CODECLAN

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.

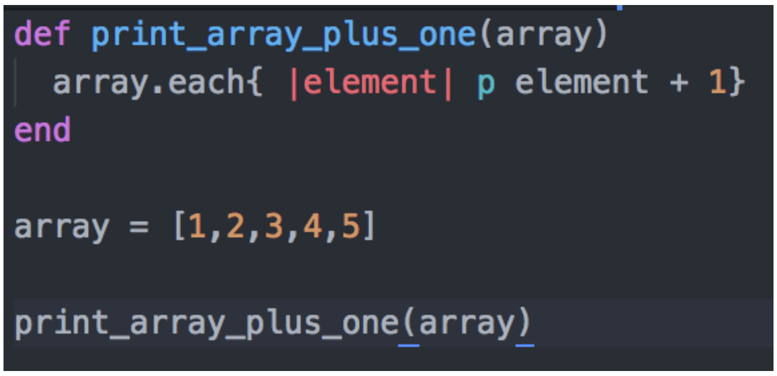
Each point that required details the Assessment Criteria (What you have to show) along with a brief description of the kind of things you should be showing.

Please fill in each point with screenshot or diagram and description.

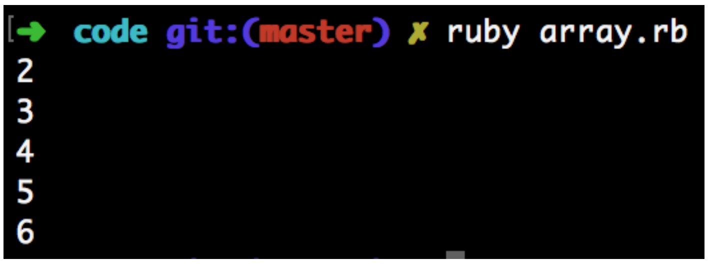
Week 2

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)
```



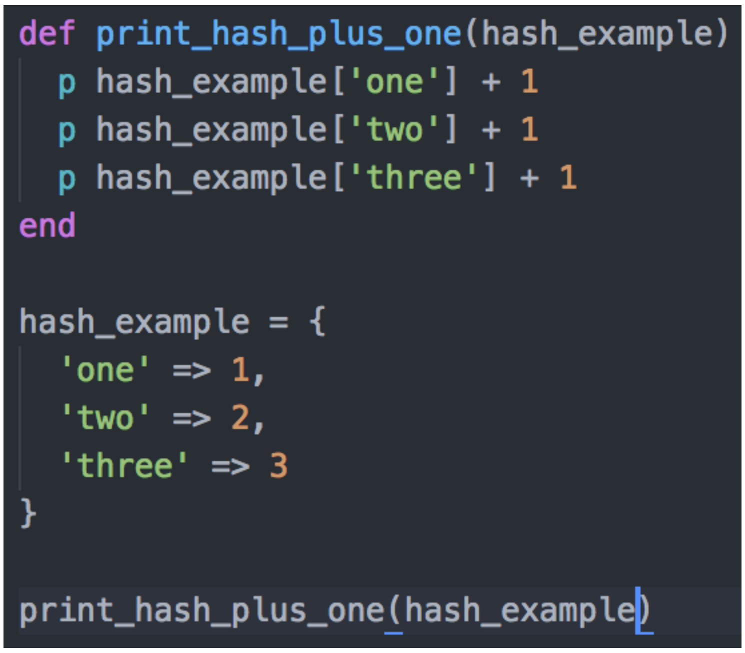
```
[➜  code git:(master) ✗ ruby array.rb
2
3
4
5
6
```

Description here

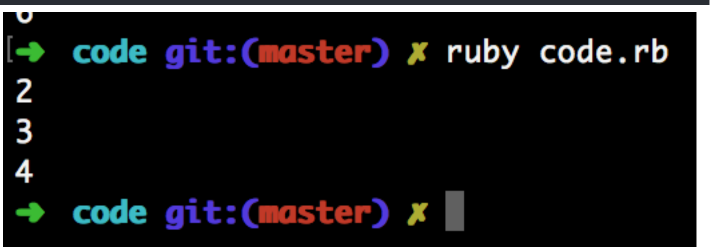
Creates an array of integers. Function prints each of those integers + 1.

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) ✗ 
```

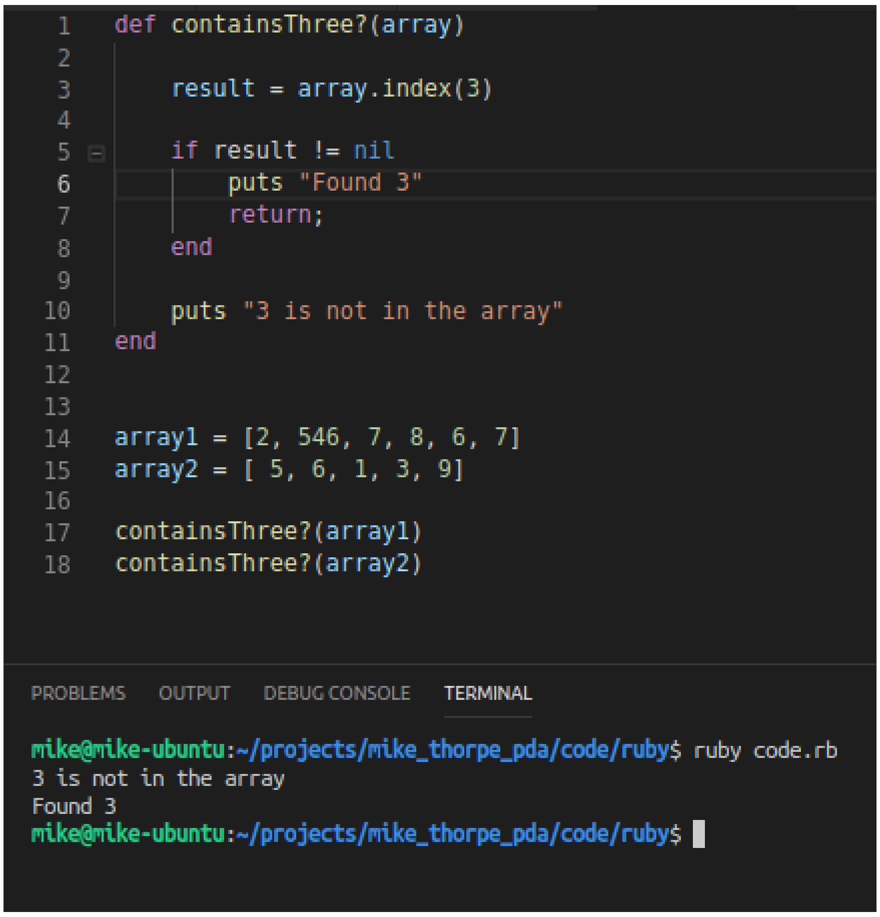
Description here

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



```ruby
def containsThree?(array)

    result = array.index(3)

    if result != nil
        puts "Found 3"
        return;
    end

    puts "3 is not in the array"
end


array1 = [2, 546, 7, 8, 6, 7]
array2 = [ 5, 6, 1, 3, 9]

containsThree?(array1)
containsThree?(array2)
```

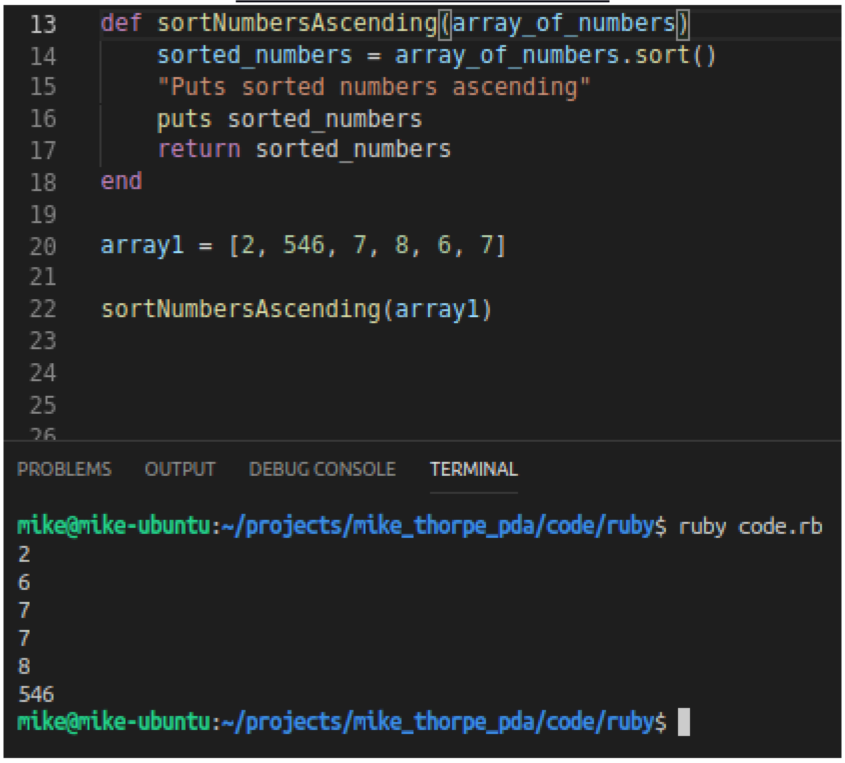
```
mike@mike-ubuntu:~/projects/mike_thorpe_pda/code/ruby$ ruby code.rb
3 is not in the array
Found 3
mike@mike-ubuntu:~/projects/mike_thorpe_pda/code/ruby$
```

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



```
def sortNumbersAscending(array_of_numbers)
    sorted_numbers = array_of_numbers.sort()
    "Puts sorted numbers ascending"
    puts sorted_numbers
    return sorted_numbers
end

array1 = [2, 546, 7, 8, 6, 7]

sortNumbersAscending(array1)
```

```
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$
```

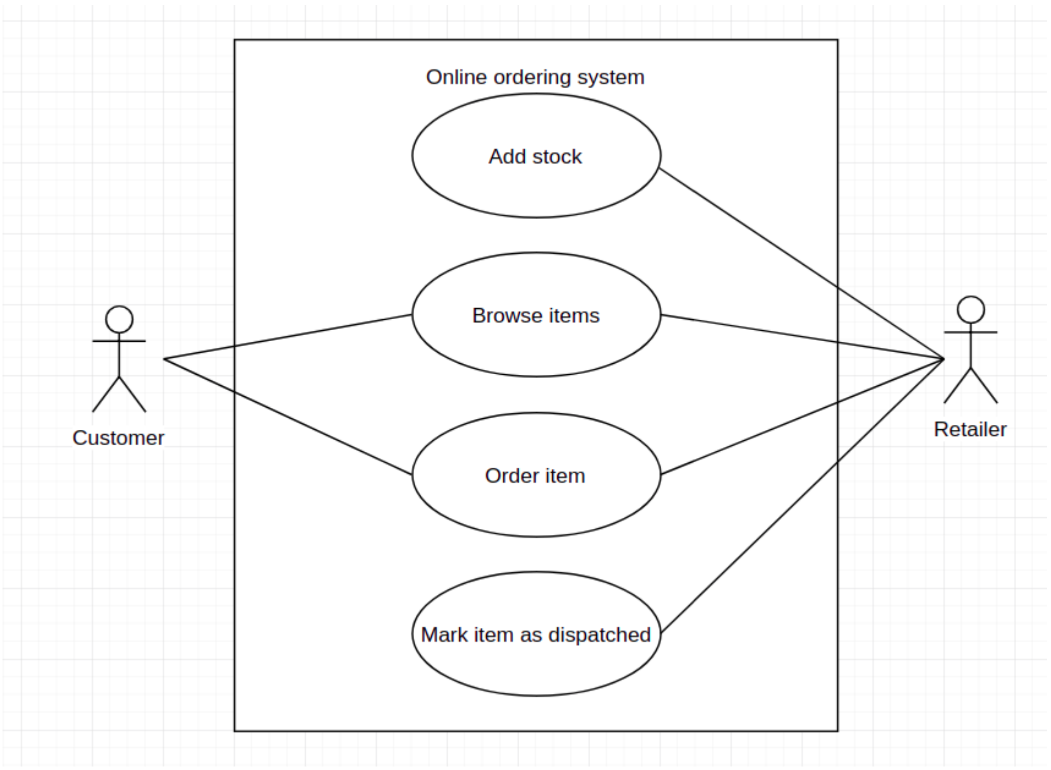
Description here

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

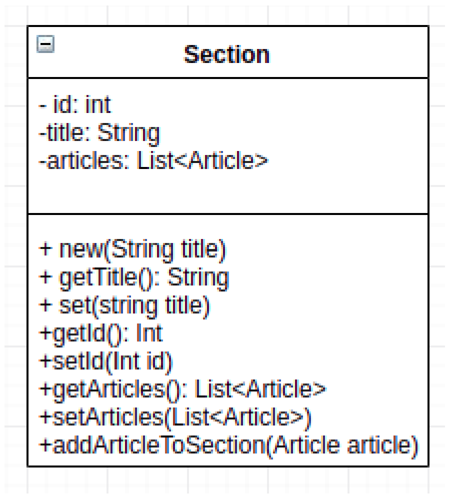


Description here

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

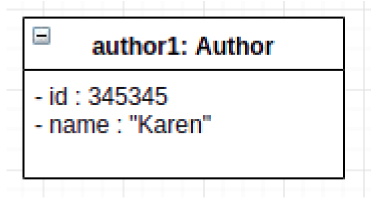


Description here

The above shows a diagram of a class – 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.

