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: arrayOfSongs is an array of song titles. The function printSongs is using the command puts to print the array of song titles to the console.

An example of an array in a program

```ruby
# arrayOfSongs is an array of song titles
arrayOfSongs = ["faith", "bohemian rhapsody", "man in the mirror", "englishman in new york"]
```

A function that uses the array

```ruby
#printSongs is a function that prints the array of song titles to the console
def printSongs(arrayOfSongs)
    puts arrayOfSongs
end

printSongs(arrayOfSongs)
```

The result of the function running

```
~/codeclan/work_files/pda  master ●  ruby ruby.rb
faith
bohemian rhapsody
man in the mirror
englishman in new york
~/codeclan/work_files/pda  master ●
```

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: listOfArtists is a hash of music artists and their respective genres. The function printArtists is using the command puts to print the hash to the console.

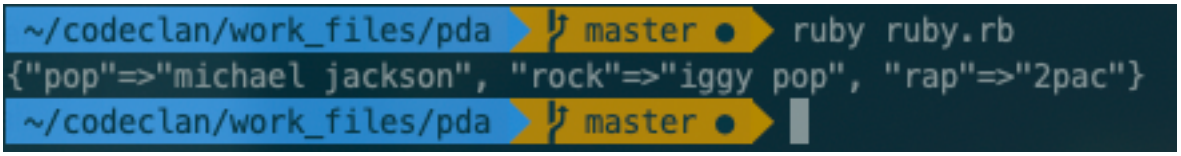
An example of a hash in a program

```
# listOfArtists is a hash of musicians and their respective genres
listOfArtists = { "pop" => "michael jackson", "rock" => "iggy pop", "rap" => "2pac"}
```

A function that uses the hash

```
# printArtists is a function that prints the hash of artists and genres to the console
def printArtists(listOfArtists)
    puts listOfArtists
end

printArtists(listOfArtists)
```

The result of the function running



Week 3

Unit	Ref	Evidence
I&T	I.T.3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running
		Description: Bounty.find_id is a function that searches a database table for an entry that has the corresponding id number. The id is passed into the function when it is called and inserted into an SQL query checking the table bounties for matching entries.

An example of a function that searches data

```ruby
#Bounty.find is a function that searches a database to find a bounty by id
  def Bounty.find_id(id)
    db = PG.connect ({dbname: 'bounties', host: 'localhost'})
    sql = "SELECT * FROM bounties
           WHERE id = $1"
    values = [id]
    db.prepare("find_id", sql)
    queried_id = db.exec_prepared("find_id", values)
    db.close()
    queried_id.count > 0 ? queried_id[0] : nil
  end
```

The result of the function running

```
[13] pry(main)> Bounty.find_id(14)
=> {"id"=>"14",
 "name"=>"Frank Corvin",
 "species"=>"Human",
 "bounty_value"=>"300",
 "danger_level"=>"medium",
 "last_known_location"=>"Saturn",
 "homeworld"=>"Earth",
 "favourite_weapon"=>"BioGun",
 "cashed_in"=>"t",
 "collected_by"=>"John Shephard"}
[14] pry(main)>
```

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: Bounty.all is a function that retrieves all entries from a database table called bounties and returns an ascending sorted list by entry bounty_value. It takes no parameters.

An example of a function that sorts data

```ruby
#Bounty.all is a function that sorts database entries of bounties by their value in ascending order
def Bounty.all()
  db = PG.connect ({dbname: 'bounties', host: 'localhost'})
  sql = "SELECT * FROM bounties ORDER BY bounty_value ASC"
  db.prepare("all", sql)
  all_bounties = db.exec_prepared("all")
  db.close()
  return all_bounties.map {|entry| Bounty.new(entry)}
end
```

The result of the function running

In Postico, the data is not sorted by bounty_value.

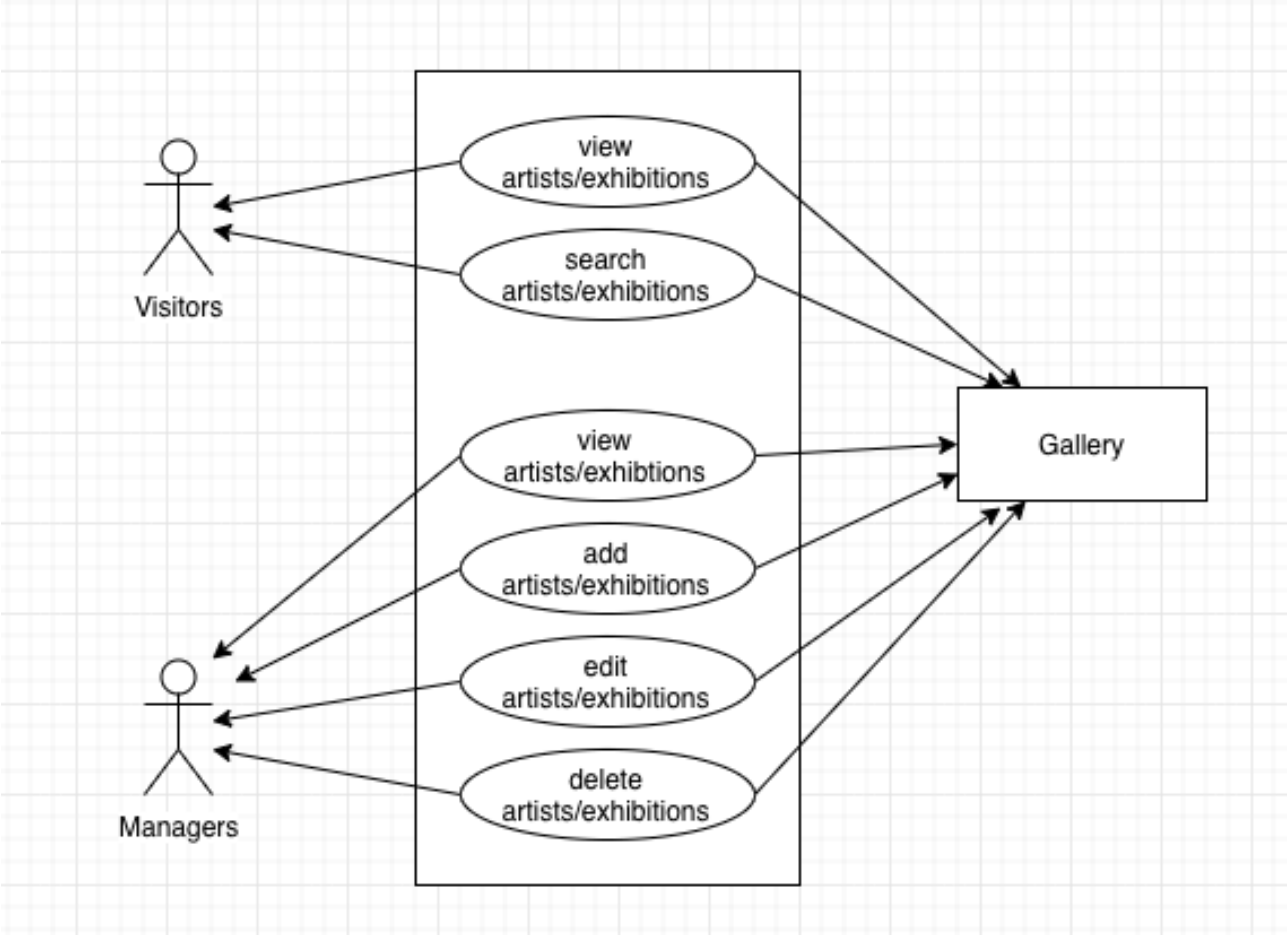
id	name	species	bounty_value	danger_level	last_known_location	homeworld	favourite_weapon	cashed_in	collected_by
18	Frank Corvin	Human	300	medium	Saturn	Earth	BioGun	TRUE	John Shephard
19	Hawk Hawkins	Centipede	500	high	Antares	Trappist A	RailGun	FALSE	
20	Tank Sullivan	Bolian	100	low	Illium	Tuchanka	Handgun	TRUE	John Shephard
21	Jerry ONeill	Ωgan	1000	ermagerdyerderd	Pylos Nebula	Caliston Rift	Biotic Flare	FALSE	

In the console, the data is being sorted in ascending order by the bounty_value field. 100 is the smallest value in the dataset while 1000 is the largest.

Week 5 and 6

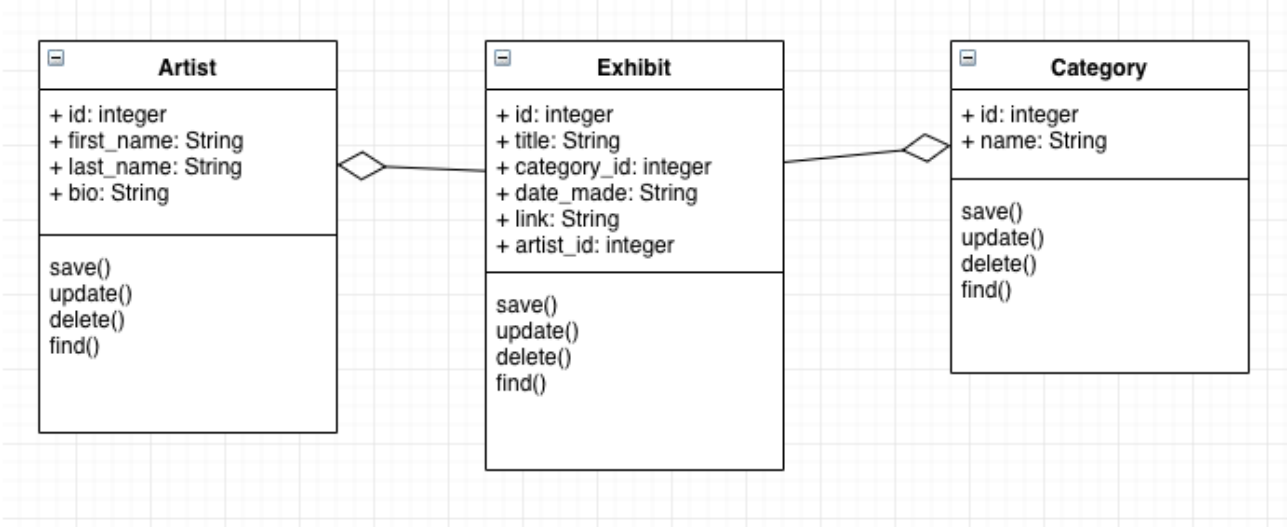
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram
		Description: This diagram shows two users and the desired functionality for their uses. Visitors only have read access to the database entries whereas managers have full read/write access.

A use case diagram



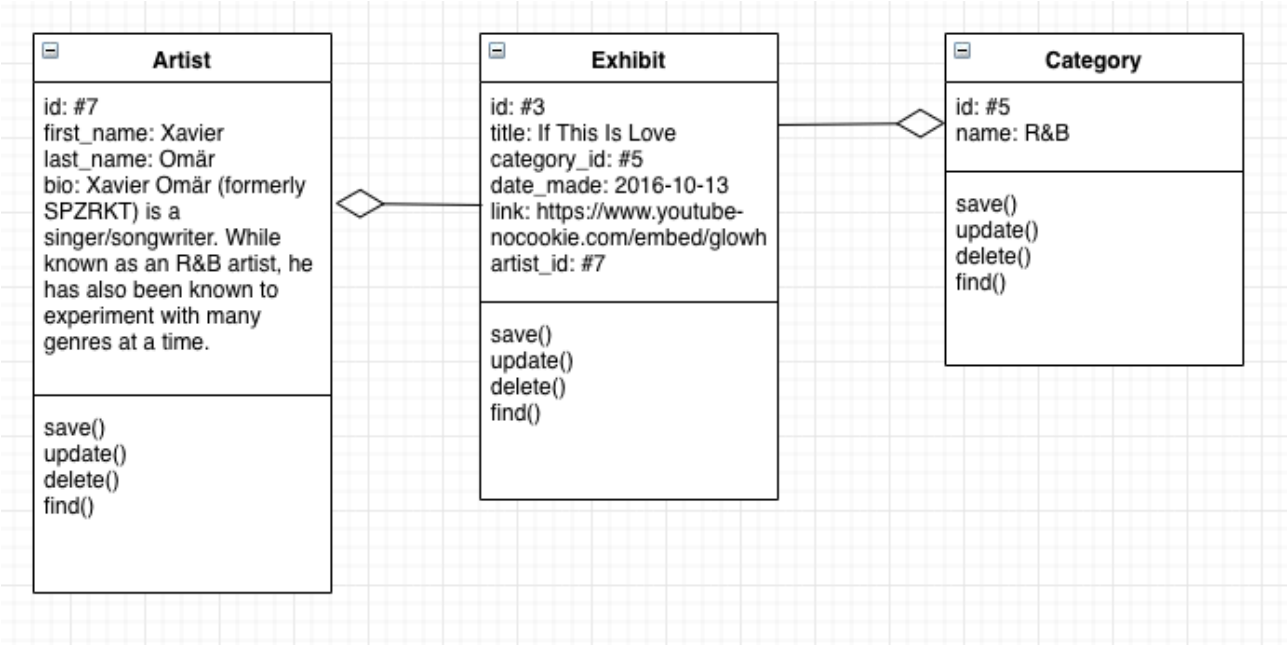
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		Description: These classes are database models with several fields of data. The Exhibit class has a many to one relationship to both the Category and Artist classes. Each class has methods for save, update delete and find.

