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	

Paste Screenshot here

```
def print_array_plus_one(array)
  array.each{ |element| p element + 1}
end

array = [1,2,3,4,5]

print_array_plus_one(array)
```

Description here

Creates an array of integers. Function prints each of those integers + 1.

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

Paste Screenshot here

```ruby
def print_hash_plus_one(hash_example)
  p hash_example['one'] + 1
  p hash_example['two'] + 1
  p hash_example['three'] + 1
end

hash_example = {
  'one' => 1,
  'two' => 2,
  'three' => 3
}

print_hash_plus_one(hash_example)
```

```
code git:(master) ✗ ruby code.rb
2
3
4
code git:(master) ✗
```

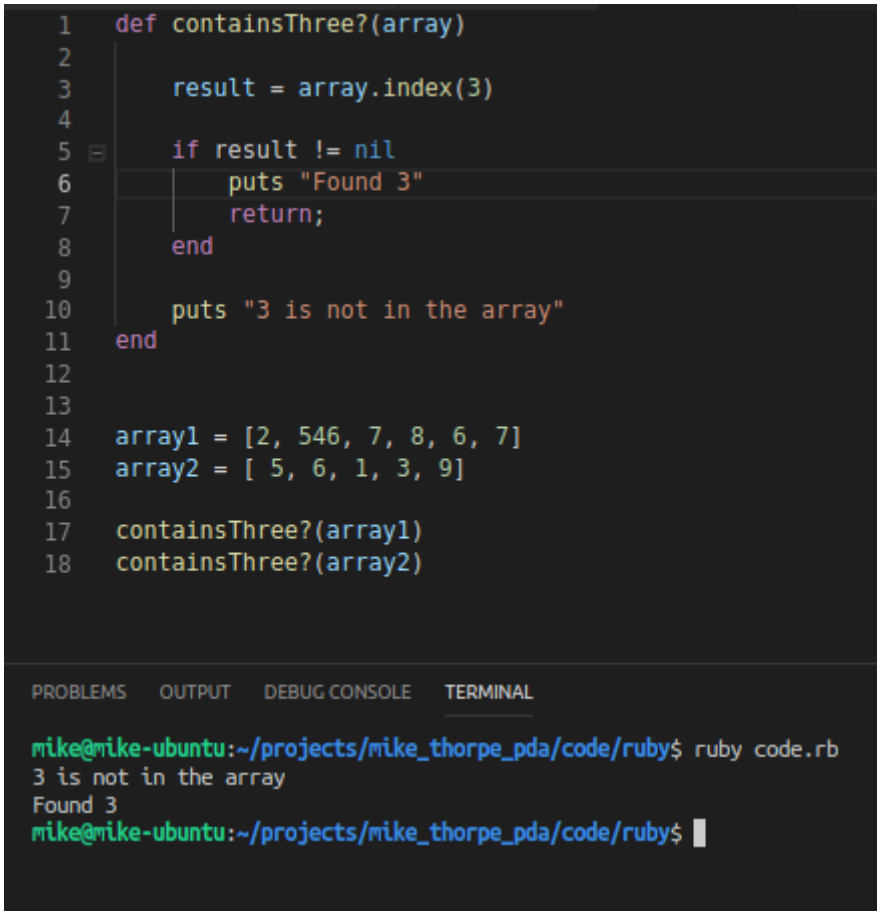
Description here

Creates a hash of string key integer values. Function prints each of those integers + 1.

Week 3

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

Paste Screenshot here



Description here

Function searches an array for the number 3. Logs a string to the console indicating if three has been found when .find_index does not return nil else returns a string indicating that 3 is not in the array passed.

Unit	Ref	Evidence	
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	

Paste Screenshot here

```
13  def sortNumbersAscending(array_of_numbers)
14      sorted_numbers = array_of_numbers.sort()
15      "Puts sorted numbers ascending"
16      puts sorted_numbers
17      return sorted_numbers
18  end
19
20  array1 = [2, 546, 7, 8, 6, 7]
21
22  sortNumbersAscending(array1)
23
24
25
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
mike@mike-ubuntu:~/projects/mike_thorpe_pda/code/ruby$ ruby code.rb
2
6
7
7
8
546
mike@mike-ubuntu:~/projects/mike_thorpe_pda/code/ruby$
```

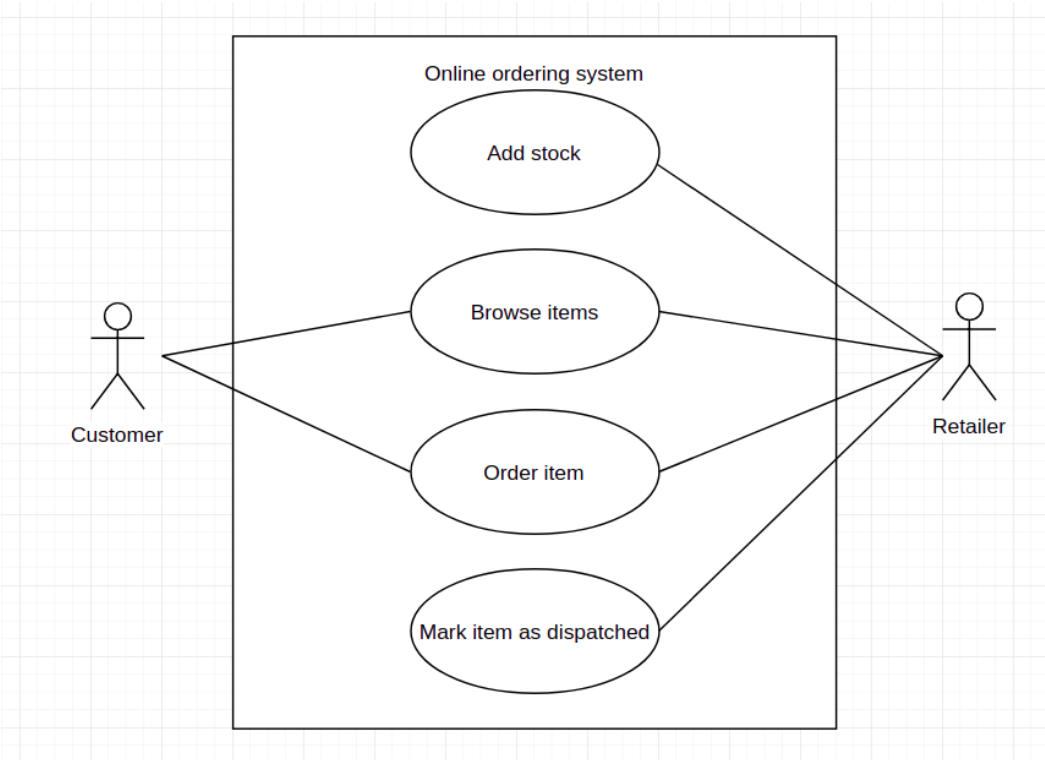
Description here

Function returns a new array of numbers sorted in ascending order given an array of numbers is passed. The new array is logged to the screen.

Week 5

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

Paste Screenshot here

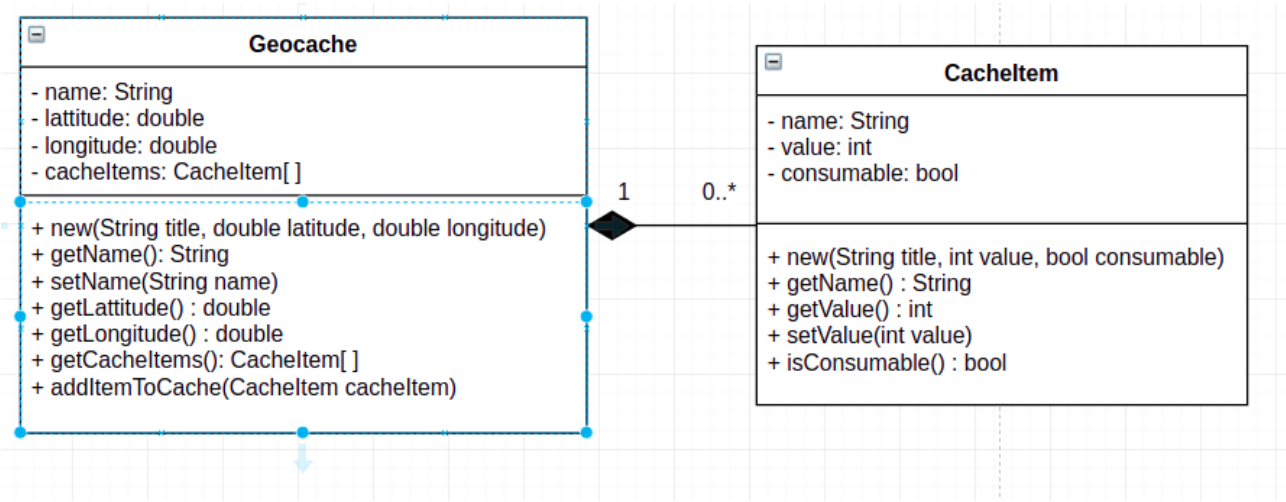


Description here

Diagram above shows interaction between customer, retailer and a simple online ordering system for buying items.

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

Paste Screenshot here

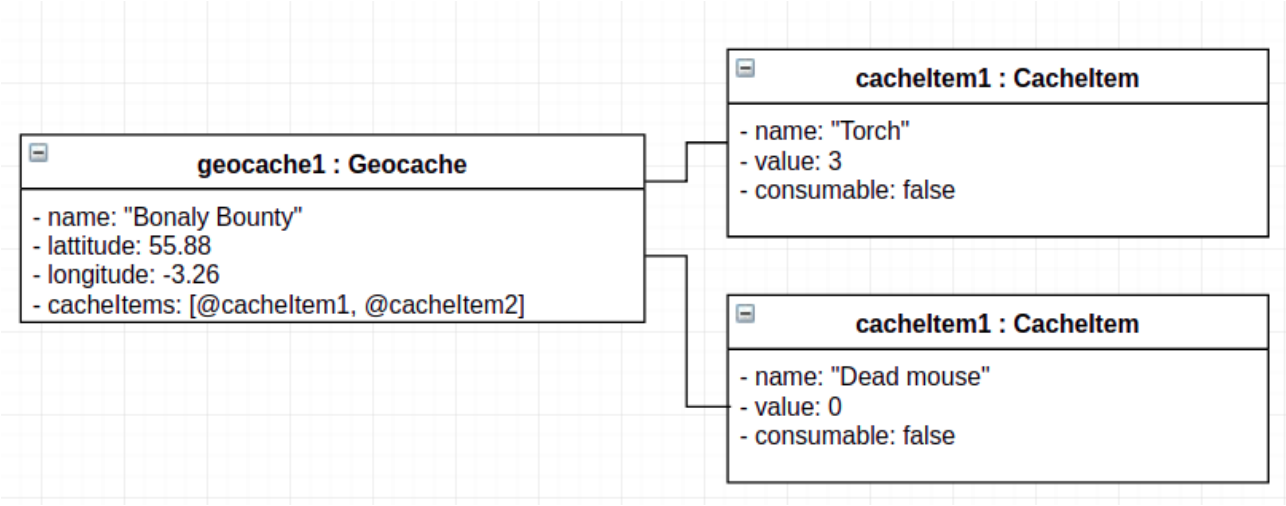


Description here

The above shows a diagram of two classes – private and public methods and properties are indicated with a '-' and '+' respectively. Top half displays properties, bottom half shows methods. Return type is indicated after ':' for each method that returns. A Geocache has between 0 and many CacheItems by composition and a CacheItem can only be in one Geocache this is indicated by the link between them.

