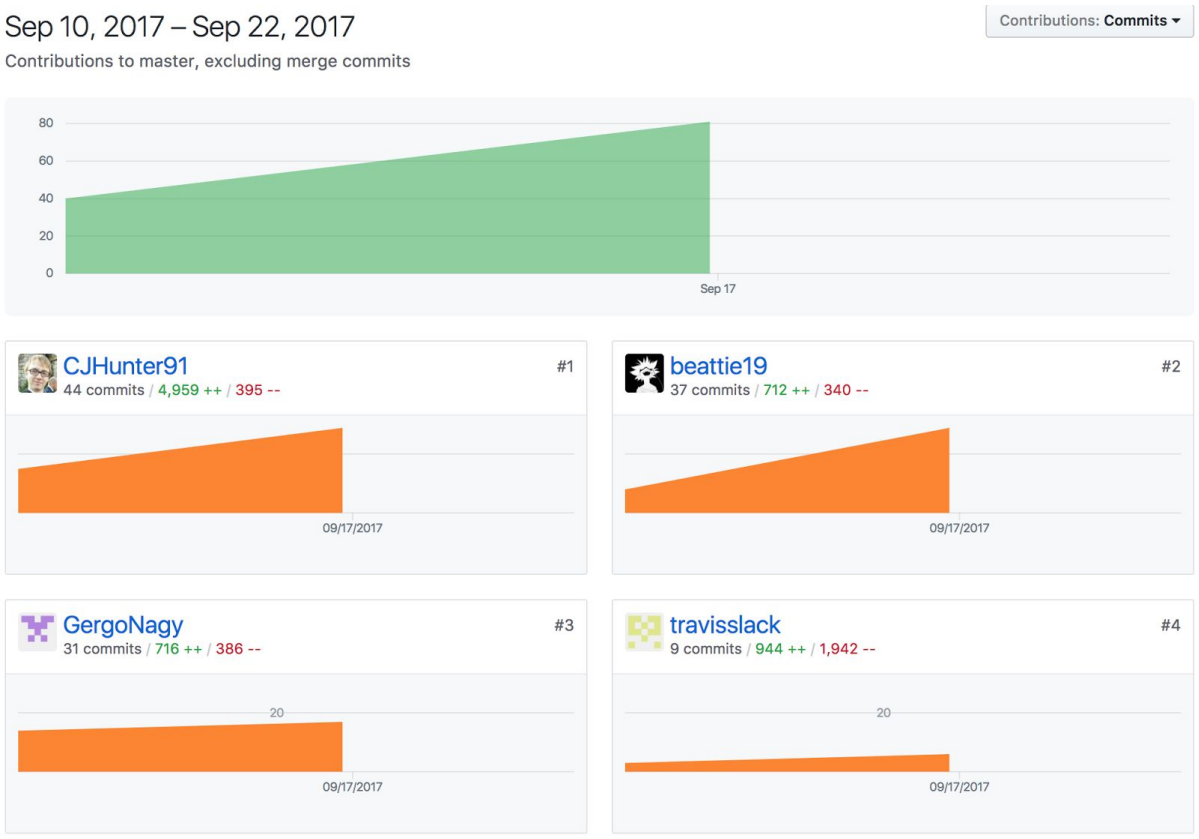
Chris Hunter - Project Unit(SQA PDA: Software Development)

P 1. Project Contributors



P 2. Project brief

Educational App

The BBC are looking to improve their online offering of educational content by developing some interactive apps that display information in a fun and interesting way.

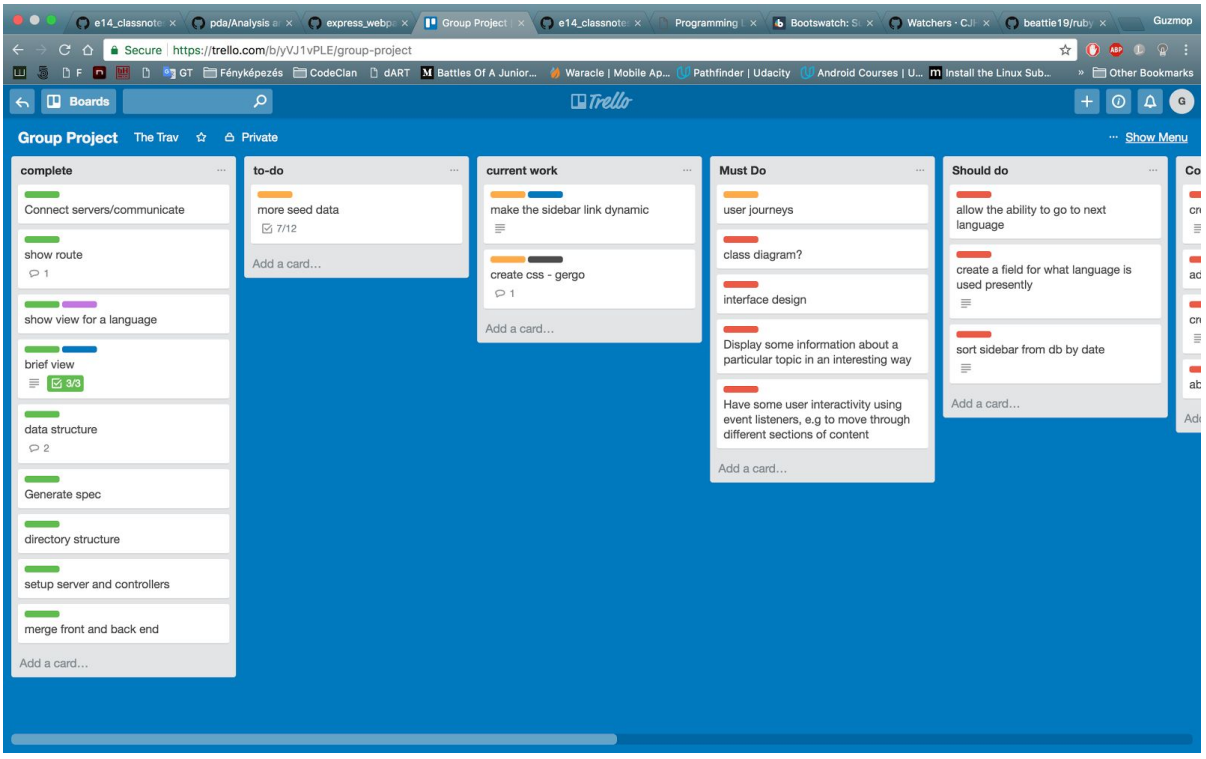
Your task is to make an MVP to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app. You might use an API to bring in content or a database to store facts. The topic of the app is your choice, but here are some suggestions you could look into:

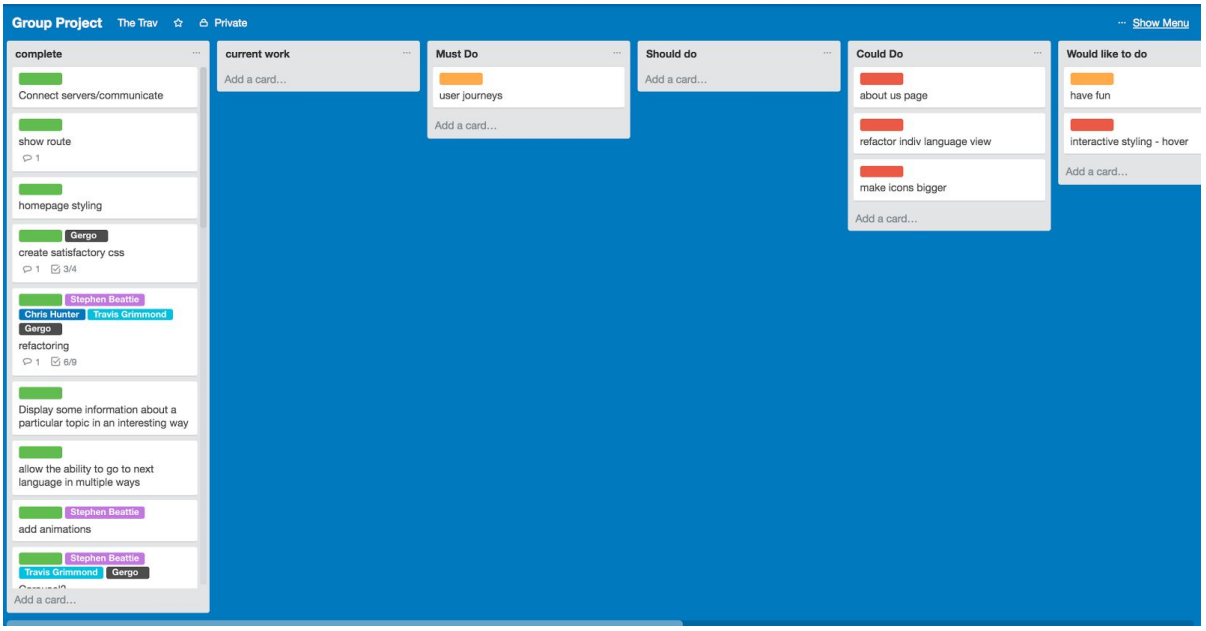
- Interactive timeline, e.g. of the history of computer programming
- Interactive map of a historical event - e.g. World War 1, the travels of Christopher Columbus

