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

```ruby
cats_and_attributes = {

  "steve" => {
      "age" => 3,
      "colour" => "Tuxedo"
    },

  "richard" => {
      "age" => 1,
      "colour" => "Ginger"
    },
  "clive" => {
      "age" => 5,
      "colour" => "Black"
    }
}

def get_cat_age(cat)
  p "the cat is #{cat["age"]}"
end

get_cat_age(cats_and_attributes["steve"])
```

```
➜  arrays_and_hashes ruby hashes.rb
"the cat is 3"
➜  arrays_and_hashes
```

Above I have created a hash of cats and their attributes, I have then created a function that takes in each cat and gets its age from the hash that is inputted then outputting this in a sentence via interpolation in a string. The function is then called with the argument of the cat named Steve who is the key of the first item in the hash which is named cats_and_attributes. When run you get the above result.

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

```ruby

cats_list = ['steve', 'bob', 'claire', 'rupert', 'beatrice']

def get_fourth_cat_name(cats)
p "the cat's name is: " + cats[3]
end

get_fourth_cat_name(cats_list)

```

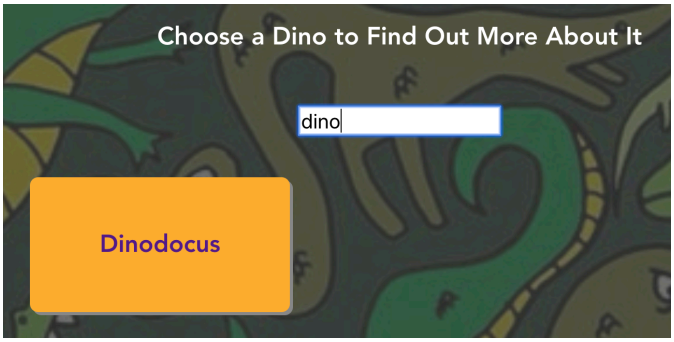
```
➜  arrays_and_hashes ruby arrays.rb
"the cat's name is: rupert"
```

Here I have created an array called cats_list which contains a list of 5 cat names. I have then created a function called get_fourth_cat_name with the argument of cats, this function then prints out the string “the cat’s name is:” and concatenates the 4th item of the list inserted as a parameter using index 3 as 0 is counted. After creating this function I then call the function with the array of cats created earlier as an argument which then prints out the string “the cat’s name is: “ and the 4th cats name as a string merged to the console.

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

```
filteredDinosaurs: function () {
  if (!this.searchTerm.length) return this.dinosaurs;
  return this.dinosaurs
  .filter(dinosaur => dinosaur.toLowerCase()
  .includes(this.searchTerm.toLowerCase()));
}
```



The above function takes in a list of dinosaur objects ‘this.dinosaurs’ from the state, and then runs through each instance of a dinosaur object and using the ‘includes’, searches for all dinosaur names matching the search term saved in the state returning only dinosaurs from ‘this.dinosaurs’ containing inputted search term. An example of this search in action showing a result based on a search on the left.

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

```
sortByPriceAsc = (adverts) => {
  return adverts.slice().sort((a, b) => {
    return a.price - b.price
  })
}
```

```
this.setState({
  adverts: this.sortByPriceAsc(data._embedded.adverts),
```

```
<AdvertList
  adverts={this.state.adverts}
  handleClick={this.handleClick}
  embedded={this.state.embedded}
/>
```

Item Title: Magic Table

Item Price: £0

Item Location: Edinburgh

Detail

Item Title: To Infinity And Beyond

Item Price: £50

Item Location: Edinburgh

Detail

Item Title: iPhone 6 - great condition

Item Price: £90

Item Location: Edinburgh

Detail

Item Title: Modern Fridge

Item Price: £100

Item Location: Edinburgh

Detail

Item Title: Hammer

Item Price: £1000

Item Location: Edinburgh

Detail

Item Title: Fast car

Item Price: £50000

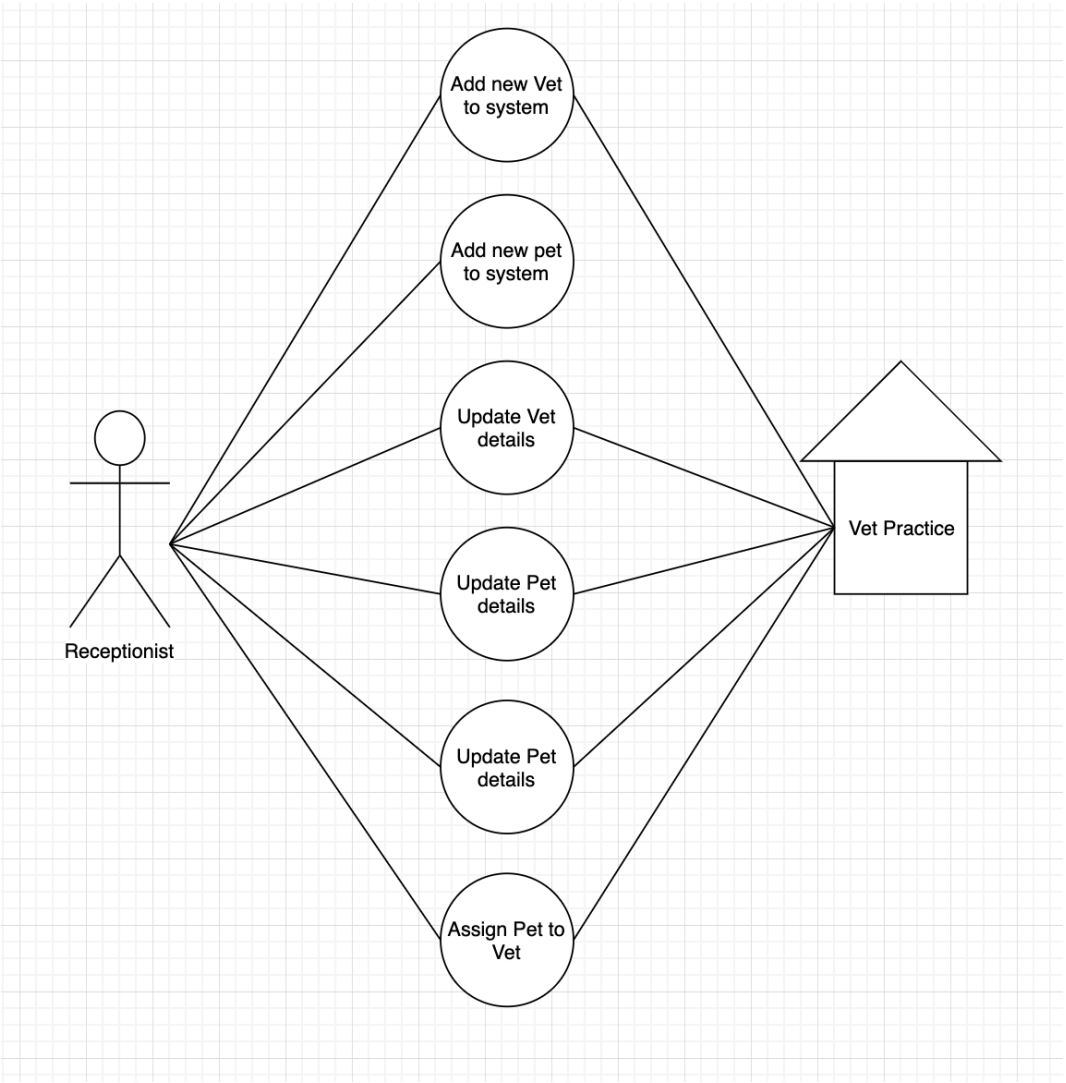
Item Location: Edinburgh

Detail

Sorting data unlike searching data does not reduce the amount of information show, it just changes the order of all given data. This example here uses a function to sort the products taken from a database from low to high. To do this, the function takes in a parameter that is an array of objects. The passed in array of objects is then looped using the sort function which is given a rule of a.price - b.price(.price is an attribute of the object), checking 2 values at a time checking if a.price is greater than, equal to or less than b.price. The function is called with an argument of advert objects and saved to the state, which is then passed to a list component where it can be displayed.