A class diagram



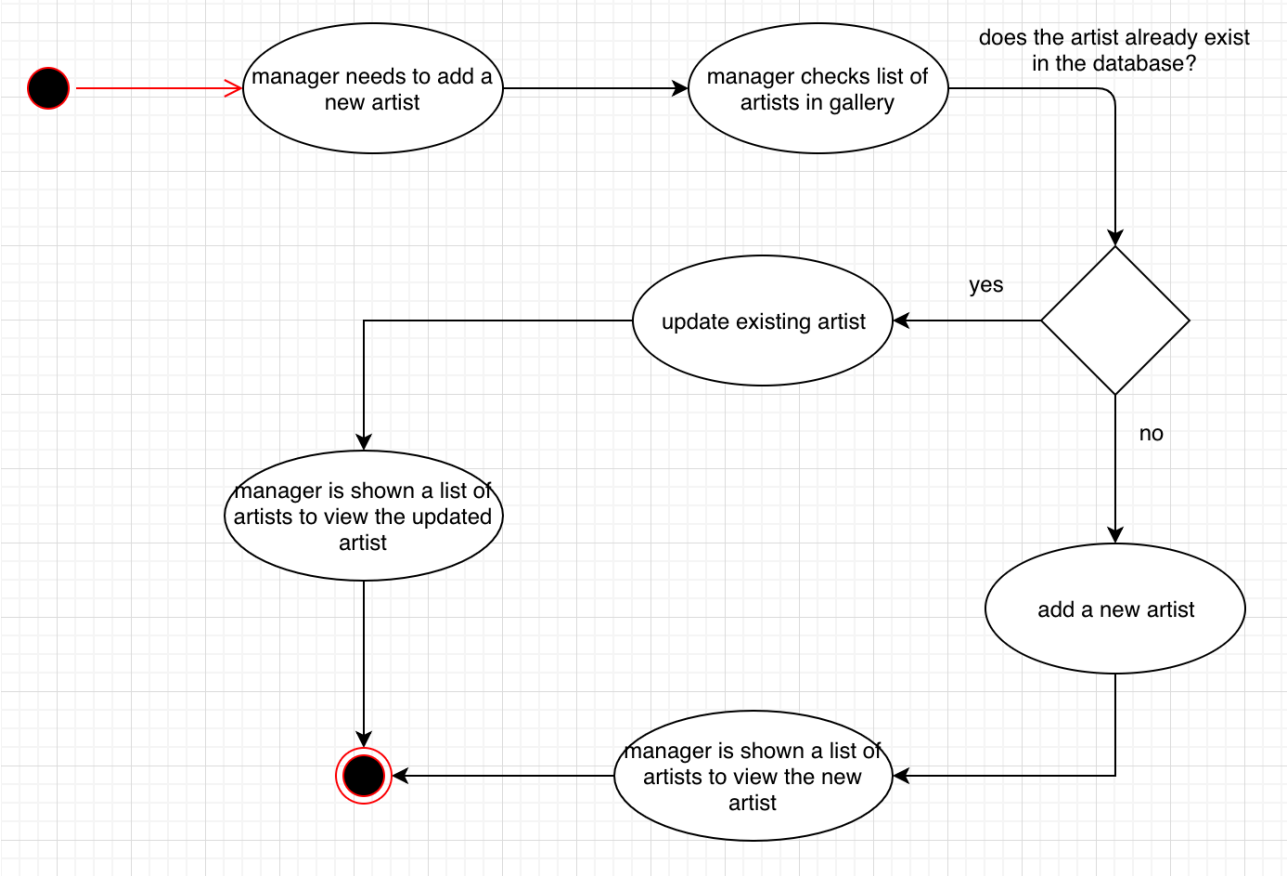
Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram
		Description: This object diagram shows three objects and their associated data fields. The Exhibit object has a many to one relationship with both the Category and Artist objects.

An object diagram



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

An activity diagram. This activity diagram illustrates the steps that should be taken in order to add a new artist to the database of a gallery



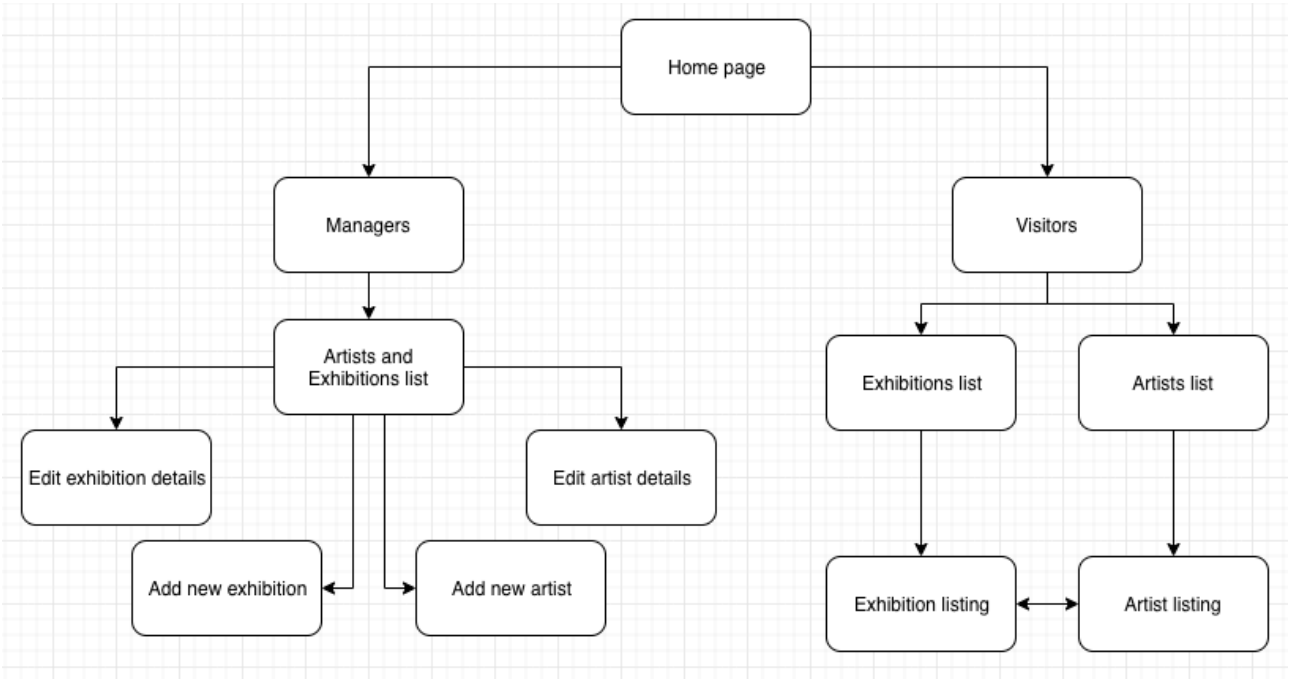
Unit	Ref	Evidence
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time
		Description:

An Implementations Constraints plan for an art gallery app.

Topic	Possible effect on constraint of product	Solution
Hardware and software platforms	Users may not be able to operate the art gallery app on different devices (smartphone, laptop, tablet). This will restrict the accessibility of the app and the number of users able to use it.	Use device agnostic design procedures in regards to CSS layouts and ensure it can fit different devices.
Performance requirements	The art gallery app loads significantly slower on mobile devices (smartphones) due to slower embedded memory.	Create a web design that uses small images sparingly that have been compressed by an image editing program.
Persistent storage and transactions	The art gallery app requires a database for all artists and exhibitions to be created, read, updated and deleted. The CRUD operations must operate exactly the same on every device in order to keep a correct inventory.	Ensure that database CRUD operations are sufficiently tested on different devices before releasing the app.
Usability	The inventory of the art gallery must be displayed in an intuitive and easy to use fashion to ensure users are not put-off using the app.	Making use of UX principles with user stories, requirements and accessibility as the basis for the design and layout of the app.
Budgets	Budget should be adhered to in order to ensure all features are present in the final app. If the budget runs out before completion, the remaining features to be implemented will have to be reconsidered.	Set realistic goals for the Minimum Viable Product and achieve this before moving on to further features. This will ensure at least a basic functioning app is created.
Time	The deadline for the app is non-negotiable. Failure to complete the app by the deadline will result in some features being dropped and possible untested bugs present.	Make a realistic MVP goal as above and carry out procedural unit and integration testing throughout development. As long as the MVP is created, extra features can be added later.

Unit	Ref	Evidence
P	P.5	User Site Map
		Description: This user site map shows the pages and their links in hierarchical form to the home page of an art gallery.

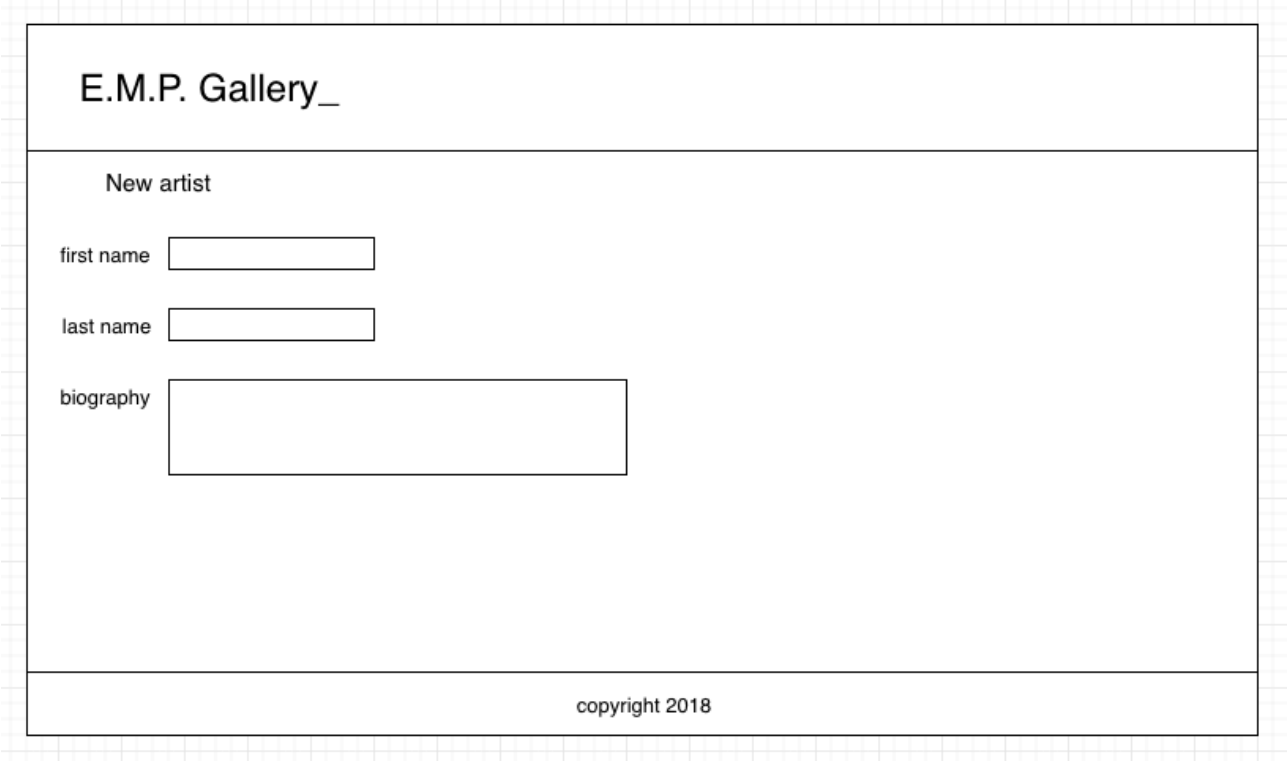
A user site map



Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams
		Description: The first wireframe diagram shows the home page of the art gallery app. There is a header with a logo, an introduction instruction, two buttons and a footer. The second diagram shows the entry of a new artist into the database page. There are fields for text entry along with the header and footer.

