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

Paste Screenshot here

```ruby
@customers = [
  {
    name: "Alice",
    pets: [],
    cash: 1000
  },
  {
    name: "Bob",
    pets: [],
    cash: 50
  },
  {
    name: "Jack",
    pets: [],
    cash: 100
  }
]
```

```ruby
def customer_can_afford_pet(customer, pet)
  return customer[:cash] >= pet[:price]
end
```

```
➜ start_point git:(master) x ruby specs/pet_shop_spec.rb
Run options: --seed 4191

# Running:

.................

Finished in 0.001892s, 8985.2010 runs/s, 8985.2010 assertions/s.
```

```ruby
def test_customer_can_afford_pet__sufficient_funds
  customer = @customers[0]
  can_buy_pet = customer_can_afford_pet(customer, @new_pet)
  assert_equal(true, can_buy_pet)
end
#
def test_customer_can_afford_pet__insufficient_funds
  customer = @customers[1]
  can_buy_pet = customer_can_afford_pet(customer, @new_pet)
  assert_equal(false, can_buy_pet)
end
#
def test_customer_can_afford_pet__exact_funds
  customer = @customers[2]
  can_buy_pet = customer_can_afford_pet(customer, @new_pet)
  assert_equal(true, can_buy_pet)
end
```

These screenshots show a hash, a function that uses the hash and the result of the function running correctly in a passed test. The function checks whether a customer can afford a pet, it takes in the customer and pet Hash and checks to see if the customers case is greater than or equal too the price of the pet. The test tests that the customer can afford each individual pet.

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

Paste Screenshot here

```ruby
class Rooms

  attr_reader :name
  attr_accessor :till

  def initialize(name, guests, songs, till)
    @name = name
    @guests = []
    @songs = []
    @till = till
  end
  #count number of guests already in guests array
  def guest_count
    return @guests.count
  end
  #add a guest to the guests array in room
  def add_guest(guest_to_be_added)
    @guests << guest_to_be_added
  end
```

```
➜  ~ cd codeclan_work/PDA
➜  PDA git:(master) open .
➜  PDA git:(master) cd ..
➜  codeclan_work cd week_02/karaoke/specs
➜  specs git:(master) ruby rooms_specs.rb
Run options: --seed 63780

# Running:

..........

Finished in 0.001466s, 6821.2822 runs/s, 8867.6668 assertions/s.

10 runs, 13 assertions, 0 failures, 0 errors, 0 skips
➜  specs git:(master)
```

```ruby
  def test_add_guests
    @room1.add_guest(@guest1)
    @room1.add_guest(@guest2)
    @room1.add_guest(@guest3)
    assert_equal(2, @room1.guest_count)
  end

  def test_remove_guests
    @room1.add_guest(@guest1)
    @room1.add_guest(@guest2)
    @room1.add_guest(@guest3)
    assert_equal(2, @room1.guest_count)
    @room1.remove_guest(@guest1)
    assert_equal(1, @room1.guest_count)
  end
```

Description here

This shows an array being used in a function and the result of the functions being tested, the test shows that guests are being added and removed to the array by counting the length of the array, the array is a list of guests which the rooms have

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

Paste Screenshot here

```ruby
def stars
  sql = 'SELECT stars.* FROM stars
        INNER JOIN castings on castings.star_id = stars.id
        WHERE movie_id = $1'
  values = [@id]
  stars_data = SqlRunner.run(sql, values)
  return stars_data.map {|star_data| Star.new(star_data) }
end
```

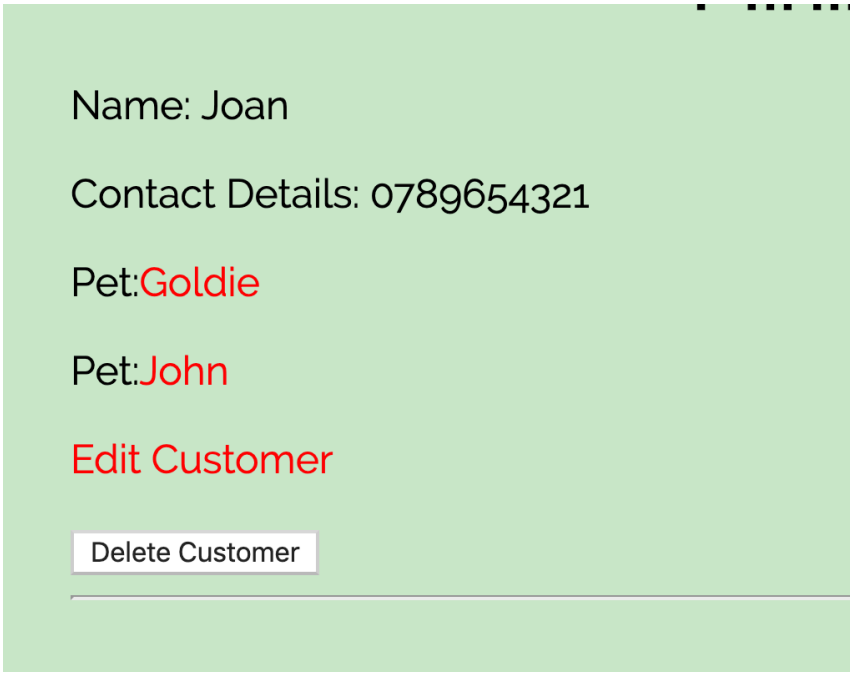
```
➜  imdb_lab git:(master) ruby start_code/console.rb

From: /Users/alexwoods/codeclan_work/week_03/day_4/imdb_lab/start_code/console.r
b @ line 30 :

    25: casting3.save
    26:
    27:
    28:
    29: binding.pry
 => 30: nil

[1] pry(main)> movie1.stars
=> [#<Star:0x007fea9d980b18 @first_name="Joe", @id=5, @second_name="Pesci">,
 #<Star:0x007fea9d9808e8 @first_name="Ray", @id=6, @second_name="Liotta">]
[2] pry(main)>
```

This function searches for all the stars that are cast in a movie using a SQL Query, when it is run in Pry it displays both Joe Pesci and Ray Liotta who are stars in the movie Goodfellas. The SQL code in the first screenshot is searching for all of the stars cast in a movie using the movie id to distinguish between the different movies

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

Paste Screenshot here

```
92
93    def self.find(id)
94      sql = "SELECT * FROM pets
95      WHERE id = $1"
96      values = [id]
97      result = SqlRunner.run(sql, values).first
98      pet = Pet.new(result)
99      return pet
00    end
01
02
```



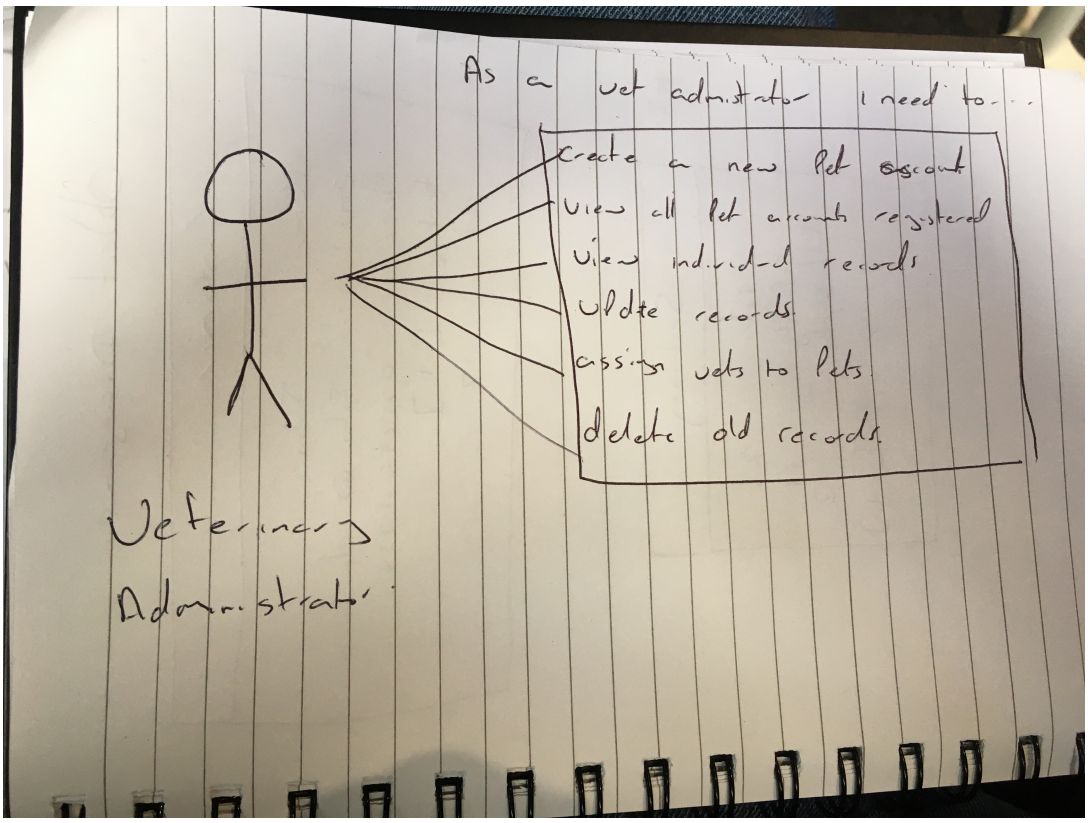
Description here

This function sorts through all pets data by id and then uses the sorted data to assign pets to a customer. It uses the pet id as a search marker because the id is unique to each individual pet. The found pet is then return by the function which can then be used in other areas of the application.

