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: An array of fruits is created with the variable name 'fruits'. A function 'get_first_item' is defined which takes the argument of 'item_list', the function then returns the first item of the array. The function is then called with 'p' to print it to the console and it returns the first item, "apple".

```
1  fruits = ["apple", "banana", "grape", "orange"]
2
3  def get_first_item(item_list)
4    return item_list.first
5  end
6
7  p get_first_item(fruits)
```

```
→  arrays ls
arrays.rb
→  arrays ruby arrays.rb
"apple"
→  arrays
```

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: This is a Bank Account class that takes in a holder_name, balance and type. When the function 'pay_fee' is called, it takes away the charge value from the the balance after looking up the account type in the hash and it's corresponding value.

```ruby
require_relative('./bank_account')

bankAccount = BankAccount.new('Ben', 1000, 'personal')
bankAccount.pay_fee
```

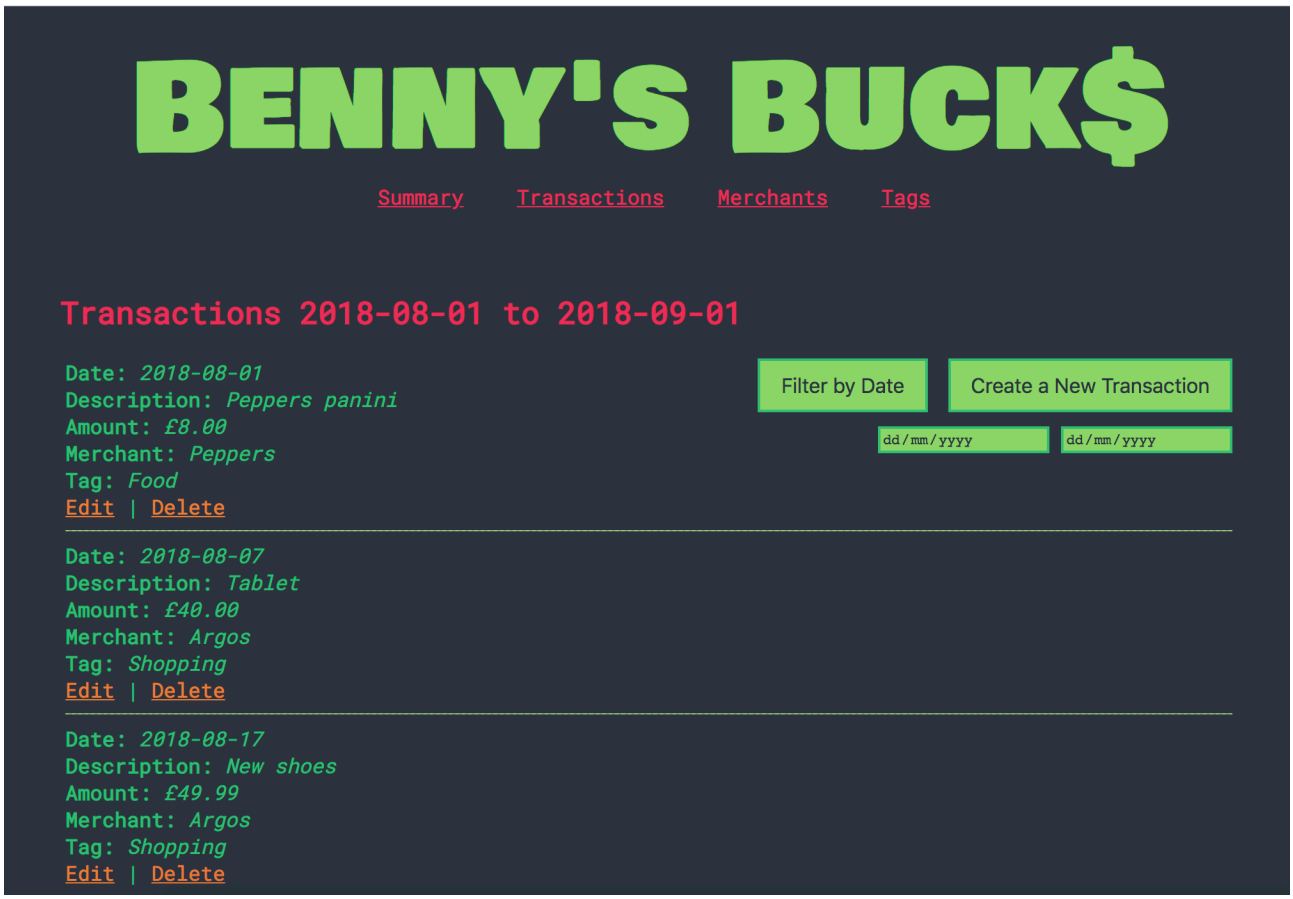
```ruby
class BankAccount
  attr_accessor(:holder_name, :balance, :type)
  def initialize(name, balance, type)
    @holder_name = name
    @balance = balance
    @type = type
  end

  def pay_in(amount)
    @balance += amount

    p "New balance: #{@balance}"
  end

  def pay_fee
    charges = {
      'personal' => 10,
      'business' => 50
    }

    @balance -= charges[@type]

    p "New balance: #{@balance}"
  end
end
```

```
→  classes ruby runner.rb
"New balance: 990"
→  classes
```

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: The function takes a start date and an end date as parameters, these are then passed to the SQL query. The SQL query then selects all the data from the database between the input start and end date and orders them in ascending order. The result is finally mapped into a new Transaction and returned.

```ruby
def self.date_between(start_date, end_date)
  sql = "SELECT * FROM transactions
         WHERE transaction_date
         BETWEEN
         $1 AND $2
         ORDER BY transaction_date ASC"
  values = [start_date, end_date]
  result = SqlRunner.run(sql, values)
  result.map { |transaction| Transaction.new(transaction) }
end
```

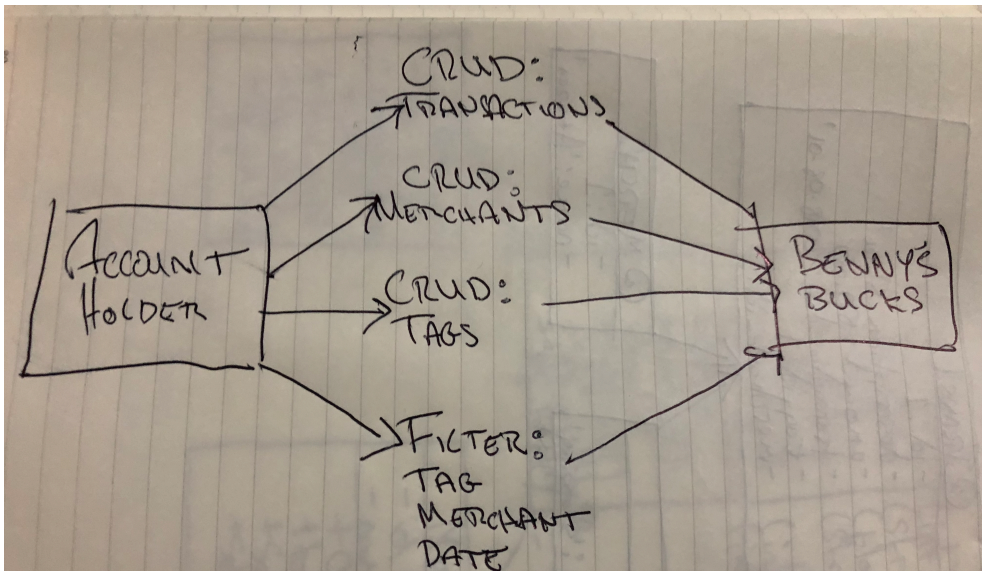


Unit	Ref	Evidence
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running
		Description: The function takes a start date and an end date parameter. The function then runs an SQL query to the database by selecting all the transactions between the given dates and orders them by the transaction_date in ascending order. The result is then mapped and instantiated into a new Transaction.