MVP

- Display some information about a particular topic in an interesting way
- Have some user interactivity using event listeners, e.g to move through different sections of content

P 3. Project Planning



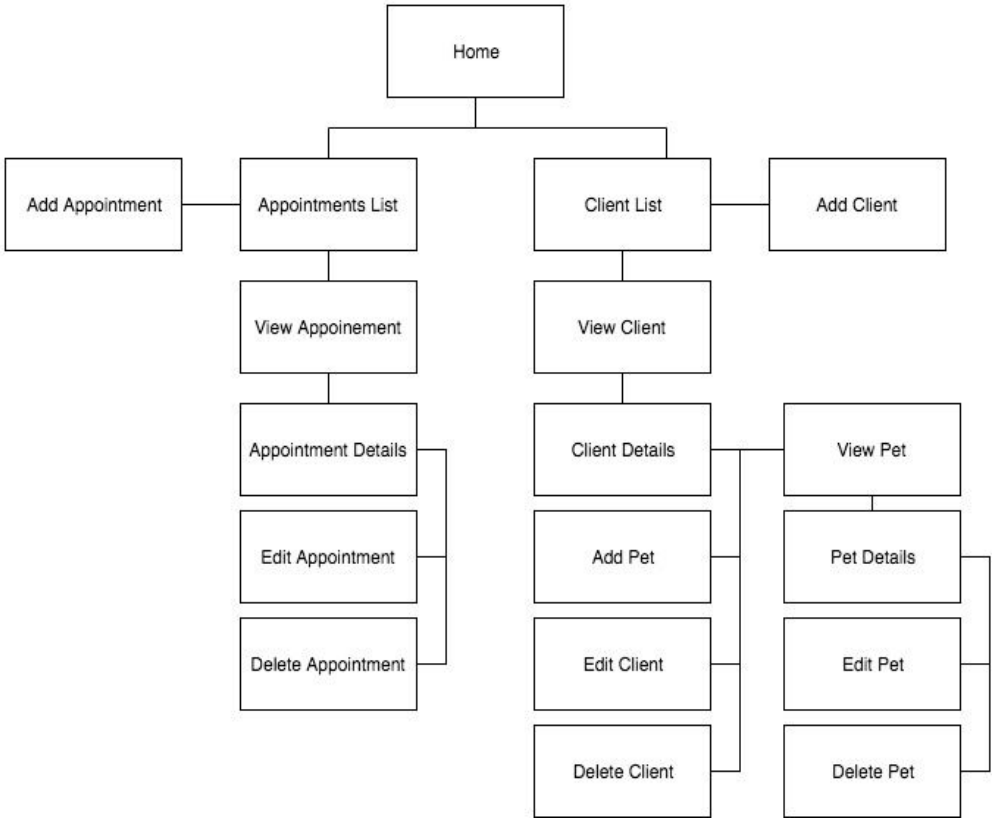


P 4.Acceptance Criteria

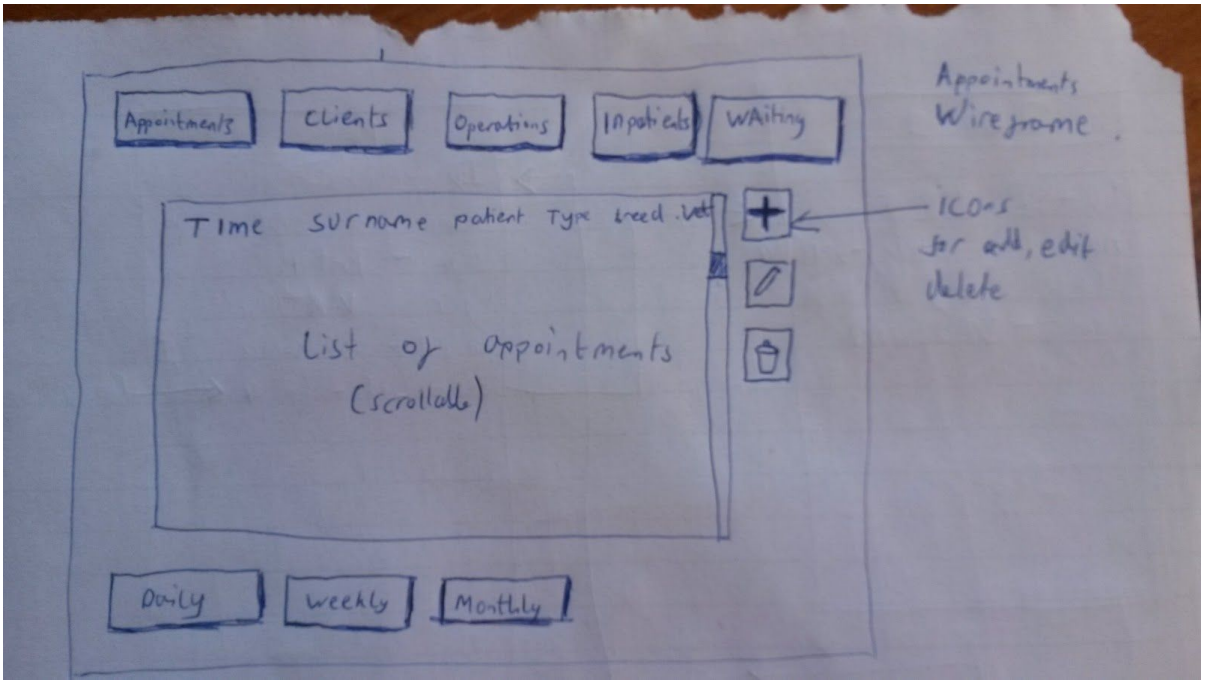
Acceptance Criteria	Expectation/Result	Pass / Fail
A user can access different programming languages.	A user clicks on the forward/back button and the next/prev language populates the view.	Pass

A user can use an interactive timeline to view languages	A user can click on the timeline arrows to scroll through the programming languages and click to select a language	Pass
A user can go back to the history main view	The user can click on the home icon to take them back to the original history view	Pass

