

# Evidence Gathering Document for SQA Level 8 Professional Developer Award.

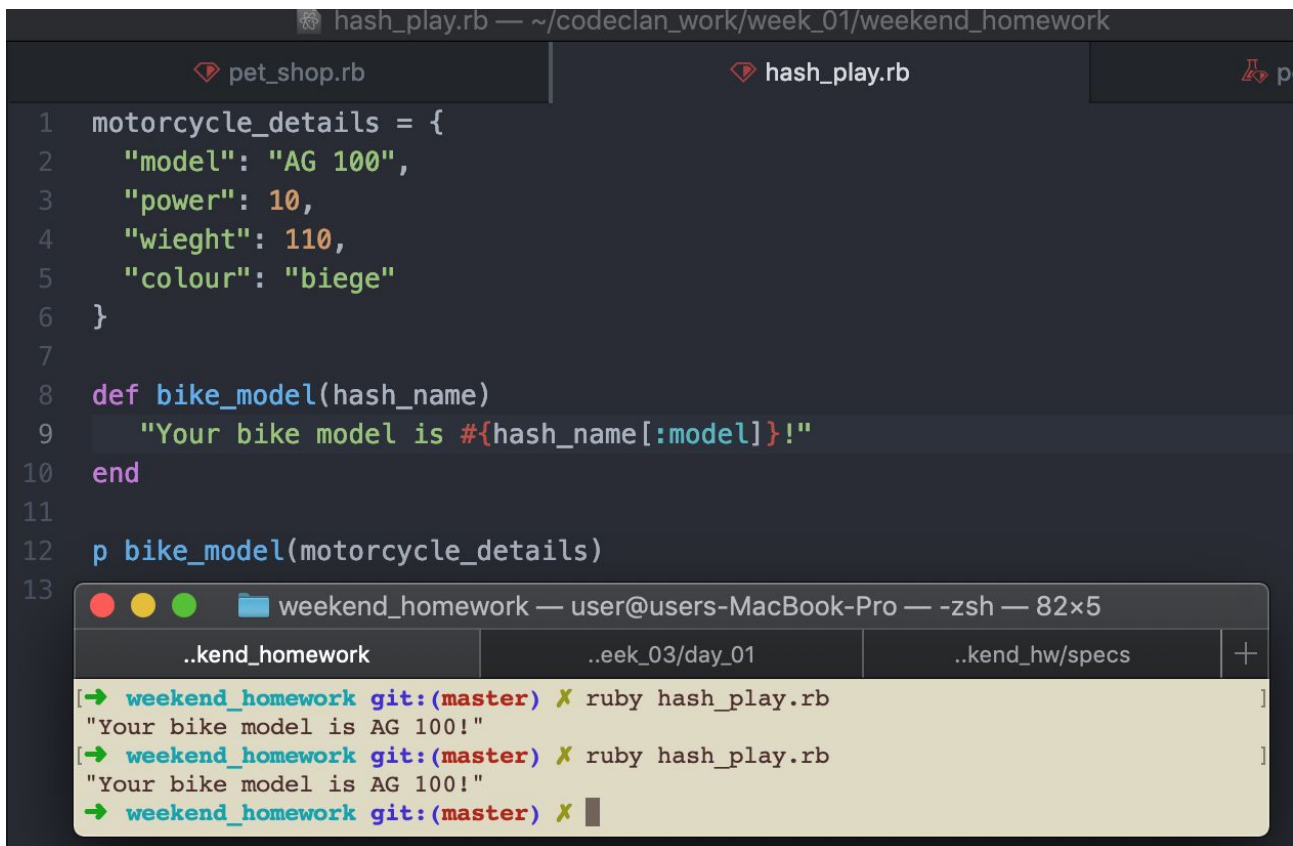
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

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

1  motorcycle_details = {
2    "model": "AG 100",
3    "power": 10,
4    "wieght": 110,
5    "colour": "biege"
6  }
7
8  def bike_model(hash_name)
9    "Your bike model is #{hash_name[:model]}!"
10 end
11
12 p bike_model(motorcycle_details)
13

