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:

Paste Screenshot here

```ruby
def reverse_array
  number_array = ["one", "two", "three", "four", "five"]
  print number_array.reverse
end
```

```
→ work_files ruby rubyArray.rb
["five", "four", "three", "two", "one"]%
```

Description here

Screenshot of reverse_array method which reverses an array of five strings within the function.
Second screenshot showing output of the method

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:

Paste Screenshot here

```ruby
word_hash = Hash["one" => "ruby", "two" => "emerald", "three" => "sapphire"]

def print_middle_hash(word_hash, key)
  print word_hash.fetch(key)
end

print_middle_hash(word_hash, "two")
```

```
➜  work_files ruby rubyArray.rb
emerald%
```

Description here

Screenshot of word_hash method which returns the value of the key given as a parameter.
Second screenshot showing output of word_hash method

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

Screenshot showing find_by_id method which takes in an id and searches for the entry matching the id

```ruby
def self.find_by_id(id)
  sql = "SELECT * FROM customers
  WHERE id = $1"
  values = [id]
  result = SqlRunner.run(sql, values)
  final_result =  Customer.map_items(result)
  return final_result if final_result.count > 0
  return nil
end
```

```
def test_find_by_id()
  Customer.find_by_id(16)
end
```

Screenshot showing result of find_by_id method running taking in the id of 16

```
[#<Customer:0x007fc6f7118e18 @name="Mike Smith", @funds=488, @tickets=0, @id=16>]
```

Description here

Class method that returns an entry to a postgreSQL by the id of the entry

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 sort_animal_array(array)
   print array.sort()
end
```