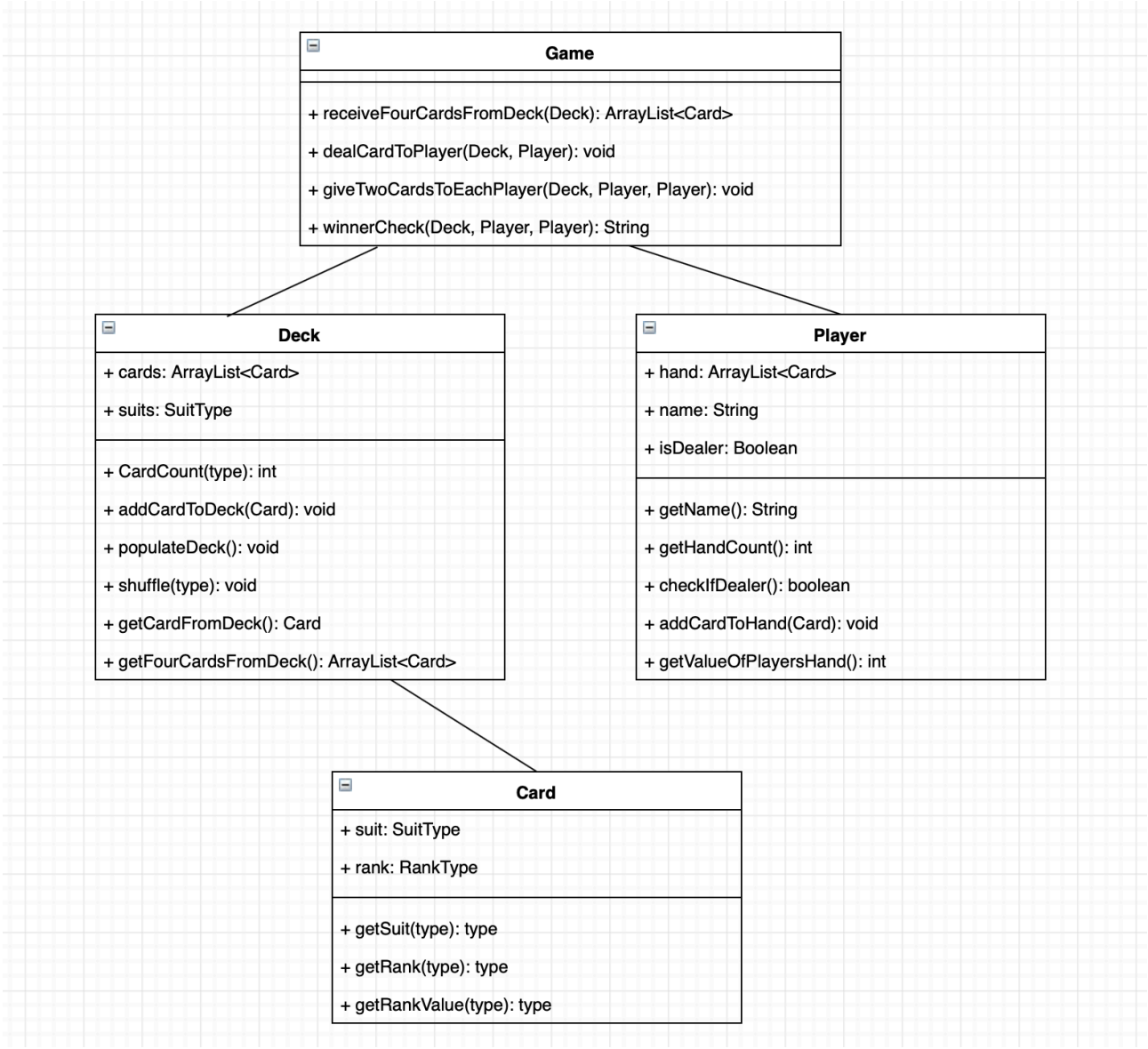
Week 4

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



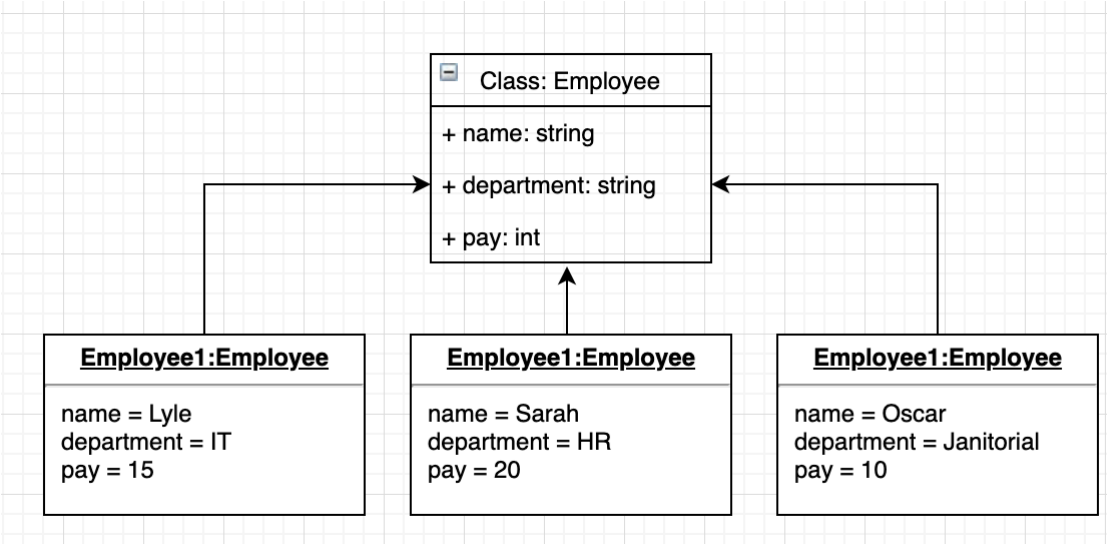
This use case diagram shows the actions that the target user of my vet practice app, the receptionist would like to be able to perform for the vet practice. This helps to lay out the requirements between the user and organisation.

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



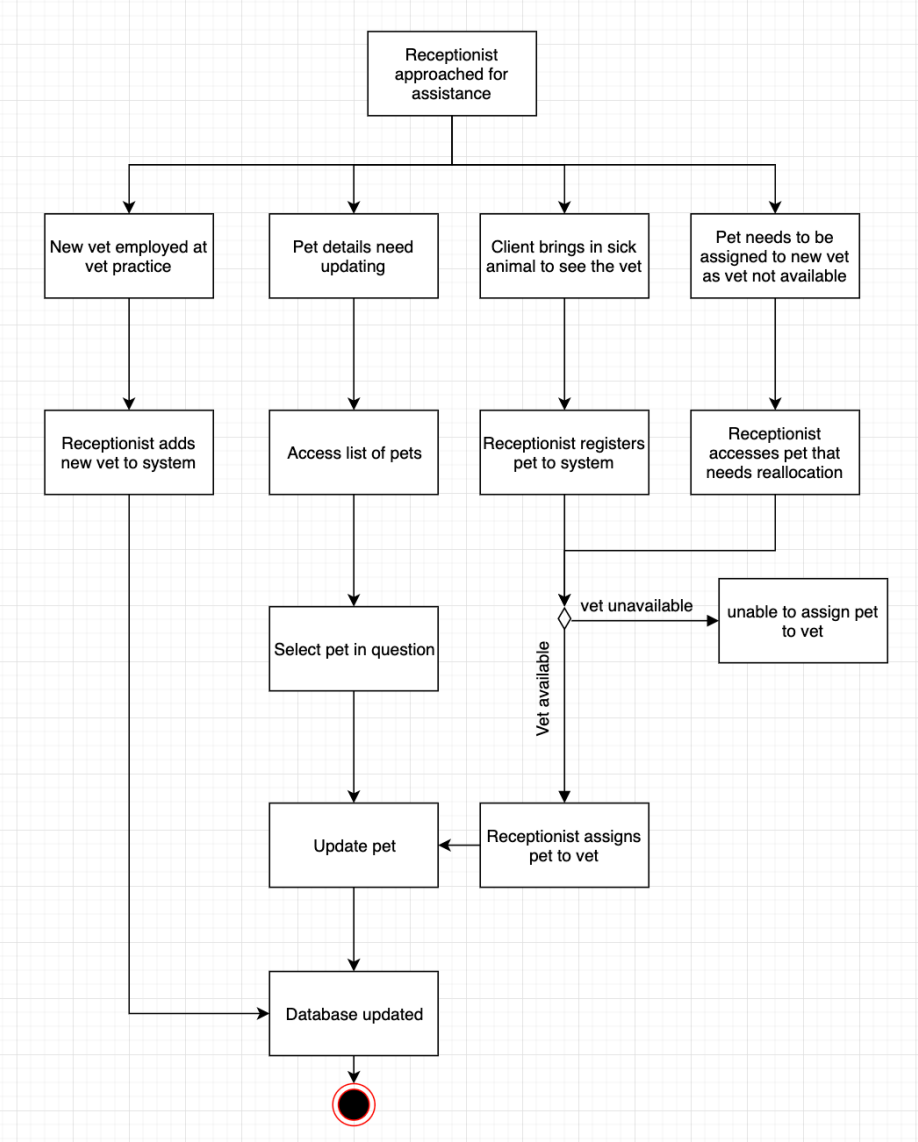
This is a class diagram for a Java BlackJack game which shows the 4 classes, their instance variables and instance methods, for both the data type appended including the parameter types of each method. The diagram also shows the relationships of the classes. This helps visualise the classes which allows you to identify repeated functionality across classes and if the relationship of each is logical.

