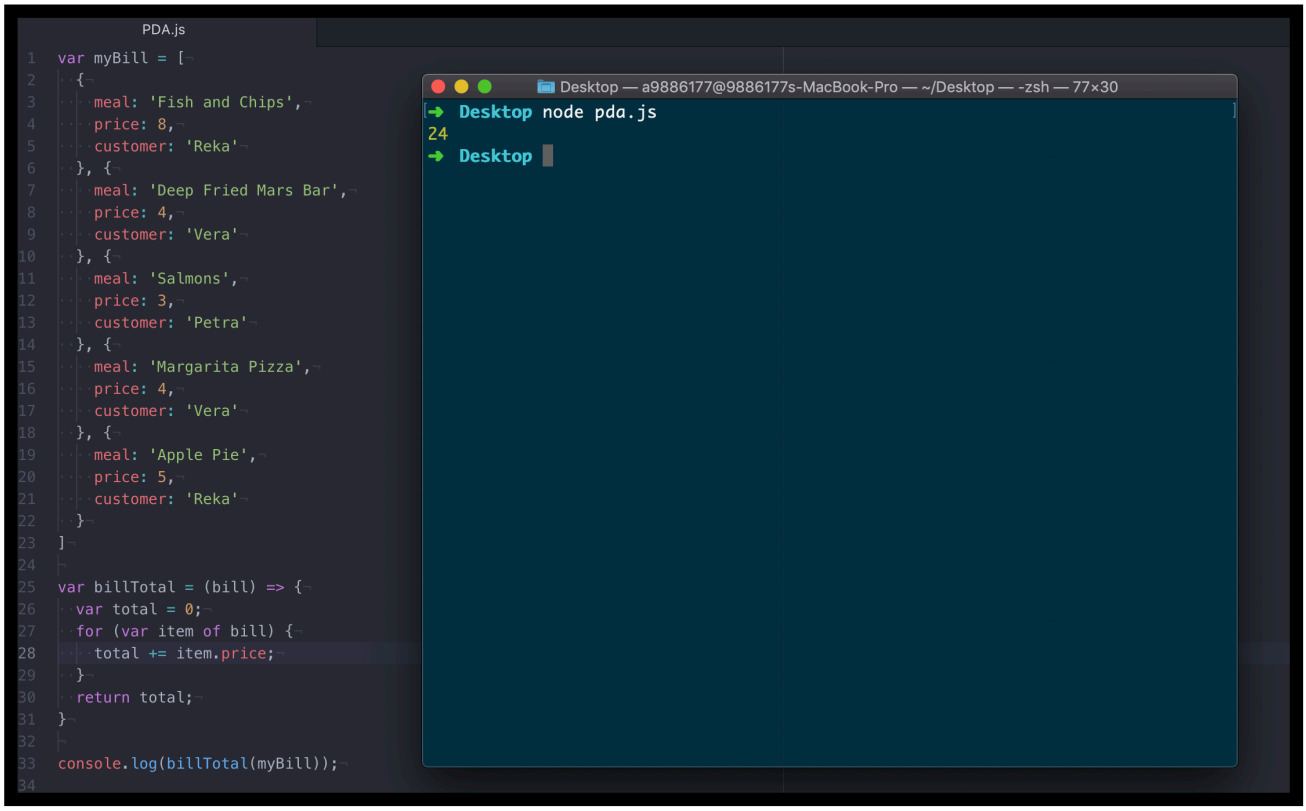
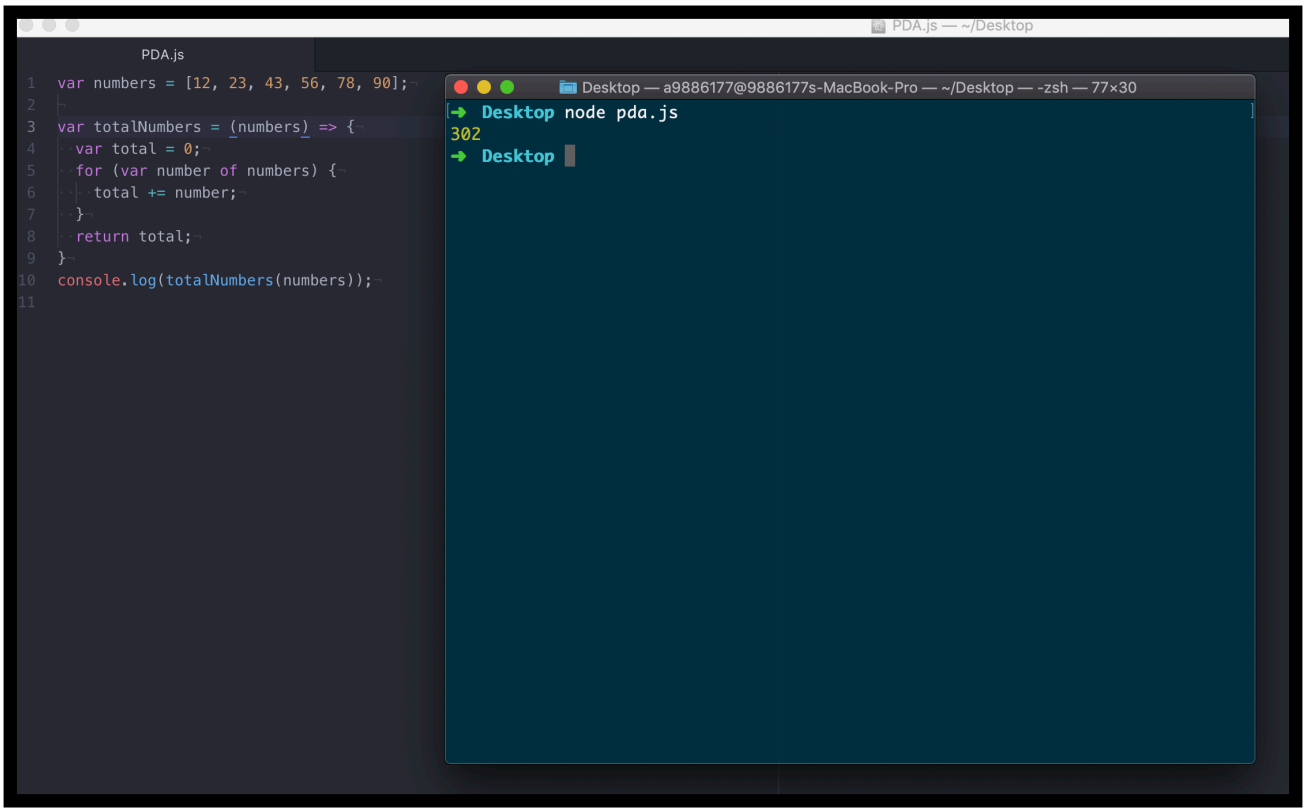
Evidence Gathering Document for SQA Level 8 Professional Developer Award - Reka Forgacs

Unit	Ref	Evidence
I&T	I.T.6	Demonstrate the use of an object literal in a program. Take screenshots of: *An object literal in a program *A function that uses the object *The result of the function running



The above code is representing the use of an object.
The function “billTotal” loops through the object of and calculates the sum by adding each meal price in the object.
The result of the function is shown in the Terminal ran by node.

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



The above code is representing the use of an array.
The function “totalNumbers” loops through the array of “numbers” and calculates the sum of the array.
The result of the function is shown in the Terminal ran by node.

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

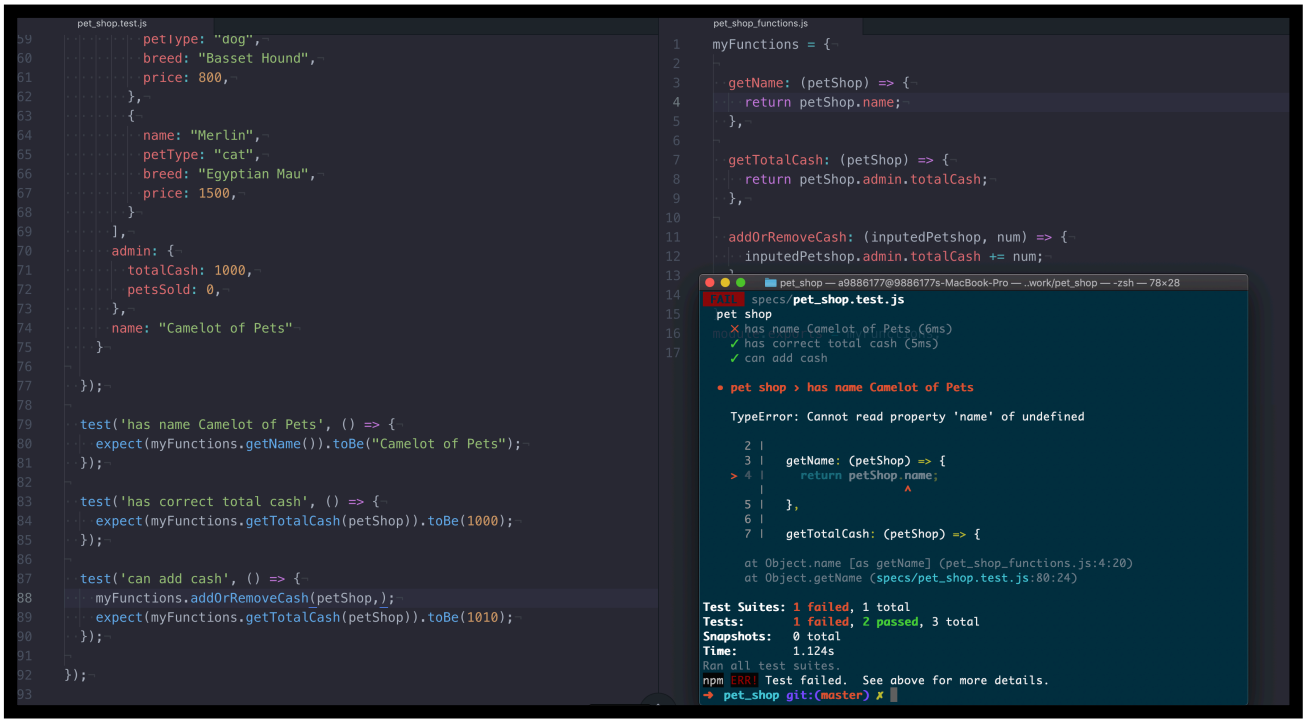
pet_shop.test.js

```
const myFunctions = require('../pet_shop_functions');

describe('pet shop', () => {

  let petShop;
  let customers;
  let newPet;

  beforeEach(() => {

    customers = [
      {
        name: "Alice",
        pets: [],
        cash: 1000
      },
      {
        name: "Bob",
        pets: [],
        cash: 50
      }
    ]

    newPet = {
      name: "Bors the Younger",
      petType: "cat",
      breed: "Cornish Rex",
      price: 100
    }

    petShop = {
      pets: [
        {
          name: "Sir Percy",
          petType: "cat",
```

pet_shop_functions.js

```
myFunctions = {

  getName: (petShop) => {
    return petShop.name;
  },

  getTotalCash: (petShop) => {
    return petShop.admin.totalCash;
  },

  addOrRemoveCash: (inputedPetshop, num) => {
    inputedPetshop.admin.totalCash += num;
  },

}
module.exports = myFunctions;
```

The above code is an example of a testing of a code.
The NPM test below shows in the Terminal that the first test fails and it points to the function, highlighting an error.



pet_shop.test.js

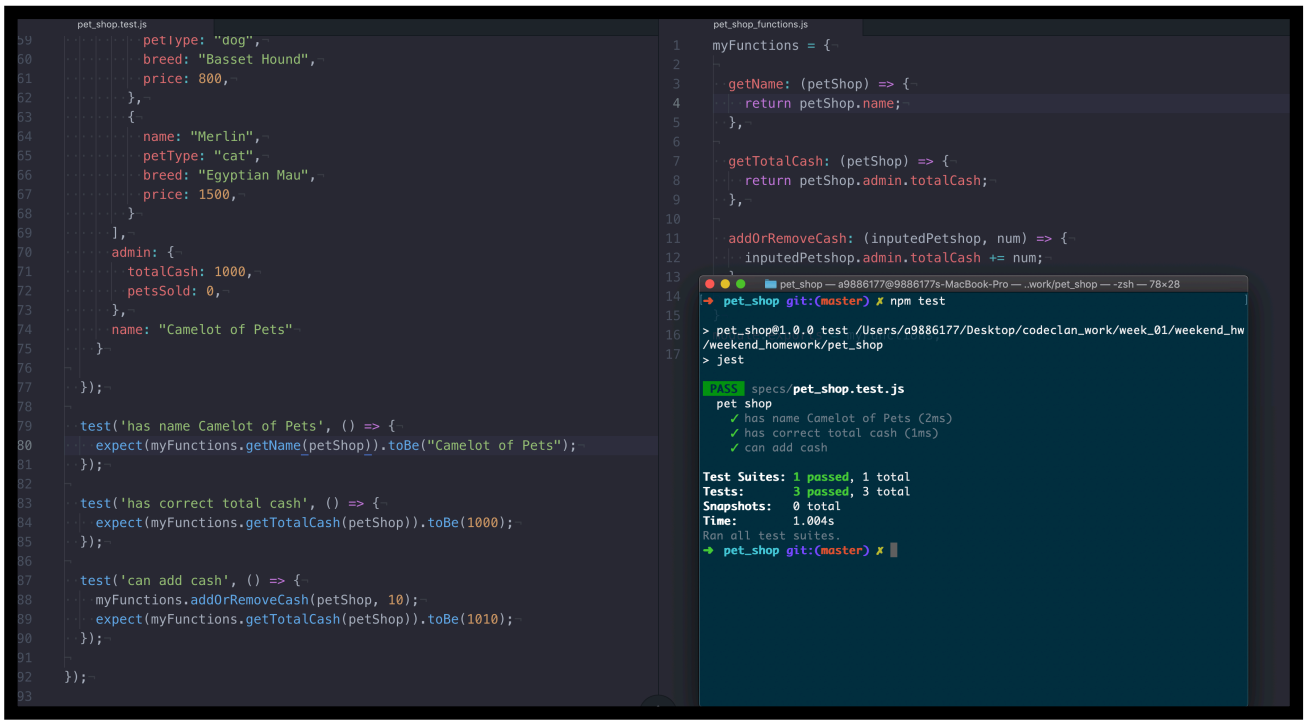
```
          petType: "dog",
          breed: "Basset Hound",
          price: 800,
        },
        {
          name: "Merlin",
          petType: "cat",
          breed: "Egyptian Mau",
          price: 1500,
        }
      ],
      admin: {
        totalCash: 1000,
        petsSold: 0,
      },
      name: "Camelot of Pets"
    }

  });

  test('has name Camelot of Pets', () => {
    expect(myFunctions.getName()).toBe("Camelot of Pets");
  });

  test('has correct total cash', () => {
    expect(myFunctions.getTotalCash(petShop)).toBe(1000);
  });

  test('can add cash', () => {
    myFunctions.addOrRemoveCash(petShop,);
    expect(myFunctions.getTotalCash(petShop)).toBe(1010);
  });

});
```

pet_shop_functions.js

```
myFunctions = {

  getName: (petShop) => {
    return petShop.name;
  },

  getTotalCash: (petShop) => {
    return petShop.admin.totalCash;
  },

  addOrRemoveCash: (inputedPetshop, num) => {
    inputedPetshop.admin.totalCash += num;
```

