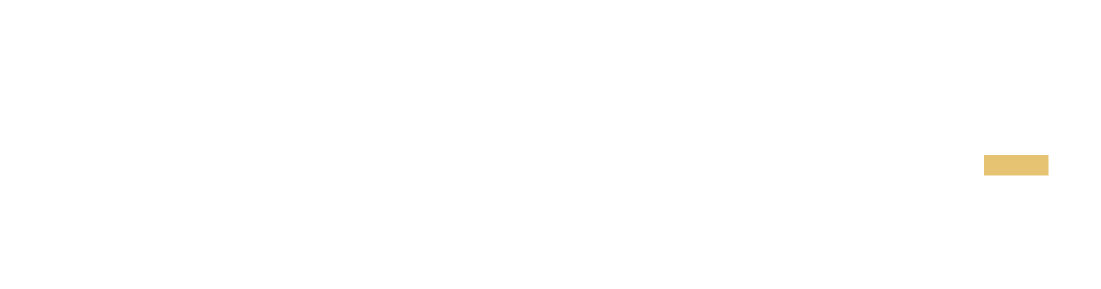
Evidence Gathering Document for SQA Level 8 Professional Developer Award.

This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

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

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

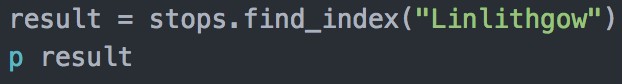


**Week 2**

| 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:** | |



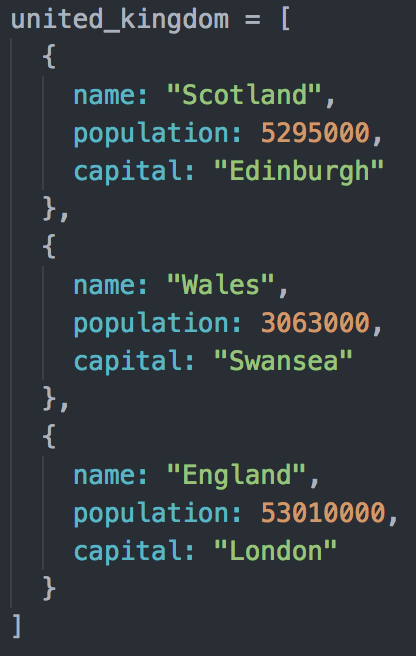
‘stops’ is an array of strings.



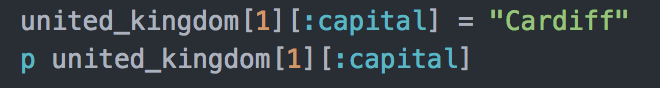
The find\_index method will run on ‘stops’ and return the index reference for the “Linlithgow” element in the array. The outcome of the method will be stored in ‘result’ which will then be printed to the screen. When this is run via terminal the result is as follows:



| 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:** | |

****

An array of hashes. Each element of the array is a hash to represent part of the United Kingdom. The hash is made up of keys and values to provide information about the particular part of the country.



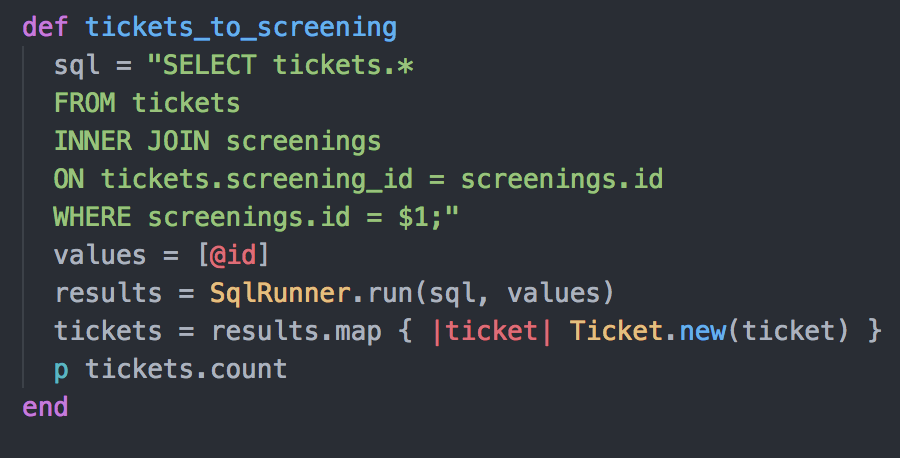
This function changes the Capital of Wales from Swansea to Cardiff. The function requires an index for the hash [1] which points it to the second element. Now the element has been located, the hash can be edited by calling on the capital symbol and reassigning it to ‘Cardiff’.