Week 4

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

Paste Screenshot here

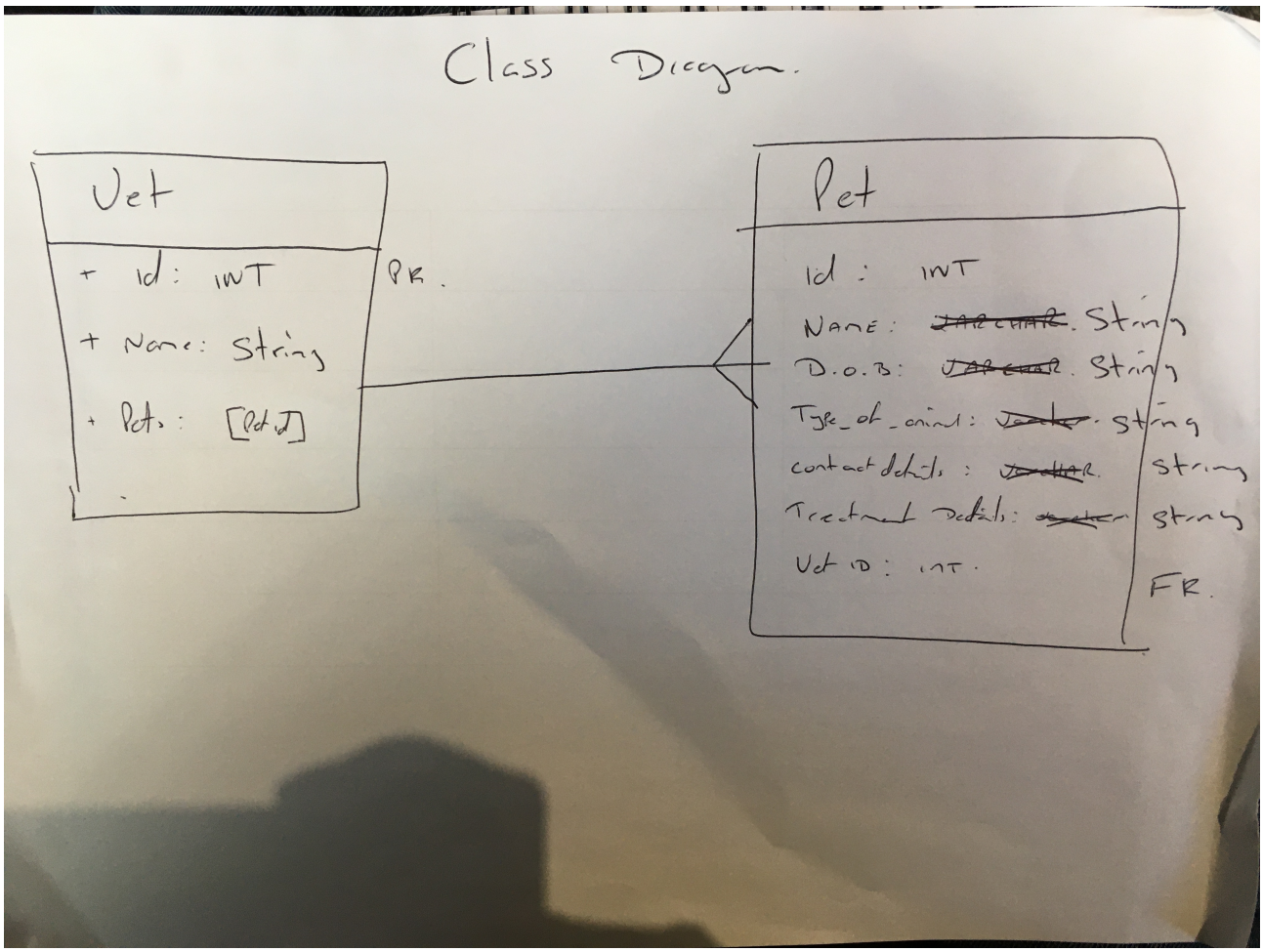


Description here

This use case diagram shows how a user would interact with a software system. It shows how an administrator working for a vet practice would need different requirements from the same application in order to be able to do their job correctly.

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

Paste Screenshot here

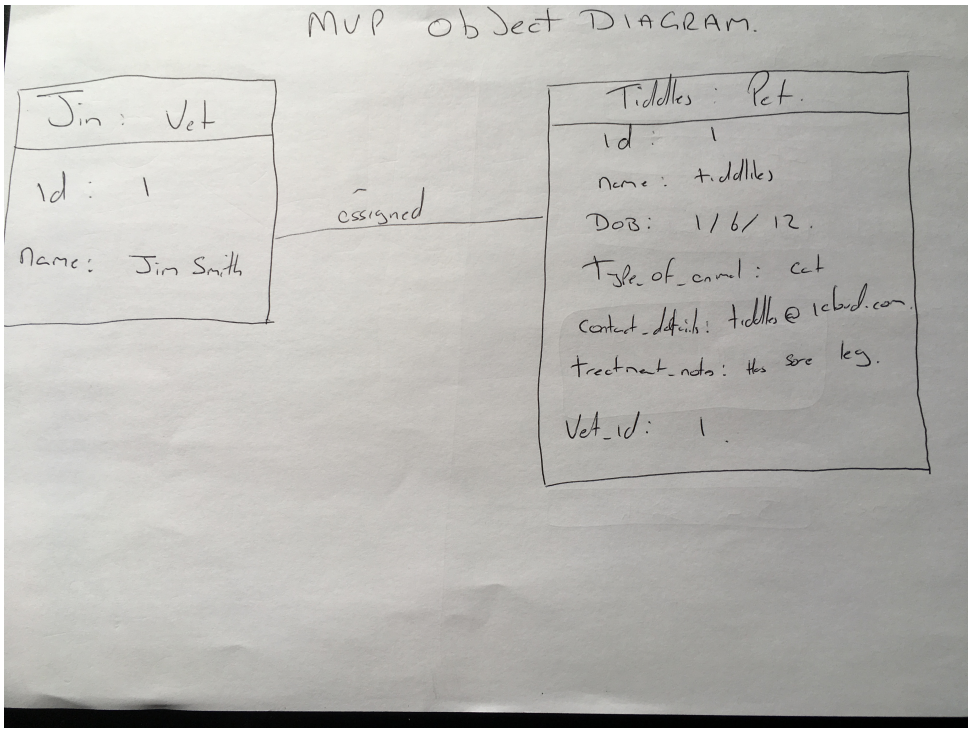


Description here

This class diagram shows the relationship between different classes and the structure of different classes. It shows that one vet can have many pets and also the properties that required in each of the vet and pet classes. This enable the programmer to build a database using this information

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

Paste Screenshot here

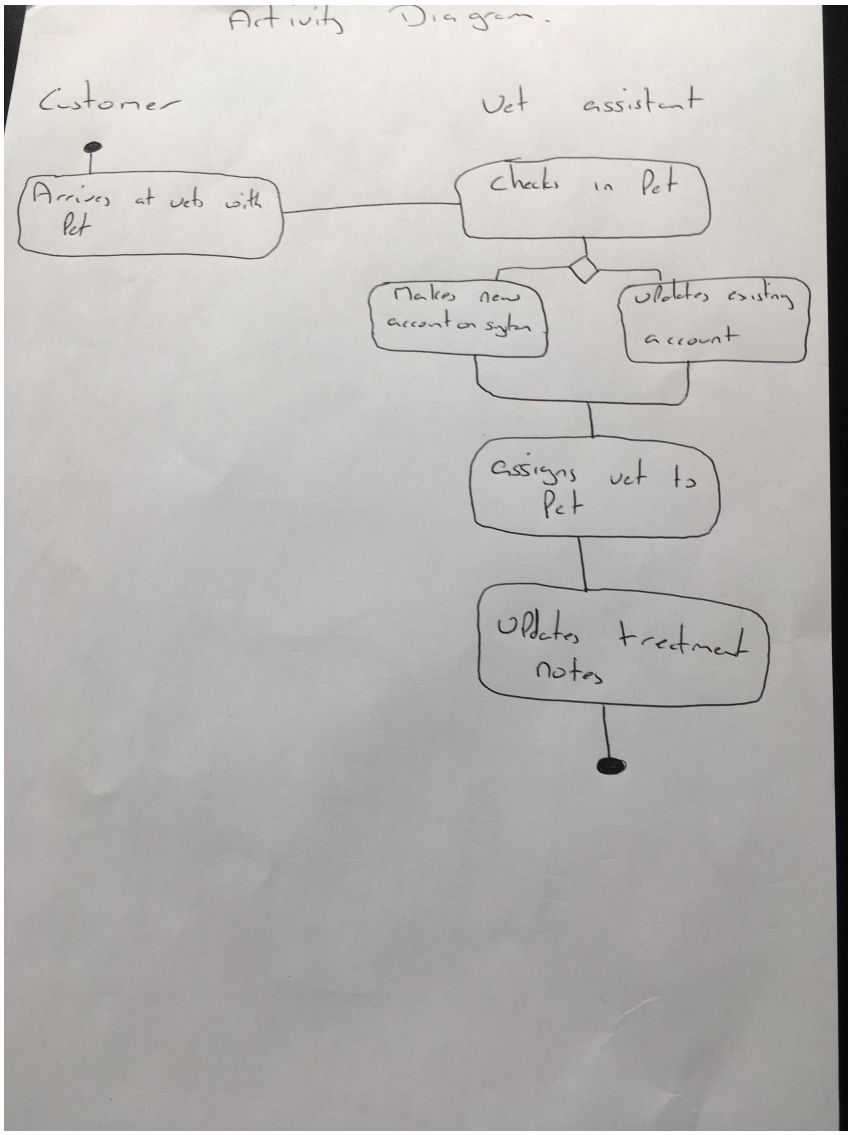


Description here

This object diagram shows how the class diagram would look once it is populated with data in a database at a certain period of time. It shows actual objects instead of the classes

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

Paste Screenshot here



Description here

This activity diagram shows how the user will use the system and the different paths they could take

