

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	<p>Demonstrate the use of an array in a program. Take screenshots of:</p> <ul style="list-style-type: none"> *An array in a program *A function that uses the array *The result of the function running <p>Description: An array of train station name and used add function to add new stop, Edinburgh Waverley to the list/array.</p>

```

1
2
3 stops = [ "Croy", "Cumbernauld", "Falkirk High", "Linlithgow", "Livingston", "Haymarket" ]
4

2
3 stops = [ "Croy", "Cumbernauld", "Falkirk High", "Linlithgow", "Livingston", "Haymarket" ]
4
5 def add(stops, new_stop)
6   return stops << new_stop
7 end
8 p add(stops, "Edinburgh Waverley")
9
10
11 # stops.push("Edinburgh Waverley")

```

Ruby Tasks - user@users-MacBook-Pro-6:~/Ruby_Tasks - zsh ->

[Ruby_Tasks- git:(master) ✘ ruby task_A.rb
["Croy", "Cumbernauld", "Falkirk High", "Linlithgow", "Livingston", "Haymarket", "Edinburgh Waverley"]
Ruby_Tasks- git:(master) ✘]

Unit	Ref	Evidence
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running
		Description: A hash of parts in the UK and with information. The function total_population shows the sum of the population from Scotland, Wales and England in the UK.

```

1  united_kingdom = [
2    {name: "Scotland",population: 5295000,capital: "Edinburgh"}, 
3    {name: "Wales",population: 3063000,capital: "Swansea"}, 
4    {name: "England",population: 53010000,capital: "London"} 
5  ]

```

countries.rb

```

21
22  def total_population(countries)
23    total = 0
24    for country in countries
25      total += country[:population]
26    end
27    return total
28  end
29
30  p total_population(united_kingdom)
31

```

Ruby_Tasks- — user@users-MBP-6 — ..W/Ruby_Task...

```
[→ Ruby_Tasks- git:(master) ✘ ruby countries.rb
61368000
→ Ruby_Tasks- git:(master) ✘ ]
```

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: Function is called <code>find_by_name</code> and it show up the value of the name key, which is "Han_Solo".

```

47
48 def Bounty.find_by_name(name)
49   db = PG.connect({dbname:
50     "space_cowboy", host: "localhost"})
51   sql = "SELECT * FROM bounties WHERE
52     name = $1"
53   values = [name]
54   db.prepare("find_by_name", sql)
55   result =
56   db.exec_prepared("find_by_name", values)
57   db.close()
58   if result.count != 0
59     return Bounty.new(result[0])
60   else
61     return nil
62   end
63
64
1  require("pry")
2 require_relative("models/space_cowboy.rb")
3
4 cowboy1 = Bounty.new(
5   'name' => "Han_Solo",
6   'danger_level' => "Medium",
7   'favourite_weapon' => "Blaster",
8   'bounty_level' => "100"
9 )
10
11 pry(main)> Bounty.find_by_name("Han_Solo")
12 => #<Bounty:0x007fb6dfb44f68
13   @bounty_level=100,
14   @danger_level="Medium",
15   @favourite_weapon="Blaster",
16   @id=12,
17   @name="Han_Solo">
18
19
20 binding.pry
21 nil

```

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: Data has screening time and tickets number. The function popular_time shows the screening time sold the most tickets.

```

13 Screening.delete_all()
14
15 screening1 = Screening.new({ 'screen_time' => '10:00'})
16 screening1.save()
17 screening2 = Screening.new({ 'screen_time' => '12:00'})
18 screening2.save()
19 screening3 = Screening.new({ 'screen_time' => '14:00'})
20 screening3.save()
21

```

```

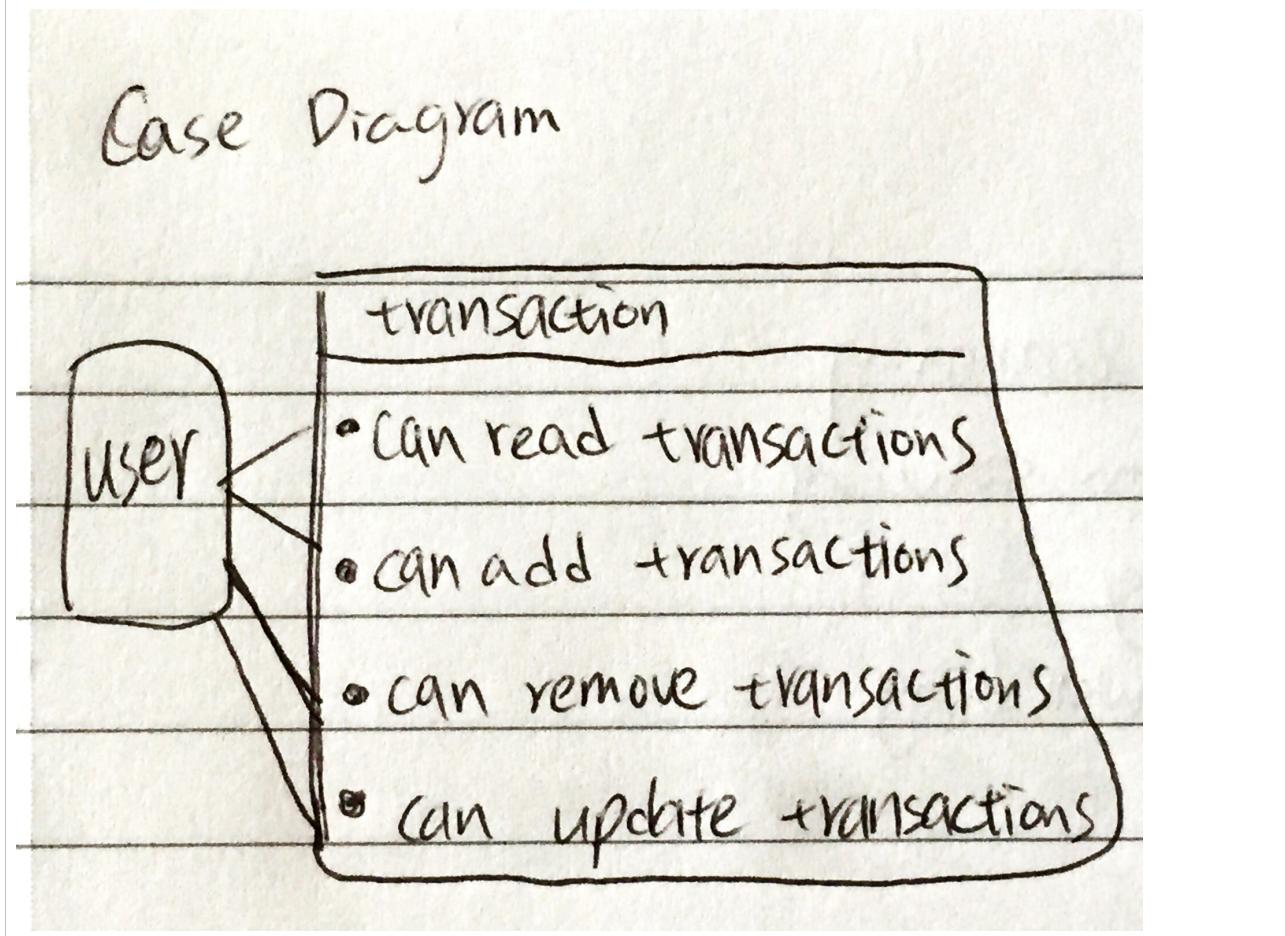
29 def popular_time()
30   sql = "SELECT screen_time, COUNT(tickets.*) FROM tickets INNER JOIN
31   screenings ON tickets.screening_id = screenings.id GROUP BY screen_time
32   ORDER BY count DESC limit 1"
33   values = []
34   tickets_count = SqlRunner.run(sql, values)
35   return tickets_count.first["screen_time"]
36 end
37