The result is printed to the screen to test the change has been successful. This displays in the terminal as follows:



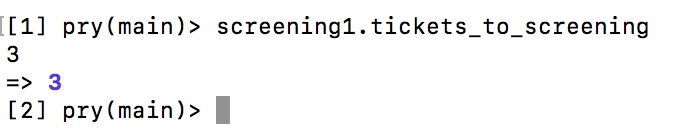
**Week 3**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.3 | Demonstrate searching data in a program. Take screenshots of:  \*Function that searches data  \*The result of the function running | |
|  |  | **Description:** | |



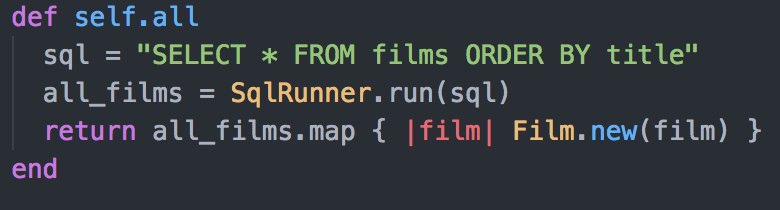
The function can be run on an instance of the Screenings class. With the information from the Screening the function checks the tickets table in the database to return the number of tickets sold for the screening.

The function uses sql to select the appropriate data from the ‘tickets’ table by joining with the ‘screenings’ table where ID for the given screening matches the Screening ID on in the ticket table.

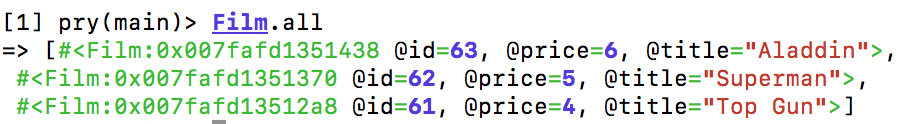


The function is run on Screening1 in terminal using Pry debugging. The function returns 3, meaning there are 3 tickets for the screening.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| I&T | I.T.4 | Demonstrate sorting data in a program. Take screenshots of:  \*Function that sorts data  \*The result of the function running | |
|  |  | **Description:** | |