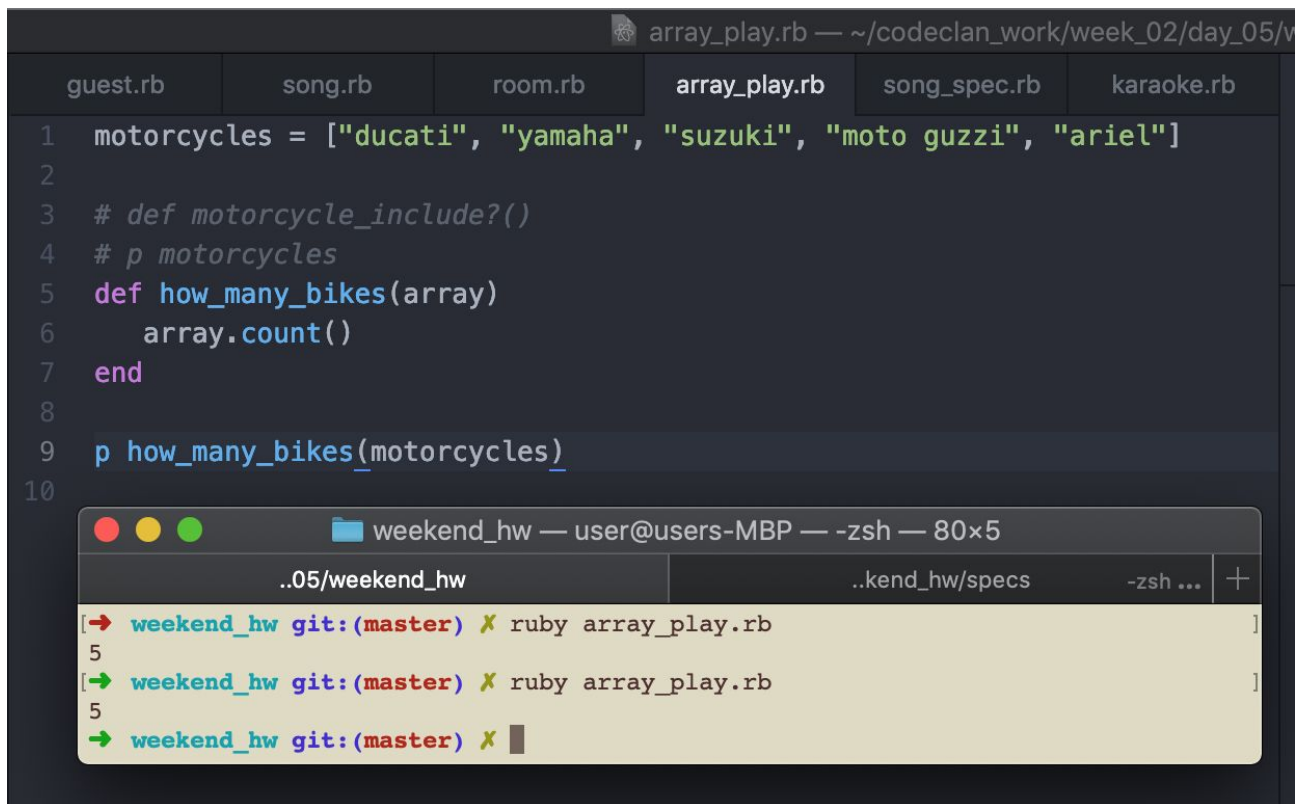
weekend_homework — user@users-MacBook-Pro — -zsh — 82x5
..kend_homework | ..eek_03/day_01 | ..kend_hw/specs | +
[→ weekend_homework git:(master) X ruby hash_play.rb |
"Your bike model is AG 100!" |
[→ weekend_homework git:(master) X ruby hash_play.rb |
"Your bike model is AG 100!" |
→ weekend_homework git:(master) X |
  
```

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



```
array_play.rb — ~/codeclan_work/week_02/day_05/v
guest.rb  song.rb  room.rb  array_play.rb  song_spec.rb  karaoke.rb
1 motorcycles = ["ducati", "yamaha", "suzuki", "moto guzzi", "ariel"]
2
3 # def motorcycle_include?()
4 # p motorcycles
5 def how_many_bikes(array)
6   array.count()
7 end
8
9 p how_many_bikes(motorcycles)
10
```

```
weekend_hw — user@users-MBP — -zsh — 80x5
..05/weekend_hw  ..kend_hw/specs  -zsh ...  +
[→ weekend_hw git:(master) X ruby array_play.rb ]
5
[→ weekend_hw git:(master) X ruby array_play.rb ]
5
→ weekend_hw git:(master) X
```

## 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.

The image shows a Ruby console window and a terminal window. The terminal window displays the code for a Ruby application, and the console window shows the output of the code.

```
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 options['customer_id']
    # @film_id = options['film_id'].to_i if options['film_id']
    @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
```

The console window shows the following output:

```
[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)>
```

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	

```

weekend_hw_cinema — ruby console.rb — ruby console.rb — 89x19
ruby ..end_hw_cinema ~/pda ../hw_cinema/db +
[[1] pry(main)> customer1
NameError: undefined local variable or method `customer1' 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 — 89x19
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()