```

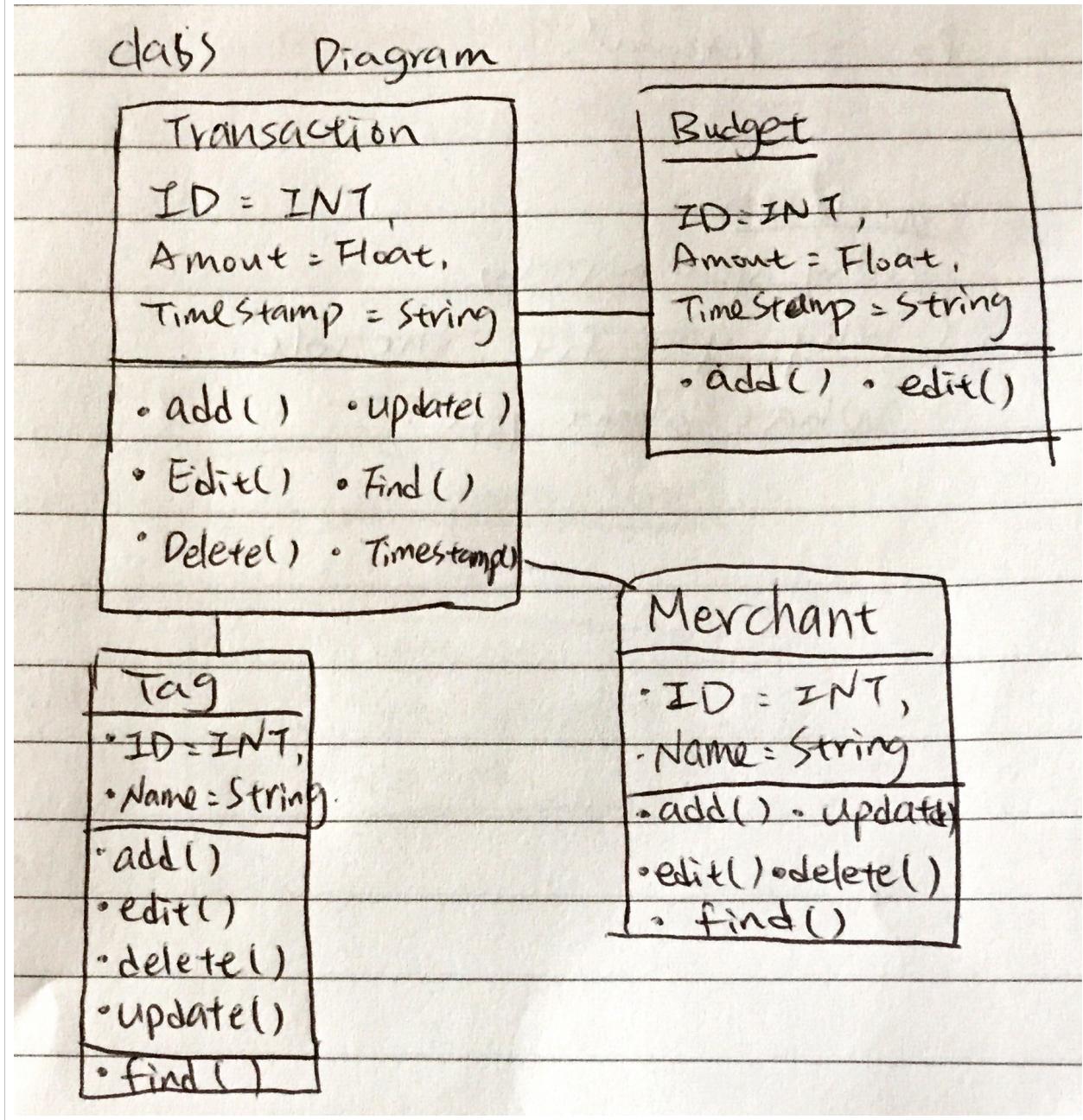
```
[1] pry(main)> screening1.popular_time
=> "10:00"
[2] pry(main)>
[3] pry(main)>
```