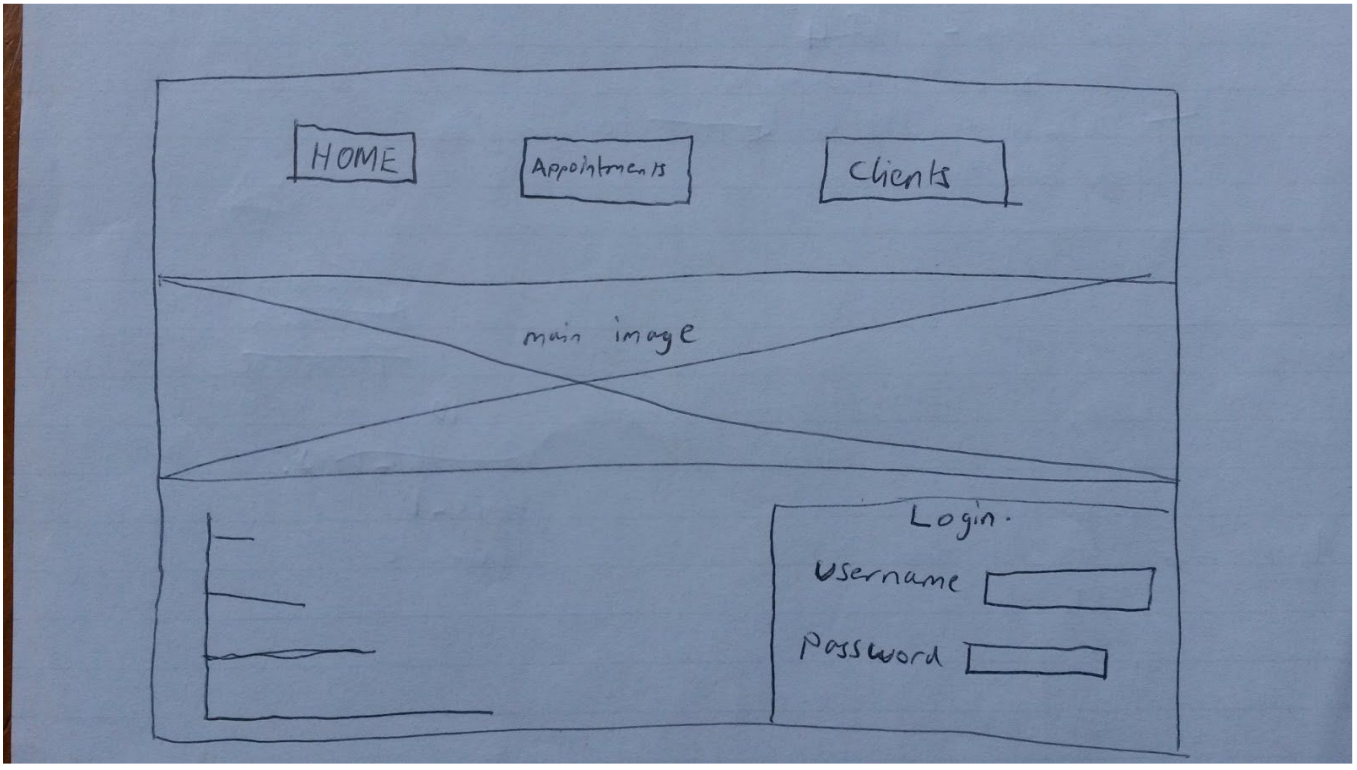
P 5. Site Map



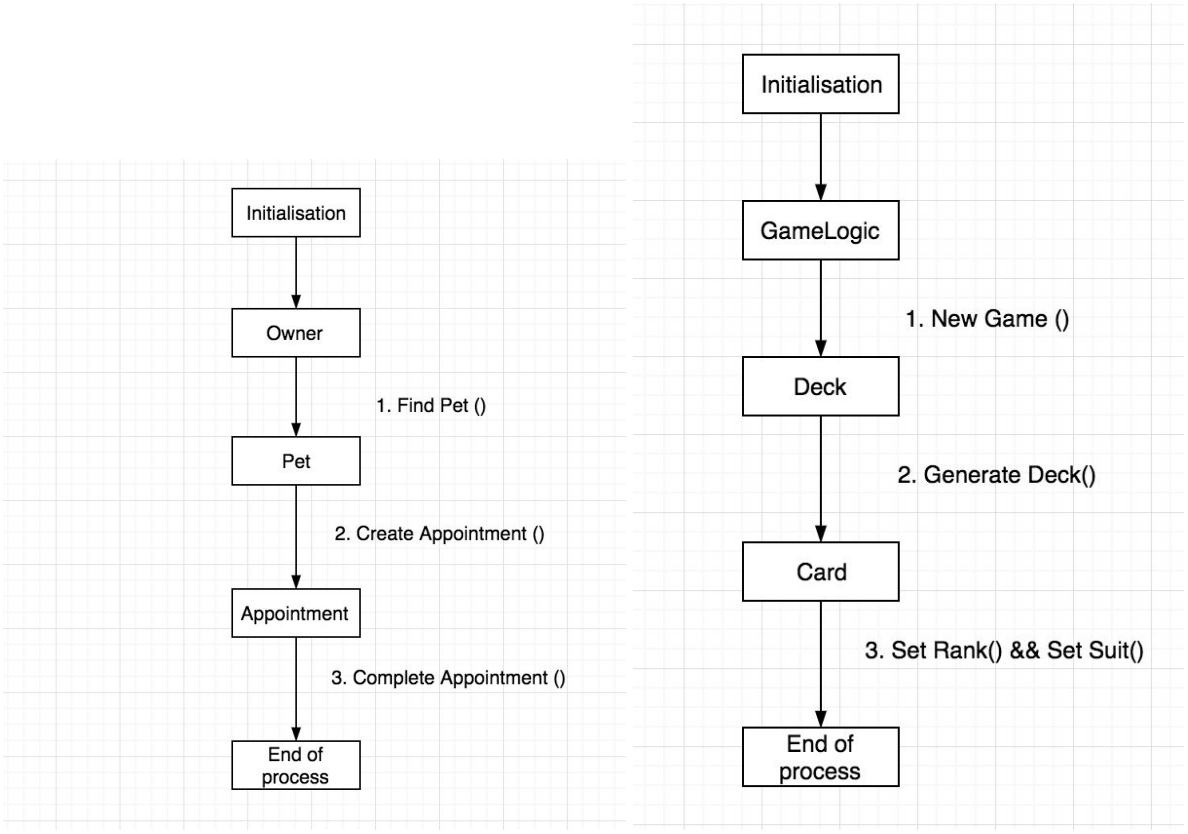
P 6. (i) Wireframe



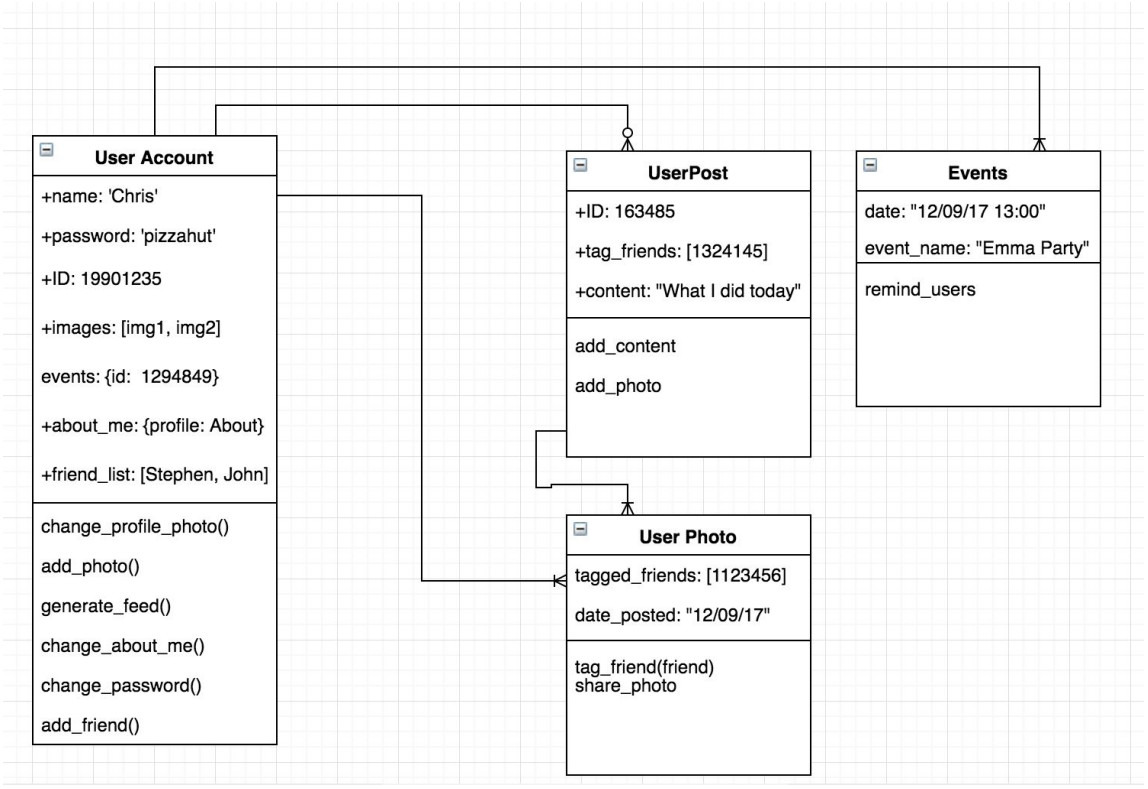
P 6. (ii) Wireframe



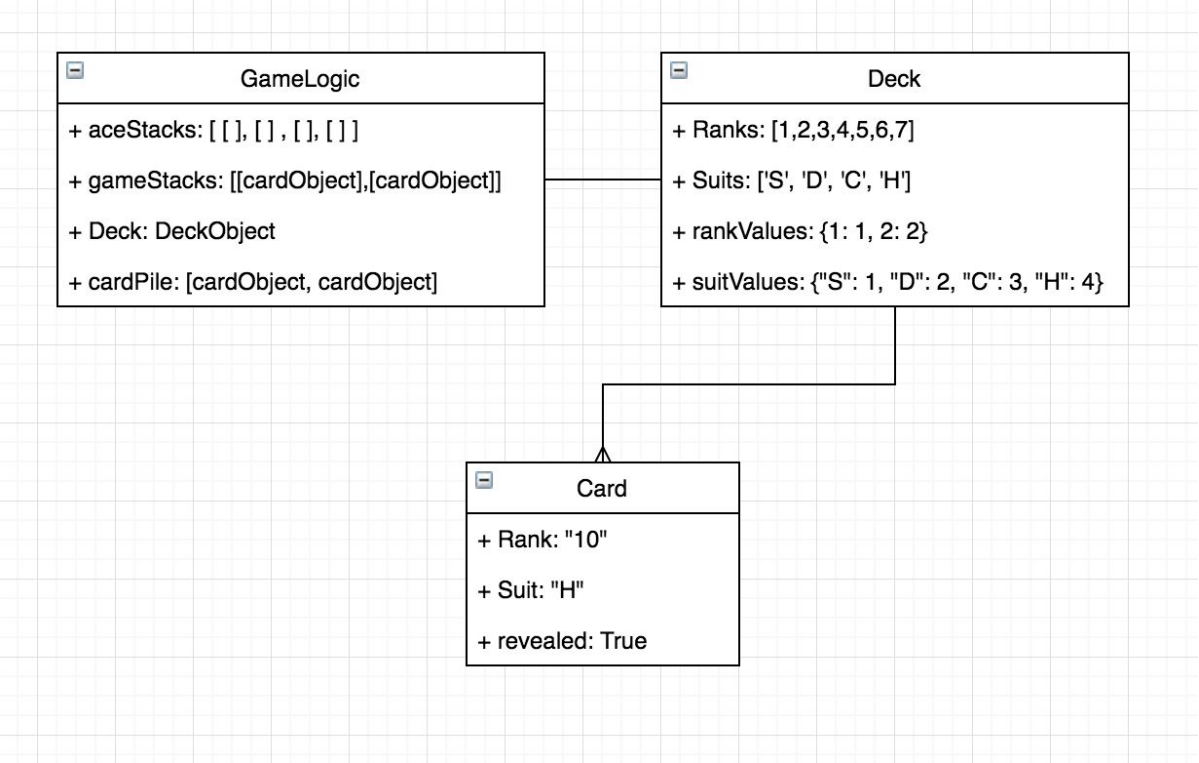
P 7. System Interaction diagrams



P 8(i). Object Diagrams



P 8(ii). Object Diagrams



P 9. Algorithms examples (i)

This algorithm was used because the size of any the arrays would be less that 52 and they are unsorted. It uses a simple nested loop to find a card in different game stacks. It is not the most efficient algorithm because of this.

```
public ArrayList<Integer> findCard(Card card) {
    ArrayList<Integer> find = new ArrayList<>();
    for(ArrayList<Card> stack: this.gameStacks) {
        for (Card eachCard : stack) {
            if (card == eachCard) {
                find.add(this.gameStacks.indexOf(stack));
                find.add(stack.indexOf(eachCard));
            }
        }
    }
    if(find.size() == 0){
        for(Card eachCard: this.pile){
            if(card == eachCard){
                find.add(-1);
                find.add(this.pile.indexOf(eachCard));
            }
        }
    }

    return find;

}
```

P 9. Algorithms examples (ii)

