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.

Week 2

Unit	Ref	Evidence	
I&T	I.T.5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running	

```ruby
class Room

  def initialize(number, capacity)
    @number = number
    @guests = []
    @playlist = []
    @capacity = capacity
  end
```

```ruby
def fav_song(song_to_find)
  for song in @playlist
    if song == song_to_find
      return "Whoo"
    end
  end
  return "Not whoo"
end
```

```ruby
def test_fav_song
  @room.add_song(@song)
  assert_equal("Whoo", @room.fav_song(@song))
end
```

```
➜  specs git:(master) ruby song_spec.rb
Run options: --seed 51502

# Running:

..

Finished in 0.001019s, 1962.7086 runs/s, 1962.7086 assertions/s.

2 runs, 2 assertions, 0 failures, 0 errors, 0 skips
```

We create an empty array list for guests and playlist with a function that returns a string if the song is in the playlist or not, we show a test of the test passing in the terminal

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

```ruby
class TestPetShop < Minitest::Test

  def setup

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

    @new_pet = {
      name: "Bors the Younger",
      pet_type: :cat,
      breed: "Cornish Rex",
      price: 100
    }
```

```ruby
def add_pet_to_customer(customers, pets)
  customers[:pets].push(pets)
  customers[:pets].length
end
```

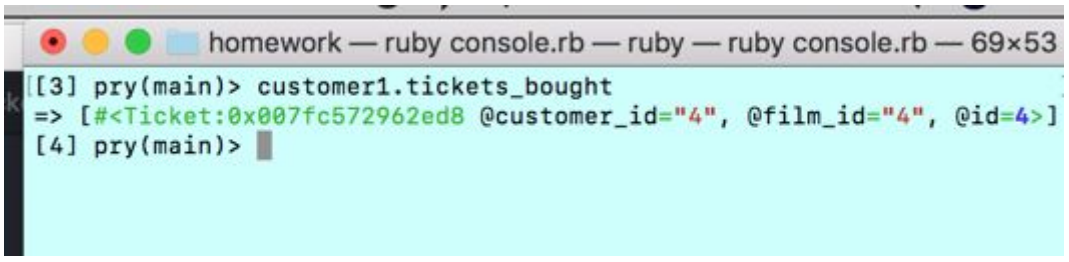
```
➜  specs git:(master) ✗ ruby pet_shop_spec.rb
Run options: --seed 36292

# Running:

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

Finished in 0.001667s, 10197.9604 runs/s, 10197.9604 assertions/s.

17 runs, 17 assertions, 0 failures, 0 errors, 0 skips
➜  specs git:(master) ✗
```

```ruby
def test_add_pet_to_customer
  customer = @customers[0]
  add_pet_to_customer(customer, @new_pet)
  assert_equal(1, customer_pet_count(customer))
end
```

I have created a hash for a customers in a pet shop with a name, pets and cash with a function to .push into pets array and using .length to see if that pet has been added.

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	




```ruby
def tickets_bought
  sql = "SELECT * FROM tickets WHERE customer_id = $1"
  values = [@id]
  results = SqlRunner.run(sql, values)
  return results.map {|result| Ticket.new(result)}
end
```

This is a method that selects all tickets from a specific customer and in the terminal we can see returned relevant data

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	

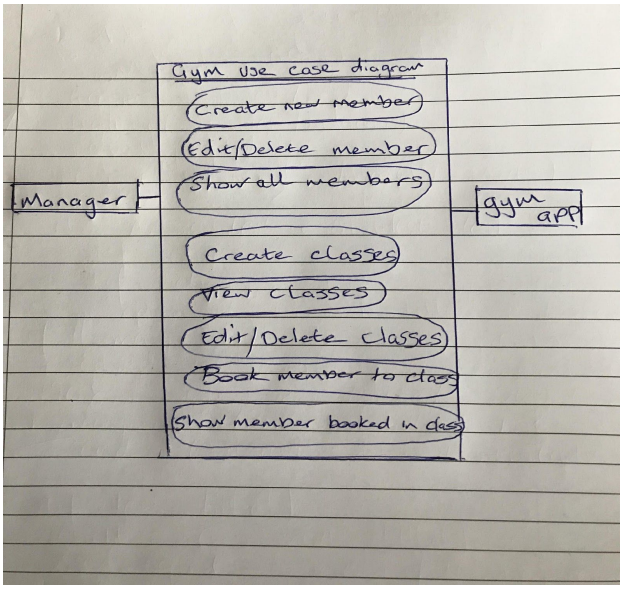
```ruby
def films()
  sql = "SELECT * FROM films INNER JOIN tickets ON film_id = films.id
  WHERE customer_id = $1 ORDER BY films.title"
  values = [@id]
   results = SqlRunner.run(sql, values)
  return results.map {|result| Film.new(result)}
end
```

```
[1] pry(main)> customer1.films
=> [#<Film:0x007f9c9d063a60 @id=67, @price=20, @title="Casino">,
 #<Film:0x007f9c9d063790 @id=64, @price=15, @title="Goodfellas">]
[2] pry(main)>
```

This method is returning all films from a specific customer by the customer id and in the terminal we can see the returned relevant data