Week 5 and 6

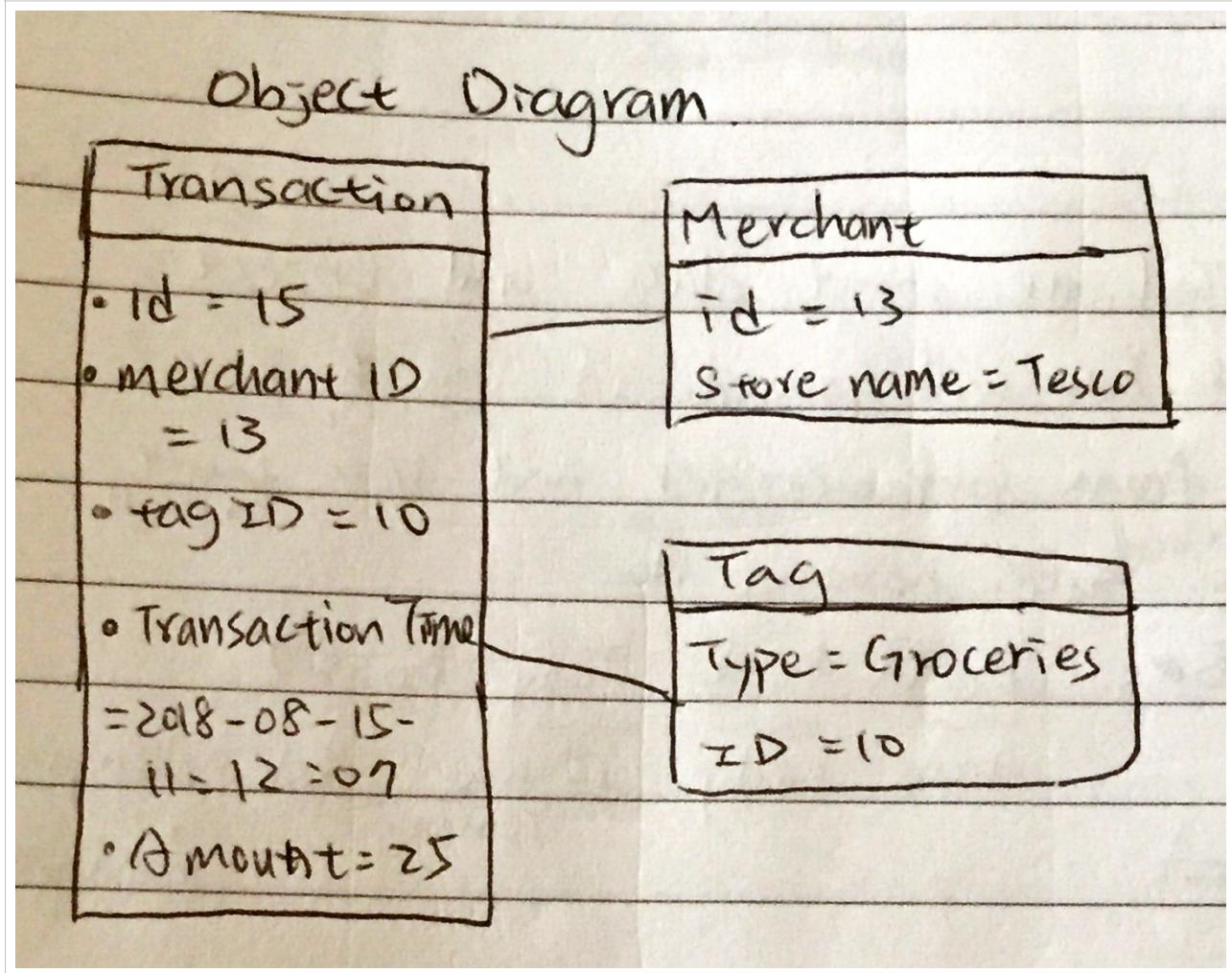
Unit	Ref	Evidence
A&D	A.D.1	A Use Case Diagram The functions of the spending tracking system that user can interact with.



Unit	Ref	Evidence
A&D	A.D.2	A Class Diagram
		The types of items and functions in different classes that used in spending tracking system



Unit	Ref	Evidence
A&D	A.D.3	An Object Diagram
		Description: the relationship between transaction table and merchant/tag table.



Unit	Ref	Evidence
A&D	A.D.4	An Activity Diagram
		The route of user using add new transaction info to spending tracking system.

Activity Diagram

```

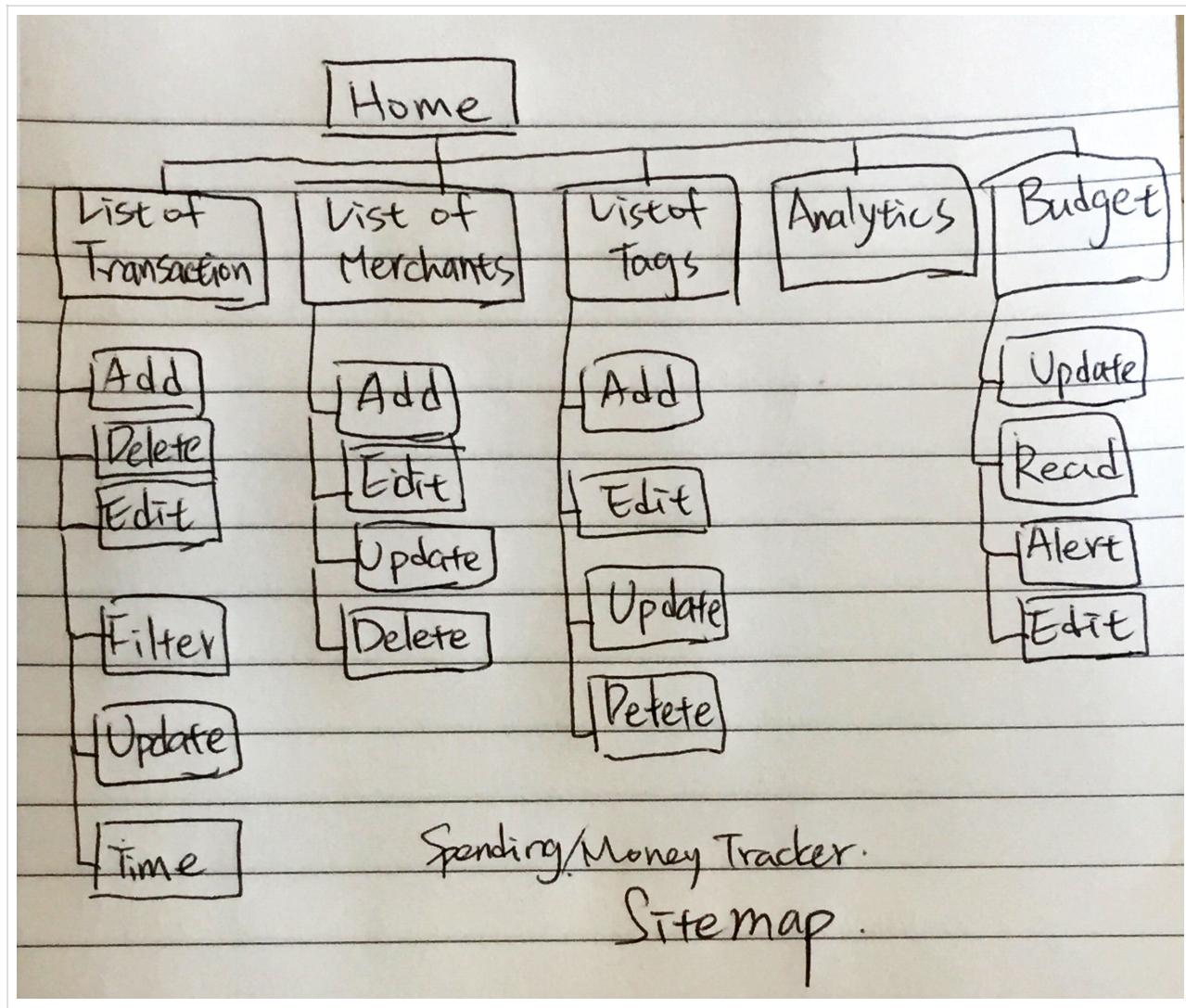
graph TD
    User((User)) -- "User log in" --> Start(( ))
    Start -- "Click Add" --> DisplayForm[Display Form]
    DisplayForm --> Decision1{ }
    Decision1 --> InputInfo[Input Info]
    InputInfo -- "Adding Item" --> AcceptingItem[Accepting added Item]
    AcceptingItem --> Decision2{ }
    Decision2 --> MissingInfo[Missing Info]
    MissingInfo --> Decision3{ }
    Decision3 --> AddingList[Adding to list  
Time Stamp]
    AddingList --> ShowList[Show list]
    ShowList --> End((( )))
    
```