```
["cat", "dog", "horse", "rabbit"]%
```

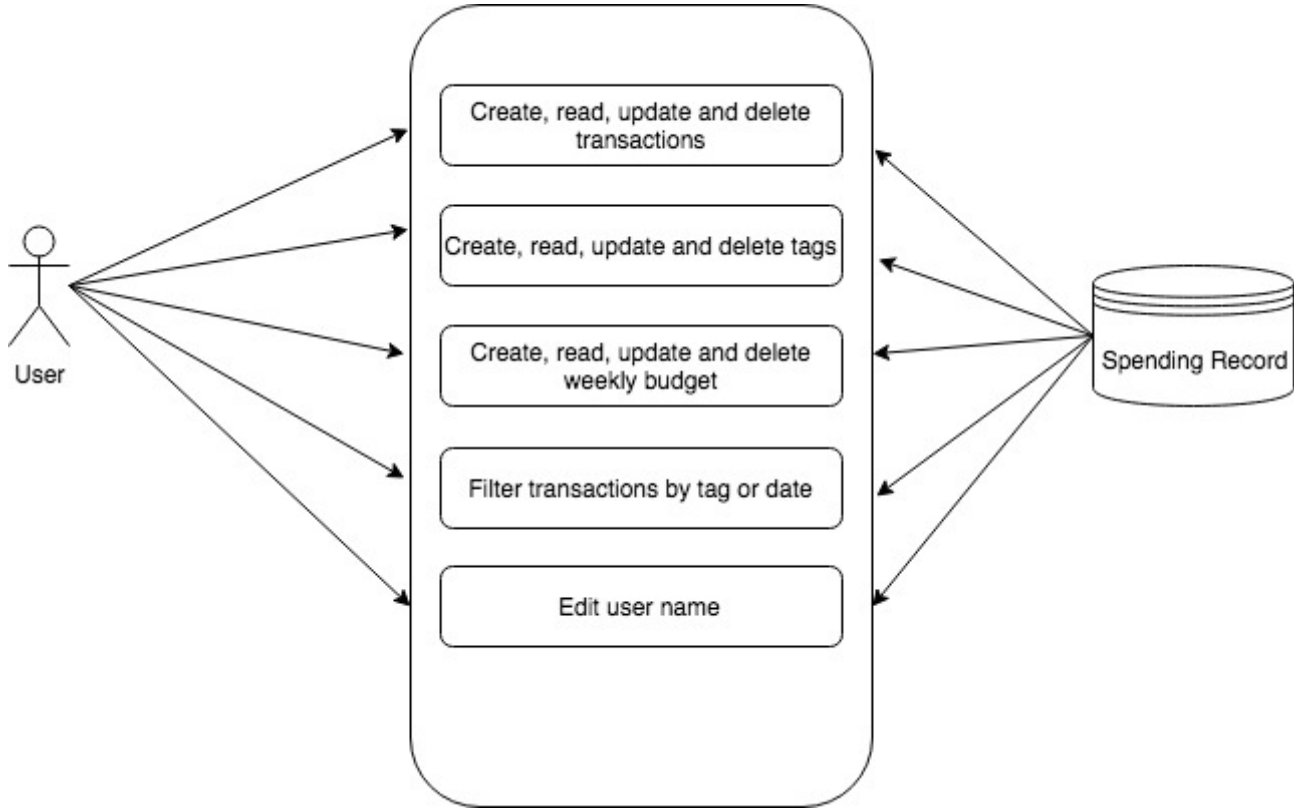
Description here

Screenshot of a sort_animal_array method which takes in an array and sorts the data within the array alphabetically. The second screenshot shows the result of running sort_animal_array.

Week 5 and 6

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

Paste Screenshot here

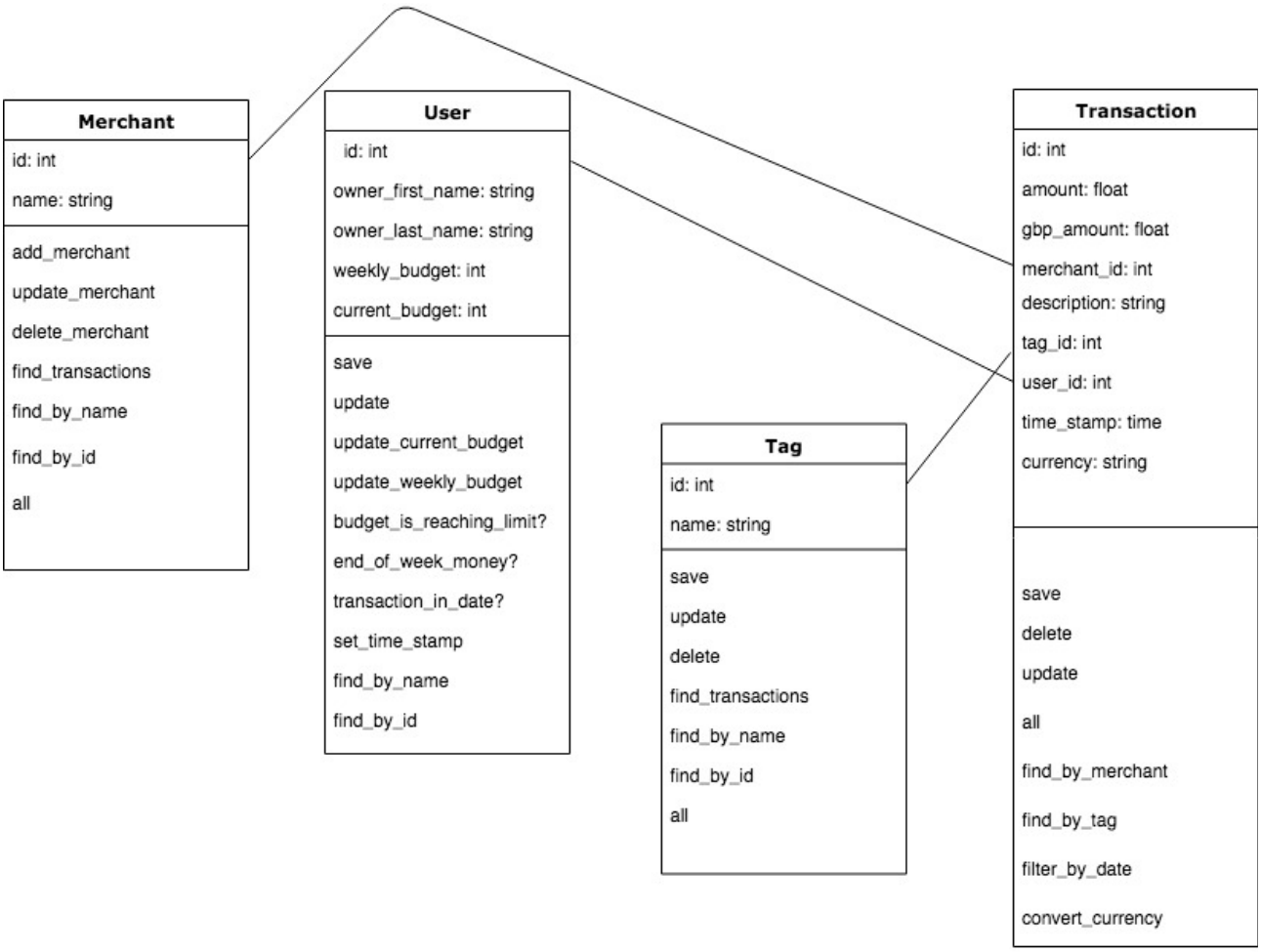


Description here

Use case diagram for spending tracker ruby app

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

Paste Screenshot here

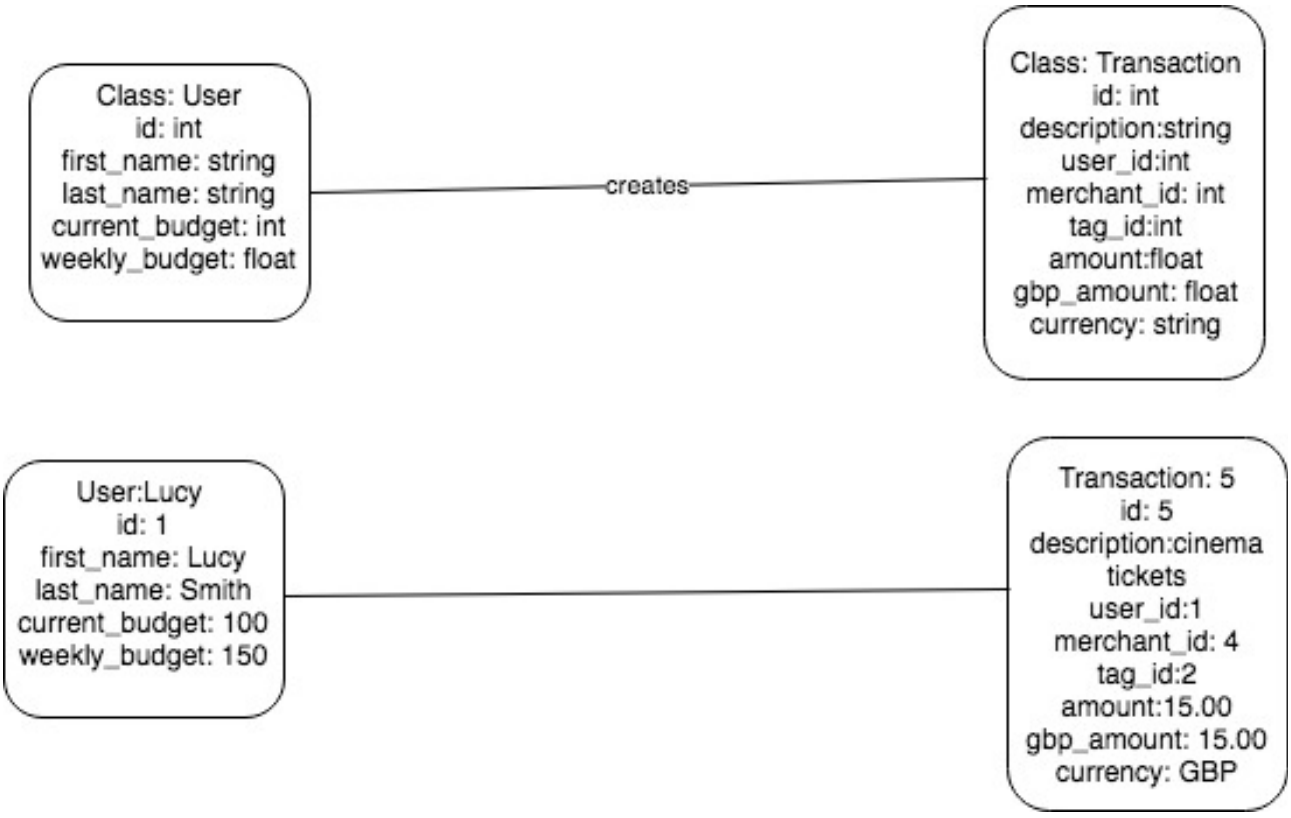


Description here

Class diagram for ruby money spending track app showing the relationships between tag to transaction, transaction to user, transaction to merchant.

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

Paste Screenshot here

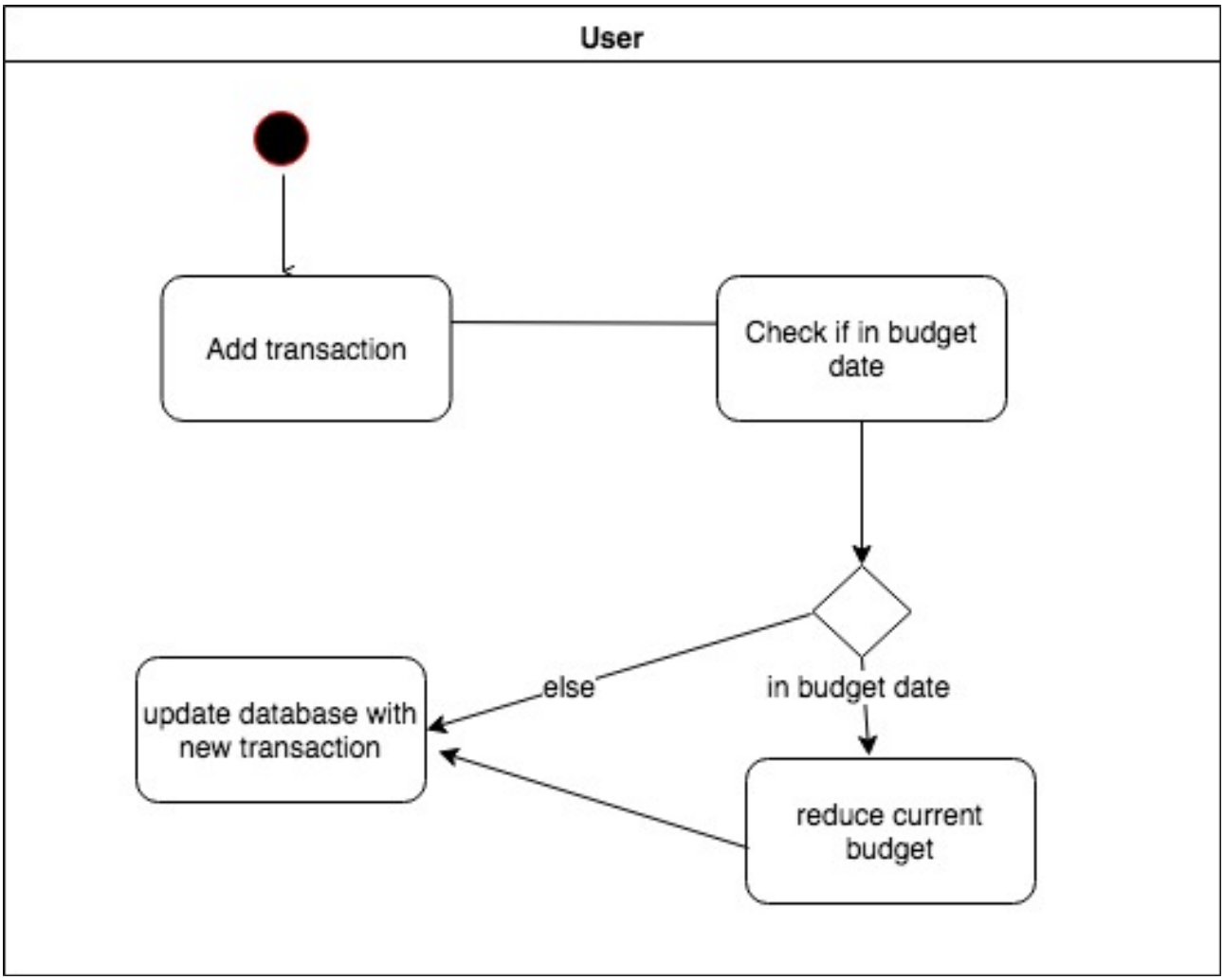


Description here

Object diagram showing instances of the user and transaction class

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

Paste Screenshot here



Description here

Activity diagram showing the action of user adding a transaction within money spending app.

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

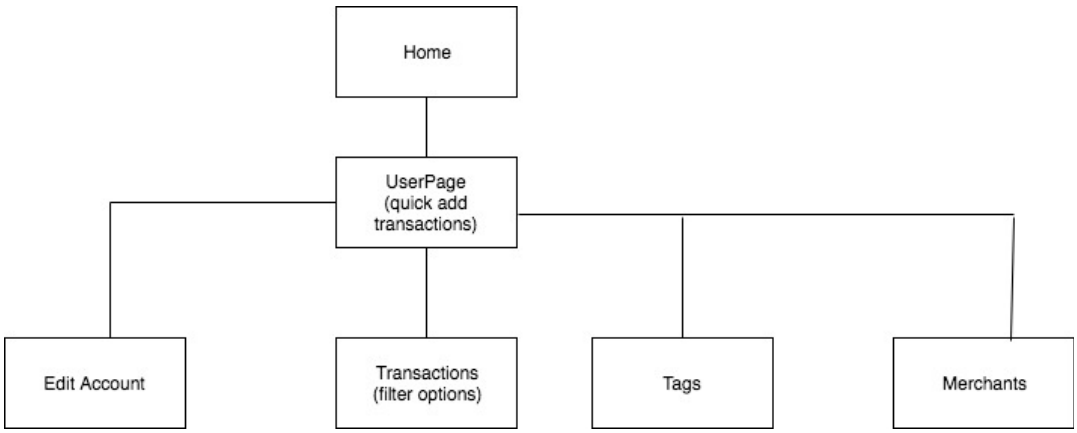
Constraint Category	Implementation Constraint	Solution