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



This object diagram for an employees application displays class instances and their relationship to the class as well as the data that will be entered, matching the class instance variables & types. Each object and its structure is the same just differing by the data contained.

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



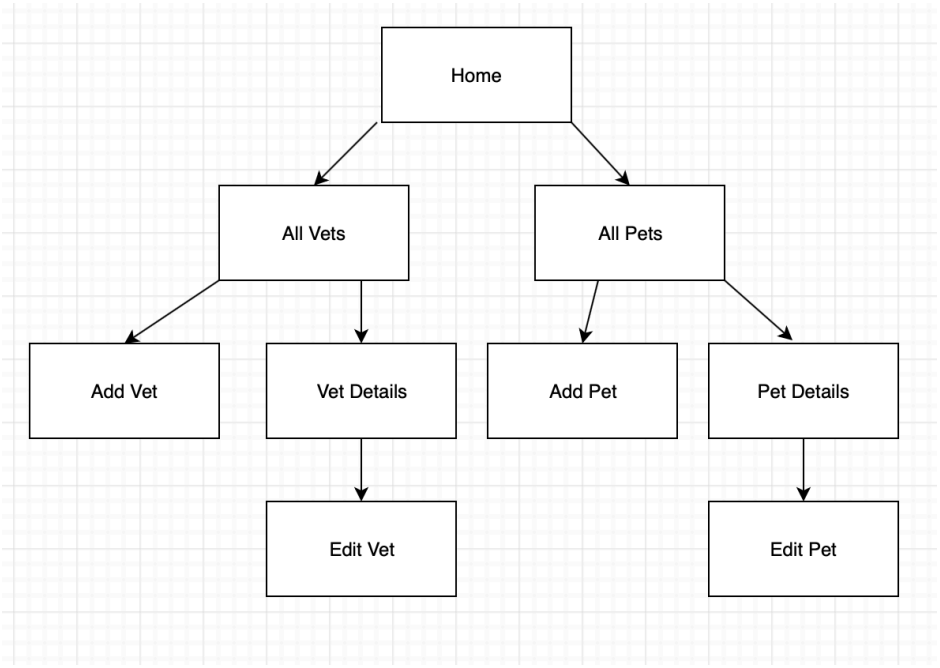
An activity diagram shows the process a user would follow to perform an action that is a required function of the program. This helps to visualise how many steps each thing takes and how the use cases interact with each other and similarities. The diagram also shows decisions in the applications process with outcomes.

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
Time limitation	Not enough time to create separate class for pet owner	Keep pet owner details within the pet class
Usability	App user might not be that tech savvy	Make interface simple
Hardware & Software Platforms	App may be used in less common web browser	Test app on different web browsers for compatibility.
Budget	Limited budget	Plan out what is achievable on budget and make working app with features based on this
Performance		

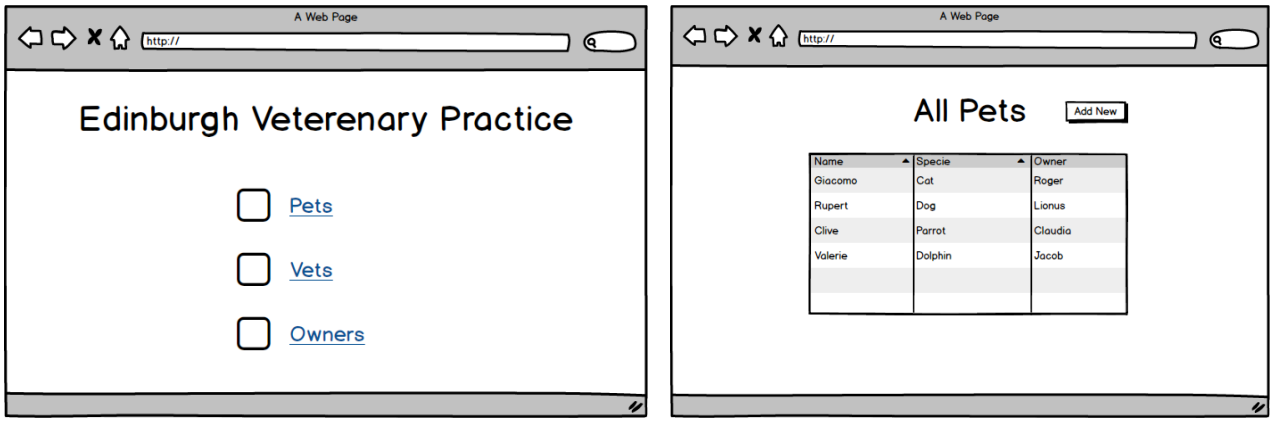
This is an implementation constraint plan for my vet practice ruby app, this plan helps you to predict potential issues with the software and state how you plan on tackling each issue/ requirement. Each constraint is categorised for organisation assignment of issues to be resolved.

Unit	Ref	Evidence
P	P.5	User Site Map



This is my sitemap for my Ruby app for a vet practice which shows the user flow whilst using the site and how each page is linked and the journey from the site index. This exercise is good for planning as well as analysis to ensure the site structure is not too complex and is logical.

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams



These wireframes were created using Balsamiq Mockups for my Ruby vet practice software project. One of my main focuses for this project was for the app to be simple and easy to use, so I planned for a minimalist design from the start. The homepage is very minimal with just 3 links to the 3 main subpages. On the next level down a list of data is displayed, I decided to go with tables as it helps data readability and categorisation.

Week 5

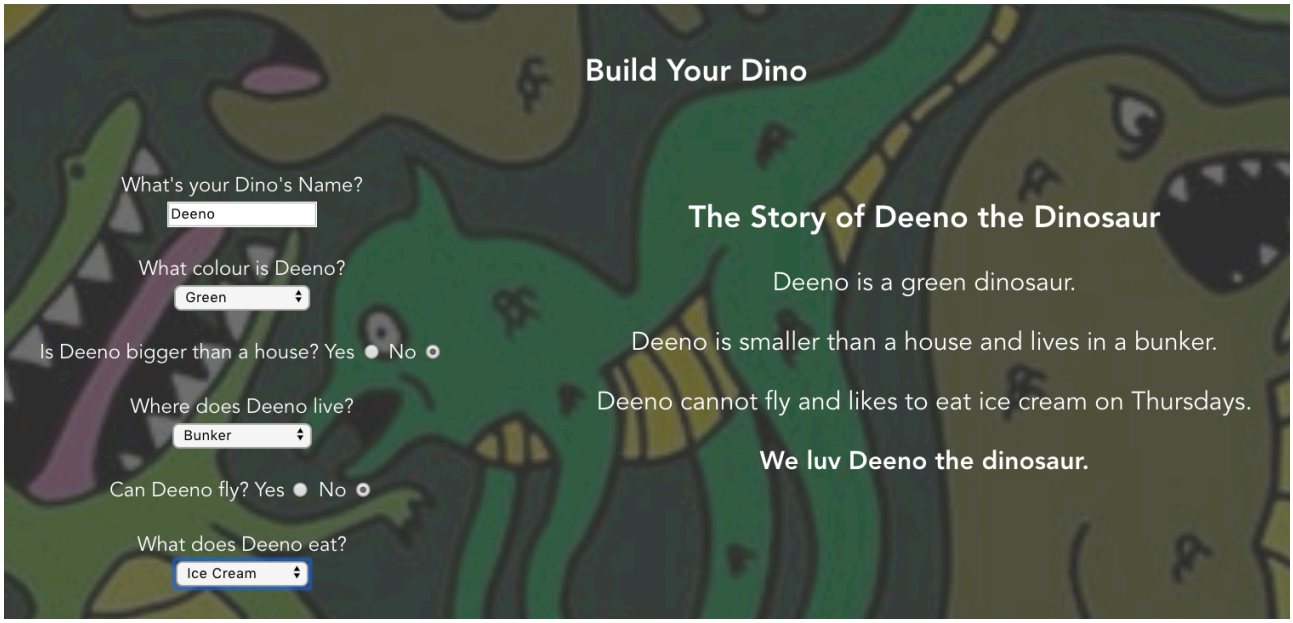
Unit	Ref	Evidence
P	P.10	Example of Pseudocode used for a method

```
// function to deal 2 cards to each of 2 participating players

public void giveTwoCardsToEachPlayer(Deck deck, Player player1, Player player2) {
    ArrayList<Card> cardsToDeal = receiveFourCardsFromDeck(deck);
    player1.addCardToHand(cardsToDeal.get(0));
    player1.addCardToHand(cardsToDeal.get(1));
    player2.addCardToHand(cardsToDeal.get(2));
    player2.addCardToHand(cardsToDeal.get(3));
}
```

For this function I wanted to be able to deal 2 cards from a deck of cards class to 2 players playing a game of blackjack, I used pseudocode to outline what I wanted the function to do and visualise it, as well as label the function for later reference and to help others to understand the code easier.