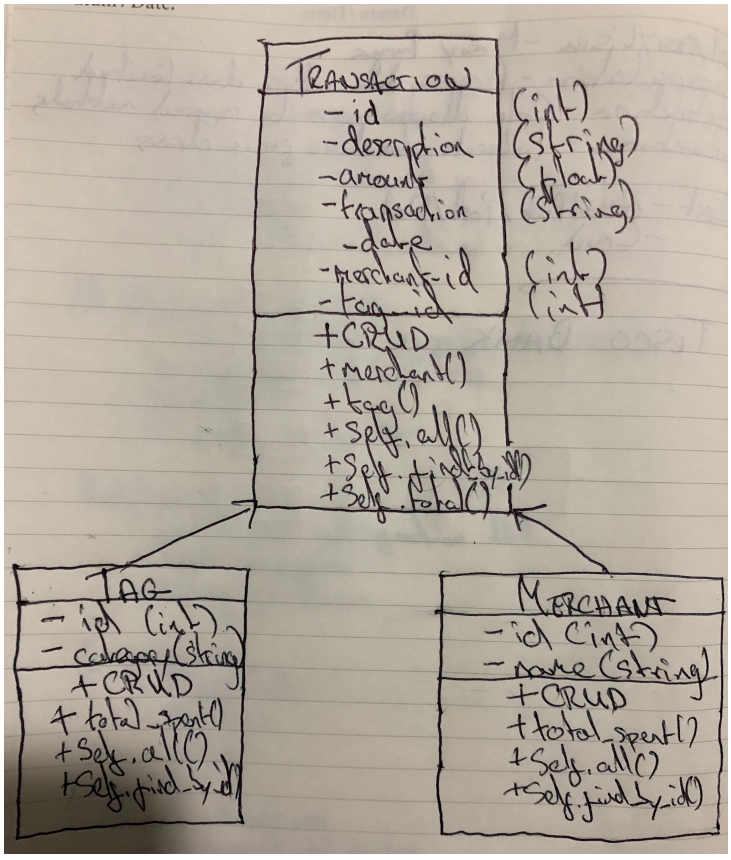
```ruby
def self.date_between(start_date, end_date)
  sql = "SELECT * FROM transactions
        WHERE transaction_date
        BETWEEN
        $1 AND $2
        ORDER BY transaction_date ASC"
  values = [start_date, end_date]
  result = SqlRunner.run(sql, values)
  result.map { |transaction| Transaction.new(transaction) }
end
```

Week 5 and 6

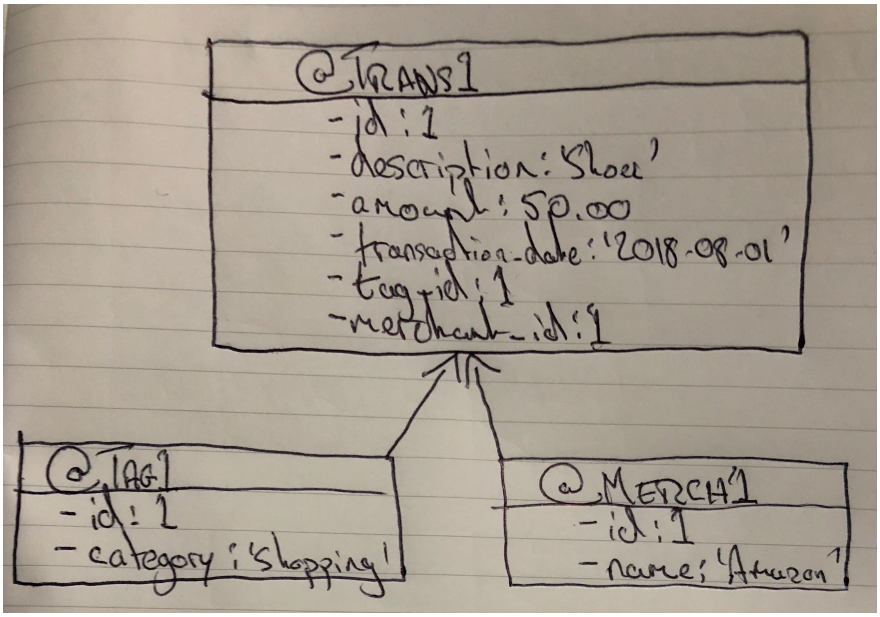
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram
		Description: A case diagram for my transaction manager project, showing the relation between the account holder and the transaction manager itself.



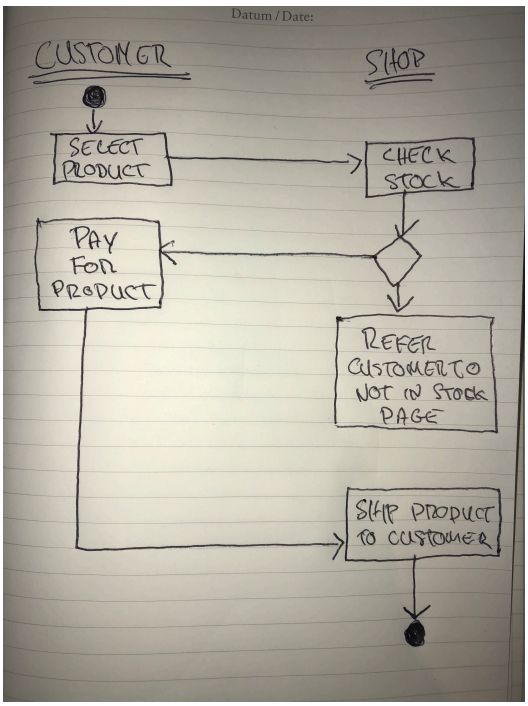
Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		Description: A class diagram for my transaction manager project, showing each class and their relevance to each other, along with their parameters and methods.



Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram
		Description: This is an object diagram for the transaction manager project, showing example objects within the project and their relation to each other with their relevant values.



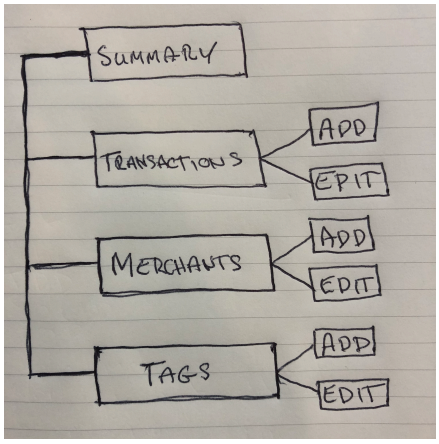
Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram
		Description: This shows an activity diagram for a Shop project. It shows the interaction between the Customer and the Shop, how the flow works dependant on whether the product is in stock or not.



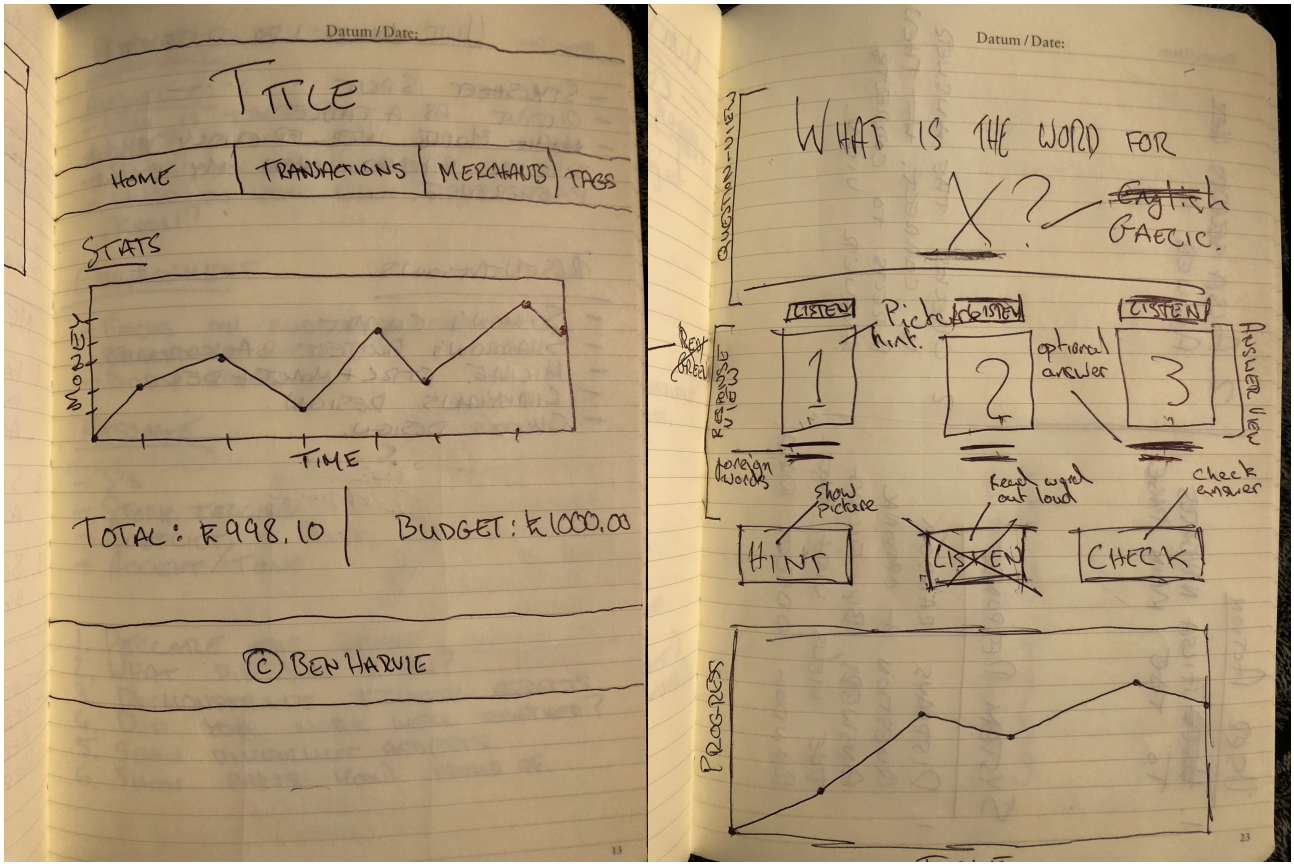
Unit	Ref	Evidence
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time
		Description: This is a table of implementation constraints planning for the transaction manager project.

Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	User may not have the required software or dependencies to run the application	Host the application on a remote website/server
Performance Requirements	As more transactions are added to the application, querying the data and displaying it will take more time	Limit the amount of results per page
Persistence Storage and Transactions	Storage limitations	Clear the database after a period of time
Usability	The app may be used on different kinds of devices (PC, mobile, tablet, etc.)	Style the page in such a way that it is generally easily visible and usable across all devices
Budgets	No budget	Use free gems/libraries to assist in the creation of the app
Time Limitation	There was a deadline of 1 week to meet and an MVP needed to be completed by this time	Focus solely on the MVP first before any extra features

Unit	Ref	Evidence
P	P.5	User Site Map
		Description: This is the Site Map for my Transaction Tracker, showing 4 main pages; Summary, Transactions, Merchants and Tags. Transactions, Merchants and Tags all branch of to Add and Edit options.



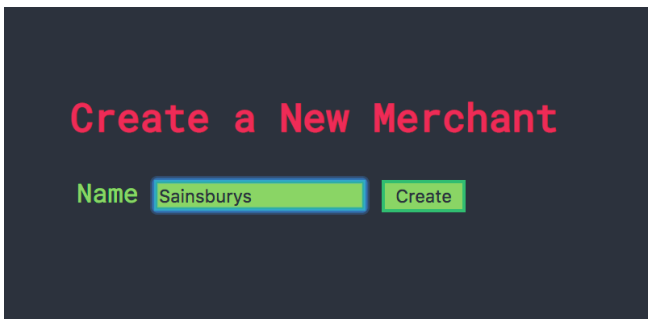
Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams
		Description: A wireframe diagram for “Benny’s Bucks” Transaction Tracker on the ***left*** showing the front summary page. A wireframe diagram for “Vocabulr” Language Learning Game on the ***right*** showing the page’s output and basic views.

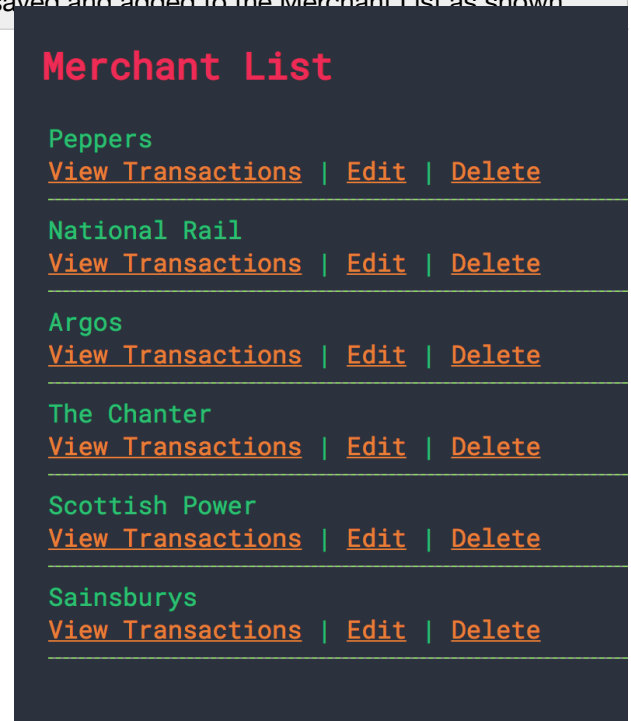


Unit	Ref	Evidence
P	P.10	Example of Pseudocode used for a method
		Description: Pseudocode for a getPrice method which takes in a Cryptocurrency and a Global Currency as arguments. The pseudocode describes each step taken by the function for it to return it's related value.

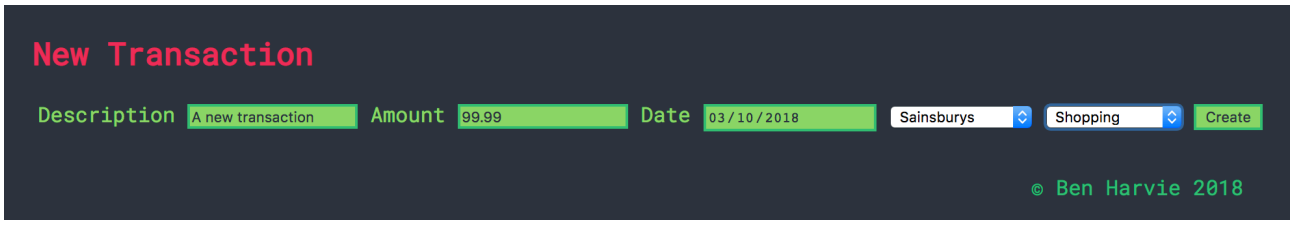
```
19 const getPrice = function(cryptoCurrency, globalCurrency) {
20   // 1. Match seleted cryptoCurrency to it's given currencyID.
21   // 2. Set up a request to the API.
22   // 3. Send the API a request with the given currencyID.
23   // 3. Get the price in the currency that matches the functions globalCurrency.
24   // 4. Return the final price.
25
```

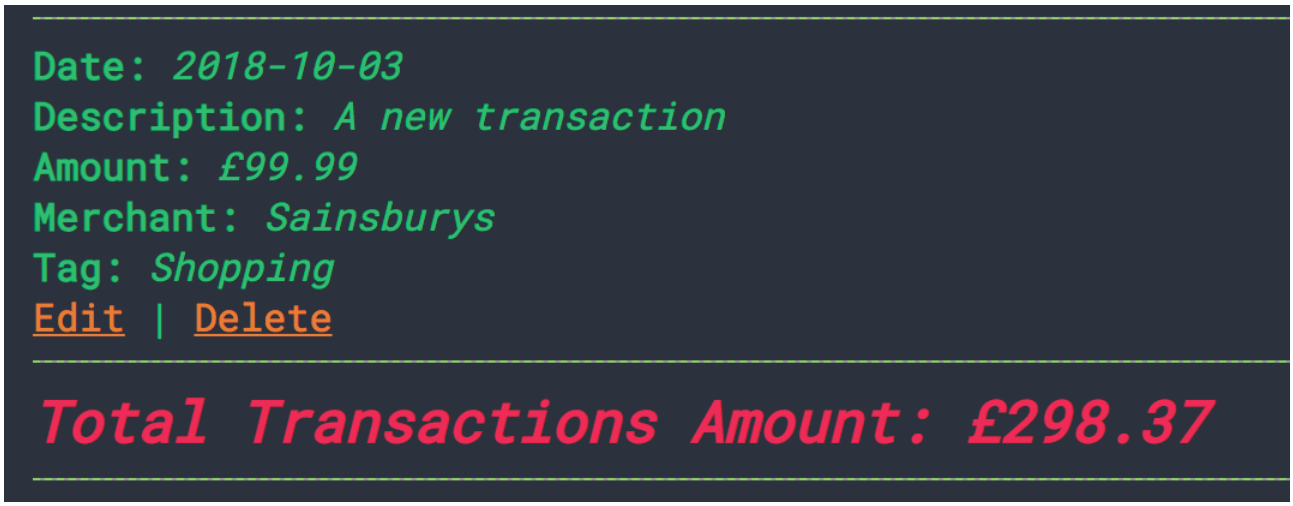
Unit	Ref	Evidence
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way
		Description: User inputs 'Sainsburys' as a new merchant. Once the user clicks 'Create', the new merchant is then saved and added to the Merchant List as shown



