certain date. Realistic timeframes are important in delivering product on time and on budget.	Increase the amount of personal working on the product to speed up delivery times.

Unit	Ref	Evidence	
P	P.5	User Site Map	

This site map represents the structure of the application. It essentially has three levels, therefore no page is more than 3 clicks away.



Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	



Welcome to PetBook

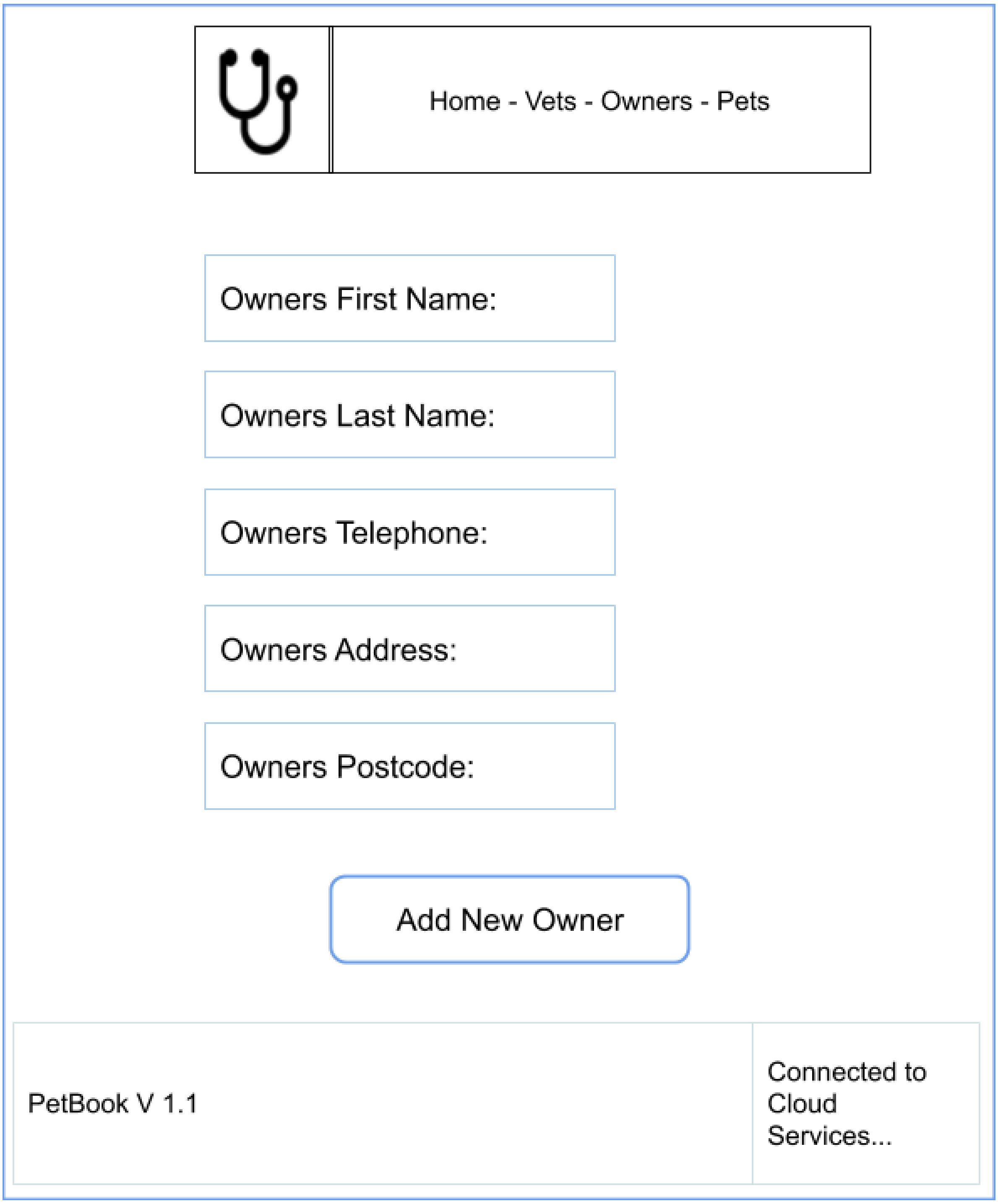
Your Cloud based Vet Management Solution

Your QuickLook DashBoard

Number of registered Vets	Number of registered Pets	Number of registered Owners
7	**1477**	**691**

PetBook V 1.1

Connected to Cloud Services...



Home - Vets - Owners - Pets

Owners First Name:

Owners Last Name:

Owners Telephone:

Owners Address:

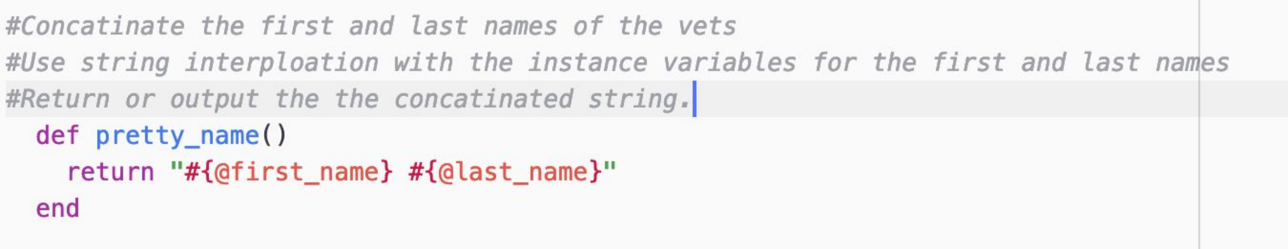
Owners Postcode:

PetBook V 1.1

Connected to
Cloud
Services...

Week 5

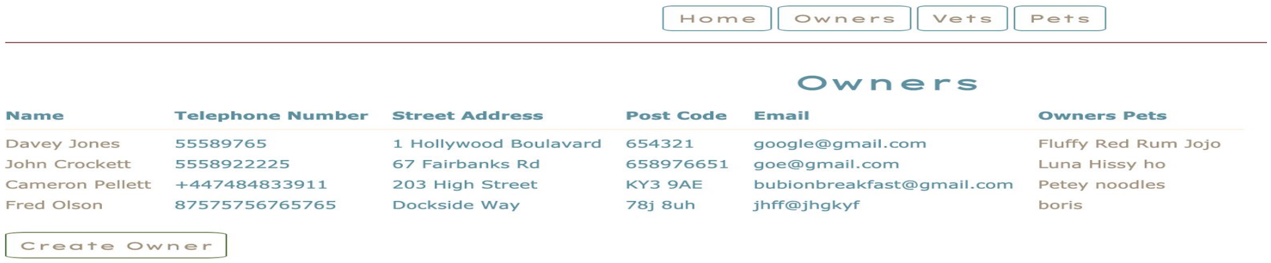
Unit	Ref	Evidence	
P	P.10	**Example of Pseudocode used for a method**	



```
#Concatinate the first and last names of the vets
#Use string interploation with the instance variables for the first and last names
#Return or output the the concatinated string.
  def pretty_name()
    return "#{@first_name} #{@last_name}"
  end
```

Unit	Ref	Evidence	
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	

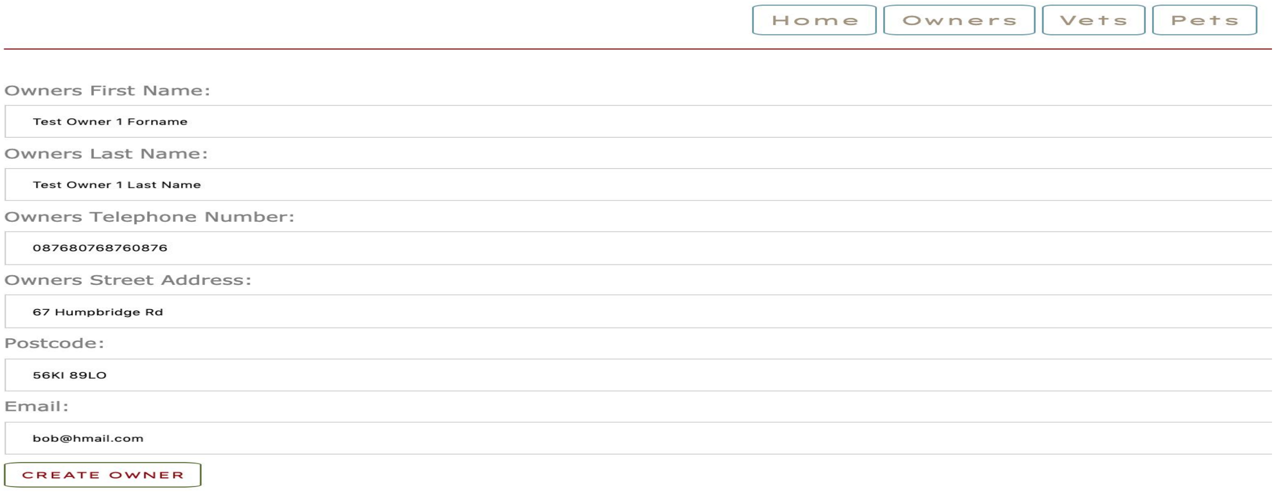
The current owner list



Owners

Name	Telephone Number	Street Address	Post Code	Email	Owners Pets
Davey Jones	55589765	1 Hollywood Boulavard	654321	google@gmail.com	Fluffy Red Rum Jojo
John Crockett	5558922225	67 Fairbanks Rd	658976651	goe@gmail.com	Luna Hissy ho
Cameron Pellett	+447484833911	203 High Street	KY3 9AE	bubionbreakfast@gmail.com	Petey noodles
Fred Olson	87575756765765	Dockside Way	78j 8uh	jhff@jhgkyf	boris

A new owner being added



Owners First Name:

Test Owner 1 Forname

Owners Last Name:

Test Owner 1 Last Name

Owners Telephone Number:

087680768760876

Owners Street Address:

67 Humpbridge Rd

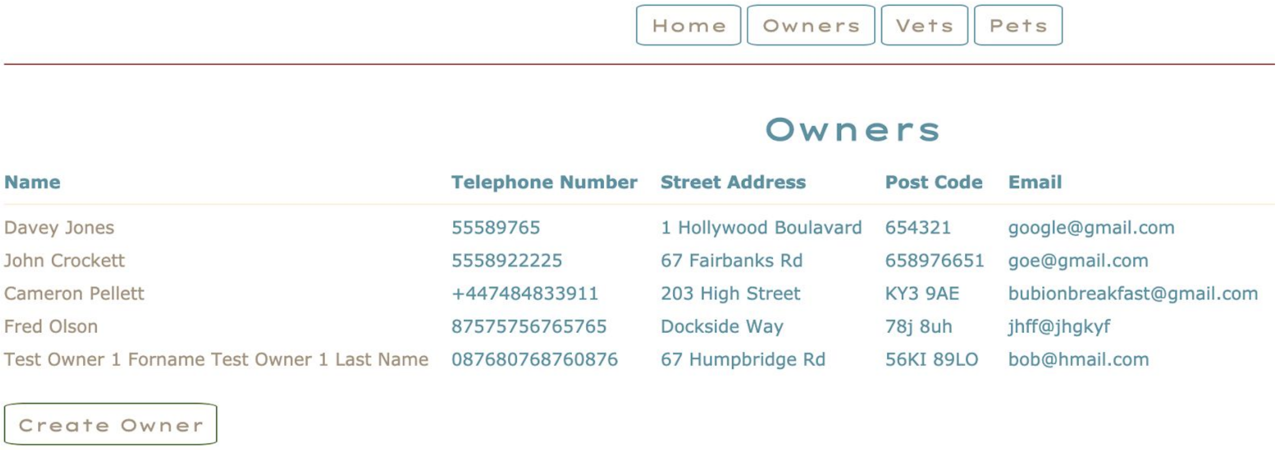
Postcode:

56KI 89LO

Email:

bob@hmail.com

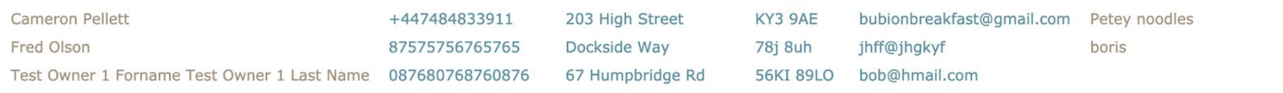
The updated owners list with the new owner appearing



Owners

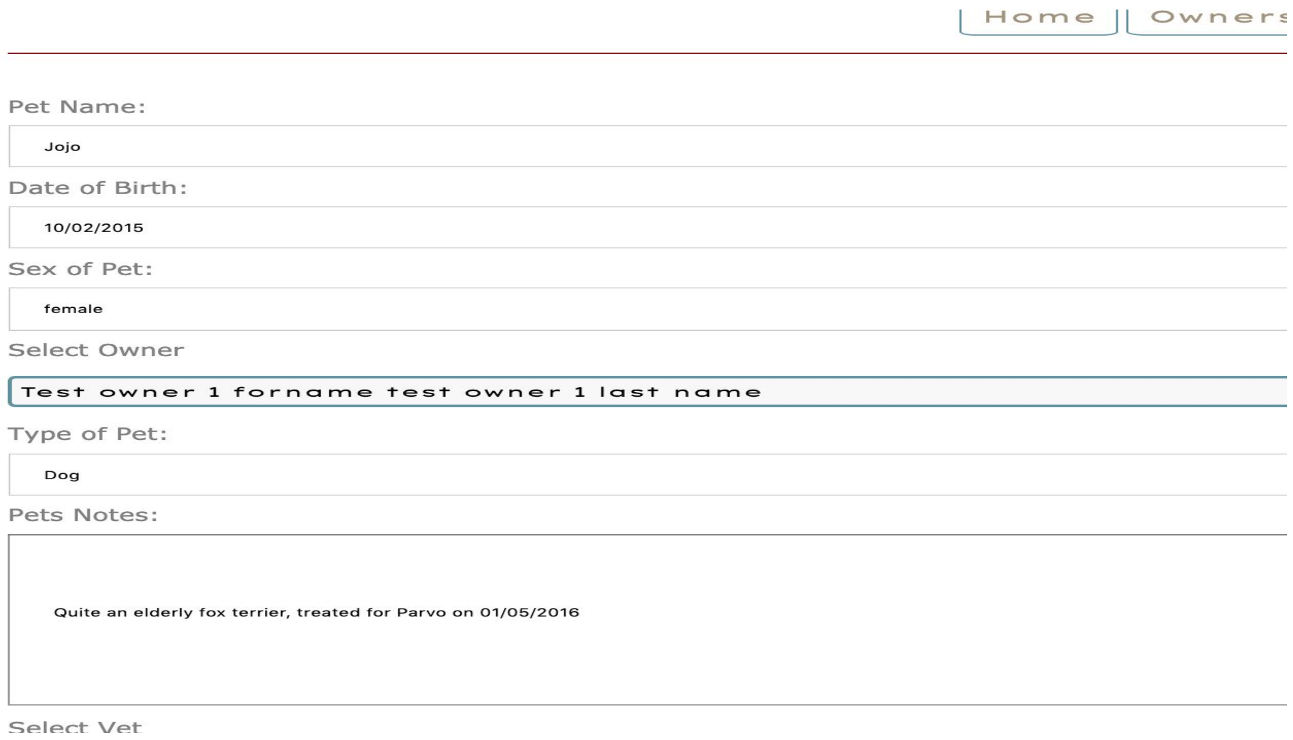
Name	Telephone Number	Street Address	Post Code	Email
Davey Jones	55589765	1 Hollywood Boulavard	654321	google@gmail.com
John Crockett	5558922225	67 Fairbanks Rd	658976651	goe@gmail.com
Cameron Pellett	+447484833911	203 High Street	KY3 9AE	bubionbreakfast@gmail.com
Fred Olson	87575756765765	Dockside Way	78j 8uh	jhff@jhgkyf
Test Owner 1 Forname Test Owner 1 Last Name	087680768760876	67 Humpbridge Rd	56KI 89LO	bob@hmail.com

Unit	Ref	Evidence	
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	



Cameron Pellett	+447484833911	203 High Street	KY3 9AE	bubionbreakfast@gmail.com	Petey noodles
Fred Olson	87575756765765	Dockside Way	78j 8uh	jhff@jhgkyf	boris
Test Owner 1 Forname Test Owner 1 Last Name	087680768760876	67 Humpbridge Rd	56KI 89LO	bob@hmail.com	

Test owner 1 has not yet been assigned to a pet.



Pet Name:

Jojo

Date of Birth:

10/02/2015

Sex of Pet:

female

Select Owner

Test owner 1 forname test owner 1 last name

Type of Pet:

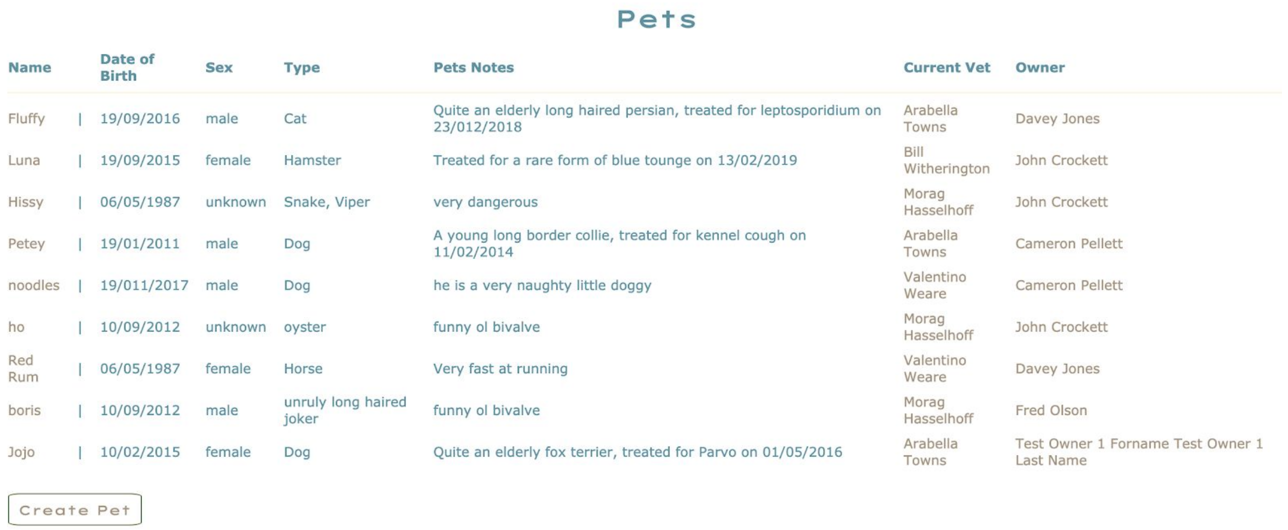
Dog

Pets Notes:

Quite an elderly fox terrier, treated for Parvo on 01/05/2016

Select Vet

Test owner has been assigned to a pet



Pets

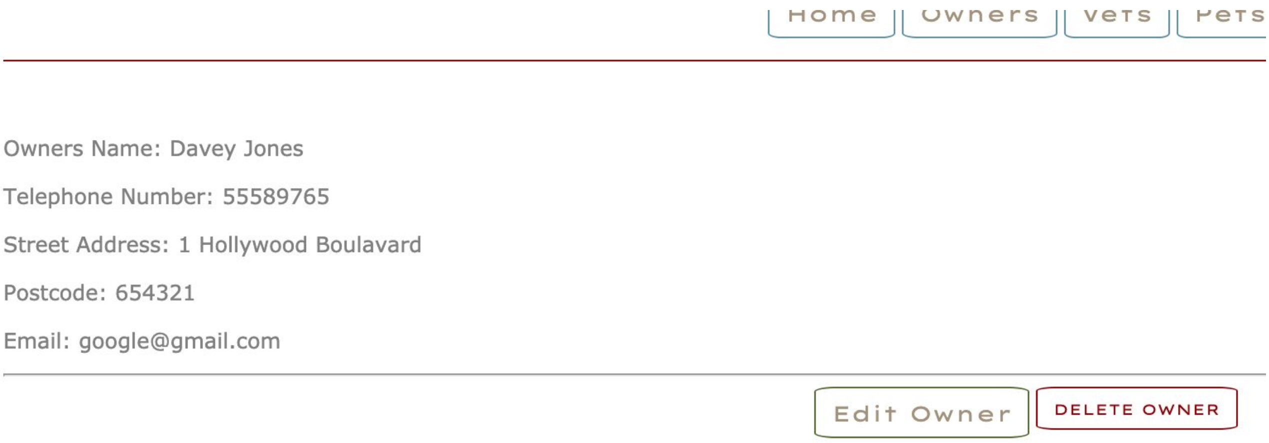
Name		Date of Birth	Sex	Type	Pets Notes	Current Vet	Owner
Fluffy	|	19/09/2016	male	Cat	Quite an elderly long haired persian, treated for leptosporidium on 23/012/2018	Arabella Towns	Davey Jones
Luna	|	19/09/2015	female	Hamster	Treated for a rare form of blue tounge on 13/02/2019	Bill Witherington	John Crockett
Hissy	|	06/05/1987	unknown	Snake, Viper	very dangerous	Morag Hasselhoff	John Crockett
Petey	|	19/01/2011	male	Dog	A young long border collie, treated for kennel cough on 11/02/2014	Arabella Towns	Cameron Pellett
noodles	|	19/011/2017	male	Dog	he is a very naughty little doggy	Valentino Weare	Cameron Pellett
ho	|	10/09/2012	unknown	oyster	funny ol bivalve	Morag Hasselhoff	John Crockett
Red Rum	|	06/05/1987	female	Horse	Very fast at running	Valentino Weare	Davey Jones
boris	|	10/09/2012	male	unruly long haired joker	funny ol bivalve	Morag Hasselhoff	Fred Olson
Jojo	|	10/02/2015	female	Dog	Quite an elderly fox terrier, treated for Parvo on 01/05/2016	Arabella Towns	Test Owner 1 Forname Test Owner 1 Last Name

Confirmation that the owner is now assigned to the pet.