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



For this section I chose to use the ‘create your own dinosaur’ activity from an information portal group project. The user can make up a dinosaur and give it custom attributes, which are then inputted into a paragraph which generates a story about the made up dinosaur.

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

As an example of interaction with data persistence I have used a form contained in my Ruby Vet Practice Software project which allows the user to enter the name and details of a new animal in the vet practice's care. Once the data is entered and the form it submitted the user is shown a confirmation message. The new data is now displayed in a table of all the patients at the vet practice as shown in the screenshot above.

Paste Screenshot here

Description here

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

Details of Pokemon

Pokemon List

Bulbasaur

Ivysaur

Venusaur

Charmander

Name: **BULBASAUR**

Move: **Razor-Wind**

Weight: **69kg**

Type: **Poison grass**

1

```
<template lang="html">
  <li v-on:click="handleClick"># {{pokemon.name}}</li>
</template>
```

```
<div class="detail-holder">

  <div class="attributes">
    <p>Name: <b> {{pokemon.name.toUpperCase()}}</b> </p>
    <p>Move: <b>{{pokemon.moves[0].move.name}}</b> </p>
    <p>Weight: <b>{{pokemon.weight}}kg</b></p>
    <p class="type-label">Type: </p>
    <p class="type" v-for="type in pokemon.types"><b>{{type.type.name}}</b> </p>
    <p># <b>{{pokemon.id}}</b></p>
  </div>
  <div class="image-container">
    <img :src="pokemon.sprites.front_default" :alt="pokemon.name + ' image'">
  </div>
</div>
```

For this section I chose my Vue Pokedex as an example. The app as default shows the user a list of Pokemon names and when the user clicks on one of Pokemon names, a new box pops up at the top of the app displaying attributes of the clicked Pokemon, as well as an image of the Pokemon.

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

Home

View All Pets

View All Vets

All Pets

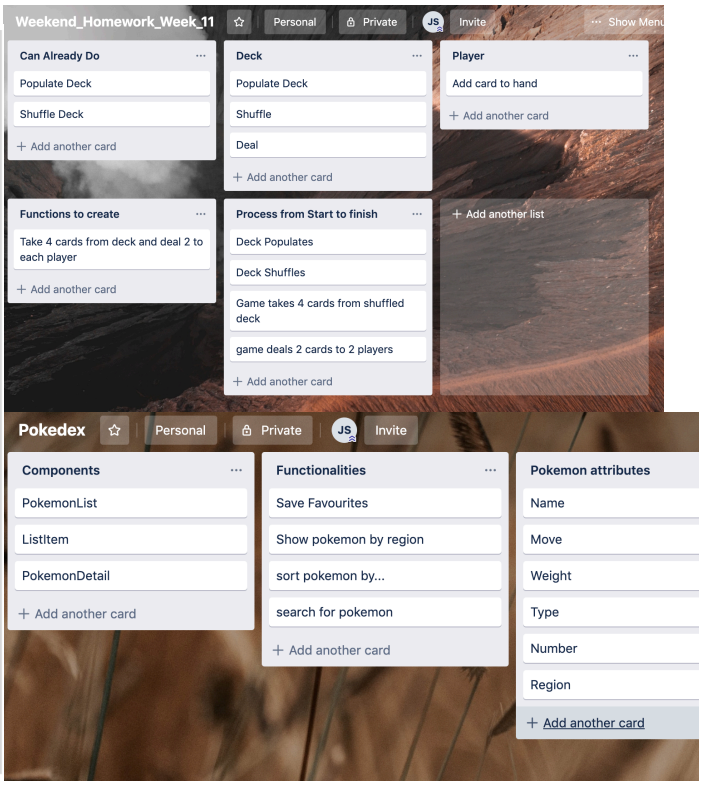
Name	Specie	Breed	Age	Vet
Xerxes	Owl	Tawney	29	Dr Cleveland Brown
Bob	cat	shorthair	4	Dr Cleveland Brown
Wallace	Seahorse	Sea Shire	26	Dr Cleveland Brown
Linus	Aligatore	Sharptooth	2014	Dr Cleveland Brown
Raymond	Snake	Python	119	Dr John Smith

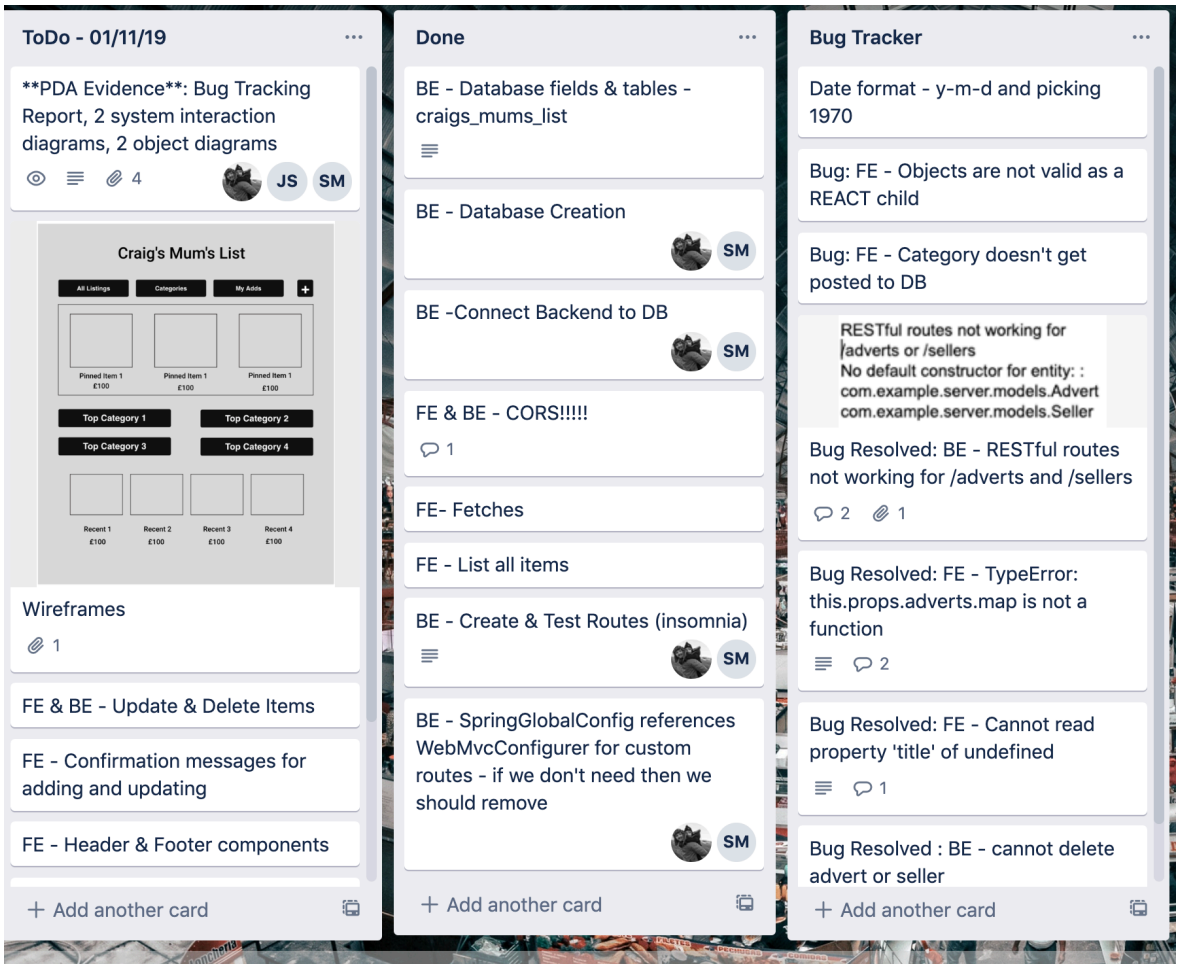
Create New Pet

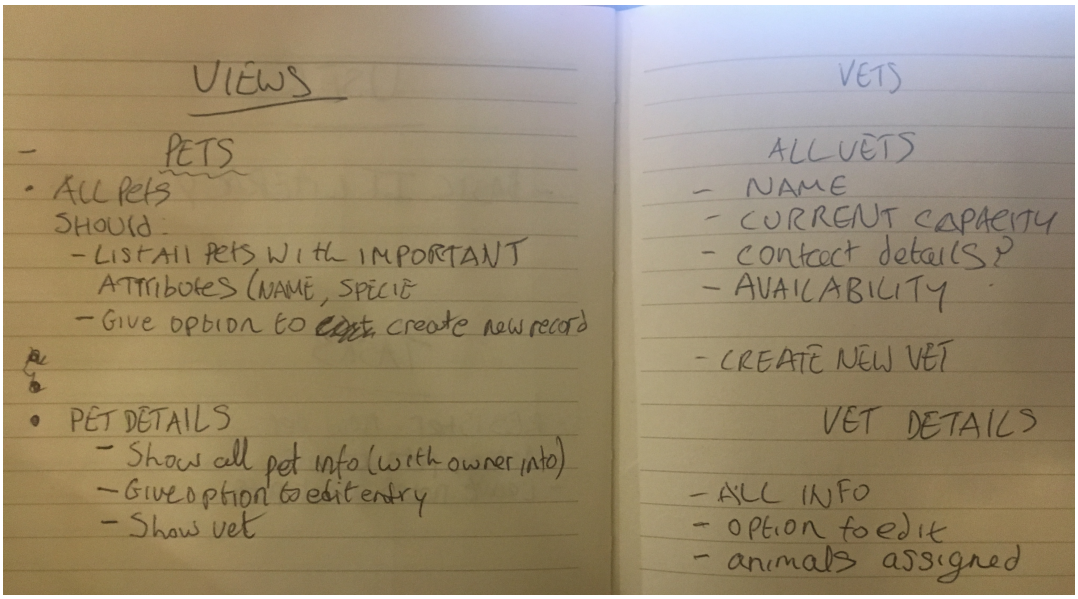
https://github.com/jordan-w-smith/wk4-vet-practice-ruby-sinatra-project