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

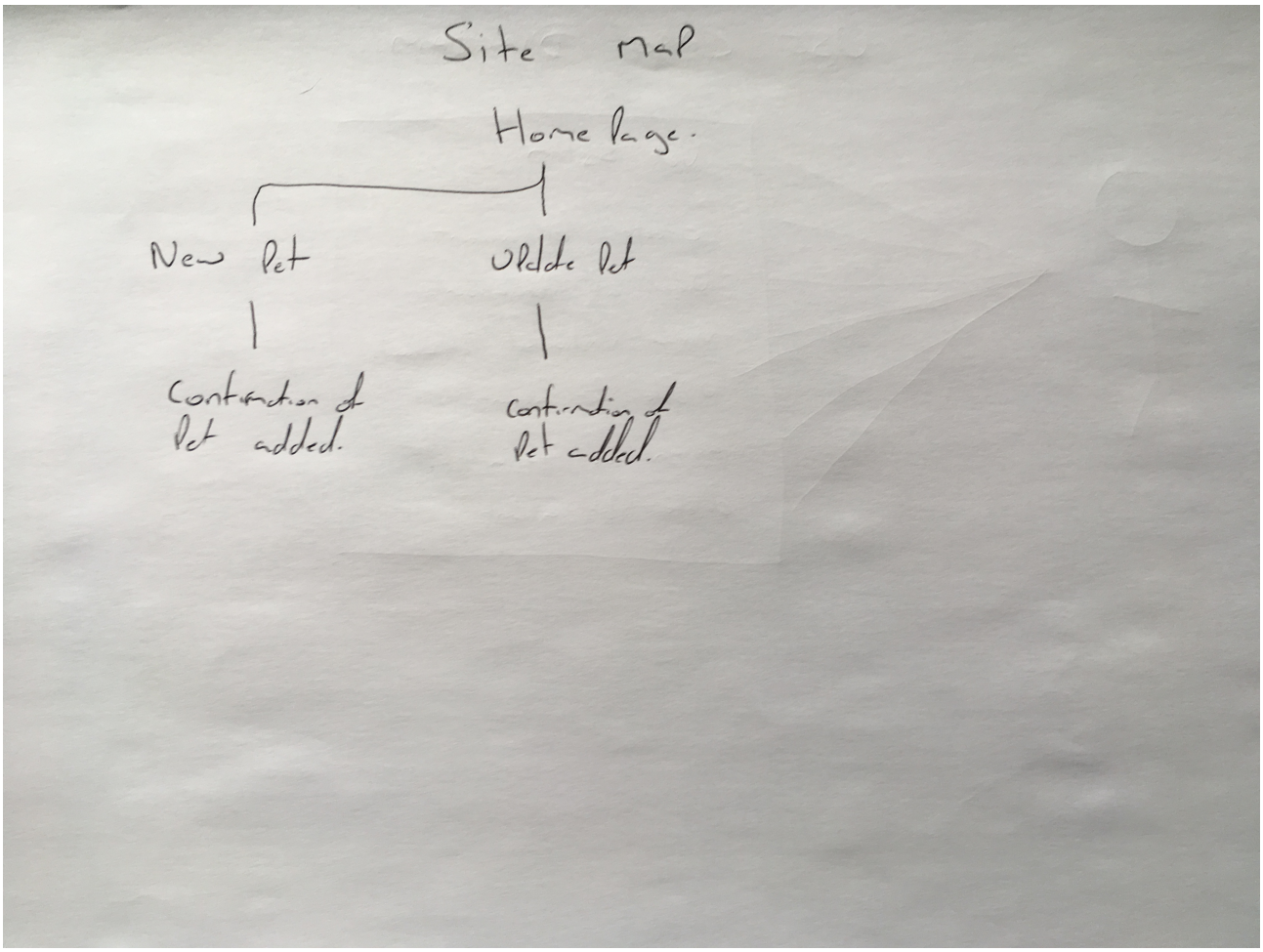
Hardware Constraints	Need to have internet access to be able to run the web app. Also need to have access to the various technologies used in the application. Technology could be designed to run automatically or training provided for the user
Performance constraints	This is a lightweight application with a small database so it should run smoothly on most modern machines. As the user base grows it may need a dedicated server and database to continue to run smoothly
Usability	Application needs to be used by people with an average or above average IT Literacy, this can be improved by simple and clear labelling in thee front end and improved accessibility for some users(i.e screen readers etc.)
Budget Constraints	There is no budget for this program therefore it would be best to use open source technology
Time Constraints	The deadline for this application is 5 days therefore it will need to be kept simple to get a working product ready in time
Persistent storage and transactions	The data will need to persist in orderer the application to be useful, therefore a database will be used

Description here

This plan shows the implementations and constraints that are placed on this project. It highlights some of the limitations that are present and provides suggestions on how to overcome these problems

Unit	Ref	Evidence
P	P.5	User Site Map

Paste Screenshot here

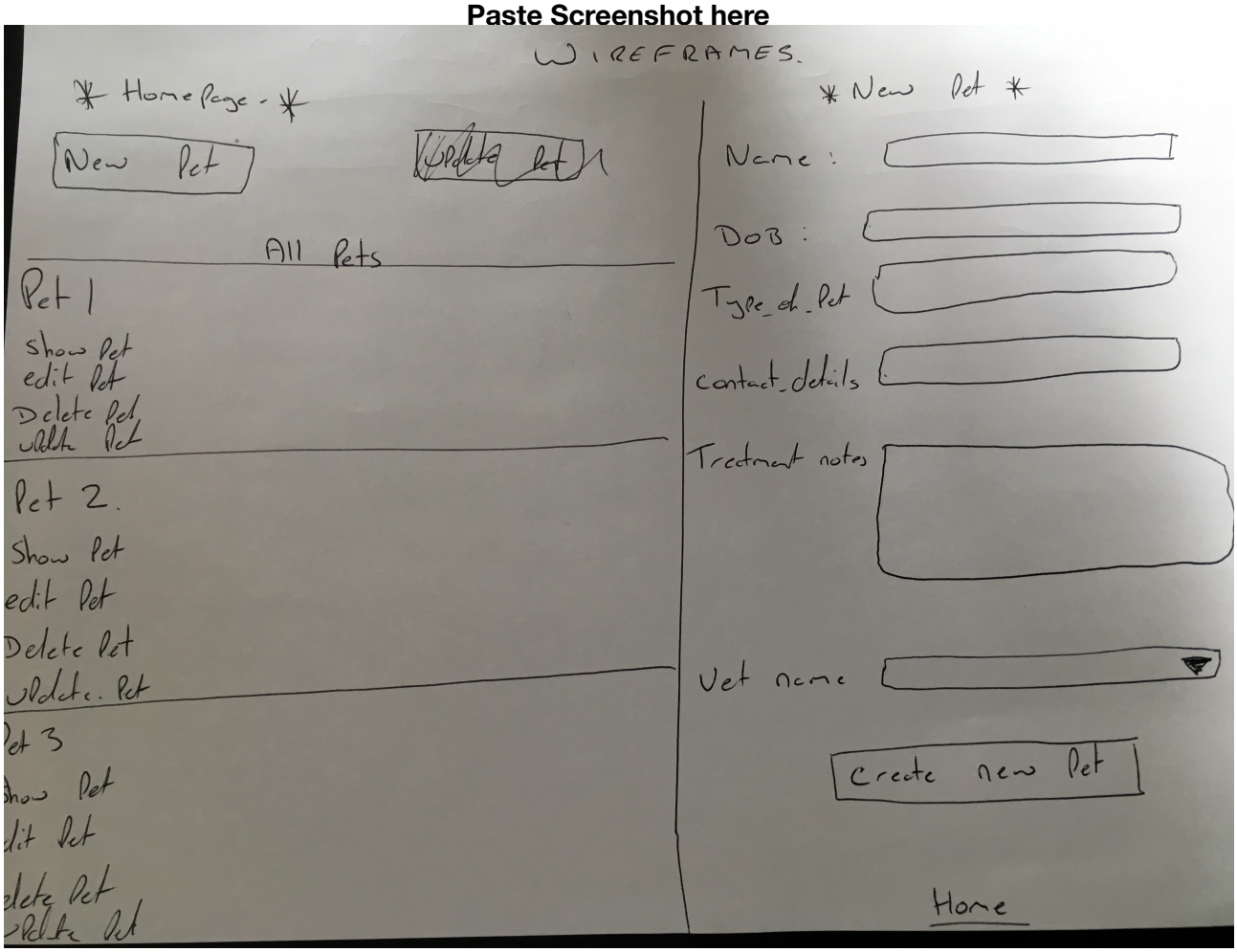


Description here

This site map shows the basic layout of the website and the navigation, it shows how a user would navigate through the site starting with the homepage and allowing them to add a new pet and update an existing pet

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams

Paste Screenshot here



Description here

These are 2 wireframe diagrams for the landing page and form to add a new pet. They show an initial plan for the layout of each web page. A developer would be able to use these to plan their HTML.

Week 5

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

Paste Screenshot here

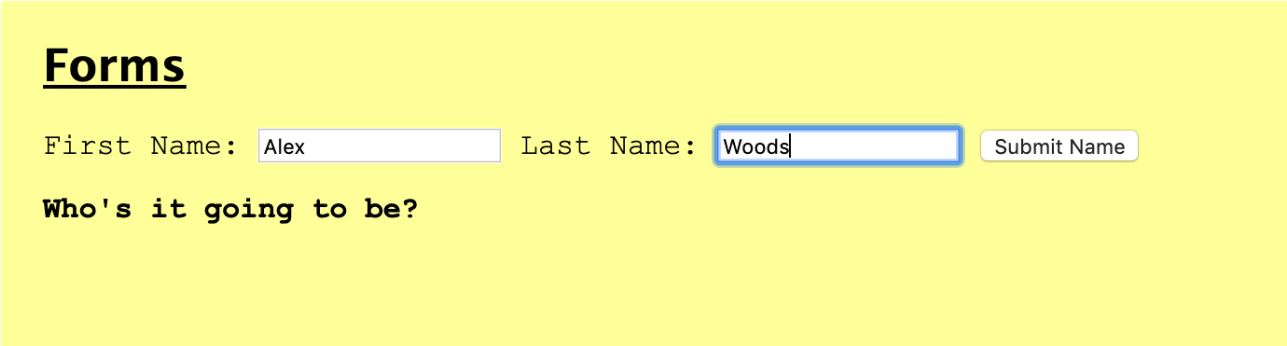
```
  //start function to update text
const handleInput = function(event) {
  //get value from input
  const inputtedText = event.target.value
  //update paragraph text
  const resultParagraph = document.querySelector('#input-result')
  resultParagraph.textContent = inputtedText + ' has been typed'
}

```

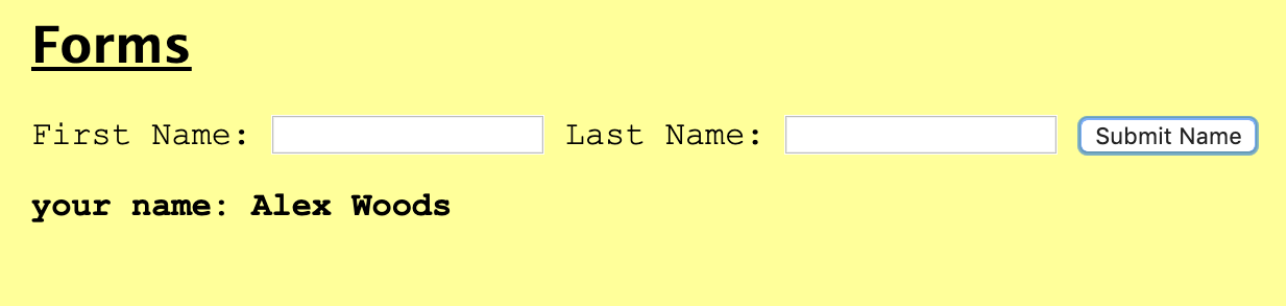
Description here

This Pseudocode shows each of the steps that a method will need to do in order to function correctly. In this case it is updating some text on a webpage. This method gets a value from an input, adds the input to paragraph text and then outputs it to a webpage.