Hardware and software platforms	-User may not have server used to run app	-make sure app is useable across different devices and investigate cloud solutions
Performance requirements	-Lightweight app but large numbers of transactions could crash/slow down app	-MVP for light weight app but look to research other options as popularity grows
Persistent storage and transactions	-The money spending app requires a database for transaction information to be stored. Database methods must be able to be used on different devices in order for app to work	-App tested thoroughly on different devices
Usability	- It must be easy to add new transactions, see past spending and budget left so the app is used regularly and users enjoy using it	-planning to include UX design with user journeys and stories to help developers put the user first.
Budgets	-Limited budget given which must be stuck to.	-Create a realistic MVP first making sure that a basic product can definitely be created within the budget
Time	-Only one week to complete project. Non- negotiable deadline.	-Stick to a tight MVP first and then work on extensions when that has been delivered

Description here

Implementations constraints plan for spending tracker app

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

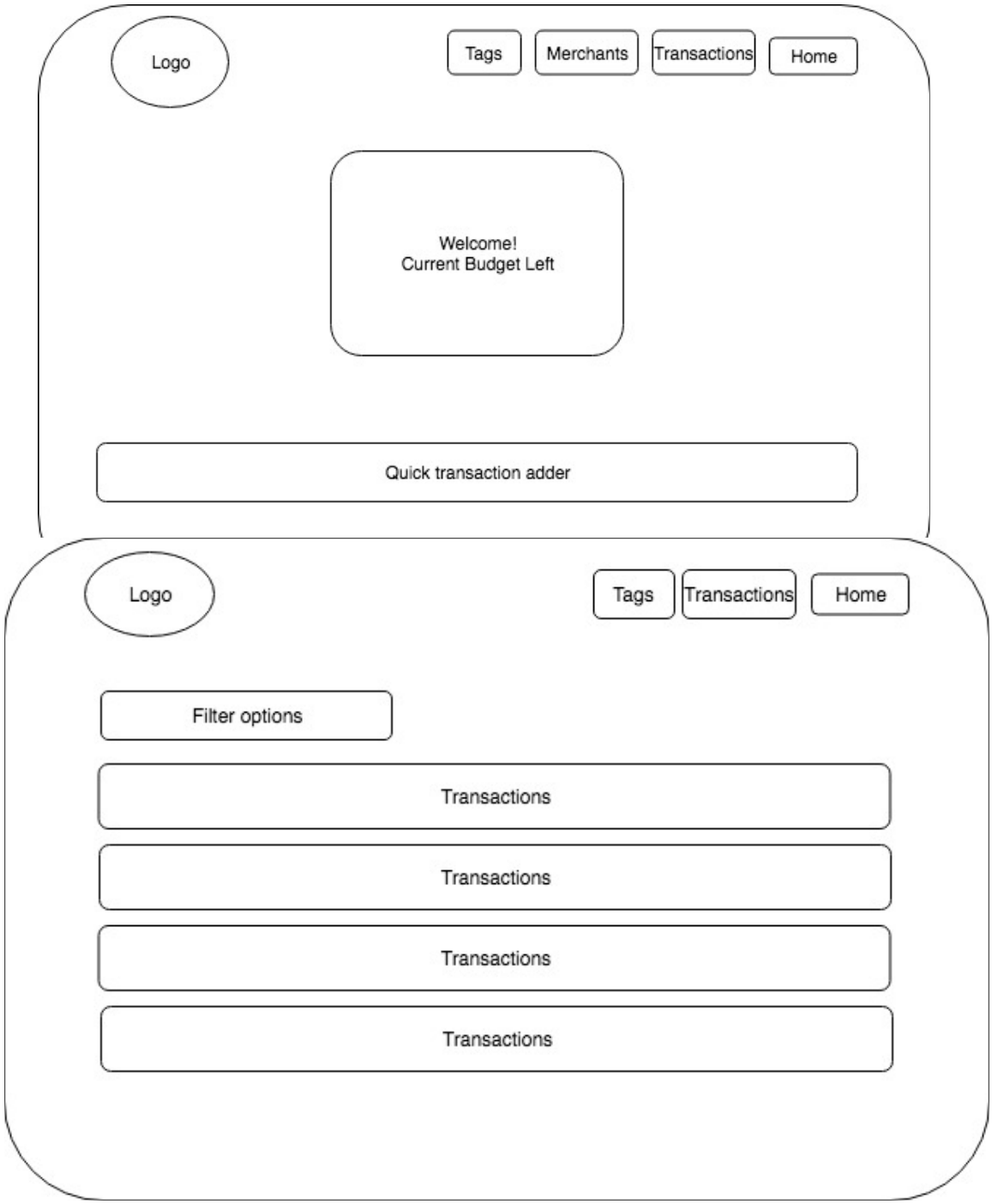
Paste Screenshot here



User site map of money spending tracker app

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