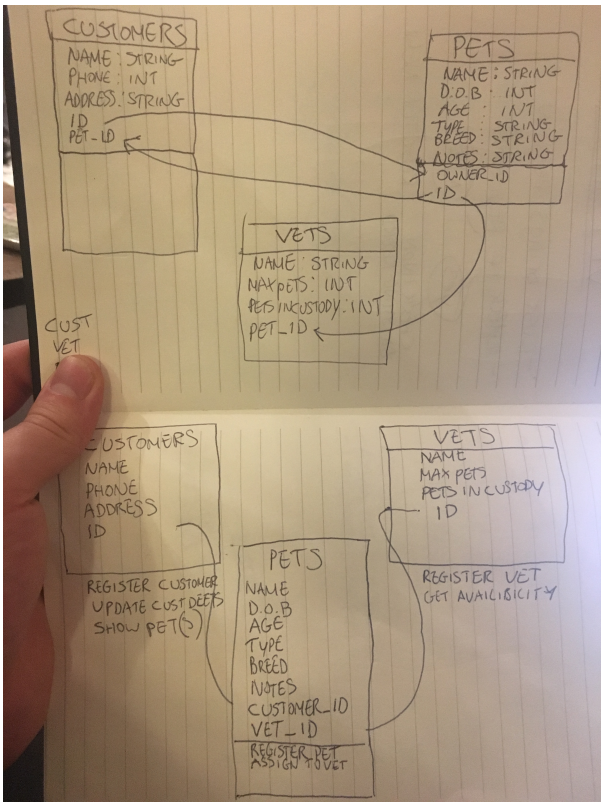
For this project I was tasked with creating an application for a receptionist to use at a vet practice to manage vets and pets. This project was built using Ruby, Sinatra & PSQL. The app has CRUD functionalities for both pets and vets.

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









For planning my primary tool of choice is the kanban board Trello which I use frequently throughout the process of building an app. I also use pen & paper, whiteboards, wireframe mockups and UML diagrams. Planning is something I revisit frequently throughout the process rather than doing all at the beginning as requirements can change as things are developed, a need for new features can become apparent, reassessment of priorities and many other reasons can make flexible planning throughout a good process. I like to create lists on kanban boards to categorise to do lists into things that need to be done, are in progress and have been done to keep track.

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

```
mounted() {
  fetch('https://pokeapi.co/api/v2/pokemon?limit=150')
  .then(res => res.json())
  .then(pokemons => this.pokemons = pokemons)

  eventBus.$on('pokemon-selected', pokemon => this.getPokemonDetails(pokemon.name) )
},
methods: {
  getPokemonDetails: function(pokemonName) {
    fetch(`https://pokeapi.co/api/v2/pokemon/${pokemonName}`)
      .then(res => res.json())
      .then(pokemon => this.selectedPokemon = pokemon)
  }
}
```

```
<div class="attributes">
  <p>Name: <b> {{pokemon.name.toUpperCase()}}</b> </p>
  <p>Move: <b>{{pokemon.moves[0].move.name}}</b> </p>
  <p>Weight: <b>{{pokemon.weight}}kg</b></p>
  <p class="type-label">Type: </p>
  <p class="type" v-for="type in pokemon.types"><b>{{type.type.name}}</b> </p>
  <p># <b>{{pokemon.id}}</b></p>
</div>
<div class="image-container">
  <img :src="pokemon.sprites.front_default" :alt="pokemon.name + ' image'">
</div>
```

Details of Pokemon

Name: **ZAPDOS**

Move: **razor-wind**

Weight: **526kg**

Type: **flying**

145

Pokemon List

Bulbasaur

Ivysaur

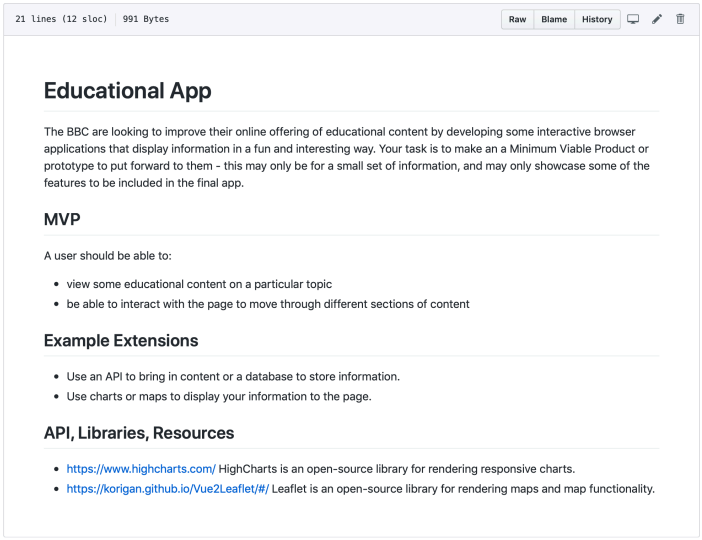
Venusaur

Charmander

For this example I have included a Vue JS application I created a basic Pokedex for Pokemon using the PokeApi. The functions screenshotted above pulls the data from 2 endpoints using ‘fetch’ pulling in all data from the api endpoint and saving it to a variable using “then”. The first API call pulls in a list of Pokemon names and the second, information about a specific Pokemon. The data is for the list of names displayed visibly by looping through through the data using a v-for(vue) and inserted in <li> elements which are inside of a <ul>. When a name is clicked this action calls the second fetch getting the details about the clicked Pokemon which are then displayed above the list.

Week 8

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



21 lines (12 sloc) | 991 Bytes — Raw | Blame | History

Educational App

The BBC are looking to improve their online offering of educational content by developing some interactive browser applications that display information in a fun and interesting way. Your task is to make an a Minimum Viable Product or prototype to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app.

MVP

A user should be able to:

- view some educational content on a particular topic
- be able to interact with the page to move through different sections of content

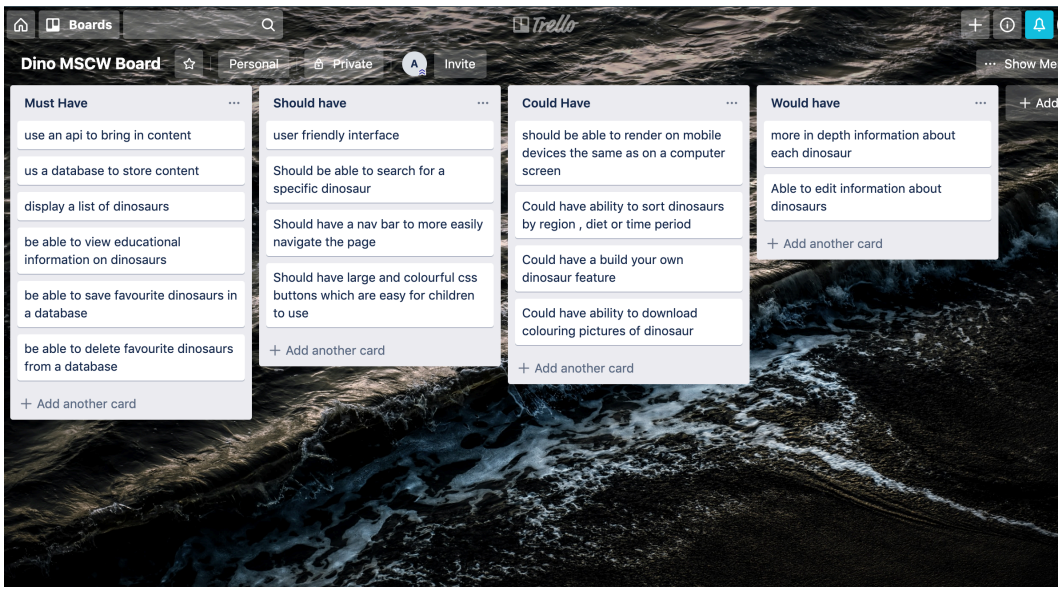
Example Extensions

- Use an API to bring in content or a database to store information.
- Use charts or maps to display your information to the page.

