Evidence Gathering Document for SQA Level 8 Professional Developer Award.

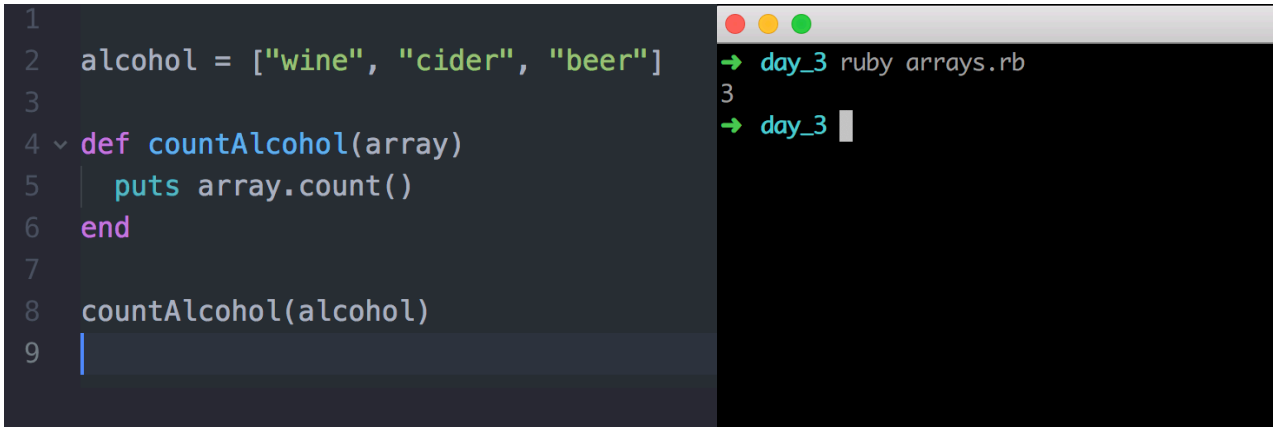
This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

Please fill in each point with screenshot or diagram and description of what you are showing.

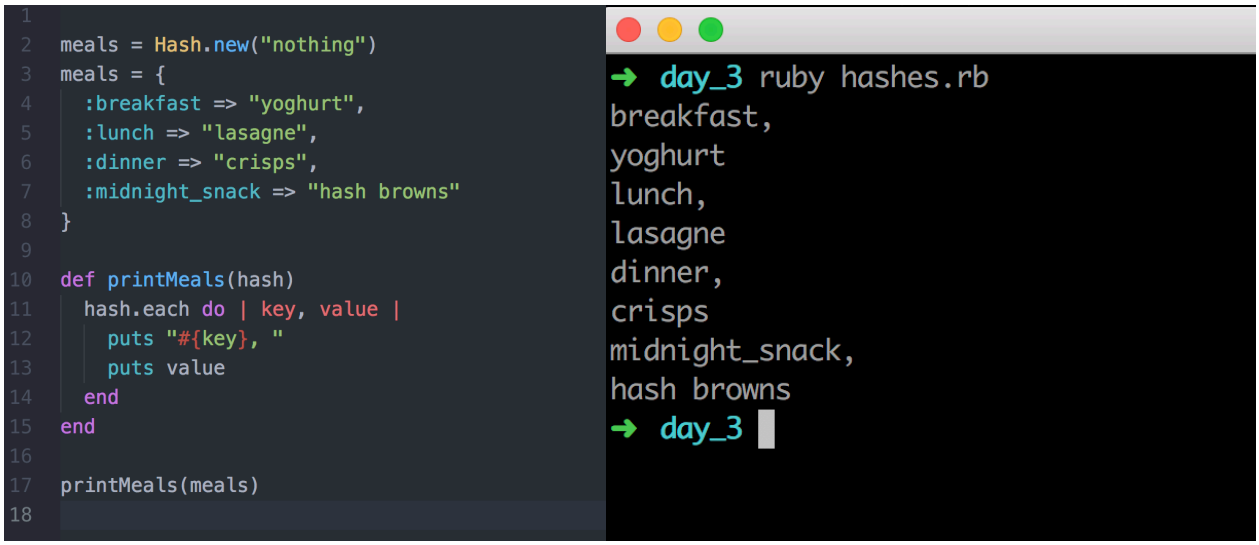
Week 2

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
		Description:



Left: An array in a program, a function using the array and the function call. The function is passed an array as an argument and it prints the number of elements in the array.
Right: The result of the function running

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
		Description:



```ruby
meals = Hash.new("nothing")
meals = {
  :breakfast => "yoghurt",
  :lunch => "lasagne",
  :dinner => "crisps",
  :midnight_snack => "hash browns"
}

def printMeals(hash)
  hash.each do | key, value |
    puts "#{key}, "
    puts value
  end
end

printMeals(meals)
```

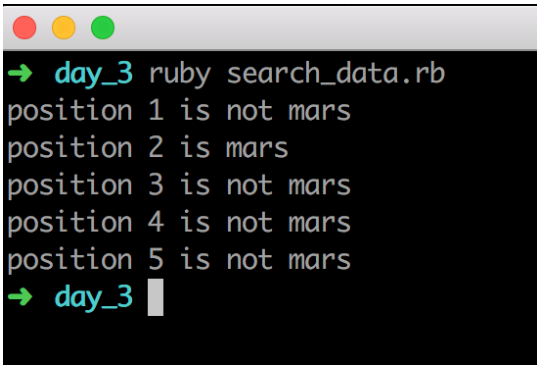
```
➜ day_3 ruby hashes.rb
breakfast,
yoghurt
lunch,
lasagne
dinner,
crisps
midnight_snack,
hash browns
➜ day_3
```

Left: A hash in a program, a function using the hash and the function call. The function is passed a hash, it loops over the items in the hash and prints the key and value for each item.
Right: The result of the function running

Week 3

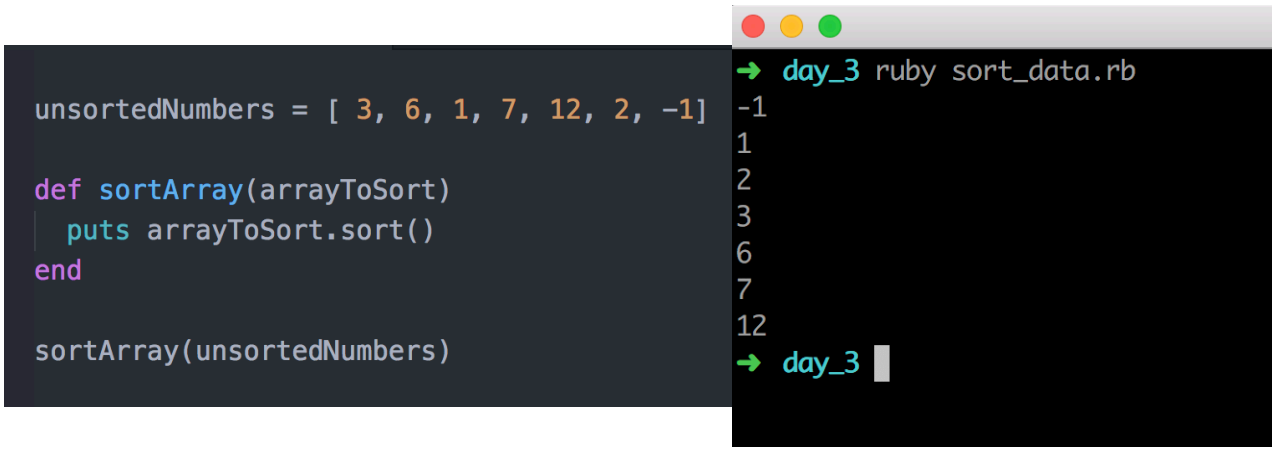
Unit	Ref	Evidence
I&T	I.T.3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running
		Description:

```ruby
planets = ["earth", "mars", "neptune", "saturn", "venus"]

def searchArrayForElement(array, searchTerm)
  i = 0;
  for element in array
    i += 1;
    if element == searchTerm
      puts "position #{i} is #{searchTerm}"
    else
      puts "position #{i} is not #{searchTerm}"
    end
  end
end

searchArrayForElement(planets, "mars")
```



Left: A function searching data and the function call. The function is passed an array and a search term, it loops over the elements in the array and prints whether or not the element matches the search term.
Right: The result of the function running

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
		Description:



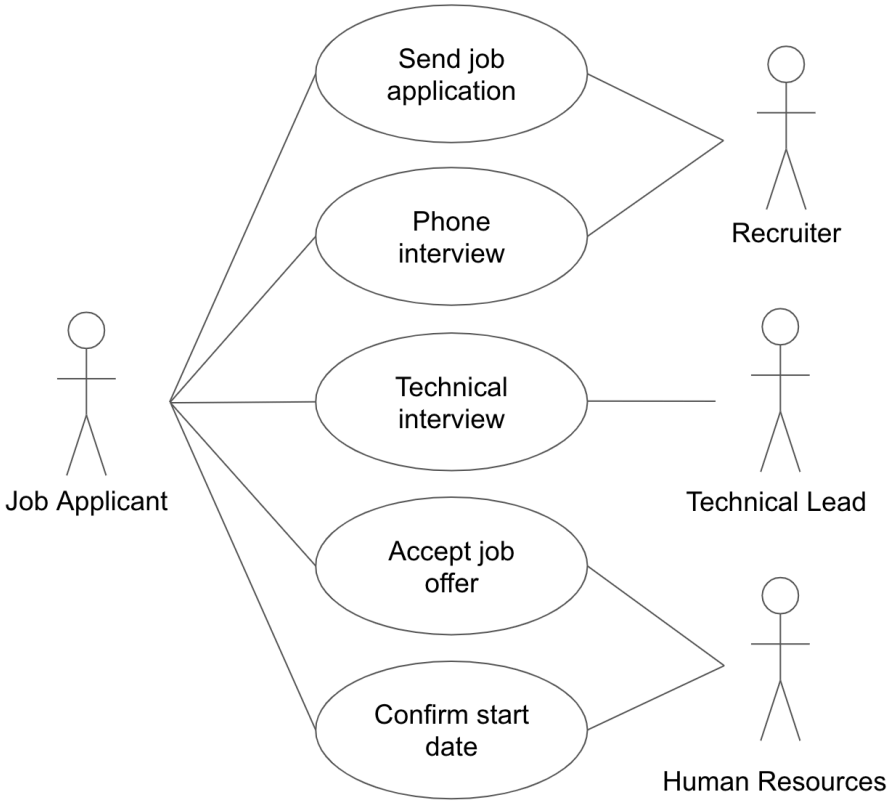
```ruby
unsortedNumbers = [ 3, 6, 1, 7, 12, 2, -1]

def sortArray(arrayToSort)
  puts arrayToSort.sort()
end

sortArray(unsortedNumbers)
```

```
➜ day_3 ruby sort_data.rb
-1
1
2
3
6
7
12
➜ day_3
```

Left: A function sorting data and the function call. The function is passed an array and it prints the sorted version of the array using Ruby's sort method.
Right: The result of the function running