This algorithm was chosen because when iterating through the game stacks, to find all the cards that come after the card we are moving, it is important to iterate in reverse due to removing elements from the array. This keeps the indices the correct number so the loop can be completed properly.

```
public void move(Card card, int targetStack){
    ArrayList<Card> temp = new ArrayList<>();
    //iterate through the stack to add each card to temp in prep for move
    //removing each card from the moving stack
    ArrayList<Integer> find = findCard(card);
    int stack = find.get(0);
    int stackItem = find.get(1);

    for(int i = this.gameStacks.get(stack).size()-1; i >= stackItem; i--) {
        temp.add(getCard(stack, i));
        this.gameStacks.get(stack).remove(i);
    }
    //adds each card from temp to the target stack
    for(int j = temp.size()-1; j >= 0; j--){
        this.gameStacks.get(targetStack).add(temp.get(j));

    }
    revealNextCard(stack);
}
```

P 10. Pseudocode

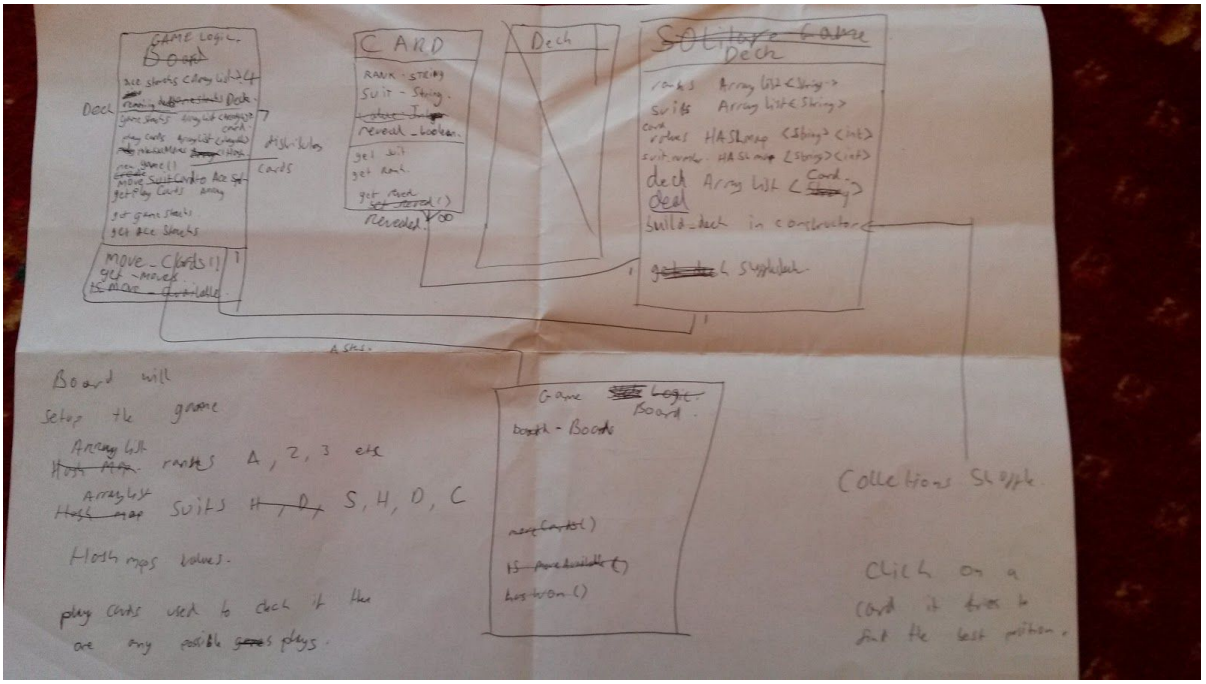
```
def 'get the time for an appoinement' ()
  # create a sql query to return the date for an appointment
  # create a database connection
  # query the database and save the result to a variable
  # create a time object using thesaved variable
  # close the database connection
  # return the time object
```