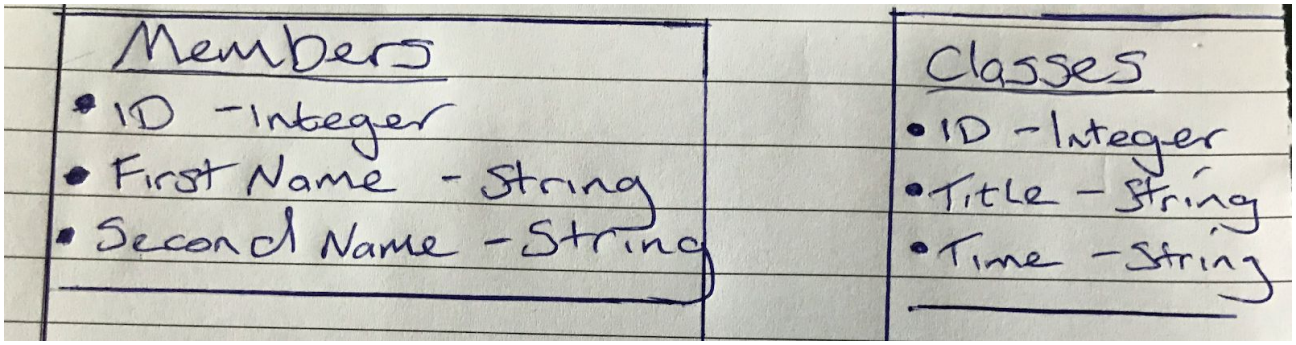
Week 5

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



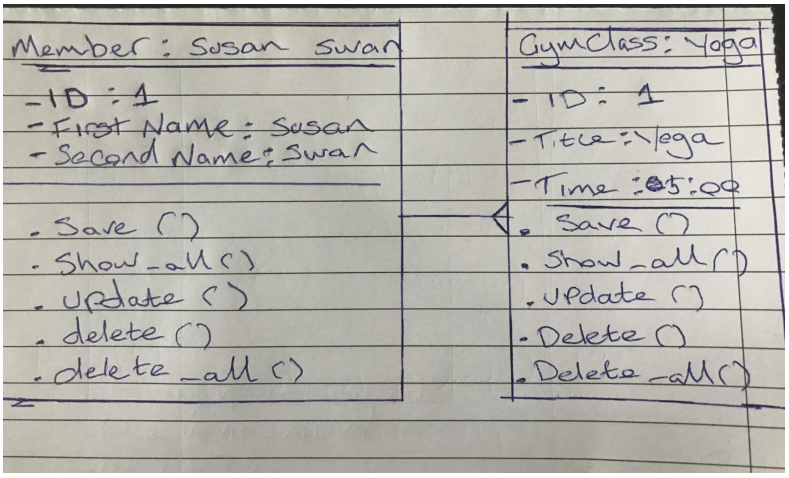
This shows what a manager should be able to do on the site.

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



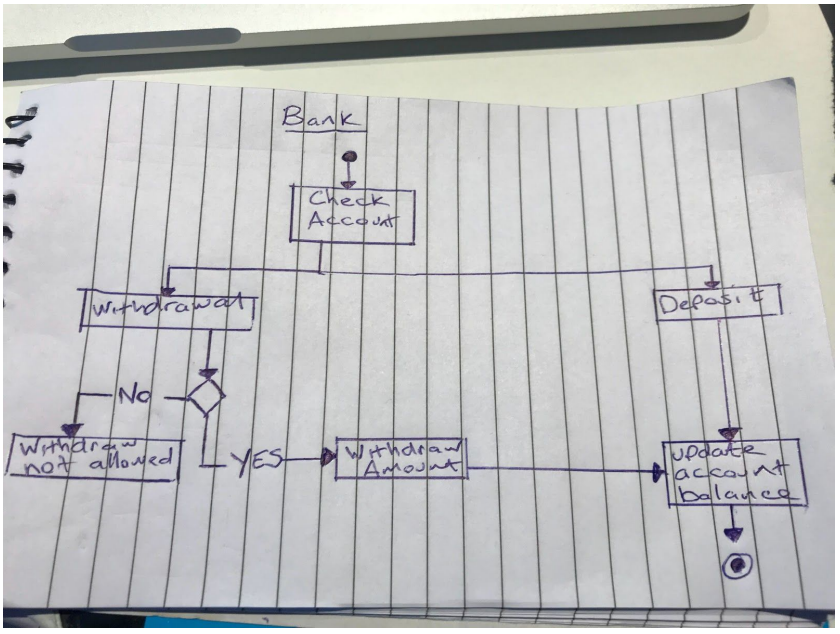
This shows all the relationships between the classes.

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



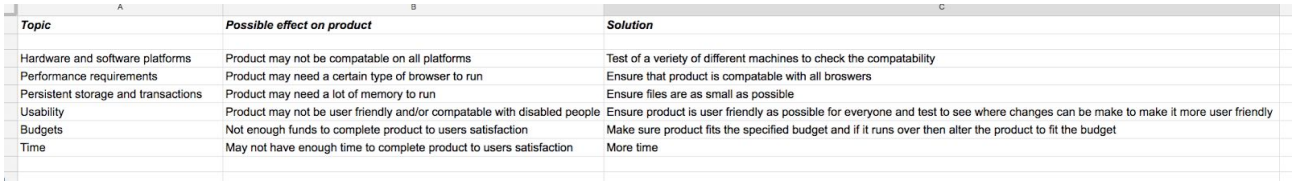
This shows all the information that would be entered.

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



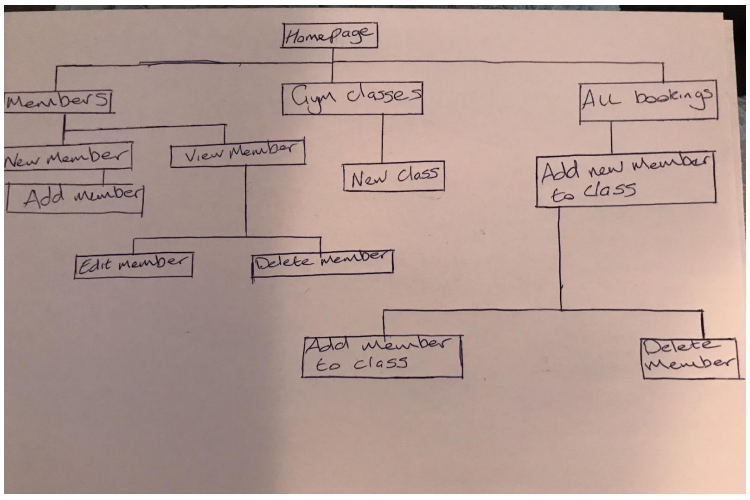
Shows the activity of a customer using an ATM