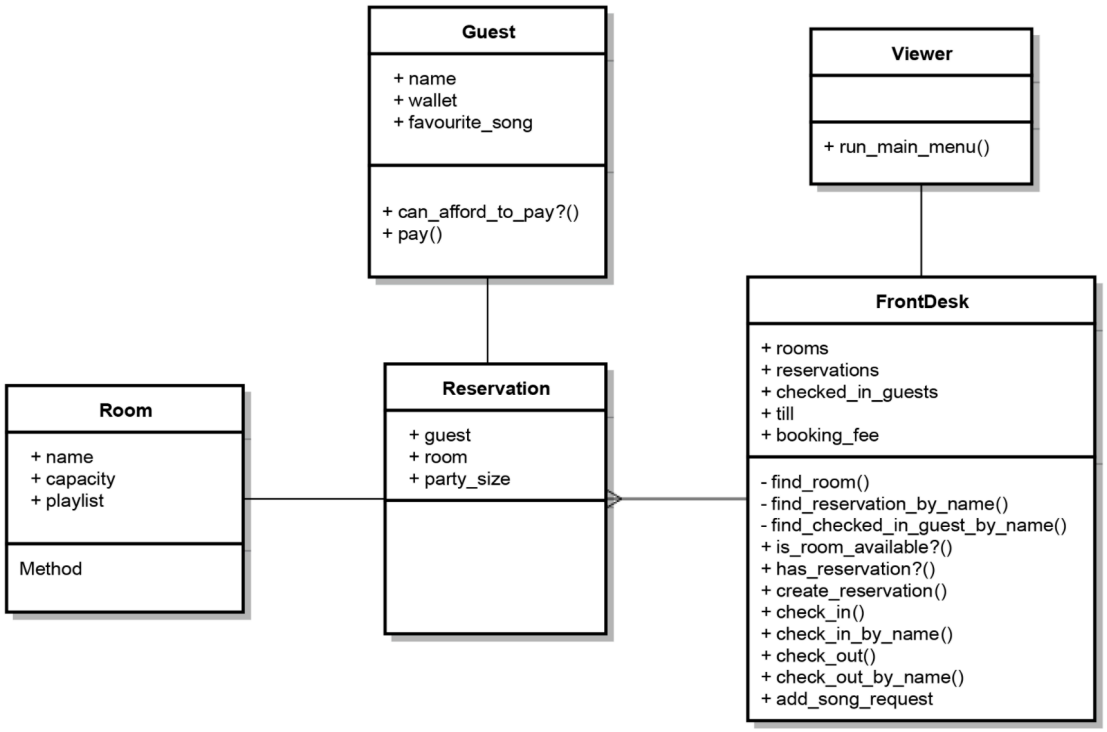
Week 5 and 6

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



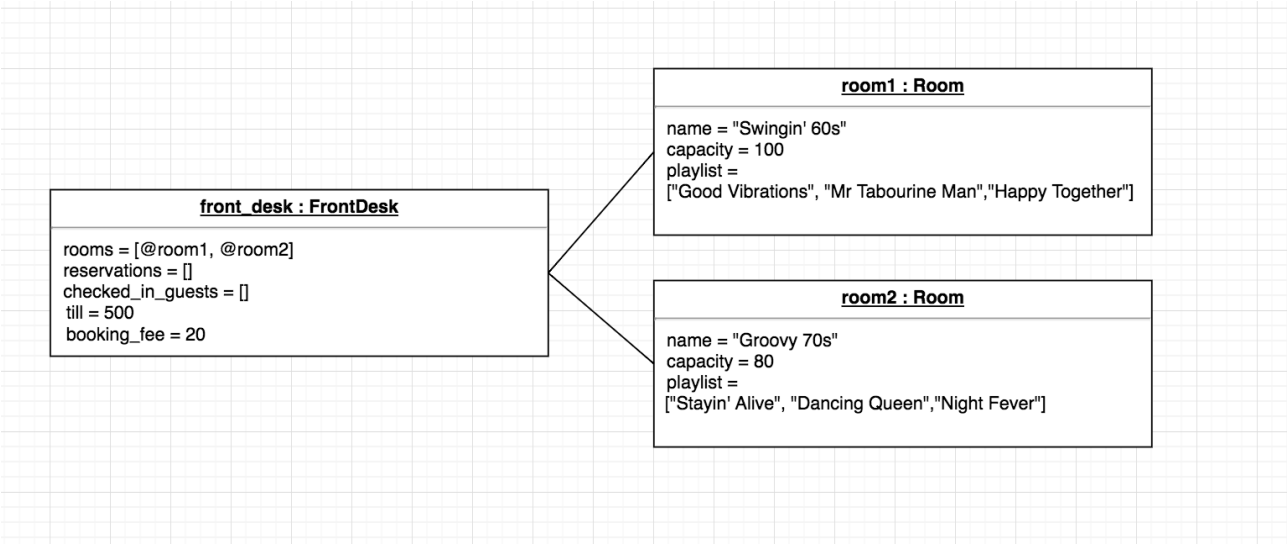
The diagram shows a system where a job applicant applies for a job. The actors are the job applicant, the recruiter, the technical lead and human resources.

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



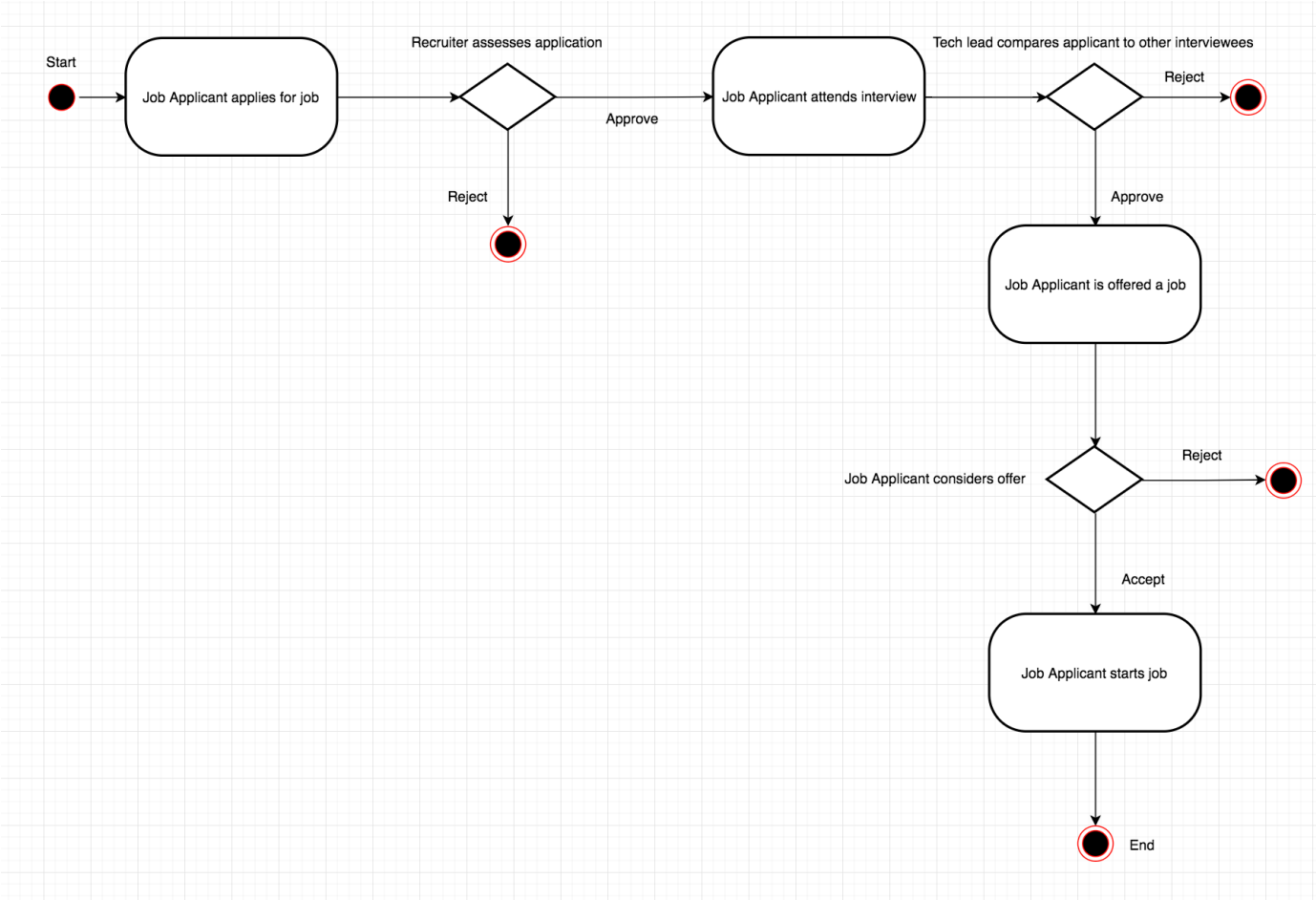
The above diagram shows the relationship between classes in a reservation system. The - and + show whether an item is private or public. Viewer is the entry point into the system, and FrontDesk is the class with the most responsibility.

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



The above diagram shows the objects that were instantiated from the classes shown in the class diagram in the previous example. The Front Desk object has an array of rooms containing two Room objects.

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



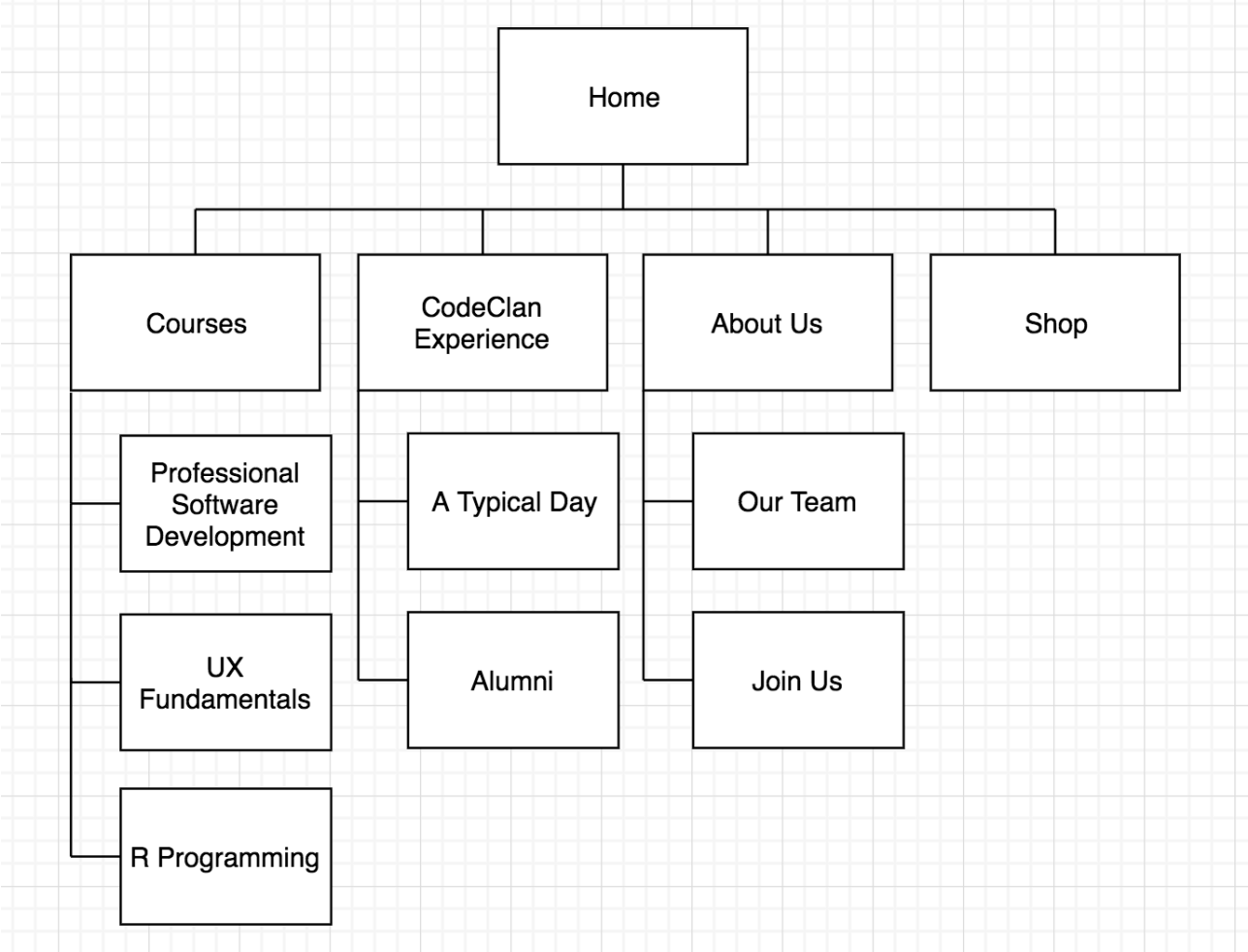
The above diagram shows the actions and decisions involved in a system where a job applicant applies for a job.

