PDA Evidence - Chris Burn, G2 Cohort

Week 2:

Demonstrate the use of an array in a program. Take screenshots of:
**An array in a program*
**A function that uses the array*

Two Arrays or strings, pulled into “add_length_of_arrays” method

my_functions.rb

```ruby
array_1 = [ "Ally", "John" ],
array_2 = [ "Steve", "Alan", "Adam" ]

def add_length_of_arrays(array_1, array_2)
 add_length_of_arrays = array_1.length() + array_2.length()
 return add_length_of_arrays
end
```

**The result of the function running*

Shows TDD testing of method through MiniTest and result of MiniTest in Terminal...

my_functions_spec.rb

```ruby
require( 'minitest/autorun' )
require( 'minitest/rg' )
require_relative( 'my_functions' )

class My_Functions < MiniTest::Test


  def test_add_length_of_arrays
    result = add_length_of_arrays( [ "Ally", "John" ], [ "Steve", "Alan", "Adam" ] )
    assert_equal( 5, result )
  end
```

```
➜  day_3 ruby my_functions_spec.rb
Run options: --seed 64857

# Running:

.

Finished in 0.001457s, 686.3417 runs/s, 686.3417 assertions/s.

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

Demonstrate the use of a hash in a program. Take screenshots of:
**A hash in a program*
WITH
**The result of the function running*

Hash and an Array with in-built Hashes, WITH testing of function (separate) via MiniTest

pet_shop_spec.rb • pet_shop.rb ×

```ruby
def setup

  @customers = [ #Array with Hashes
    {
      name: "Craig",
      pets: [],
      cash: 1000
    },
    {
      name: "Zsolt",
      pets: [],
      cash: 50
    }
  ]

  @new_pet = {   #Simple Hash
          name: "Bors the Younger",
          pet_type: :cat,
          breed: "Cornish Rex",
          price: 100
        }
end

def test_add_pet_to_customer
  customer = @customers[0]
  add_pet_to_customer(customer, @new_pet) #already defined cust[0]
  assert_equal(1, customer_pet_count(customer))
end
```

**A function that uses the hash*

```ruby
def add_pet_to_customer(customers, new_pet)
  customers[:pets] << new_pet
end
```

Week 3:

Demonstrate searching data in a program. Take screenshots of:
***Function that searches data**

merchant.rb | tag.rb | transaction.rb | SqlRunner.rb | user.rb

```ruby
require_relative '../db/SqlRunner'
require 'pry-byebug'

class Transaction

  attr_accessor :id, :value, :date_id, :comment, :
    tag_id, :merchant_id

  def initialize(options)
    @id = options['id'].to_i() if options['id']
    @value = options['value'].to_i if options['
      value']
    @date_id = options['date_id']
    @comment = options['comment']
    @tag_id = options['tag_id'].to_i()
    @merchant_id = options['merchant_id'].to_i()
  end

  def self.find(id)
    sql = "SELECT * FROM transactions WHERE id =
      #{id};"
    transaction = SqlRunner.run(sql)
    result = Transaction.new(transaction.first)
    return result
  end
```

***The result of the function running**

```
moneyapp=# SELECT * FROM transactions WHERE id = 1;
 id | value |  date_id   |   comment    | tag_id | merchant_id
----+-------+------------+--------------+--------+-------------
  1 |    30 | 2017-06-04 | Weekly shop. |      1 |           2
(1 row)
```

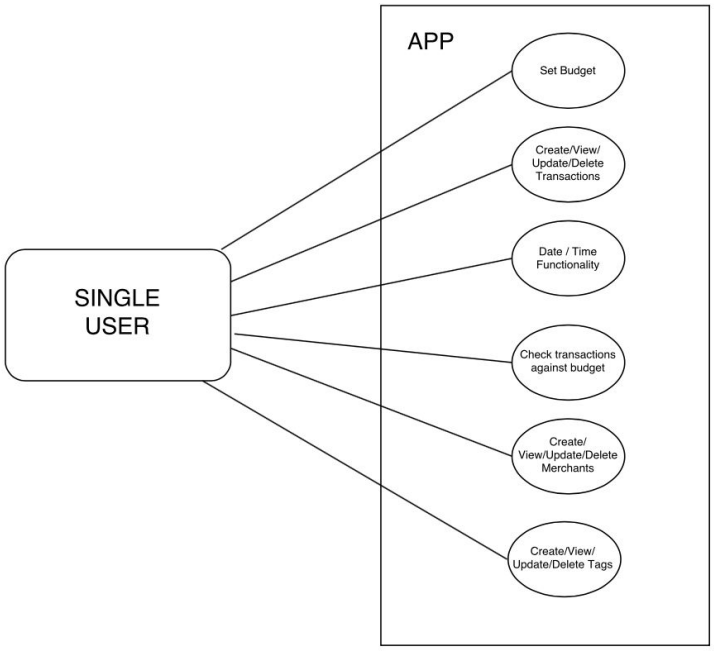
Demonstrate sorting data in a program. Take screenshots of:
***Function that sorts data**

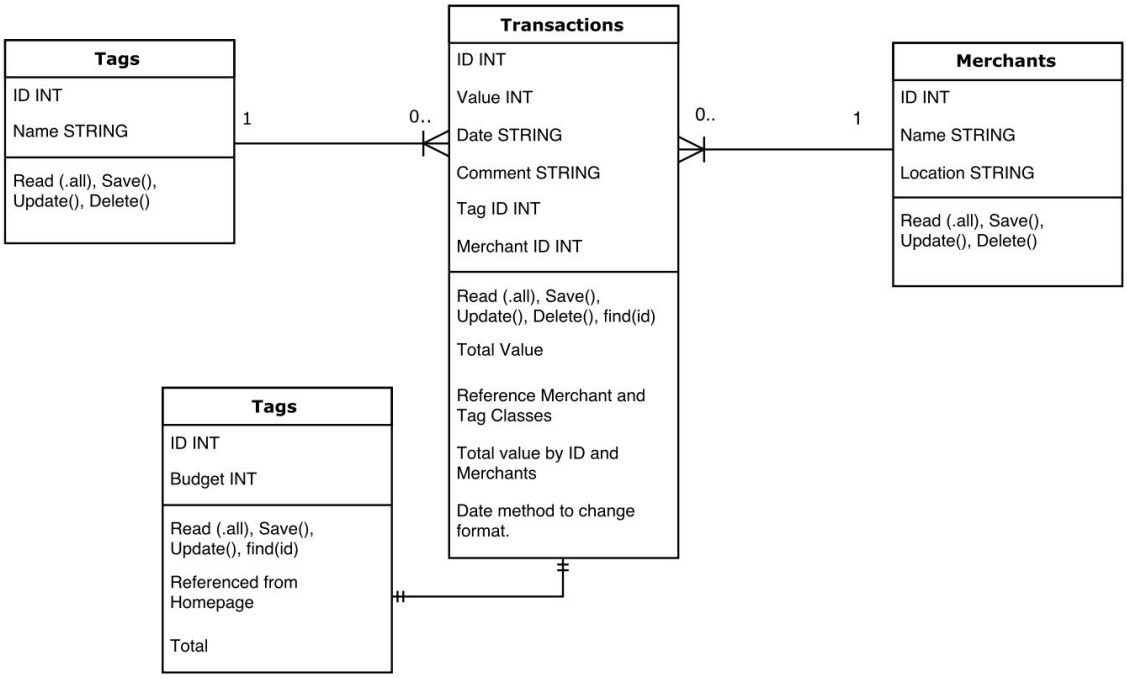
```ruby
require_relative '../db/SqlRunner'
require 'pry-byebug'

class Transaction

  attr_accessor :id, :value, :date_id, :comment, :
    tag_id, :merchant_id

  def initialize(options)
    @id = options['id'].to_i() if options['id']
    @value = options['value'].to_i if options['
      value']
    @date_id = options['date_id']
    @comment = options['comment']
    @tag_id = options['tag_id'].to_i()
    @merchant_id = options['merchant_id'].to_i()
  end


  def self.sort_by_date
    sql = "SELECT * FROM transactions ORDER BY
      date_id;"
   end
```

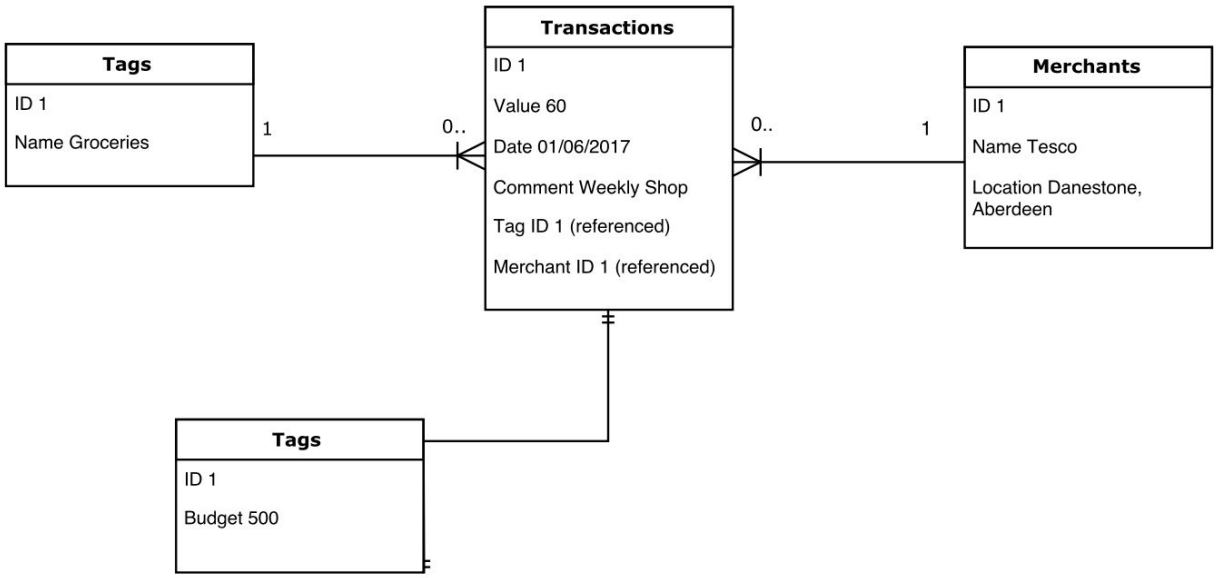
***The result of the function running**

