CodeClan_
Bridging the digital skills gap.

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.

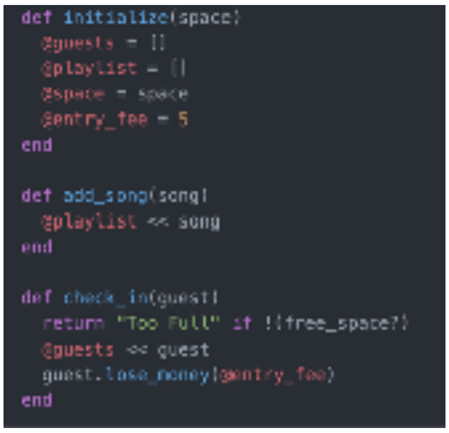
Please fill in each point with screenshot or diagram and description of what you are showing.

Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

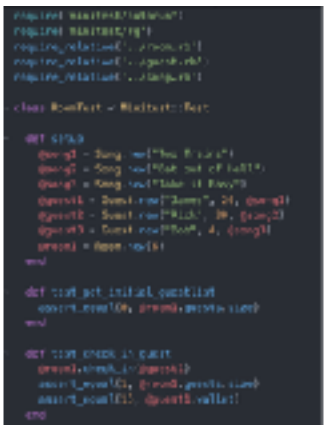
<u>Week 2</u>

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:

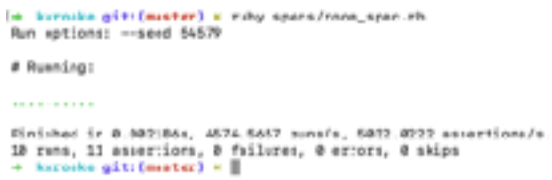
<u>Paste Screenshot here</u>



```ruby
def initialize(space)
  @guests = []
  @playlist = []
  @space = space
  @entry_fee = 5
end

def add_song(song)
  @playlist << song
end

def check_in(guest)
  return "Too Full" if !(free_space?)
  @guests << guest
  guest.lose_money(@entry_fee)
end
```



```ruby
require('minitest/autorun')
require('minitest/rg')
require_relative('../room.rb')
require_relative('../guest.rb')
require_relative('../song.rb')

class RoomTest < MiniTest::Test

  def setup
    @song1 = Song.new([illegible])
    @song2 = Song.new([illegible])
    @song3 = Song.new([illegible])
    @guest1 = Guest.new([illegible], 20, @song1)
    @guest2 = Guest.new([illegible], 30, @song2)
    @guest3 = Guest.new([illegible], 4, @song3)
    @room1 = Room.new(5)
  end

  def test_get_initial_guestlist
    assert_equal(0, @room1.guests.size)
  end

  def test_check_in_guest
    @room1.check_in(@guest1)
    assert_equal(1, @room1.guests.size)
    assert_equal(15, @guest1.wallet)
  end
```



```
→ karaoke git:(master) ✗ ruby specs/room_spec.rb
Run options: --seed 54579

# Running:

..........

Finished in [illegible]
10 runs, 11 assertions, 0 failures, 0 errors, 0 skips
→ karaoke git:(master) ✗
```

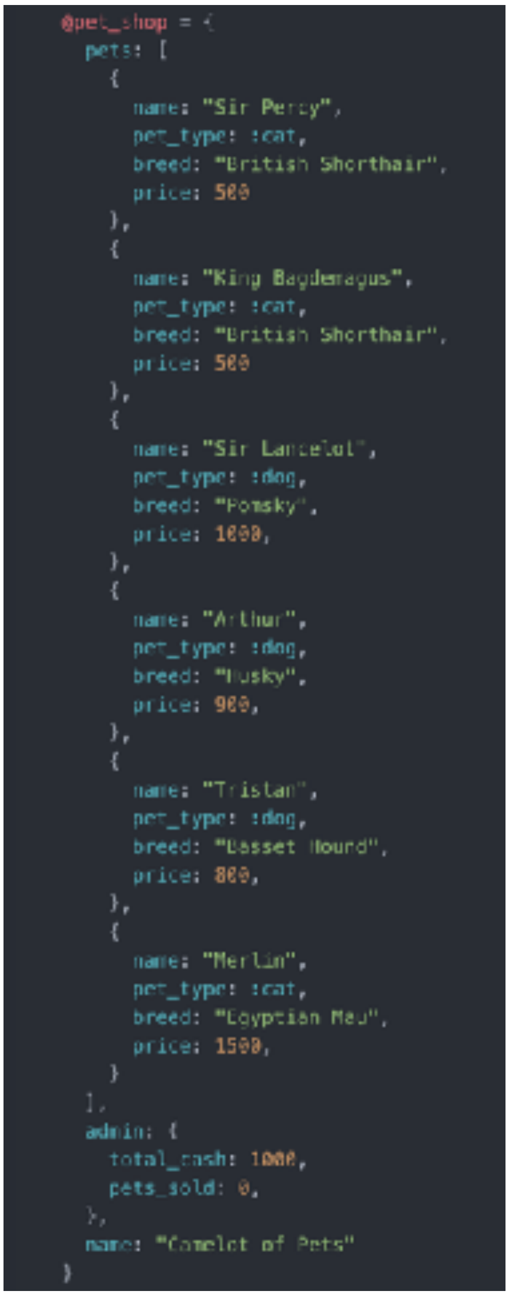
<u>Description here</u>

From the first screenshot, this program creates an array called @guests. The function check_in(guest) uses the guests array, by adding a guest object to the array. The function is called in the test code in the second screenshot, in the test_check_in_guest function. The third screenshot shows the result of running that test code.

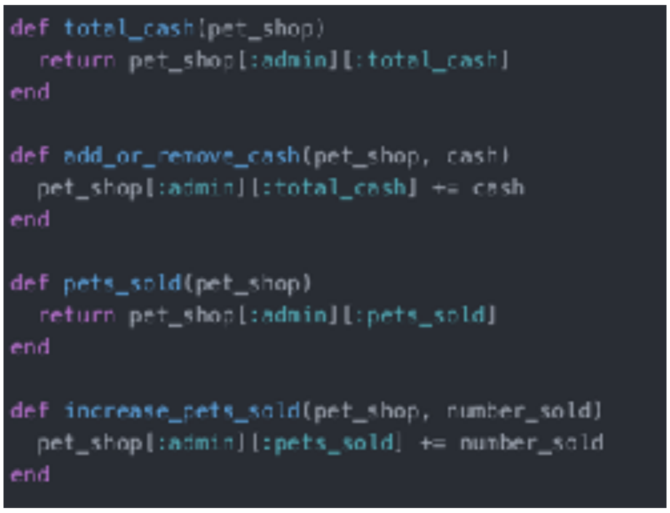
Unit	Ref	Evidence
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running

Unit	Ref	Evidence
		Description:

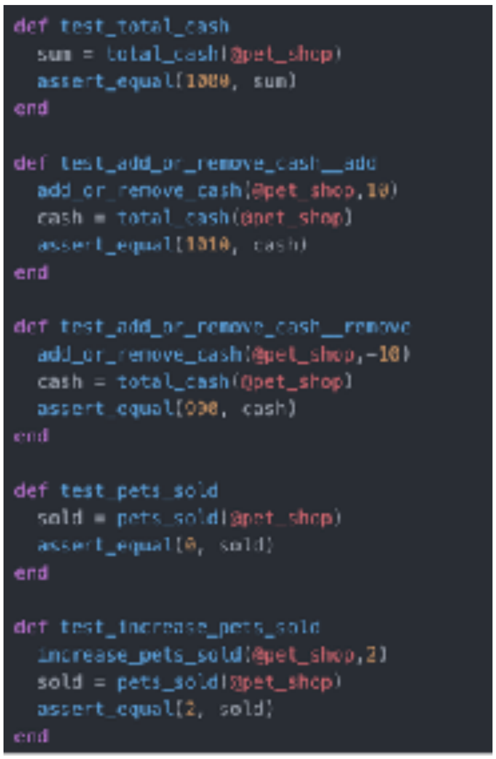
<u>Paste Screenshot here</u>



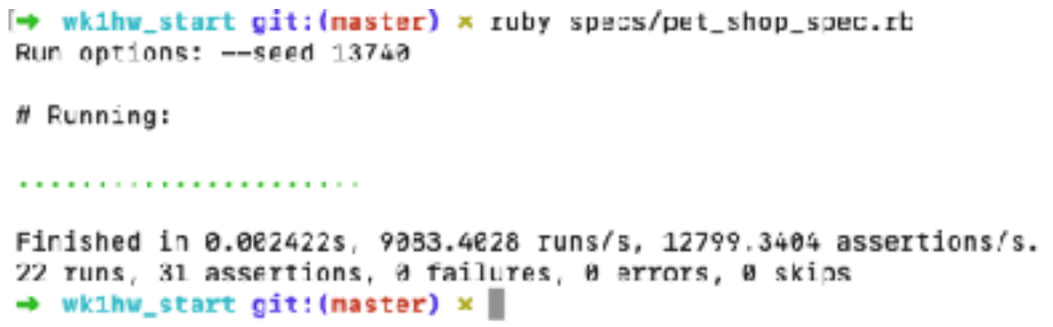
```ruby
@pet_shop = {
  pets: [
    {
      name: "Sir Percy",
      pet_type: :cat,
      breed: "British Shorthair",
      price: 500
    },
    {
      name: "King Bagdemagus",
      pet_type: :cat,
      breed: "British Shorthair",
      price: 500
    },
    {
      name: "Sir Lancelot",
      pet_type: :dog,
      breed: "Pomsky",
      price: 1000,
    },
    {
      name: "Arthur",
      pet_type: :dog,
      breed: "Husky",
      price: 900,
    },
    {
      name: "Tristan",
      pet_type: :dog,
      breed: "Basset Hound",
      price: 800,
    },
    {
      name: "Merlin",
      pet_type: :cat,
      breed: "Egyptian Mau",
      price: 1500,
    }
  ],
  admin: {
    total_cash: 1000,
    pets_sold: 0,
  },
  name: "Camelot of Pets"
}
```



```ruby
def total_cash(pet_shop)
  return pet_shop[:admin][:total_cash]
end

def add_or_remove_cash(pet_shop, cash)
  pet_shop[:admin][:total_cash] += cash
end

def pets_sold(pet_shop)
  return pet_shop[:admin][:pets_sold]
end

def increase_pets_sold(pet_shop, number_sold)
  pet_shop[:admin][:pets_sold] += number_sold
end
```



```ruby
def test_total_cash
  sum = total_cash(@pet_shop)
  assert_equal(1000, sum)
end

def test_add_or_remove_cash__add
  add_or_remove_cash(@pet_shop,10)
  cash = total_cash(@pet_shop)
  assert_equal(1010, cash)
end

def test_add_or_remove_cash__remove
  add_or_remove_cash(@pet_shop,-10)
  cash = total_cash(@pet_shop)
  assert_equal(990, cash)
end

def test_pets_sold
  sold = pets_sold(@pet_shop)
  assert_equal(0, sold)
end

def test_increase_pets_sold
  increase_pets_sold(@pet_shop,2)
  sold = pets_sold(@pet_shop)
  assert_equal(2, sold)
end
```



```
[➜  wk1hw_start git:(master) ✗ ruby specs/pet_shop_spec.rb
Run options: --seed 13740

# Running:

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

Finished in 0.002422s, 9083.4028 runs/s, 12799.3404 assertions/s.
22 runs, 31 assertions, 0 failures, 0 errors, 0 skips
➜  wk1hw_start git:(master) ✗ ▌
```

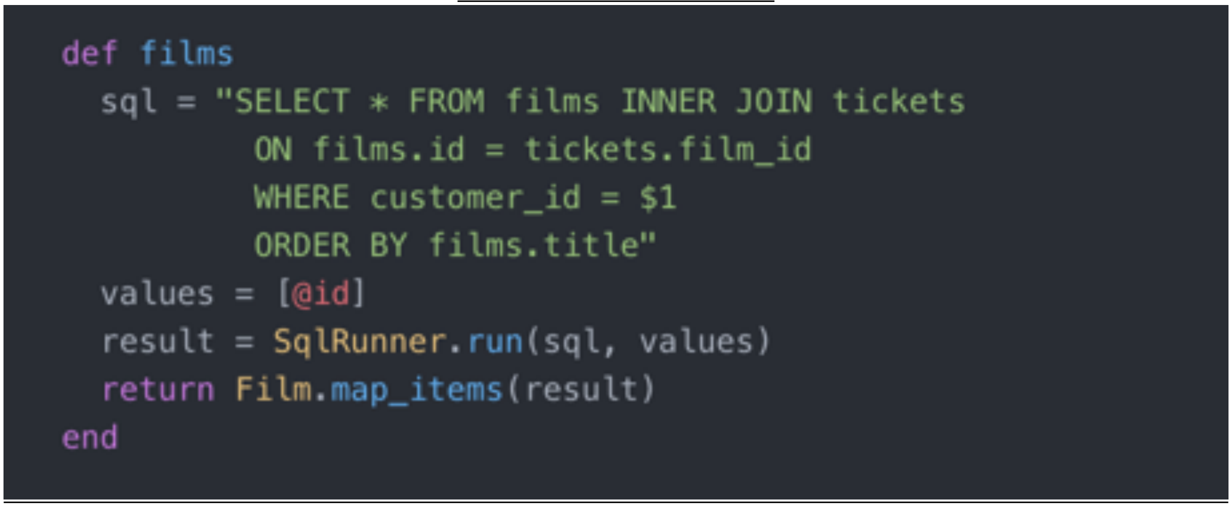
Description here

The first screenshot demonstrates the @pet_shop hash being initiated. The second screenshot shows four functions that use the pet_shop hash, by either finding a value and returning it, or finding a value and mutating it. The third screenshot has five functions in a test file, using the previous four functions. The final screenshot is the result of running that test file.

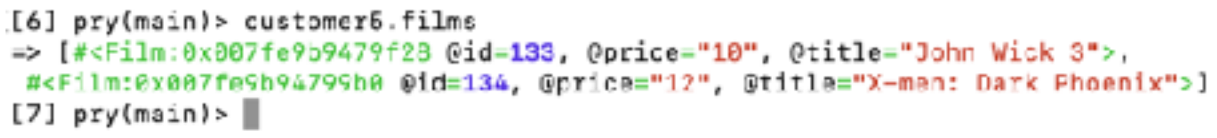
Week 3

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:

Paste Screenshot here



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



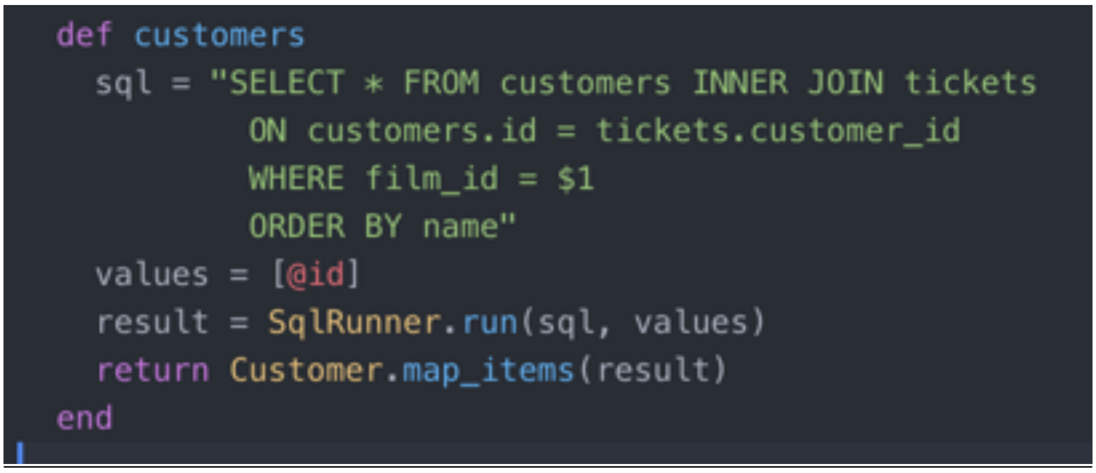
```
[6] pry(main)> customer6.films
=> [#<Film:0x007fe9b9479f28 @id=133, @price="10", @title="John Wick 3">,
 #<Film:0x007fe9b94799b0 @id=134, @price="12", @title="X-men: Dark Phoenix">]
[7] pry(main)> 
```

Description here

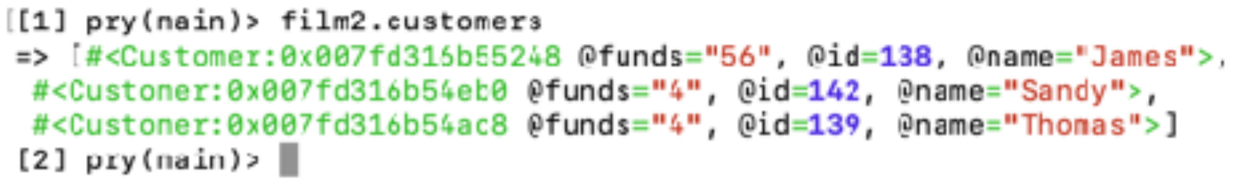
The first screenshot shows a method films, that contains an SQL statement which is used to talk to a PSQL database. The SQL statement looks for all films that a specific customer has tickets for. The second screenshot shows the result of calling that method on a customer, returning two films.