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
		Description:

Constraint Category	Implementation Constraint	Solution
Hardware and software platforms	The app must work on both desktop and mobile. The UI layout could mess up depending on the platform.	Design for mobile first, so all relevant data is always displayed. Use media queries to detect platform and make responsive CSS for the most common sizes.
Performance requirements	The app is free and may attract hundreds of thousands of users. We may not be able to handle high numbers of requests to the database. The app could end up slow during high traffic.	Use auto scaling servers and cache data where possible and legal to do so.
Persistent storage and transactions	We may not be allowed to store some data due to GDPR. If we store data incorrectly, the team could be in legal trouble.	Follow a guide to ensure only relevant data is being stored. Keep a note of when and where data is being stored and how long it is stored for.
Usability	This kind of product has never been released before. Users may not understand how to use the product. The UI could confuse them, and they might give up using it.	Test the UI frequently throughout the project. Gather as much data as possible to assess the intuitiveness of the UI and adapt the flow where necessary.
Budgets	There is no money available for the project as it's a proof of concept. The project could run into blockers as the cost of the scope is uncovered.	Use free, open source third party libraries and APIs where possible. Be prepared to change initial planned solutions to accommodate this.
Time	The project is only a week long. There is a high risk that the project could have unfinished parts, or lots of bugs in production. Either of these things could lead to the project being unusable.	Work iteratively, making sure that each component is tested and complete before moving onto another component, thereby ensuring the project can be shipped at any point.

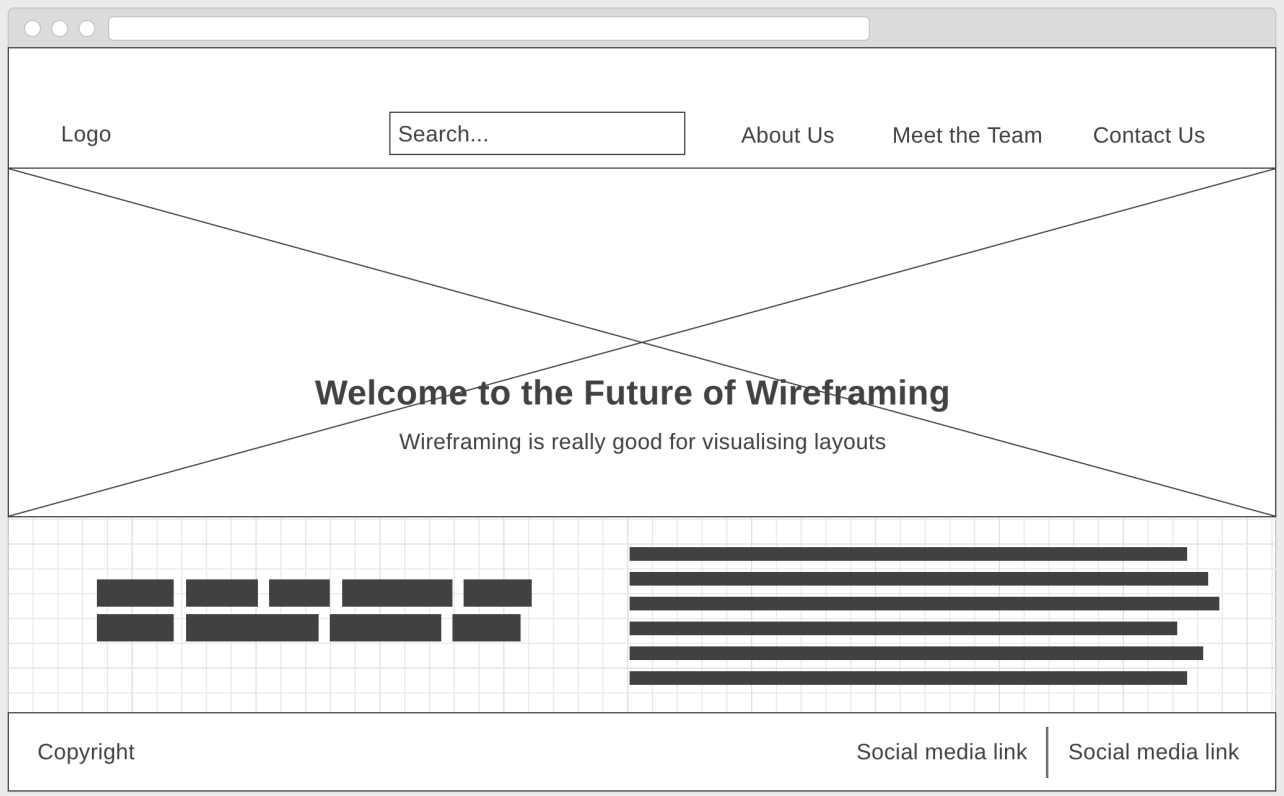
Above is a plan detailing project constraints and mitigations for risk.

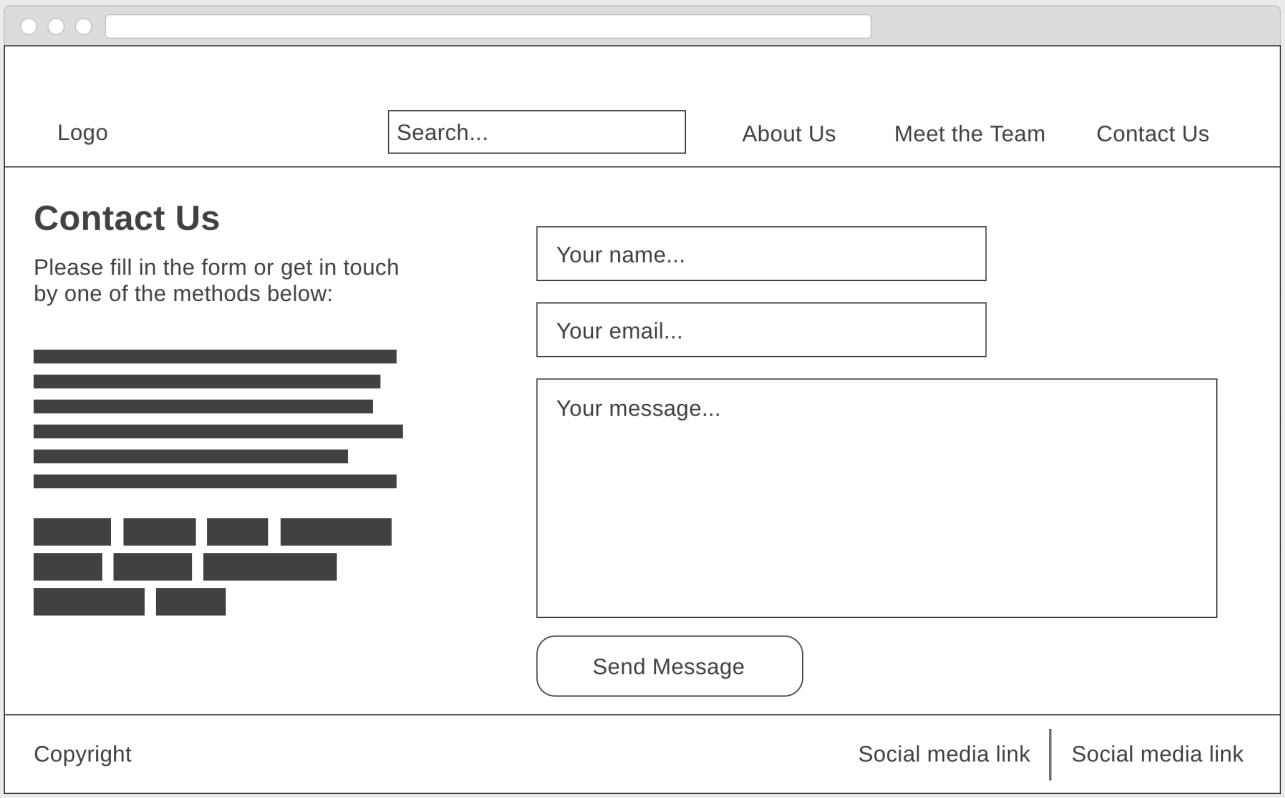
Unit	Ref	Evidence
P	P.5	User Site Map
		Description:



The above diagram shows a user site map of part of the CodeClan website. It starts on the landing page (Home), then branches into the primary navigation, then the secondary navigation.

Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams
		Description:





Above are two wireframes, the first suggesting a layout for a Home page and the second suggesting a layout for a Contact Us page.