```
 id | value |  date_id   |               comment                | tag_id |
merchant_id
----+-------+------------+--------------------------------------+--------+-
------------
  5 |    39 | 2017-05-20 | Train tickets to Glasgow.            |      6 |
          7
  4 |    50 | 2017-05-21 | Night out with mates.                |      7 |
          5
  3 |    15 | 2017-05-27 | Mid-week takeaway.                   |      4 |
          4
  7 |    40 | 2017-05-28 | CDs and Blu-Ray for sisters birthday. |      2 |
          6
 10 |    50 | 2017-05-29 | Whisky for me!                       |      9 |
          9
  6 |    45 | 2017-05-30 | Weekly shop.                         |      1 |
          1
  8 |    70 | 2017-06-01 | Paint for home decoration.           |      3 |
          3
  1 |    30 | 2017-06-04 | Weekly shop.                         |      1 |
          2
  2 |    19 | 2017-06-05 | Replacement window wipers for car.   |      5 |
          8
 11 |    25 | 2017-06-09 | More pizza!                          |      4 |
          4
  9 |   200 | 2017-06-11 | Flights to Madrid                    |      6 |
         10
(11 rows)
```

Week 5:

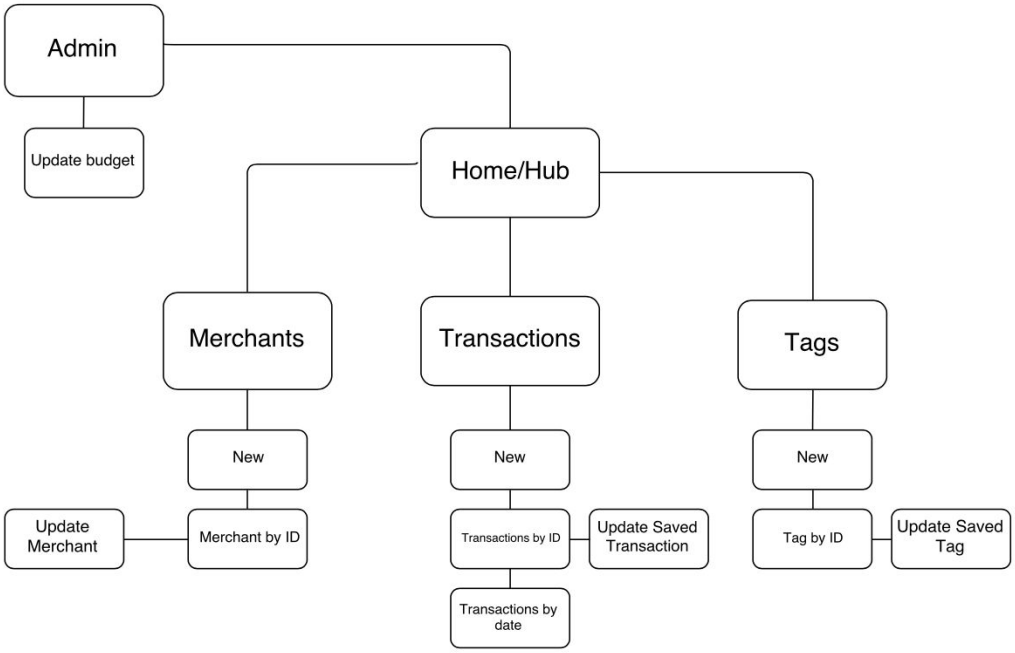

APP
Set Budget
Create/View/ Update/Delete Transactions
Date / Time Functionality
Check transactions against budget
Create/ View/Update/Delete Merchants
Create/View/ Update/Delete Tags
SINGLE USER

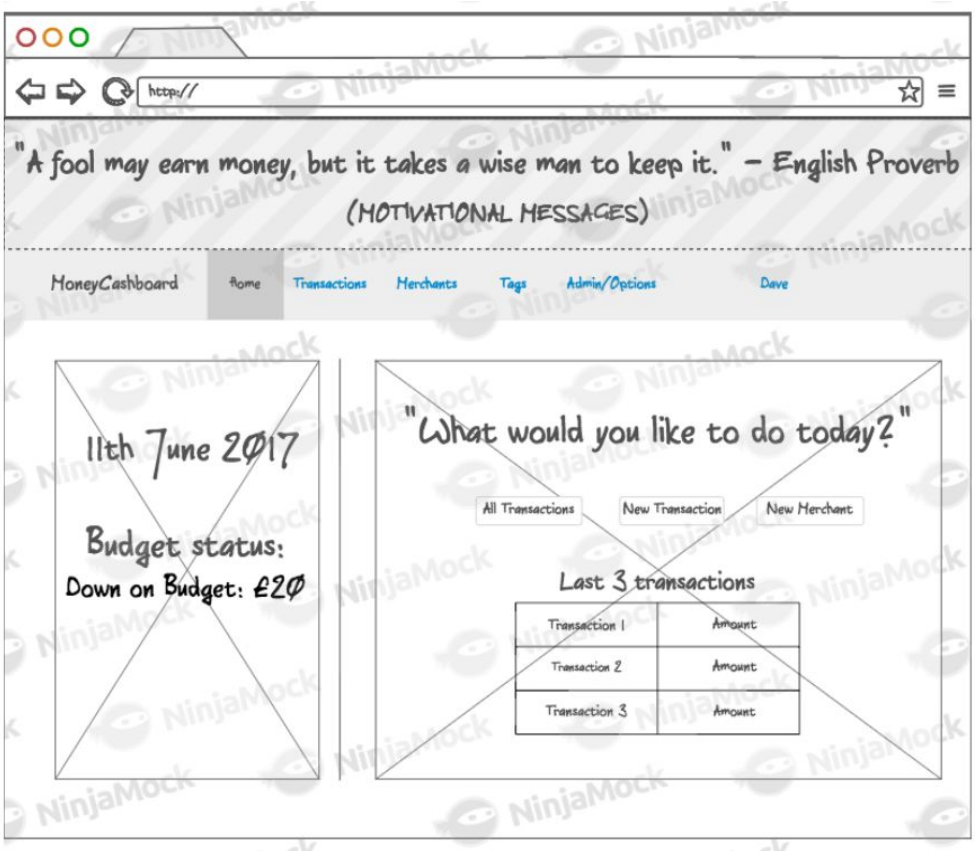

Tags
ID INT
Name STRING
Read (.all), Save(), Update(), Delete()
1
0..
Transactions
ID INT
Value INT
Date STRING
Comment STRING
Tag ID INT
Merchant ID INT
Read (.all), Save(), Update(), Delete(), find(id)
Total Value
Reference Merchant and Tag Classes
Total value by ID and Merchants
Date method to change format.
0..
1
Merchants
ID INT
Name STRING
Location STRING
Read (.all), Save(), Update(), Delete()
Tags
ID INT
Budget INT
Read (.all), Save(), Update(), find(id)
Referenced from Homepage
Total

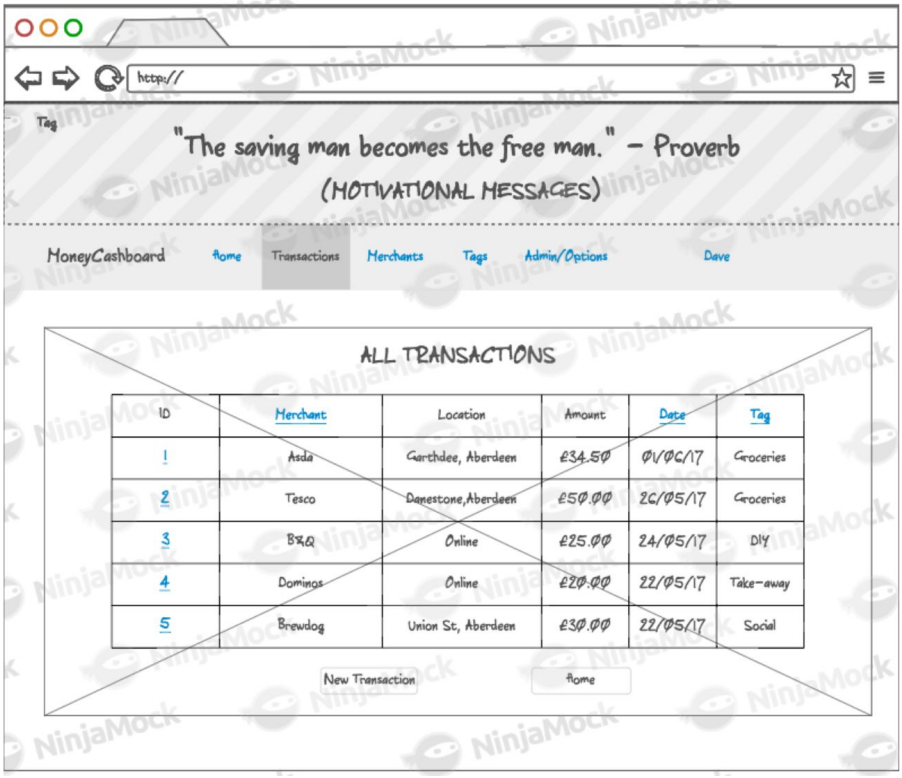


Money CashBoard Implementation Constraints Plan - Sheet1

IMPLEMENTATION CONSTAINTS PLAN		
Constraint	Possible Effect on Constraint on Product	Solution
Hardware and Software Problems	Reduced user-experience cross platforms, and on different user machines.	Program to compatible cross browsers and machines including mobile platforms. Use of dynamic displays using FlexBox should help to achieve this.
Performance Requirements	Reduced performance when frequently calling on database data. Potential for reduced user experience.	Unlikely to have mass traffic as it's an application for personal use. Data being called from database is of relatively small size and is unlikely to cause an issue with performance.
Persistant Storage and Transactions	Potential for future loss or reduction of free/cheap database capacity may curtail use of application going forward.	To get around this the website design will avoid the use of large media files / graphics /styling which also has potential for an upside on meeting budget requirements.
Usability	User unable to navigate site with ease, reducing user experience and reduce likelihood of future use and roll out.	Personal use of application, not being used in public. Simple design with large menus/buttons and bold text and colours should reduce potantial for navigation errors.
Budgets	No budget considered but in theory a limited budget will curtail use of premium servers meaning potential for slower performance and risk of downtime. In additon, use of non premium graphics and design may limit the visual appeal of the finished product.	Use of free web tooling and server space will keep web app cost minimal for as long a period of time as application is required.
Time Limitations	6 day turnaround for finished product will mean limited tit to perfect usability and design/finish of product. Additional functionality above MVP may be limited.	Careful planning of project should reduce potential for road-blocks in execution and possible delays to project.


Admin
Update budget
Home/Hub
Merchants
Transactions
Tags
New
New
New
Update Merchant
Merchant by ID
Transactions by ID
Update Saved Transaction
Tag by ID
Update Saved Tag
Transactions by date


http://
"A fool may earn money, but it takes a wise man to keep it." – English Proverb
(MOTIVATIONAL MESSAGES)
MoneyCashboard
Home
Transactions
Merchants
Tags
Admin/Options
Dave
11th June 2017
Budget status:
Down on Budget: £20
"What would you like to do today?"
All Transactions
New Transaction
New Merchant
Last 3 transactions
Transaction 1
Amount
Transaction 2
Amount
Transaction 3
Amount



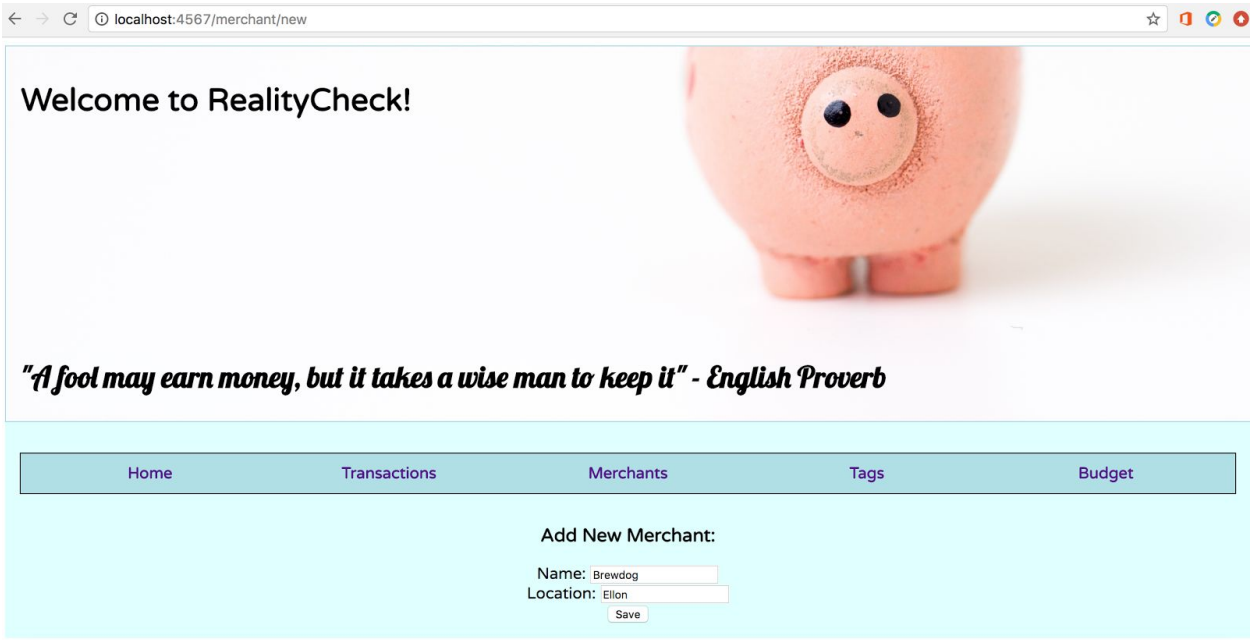
Psudo-code example showing a method calling on the result of an INNER JOIN on two database tables:

```
#make a Class method on Transaction Class to
dislay Transaction value by Merchant from
Merchant Class.

def self.total_by_merchant(id)
  #write SQL to join Tranasction and Merchant
  database tables against ID(Transaction) and
  Merchant ID(Merchant)
  sql = "SELECT * FROM transactions INNER JOIN
    merchants ON merchants.id =
    transactions.merchant_id WHERE merchant_id
    = #{id};"
  #assign local variable to running of SQL code
  through SqlRunner Class Ruby file
  total = SqlRunner.run(sql)
  #return "total" variable and 'map'/loop
  'total' to pull out individual transactions
  #sum all 'value' properties (coverted to
  Integer) of Transactions called
  return total.map {|transaction| transaction['
    value'].to_i}.sum
end
```

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

Show an interaction with data persistence. Take a screenshot of:
* Data being inputted into your program
* Confirmation of the data being saved

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




```ruby
get '/merchant/new' do
  erb(:merchant_new)
end

post '/merchants' do
  @merchant = Merchant.new(params)
  @merchant.save()
  erb(:merchant_create)
end
```

controller.rb | merchant_new.erb | console.rb | styles.css

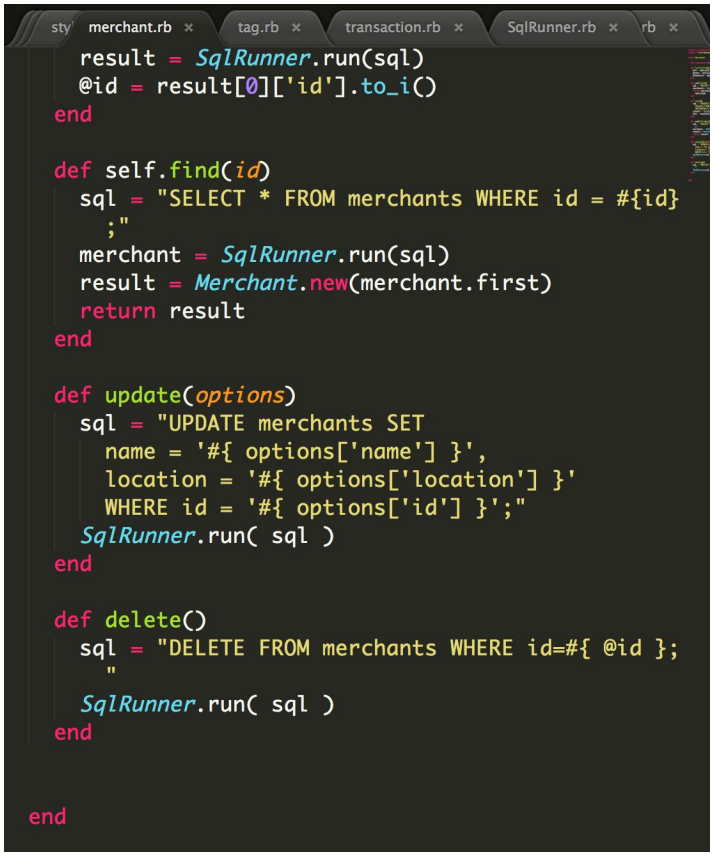
```erb
<div class="cell">

<h3>Add New Merchant:</h3>

<form action="/merchants" method="POST">

<div>
<label for="name">Name:</label>
  <input type="text" id="name" name="name" required
  >
</div>

<div>
<label for="location">Location:</label>
  <input type="text" id="location" name="location"
  required>
</div>

<input type="submit" value="Save">

</form>
</div>

```

controller.rb | merchant_create.erb | console.rb | styles.css

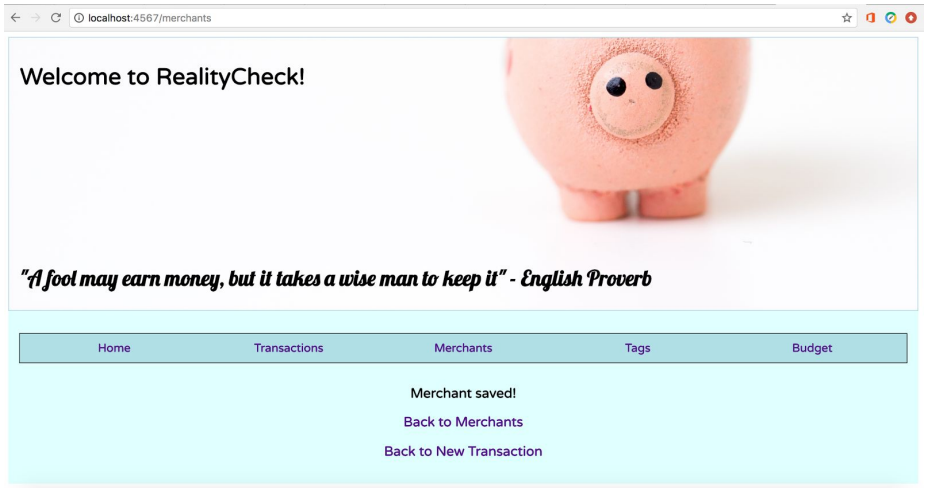
```erb
<div class="cell">
<h3>Merchant saved!</h3>

<h3><a href="/merchants">Back to Merchants</a></h3>

<h3><a href="/transaction/new">Back to New
Transaction</a></h3>
</div>

```

merchant.rb | tag.rb | transaction.rb | SqlRunner.rb