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

Paste Screenshot here



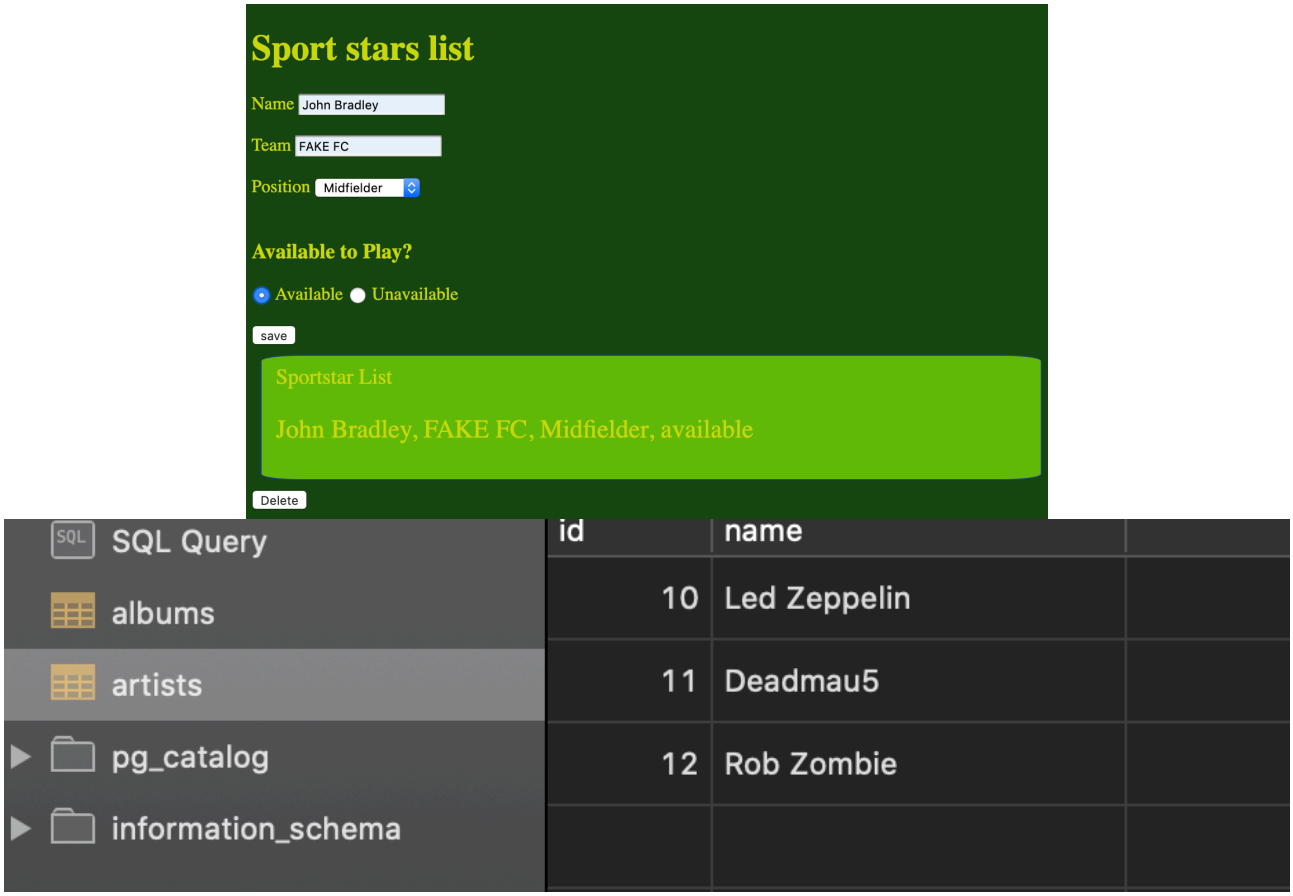


Description here

These screenshots show a user inputting information into a form, this information is then saved and logged to the paragraph below the form

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

Paste Screenshot here

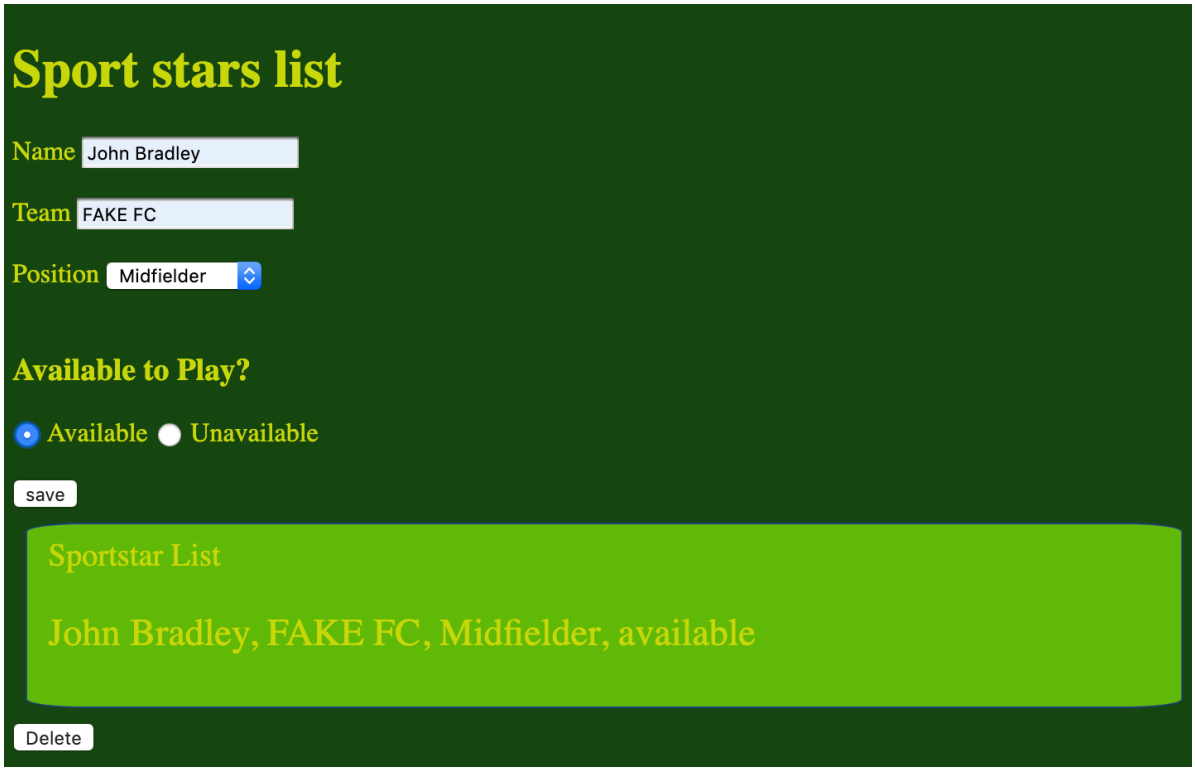


Description here

This screenshot shows an application that takes data into a program and saves/returns it, this data will persist even if the webpage is refreshed as it is being stored and recalled from an existing database

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

Paste Screenshot here

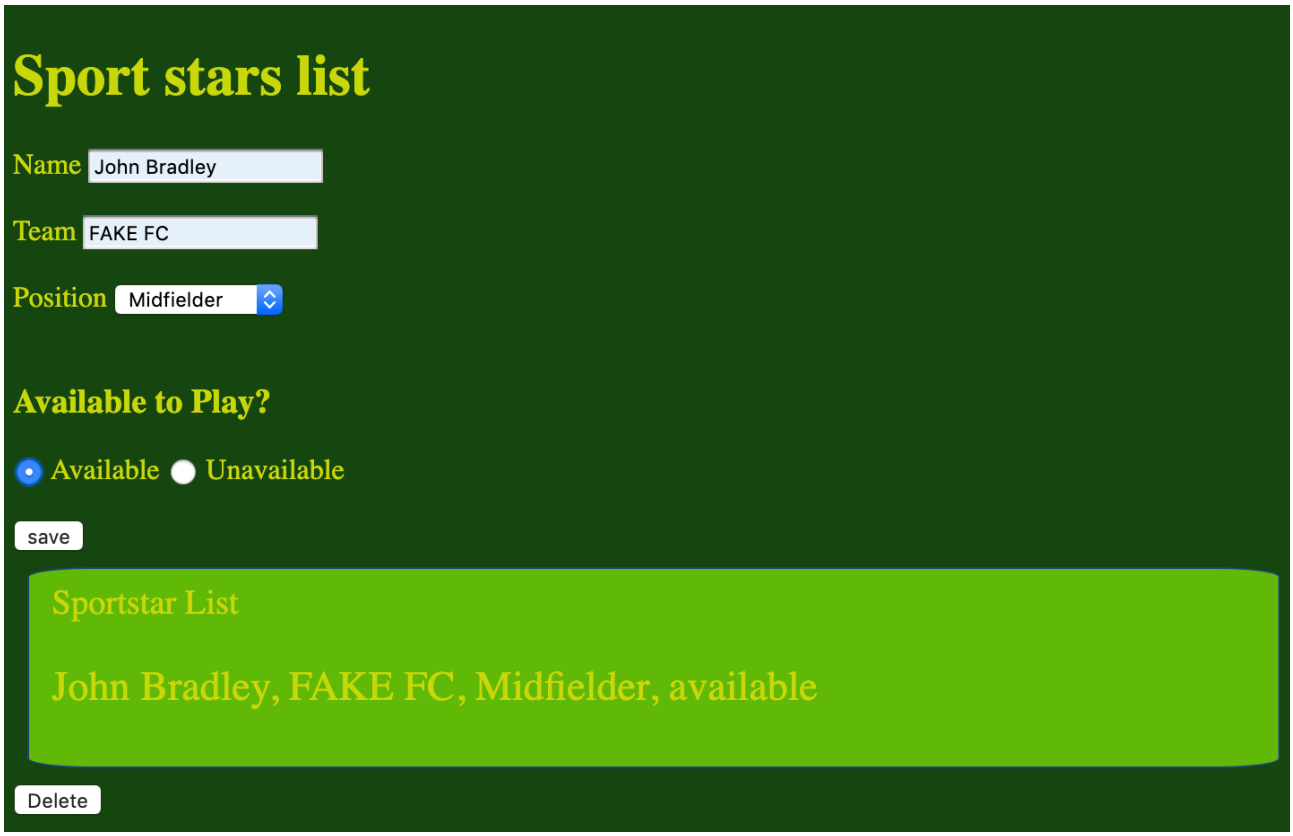


Description here

In this program the user requests the data they have inputted to be saved and it saves and returns it on the same webpage. This data in the form is saved to a database, the application then calls the information from the database and renders it in the light green box at the bottom of the screenshot.