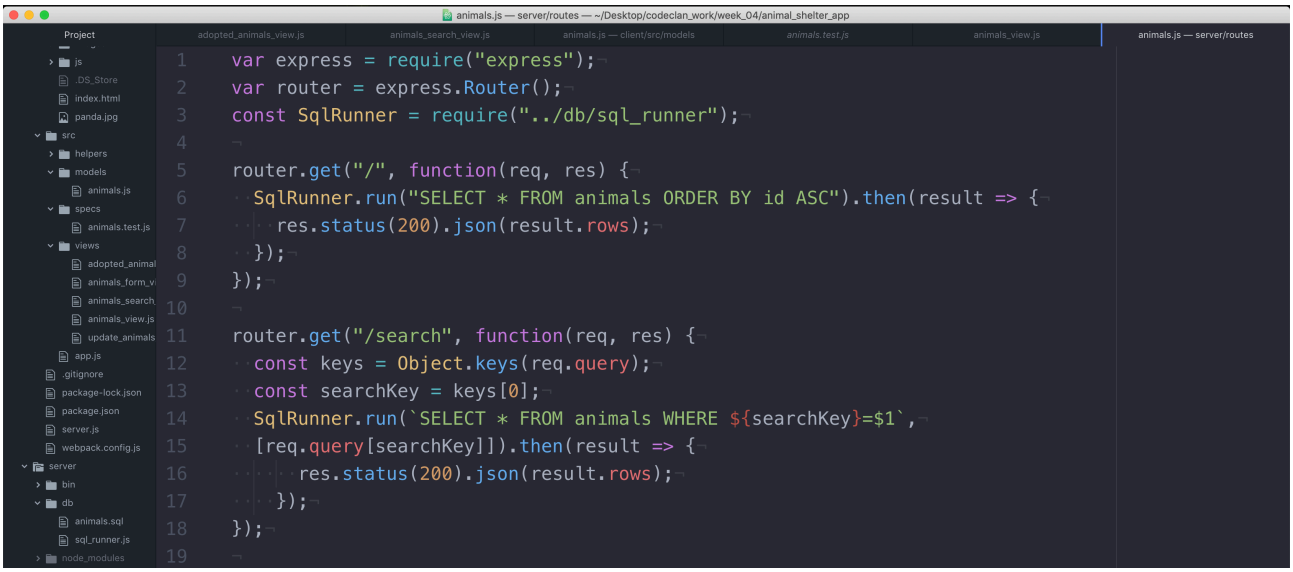
pet_shop — a9886177@9886177s-MacBook-Pro — ..work/pet_shop — -zsh — 78×28

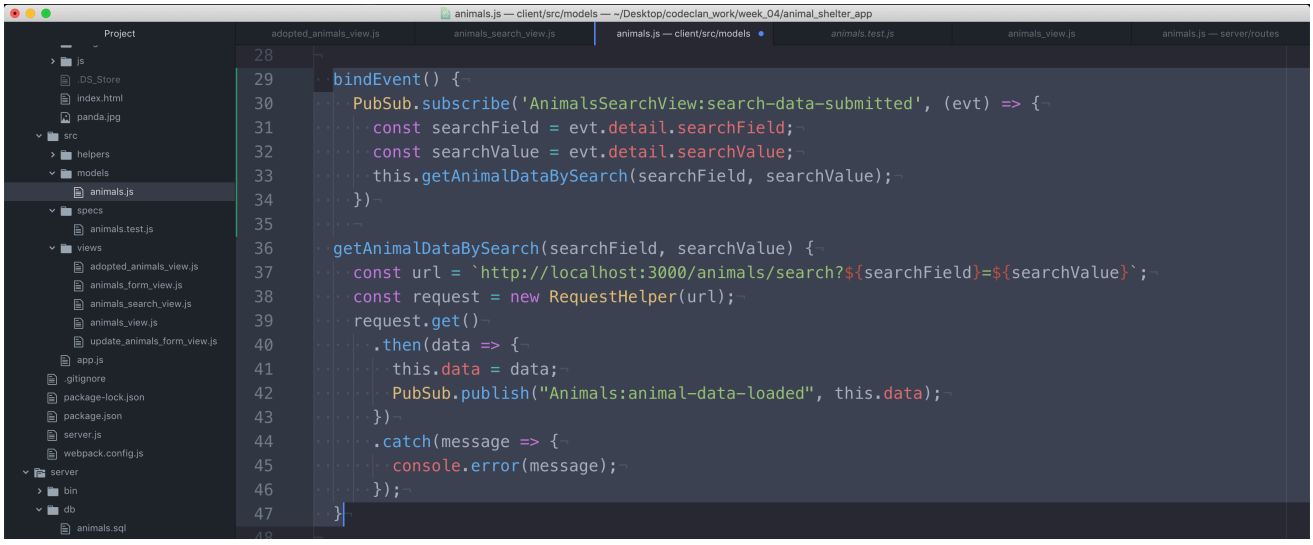
```
FAIL specs/pet_shop.test.js
 pet shop
   ✕ has name Camelot of Pets (6ms)
   ✓ has correct total cash (5ms)
   ✓ can add cash

 ● pet shop › has name Camelot of Pets

   TypeError: Cannot read property 'name' of undefined

     2 |
     3 |   getName: (petShop) => {
   > 4 |     return petShop.name;
       |                    ^
     5 |   },
     6 |
     7 |   getTotalCash: (petShop) => {

     at Object.name [as getName] (pet_shop_functions.js:4:20)
     at Object.getName (specs/pet_shop.test.js:80:24)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        1.124s
Ran all test suites.
npm ERR! Test failed.  See above for more details.
➜  pet_shop git:(master) ✗
```

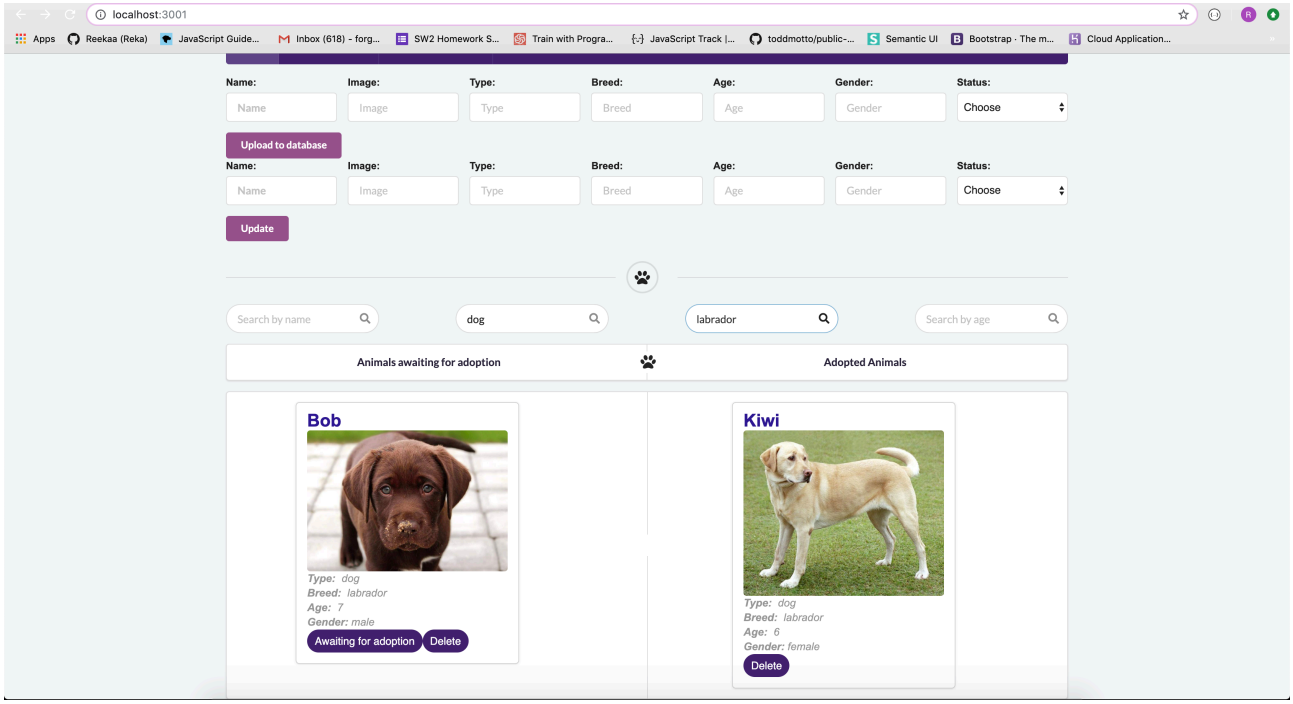


The above NPM test shows the tests passed after I've corrected the error in the test code and now calling the correct function.

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







I took this example of searching data from my own project, which is an animal shelter app. The function get AnimalDataBySearch( ) finds all animals and their details with the search id that is passed into the function. The above SQL searches a database called animals, which has a table of Animals, and matches the query parameter(s) to the table. The function gets the returned data from the database into an object of animals and display it on the application page.

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

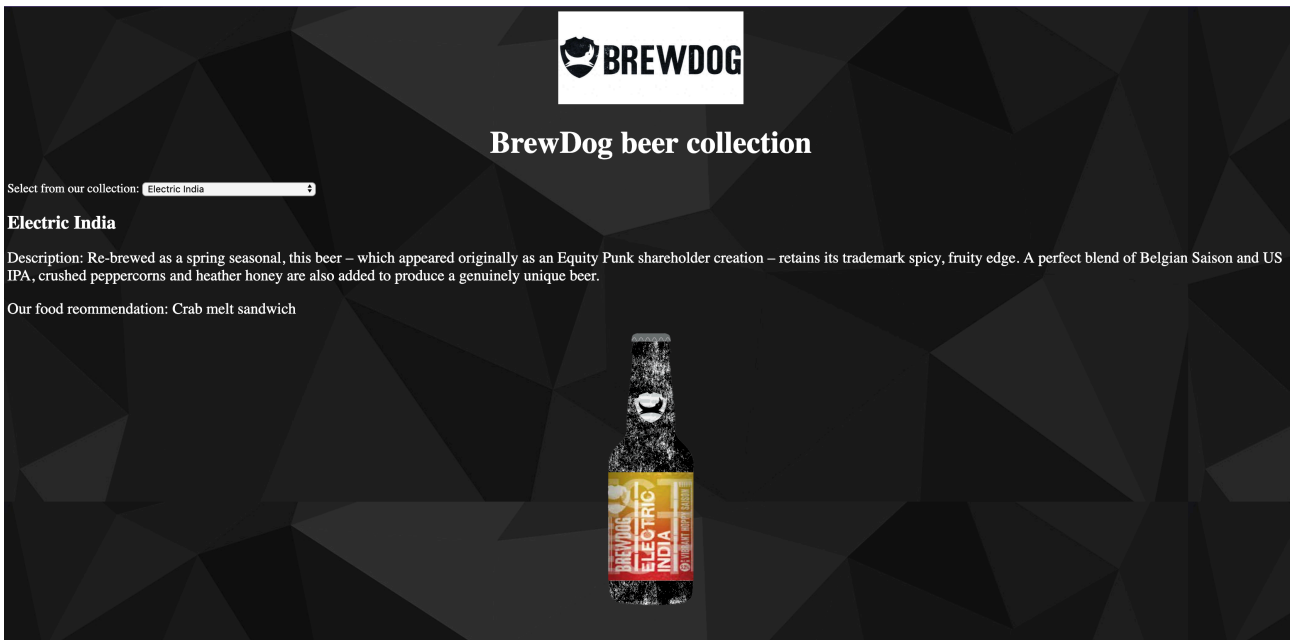
app.js | beers.js | index.html | beer_info.js | request_helper.js | main.css | beer_selection_view.js

```
const PubSub = require('../helpers/pub_sub.js');
const RequestHelper = require('../helpers/request_helper.js');

class Beers {

  constructor(){
    this.data = [];
  }

  bindEvent(){
    PubSub.subscribe('BeerSelection:change', (event) => {
      const selectedIndex = event.detail;
      const selectedBeer = this.data[selectedIndex];
    PubSub.publish('Beer:SelectedBeer-ready', selectedBeer)
    })
  }

  getData() {
    const url = `https://api.punkapi.com/v2/beers`
    const requestHelper = new RequestHelper(url);

    requestHelper.get()
      .then(data => {
        this.data = data;
        PubSub.publish('Beers:beer-data-loaded', this.data)
      })
      .catch(message => {
        console.error(message);
      })
  }
}
module.exports = Beers;
```

```
class RequestHelper {

  constructor(url){
    this.url = url;
  }

  get(){
    return fetch(this.url)
      .then(res => {
        return res.json()
      })
  }
}

module.exports = RequestHelper;

```

The code above shows the API used in my code to get the data.

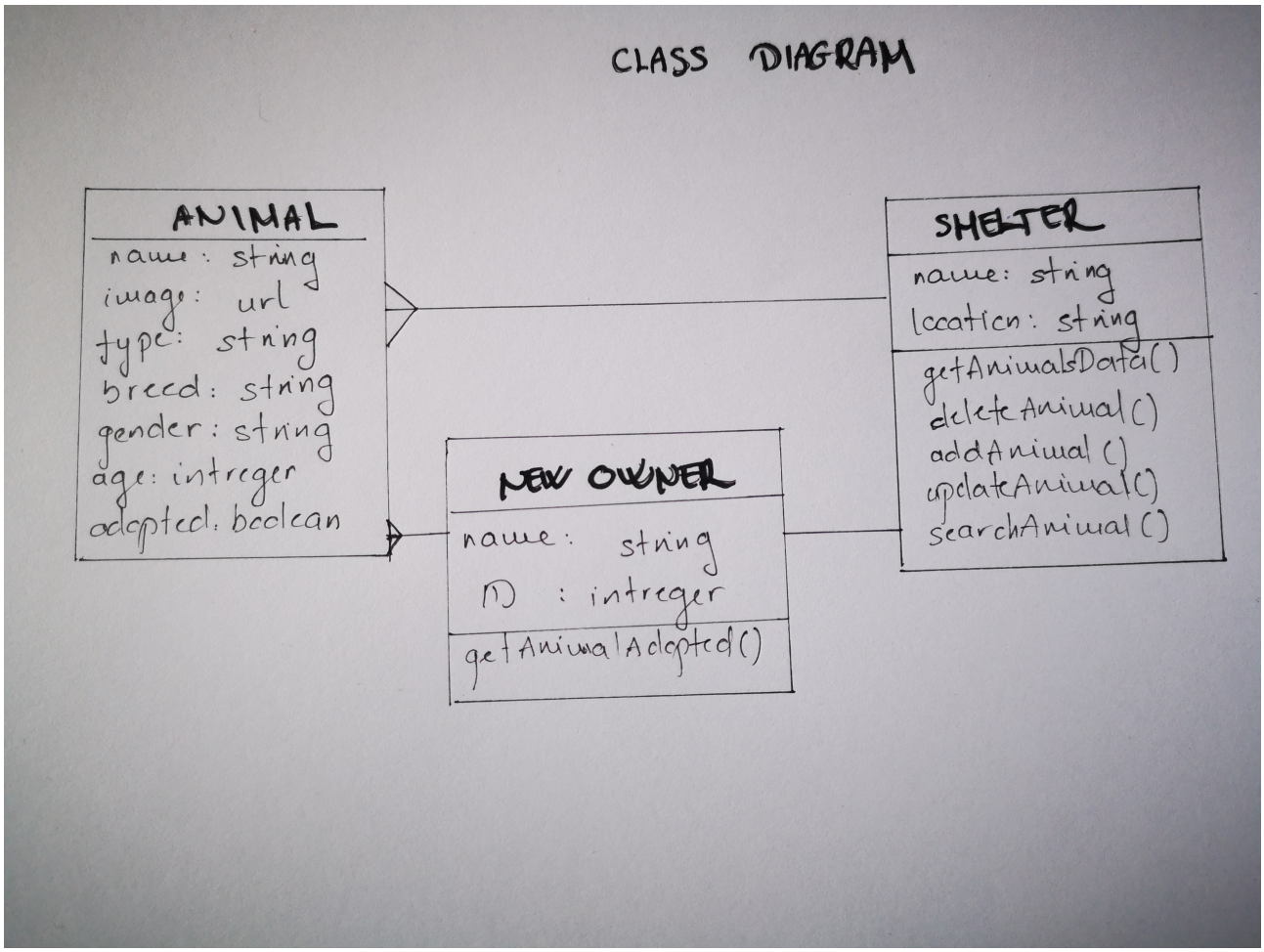
The application above that uses the API from my code.