Unit	Ref	Evidence	

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

The user can edit the owners phone number



Owners Name: Davey Jones

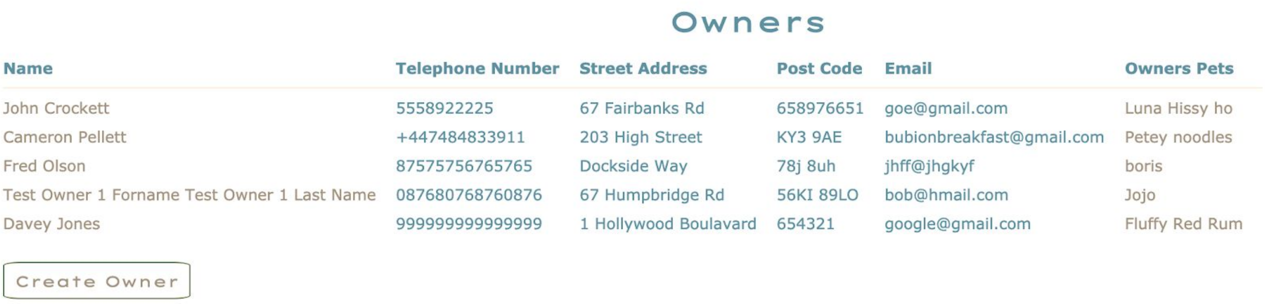
Telephone Number: 55589765

Street Address: 1 Hollywood Boulavard

Postcode: 654321

Email: google@gmail.com

The owners phone number is now edited



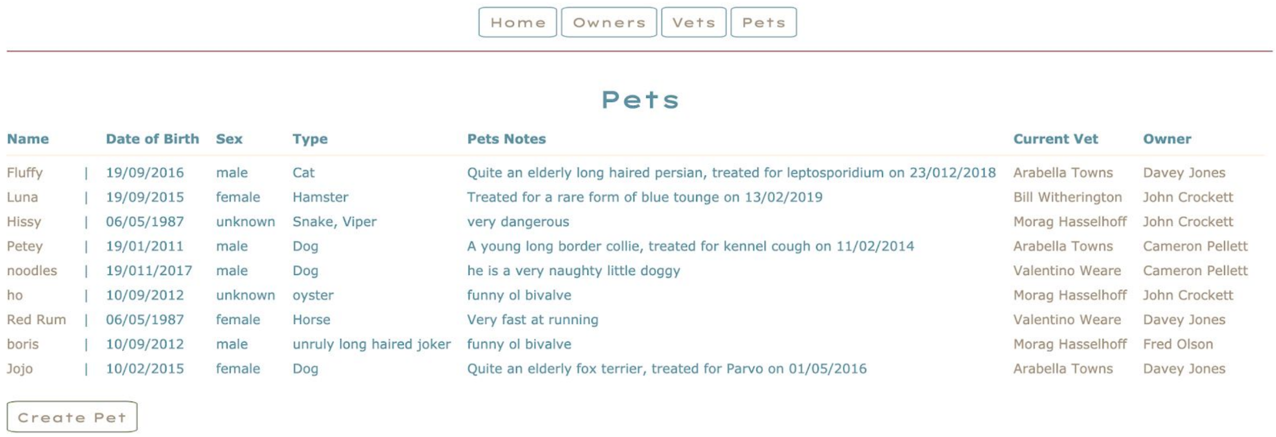
Owners

Name	Telephone Number	Street Address	Post Code	Email	Owners Pets
John Crockett	5558922225	67 Fairbanks Rd	658976651	goe@gmail.com	Luna Hissy ho
Cameron Pellett	+447484833911	203 High Street	KY3 9AE	bubionbreakfast@gmail.com	Petey noodles
Fred Olson	87575756765765	Dockside Way	78j 8uh	jhff@jhgkyf	boris
Test Owner 1 Forname Test Owner 1 Last Name	087680768760876	67 Humpbridge Rd	56KI 89LO	bob@hmail.com	Jojo
Davey Jones	999999999999999	1 Hollywood Boulavard	654321	google@gmail.com	Fluffy Red Rum

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

Vet Management App A veterinary practice has approached you to build a web application to help them manage their animals and vets. A vet may look after many animals at a time. An animal is registered with only one vet. This App is built using an Object Oriented Model. It has Classes that define individual instances of each object. This App is written in: Ruby HTML CSS Uses RESTful Routes with Sinatra

https://github.com/bubionbreakfast/vet_app

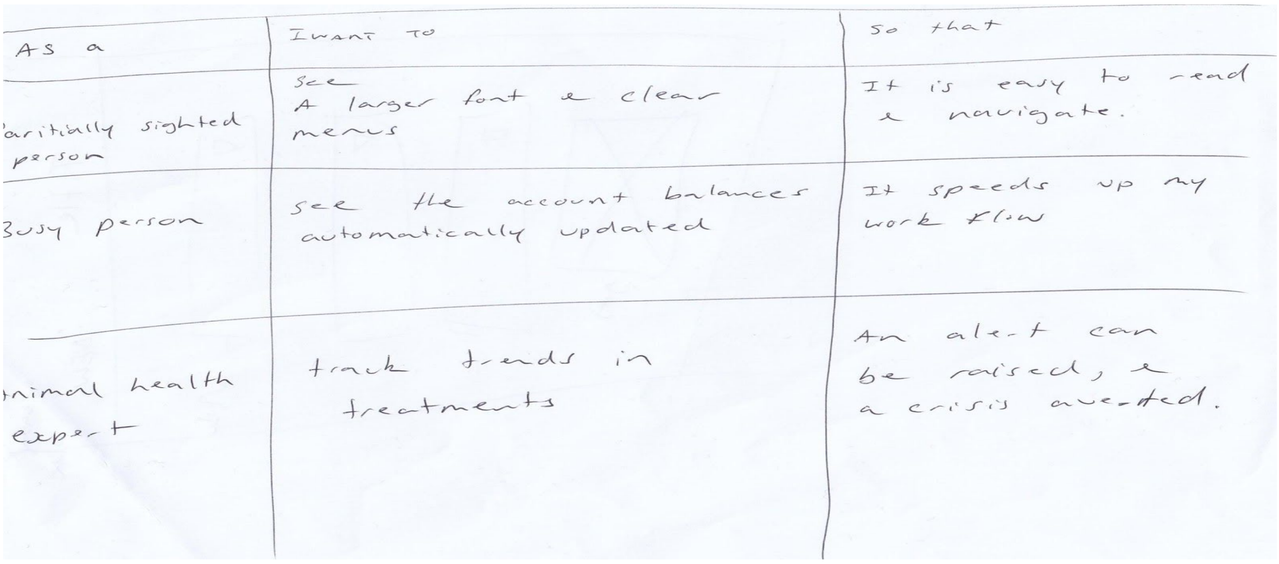


Pets

Name	Date of Birth	Sex	Type	Pets Notes	Current Vet	Owner
Fluffy	19/09/2016	male	Cat	Quite an elderly long haired persian, treated for leptosporidium on 23/012/2018	Arabella Towns	Davey Jones
Luna	19/09/2015	female	Hamster	Treated for a rare form of blue tounge on 13/02/2019	Bill Witherington	John Crockett
Hissy	06/05/1987	unknown	Snake, Viper	very dangerous	Morag Hasselhoff	John Crockett
Petey	19/01/2011	male	Dog	A young long border collie, treated for kennel cough on 11/02/2014	Arabella Towns	Cameron Pellett
noodles	19/011/2017	male	Dog	he is a very naughty little doggy	Valentino Weare	Cameron Pellett
ho	10/09/2012	unknown	oyster	funny ol bivalve	Morag Hasselhoff	John Crockett
Red Rum	06/05/1987	female	Horse	Very fast at running	Valentino Weare	Davey Jones
boris	10/09/2012	male	unruly long haired joker	funny ol bivalve	Morag Hasselhoff	Fred Olson
Jojo	10/02/2015	female	Dog	Quite an elderly fox terrier, treated for Parvo on 01/05/2016	Arabella Towns	Davey Jones

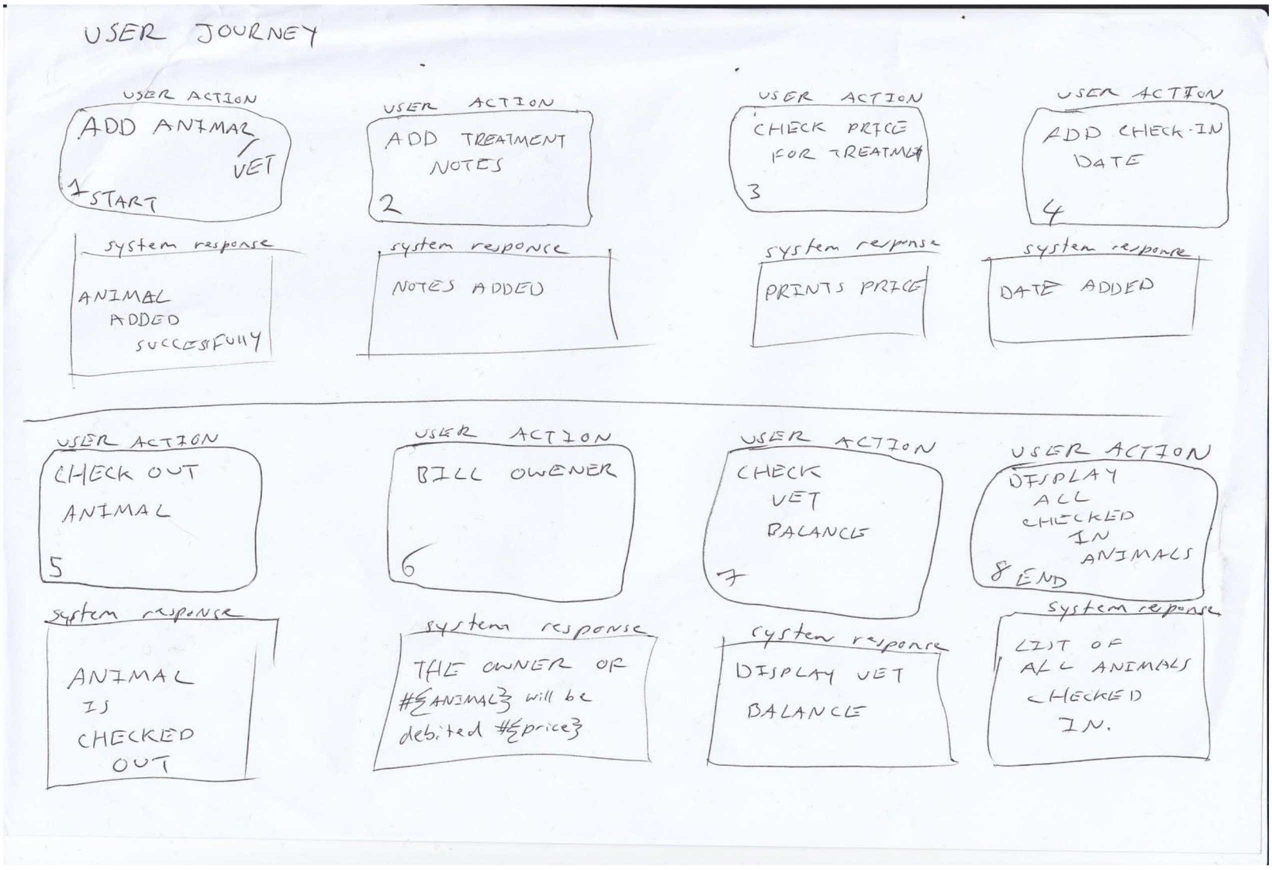
Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	

User Needs planning. Three different users need detailed with their differing requiremnents.



As a	I want to	So that
aritially sighted person	See A larger font & clear menus	It is easy to read & navigate.
3usy person	see the account balances automatically updated	It speeds up my work flow
nimal health expert	track trends in treatments	An alert can be raised, & a crisis averted.

A user journey through the application.