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

```
def verifyUsernameAndPassword(username, password)
  # if username exists in the users table
  # and that user is active (e.g. not banned)
  # and the password matches the password for that user
  # then allow that user to log in with these credentials
  # and display the logged-in dashboard for the credentials
  # else display an inline error
  # and clear the form
end
```

The above pseudocode shows the steps involved in checking whether or not a provided username and password match entries in a 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
		Description:

ScotlandPHP
powered by Ruby

Home > Admin Dashboard > Speakers > Edit speaker

First Name:
John

Last Name:
Smith

Job title:
Senior Developer

Organisation:
Oracle

Bio:
Lorem ipsum dolar sit amet

Email:
john_smith@oracle.net

Twitter: @
john_smith

LinkedIn:
john_smith

Image URL:
https://images.pexels.com/photos/220453/pexels-photo-220453.jpeg?auto=compress&cs=

Cancel **save changes**

ScotlandPHP
powered by Ruby

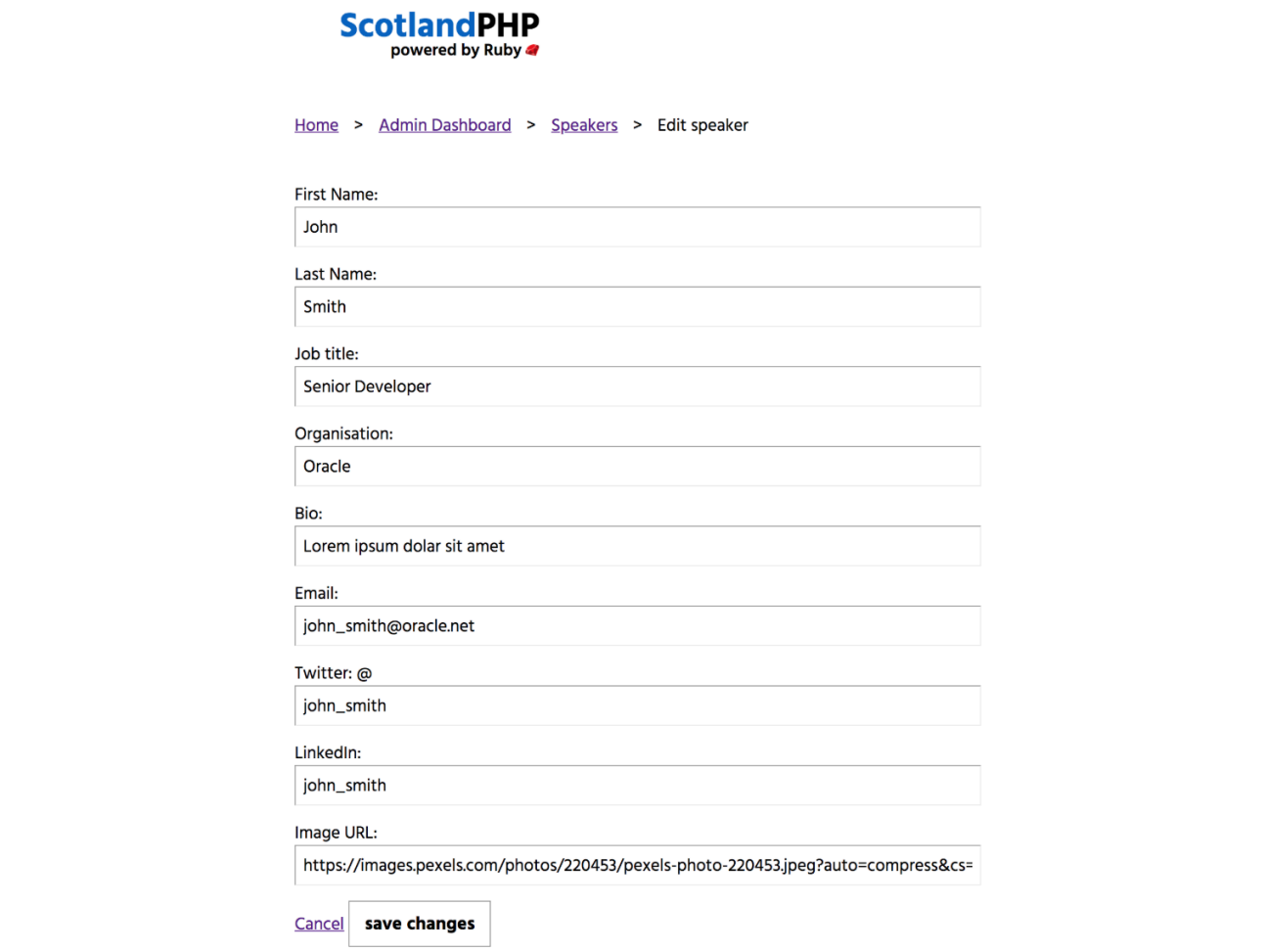
Home > Admin Dashboard > Speakers

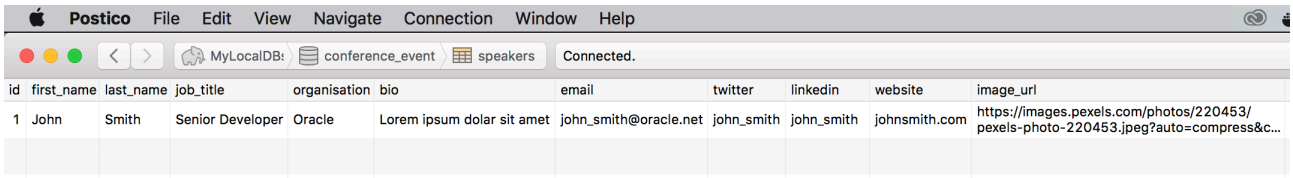
You have successfully updated John Smith! ×

John Smith
Senior Developer at Oracle
Edit speaker
delete speaker

The above (left) shows a user inputting details into a form, and above (right) shows the result of pressing save and being taken to another page.

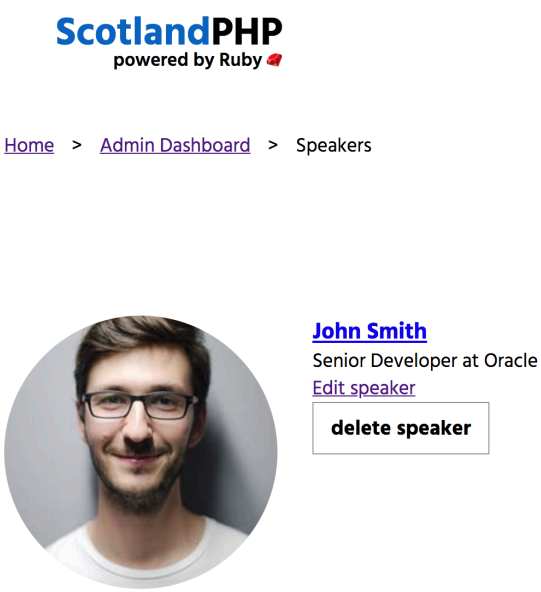
Unit	Ref	Evidence
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved
		Description:





Above is a screenshot of a user inputting details into a form and saving them to a Postgres database, which is being viewed via Postico.

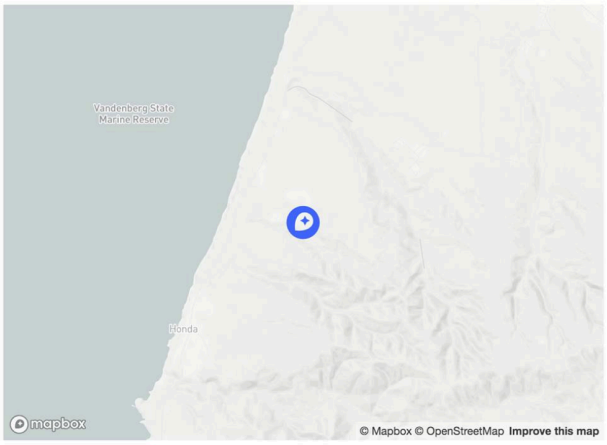
Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program
		Description:

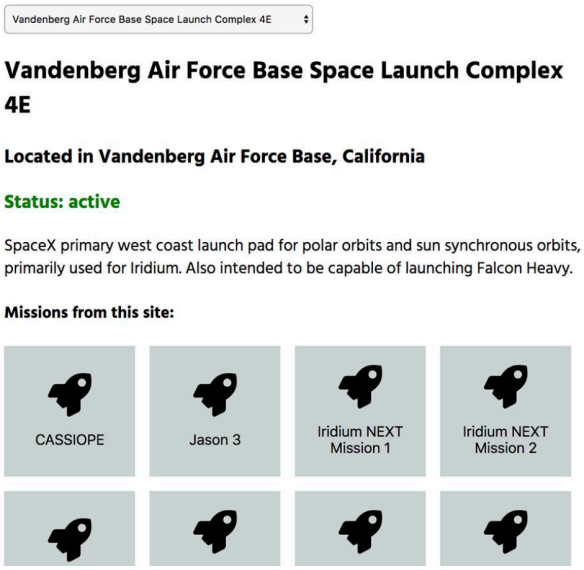


Above (left) shows the screen where a user can select to delete a speaker from the system. Above (right) shows the output after the action of deleting the speaker was performed.