Paste Screenshot here

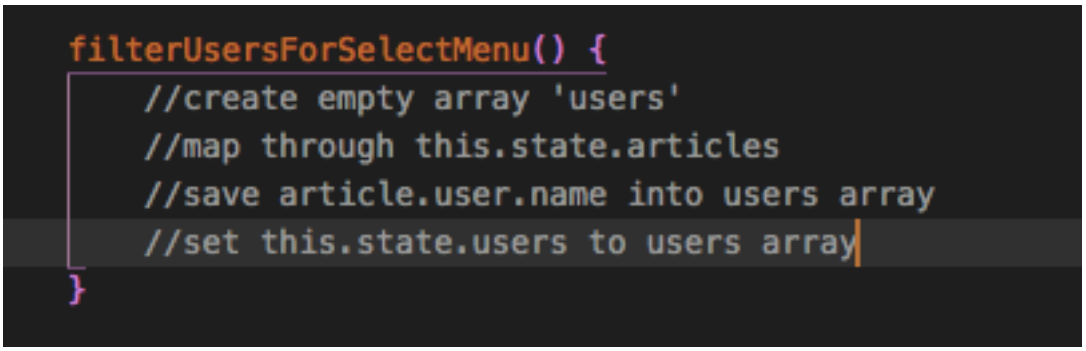


Description here

Two wireframe documents showing initial plans for user home and transaction page

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

Paste Screenshot here



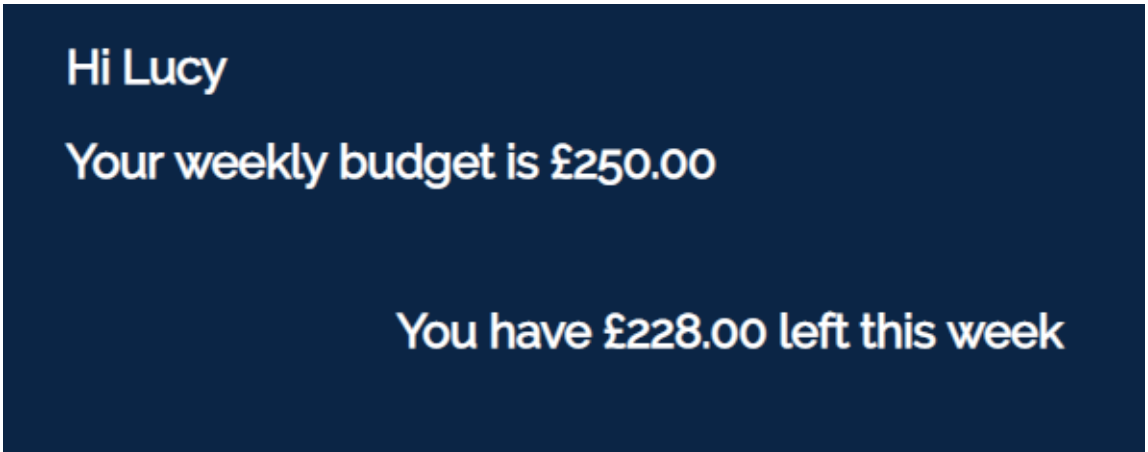
Description here

Pseudocode which talks through a method to get all the articles authors' name and setting this.state.users to the these names.

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

Screenshot showing transaction being inputted of £22

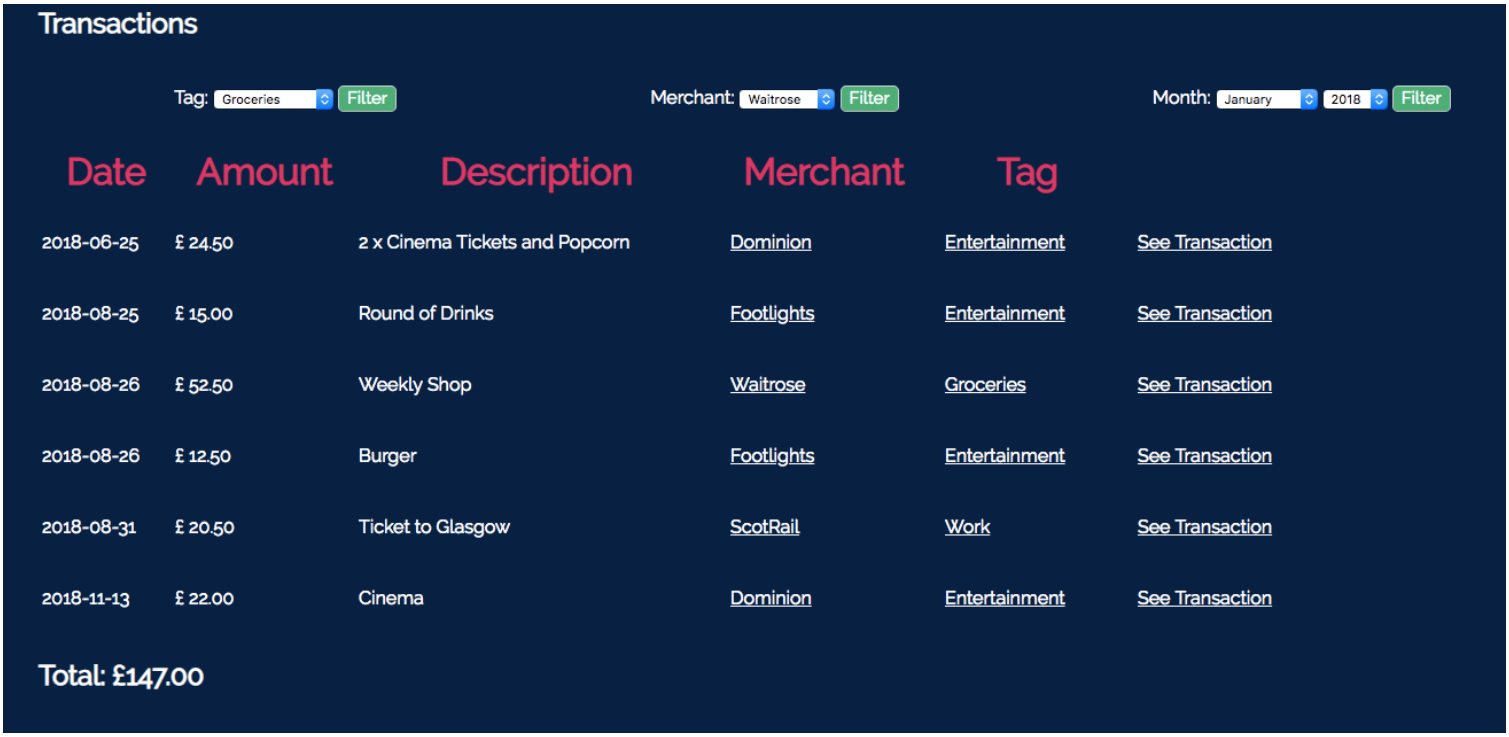


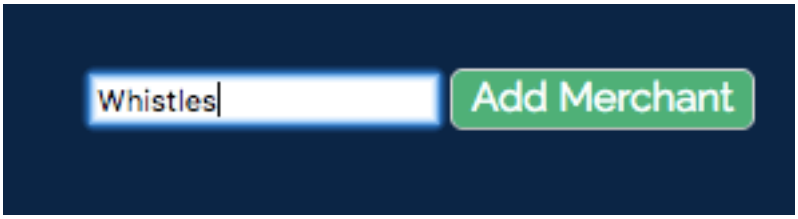
Screenshot showing the £22.00 taken off the weekly budget

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

Screenshot showing transaction of £22 being spent at the cinema with the merchant being Dominion



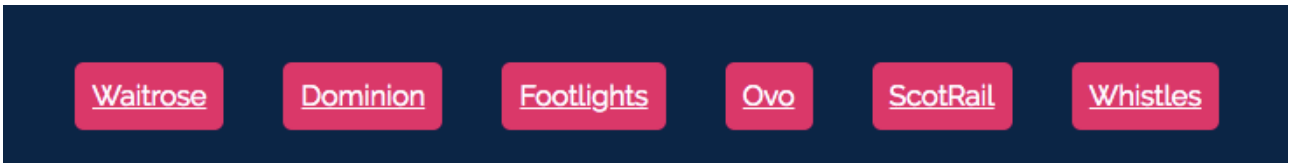


Screenshot transaction being saved in the full transaction page at the bottom of the table

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

Screenshot showing user requesting Whistles added as a merchant



Screenshot showing Whistles added to the merchants, having been processed by the program

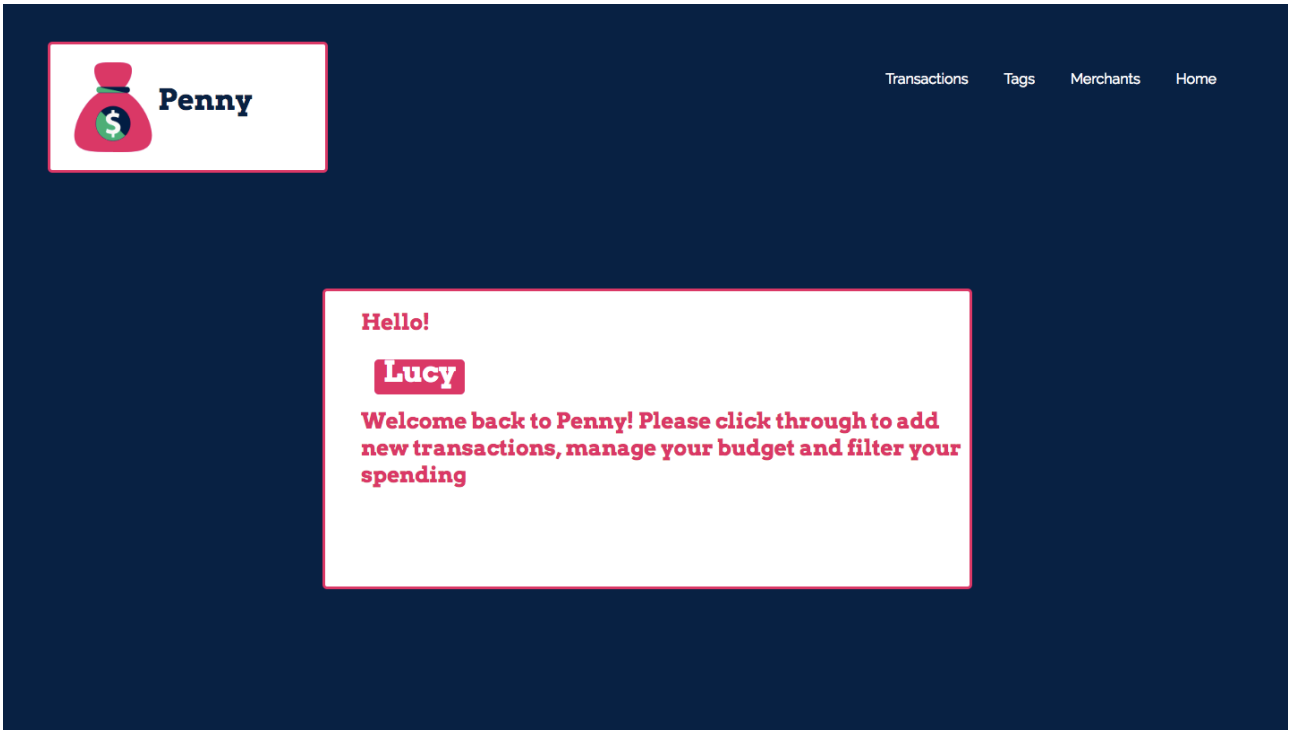
Unit	Ref	Evidence
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.
		Description:

Paste Screenshot here

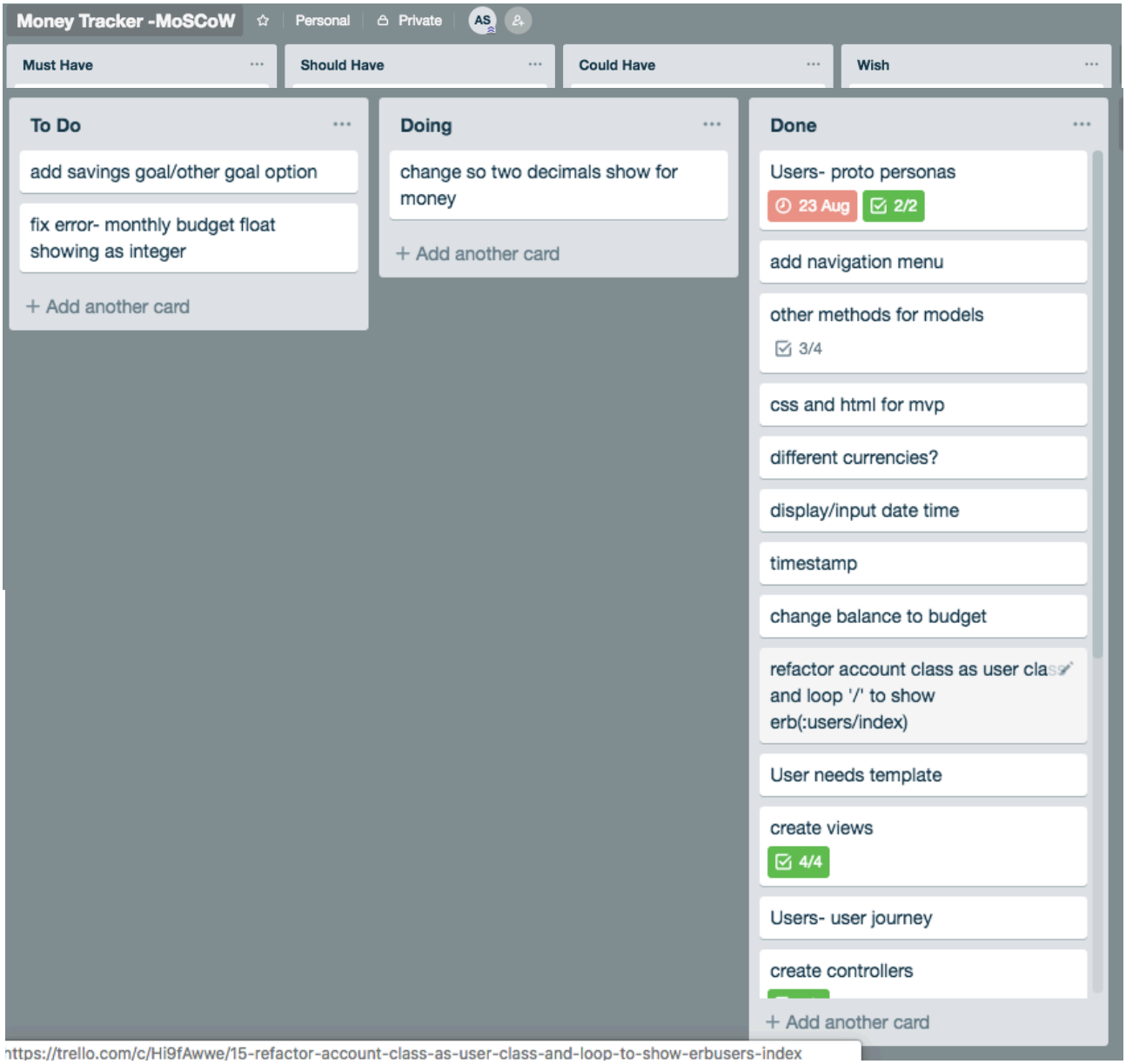