A wireframe diagram for the homepage of an art gallery app
A wireframe diagram for the add new artist page on an art gallery app

Unit	Ref	Evidence
P	P.10	Example of Pseudocode used for a method
		Description: This pseudocode explains a method to retrieve a JSON file from an API with questions of a particular category.

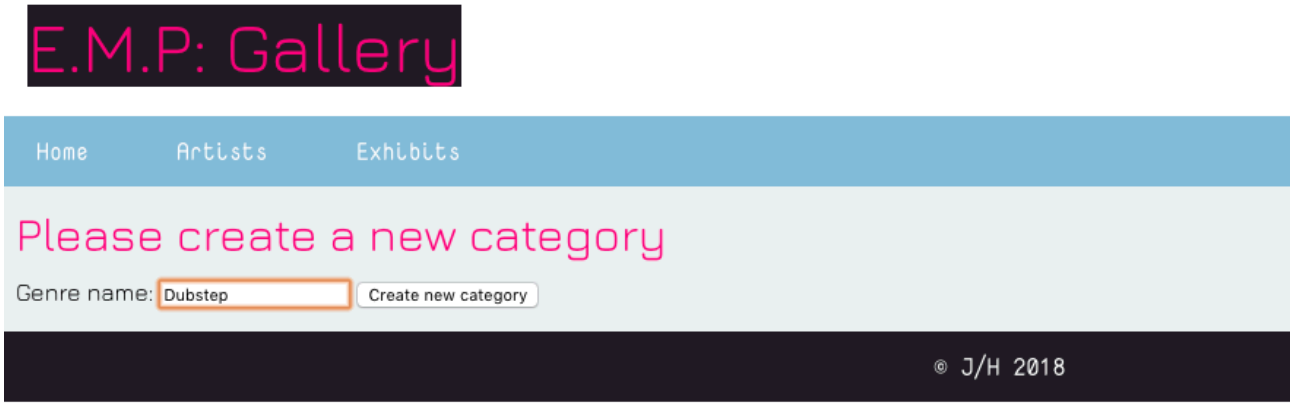


An example of pseudocode used for a method

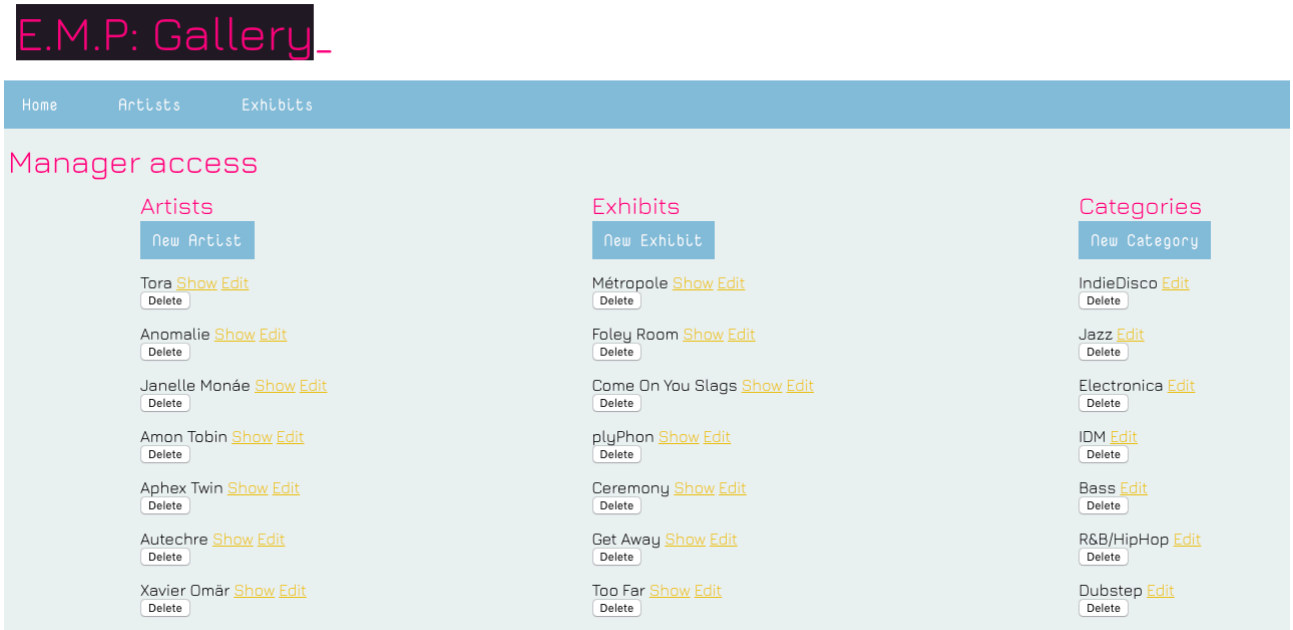
```
// loadCategoryQuestions function needs to load a JSON file of questions from an api
// for a particular category

// pass in one argument to the function: the category of the questions to be retrieved.

// prep the base URL for the API along with the chosen category
// call a new request with the preped URL
// handle asynchronous request event
// send the received JSON file contents into a hashmap for the particular category
```

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:

The user inputting a new category into the EMP gallery app

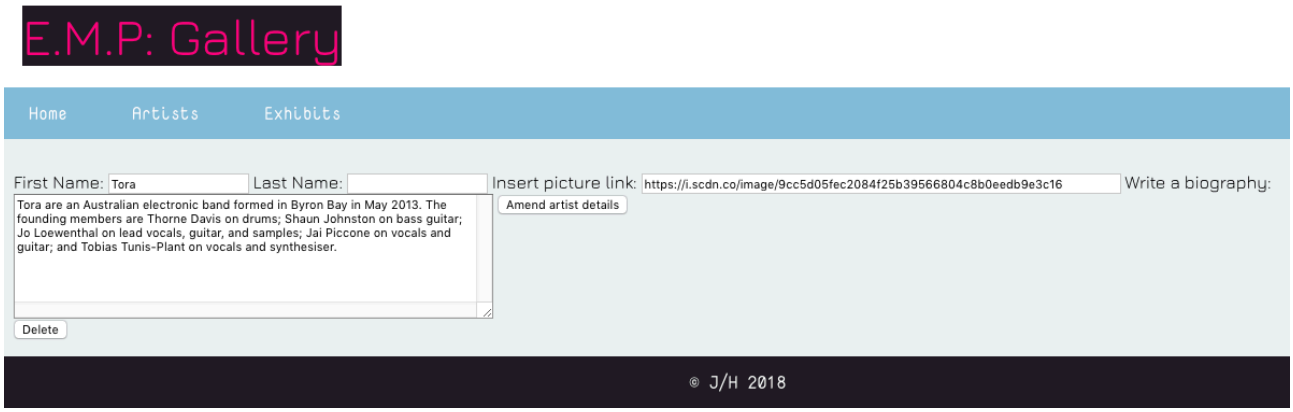


The user input being saved and displayed in a list of categories. The new category Dubstep can be seen at the bottom of the list of categories.

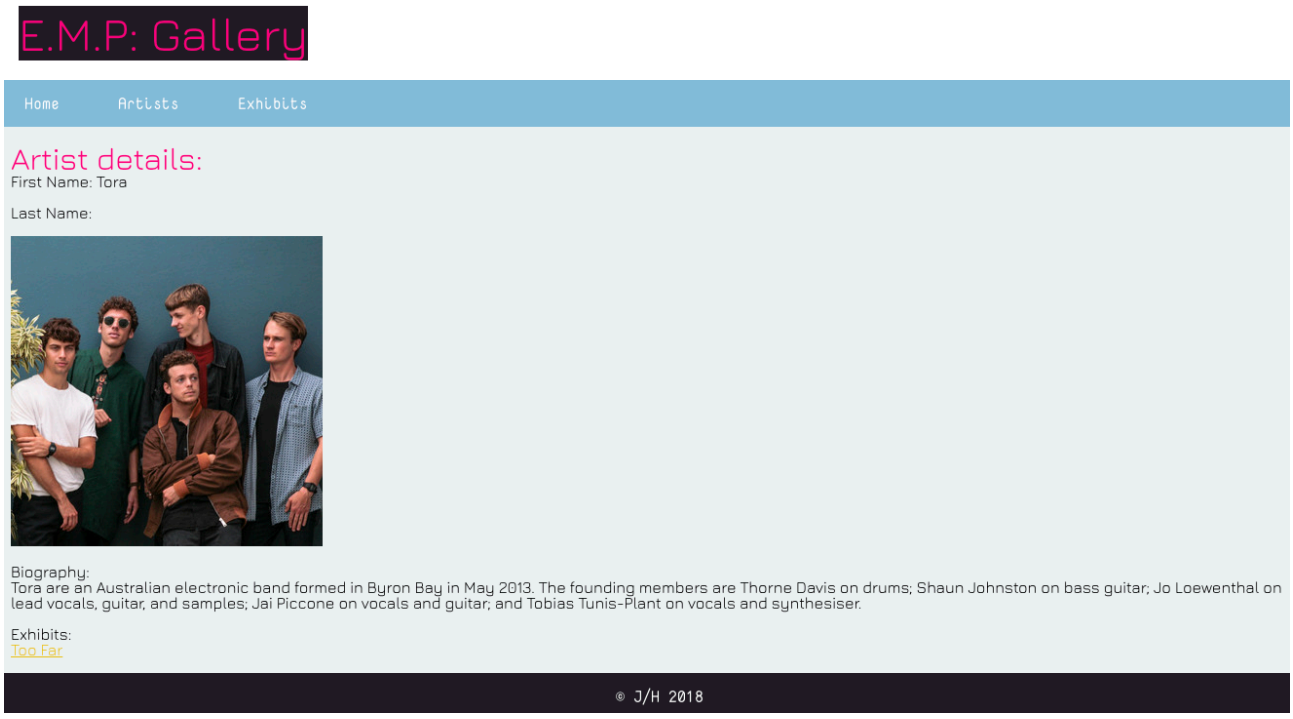


Unit	Ref	Evidence
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved
		Description:

An example of data being inputted into the EMP gallery app. A new artist is being created.



Confirmation of the data being saved. The new artist is displayed with all the details.

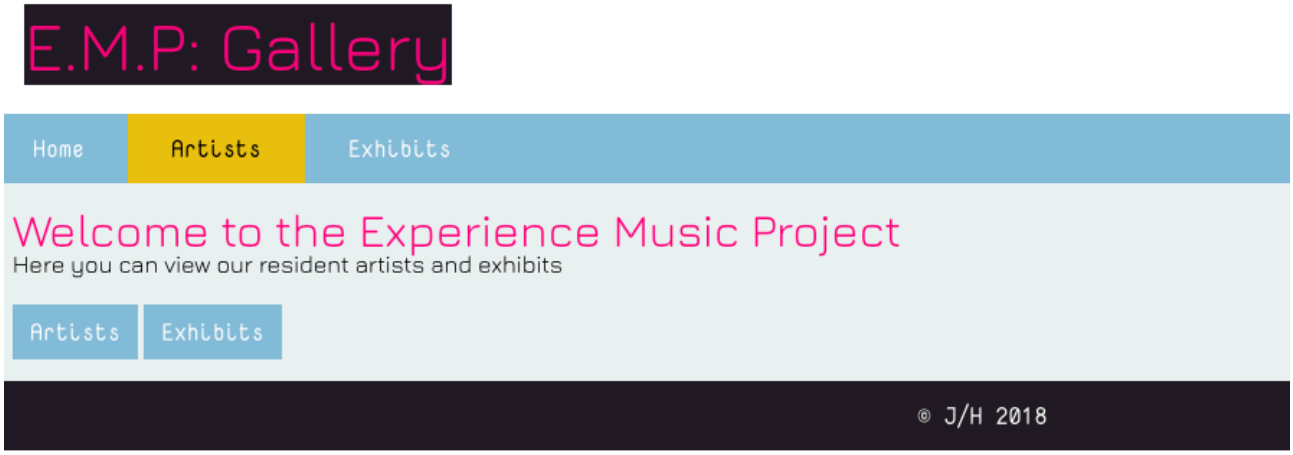


In Postico, the artist details have been saved under Tora.

