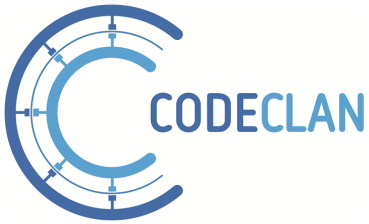


Evidence Gathering Document for SQA Level 8 Professional Developer Award.

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

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

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

Week 2

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

```ruby
class Rooms

  attr_reader :roomNo, :playlist, :guests, :waitingList, :rejected , :fee

  def initialize(roomNo, fee, max)
    @roomNo = roomNo # Set room number
    @fee = fee # Entry fee set during room instatiation
    @guests = [] # Guests checked-in to this
    @playlist = [] # Songs added to this
    @waitingList = [] # Guests waiting to get in a room
    @rejected = 0 # Guests rejected due to insufficient funds
    @max = max # Maximum limit of guests in a room
    # @till = 0
  end

  def addSong(song)
    @playlist.push(song.songName)
  end
```

Figure 1: 'Rooms' Class with an empty @playlist array within its constructor

```ruby
def test_songInclude # Test if song is included on playlist
  @room1.addSong(@song1)
  # binding.pry
  assert_equal(true, @room1.playlistIncludes("Perfect"))
  p "This test shows the @playlist array includes the song Perfect"
end
```

Figure 2: Test function verifies the 'addSong( )' function to @playlist array is working.

```
➜ weekend_homework git:(master) × ruby spec/rooms_spec.rb
Run options: --seed 41396

# Running:

"This test shows the @playlist includes the song Perfect"
.........

Finished in 0.001541s, 5840.3631 runs/s, 6489.2923 assertions/s.

9 runs, 10 assertions, 0 failures, 0 errors, 0 skips
```

Figure 3: Shows successful execution of the test function 'test_songInclude'.

Description

Figure 1 shows the class **'Rooms'** with the constructor containing a series of empty arrays, namely the **@playlist** array. Songs are added to this array by calling the **'addSong( )'** function as shown in the same figure above. It takes in a song object as an argument. A test function was written to verify that a specific song has been successfully added using the **'playlistIncludes( )'** function which takes in a song name as a string argument.
The result of the successful execution of the test function is displayed in figure 3.

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

```ruby
@pet_shop = {
  pets: [
    {
      name: "Sir Percy",
      pet_type: :cat,
      breed: "British Shorthair",
      price: 500
    },
    {
      name: "King Bagdemagus",
      pet_type: :cat,
      breed: "British Shorthair",
      price: 500
    },
    {
      name: "Sir Lancelot",
      pet_type: :dog,
      breed: "Pomsky",
      price: 1000,
    },
    {
      name: "Arthur",
      pet_type: :dog,
      breed: "Husky",
      price: 900,
    },
```

Figure 4: Hash named @pet_shop containing pets data set such as names

```ruby
def test_find_pet_by_name__returns_pet
  pet = find_pet_by_name(@pet_shop, "Arthur")
  assert_equal("Arthur", pet[:name])
end
```

Figure 5: Test function to verify 'find_pet_by_name( )' function is working

```ruby
def find_pet_by_name(pet_shop, petName)
  for each in pet_shop[:pets]
    if (each[:name] == petName)
      return each
    end
  end
  return nil
end
```

Figure 6: Accessing the hash in the program

```
➜  homework_start_point git:(master) ✗ ruby specs/pet_shop_spec.rb
Run options: --seed 48603

# Running:

.."This test shows the pet name Arthur exist in the hash"
.................

Finished in 0.002209s, 8601.1768 runs/s, 8601.1768 assertions/s.

19 runs, 19 assertions, 0 failures, 0 errors, 0 skips
➜  homework_start_point git:(master) ✗
```

Figure 7: Shows successful execution of the test function 'test_find_pet_by name__returns_pet'.

Description here

Figure 4 shows a hash named **'@pet_shop'** containing a series of sub-hashes within an array named **pets.** Pet hashes are accessed using the **'find_pet_by_name( )'** function as shown in Figure 5. It takes in a hash object and a string name. A test function was written to verify that their function works as intended. An existing pet name "Arthur" was used to verify that the function is able to retrieve the specific object with the matching name. The result of the successful execution of the test function is displayed in Figure 6.

Week 3

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

```ruby
def self.find(id)
  sql = "SELECT * FROM members WHERE id = $1"

  values = [id]
  member_hash = SqlRunner.run(sql, values).first
  return Member.new(member_hash)
end
```

Figure 7: Function within the program containing a search SQL command.

```
GymApp=# select * from members where id = 122;
 id  | first_name | last_name | member_type
-----+------------+-----------+-------------
 122 | Stephen    | Hart      | premium
(1 row)
```

Figure 8: Result of the search by id (122) SQL command.

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

```
GymApp=# SELECT * FROM members;
 id  | first_name | last_name | member_type
-----+------------+-----------+-------------
 127 | Raymond    | Yau       | premium
 119 | Rique      | Batista   | standard
 120 | Colin      | Bell      | standard
 121 | Leah       | Meromy    | premium
 122 | Stephen    | Hart      | premium
 123 | Anna       | Henderson | premium
 124 | Louise     | Reid      | premium
 125 | David      | Bell      | premium
 126 | Neil       | Davidson  | premium
 118 | Ray        | Yau       | premium
(10 rows)
```

Figure 9: All selected data from members table

```
GymApp=# SELECT * FROM members ORDER BY first_name;
 id  | first_name | last_name | member_type
-----+------------+-----------+-------------
 123 | Anna       | Henderson | premium
 120 | Colin      | Bell      | standard
 125 | David      | Bell      | premium
 121 | Leah       | Meromy    | premium
 124 | Louise     | Reid      | premium
 126 | Neil       | Davidson  | premium
 118 | Ray        | Yau       | premium
 127 | Raymond    | Yau       | premium
 119 | Rique      | Batista   | standard
 122 | Stephen    | Hart      | premium
(10 rows)
```

Figure 10: All selected data in members table ORDER BY 'first_name' column.

Figure 12: Displayed result of the sorting function on the web page.

```ruby
def self.find_all()
  sql = "SELECT * FROM members ORDER BY first_name"
  members_hash = SqlRunner.run(sql)
  return members_hash.map {|member| Member.new(member)}
end
```

Figure 11: Function within the program containing sorting SQL command.

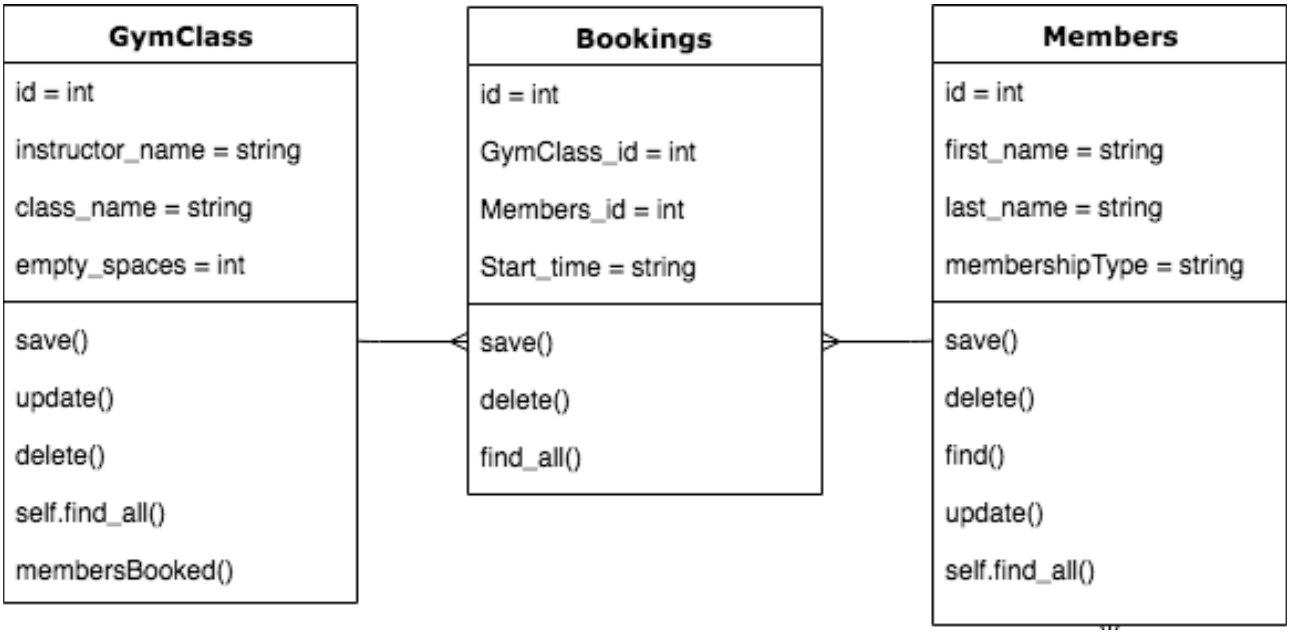
Description here

SQL command 'SELECT * FROM members ORDER BY first_name' was used to select all entries within the members table and sorted alphabetically by the first_name column. Results are tested within the Unix terminal window. Figure 11 & 12 also shows the results of the output on the website and the function containing the SQL sorting command.