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

Paste Screenshot here

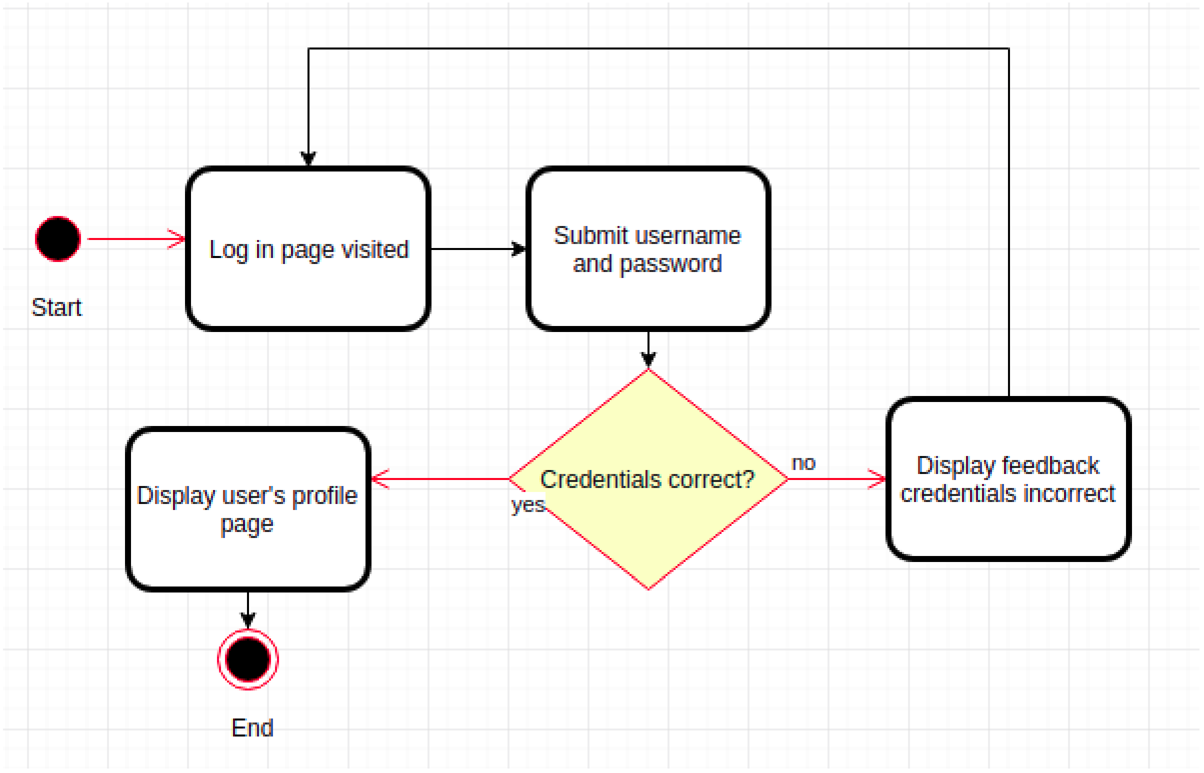


Description here

The above shows an object diagram, indicating the values of its properties and the Class it was instantiated from.

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

Paste Screenshot here


Start
Log in page visited
Submit username and password
Credentials correct?
no
yes
Display feedback credentials incorrect
Display user's profile page
End

Description here

Above shows an activity diagram of the process of logging on to a system. Activites are in rectangular boxes, conditionals in diamonds and start and termination points are represented by circles.

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	Application must be cross platform	Create web app based solution capable of running in any modern browser
Performance requirements	App must start within 20s and all user requests must complete within 2s	Hosting web app on Heroku should give specified performance

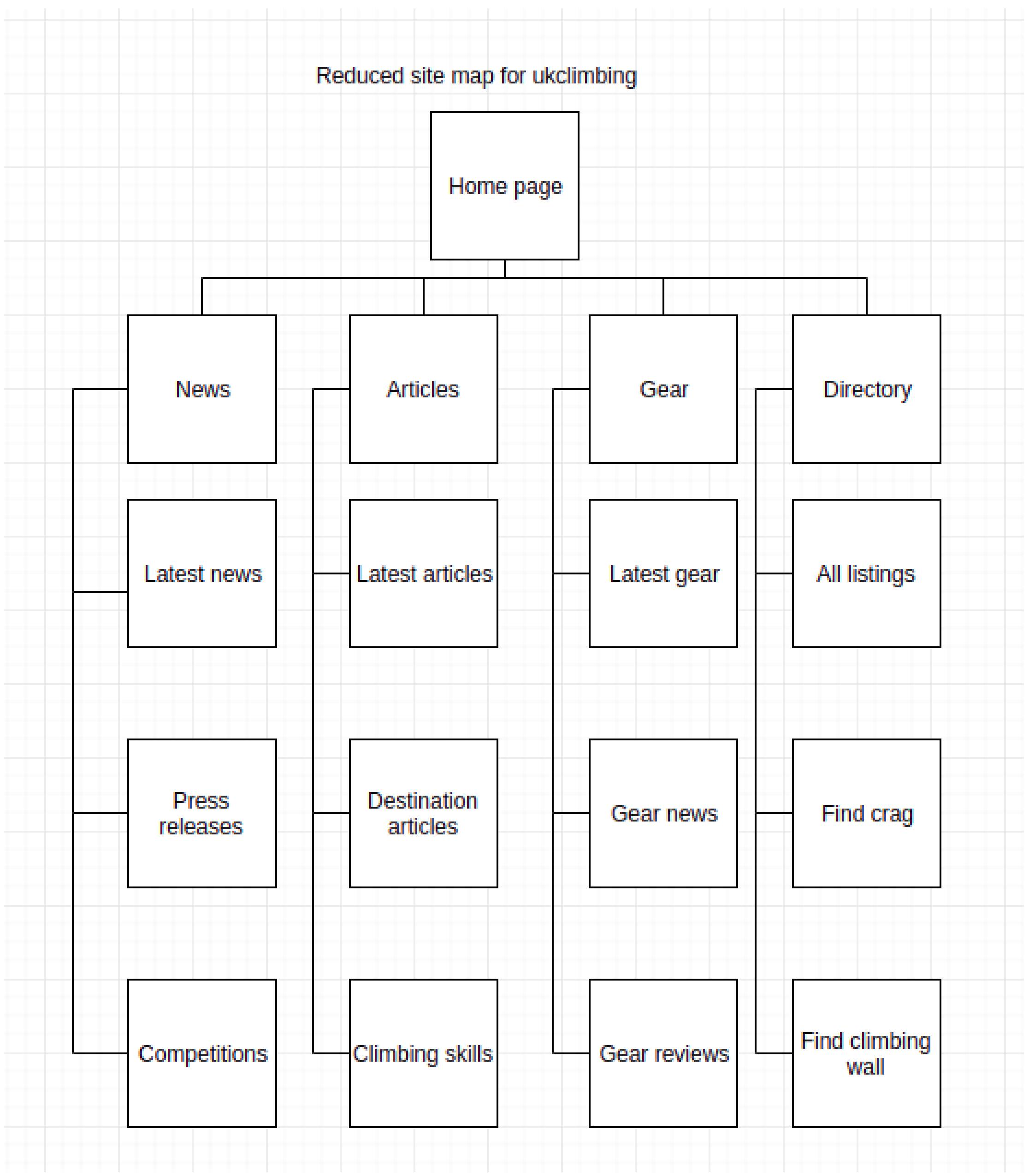
Persistent storage and transactions	Exercise and workout data must be stored persistently. One workout can have many exercises and each exercise can be associated with many workouts.	Persist data via relational database (Postgres). Transactions to and from the db should be facilitated by restful routes.
Usability	A new user can quickly learn how to use the app	Implement a help view which is always accessible via the nav bar with a guide on how to use the app
Budgets	No monetary budget for project	Use only free open source tools and libraries for development
Time	Application must be produced in 7 days	Tight timescale so planning must be stringent to only include necessary features in MVP. Use of Trello to prioritize features. 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

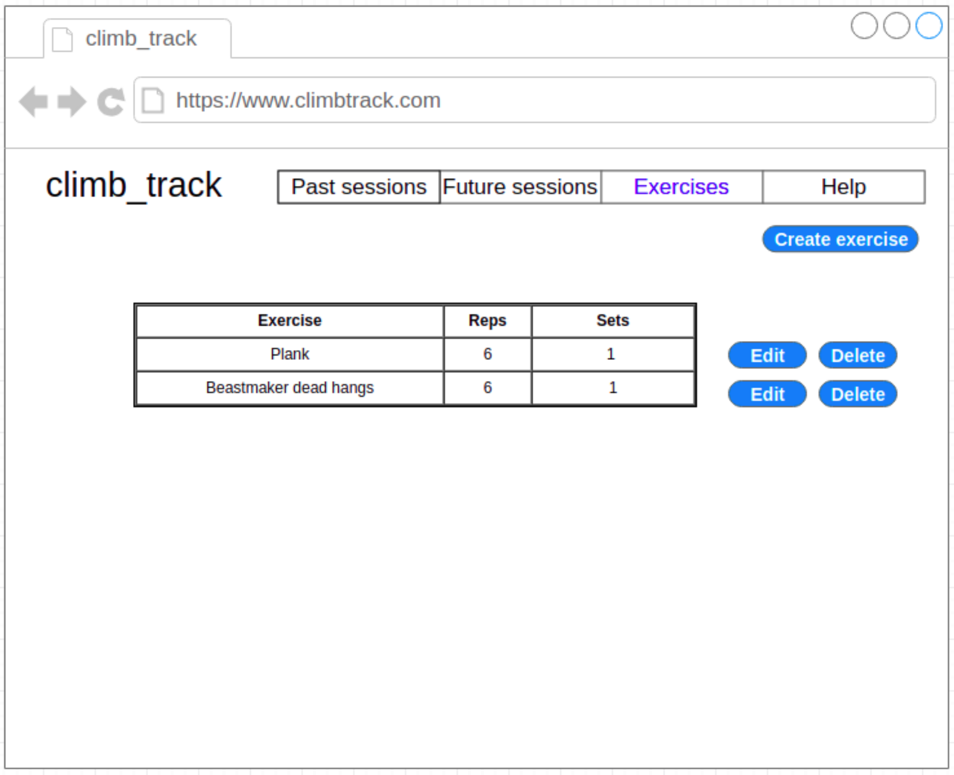


Description here

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

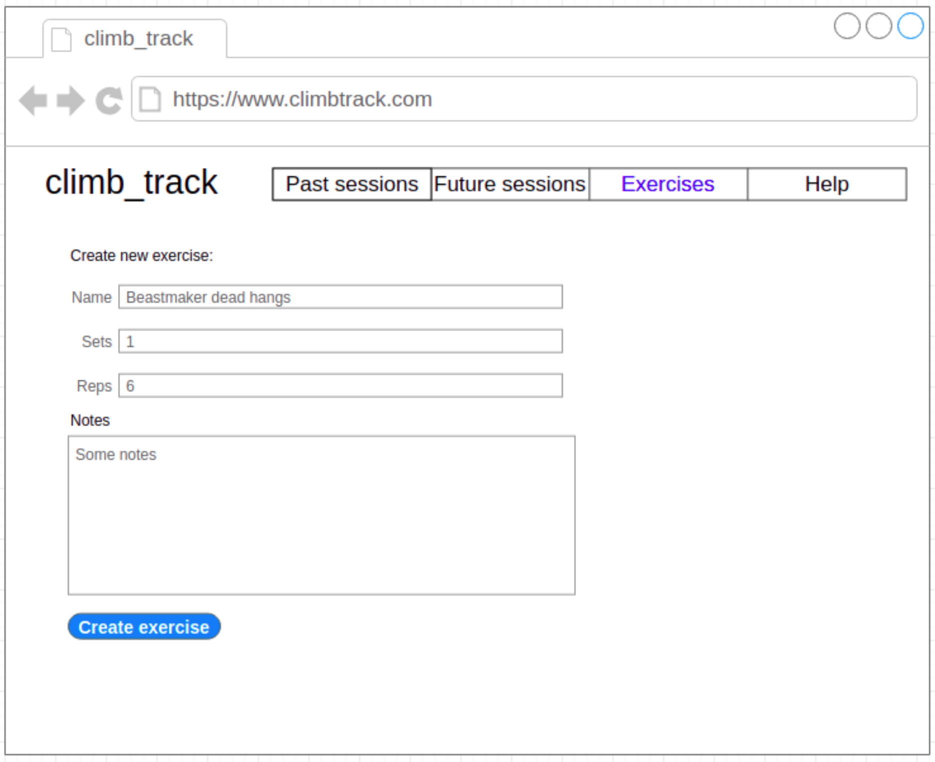


climb_track

https://www.climbtrack.com

climb_track

Exercise	Reps	Sets
Plank	6	1
Beastmaker dead hangs	6	1



climb_track

https://www.climbtrack.com

climb_track

Create new exercise:

Name	Beastmaker dead hangs
Sets	1
Reps	6

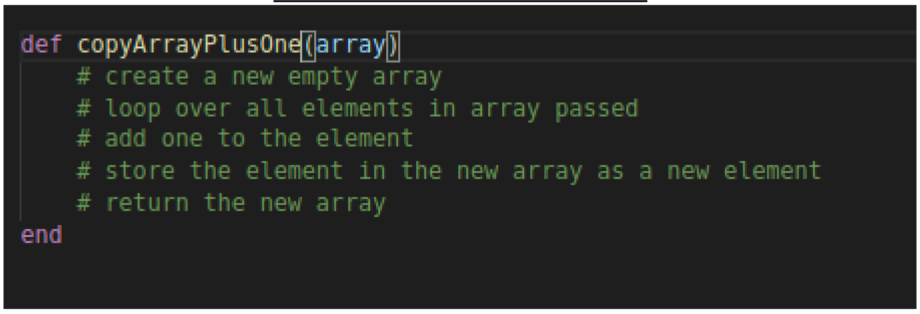
Notes

Some notes

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

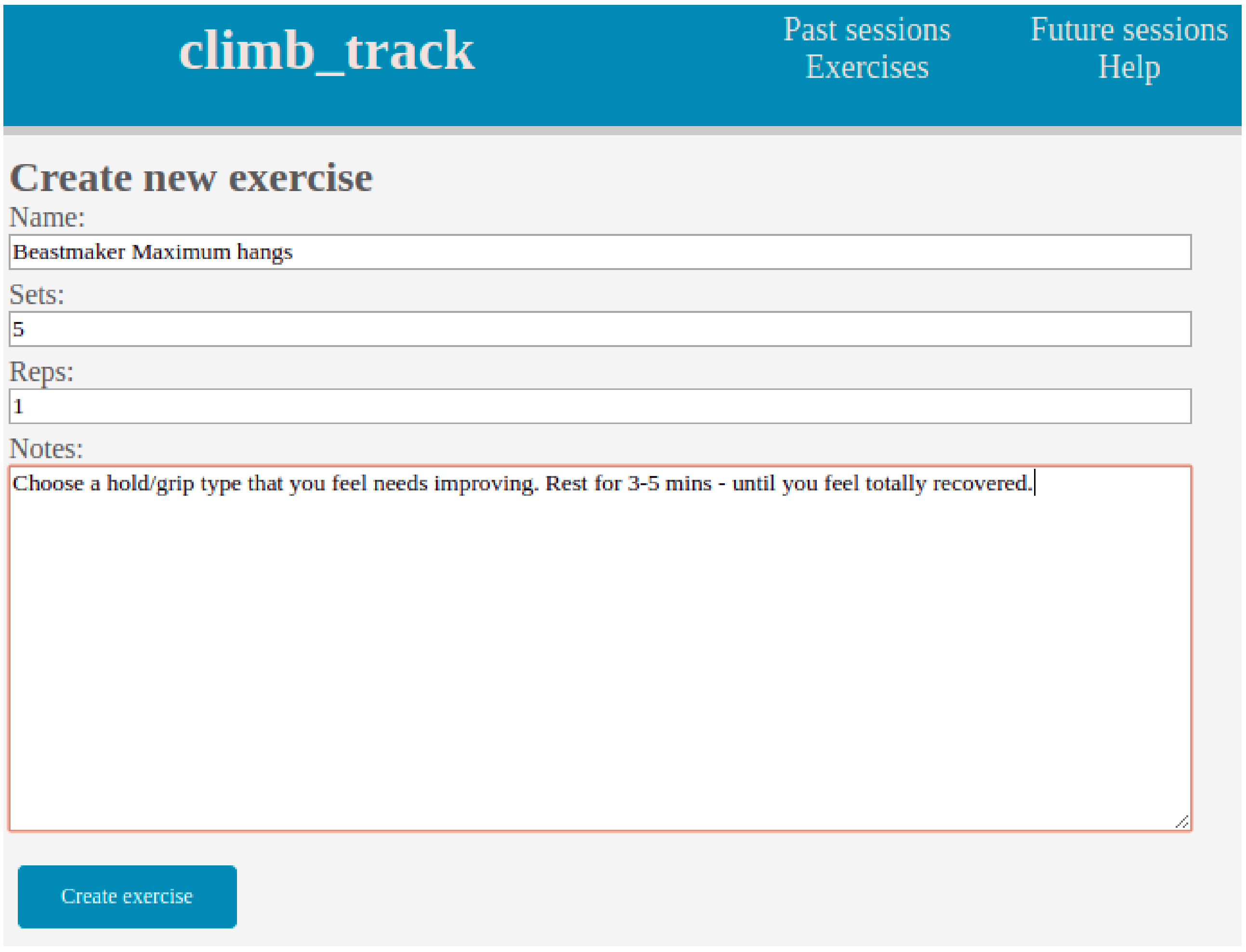


```
def copyArrayPlusOne(array)
    # create a new empty array
    # loop over all elements in array passed
    # add one to the element
    # store the element in the new array as a new element
    # return the new array
end
```

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



climb_track

Past sessions
Exercises
Future sessions
Help

Create new exercise

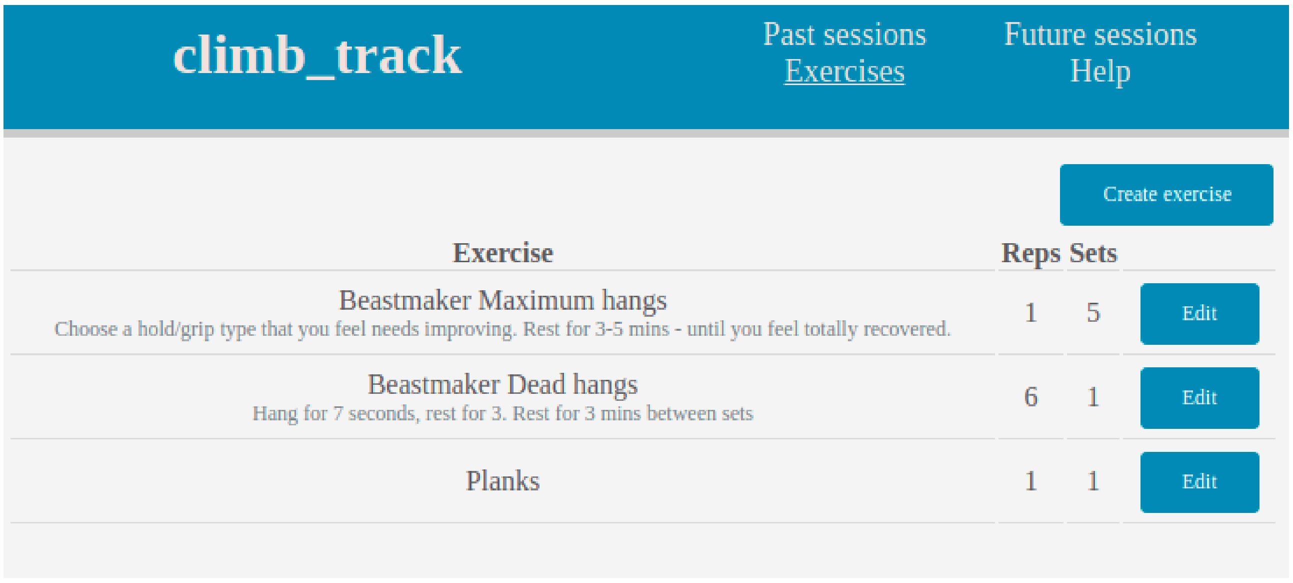
Name:
Beastmaker Maximum hangs

Sets:
5

Reps:
1

Notes:
Choose a hold/grip type that you feel needs improving. Rest for 3-5 mins - until you feel totally recovered.

Create exercise



climb_track

Past sessions
Exercises
Future sessions
Help

Create exercise

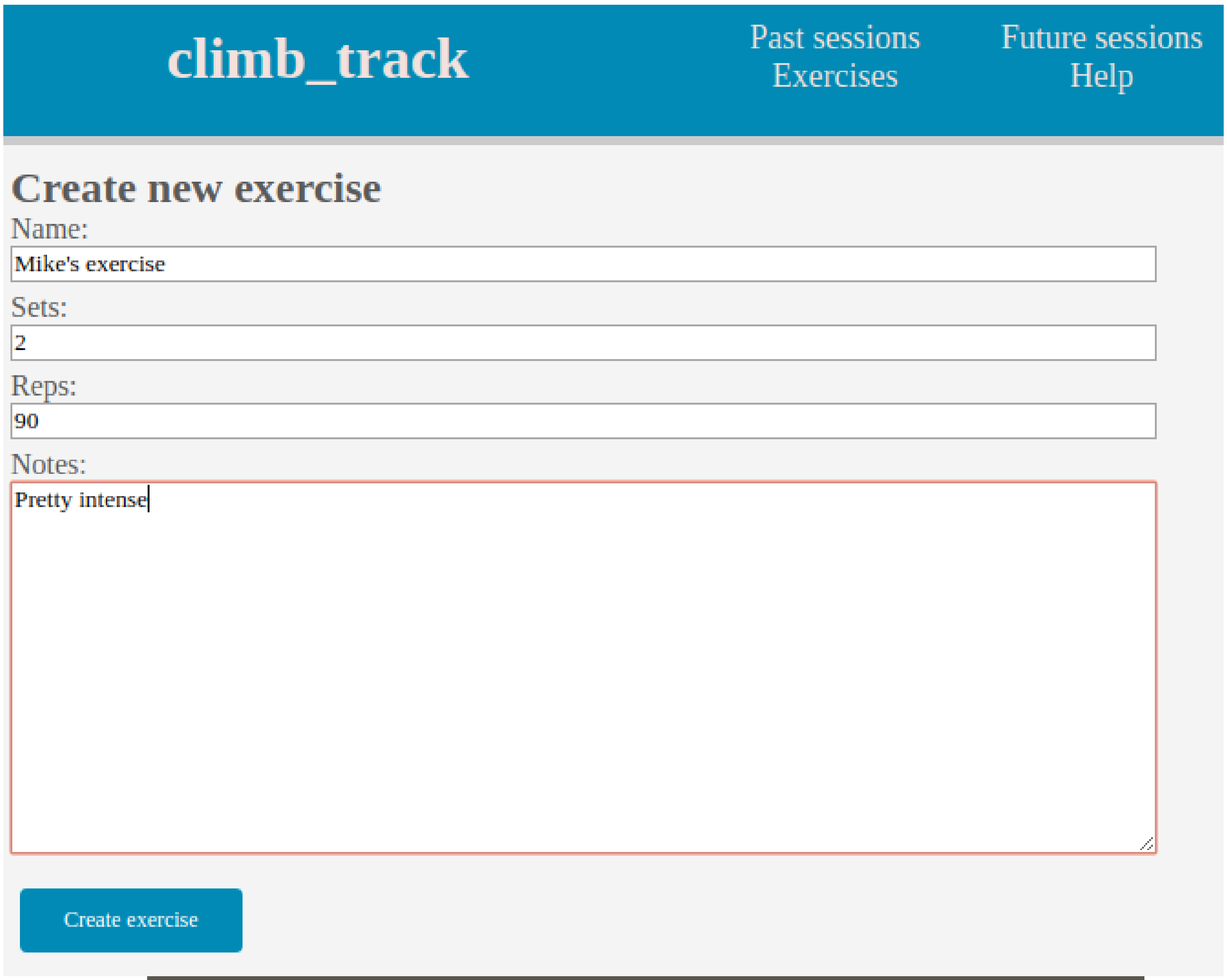
Exercise	Reps	Sets	
Beastmaker Maximum hangs Choose a hold/grip type that you feel needs improving. Rest for 3-5 mins - until you feel totally recovered.	1	5	Edit
Beastmaker Dead hangs Hang for 7 seconds, rest for 3. Rest for 3 mins between sets	6	1	Edit
Planks	1	1	Edit