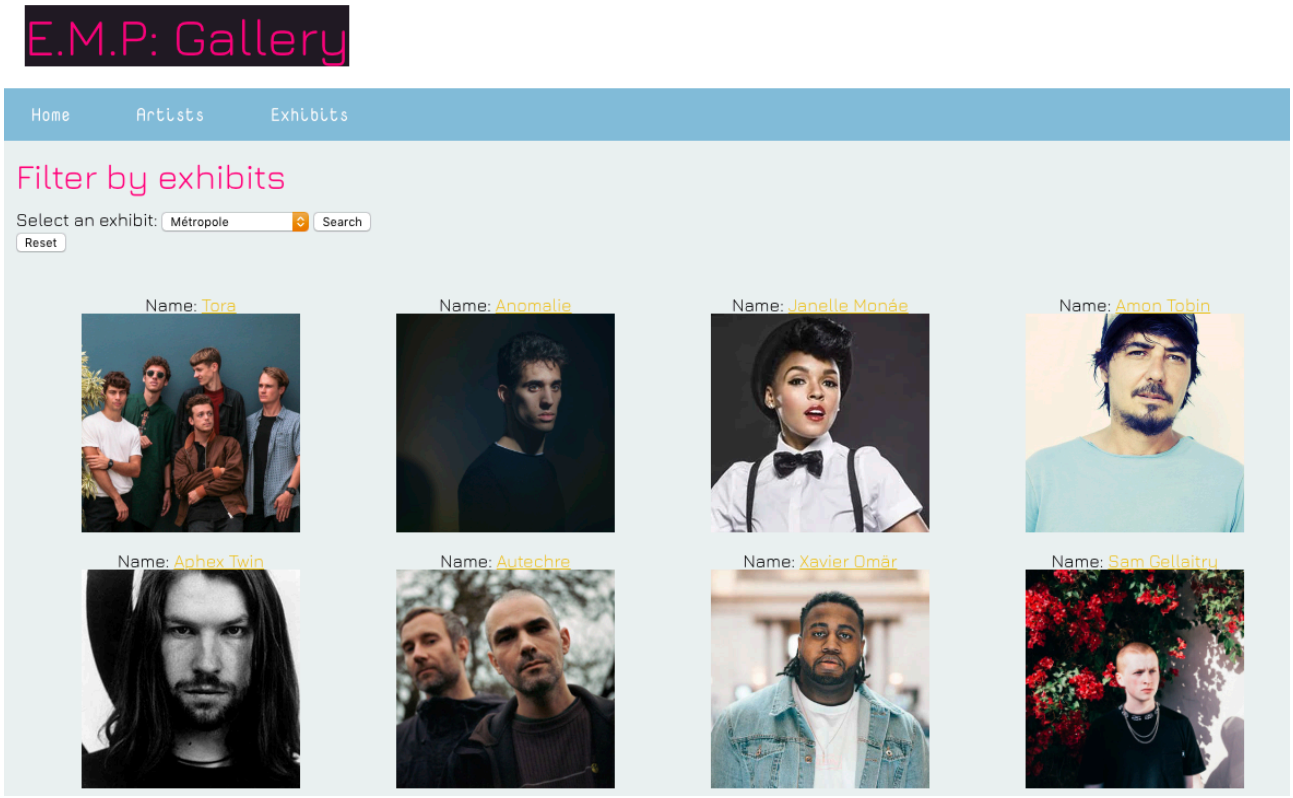
id	first_n...	last_name	bio	link
41	Amon	Tobin	Amon Adonai Santos de Araújo Tobin born February 7, 1972, known as Amon Tobin, is a Brazilian musician, composer and producer of electron...	https://pp.userapi.com/c620316/v620316524/14d3a/Qo07krYzCuc.jpg?ava=1
39	Anomalie	NULL	In 2012, classically trained keyboardist Nicolas Dupuis decided to hedge his bets and develop a new sound after becoming the first ever Yannick...	https://d3jjg4nf4bbybe.cloudfront.net/u/195793/602470c0c92cd969f341fada58d8e32c0...
42	Aphex	Twin	Richard David James born 18 August 1971, best known by his main alias Aphex Twin, is an English electronic musician raised in Cornwall. He is b...	https://pbs.twimg.com/profile_images/1096942082/28ad84ebe563235ce83aef72a3a96...
43	Autechre	NULL	Autechre are an English electronic music duo consisting of Rob Brown and Sean Booth, both from Rochdale, Greater Manchester. Formed in 1...	https://images.sk-static.com/images/media/profile_images/artists/49992/huge_avatar
40	Janelle	Monáe	Janelle Monáe Robinson born December 1, 1985 is an American singer, songwriter, rapper, actress, and model. She is signed to her own imprint...	https://vignette.wikia.nocookie.net/supernatural-ties/images/6/64/Janelle-monae.jpg/revision/latest?cb=...
45	Sam	Gellaitry	Gellaitry, now 20, started producing at the age of 12 Scotland where he has spent much of his youth. Having been influenced by a wide-range o...	https://yt3.ggpht.com/a-/ACSszfEzTyDTtmvF9xkjXQqI3VKtsZ_2YJRM0cjUkg=...
46	The	Internet	The Internet is an American band from Los Angeles, California. It currently consists of Syd, Matt Martians, Patrick Paige II, Christopher S...	https://video-images.vice.com/articles/5b0d772c3b76b60008a48d90/lede/1527841591...
38	Tora	NULL	Tora are an Australian electronic band formed in Byron Bay in May 2013. The founding members are Thorne Davis on drums; Shaun Johnston on...	https://i.scdn.co/image/9cc5d05fec2084f25b39566804c8b0eedb9e3c16
44	Xavier	Omär	Xavier Omär (formerly SPZRKT) is a singer/songwriter. While known as an R&B artist, he has also been known to experiment with many genres at...	https://mk0bookingagent97efu.kinstacdn.com/wp-content/uploads/2017/07/Xavier-Omar-Contact-Info...

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:

The user clicks on the artists section in order to see a list of all artists in the gallery.

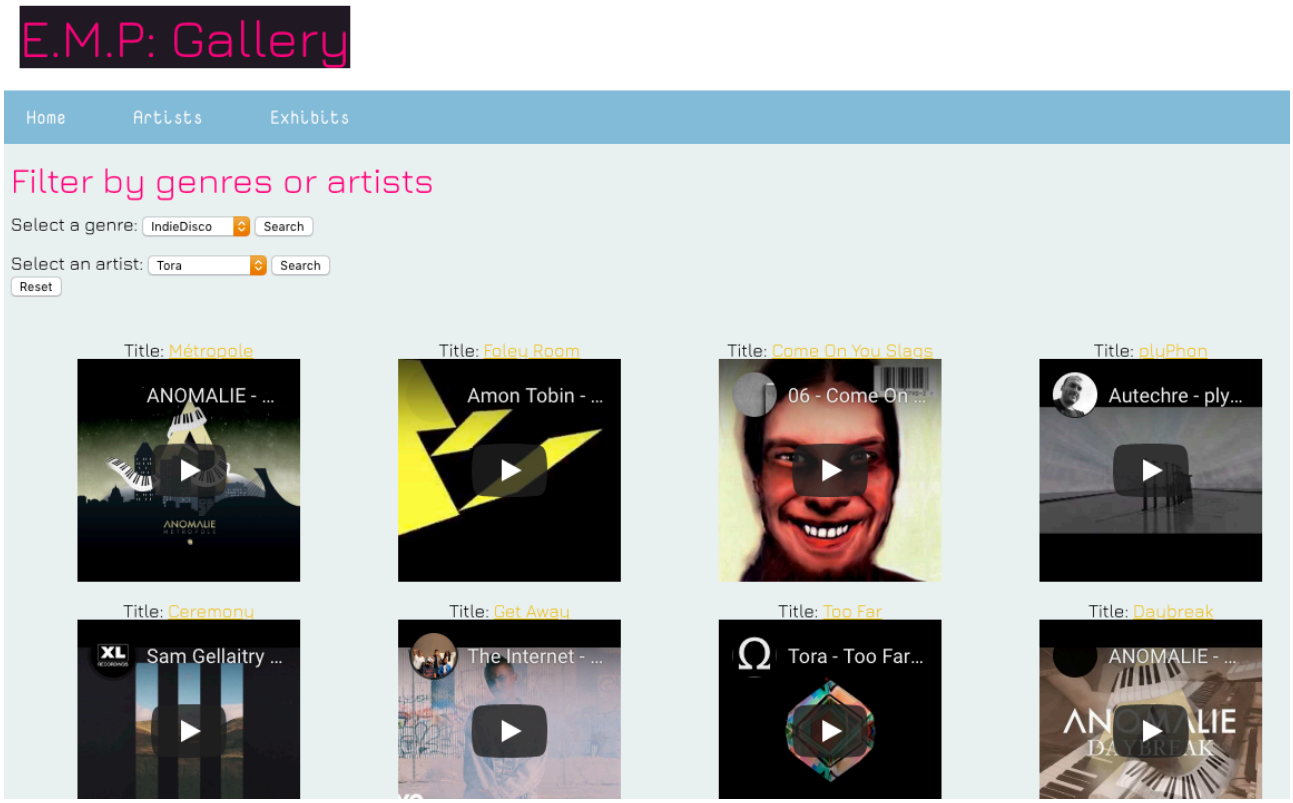


The user request being processed correctly and demonstrated in the program



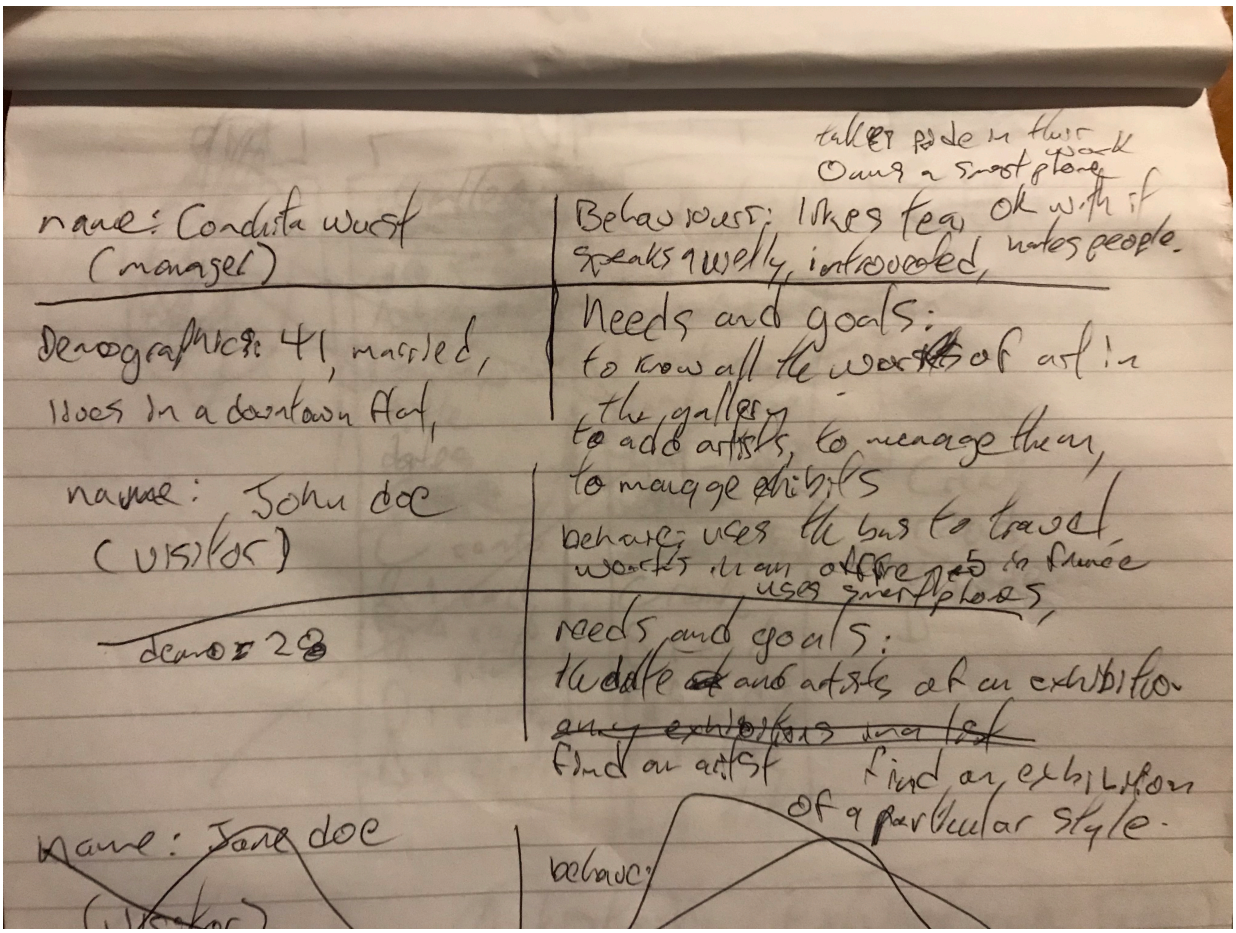
Unit	Ref	Evidence
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.
		Description:

This EMP gallery project I worked alone to create. The GitHub link is: https://github.com/self-unit/EMP_gallery