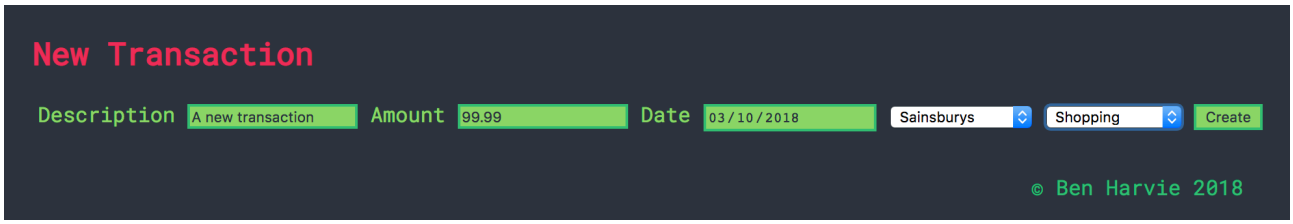


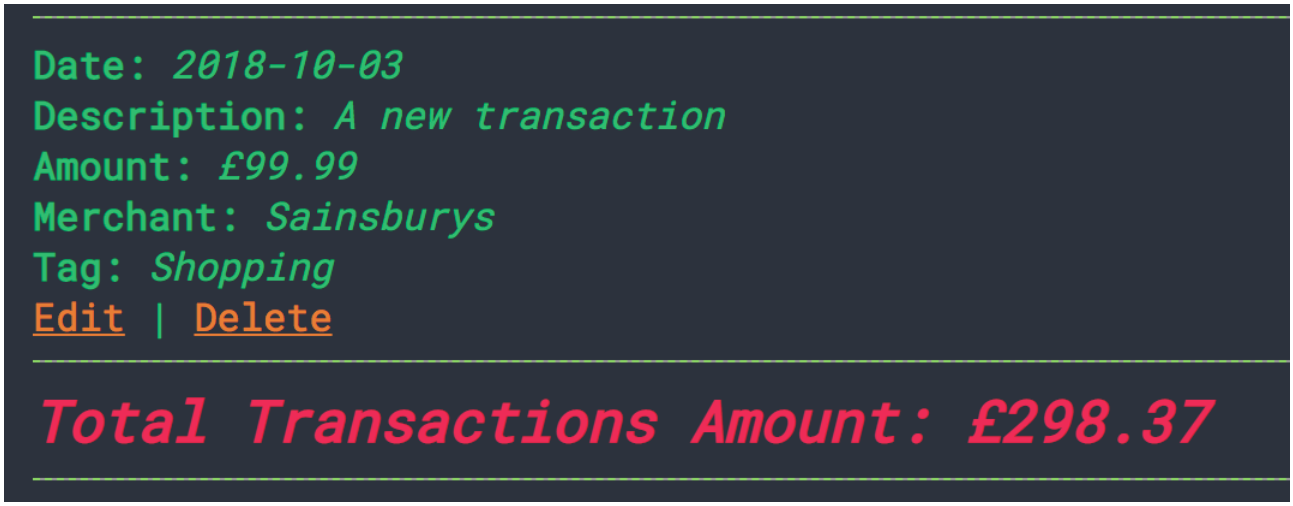
Unit	Ref	Evidence
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved
		Description: A user enters a new transaction and their related details. Once clicking 'Create' the data is added to the database and the data is then visible on the transaction list.



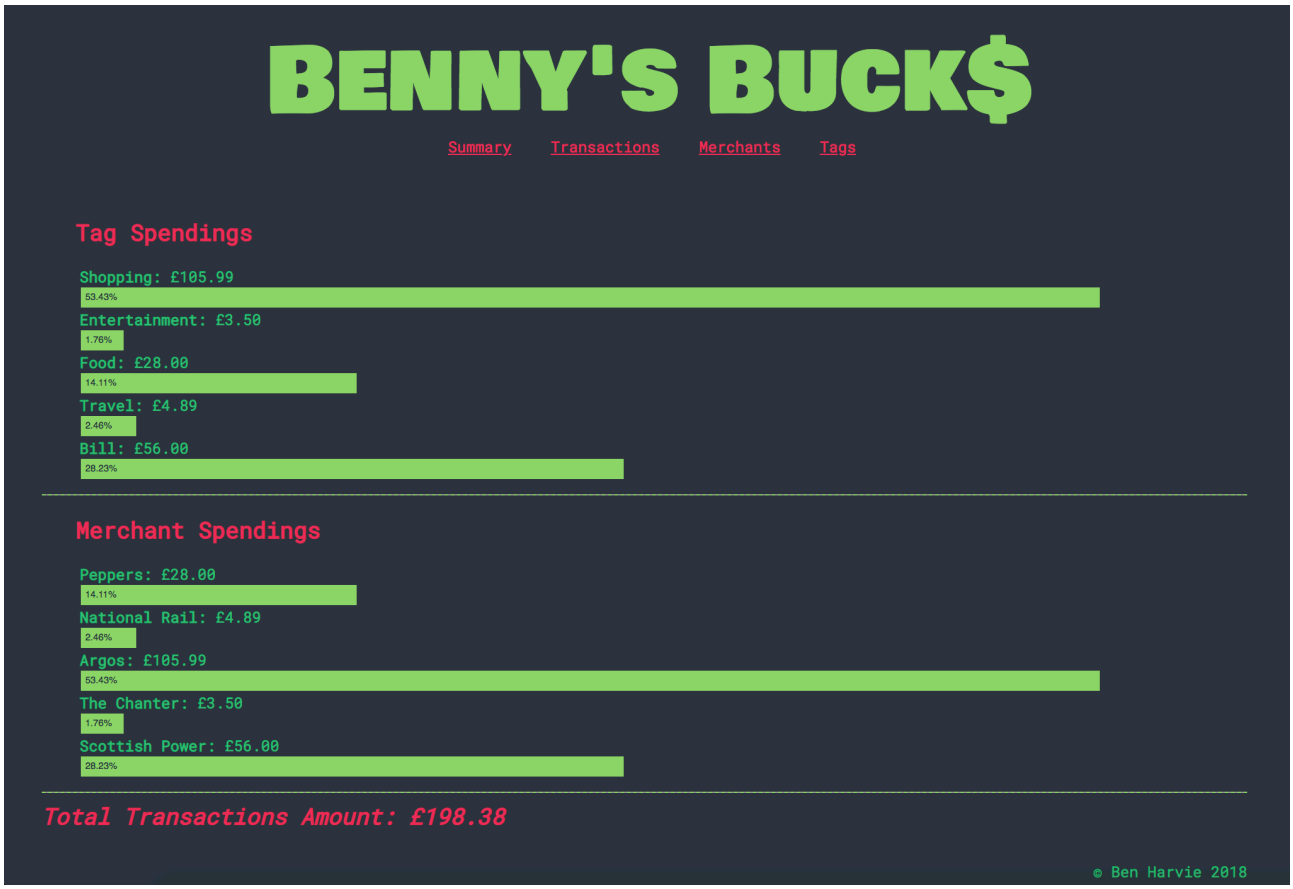


Unit	Ref	Evidence
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program
		Description: The user inputs data including description, amount, date, merchant and tag. The user can then create the transaction by hitting the 'Create' button and the data will be saved to the database and be visible within the application.