```ruby
require_relative '../db/SqlRunner'
require 'pry-byebug'

class Merchant

  attr_accessor :id, :name, :location

  def initialize(options)
    @id = options['id'].to_i if options['id']
    @name = options['name']
    @location = options['location']
  end

  def self.all()
    sql = "SELECT * FROM merchants;"
    merchants = SqlRunner.run(sql)
    return merchants.map {|options| Merchant.new(
      options)}
  end

  def save()
    sql = "INSERT INTO merchants (name, location)
      VALUES ('#{@name}', '#{@location}')
      RETURNING *;"
    result = SqlRunner.run(sql)
    @id = result[0]['id'].to_i()
  end
```




```
17 | Amazon UK       | Online
18 | Scotrail        | Train Station, Aberdeen
19 | Halfords        | Berryden, Aberdeen
20 | Whisky Exchange | Online
21 | British Airways | Online
22 | Falkirk Council | West Bridge Street, Falkirk
23 | Asda            | Garthdee, Aberdeen
24 | Tesco           | Danestone, Aberdeen
25 | B&Q             | Online
26 | Dominos         | Online
27 | Brewdog         | Merchant City, Glasgow
28 | Amazon UK       | Online
29 | Scotrail        | Train Station, Aberdeen
30 | Halfords        | Berryden, Aberdeen
31 | Whisky Exchange | Online
32 | British Airways | Online
33 | Falkirk Council | West Bridge Street, Falkirk
34 | Brewdog         | Ellon
(34 rows)

(END)
```



Week 6:

Demonstrate the use of Polymorphism in a program:
POLYMORPHISM

- Capacity to take on different forms.
- A language's ability to process objects of various types and classes through a single, uniform interface.
- Example, how a parent class refers to a child class object
- Any object that satisifies more than one IS-A relationship is polymorphic in nature.
- Fair to say that every Object in Java is polymorphic in nature, each on passes IS-A test for itself and for Object class.

E.g. Animal class, Cat IS sub-class of Animal. Cat satisfies IS-A relationship of own type and its Super-class, Animal.

```java
package com.example.user.multipleclasshomework;

import java.util.IllegalFormatException;

/**
 * Created by user on 20/06/2017.
 */

public class Employee {

    private int empId;
    private String name;
    private String ssn;
    private double salary;

    public Employee(int empId, String name, String ssn, double salary) {
        this.empId = empId;
        this.name = name;
        this.ssn = ssn;
        this.salary = salary;
    }

    public int getEmpId() {
        return empId;
    }


    public String getName() {
        return name;
    }

    public void setName(String name) {
        if(name != null && name != "") {
```

```java
package com.example.user.multipleclasshomework;

/**
 * Created by user on 20/06/2017.
 */

public class Manager extends Employee {

    private String deptName;

    public Manager(int empId, String name, String ssn, double salary, String deptName) {
        super(empId, name, ssn, salary);
        this.deptName = deptName;
    }

    public String getDeptName() {
        return this.deptName;
    }

}
```

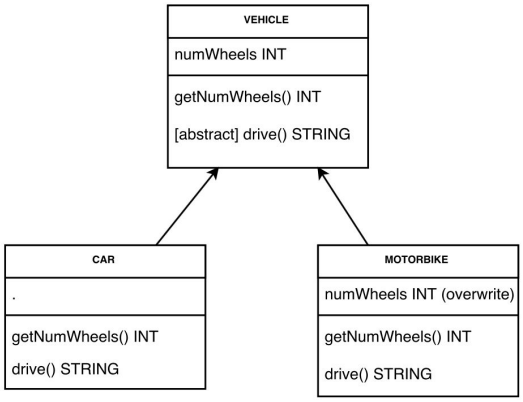
```java
package com.example.user.multipleclasshomework;

/**
 * Created by user on 20/06/2017.
 */

public class Director extends Manager {

    private double budget;

    public Director(int empId, String name, String ssn, double salary, String deptName, double budget) {
        super(empId, name, ssn, salary, deptName);
        this.budget = budget;
    }


    public double getBudget(){
        return this.budget;
    }

}
```

```java
import org.junit.Test;

import static org.junit.Assert.assertEquals;

/**
 * Created by user on 20/06/2017.
 */

public class TestDirector {

    Director director;

    @Before
    public void before(){
        director = new Director(20, "Sean Jones", "FGHIJ", 50000.00, "Sausages", 500000.00);
    }

    @Test
    public void canRaiseSalary(){
        director.raiseSalary(80000.00);
        assertEquals(80000.00, director.getSalary(), 0.01);
    }

    @Test
    public void canSetName(){
        director.setName("Brent Jones");
        assertEquals("Brent Jones", director.getName());
    }
```

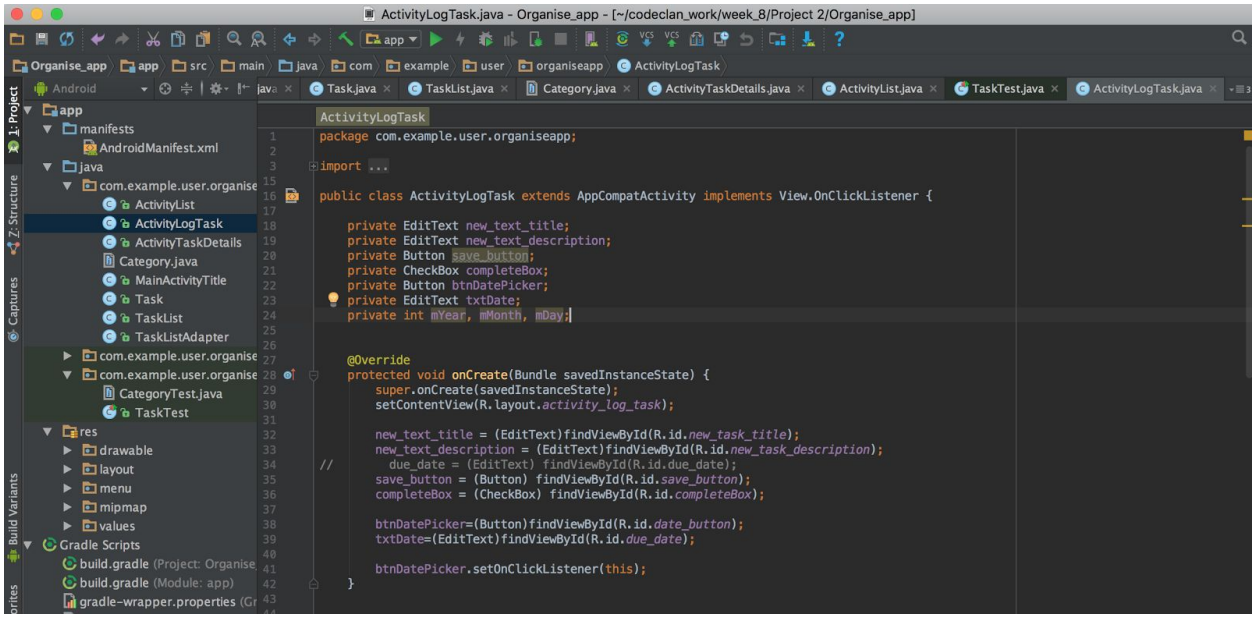
All 2 tests passed - 5ms

Week 7:

An Inheritance Diagram



Take a screenshot of an example of encapsulation in a program



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.*

```
package example.codeclan.com.godzilla;

import static android.R.attr.value;

public abstract class Kaiju {

    public String name;
    public int healthValue;
    public int destructiveForce;

    public Kaiju(String name, int healthValue, int destructiveForce) {
        this.name = name;
        this.healthValue = healthValue;
        this.destructiveForce = destructiveForce;
    }

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

    public int getHealthValue(){
        return this.healthValue;
    }

    public int getDestructiveForce(){
        return this.destructiveForce;
    }


}
```

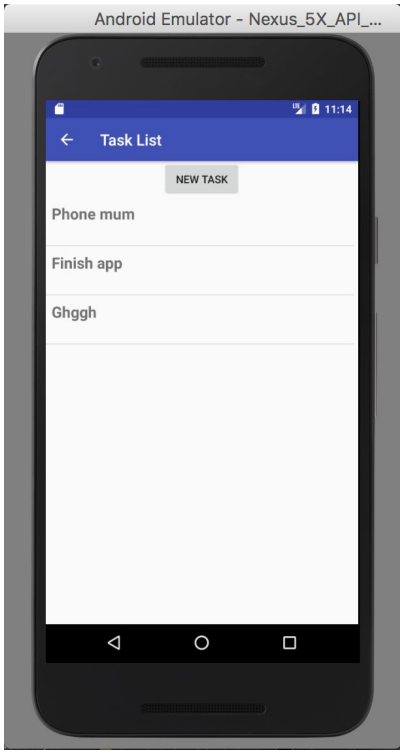
```java
package example.codeclan.com.godzilla;

/**
 * Created by user on 21/06/2017.
 */

public class Godzilla extends Kaiju implements Attack {

        public Godzilla(String name, int healthValue, int destructiveForce) {
            super(name, healthValue, destructiveForce);
        }

//...

        public String roar(){
            return "ROAR!";
        }

}
```

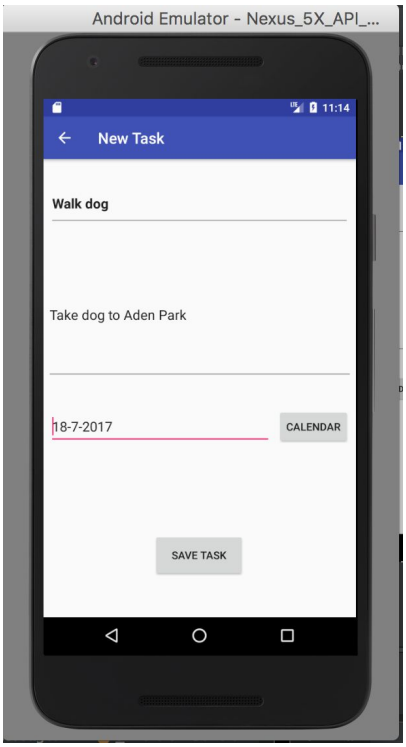
```java
package example.codeclan.com.godzilla;