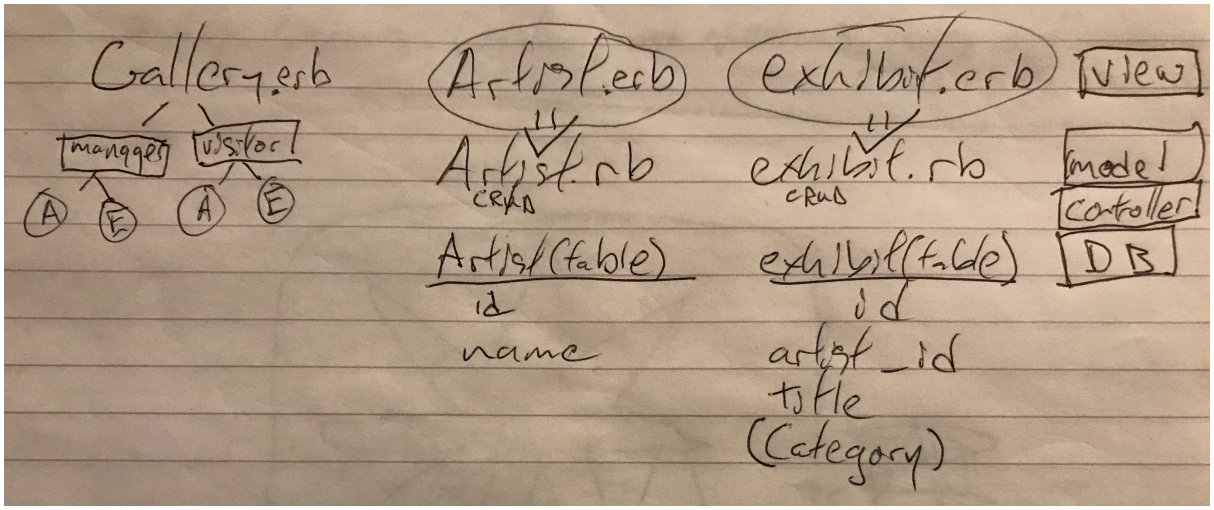


Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.
		Description:

A user needs profile for two imagined people showing their lifestyles and expectations when using the art gallery app.

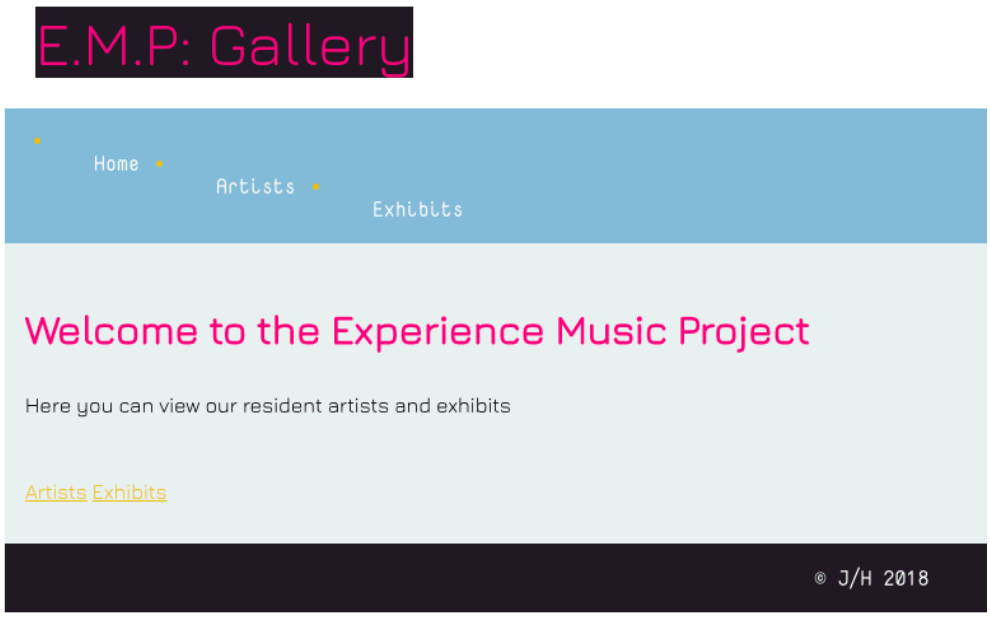


An early diagram showing the file structure and functionality of each file in the project.

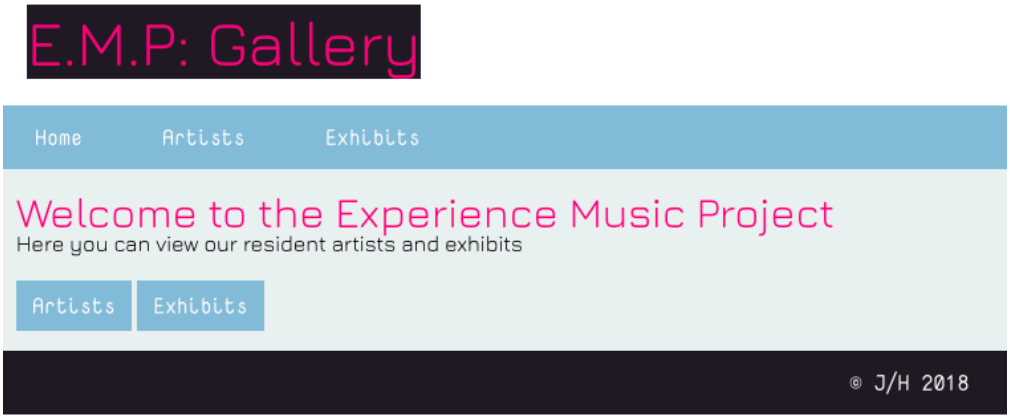


A screenshot of the very first look of the website homepage

A screenshot after some layout work with CSS



A screenshot showing the final design with buttons for the artist and exhibitions sections



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:

A helper function that takes in a parameter URL for an API and requests data from it using the fetch command.

```
const Request = function (url) {
  this.url = url;
};

Request.prototype.get = function () {
  return fetch(this.url)
  .then((response) => response.json());
};
```

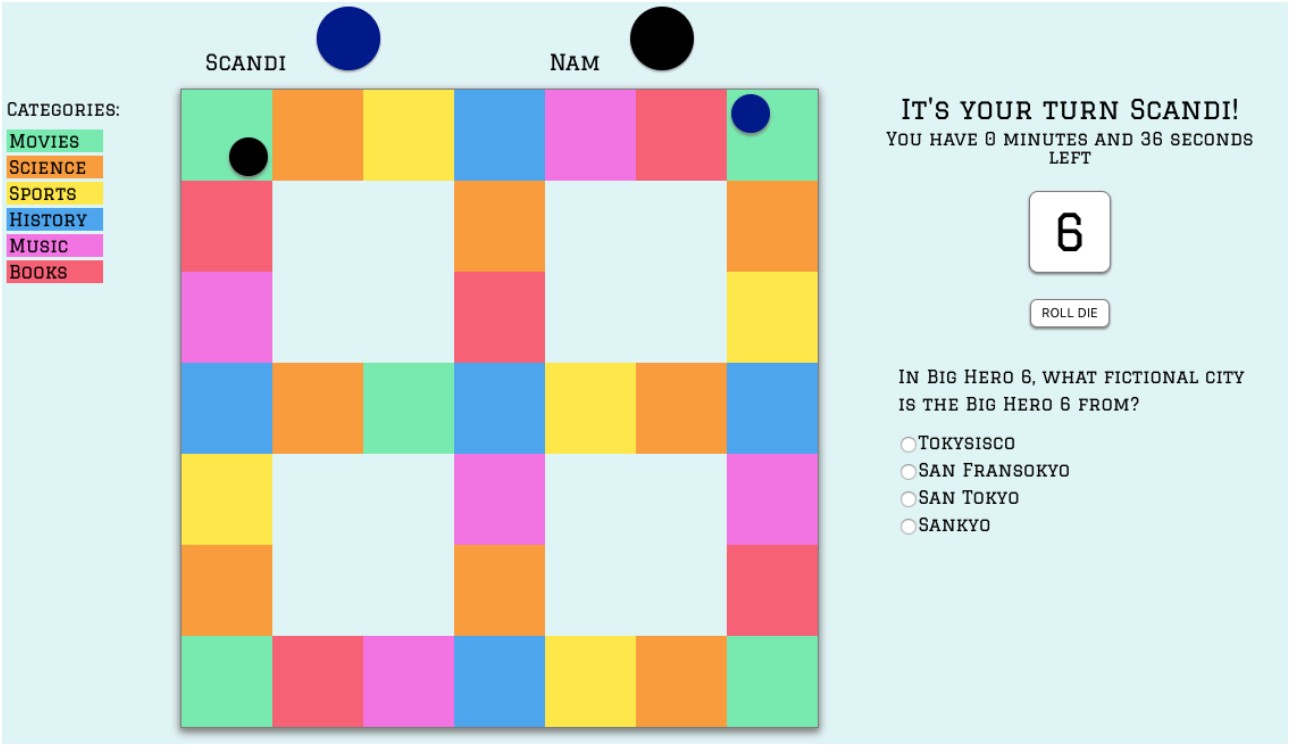
A code fragment that has the URL for the API to be requested from

```
const Request = require('../helpers/request.js');
const PubSub = require('../helpers/pub_sub.js');
const randomizeArray = require('../helpers/randomize_array.js');

const Card = function(difficulty) {
  this.baseUrl = 'https://opentdb.com/api.php?amount=20&category=';
  this.currentQuestion = 0;
  this.difficulty = difficulty;

  this.categories = [{
    "name": "movies",
    "categoryId": 11,
    "currentCard": 0,
    "cards": null,
  }, {
    "name": "science",
    "categoryId": 17,
    "currentCard": 0,
    "cards": null,
  }, {
```

A function that takes in a category as a parameter and uses this to amend the this.baseUrl with the category and difficulty before passing it into the request helper function that uses asynchronous fetch to retrieve the JSON file

```
Card.prototype.loadCategoryQuestions = function (category) {
  const quizUrl = `${this.baseUrl}${category[0]['categoryId']}&difficulty=${this.difficulty}&type=multiple`;
  const request = new Request(quizUrl);

  request.get()
  .then((cards) => {
    category[0]['cards'] = cards.results.splice(0, 25)
    this.sortQuestion(category[0])
  });
};
```

A screenshot of the API being used by the app. This API has been queried to get a JSON file for a category that the player token has landed on. The file has a question and multiple answers which are displayed on the bottom right of the screen



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:

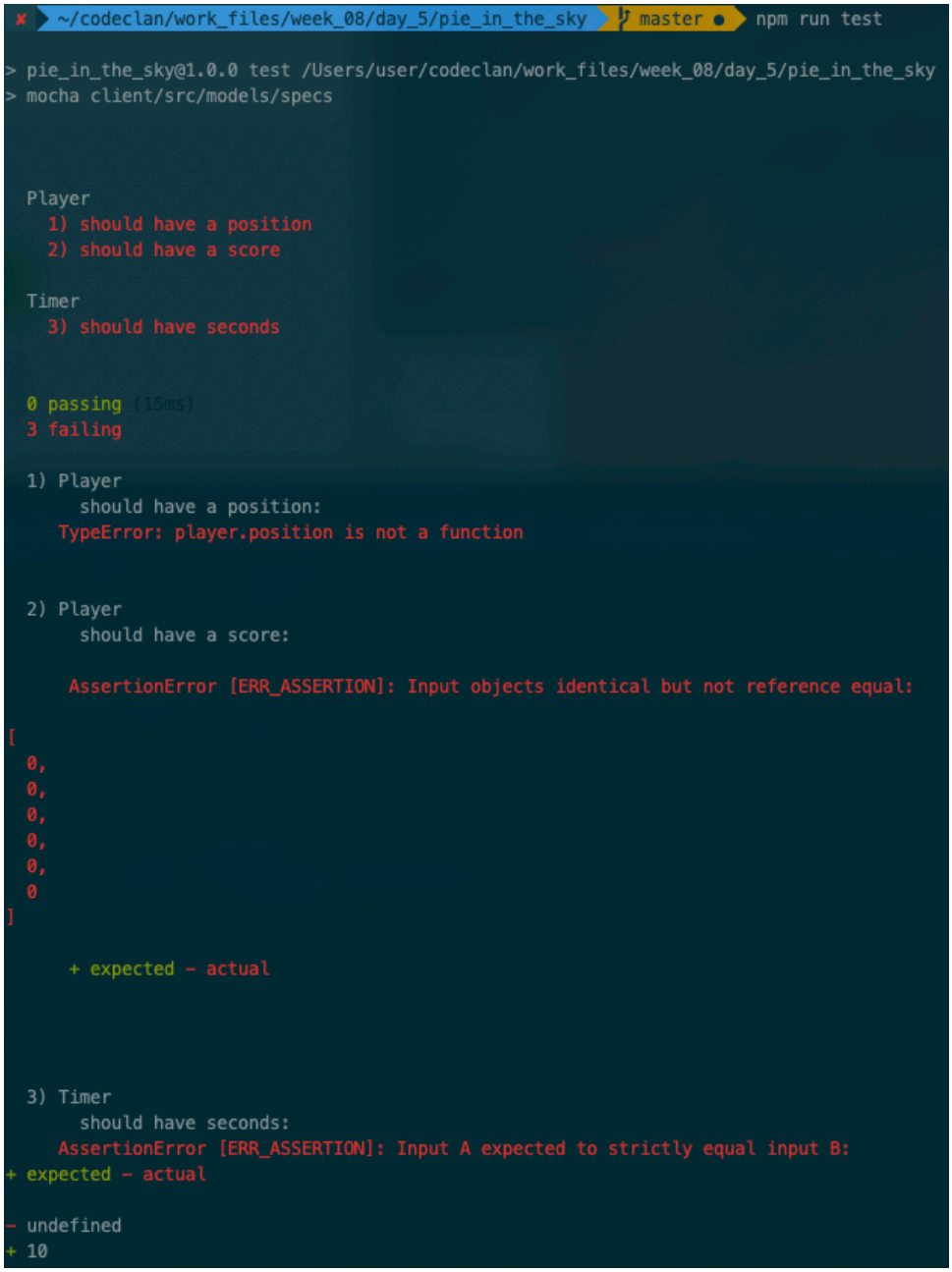
A class called Player which takes in several properties name, id & player. It has a function rollDie which uses the Math.floor(Math.random()) function with arguments to make sure the final result is always between 1 and 6.

```
const PubSub = require('../helpers/pub_sub.js');

const Player = function (name, id, player) {
  this.icon = null;
  this.name = name;
  this.id = id;
  this.score = [0, 0, 0, 0, 0, 0];
  this.position = 'a1';
  this.player = player;
  this.colour = null;
};

Player.prototype.rollDie = function () {
  const randomNumber = Math.floor(Math.random() * 6 + 1 );
  PubSub.publish('Player:rollnumber', randomNumber);
};

module.exports = Player;
```

Test code that checks if the player class has a starting position of a1 and an empty score of 0 for 6 array values.

```
const assert = require('assert');
const Player = require('../player.js');

describe("Player", function () {

beforeEach(function () {
  player = new Player ('Mike', 'p2', 'player2');
})

  it("should have a position", function () {
    const actual = player.position();
    assert.strictEqual(actual, 'a1');
  });

  it("should have a score", function () {
    const actual = player.score;
    assert.strictEqual(actual, [0,0,0,0,0,0]);
  });


});
```

The incorrectly written test code failing to pass due to several errors.



Correctly written test code that has been amended to change the actual value in the first test to a variable of player instead of a function and to use deepStrictEqual to compare the array of 6 0's.

```
const assert = require('assert');
const Player = require('../player.js');

describe("Player", function () {

beforeEach(function () {
  player = new Player ('Mike', 'p2', 'player2');
})

  it("should have a position", function () {
    const actual = player.position;
    assert.strictEqual(actual, 'a1');
  });

  it("should have a score", function () {
    const actual = player.score;
    assert.deepStrictEqual(actual, [0,0,0,0,0,0]);
  });


});
```

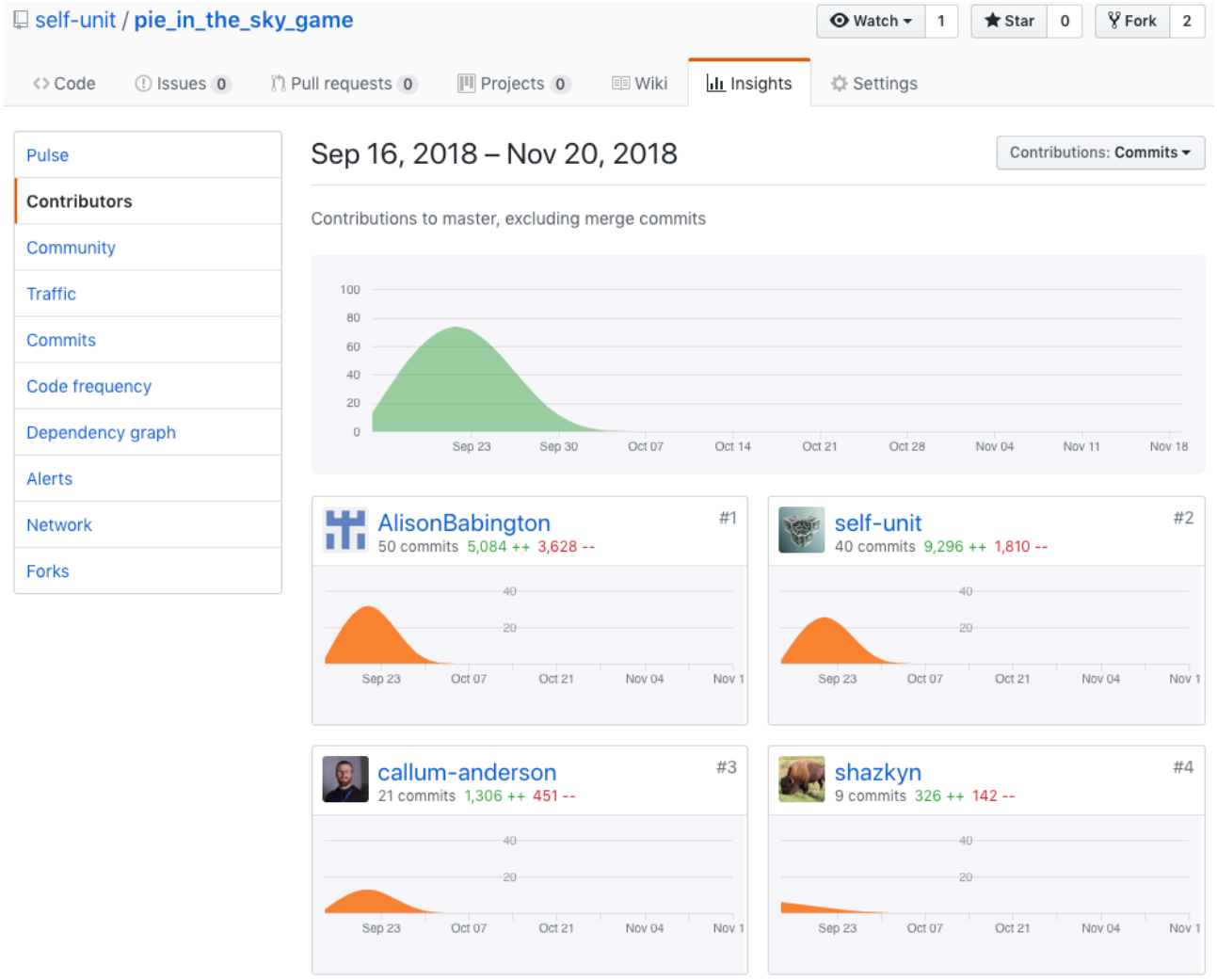
The final test working as expected.

```
✘ ~/codeclan/work_files/week_08/day_5/pie_in_the_sky  master ● npm run test

> pie_in_the_sky@1.0.0 test /Users/user/codeclan/work_files/week_08/day_5/pie_in_the_sky
> mocha client/src/models/specs



  Player
    ✓
    ✓

  Timer
    ✓


  3 passing
```

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:

A screenshot of the contributor's page on my Github group project Pie in the Sky.



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

A screenshot of the project brief for our group project

Browser Game

Create a browser game based on an existing card or dice game. Model and test the game logic and then display it in the browser for a user to interact with.

Write your own MVP with some specific goals to be achieved based on the game you choose to model.

You might use persistence to keep track of the state of the game or track scores/wins. Other extended features will depend on the game you choose.

A screenshot of the additional brief details our group created for the project

Trivial Pursuit MVP

A user should be able to roll the die.

A user should be able to move around the board.

A user should be able to answer multi-category questions.

A user should have a score.

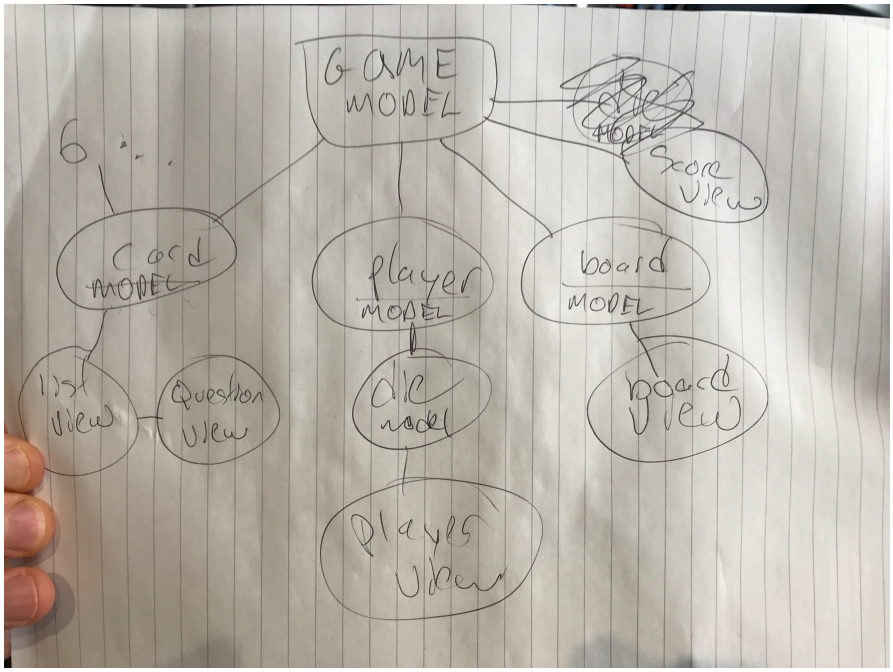
Possible Extensions

A user should be able to win the game.

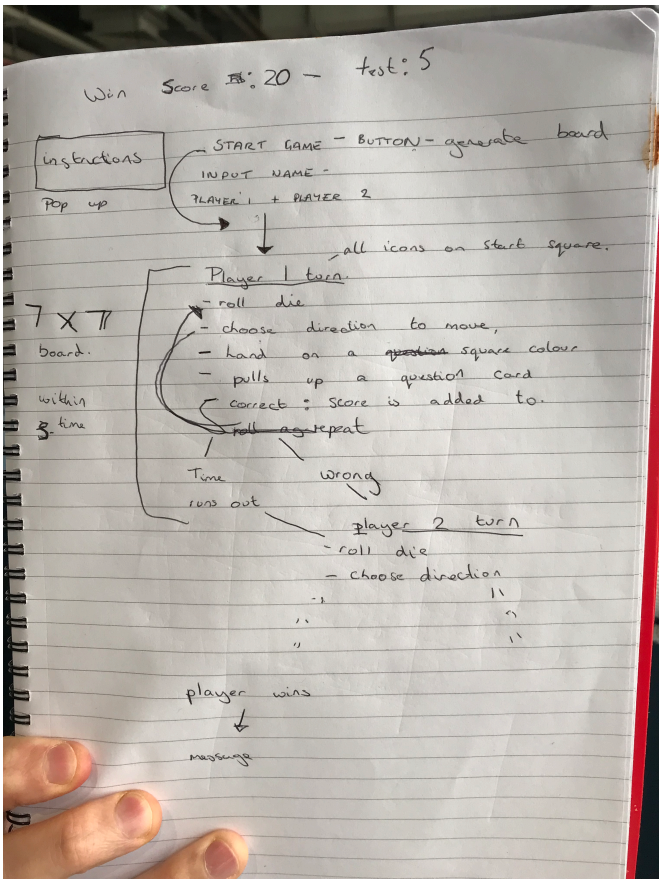
A user should be able to enter their name.