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	



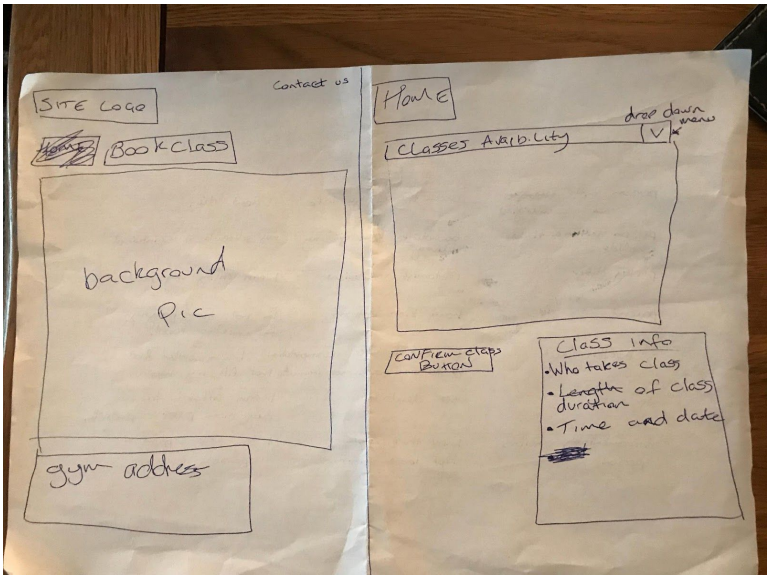
	A	B	C
	Topic	***Possible effect on product***	***Solution***
	Hardware and software platforms	Product may not be compatable on all platforms	Test of a veriety of different machines to check the compatability
	Performance requirements	Product may need a certain type of browser to run	Ensure that product is compatable with all broswers
	Persistent storage and transactions	Product may need a lot of memory to run	Ensure files are as small as possible
	Usability	Product may not be user friendly and/or compatable with disabled people	Ensure product is user friendly as possible for everyone and test to see where changes can be make to make it more user friendly
	Budgets	Not enough funds to complete product to users satisfaction	Make sure product fits the specified budget and if it runs over then alter the product to fit the budget
	Time	May not have enough time to complete product to users satisfaction	More time

Unit	Ref	Evidence	
P	P.5	User Site Map	



Site map for the full website.

Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	



The wireframe on the left shows the home/landing page and the right one shows what happens when you click on “book class”

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

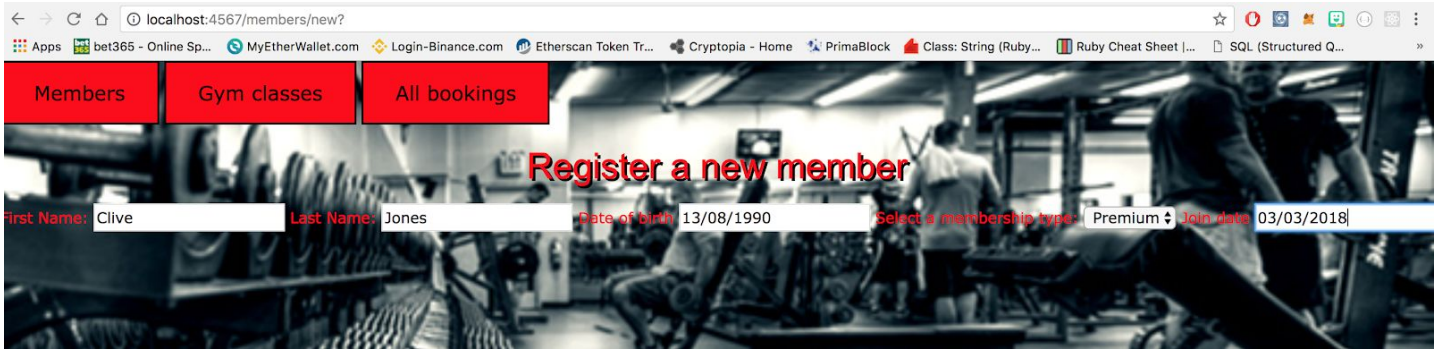
```
Pseudo code example for pda.

We want to charge a visitor double price if their height is over 200cm.

We will declare the type of double and name the method, pass in a Visitor type and name them "visitor".

We will write an if statement, use the visitor we passed in and use the getHeight method that we wrote for
it then check to see if hight is over >200

If true we return the defaultPrice() times 2 (*2), else return the defaultPrice


public double priceForVisitor(Visitor visitor){
        if (visitor.getHeight() > 200 ){
        return defaultPrice()*2;
        }
        return defaultPrice();
    }
```

Pseudocode i wrote for is a method that has to charge a visitor double price if their height is over 200cm.

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	

A user adding a new gym class and a time and that class then appearing in the gym classes list.

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	




```
gym_members=# select * from members;
 id | first_name | second_name |     dob     | membership_type | join_date
----+------------+-------------+-------------+-----------------+------------
  1 | Clive      | Jones       | 13/08/1990  | Premium         | 03/03/2018
  2 | Susan      | Swan        | 22/09/1981  | Regular         | 03/03/2014
  3 | Paul       | Johnson     | 12/097/1983 | Regular         | 13/09/2016
  4 | Clive      | Jones       | 13/08/1990  | Premium         | 03/03/2018
  5 | Susan      | Swan        | 22/09/1981  | Regular         | 03/03/2014
  6 | Paul       | Johnson     | 12/097/1983 | Regular         | 13/09/2016
  7 | Clive      | Jones       | 13/08/1990  | Premium         | 03/03/2018
  8 | Susan      | Swan        | 22/09/1981  | Regular         | 03/03/2014
 10 | Clive      | Jones       | 13/08/1990  | Premium         | 03/03/2018
 11 | Susan      | Swan        | 22/09/1981  | Regular         | 03/03/2014
 12 | Paul       | Johnson     | 12/097/1983 | Regular         | 13/09/2016
(11 rows)

gym_members=#
```