USER JOURNEY

USER ACTION
ADD ANIMAL / VET
1 START

system response
ANIMAL ADDED SUCCESSFully

USER ACTION
ADD TREATMENT NOTES
2

system responce
NOTES ADDED

USER ACTION
CHECK PRICE FOR TREATMEN
3

system response
PRINTS PRICE

USER ACTION
ADD CHECK-IN DATE
4

system responce
DATE ADDED

USER ACTION
CHECK OUT ANIMAL
5

system response
ANIMAL IS CHECKED OUT

USER ACTION
BILL OWENER
6

system response
THE OWNER OF #{ANIMAL} will be debited #{price}

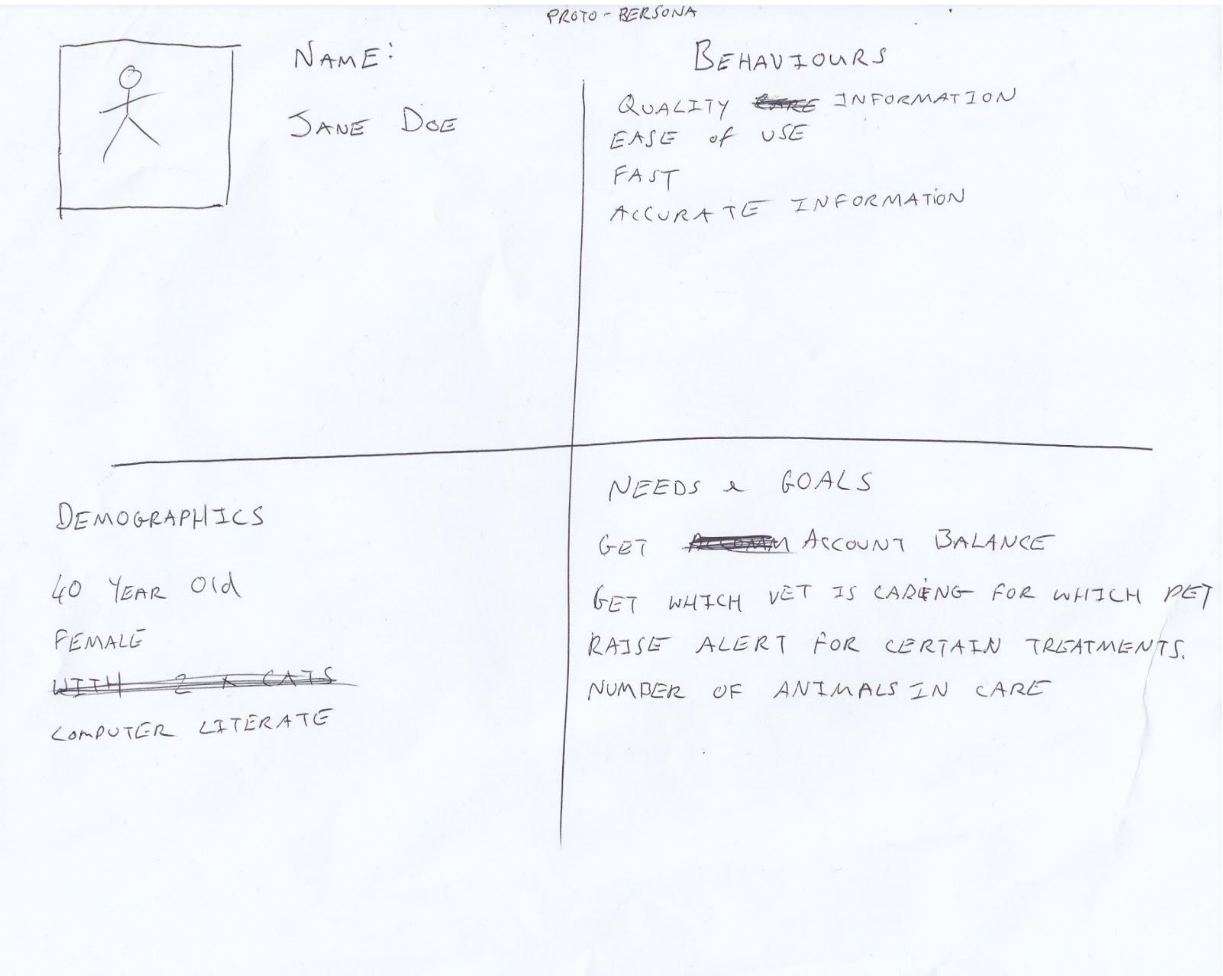
USER ACTION
CHECK VET BALANCE
7

cystem responce
DISPLAY VET BALANCE

USER ACTION
DISPLAY ALL CHECKED IN ANIMALS
8 END

system response
LIST OF ALL ANIMALS CHECKED IN.

Protopersona planning,



PROTO - PERSONA

NAME:

JANE DOE

BEHAVIOURS

QUALITY ~~CARE~~ INFORMATION

EASE of USE

FAST

ACCURATE INFORMATION

DEMOGRAPHICS

40 YEAR Old

FEMALE

~~WITH 2 x CATS~~

COMPUTER LITERATE

NEEDS & GOALS

GET ~~ACCOUM~~ ACCOUNT BALANCE

GET WHICH VET IS CARING FOR WHICH PET

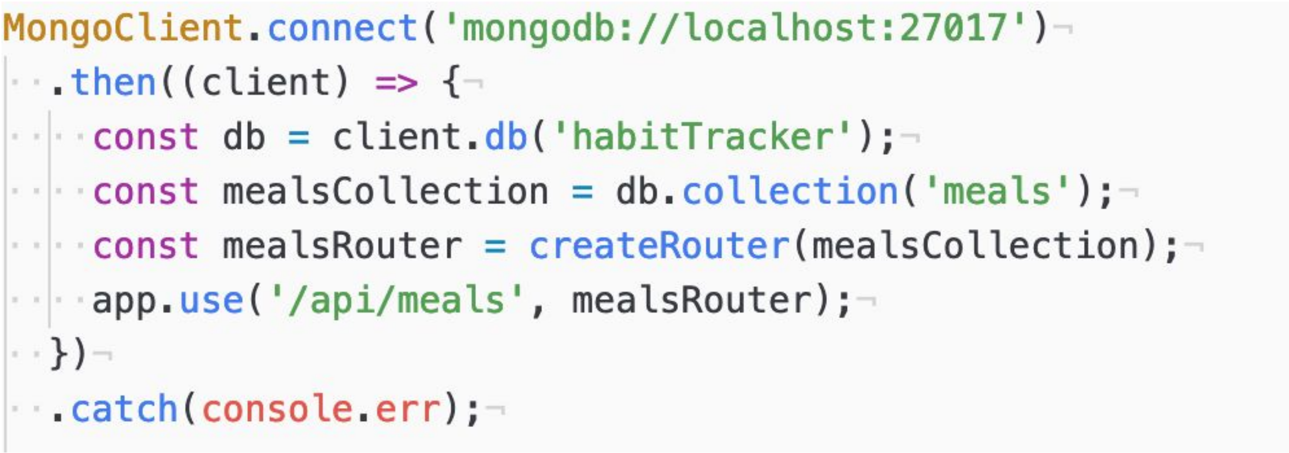
RAISE ALERT FOR CERTAIN TREATMENTS.

NUMPER OF ANIMALS IN CARE

Week 7

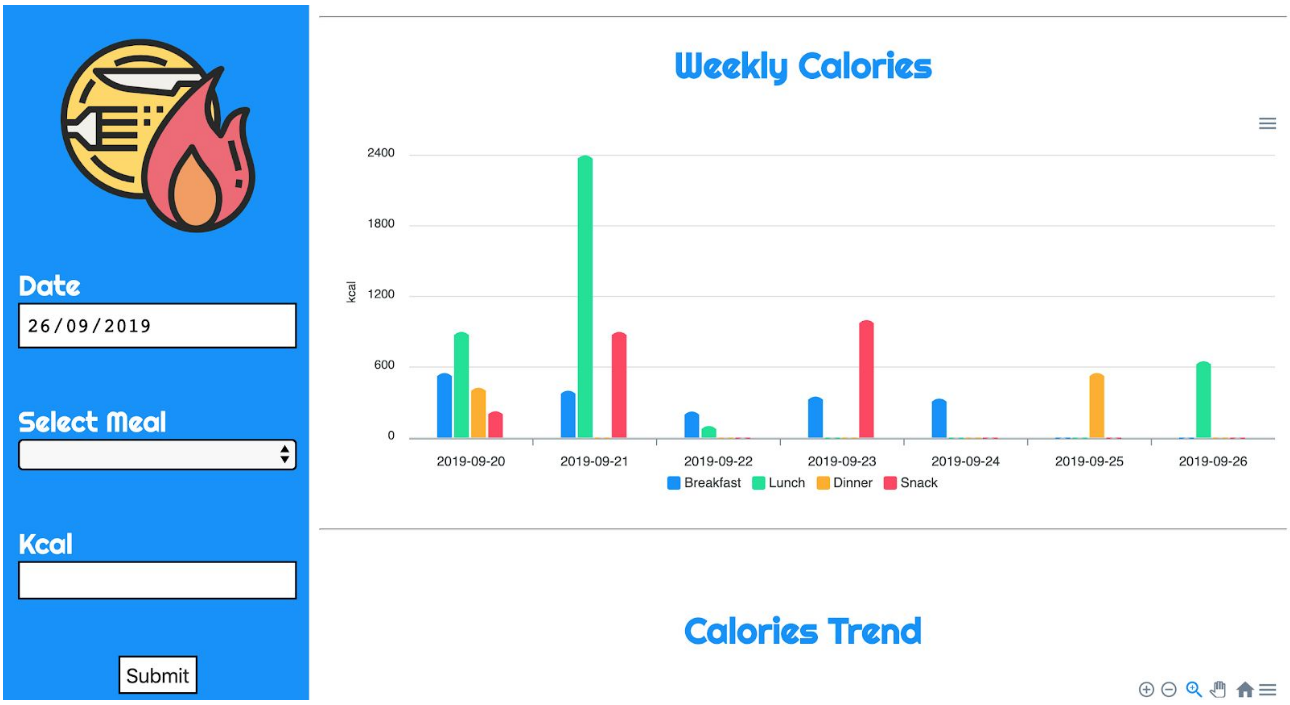
Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	

This is the code that connects to the database imlpementing the API



```
MongoClient.connect('mongodb://localhost:27017')
  .then((client) => {
    const db = client.db('habitTracker');
    const mealsCollection = db.collection('meals');
    const mealsRouter = createRouter(mealsCollection);
    app.use('/api/meals', mealsRouter);
  })
  .catch(console.err);
```

API bieng used while the programme is running



Date

26/09/2019

Select Meal

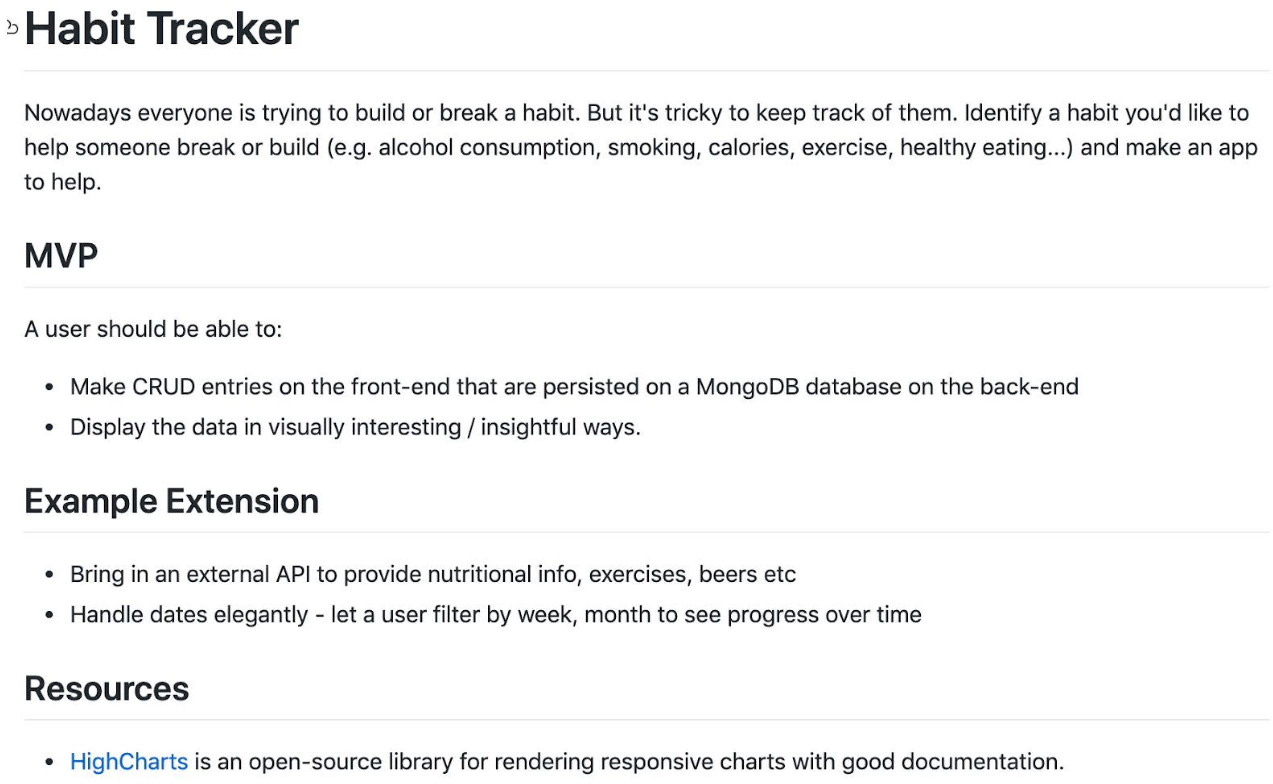
Kcal

Submit

Week 8

Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	

Project group project brief



Habit Tracker

Nowadays everyone is trying to build or break a habit. But it's tricky to keep track of them. Identify a habit you'd like to help someone break or build (e.g. alcohol consumption, smoking, calories, exercise, healthy eating...) and make an app to help.