API, Libraries, Resources

- https://www.highcharts.com/ HighCharts is an open-source library for rendering responsive charts.
- https://korigan.github.io/Vue2Leaflet/#/ Leaflet is an open-source library for rendering maps and map functionality.

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



Above is a MOSCOW board created for a group project making a dinosaur learning portal. This ‘must, should, could, would’ model helps to outline features and functionalities from vital to un necessary as well as prioritising them based on the category each item is put into.

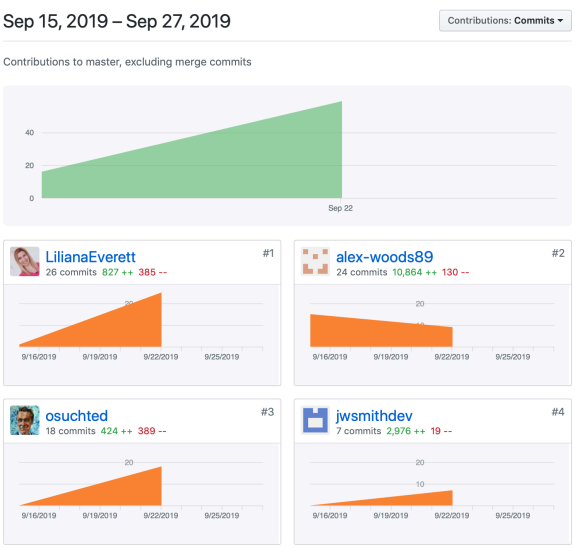
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

Acceptance Criteria	Expected Result	Pass/Fail
A user can see names of all dinosaurs	On page load a list of dinosaurs is pulled from the API and displayed to user	Pass
A user can find out information about an individual dinosaur	User clicks dinosaur name from the list of dinosaurs and details about this dinosaur show	Pass
A user can save a dinosaur they like to a list of favourites	When user is looking at details of individual dinosaur they can click a button which will add the dinosaur to a favourites list which displays on the top of the page	Pass
A user is able to create their own dinosaur	User fills out form with details about a made up dinosaur, these details are then taken and inputted into a paragraph below the form, making a summary of newly created dinosaur	Pass
A user can find a dinosaur they know the name of and see information about it	User inputs the name of dinosaur they want to find out about into the search bar and are shown results containing the name of dinosaur.	Pass
A user is able to navigate and read the website easily	Text is a good size and readable, site is colourful with whitespace. Navigation can take user to each section of application.	Pass

An acceptance criteria and test plan allows you to keep track of processes and features deemed necessary for a user. The first column is the users need for function or feature, the second is the expected result of a function created to fulfil the criteria laid out in the first column and finally the 3rd column is to evaluate if the expected result is true.

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: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing

I wrote tests for the sample code using the Ruby Gem MiniTest to establish if each given function can fulfil its purpose. Each test function contains newly created objects for the test environment as well as an equal assertion statement that checks that the thing you are returning is equal to what you expect it to be and will only pass if this is true. In order for these tests to run, all syntax must be correct in the imported file to be tested.

```ruby
class CardGame


  def checkforAce(card)
    if card.value = 1
      return true
    else
      return false
    end
  end

  dif highest_card(card1 card2)
  if card1.value > card2.value
    return card
  else
    return card2
  end
end
end

def self.cards_total(cards)
  total
  for card in cards
    total += card.value
    return "You have a total of" + total
  end
end
```

```
➜  Static_and_Dynamic_Task_A git:(master) ✗ ruby specs/card_game_spec.rb
specs/card_game_spec.rb:3:in `require_relative': /Users/CodeClan/codeclan_work/Jordan_Smith_pda/practical_tests/Static
_and_Dynamic_Task_A/card_game.rb:24: syntax error, unexpected keyword_end, expecting end-of-input (SyntaxError)
        from specs/card_game_spec.rb:3:in `<main>'
```

This first error is stating that there is an ‘end’ to a function that doesn’t belong on line 24 of the card_game.rb file.

```
➜  Static_and_Dynamic_Task_A git:(master) ✗ ruby
specs/card_game_spec.rb
/Users/CodeClan/codeclan_work/Jordan_Smith_pda/pr
actical_tests/Static_and_Dynamic_Task_A/card_game
.rb:17:in `<class:CardGame>': undefined local var
iable or method `card2' for CardGame:Class (NameE
rror)
        from /Users/CodeClan/codeclan_work/Jordan
_Smith_pda/practical_tests/Static_and_Dynamic_Tas
k_A/card_game.rb:6:in `<top (required)>'
        from specs/card_game_spec.rb:3:in `requir
e_relative'
        from specs/card_game_spec.rb:3:in `<main>
'
```

After removing the extra end from line 24 and running the test file again I now get a fresh error caused by line 17 of the card_game.rb file. This is due to the space in-between the 2 parameters passed into the function on line 17, they should be separated by a comma.

```
17    dif highest_card(card1, card2)
```

We are still getting an error because of the syntax on line 17 declaring the function ‘dif’, should be ‘def’.

```
def highest_card(card1, card2)
```

This next error is occurring on the final line of code on the card_game.rb file as there is a missing ‘end’ to the class.

```
➜  Static_and_Dynamic_Task_A git:(master) ✗ ruby specs/card_game_spec.rb
specs/card_game_spec.rb:3:in `require_relative': /Users/CodeClan/codeclan_work/Jordan_Smith_pda/prac
tical_tests/Static_and_Dynamic_Task_A/card_game.rb:31: syntax error, unexpected end-of-input, expect
ing keyword_end (SyntaxError)
```

```ruby
class CardGame


  def check_for_ace(card)
    if card.value = 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card
    else
      return card2
    end
  end

  def self.cards_total(cards)
    total
    for card in cards
      total += card.value
      return "You have a total of" + total
    end
  end

end
```

I added in an end statement at the end of the exiting code to end the class and whilst I was at it, indented the code correctly for visibility as well as renaming the first function to be in snake case to follow suit to the other 2 and being the general convention. The tests are now all running, with 2 errors occurring and 1 passing. The first error is an undefined method error on line 27, because the function is tied to the class with self, removing the self will allow its use by an object. The second error is also an undefined method, this time due to a single equals on line 10 which should be ‘==‘, the comparison operator rather than its current state using an assignment operator.

```ruby
class CardGame


  def check_for_ace(card)
    if card.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card
    else
      return card2
    end
  end

  def cards_total(cards)
    total
    for card in cards
      total += card.value
      return "You have a total of" + total
    end
  end

end
```

After making these 2 changes and getting a 2nd test passing, there is a further error advising of an undefined local variable of method ‘total’, which is declared on line 26. This is due to ‘total’ not being assigned an initial value before being used in the for loop to store integer values.

```ruby
def cards_total(cards)
  total = 0
  for card in cards
    total += card.value
    return "You have a total of" + total
  end
end
```

```
➜  Static_and_Dynamic_Task_A git:(master) ✗ ruby specs/card_game
_spec.rb
Run options: --seed 21780

# Running:

.E.

Finished in 0.000873s, 3436.4261 runs/s, 2290.9507 assertions/s.

  1) Error:
CardGameTest#test_can_get_cards_total:
TypeError: no implicit conversion of Integer into String
    /Users/CodeClan/codeclan_work/Jordan_Smith_pda/practical_tes
ts/Static_and_Dynamic_Task_A/card_game.rb:29:in `+'
    /Users/CodeClan/codeclan_work/Jordan_Smith_pda/practical_tes
ts/Static_and_Dynamic_Task_A/card_game.rb:29:in `block in cards_
total'
    /Users/CodeClan/codeclan_work/Jordan_Smith_pda/practical_tes
ts/Static_and_Dynamic_Task_A/card_game.rb:27:in `each'
    /Users/CodeClan/codeclan_work/Jordan_Smith_pda/practical_tes
ts/Static_and_Dynamic_Task_A/card_game.rb:27:in `cards_total'
    specs/card_game_spec.rb:27:in `test_can_get_cards_total'

3 runs, 2 assertions, 0 failures, 1 errors, 0 skips
➜  Static_and_Dynamic_Task_A git:(master) ✗
```

The next error is advising a type error caused by line 29 which is attempting to add a number to a string, which is not possible. The number needs to become a string to be added to the string, using the to_int operator.

```
  1) Failure:
CardGameTest#test_can_get_cards_total [specs/card_game_spec.rb:27]:
Expected: "you have a total of 20"
  Actual: "You have a total of5"
```