/**
 * Created by user on 21/06/2017.
 */

public class Pikachu extends Kaiju implements Attack {

    public Pikachu(String name, int healthValue, int destructiveForce){
        super(name, healthValue, destructiveForce);
    }

    public String attack(){
        return "Kablammo!!";
    }

    public String roar(){
        return "PIKA! PIKA!";
    }
}
```

```java
package example.codeclan.com.godzilla;

/**
 * Created by user on 21/06/2017.
 */

public class Runner {

    public static void main(String[] args){

        Godzilla godzilla = new Godzilla("Zilla", 120, 50);
        Pikachu pikachu = new Pikachu("Pika", 90, 35);
        SkyScraper skyScraper = new SkyScraper("largebuilding", 30);
        Townhall townhall = new Townhall("smallbuilding", 20);

        System.out.println("Zilla enters the City, looks like trouble!");
        System.out.println("Zilla looks pretty healthy with health " + godzilla.getHealthValue());


    }
}
```

Take a screenshot of one of your projects where you have worked alone and attach the Github link.

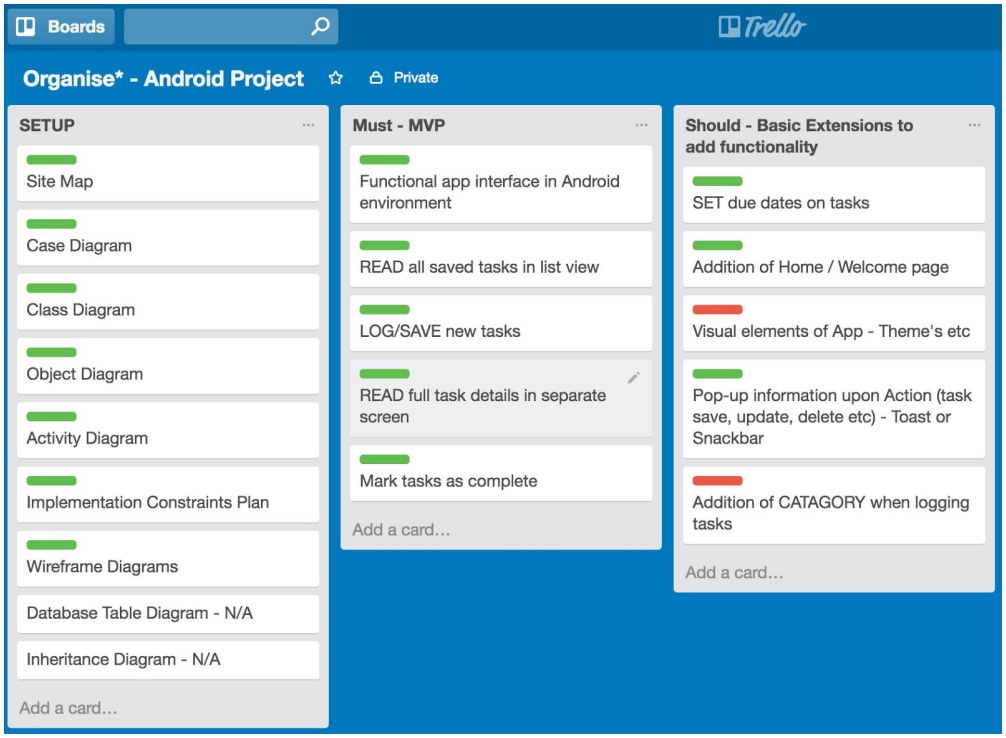
https://github.com/chris-burn/To-Do_list_project

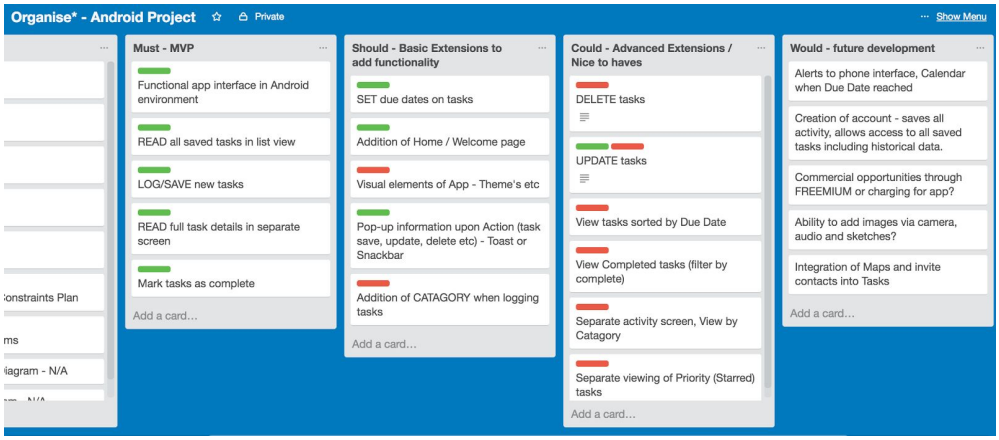


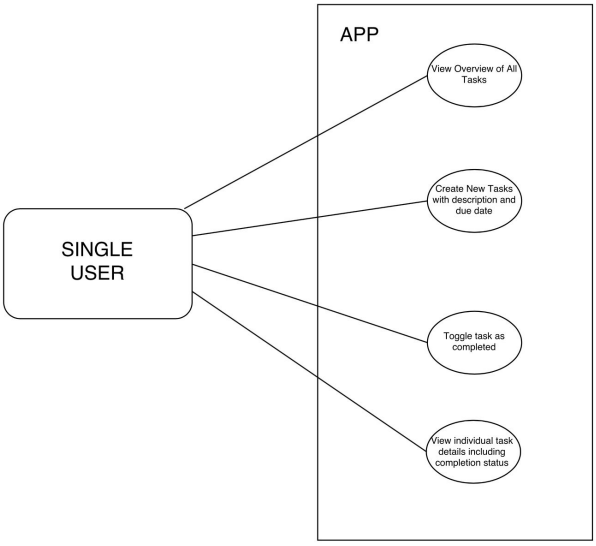


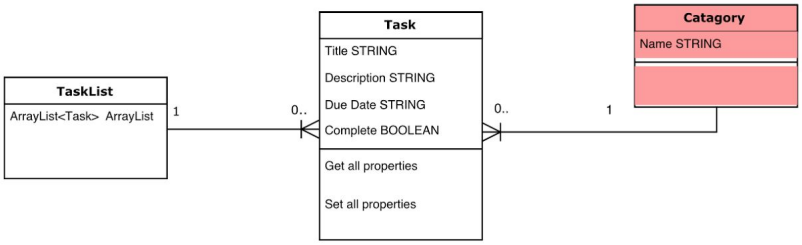
Take screenshots or photos of your planning and the different stages of development to show changes.

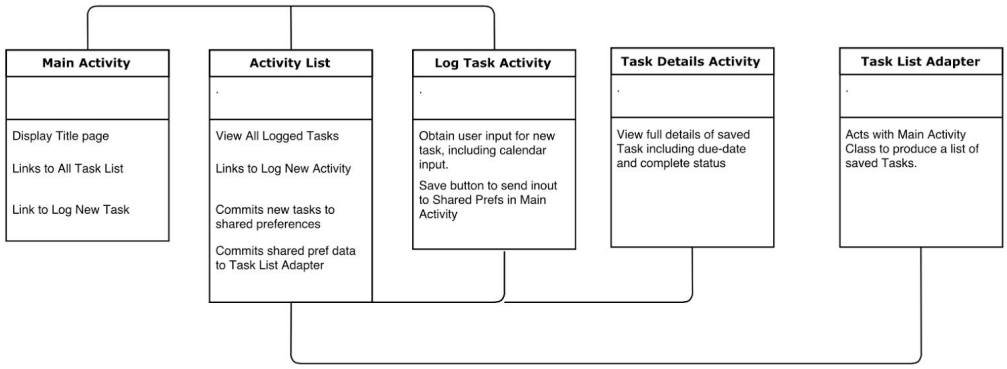
Android App, PROJECT PLANNING:

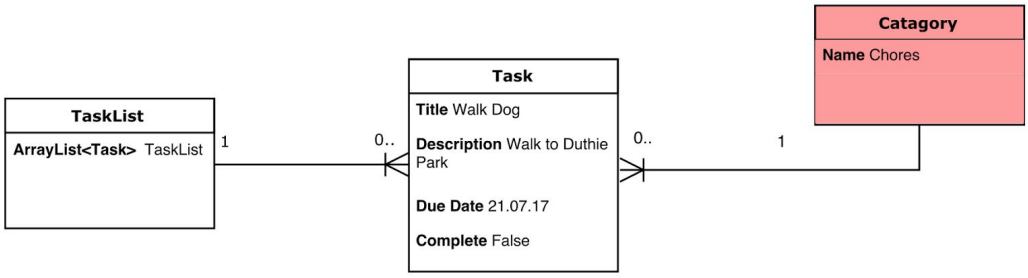


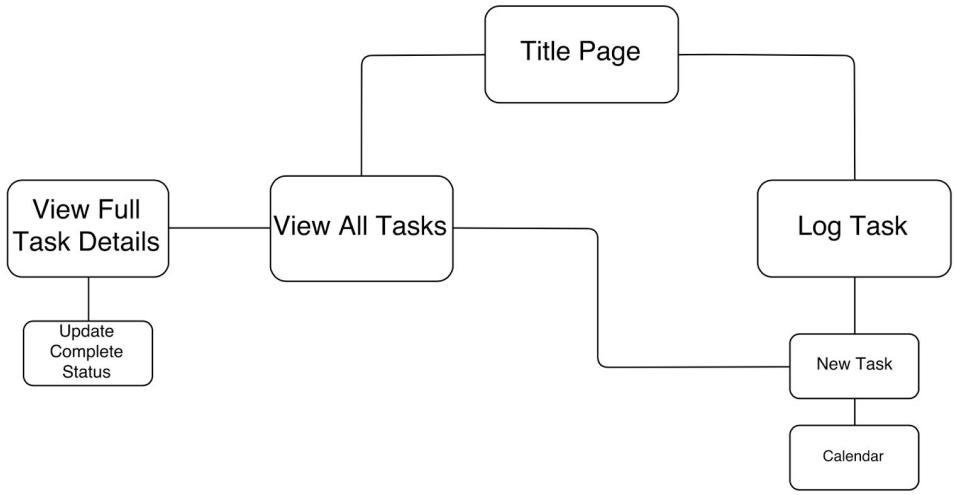



APP
View Overview of All Tasks
Create New Tasks with description and due date
Toggle task as completed
View individual task details including completion status
SINGLE USER

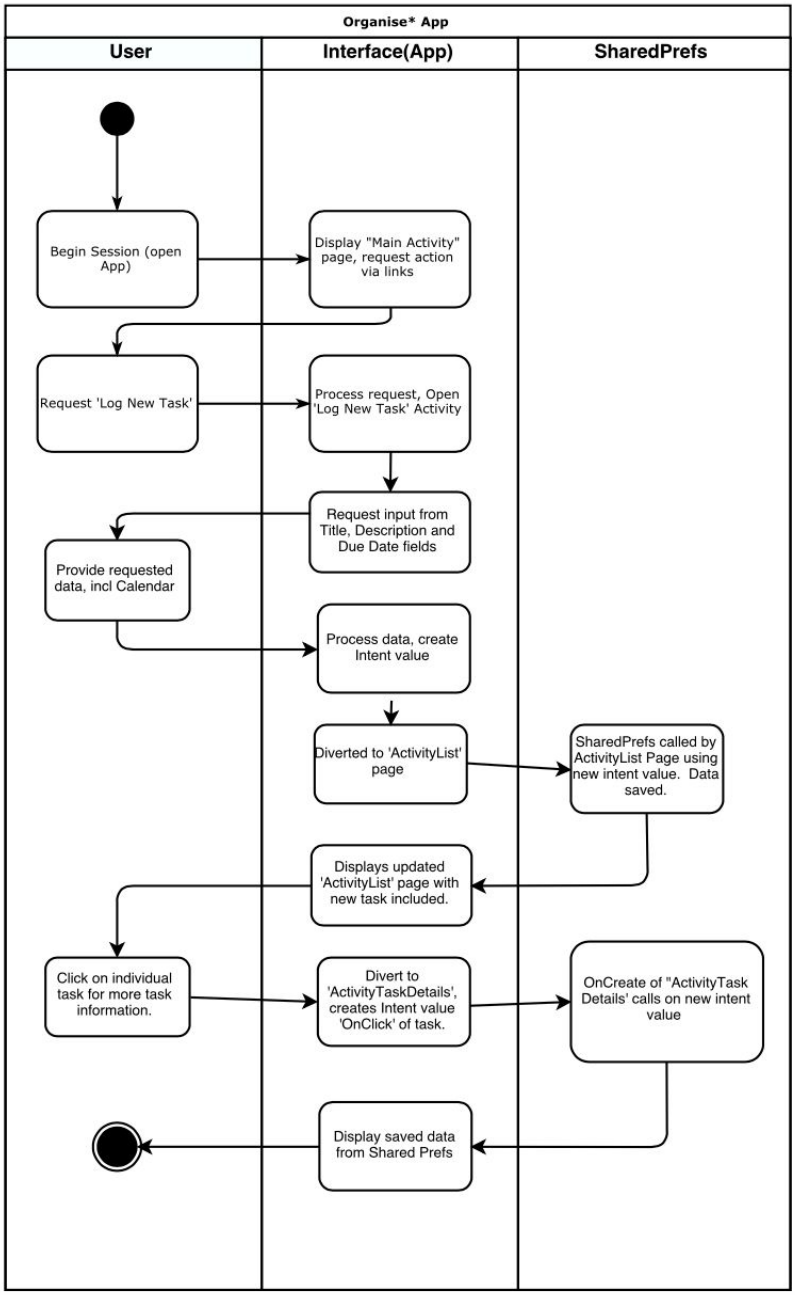

Task
Title STRING
Description STRING
Due Date STRING
Complete BOOLEAN
Get all properties
Set all properties
Catagory
Name STRING
TaskList
ArrayList<Task> ArrayList
1
0..
0..
1


Main Activity
Display Title page
Links to All Task List
Link to Log New Task
Activity List
View All Logged Tasks
Links to Log New Activity
Commits new tasks to shared preferences
Commits shared pref data to Task List Adapter
Log Task Activity
Obtain user input for new task, including calendar input.
Save button to send inout to Shared Prefs in Main Activity
Task Details Activity
View full details of saved Task including due-date and complete status
Task List Adapter
Acts with Main Activity Class to produce a list of saved Tasks.


Catagory
Name Chores
Task
Title Walk Dog
Description Walk to Duthie Park
Due Date 21.07.17
Complete False
TaskList
ArrayList<Task> TaskList
1
0..
0..
1



IMPLEMENTATION CONSTAINTS PLAN		
Constraint	Possible Effect on Constraint on Product	Solution
Hardware and Software Problems	Limited use - Android App only	Multiple versions of apps would be required. No instant solution.