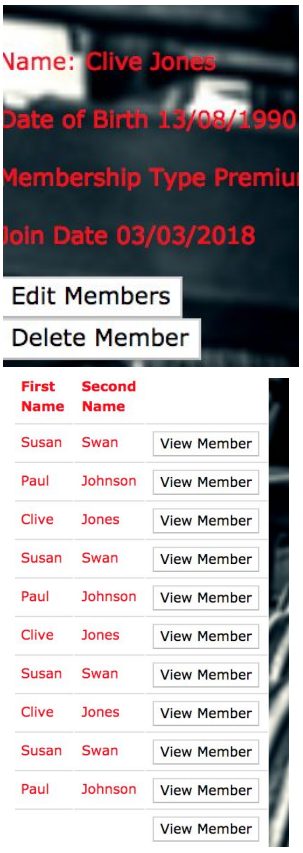
Adding a member then it appearing in the sql database

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	

```ruby
def self.delete(id) #deletes members
  sql = "DELETE from members WHERE id = $1"
  values = [id]
  SqlRunner.run(sql, values)
end
```

```ruby
post '/members/:id/delete' do #delete
  Member.delete(params[:id])
  redirect to '/members'
end
```

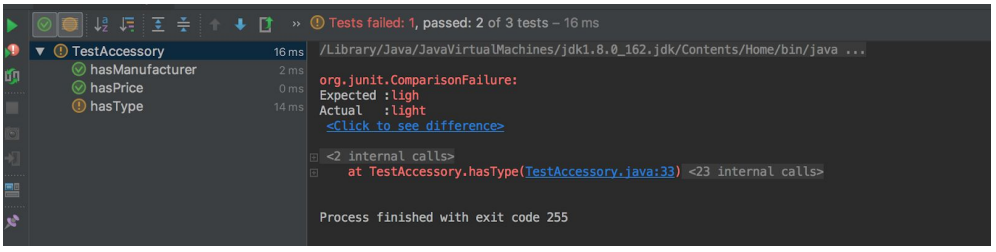
```erb
<form action ="/members/<%= @members.id %>/delete" method="post">
  <input type ="submit" value ="Delete Member"/>
</form>
```



Screenshots from code and browser, the code is deleting a member and the result of performing that in the browser

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	

```
public class TestAccessory {

    DrumSticks drumsticks;
    GuitarStrings guitarStrings;

//    manu   type  price
    @Before
    public void setUp() {
        drumsticks = new DrumSticks( manufacturer: "Lars Urlich",  type: "5A",  buyPrice: 15.99,  sellPrice: 25.50);
        guitarStrings = new GuitarStrings( manufacturer: "Elixir",  type: "light",  buyPrice: 22.99,  sellPrice: 35.50);

        }

    @Test
    public void hasManufacturer() {
        assertEquals( expected: "Lars Urlich", drumsticks.getManufacturer());
        assertEquals( expected: "Elixir", guitarStrings.getManufacturer());

        }

    @Test
    public void hasType() {
        assertEquals( expected: "5A", drumsticks.getType());
        assertEquals( expected: "light", guitarStrings.getType());

    }

    @Test
    public void hasPrice() {
        assertEquals( expected: 15.99, drumsticks.getBuyPrice(), delta: 0.01);
        assertEquals( expected: 22.99, guitarStrings.getBuyPrice(), delta: 0.01);
    }
}
```



```
Tests failed: 1, passed: 2 of 3 tests – 16 ms
TestAccessory 16 ms
  hasManufacturer 2 ms
  hasPrice 0 ms
  hasType 14 ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java ...

org.junit.ComparisonFailure:
Expected :ligh
Actual   :light
 <Click to see difference>

 <2 internal calls>
    at TestAccessory.hasType(TestAccessory.java:33) <23 internal calls>

Process finished with exit code 255
```

```
public class TestAccessory {

    DrumSticks drumsticks;
    GuitarStrings guitarStrings;

//    manu   type  price
    @Before
    public void setUp() {
        drumsticks = new DrumSticks( manufacturer: "Lars Urlich",  type: "5A",  buyPrice: 15.99,  sellPrice: 25.50);
        guitarStrings = new GuitarStrings( manufacturer: "Elixir",  type: "light",  buyPrice: 22.99,  sellPrice: 35.50);

        }

    @Test
    public void hasManufacturer() {
        assertEquals( expected: "Lars Urlich", drumsticks.getManufacturer());
        assertEquals( expected: "Elixir", guitarStrings.getManufacturer());

        }

    @Test
    public void hasType() {
        assertEquals( expected: "5A", drumsticks.getType());
        assertEquals( expected: "light", guitarStrings.getType());

    }
```

```
Run: TestAccessory
Tests passed: 3 of 3 tests – 2 ms
TestAccessory 2 ms
  hasManufacturer 1 ms
  hasPrice 1 ms
  hasType 0 ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:/Applications/Intel

Process finished with exit code 0
```

Example of test data which fails because we have the wrong string, we change the string to the correct string and it passes.

Week 7

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

```java
import java.util.ArrayList;

public class Hotel {

    private ArrayList<Room> rooms;
    private ArrayList<Guest> guests;

    public Hotel(){
        this.rooms = new ArrayList<Room>();
        this.guests = new ArrayList<Guest>();
    }
```