****

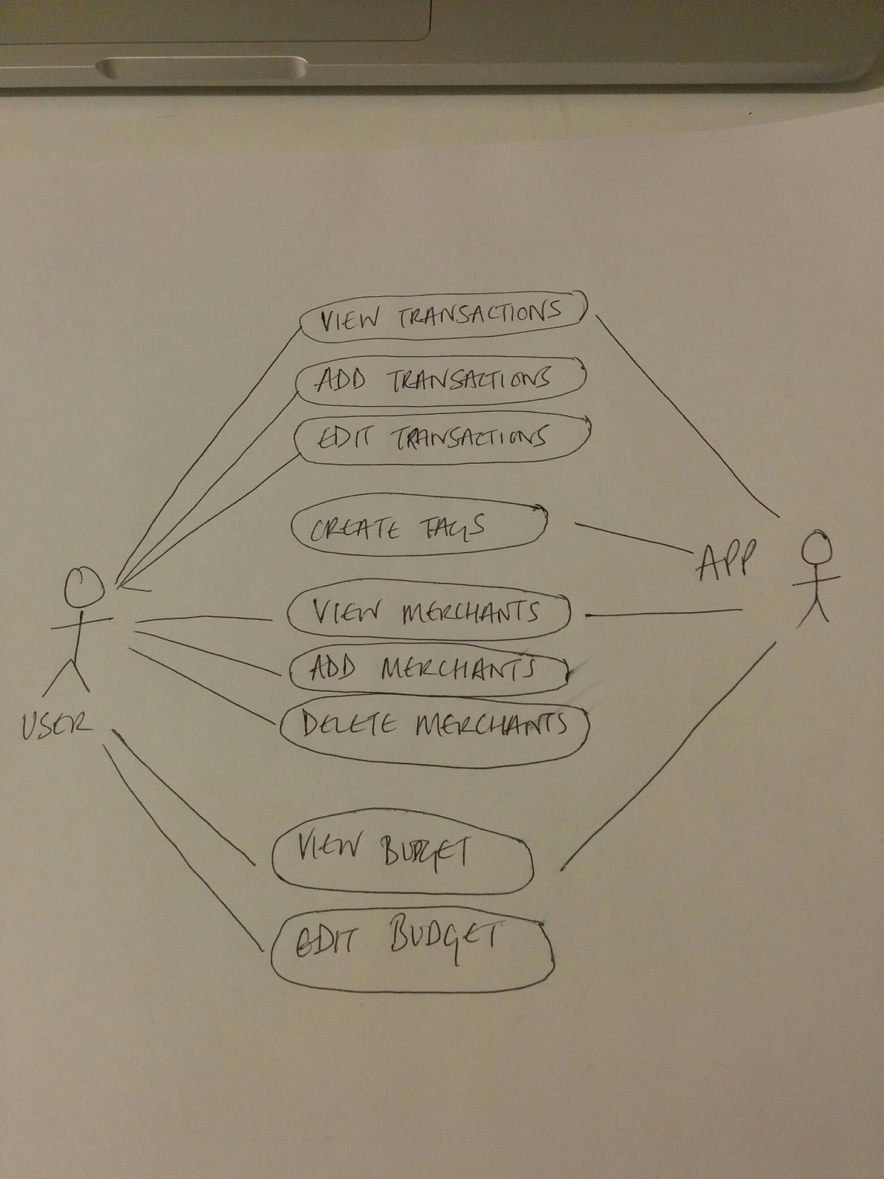
The function uses sql to return all instances of the film class in the table, and the results are ordered by film title.



The result of the function returns each film with all its associated information. The films are listed in alphabetical order by Title.