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

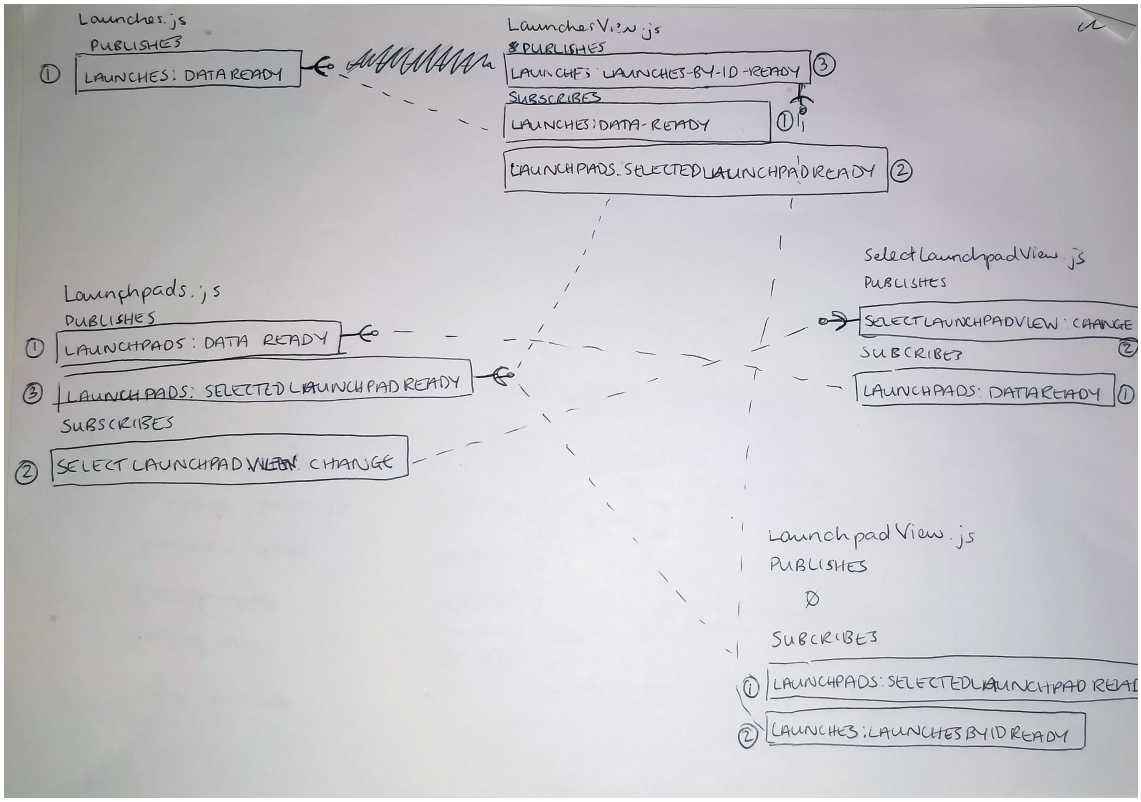
SpaceX Launch Sites

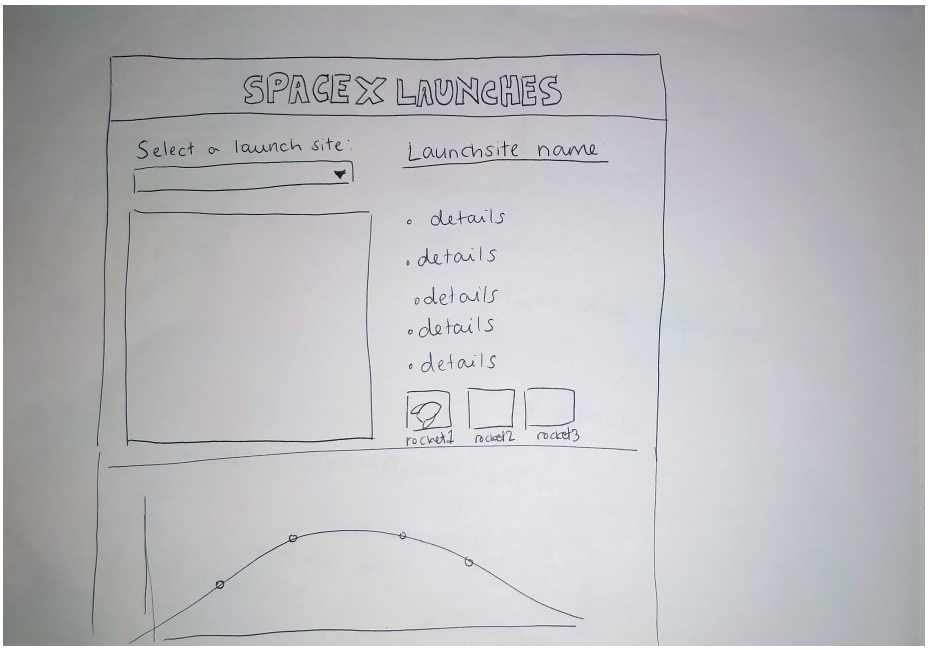


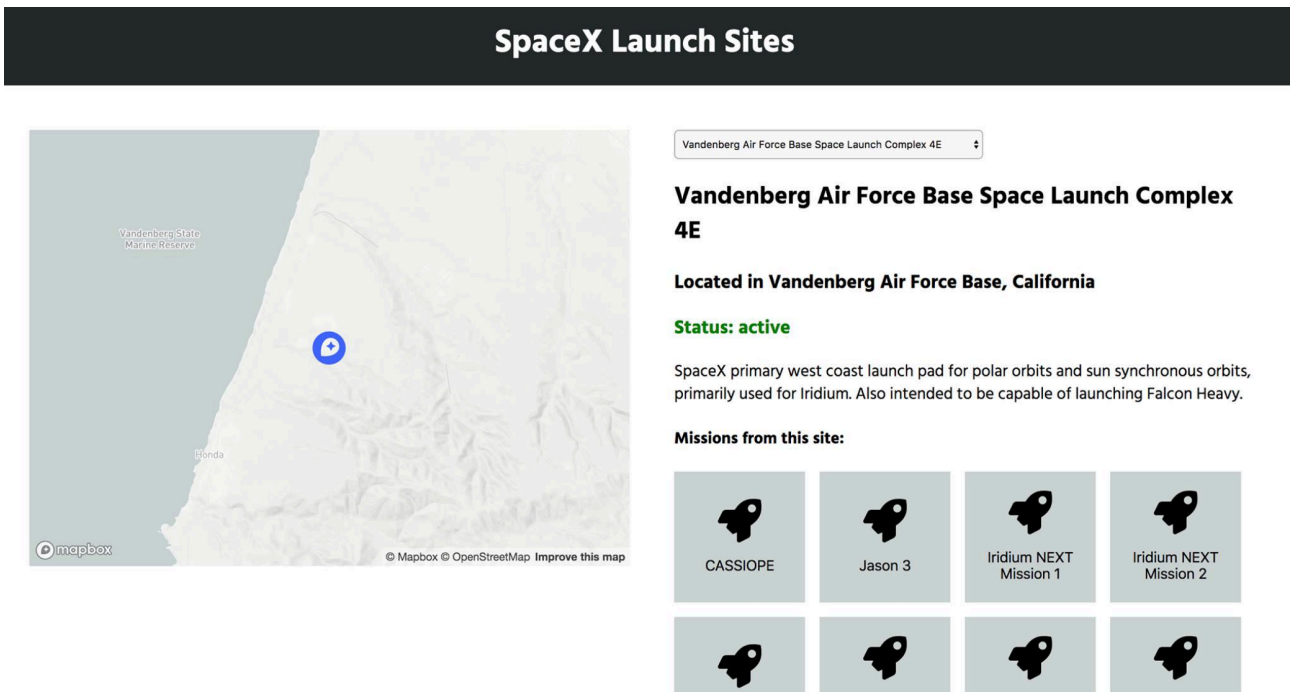


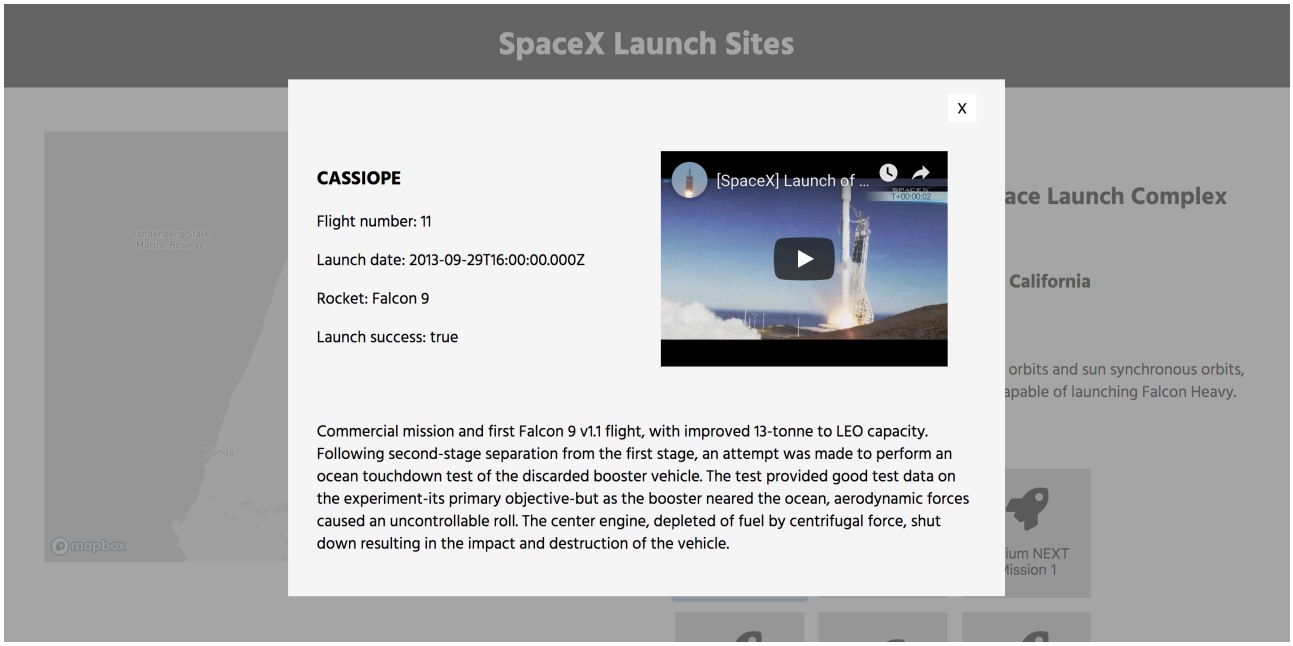
A JS project using external APIs and reusable views:
https://github.com/vickyjackson/wk12_d5

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









The above images show the planning involved in architecting the pubsub structure of the program, and the original sketch of what the site would look like. The layout and content of the site changed during the build, and some extra content was added into a popup.

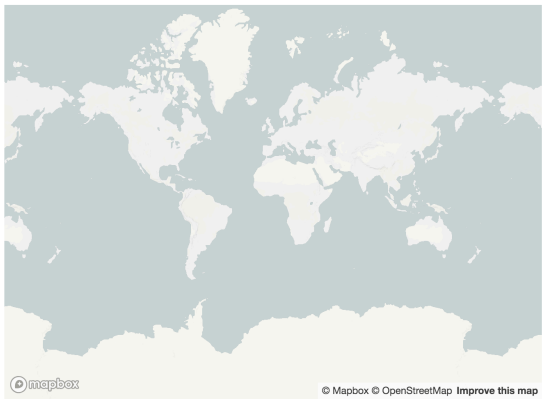
Week 7

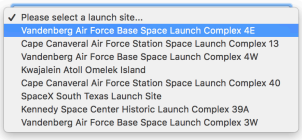
Unit	Ref	Evidence
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running
		Description:

```
const RequestHelper = function (url) {
  this.url = url
}

RequestHelper.prototype.get = function () {

  return fetch(this.url)
    .then(response => response.json());
}
```

```
LaunchPads.prototype.getData = function () {
  const url = `https://api.spacexdata.com/v2/launchpads/`;
  const request = new RequestHelper(url);

  request.get()
    .then((data) => {
      this.data = data;
      PubSub.publish('LaunchPads:data-ready', this.data);
    })
    .catch((err) => {
      console.log(err);
    })

}
```