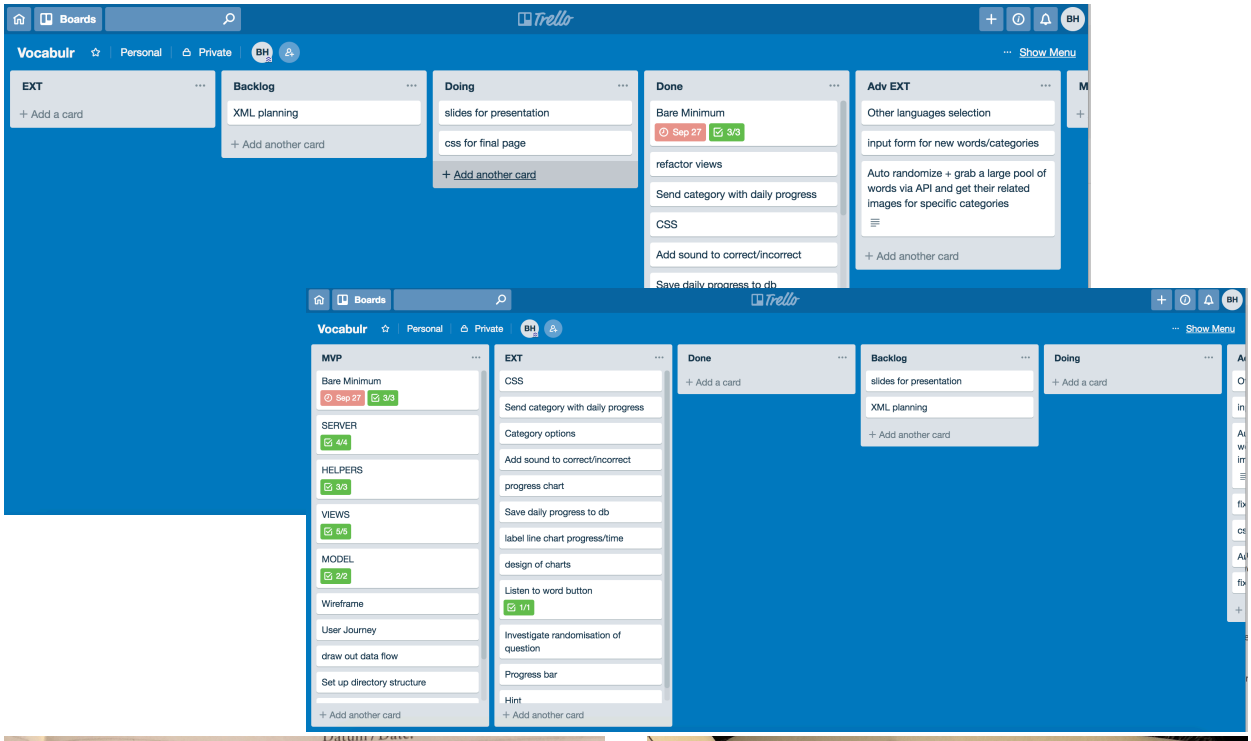


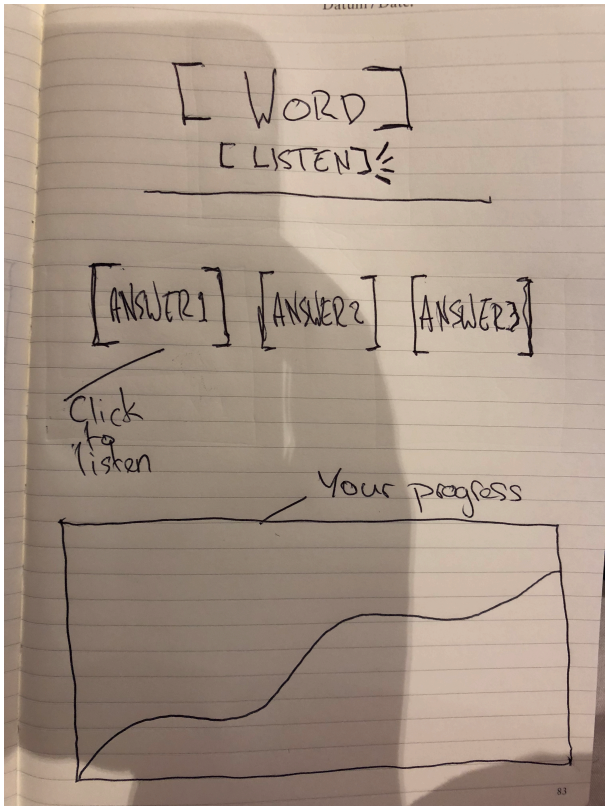


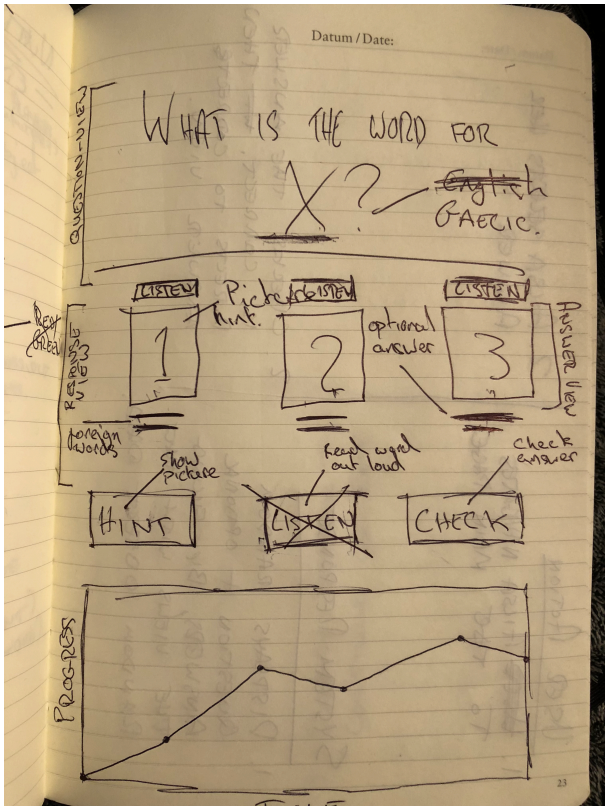
Unit	Ref	Evidence
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.
		Description: My spending tracker application which I worked on as a solo project. GitHub link: **https://github.com/benharvie/week_4_spending_tracker_project**



Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.
		Description: The first screenshots show the Trello board for initial planning and what we our MVP was, extensions, advanced extensions, what was in progress and in the backlog. Each screenshot showing different stages in the planning. Below that are two wireframe diagrams, the initial plan and then a more complexed and in-depth detailed plan including what the views were and subtle changes to how the design was going to be changed.



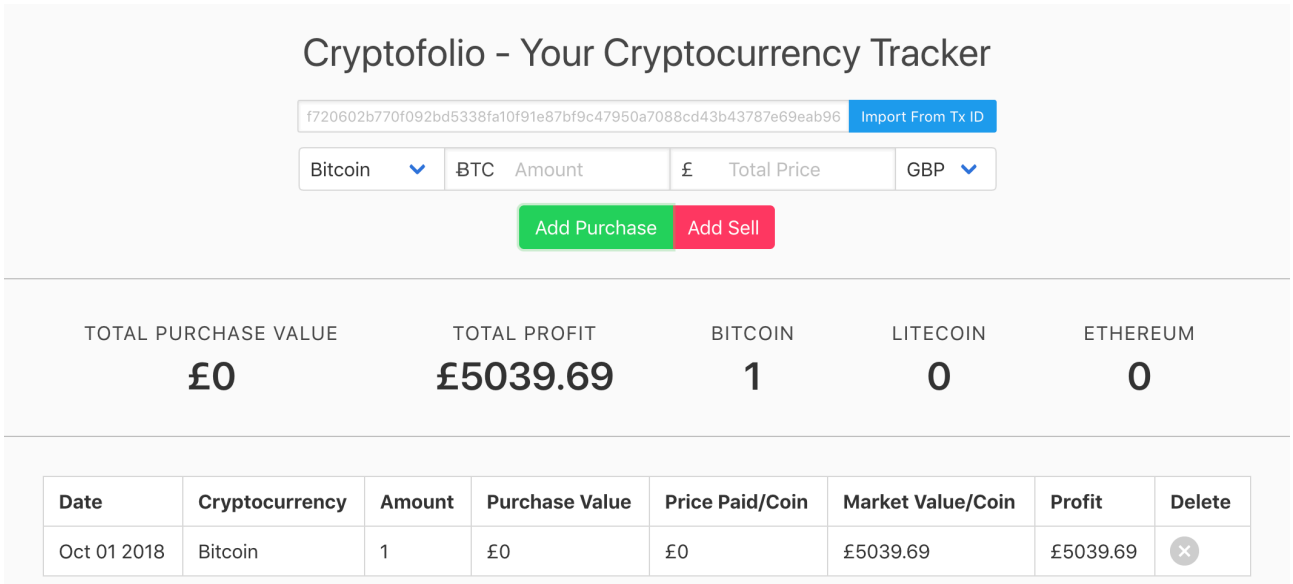




Week 7

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: This function takes the input Cyrptocurrency and utilises the CoinMarketCap.com API to find the relevant price as shown on the lower screenshot.

```
const getPrice = function(cryptoCurrency, globalCurrency) {
  // 1. Match seleted cryptoCurrency to it's given currencyID.
  // 2. Set up a request to the API.
  // 3. Send the API a request with the given currencyID.
  // 3. Get the price in the currency that matches the functions globalCurrency.
  // 4. Return the final price.

  let currencyID
  if (cryptoCurrency === 'Bitcoin') {
      currencyID = 1;
  } else if (cryptoCurrency === 'Litecoin') {
      currencyID = 2;
  } else if (cryptoCurrency === 'Ethereum') {
      currencyID = 1027;
  }

  let price
  const priceInfo = JSON.parse(httpGET(`https://api.coinmarketcap.com/v2/ticker/${currencyID}/?convert=GBP&limit=10/`));
  if (globalCurrency === 'USD') {
    price = roundToTwo(priceInfo.data.quotes.USD.price);
  } else if (globalCurrency === 'GBP') {
    price = roundToTwo(priceInfo.data.quotes.GBP.price);
  }
  return price;
}
```