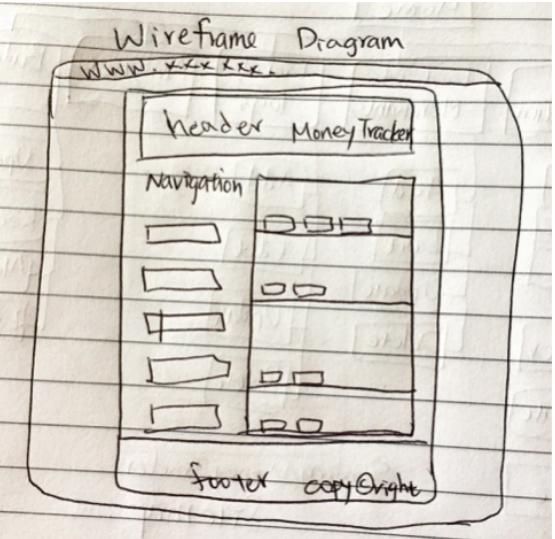
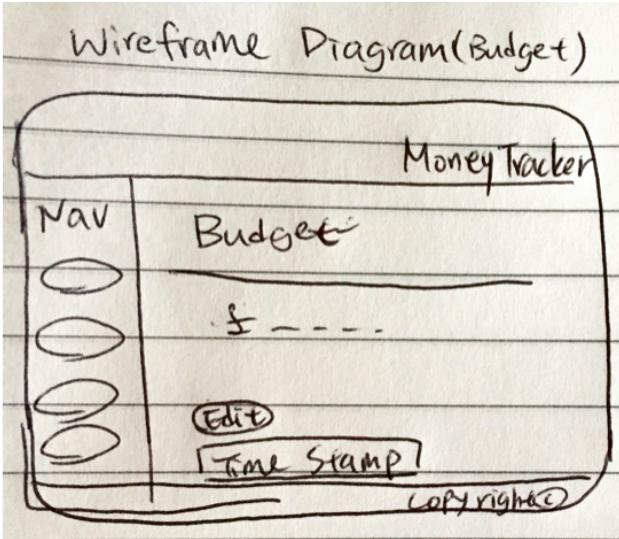
The diagram illustrates the workflow for adding a new transaction. It starts with a user logging in. The user then clicks 'Add', which triggers a 'Display Form' activity. From the display form, the flow splits into two parallel paths. One path leads to 'Input Info' (labeled 'Adding Item'), which then leads to 'Accepting added Item'. From there, it branches again: one path leads to 'Missing Info', which then leads to a decision point; the other path leads directly to 'Adding to list Time Stamp'. Both paths converge at the decision point, which then leads to 'Show list', and finally to the end state.

Unit	Ref	Evidence	
A&D	A.D.6	<p>Produce an Implementations Constraints plan detailing the following factors:</p> <ul style="list-style-type: none"> *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time 	
		Description: The difficulties and solutions.	

Topic	Possible Effect of Constraint on Product	Solution
Hardware and software platforms	Content may display differently in Mobile screen.	Plan for mobile and set up in CSS.
Performance requirements	Occurring errors.	Regularly checking app performance and ask third person to test the product.
Persistent storage and transactions	Memory storage may not be enough.	Regular clear out system automatically. Extend to bigger memory space.
Usability	For people have color blind.	Different font and size or add extra underlines.
Budgets	Limited budget.	Use more free online resources.
Timelimitations	One week	Follow the plan and well organised teamwork

Unit	Ref	Evidence
P	P.5	User Site Map
		Description: The navigation options in transaction webpage and its' own functions.



Unit	Ref	Evidence
P	P.6	2 Wireframe Diagrams
Description: The frame work of spending tracker application website.		
 		

Unit	Ref	Evidence
P	P.10	Example of Pseudocode used for a method
Description: A plan of a functionality, <code>filter_by_tag_and_merchant()</code> in transaction class in spending tracker application.		

This function should filter out transactions with specific tag and merchants and display in time desc order, `filter_by_tag_and_merchant() {`

Each filter require one tag type and one merchant store name

A list of tag type

A list of merchant store names

Each transaction should have timestamp

Filter out transactions with tags are not the same as the selected tag

Filter out transactions with merchants are not the same as the selected merchant

Organise selected transactions in Desc time order.

}

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: Add New Merchant in to merchant list.	

The screenshot shows the Money Tracker application interface. At the top, there's a dark blue header with the title "Money Tracker". Below the header, on the left, is a sidebar with a "Navigation" section containing links: "List of Transactions", "List of Merchants", "List of Tags", "Analytics", and "Budget". The main content area has a light blue background. It features a heading "New Merchant" and a form field labeled "Merchant Store Name:" with the value "Tesco" entered. Below the form is a button labeled "save new merchant store". At the bottom right of the main area, there's a copyright notice: "Copyright © 2018 YingYing Chen."

The screenshot shows the Money Tracker application interface. At the top, there's a dark blue header with the title "Money Tracker". Below the header, on the left, is a sidebar with a "Navigation" section containing links: "List of Transactions", "List of Merchants", "List of Tags", "Analytics", and "Budget". The main content area has a light blue background. It features a heading "All Merchants" and a button labeled "Add New Merchant". Below this, there are three entries, each representing a merchant: "Aldi", "Scotrail", and "CodeClan". Each entry has a "Delete" and an "Edit Merchant" button. At the bottom right of the main area, there's a copyright notice: "Copyright © 2018 YingYing Chen."

The screenshot shows a web application interface. On the left, there is a navigation sidebar with links: "List of Transactions", "List of Merchants", "List of Tags", "Analytics", and "Budget". The main content area has a title "All Merchants" and a button "Add New Merchant". Below this, there is a list of four merchants, each with a "Delete" and "Edit Merchant" button:

- Aldi
- Scotrail
- CodeClan
- Tesco

Below the merchant list, there is a terminal-like window displaying PostgreSQL (psql) command-line output:

```
[psql (10.4)
Type "help" for help.

moneytracker=# select * from merchants;
 id | store_name
----+-----
 3  | Aldi
 4  | Scotrail
 5  | CodeClan
 6  | Tesco
(4 rows)

moneytracker=# ]
```

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: Update merchant name in transaction.

Money Tracker

Navigation

- [List of Transactions](#)
- [List of Merchants](#)
- [List of Tags](#)
- [Analytics](#)
- [Budget](#)

All Money Transactions

Transaction Tag: Merchant:

Amount: 50.0

Merchant: Desco00000

Transaction Tag: Groceries

Transaction Time: 17 August, 2018

Copyright © 2018 YingYing Chen

```
moneytracker=# select * from merchants
moneytracker-# ;
[ id | store_name
[----+-----
  3 | Aldi
  4 | Scotrail
  8 | Desco00000
(3 rows)

moneytracker=#

```

Money Tracker

Navigation

- [List of Transactions](#)
- [List of Merchants](#)
- [List of Tags](#)
- [Analytics](#)
- [Budget](#)

Merchant Store Name:

Copyright © 2018 YingYing Chen

Money Tracker

All Money Transactions

Transaction Tag: Merchant:

Amount: 50.0
Merchant: Tesco
Transaction Tag: Groceries
Transaction Time: 17 August, 2018

Copyright © 2018 YingYing Chen

```
moneytracker=# select * from merchants;
+----+-----+
| id | store_name |
+----+-----+
| 3  | Aldi      |
| 4  | Scotrail   |
| 8  | Tesco      |
+----+
(3 rows)

moneytracker=#

```

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: Filter out transactions with tag, Groceries and merchant, Tesco.