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

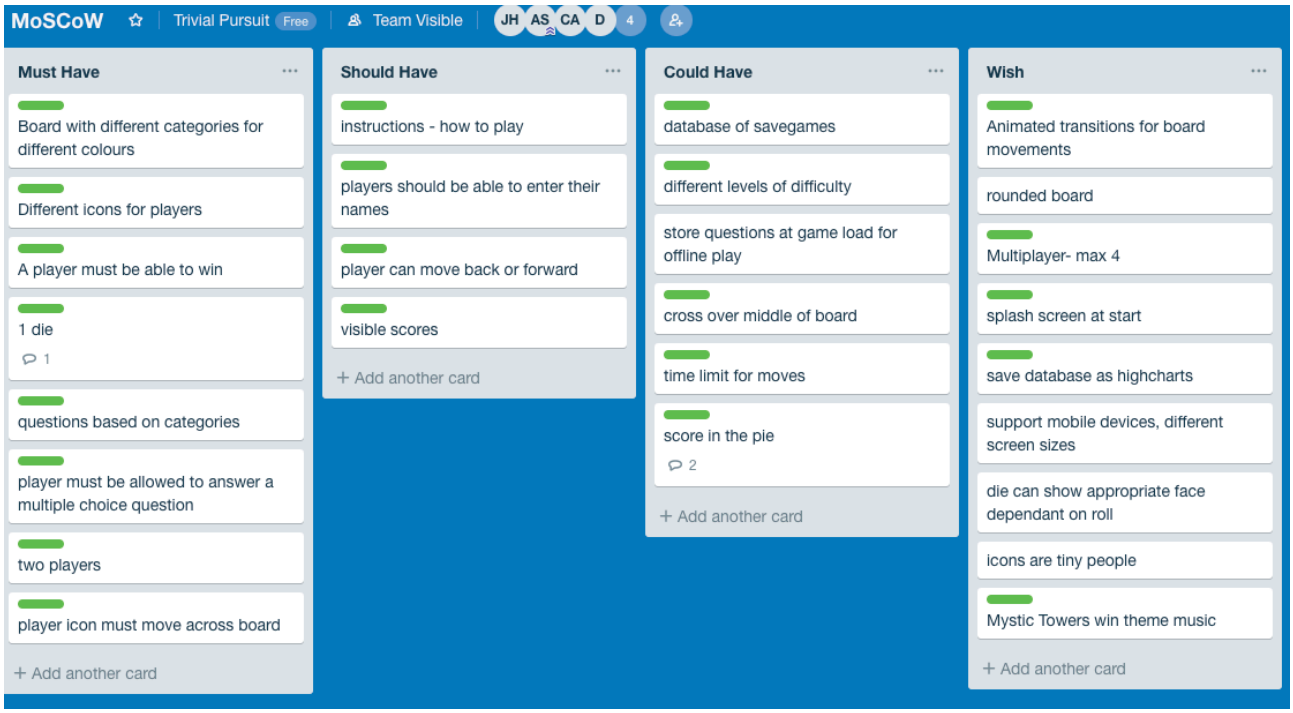
Hierarchy diagram of how all classes were to be related in the project



A user journey diagram that explains the progress an individual player would have when playing



A screenshot of the MoSCow board used in our group project with green highlights on all completed tasks



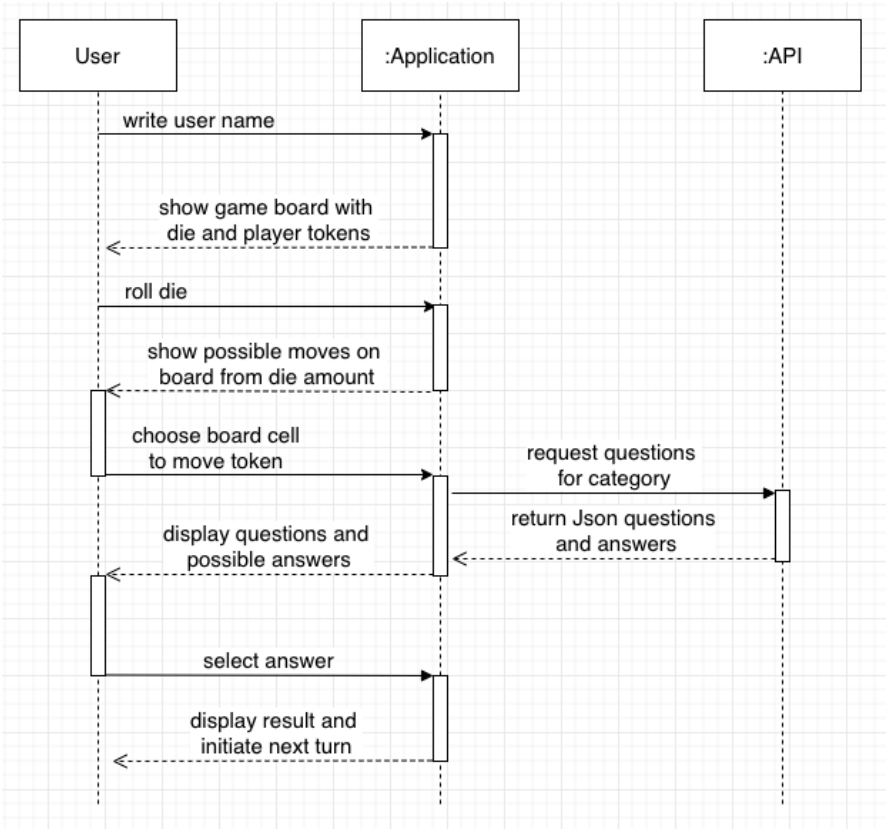
Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

This acceptance criteria test plan shows the functions expected from the app

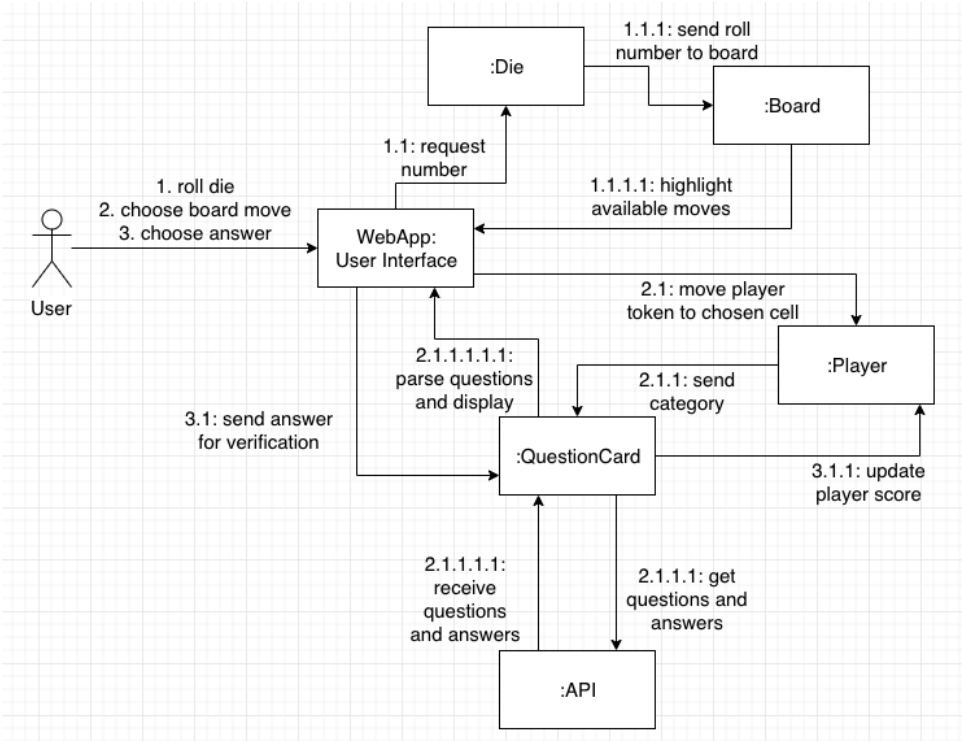
Acceptance criteria	Expected result/output	Pass/Fail
A player must be able to roll the die	A random number between 1 and 6 should be displayed after clicking the roll the die button	Pass
A player must be able to move their token across the board	A highlighted cell on the board should move the player's token when it is clicked	Pass
A player must be able to answer a trivial pursuit question based on the category their token lands on	A trivial pursuit question should be displayed with multiple choice answers that can be clicked individually	Pass
A player should be able to gain a point for every correct answer	A colour corresponding to the category of the correctly answered trivial pursuit question should be added to the player's high score display	Pass
A timer must limit the amount of time a player has to answer questions	A timer should count down to zero and at that point switch control to the next player	Pass

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

A sequence diagram for a user turn when playing the trivial pursuit game. It shows the actions a user takes and

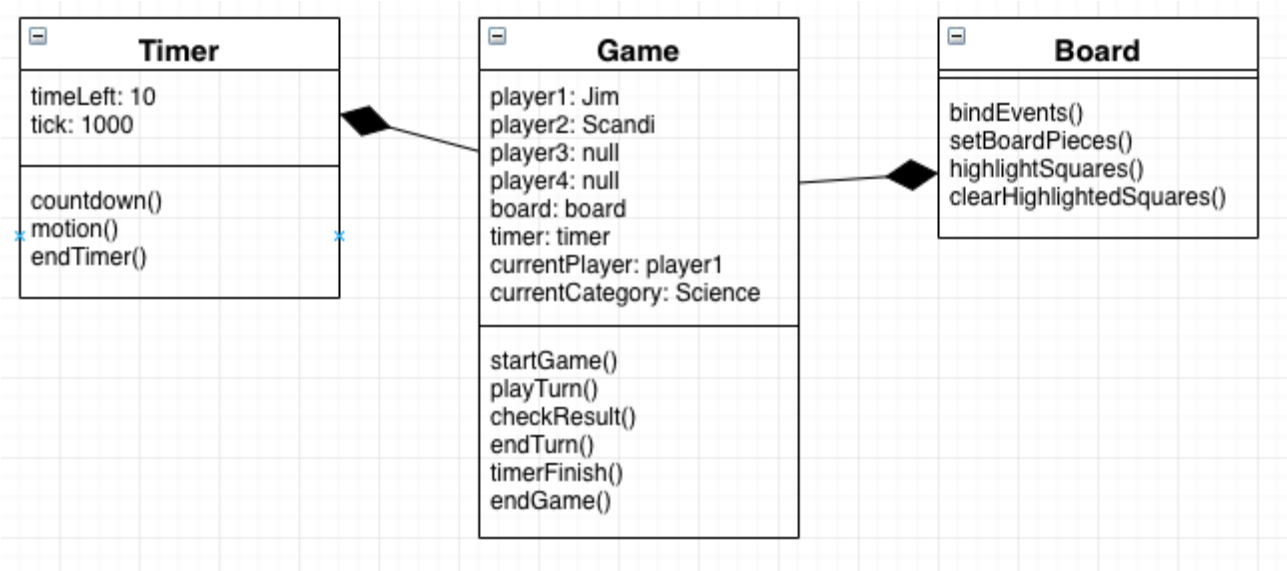


A collaboration diagram for the trivial pursuit game

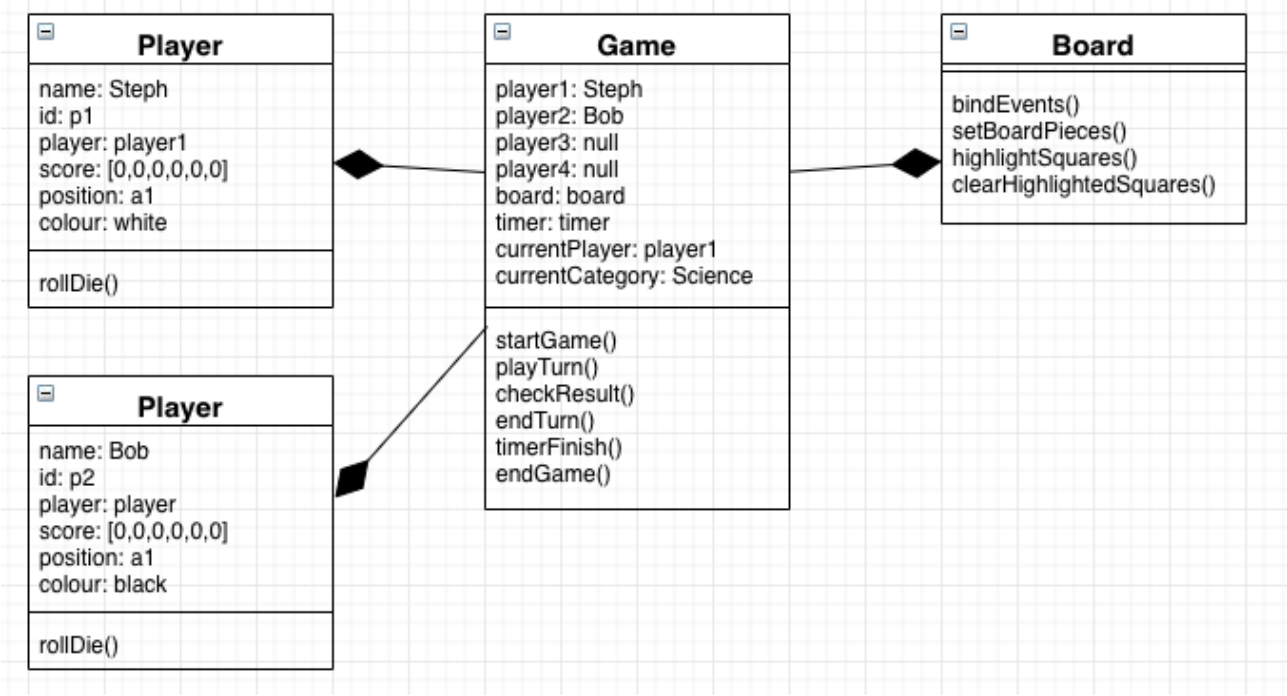


Unit	Ref	Evidence
P	P.8	Produce two object diagrams.
		Description:

An object diagram of the Timer class showing an example of values for time left and the tick interval and its composition relationship to game as well as board.