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:

Paste Screenshot here



```
def customers
  sql = "SELECT * FROM customers INNER JOIN tickets
          ON customers.id = tickets.customer_id
          WHERE film_id = $1
          ORDER BY name"
  values = [@id]
  result = SqlRunner.run(sql, values)
  return Customer.map_items(result)
end
```



```
[1] pry(main)> film2.customers
 => [#<Customer:0x007fd315b55248 @funds="56", @id=138, @name="James">,
  #<Customer:0x007fd316b54eb0 @funds="4", @id=142, @name="Sandy">,
  #<Customer:0x007fd316b54ac8 @funds="4", @id=139, @name="Thomas">]
 [2] pry(main)>
```

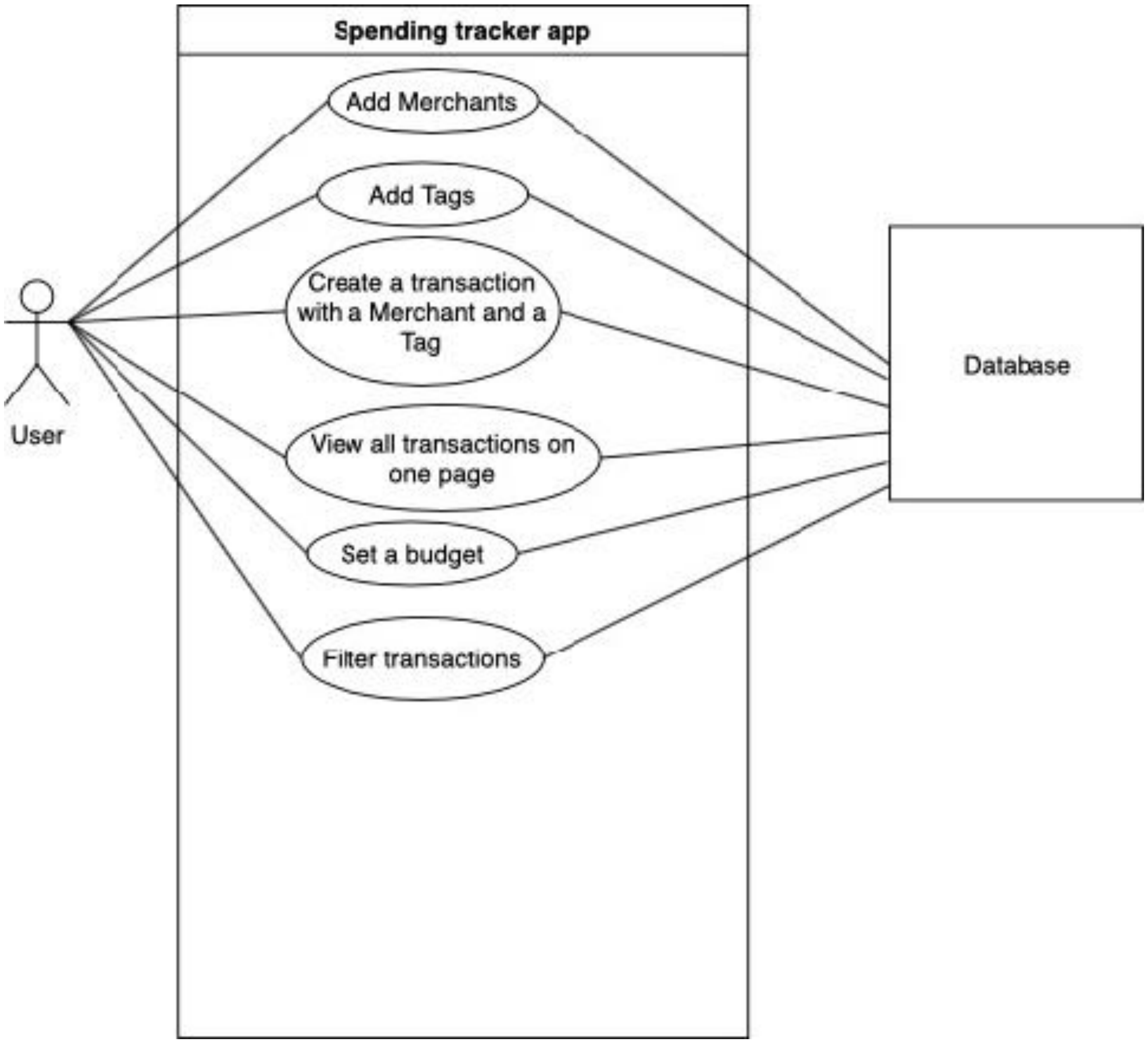
Description here

The first screenshot shows a method customers, that contains an SQL statement which is used to talk to a PSQL database. The SQL statement looks for all customers who have a ticket to a given film, and then sorts them by their name. The second screenshot shows the result of calling that method on a film, returning three sorted customers.

Week 5 and 6

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

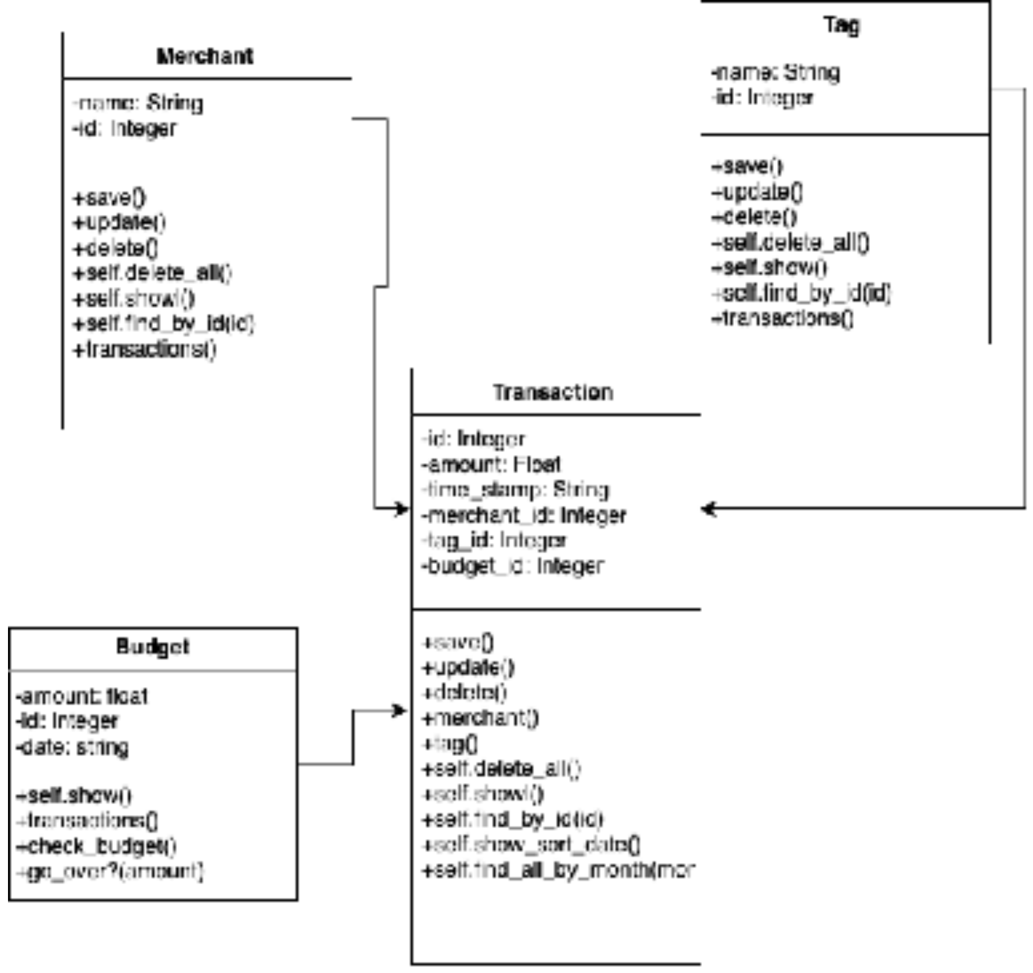
<u>Paste Screenshot here</u>



<u>Description here</u>

The Use Case diagram above, based on my solo ruby project, shows a User, interacting with the Spending Tracker App, which returns information from the database for the User to view. In the center are the use cases.

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

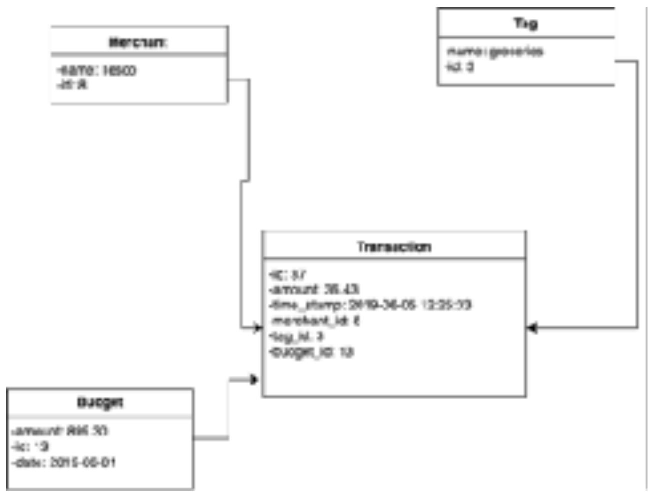


Paste Screenshot here

Description here

Above is a Class diagram for my solo ruby project which depicts four classes: Merchant, Tag, Budget and Transaction. The arrows point towards the class which holds instances of the class the arrow points away from. Each Class has a list of variables and their types, as well as a list of methods.

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



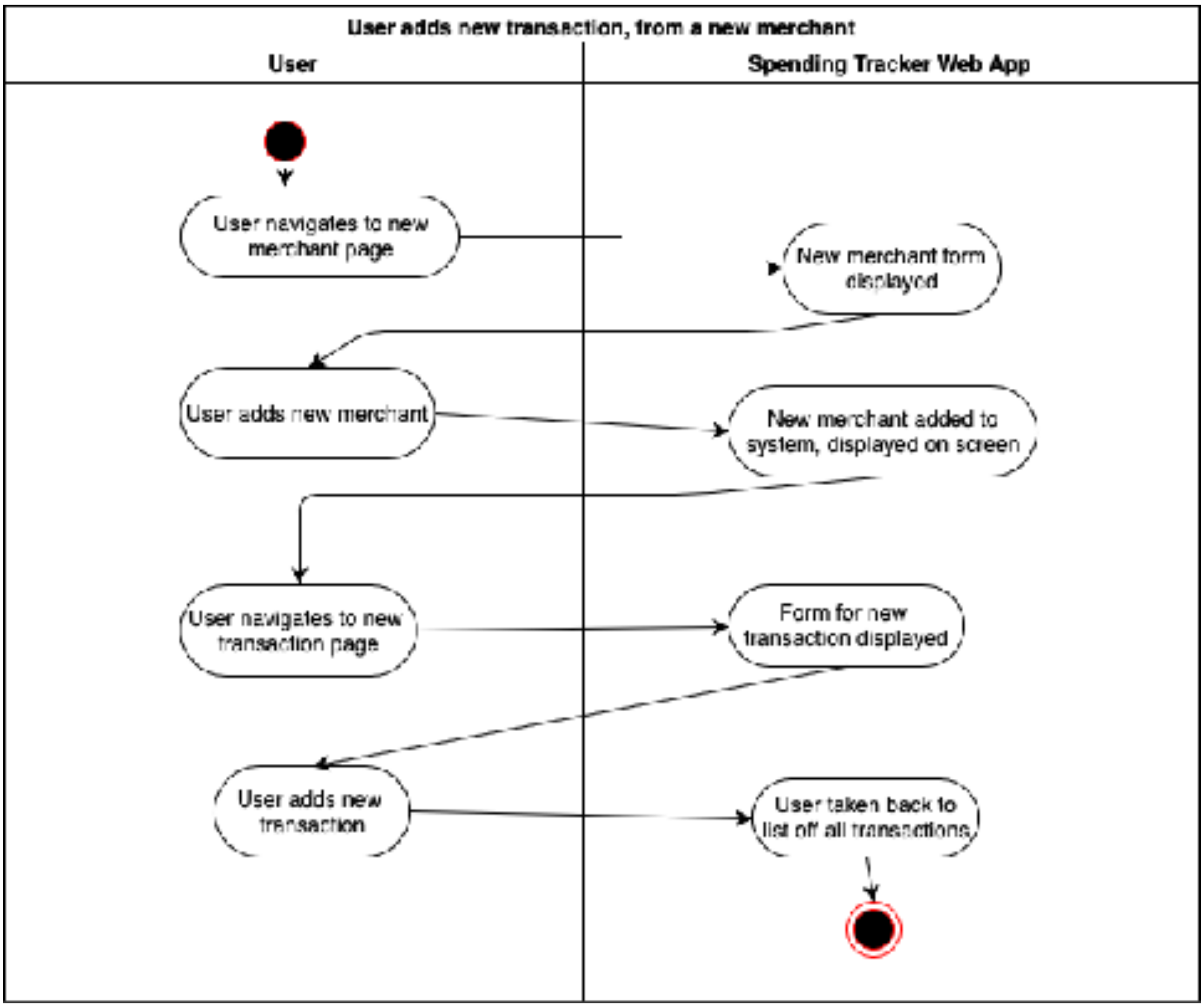
Paste Screenshot here

Description here

Above is an Object diagram for my solo ruby project, which depicts four instances of classes: Merchant, Tag, Budget and Transaction. The arrows point towards the object which contains the object the arrow points away from. Each object has a list of variables, and example data that would be held in that variable.

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

Paste Screenshot here

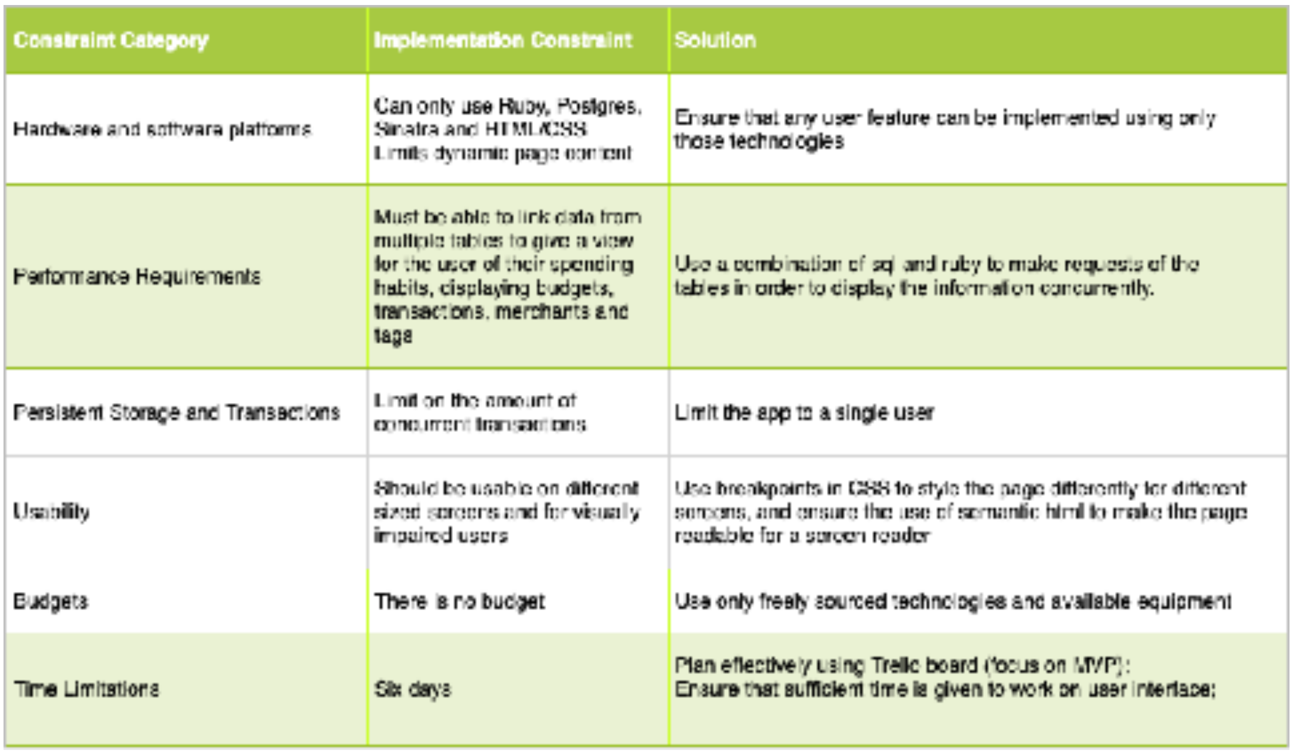

User adds new transaction, from a new merchant
User
Spending Tracker Web App
User navigates to new merchant page
New merchant form displayed
User adds new merchant
New merchant added to system, displayed on screen
User navigates to new transaction page
Form for new transaction displayed
User adds new transaction
User taken back to list off all transactions

Description here

Above is an activity diagram for my solo ruby project, which depicts the steps a user and the app must take to add a new transaction using a new merchant.