API in use on the application.

I've used an API from BrewDog open source https://punkapi.com/documentation/v2. The API used in my code give the date for my application. On the first screenshot you can see how I called the date using the API. The second screenshot shows the basic application. The third screenshot shows the data I've requested from the API and it displays after user selected an option from the dropdown.

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



Description:

The above is a class diagram from my animal shelter application. I have used 3 classes in the application.

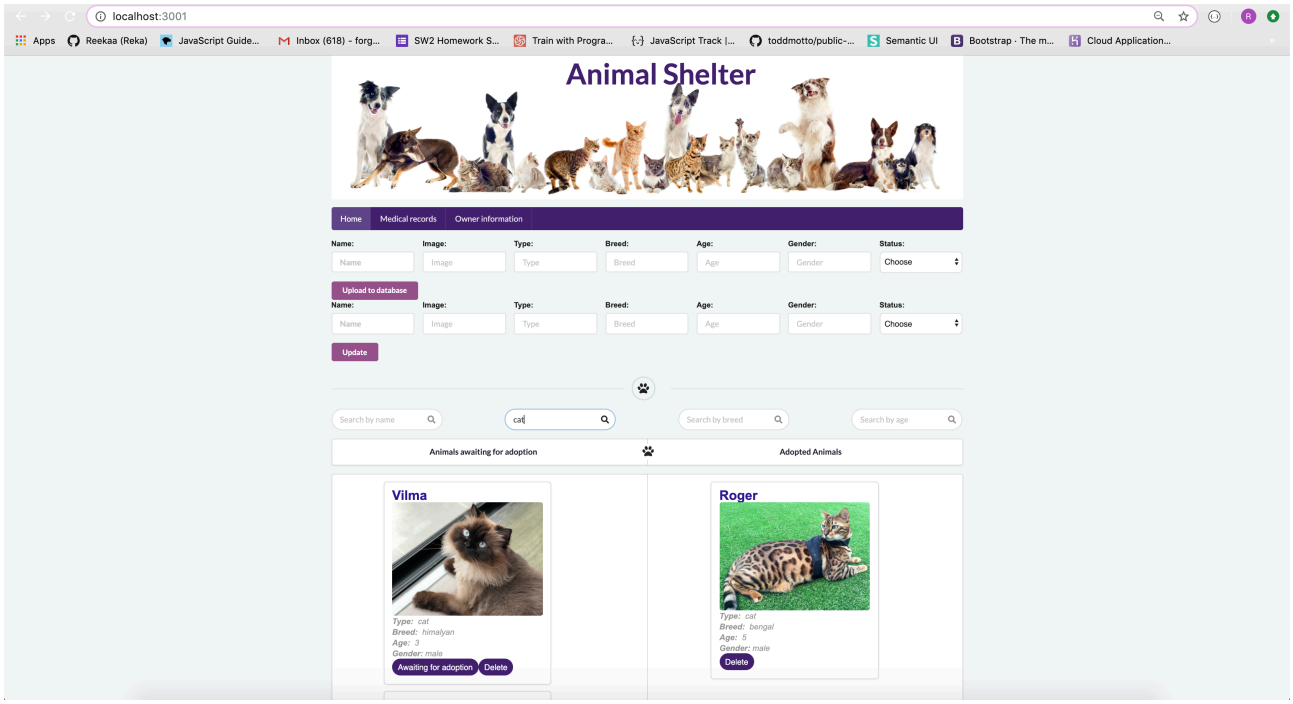
The relationships are between the classes:

Shelter to Animal - 1 to many

Shelter to New Owner - 1 to 1

New Owner to Animal - 1 to many

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

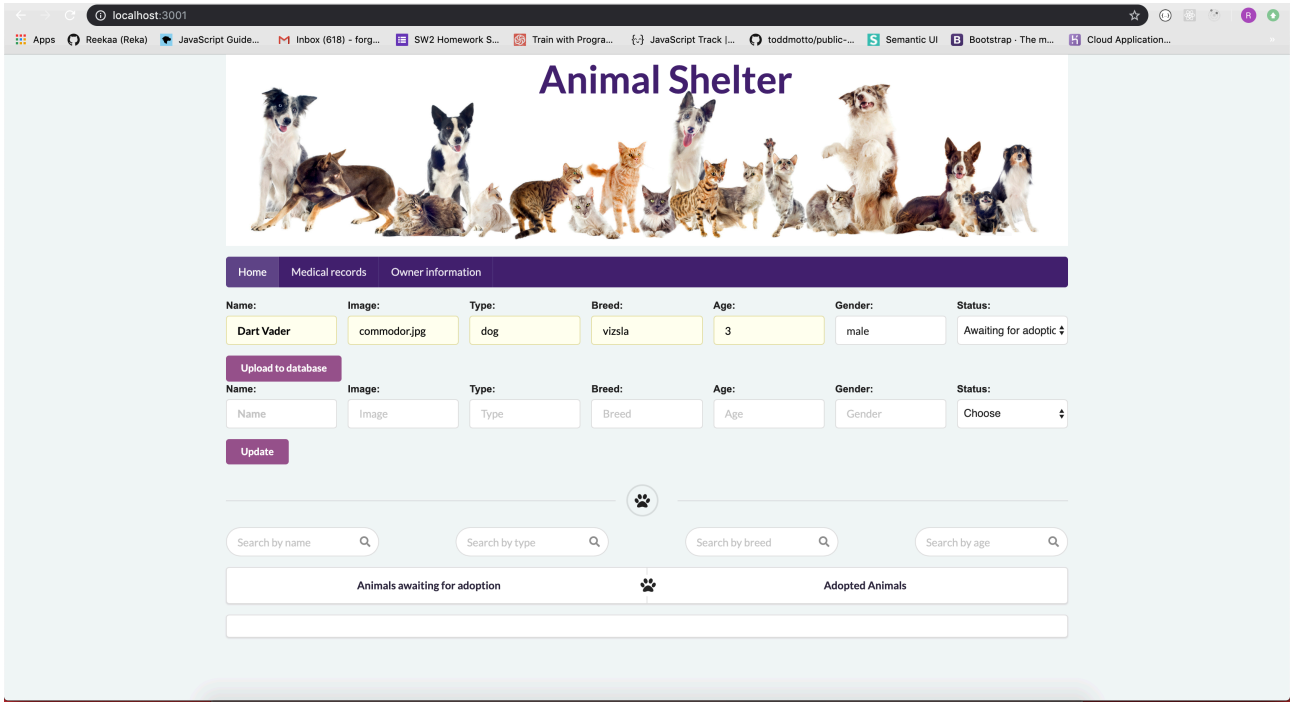


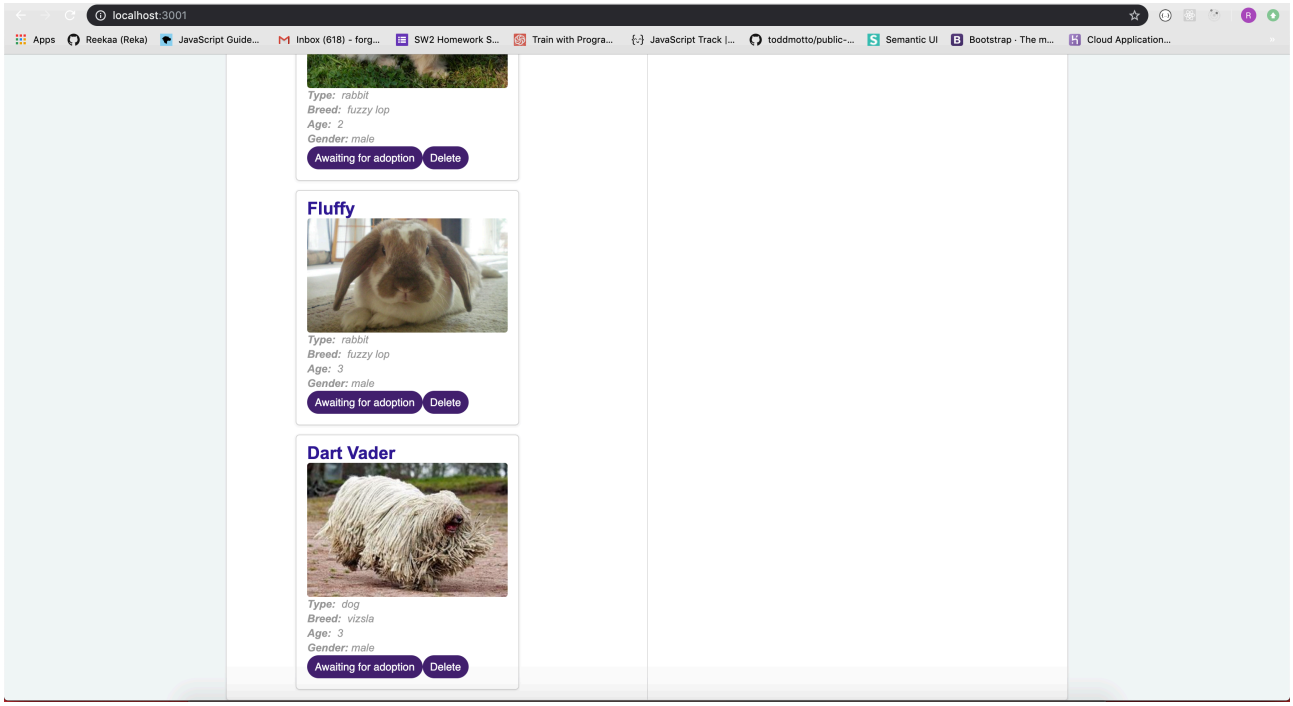
https://github.com/Reekaa/animal_shelter_app

Description:

This is my project that I worked on at week5. It is an application for an animal shelter to make easier for them to handle their database. I can update the database with new animals and update existing animal's details. Animals can be marked as adopted and deleted from the database. It was build in JavaScript and used SQL to build and handle the backend database.

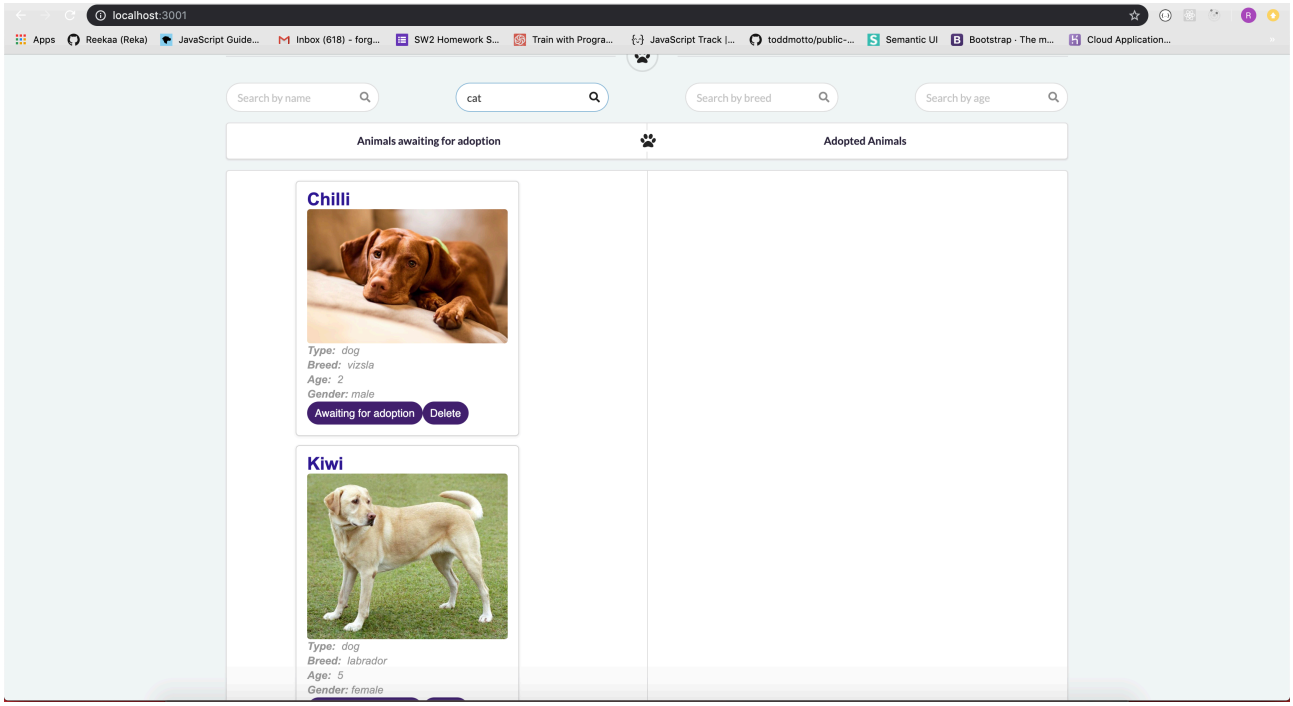
Unit	Ref	Evidence
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way

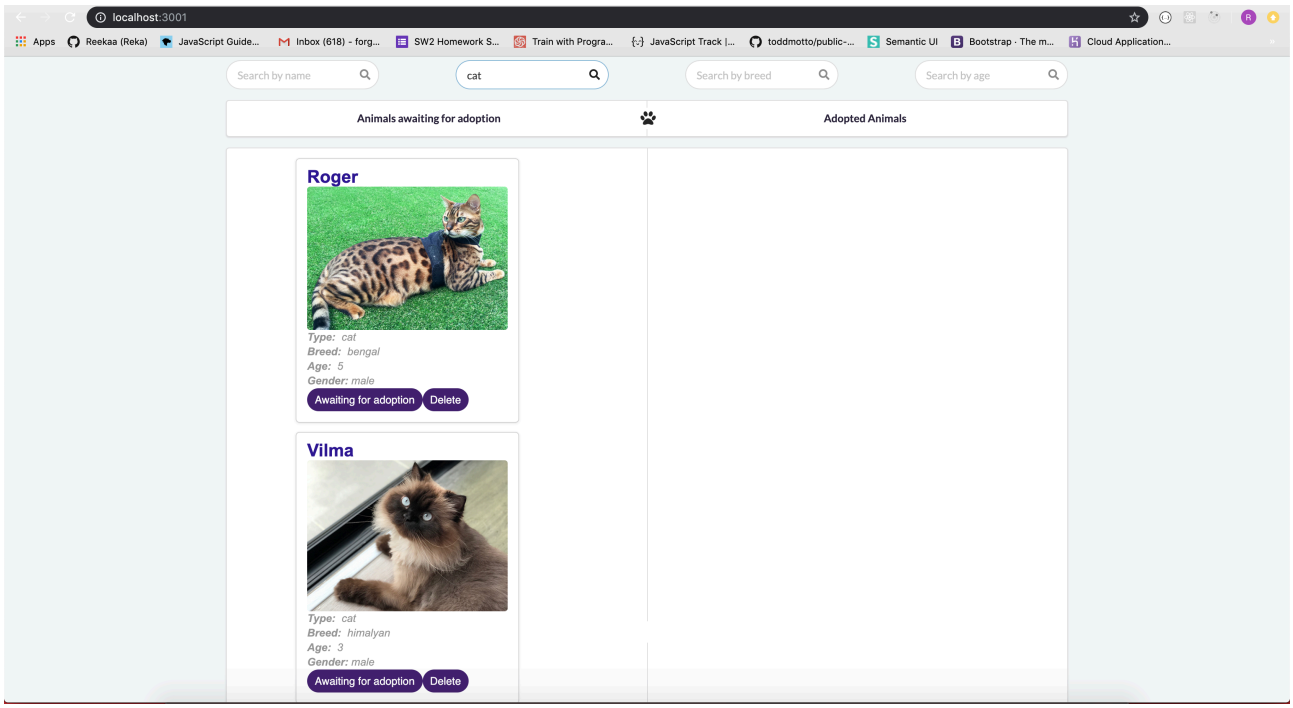




The above screenshots show the Animal Shelter App from my solo project.
On the first screenshot the user having inputted details of a new animal that they want to add to the database.
The second screenshot shows that the details have been processed and the page shows the animal details that have been submitted. These details now are in the database.