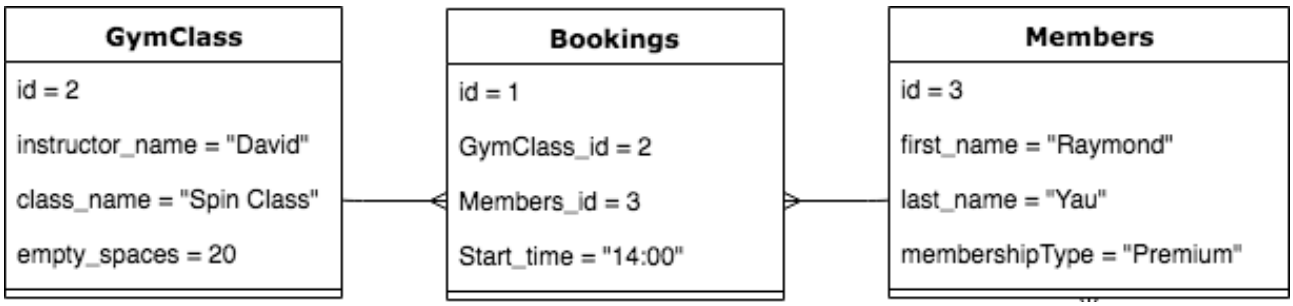
Week 5 and 6

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

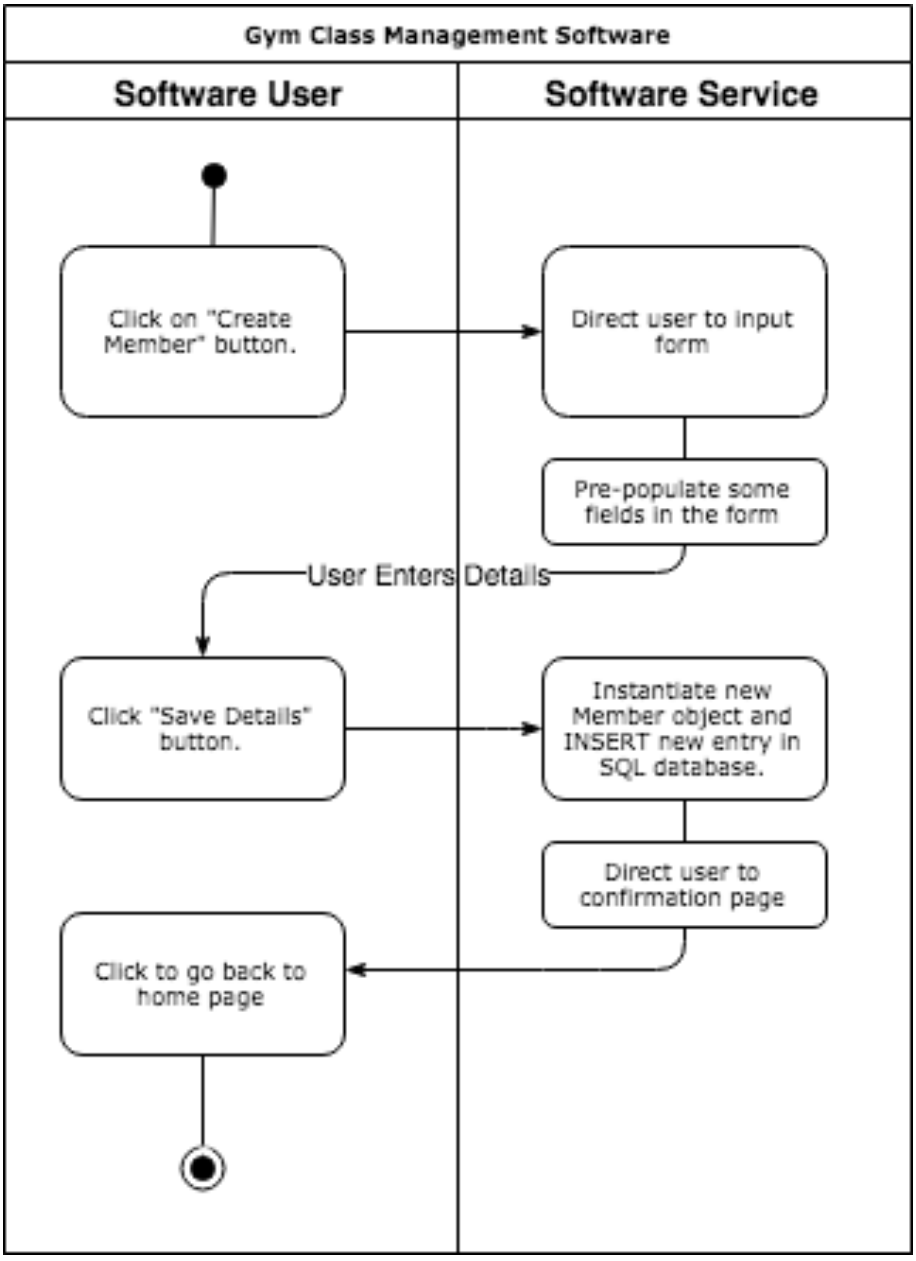
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		Description:



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



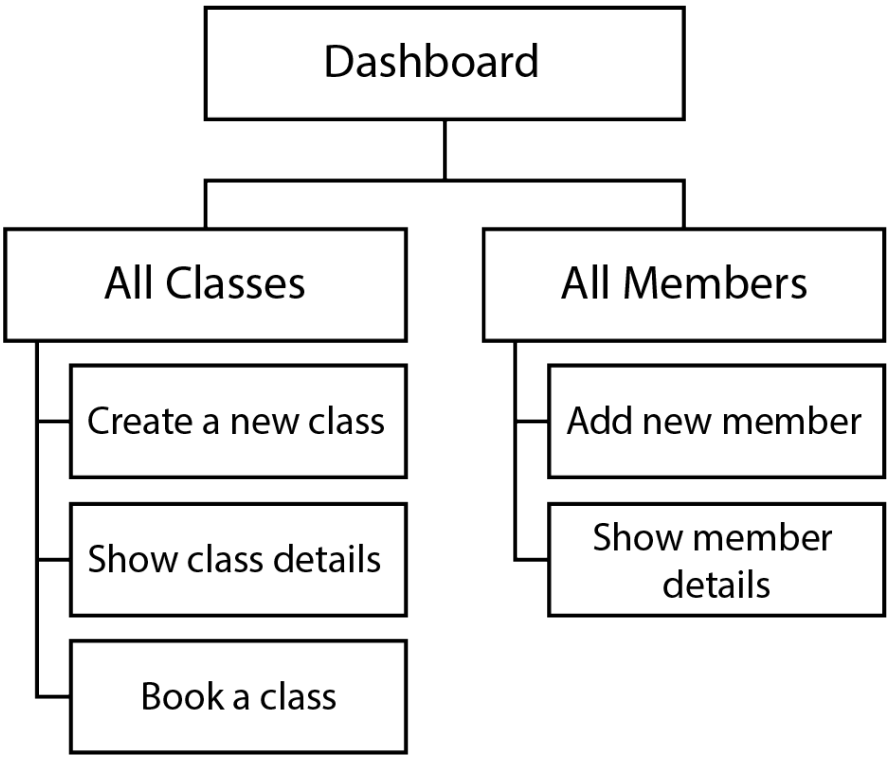
Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram
		Description:



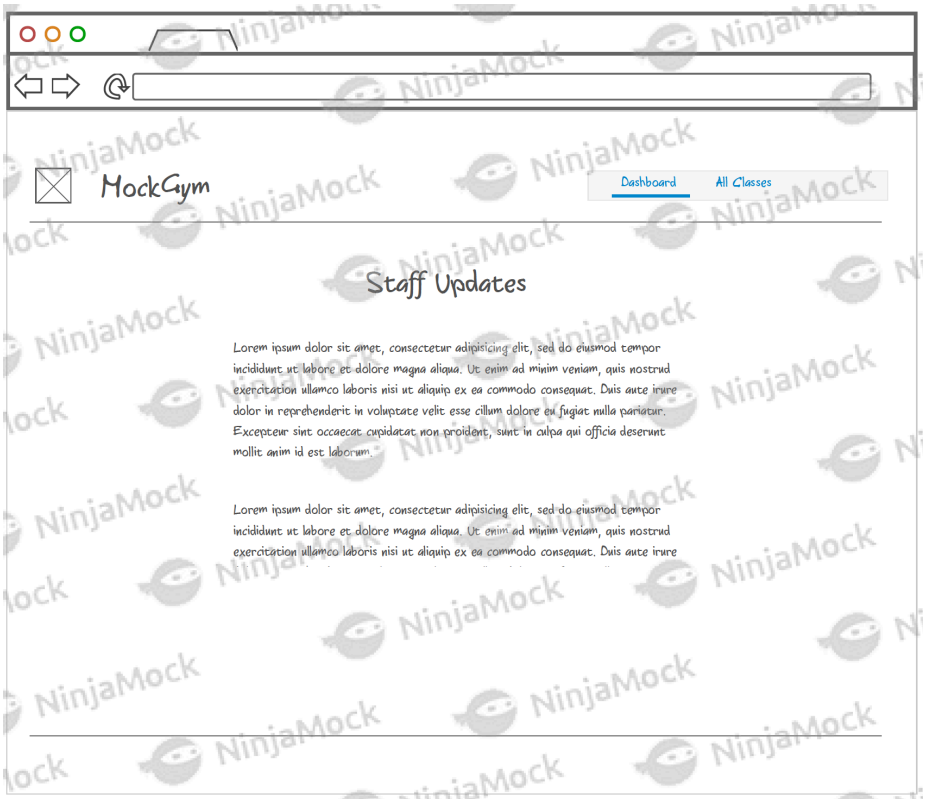
Unit	Ref	Evidence
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time
		Description:

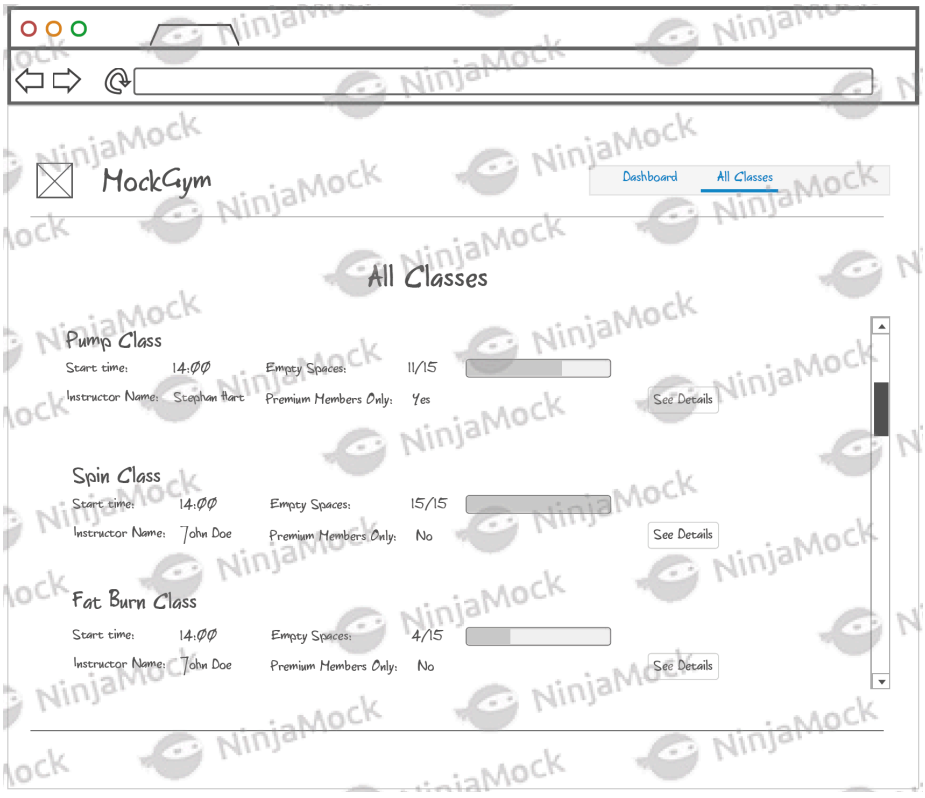
Topic	Possible Effect of Constraint on Product	Solution
Hardware and software platforms	Different devices used may lead to content formatting issues.	Use CSS media query to specify requirements for different hardware platforms.
Performance requirements	Use of any images may lead to slow loading of website application content.	Avoid the use of images on website if not necessary.
Persistant storage and transactions	Use of PostgreSQL to store image data can occupy too much storage space and affect read performance, making it costly	Store image location references for each image in the database instead of the actual image data.
Usabilitv	Website is predominantly text heavy and may be hard to read.	Use minimum 12pt font size. Use tables to format information.
Budgets	Not applicable	Not applicable
Time limitations	1 week project	Prioritise MVP and reuse boilerplate code where possible.

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



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

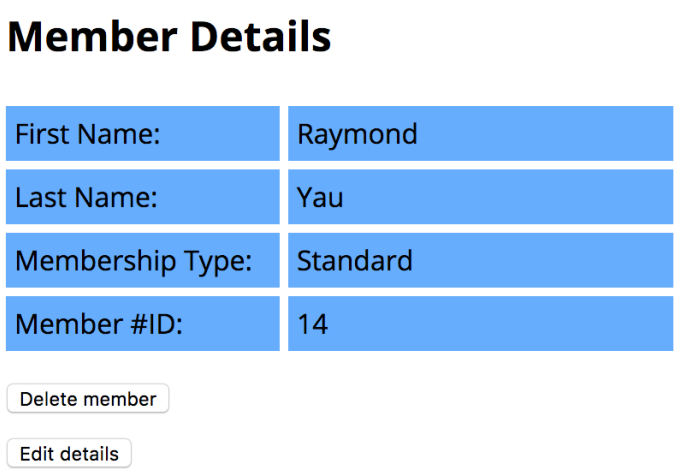




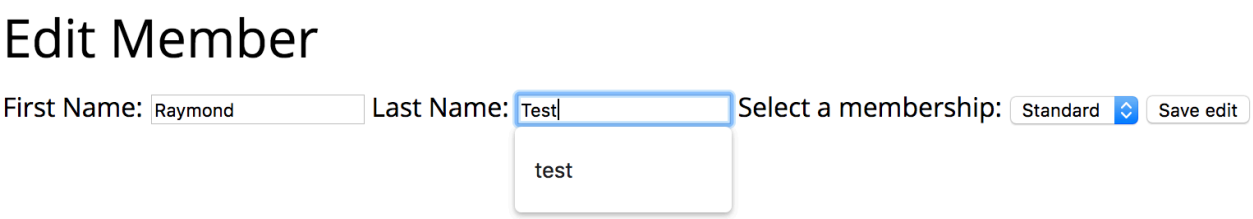
Unit	Ref	Evidence
P	P.10	Example of Pseudocode used for a method
		Description:

```
def peakTime()
  # peaktime variable set to false by default
  peaktime = false
  # Hard coded peak times and string format
  # Morning peak times
  t1 = Time.new(2018, 1, 1, 7, 0, 0, 0).strftime("%H%M%S")
  t2 = Time.new(2018, 1, 1, 9, 0, 0, 0).strftime("%H%M%S")
  # Evening peak times
  t3 = Time.new(2018, 1, 1, 17, 0, 0, 0).strftime("%H%M%S")
  t4 = Time.new(2018, 1, 1, 20, 0, 0, 0).strftime("%H%M%S")

  # Set gymclass start time in string format
  current_time = self.start_time.strftime("%H%M%S")

  # Check if current selected time is between the peak times t1-t2 or t3-t4
  if current_time.between?(t1,t2) || current_time.between?(t3,t4)
    peaktime = true
  end
  return peaktime
end
```

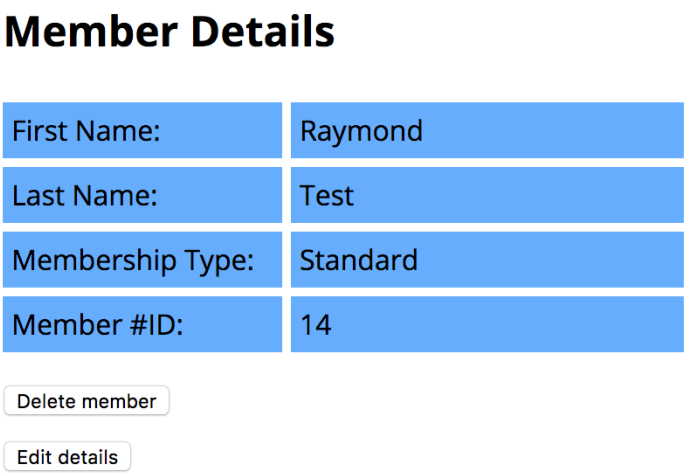
Unit	Ref	Evidence
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way
		Description:



Member Details Screen: Before data field ‘Last Name’ is edited

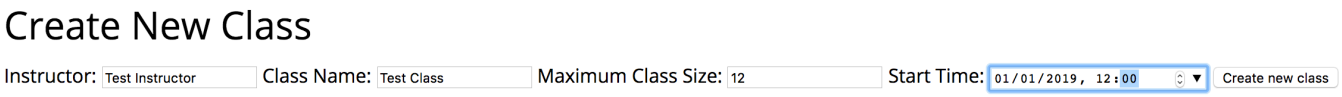


Edit Detail Screen: New ‘Last Name’ value is entered

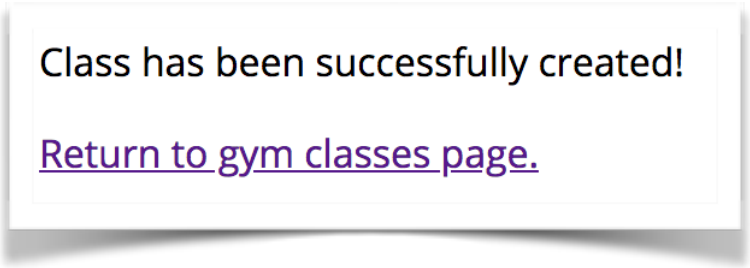


Resulting Member Details: ‘Last Name’ value is changed to ‘Test’