P 11 Screenshot of personal project and GitHub Link:
(https://github.com/Orkem/CC_Solitare)

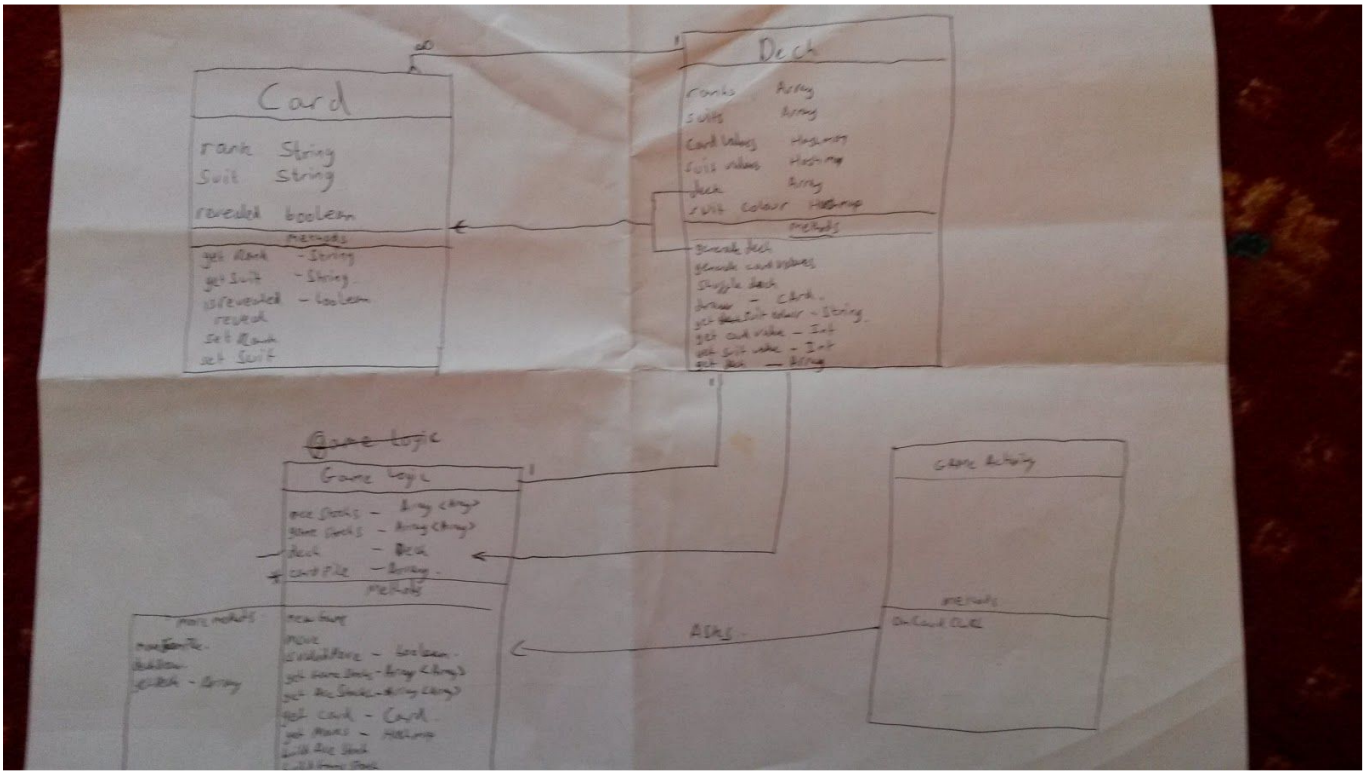
```java
package com.codeclan.solitare;

import ...

/**...*/

public class GameLogic {
    private ArrayList<ArrayList<Card>> aceStacks;
    private ArrayList<ArrayList<Card>> gameStacks;
    private Deck deck;
    private ArrayList<Card> pile;
    private boolean isWon;

    public GameLogic(){
        this.deck = new Deck();
        this.gameStacks = new ArrayList<>();
        this.aceStacks = new ArrayList<>();
        this.pile = new ArrayList<>();
        this.isWon = false;
    }

    private void  buildAceStack(){
        for(int i=0; i <4; i++){
            aceStacks.add(new ArrayList<Card>());
        }
    }

    private void buildGameStack(){...}

    public ArrayList<ArrayList<Card>> getGameStacks() { return gameStacks; }

    public ArrayList<ArrayList<Card>> getAceStacks() { return aceStacks; }

    public String getColour(Card card) { return deck.getSuitColour(card); }
    public Card getCard(int stack, int stackItem){...}