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



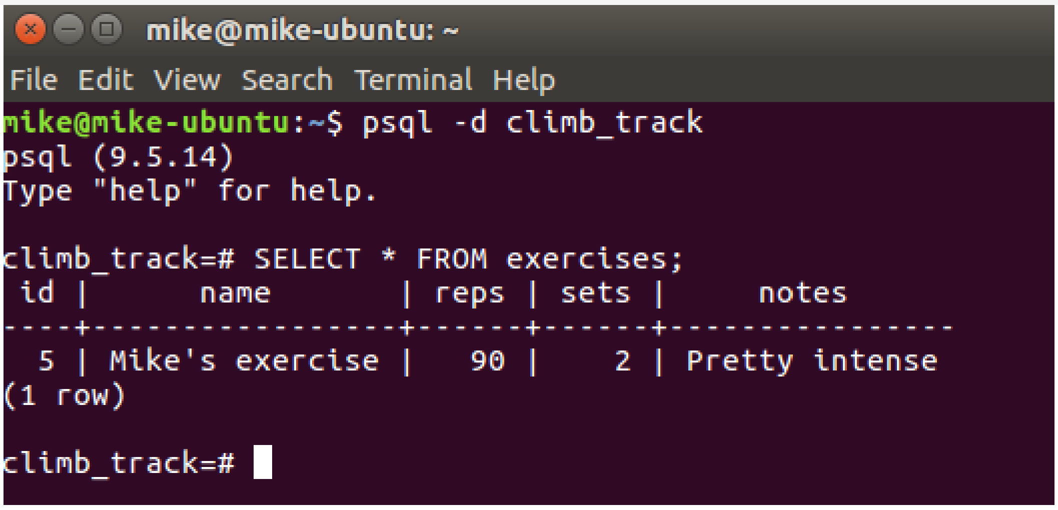
Create new exercise

Name:
Mike's exercise

Sets:
2

Reps:
90

Notes:
Pretty intense



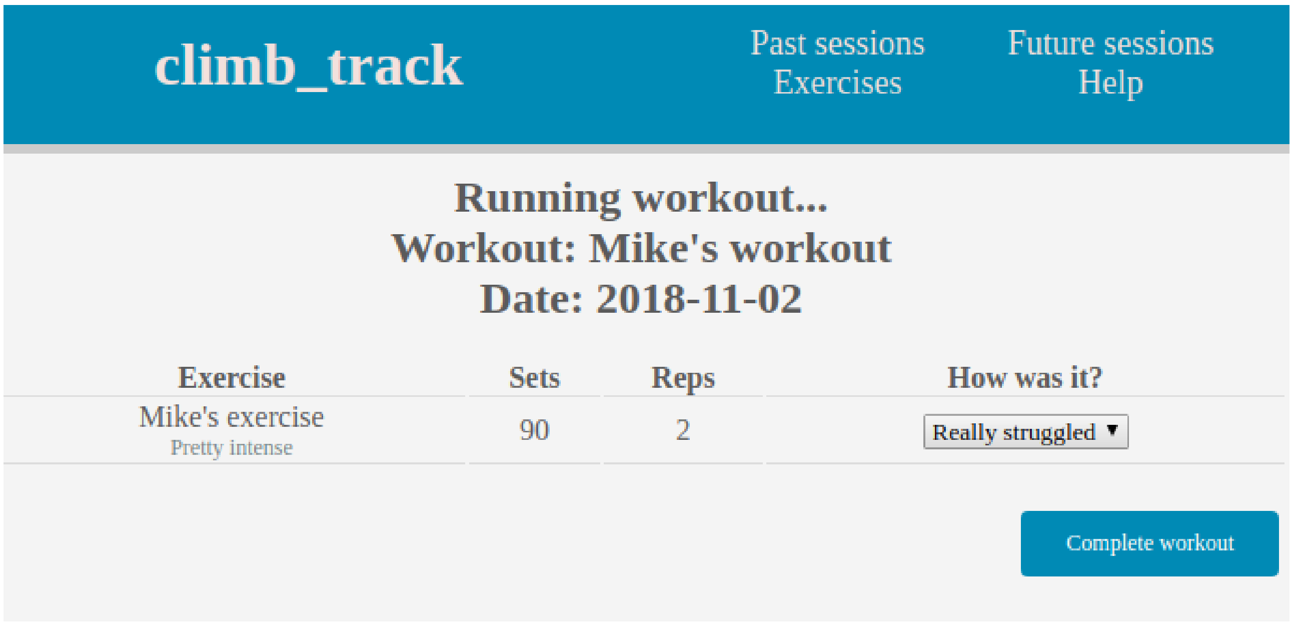
```
mike@mike-ubuntu:~$ psql -d climb_track
psql (9.5.14)
Type "help" for help.

climb_track=# SELECT * FROM exercises;
 id |       name        | reps | sets |      notes
----+-------------------+------+------+----------------
  5 | Mike's exercise |   90 |    2 | Pretty intense
(1 row)

climb_track=# 
```

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.

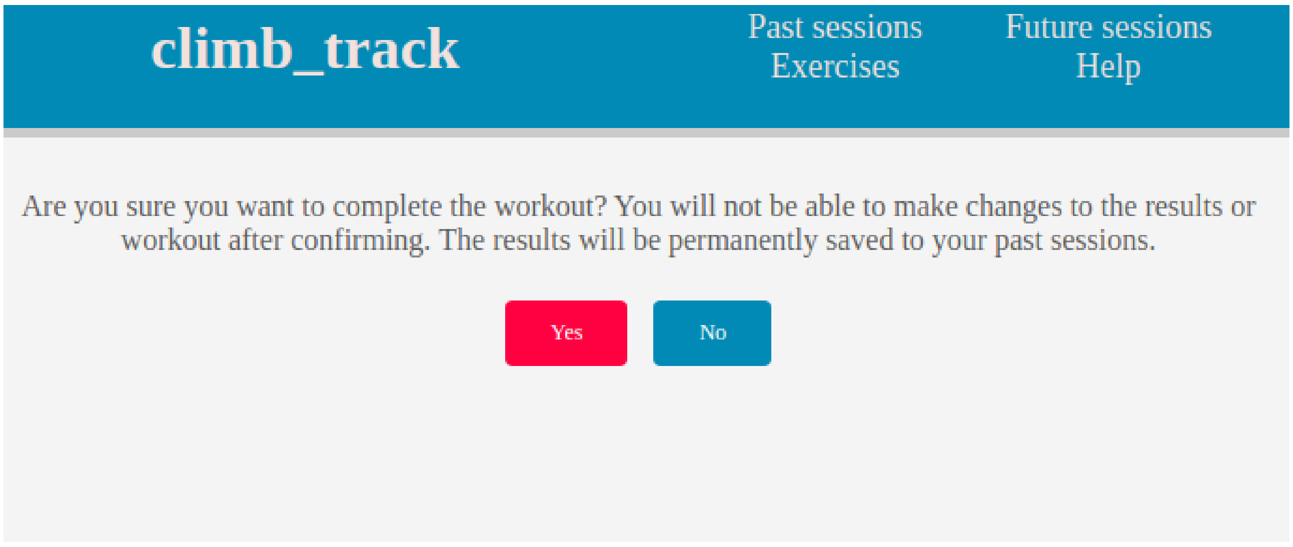
Unit	Ref	Evidence	
P	P.15	Show the correct output of results and feedback to user. Take a screen-shot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program	

Paste Screenshot here

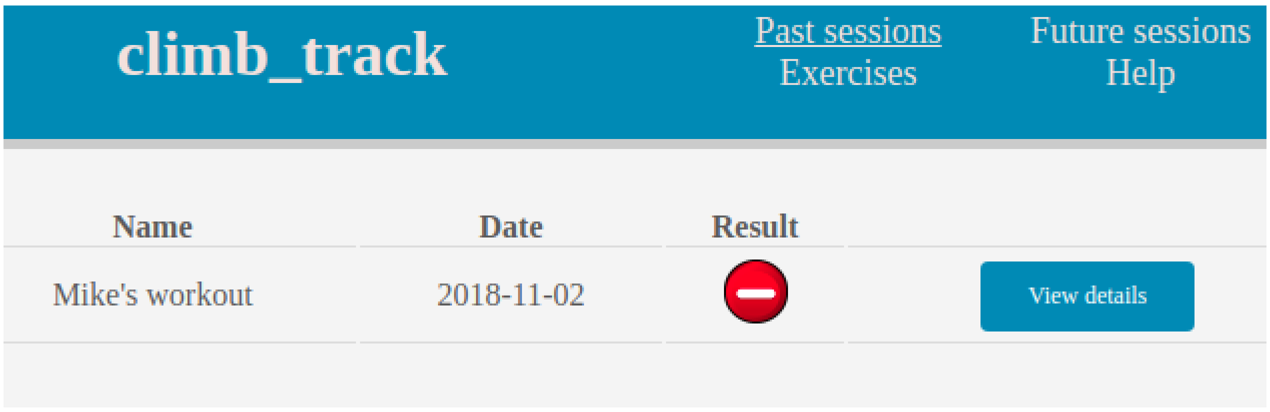


Running workout...
Workout: Mike's workout
Date: 2018-11-02

Exercise	Sets	Reps	How was it?
Mike's exercise Pretty intense	90	2	Really struggled



Are you sure you want to complete the workout? You will not be able to make changes to the results or workout after confirming. The results will be permanently saved to your past sessions.


climb_track

Past sessions
Exercises
Future sessions
Help

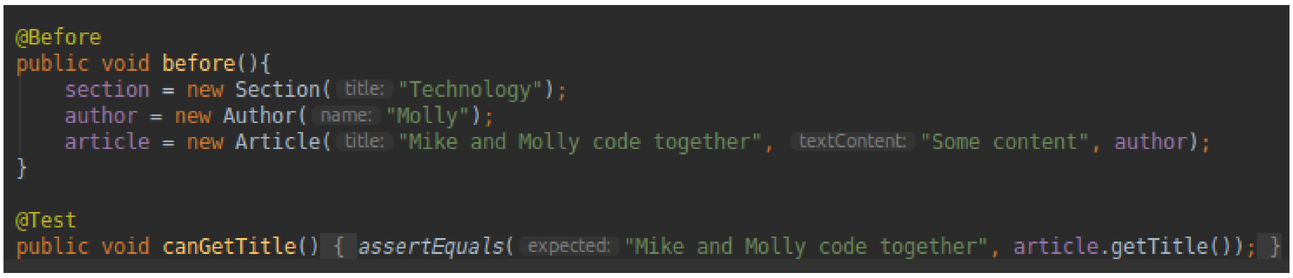
Name	Date	Result	
Mike's workout	2018-11-02	⛔	View details

Description here

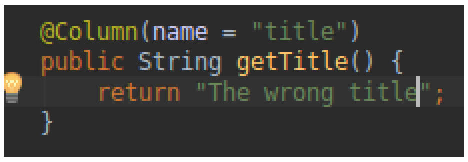
By clicking the complete workout button and confirming the completion of the workout, the workout is marked as complete and moved from the list of future sessions to the list of past sessions.