SpaceX Launch Sites



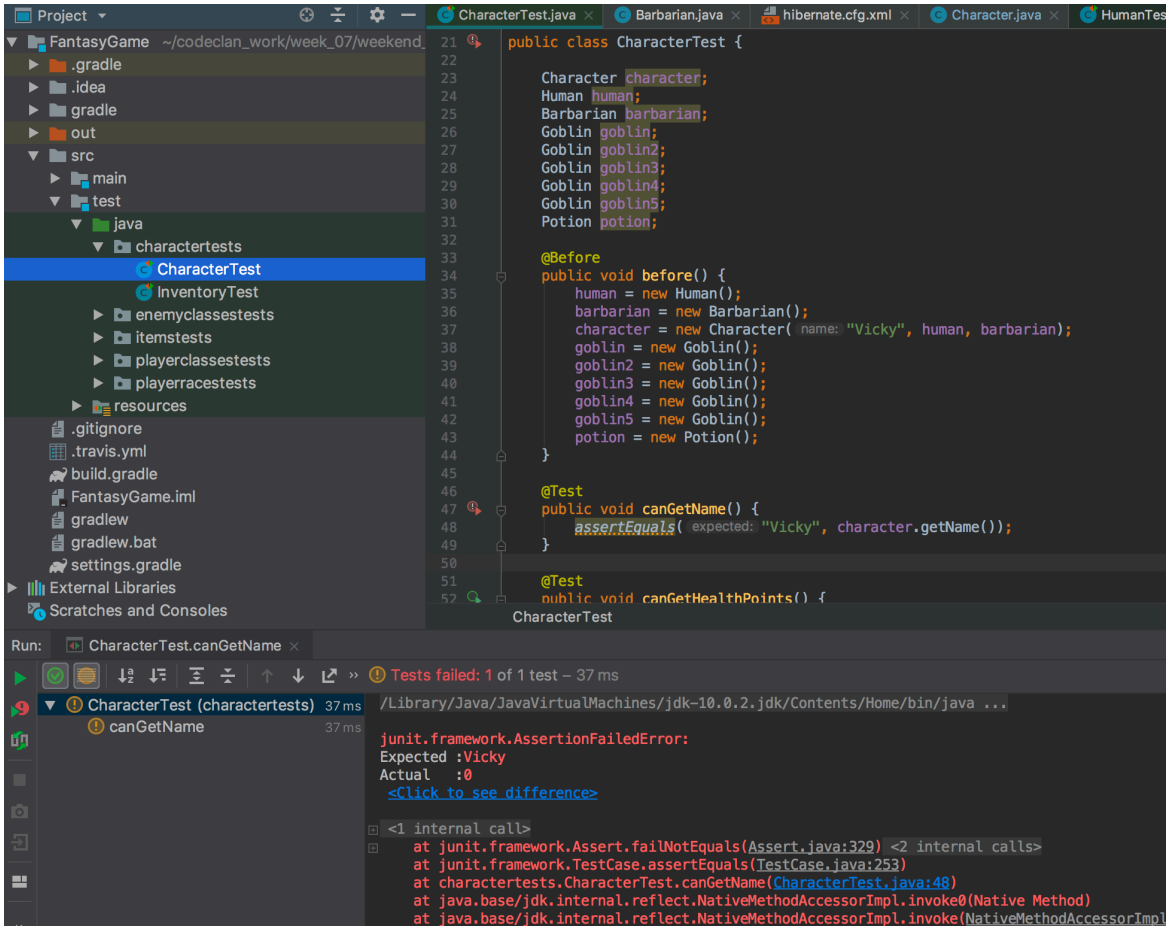


The above images show the code interacting with the API, the user selected an item from a dropdown which is populated by the API request.

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
		Description:

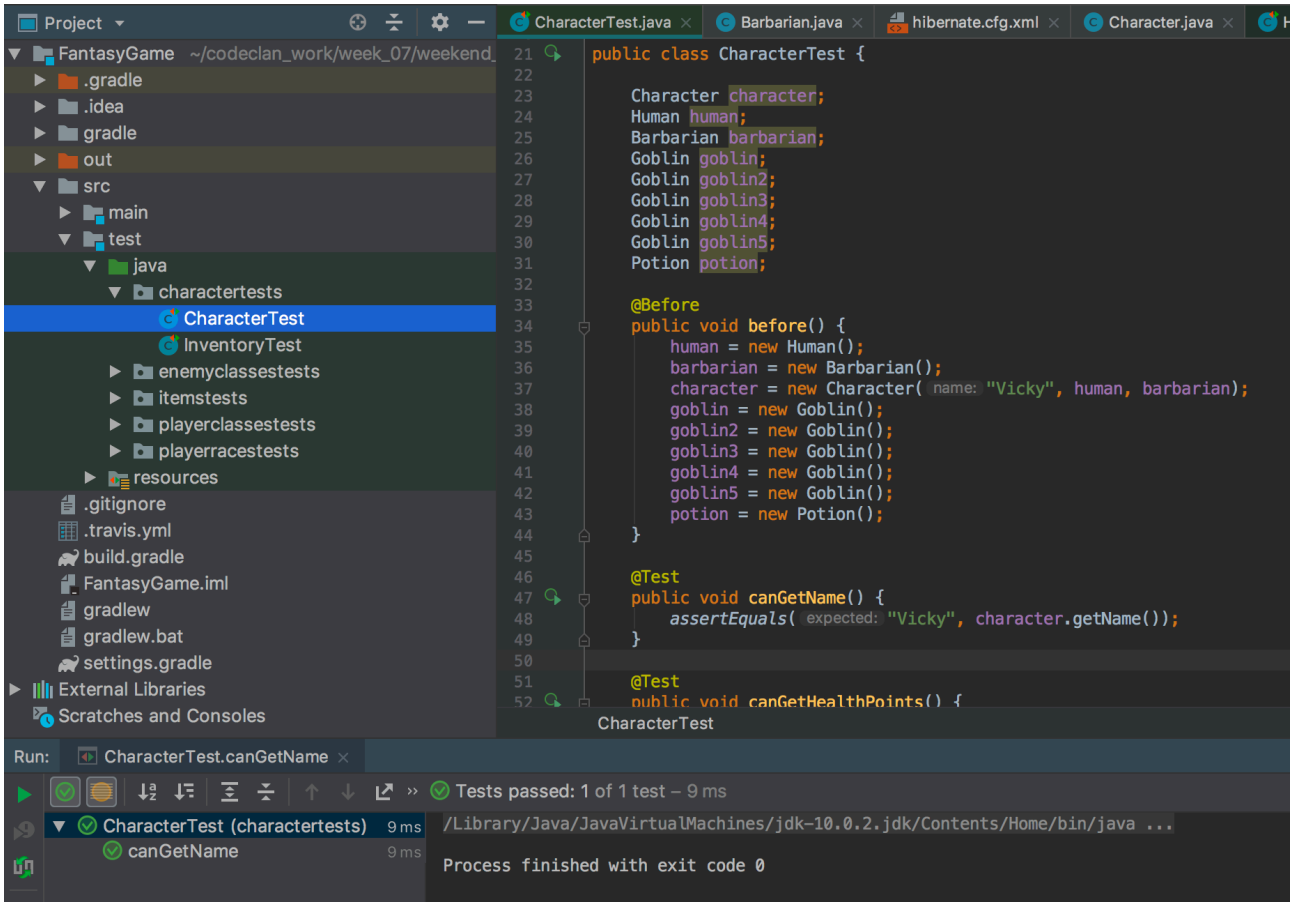
```
public Character(String name, IPlayerRace playerRace, IPlayerClass playerClass) {
    this.name = name;
    this.healthPoints = 100;
    this.playerRace = playerRace;
    this.playerClass = playerClass;
    this.inventory = new Inventory();
    this.level = LevelType.LEVEL_01;
    this.experiencePoints = 0;
    this.weapon = playerClass.getWeapon();
    addTitles();
}
```

```
@Column(name="name")
public int getName() {
    return id;
}
```



```java
public Character(String name, IPlayerRace playerRace, IPlayerClass playerClass) {
    this.name = name;
    this.healthPoints = 100;
    this.playerRace = playerRace;
    this.playerClass = playerClass;
    this.inventory = new Inventory();
    this.level = LevelType.LEVEL_01;
    this.experiencePoints = 0;
    this.weapon = playerClass.getWeapon();
    addTitles();
}
```

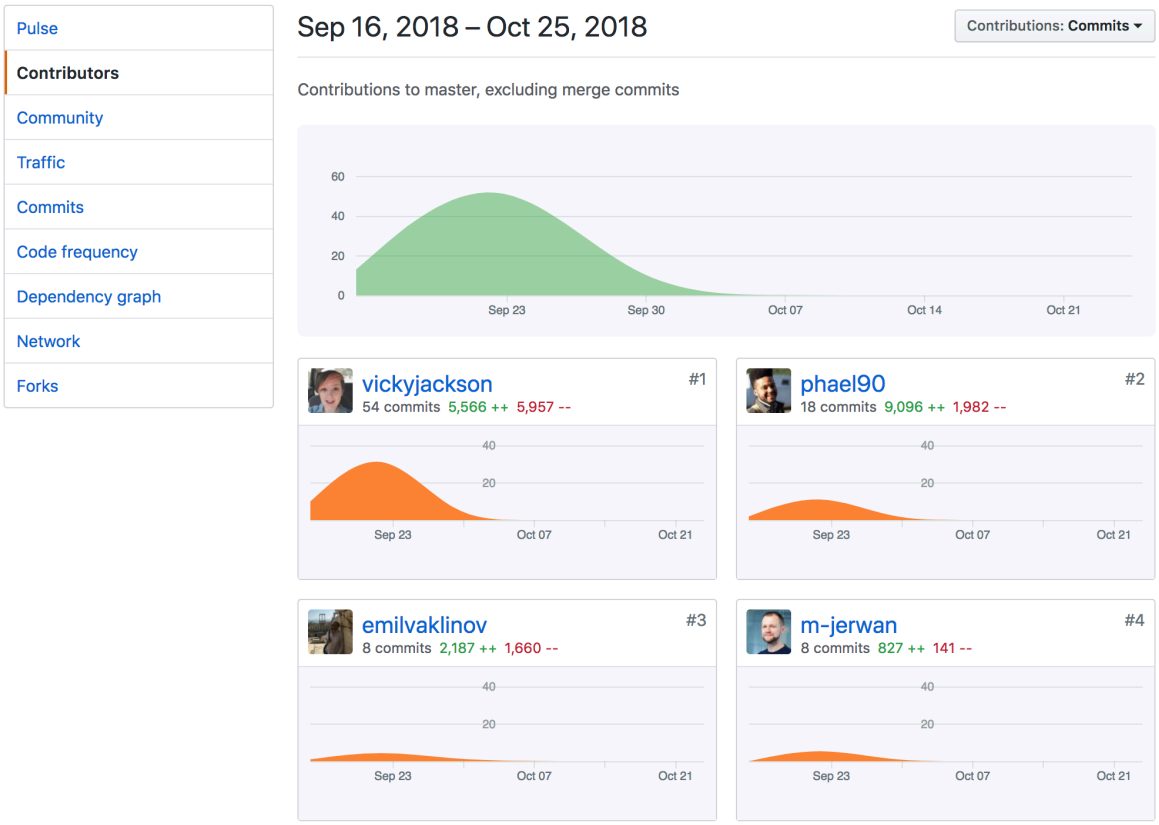
```java
@Column(name="name")
public String getName() {
    return name;
}
```



The first three images show code that fails a test. The test expects a String name to be returned from the getName method, but gets a 0 instead. The next three images show the code passing the test.

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.
		Description:



The contributors page for our final group project, with my team members: Raphael, Emil and Marcin.

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

CodeClan Collaborative Project - Tweet analysis app

We've been approached by an organisation that would like the ability to find out tweet frequency or sentiment by location. We will build a Minimum Viable Product that will retrieve data from Twitter and display it to the user, then explore ways to analyze and display meaningful data in an interesting way.