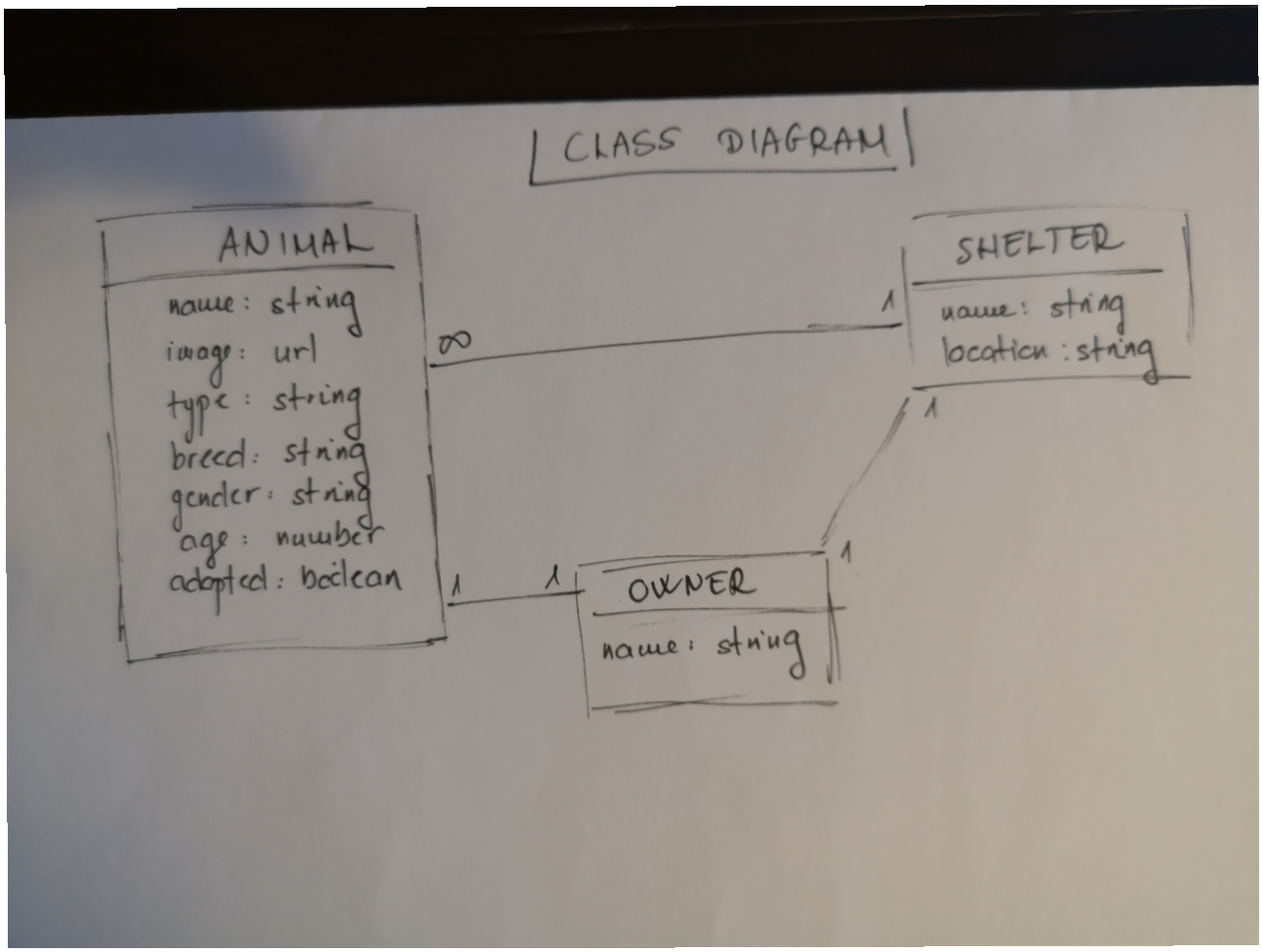
Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program





The above screenshots demonstrate my application showing the feedback that the user requested in the search bar. The first screenshot shows the dogs in the database and the user input “cat” into the searchbar. The second screenshot shows the return to the user after the search has been performed. The user can now see the cats in the database.

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



User needs

As a...	I want to...	So that...
administrator of the animal shelter	be able to see all animal data ~~in the~~ from the database	I can keep track of the animals.
— " —	be able to upload/add new animals	I can keep the database up to date.
— " —	be able to update the animal ~~data~~ details.	— " —
— " —	be able to delete animals from the database.	— " —
— " —	be able to search for animals by different options.	I can find the animal I look for quickly.
— " —	be able to mark an animal as adopted.	~~the~~ I can see which animal is adopted / which is available for adoption.
— " —	be able to see medical records for animals.	I can see which animal needs vet care.

User journey

Your name: shelter administrater Date: 26/04/2019

1 (Start)
User action: Open app to upload new animal details / filling in form.
System response: ~~The~~ App load up with all animal details. Gets the inputed data from form and update database.

2
User action: Misstyped animal name - click on name and update name to the correct ~~the~~ one.
System response: App loads up animal details after name has been clicked. After change from is submitted database updated.

3
User action: Search for all dogs data in the shelter. Type "dog" to searchbar.
System response: ~~Get~~ Get data from searchbar and filter database to provide list of dogs by ID.

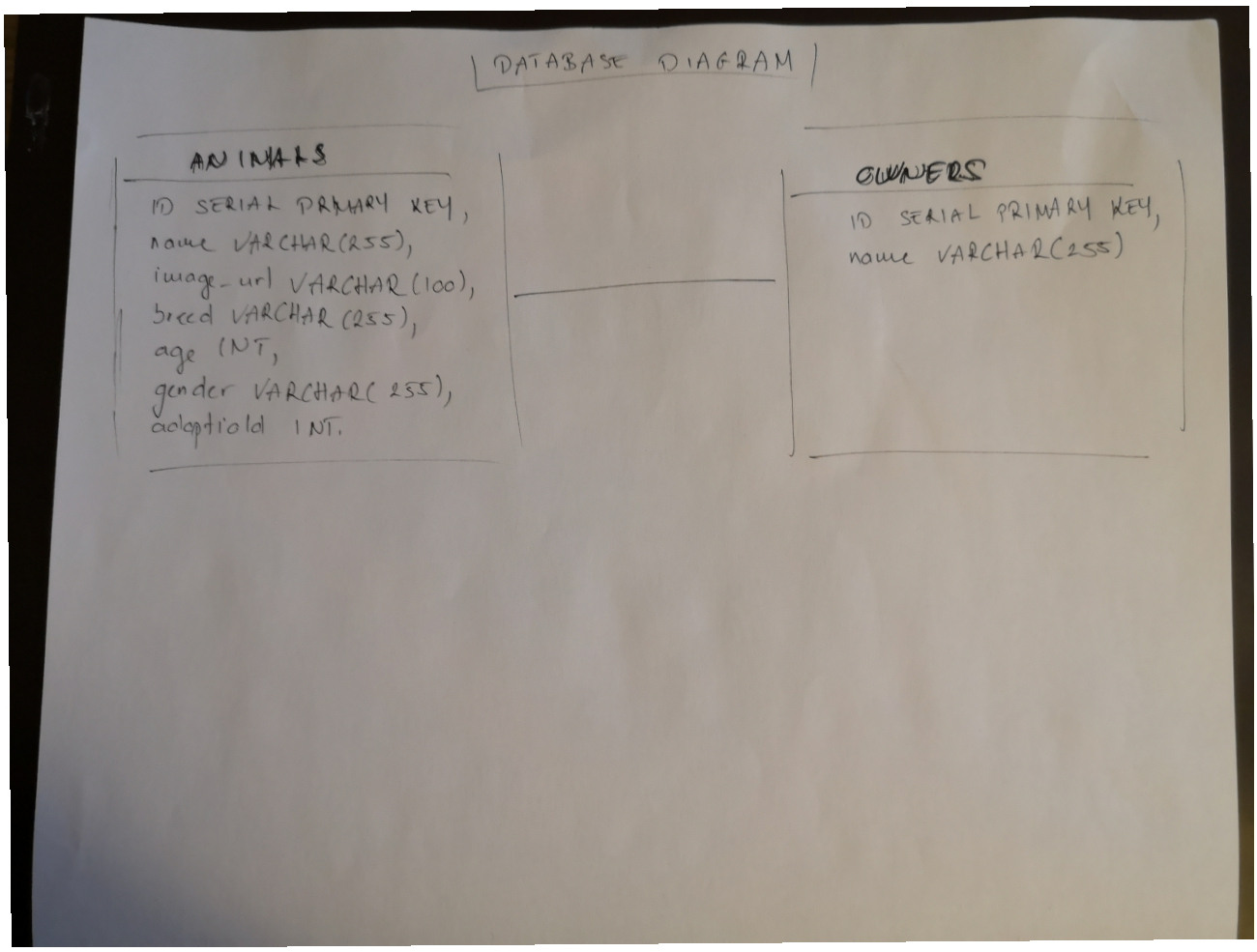
4
User action: Delete animal from application. by click on delete button.
System response: Delete button gets animal data by it's ID and delete animal from database by ID.

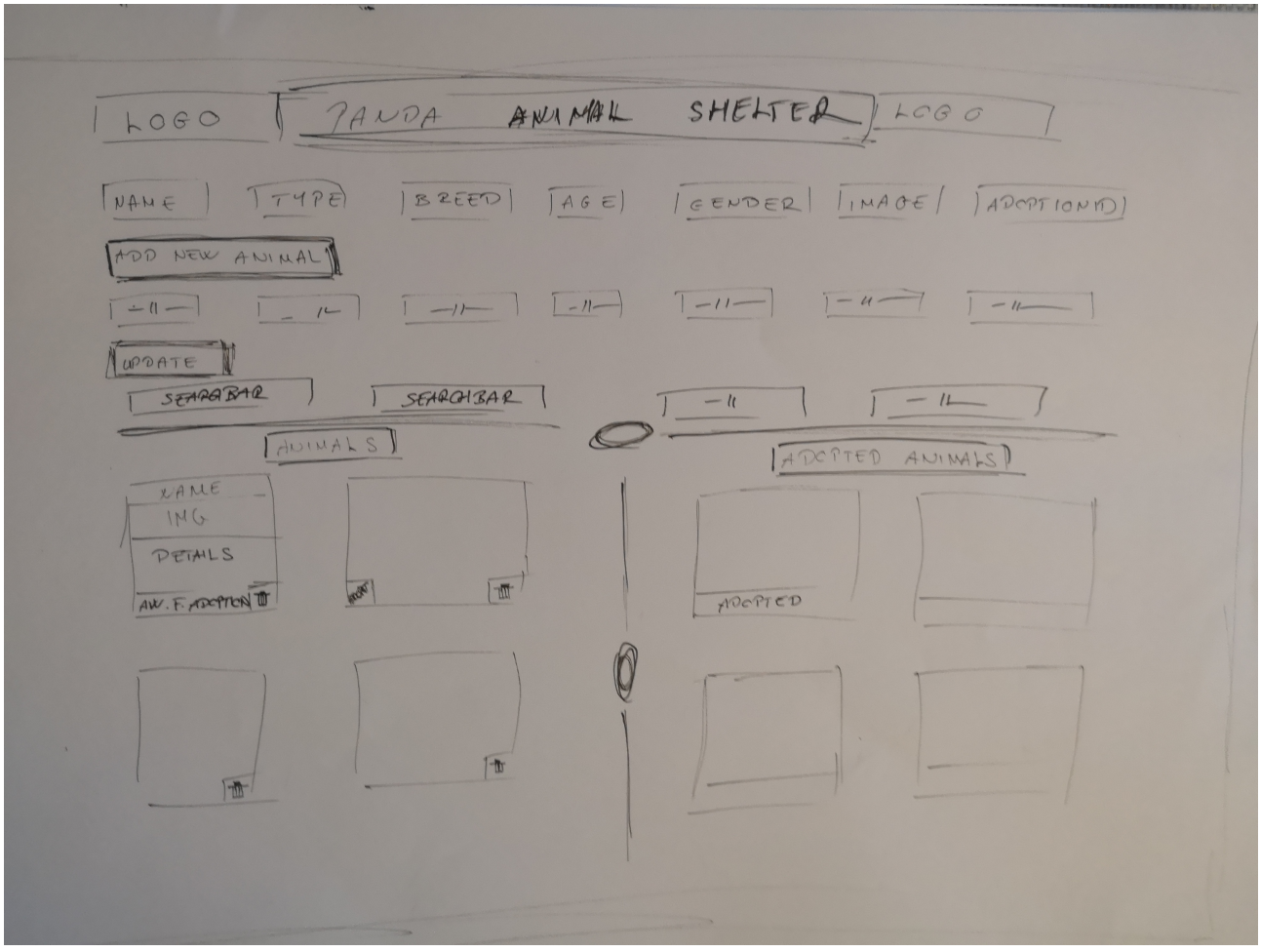
5
User action: Click on adoption button to change it to adopted and more animal details.
System response: Button click gets animal data by ID and changes boolean set to false to true and display "Adopted" and moves animal to new column.