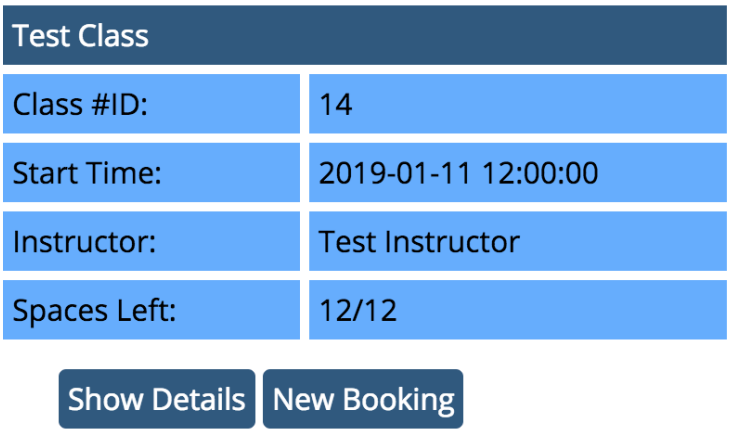
Unit	Ref	Evidence
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved
		Description:



Create New Class Screen: Create new class entry in database by inputting data into a form

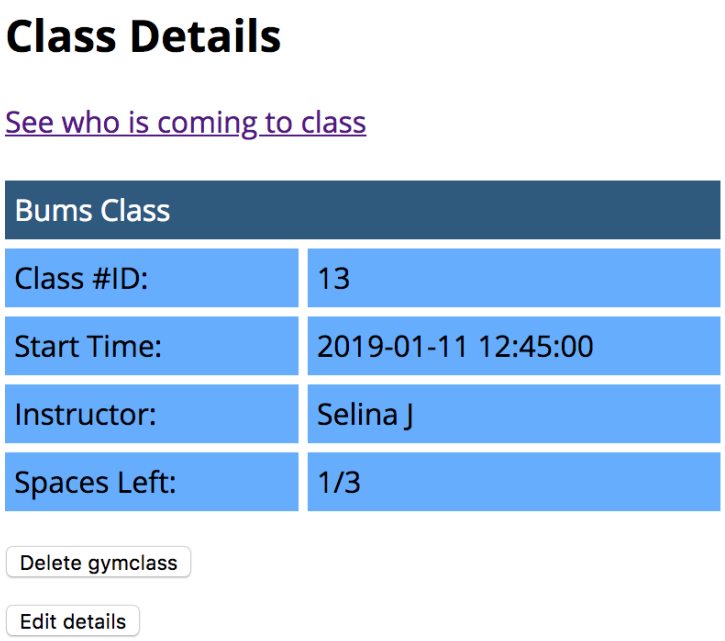


Caption: Confirmation page confirming the successful creation of a class.

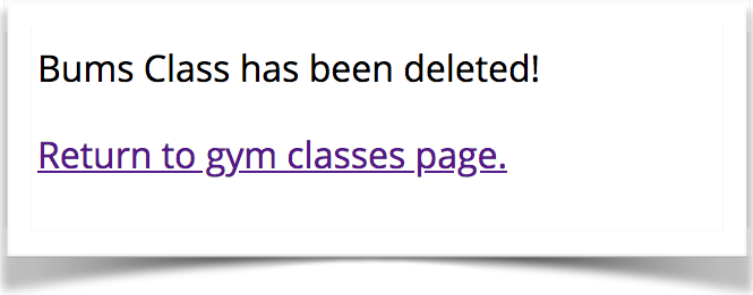


Caption: Index page displaying the new entry of the class created.

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



Class Details Screen: Details screen shows the class with a delete button



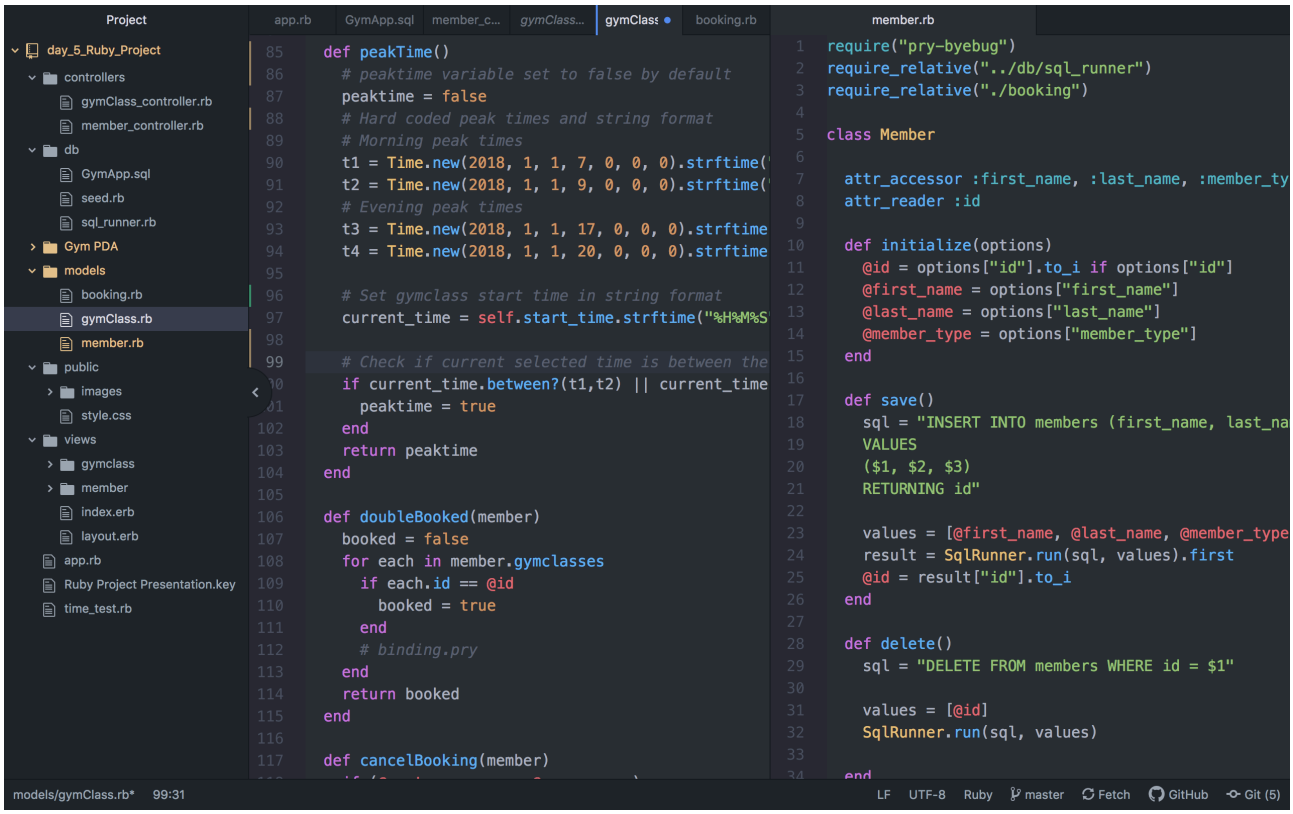
Delete Action Confirmation: Class deleted when Delete button is clicked.