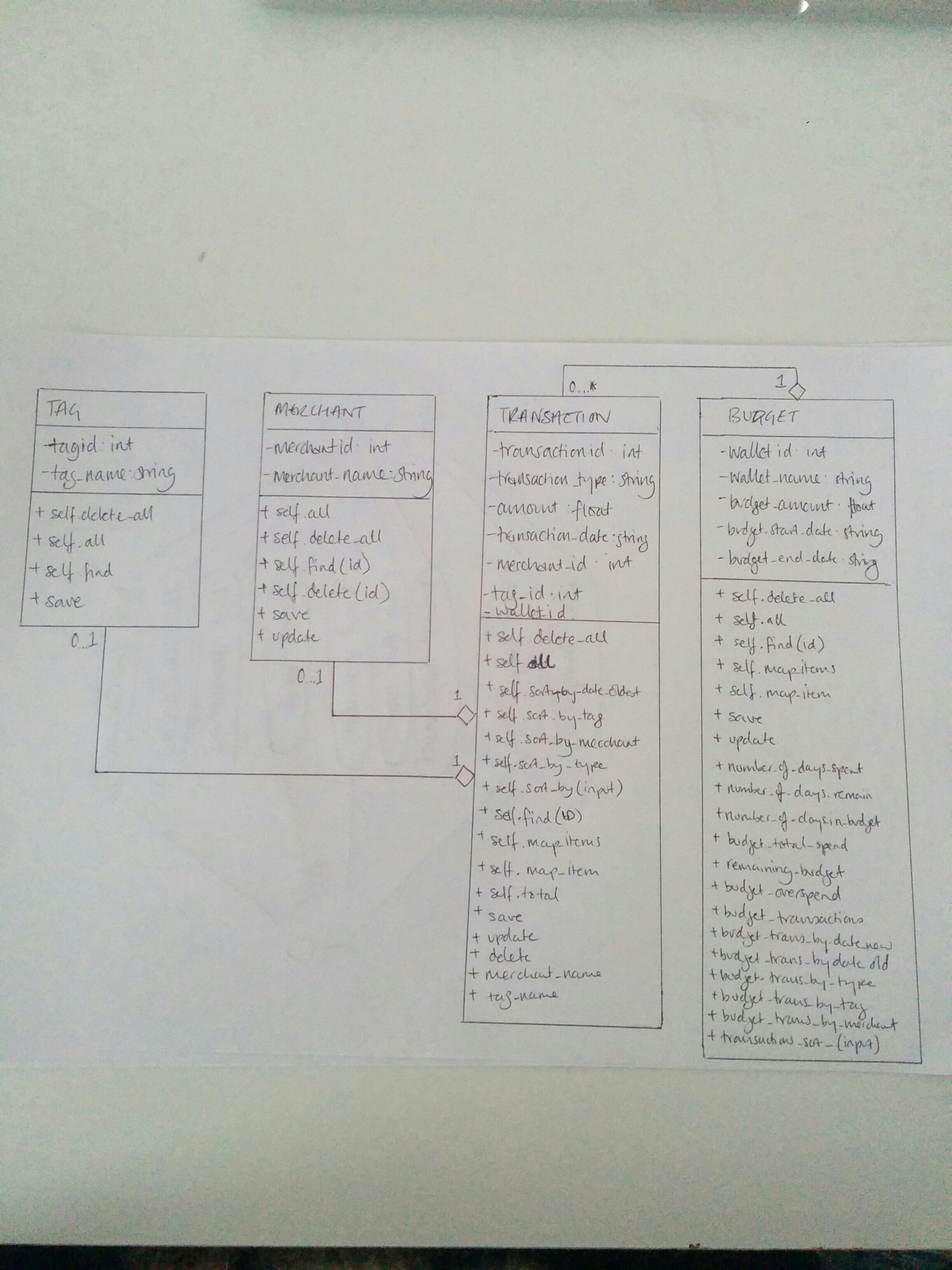
**Week 5 and 6**

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

****

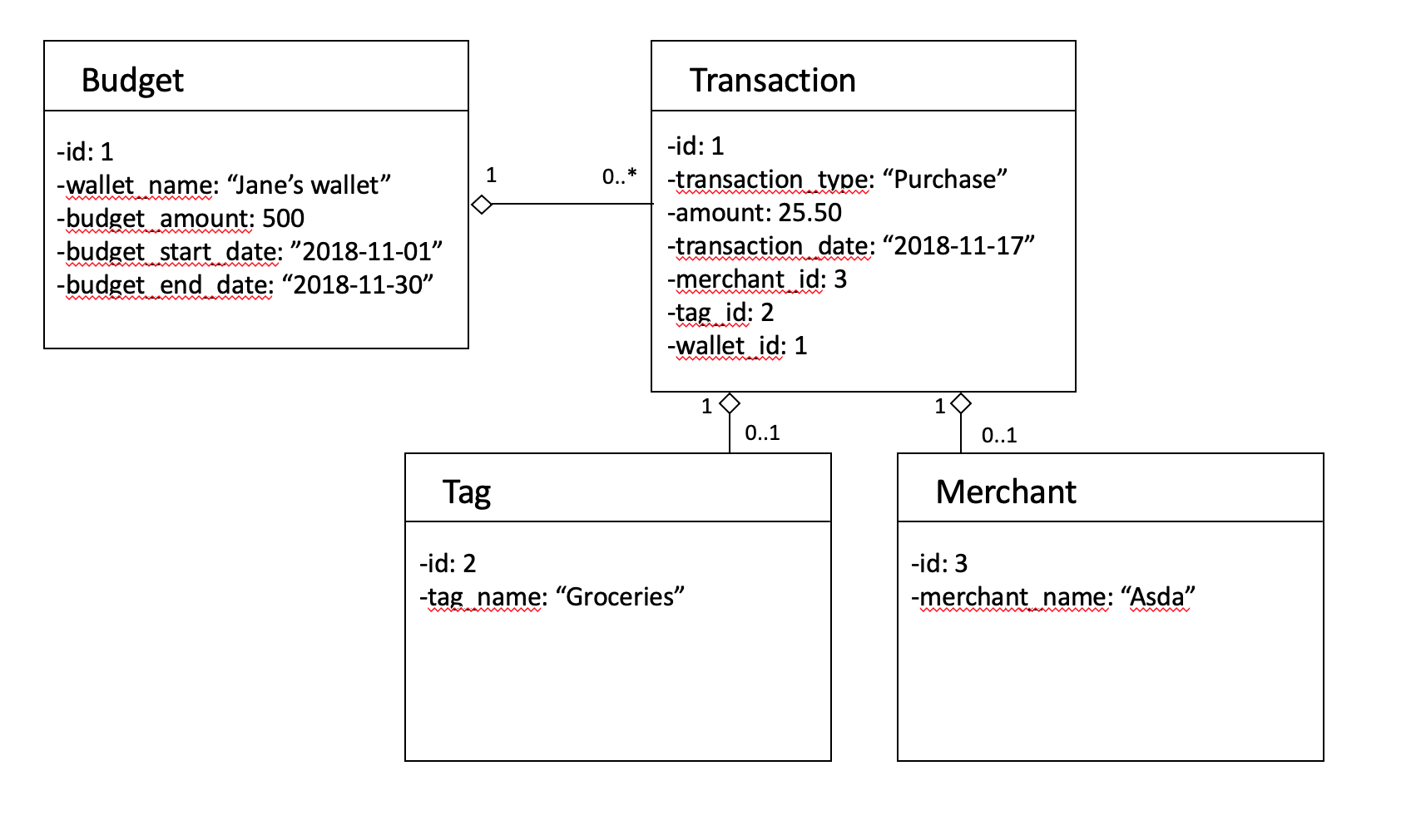
This Use Case diagram shows the options available to the user for a budget tracker app.