Screenshot of the first page of Penny, solo money tracker project

GitHub link: https://github.com/AlisonBabington/PennyMoneyTracker

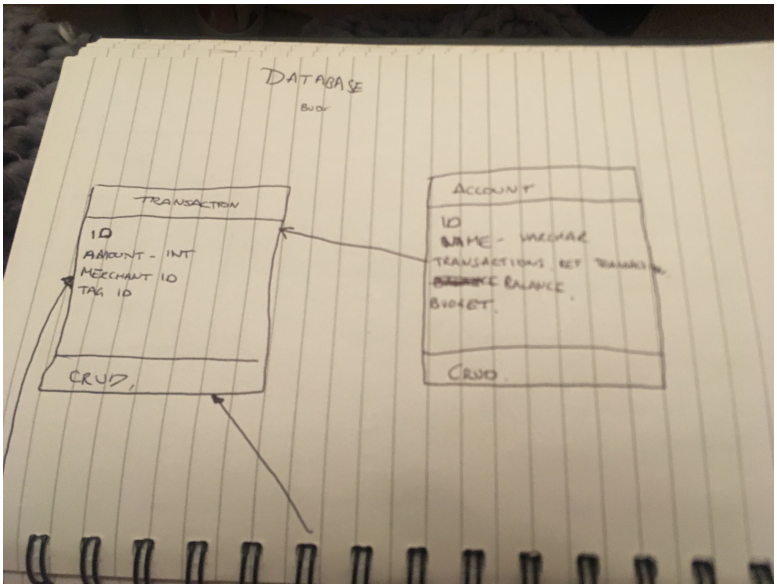
Unit	Ref	Evidence
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.
		Description:



Paste Screenshot here



Screenshot showing MoSCoW board from Penny project



I want to:
See how much we spe
on household things

So that:
Can check how we've split thi

John Smith

As a:
person who is not used
to bills + other financial
responsibilities.
I want to:
see clearly what my
budget is
So that:
I dont run out of money!

As a:

Screenshot showing user needs

Penny

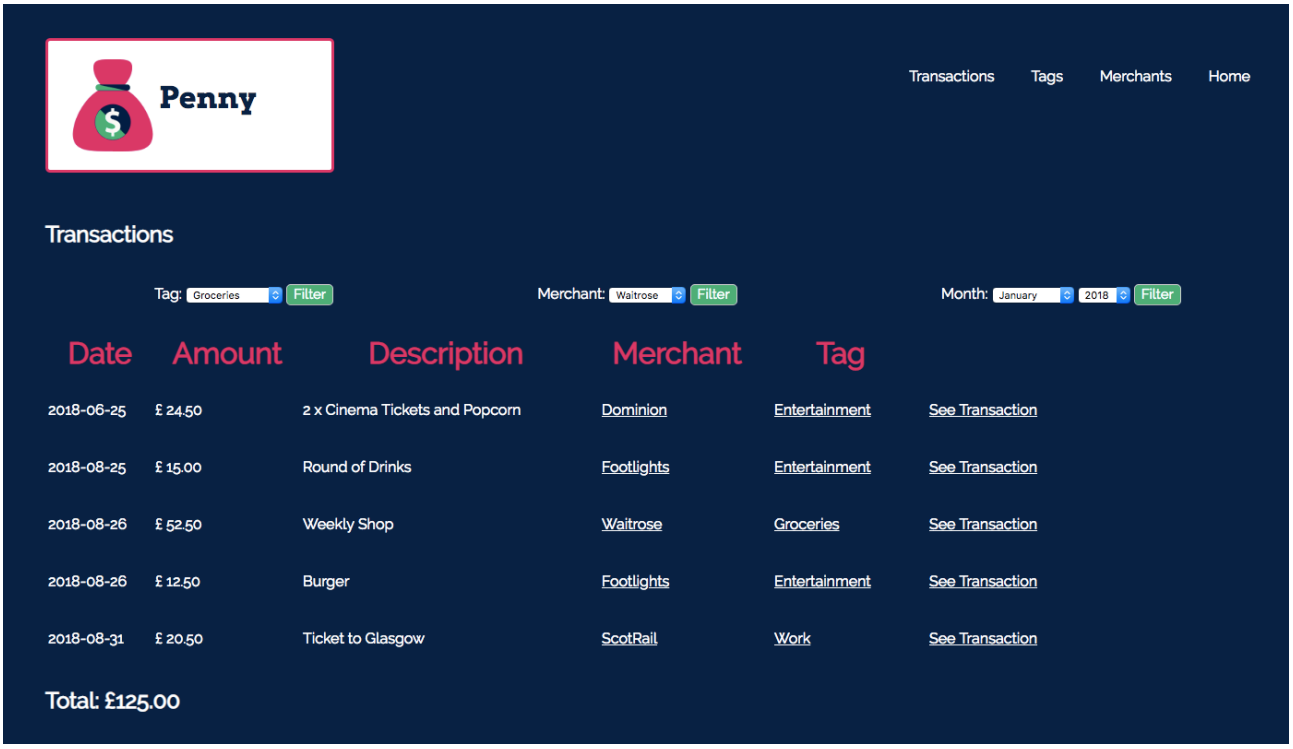
Transactions

Tag: Groceries Filter
Merchant: Waitrose Filter
Month: January 2018 Filter

Date	Amount	Description	Merchant	Tag	
2018-06-25	£ 24.50	2 x Cinema Tickets and Popcorn	Dominion	Entertainment	See Transaction
2018-08-25	£ 15.00	Round of Drinks	Footlights	Entertainment	See Transaction
2018-08-26	£ 52.50	Weekly Shop	Waitrose	Groceries	See Transaction
2018-08-26	£ 12.50	Burger	Footlights	Entertainment	See Transaction
2018-08-31	£ 20.50	Ticket to Glasgow	ScotRail	Work	See Transaction

Total: £125.00

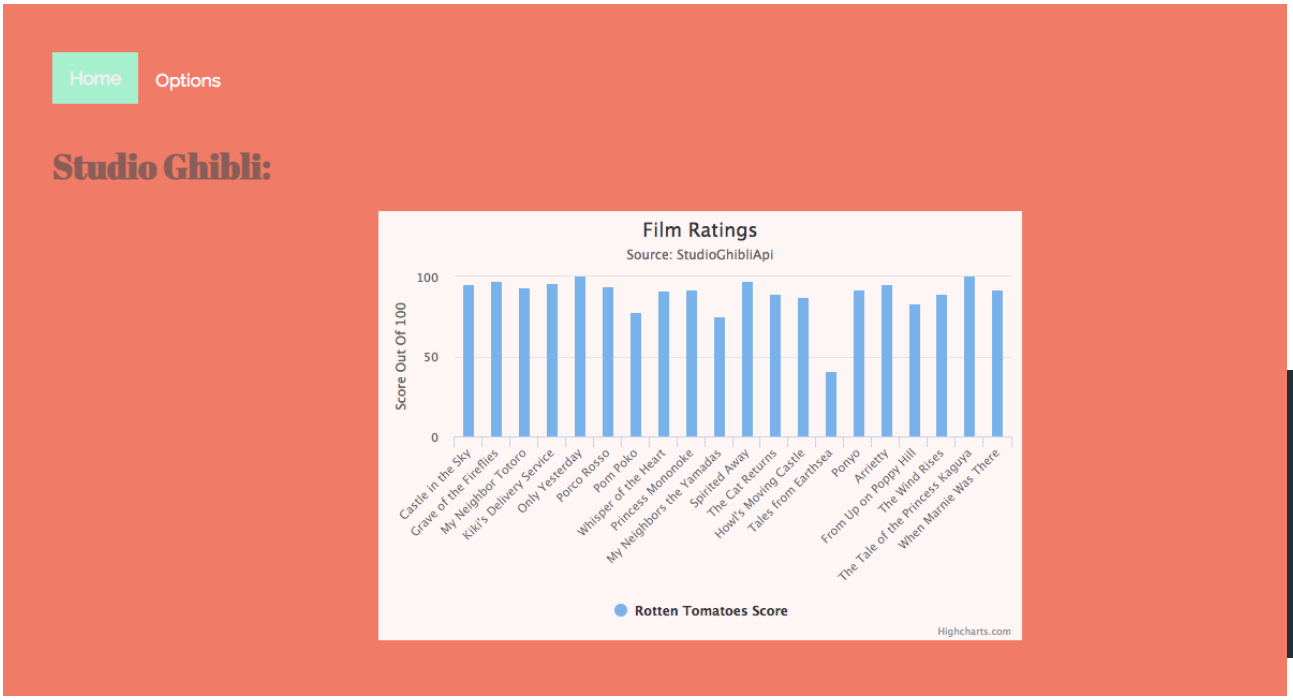
Screenshot showing transaction table before css styling



Screenshot showing transaction table after styling with nabber and logo also added