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:

Paste Screenshot here



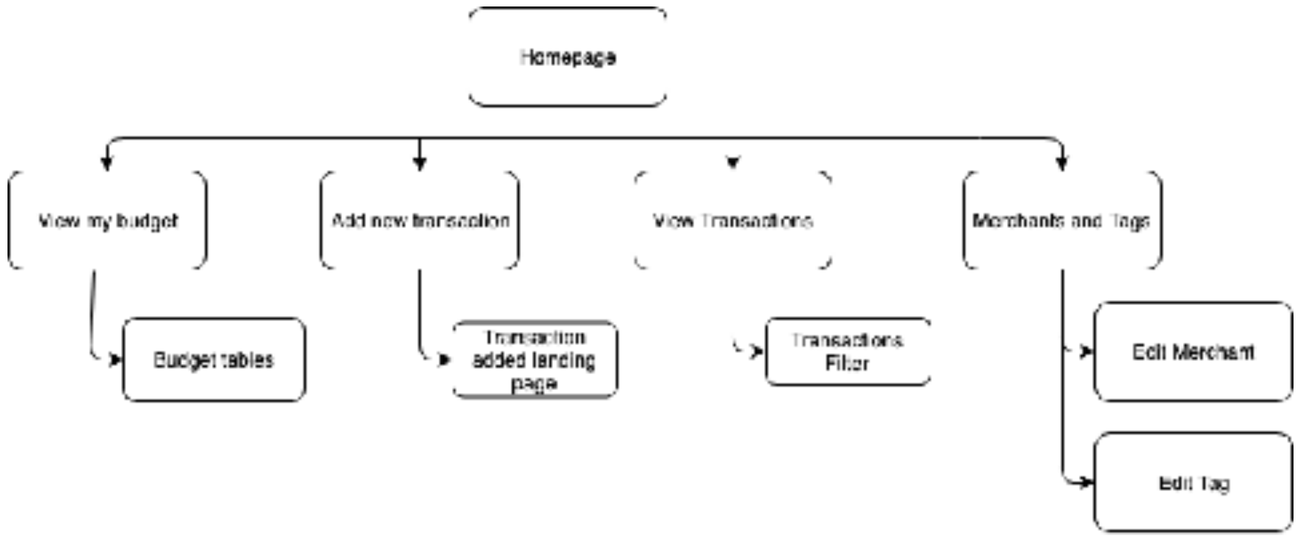
Constraint Category	Implementation Constraint	Solution
Hardware and software platforms	Can only use Ruby, Postgres, Sinatra and HTML/CSS Limits dynamic page content	Ensure that any user feature can be implemented using only those technologies
Performance Requirements	Must be able to link data from multiple tables to give a view for the user of their spending habits, displaying budgets, transactions, merchants and tags	Use a combination of sql and ruby to make requests of the tables in order to display the information concurrently.
Persistent Storage and Transactions	Limit on the amount of concurrent transactions	Limit the app to a single user
Usability	Should be usable on different sized screens and for visually impaired users	Use breakpoints in CSS to style the page differently for different screens, and ensure the use of semantic html to make the page readable for a screen reader
Budgets	There is no budget	Use only freely sourced technologies and available equipment
Time Limitations	Six days	Plan effectively using Trello board (focus on MVP); Ensure that sufficient time is given to work on user interface;

Description here

Above is an implementation constraints plan for my solo ruby project.

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

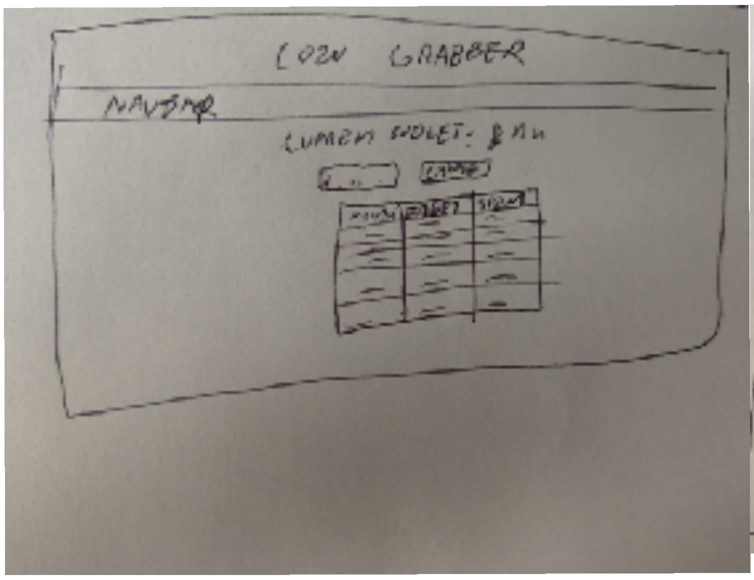
Paste Screenshot here

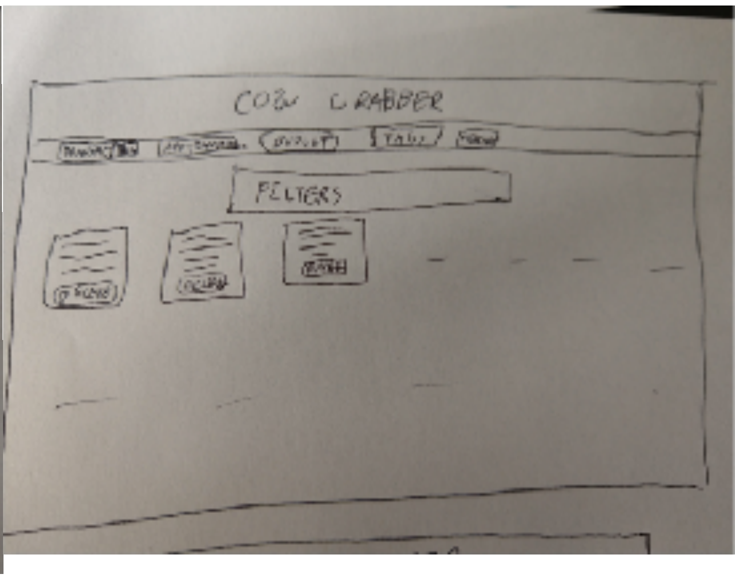


Description here

Above is a planned user site map for my solo ruby project, with arrows pointing towards potential routes within the web app.

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

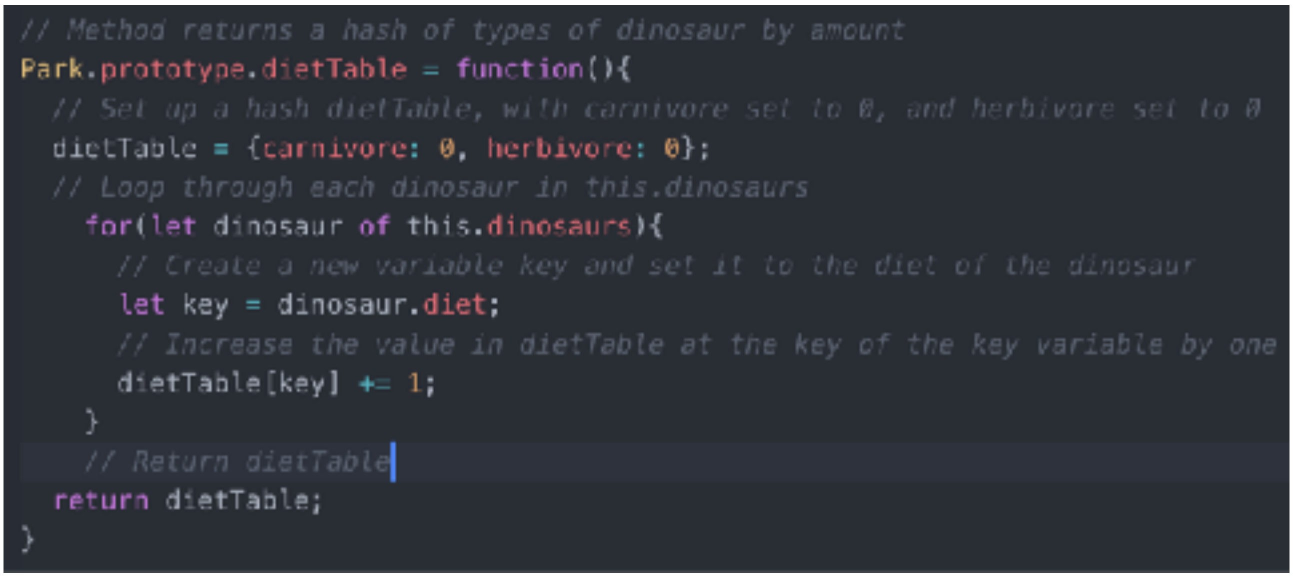




Paste Screenshot here
Description here
Two wireframes for my solo project, depicting the Budget page and the Transactions page.

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

Paste Screenshot here

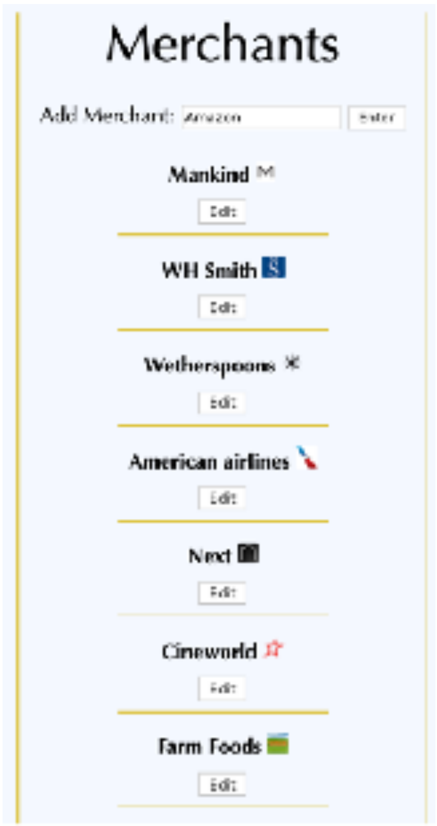


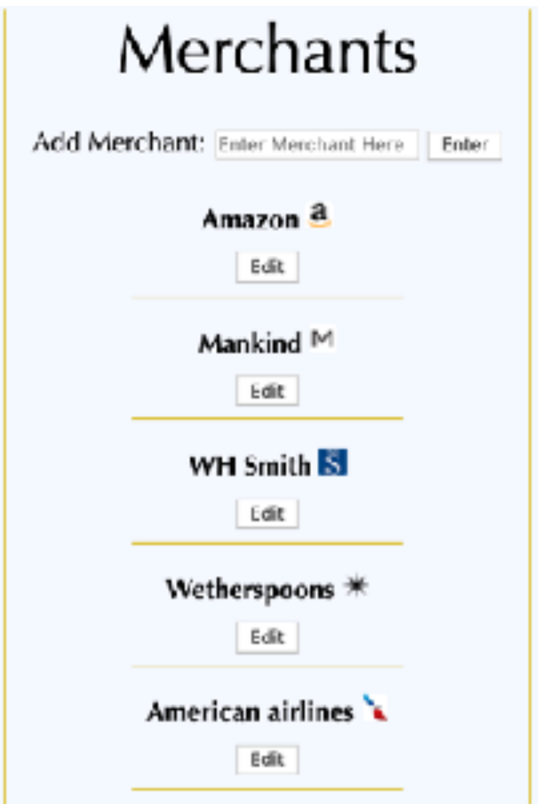
```
// Method returns a hash of types of dinosaur by amount
Park.prototype.dietTable = function(){
  // Set up a hash dietTable, with carnivore set to 0, and herbivore set to 0
  dietTable = {carnivore: 0, herbivore: 0};
  // Loop through each dinosaur in this.dinosaurs
    for(let dinosaur of this.dinosaurs){
      // Create a new variable key and set it to the diet of the dinosaur
      let key = dinosaur.diet;
      // Increase the value in dietTable at the key of the key variable by one
      dietTable[key] += 1;
    }
    // Return dietTable
  return dietTable;
}
```

Description here

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:

Paste Screenshot here



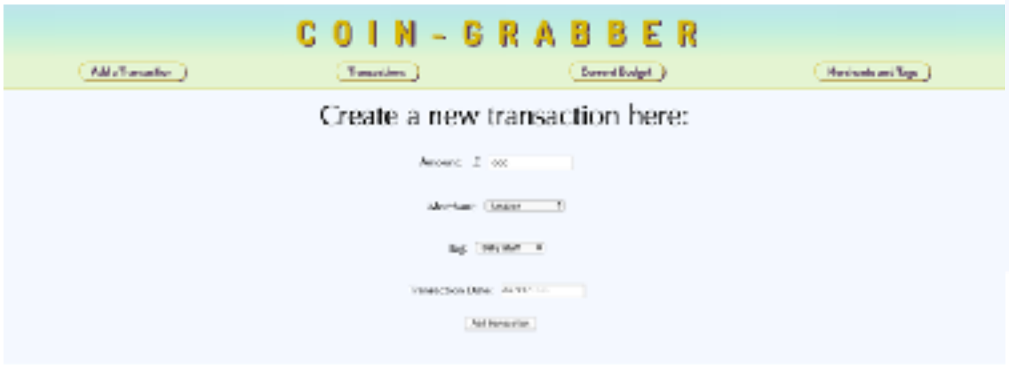


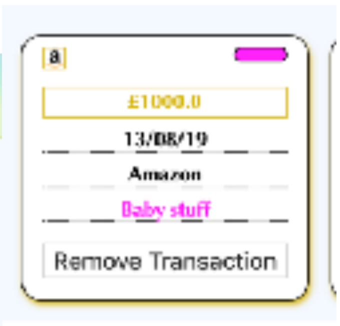
Description here

User enters Amazon as a merchant. Upon pressing enter, the input is added to the page, along with a favicon that was acquired using the inputted string.