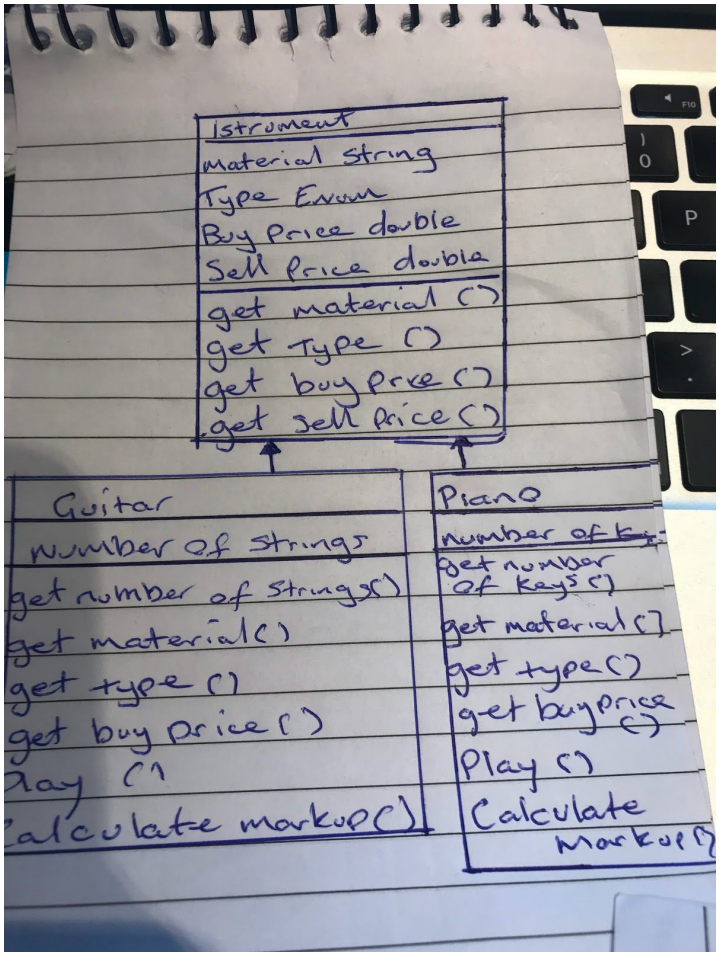
```java
public class Confroom extends Room {

    private String name;
    private Integer capacity;
    private ArrayList guests;
    private Integer dailyrate;

    public Confroom(String name, int capacity, int dailyrate) {
        super(capacity);
        this.guests = new ArrayList<Guest>();
        this.dailyrate = dailyrate;
        this.name = name;
    }

    public Integer getDailyRate() {
        return dailyrate;
    }

    public String getName() {
        return name;
    }
```

```java
public void checkInGuest(Room room, Guest guest) {
    room.addGuest(guest);
    }
```

```java
@Test
 public void getGuestsInRoom() {
     hotel.addBedRoom(conferenceRoom1);
     ArrayList<Guest> guests = new ArrayList<Guest>();
     guests.add(guest1);
     guests.add(guest2);
     hotel.checkInGuest(conferenceRoom1.getNumber(), guests);
     assertTrue(guests.equals.getGuestsCheckedIntoRoom(conferenceRoom1.getNumber));


 }
```

We demonstrate polymorphism here as the Hotel class has an arraylist of rooms… anything that extends room can use the properties of a room, conferenceRoom can be a Room and a conferenceRoom object and can be added to a hotel.

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



An Inheritance diagram showing a guitar and a piano inheriting from an Instrument class.

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

```java
package Instruments;

public abstract class Instrument {

    private String manufacturer;
    private String type;
    private String colour;

    public Instrument(String manufacturer, String type, String colour){
        this.manufacturer = manufacturer;
        this.type = type;
        this.colour = colour;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public String getType() {
        return type;
    }

    public String getColour() {
        return colour;
    }
}
```

Example of an instrument class with private properties that can be accessed by using getters and setters elsewhere in the program.

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.	

```java
package Accessories;

import Accessories.DrumSticks;
import Accessories.GuitarStrings;
import Interfaces.ISell;

public abstract class Accessory implements ISell {

    private String manufacturer;
    private String type;
    private double buyPrice;
    private double sellPrice;


    public Accessory(String manufacturer, String type, double buyPrice, double sellPrice){
        this.manufacturer = manufacturer;
        this.type = type;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getManufacturer() {
        return manufacturer;
    }
```

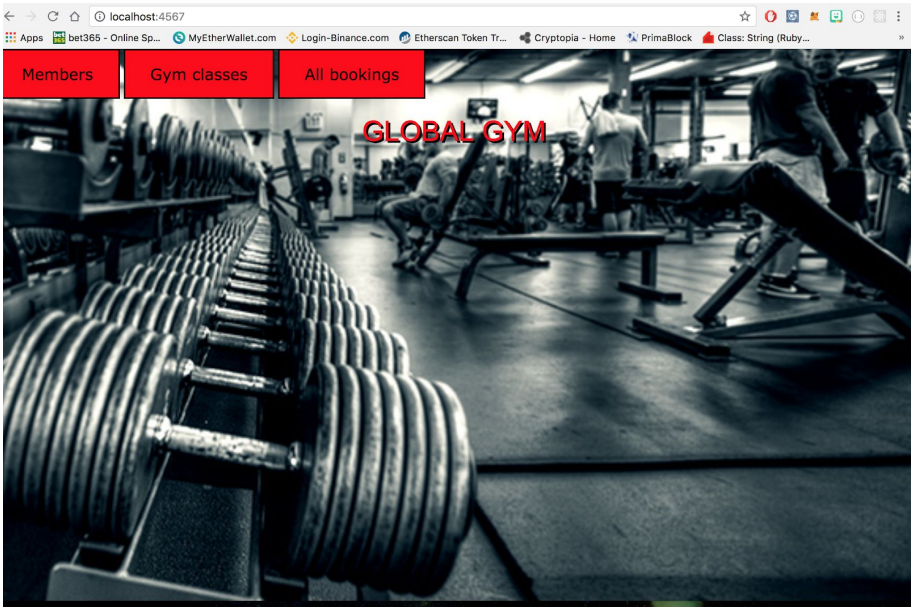
```java
package Accessories;

import Interfaces.ISell;

public class DrumSticks extends Accessory {

    public DrumSticks(String manufacturer, String type, double buyPrice, double sellPrice) {
        super(manufacturer, type, buyPrice, sellPrice);
    }
```

```java
public class TestAccessory {

    DrumSticks drumsticks;
    GuitarStrings guitarStrings;

//    manu   type  price
    @Before
    public void setUp() {
        drumsticks = new DrumSticks("Lars Urlich", "5A", 15.99, 25.50);
        guitarStrings = new GuitarStrings("Elixir", "light", 22.99, 35.50);

        }

    @Test
    public void hasManufacturer() {
        assertEquals("Lars Urlich", drumsticks.getManufacturer());
        assertEquals("Elixir", guitarStrings.getManufacturer());
```

Screenshots of an Accessory and Drumstick class which inherits from Accessory and test examples of new Drumsticks object to make methods.