MVP

A user should be able to:

- Make CRUD entries on the front-end that are persisted on a MongoDB database on the back-end
- Display the data in visually interesting / insightful ways.

Example Extension

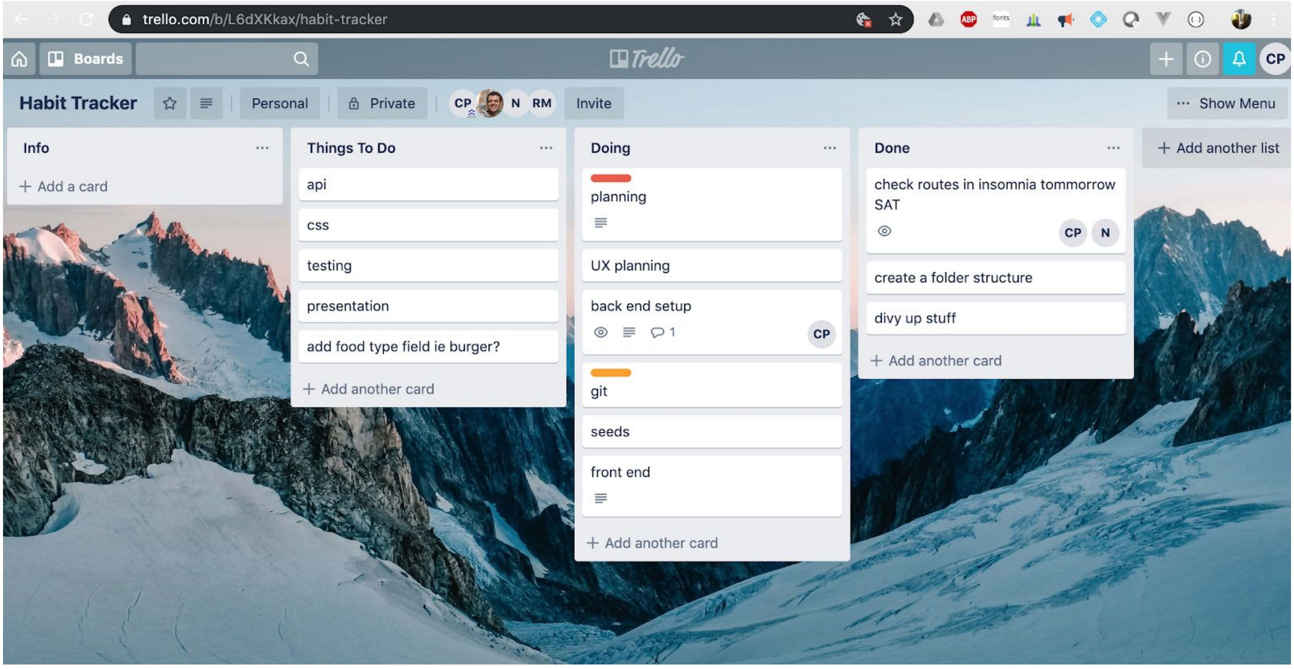
- Bring in an external API to provide nutritional info, exercises, beers etc
- Handle dates elegantly - let a user filter by week, month to see progress over time

Resources

- HighCharts is an open-source library for rendering responsive charts with good documentation.

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

Group project planning on the Kanban Board Trello



Info

+ Add a card

Things To Do

- api
- css
- testing
- presentation
- add food type field ie burger?

+ Add another card

Doing

- planning
- UX planning
- back end setup
- git
- seeds
- front end

+ Add another card

Done

- check routes in insomnia tommorrow SAT
- create a folder structure
- divy up stuff

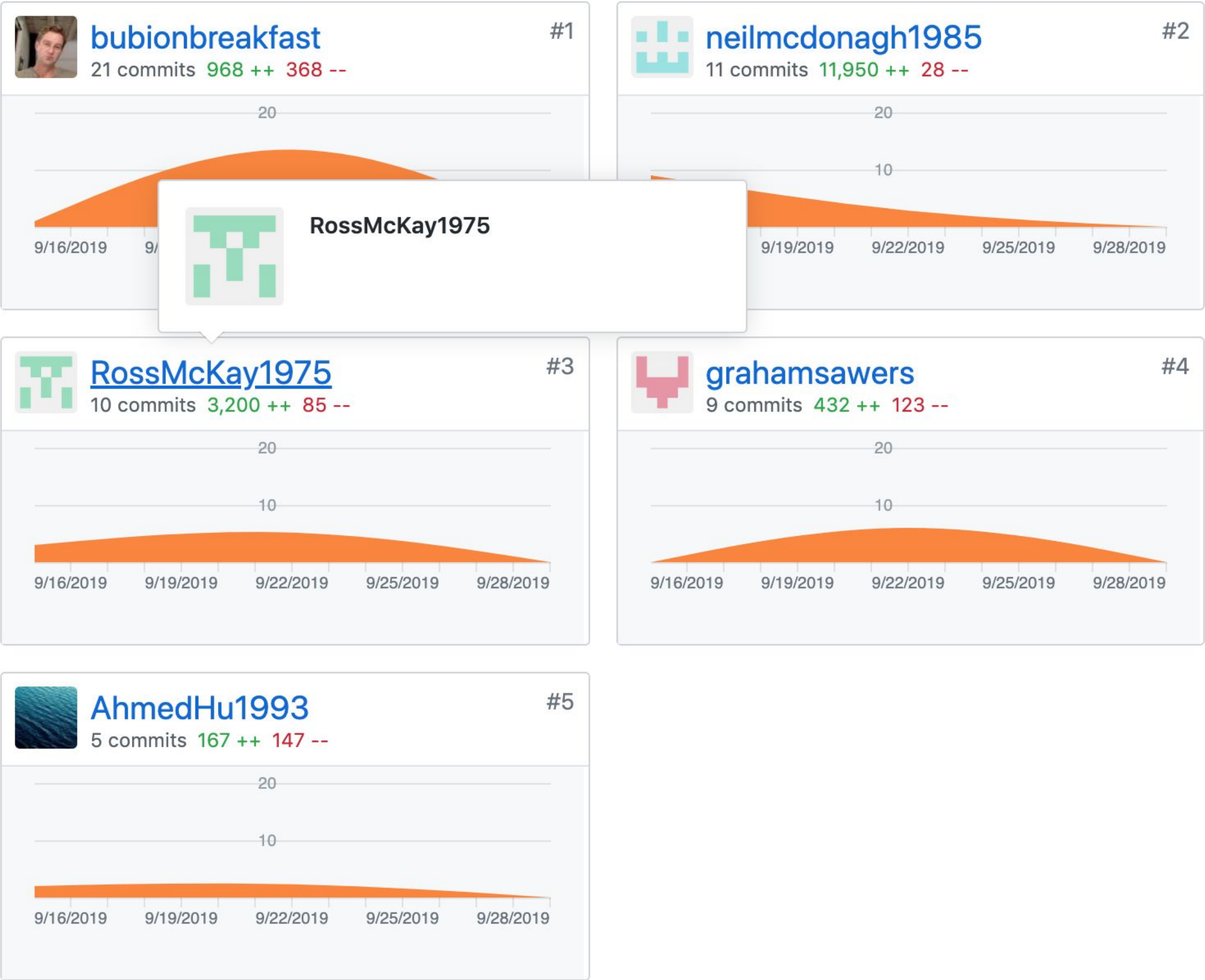
+ Add another card

Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

Acceptance Criteria	**Expected Result**	**Pass/Fail**
A user is able to add a meal	breakfast appears	**pass**
A user is able to add calories	the number of calories appears next to the meal	**pass**
A user is able to view 7 days of data	A graph with 7 days of recorded data appears	**pass**
A user can see calories trends over time	A graph with all of the users data appears showing increase/decrease	**pass**
A user can delete a meal if added erroneously	The meal data is destroyed, and removed from the list	**pass**

Week 9

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



bubionbreakfast #1
21 commits 968 ++ 368 --

neilmcdonagh1985 #2
11 commits 11,950 ++ 28 --

RossMcKay1975

RossMcKay1975 #3
10 commits 3,200 ++ 85 --

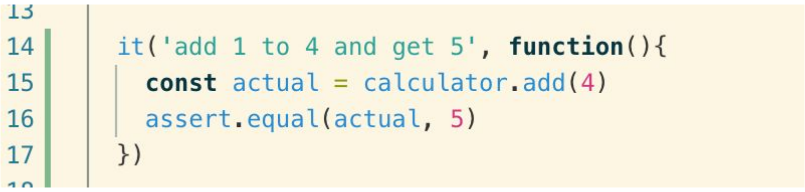
grahamsawers #4
9 commits 432 ++ 123 --

AhmedHu1993 #5
5 commits 167 ++ 147 --

Week 11

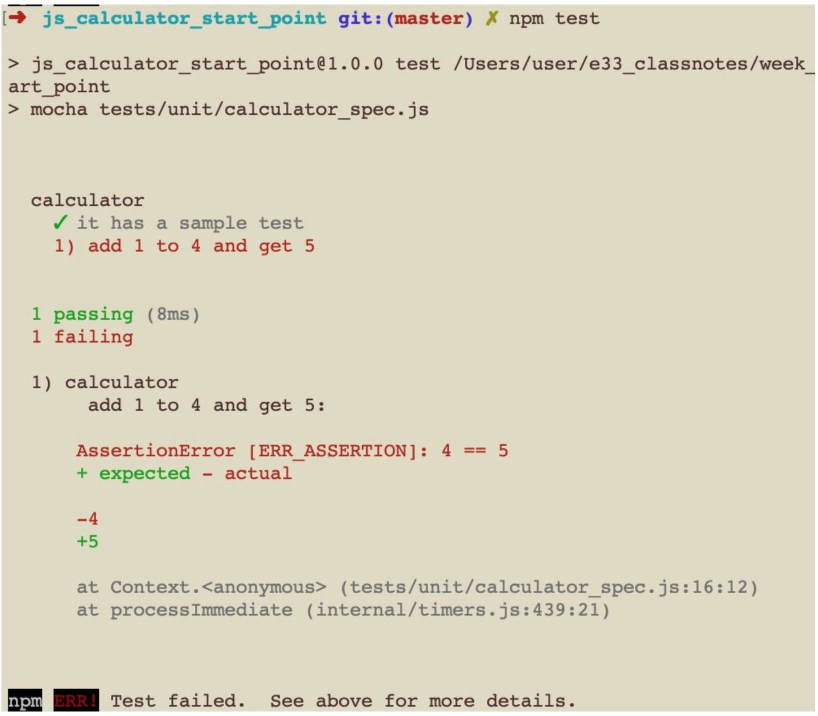
Unit	Ref	Evidence	
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	

Example of test code.