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	

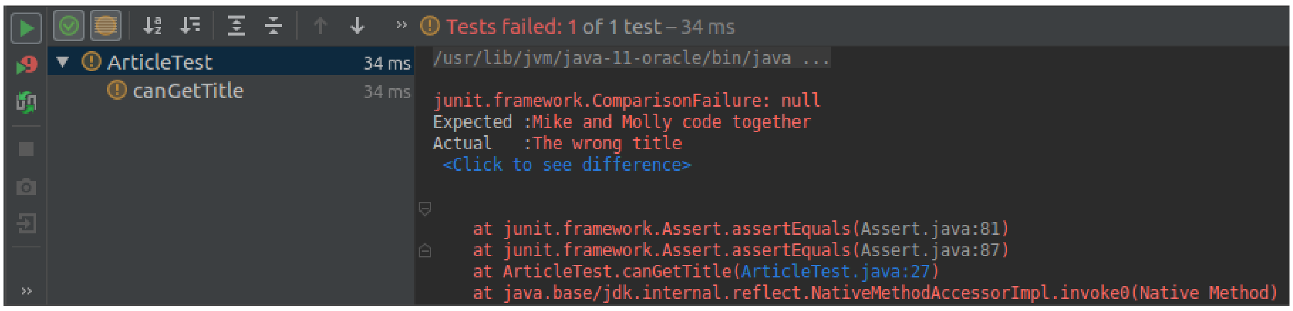
Paste Screenshot here



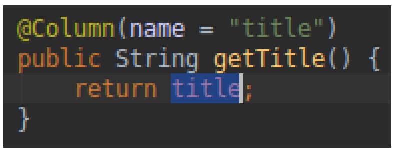
```
@Before
public void before(){
    section = new Section( title: "Technology");
    author = new Author( name: "Molly");
    article = new Article( title: "Mike and Molly code together",  textContent: "Some content", author);
}

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

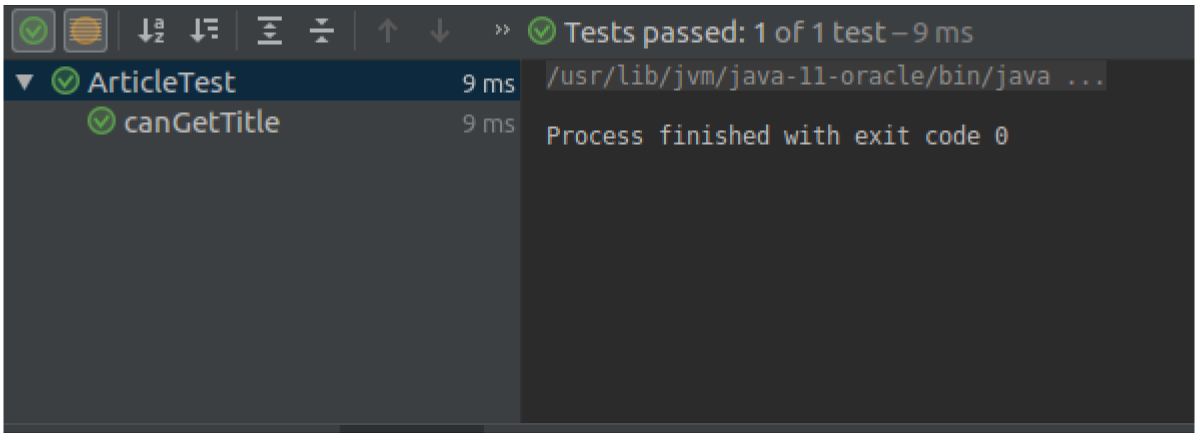


```
@Column(name = "title")
public String getTitle() {
    return "The wrong title";
}
```



```
Tests failed: 1 of 1 test – 34 ms
▼ ArticleTest      34 ms
    canGetTitle    34 ms

/usr/lib/jvm/java-11-oracle/bin/java ...

junit.framework.ComparisonFailure: null
Expected :Mike and Molly code together
Actual   :The wrong title
 <Click to see difference>

    at junit.framework.Assert.assertEquals(Assert.java:81)
    at junit.framework.Assert.assertEquals(Assert.java:87)
    at ArticleTest.canGetTitle(ArticleTest.java:27)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```


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



```
/usr/lib/jvm/java-11-oracle/bin/java ...

Process finished with exit code 0
```

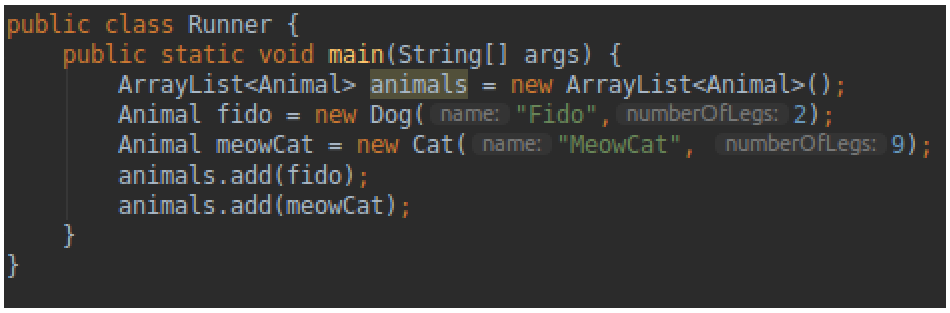
Description here

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

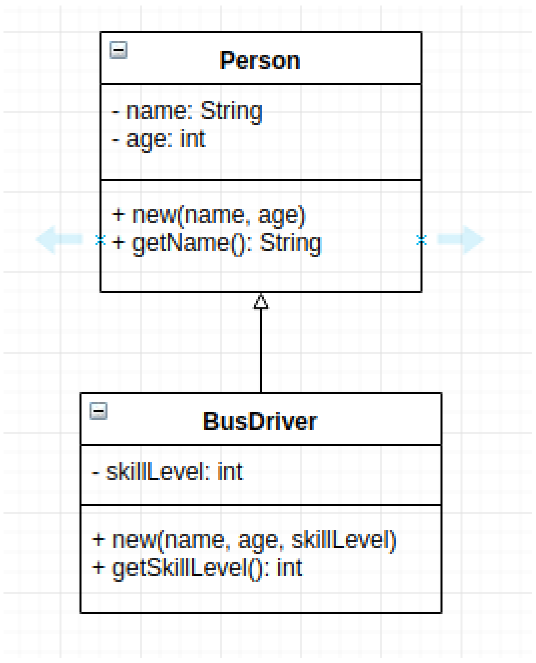

```
public class Runner {
    public static void main(String[] args) {
        ArrayList<Animal> animals = new ArrayList<Animal>();
        Animal fido = new Dog( name: "Fido", numberOfLegs: 2);
        Animal meowCat = new Cat( name: "MeowCat", numberOfLegs: 9);
        animals.add(fido);
        animals.add(meowCat);
    }
}
```

Description here

The above is an example of polymorphism as we are treating the Dog and Cat objects as if they were Animal objects as they both inherit from the Animal class.

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

Paste Screenshot here

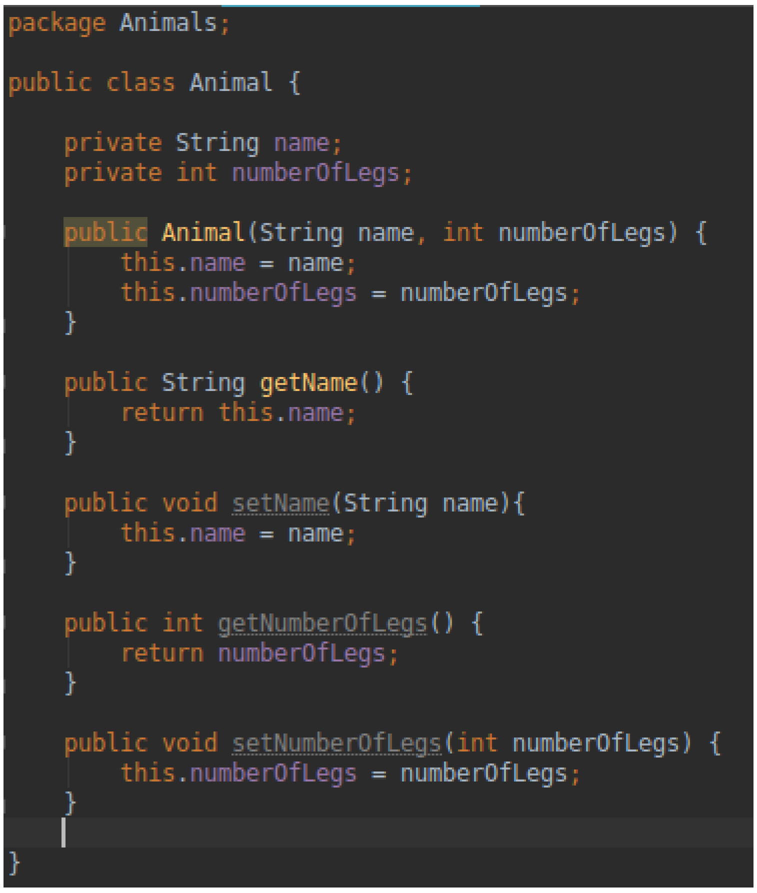


Description here

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



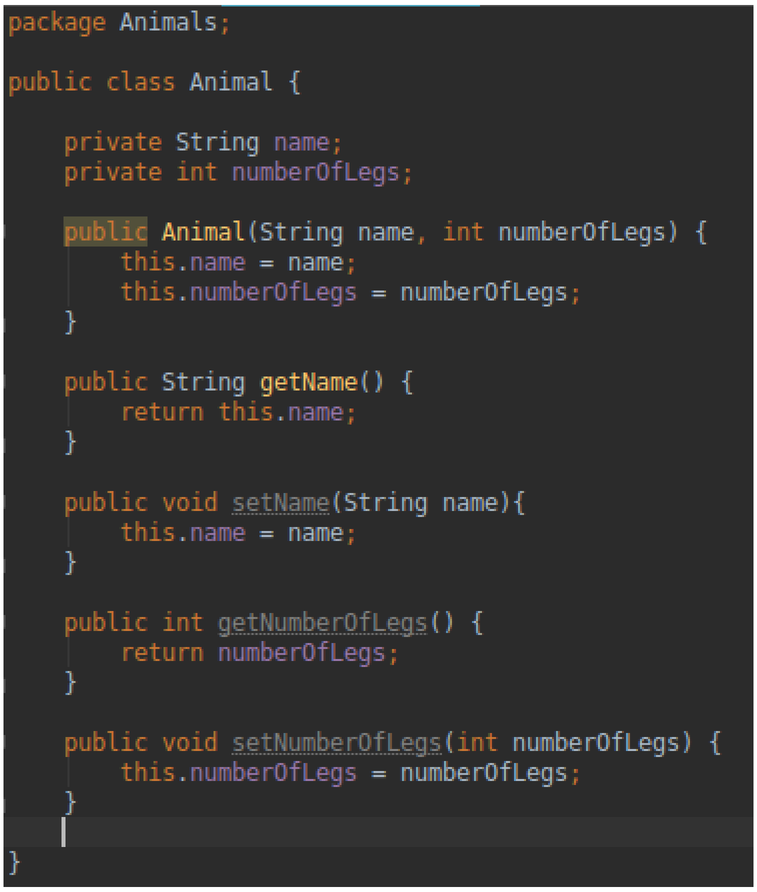
```java
package Animals;

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;
    }

}
```

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.

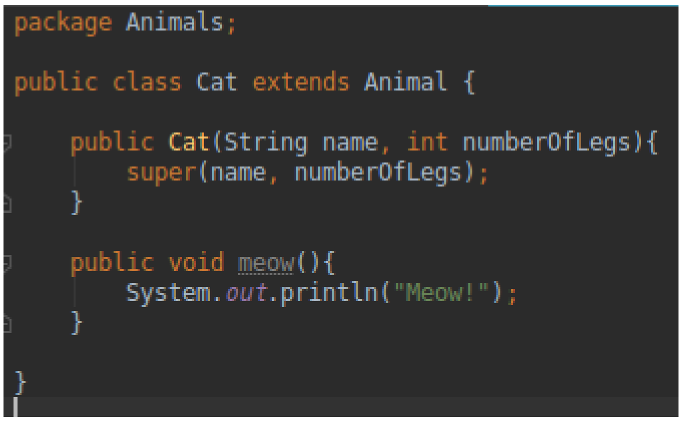
Unit	**Ref**	**Evidence**	
I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screen-shots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.	