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

Paste Screenshot here

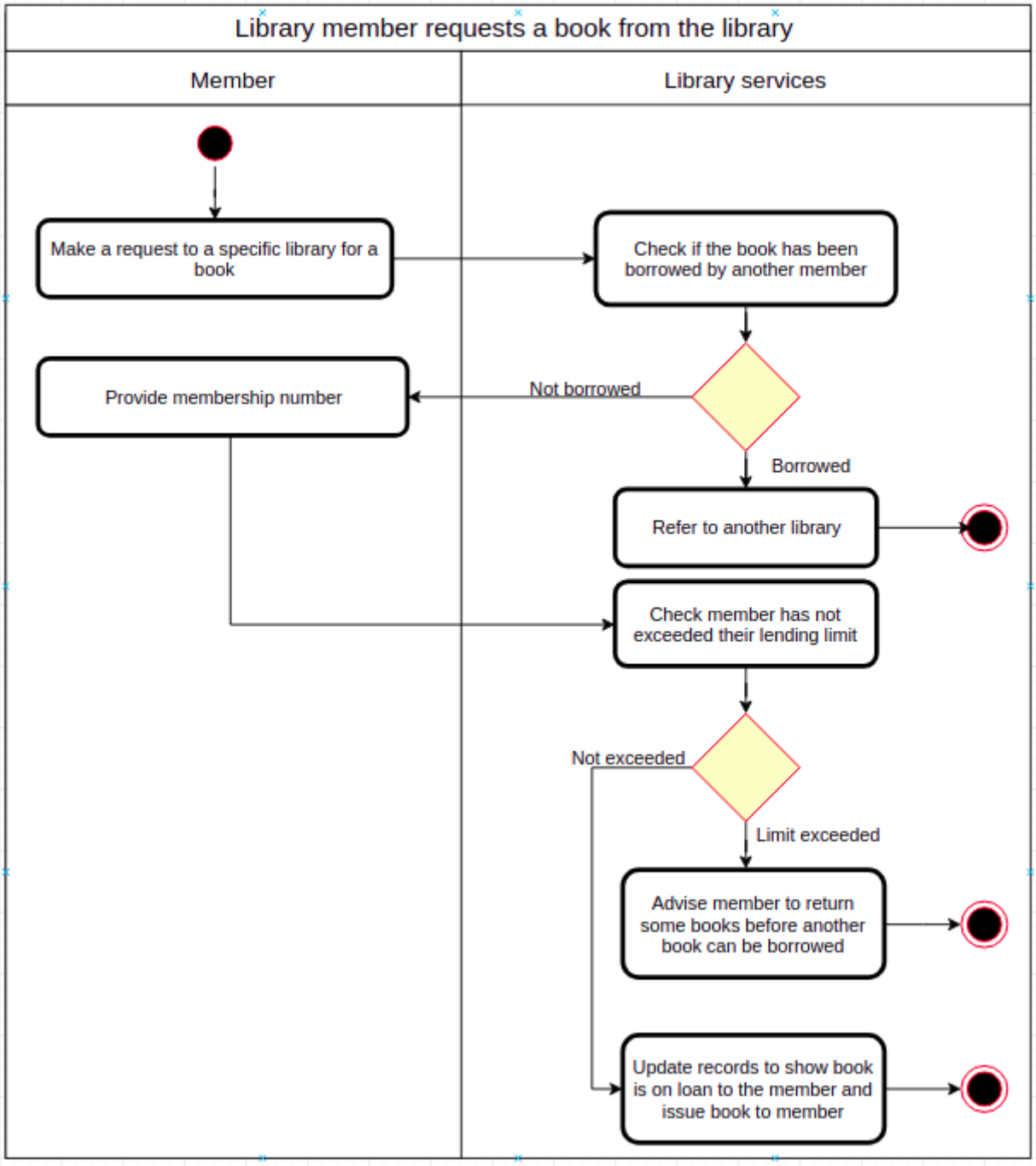


Description here

The above shows an object diagram, indicating the values of properties and the the Classes instantiated from. The links show how the Geocache is composed of the cache items.

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

Paste Screenshot here



Description here

Above shows an activity diagram demonstrating a library member requesting a book from their local library

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	

Paste Screenshot here

Topic	Constraint	Solution
Hardware and software platforms	The app must be available to as many users on as many platforms as possible. Might constrain development time as developing for many different platforms is time consuming. This may lead to poor quality of product on each platform due to time drain of maintaining multiple code bases.	Avoid time drain by having a single code base and maximize number of platforms targeted by creating web app based solution capable of running in any modern browser – these run on most platforms.
Performance requirements	The app must start within 10s and all user requests must complete within 5s. If the load times are greater than the user may navigate away from the site due to frustration or a perceived error. This could lead to increased requests for support and frustration for the user, ultimately resulting in them migrating to another service.	Mitigate the risk of long load times by hosting web app on paid version of Heroku that prevents dynos from sleeping, thus the pages of the application should load quickly in the browser and provide the responsive service the user expects.
Persistent storage and transactions	User data must be persisted in data base for later access. Risk data may be accessed by the wrong user. This could lead to data breaches and access to a users private information by third parties negatively impacting the user's privacy.	Implement user authentication system so that user has to log in to see their data. Test to ensure they can only see their data. For extra security data should be stored in DB as encrypted strings.
Usability	The user should be able to use the product without the need for a user manual. Risk the user will not know how to use the app. This could cause the user to give up interacting with the app and look for alternative solutions.	User experience should be planned at the start of the project with wireframes using common user interface design patterns. This will ensure that although the app is new the user will know how to interact with it. Use off tooltips throughout the app can prompt the user where more clarification on an interaction is needed. User testing should take place with new users and

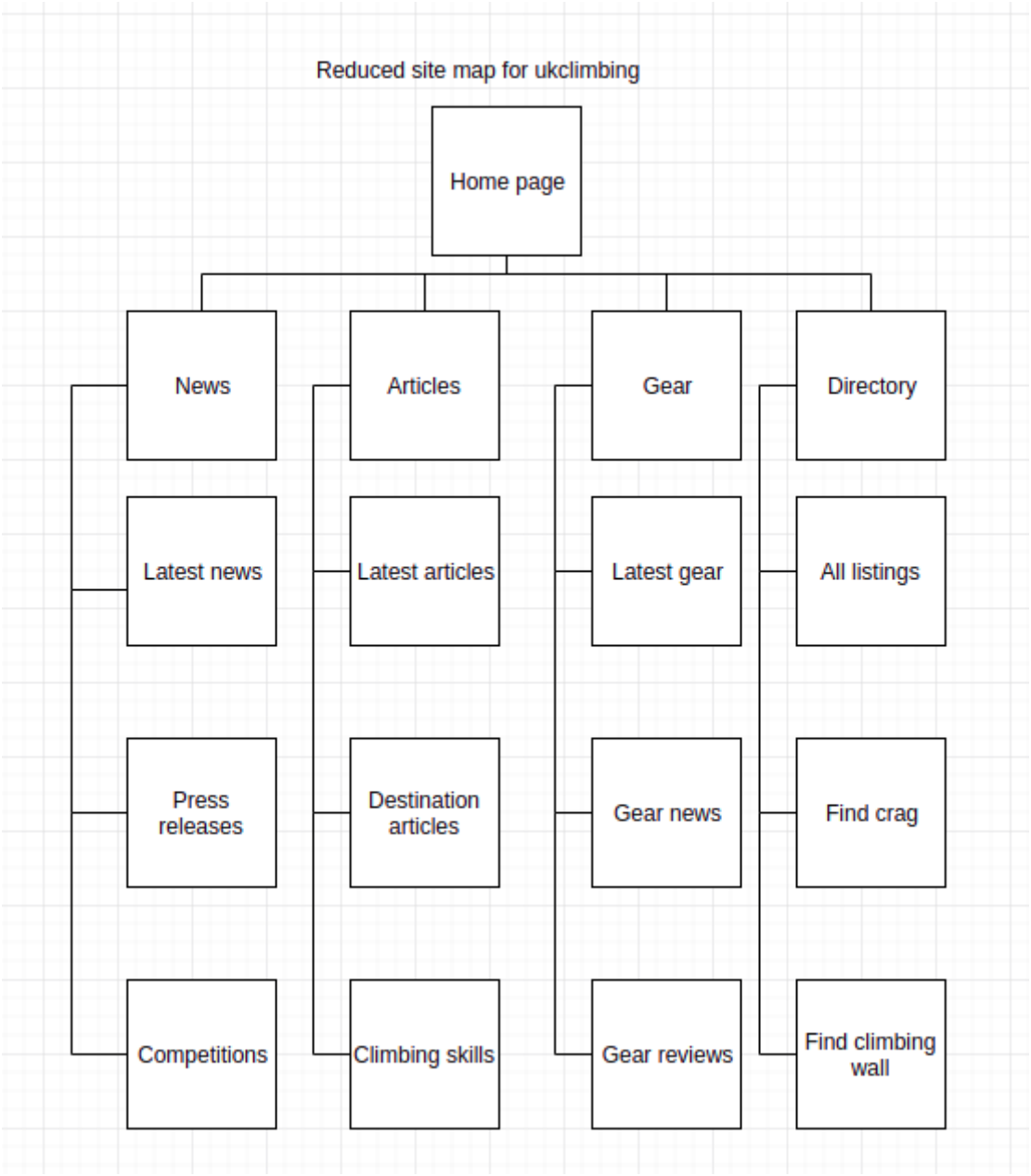
		revisions to the UI should be made accordingly.
Budgets	No monetary budget for project. Risk that project cannot be built due to lack of human and technological resource therefore there will be no product for the end user.	Build using students looking for work experience. Use only free open source tools and libraries for development
Time	Application must be produced in 7 days. Risk of the wrong features being developed to satisfy the MVP. This could lead to the product being unusable by the end user at the end of the development period.	Reduce risk of incomplete core features by use of Trello to prioritize features and plan those necessary to satisfy MVP. Extension features must only be implemented when MVP features are complete.