A string is now being returned, but with the wrong figure being returned for the cards_total function. This is due to the return statement being inside the for loop, meaning the loop finishes at the first run and does not get the true total. The return statement needs to be moved to after the for loop ends. There is also no space between the word “of” and the number, a space added to the end of the string will fix that.

```ruby
def cards_total(cards)
  total = 0
  for card in cards
    total += card.value
  end
  return "You have a total of " + total.to_s
end
```

```
➜  Static_and_Dynamic_Task_A git:(master) ✗ ruby specs/card_game_spec.rb
Run options: --seed 37867

# Running:

...

Finished in 0.000793s, 3783.1021 runs/s, 3783.1021 assertions/s.

3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

All 3 tests are now passing!

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

```java
public abstract class Employee {
    private String name;
    private int nationalInsuranceNumber;
    private double salary;
```

Encapsulation is the restriction of data to a single unit, the example here is private variables within a Java class, meaning inaccessible outside of this class. The example is instance variables of the abstract class Employee.

Week 12

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

src > main > java > ICrushable.java

```java
1   public interface ICrushable {
2       int getMetalContent();
3       void setMetalCountTo0();
4   }
5
```

```java
public class Car implements ICrushable {
```

Car.java

src > main > java > Car.java

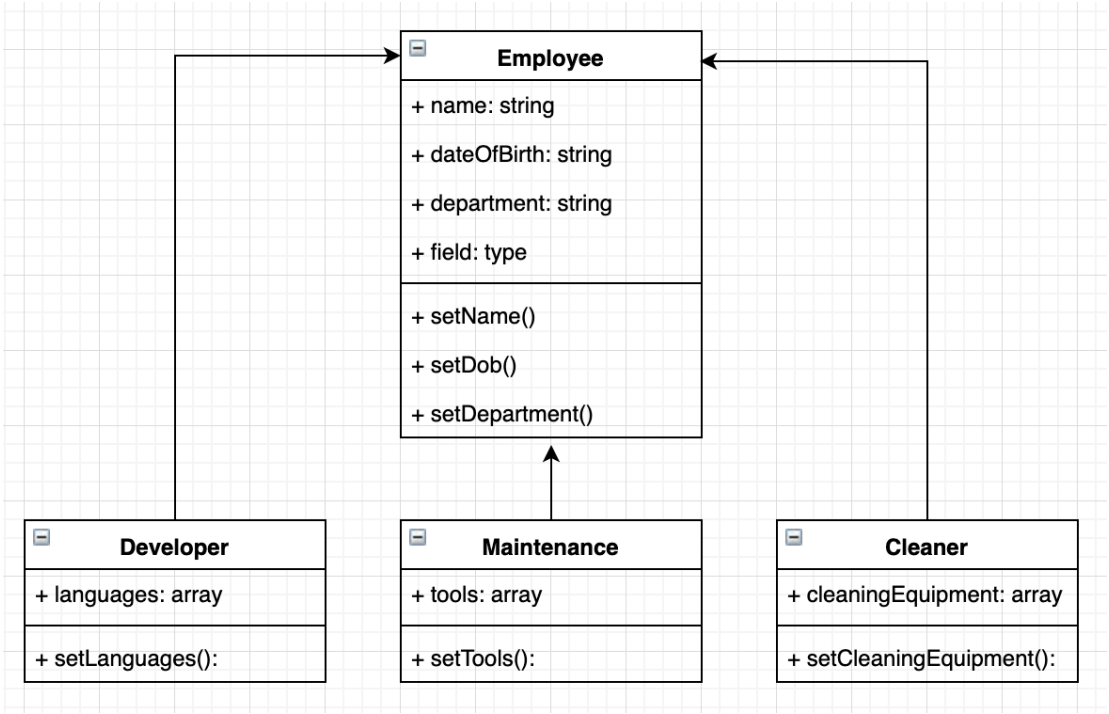
```java
9
10      public int getMetalContent() {
11          return this.metalContentKG;
12      }
13
14      public void setMetalCountTo0() {
15          this.metalContentKG = 0;
16      }
```

```java
    public void addCrushable(ICrushable crushable) {
        this.crushables.add(crushable);
    }

    public void salvageMetal(ICrushable crushable) {
        this.scrapMetalKG += this.crushingMachine.crush(crushable);
    }
```

The literal meaning of polymorphism is ‘many shapes’, in this example there is an interface which is like a superclass but not a class in itself, called ‘ICrushable’ which has 2 set functions that must be implemented when a class implements the interface. The class ‘Car’ implements the interface ‘ICrushable’ to morph the class into an ‘ICrushable’. This class must define the body of the functions from ‘ICrushable’ for it to be usable.

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



This inheritance diagram shows the relationship between classes and superclasses, the superclass ‘Employee’ contains instance variables and methods that the 3 subclasses ‘developer’, ‘maintenance’ and ‘cleaner’ inherit and have access to. Inside of the subclasses are unique instance variables and methods not shared by the other sub classes of ‘employee’.

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.

```
public abstract class Employee {
    private String name;
    private int nationalInsuranceNumber;
    private double salary;

    public Employee(String name, int nationalInsuranceNumber,
        this.name = name;
        this.nationalInsuranceNumber = nationalInsuranceNumbe
        this.salary = salary;
    }

    public String getName() { return name; }

    public int getNationalInsuranceNumber() { return national

    public double getSalary() { return salary; }

    public double raiseSalary(double toRaiseBy) {
        return salary += toRaiseBy;
    }

    public double payBonus(double salary) { return salary / 1


}
```

```
package techStaff;

import staff.Employee;

public class Developer extends staff.Employee {

    public Developer(String name, int nationalInsuranceNumber, double salary) {
        super(name, nationalInsuranceNumber, salary);
    }

}
```

```
public class Developer extends staff.Employee {

    public Developer(String name, int nationalInsuranceNumber, double salary) {
        super(name, nationalInsuranceNumber, salary);
    }

    public String getNameAndNiNumber() {
        return this.name + this.nationalInsuranceNumber;
    }

}
```

```
developer = new Developer( name: "Jimmy",  nationalInsuranceNumber: 54321,  salary: 5000);
```

For this example I chose my employee management application. The screenshots show a superclass called ‘Employee’ & a subclass of ‘Employee’ called ‘Developer’ as well as a ‘Developer’ object including instance variables being used from the parent class. There is also a demonstration of a function in the subclass ‘Developer’ using data/instance variables from the superclass ‘Employee’ in the function named ‘getNameAndNiNumber()’.

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.

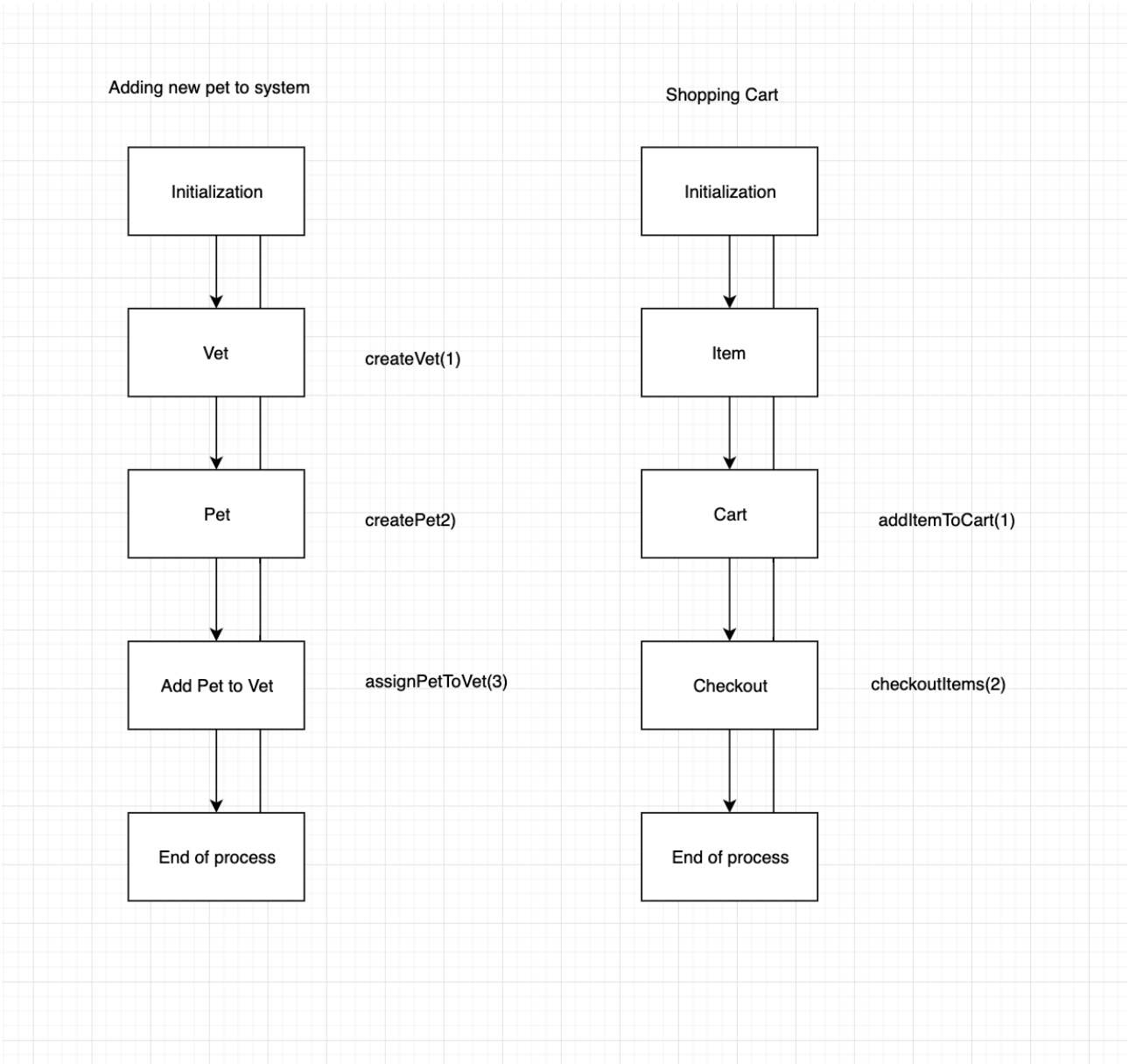
```
public double calculateTotal() {
    for (Item item : items) {
        cartTotal += item.getPrice();
    }
    return cartTotal;
}
```

The first algorithm I have chosen as an example is from a shopping cart program I made using Java, this particular algorithm calculates the shopping cart total. The function achieves this by looping through all items in the shopping cart and selecting the price attribute and adding it into the ‘cartTotal’ variable using ‘+=‘. The return statement then returns the value of all items in the cart added together.