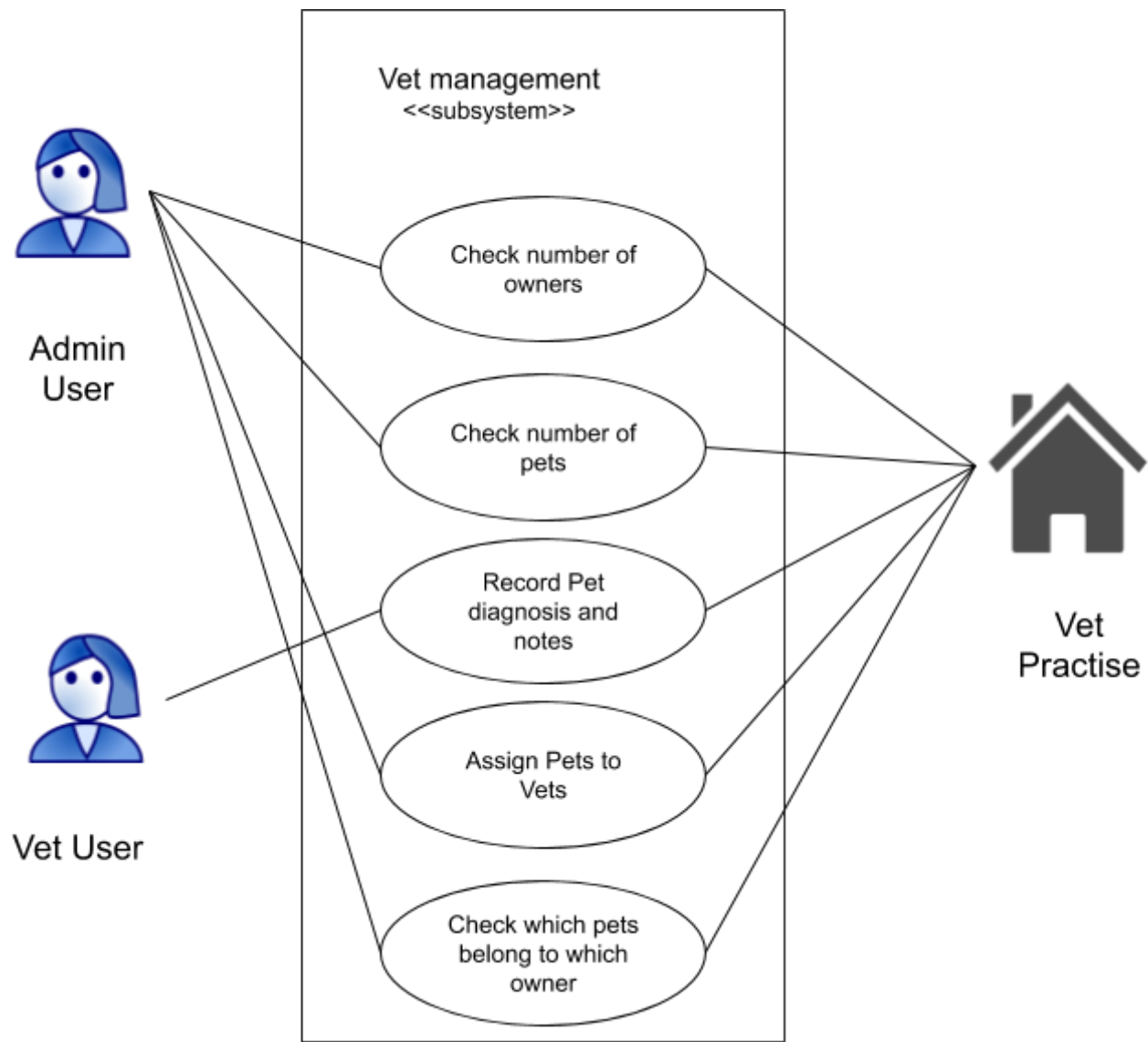
  sql = "SELECT customers.* FROM customers INNER JOIN tickets ON customers.id = tickets.customer_id WHERE film_id = $1"
  values = [@id]
  customer_data = SqlRunner.run(sql, values)
  return Customer.map_items(customer_data)