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:

Paste Screenshot here

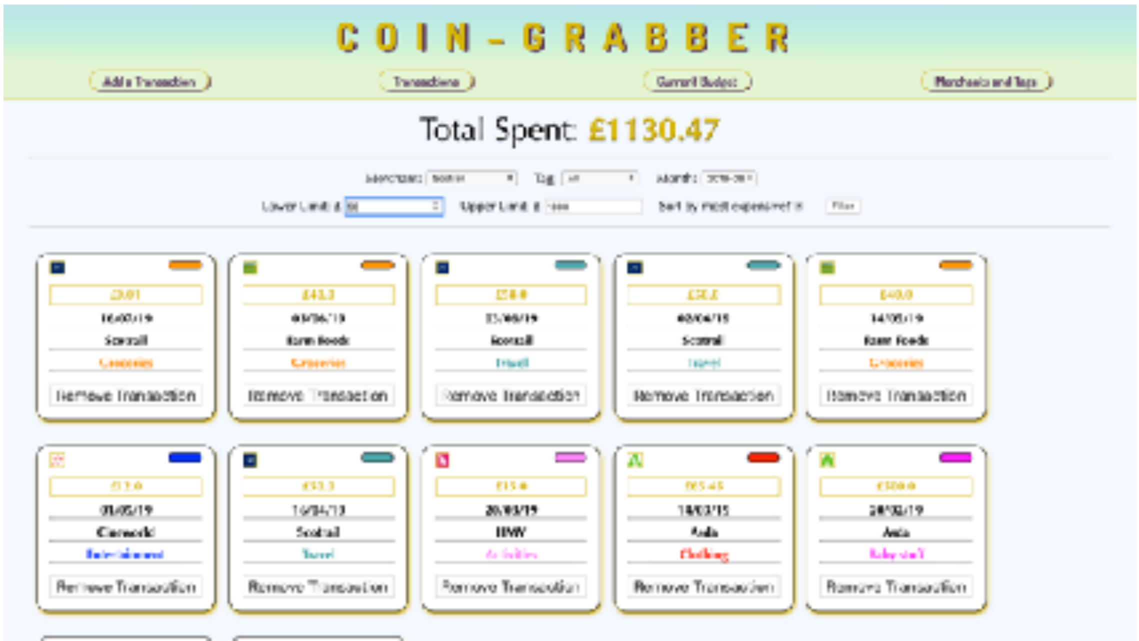


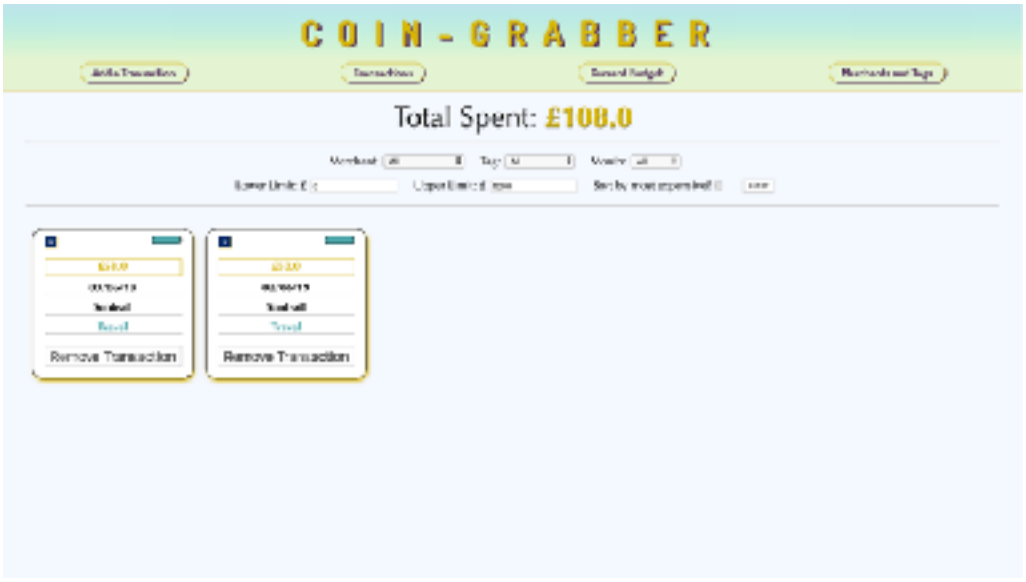


Description here

Data is entered into the add transaction page, and appears on the transactions page in the form seen to the left, being saved into the program.

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:





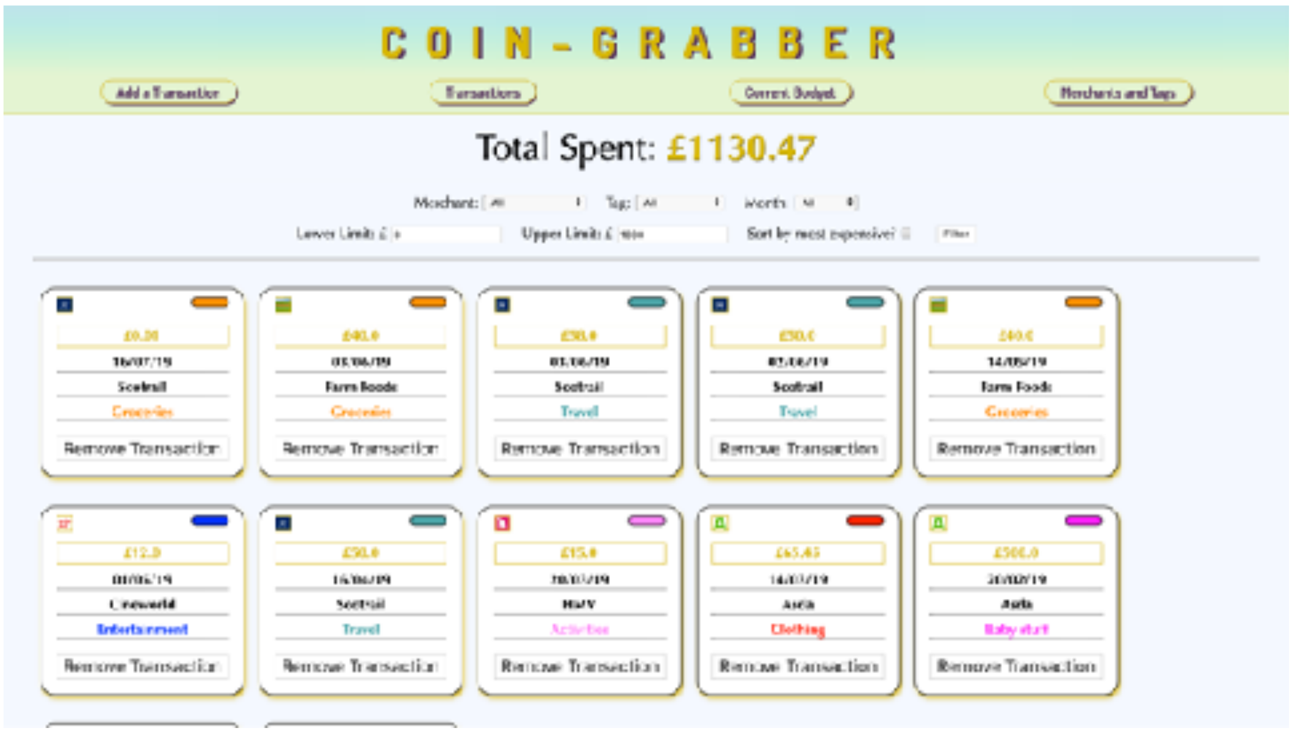
Paste Screenshot here

Description here

On the transactions page, the user selects different options to filter, and clicks the filter button. The request is processed and the transactions on the page are filtered down.

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:

Paste Screenshot here

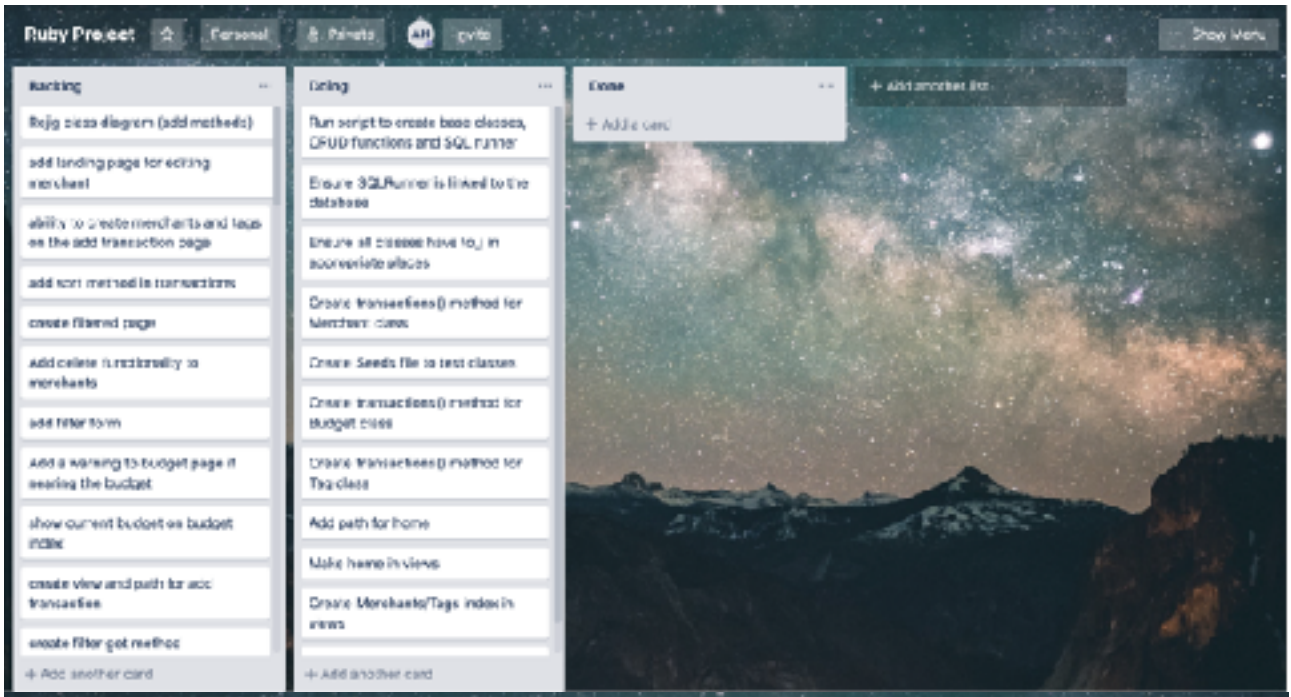


Description here

https://github.com/Adamsh6/ruby_spending_tracker

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

Paste Screenshot here



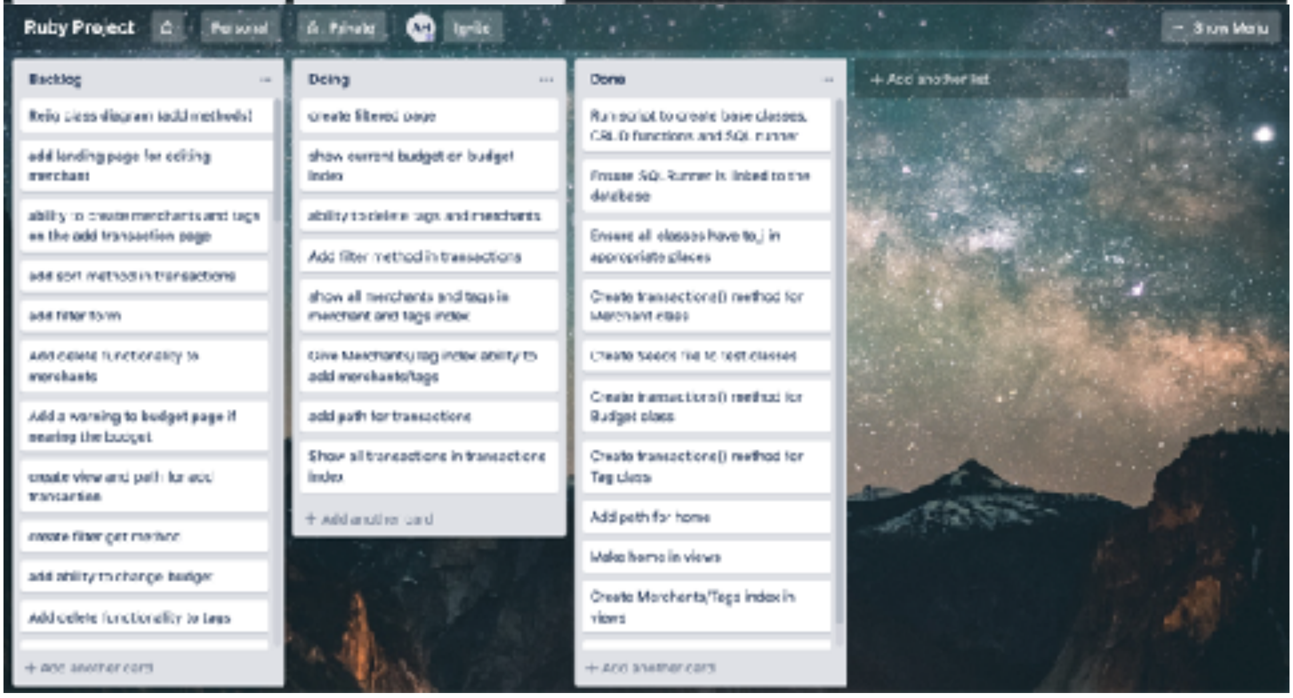
Ruby Project · Personal · Private · Invite · Show Menu

Backlog
- Redo class diagram (add methods)
- add landing page for editing merchant
- ability to create merchants and tags on the add transaction page
- add sort method in transactions
- create filtered page
- Add delete functionality to merchants
- add filter form
- Add a warning to budget page if nearing the budget
- show current budget on budget index
- create view and path for add transaction
- create filter get method
- + Add another card

Doing
- Run script to create base classes, CRUD functions and SQL runner
- Ensure SQLRunner is linked to the database
- Ensure all classes have to_i in appropriate places
- Create transactions() method for Merchant class
- Create Seeds file to test classes
- Create transactions() method for Budget class
- Create transactions() method for Tag class
- Add path for home
- Make home in views
- Create Merchants/Tags index in views
- + Add another card

Done
- + Add a card

+ Add another list



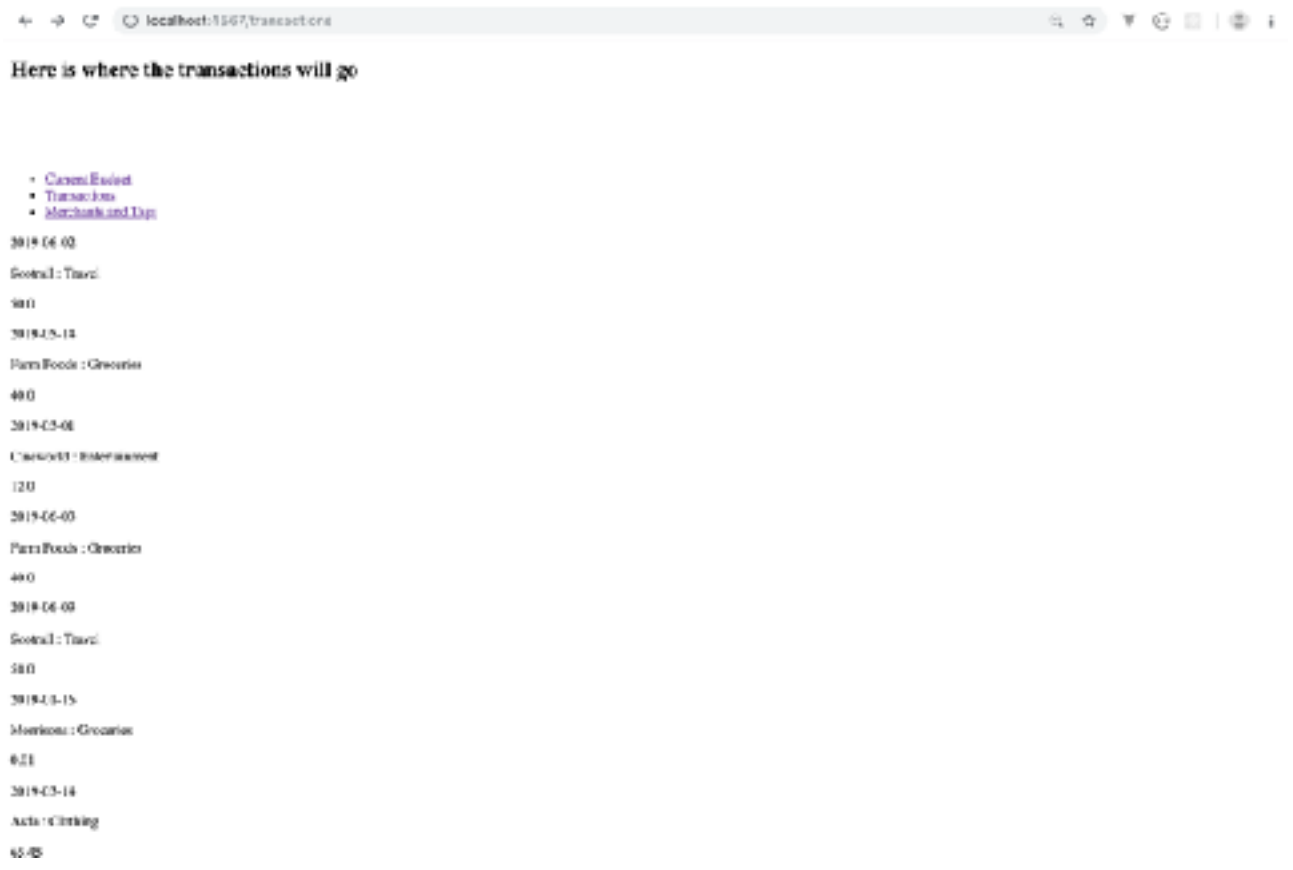
Ruby Project · Personal · Private · Invite · Show Menu