Performance Requirements	Potential limit to performance especially as the data size increases. Minimal visuals in app at present should limit issues with regards app performance.	Unlikely to have mass traffic as it's an application for personal use. Data being used for App use is of relatively small size and is unlikely to cause an issue with performance. Use of SQL database would solve any data volume concerns.
Persistant Storage and Transactions	Data persistance achieved through SharedPreferences internal to app only, no database use which may limit app potential.	To get around this the website design will avoid the use of large media files / graphics /styling which also has potential for an upside on meeting budget requirements.
Usability	User unable to navigate site with ease, reducing user experience and reduce likelihood of future use and roll out.	Simple design layout with large menus/buttons and bold text and colours should reduce potantial for navigation errors. Personal use of application, not for public distribution in current state.
Budgets	No budget considered but in theory a limited budget will curtail use of premium servers meaning potential for slower performance and risk of downtime. In additon, use of non premium graphics and design may limit the visual appeal of the finished product.	Use of free web tooling and server space will keep web app cost minimal for as long a period of time as application is required.
Time Limitations	6 day turnaround for finished product means limited design/finish of product. Additional functionality above MVP may be limited.	Careful planning of project should reduce potential for road-blocks in execution and possible delays to project.


Organise* App
User
Interface(App)
SharedPrefs
Begin Session (open App)
Display "Main Activity" page, request action via links
Request 'Log New Task'
Process request, Open 'Log New Task' Activity
Request input from Title, Description and Due Date fields
Provide requested data, incl Calendar
Process data, create Intent value
Diverted to 'ActivityList' page
SharedPrefs called by ActivityList Page using new intent value. Data saved.
Displays updated 'ActivityList' page with new task included.
Click on individual task for more task information.
Divert to 'ActivityTaskDetails', creates Intent value 'OnClick' of task.
OnCreate of "ActivityTask Details' calls on new intent value
Display saved data from Shared Prefs

Week 10:

Demonstrate testing in your program. Take screenshots of:

** Example of test code*

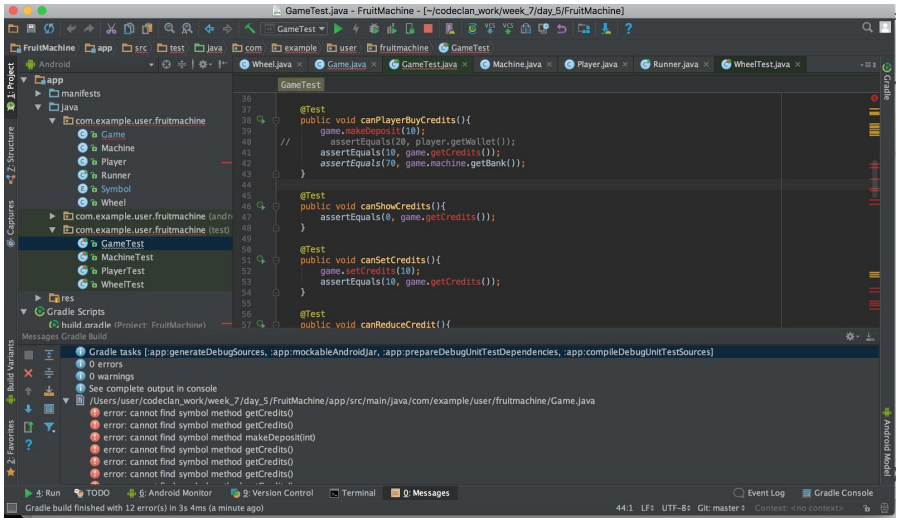
** The test code failing to pass*

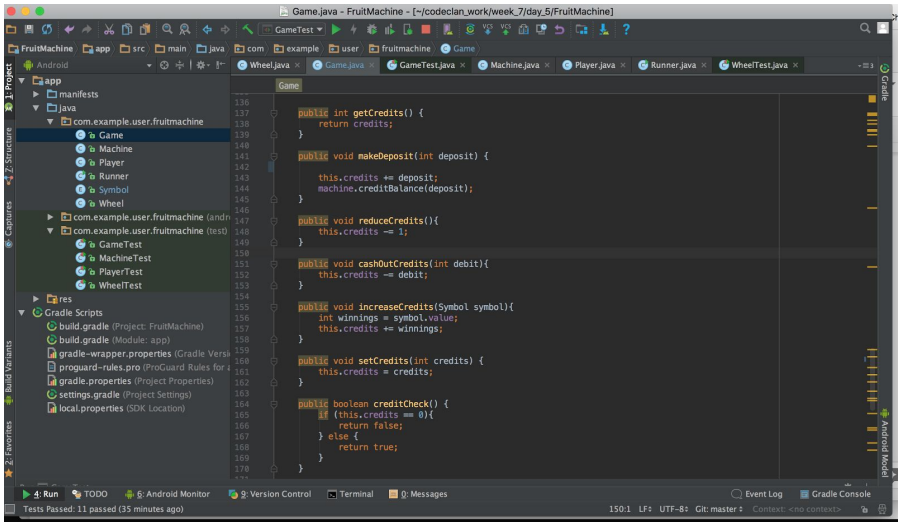
** Example of the test code once errors have been corrected*

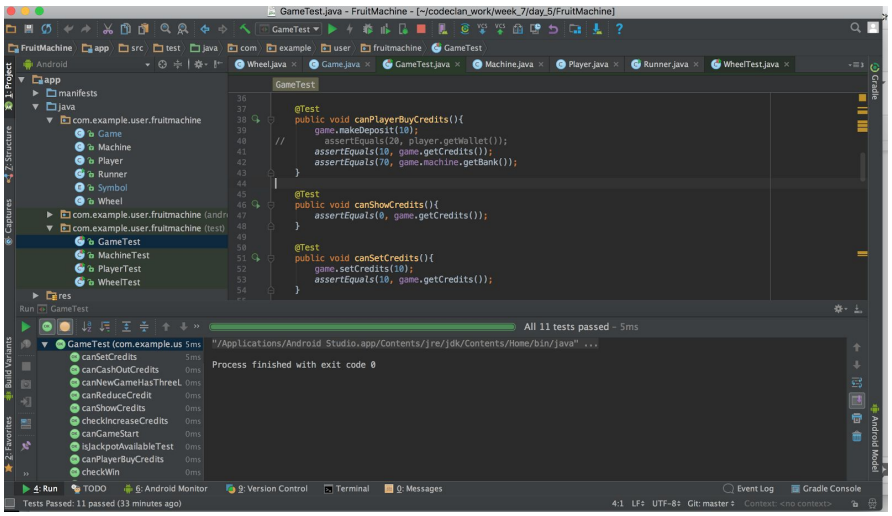
** The test code passing*

GameTest