end

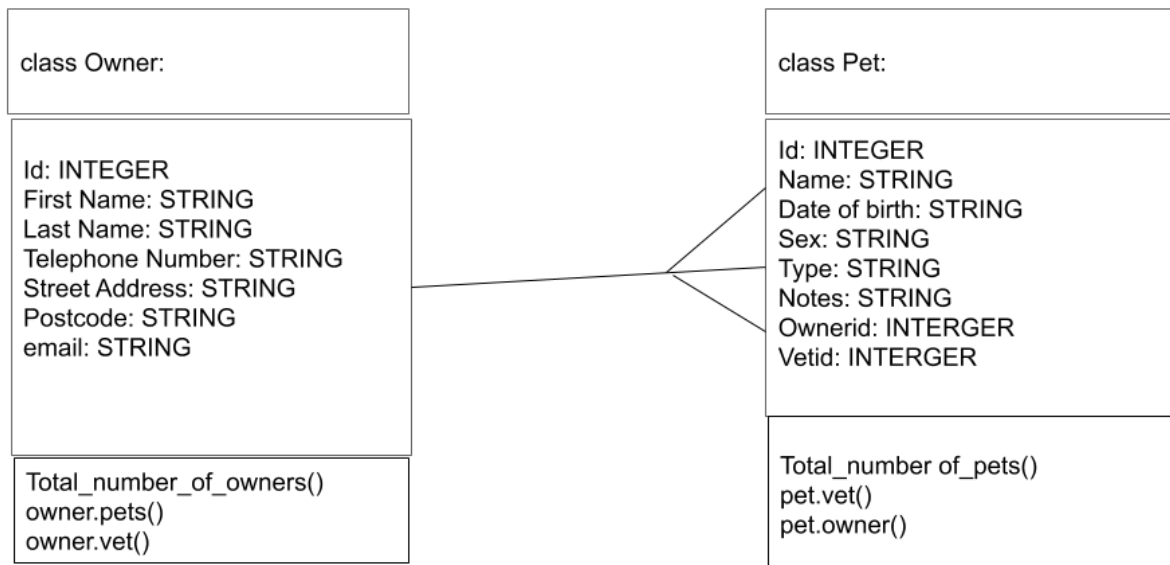
```

Week 4

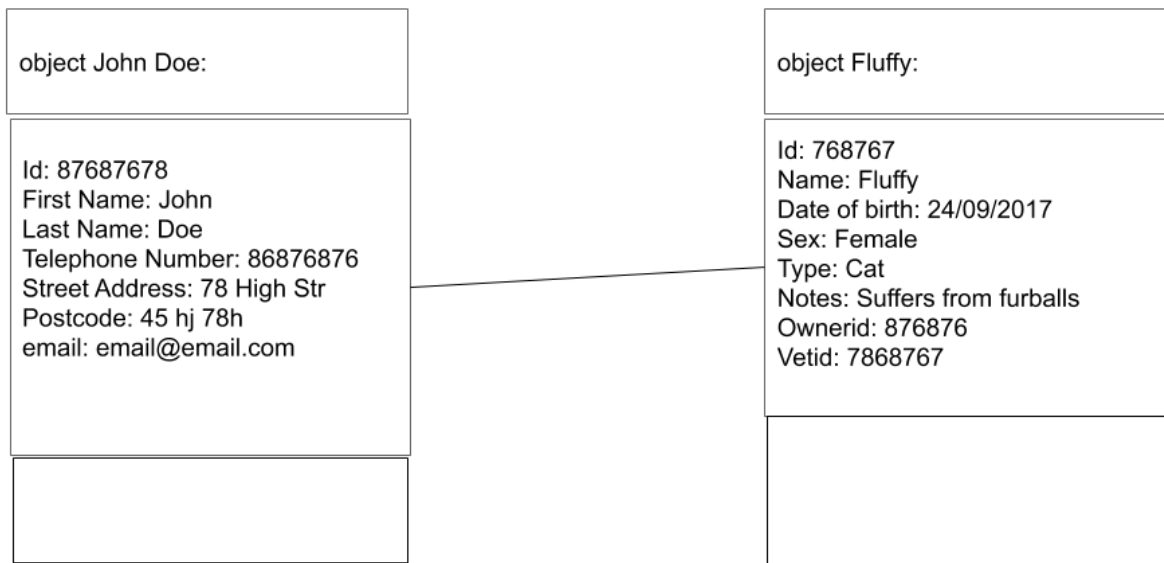
Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	



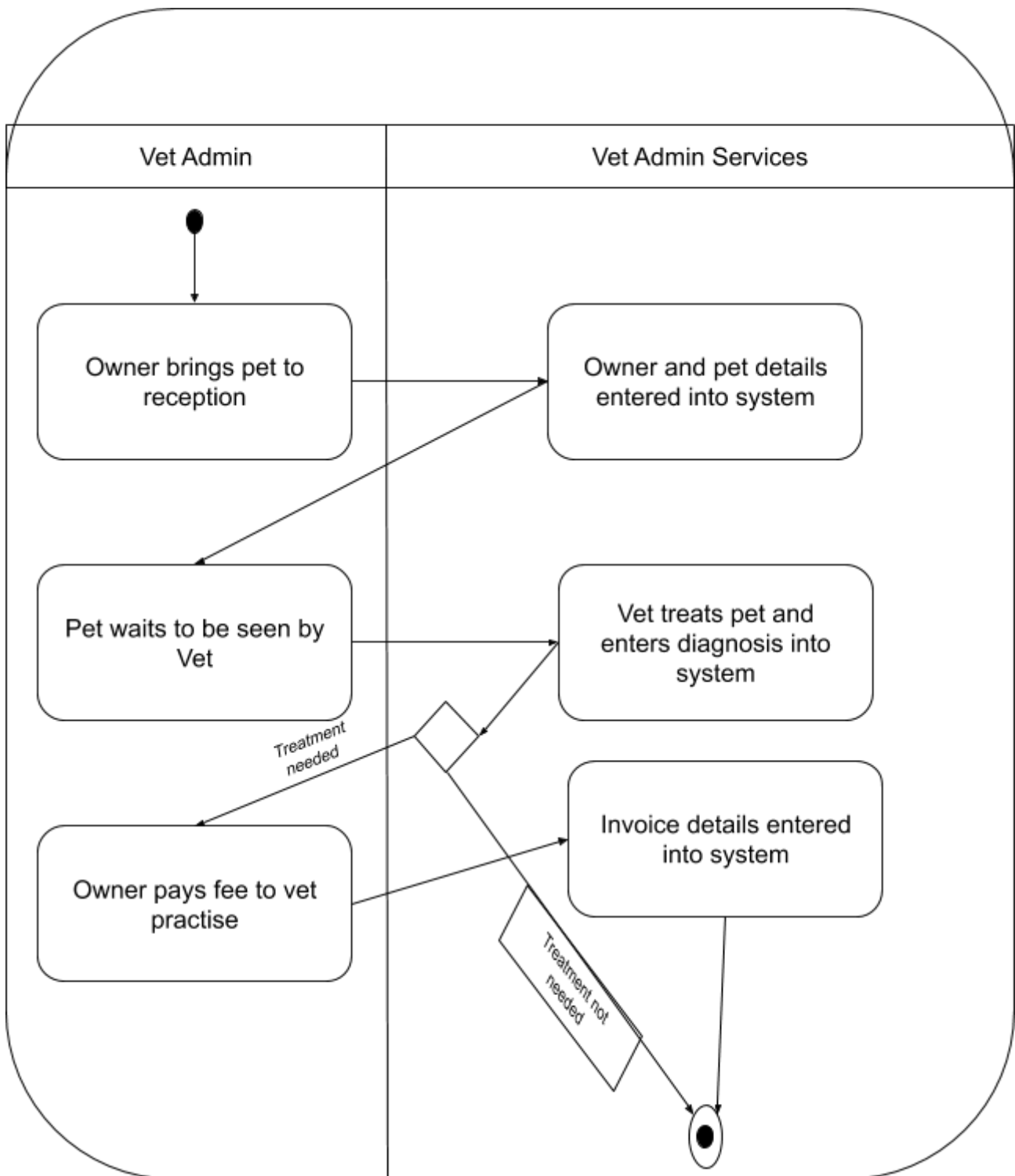
Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	



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



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

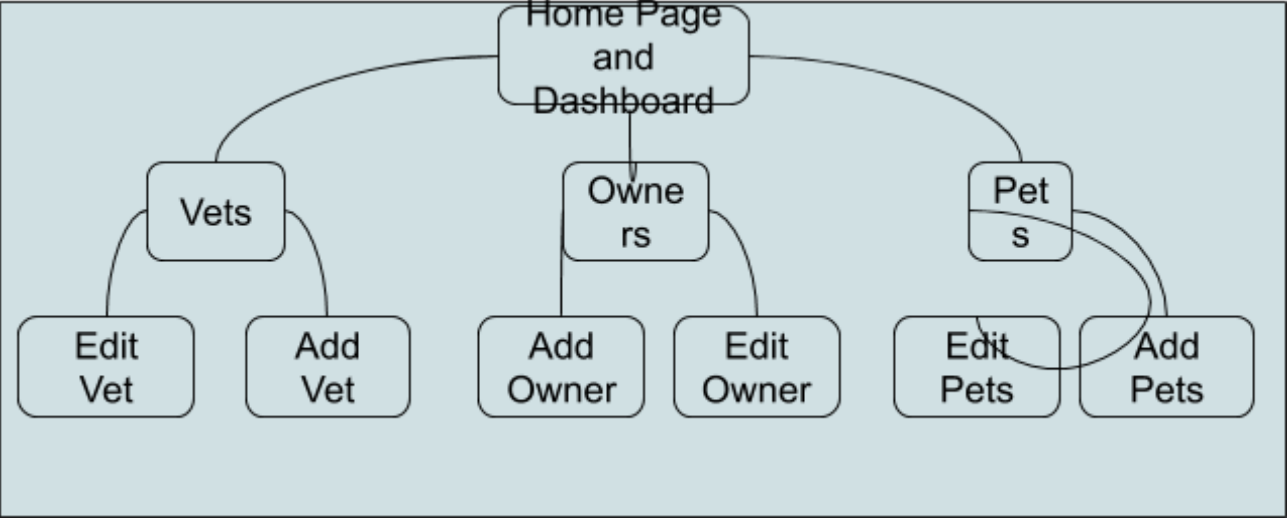




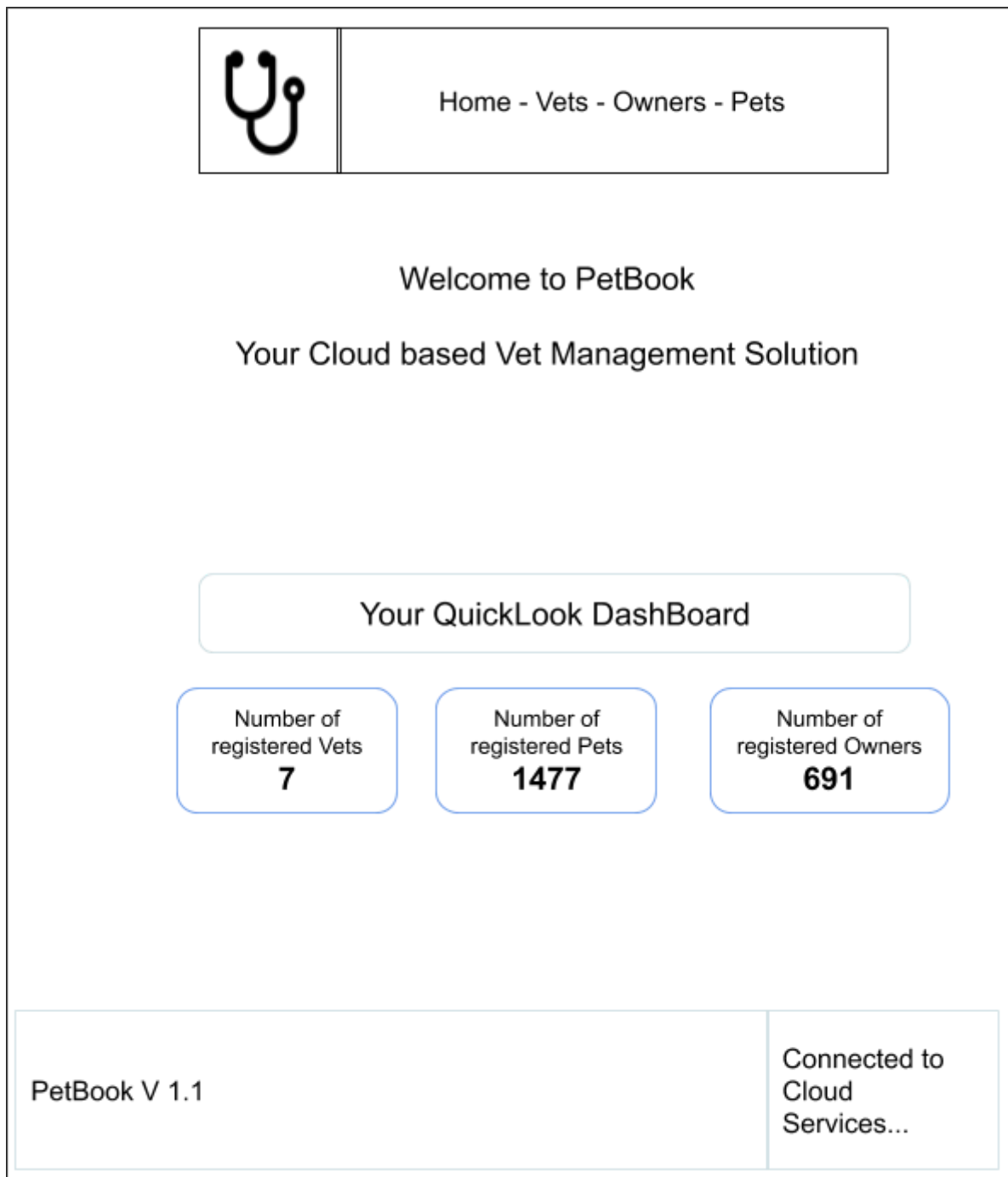
Unit	Ref	Evidence	
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	

Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	Not enough memory to run the desired software	Increase the available ram
Performance Requirements	Fast enough to server x number of concurrent users	Purchase extra system resources
Persistent Storage and Transactions	The current size and type of the storage solution	Increase the size and upgrade the type of storage solution ie Cloud Based
Usability	Difficult for user to efficiently use the system	Streamline the UX to improve workflow
Budgets	Limited funding prevents additional features and functionality being added	Secure additional funding resources
Time Limitations	The product has to be delivered to the client in a certain date	Increase the amount of personal working on the product to speed up delivery times.

Unit	Ref	Evidence	
P	P.5	User Site Map	



Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	





Home - Vets - Owners - Pets

Owners First Name:

Owners Last Name:

Owners Telephone:

Owners Address:

Owners Postcode:

Add New Owner

PetBook V 1.1

Connected to  
Cloud  
Services...

## Week 5

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