Description here

The table above details project constraints and how they can be met using technical and management solutions.

Unit	Ref	Evidence	
P	P.5	User Site Map	

Paste Screenshot here

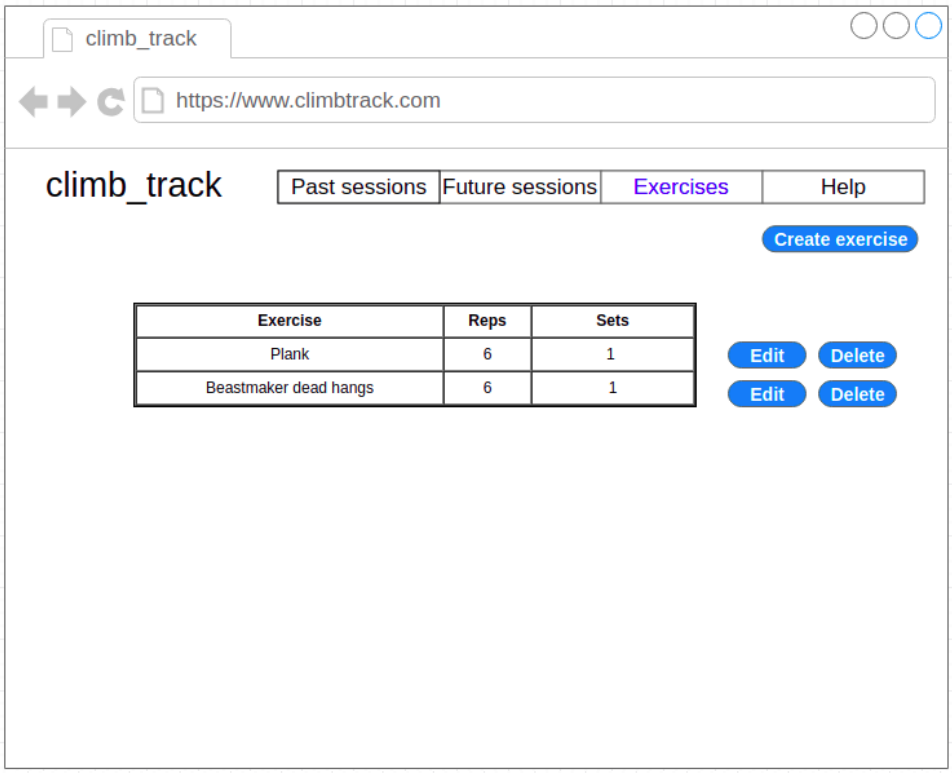


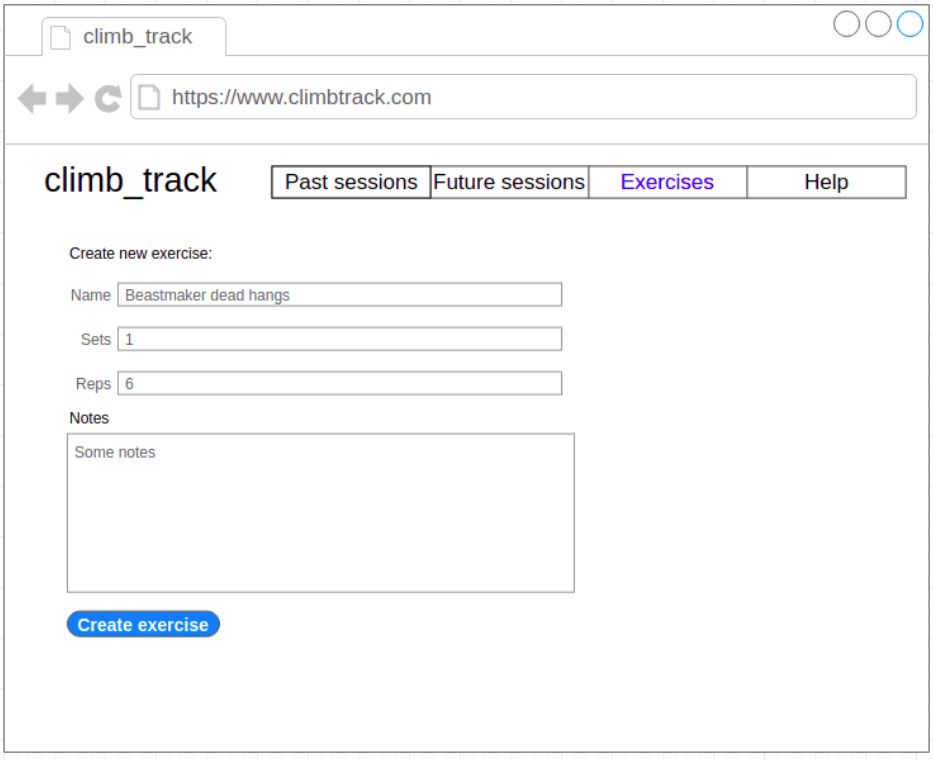
Description here

The above is a reduced site map for ukclimbing mapping out the tree of links from the homepage.

Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	

Paste Screenshot here



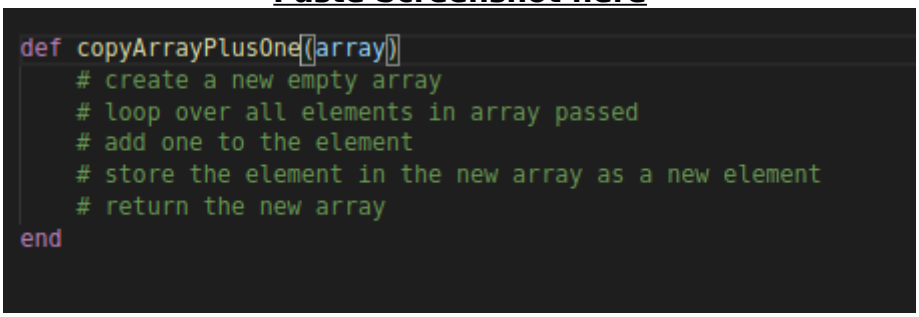


Description here

Wireframes showing exercise creation and list of exercises page for climb_track solo ruby project.