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



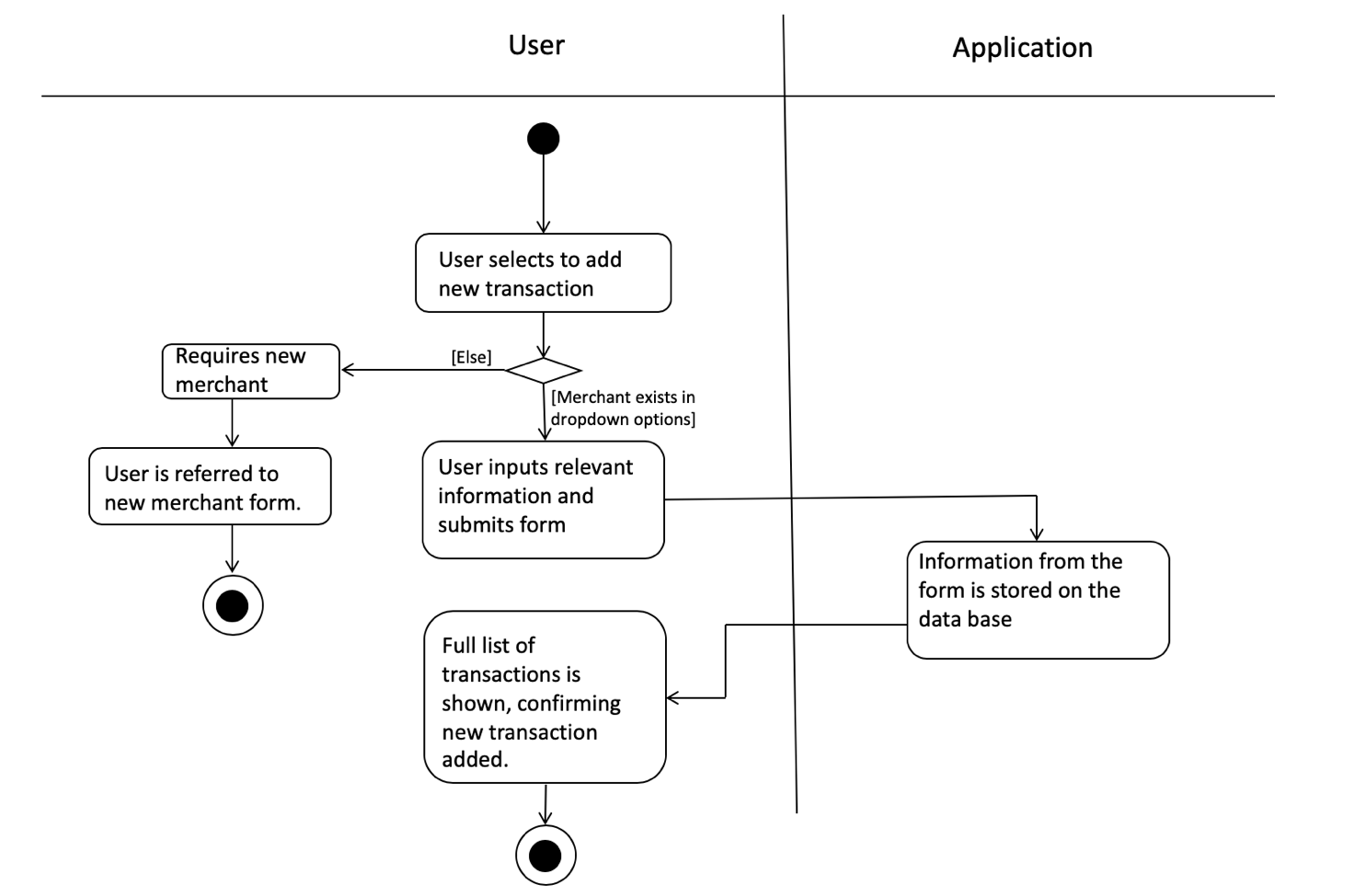
Class diagram outlining the tag, merchant, transaction and budget classes for a budget tracker app.