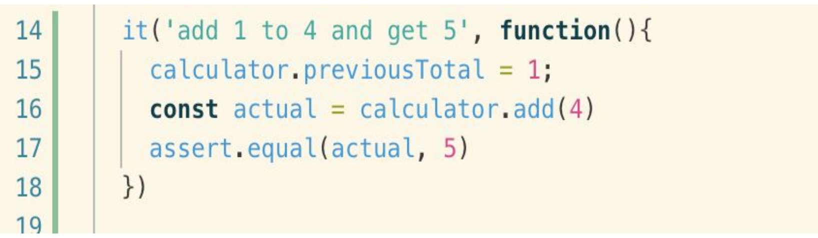
```
it('add 1 to 4 and get 5', function(){
  const actual = calculator.add(4)
  assert.equal(actual, 5)
})
```

Test code failing to pass.



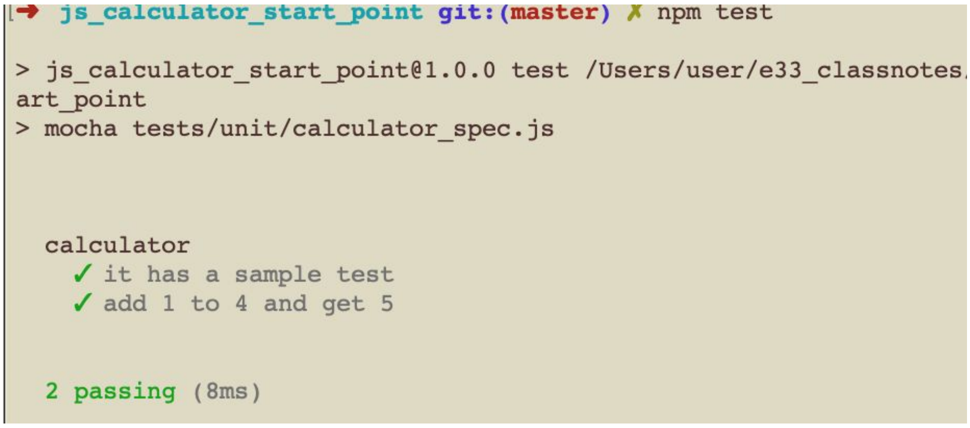
```
[➜  js_calculator_start_point git:(master) ✗ npm test

> js_calculator_start_point@1.0.0 test /Users/user/e33_classnotes/week_
art_point
> mocha tests/unit/calculator_spec.js



  calculator
    ✓ it has a sample test
    1) add 1 to 4 and get 5


  1 passing (8ms)
  1 failing

  1) calculator
       add 1 to 4 and get 5:

      AssertionError [ERR_ASSERTION]: 4 == 5
      + expected - actual

      -4
      +5

      at Context.<anonymous> (tests/unit/calculator_spec.js:16:12)
      at processImmediate (internal/timers.js:439:21)



npm ERR! Test failed.  See above for more details.
```

_Example of the test code once errors have been corrected



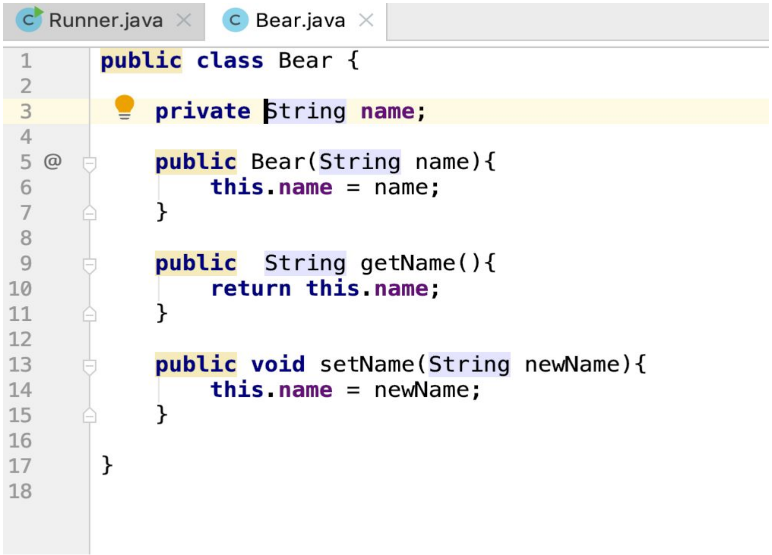
```
it('add 1 to 4 and get 5', function(){
  calculator.previousTotal = 1;
  const actual = calculator.add(4)
  assert.equal(actual, 5)
})
```

Test code passing.



```
➜  js_calculator_start_point git:(master) ✗ npm test

> js_calculator_start_point@1.0.0 test /Users/user/e33_classnotes/
art_point
> mocha tests/unit/calculator_spec.js



  calculator
    ✓ it has a sample test
    ✓ add 1 to 4 and get 5


  2 passing (8ms)
```

Unit	Ref	Evidence	
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.	

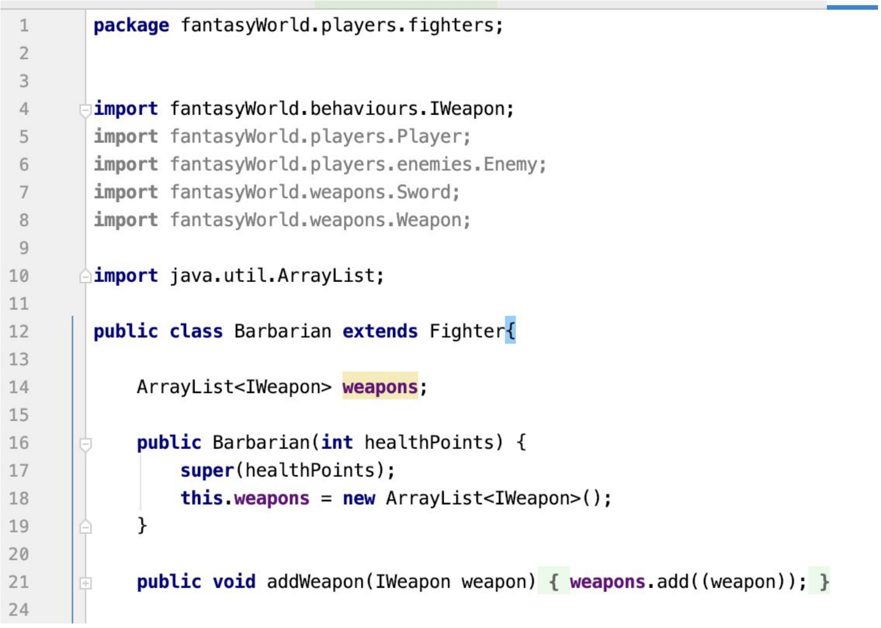


```
public class Bear {

    private String name;

    public Bear(String name){
        this.name = name;
    }

    public  String getName(){
        return this.name;
    }

    public void setName(String newName){
        this.name = newName;
    }

}
```

Here the Bear class encapsulates the data in the form of the name of the Bear. This data is restricted to be private to the class of Bear. The class includes a method in the form of “getName” to make this data available for other parts of the program. Therefore the Bear class is an example of encapsulation

Week 12

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	

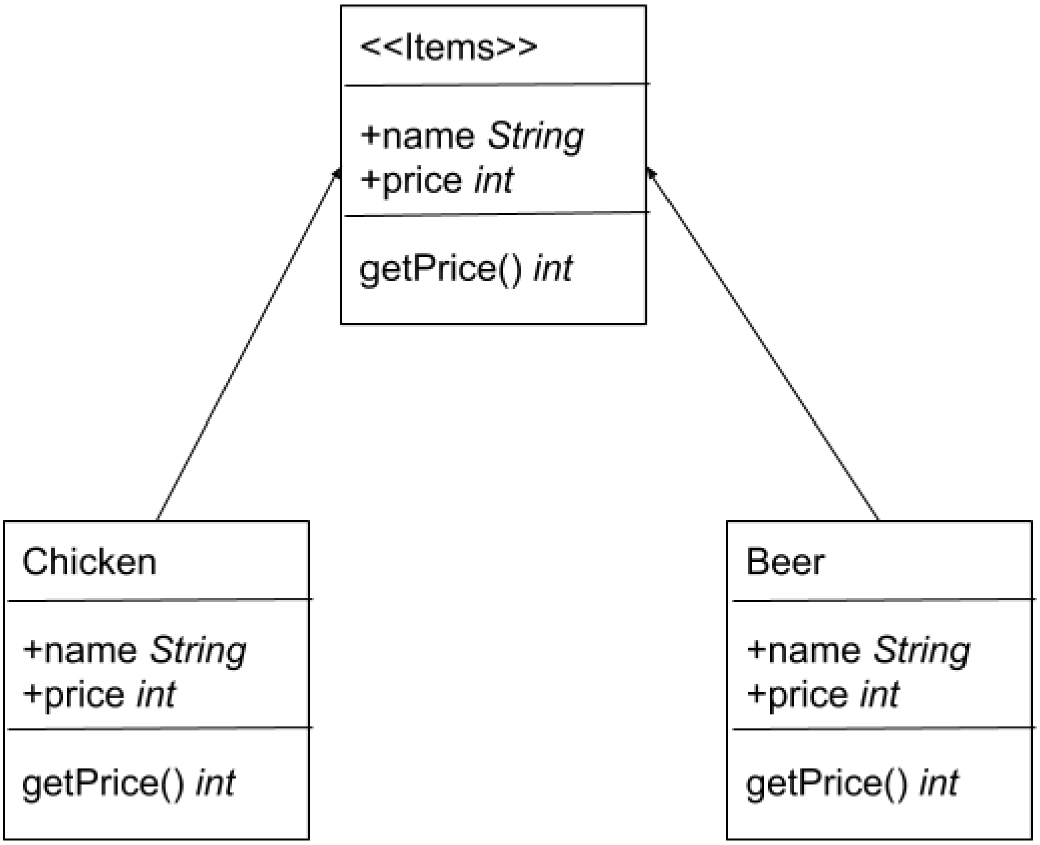


```
package fantasyWorld.players.fighters;


import fantasyWorld.behaviours.IWeapon;
import fantasyWorld.players.Player;
import fantasyWorld.players.enemies.Enemy;
import fantasyWorld.weapons.Sword;
import fantasyWorld.weapons.Weapon;

import java.util.ArrayList;

public class Barbarian extends Fighter{

    ArrayList<IWeapon> weapons;

    public Barbarian(int healthPoints) {
        super(healthPoints);
        this.weapons = new ArrayList<IWeapon>();
    }

    public void addWeapon(IWeapon weapon) { weapons.add((weapon)); }

```

This is an example of polymorphism, where the barbarian character can have/change weapons through the IWeapon interface. These can be added as IWeapon objects to the weapons array.