Backlog
- Redo class diagram (add methods)
- add landing page for editing merchant
- ability to create merchants and tags on the add transaction page
- add sort method in transactions
- add filter form
- Add delete functionality to merchants
- Add a warning to budget page if nearing the budget
- create view and path for add transaction
- create filter get method
- add ability to change budget
- Add delete functionality to tags
- + Add another card

Doing
- create filtered page
- show current budget on budget index
- ability to delete tags and merchants
- Add filter method in transactions
- show all merchants and tags in merchant and tags index
- Give Merchants/Tag index ability to add merchants/tags
- add path for transactions
- Show all transactions in transactions index
- + Add another card

Done
- Run script to create base classes, CRUD functions and SQL runner
- Ensure SQLRunner is linked to the database
- Ensure all classes have to_i in appropriate places
- Create transactions() method for Merchant class
- Create Seeds file to test classes
- Create transactions() method for Budget class
- Create transactions() method for Tag class
- Add path for home
- Make home in views
- Create Merchants/Tags index in views
- + Add another card

+ Add another list



localhost:1567/transactions

Here is where the transactions will go

- Current Budget
- Transactions
- Merchants and Tags

2019-06-02

Scotrail : Travel

50.0

2019-05-14

Farm Foods : Groceries

40.0

2019-03-08

Cineworld : Entertainment

12.0

2019-06-03

Farm Foods : Groceries

40.0

2019-06-03

Scotrail : Travel

50.0

2019-03-15

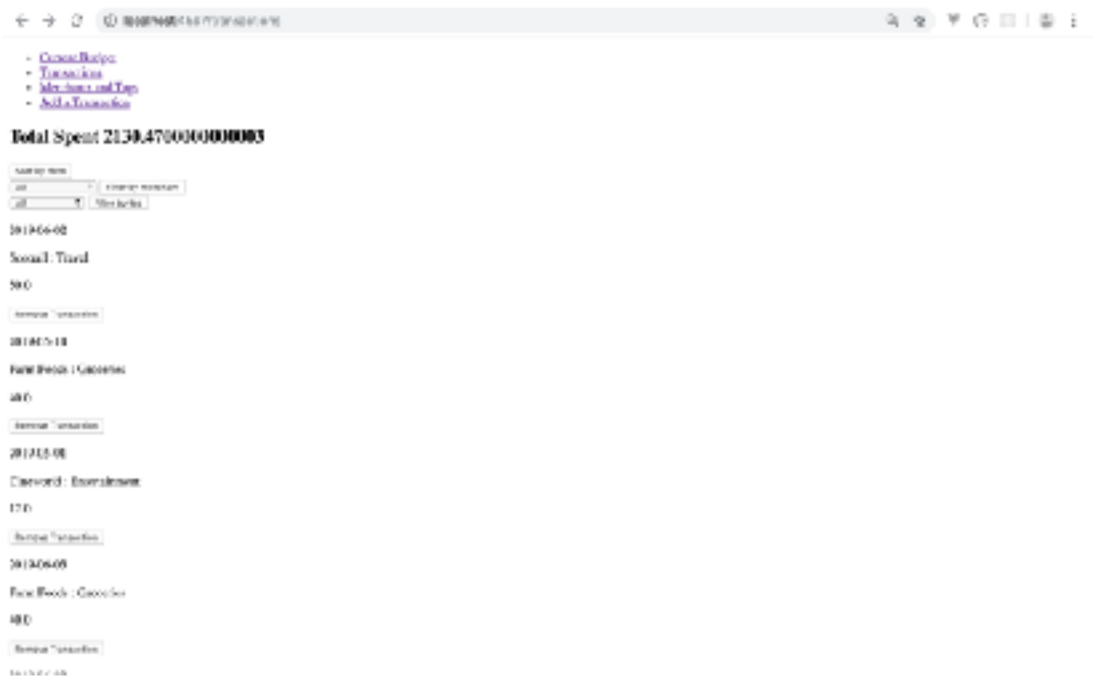
Morrisons : Groceries

8.21

2019-03-14

Asda : Clothing

65.45



- Current Budget
- Transactions
- Merchants and Tags
- Add a Transaction

Total Spent 2130.4700000000003

2019-06-02

Scotrail : Travel

50.0

2019-05-14

Farm Foods : Groceries

40.0

2019-03-08

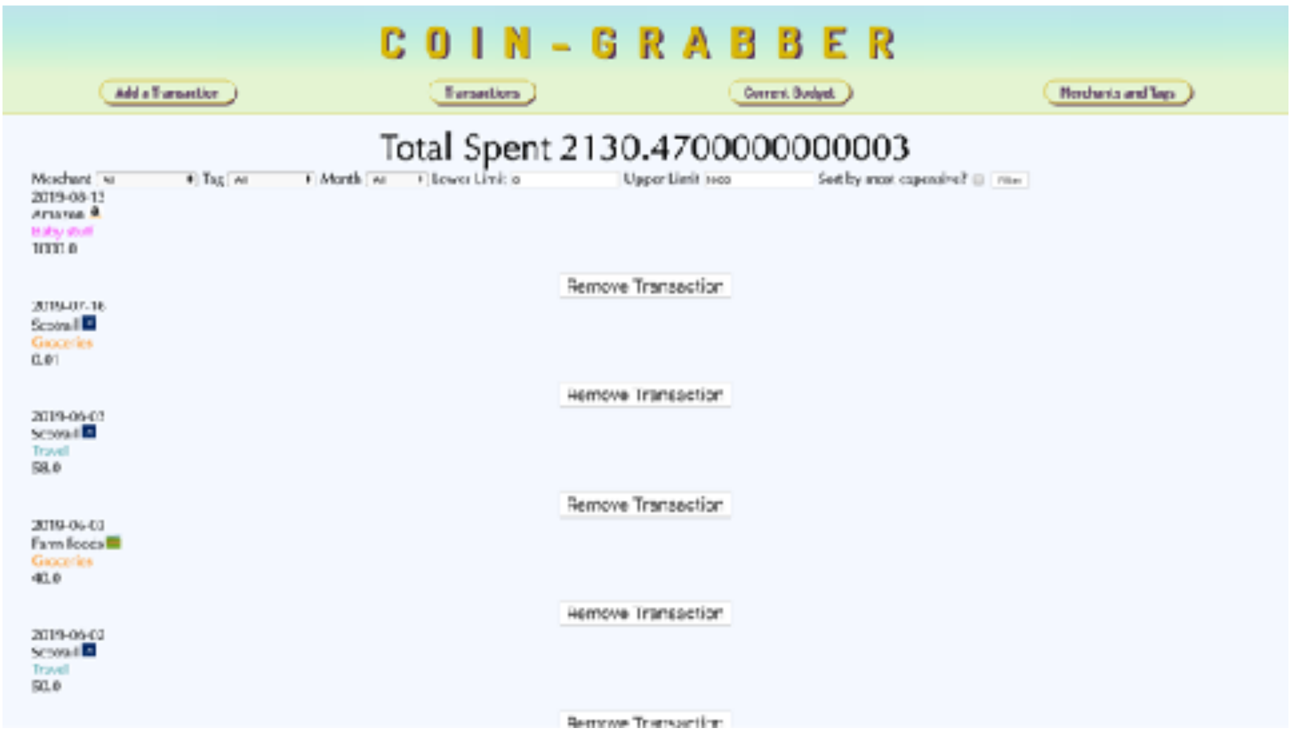
Cineworld : Entertainment

12.0

2019-06-03

Farm Foods : Groceries

40.0



Total Spent 2130.4700000000003

Merchant All Tag All Month All Lower Limit: 0 Upper Limit 1000 Sort by most expensive?

2019-08-13
Amazon
Baby stuff
1000.0

2019-07-16
Scotrail
Groceries
0.01

2019-06-03
Scotrail
Travel
58.0

2019-06-03
Farm foods
Groceries
40.0

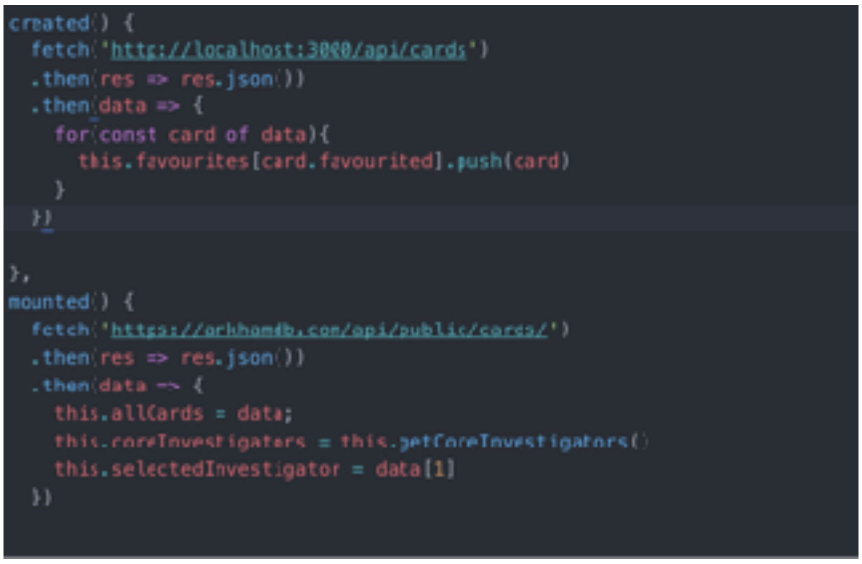
2019-06-02
Scotrail
Travel
50.0

Description here

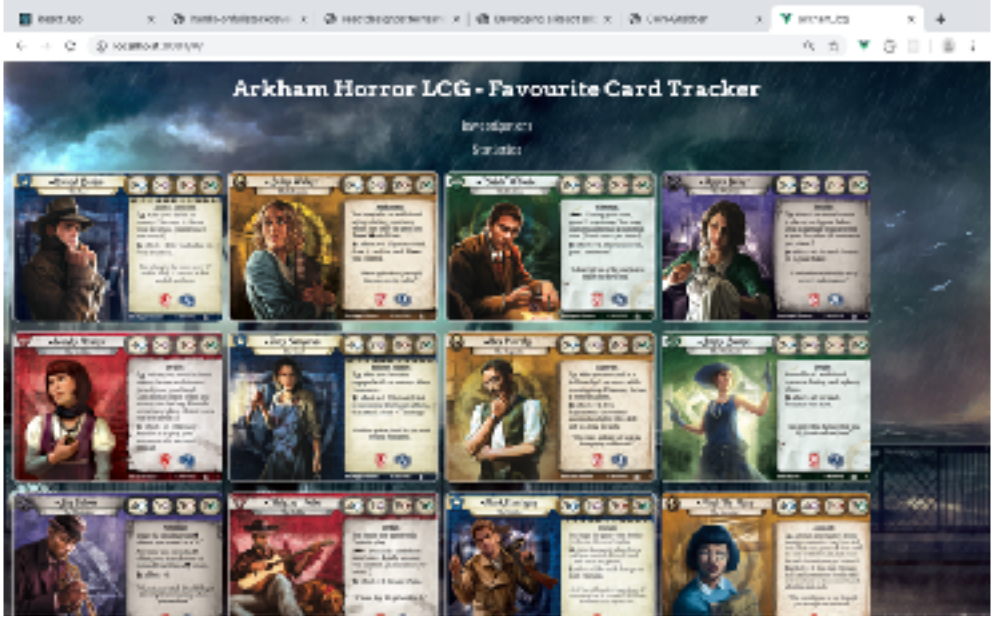
Week 7

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:

Paste Screenshot here



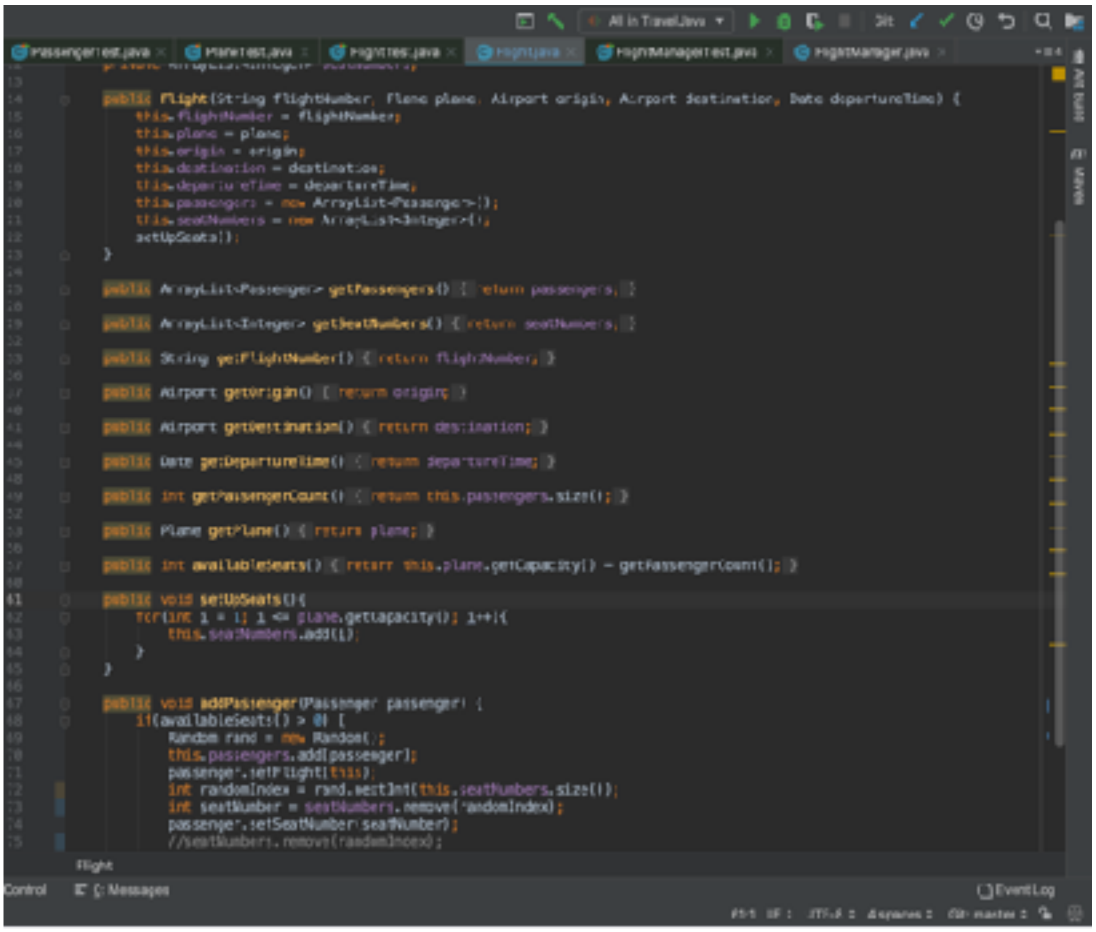
```
created() {
  fetch('http://localhost:3000/api/cards')
  .then(res => res.json())
  .then(data => {
    for(const card of data){
      this.favourites[card.favourited].push(card)
    }
  })

},
mounted() {
  fetch('https://arkhamdb.com/api/public/cards/')
  .then(res => res.json())
  .then(data => {
    this.allCards = data;
    this.coreInvestigators = this.getCoreInvestigators()
    this.selectedInvestigator = data[1]
  })
```



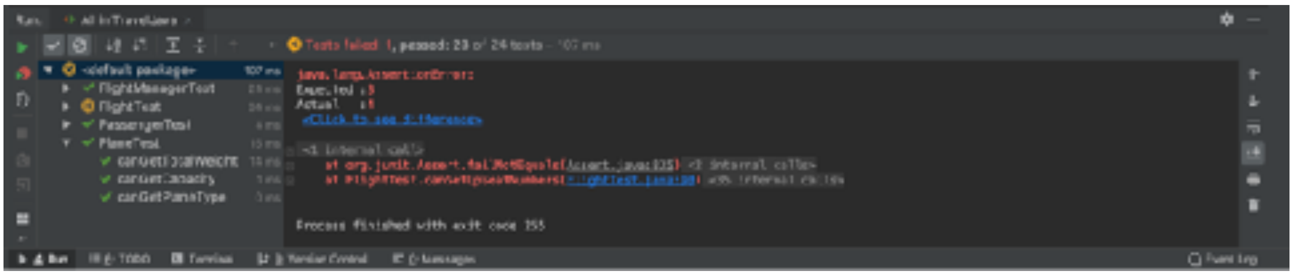
Description here

When mounted, the program fetches data from the arkhamdb.com api, which is then saved as data on the app. The app then uses that data to display the card images as shown in the second screenshot.