```
13  * Created by user on 01/07/2017.
14  */
15
16  public class GameTest {
17
18      Game game;
19      Wheel wheel1;
20      Wheel wheel2;
21      Wheel wheel3;
22      Machine machine;
23      Player player;
24      Symbol symbol;
25
26      @Before
27      public void before(){
28          this.player = new Player(20);
29          this.wheel1 = new Wheel();
30          this.wheel2 = new Wheel();
31          this.wheel3 = new Wheel();
32          this.machine = new Machine(60, wheel1, wheel2, wheel3);
33          this.game = new Game(machine, player);
34      }
35
36
37      @Test
38      public void canPlayerBuyCredits(){
39          game.makeDeposit(10);
40  //          assertEquals(20, player.getWallet());
41          assertEquals(10, game.getCredits());
42          assertEquals(70, game.machine.getBank());
43      }
44
```


GameTest.java - FruitMachine - [~/codeclan_work/week_7/day_5/FruitMachine]
Gradle tasks [:app:generateDebugSources, :app:mockableAndroidJar, :app:prepareDebugUnitTestDependencies, :app:compileDebugUnitTestSources]
0 errors
0 warnings
See complete output in console
/Users/user/codeclan_work/week_7/day_5/FruitMachine/app/src/main/java/com/example/user/fruitmachine/Game.java
error: cannot find symbol method getCredits()
error: cannot find symbol method getCredits()
error: cannot find symbol method makeDeposit(int)
error: cannot find symbol method getCredits()
error: cannot find symbol method getCredits()
error: cannot find symbol method getCredits()
Gradle build finished with 12 error(s) in 3s 4ms (a minute ago)


Game.java - FruitMachine - [~/codeclan_work/week_7/day_5/FruitMachine]
Tests Passed: 11 passed (35 minutes ago)

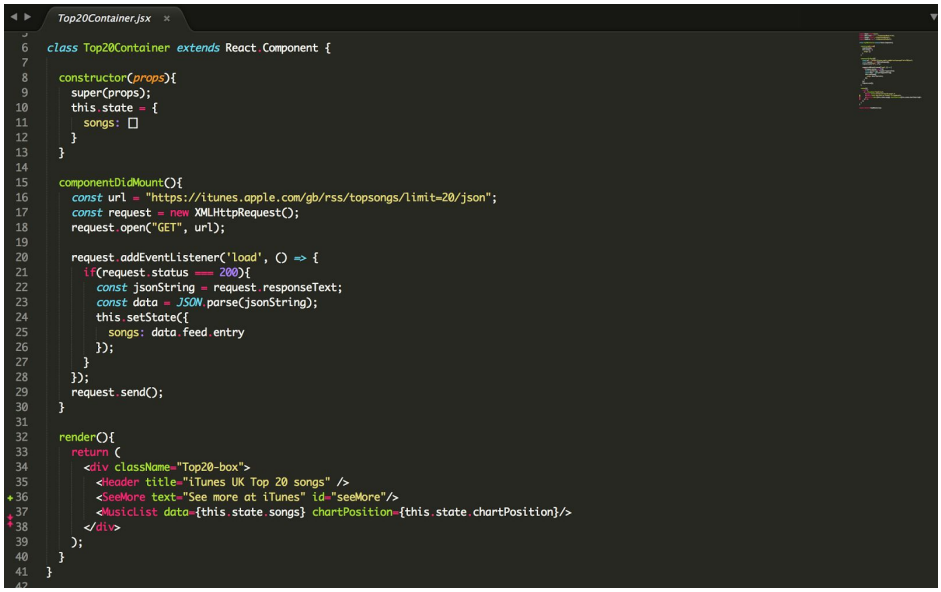

GameTest.java - FruitMachine - [~/codeclan_work/week_7/day_5/FruitMachine]
All 11 tests passed - 5ms
Process finished with exit code 0
Tests Passed: 11 passed (33 minutes ago)

Week 11:

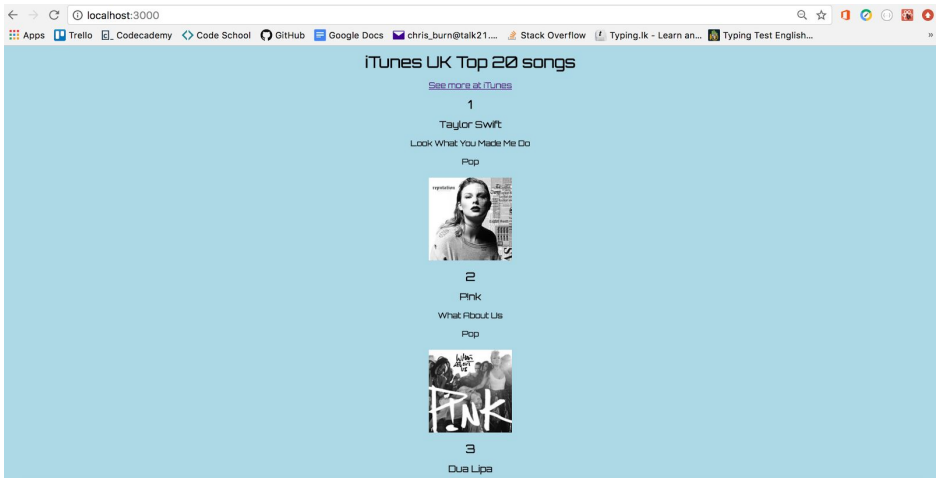
Show an API being used within your program. Take a screenshot of:

** The code that uses or implements the API*

** The API being used by the program whilst running*

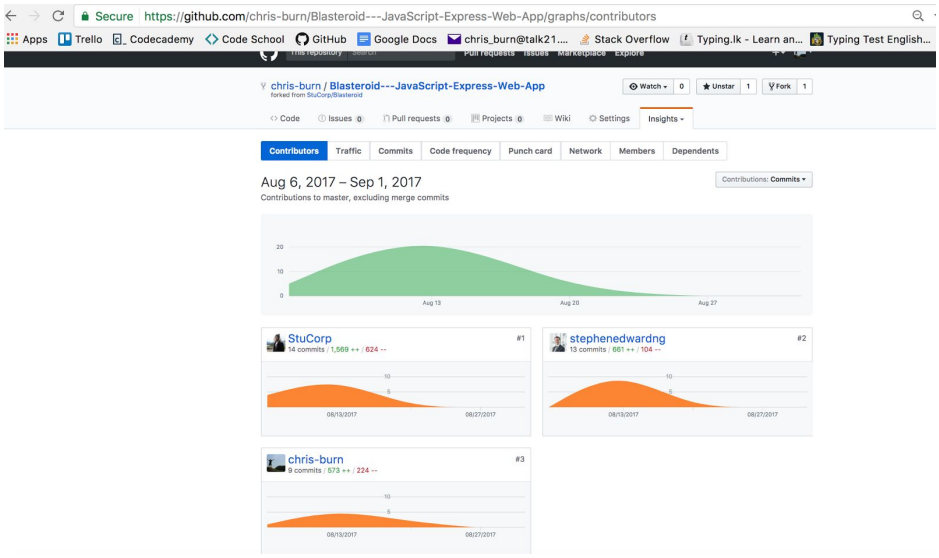