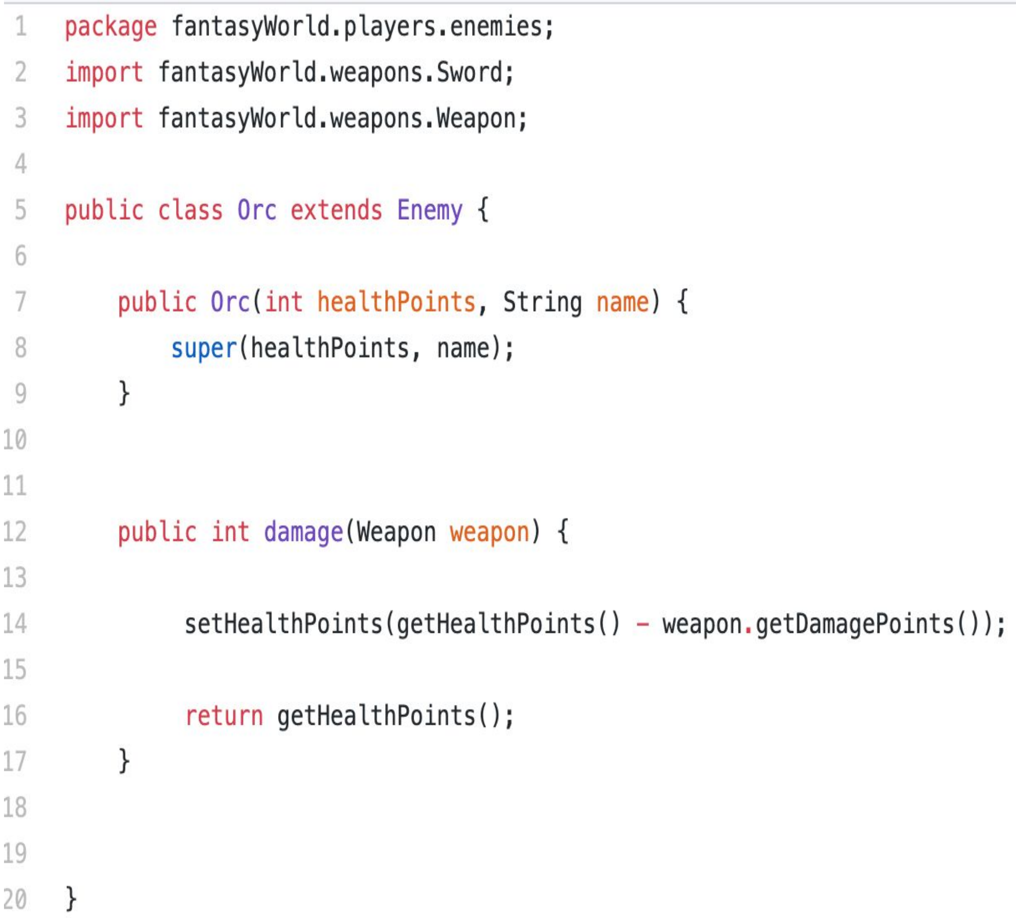
Unit	Ref	Evidence	
A&D	A.D.5	An Inheritance Diagram	

In this diagram both the Chicken sub-class and Beer subclasses inherit the properties of +name, +price and also the method getPrice() from the Items super class.



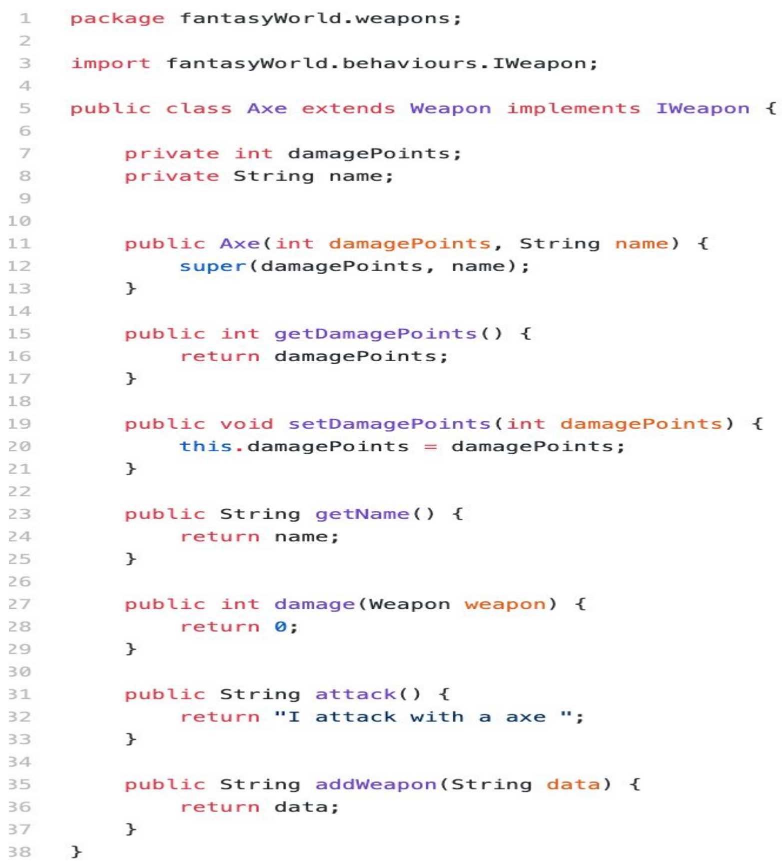
Unit	Ref	Evidence	
I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screenshots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.	

This is the Class Orc, it inherits from the previous Class of Enemy. This Class also includes the inherited weapon object. This Class has the inherited method called damage, which takes in the weapon object.



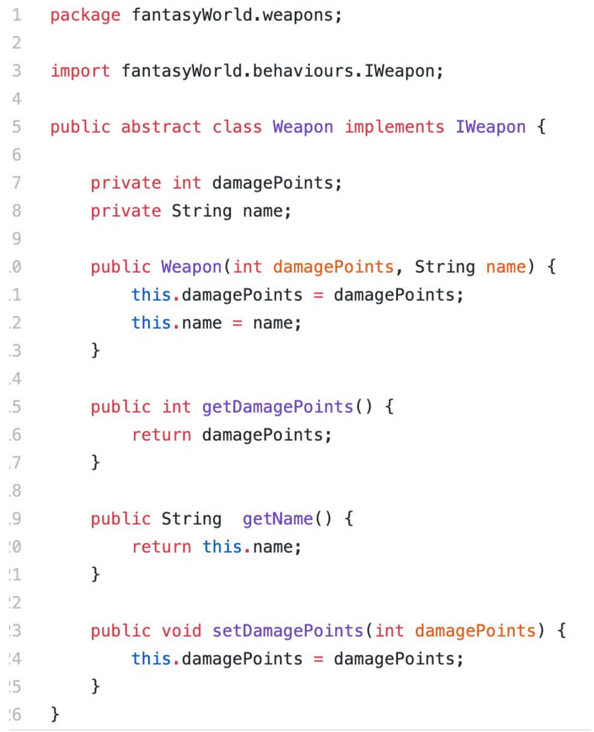
```
package fantasyWorld.players.enemies;
import fantasyWorld.weapons.Sword;
import fantasyWorld.weapons.Weapon;

public class Orc extends Enemy {

    public Orc(int healthPoints, String name) {
        super(healthPoints, name);
    }


    public int damage(Weapon weapon) {

         setHealthPoints(getHealthPoints() - weapon.getDamagePoints());

         return getHealthPoints();
    }


}
```

An Object in the inherited class



```
package fantasyWorld.weapons;

import fantasyWorld.behaviours.IWeapon;

public class Axe extends Weapon implements IWeapon {

    private int damagePoints;
    private String name;


    public Axe(int damagePoints, String name) {
        super(damagePoints, name);
    }

    public int getDamagePoints() {
        return damagePoints;
    }

    public void setDamagePoints(int damagePoints) {
        this.damagePoints = damagePoints;
    }

    public String getName() {
        return name;
    }

    public int damage(Weapon weapon) {
        return 0;
    }

    public String attack() {
        return "I attack with a axe ";
    }

    public String addWeapon(String data) {
        return data;
    }
}
```

Here is the Method that is used the information inherited from another class.

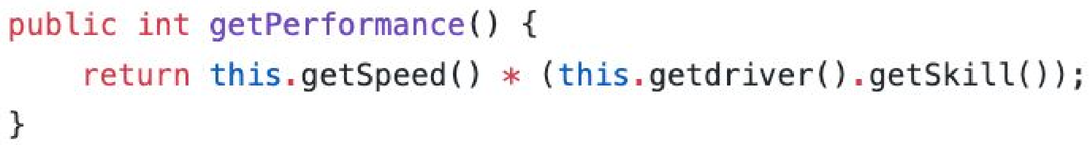


```
package fantasyWorld.weapons;

import fantasyWorld.behaviours.IWeapon;

public abstract class Weapon implements IWeapon {

    private int damagePoints;
    private String name;

    public Weapon(int damagePoints, String name) {
        this.damagePoints = damagePoints;
        this.name = name;
    }

    public int getDamagePoints() {
        return damagePoints;
    }

    public String  getName() {
        return this.name;
    }

    public void setDamagePoints(int damagePoints) {
        this.damagePoints = damagePoints;
    }
}
```

Week 14

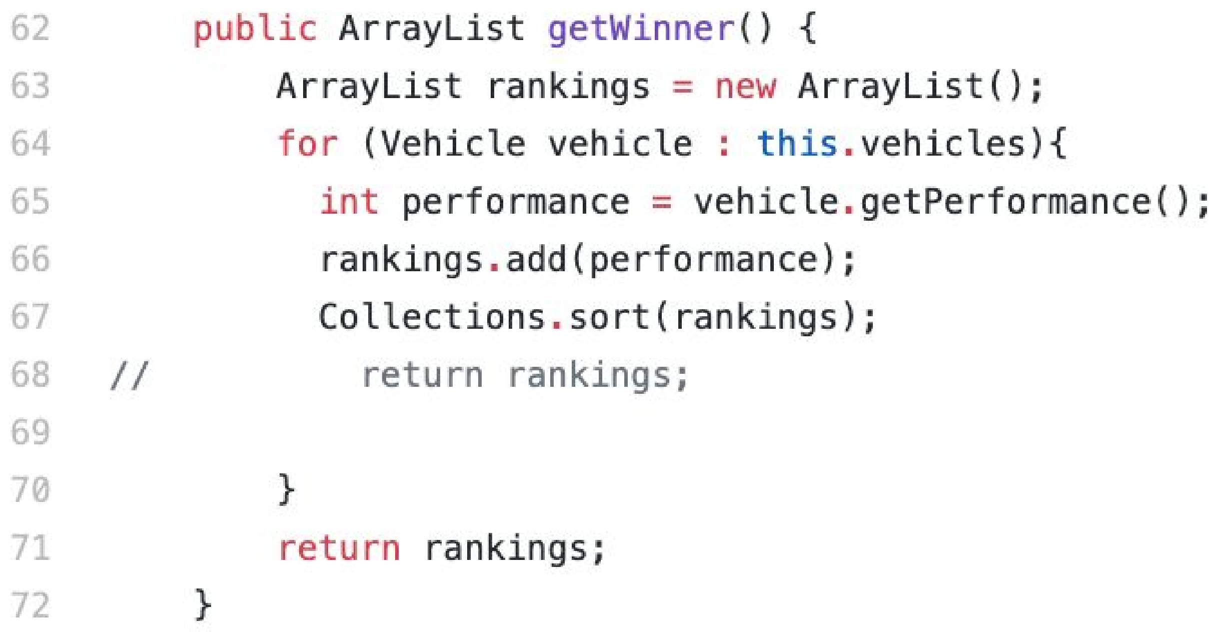
Unit	Ref	Evidence	
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.	