Paste Screenshot here



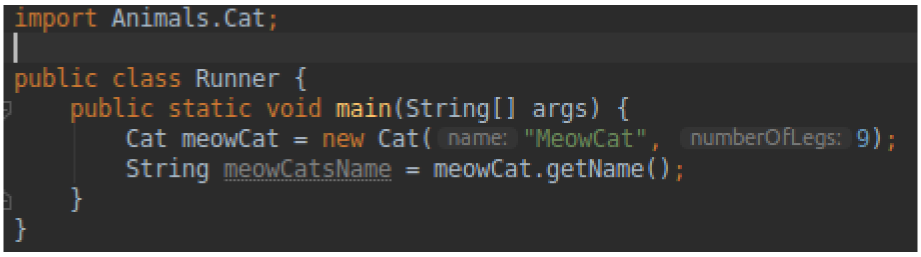
```java
package Animals;

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



```
package Animals;

public class Cat extends Animal {

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

    public void meow(){
        System.out.println("Meow!");
    }

}
```

The class that inherits from the previous class



```
import Animals.Cat;

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

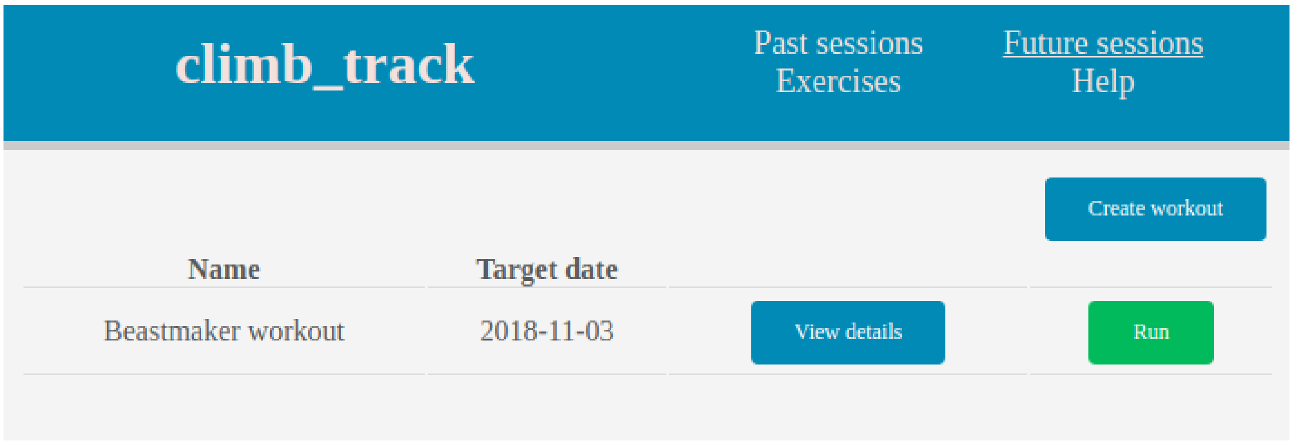
Description here

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

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



climb_track

Past sessions | Future sessions | Exercises | Help

Create workout

Name	Target date		
Beastmaker workout	2018-11-03	View details	Run

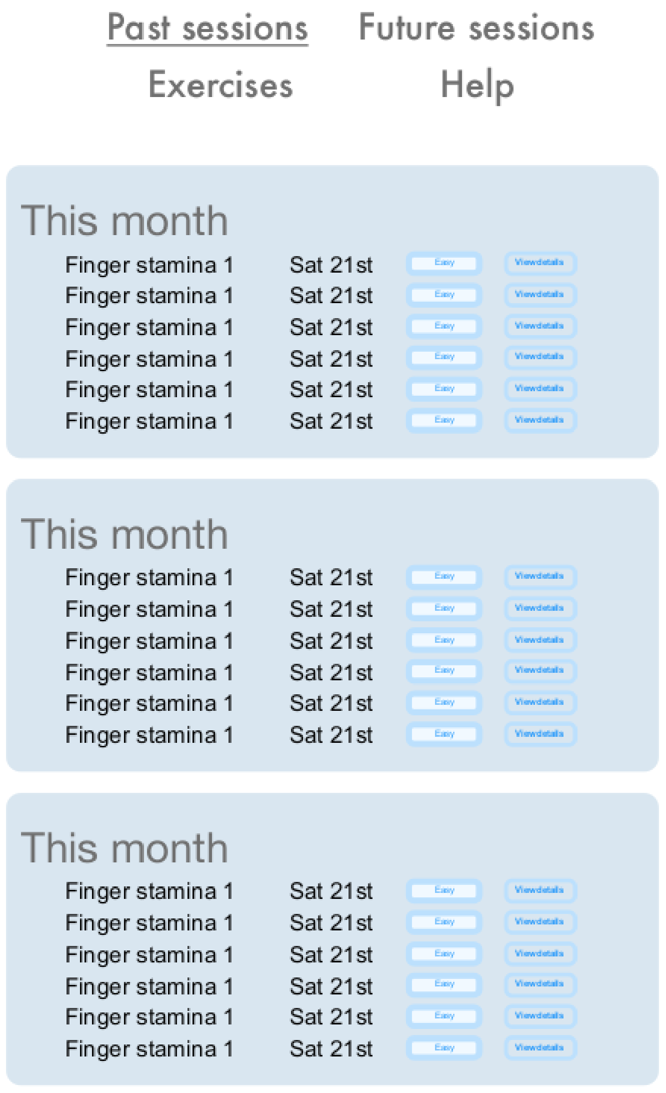
https://github.com/mikethorpe/climb_track

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



Past sessions Future sessions

Exercises Help

This month

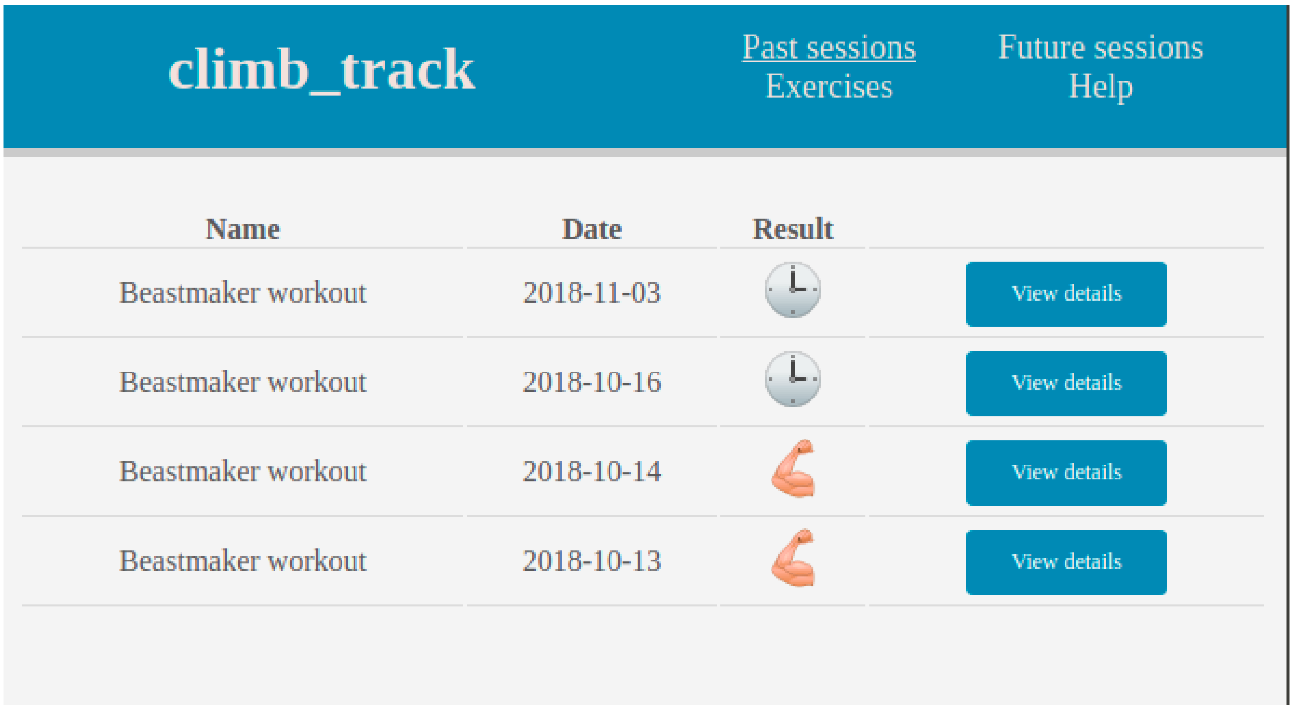
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details

This month

Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details

This month

Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details
Finger stamina 1	Sat 21st	Easy	View details



Name	Date	Result	
Beastmaker workout	2018-11-03	🕐	View details
Beastmaker workout	2018-10-16	🕐	View details
Beastmaker workout	2018-10-14	💪	View details
Beastmaker workout	2018-10-13	💪	View details

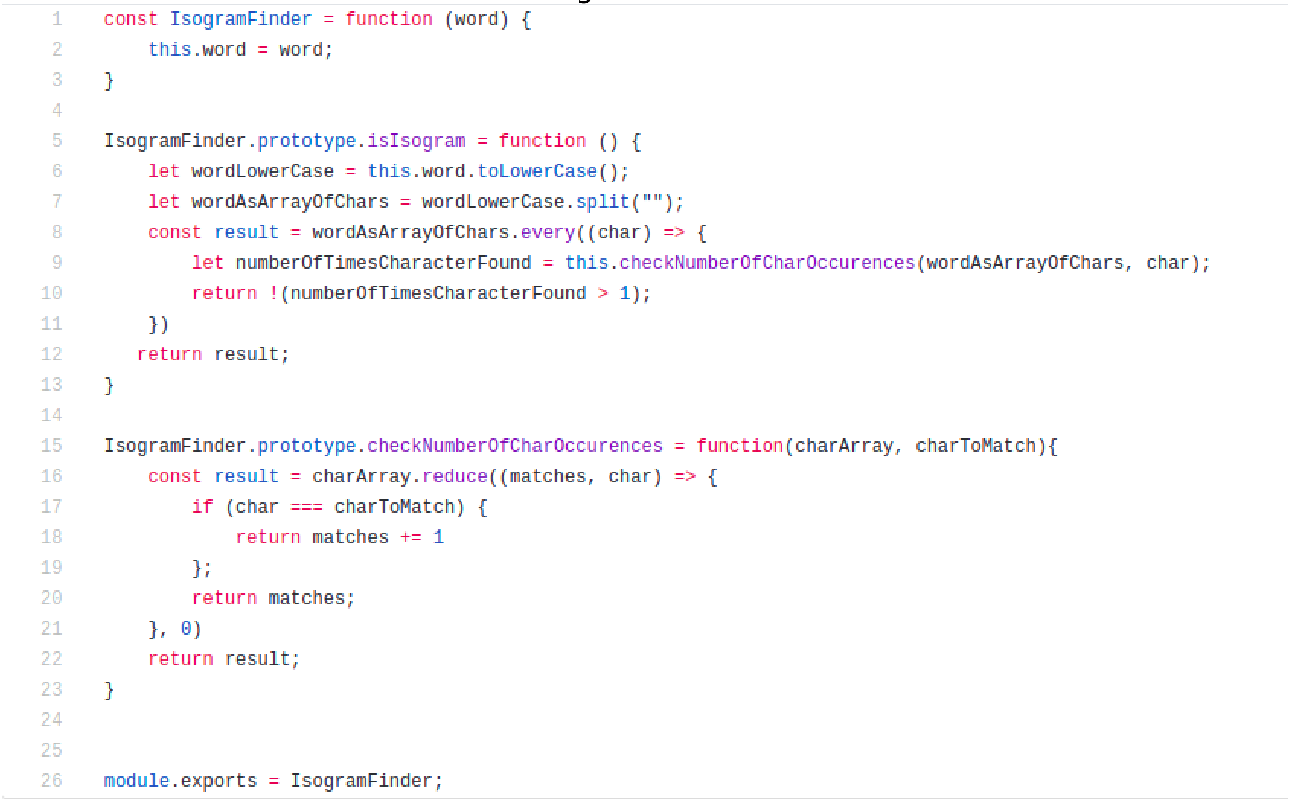
<u>Description here</u>

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.	