```

P 12 Pictures/Screenshots of different planning stages to show development

Class Diagram initial design

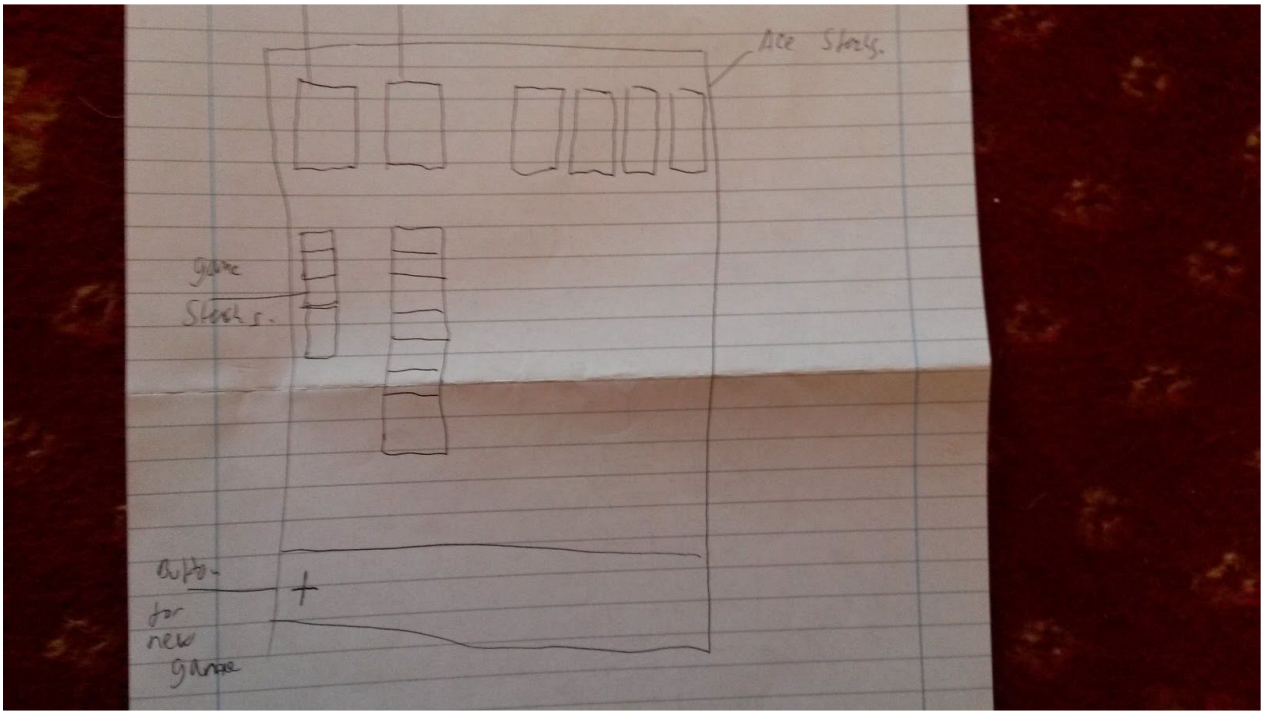


Class Diagram after

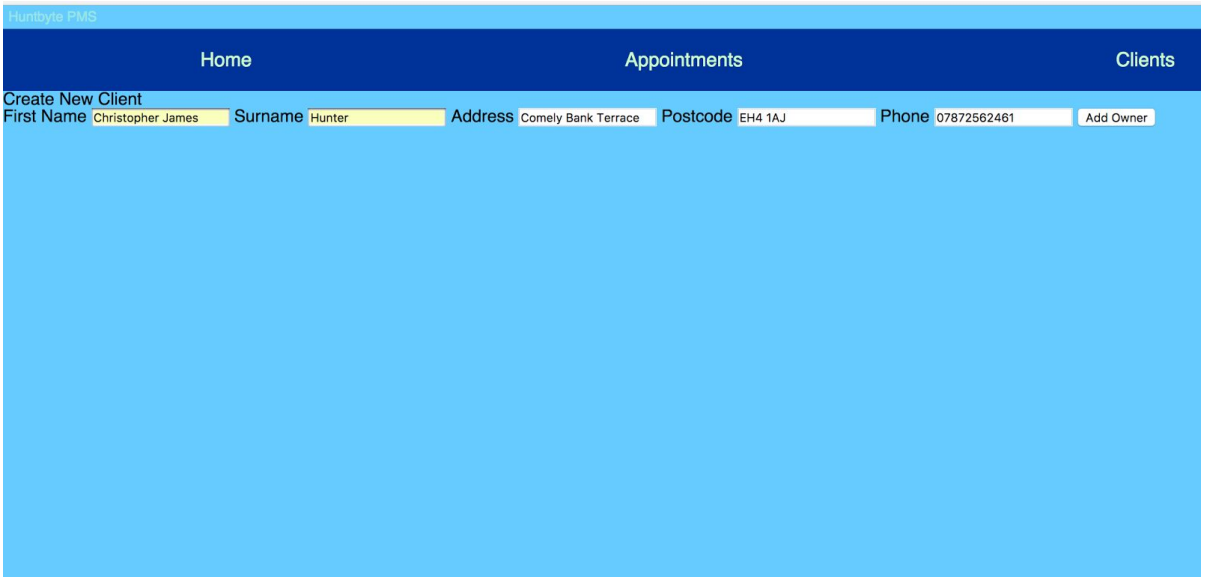


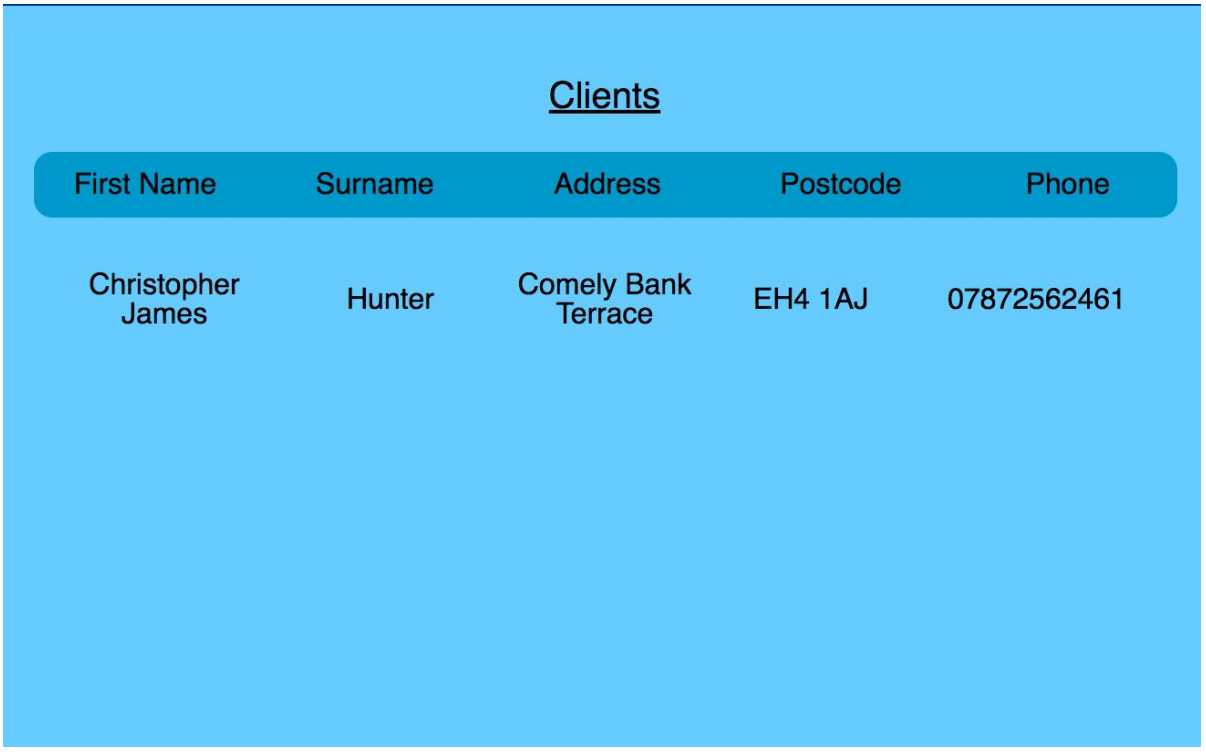
WireFrame of GameActivity initial design

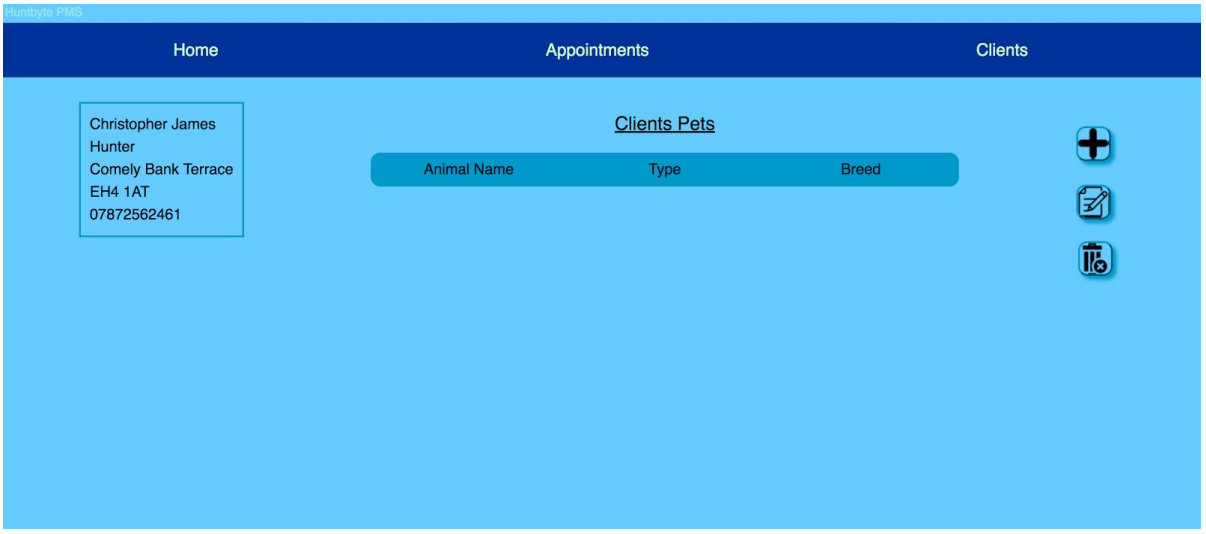
WireFrame of GameActivity After



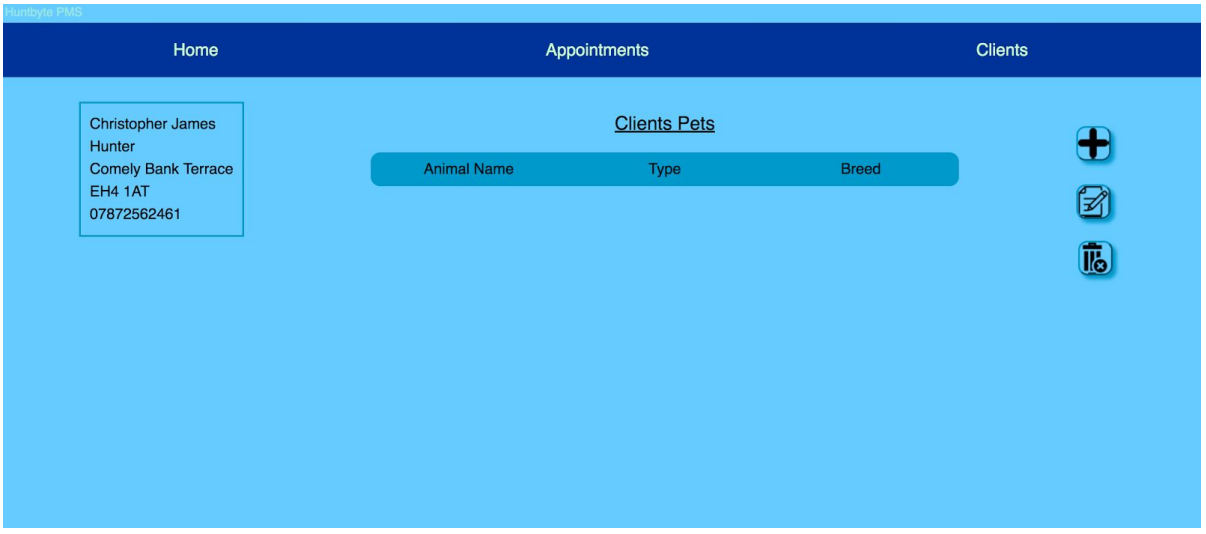
P 13. User input being processed according to design requirements.