```
#Concatenate the first and last names of the vets
#Use string interpolation with the instance variables for the first and last names
#Return or output the the concatenated 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

<div> <a href="#">Home</a> <a href="#">Owners</a> <a href="#">Vets</a> <a href="#">Pets</a> </div>					
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
<div>Create Owner</div>					

A new owner being added

<div> <a href="#">Home</a> <a href="#">Owners</a> <a href="#">Vets</a> <a href="#">Pets</a> </div>
<p>Owners First Name:</p> <input type="text" value="Test Owner 1 Forname"/>
<p>Owners Last Name:</p> <input type="text" value="Test Owner 1 Last Name"/>
<p>Owners Telephone Number:</p> <input type="text" value="087680768760876"/>
<p>Owners Street Address:</p> <input type="text" value="67 Humpbridge Rd"/>
<p>Postcode:</p> <input type="text" value="56KI 89LO"/>
<p>Email:</p> <input type="text" value="bob@hmail.com"/>
<div>CREATE OWNER</div>

The updated owners list with the new owner appearing

<div> <a href="#">Home</a> <a href="#">Owners</a> <a href="#">Vets</a> <a href="#">Pets</a> </div>					
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	
<div>Create Owner</div>					

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+447484833911203 High StreetKY3 9AEBubionbreakfast@gmail.comPetey noodles

Fred Olson87575756765765Dockside Way78j 8uhjhff@jhgkyfboris

Test Owner 1 Forname Test Owner 1 Last Name08768076876087667 Humpbridge Rd56KI 89LOBob@hmail.com

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

HomeOwners

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
Create Pet						

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

[Home](#)
[Owners](#)
[Vets](#)
[Pets](#)

Owners Name: Davey Jones

Telephone Number: 55589765

Street Address: 1 Hollywood Boulevard

Postcode: 654321

Email: google@gmail.com

Edit Owner

DELETE OWNER

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	99999999999999	1 Hollywood Boulevard	654321	google@gmail.com	Fluffy Red Rum

Create Owner



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

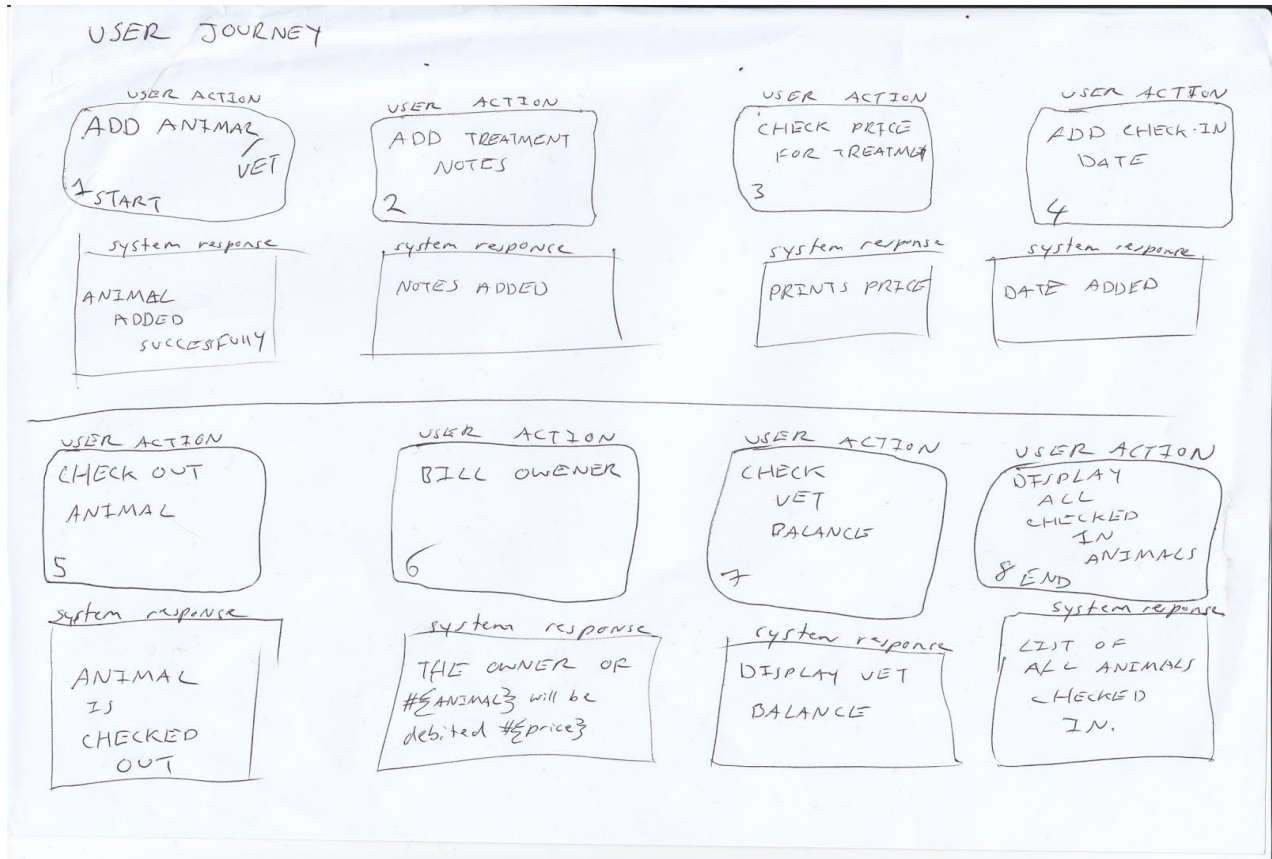
[https://github.com/bubionbreakfast/vet\\_app](https://github.com/bubionbreakfast/vet_app)