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

Paste Screenshot here

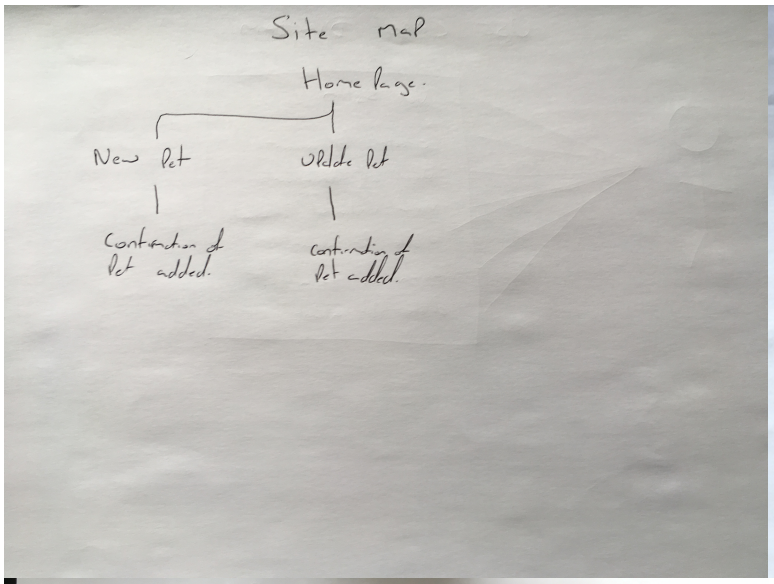


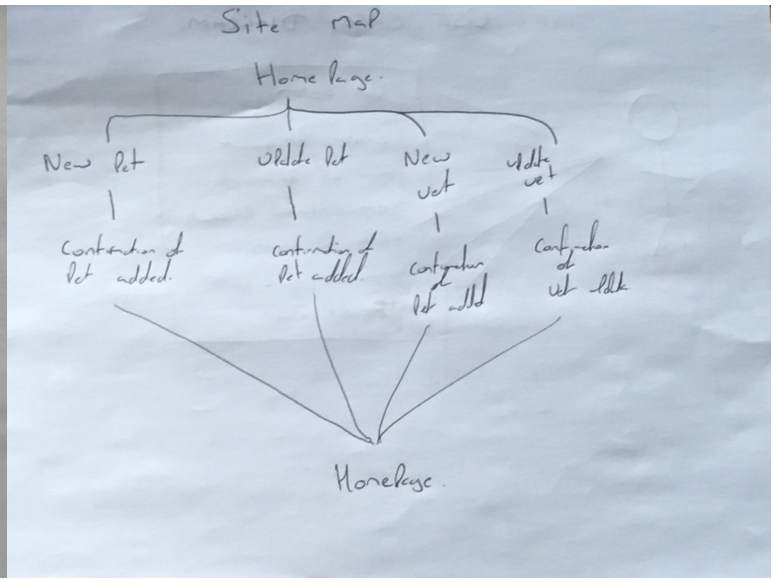
Description here

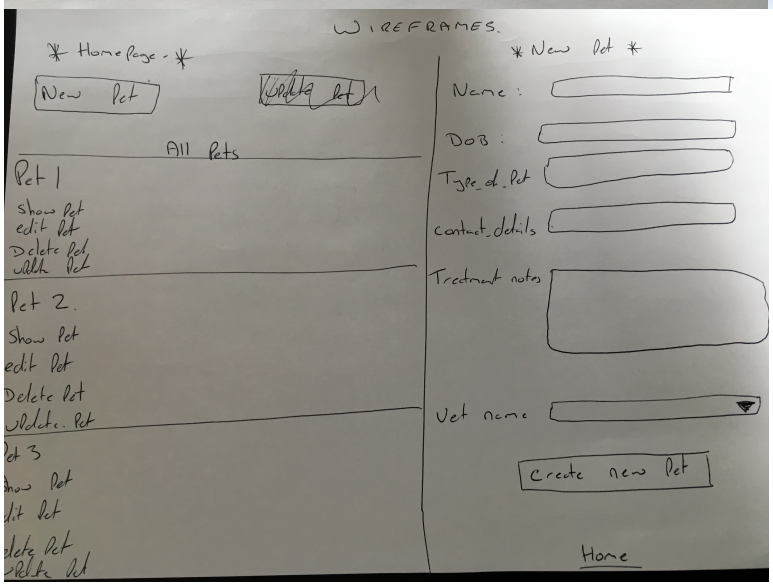
https://github.com/alex-woods89/w6-wkend-hw

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

Paste Screenshot here







Description here

These screenshots show how my sitemap evolved during the Ruby Project as I added more functionality to my web app. The sitemap was initially very simple however after each stage of development I decided to make it more complex and add more pages and tables to the sit

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

Paste Screenshot here

```
45    mounted(){
46      fetch('https://api.punkapi.com/v2/beers')
47      .then(result => result.json())
48      .then(beers => this.beers = beers)
49
```

All the beers!

Buzz
Trashy Blonde
Berliner Weisse With Yuzu - B-Sides
Pilsen Lager
Avery Brown Dredge
Electric India
AB:12
Fake Lager
AB:07
Bramling X
Misspent Youth
Arcade Nation
Movember
Alpha Dog
Mixtape 8
Libertine Porter
AB:06
Russian Doll – India Pale Ale
Hello My Name Is Mette-Marit
Rabiator
Vice Bier
Devine Rebel (w/ Mikkeller)
Storm
The End Of History
Bad Pixie

Description here

This fetch request in the first screenshot gets the api and the second screenshot shows the result of the program running and using the API to display a list of all the beers.

Week 8

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

Paste Screenshot here

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

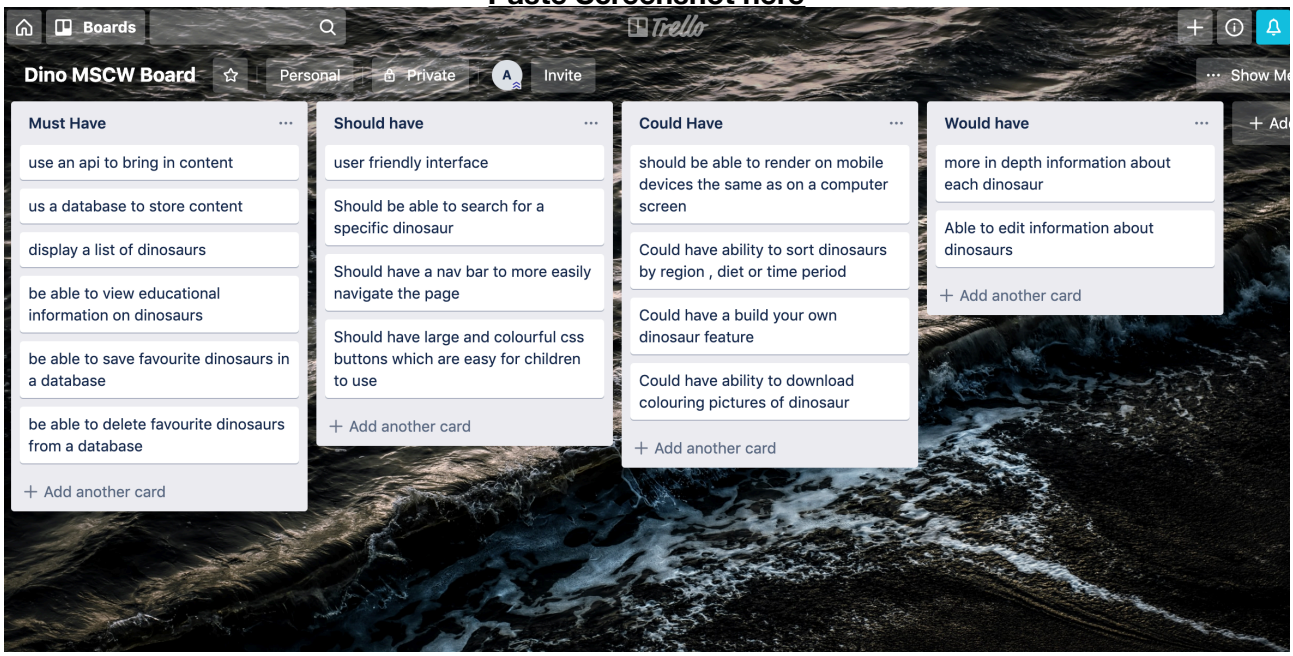
- https://www.highcharts.com/ HighCharts is an open-source library for rendering responsive charts.
- https://korigan.github.io/Vue2Leaflet/#/ Leaflet is an open-source library for rendering maps and map functionality.

Description here

This screenshot shows the project brief from the group project, we decided to produce an educational app.