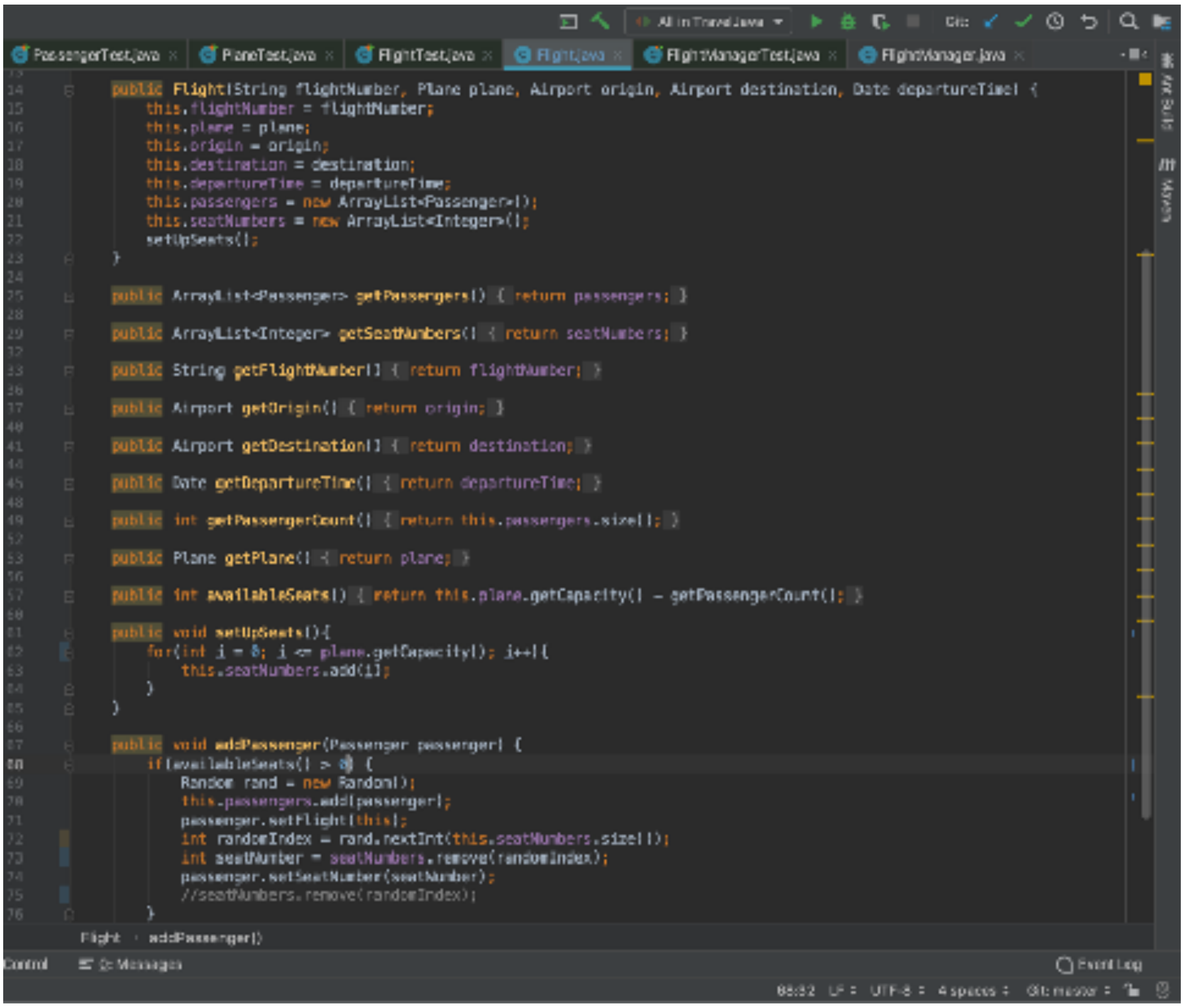
Unit	Ref	Evidence
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing
		Description:



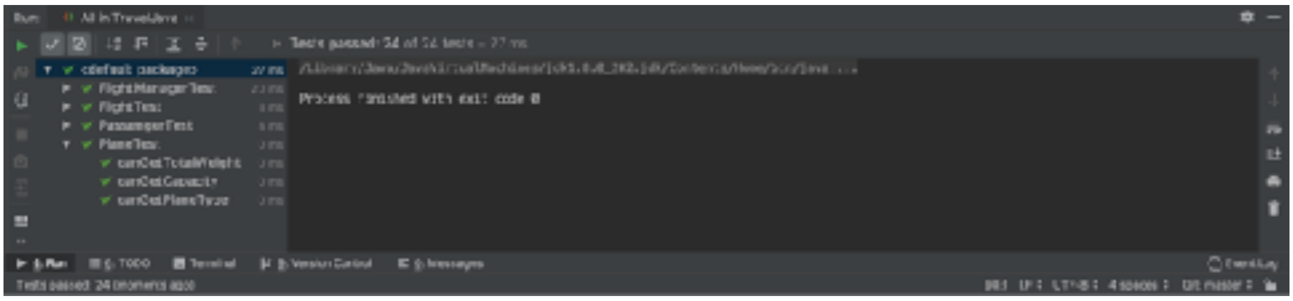
```java
    public Flight(String flightNumber, Plane plane, Airport origin, Airport destination, Date departureTime) {
        this.flightNumber = flightNumber;
        this.plane = plane;
        this.origin = origin;
        this.destination = destination;
        this.departureTime = departureTime;
        this.passengers = new ArrayList<Passenger>();
        this.seatNumbers = new ArrayList<Integer>();
        setUpSeats();
    }

    public ArrayList<Passenger> getPassengers() { return passengers; }

    public ArrayList<Integer> getSeatNumbers() { return seatNumbers; }

    public String getFlightNumber() { return flightNumber; }

    public Airport getOrigin() { return origin; }

    public Airport getDestination() { return destination; }

    public Date getDepartureTime() { return departureTime; }

    public int getPassengerCount() { return this.passengers.size(); }

    public Plane getPlane() { return plane; }

    public int availableSeats() { return this.plane.getCapacity() - getPassengerCount(); }

    public void setUpSeats(){
        for(int i = 1; i <= plane.getCapacity(); i++){
            this.seatNumbers.add(i);
        }
    }

    public void addPassenger(Passenger passenger) {
        if(availableSeats() > 0) {
            Random rand = new Random();
            this.passengers.add(passenger);
            passenger.setFlight(this);
            int randomIndex = rand.nextInt(this.seatNumbers.size());
            int seatNumber = seatNumbers.remove(randomIndex);
            passenger.setSeatNumber(seatNumber);
            //seatNumbers.remove(randomIndex);
```



```
Tests failed: 1, passed: 23 of 24 tests – 107 ms

java.lang.AssertionError:
Expected :3
Actual   :0
<Click to see difference>

<1 internal call>
    at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal calls>
    at FlightTest.canGetSeatNumbers(FlightTest.java:98) <35 internal calls>

Process finished with exit code 255
```



```java
    public Flight(String flightNumber, Plane plane, Airport origin, Airport destination, Date departureTime) {
        this.flightNumber = flightNumber;
        this.plane = plane;
        this.origin = origin;
        this.destination = destination;
        this.departureTime = departureTime;
        this.passengers = new ArrayList<Passenger>();
        this.seatNumbers = new ArrayList<Integer>();
        setUpSeats();
    }

    public ArrayList<Passenger> getPassengers() { return passengers; }

    public ArrayList<Integer> getSeatNumbers() { return seatNumbers; }

    public String getFlightNumber() { return flightNumber; }

    public Airport getOrigin() { return origin; }

    public Airport getDestination() { return destination; }

    public Date getDepartureTime() { return departureTime; }

    public int getPassengerCount() { return this.passengers.size(); }

    public Plane getPlane() { return plane; }

    public int availableSeats() { return this.plane.getCapacity() - getPassengerCount(); }

    public void setUpSeats(){
        for(int i = 0; i <= plane.getCapacity(); i++){
            this.seatNumbers.add(i);
        }
    }

    public void addPassenger(Passenger passenger) {
        if(availableSeats() > 0) {
            Random rand = new Random();
            this.passengers.add(passenger);
            passenger.setFlight(this);
            int randomIndex = rand.nextInt(this.seatNumbers.size());
            int seatNumber = seatNumbers.remove(randomIndex);
            passenger.setSeatNumber(seatNumber);
            //seatNumbers.remove(randomIndex);
        }
```



```
Tests passed: 24 of 24 tests – 27 ms

Process finished with exit code 0
```

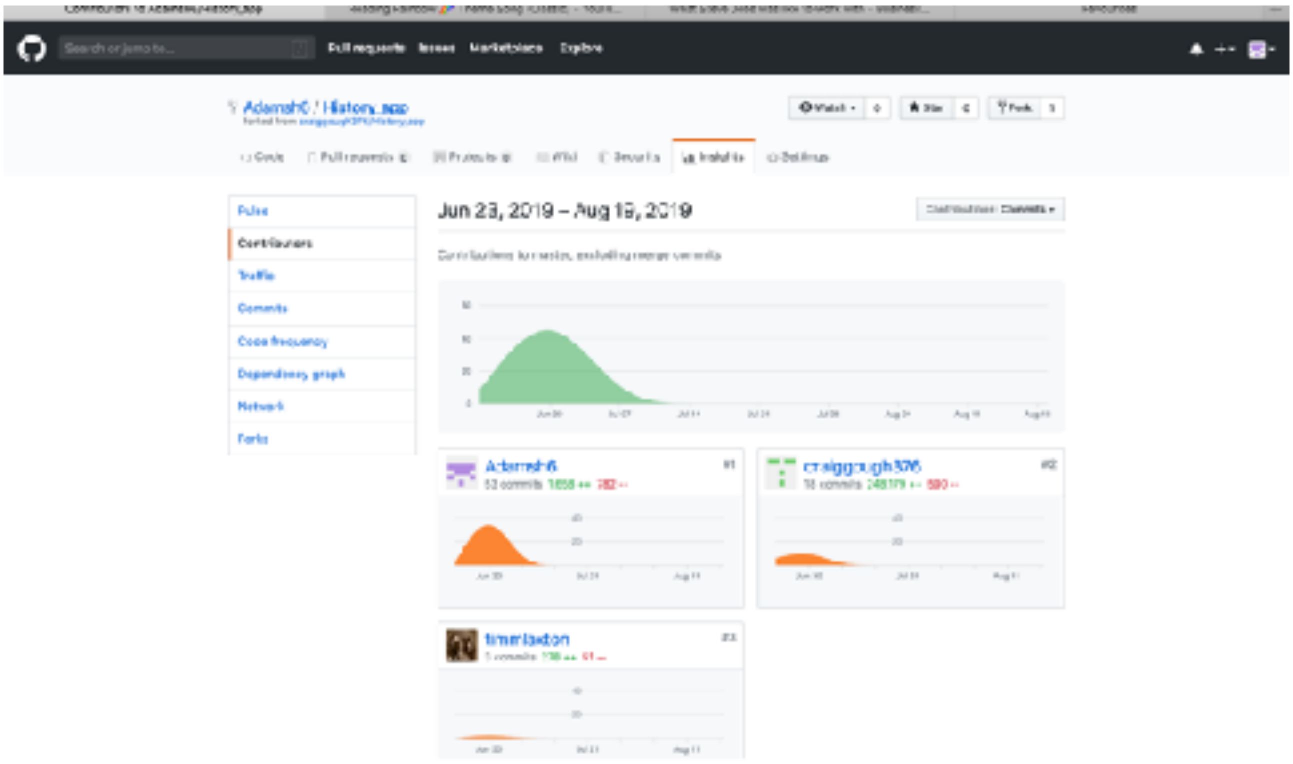
Paste Screenshot here

Description here

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:

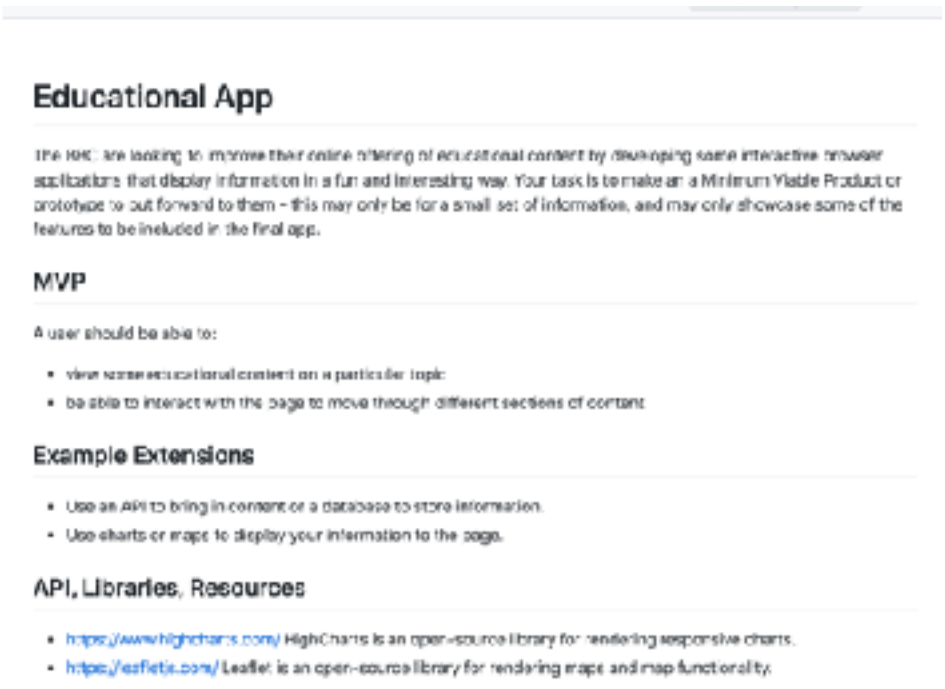
Paste Screenshot here



Description here

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

<u>Paste Screenshot here</u>



Educational App

The BBC are looking to improve their online offering of educational content by developing some interactive browser applications that display information in a fun and interesting way. Your task is to make an a Minimum Viable Product or prototype to put forward to them - this may only be for a small set of information, and may only showcase some of the features to be included in the final app.

MVP

A user should be able to:

- view some educational content on a particular topic
- be able to interact with the page to move through different sections of content

Example Extensions

- Use an API to bring in content or a database to store information.
- Use charts or maps to display your information to the page.

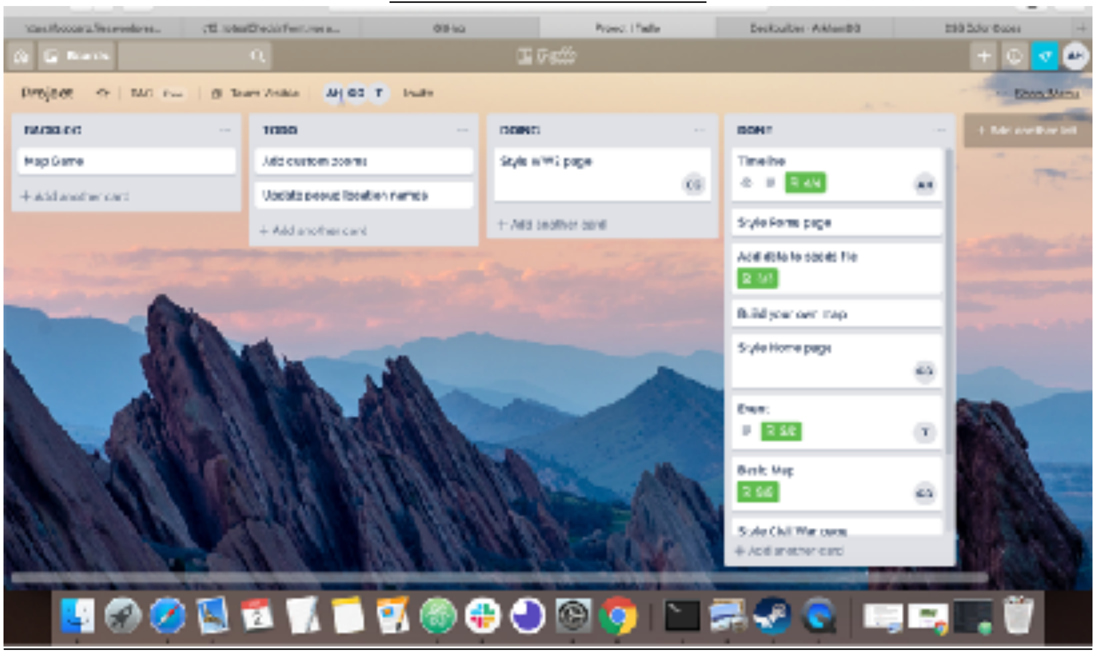
API, Libraries, Resources

- https://www.highcharts.com/ HighCharts is an open-source library for rendering responsive charts.
- https://leafletjs.com/ Leaflet is an open-source library for rendering maps and map functionality.