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

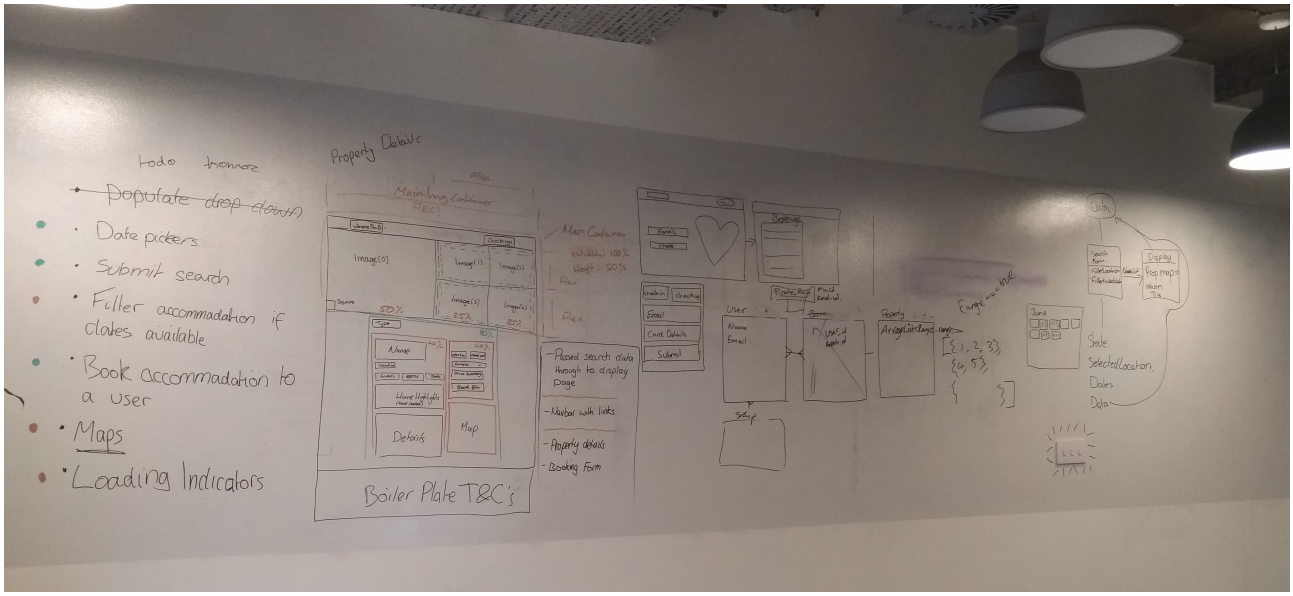
Link to Github individual project

https://github.com/xeb10154/GymManager-Ruby-Sinatra-Project

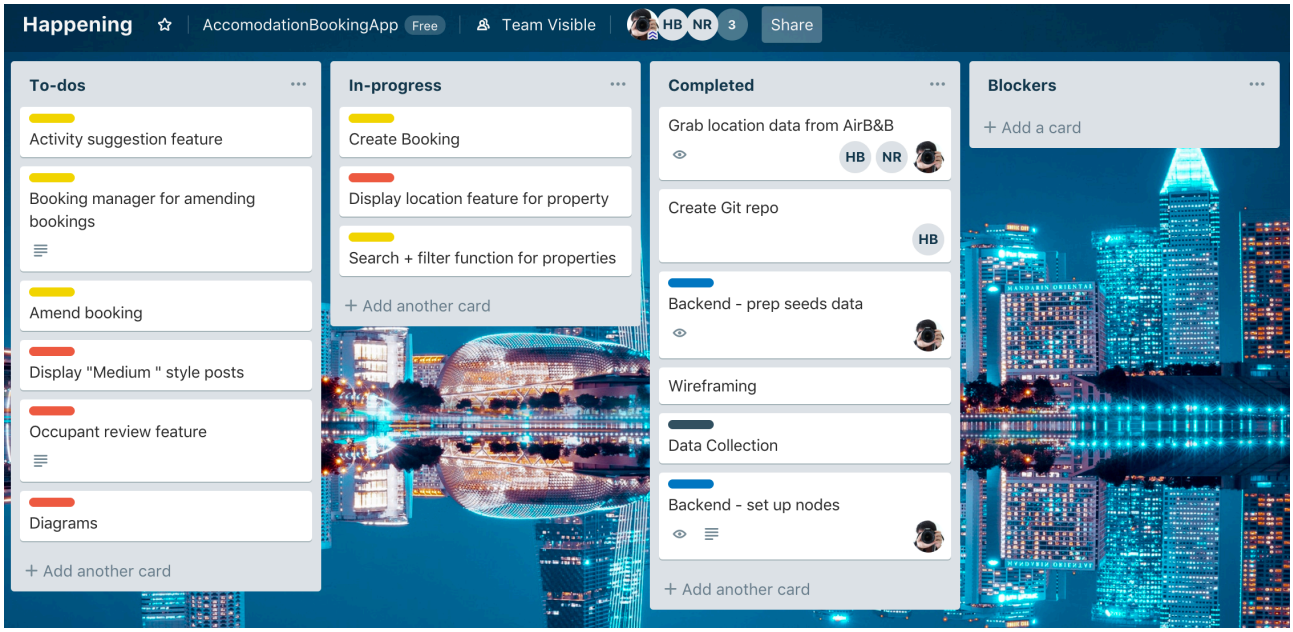
Screenshot of the individual project



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



Left: initial goals. Middle: View layout planning & development. Right: Flow diagrams planning.



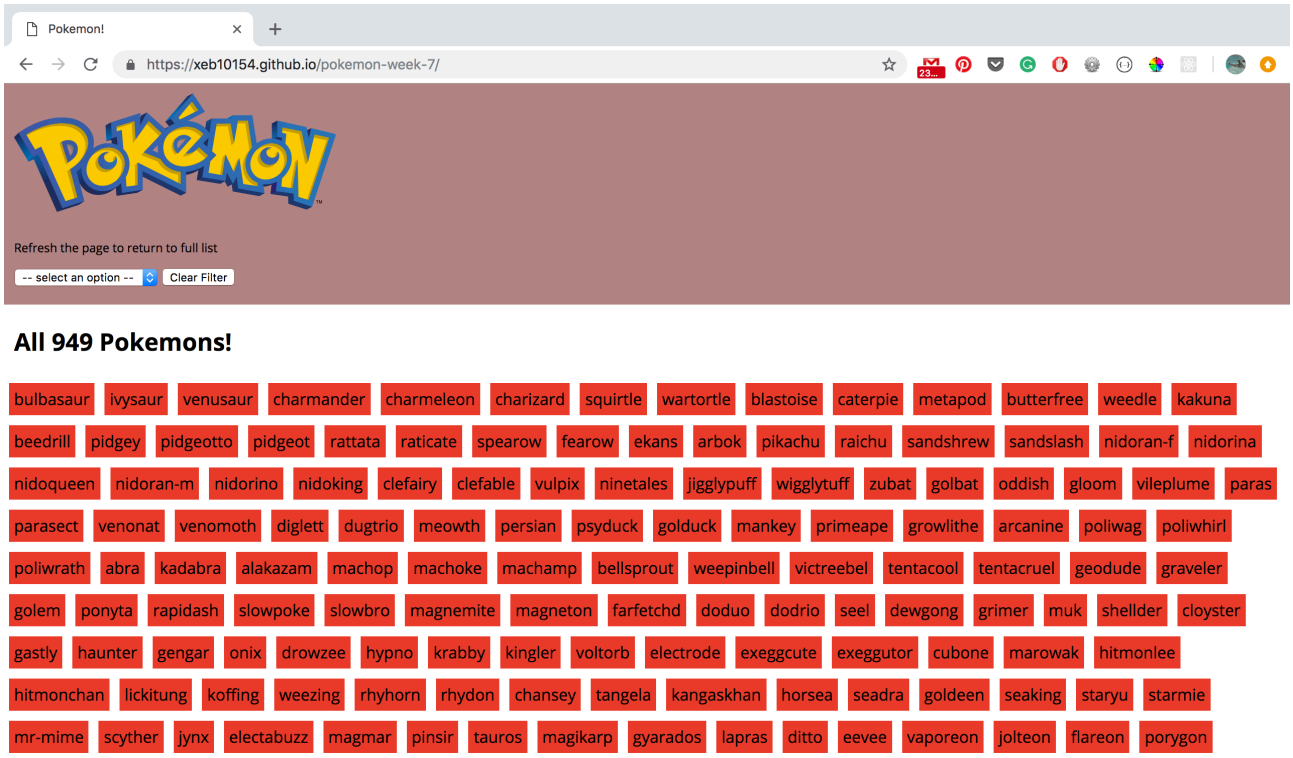
Trello board: Agreed goals and features documented and updated throughout the project.

Week 7

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

```
1  const PubSub = require('../helpers/pub_sub')
2  const Request = require('../helpers/request')
3
4  const Pokemon = function(data){
5    this.data = []; // Stores full list of pokemons and urls
6  }
7
8  Pokemon.prototype.bindEvents = function(){
9    this.getData();
10   this.getTypes();
11   PubSub.subscribe('PokemonListView:PokeIndex', (event) => {
12     const pokemonIndex = event.detail;
13     this.findPokemonInfo(pokemonIndex)
14   });
15   PubSub.subscribe('FilterView:TypeIndex', (event) => {
16     const typeIndex = event.detail;
17     this.findSelectedTypes(typeIndex)
18   });
19 }
20
21 Pokemon.prototype.getData = function(){
22   url = `https://pokeapi.co/api/v2/pokemon/`;
23   const request = new Request(url);
24   request.get().then((data) => {
25     this.data = data.results
26     PubSub.publish('Pokemon:All-Data-Ready', this.data)
27   });
28 }
```

Pokemon API Code: Last method in the screenshot (lines 21-28) shows the url route for accessing the API data. Retrieved data is displayed on the website main page.