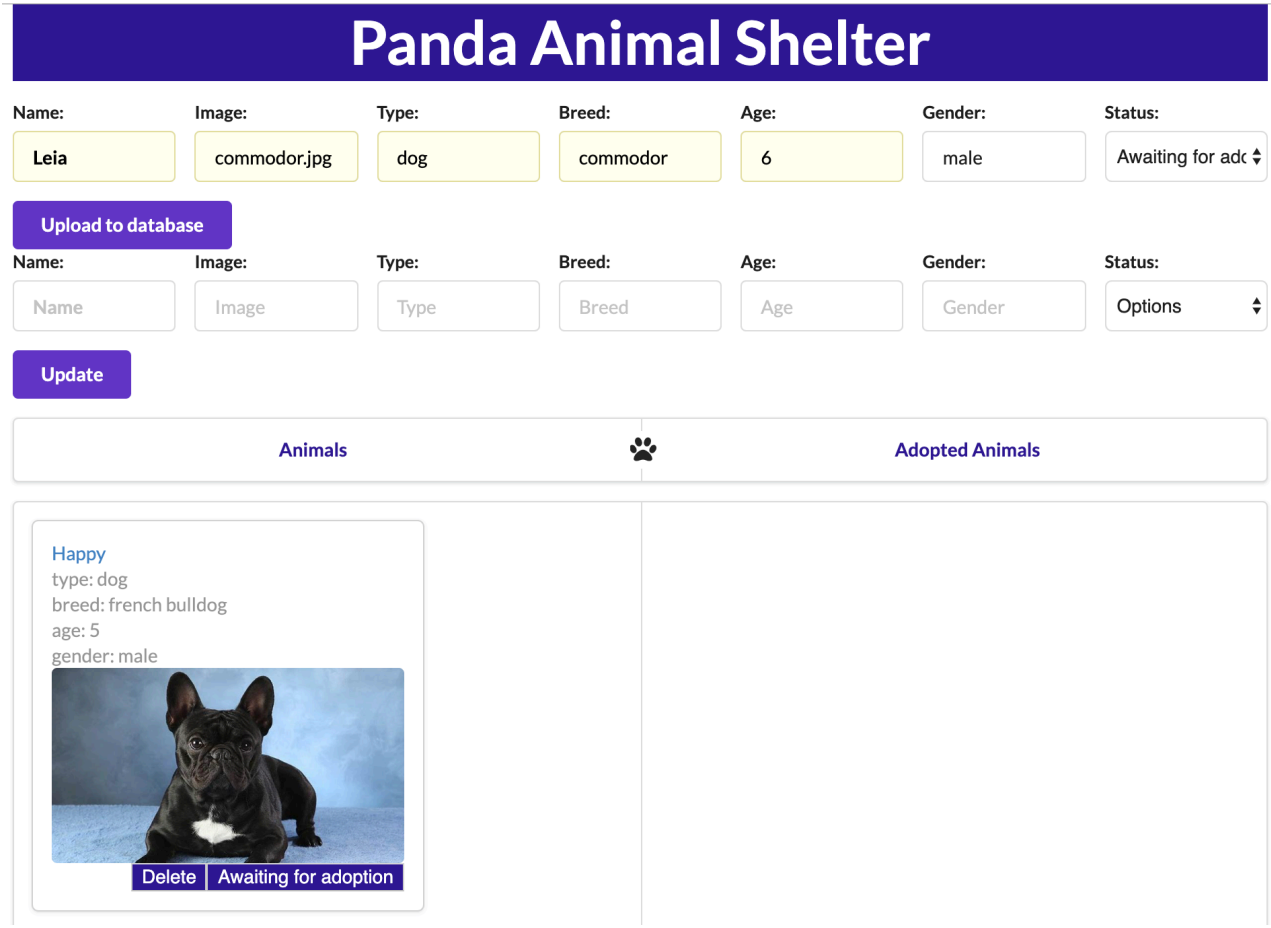
6
User action: Click medical data button to see medical history of the animal.
System response: Get data from database by id, display it on image (with CSS)

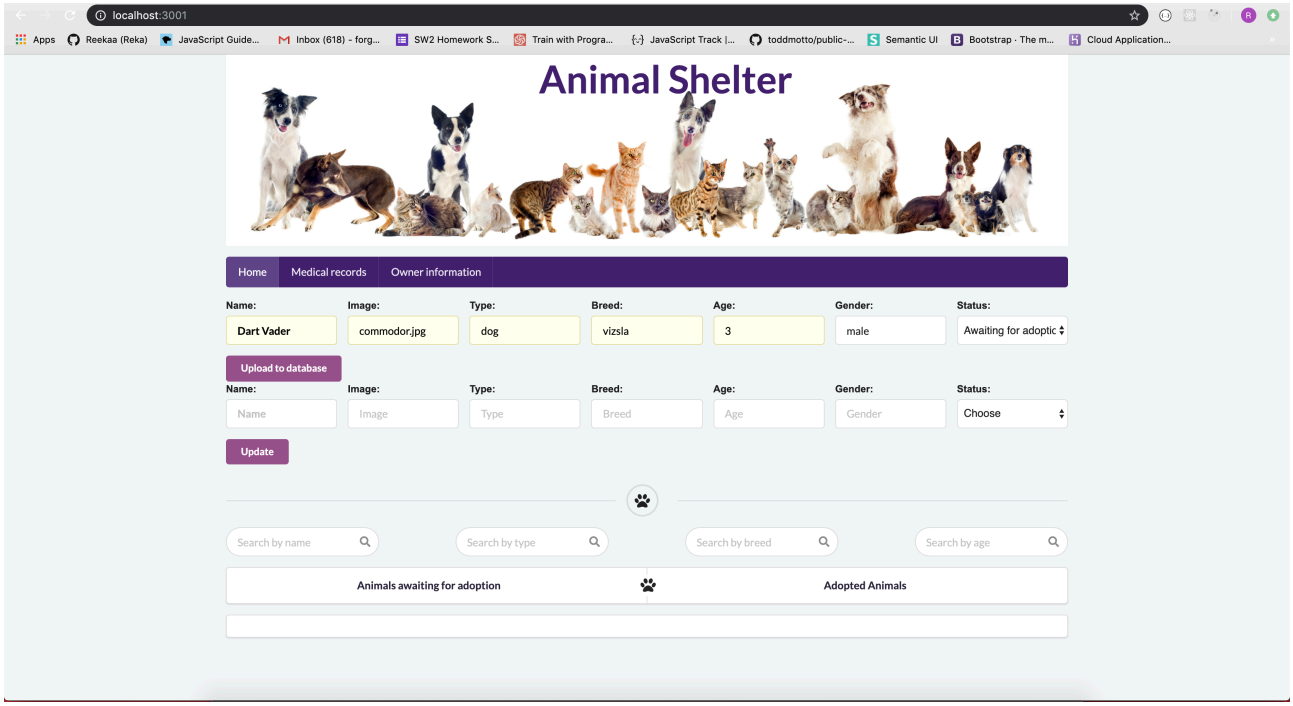
7
User action:
System response:

8 (End)
User action:
System response:


DATABASE DIAGRAM
ANIMALS
ID SERIAL PRIMARY KEY,
name VARCHAR(255),
image_url VARCHAR(100),
breed VARCHAR(255),
age INT,
gender VARCHAR(255),
adoptiond INT.
OWNERS
ID SERIAL PRIMARY KEY,
name VARCHAR(255)


LOGO
PANDA ANIMAL SHELTER
LOGO
NAME
TYPE
BREED
AGE
GENDER
IMAGE
ADOPTIONID
ADD NEW ANIMAL
UPDATE
SEARCHBAR
SEARCHBAR
ANIMALS
ADOPTED ANIMALS
NAME
IMG
DETAILS
AN. F. ADOPTION
ADOPTED

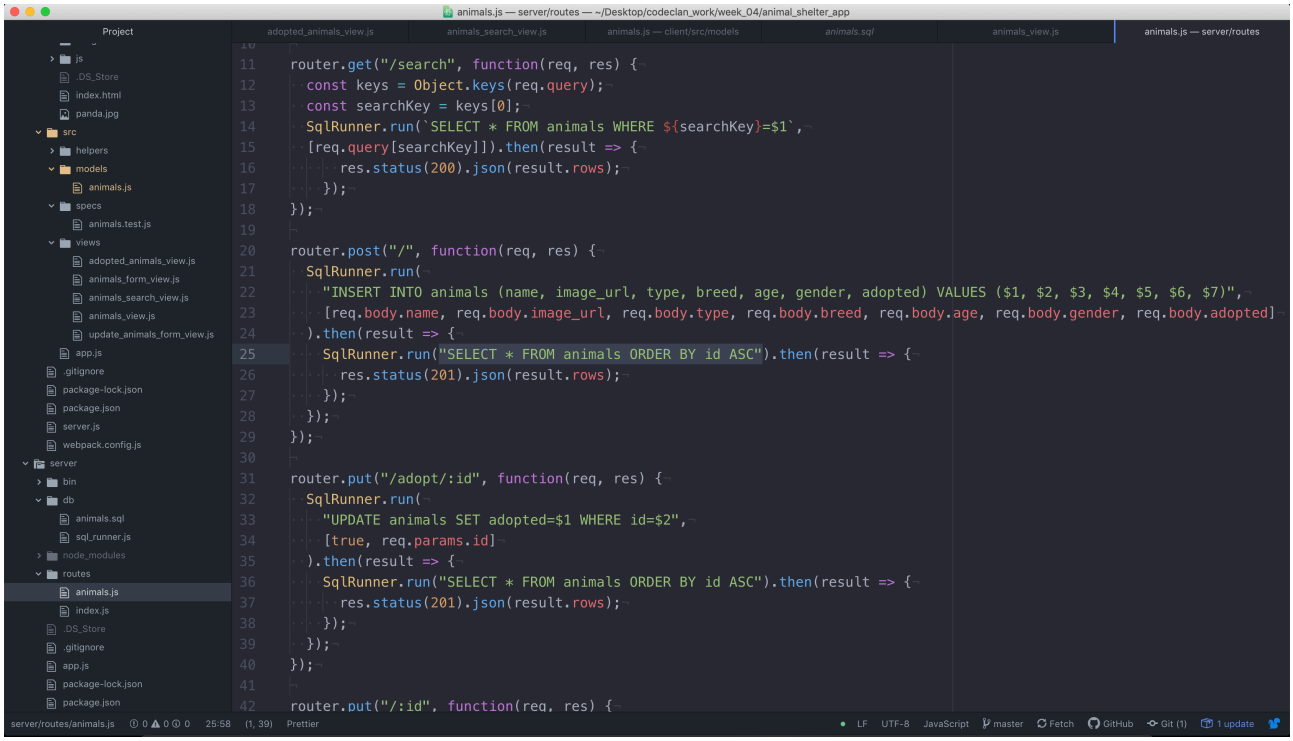


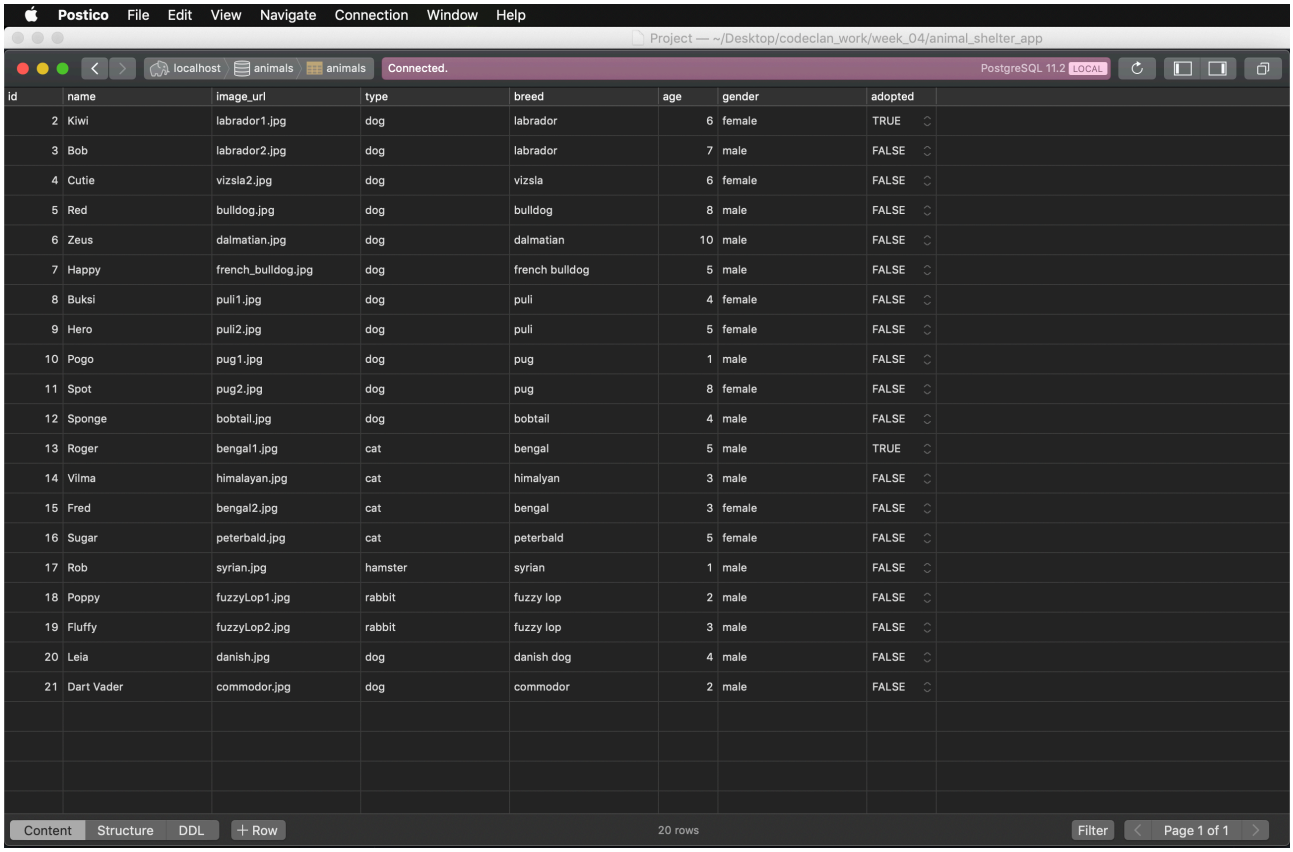


The above photos and screenshoots show my planning and different stages of the development of my solo project for the animal shelter application.
The first photos show the planning at the beginning of the project, the class and database diagram. I've also complete a user journey and user needs form to make the application as useful for the user as possible.

Once I had the diagrams and the user journey I've complete a user interface to see how the page should look like.
The last two screenshot show the different stages of the development. On the first screenshot the user can input data to the database and update the database. On the second screenshot I've implemented my extension, the search bars so the user can search in the database.

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





id	name	image_url	type	breed	age	gender	adopted
2	Kiwi	labrador1.jpg	dog	labrador	6	female	TRUE
3	Bob	labrador2.jpg	dog	labrador	7	male	FALSE
4	Cutie	vizsla2.jpg	dog	vizsla	6	female	FALSE
5	Red	bulldog.jpg	dog	bulldog	8	male	FALSE
6	Zeus	dalmatian.jpg	dog	dalmatian	10	male	FALSE
7	Happy	french_bulldog.jpg	dog	french bulldog	5	male	FALSE
8	Buksi	puli1.jpg	dog	puli	4	female	FALSE
9	Hero	puli2.jpg	dog	puli	5	female	FALSE
10	Pogo	pug1.jpg	dog	pug	1	male	FALSE
11	Spot	pug2.jpg	dog	pug	8	female	FALSE
12	Sponge	bobtail.jpg	dog	bobtail	4	male	FALSE
13	Roger	bengal1.jpg	cat	bengal	5	male	TRUE
14	Vilma	himalayan.jpg	cat	himalyan	3	male	FALSE
15	Fred	bengal2.jpg	cat	bengal	3	female	FALSE
16	Sugar	peterbald.jpg	cat	peterbald	5	female	FALSE
17	Rob	syrian.jpg	hamster	syrian	1	male	FALSE
18	Poppy	fuzzyLop1.jpg	rabbit	fuzzy lop	2	male	FALSE
19	Fluffy	fuzzyLop2.jpg	rabbit	fuzzy lop	3	male	FALSE
20	Leia	danish.jpg	dog	danish dog	4	male	FALSE
21	Dart Vader	commodor.jpg	dog	commodor	2	male	FALSE

The above example is to demonstrate how an Order By command within SQL can sort the result of a method. It shows the animals in the database sorted by their ID once the method has been run.