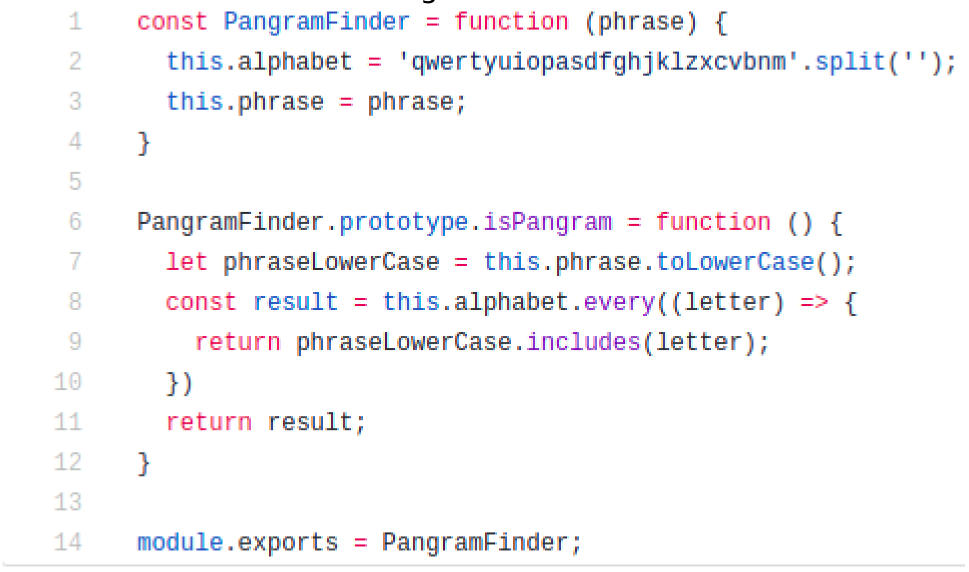
Paste Screenshot here

Isogram finder:



```javascript
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:



```javascript
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;
```

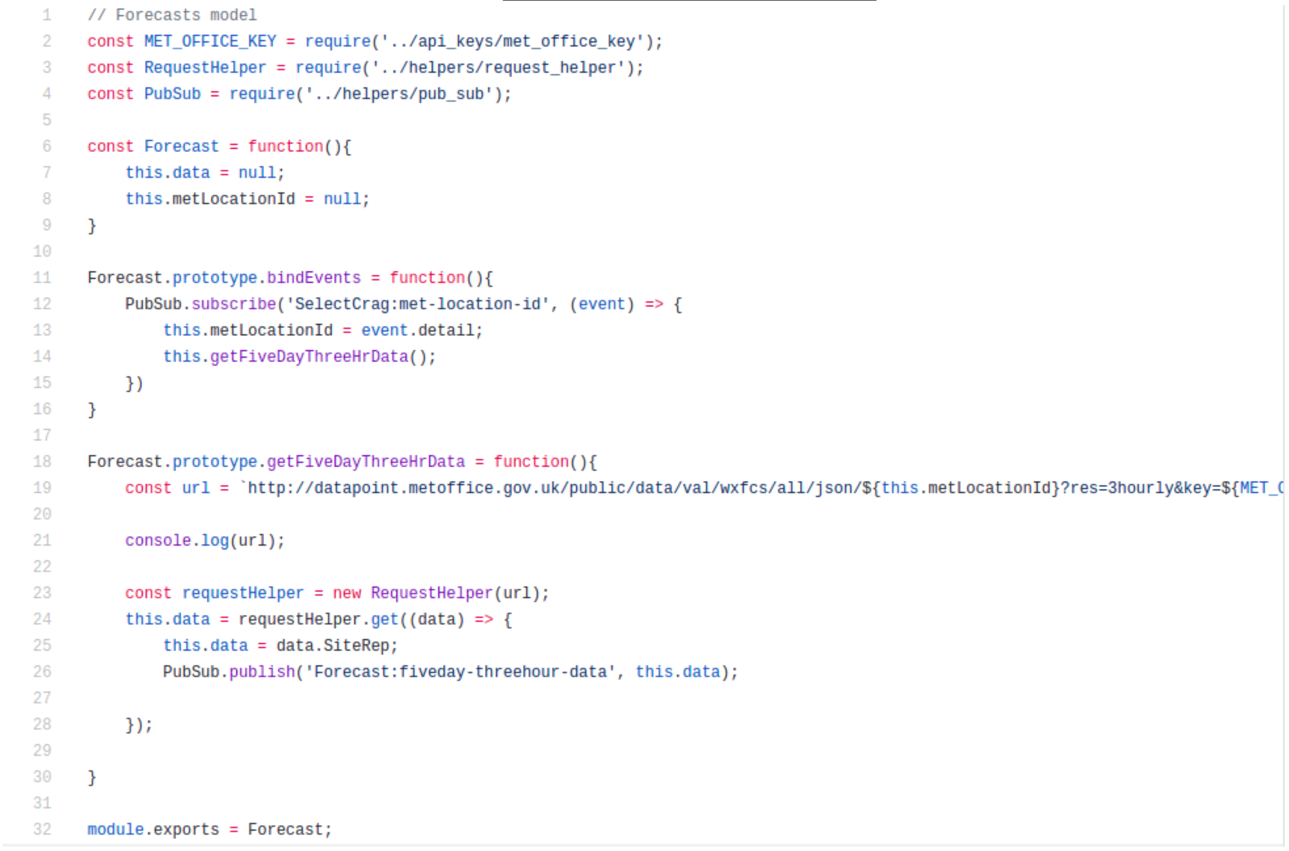
Description here

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.

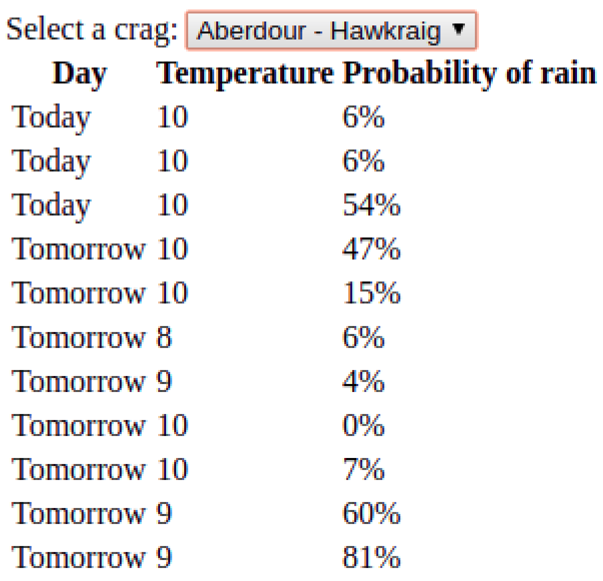
Week 12

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_

    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%

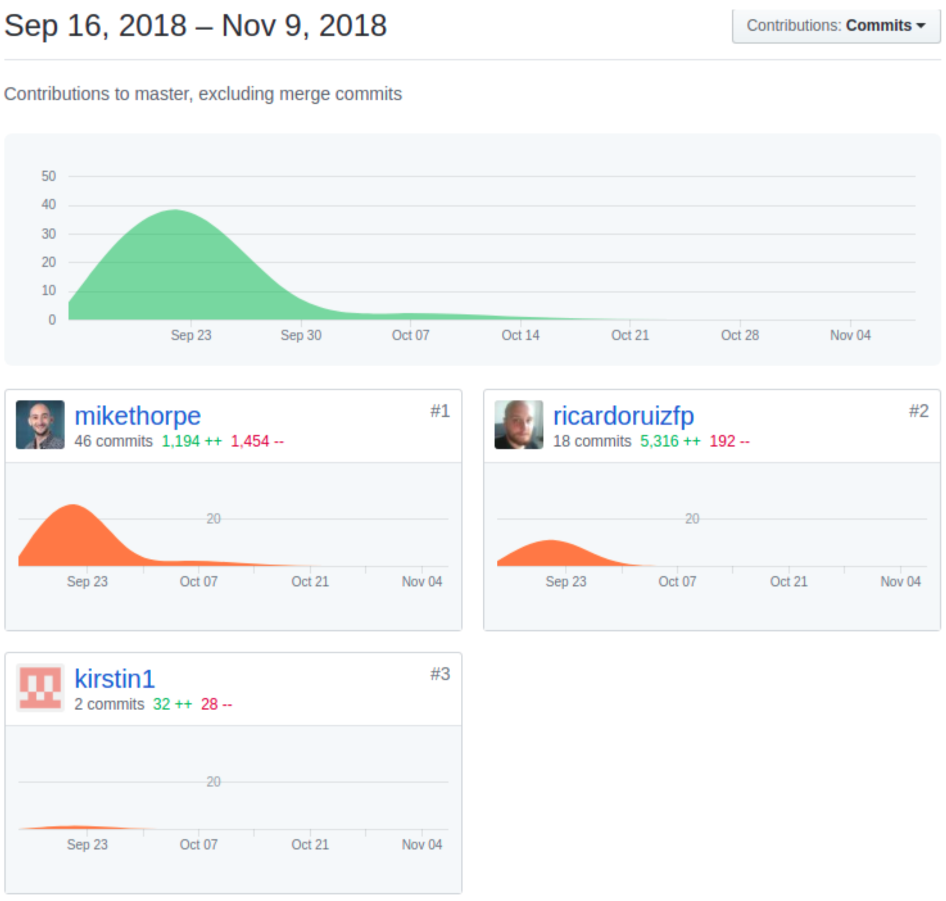
Description here

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



Sep 16, 2018 – Nov 9, 2018

Contributions: **Commits**

Contributions to master, excluding merge commits

mikethorpe #1
46 commits 1,194 ++ 1,454 --

ricardoruizfp #2
18 commits 5,316 ++ 192 --

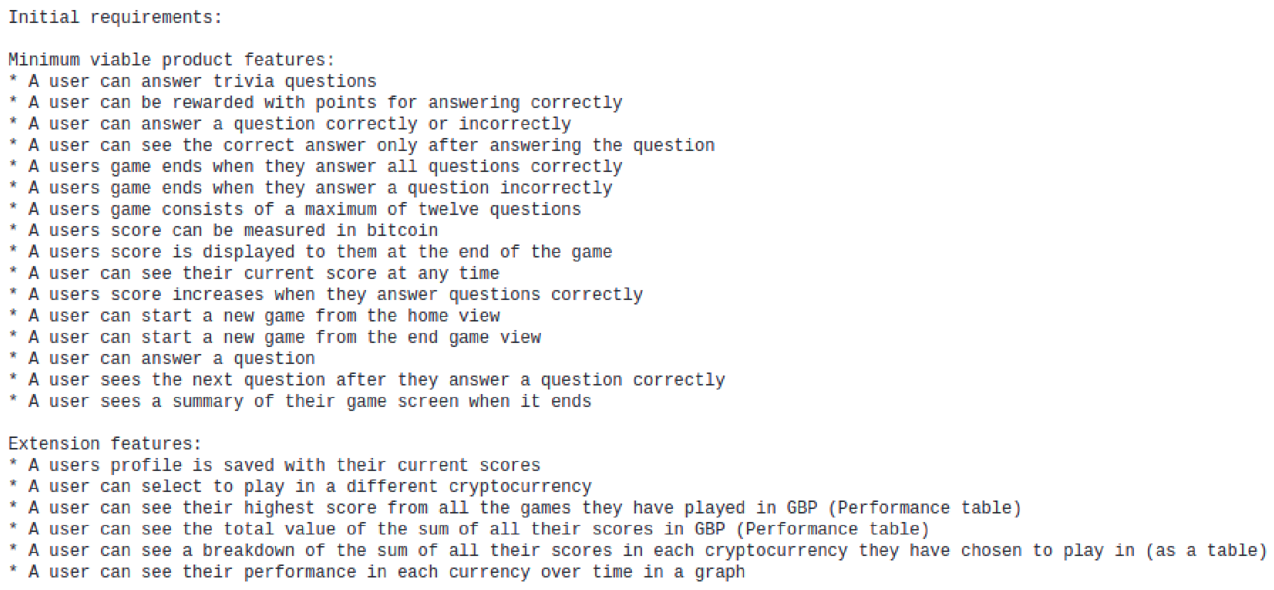
kirstin1 #3
2 commits 32 ++ 28 --

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



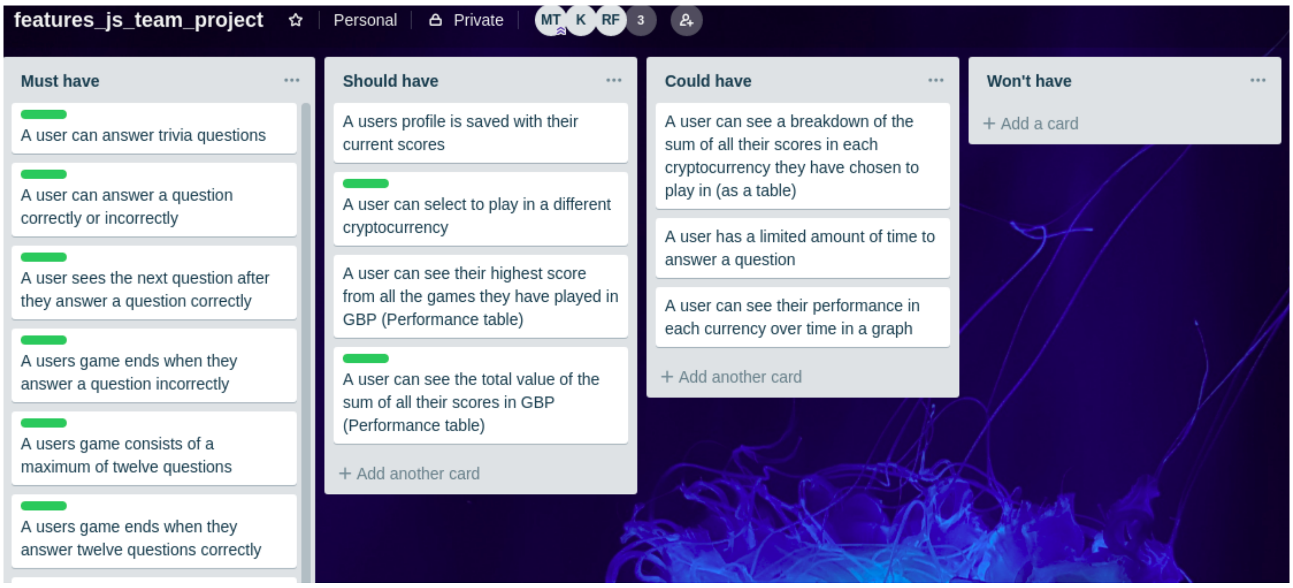
```
Initial requirements:

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



features_js_team_project | Personal | Private | MT K RF 3

Must have
- A user can answer trivia questions
- A user can answer a question correctly or incorrectly
- A user sees the next question after they answer a question correctly
- A users game ends when they answer a question incorrectly
- A users game consists of a maximum of twelve questions
- A users game ends when they answer twelve questions correctly

Should have
- 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)
- + Add another card

Could have
- 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 has a limited amount of time to answer a question
- A user can see their performance in each currency over time in a graph
- + Add another card

Won't have
- + Add a card

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

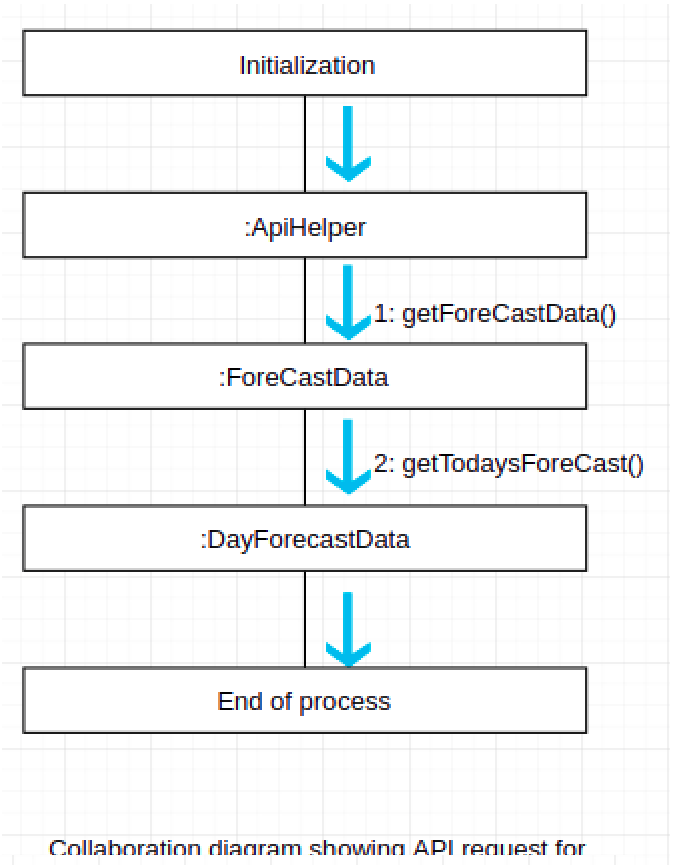
Acceptance criteria	Expected result / output	Pass / Fail
A user can create a new exer-cise	Given the create new exercise view has been loaded. And exercise details have been entered. When create exercise is clicked. Then the exercise is added to a list of all exercises	Pass
A user can change the name of an existing exercise	Given the edit page for an ex-isting 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 as-sociated 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

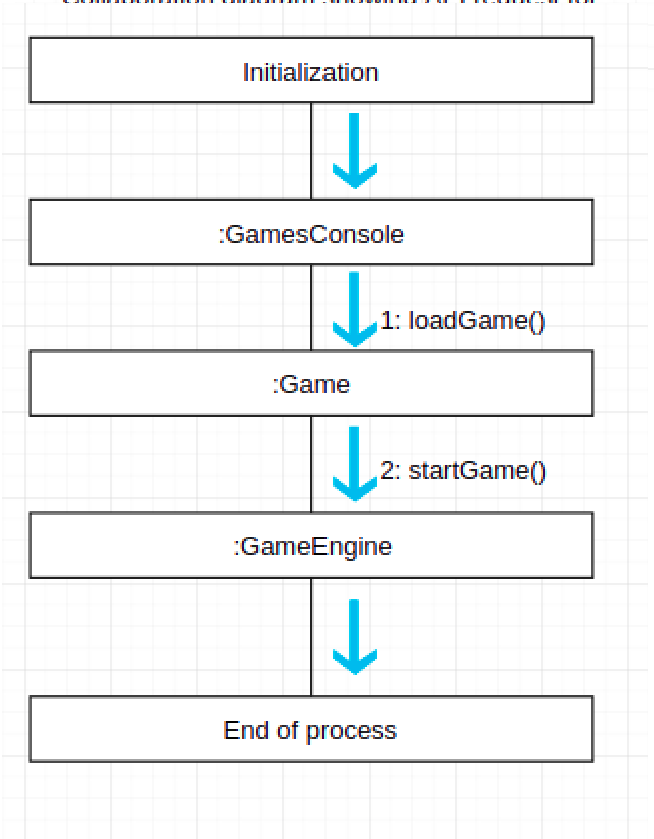
Above is an acceptance criteria and test plan. Acceptance criteria – workflows that theh application should satisfy are displayed in the left hand column. The middle column shows the test to be performed to ensure the acceptance criteria are met. The right hand column indicates whether or not the test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

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

Paste Screenshot here



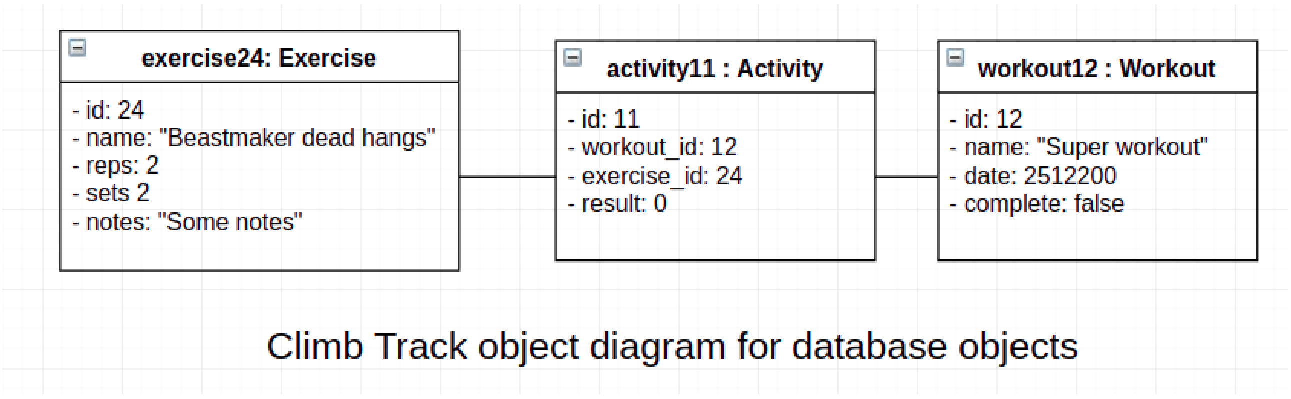
Collaboration diagram showing API request for

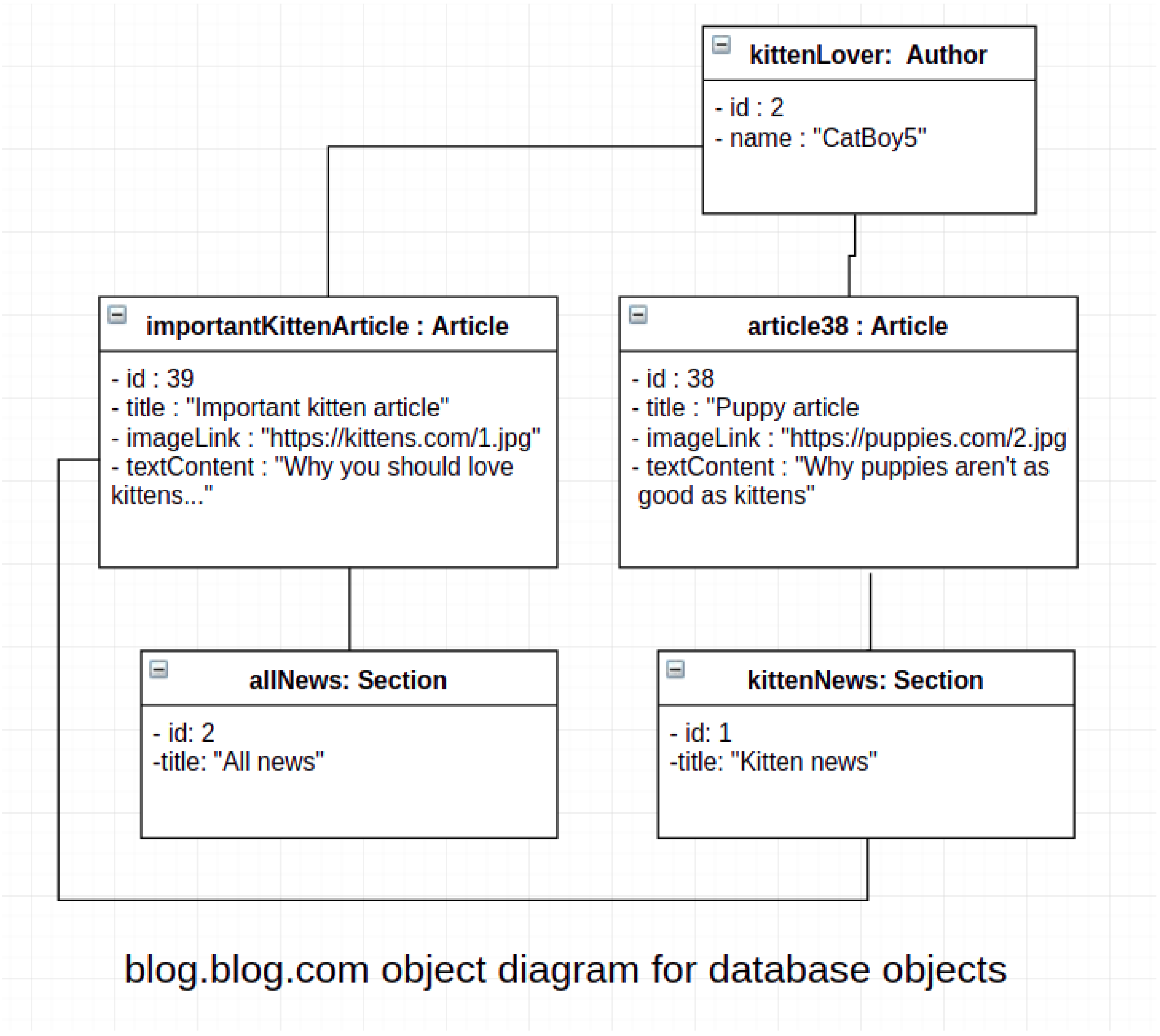


Collaboration diagram showing loading and starting game from virtual games console

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

<u>Paste Screenshot here</u>





kittenLover: Author
- id : 2
- name : "CatBoy5"
importantKittenArticle : Article
- id : 39
- title : "Important kitten article"
- imageLink : "https://kittens.com/1.jpg"
- textContent : "Why you should love kittens..."
article38 : Article
- id : 38
- title : "Puppy article
- imageLink : "https://puppies.com/2.jpg
- textContent : "Why puppies aren't as good as kittens"
allNews: Section
- id: 2
-title: "All news"
kittenNews: Section
- id: 1
-title: "Kitten news"

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.