```
public void applyTenPercentDiscountToOrdersOver20() {
    double tenPercent = cartTotal / 10;
    if (cartTotal > 20) {
        cartTotal -= tenPercent;
    }
}
```

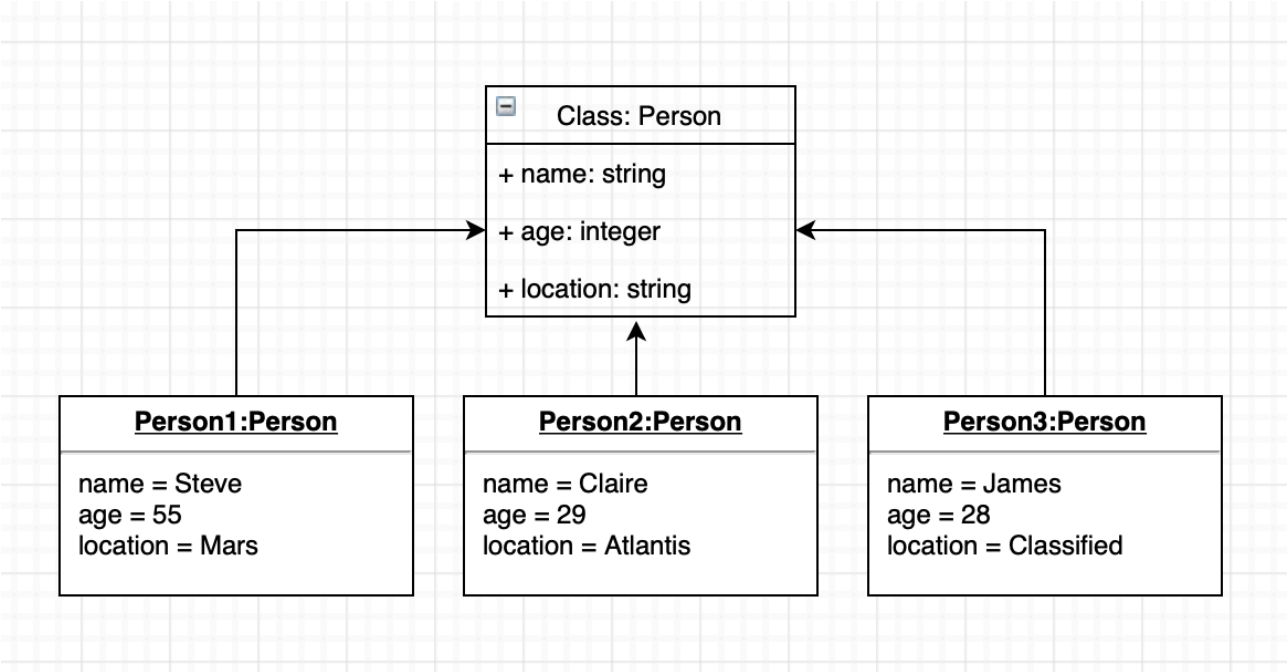
The second example I have chosen, again from a Java shopping cart program is an algorithm that applies a 10 percent discount to the shopping cart when the basked total is over £20. To do this the function first of all calculates 10% of the value of the shopping cart and assigned it to the variable ‘tenPercent’, the function then checks if the value of the cart is above 20. If the value of the cart is over 20 then the function will take the value saved to the ‘tenPercent’ variable from the ‘cartTotal’ instance variable.

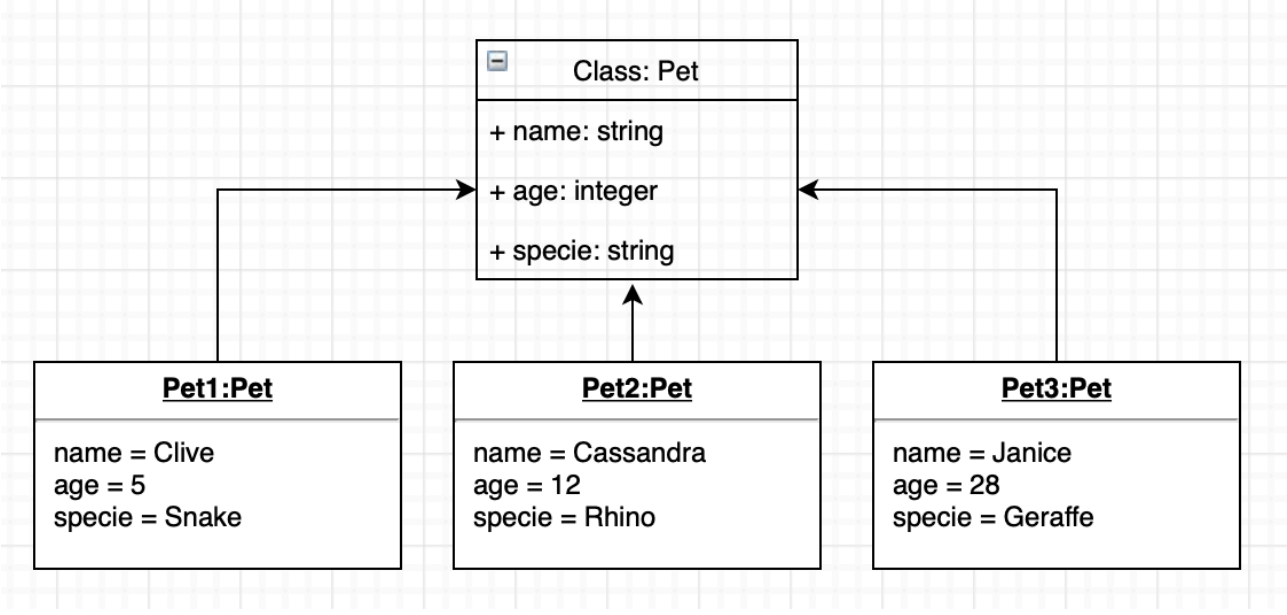
Unit	Ref	Evidence
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).



Above are two collaboration system interaction diagrams, one for a vet practice app adding a new pet to the system and another for adding an item to cart and checking out with a shopping cart app. These diagrams help to map out the flow control along with the data in the system.

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





An object diagram visualises class instances and how the data in the object follows the class rules. These 2 diagrams display object diagrams for 2 classes, a Person class and a Pet class with 3 objects and their data mapped out in a UML diagram.

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

Bug/Error	Solution	Date
RESTful routes not working for /adverts and/sellers.	Added default constructor to the Advert & seller classes as missing,	03/11/19
TypeError: this.props.adverts.map is not a function	State set to wrong api endpoint, amended.	04/11/19
Cannot read property 'title' of undefined	Id being passed into advert not strictly equal due to type, parsed it into integer.	04/11/19
cannot delete advert or seller	Delete request changed to GET request	05/11/19
Product image bigger than containing box	Changed img width to %	06/11/19

Bug tracking reports are like change logs for bugs and errors, you can keep track of the errors you encounter and how you fixed them as well as the date. This can be helpful for things like tracking repeat errors and error fixes causing new errors and getting a fix for an error that has occurred before.