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

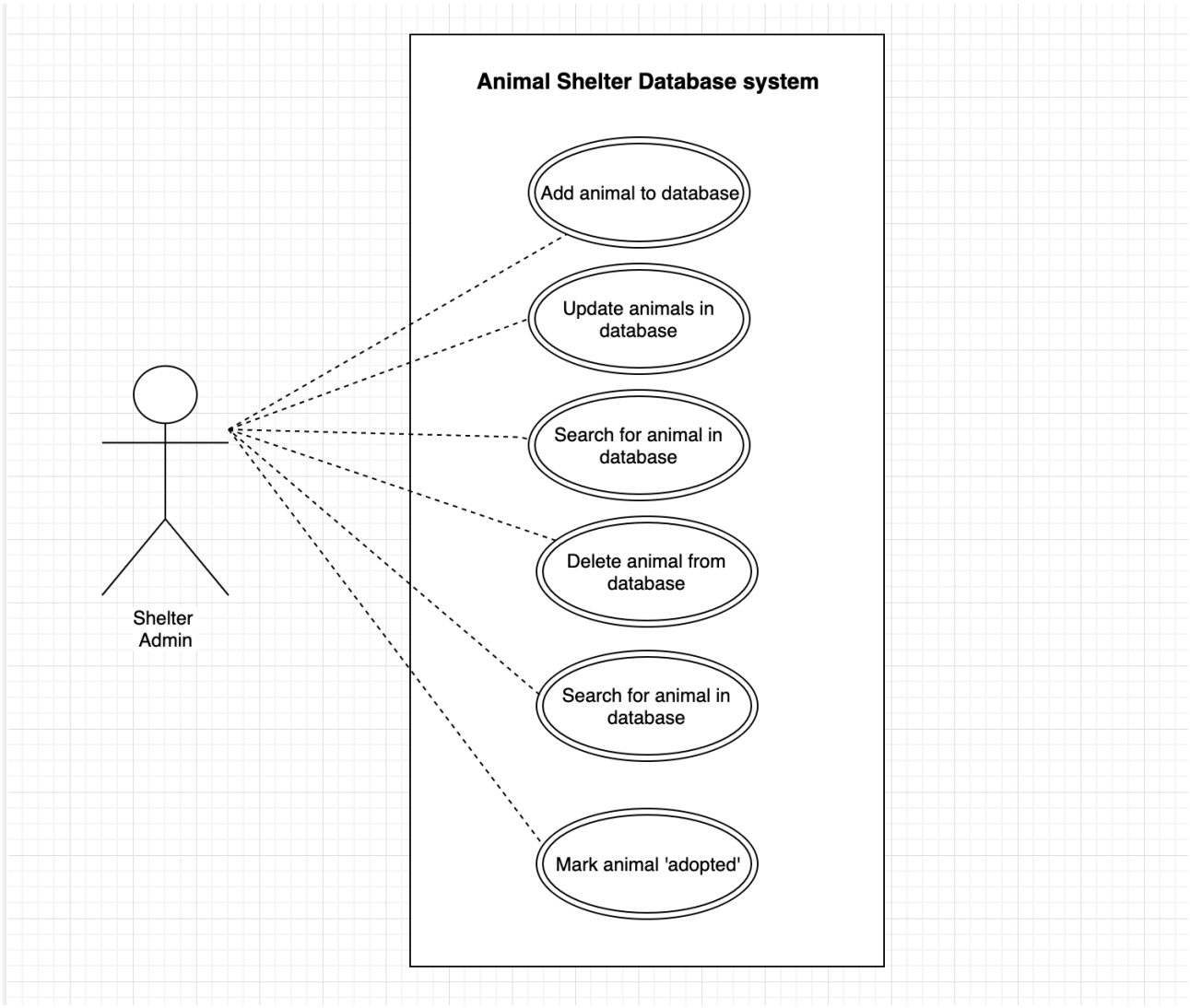
Encapsulation.js

```
class Train {

    constructor() {
        let miles = 0; // private instance data
        this.getMiles = function() {
            return miles;
        }; // function that has access to the private instance above
        this.setMiles = function(a) {
            miles += a;
        };
    }
}

var myTrain = new Train();
console.log(myTrain.getMiles()); // returns 0

myTrain.setMiles(20);
console.log(myTrain.getMiles()); // returns 20

console.log(myTrain.miles); // undefined

```

-

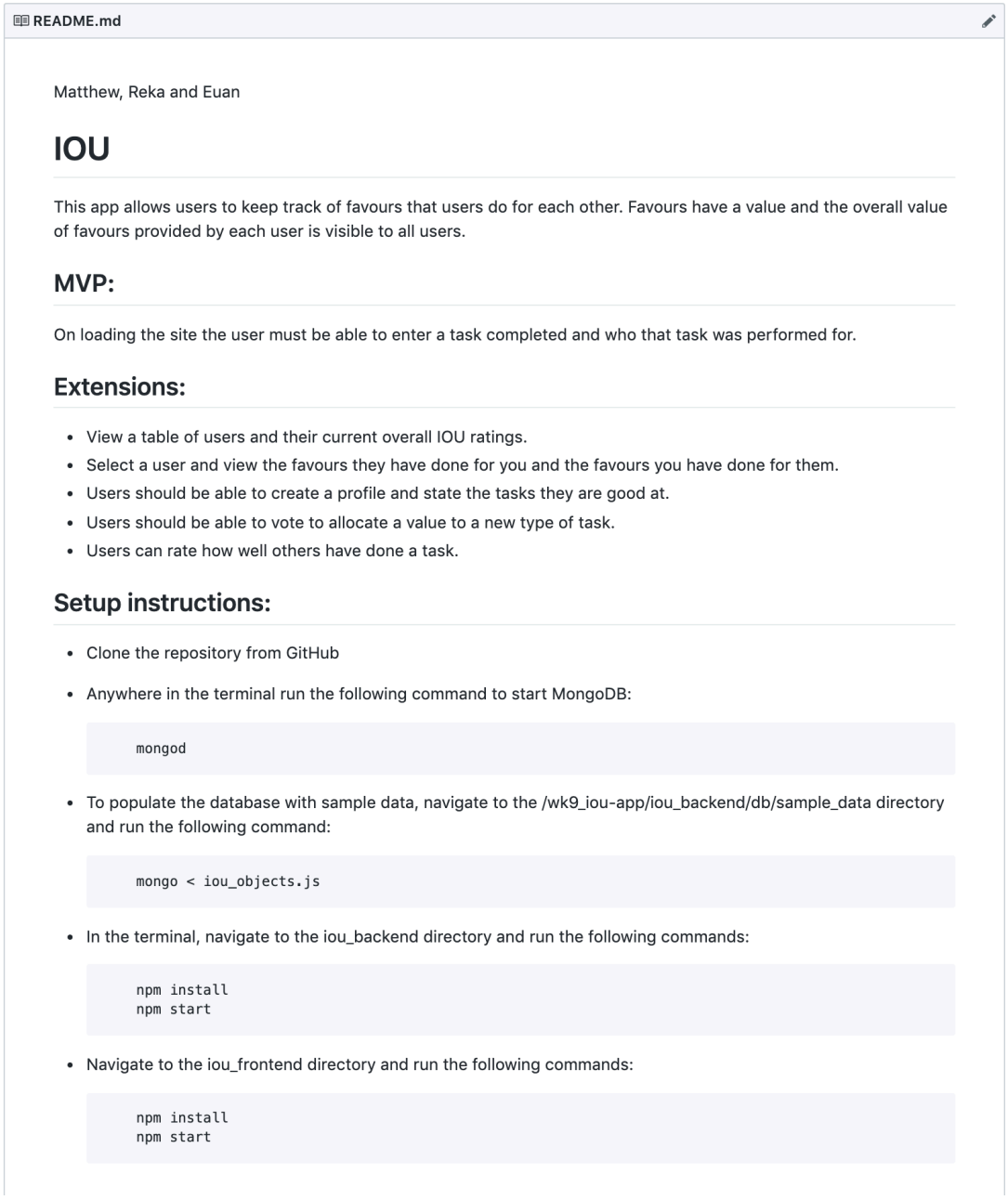
The above screenshot is an example for a use of Encapsulation. The variable ‘miles’ is not accessible directly. I’ve created a method setMiles() so that it can be accessed in other methods.

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



I have created the above use case diagram to visualise how my solo project application going to be used by an animal shelter administrator. The administrator can use the system to update the database, delete from it, use the search functions and mark animals as ‘adopted’.

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



README.md

Matthew, Reka and Euan

IOU

This app allows users to keep track of favours that users do for each other. Favours have a value and the overall value of favours provided by each user is visible to all users.

MVP:

On loading the site the user must be able to enter a task completed and who that task was performed for.

Extensions:

- View a table of users and their current overall IOU ratings.
- Select a user and view the favours they have done for you and the favours you have done for them.
- Users should be able to create a profile and state the tasks they are good at.
- Users should be able to vote to allocate a value to a new type of task.
- Users can rate how well others have done a task.

Setup instructions:

- Clone the repository from GitHub
- Anywhere in the terminal run the following command to start MongoDB:

```
mongod
```

- To populate the database with sample data, navigate to the /wk9_iou-app/iou_backend/db/sample_data directory and run the following command:

```
mongo < iou_objects.js
```

- In the terminal, navigate to the iou_backend directory and run the following commands:

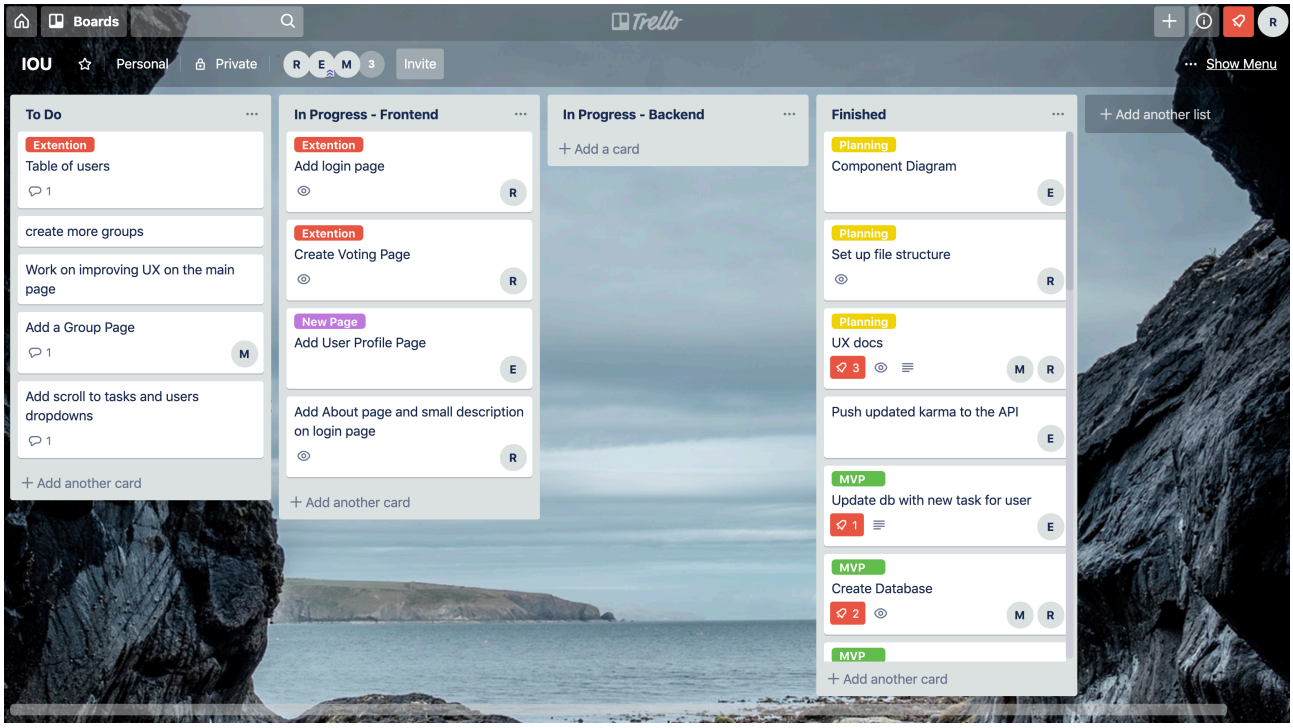
```
npm install
npm start
```

- Navigate to the iou_frontend directory and run the following commands:

```
npm install
npm start
```

The above screenshot is from our group project brief. We've created an app that allows the users to keep track of their favours that users do for each other.
We've defined our MVP and extension and also set up a simple setup instruction to help other people to run our app.

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



This screenshot show our Trello board that we used to plan and organise our workload throughout the group project.
We've had a stand up every day at 10am to discussed with each other what we have done so far and what we plan to achieve that day. We've shared the work load equally and allowed everyone to work an every different step in the process (planing, MVP, extensions).
The planning helped us to keep track of the progress of our application and we could decide on the next step together once a feature has been completed.

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

Animal Shelter

Home | Medical records | Owner information

Name:	Image:	Type:	Breed:	Age:	Gender:	Status:
Dart Vader	commodor.jpg	dog	commodor	2	male	Awaiting for adoptio

Upload to database

Postico File Edit View Navigate Connection Window Help

Project — ~/Desktop/codeclan_work/week_04/ani

localhost | animals | animals | Connected.

id	name	image_url	type	breed	age	gender	adopted
2	Kiwi	labrador1.jpg	dog	labrador	6	female	TRUE
3	Bob	labrador2.jpg	dog	labrador	7	male	FALSE
4	Cutie	vizsla2.jpg	dog	vizsla	6	female	FALSE
5	Red	bulldog.jpg	dog	bulldog	8	male	FALSE
6	Zeus	dalmatian.jpg	dog	dalmatian	10	male	FALSE
7	Happy	french_bulldog.jpg	dog	french bulldog	5	male	FALSE
8	Buksi	puli1.jpg	dog	puli	4	female	FALSE
9	Hero	puli2.jpg	dog	puli	5	female	FALSE
10	Pogo	pug1.jpg	dog	pug	1	male	FALSE
11	Spot	pug2.jpg	dog	pug	8	female	FALSE
12	Sponge	bobtail.jpg	dog	bobtail	4	male	FALSE
13	Roger	bengal1.jpg	cat	bengal	5	male	TRUE
14	Vilma	himalayan.jpg	cat	himalyan	3	male	FALSE
15	Fred	bengal2.jpg	cat	bengal	3	female	FALSE
16	Sugar	peterbald.jpg	cat	peterbald	5	female	FALSE
17	Rob	syrian.jpg	hamster	syrian	1	male	FALSE
18	Poppy	fuzzyLop1.jpg	rabbit	fuzzy lop	2	male	FALSE
19	Fluffy	fuzzyLop2.jpg	rabbit	fuzzy lop	3	male	FALSE
20	Leia	danish.jpg	dog	danish dog	4	male	FALSE
21	Dart Vader	commodor.jpg	dog	commodor	2	male	FALSE

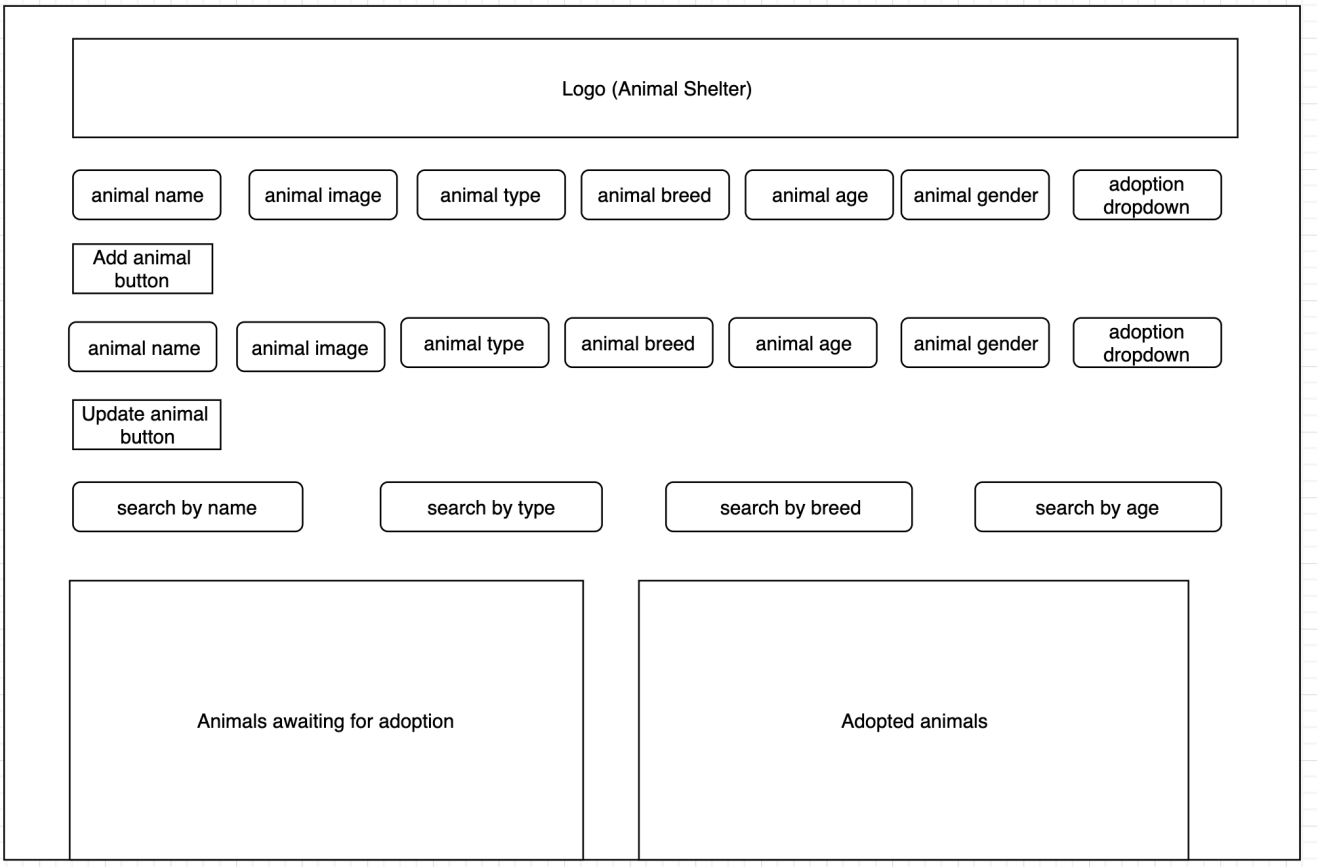
The first screenshot above shows that a user input data in my animal shelter app (name, image, type, breed, age, gender and adoption status).
The second screenshot shows that this data has been save into my SQL database (on line 21).