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

Paste Screenshot here



Description here

This is a screenshot of a trello Moscow board which shows the things our app must have, should have, could have and would have. We prioritised the must have features and did not work on the would have features due to time constraints

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

Paste Screenshot here

User	As a vet administrator I want to be able to store a list of all the pets in the practice	As a vet administrator I want to be able to edit details of all the pets in the practice as their case develops
Acceptance	Data needs to be persisted in the database to ensure it is safe	User needs to be able to edit data in the database
Test	Pass	Pass

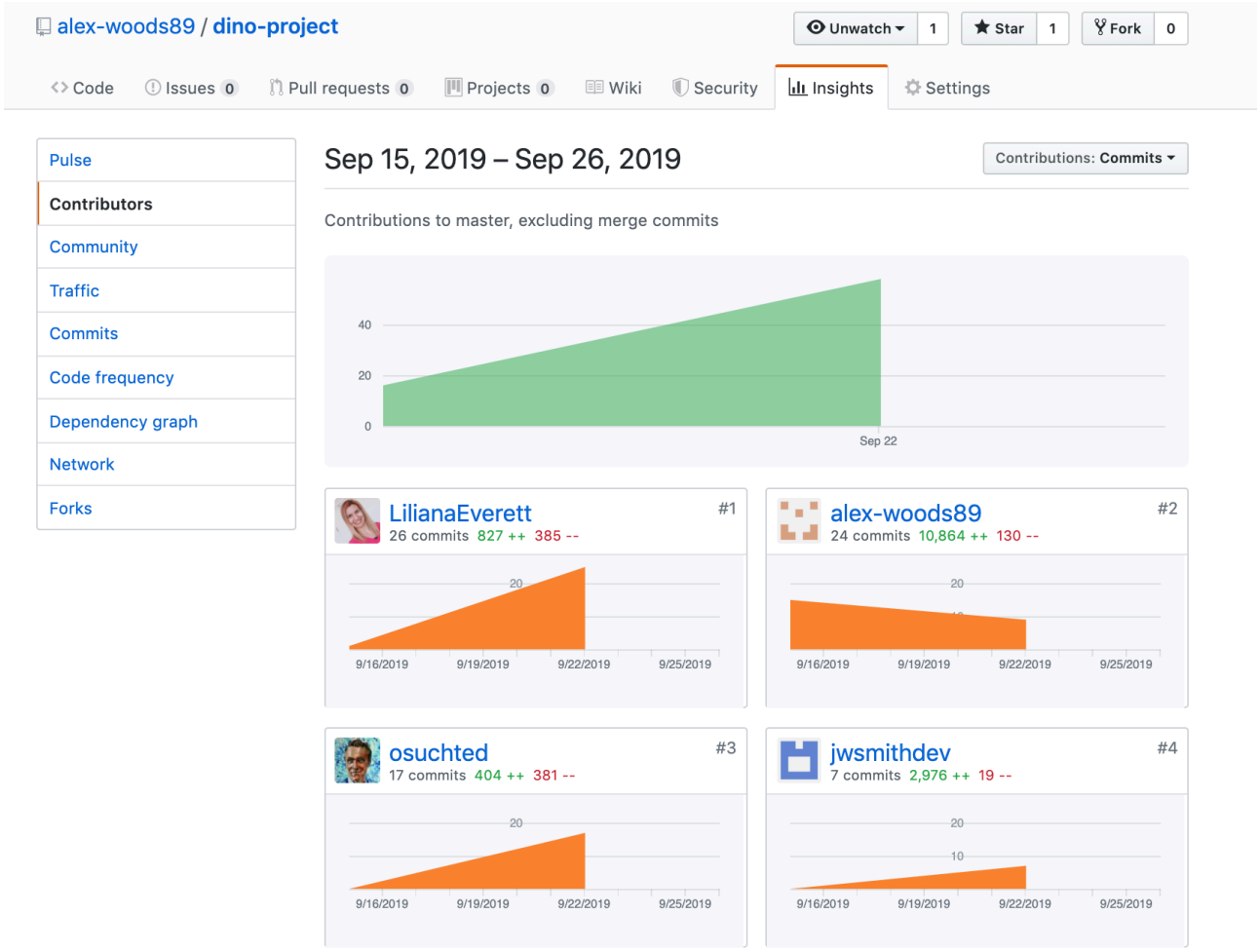
Description here

This is the acceptance, criteria and test plan used for the project, it shows the criteria that the user would need, what criteria it could be tested against and also whether or not these tests 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.

Paste Screenshot here



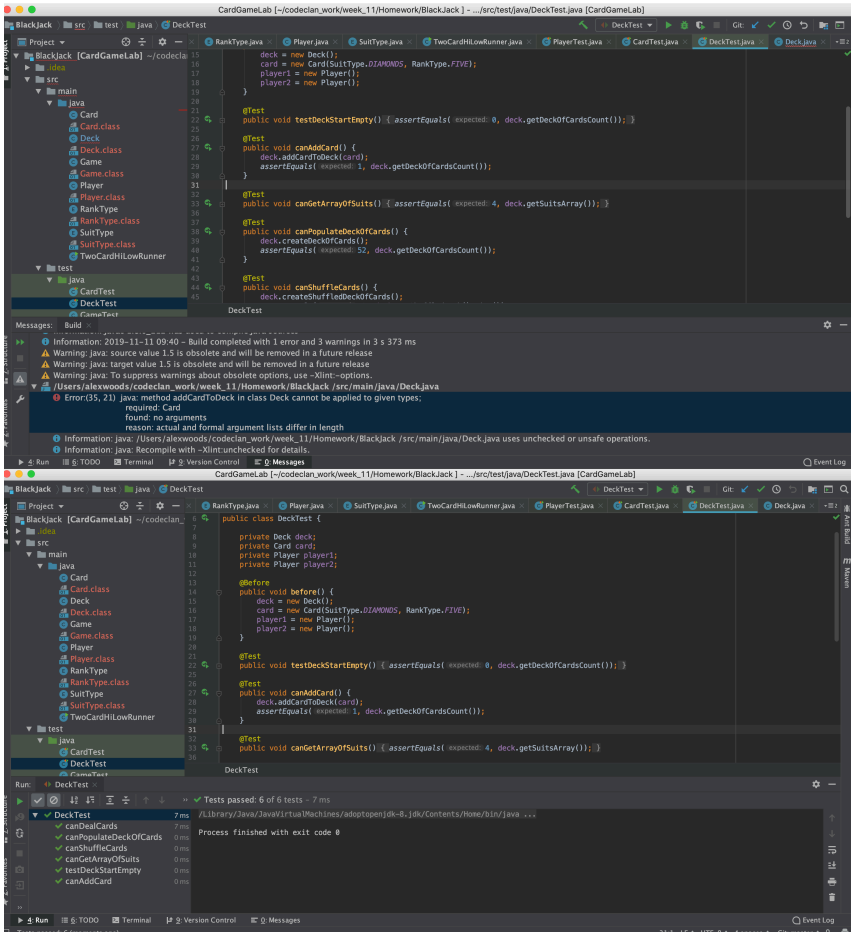
Description here

This screenshot shows the contributors page on GitHub for the Javascript group project

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

Paste Screenshot here



```
public void addCardToDeck(Card card) { this.deckOfCards.add(); }
```

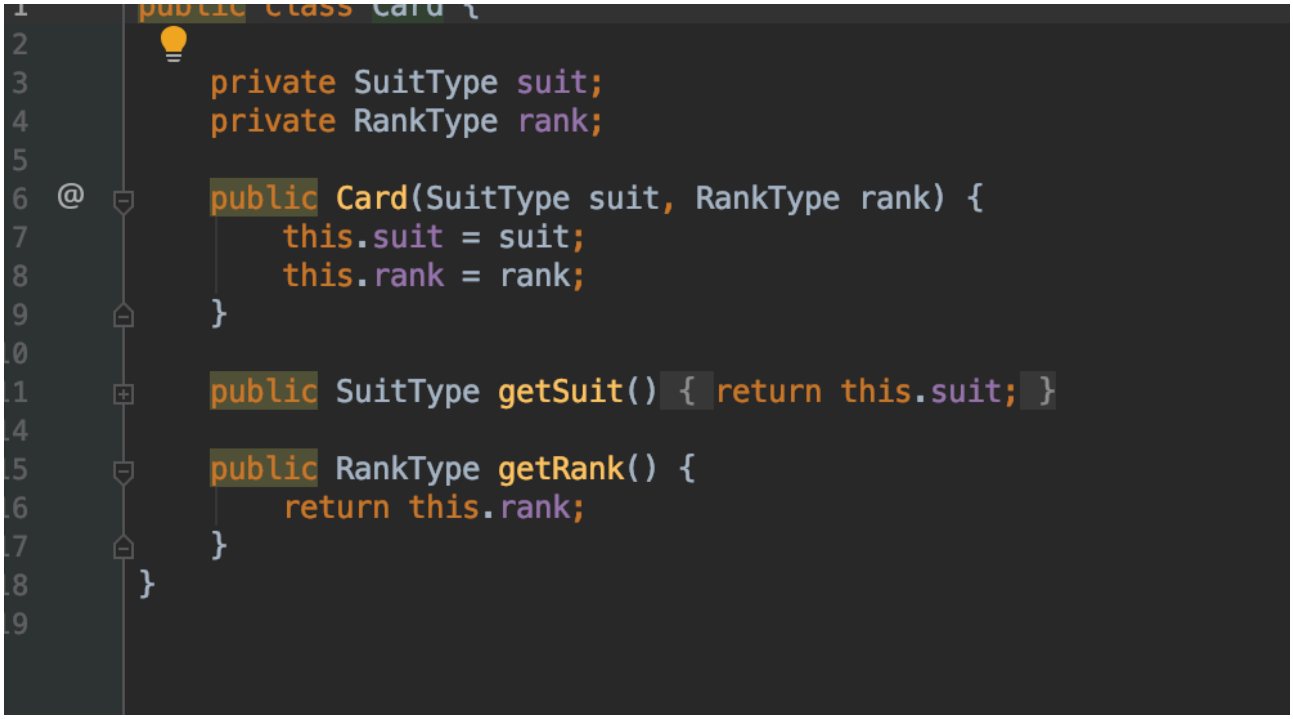
```
public void addCardToDeck(Card card) { this.deckOfCards.add(card); }
```

Description here

The screenshots above show a series of tests for a BlackJack game, the aim of the tests is to test that a deck of cards can be populated. The first screenshot shows the test failing because a card isn't being added to the deck correctly, the second set of tests shows the tests passing because the initial error has been fixed. The third and fourth screenshots show the code before and after the error was fixed

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

Paste Screenshot here



```
1   public class Card {
2
3       private SuitType suit;
4       private RankType rank;
5
6   @   public Card(SuitType suit, RankType rank) {
7           this.suit = suit;
8           this.rank = rank;
9       }
10
11      public SuitType getSuit() { return this.suit; }
14
15      public RankType getRank() {
16          return this.rank;
17      }
18  }
19
```