```
Ghibli.prototype.getData = function ( url, channel) {
  const request = new Request(url)
request.get()
  .then((data) => {
    PubSub.publish(channel, data);
  })
  .catch((err) => {
   console.error(err);
 });
};
```

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

Screenshot showing method which sends fetch request for data from Studio Ghibli api

Screenshot showing film rating chart plotting the rotten tomatoes score data received from the studio Ghibli app

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:

Paste Screenshot here

```
@Test
public void canAddPlanes() {
    airport.buyPlane(plane1);
    assertEquals(1, airport.checkHangerSize());
    airport.buyPlane(plane2);
    assertEquals(2, airport.checkHangerSize());
}
```

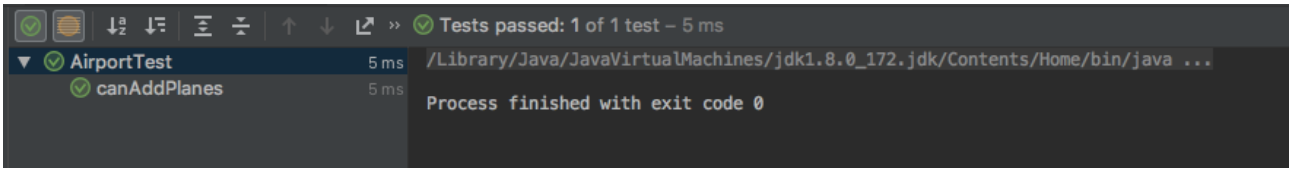
```
public void buyPlane(Plane plane) {