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: The test code is testing if the bear has a name. The test codes check to see if “Yogi” is equal to the @bear.name method, after testing it shows that this is not the expected result. After adjusting the instantiation of the @bear’s name to the correct spelling, the test is then re-ran and the test code is passing.

```ruby
class BearTest < MiniTest::Test
  def setup
    @fish = ["Fishy", "George", "Coddy"]
    @river = River.new("Amazon", @fish)
    @bear = Bear.new("Yogy", @river)
  end

  def test_bear_has_name
    assert_equal("Yogi", @bear.name)
  end
```

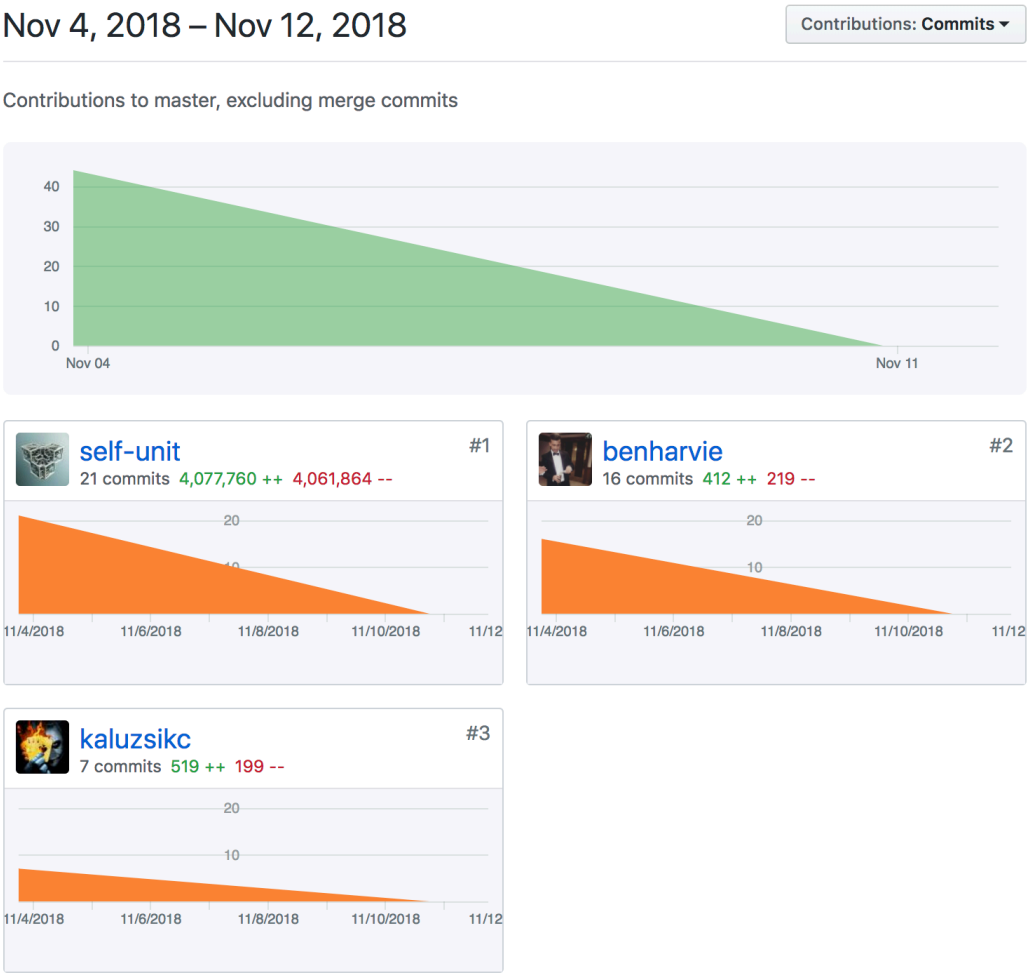
```
➜  specs git:(master) ruby bear_spec.rb
Run options: --seed 61354

# Running:

...F.

Finished in 0.001230s, 4065.0406 runs/s, 4065.0406 assertions/s.

  1) Failure:
BearTest#test_bear_has_name [bear_spec.rb:15]:
Expected: "Yogi"
  Actual: "Yogy"

5 runs, 5 assertions, 1 failures, 0 errors, 0 skips
```

```ruby
class BearTest < MiniTest::Test
  def setup
    @fish = ["Fishy", "George", "Coddy"]
    @river = River.new("Amazon", @fish)
    @bear = Bear.new("Yogi", @river)
  end

  def test_bear_has_name
    assert_equal("Yogi", @bear.name)
  end
```

```
➜  specs git:(master) ✗ ruby bear_spec.rb
Run options: --seed 43299

# Running:

.....

Finished in 0.002025s, 2469.1358 runs/s, 2469.1358 assertions/s.

5 runs, 5 assertions, 0 failures, 0 errors, 0 skips
```

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: This is the contributor's page on GitHub from the international price comparison app, including myself and two other contributors.



Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.
		Description: This is the project brief for the International price comparison app along with some of it's intended extensions.

Project #3 Brief

Summary

Full-stack application (Java/React) that allows users to search across Amazon's multiple international websites to compare prices on searched products. It will allow the user to compare prices between countries and how much money they could potentially save.

MVP

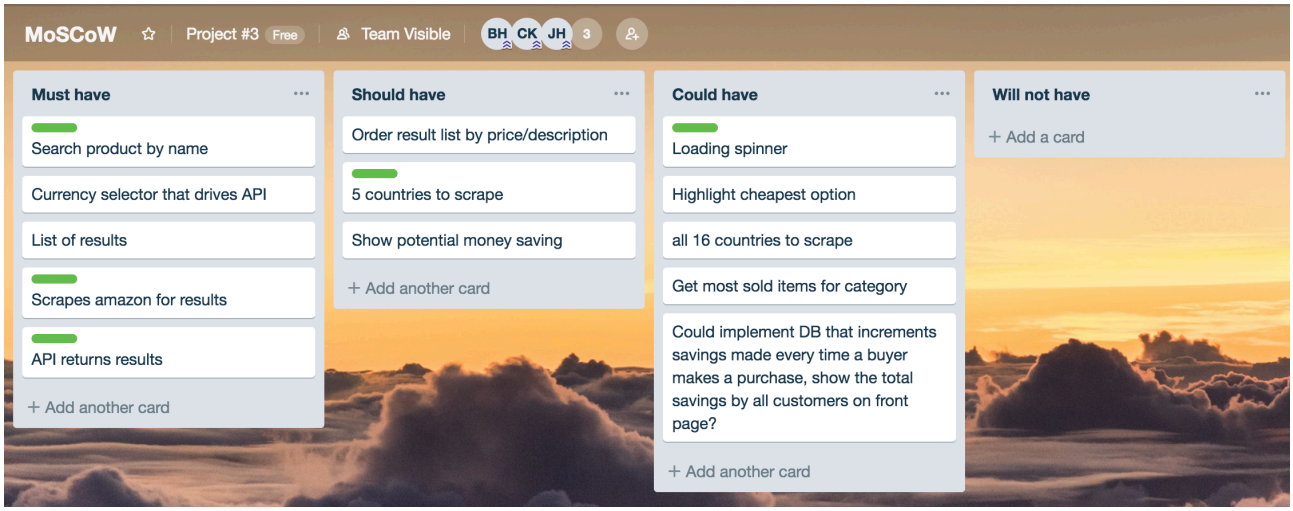
FRONT-END

- Search product input
- Display list of results (Country, Description, Rating, Currency, Price, Buy Button)
- Order by price/description [EXT]
- Highlight cheapest option [EXT]
- Currency selection [EXT]
- Grab Amazon's most sold items (from related category searched by user) [EXT]