Money Tracker

Navigation

[List of Transactions](#) [List of Merchants](#) [List of Tags](#) [Analytics](#) [Budget](#)

All Money Transactions

Transaction Tag: Merchant: [filter transactions](#)

[Add New Transaction](#) [Merchant List](#) [Tag List](#)

Amount: 23.0

Merchant: Tesco

Transaction Tag: Groceries

Transaction Time: 17 August, 2018

[Delete Transaction](#) [Edit Transaction](#)

Amount: 33.0

Merchant: Scotrail

Transaction Tag: Transport

Transaction Time: 17 August, 2018

[Delete Transaction](#) [Edit Transaction](#)

Amount: 25.0

Merchant: Tesco

Transaction Tag: Groceries

Transaction Time: 17 August, 2018

[Delete Transaction](#) [Edit Transaction](#)

Amount: 67.0

Merchant: Aldi

Transaction Tag: Groceries

Transaction Time: 17 August, 2018

[Delete Transaction](#) [Edit Transaction](#)

Copyright © 2018 YingYing Chen.

Navigation

[List of Transactions](#)
[List of Merchants](#)
[List of Tags](#)
[Analytics](#)
[Budget](#)

All Money Transactions

Transaction Tag: Merchant: [filter transactions](#)

[Add New Transaction](#) [Merchant List](#) [Tag List](#)

Amount: 23.0
Merchant: Tesco
Transaction Tag: Groceries
Transaction Time: 17 August, 2018

[Delete Transaction](#) [Edit Transaction](#)

Amount: 25.0
Merchant: Tesco
Transaction Tag: Groceries
Transaction Time: 17 August, 2018

[Delete Transaction](#) [Edit Transaction](#)

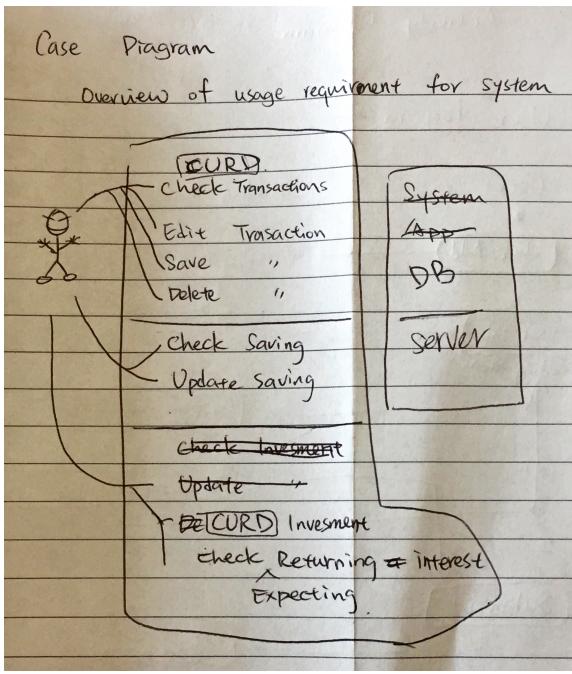
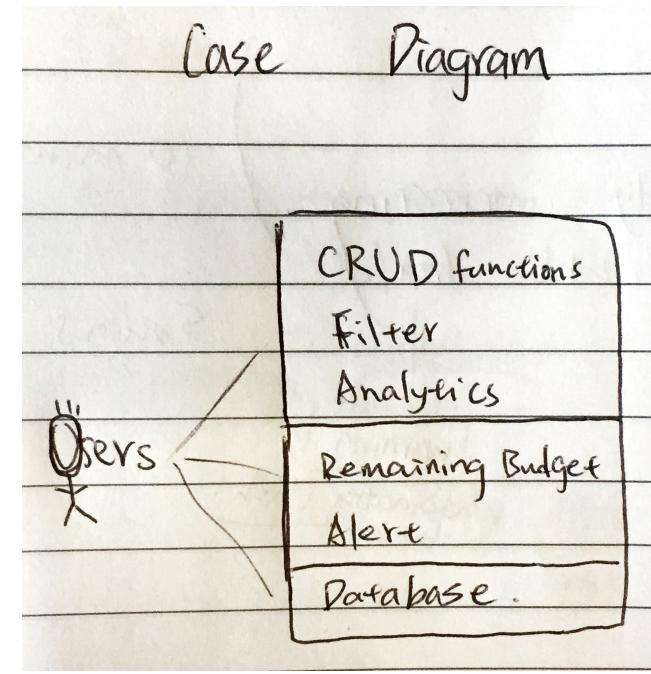
Copyright © 2018 YingYing Chen

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: https://github.com/YYChen01988/Transaction_Tracker_Project

The screenshot shows the 'Money Tracker' application interface. On the left, there is a navigation sidebar with links: 'List of Transactions', 'List of Merchants', 'List of Tags', 'Analytics', and 'Budget'. The main content area is titled 'All Money Transactions'. It displays three transaction entries:

- Amount: 25.0
Merchant: Tesco
Transaction Tag: Groceries
Transaction Time: 17 August, 2018
[Delete Transaction](#) [Edit Transaction](#)
- Amount: 67.0
Merchant: Aldi
Transaction Tag: Groceries
Transaction Time: 17 August, 2018
[Delete Transaction](#) [Edit Transaction](#)
- Amount: 100.0
Merchant: Scotrail
Transaction Tag: Transport
Transaction Time: 17 August, 2018
[Delete Transaction](#) [Edit Transaction](#)

At the bottom right of the main content area, there is a copyright notice: 'Copyright © 2018 YingYing Chen.'