An object diagram of the Player class showing an example of values for name, id, player, score, position and colour. There are 2 players composed in Game and one Board.



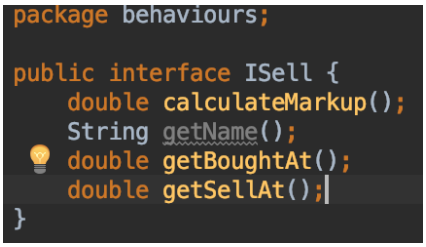
Unit	Ref	Evidence
P	P.17	Produce a bug tracking report
		Description:

Expected result	1st testing attempt		2nd testing attempt
The user must be able to enter their name	Passed		
The timer must start counting down as soon as the game starts	Failed	The callback setInterval did not have a value set after it received instruction from the called function to decrease the time value. A value of 1000 was input after the callback which made the timeLeft value decrease correctly.	Passed
The user must be able to move their token across the board	Passed		
The user must be able to see questions and multiple choice answers displayed in English	Failed	The questions retrieved from the API were being returned with HTML escapes embedded in the Json objects. It was necessary to use an escapes parser and pass the Json data through before displaying to the screen	Passed

Week 12

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

This is an interface ISell, which has several methods which are implemented by any class that implements the interface



```java
package behaviours;

public interface ISell {
    double calculateMarkup();
    String getName();
    double getBoughtAt();
    double getSellAt();
}
```

This is an abstract class Instrument, which implements ISell. Every method listed in the interface is visible here and they are available to every class that inherits from this abstract class.

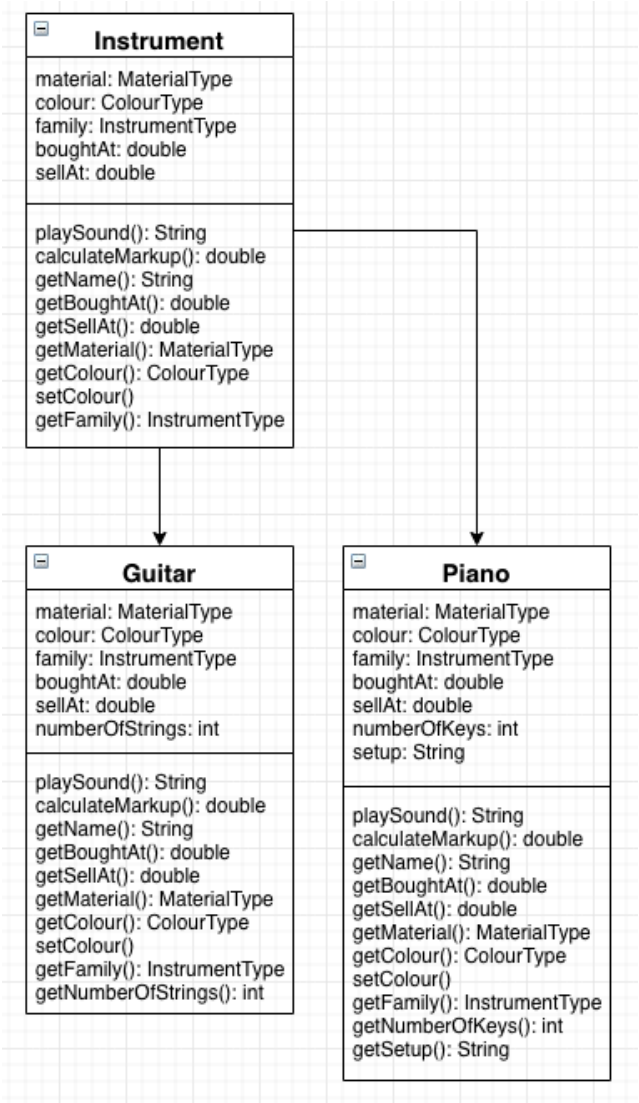
```java
public abstract class Instrument implements IPlay, ISell {

    private MaterialType material;
    private ColourType colour;
    private InstrumentType family;
    private double boughtAt;
    private double sellAt;

    public Instrument(MaterialType material, ColourType colour, InstrumentType family, double boughtAt, double sellAt){
        this.material = material;
        this.colour = colour;
        this.family = family;
        this.boughtAt = boughtAt;
        this.sellAt = sellAt;
    }

    @Override
    public String playSound(){
        return "This " + this.getClass().getSimpleName() +" makes a '" + family.getSound() + "' sound.";
    }

    @Override
    public double calculateMarkup() { return ((sellAt - boughtAt) / boughtAt); }

    @Override
    public String getName() { return this.getClass().getSimpleName(); }

    @Override
    public double getBoughtAt() { return boughtAt; }

    @Override
    public double getSellAt() { return sellAt; }

    public MaterialType getMaterial() { return material; }

    public ColourType getColour() { return colour; }

    public void setColour(ColourType colour) { this.colour = colour; }

    public InstrumentType getFamily() { return family; }
}
```

In the Shop class, all instruments are loaded into an ArrayList as ISell types. This allows all the different classes that inherit from the abstract instrument class to be loaded there with all the methods available to the Shop class. In the sellStockItem() method, the ISell methods getSellAt() and getBoughtAt() are used for each instrument in the stock.

```java
public class Shop {

    private ArrayList<ISell> stock;
    private double cashRegister;
    private double profit;

    public Shop(double cashRegister){
        this.cashRegister = cashRegister;
        this.stock = new ArrayList<>();
        this.profit = 0;
    }

    public double getCashRegister() {
        return cashRegister;
    }

    public double getProfit() { return profit; }

    public ArrayList<ISell> getStock() { return stock; }

    public void addStockItem(ISell stockItem) { this.stock.add(stockItem); }

    public boolean findStockItem(ISell stockItem) { return this.stock.contains(stockItem); }

    public void sellStockItem(ISell stockItem) {
        for (ISell item : stock) {
            if (item == stockItem) {
                this.cashRegister += item.getSellAt();
                this.profit += (item.getSellAt() - item.getBoughtAt());
            }
        }
        this.stock.remove(stockItem);
    }

    public double getPotentialProfit(){
        for (ISell item : stock) {
            this.profit += (item.calculateMarkup() * item.getBoughtAt());
        }
        return this.profit;
    }
}
```

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

This is an inheritance diagram showing an abstract class Instrument and two subclasses Guitar and Piano that inherit all methods and variables from the abstract class



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

This is an example of encapsulation in a program. The abstract class Accessory has several data variables which are all private. They are not accessible outside of the class except through the explicit getters that have been created eg: getSellAt. Additionally this is an example of another form of encapsulation wherein data and methods are bundled into a class.

```
package accessories;

import ...

public abstract class Accessory implements ISell {

    private InstrumentType family;
    private double boughtAt;
    private double sellAt;

    public Accessory(InstrumentType family, double boughtAt, double sellAt){
        this.family = family;
        this.boughtAt = boughtAt;
        this.sellAt = sellAt;
    }

    @Override
    public double calculateMarkup() { return ((sellAt - boughtAt) / boughtAt); }

    @Override
    public String getName() { return this.getClass().getSimpleName(); }

    @Override
    public double getBoughtAt() { return boughtAt; }

    @Override
    public double getSellAt() { return sellAt; }

    public InstrumentType getFamily() { return family; }
}
```

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:

This is an abstract class that has several variables which are private. There are methods to access these variables like getSellAt() and other methods that return a string or the arithmetic result of two variables.

```
public abstract class Instrument implements IPlay, ISell {

    private MaterialType material;
    private ColourType colour;
    private InstrumentType family;
    private double boughtAt;
    private double sellAt;

    public Instrument(MaterialType material, ColourType colour, InstrumentType family, double boughtAt, double sellAt){
        this.material = material;
        this.colour = colour;
        this.family = family;
        this.boughtAt = boughtAt;
        this.sellAt = sellAt;
    }

    @Override
    public String playSound(){
        return "This " + this.getClass().getSimpleName() +" makes a '" + family.getSound() + "' sound.";
    }

    @Override
    public double calculateMarkup() { return ((sellAt - boughtAt) / boughtAt); }

    @Override
    public String getName() { return this.getClass().getSimpleName(); }

    @Override
    public double getBoughtAt() { return boughtAt; }

    @Override
    public double getSellAt() { return sellAt; }

    public MaterialType getMaterial() { return material; }

    public ColourType getColour() { return colour; }

    public void setColour(ColourType colour) { this.colour = colour; }

    public InstrumentType getFamily() { return family; }
}
```

This is a subclass that inherits all the variables and methods from the abstract class above. It also adds a new variable called range along with a getRange method.

```
package instruments;

import behaviours.IPlay;
import behaviours.ISell;

public class Marimba extends Instrument implements IPlay, ISell {

    private String range;

    public Marimba(MaterialType material, ColourType colour, InstrumentType family, double boughtAt, double sellAt,
                   String range){
        super(material, colour, family, boughtAt, sellAt);
        this.range = range;
    }

    public String getRange() { return range; }
}
```

This is an example of an object instance of the Marimba class, called marimba. It has several parameters being assigned, wood for the material type, indigo for the colour type, percussion for the instrument type, 5099.00 for the bought at price, 6000.00 for the sell at price and contra-bass for the range.

```
import instruments.ColourType;
import instruments.InstrumentType;
import instruments.Marimba;
import instruments.MaterialType;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class MarimbaTest {

    Marimba marimba;

    @Before
    public void before() {
        marimba = new Marimba(MaterialType.WOOD, ColourType.INDIGO, InstrumentType.PERCUSSION,  boughtAt: 5099.00,
  sellAt: 6000.00,  range: "Contra-Bass");
    }
```

The method getFamily is being used on the marimba object instance to access the variable family in the instrument abstract class and inherited by the marimba.

```
    @Test
    public void hasType() { assertEquals(MaterialType.WOOD, marimba.getMaterial()); }

    @Test
    public void hasColour() { assertEquals(ColourType.INDIGO, marimba.getColour()); }

    @Test
    public void hasFamily() { assertEquals(InstrumentType.PERCUSSION, marimba.getFamily()); }

    @Test
    public void hasBoughtValue() { assertEquals( expected: 4099.00, marimba.getBoughtAt(),  delta: 0.01); }

    @Test
    public void hasSellValue() { assertEquals( expected: 6000.00, marimba.getSellAt(),  delta: 0.01); }

    @Test
    public void hasRange() { assertEquals( expected: "Contra-Bass", marimba.getRange()); }

    @Test
    public void canCalculateMarkup() { assertEquals( expected: 0.18, marimba.calculateMarkup(),  delta: 0.01); }

    @Test
    public void canPlaySound() { assertEquals( expected: "This Marimba makes a 'Clack' sound.", marimba.playSound()); }
}
```

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:

This algorithm is used to sell a stock item. It takes in a parameter (the stock item to be sold) and then searches through an ArrayList of stock items using a for loop and an if statement. Once the stock item is found, the getSellAt method returns the price and adds it to the cashRegister variable. After this, the profit is calculated as the difference of the sale price and the purchase price.

```
public void sellStockItem(ISell stockItem) {
    for (ISell item : stock) {
        if (item == stockItem) {
            this.cashRegister += item.getSellAt();
            this.profit += (item.getSellAt() - item.getBoughtAt());
        }
    }
    this.stock.remove(stockItem);
}
```

This algorithm is used to display song details in list. It takes in an array of charts which then is mapped into a new array and each song is then passed into a Song class with every attribute sent as React params to drive the rendering inside of the Song class.

```
const TopChartList = ({charts}) => {

    const chartItems = charts.map (data =>
        <Song key={data.id.attributes['im:id']} title={data.title.label} artist={data['im:artist']
        .label} album={data['im:collection']['im:name'].label} preview={data.link[1]} picture={data
        ['im:image'][2]}></Song>
        )

    return(
        <div className="top-charts-list">
            {chartItems}
        </div>
    );

}

export default TopChartList;
```