```
Top20Container.jsx

class Top20Container extends React.Component {

  constructor(props){
    super(props);
    this.state = {
      songs: []
    }
  }

  componentDidMount(){
    const url = "https://itunes.apple.com/gb/rss/topsongs/limit=20/json";
    const request = new XMLHttpRequest();
    request.open("GET", url);

    request.addEventListener('load', () => {
      if(request.status === 200){
        const jsonString = request.responseText;
        const data = JSON.parse(jsonString);
        this.setState({
          songs: data.feed.entry
        });
      }
    });
    request.send();
  }

  render(){
    return (
      <div className="Top20-box">
        <Header title="iTunes UK Top 20 songs" />
        <SeeMore text="See more at iTunes" id="seeMore"/>
        <MusicList data={this.state.songs} chartPosition={this.state.chartPosition}/>
      </div>
    );
  }
}
```

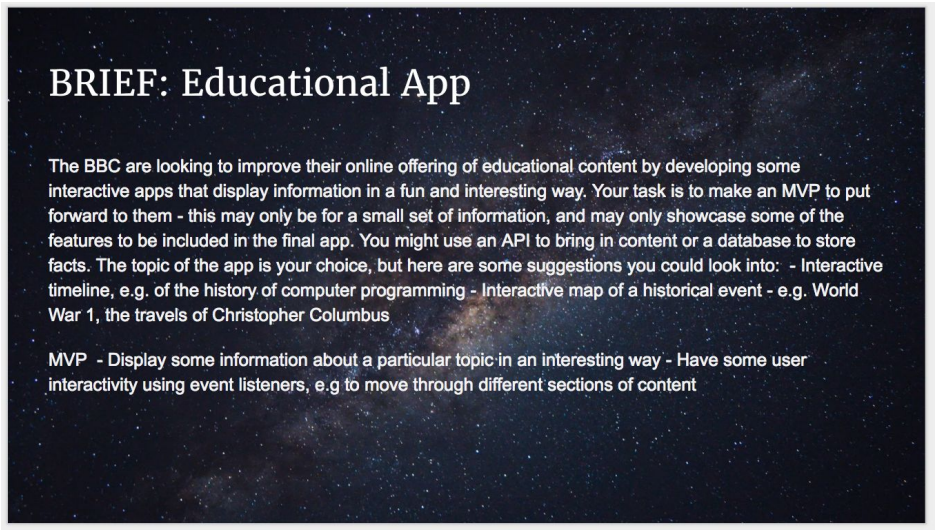


Week 13:

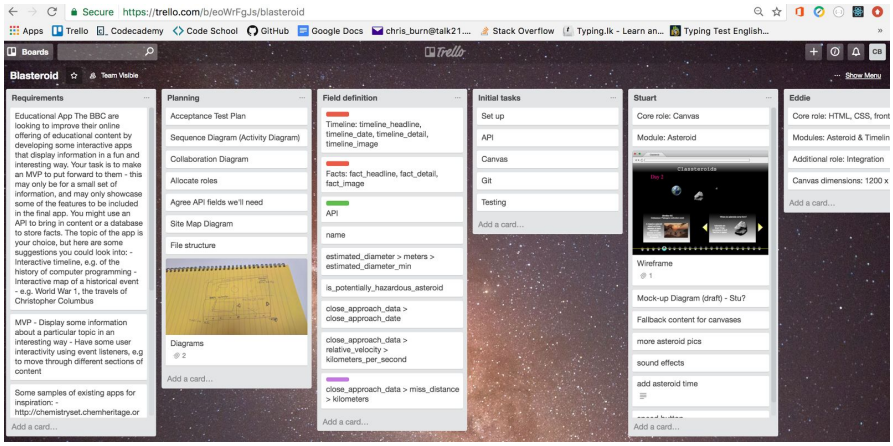
Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.



Take a screenshot of the project brief from your group project.



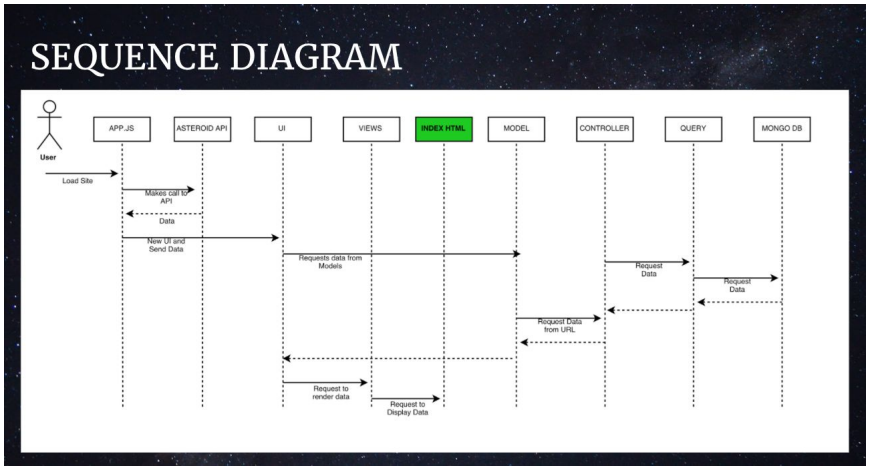
Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.

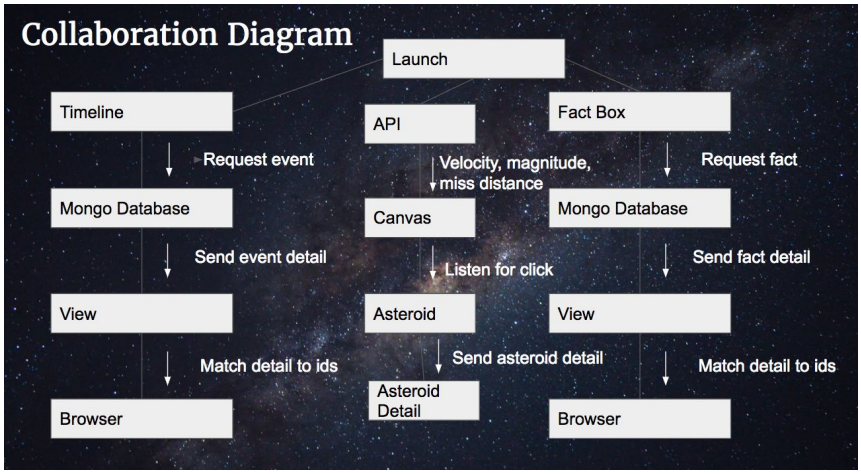


Write an acceptance criteria and test plan.

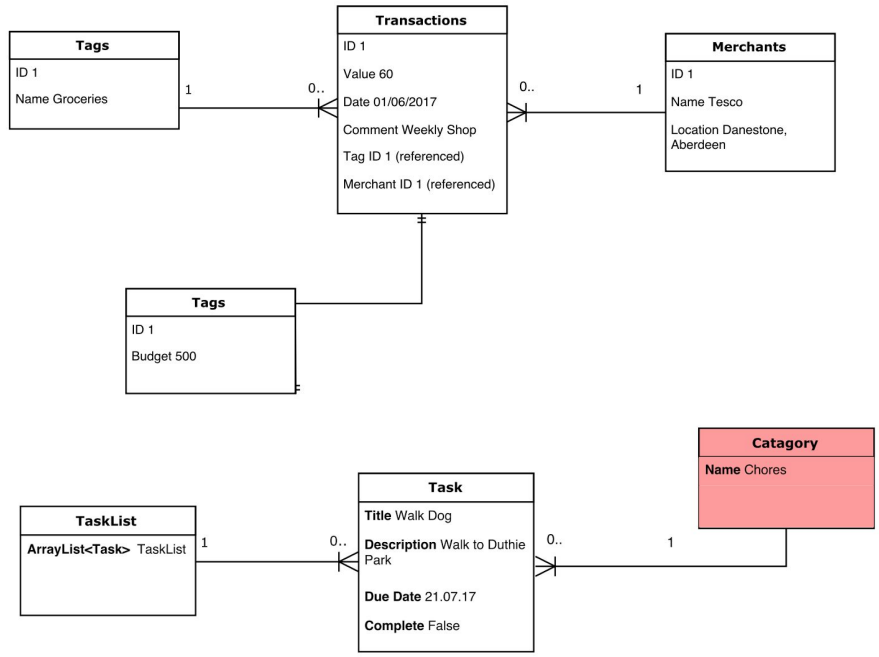
ACCEPTANCE CRITERIA & TEST PLAN		
ACCEPTANCE CRITERIA	EXPECTED RESULT / OUTPUT	PASS / FAIL
Click icon on timeline and view details of event (any event)	Timeline item details box populated with data from database.	PASS
User can view asteroid facts upon load of site, then able to scroll between facts using click button functionality.	Asteroid box autopopulated and can display more facts upon click of 'next fact' arrow.	PASS
Canvas animation should run when prompted by load of site, showing animated asteroids able to display 'real-time' API data.	Display moving asteroid on canvas taking in parameters from asteroid API.	PASS
User can interact with canvas asteroid animations to display data.	Each asteroid is an object in the asteroid API which is able to display object data in window upon click of object.	PASS

Produce two system interaction diagrams (sequence and/or collaboration diagrams).





Produce two object diagrams.



Algorithms:
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.

Weekend homework (Blackjack):
This method populates a 'deck' (ArrayList) with card types and numbers via a nested for loop which loops through a designated set of ENUMS for card type (suits) and loads the ArrayList with a connected set of ENUMS for card numbers. This means for each card type there are a set of card numbers. I.e. Hearts has an ArrayList of (Ace, 2, 3, 4 … King), as does Diamonds

*and Clubs. The result is an ArrayList of 52 cards (4 *13) which makes up the playing deck for the game of Blackjack.*

```
    public void populateDeck() {
        for (CardSuit type : CardSuit.values()){
            for (CardNumber number : CardNumber.values()){
                deck.add(new Card(number, type));
//                  Collections.shuffle(deck);
            }
        }
    }
```

Weekend homework (WordCount Android App, homework):
This method is used to calculate the length of a String input from an EditText box. Setting a counter at the start of the method via the 'words' variable, the for loop iterates over the String and splits the string into separate characters via the split function. It then increments the word count for each input of a blank entry. The words variable is then converted to a String via another variable designation as required by the String method output.

```
    public String getWordCount(String input){
        int words = 0;
        for (String string : input.split(" ")){
            words += 1;
        }
        int number = words;
        String numberToString = Integer.toString(number);
        return numberToString;
    }
```

Produce a bug tracking report - Group Project:

Bug Tracking Report

Issue	Outcome	Solution	Outcome
User must be able to click element on timeline and the details of event shows in asteroid hit box.	Fail The id in asteroid hits view was not matching with the id in index.html.	Ids homogenised; underscore and hyphen conflict resolved.	Pass
Asteroid should spin while approaching.	Fail Asteroids either static or whole screen rotates.	Multiple attempts at spinning asteroids.	Fail Asteroids remain unrotating.
UI should direct data flows and not manipulate data.	Fail UI was parsing data sent from the database.	Methods for manipulating data and populating html elements dynamically moved into separate views.	Pass UI is uncluttered and simple. Views have responsibility for managing their own data and display.