Unit	Ref	Evidence
P	P.12	<p>Take screenshots or photos of your planning and the different stages of development to show changes.</p> <p>Description: The picture on the left is at the first stage of Spending tracker project and the right picture is a new case diagram in the end of project.</p>
		 <p>Case Diagram Overview of usage requirement for system</p> <p>The diagram illustrates the initial usage requirements for a spending tracker system. It features a stick figure representing a user interacting with a central 'System APP DB SERVER' block. The user has access to three main functions: CURD (Check Transactions, Edit Transaction, Save, Delete), CURD (Check Saving, Update Saving), and CURD (Investment, Check Returning ≠ interest, Expecting).</p>  <p>Case Diagram</p> <p>The final case diagram shows the evolution of the system. The user interacts with a 'CRUD functions' block, which now includes 'Filter', 'Analytics', 'Remaining Budget', 'Alert', and 'Database' features. The original CURD requirements have been integrated into this more advanced set of functions.</p>

Week 7

Unit	Ref	Evidence
P	P.16	<p>Show an API being used within your program. Take a screenshot of:</p> <ul style="list-style-type: none"> * The code that uses or implements the API * The API being used by the program whilst running <p>Description: The link of the API in the example application (https://restcountries.eu/rest/v2/all)</p>

```

1 const PubSub = require('../helpers/pub_sub.js');
2
3 const SelectView = function(select){
4   this.select = select;
5 }
6
7 SelectView.prototype.bindEvents = function(){
8   PubSub.subscribe("CurrencyList:names-ready", (event) => {
9     this.populateList(event.detail);
10    });
11
12   this.select.addEventListener('change', (event) => {
13     PubSub.publish("SelectView:currency_name-selected", event.target.value);
14   });
15 };
16
17 SelectView.prototype.populateList = function(names){
18   names.forEach((name, index) => {
19     const option = document.createElement('option');
20     option.textContent = name;
21     option.value = index;
22     this.select.appendChild(option);
23   });
24 };
25
26 module.exports = SelectView;
27

```

Currency Map

For selecting the currency, you will see the details of the nation, who use the currency and the location of the nation.

[GitHub](#)

Please select a currency

Australian dollar

Antarctica

- Region: Polar
- Capital:
- Currency: Australian dollar, British pound

Australia

- Region: Oceania
- Capital: Canberra
- Currency: Australian dollar

Christmas Island

- Region: Oceania



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

const MapWrapper = function(container){
  this.container = container;
  this.coords = [55.8654192, -4.25802099999999];
  this.map = L.map(this.container);
  this.osmLayer = new L.TileLayer("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png");
  this.map.setView(this.coords, 10).addLayer(this.osmLayer);
  this.markers = []
}

MapWrapper.prototype.bindEvents = function(){
  PubSub.subscribe("CurrencyList:country-ready", (event) => {
    // map.removeLayer(marker);
    this.markers.forEach((m)=>{
      this.map.removeLayer(m);
    });
  });
}
```

Unit	Ref	Evidence	
P	P.18	<p>Demonstrate testing in your program. Take screenshots of:</p> <ul style="list-style-type: none"> * 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: Running unit test with ruby code.	

```
require_relative("../card.rb")
require_relative("../testing_task_2.rb")

class CardTest < MiniTest::Test
  def setup
    @diamond1 = Card.new("diamond", 1)
    @cardgame1 = CardGame.new()
  end

  def test_check_for_Ace()
    result = @cardgame1.check_for_Ace(@diamond1)
    assert_equal(result, true)
  end
end
```

```
pda_static_and_dynamic_testing_tasks ruby spec/testing_task_spec.rb
Run options: --seed 60332

# Running:

E

Finished in 0.001014s, 986.1930 runs/s, 0.0000 assertions/s.

1) Error:
CardTest#test_check_for_Ace:
NoMethodError: undefined method `value=' for #<Card:0x007fc99c03eec8 @suit="diamond", @value=1>
Did you mean? value
  /Users/user/codeclan_work/pda_static_and_dynamic_testing_tasks/testing_task_2.rb:11:in `check_for_Ace'
  spec/testing_task_spec.rb:13:in `test_check_for_Ace'

1 runs, 0 assertions, 0 failures, 1 errors, 0 skips
pda_static_and_dynamic_testing_tasks ruby spec/testing_task_spec.rb
```

```
2 require_relative("../card.rb")
3 require_relative("../testing_task_2.rb")
4
5
6 class CardTest < MiniTest::Test
7   def setup
8     @diamond1 = Card.new("diamond", 1)
9     @cardgame1 = CardGame.new()
10   end
11
12  def test_check_for_Ace()
13    result = @cardgame1.check_for_Ace(@diamond1)
14    assert_equal(result, true)
15  end
16 end
17
```

```
pda_static_and_dynamic_testing_tasks ruby spec/testing_task_spec.rb
Run options: --seed 33581

# Running:

.

Finished in 0.000835s, 1197.6047 runs/s, 1197.6047 assertions/s.

1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
pda_static_and_dynamic_testing_tasks ruby spec/testing_task_spec.rb
Run options: --seed 58187

# Running:

.

Finished in 0.000866s, 1154.7343 runs/s, 1154.7343 assertions/s.

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

```

def self.cards_total(cards)
  total=0
  for card in cards
    total += card.value
  return "You have a total of" + total
end
end
# Running:
E..
Finished in 0.000988s, 3036.4370 runs/s, 2024.2913 assertions/s.

  1) Error:
CardTest#test_cards_total:
NoMethodError: undefined method `cards_total' for #<CardGame:0x007fe5938276e0>
spec/testing_task_spec.rb:25:in `test_cards_total'
'

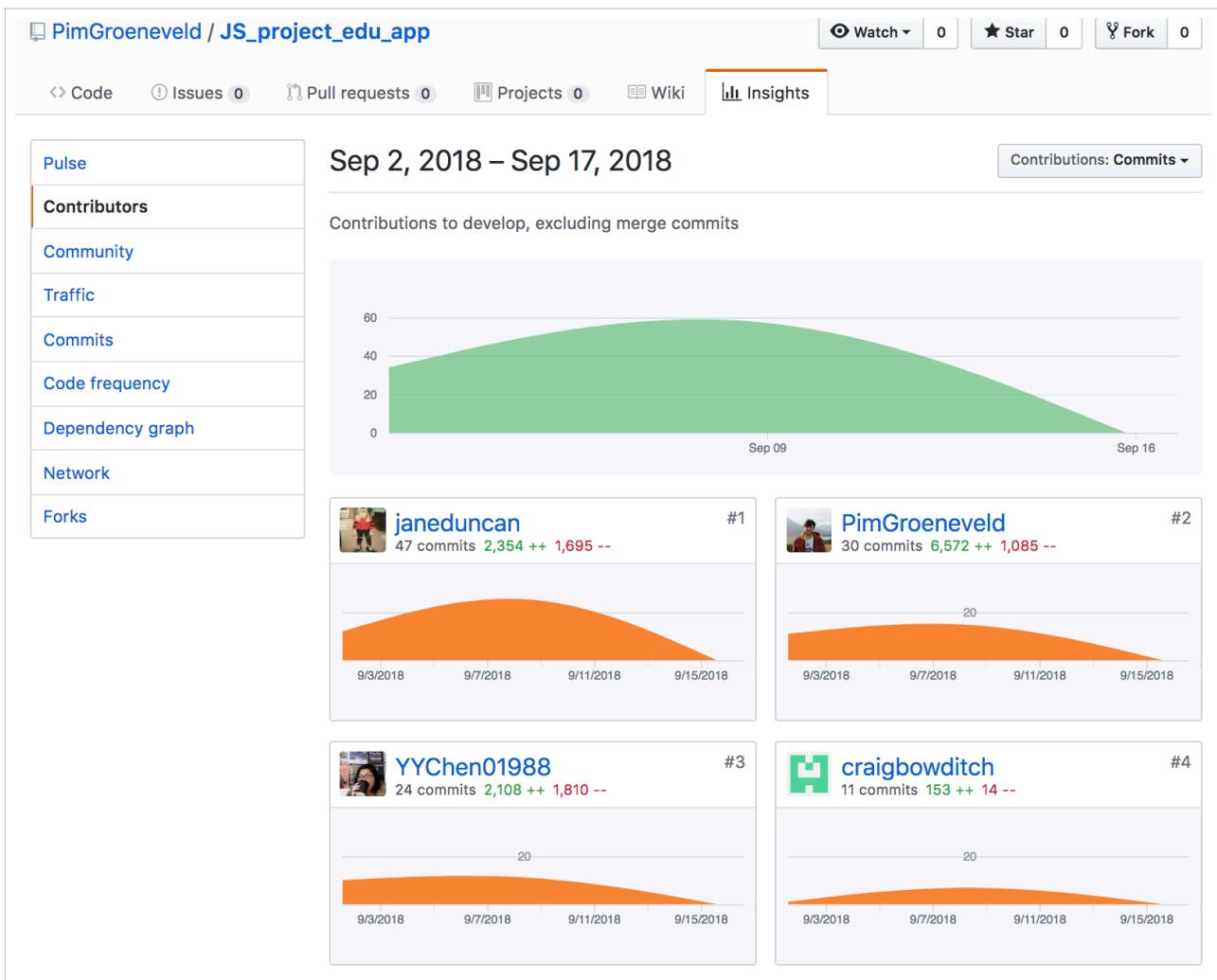
3 runs, 2 assertions, 0 failures, 1 errors, 0 skips
→ pda_static_and_dynamic_testing_tasks