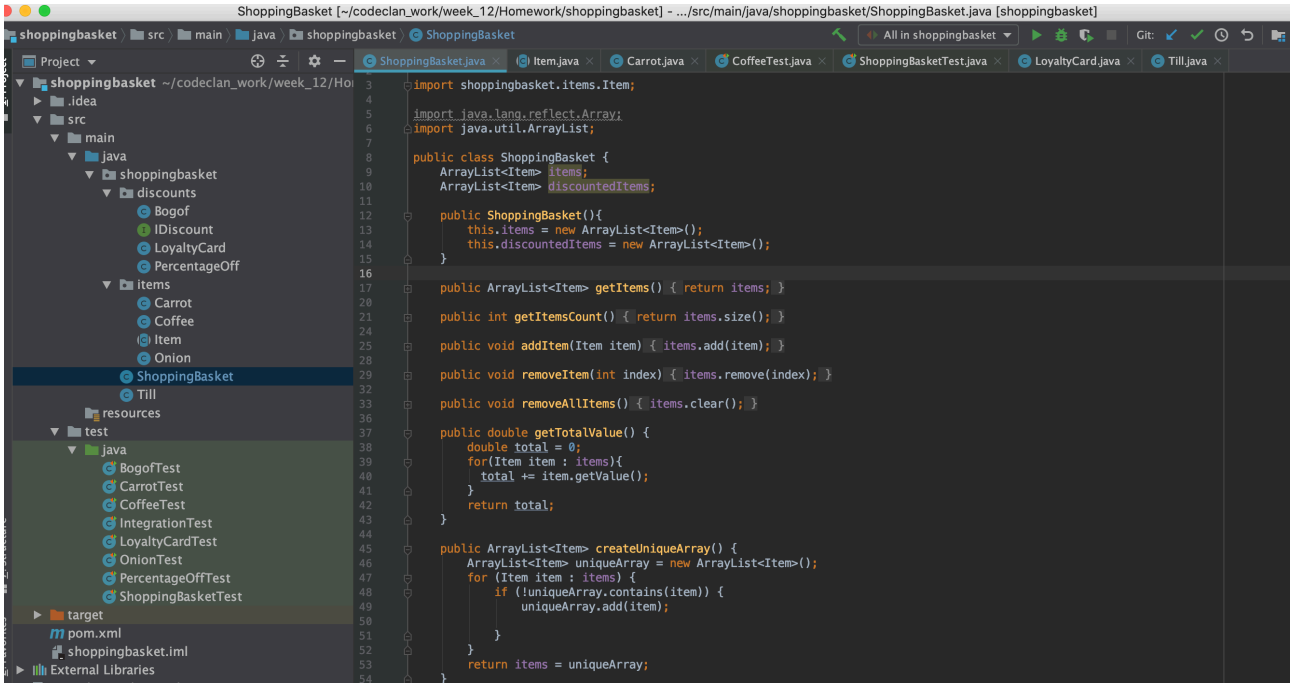
Description here

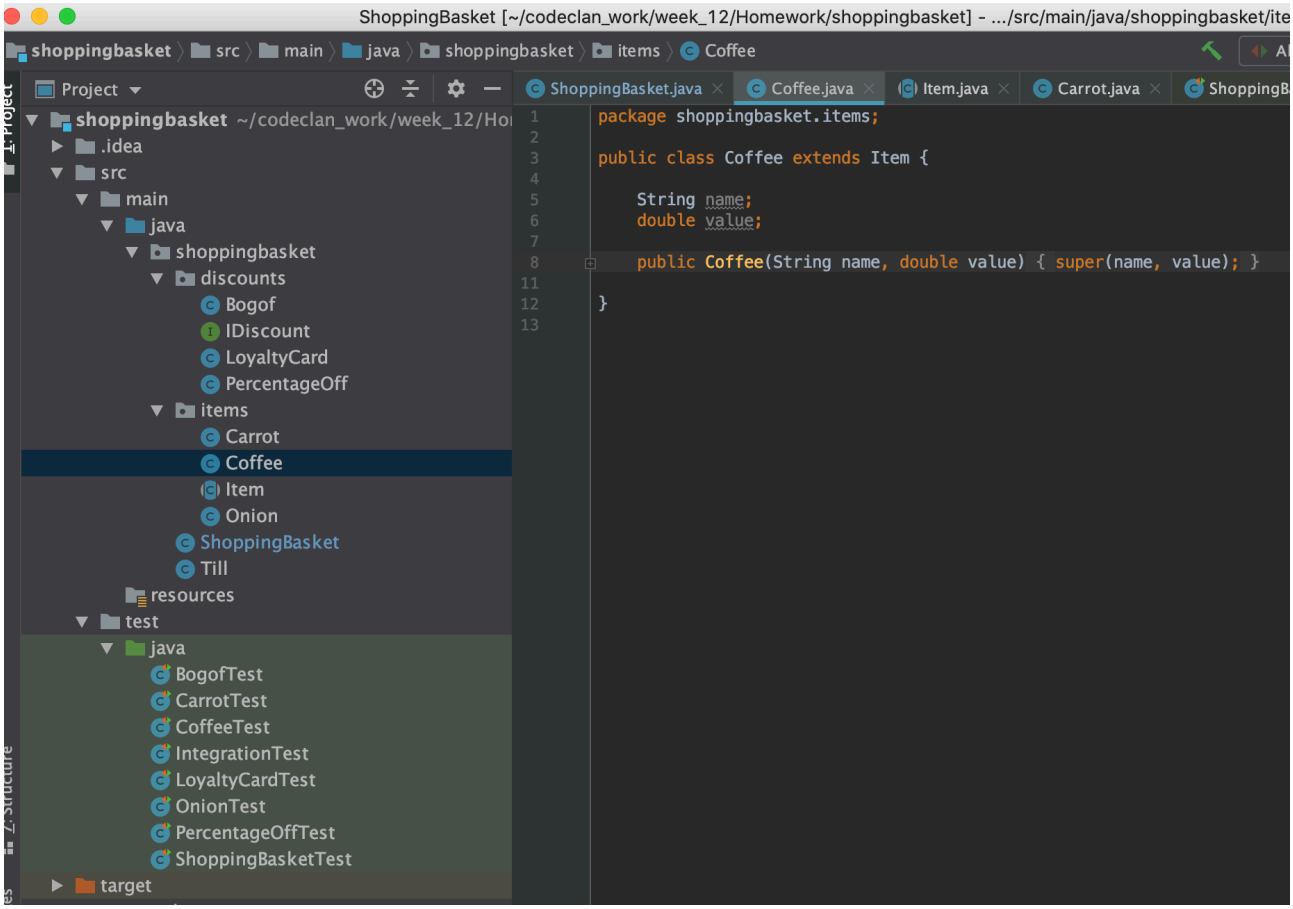
This screenshot shows encapsulation in a card class for a BlackJack Program, the suit and rank of each of the cards is encapsulated in the class therefore only the card class knows the suit and rank of each card. If other classes want to access this data they need to use the getSuit and getRank methods.

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



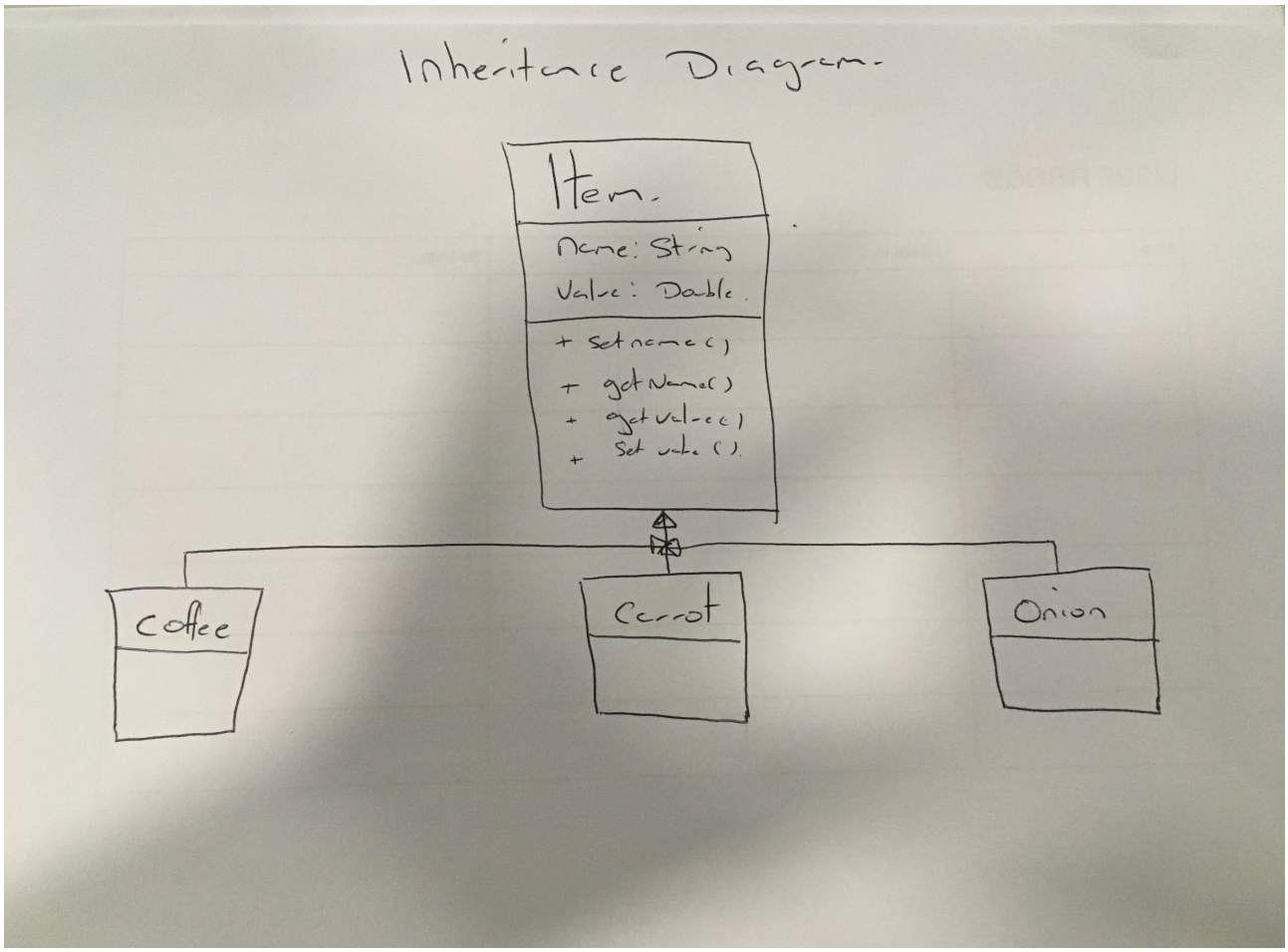


Description here

The above screenshots show an example of polymorphism, the shopping basket can take in a list of items. The coffee class is extending from the item class therefore it can be both a coffee object and an item object as required. It also means that the shopping basket can take in different types of items and not just coffee.