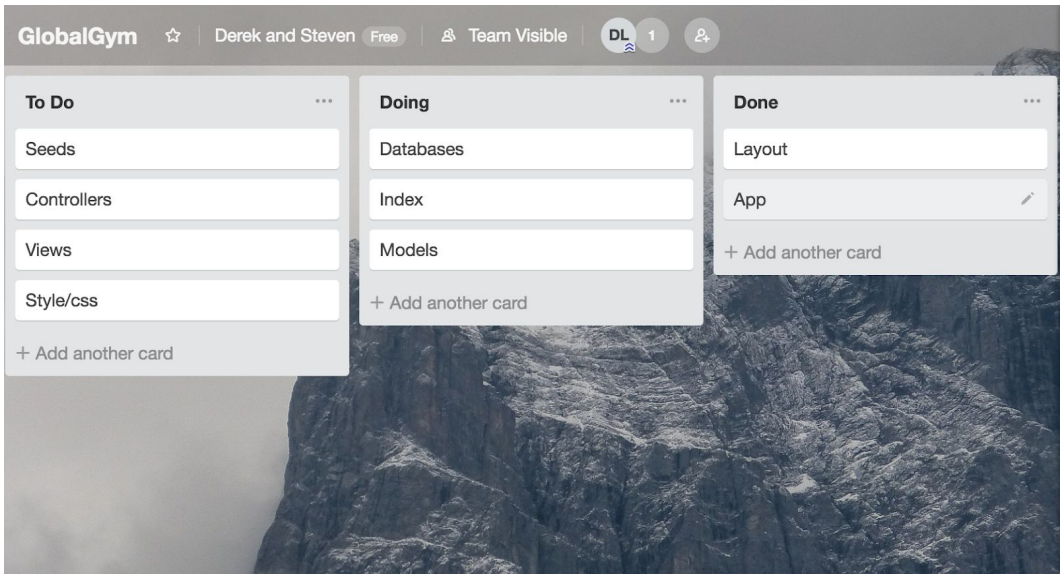
Week 1

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/dl184/RubyProject

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



Unit	Ref	Evidence	
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.	

```
public void addGuest(Guest guest){
    if (guests.size() < this.capacity){
        guests.add(guest);
    }
}
```

I chose this algorithm because it seems like a good way to test to see if you should add a guest if the guest number is less than the capacity.

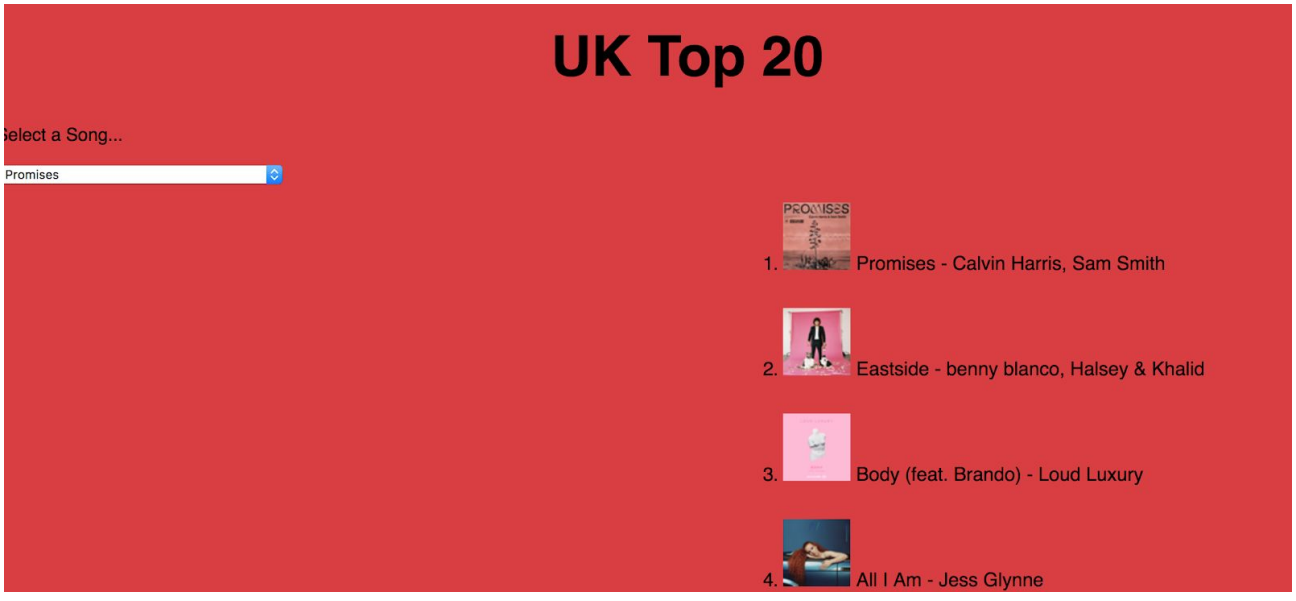
```
import React from 'react';

class SongList extends React.Component {
  render(){
    const songList = this.props.songs.map((song, index) => (
      <li key={song.id.attributes["im:id"]}><img src={song["im:image"][1].label} alt="song art" /> {song.title.label}</li>
    ))

    return(
        <ol className='chart-list'>{songList}</ol>
      )
    }
  }


    export default SongList;
```

I chose this algorithm that maps over an array of songs from an api and displays relevant information in an ordered list to see each position in the chart.