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

Paste Screenshot here

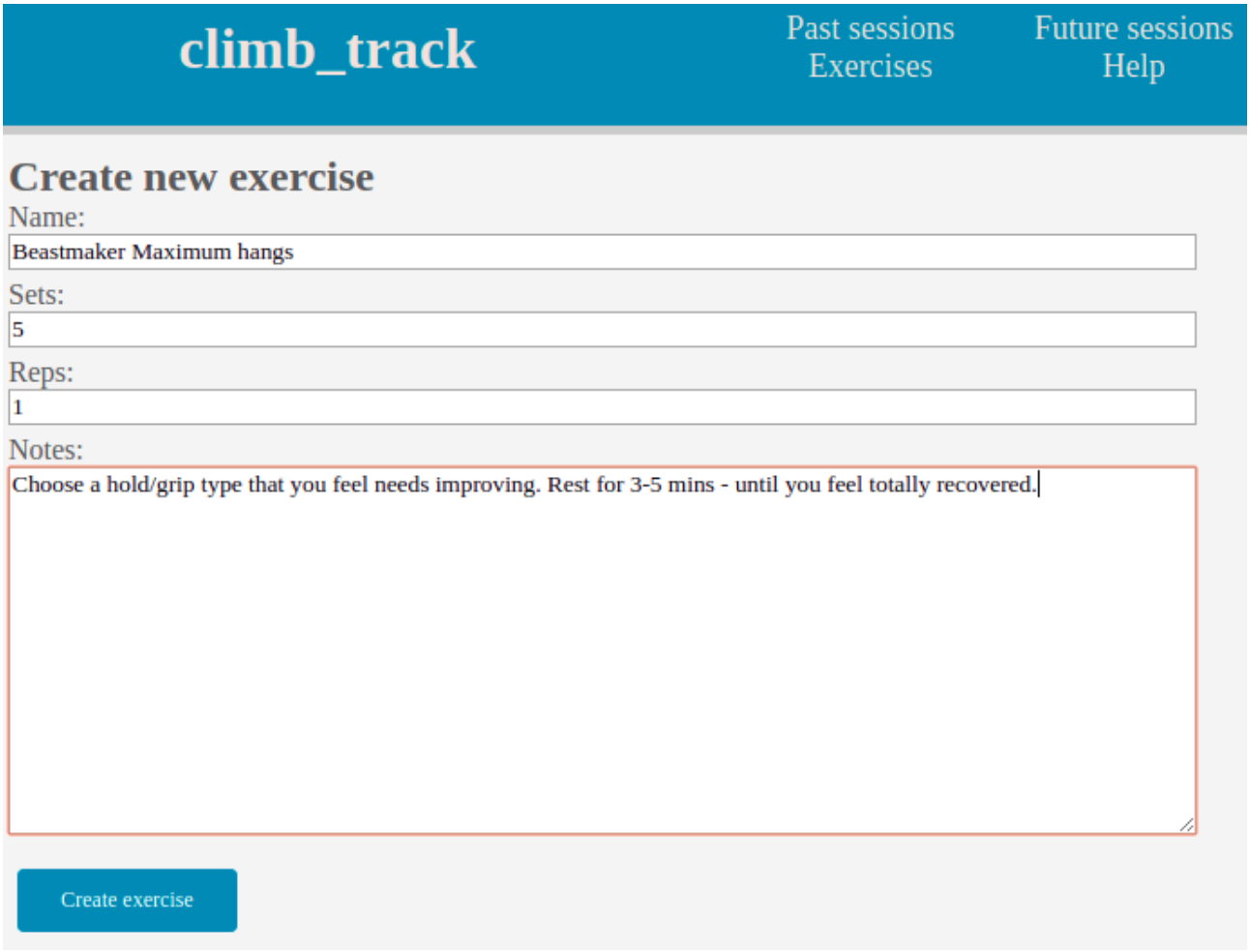


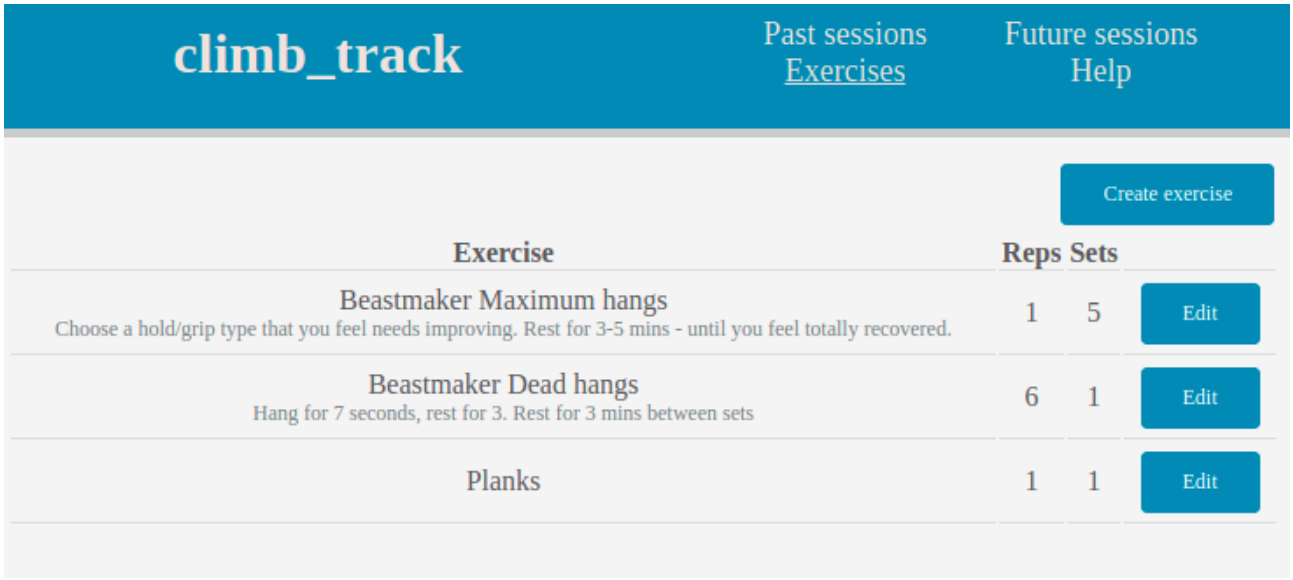
Description here

Psudo code for a method to create a copy of an array with 1 added to each element.