}
```

Screenshot showing test code for buyPlane method

```
AirportTest 17 ms
  canAddPlanes 17 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_172.jdk/Contents/Home/bin/java ...

java.lang.AssertionError:
Expected :1
Actual   :0
 <Click to see difference>

<1 internal call>
    at org.junit.Assert.failNotEquals(Assert.java:834) <2 internal calls>
    at AirportTest.canAddPlanes(AirportTest.java:26) <23 internal calls>
```

Screenshot showing test failing for buyPlane method

```
public void buyPlane(Plane plane) {
    hanger.add(plane);
}
```

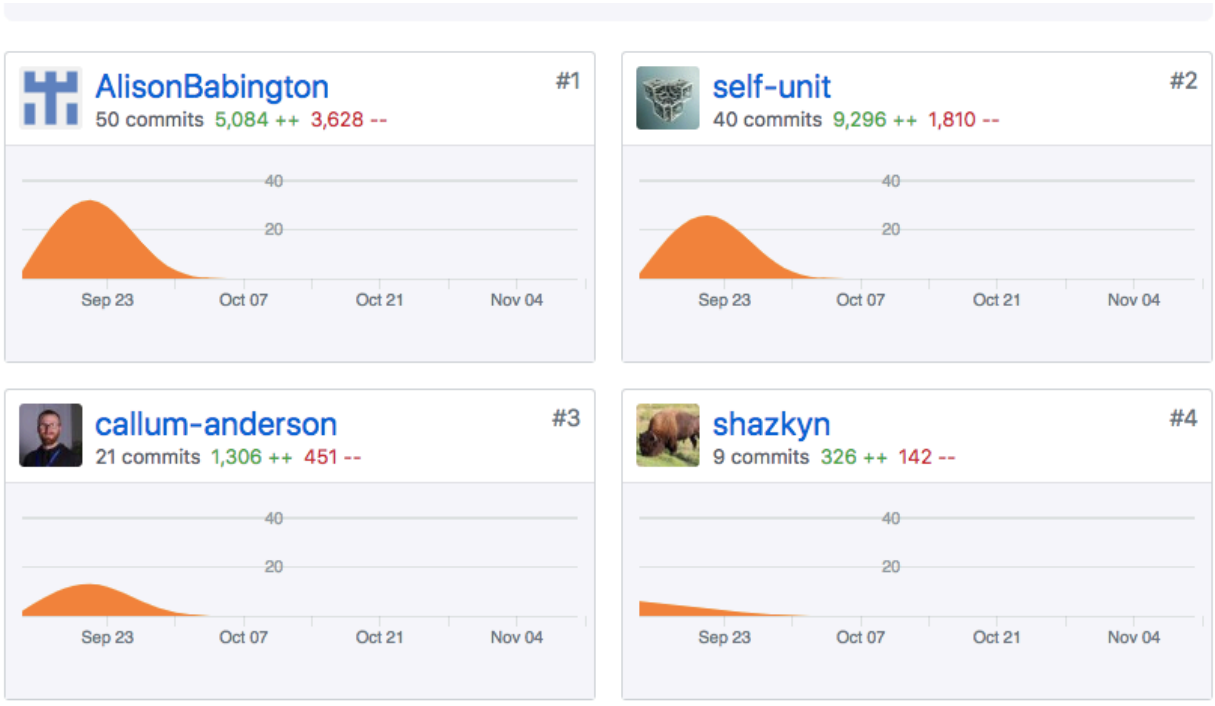


Screenshot showing buyPlane method corrected

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:

Screenshot showing test for buyPlane method now passing

Week 9



Github repo's contributor page from group project showing 4 contributors including me

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

Paste Screenshot here

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.

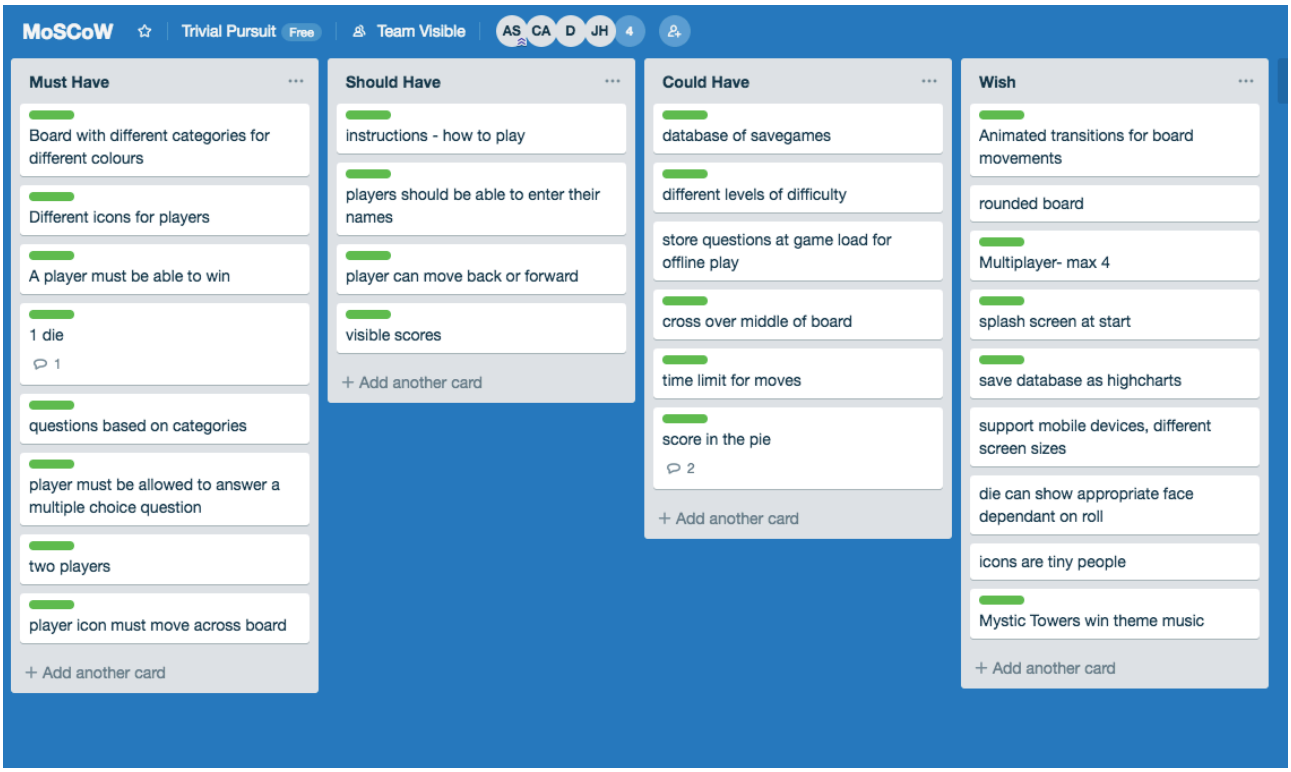
A user should be able to enter their name.

Description here

MVP and extensions brief from the JavaScript group project

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

Paste Screenshot here



Description here

MoSCoW Trello board used for the planning in the JavaScript group project.

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

Paste Screenshot here

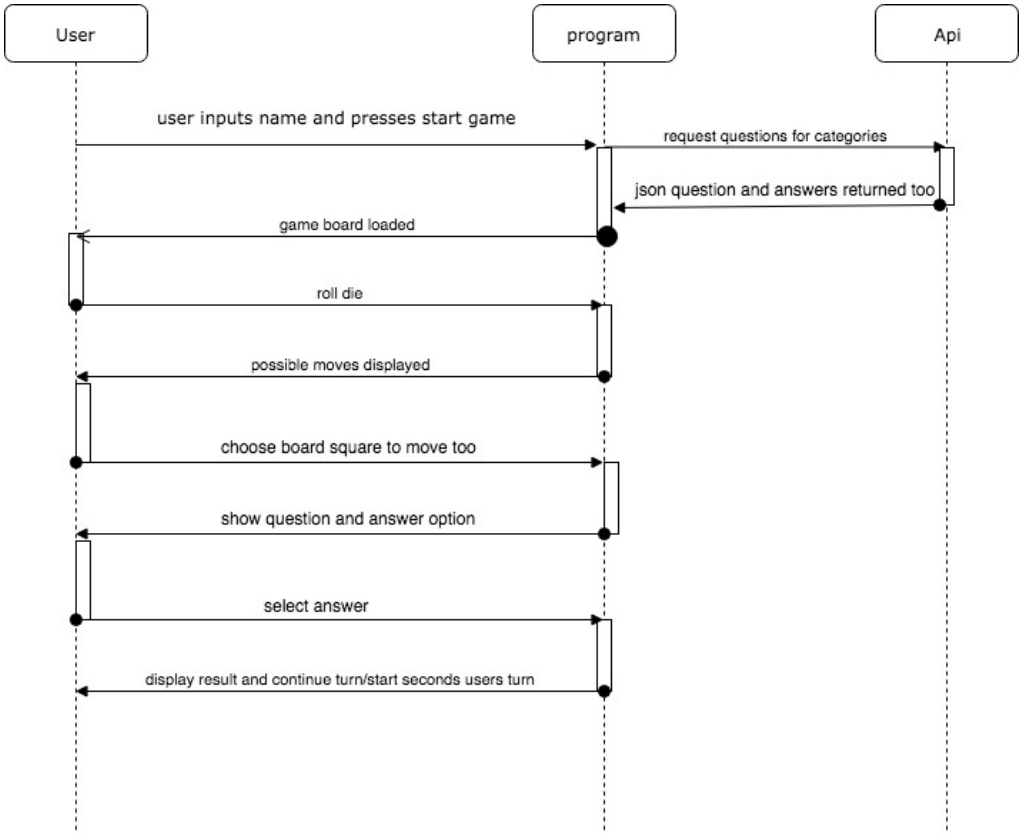
Acceptance criteria	Expected Outcome	Pass/Fail
A user is able to submit articles	The database saves new articles when they are submitted	Pass
A user is able to edit articles	The database edits the article entry when the user edits them	Pass
A user is able to filter articles	A filtered list of articles is shown when the filters are selected	Pass
A user is able to securely log in	The program securely checks authentication when a user logs in	Fail
A user is able to create a new account	The database saves a new user into the database when it is created	Pass

Description here

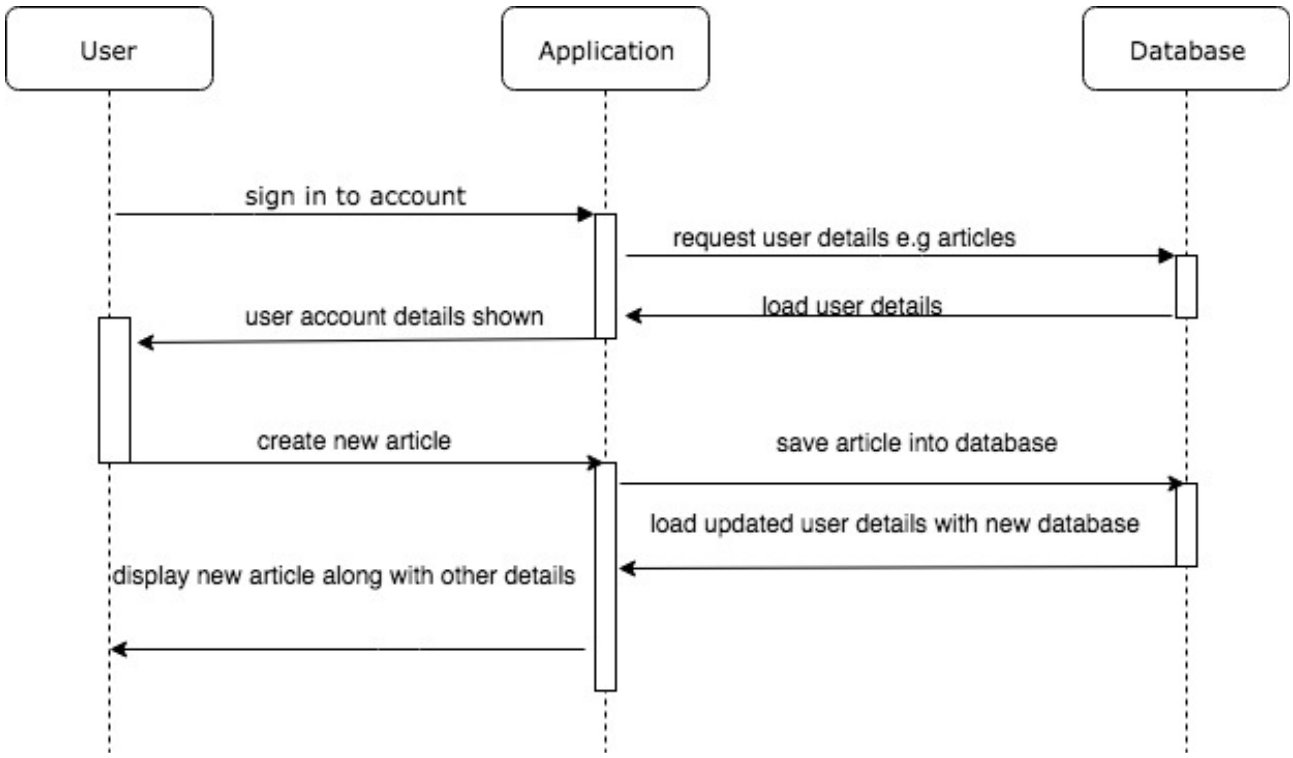
Screenshot showing acceptance criteria for CMS app

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

Paste Screenshot here



Sequence diagram showing user turn during JavaScript Pie in the Sky game

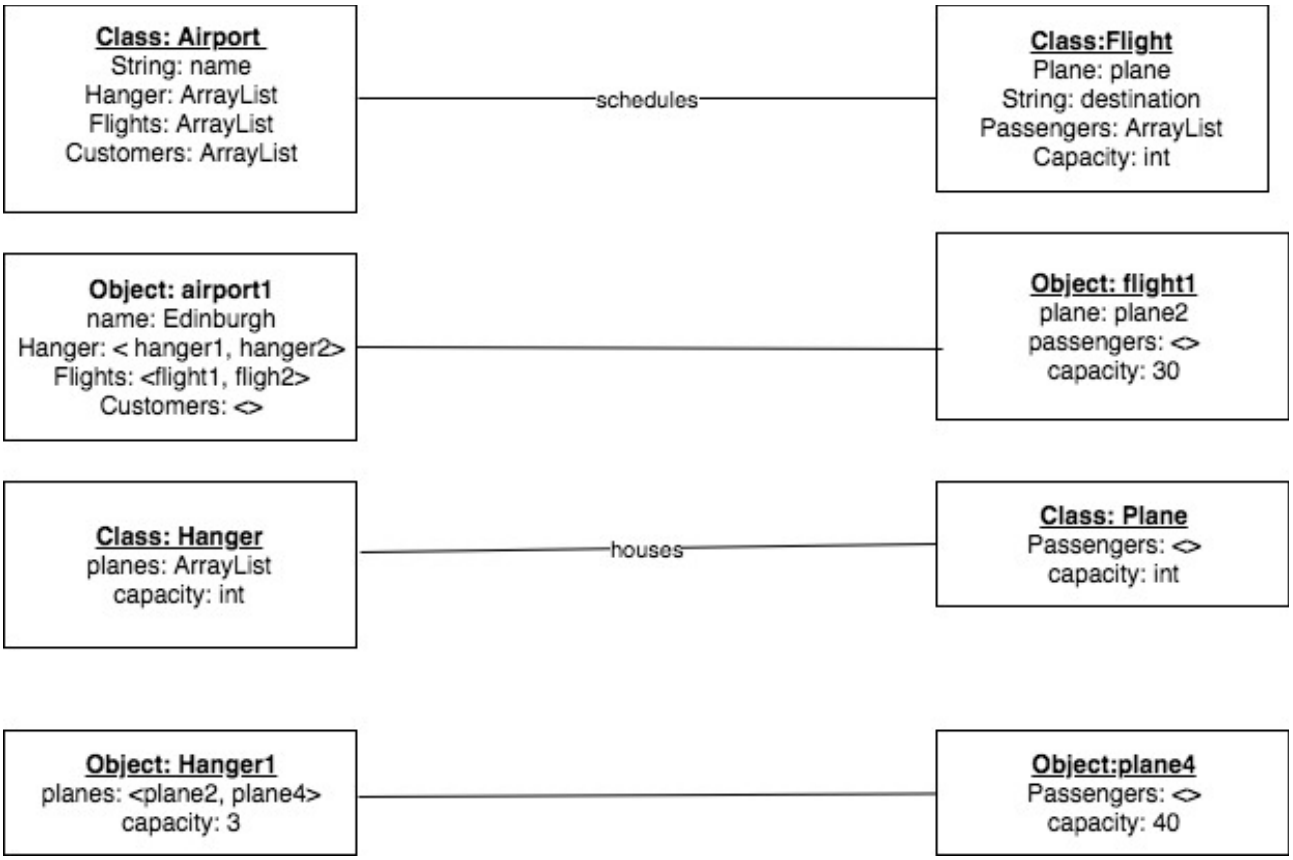


Sequence diagram showing user adding article in CMS project

Description here

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

Paste Screenshot here



Description here

Two object diagrams. The first showing airport and flight classes and the second showing hanger and plane classes

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

Unit	Ref	Evidence
		Description:

Paste Screenshot here

Bug/Error	Solution	Date
Filter not going back all articles after filtering by most/least popular	Change previous results parameter which gets entered into filterByView method so it is not changing this.state.articles when it is filtering	8/11/2018
Unable to submit patch requests to database	Correct routes in router files	7/11/2018
Page reloading before this.state reloads to show all articles	Add the reload method as a callback so this.setState is always finished first	4/11/2018
When article is displayed from database line breaks are not showing	Map through article creating a new react fragments with html break when it hits '\n'	2/11/2018

Description here

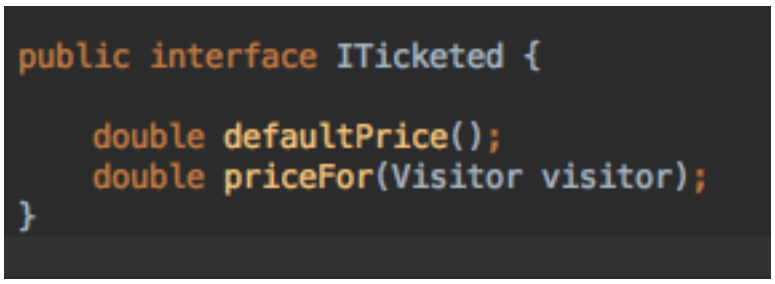
Bug tracking report for CMS JavaScript React and Java Spring project.

Week 12

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

Unit	Ref	Evidence
		Description:

Paste Screenshot here