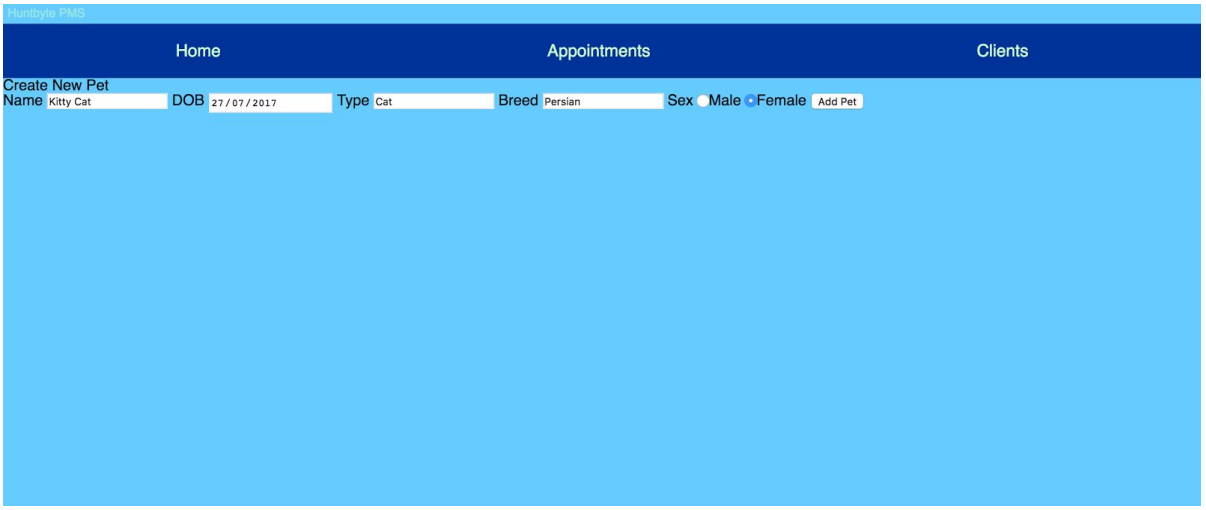

Huntbyte PMS
Home
Appointments
Clients
Create New Client
First Name Christopher James
Surname Hunter
Address Comely Bank Terrace
Postcode EH4 1AJ
Phone 07872562461
Add Owner

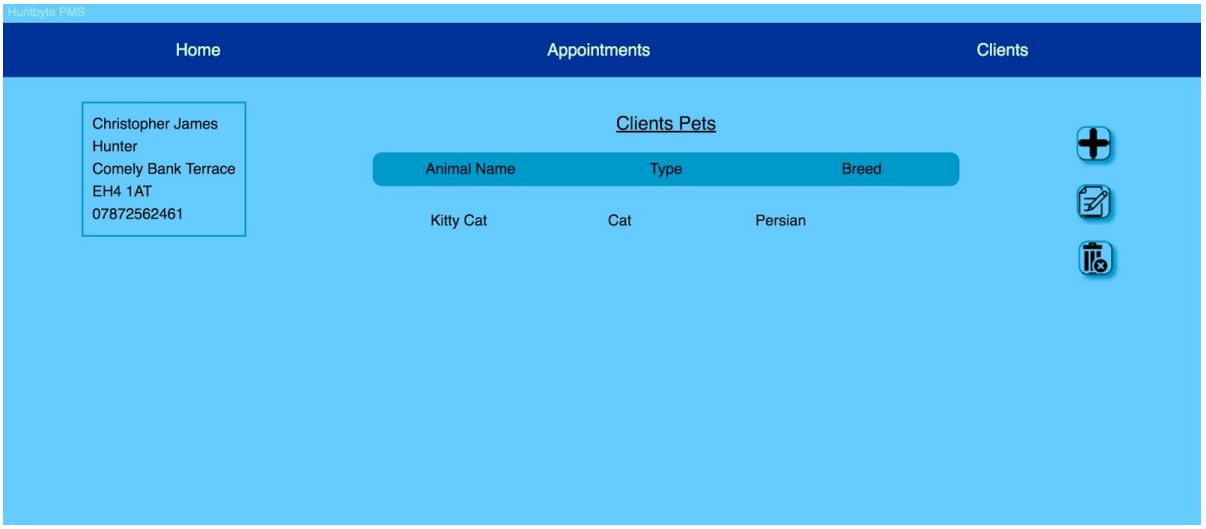

Clients
First Name
Surname
Address
Postcode
Phone
Christopher James
Hunter
Comely Bank Terrace
EH4 1AJ
07872562461



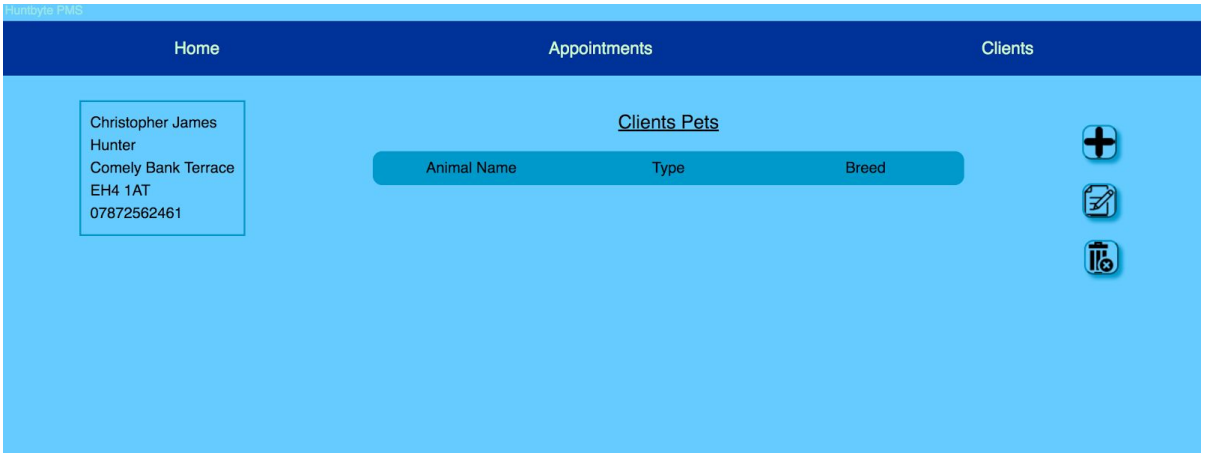
P 14. Data persistence







P 15. Output of results and feedback to user



P 16 API use example

ajax_request.js

```
var AjaxRequest = function(url){
  this.url = url;
  this.onUpdate = null;
}

AjaxRequest.prototype.getData = function() {
  var xhr = new XMLHttpRequest()
  xhr.open("GET", this.url)

  //add callback
  xhr.addEventListener('load', function(){
    if(xhr.status !== 200) return;
    var data = JSON.parse(xhr.responseText);
    this.onUpdate(data);
  }.bind(this));

  //send request
  xhr.send();
};