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

Paste Screenshot here



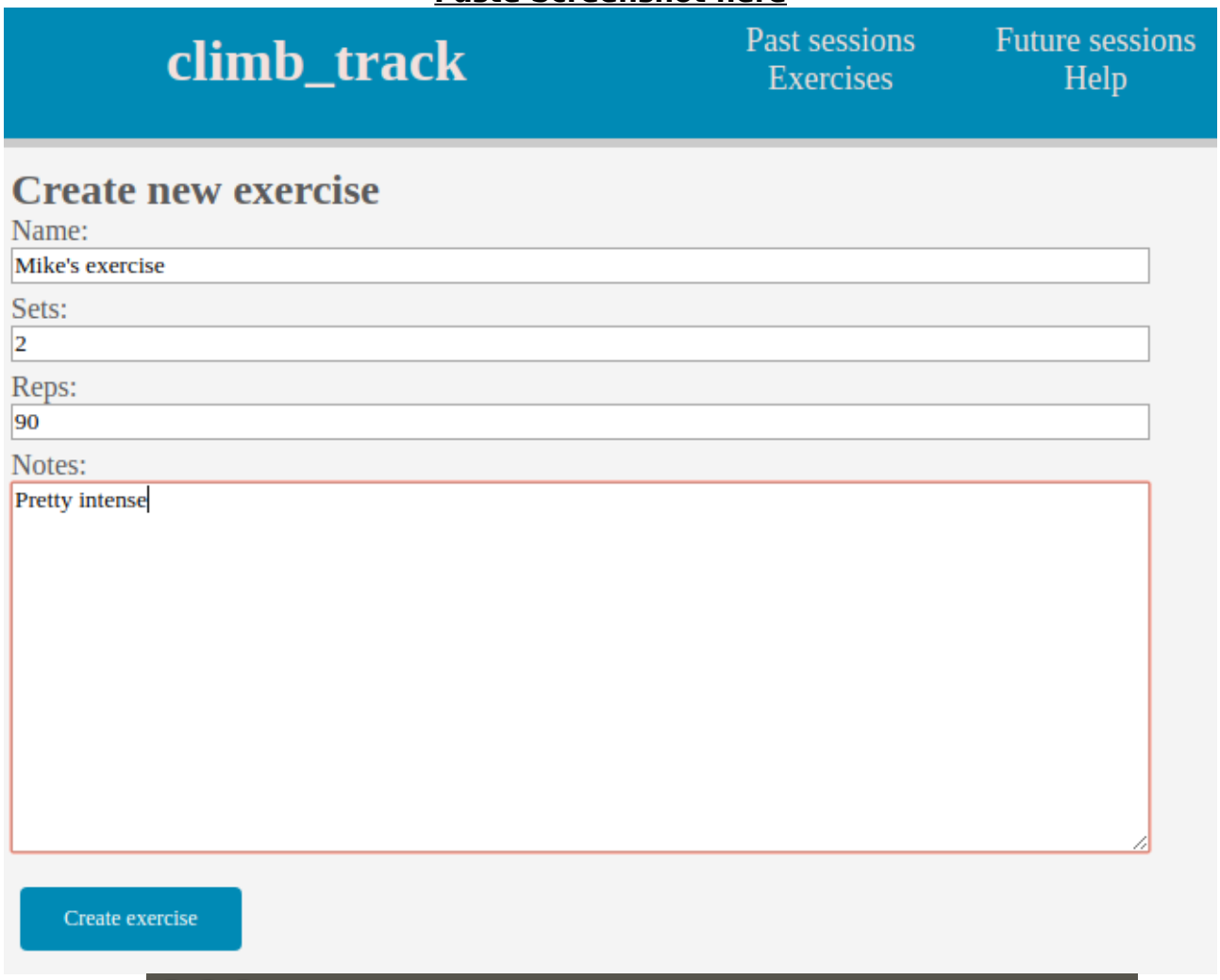


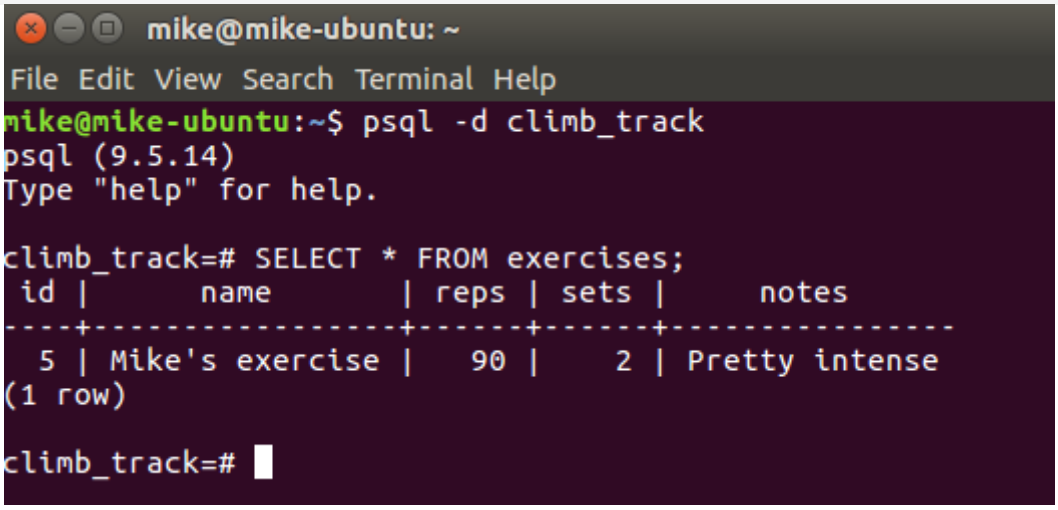
Description here

Adding new exercise data to the list of exercises in the climb_track app

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

Paste Screenshot here



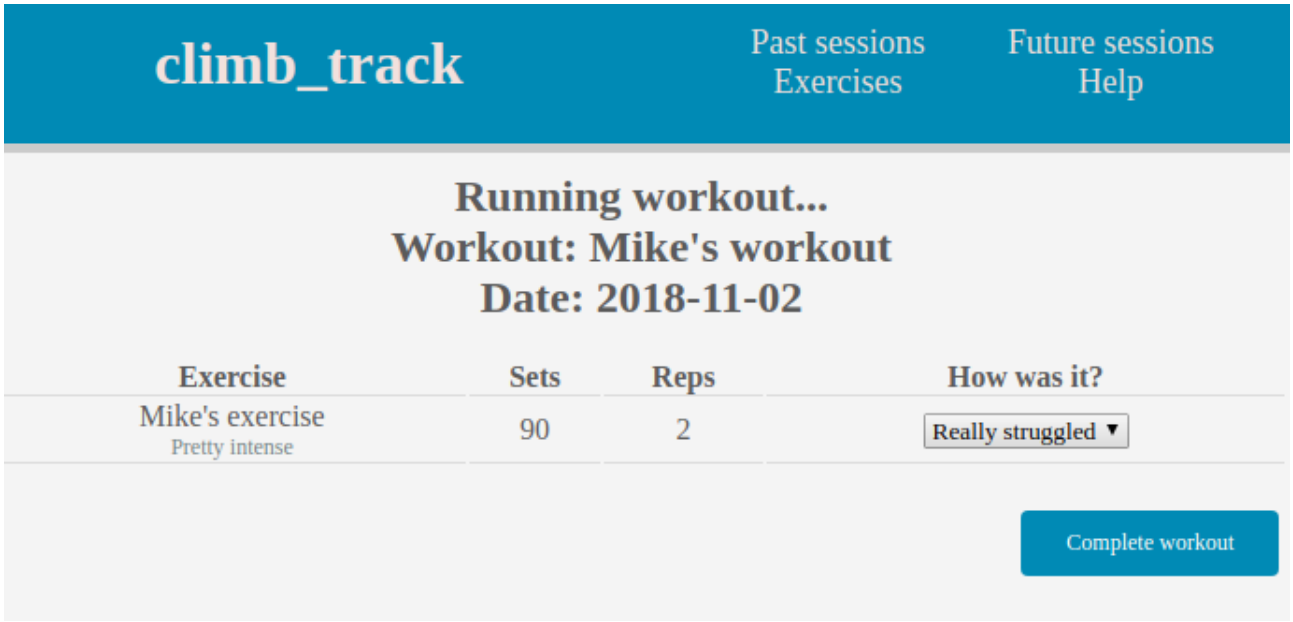


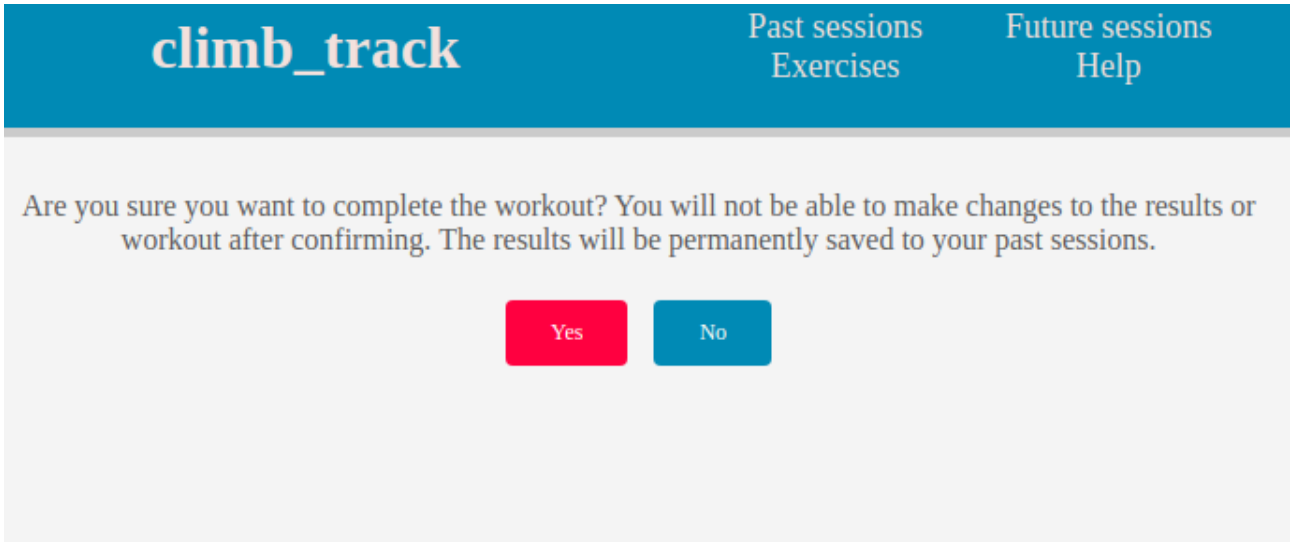
Description here

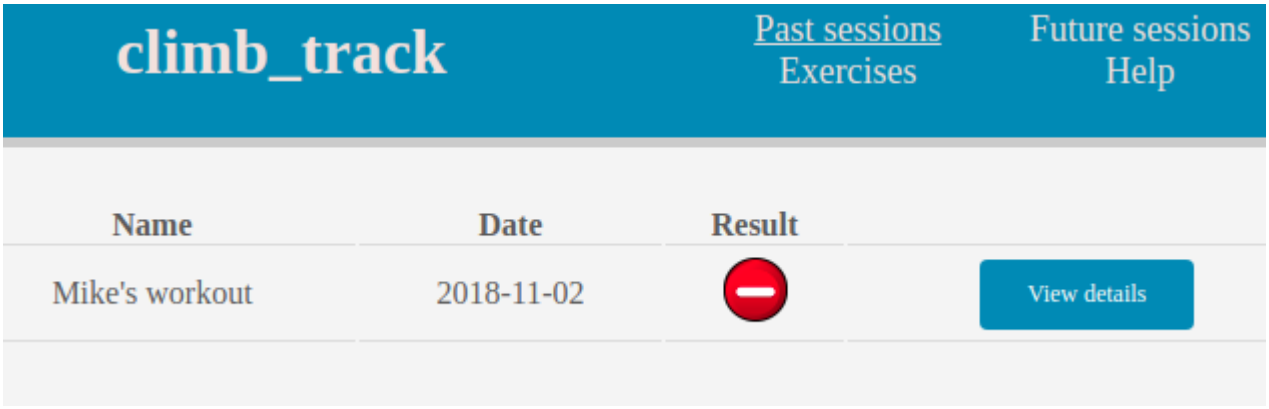
The above shows an exercise being added to the climb_track db via the climb_track program – results are shown using psql at the command line.

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

Paste Screenshot here







Description here

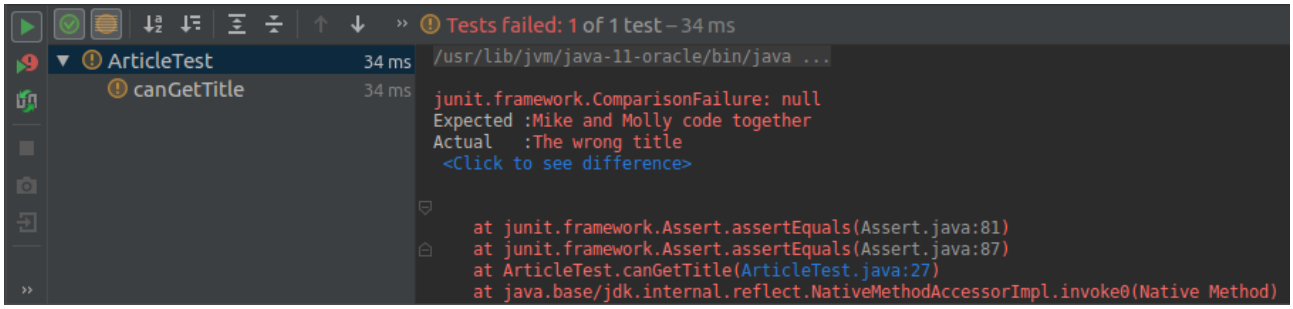
By clicking the complete workout button and confirming the completion of the workout, the workout is marked as complete and moved from the list of future sessions to the list of past sessions.

Unit	Ref	Evidence	
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	

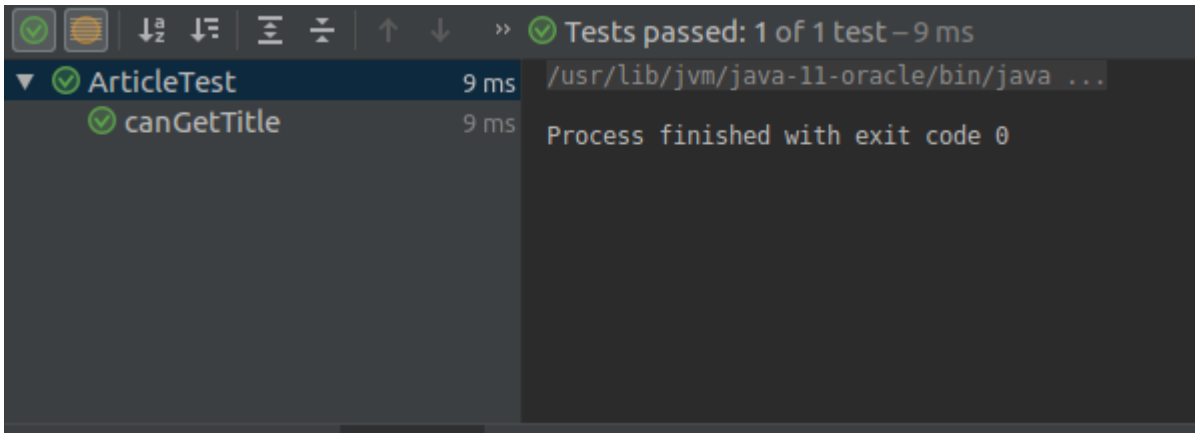
Paste Screenshot here

```
@Before
public void before(){
    section = new Section( title: "Technology");
    author = new Author( name: "Molly");
    article = new Article( title: "Mike and Molly code together",  textContent: "Some content", author);
}

@Test
public void canGetTitle() { assertEquals( expected: "Mike and Molly code together", article.getTitle()); }
```

```
@Column(name = "title")
public String getTitle() {
    return "The wrong title";
}
```




```
@Column(name = "title")
public String getTitle() {
    return title;
}
```



Description here

Above is shown the unit test for a getter, the broken getter, the test failing to pass when run on the broken (hard coded) getter, the fix (return the value of the property from the getter) and the test code passing after the fix.

Week 7

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

Paste Screenshot here

```
import Vehicles.IFlyable;
import java.util.ArrayList;

public class AirportHangar {

    private String name;
    private ArrayList<IFlyable> aircraft;

    public AirportHangar(String name) {
            this.name = name;
            this.aircraft = new ArrayList<IFlyable>();
    }

    public void addAircraft(IFlyable vehicle) {
        this.aircraft.add(vehicle);
    }
}
```

The AirportHangar can have any object added to it that implements the Iflyable interface via addAircraft().

```java
package Vehicles;

public class Plane implements IFlyable {

    private String model;
    private String brand;

    public Plane(String model, String brand) {
        this.model = model;
        this.brand = brand;
    }

    public void fly() {
        System.out.println("Sonic boom...!");
    }
}
```

```java
package Vehicles;

public class Helicopter implements IFlyable {

    private String model;
    private String brand;

    public Helicopter(String model, String brand) {
        this.model = model;
        this.brand = brand;
    }

    public void fly() {
        System.out.println("choppa, choppa, choppa...");
    }
}
```

Both Plane and Helicopter implement the IFlyable interface below.

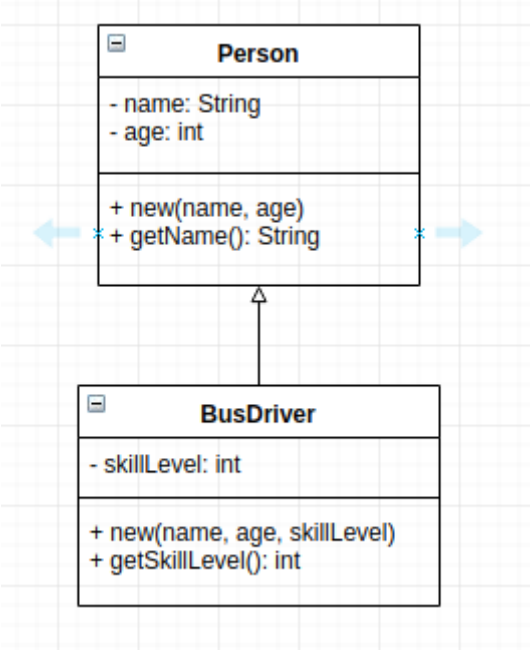
```java
package Vehicles;

public interface IFlyable {

    public void fly();
}
```

Description here

In this example of Polymorphism, both the plane and the helicopter can be treated the same by the AirportHangar when they are being added to the ArrayList of Iflyables as both helicopter and plane implement the interface Iflyable – they both have a fly method.

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

Paste Screenshot here



Description here

The above diagram shows the inheritance hierarchy between a super (Person) and sub class (bus driver). The sub class inherits all of the properties and methods of its parent and has its own in addition.