<u>Description here</u>

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

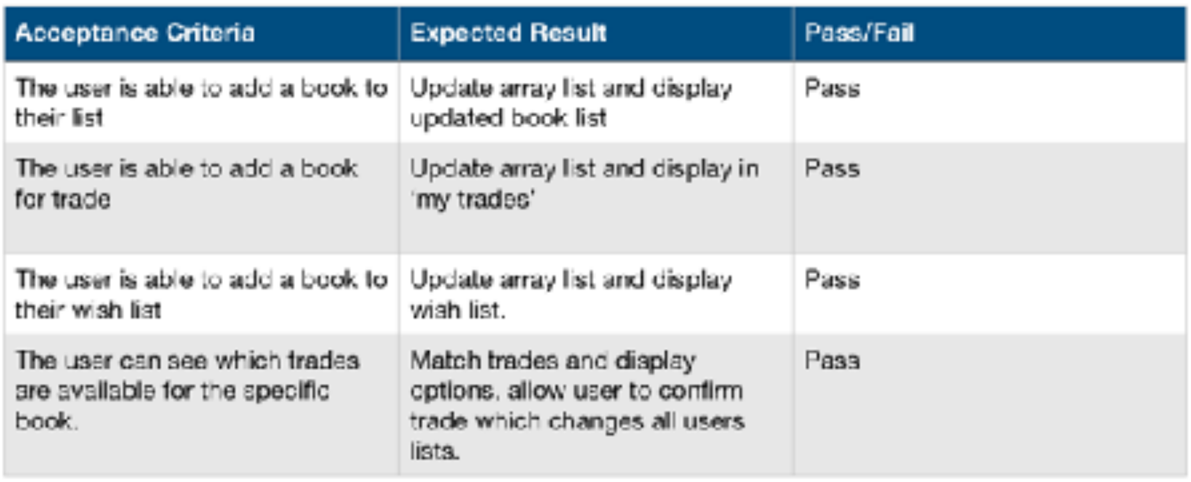
<u>Paste Screenshot here</u>



<u>Description here</u>

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

<u>Paste Screenshot here</u>

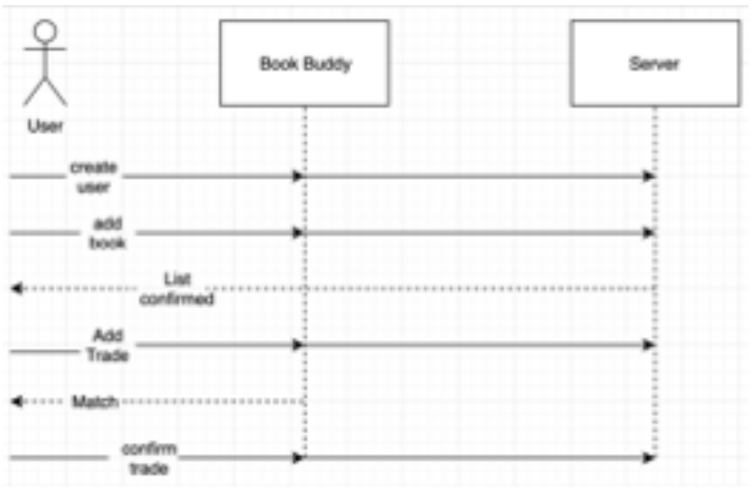


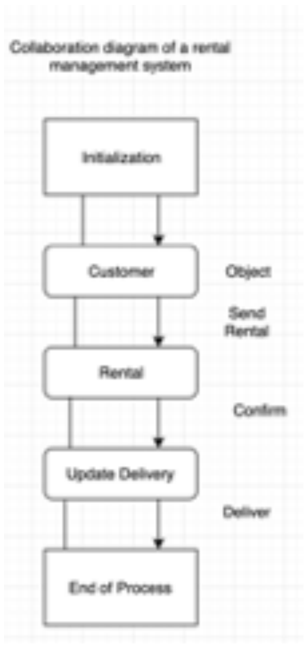
Acceptance Criteria	Expected Result	Pass/Fail
The user is able to add a book to their list	Update array list and display updated book list	Pass
The user is able to add a book for trade	Update array list and display in 'my trades'	Pass
The user is able to add a book to their wish list	Update array list and display wish list.	Pass
The user can see which trades are available for the specific book.	Match trades and display options, allow user to confirm trade which changes all users lists.	Pass

Description here

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

Paste Screenshot here

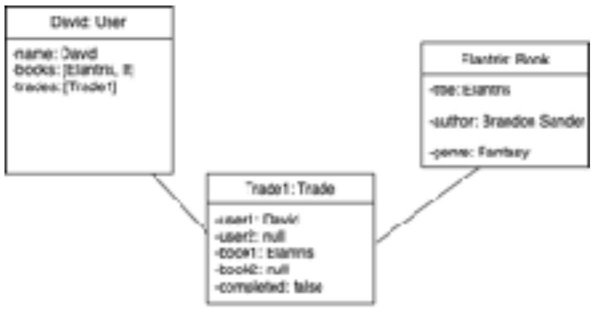


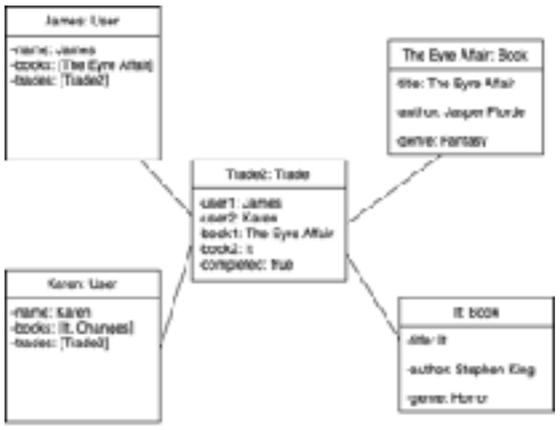


Description here

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

Paste Screenshot here

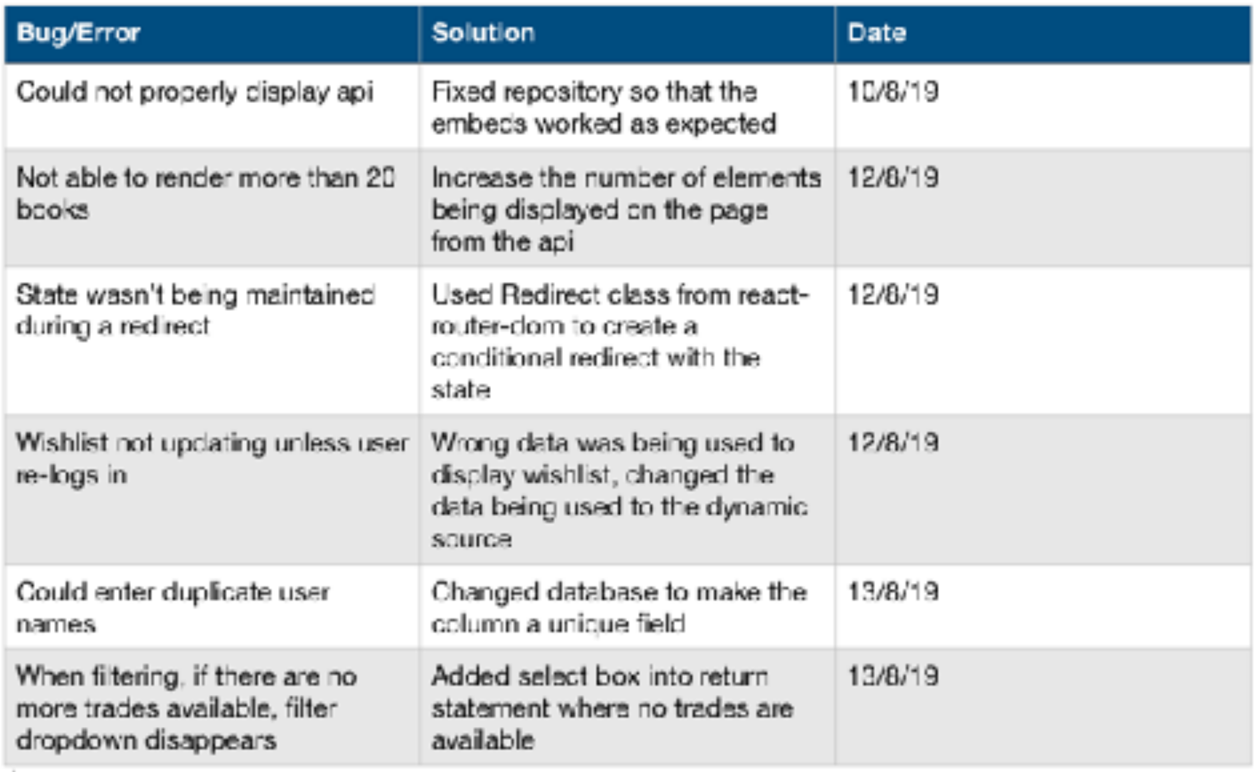




Description here

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

Paste Screenshot here



Bug/Error	Solution	Date
Could not properly display api	Fixed repository so that the embeds worked as expected	10/8/19
Not able to render more than 20 books	Increase the number of elements being displayed on the page from the api	12/8/19
State wasn't being maintained during a redirect	Used Redirect class from react-router-dom to create a conditional redirect with the state	12/8/19
Wishlist not updating unless user re-logs in	Wrong data was being used to display wishlist, changed the data being used to the dynamic source	12/8/19
Could enter duplicate user names	Changed database to make the column a unique field	13/8/19
When filtering, if there are no more trades available, filter dropdown disappears	Added select box into return statement where no trades are available	13/8/19

Description here

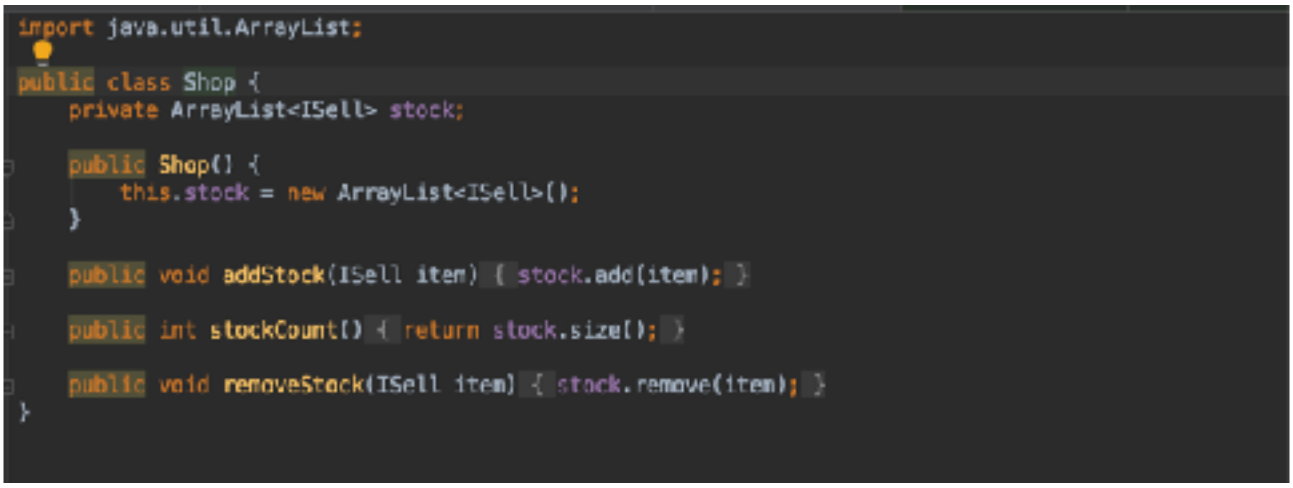
Week 12

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

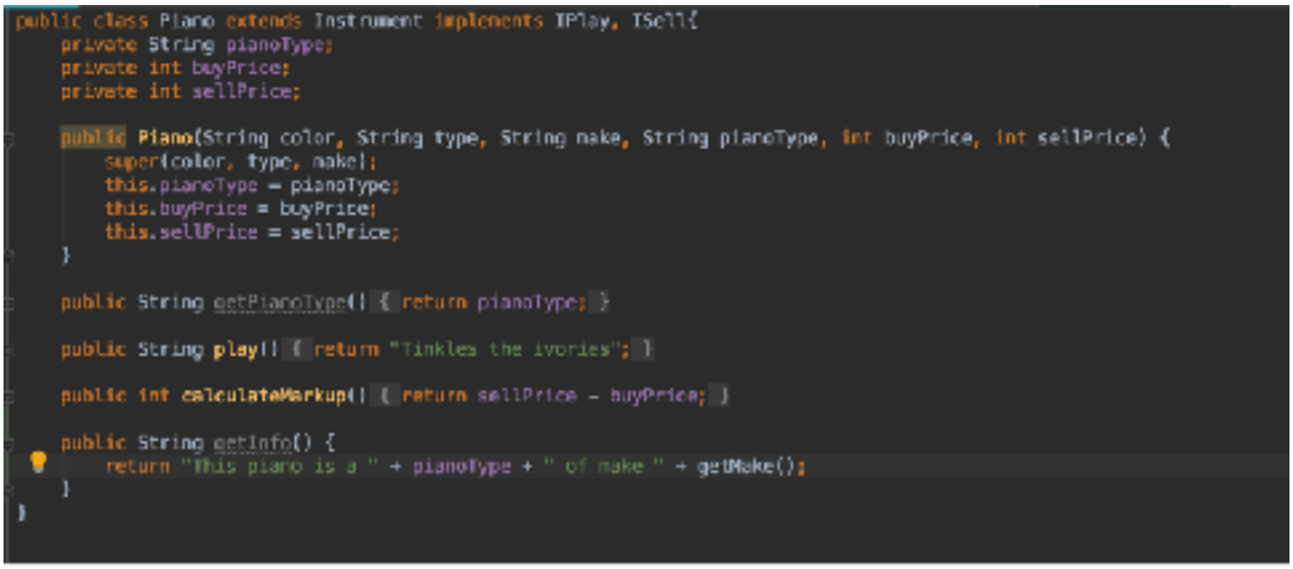
Paste Screenshot here

Description here

A Shop class with an ArrayList that holds the type ISell, which is an interface. The second screenshot is of a class that implements ISell. The shop would be able to store Piano instances in its stock ArrayList, along with instances of other classes that implement ISell.