```

testing_ta...	customer.rb	testing_ta...	card.rb	customer_spec.rb	testing_task_spec.rb
o				6 class CardTest < MiniTest::Test	...
9	def check_for_Ace(card)			7 def setup	Finished in 0.000952s, 3151.2596 runs/s, 3151.2596 assertions/s.
10	if card.value == 1			8 @diamond1 = Card.new("diamond", 1)	3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
11	return true			9 @club2 = Card.new("club", 2)	→ pda_static_and_dynamic_testing_tasks ruby spec/test
12	else			10 @cardgame1 = CardGame.new()	Run options: --seed 49566
13	return false			11 @cards= [@diamond1, @club2]	# Running:
14	end			12 end	E..
15	end			13	Finished in 0.001076s, 2788.1034 runs/s, 1858.7356 assertions/s.
16	def highest_card(card1,			14 def test_check_for_Ace()	1) Error:
.	card2)			15 result =	CardTest#test_cards_total:
17	if card1.value > card2.value			16 @cardgame1.check_for_Ace(@diamond1)	NoMethodError: undefined method `cards_total' for #<Ca
18	return card1.suit			17 assert_equal(true, result)	rdGame:0x007fd7ad834bb8>
19	else			18 end	spec/testing_task_spec.rb:25:in `test_cards_total'
20	card2.suit			19	3 runs, 2 assertions, 0 failures, 1 errors, 0 skips
21	end			20	→ pda_static_and_dynamic_testing_tasks ruby spec/test
22	end			21	Run options: --seed 25510
23	def cards_total(cards)			22	# Running:
24	total=0			23	...
25	for card in cards do			24	Finished in 0.000970s, 3092.7822 runs/s, 3092.7822 assertions/s.
26	total += card.value			25	3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
27	end			26	→ pda_static_and_dynamic_testing_tasks
28	return "You have a total of			27	
29	" + total.to_s			28	
30	end			29	
31	end			30	
	testing_task_2.rb 29:4			31	
					LF UTF-8 Ruby 0 fil

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: https://github.com/PimGroeneveld/JS_project_edu_app/graphs/contributors



Unit	Ref	Evidence
P	P.2	Take a screenshot of the project brief from your group project.
		Description: This is the details of our project MVP and additional extension parts.

README.md

Educational App

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

MVP

Education Language App

A user should be able to:

- Select a language from a dropdown menu
- A flashcard then displays a phrase in the chosen language
- Click a button to reveal the phrase in English
- Click a button to listen to audio of phrase

Extensions

- Word guessing game
- Facts section (using an API to display map)
- Machine learning API
- Add phonetic spelling of phrase to flashcard

API, Libraries, Resources

- <https://cloud.google.com/maps-platform/> An open-source library for rendering maps and map functionality.
- <https://developer.oxforddictionaries.com> An open-source dictionary API (Pim has key)

(Put the API key file in .gitignore)

Unit	Ref	Evidence
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.
		Description: Trello shows the features and functions we want to finished in a time scale. https://trello.com/b/Hdht0ySA/javascript-project

Unit	Ref	Evidence
P	P.4	Write an acceptance criteria and test plan.
		The acceptance criteria for language learning application for group project.

Acceptance Criteria	Expected Result/output	Pass/Fail
Simple user interface	Clean flashcard view	Passed
Practical Content	Add audio and extra information about the foreign languages	Passed
Fun in Learning	Add Quiz part	Passed

Unit	Ref	Evidence
P	P.7	<p>Produce two system interaction diagrams (sequence and/or collaboration diagrams).</p> <p>Description: user select change and pass it to database and cause the response to other functions.</p> <pre> sequenceDiagram participant User participant Server participant DataBase User->>Server: dropdown-list activate Server Note over Server: model: flashcar.js ch:Languages:languages-data-ready Note over Server: select_view.js ch: SelectView:change Note over Server: model: flashcar.js ch:Flashcard:selected-language-and-answer Note over Server: model: flashcar_list_view.js ch:Flashcard:selected-language-and-answer User-->Server: selected option activate DataBase DataBase-->Server: deactivate DataBase Server-->User: flash card deactivate Server </pre>
		<pre> sequenceDiagram participant User participant Server participant DataBase User->>Server: activate Server Note over Server: Add_form_view.js Pub Ch: "AddWordFormView: item-submitted" Note over Server: Model: FlashCard.js Pub Ch: " Map: countries-objects-ready" Note over Server: MappWrapper.js activate DataBase DataBase-->Server: deactivate DataBase Server-->User: deactivate Server </pre>

Paste Screenshot here

Unit	Ref	Evidence
P	P.8	Produce two object diagrams. Description: Fist Object Diagram shows a shop sells items, which includes Guitar and Apple Juice. Second Object Diagram shows two Kaiju objects but they are not related.
		<p>Object Diagram1. MusicShop Project (Shop sells items, which includes Guitar and Apple Juice)</p> <pre>graph TD; Shop[Shop] --> Guitar[Guitar]; Shop --> AppleJuice[AppleJuice]</pre> <p>The diagram illustrates an object structure. At the top is a rounded rectangle labeled "Shop". Inside "Shop", the following attributes are listed: stocks = arrayList soldItems = arrayList</p> <p>Two arrows point downwards from "Shop" to two separate rounded rectangles below it. The left rectangle is labeled "Guitar" and contains the following attributes: material:"Wood", color:"Black", boughtPrice: 200, sellPrice:450, stringNumber: 6, InstrumentType: GUITAR</p> <p>The right rectangle is labeled "AppleJuice" and contains the following attributes: type: "Apple Juice", boughtPrice: 5, sellPrice: 7, JuiceType: APPLE</p>

Object Diagram2. Kaiju Project

(**G**ozilla and **M**othra inherit from the abstract class **Kaiju**. However, **G**ozilla and **M**othra are not related.)