Week 12

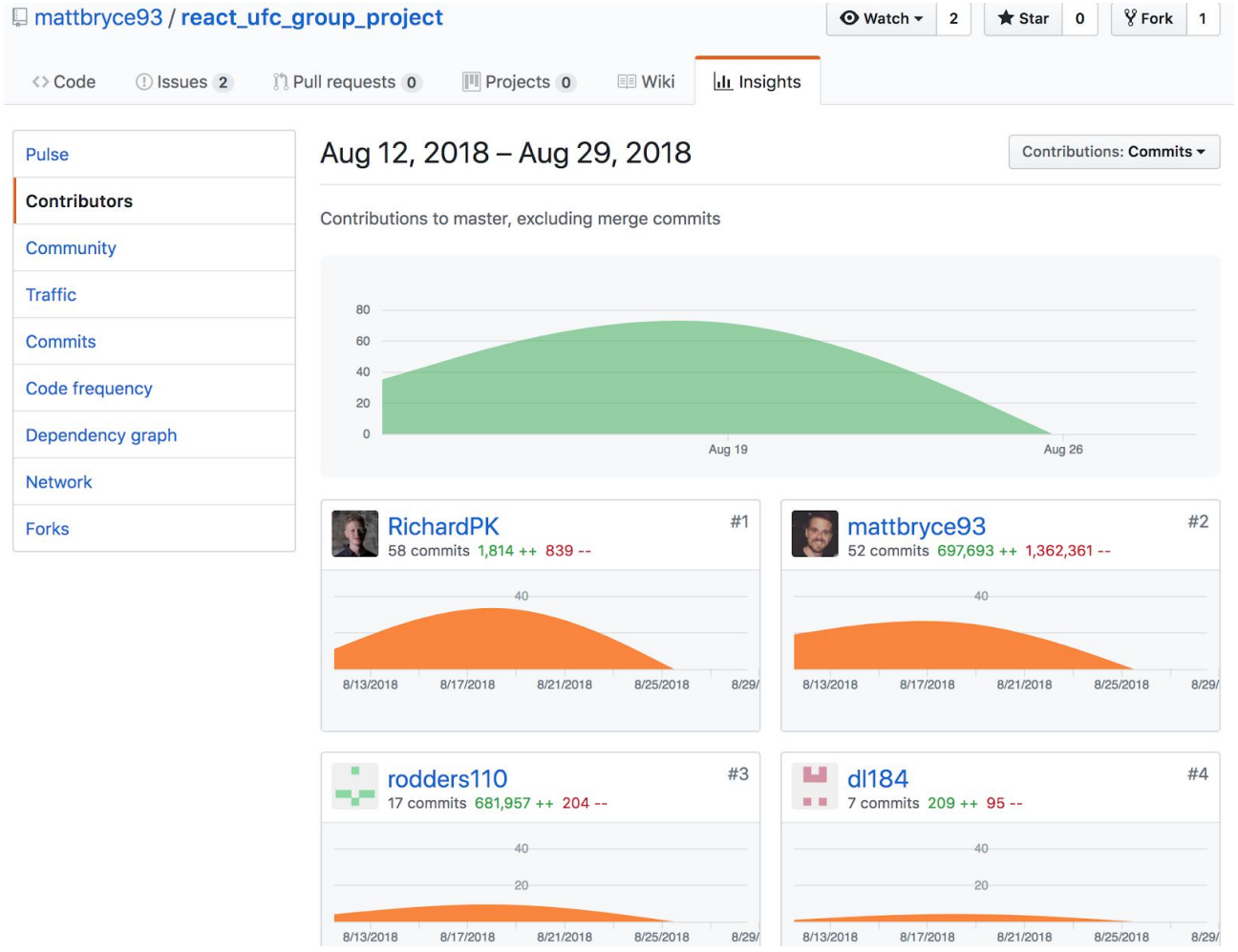
Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	

```
componentDidMount(){
  const url = 'https://itunes.apple.com/gb/rss/topsongs/limit=20/json'
  fetch(url).then(res => res.json()).then(songs => this.setState({songs: songs.feed.entry}));
}
```



Week 15

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.	



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

README.md

Fan UFC Project Brief

Sports Dashboard

Sports fans want to be able to view relevant sporting events on a dashboard. With a sport of your choice, use an existing API or create a new API to display information about fixtures, news and travel information for events.

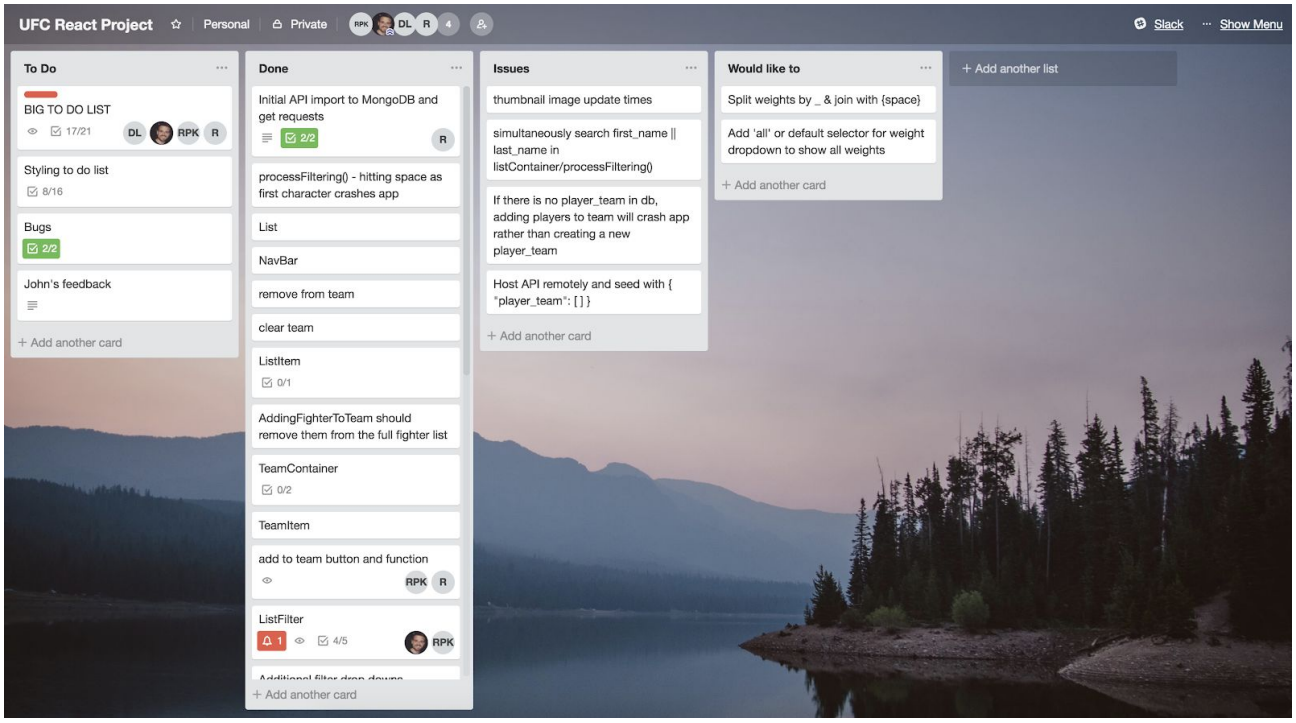
Possible APIs to use:

- UFC: http://ufc-data-api.ufc.com/api/v1/us
- Football: http://api.football-data.org/index
- Triathalon: https://developers.triathlon.org/docs

MVP

- Display upcoming events on a map
- Display results and ranking of players/teams
- Allow users to add events to a favourites list

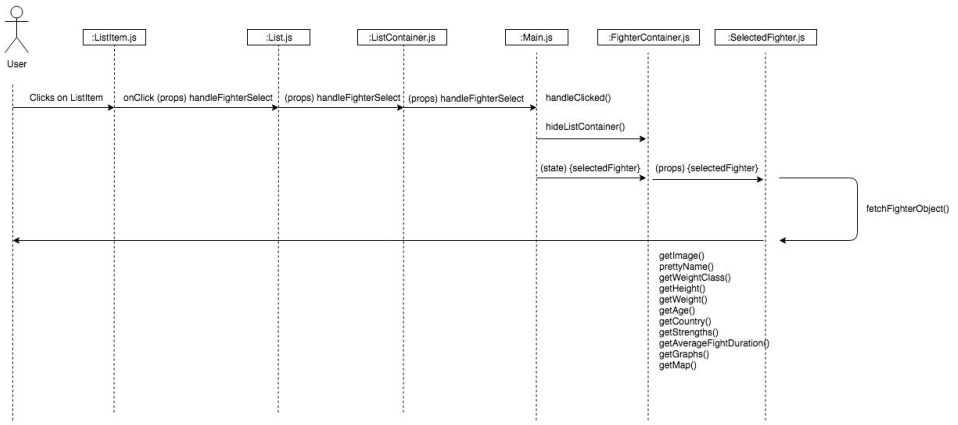
Unit	Ref	Evidence	
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	

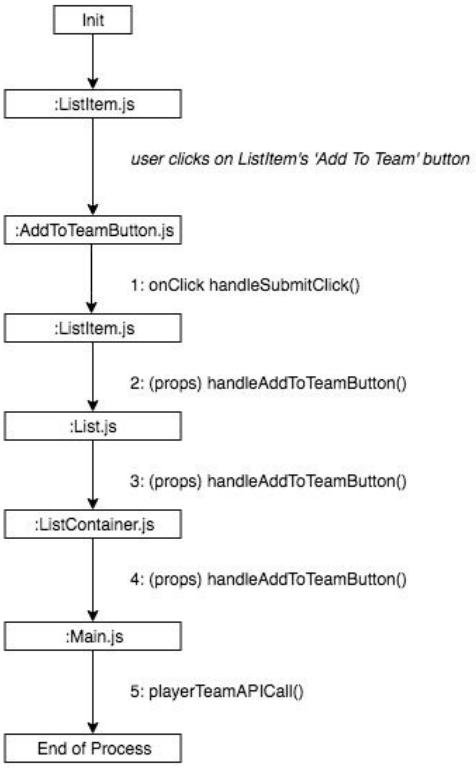


Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

Accepted Criteria	Expected Result/Output	Pass/Fail
A user is able to add Fighters to their Team	When a user selects add fighter the fighter should appear in the Team section	Pass
A user should be able to filter Fighters by weight class	When a user selects a weight class for fighters, the populated list should only contain fighters within that weight class	Pass
A user should be able to search by the name of the fighter	When a user types a string into the name search box the populated list should contain only fighters with matching names	Pass
A user should be able to see extra details of any fighter	When a user clicks on a fighter they should be presented with up to date statistics of that fighters historical performance.	Pass
A user should be able to compare their team with other teams	When a user inspects all teams they should be presented with a table showing the ranking of their team against other teams	Fail
A user should be able to remove fighters from their team	When a user selects to remove a fighter from their team the fighter should disappear from the team section	Pass

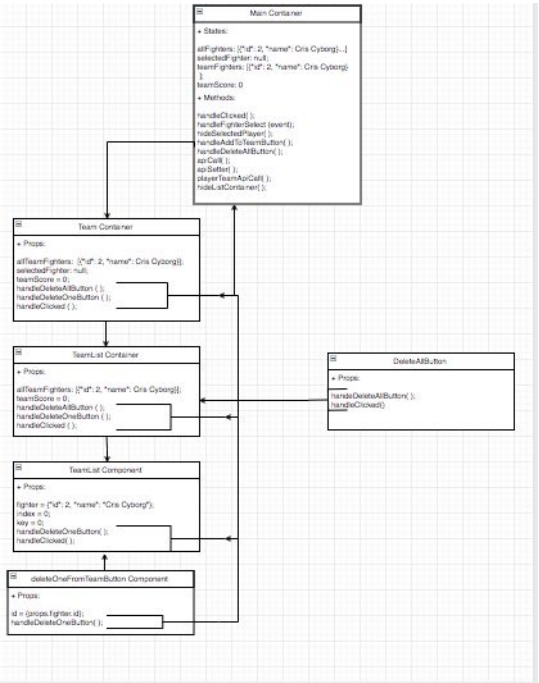
Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	





Unit	Ref	Evidence	

P	P.8	Produce two object diagrams.



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

A	B	C	D
User must be able to add a fighter to team selection	FAILED	Had a syntax error of an extra comma	PASSED
Some of the fighters didn't have past fight data so we couldn't populate the map to show past fights and it was causing the map to crash	FAILED	Set a condtion to show the map without being populated if the fighter did not have any data	PASSED
Couldn't show the nickname of any fighters	FAILED	Added quote marks around the "nickname" from the api in the player view which meant they added the nickname of the fighte	PASSED
Images were loading very slow and it was slowing down the app	FAILED	Added lazy load which loaded a certain amount of images then more as you scroll down down the page	PASSED
Couldn't show locations of last fights on map where the fighter did have historical data for it but it was as a string	FAILED	Converted the string location to coorinates and implemented them to the map	PASSED