MVP

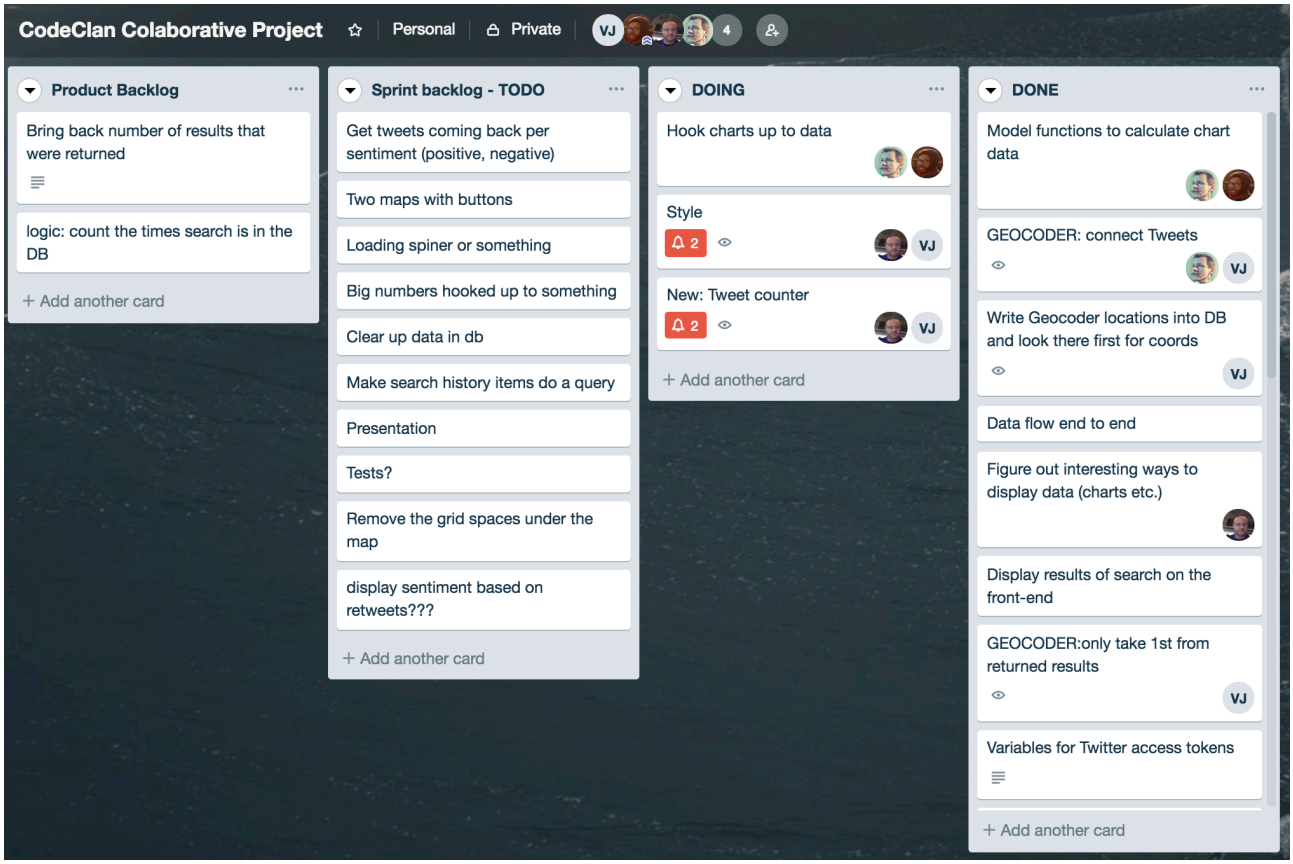
- Be able to fetch tweets from the Twitter API
- Be able to filter the data retrieved from the API
- Be able to display filtered API data on the front end
- Be able to persist user input from the front end into the database
- Be able to retrieve data from the database
- Be able to display the data from the database on the front end
- Practice PubSub and MVC design patterns

Possible Extensions

- Implement a mapping library with basic functionality (points on map)
- Try to use the map library to display heatmaps of tweet data by geolocation (dependant on the usefulness of the data retrieved)
- Explore other libraries for analysing twitter data (i.e tweet sentiment, gender analysis)
- Display the analysis on the front end
- Explore chart libraries to find other interesting ways to display data on the front end

Above is the project brief for our final group project. We created our own brief based on the consolidating lessons, which was approved by the tutors.

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



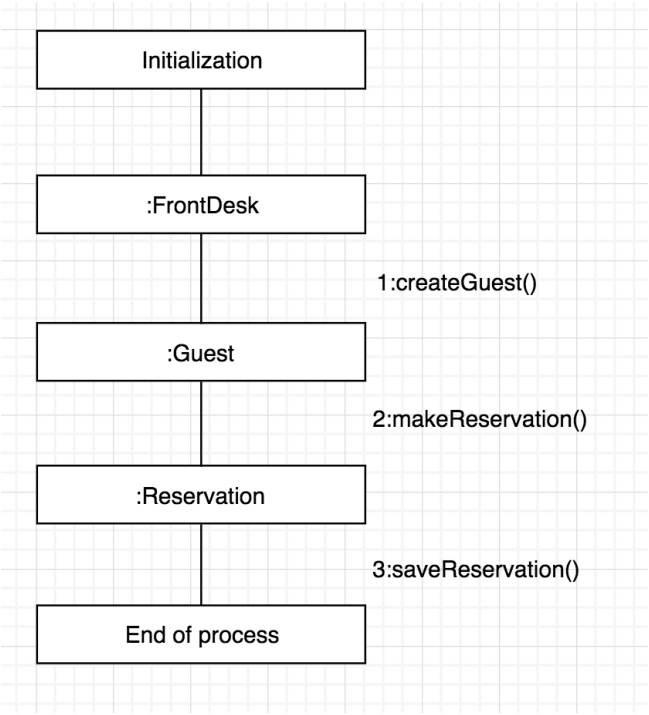
Above is a screenshot of our Trello board for our final group project. We had three columns, To Do, Doing and Done. We also had a product backlog where we put all of our ideas before they were reviewed and accepted as work items.

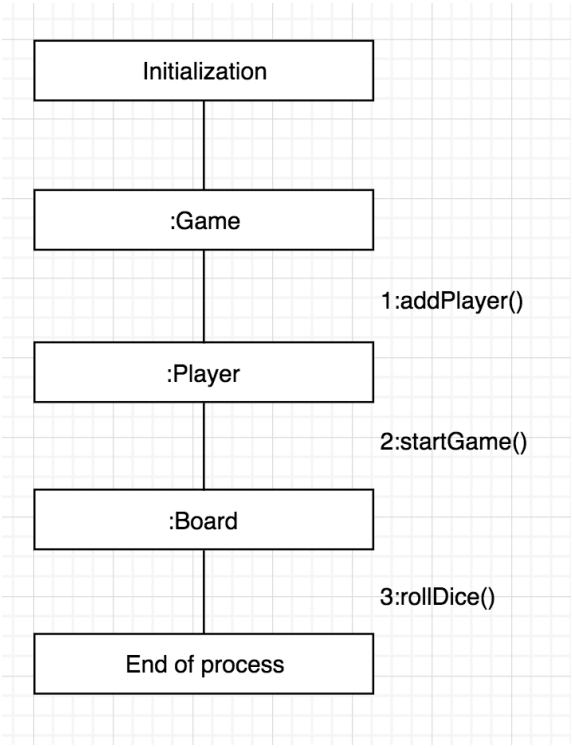
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

Acceptance Criteria	Expected result / output	Pass/Fail
A user is able to add to the quantity of items in a shopping basket	The quantity value increments by 1 when the user adds 1 to an existing item in the basket and clicks Save. The change is recorded in a database.	Pass
A user is able to subtract from the quantity of items in a shopping basket	The quantity value decrements by 1 when the user removes 1 from an existing item in the basket and clicks Save. The change is recorded in a database.	Pass
A user is able to delete all items from a shopping basket	The items in the basket will be cleared when the user selects to delete all items. The change is recorded in a database. The user will be given feedback that the action was successful.	Pass
A user is able to see the total cost of all the items in the basket	The total cost value will reflect the cost of all the items costs added together. When an item is added or removed, the total cost will reflect this change. The change is recorded in a database.	Pass
A user is able to navigate away from and back to the basket without losing the state of the basket	The basket state persists when the user clicks away from the basket page. If the basket contains 2 items, and the user visits another page and then returns, the basket will still contain 2 items. When the page is loaded, the data will be fetched from the database.	Pass

Above is an example of acceptance criteria that you would find for shopping cart functionality, the test steps to be carried out to make sure the criteria is met and a note of whether or not those tests passed.

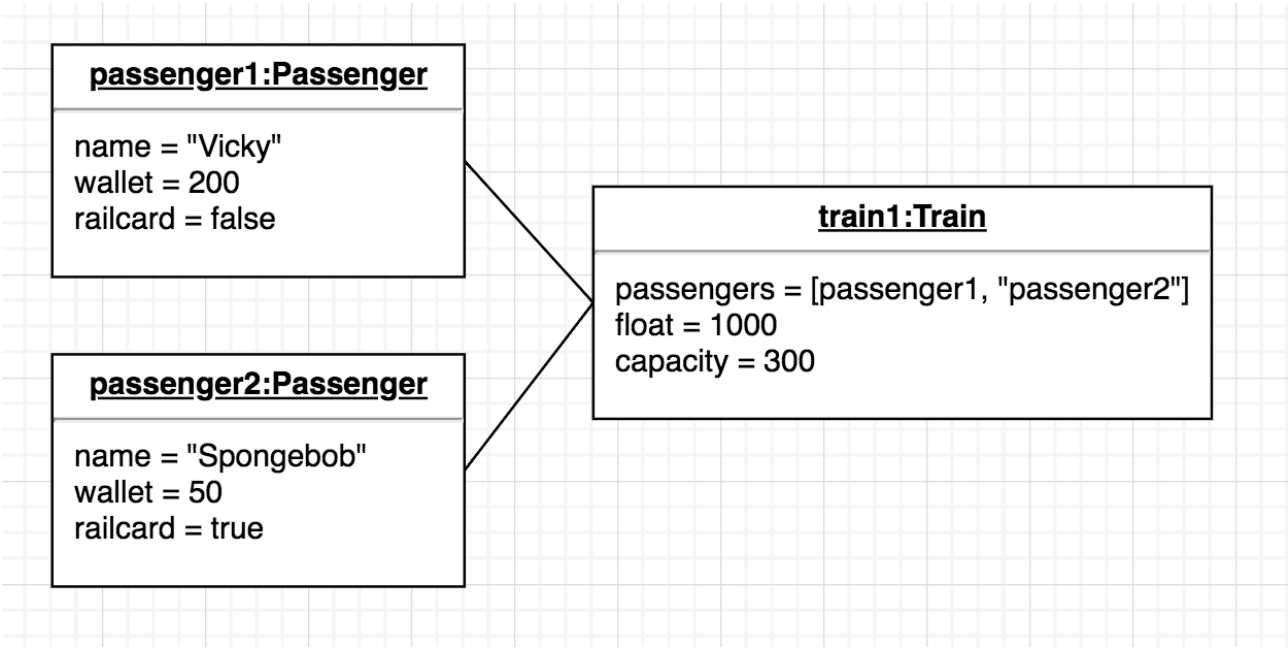
Unit	Ref	Evidence
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).
		Description:

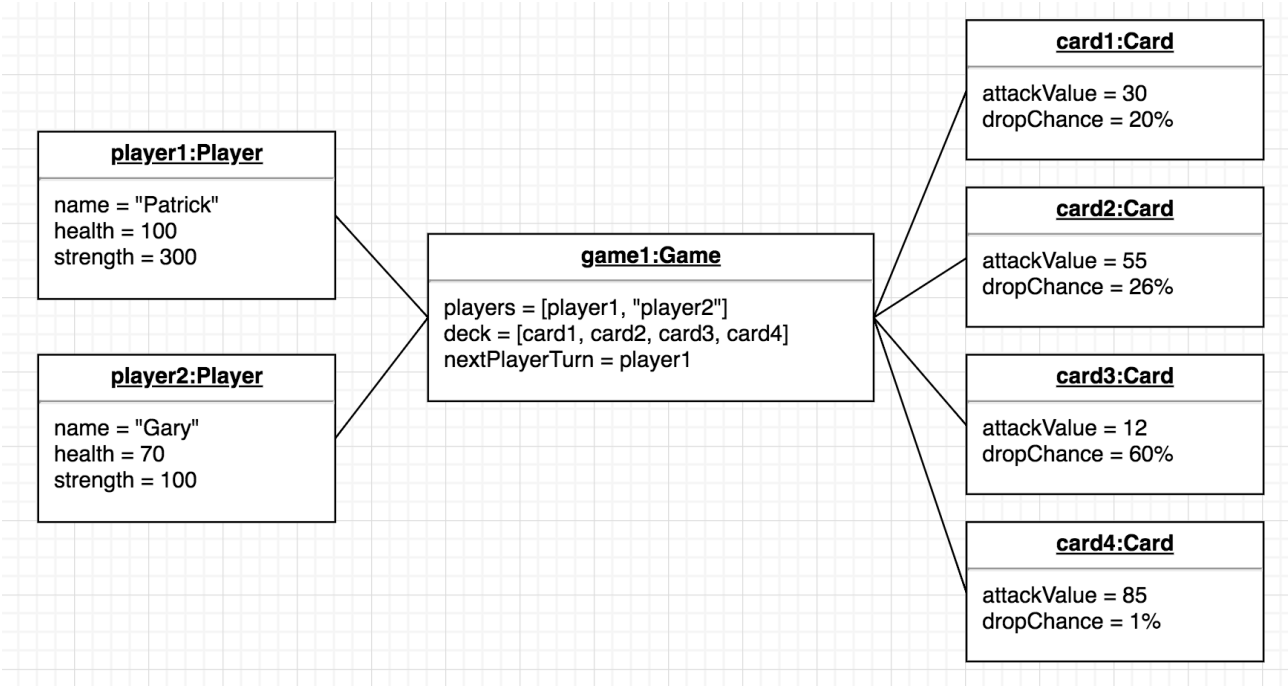




Above are two examples of collaboration diagrams. The first one shows the relationship between objects in a reservation system, and the second shows the relationship between objects in a board game.

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





Above are two examples of relationships between objects. The first is a system where passengers board a train. The second is a system where players play a game of cards.

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

Bug / Error	Solution	Date
User is not able to add to the quantity of items in a basket. When they hit save, a server 500 error is produced.	Update the database credentials so they are correct and the request to update it can complete. They were rotated without us noticing!	12/10/18
A user is not able to subtract from the quantity of items in a shopping basket. When they try to edit it, they find they can't click on it.	There is an invisible div sitting on top of the edit value element. Remove the div to be able to access the element underneath.	15/10/18
A user is unable to delete all items from a shopping basket. All items are deleted but one.	The condition of the loop that checks if there are items is set to check for more than one item. It should be checking for more than zero items.	17/10/18
A user is not able to see the total cost of all the items in the basket. They see a number, but it isn't correct.	There is a problem with the calculation where a deleted item's cost is persisting in the total value. Make sure that when an item is deleted, the cost of the item is subtracted from the total.	17/10/18
A user is not able to navigate away from and back to the basket without losing the state of the basket. When they come back to the page, all the items that were there are gone.	The data is not persisting to the database. It looks like we forgot to hook it up. Make sure a request is being made to the database when items are added, removed or updated on the page.	21/10/18

The above spreadsheet shows an example of a bug report with some bugs listed for a shopping basket page. It describes the bugs, their solutions and the date they were fixed.

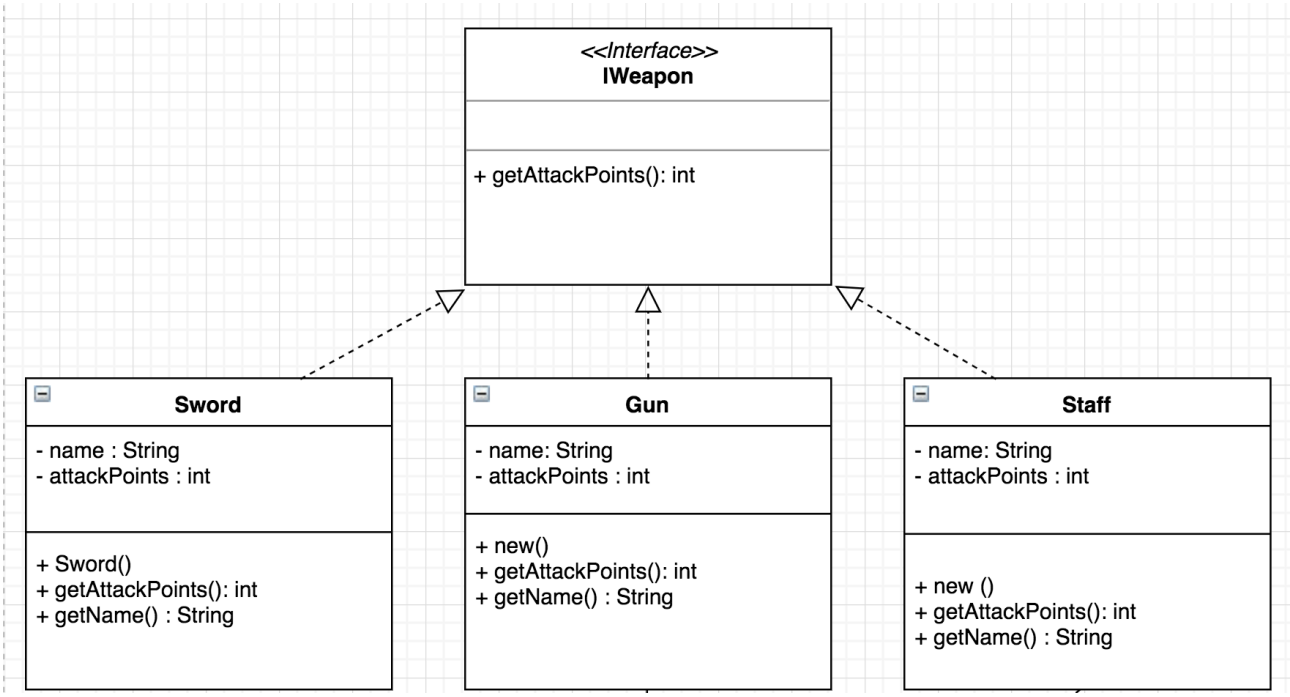
Week 12

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

```java
public class Student extends Person {

    public Student(String name, String cohort) {
        super(name, cohort);
    }

    @Override
    public String talk(String language) {
        return "I love learning " + language;
    }
}
```

Above is an example of polymorphism where the method “talk” (which is inherited from Person) is being overridden in the sub-class Student.

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



Above is an example of inheritance. The Sword, Gun and Staff classes are inheriting from the Interface IWeapon. Specifically, they inherit a method called getAttackPoints().

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

```ruby
require_relative('../db/sql_runner.rb')

# A class for creating, and reading talks
class Talk
  attr_reader :id
  attr_accessor :title, :abstract, :category_id

  # Pass in a hash for more flexibility
  def initialize(options)
    @id = options['id'].to_i unless options['id'].nil?
    @title = options['title']
    @abstract = options['abstract']
    @category_id = options['category_id']
  end

  # Create
  def save
    sql = "INSERT INTO talks (title, abstract, category_id)
    VALUES ($1, $2, $3) RETURNING id"
    values = [@title, @abstract, @category_id]
    result = SqlRunner.run(sql, values)
    @id = result[0]['id']
  end

  # Read
  def self.all
    sql = 'SELECT * from talks ORDER BY title, abstract ASC'
    result = SqlRunner.run(sql)
    talks = result.map { |talk| Talk.new(talk) }
    talks
  end
end
```

A class is an example of encapsulation. The above class has properties that are accessible via getters and setters (attr_reader and attr_accessor). It has an object method and a class method.

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.
		Description:

```java
package models.enemy;

public abstract class Enemy {

    private int healthPoints;
    private boolean isDead;
    protected int experiencePointsToAward;

    public Enemy() {
        this.healthPoints = 100;
        this.isDead = false;
        this.experiencePointsToAward = 0;
    }

    public int getHealthPoints(){
        return this.healthPoints;
    }

    public int getExperiencePointsToAward(){
        return experiencePointsToAward;
    }

    public String takeDamage(int attackPoints){
        this.healthPoints -= attackPoints;
        if (this.healthPoints <= 0){
            this.isDead = true;
            return "You killed the enemy!";
        }
        return "You hit the enemy and reduced its health by " + attackPoints + " points!";
    }

    public boolean isDead(){
        return this.isDead;
    }

}
```

```java
package models.enemyclasses;

import models.enemy.Enemy;

public class Goblin extends Enemy {

    public Goblin(){
        super();
        this.experiencePointsToAward += 25;
    }

}
```

```java
package enemyclassestests;

import ...

public class GoblinTest {

    Goblin goblin;
    @Before
    public void before(){
        goblin = new Goblin();
    }

    @Test
    public void canGetHealthPoints(){
        assertEquals( expected: 100, goblin.getHealthPoints());
    }

    @Test
    public void canGetExperiencePointsToAward(){
        assertEquals( expected: 25, goblin.getExperiencePointsToAward());
    }

    @Test
    public void canTakeDamage(){
        goblin.takeDamage( attackPoints: 50);
        assertEquals( expected: 50, goblin.getHealthPoints());
    }

    @Test
    public void canReturnDamageTaken(){
        assertEquals( expected: "You hit the enemy and reduced its health by 10 points!", goblin.takeDamage( attackPoints: 10));
    }
}
```

The first image shows the base class, Enemy. The second image shows the sub-class, Goblin, which inherits from Enemy. The last image shows the tests, which create a Goblin object and use information from the base class.

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.
		Description:

```
let highestPositiveNumber = 1;

function solution(arrayToCheck) {
    // write your code in JavaScript (Node.js 8.9.4)
    sortedArray = arrayToCheck.sort((a,b) => { return a-b });
    sortedArray.forEach((element) => {
        if (element === highestPositiveNumber){
            highestPositiveNumber = element +1;
        }
    })
    return highestPositiveNumber;
}
```

```
function isPangram(phrase){
  let alphabet = 'qwertyuiopasdfghjklzxcvbnm'.split('');
  return alphabet.every(letter => phrase.includes(letter));
}
```

Algorithm 1:

This algorithm looks through an array to find the first missing positive number.

The function is passed an array. It sorts the array alphabetically. For each element in the sorted array, if the element is the same as the highestPositiveNumber, the highestPositiveNumber is changed to equal the element value + 1. The new highestPositiveNumber is returned when the loop is complete.

Algorithm 2:

This algorithm looks through a given string to check if it contains every letter in the alphabet.

The function is passed a string. It creates an array with each letter in the alphabet as an element. For each element in the array, it checks that every letter in the array matches a letter in the string.