```

```
var requestWeatherUpdate = function(googleMap){
  var allCoords = googleMap.getMarkerCoords();
  document.querySelector('#weather-container').innerHTML = "";
  allCoords.forEach(function(coords){
    var weatherData = new AjaxRequest("http://api.openweathermap.org/data/2.5/weather?lat="+coords.lat +"&lon="+
      coords.lon + getKey());
    var weatherView = new WeatherView(document.querySelector('#weather-container'), weatherData, googleMap);
    weatherData.getData();
  })
```

P 17 Bug Tracking Report

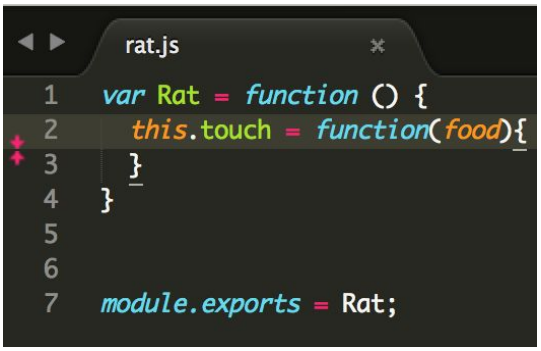
A user can view different languages	Fail	Added buttons to move to a different language	Pass
Languages contain a graph on popularity over time	Fail	Highcharts API used to change popularity data into graph	Pass
A user can view all languages available	Fail	Added a timeline with language icons	Pass
Timeline can scroll from language to language	Fail	Added timeline animation.	Pass
A user can go back to the history landing page	Fail	Added home button to reset the view	Pass
When scrolling through languages the timeline positions at that language.	Fail	Added extra logic to forward and back event listeners to change timeline position	Pass
It is possible to click on a timeline language and load the details	Fail	Add event listeners to each list item in the timeline to change view onclick	Pass

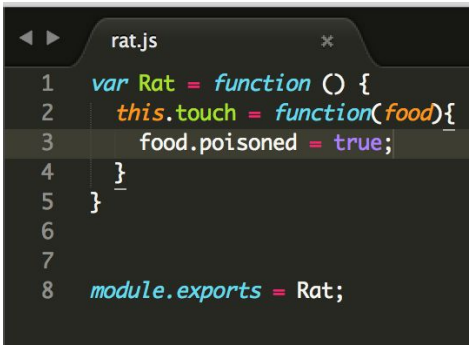
P 18 Testing example

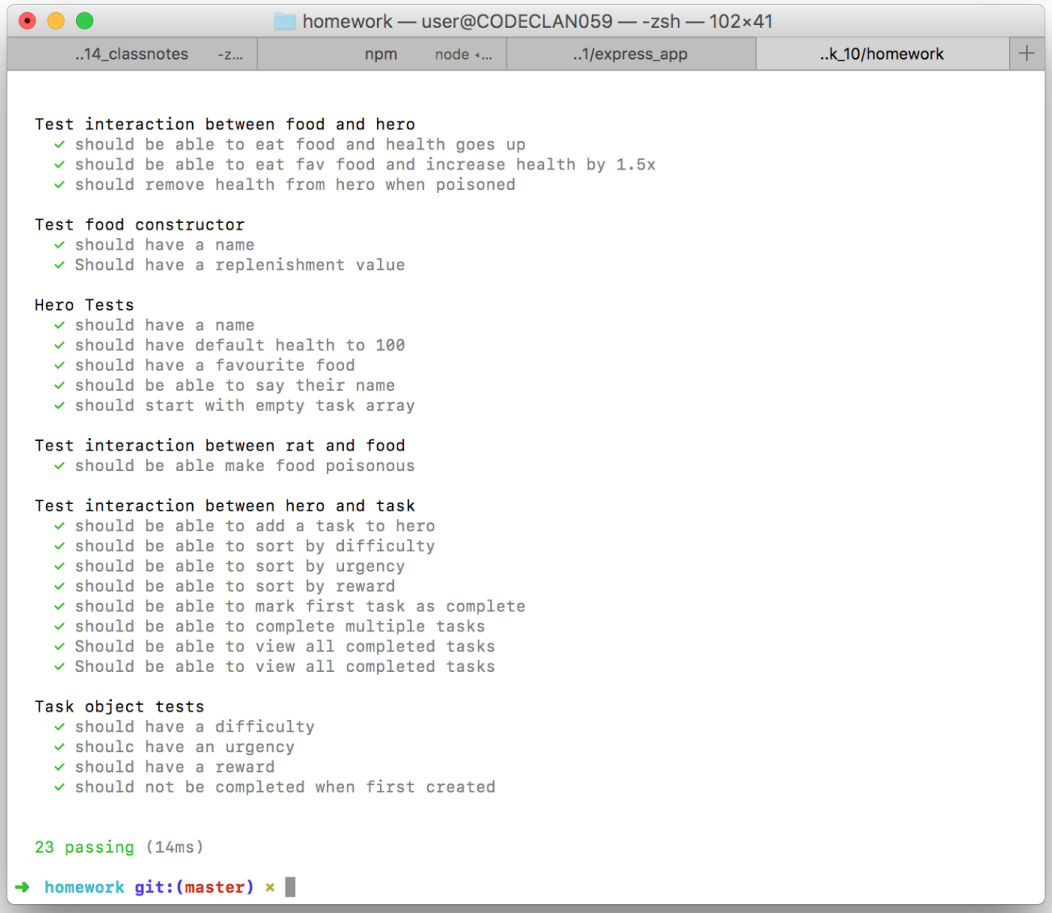
rat.js | food_hero_integration_spec.js

```
var assert = require("assert");
var Food = require("../food.js");
var Hero = require("../hero.js");
var Rat = require("../rat.js");

describe("Test interaction between food and hero", function(){

  var tuna;
  var cheese;
  var hero;
  var rat;

  beforeEach(function(){
    rat = new Rat();
    cheese = new Food("Macaroni Cheese", 20);
    tuna = new Food("Tuna", 20);
    hero = new Hero("Chris", "Macaroni Cheese");
  })

  it("should be able to eat food and health goes up", function(){
    hero.eat(tuna);
    assert.strictEqual(hero.health, 120);
  })

  it("should be able to eat fav food and increase health by 1.5x", function(){
    hero.eat(cheese);
    assert.strictEqual(hero.health, 130);
  })

  it("should remove health from hero when poisoned", function(){
    rat.touch(cheese);
    hero.eat(cheese);
    assert.strictEqual(hero.health, 70);
  })
})
```

```
21 passing (17ms)
2 failing

1) Test interaction between food and hero should remove health from hero when poisoned:

    AssertionError [ERR_ASSERTION]: 130 === 70
    + expected - actual

    -130
    +70

    at Context.<anonymous> (specs/food_hero_integration_spec.js:33:12)

2) Test interaction between rat and food should be able make food poisonous:

    AssertionError [ERR_ASSERTION]: false === true
    + expected - actual

    -false
    +true

    at Context.<anonymous> (specs/rat_food_integration_spec.js:11:12)


npm ERR! Test failed.  See above for more details.
➜  homework git:(master) ×
```






homework — user@CODECLAN059 — -zsh — 102×41
..14_classnotes -z...
npm node ...
..1/express_app
..k_10/homework
Test interaction between food and hero
✓ should be able to eat food and health goes up
✓ should be able to eat fav food and increase health by 1.5x
✓ should remove health from hero when poisoned
Test food constructor
✓ should have a name
✓ Should have a replenishment value
Hero Tests
✓ should have a name
✓ should have default health to 100
✓ should have a favourite food
✓ should be able to say their name
✓ should start with empty task array
Test interaction between rat and food
✓ should be able make food poisonous
Test interaction between hero and task
✓ should be able to add a task to hero
✓ should be able to sort by difficulty
✓ should be able to sort by urgency
✓ should be able to sort by reward
✓ should be able to mark first task as complete
✓ should be able to complete multiple tasks
✓ Should be able to view all completed tasks
✓ Should be able to view all completed tasks
Task object tests
✓ should have a difficulty
✓ shoulc have an urgency
✓ should have a reward
✓ should not be completed when first created
23 passing (14ms)
➜ homework git:(master) ×