Pokemon Website: Displayed all API data retrieved from the API url route defined in code

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

Example of test code:

```ruby
require_relative('card.rb')
class CardGame


  def checkforAce(card)
    if card.value = 1 # assignment operator used instead of a comparator
      return true
    else
      return false
    end
  end

  dif highest_card(card1 card2) # typo error, def instead of dif. Comma
  is also needed to separate args.
  if card1.value > card2.value
    return card.name # should be card1.name
  else # elsif statement to take into account card of equal value
    card2 # should be card2.name
  end
end
end # extra dangling end keyword

def self.cards_total(cards)
  total # No value or datatype set for variable total
  for card in cards
    total += card.value
    return "You have a total of" + total # Should be interpolated with a
    space. Return should be outside for loop
  end
end
```

The test code failing to pass:

```
[➜  Static_and_Dynamic_Task_A git:(master) ✗ ruby spec/testing_task_2.rb          ]
spec/testing_task_2.rb:3:in `require_relative': /Users/raymond/Documents/codecla
n_work/PDA/Static_and_Dynamic_Task_A/testing_task_2.rb:25: syntax error, unexpec
ted keyword_end, expecting end-of-input (SyntaxError)
end # extra dangling end keyword
   ^
        from spec/testing_task_2.rb:3:in `<main>'
```

Example of test code passing after errors have been corrected

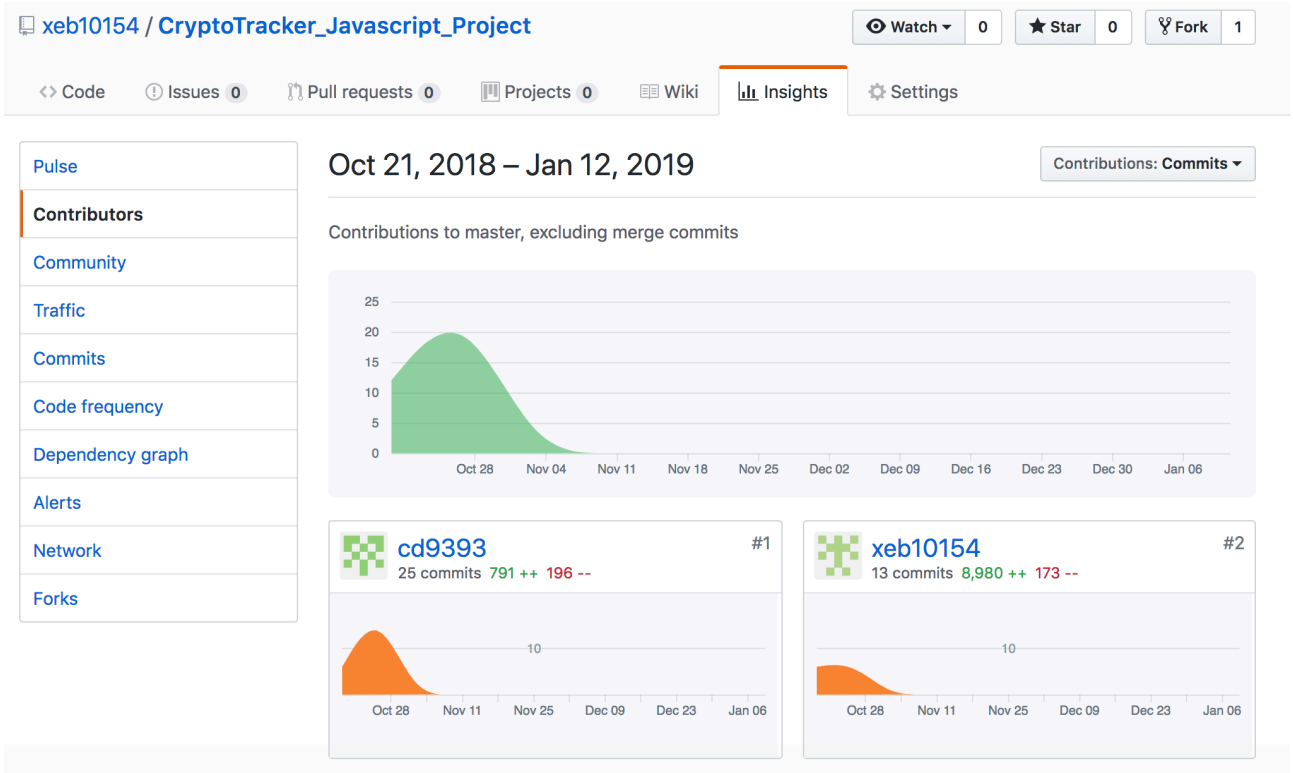
```ruby
require_relative('./card')
class CardGame

  def checkforAce(card)
    if card.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card1.name
    elsif card1.value == card2.value
      return card1.name
    else
      return card2.name
    end
  end

  def self.cards_total(cards)
    total = 0
    for card in cards
      total += card.value
    end
    return "You have a total of #{total}."
  end
end

```

The test code passing:

```
➜ Static_and_Dynamic_Task_A git:(master) ✗ ruby spec/testing_task_2.rb
Run options: --seed 21960

# Running:

.....

Finished in 0.001673s, 2988.6433 runs/s, 2988.6433 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
➜ Static_and_Dynamic_Task_A git:(master) ✗
```

Week 9

Unit	Ref	Evidence
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.
		Description:



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

Shares Portfolio Application

A local trader has come to you with a portfolio of shares. She wants to be able to analyse it more effectively. She has a small sample data set to give you and would like you to build a Minimum Viable Product that uses the data to display her portfolio so that she can make better decisions.

MVP

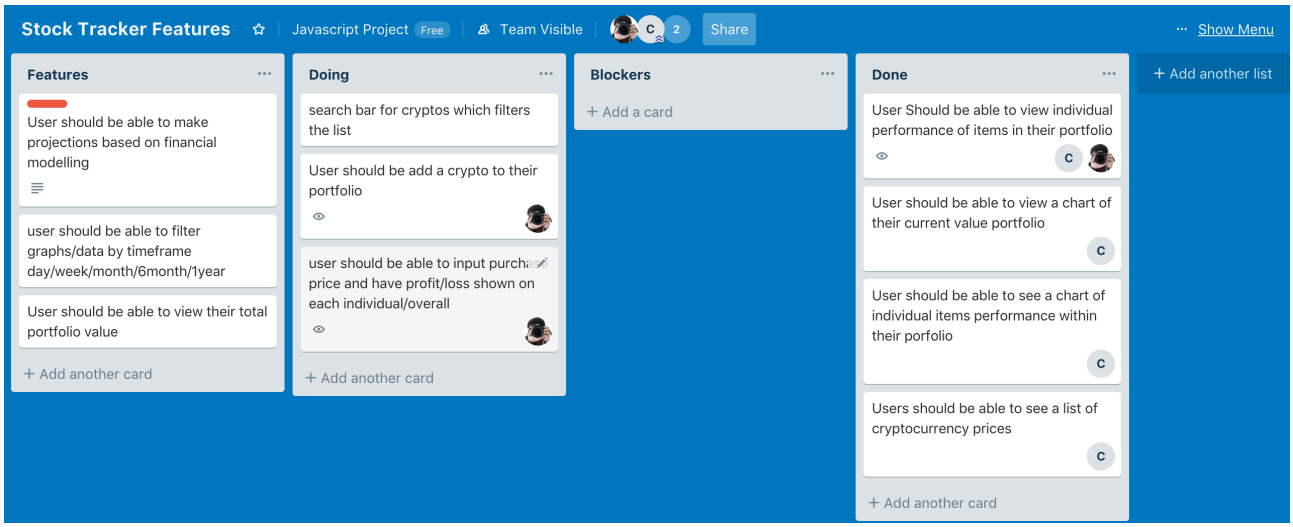
A user should be able to:

- view total current value.
- view individual and total performance trends.
- retrieve a list of share prices from an external API and allow the user to add shares to her portfolio.
- View a chart of the current values in her portfolio.

Example Extensions

- Speculation based on trends and further financial modelling using projections.