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

Paste Screenshot here

```
package Animals;

public class Animal {

    private String name;
    private int numberOfLegs;

    public Animal(String name, int numberOfLegs) {
        this.name = name;
        this.numberOfLegs = numberOfLegs;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name){
        this.name = name;
    }

    public int getNumberOfLegs() {
        return numberOfLegs;
    }

    public void setNumberOfLegs(int numberOfLegs) {
        this.numberOfLegs = numberOfLegs;
    }

}
```

Description here

The name and number of legs properties are encapsulated in the Animal class – they are kept private and the only way they are exposed for modification is via the respective setter methods. This means that we can control how and if they are modified. We can also control if they can be read from outside the class with the use of getter methods.

Unit	Ref	Evidence	
I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screen-shots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.	

Paste Screenshot here

```java
package Animals;

public class Animal {

    private String name;
    private int numberOfLegs;

    public Animal(String name, int numberOfLegs) {
        this.name = name;
        this.numberOfLegs = numberOfLegs;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name){
        this.name = name;
    }

    public int getNumberOfLegs() {
        return numberOfLegs;
    }

    public void setNumberOfLegs(int numberOfLegs) {
        this.numberOfLegs = numberOfLegs;
    }

}
```

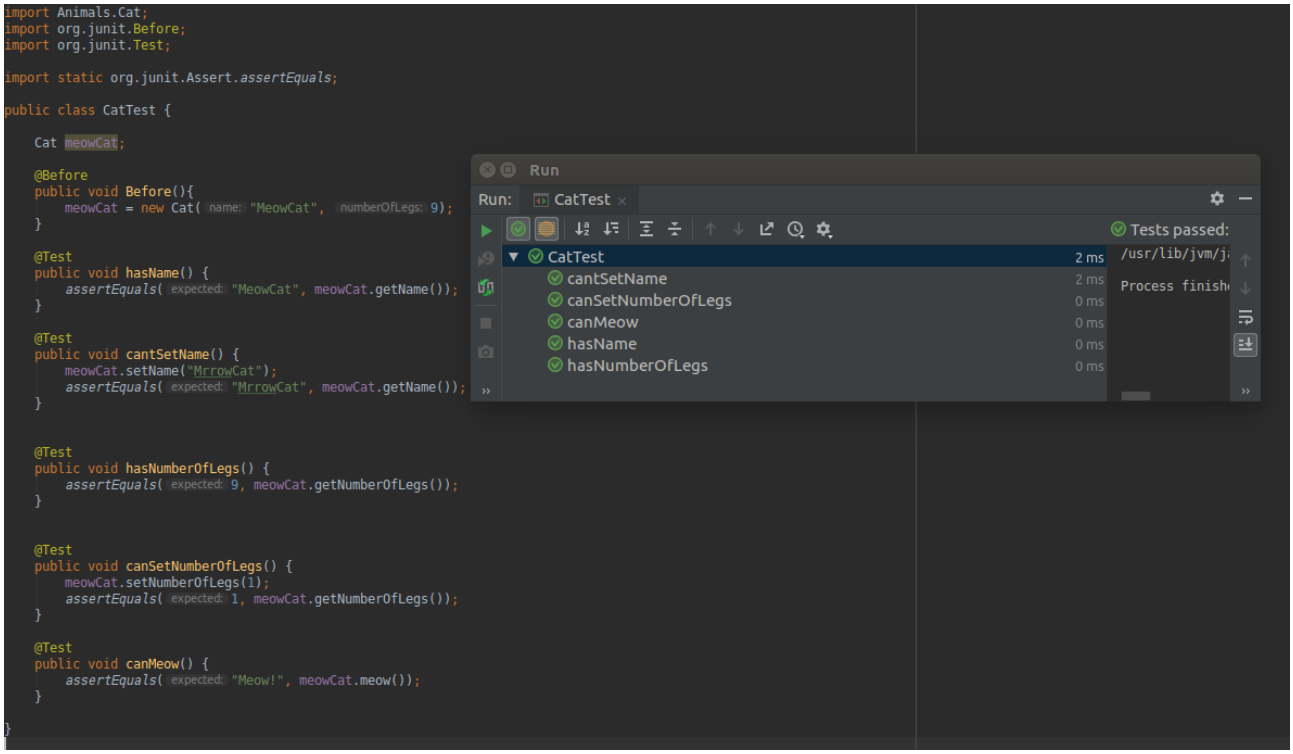
The class

```java
package Animals;

public class Cat extends Animal {

    public Cat(String name, int numberOfLegs){
        super(name, numberOfLegs);
    }

    public String meow(){
        return "Meow!";
    }

}
```

The class that inherits from the previous class

```java
import Animals.Cat;

public class Runner {
    public static void main(String[] args) {
        Cat meowCat = new Cat( name: "MeowCat",  numberOfLegs: 9);
        String meowCatsName = meowCat.getName();
    }
}
```

Tests demonstrating the inherited properties and methods passing:



```
import Animals.Cat;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CatTest {

    Cat meowCat;

    @Before
    public void Before(){
        meowCat = new Cat( name: "MeowCat",  numberOfLegs: 9);
    }

    @Test
    public void hasName() {
        assertEquals( expected: "MeowCat", meowCat.getName());
    }

    @Test
    public void cantSetName() {
        meowCat.setName("MrrowCat");
        assertEquals( expected: "MrrowCat", meowCat.getName());
    }


    @Test
    public void hasNumberOfLegs() {
        assertEquals( expected: 9, meowCat.getNumberOfLegs());
    }


    @Test
    public void canSetNumberOfLegs() {
        meowCat.setNumberOfLegs(1);
        assertEquals( expected: 1, meowCat.getNumberOfLegs());
    }

    @Test
    public void canMeow() {
        assertEquals( expected: "Meow!", meowCat.meow());
    }

}
```

```
Run
Run:  CatTest ×
Tests passed:
CatTest 2 ms
cantSetName 2 ms
canSetNumberOfLegs 0 ms
canMeow 0 ms
hasName 0 ms
hasNumberOfLegs 0 ms
/usr/lib/jvm/j
Process finish
```

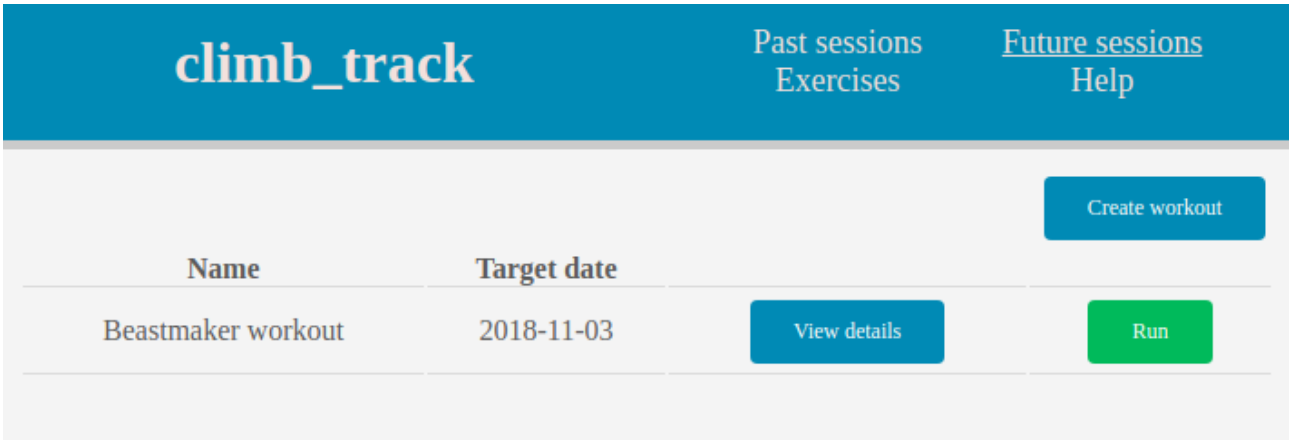
Description here

An Object in the inherited class (Cat)
A Method that uses the information inherited from another class. - we call the getName() method from the Animal class on the Cat object. Tests demonstrating that the inherited properties and methods are available in a Cat object.

Week 10

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

Paste Screenshot here



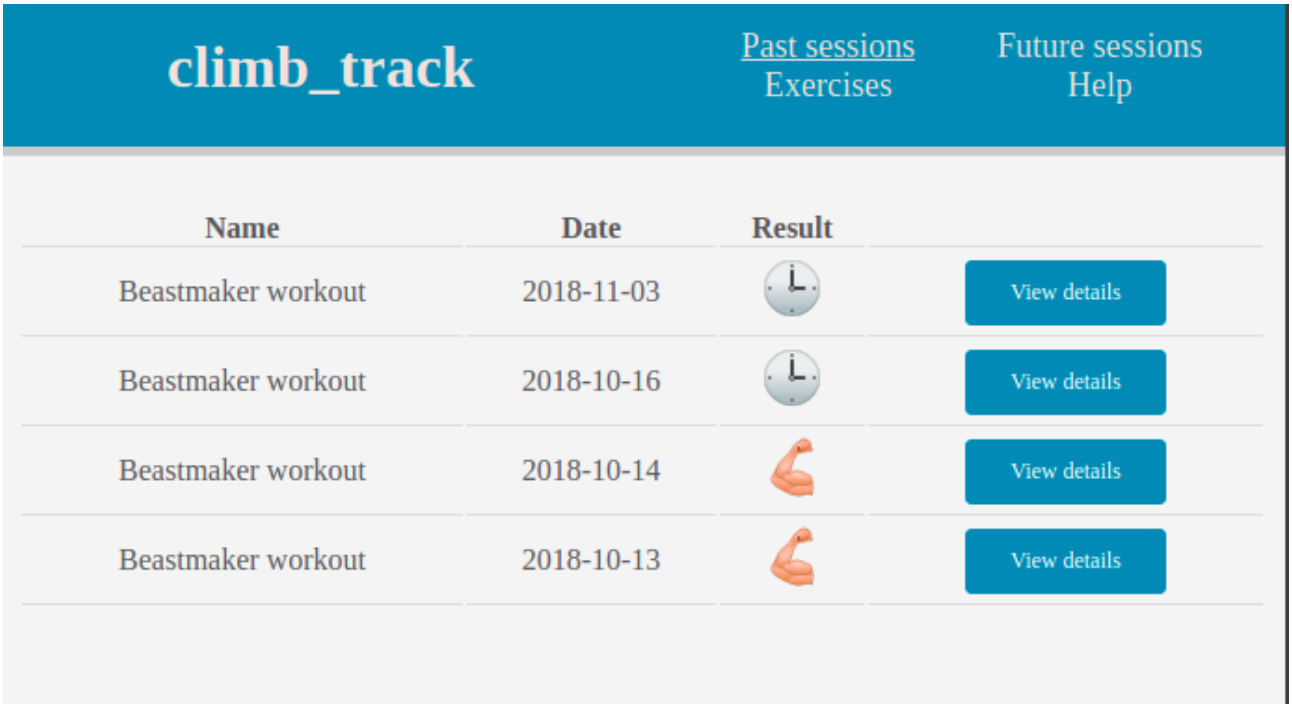
https://github.com/mikethorpe/climb_track

Description here

climb_track solo Ruby project

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

Paste Screenshot here



Description here

Wireframe shows evidence of planning of structure of past sessions view and is demonstrated here in the image of the implemented past sessions view. The result column was updated during development to feature an emoji symbol to indicate how difficult a workout was and a help view was created. Grouping of workouts into blocks of months was dropped as feature due to time constraints.