This algorithm returns the performance of a vehicle object. It gets the speed of the vehicle and multiples that by the selected drivers skill level. I have chosen this means that each combination of vehicle and river can return a different result.



```java
public int getPerformance() {
    return this.getSpeed() * (this.getdriver().getSkill());
}
```

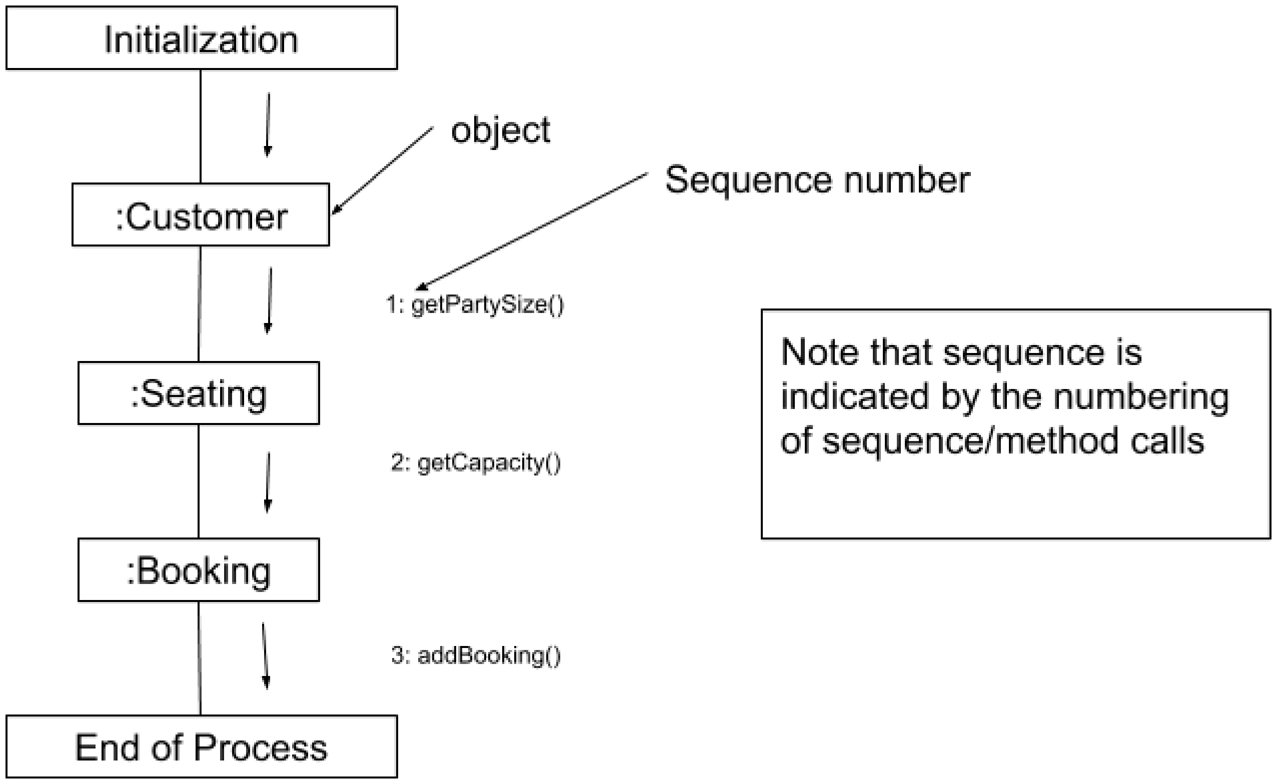
This algorithm returns a list of the results of the race in order of results.   I have chosen this as it is the most important part of this programme, taking in all the vehicles in the race to a list which is then sorted to return the winner.



```java
    public ArrayList getWinner() {
        ArrayList rankings = new ArrayList();
        for (Vehicle vehicle : this.vehicles){
          int performance = vehicle.getPerformance();
          rankings.add(performance);
          Collections.sort(rankings);
//            return rankings;

        }
        return rankings;
    }
```

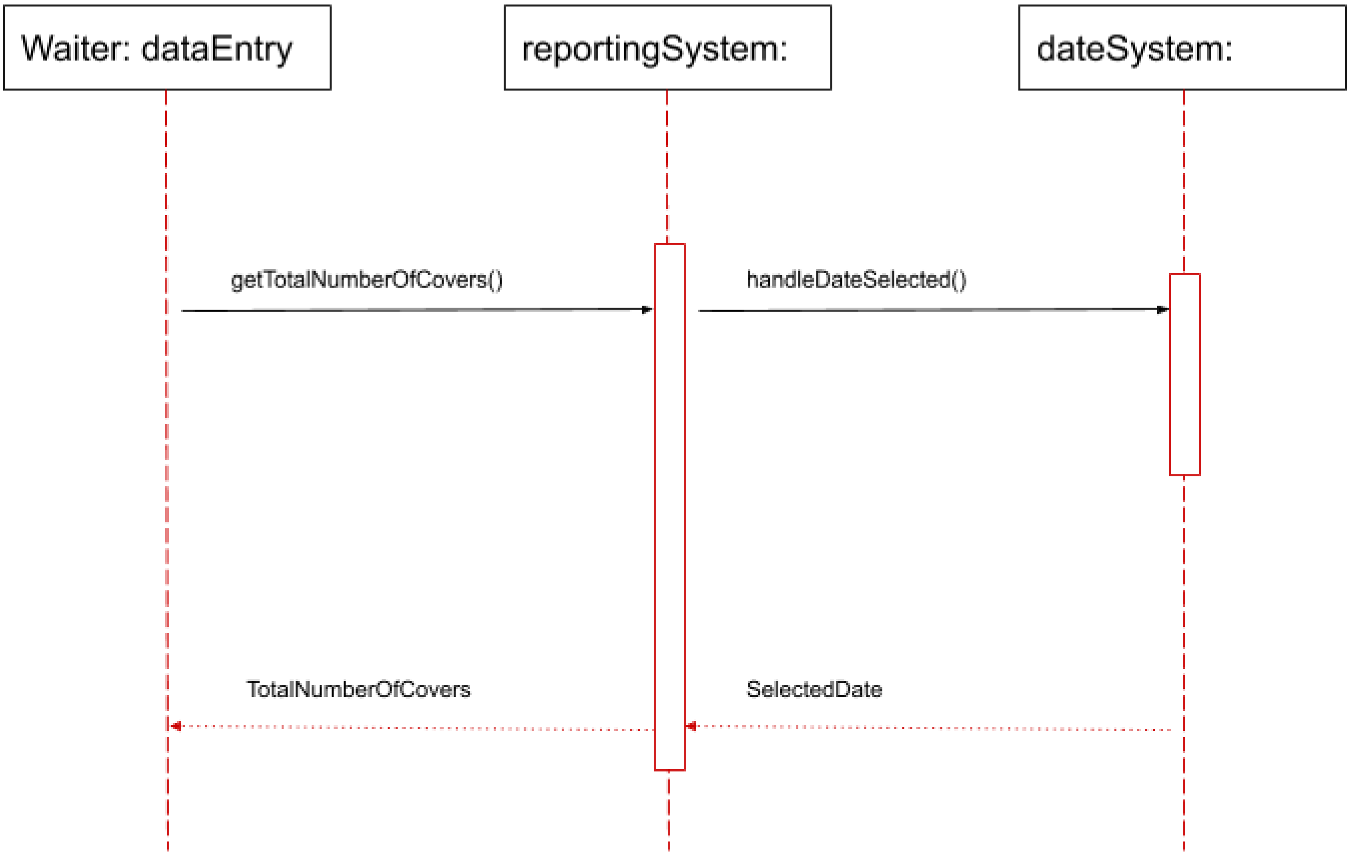
Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	

Collaboration diagram of restaurant booking system

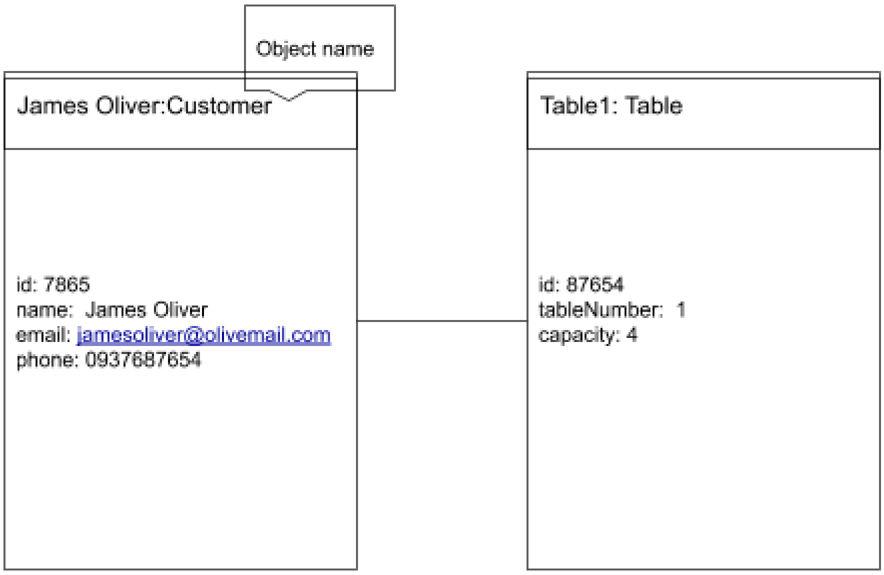


Note that sequence is indicated by the numbering of sequence/method calls

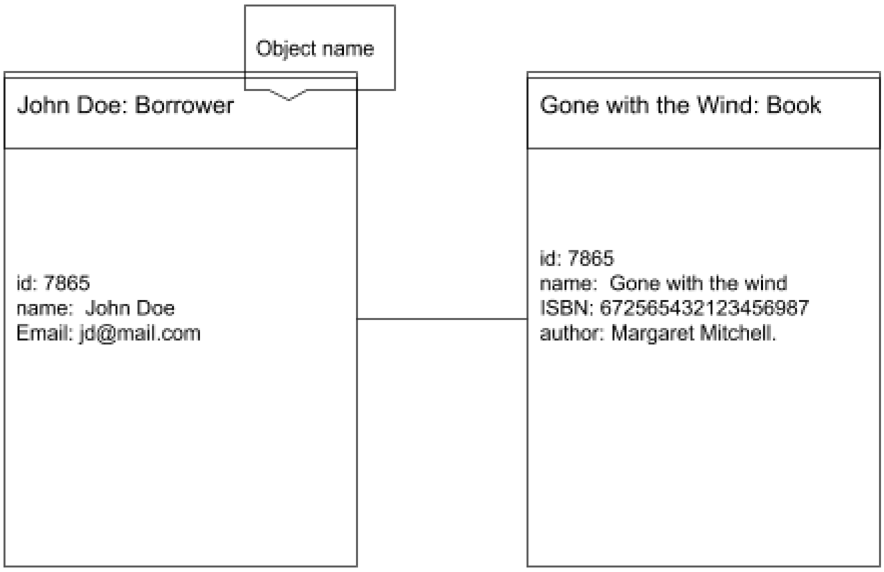
Sequence diagram of restaurant booking system



Unit	Ref	Evidence	
P	P.8	Produce two object diagrams.	



Object Diagram



Object Diagram

Unit	Ref	Evidence	
P	P.17	Produce a bug tracking report	

Bug/Error	**Solution**	**Date**
null pointer exception	added guard clause to table capacity property to prevent overbooking a table	4/11/2019
invalid DOM property “class”	changed “class” to “className”	5/11/2019
internal server error could not write JSON	added JSON body to header	6/11/2019
Objects are not valid as a REACT child	changed variable to be an array	6/11/2019
object “undefined”	added props to state	7/11/2019