[illegible]

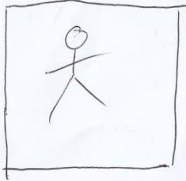
Create Pet

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

AS a	I want to	so that
visually sighted person	see A larger font & clear menus	It is easy to read & navigate.
Busy person	see the account balances automatically updated	It speeds up my work flow
animal health expert	track trends in treatments	An alert can be raised, & a crisis averted.



PROTO-PERSONA



NAME:

JANE DOE

BEHAVIOURS

QUALITY ~~USE~~ INFORMATION  
EASE of USE  
FAST  
ACCURATE INFORMATION

DEMOGRAPHICS

40 YEAR OLD

FEMALE

~~WITH 2 CATS~~

COMPUTER LITERATE

NEEDS & GOALS

GET ~~ACCOUNT~~ ACCOUNT BALANCE

GET WHICH VET IS CARING FOR WHICH PET


RAISE ALERT FOR CERTAIN TREATMENTS.

NUMBER 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	

```
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);
```



**Date**

**Select Meal**

**Kcal**



## Week 8

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

### ↳ **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.

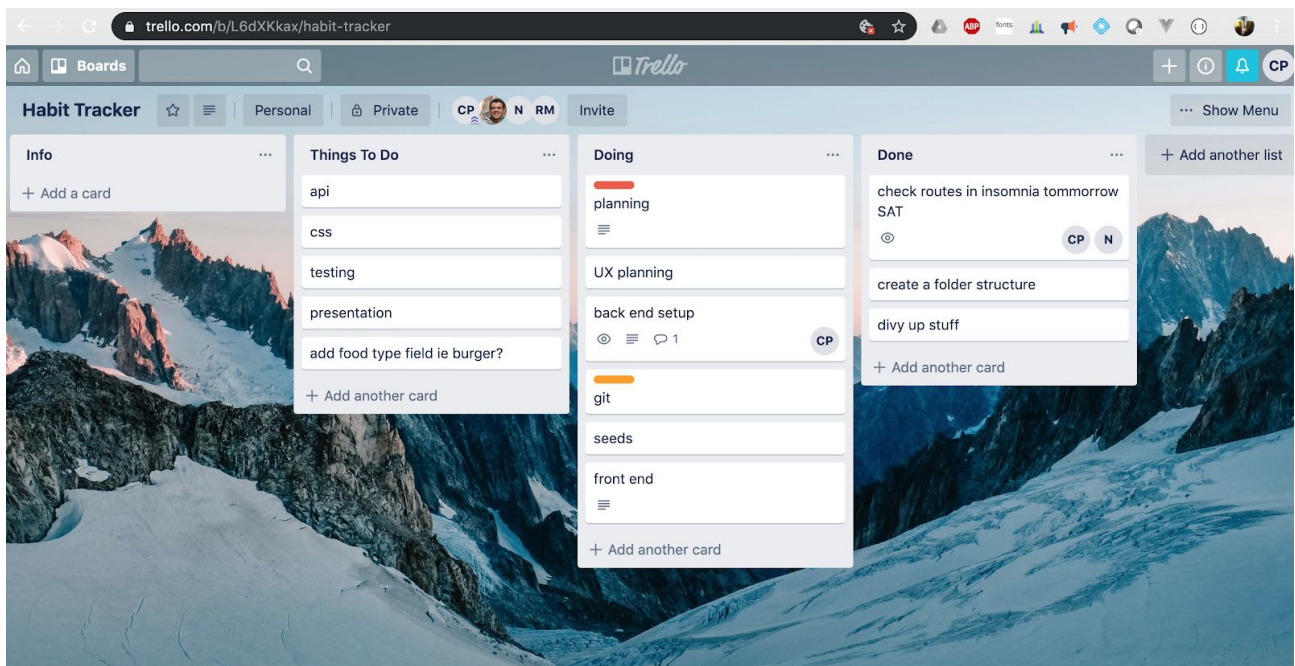
#### **Example Extension**

- 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.	