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

Paste Screenshot here

Isogram finder:

```
const IsogramFinder = function (word) {
    this.word = word;
}

IsogramFinder.prototype.isIsogram = function () {
    let wordLowerCase = this.word.toLowerCase();
    let wordAsArrayOfChars = wordLowerCase.split("");
    const result = wordAsArrayOfChars.every((char) => {
        let numberOfTimesCharacterFound = this.checkNumberOfCharOccurences(wordAsArrayOfChars, char);
        return !(numberOfTimesCharacterFound > 1);
    })
   return result;
}

IsogramFinder.prototype.checkNumberOfCharOccurences = function(charArray, charToMatch){
    const result = charArray.reduce((matches, char) => {
        if (char === charToMatch) {
            return matches += 1
        };
        return matches;
    }, 0)
    return result;
}


module.exports = IsogramFinder;
```

Pangram finder:

```
const PangramFinder = function (phrase) {
  this.alphabet = 'qwertyuiopasdfghjklzxcvbnm'.split('');
  this.phrase = phrase;
}

PangramFinder.prototype.isPangram = function () {
  let phraseLowerCase = this.phrase.toLowerCase();
  const result = this.alphabet.every((letter) => {
    return phraseLowerCase.includes(letter);
  })
  return result;
}

module.exports = PangramFinder;
```

Description here

The pangram finder identifies if a string is a pangram regardless of case. The isogram finder finds words without a repeating letter regardless of case. I chose these as they are both clear examples of problem solving with logic (i.e. algorithms) written in code.

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	

Paste Screenshot here

```
// Forecasts model
const MET_OFFICE_KEY = require('../api_keys/met_office_key');
const RequestHelper = require('../helpers/request_helper');
const PubSub = require('../helpers/pub_sub');

const Forecast = function(){
    this.data = null;
    this.metLocationId = null;
}

Forecast.prototype.bindEvents = function(){
    PubSub.subscribe('SelectCrag:met-location-id', (event) => {
        this.metLocationId = event.detail;
        this.getFiveDayThreeHrData();
    })
}

Forecast.prototype.getFiveDayThreeHrData = function(){
    const url = `http://datapoint.metoffice.gov.uk/public/data/val/wxfcs/all/json/${this.metLocationId}?res=3hourly&key=${MET_C

    console.log(url);

    const requestHelper = new RequestHelper(url);
    this.data = requestHelper.get((data) => {
        this.data = data.SiteRep;
        PubSub.publish('Forecast:fiveday-threehour-data', this.data);

    });

}

module.exports = Forecast;
```

Select a crag: Aberdour - Hawkraig ▾

Day	Temperature	Probability of rain
Today	10	6%
Today	10	6%
Today	10	54%
Tomorrow	10	47%
Tomorrow	10	15%
Tomorrow	8	6%
Tomorrow	9	4%
Tomorrow	10	0%
Tomorrow	10	7%
Tomorrow	9	60%
Tomorrow	9	81%

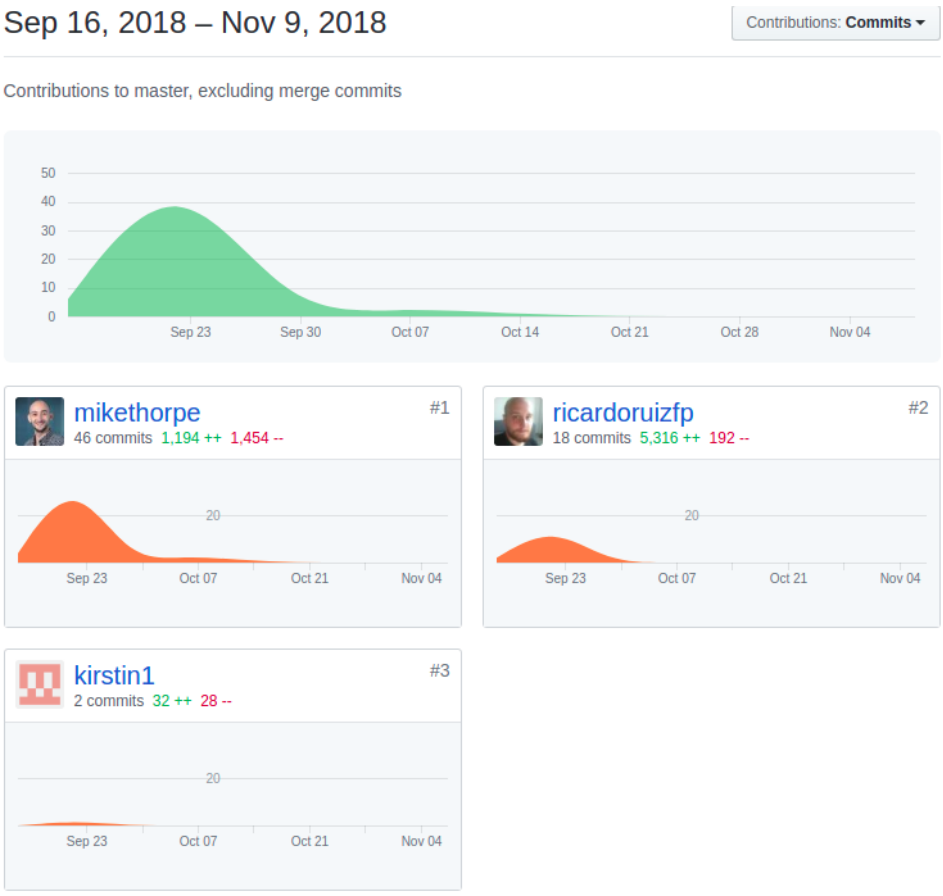
Description here

The above shows the code that gets the data from the API and rebroadcasts it to a view where it can be rendered. The rendering of the data in the view as the program runs is also shown above.

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.	

Paste Screenshot here



Description here

Contributions to the JS team project

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

Paste Screenshot here

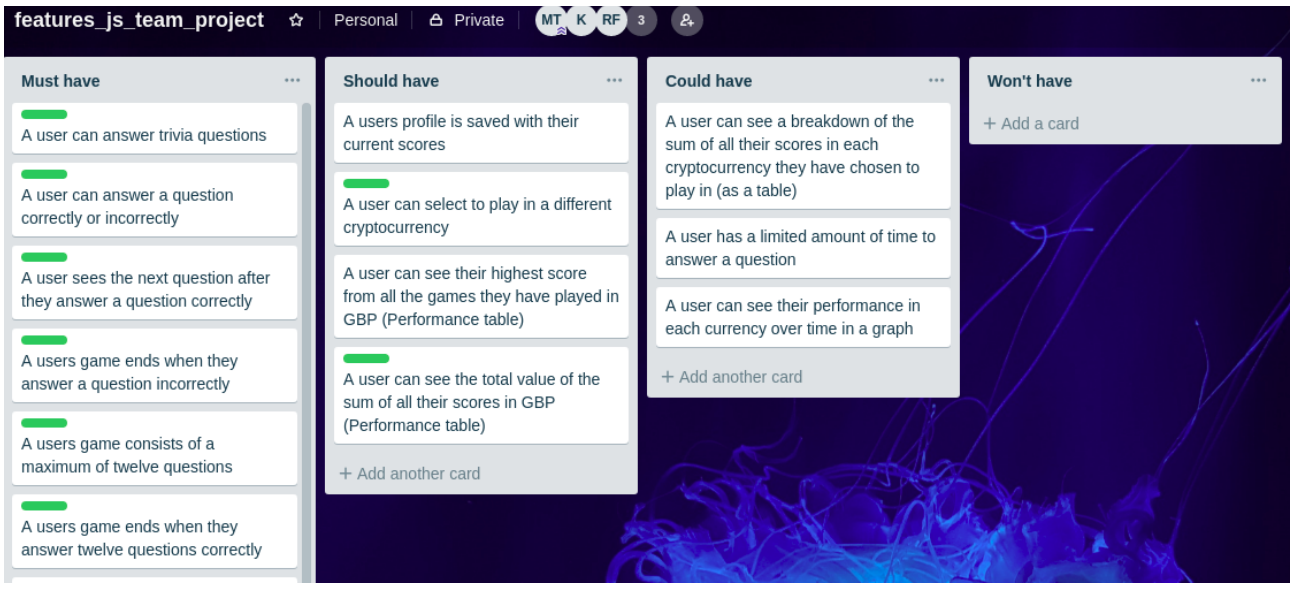
```
Initial requirements:

Minimum viable product features:
* A user can answer trivia questions
* A user can be rewarded with points for answering correctly
* A user can answer a question correctly or incorrectly
* A user can see the correct answer only after answering the question
* A users game ends when they answer all questions correctly
* A users game ends when they answer a question incorrectly
* A users game consists of a maximum of twelve questions
* A users score can be measured in bitcoin
* A users score is displayed to them at the end of the game
* A user can see their current score at any time
* A users score increases when they answer questions correctly
* A user can start a new game from the home view
* A user can start a new game from the end game view
* A user can answer a question
* A user sees the next question after they answer a question correctly
* A user sees a summary of their game screen when it ends

Extension features:
* A users profile is saved with their current scores
* A user can select to play in a different cryptocurrency
* A user can see their highest score from all the games they have played in GBP (Performance table)
* A user can see the total value of the sum of all their scores in GBP (Performance table)
* A user can see a breakdown of the sum of all their scores in each cryptocurrency they have chosen to play in (as a table)
* A user can see their performance in each currency over time in a graph
```

Description here

JS Team project brief – a list of initial requirements

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

Paste Screenshot here



Description here

Trello MOSCOW board from JS team project showing features and their relative priority for implementation