Paste Screenshot here

Unit	Ref	Evidence
P	P.17	Produce a bug tracking report
		Description: Error show up in the group project.

Debugging Event	Status	Solution	Result
Empty lines in the end of seeds.js file	Error, Mongoldb can not read the file	Remove empty lines	Pass. Data displayed in Mongodb
Missing comma in the seeds.js	Error, Mongoldb can not read the file	Add the comma	Pass. Data displayed in Mongodb
Database name is incorrect in the server.js	Insomnia can not find the route	Change the dbname to correct one	Route available, passed.
Typo in server.js (bdoy.parser)	Insomnia shows error	Correct typo	Passed
Logic error; try to get value from key [language.translation[0]	Undefined content	Console log event and pick the data we want. [language.translation[event.detail]]	Passed

[Paste Screenshot here](#)

Week 12

Unit	Ref	Evidence
I&T	I.T.7	<p>The use of Polymorphism in a program and what it is doing.</p> <p>Description: two classes using same interface e.g.. instrument class and Juice class using ISell interface, so they can be sell in the shop</p> <pre> src └── main └── java └── behaviour ├── IPlay ├── ISell └── Upgradable └── instruments ├── Guitar └── Instrument ├── InstrumentType ├── Piano └── Trumpet └── juiceBar ├── AppleJuice ├── Juice ├── JuiceType └── PapayaJuice └── Shop └── behaviour ├── IPlay ├── ISell └── Upgradable └── instruments ├── Guitar └── Instrument ├── InstrumentType ├── Piano └── Trumpet └── juiceBar ├── AppleJuice ├── Juice ├── JuiceType └── PapayaJuice └── Shop 5 public abstract class Instrument implements IPlay, ISell { 6 private String material; 7 private String color; 8 private int boughtPrice; 9 private int sellPrice; 10 11 public Instrument(String material, String color, int boughtPrice, 12 int sellPrice) { 13 this.material = material; 14 this.color = color; 15 this.boughtPrice = boughtPrice; 16 this.sellPrice = sellPrice; 17 } 18 19 public String getMaterial() { 20 21 22 public class Juice implements ISell, Upgradable { 23 private int boughtPrice; 24 private int sellPrice; 25 26 public Juice(int boughtPrice, int sellPrice) { 27 this.boughtPrice = boughtPrice; 28 this.sellPrice = sellPrice; 29 } 30 31 public int getBoughtPrice() { 32 return boughtPrice; 33 } 34 } 35 36 37 38 39 3 </pre>

Unit	Ref	Evidence
A&D	A.D.5	<p>An Inheritance Diagram</p> <p>Description: Like class diagram but subclasses have arrow to superclass</p> <pre> Instrument Private String material; Private String color; Private int boughtPrice; Private int sellPrice; getMaterial(), getColor(), getBoughtPrice(), getSellPrice(), play(), calculateMarkup() Piano Private InstrumentType type getType() Trumpet Private int valvesNumbers; Private InstrumentType Type getType(), getValvesNumber() </pre>

Unit	Ref	Evidence
I&T	I.T.1	<p>The use of Encapsulation in a program and what it is doing.</p> <p>Description: name is private, so the name can not be modified directly.</p>

```

public class SweetSlot extends Slot {

    private String name;
    private ArrayList<Sweet> sweets;
    private ProductType productType;
    public SweetSlot(double price, int code, int capacity) {
        super(price, code, capacity);
        this.productType = ProductType.SWEET;
        this.sweets = new ArrayList<>();
        this.name = "Chocolate";
    }

    public String getName() {
        return name;
    }

    @Test
    public void changeName() {
        sweetSlot.name = "Candy";
        assertEquals("Candy", sweetSlot.getName());
    }
}

```

Unit	Ref	Evidence	
I&T	I.T.2	<p>Take a screenshot of the use of Inheritance in a program. Take screenshots of:</p> <ul style="list-style-type: none"> *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: Godzilla inherits from Kaiju class.	

```

public abstract class Kaiju {
    private String name;
    private int healthValue;
    private int attackValue;
    public Kaiju(String name, int healthValue, int attackValue) {
        this.name = name;
        this.healthValue = healthValue;
        this.attackValue = attackValue;
    }
    public void attack(Vehicle vehicle){
        vehicle.healthValue -= this.attackValue;
    }
}
public class Godzilla extends Kaiju{
    public Godzilla(String name, int healthValue, int attackValue) {
        super(name, healthValue, attackValue);
    }
    public void attack(Vehicle vehicle){
        vehicle.healthValue -= this.getAttackValue()*2;
    }
}

```

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: BlackJack project. I choose them is because each of they stands for simple and complicate algorithms.

```

public void playersDecide(ArrayList<Player> players, Dealer dealer){
    for (Player player : players){
        if (player.getHandValue() <= 16){
            Card card = dealer.deal(deck);
            player.getCard(card);
        }
    }
}

```

In the BlackJack, the player meant to take another card if the total cards value is less than 16.

```

public ArrayList<Player> getWinners(ArrayList<Player> players, Dealer dealer){
    //check who is winner 1. dealer busted(players who stay win
    ArrayList<Player> winners = new ArrayList<>();
    if (dealer.getHandValue() > 21){

        for (Player player : players){
            if (player.getBusted() == false){
                winners.add(player);
            }
        }
        return winners;
    }else{
        Boolean isDraw = false;
        for (Player player: players){
            //!player.getBusted() == false
            if (player.getHandValue() >= dealer.getHandValue() && !player.getBusted()){
                winners.add(player);

                if (player.getHandValue() == dealer.getHandValue() && !player.getBusted()){
                    if (isDraw == false){
                        winners.add(dealer);
                    }
                    isDraw = true;
                }
            }
        }
        return winners;
    }
}

```

In the Black Jack, there are situations that indicate the winner.

1. When dealer is busted and player is not then the player win.
2. When dealer is not busted but player is, then dealer wins.
3. When dealer is not busted and so as player. And dealer and player have same value of the cards then its a draw.
4. When dealer is not busted and so as player. And dealer has bigger value of the cards than player, then dealer wins.
5. When dealer is not busted and so as player. And player has bigger value of the cards than dealer, then player wins.