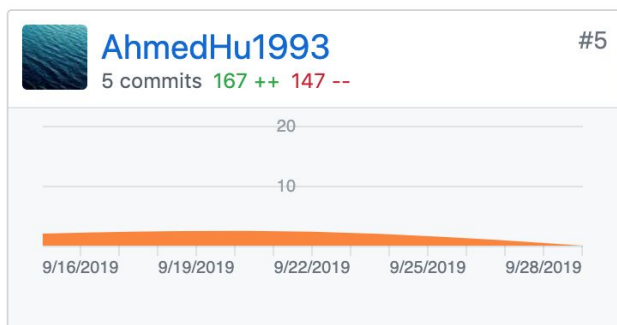
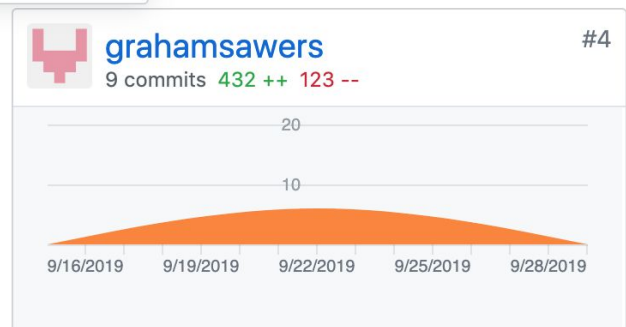
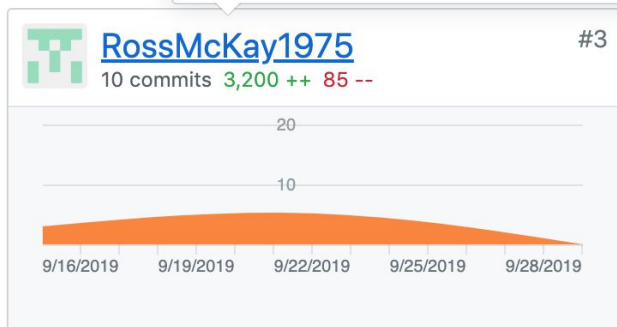
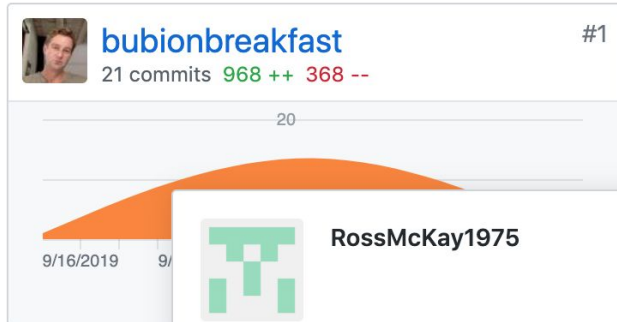
Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

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



## 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.	



## Week 11

Unit	Ref	Evidence	
P	P.18	Demonstrate testing in your program. Take screenshots of: <ul style="list-style-type: none"><li>* Example of test code</li><li>* The test code failing to pass</li><li>* Example of the test code once errors have been corrected</li><li>* The test code passing</li></ul>	

```
13
14   it('add 1 to 4 and get 5', function(){
15       const actual = calculator.add(4)
16       assert.equal(actual, 5)
17   })
```

```
➔ 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.
```

```
➔ js_calculator_start_point git:(master) ✖ npm test

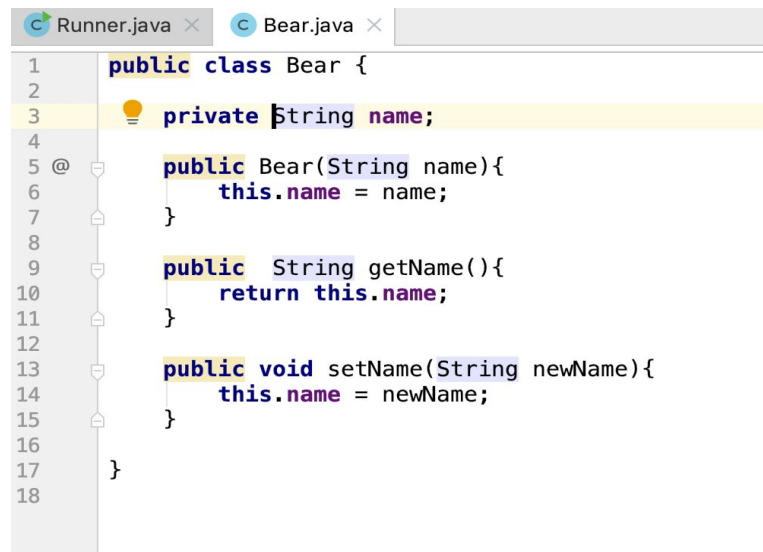
> js_calculator_start_point@1.0.0 test /Users/user/e33_classnotes.
art_point
> mocha tests/unit/calculator_spec.js

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.	



```

1 public class Bear {
2
3     private String name;
4
5     public Bear(String name){
6         this.name = name;
7     }
8
9     public String getName(){
10        return this.name;
11    }
12
13    public void setName(String newName){
14        this.name = newName;
15    }
16
17 }
18

```

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.	

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

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.	

**Paste Screenshot here**

**Description here**

## 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.	

**Paste Screenshot here**

**Description here**

Unit	Ref	Evidence	
------	-----	----------	--

<b>P</b>	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).
----------	-----	---

**Paste Screenshot here**

**Description here**

<b>Unit</b>	<b>Ref</b>	<b>Evidence</b>	
<b>P</b>	P.8	Produce two object diagrams.	

**Paste Screenshot here**

**Description here**

<b>Unit</b>	<b>Ref</b>	<b>Evidence</b>	
<b>P</b>	P.17	Produce a bug tracking report	

**Paste Screenshot here**

**Description here**