BACK-END

- Scrapes each individual website (by ASIN)
- Returns an API with a list of product links, prices, description and image
- Show potential money saved (highest - lowest price) [EXT]

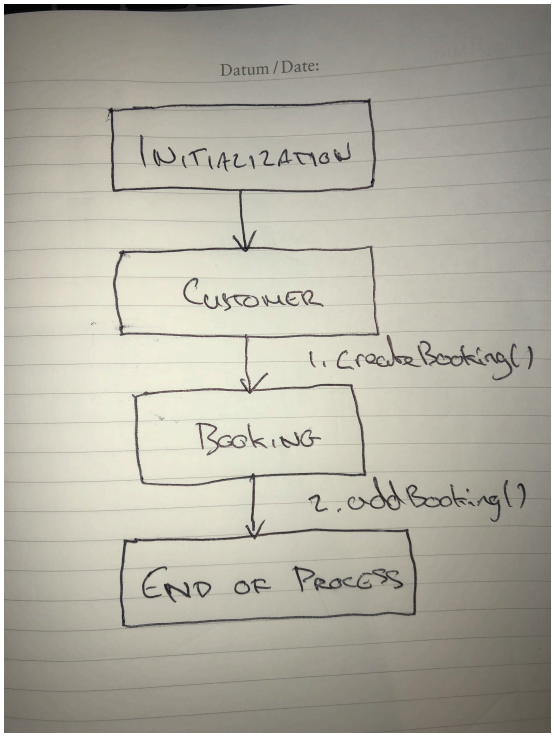
Unit	Ref	Evidence
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.
		Description: This was our Trello board for the price comparison app.

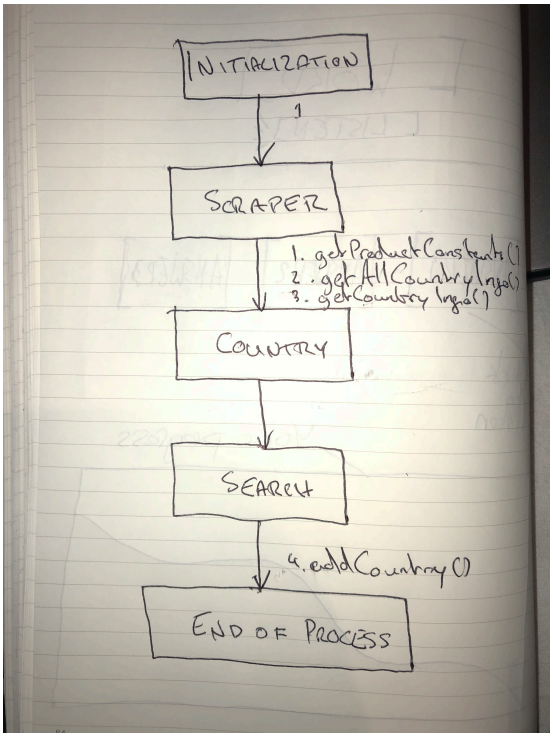


Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.

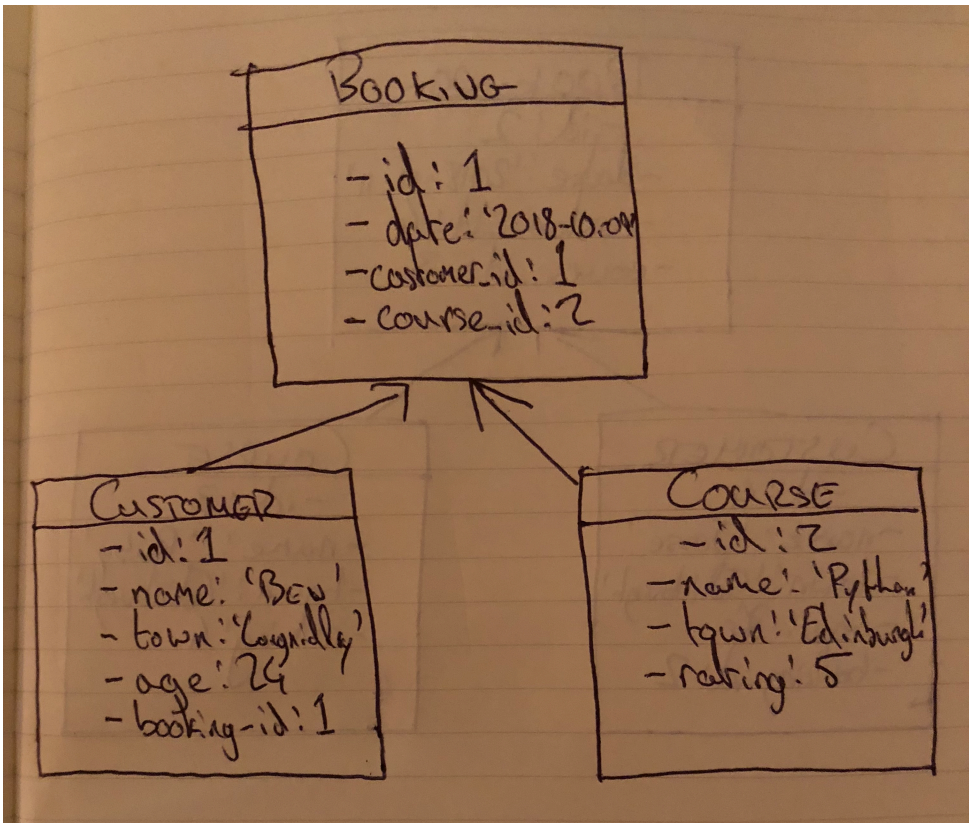
Acceptance Criteria	Expected Result	Pass/Fail
The function returns null when no product is found	Null return	Pass
The hasBadKeyword() function returns True when containing “did not match any products”	TRUE	Pass
The convertCurrency() method cleans the currency to a BigDecimal	A BigDecimal return with no symbols, commas, full-stops etc.	Pass
The convertDomainToLocale() method returns Locale.UK for “.co.uk" domain	Locale.UK	Pass
The getAllCountriesPrices() method should return prices for all countries in the array	10 prices	Fail, not all countries had the product in stock

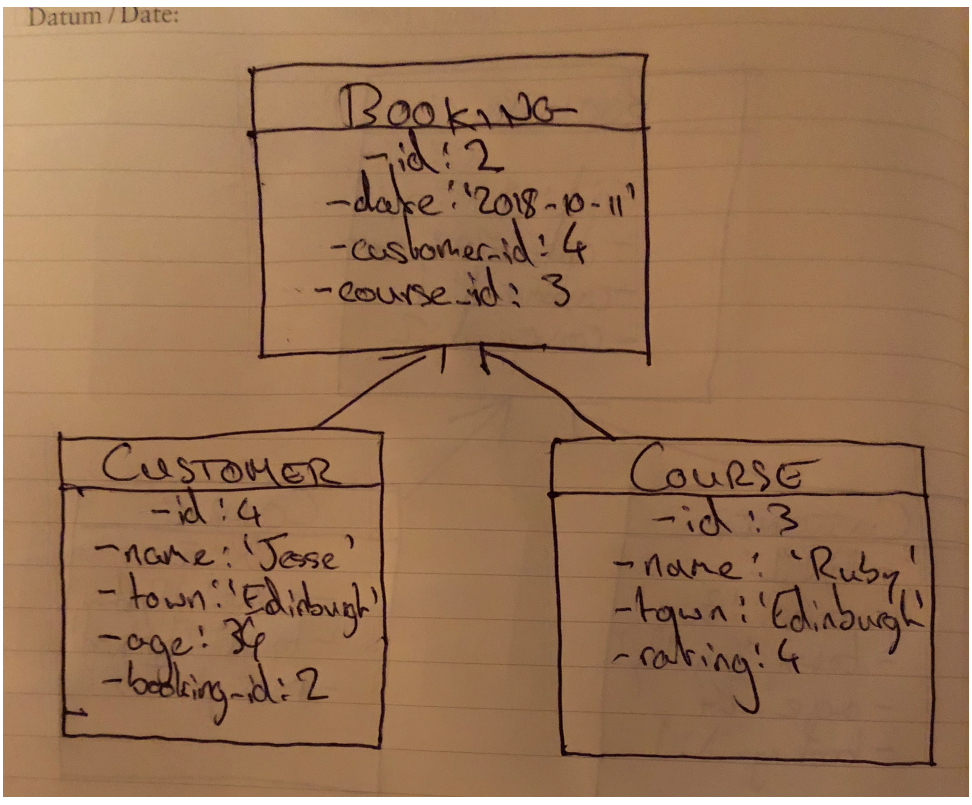
Unit	Ref	Evidence
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).
		Description: A collaboration diagram for a booking system and the price comparison project, showing the relation between the objects, methods and order that they are called .