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

Paste Screenshot here

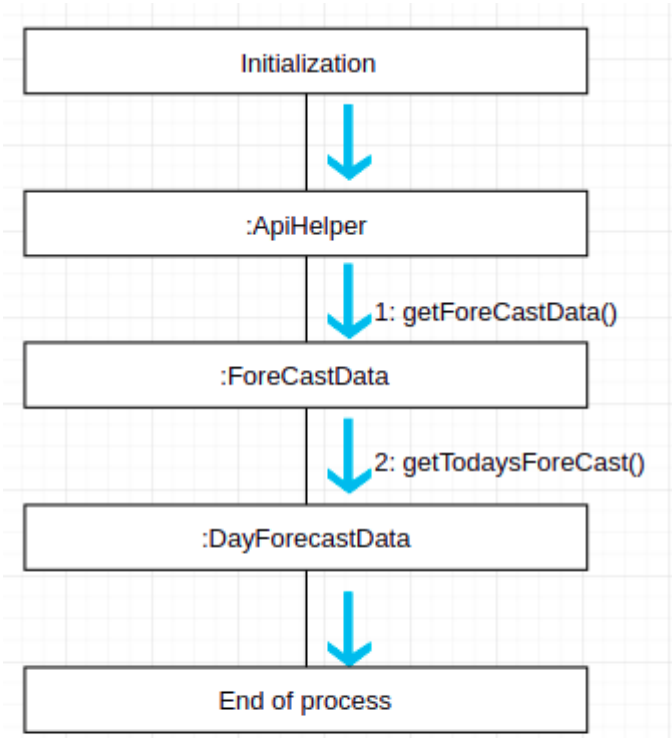
Acceptance criteria	Expected result / output	Pass / Fail
A user can create a new exercise	Given the create new exercise view has been loaded. And exercise details have been entered. When create exercise is clicked. Then the exercise is added to a list of all exercises	Pass
A user can change the name of an existing exercise	Given the edit page for an existing exercise has been loaded. And the new exercise name has been entered When update exercise is clicked Then the name of the exercise is updated	Fail
A user can delete an exercise that is not associated with a workout	Given the exercises page has been loaded And an exercise that is not associated with a workout exists When the option to delete the exercise is chosen And the option is confirmed Then the exercise is deleted	Pass
A user can view the details of an exercise	Given the exercises page has been loaded And an exercise exists When the option to view the exercise is selected The the details of the exercise are displayed	Pass

Description here

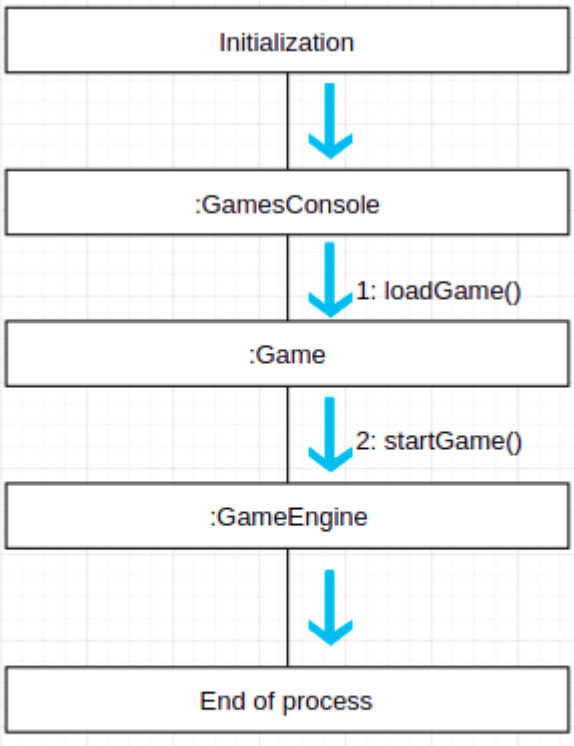
Above is an acceptance criteria and test plan. Acceptance criteria – workflows that theh application should satisfy are displayed in the left hand column. The middle column shows the test to be performed to ensure the acceptance criteria are met. The right hand column indicates whether or not the test passed on its last run.

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

Paste Screenshot here



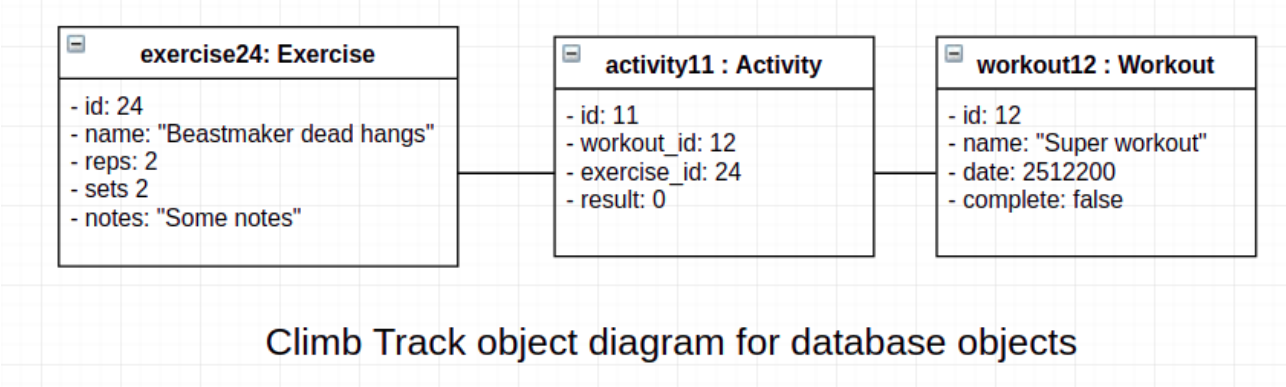
Collaboration diagram showing API request for



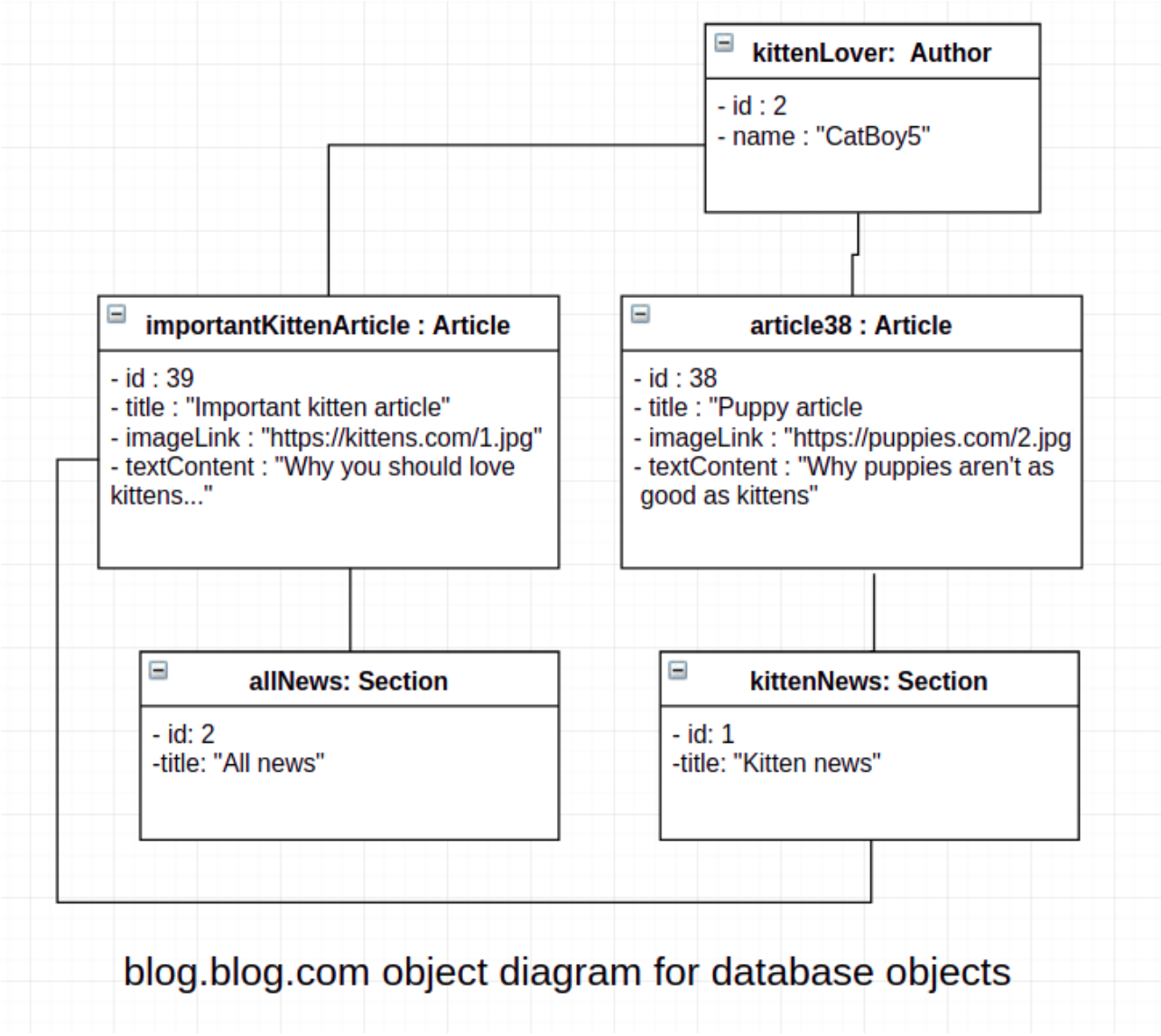
Collaboration diagram showing loading and starting game from virtual games console

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

Paste Screenshot here



Climb Track object diagram for database objects



blog.blog.com object diagram for database objects

Description here

The above shows object diagrams for the climb track (ruby project) db objects – all have a 1 to 1 relationship, and the blog.blog.com (java project) db objects – an author can have many articles and each article can be associated with many sections. Each object has been instantiated with values for all properties.

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

Paste Screenshot here

Bug / Error:	Solution:	Date:
Name of new exercise is not displayed on exercise details or list view	'name' param in post request incorrectly referenced as Name in exercise.rb model options. Now referenced correctly.	30/10/2018
Foreign key violation error thrown when attempting to delete and exercise that is associated with a workout	Modified design and implementation so delete button is not shown for exercises that are associated with a workout(s). No longer possible for user to delete exercise associate with a workout.	25/07/2018
Trying to add exercise to workout when no exercises exist throws error	Exercise id is passed as nil from selector in update_activities.erb when no exercises exist. Added conditional statement in workouts_controller.rb to check exercise id is not nil before adding exercise to workout.	26/07/2018
Exercise reps not displayed on list of exercises	Reps not referenced from params in show exercises/index.erb table. Reference to exercises.reps in params updated. Fixed	10/10/2018
Exercise can be created with negative number of sets	Min attribute added to number input html tag for sets input in exercises/new.erb	10/10/2018

Description here

Bug tracking report show above demonstrating a description of the bug / error without technical assumption of the cause. Solution indicates the technical solution including design and implementation changes.