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



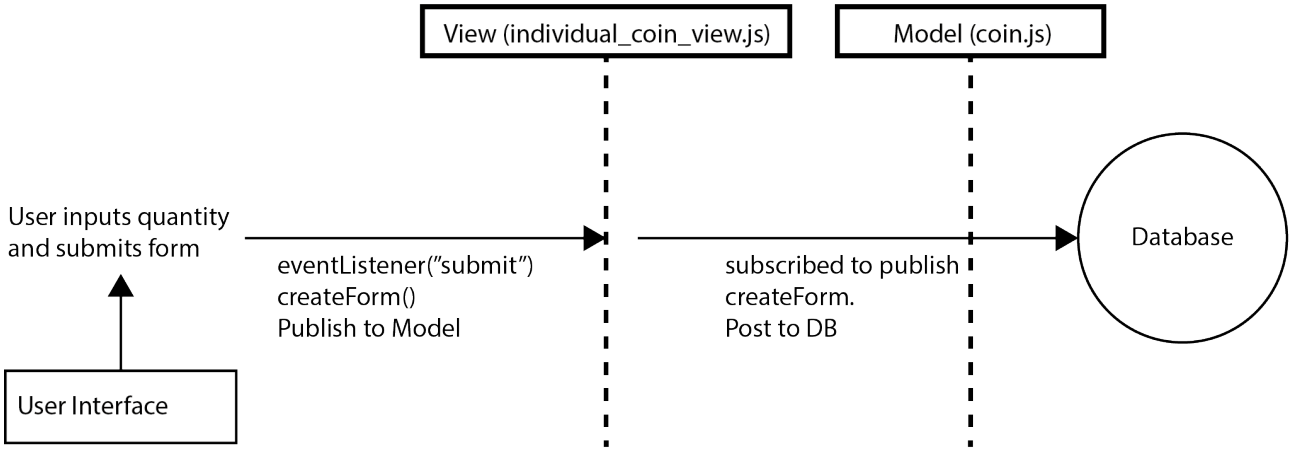
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

CryptoTracker group project,
Acceptance criteria and test plan

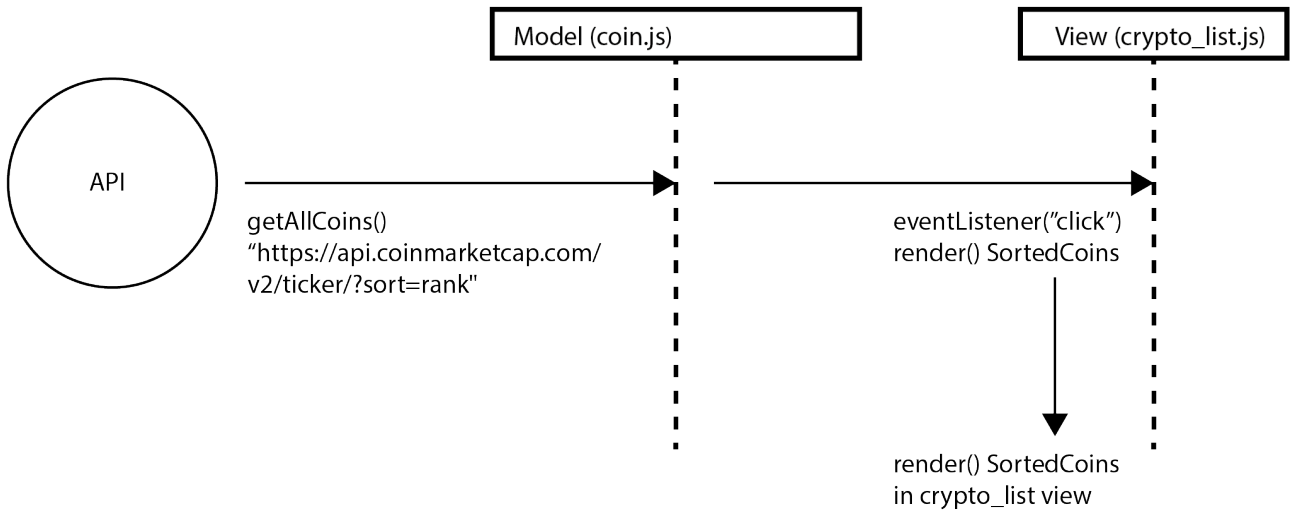
Acceptance Criteria - "A user should be able to…"	**Expected Output/Result**	**Test Result** (Pass/Fail)
view total current value.	• The main page should display total current value when a user visits the home page of the website	Pass
view individual and total performance trends.	• The user should be able to view total performance trends when the individual element is clicked	Pass
retrieve a list of share prices from an external API and allow the user to add shares to her portfolio.	• A user should be able to view a list of all API entries retrieved when the 'Crypto' button is clicked • A user should be able to add a Crypto asset to the list when the 'add' button is clicked	Pass
View a chart of the current values in her portfolio.	• The user should be able to view a performance chart of the asset when the portfolio asset is clicked	Pass

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

Sequence Diagram 1:

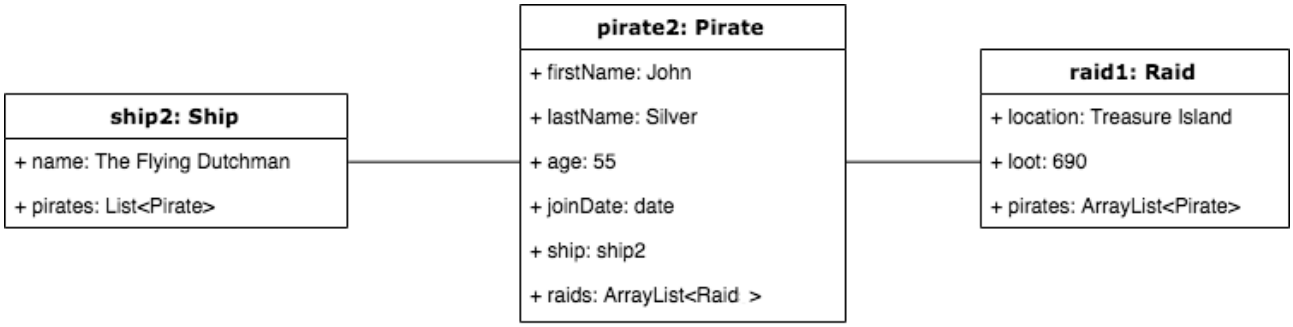


Sequence Diagram 2:

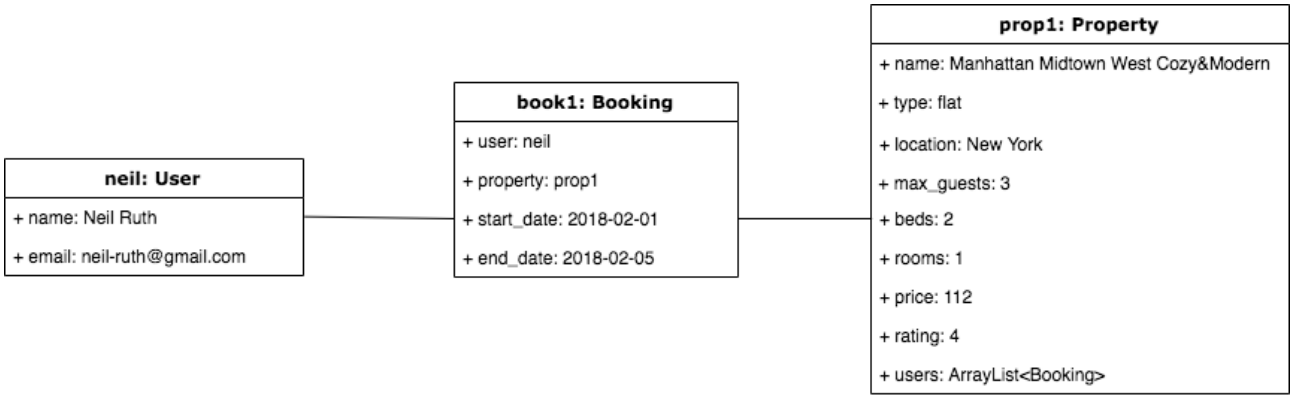


Unit	Ref	Evidence
P	P.8	Produce two object diagrams.
		Description:

Object diagram 1



Object diagram 2



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

CryptoTracker Application: Bug report.

Bug Identified	Solution	Date
Portfolio not displaying on screen - waiting for data to be returned from multiple API calls, leading to asynchronous behaviour.	Declare ‘async’ function and add ‘await’ expression to resolve asynchronous behaviour.	04 Nov 2018
Portfolio not rendering when delete button is clicked - subscribe eventListener not receiving data.	Initiate top level function in App.js (entry point), triggering bindEvents() function to pass data to eventListener	04 Nov 2018
Webpage refreshes when submit buttons is clicked.	Event trigger - added preventDefault() to resolve form submit issue.	02 Nov 2018
SearchCoin() function is returning an ‘undefined’ value in the model	‘return’ keyword is required outside of the loop statement	02 Nov 2018
Adding of API ‘Price’ values to existing database price results in a ‘NaN’	API price require to be parsed to decimal, using parseFloat(Coin-Price)	02 Nov 2018

Week 12

Unit	Ref	Evidence
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.
		Description: The model of a computer network (Network.java) has an array list called “devices” that contains objects implementing Polymorphism through an interface called IConnect (line 11). IConnect objects also belong to different classes e.g. a printer in the network would be a Printer object as well as an IConnect object. Since a Printer is both a Printer object as well as an IConnect object, this makes it Polymorphic and allows it to be added to the ‘ArrayList<IConnect> devices’.