Unit	Ref	Evidence
P	P.8	Produce two object diagrams.
		Description: These are two object diagrams from a Course Booking System, showing example objects and their relevant values and relations to one another.





Unit	Ref	Evidence
P	P.17	Produce a bug tracking report
		Description: This is a bug report from the final group project showing the bug/error, solution and the date.

Bug/Error	Solution	Date
Scraper was returning an irrelevant product	Debugged the code to find it was grabbing an advert rather than the product, filtered out the advert so it did not happen again	04/11/2018
convertDomainToLocale() function was not working for Spain	Converted the currency to another country that also used the same format (Euros)	05/11/2018
Some products could not be found and would return null	Added a check to see if the product existed and handled it accordingly	05/11/2018
Remote host was terminating the handshake in the web request causing the server to crash/error	Created a Try/Catch to catch when this error occurred so that the app could continue without crashing	06/11/2018
Some countries would not have the products price available and this would cause a crash/error	Created a check to see if a price was available before trying to parse it from the page	07/11/2018

Week 12

Unit	Ref	Evidence
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.
		The screenshots below show polymorphism being used within a Music Shop project, where the Shop class has an ArrayList of ISell named 'stock', the Instrument class then implements this interface of ISell. The Guitar class shows the Instrument sub class being extended from which takes in all it's properties and methods (and implements the ISell interface).

Shop.java

```java
package musicShop;

import interfaces.ISell;
import java.util.ArrayList;

public class Shop {
    private String name;
    private ArrayList<ISell> stock;
    private int cash;

    public Shop(String name, int cash) {
        this.name = name;
        this.cash = cash;
        this.stock = new ArrayList<>();
    }

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

    public int getCash() {
        return cash;
    }

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

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

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

Instrument.java

```java
package musicShop;

import interfaces.IPlay;
import interfaces.ISell;

public abstract class Instrument implements IPlay, ISell {
    private String colour;
    private String type;
    public int buyPrice;
    public int sellPrice;

    public Instrument(String colour, String type, int buyPrice, int sellPrice) {
        this.colour = colour;
        this.type = type;
        this.buyPrice = buyPrice;
        this.sellPrice = sellPrice;
    }

    public String getColour() { return this.colour; }

    public String getType() { return this.type; }

    public int getBuyPrice() { return this.buyPrice; }

    public int getSellPrice() { return this.sellPrice; }
}
```

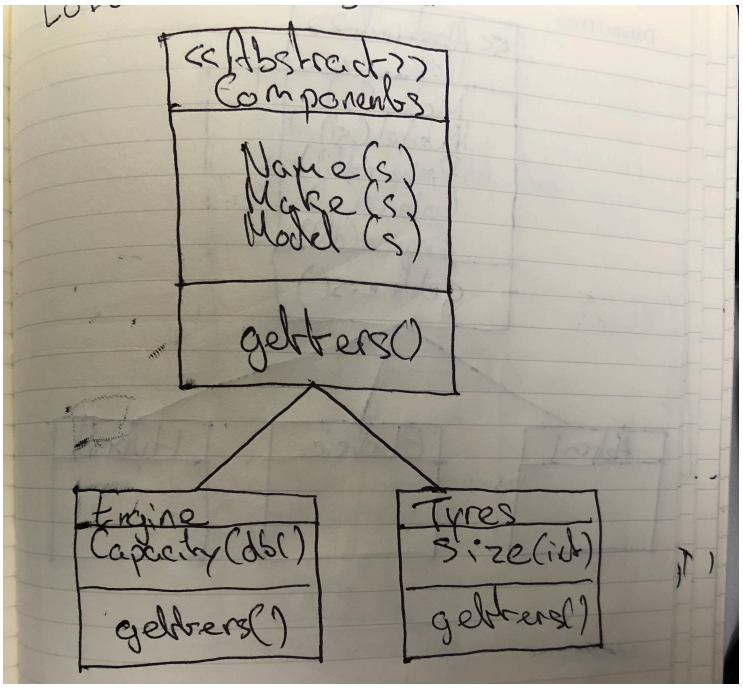
ISell.java

```java
package interfaces;

public interface ISell {
    int calculateProfit();
}
```

Guitar.java

```java
package musicShop;

public class Guitar extends Instrument  {
    private String model;

    public Guitar(String model, String colour, int costPrice, int rrpPrice) {
        super(colour,  type: "String", costPrice, rrpPrice);
        this.model = model;
    }

    public String play() { return "DumDeeDum"; }

    public String getModel() { return this.model; }

    public int calculateProfit() { return this.sellPrice - this.buyPrice; }

}
```

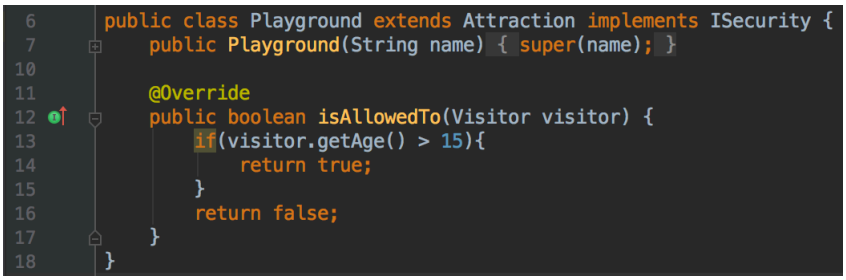
Unit	Ref	Evidence
A&D	A.D.5	An Inheritance Diagram
		Description: An inheritance diagram showing an abstract Components class passing it's methods down to Engine and Tyres class through the use of inheritance. Methods include Name/Make/Model getters, then the Engine class having it's own individual getter for the Capacity property and Tyre having it's individual getter for it's Size property.



Unit	Ref	Evidence
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.
		Description: This is a class that declares private variables, meaning that these particular variables have restricted access to just within the class itself.

```
14      public class Scraper {
15          private String searchName;
16          private String productName;
17          private String image;
18          private String ASIN;
19          private double rating;
20
21  @       public Scraper(String searchName) {
22              this.searchName = searchName.replace( target: " ", replacement: "%20");
```

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: The Playground class extends (inherits) from the Attraction class, meaning it takes in all of the same methods as the Attraction class. The 3rd screenshot shows a new Playground being instantiated, a new Visitor being instantiated and calling the isAllowedTo() method from Playground on the Visitor object to see if the Visitor is allowed on the Playground.



```
public class Playground extends Attraction implements ISecurity {
    public Playground(String name) { super(name); }

    @Override
    public boolean isAllowedTo(Visitor visitor) {
        if(visitor.getAge() > 15){
            return true;
        }
        return false;
    }
}
```

```
public abstract class Attraction {

    private String name;
    private ArrayList<Visitor> visitors;

    public Attraction(String name){
        this.name = name;
        this.visitors = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public void addVisitor(Visitor visitor){
        visitors.add(visitor);
    }

    public ArrayList<Visitor> getVisitors(){
        return visitors;
    };
}
```

```
public class PlaygroundTest {

    Playground playground;
    Visitor visitor;

    @Before
    public void before() {
        playground = new Playground( name: "FunFunFun");
    }

    @Test
    public void visitorUnderageNotAllowed(){
        visitor = new Visitor( age: 12, height: 5.5, money: 24.50);
        assertEquals( expected: false, playground.isAllowedTo(visitor));
    }
}
```

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: The first screenshot shows a function that will return the count of distinct case-insensitive alphabetic characters and numeric digits that occur more than once in the input 'text' string. The second screenshot shows a function that takes in a number, the function will then square each individual digit of that given number and concatenate it, returning it as an integer.

```ruby
def duplicate_count(text)
  count = 0
  text.upcase!
  original = text.split('')
  unique = text.split('')
  unique.uniq!
  for i in 0..unique.length
    count += 1 if original.count(unique[i]) > 1
  end
  return count
end
```

```javascript
function squareDigits(num){
  value = ''
  String(num).split('').map(x => Number(x)**2).forEach(x => value += String(x))
  return Number(value)
}
```