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

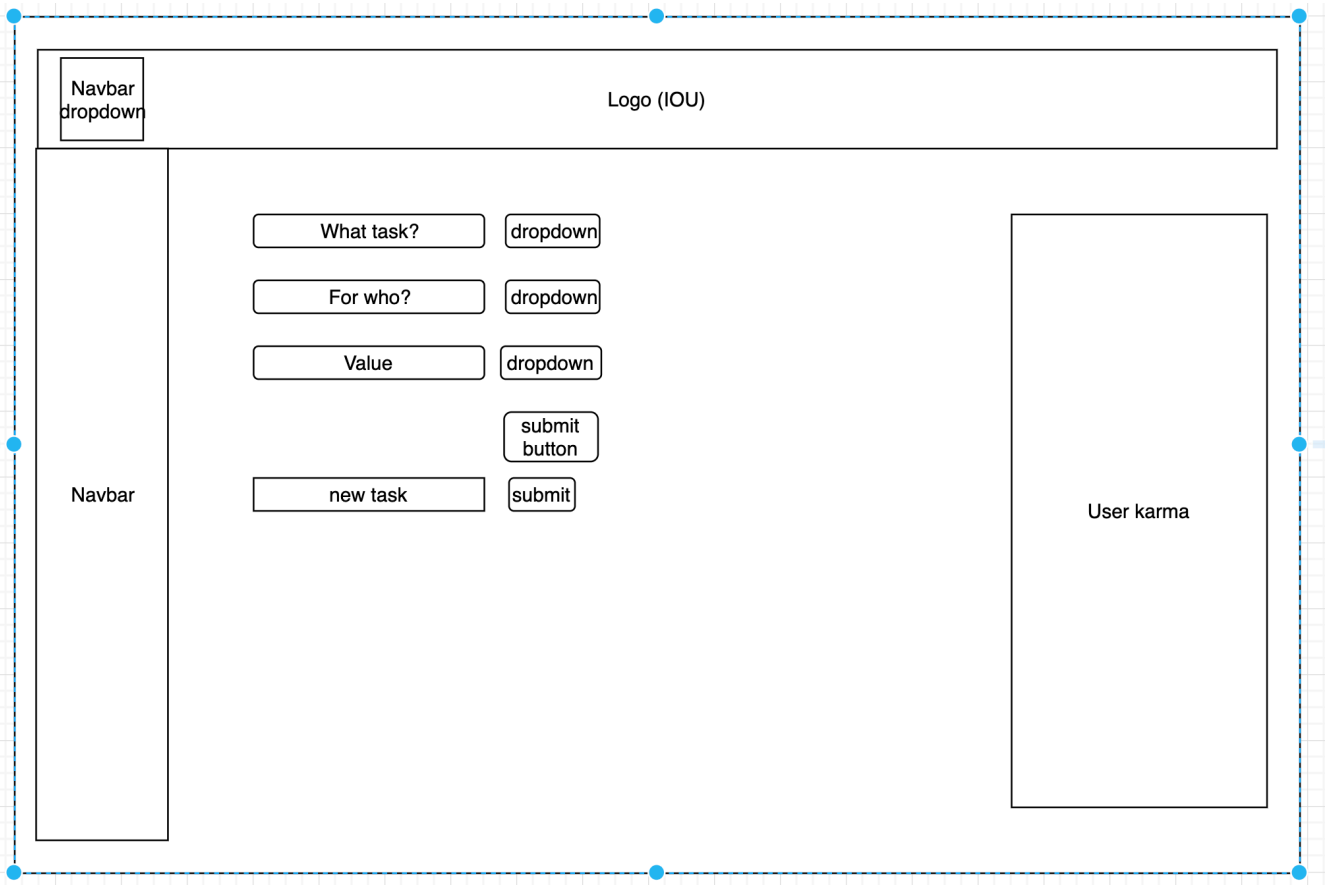
Acceptance Criteria	Expected Result	Pass/Fail
The user is able to login.	The user types the username on the Login page and it takes the user to the Profile page.	Pass
The user can add new task.	After user chose task, cost and clicked “add task” program confirms that “task has been added”	Pass
The user is able to add new task that aren’t on the list yet.	When user types in a new task on New Task page and click Submit, the new task is added to the database.	Pass
The user is able to logout.	When user clicks on Logout program returns the Login page.	Pass
The user can create new groups.	When user logged in new group added on group page that is saved to the database.	Pass

The above is an Acceptance Criteria and Test Plan for my group project. I’ve created this to help me with my Test driven development to make sure that we have all the basic functionality in the program that makes it work. We carried out test on all these features.

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams



The above is a wireframe diagram I've created for my solo project (Animal Shelter). It shows what type of information the user can type in into each input and submit buttons. It also shows the search bars and the two different grid that separates the adopted animals and the animals that are awaiting for adoption.



The above is a wireframe diagram of the new task page from our IOU group project.

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

```java
import java.util.*;

public class Shop {

  private ArrayList<IPlay> stock;
  private String name;
  private int price;

  public Shop(String name){
    this.name = name;
    this.stock = new ArrayList<IPlay>();
  }

  public void addInstrument(IPlay stock){
    this.stock.add(stock);
  }

  public ArrayList<IPlay> getInstrument(){
    return this.stock;
  }

  public static void main(String[] args) {
    Shop shop = new Shop("Jazz");
    Gitar g = new Gitar("gitar", 100);
    Piano p = new Piano("piano", 300);
    Drum d = new Drum("drum", 200);

    shop.addInstrument(g);
    shop.addInstrument(p);
    shop.addInstrument(d);

    for (int i=0; i<shop.getInstrument().size(); i++ ) {
      System.out.println(shop.getInstrument().get(i).makeNoise());
    }
  }
```

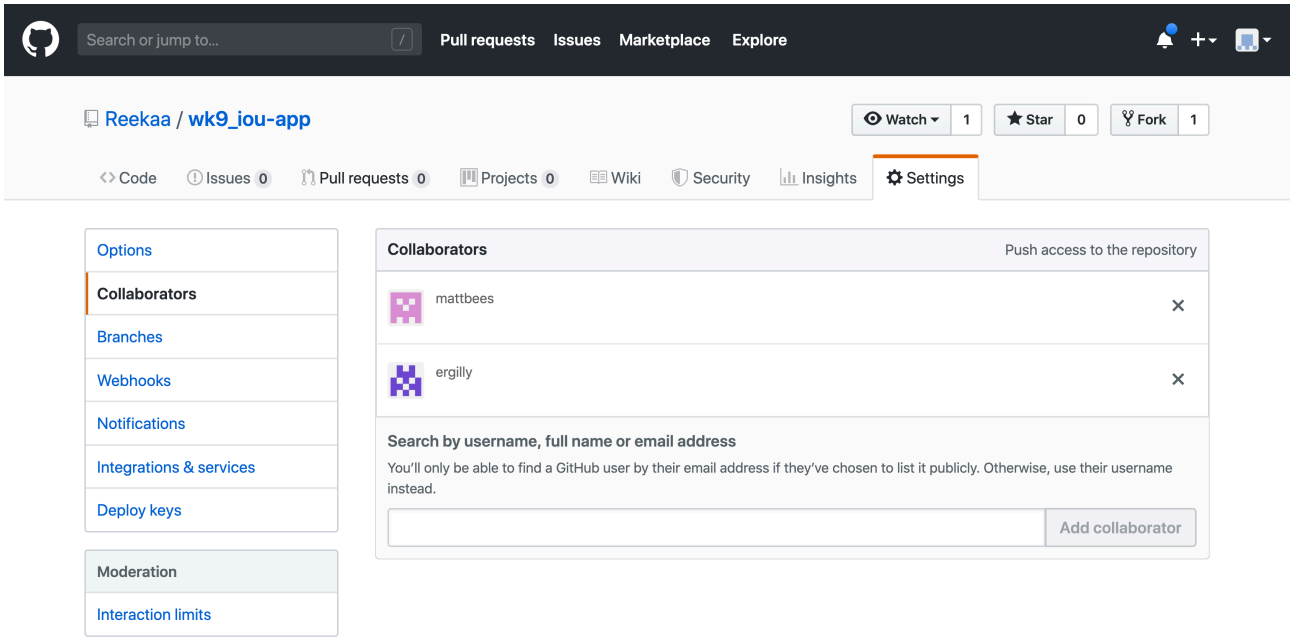
```java
public class Piano implements IPlay{

    private String name;
    private int price;

    public Piano(String name, int price){
        this.name = name;
        this.price = price;
    }

    public String makeNoise(){
        return "mmmmmmm";
    }

}

```

```java
public class Gitar implements IPlay{

    private String name;
    private int price;

    public Gitar(String name, int price){
        this.name = name;
        this.price = price;
    }

    public String makeNoise(){
        return "aaaaaaa";
    }

}

```

```java
public class Drum implements IPlay{

    private String name;
    private int price;

    public Drum(String name, int price){
        this.name = name;
        this.price = price;
    }

    public String makeNoise(){
        return "buuuuu";
    }

}

```

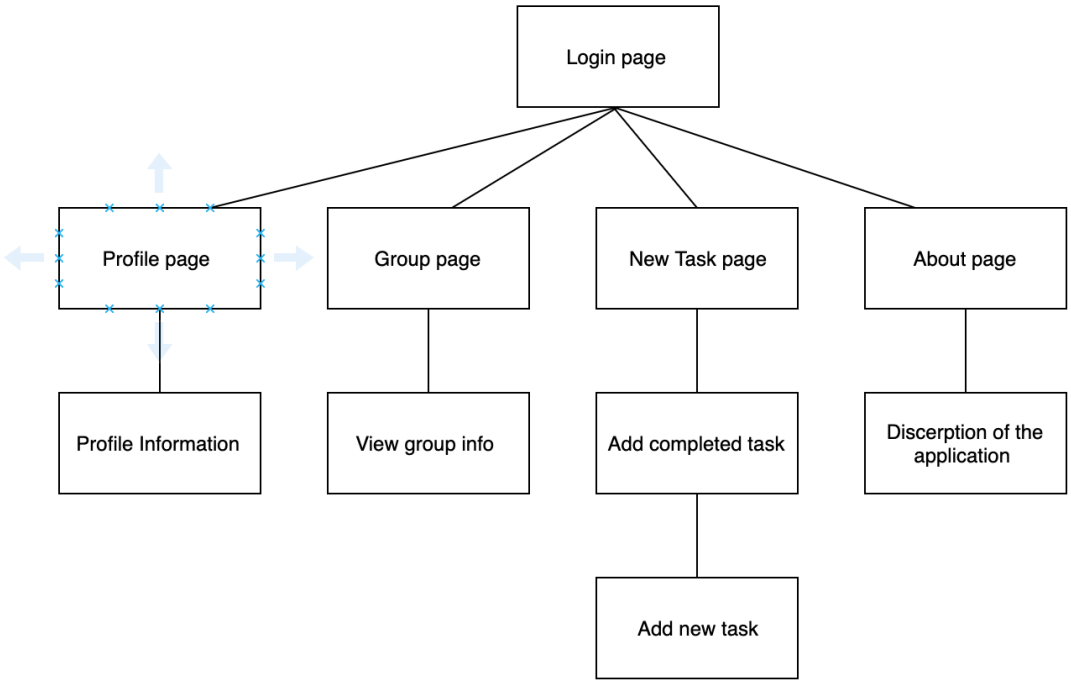
The above screenshots are demonstrating Polymorphism. On the music shop app above I've created an ArrayList (IPlay) to be able to add an instrument to the store with the shared makeSound() method. Each of the above instrument class return what sound an instrument returns. The makeSound() method that takes polymorphism.

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.



I've created the repository for our group project on GitHub. Matthew Beeston and Euan Gilmour were the other contributors in the project.

Unit	Ref	Evidence
P	P.5	User Site Map



This is a screenshot of the User Site Map that I've created for the IOU project as part of the planning of the project. It shows the important part of the application. It shows the different pages of the application. The main page is the Login Page, the user can access the other pages through the navigation bar.

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

Bug/Error	Solution	Data
Try to run application at the first time in Google Chrome and it fails to compile.	Store hasn't been passed into the React-Redux Provider, bug has been fixed by updating index.js with store={store}	29/05/2019
When type into Login box no text appears.	Missing handleInput( ) method in Login.js. Updated the file to set the state to '({userNameInput: evt.target.value)}'	30/05/2019
Couldn't get current user data out of the database on the frontend to display.	Bug fixed: two 'get' methods had been used on the backend database with the same name ('get') I've changed the second method to getByName( )	31/05/2019
Clicking on "Logout" button redirect to Profile Page instead of Login Page.	Bug fixed by correcting the link in the 'Redirect' method to '/login'	03/06/2019
Adding new task on New Task Page crashes backend server.	Bug fixed in addNewTask( ) method by correcting a typo. Also had to restart backend server.	03/06/2019

The above is the bug tracking report I've used when I worked on our group project. We've had list of bugs on our Trello board to keep track of them and make sure that they are fixed. The project had different bugs I've solved, see examples above.

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

Animal Shelter Application		
Constrain Category	**Implementation Constraint**	**Solution**
Hardware and Software Platforms	The application maybe not work an all type of browsers. It works on GoogleChrome and Firefox but not on unsupported browsers. This could stop users to access the application if they don't have up to date browsers.	Remind clients through the website that the website run on GoogleChrome and Firefox better.
Performance Requirements	Application maybe update the database too slowly or it hasn't update it at all due to the lack of space in the database. It slows down the application. That effects negatively the user experience.	Upgrade backend server to create more space.
Persistent Storage and Transactions	Database for this project has been created on my laptop, for other developers to run my code the database needs to be set up on each laptop separately. This make it hard to run the application on other devices.	Database transferred over to an outside server where everyone can access it easily.
Usability	Users could have sight impaired. Means they can't read the normal user interface and they unable to use the application. They potentially miss out on information they need.	Build in option to the application to change user interface size. This makes all information visible for all users.
Budgets	There was no budget to develop this application. That means had I had a limit of available programs and resources that I could have used. If I'd like use extra resources maybe it would have cost me.	I've chosen resources after I thought trough what I need exactly to complete the application.
Time Limitations	The time limit for the project was 1 week. It means they were features that I couldn't complete on this time limit.	I've had a plan for the full week manage my time throughout the week as I had to make sure that I've developed a working application.