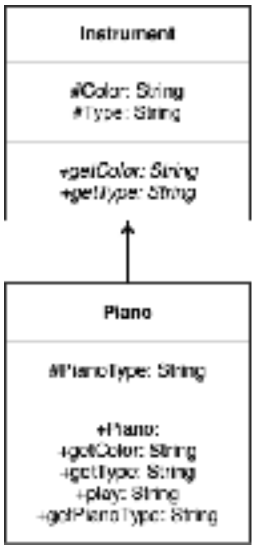
```java
import java.util.ArrayList;

public class Shop {
    private ArrayList<ISell> stock;

    public Shop() {
        this.stock = new ArrayList<ISell>();
    }

    public void addStock(ISell item) { stock.add(item); }

    public int stockCount() { return stock.size(); }

    public void removeStock(ISell item) { stock.remove(item); }
}
```



```java
public class Piano extends Instrument implements IPlay, ISell{
    private String pianoType;
    private int buyPrice;
    private int sellPrice;

    public Piano(String color, String type, String make, String pianoType, int buyPrice, int sellPrice) {
        super(color, type, make);
        this.pianoType = pianoType;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getPianoType() { return pianoType; }

    public String play() { return "Tinkles the ivories"; }

    public int calculateMarkup() { return sellPrice - buyPrice; }

    public String getInfo() {
        return "This piano is a " + pianoType + " of make " + getMake();
    }
}
```

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

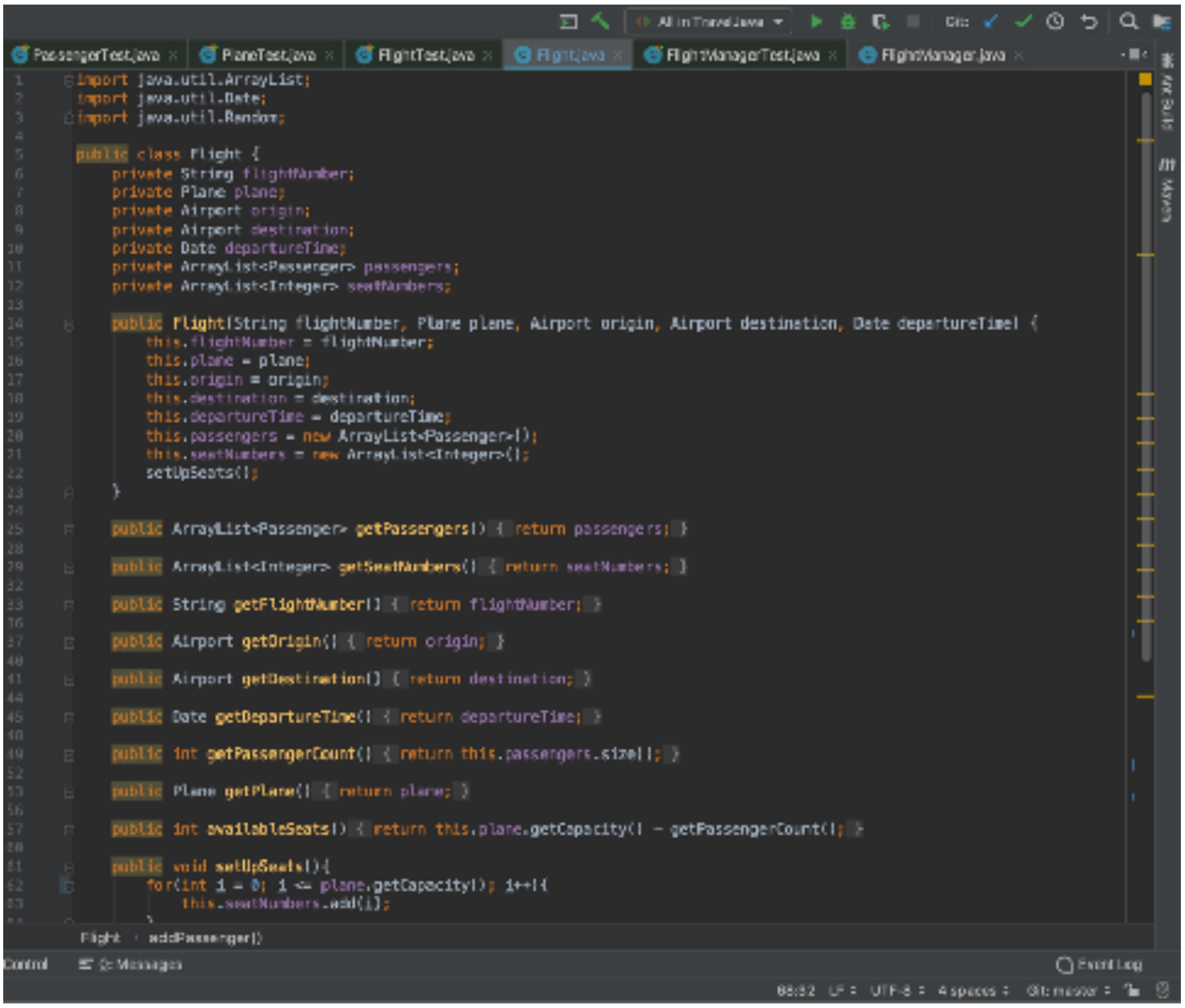
Paste Screenshot here



Description here

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

Paste Screenshot here



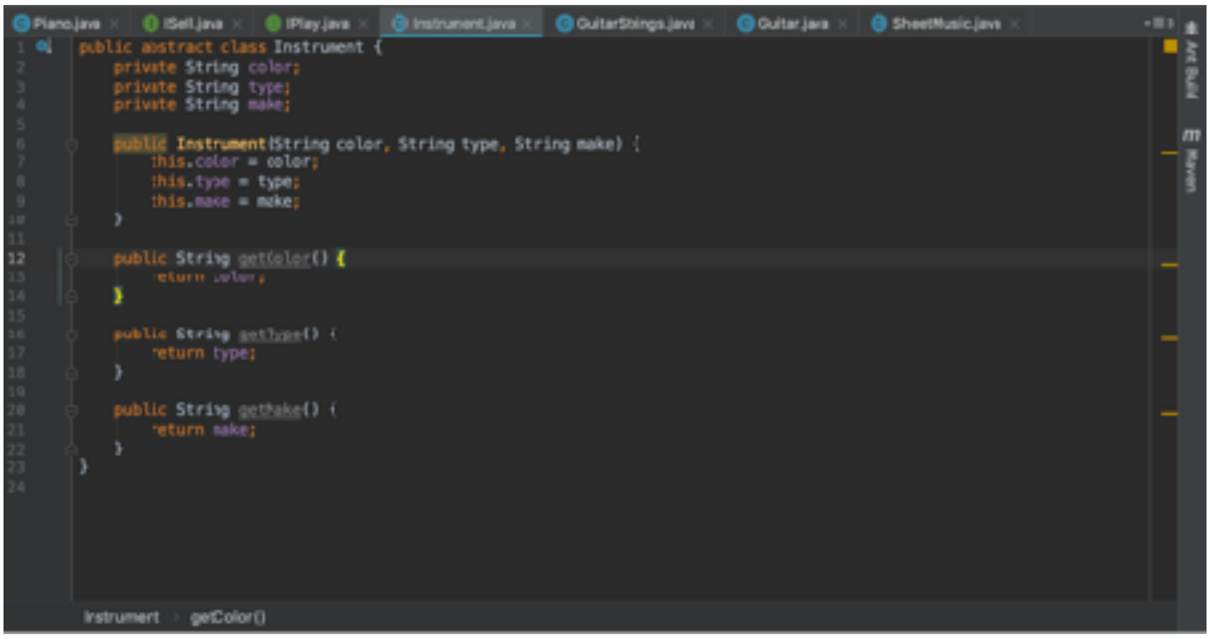
```java
import java.util.ArrayList;
import java.util.Date;
import java.util.Random;

public class Flight {
    private String flightNumber;
    private Plane plane;
    private Airport origin;
    private Airport destination;
    private Date departureTime;
    private ArrayList<Passenger> passengers;
    private ArrayList<Integer> seatNumbers;

    public Flight(String flightNumber, Plane plane, Airport origin, Airport destination, Date departureTime) {
        this.flightNumber = flightNumber;
        this.plane = plane;
        this.origin = origin;
        this.destination = destination;
        this.departureTime = departureTime;
        this.passengers = new ArrayList<Passenger>();
        this.seatNumbers = new ArrayList<Integer>();
        setUpSeats();
    }

    public ArrayList<Passenger> getPassengers() { return passengers; }

    public ArrayList<Integer> getSeatNumbers() { return seatNumbers; }

    public String getFlightNumber() { return flightNumber; }

    public Airport getOrigin() { return origin; }

    public Airport getDestination() { return destination; }

    public Date getDepartureTime() { return departureTime; }

    public int getPassengerCount() { return this.passengers.size(); }

    public Plane getPlane() { return plane; }

    public int availableSeats() { return this.plane.getCapacity() - getPassengerCount(); }

    public void setUpSeats(){
        for(int i = 0; i <= plane.getCapacity(); i++){
            this.seatNumbers.add(i);
```

Description here

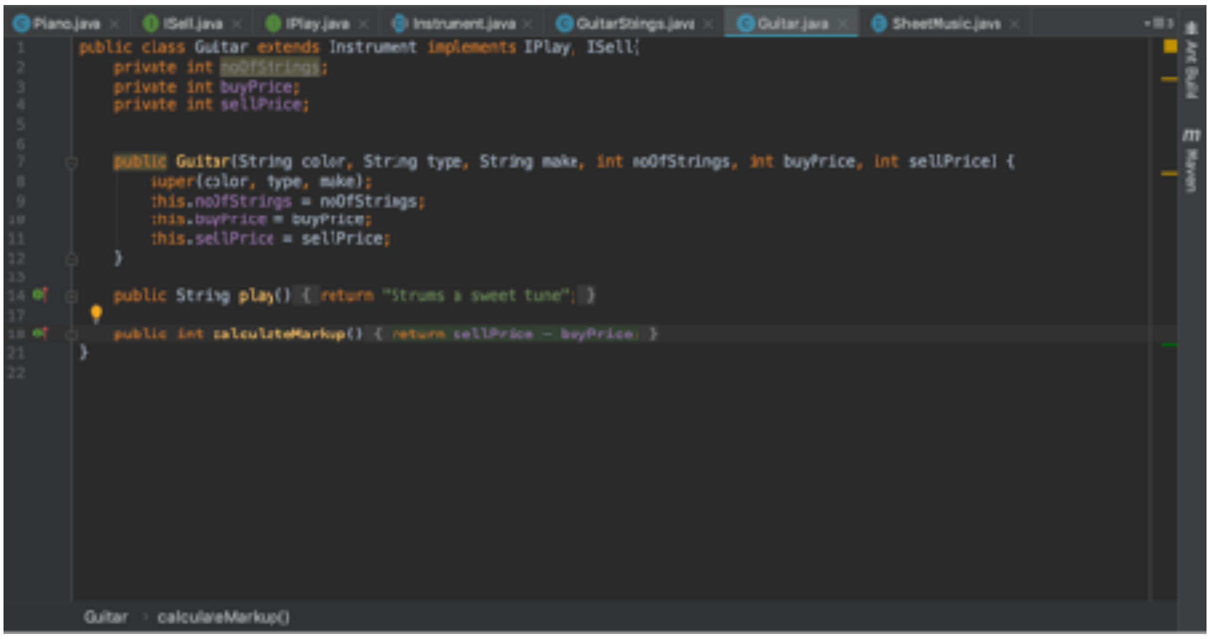
The Flight class is using encapsulation - all the variables are private, and can only be accessed via public getter and setter methods within the class.

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:

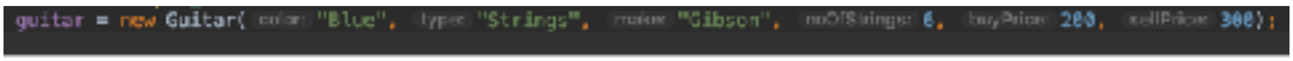
Paste Screenshot here



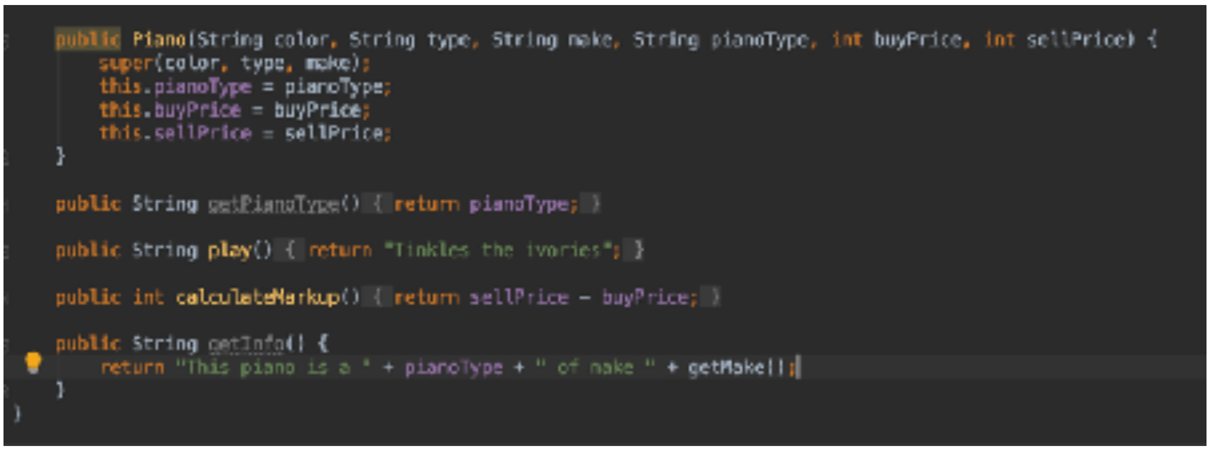
```java
public abstract class Instrument {
    private String color;
    private String type;
    private String make;

    public Instrument(String color, String type, String make) {
        this.color = color;
        this.type = type;
        this.make = make;
    }

    public String getColor() {
        return color;
    }

    public String getType() {
        return type;
    }

    public String getMake() {
        return make;
    }
}
```



```java
public class Guitar extends Instrument implements IPlay, ISell{
    private int noOfStrings;
    private int buyPrice;
    private int sellPrice;


    public Guitar(String color, String type, String make, int noOfStrings, int buyPrice, int sellPrice) {
        super(color, type, make);
        this.noOfStrings = noOfStrings;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String play() { return "Strums a sweet tune"; }

    public int calculateMarkup() { return sellPrice - buyPrice; }
}
```



```java
guitar = new Guitar( color: "Blue",  type: "Strings",  make: "Gibson",  noOfStrings: 6,  buyPrice: 200,  sellPrice: 300);
```

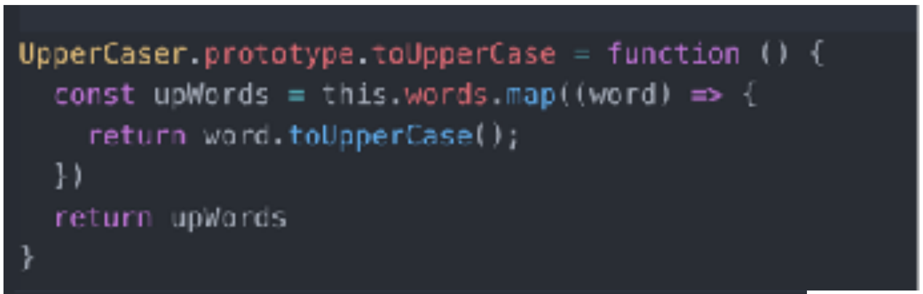


```java
    public Piano(String color, String type, String make, String pianoType, int buyPrice, int sellPrice) {
        super(color, type, make);
        this.pianoType = pianoType;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getPianoType() { return pianoType; }

    public String play() { return "Tinkles the ivories"; }

    public int calculateMarkup() { return sellPrice - buyPrice; }

    public String getInfo() {
        return "This piano is a " + pianoType + " of make " + getMake();
    }
}
```

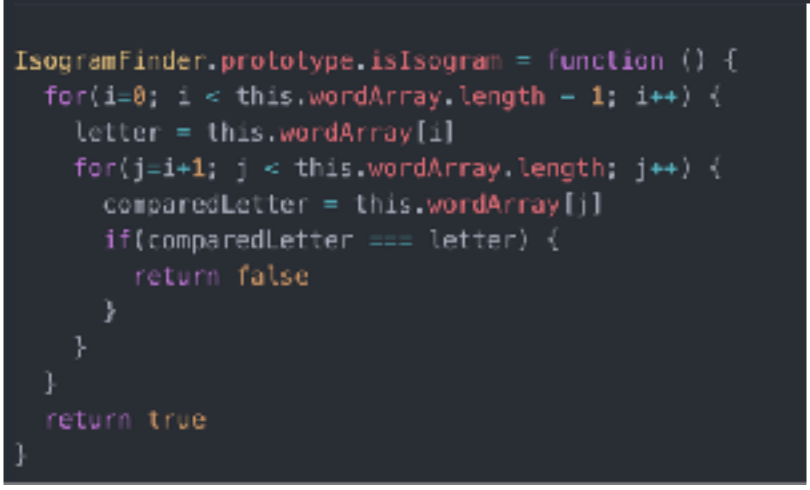
Description here

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:

Paste Screenshot here



```javascript
UpperCaser.prototype.toUpperCase = function () {
  const upWords = this.words.map((word) => {
    return word.toUpperCase();
  })
  return upWords
}
```



```javascript
IsogramFinder.prototype.isIsogram = function () {
  for(i=0; i < this.wordArray.length - 1; i++) {
    letter = this.wordArray[i]
    for(j=i+1; j < this.wordArray.length; j++) {
      comparedLetter = this.wordArray[j]
      if(comparedLetter === letter) {
        return false
      }
    }
  }
  return true
}
```

Description here

For the first algorithm, I chose to use it because the built in functionality of the map function in javascript meant that I could easily mutate every element in an array in the same way.

For the second algorithm, I chose to use it because I could use nested for loops to compare every element of my array against every other element. Using an explicit for loop meant that I could set the starting index of the array, meaning I would never have to repeat the same comparison twice.