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

Paste Screenshot here



Description here

This diagram shows how individual items all inherit from the item class, as they all have similar properties and structure they can extend the item class which makes the code much more DRY and easier to understand. It also means you can add new items quickly and easily.

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

```java
package players;

import ...

public class Barbarian extends AbstractPlayer {

    public Barbarian(String name, IWeapon weapon, ISpell spell) { super(name, weapon, spell,  hp: 100); }

}
```

```java
package players;

import ...

public abstract class AbstractPlayer {
    private String name;
    private IWeapon weapon;
    private ISpell spell;
    private int hp;

    public AbstractPlayer(String name, IWeapon weapon, ISpell spell, int hp) {
        this.name = name;
        this.weapon = weapon;
        this.spell = spell;
        this.hp = hp;
    }

    public String getName() { return name; }

    public IWeapon getWeapon() { return weapon; }

    public void setWeapon(IWeapon weapon) { this.weapon = weapon; }

    public ISpell getSpell() { return spell; }

    public void setSpell(ISpell spell) { this.spell = spell; }

    public int getHp() { return hp; }

    public void increaseHp(int value) { hp += value; }

    public void decreaseHp(int value) { hp -= value; }

    public int attack() { return weapon.attack(); }

    public int cast() { return spell.cast(); }
}
```

```java
@Before
public void setUp() { player = new Barbarian( name: "Alex", weapon, spell); }

@Test
public void getName() { assertEquals( expected: "Alex", player.getName()); }
```

Description here

This screenshot shows the use of inheritance in a program. The barbarian class extends the AbstractPlayer class so it has all the methods and properties of the AbstractPlayer class but also the barbarian class. The second screenshot shows the Abstract Player Class. The third screenshot shows a Barbarian Object and also the getName method being tested

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

```
public int dealersTurn() {
    int dealerHandValue = dealer.getHandValue();
    if (dealerHandValue <= 16) {
        this.deck.drawCardFromDeckToPlayer(dealer);
        this.dealersTurn();
    }
    return dealerHandValue;
}
```

```
private void declareWinner() {
    if (player.getHp() <= 0) {
        System.out.println("Enemy wins");
        System.out.println("Game over loser!");
    } else {
        System.out.println("Player wins");
        System.out.println();;
        fight();
    }
}
```

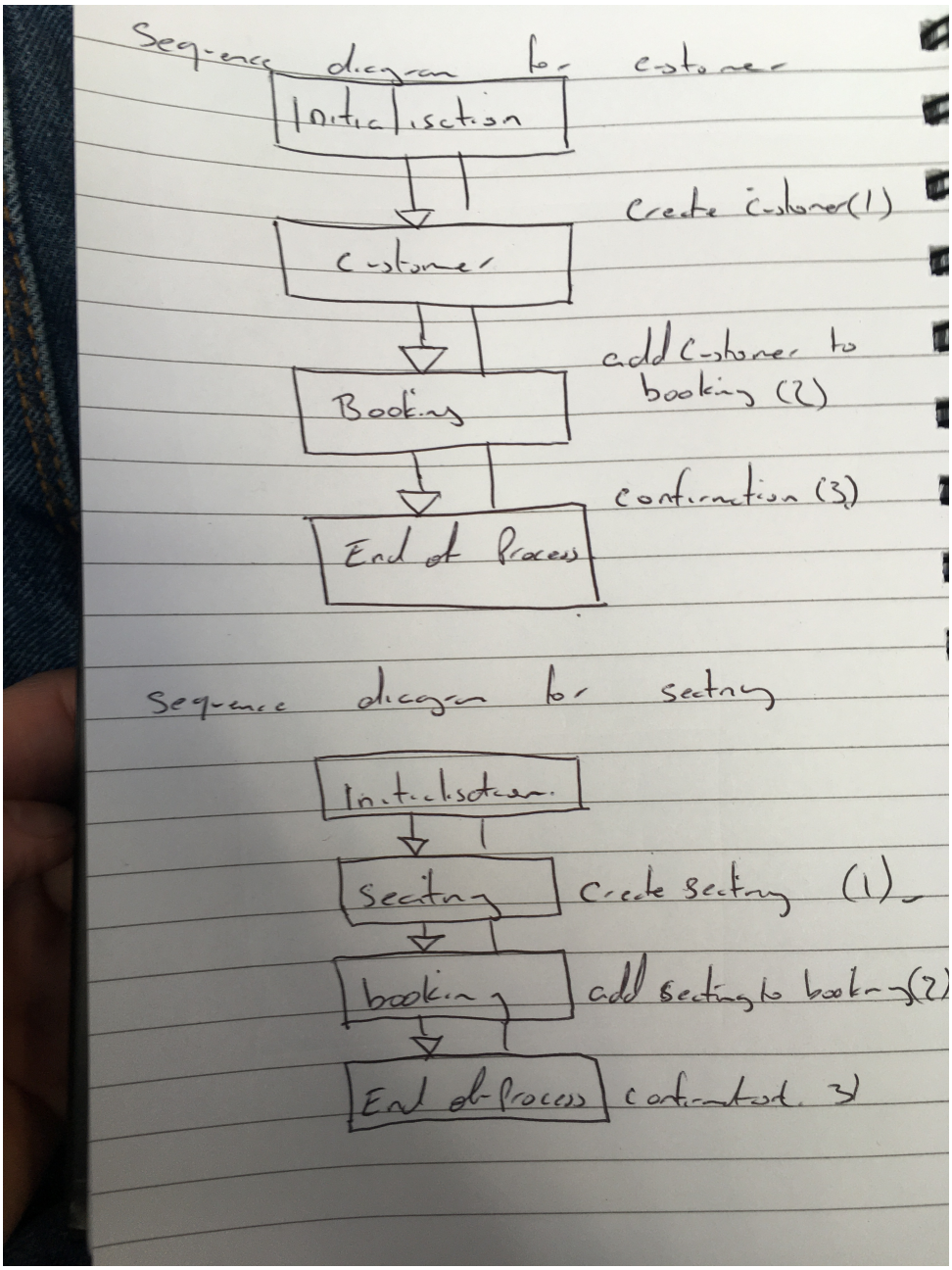
Description here

The first algorithm shows the logic for a dealer in a game of BlackJack, the dealer will draw a card from a deck if the dealers hand value is less than or equal to 16. The dealer will continue to draw cards from the deck until their hand value is higher than 16 at which point the loop will break and the algorithm will return the finalised hand value for the dealer.

The second algorithm shows the logic for deciding who the winner is in a game, it checks to see if the player has any hp left and if they do it returns that they have won and restarts the game else it returns that the enemy has won

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

Paste Screenshot here

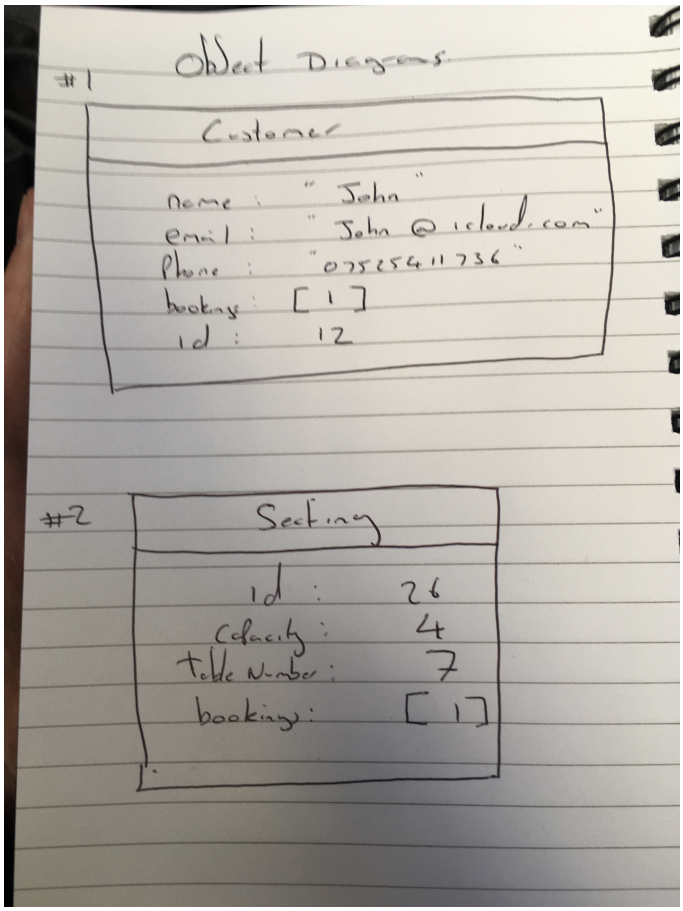


Description here

The screenshot above shows a system interaction diagram for a restaurant booking system, first of all a customer and seating object are created which are then added to a booking object. The booking object is then stored in a database which calls a confirmation to be sent back to the user.

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

Paste Screenshot here



Description here

This diagram shows an 2 objects. The Booking class has a one to many relationship with both customer and seating, the Seating and Customer class have a many to one relationship with Booking. The diagram shows all the properties each individual object will have and the type of each of the properties.

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

Paste Screenshot here

Who detected bug	Joe Cooper	Alex Woods	Jordan	Karolina
How was bug detected	Testing restful routes	Testing code	Testing user experience	Testing user experience
Severity	Breaks application	Tests fail	Minor	Severe
Status	Fixed	Fixed	fixed	To be fixed
Fixed by	Joe Cooper	Alex Woods	Jordan	Karolina
Notes	Internal server error has been resolved	Wrong array method being used	Submit form not resetting properly	Anchor links wrong way round

Description here

The above screenshots shows bugs and the bug tracking report that went along with it. The bugs were discovered during testing and found to be quite severe so we tracked them. This meant that if similar bugs were found in other areas of the application that we could quickly and easily fix them as they had already been encountered and documented.