Network model containing arrayList of Polymorphic IConnect objects

Network.java | Printer.java | Desktop.java | InternetRadio.java | IConnect.java

```
1  import behaviours.IConnect;
2
3  import java.util.*;
4
5  public class Network {
6      private String name;
7      private ArrayList<IConnect> devices;
8      private int maxConnections;
9
10     public Network(String name){
11         this.devices = new ArrayList<IConnect>();
12         this.name = name;
13         this.maxConnections = 2;
14     }
15
16     public String getName() { return name; }
```

Interface: IConnect

Network.java | Printer.java | Desktop.java | InternetRadio.java | IConnect.java

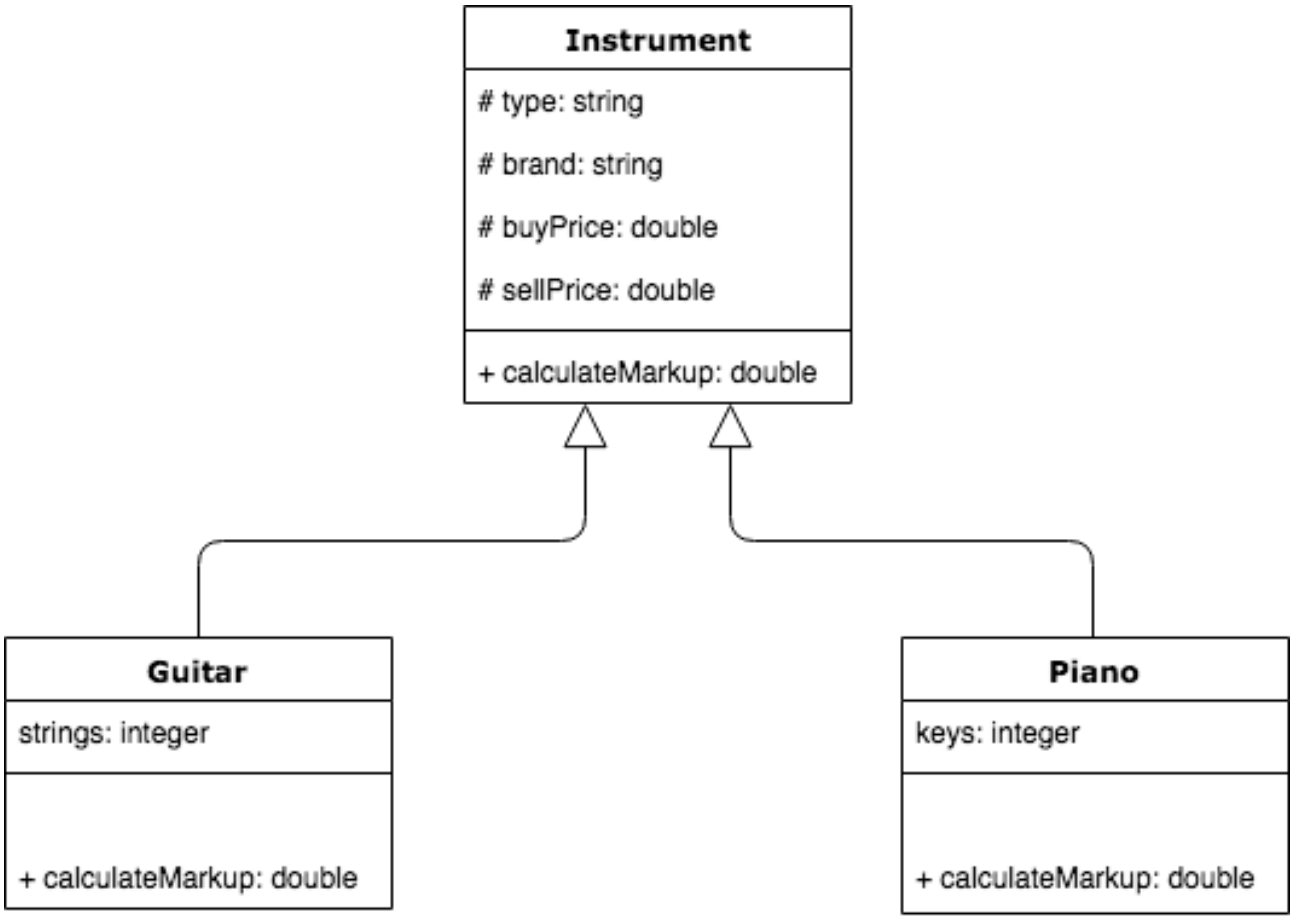
```
1  package behaviours;
2
3  public interface IConnect {
4
5      String connect(String data);
6      // Everything in an interface is assumed ot be Public access.
7  }
8
```

Printer model implementing Polymorphism using IConnect Interface

Network.java | Printer.java | Desktop.java | InternetRadio.java | IConnect.java

```
1  import behaviours.IConnect;
2
3  public class Printer implements IConnect {
4
5      public Printer() {
6
7      }
8
9      public String print(String data) { return "Printing: " + data; }
12
13     public String connect(String data) {
14         return "connecting to network: " + data;
15     }
16 }
```

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



Unit	Ref	Evidence
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.
		Description: All instance variables are set as private, making them only accessible inside this class named 'Instrument'. A public "getter" method e.g. getType() is used for external components to access the variables belonging to this class.

Screenshot of abstract class using Encapsulation

Piano.java | Guitar.java | SheetMusic.java | Instrument.java | Accessories.java

```java
package Instruments;

import Behaviours.IPlay;
import Behaviours.ISell;

public abstract class Instrument implements IPlay, ISell {

    private String type;
    private String brand;
    private double buyPrice;
    private double sellPrice;

    public Instrument(String type, String brand, double buyPrice, double sellPrice) {
        this.type = type;
        this.brand = brand;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getType() { return type; }

    public String getBrand() { return brand; }

    public double getBuyPrice() { return buyPrice; }

    public double getSellPrice() { return sellPrice; }

    public double calculateMarkup() {
        return this.sellPrice - this.buyPrice;
    }

}
```

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

Screenshot of a class: Accessories

```
1   package Accessories;
2
3   import Behaviours.ISell;
4
5   public abstract class Accessories implements ISell {
6
7       private String type;
8       private double buyPrice;
9       private double sellPrice;
10
11      public Accessories(String type, double buyPrice, double sellPrice) {
12          this.type = type;
13          this.buyPrice = buyPrice;
14          this.sellPrice = sellPrice;
15      }
16
17      public String getType() { return type; }
20
21      public void setType(String type) { this.type = type; }
24
25      public double getBuyPrice() { return buyPrice; }
28
29      public void setBuyPrice(double buyPrice) { this.buyPrice = buyPrice; }
32
33      public double getSellPrice() { return sellPrice; }
36
37      public void setSellPrice(double sellPrice) { this.sellPrice = sellPrice; }
40
41      public double calculateMarkup() { return this.sellPrice - this.buyPrice; }
44  }
45
```

Screenshot of a class that inherits from the previous class: SheetMusic

```
1   package Accessories;
2
3   import Accessories.Accessories;
4
5   public class SheetMusic extends Accessories {
6
7       private int sheets;
8
9       public SheetMusic(String type, double buyPrice, double sellPrice) {
10          super(type, buyPrice, sellPrice);
11      }
12
13
14  }
15
```

Screenshot of an Object in the inherited class (line 14) and a screenshot of a method using information inherited from another class i.e. line 23-25 shows an inherited method using the properties buyPrice and sellPrice.

```
12      @Before
13      public void setUp(){
14          sheetMusic = new SheetMusic( type: "Classical",  buyPrice: 5,  sellPrice: 12.5);
15      }
16
17      @Test
18      public void getBuyPrice(){
19          assertEquals( expected: 5, sheetMusic.getBuyPrice(),  delta: 0.01);
20      }
21
22      @Test
23      public void calculateMarkup(){
24          assertEquals( expected: 7.5, sheetMusic.calculateMarkup(),  delta: 0.01);
25      }
26
27  }
28
```

Unit	Ref	Evidence
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.
		Description: **Algorithm 1:** This algorithm verifies that a member has not been double booked for a particular gym class by examining each member's gym class bookings. This method was chosen as it demonstrates the use of a for loop with a nested conditional if-statement to verify that the current instance of the gym class does not match any of the existing booked instances. If a matching instance is found, the algorithm returns 'true'. **Algorithm 2:** This algorithm checks whether the 'current_time' selected falls within the peak time hours. This was chosen to demonstrate the use of conditional if-statement with logical operators (| | - OR sign) in conjunction with formatted times. If 'current_time' falls within the peak time hours, the algorithm returns 'true'.

Algorithm 1:

```
103     def doubleBooked(member)
104       booked = false
105       for each in member.gymclasses
106         if each.id == @id
107           booked = true
108         end
109         # binding.pry
110       end
111       return booked
112     end
```

Algorithm 2:

```
85    def peakTime()
86      peaktime = false
87      # Hard coded times
88      # Morning peak times
89      t1 = Time.new(2018, 1, 1, 7, 0, 0, 0).strftime("%H%M%S")
90      t2 = Time.new(2018, 1, 1, 9, 0, 0, 0).strftime("%H%M%S")
91      # Evening peak times
92      t3 = Time.new(2018, 1, 1, 17, 0, 0, 0).strftime("%H%M%S")
93      t4 = Time.new(2018, 1, 1, 20, 0, 0, 0).strftime("%H%M%S")
94
95      current_time = self.start_time.strftime("%H%M%S")
96      # binding.pry
97      if current_time.between?(t1,t2) || current_time.between?(t3,t4)
98        peaktime = true
99      end
100     return peaktime
101   end
```