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

```
//handle user input by setting the state to the target value of the input
//handle when user press submit button
//- get current user from database
//- set state to true to redirect the page if the user entered the a correct username
//add if statement to redirect the page to user profile with current user information
```

I've created the above pseudocode for the login page on our group project. It goes over step by step how the user input will be handled and used in the code to redirect the page to the user page with the current users details.

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.

1.

```
15    dealCards(){
16      this.deck.forEach((card, index) => {
17        if(index % 2 === 0){
18          const player1 = this.players[0];
19          player1.hand.push(card);
20        }else{
21          const player2 = this.players[1];
22          player2.hand.push(card);
23        };
24      });
25    };
```

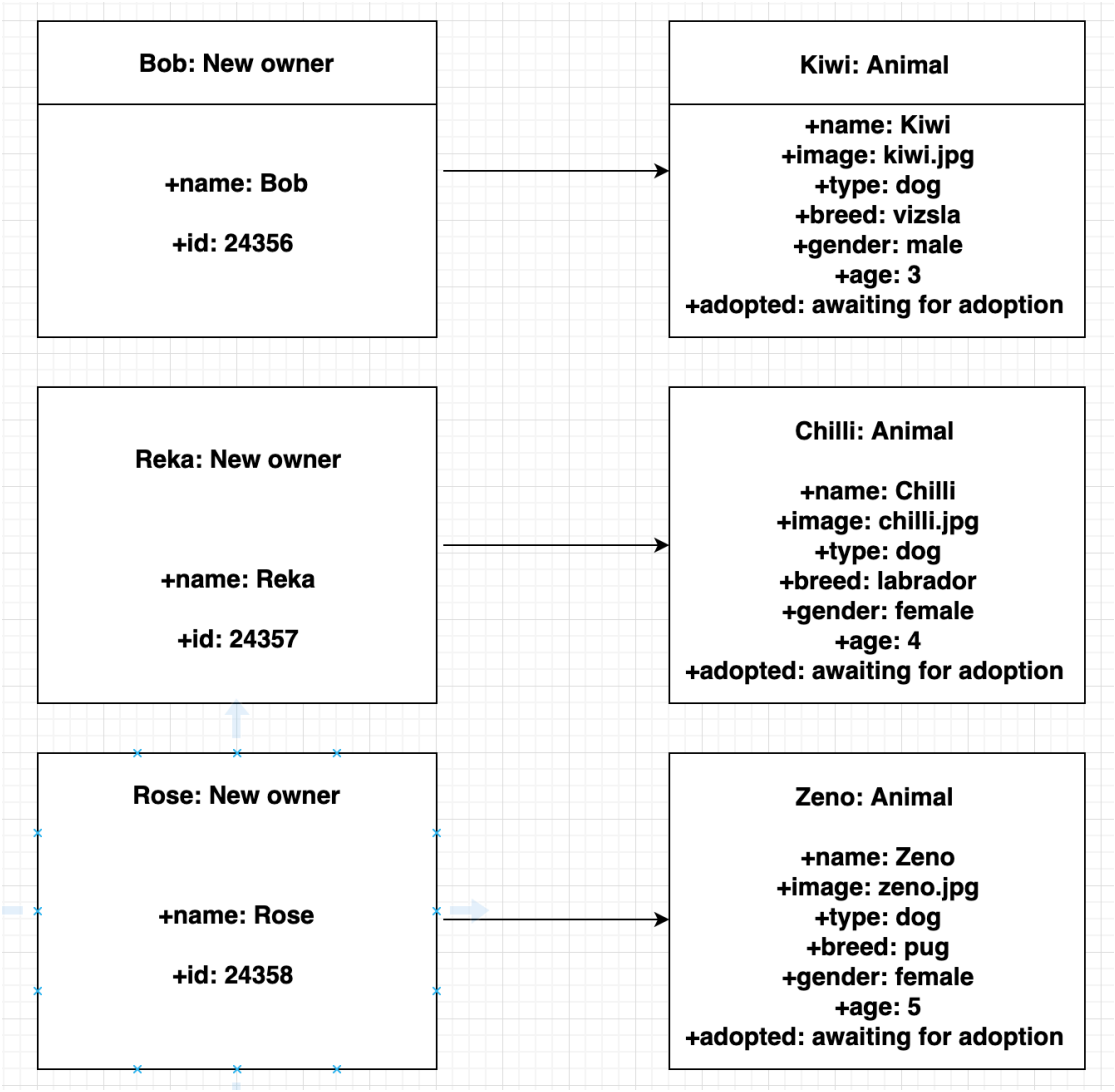
The above algorithm is from my Top Trumps card game homework. This method is dealing the cards evenly to the players. I've chosen this method to iterate through every card in the deck and if the card index divided by 2 equals to 0 push the card into player1's hand and if doesn't equals to 0 push the card into player2's hand.

2.

```
checkWinner(){
    let winningLines =
      [
        ['0', '1', '2'],
        ['3', '4', '5'],
        ['6', '7', '8'],
        ['0', '3', '6'],
        ['1', '4', '7'],
        ['2', '5', '8'],
        ['0', '4', '8'],
        ['2', '4', '6'],
      ]
      this.checkMatch(winningLines)
}

checkMatch(winningLines){
  for (var i = 0; i < winningLines.length; i++) {
    const [a,b,c] = winningLines[i];
    let board = this.state.board;
    if(board[a] && board[a] === board[b] && board[a] === board[c]){
      this.setState({winner: this.state.player})
    }
  }
}
```

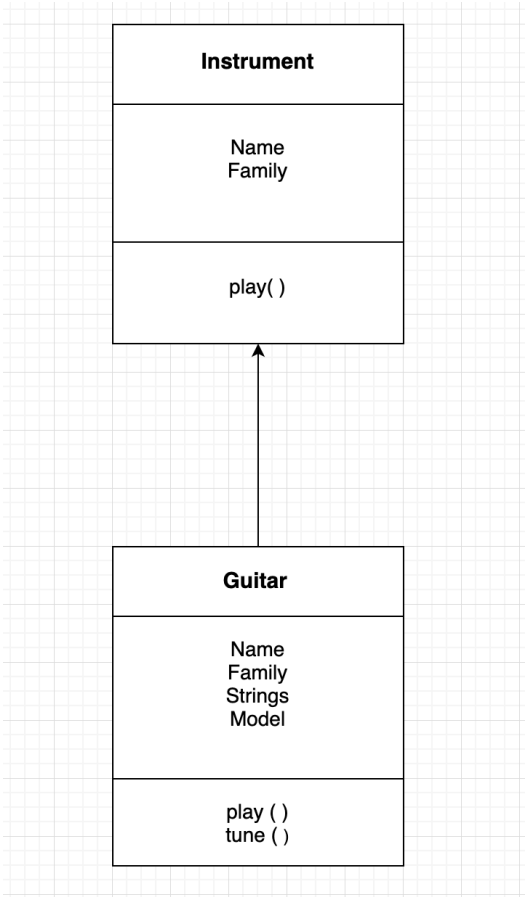
This example is from my Tic Tac Toe homework. It shows the game logic of how the game determinate who is the winner. I have chosen this option to check the winner of the game. The function checkMatch( ) loops through all possible winning options and give me the result. Once the loop went through all possible options it sets the winner.

Unit	Ref	Evidence
A&D P	A.D.3 P.8	Produce three Object Diagrams



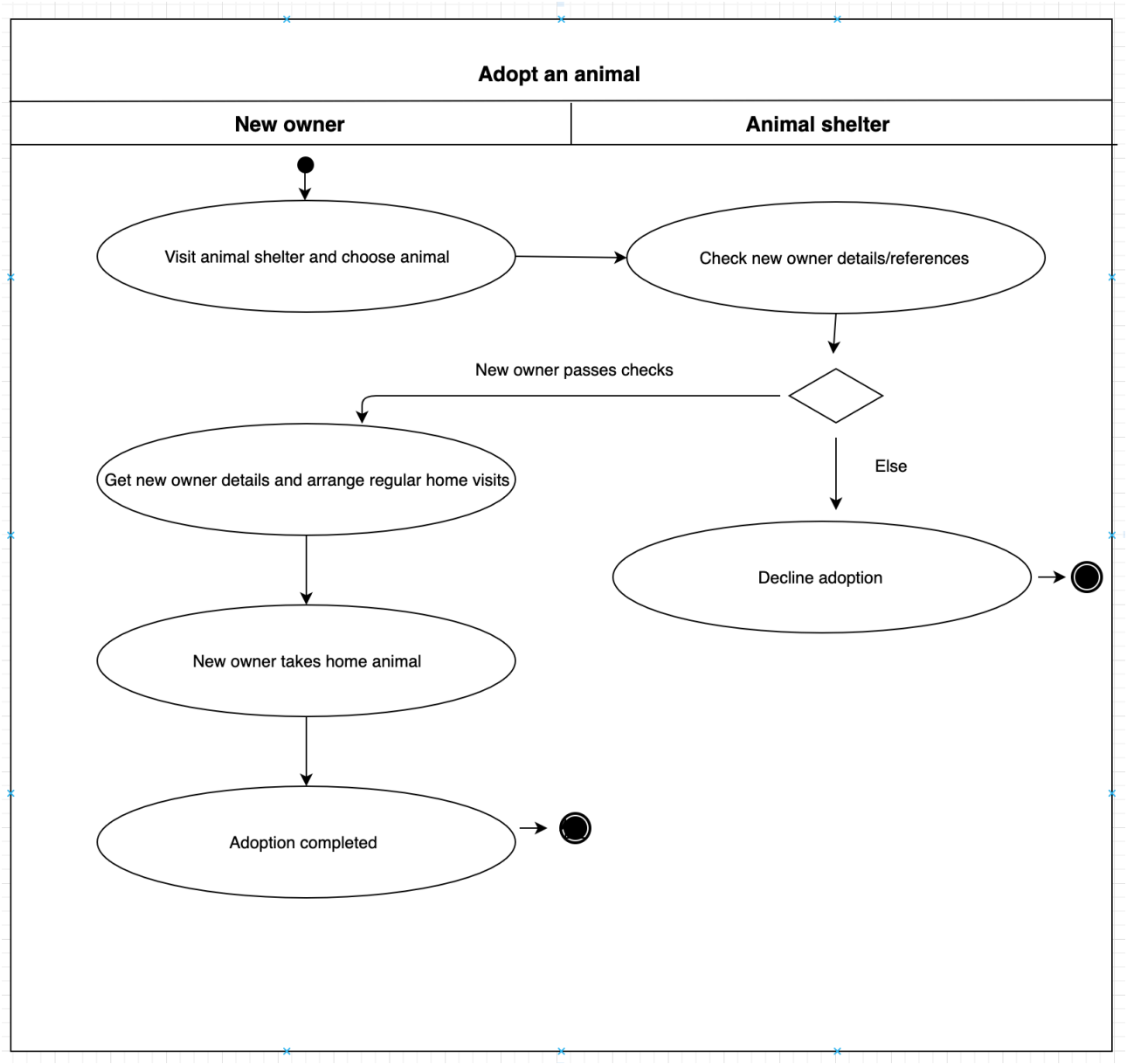
The above is an Object Diagram that I've made for my solo project of Animal Shelter. It shows to classes the New Owner and the Animal and the object of these classes. A new owner can adopts an animal from the shelter and the diagram shows the details that goes into the system with both the new owner and the animal details.

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



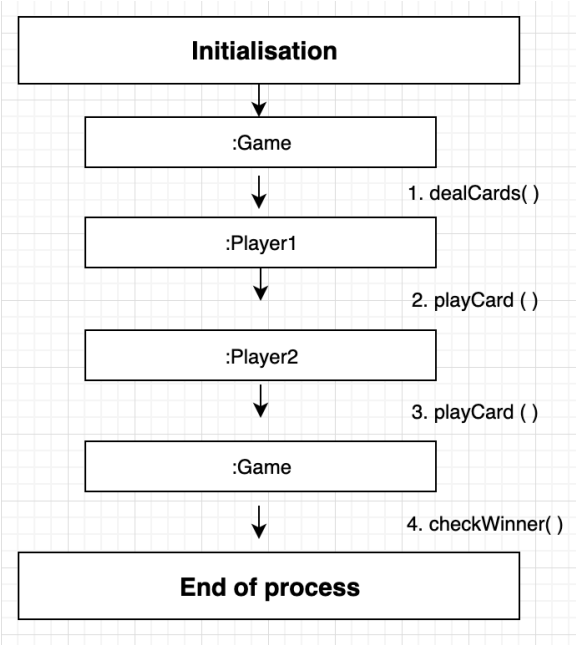
The above is an Inheritance Diagram I've created for a homework. The instrument class has a function (play( )) and name and family. The guitar inherits this play function, name and family from the instrument.

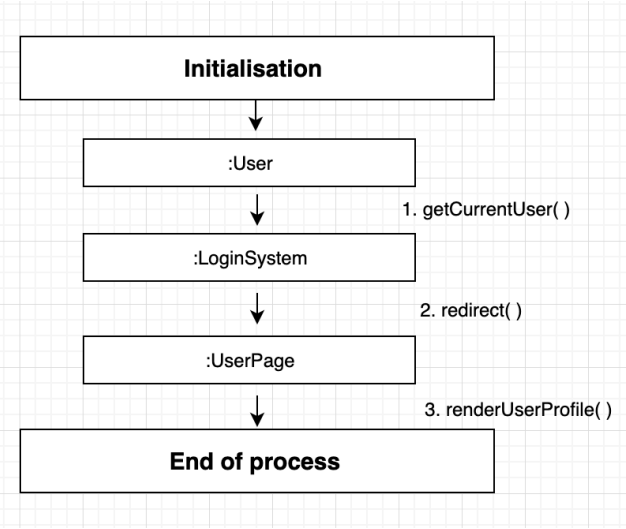
Unit	Ref	Evidence
A&D	A.D.4	Produce an Activity Diagram



The above is an activity diagram that details how someone can adopt and animal from the animal shelter.

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



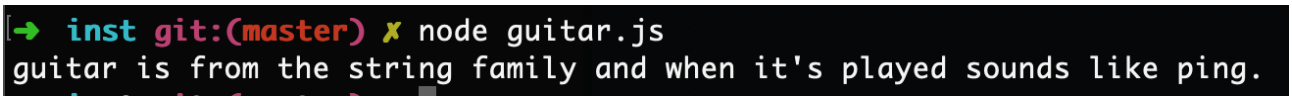


I have produces the above system interaction diagrams for two different projects I've worked on.

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.

```
class Instrument{

  constructor(name, family){
  this.name
  this.family
  }
}

module.exports = Instrument;
```

```
const Instrument = require('./instrument.js')

class Guitar extends Instrument {

  constructor(name, family, play){
    super(name, family)
    this.name = name
    this.play = play
    this.family = family
  }

  guitar(){
    return `${this.name} is from the ${this.family} family and when it's played sounds like ${this.play}.`
  }
}

const myGuitar = new Guitar('guitar', 'string', 'ping')

console.log(myGuitar.guitar());

module.exports = Guitar;
```



The above is an example for inheritance in my instrument homework. The guitar inherits the name and the family from the instrument.