```
public interface ITicketed {

    double defaultPrice();
    double priceFor(Visitor visitor);
}
```

Screenshot of public interface ITicketed which has two methods which are used by any class which implement the interface

```
public class Dodgem extends Attraction implements ITicketed {

    public Dodgem(String name, int rating) {
        super(name, rating);
    }

    @Override
    public double defaultPrice() {
        return 4.50;
    }

    @Override
    public double priceFor(Visitor visitor) {
        if (visitor.getAge() < 12) return defaultPrice() / 2;
        return defaultPrice();
    }

    @Override
    public boolean isAllowedToVisitors(Visitor visitor) {
        return true;
    }
}
```

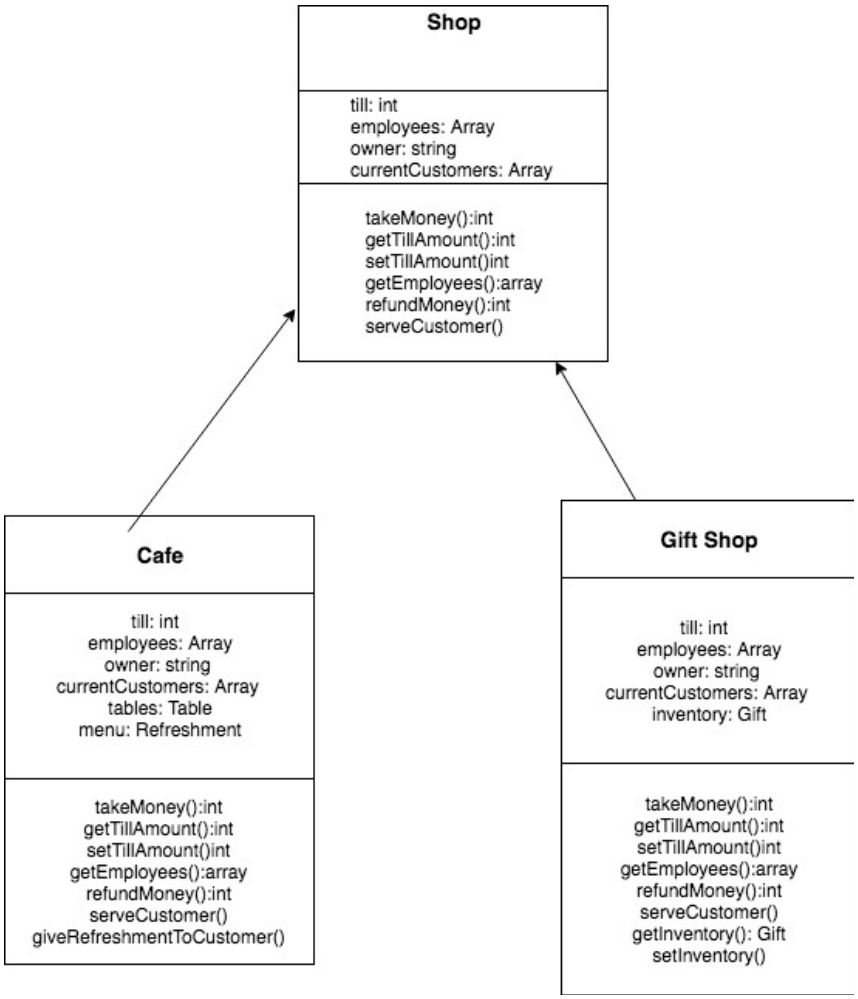
Screenshot of Dodgem class which implements ITicketed and therefore ITicketed methods. It implements ITicketed method that other classes also use but is able to specify own requirements - e.g if visitor is under 12 then the price is halved.

```java
public class Visitor {


    private int age;
    private double height;
    private double money;
    private ArrayList<ITicketed> attractionTickets;

    public Visitor(int age, double height, double money) {
        this.age = age;
        this.height = height;
        this.money = money;
        this.attractionTickets = new ArrayList<>();
    }
```

Screenshot of visitor class which has a array list into which any class which implements ITicketed can be put.

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

Paste Screenshot here



Description here

Inheritance diagram showing a Shop abstract class and two subclasses, Cafe and Gift Shop which inherit from the abstract Shop class

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

```java
public class Flight {

    private Plane plane;
    private int ticketPrice;
    private Airport destination;
    private int capacity;
    private ArrayList<Passenger> passengers;

    public Flight(int ticketPrice, Airport destination, int capacity) {
        this.ticketPrice = ticketPrice;
        this.plane = null;
        this.destination = destination;
        this.capacity = capacity;
        this.passengers = new ArrayList<>();
    }

    public int getTicketPrice() { return ticketPrice; }

    public void assignPlane(Plane plane) { this.plane = plane; }

    public Plane getPlane() { return plane; }

    public Airport getDestination() { return destination; }

    public int getFlightListSize() { return passengers.size(); }

    public void addToFlightList(Passenger passenger) { passengers.add(passenger); }
}
```

Description here

Example of encapsulation. The abstract Flight class holds several private attributes (e.g private plane, private ticketPrice, private capacity) which are only accessible through explicitly created getter and setter methods.

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

```
public abstract class Stall implements IReviewed, ISecurity {

    private String name;
    private String ownersName;
    private int parkingSpot;
    private int rating;

    public Stall (String name, String ownersName, int parkingSpot, int rating) {
        this.name = name;
        this.ownersName = ownersName;
        this.parkingSpot = parkingSpot;
        this.rating = rating;
    }

    public String getName() { return name; }

    public int getRating() { return rating; }

    public void setName(String name) { this.name = name; }

    public String getOwnersName() { return ownersName; }

    public void setOwnersName(String ownersName) { this.ownersName = ownersName; }

    public int getParkingSpot() { return parkingSpot; }

    public void setParkingSpot(int parkingSpot) { this.parkingSpot = parkingSpot; }
}
```

Screenshot of abstract class Stall

```
public class IceCreamStall extends Stall implements ITicketed {

    public IceCreamStall(String name, String ownersName, int parkingSpot, int rating) {
        super(name, ownersName, parkingSpot, rating);
    }


    @Override
    public double defaultPrice() { return 2.80; }

    @Override
    public double priceFor(Visitor visitor) { return defaultPrice(); }

    @Override
    public boolean isAllowedToVisitors(Visitor visitor) { return true; }
}
```

```
IceCreamStall iceCreamStall;

@Before
public void before() { iceCreamStall = new IceCreamStall( name: "Ice Cream",  ownersName: "Tam",  parkingSpot: 5,  rating: 9); }
```

Screenshot of IceCreamStall class inheriting from abstract Stall class

Object instance of the IceCreamStall class with assigned attributes- name, ownerName, parkingSpot and rating.

```
@Test
public void canGetName() { assertEquals( expected: "Ice Cream", iceCreamStall.getName()); }

@Test
public void canGetOwnersName() { assertEquals( expected: "Tam", iceCreamStall.getOwnersName()); }

@Test
public void canGetParkingSpot() { assertEquals( expected: 5, iceCreamStall.getParkingSpot()); }
```

Screenshot showing IceCreamStall class using .getName(), .getOwnersName() and .getParkingSpot() inherited from stall class.

```
setCurrentAnswers() {
    const currentAnswers = this.props.answers.answers.filter((answer) => answer.question_id === this.state.currentIndex)
    shuffle(currentAnswers);
    this.setState({ currentAnswers: currentAnswers })
}
```

```
const songs = props.songs.map ((song, index) => {
    return <Song song = {song} key = {index} position = {index + 1}></Song>
})
```

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

This algorithm filters through an array and selects the elements where the id matches this.state.currentIndex and sets the filtered array to currentAnswers. This is then shuffled then set to the state currentAnswers

This algorithm maps through an array of songs returning a Song object for each song element using the index of the element in the array as well as the element.