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

****

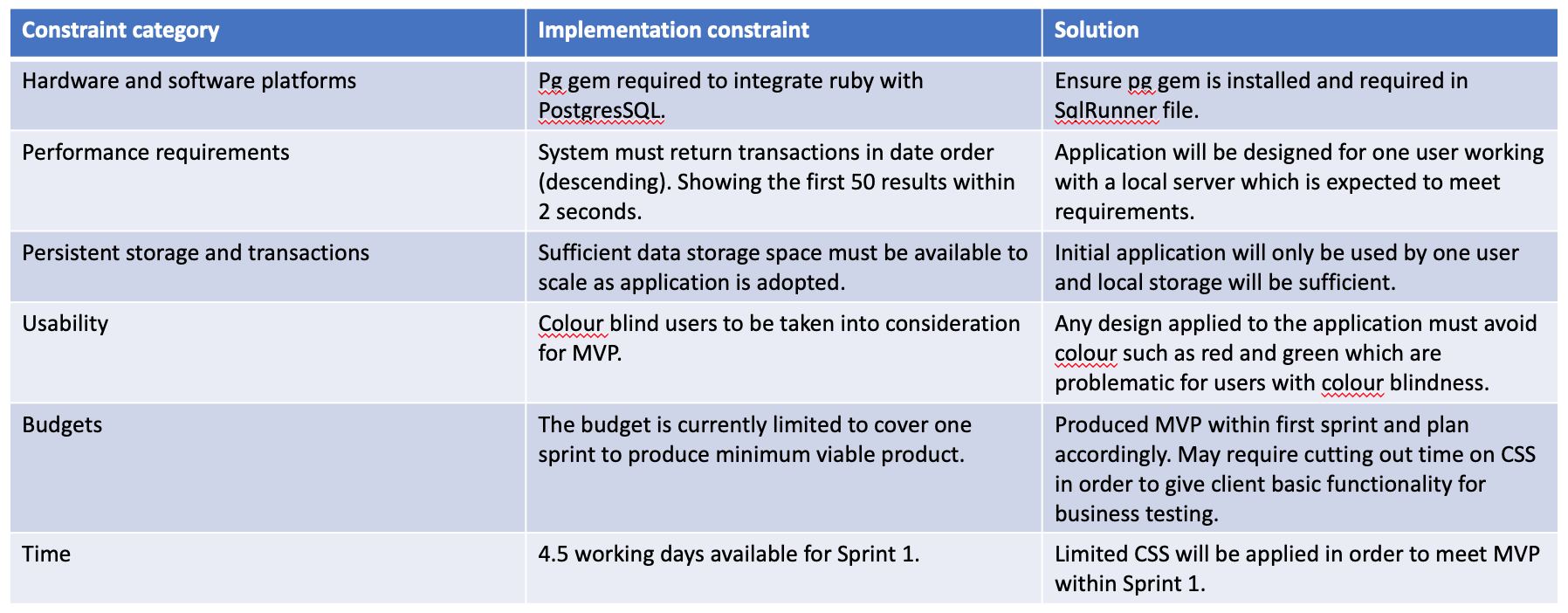
This object diagram shows examples of the budget, transaction, tag and merchant classes, including example attributes.

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

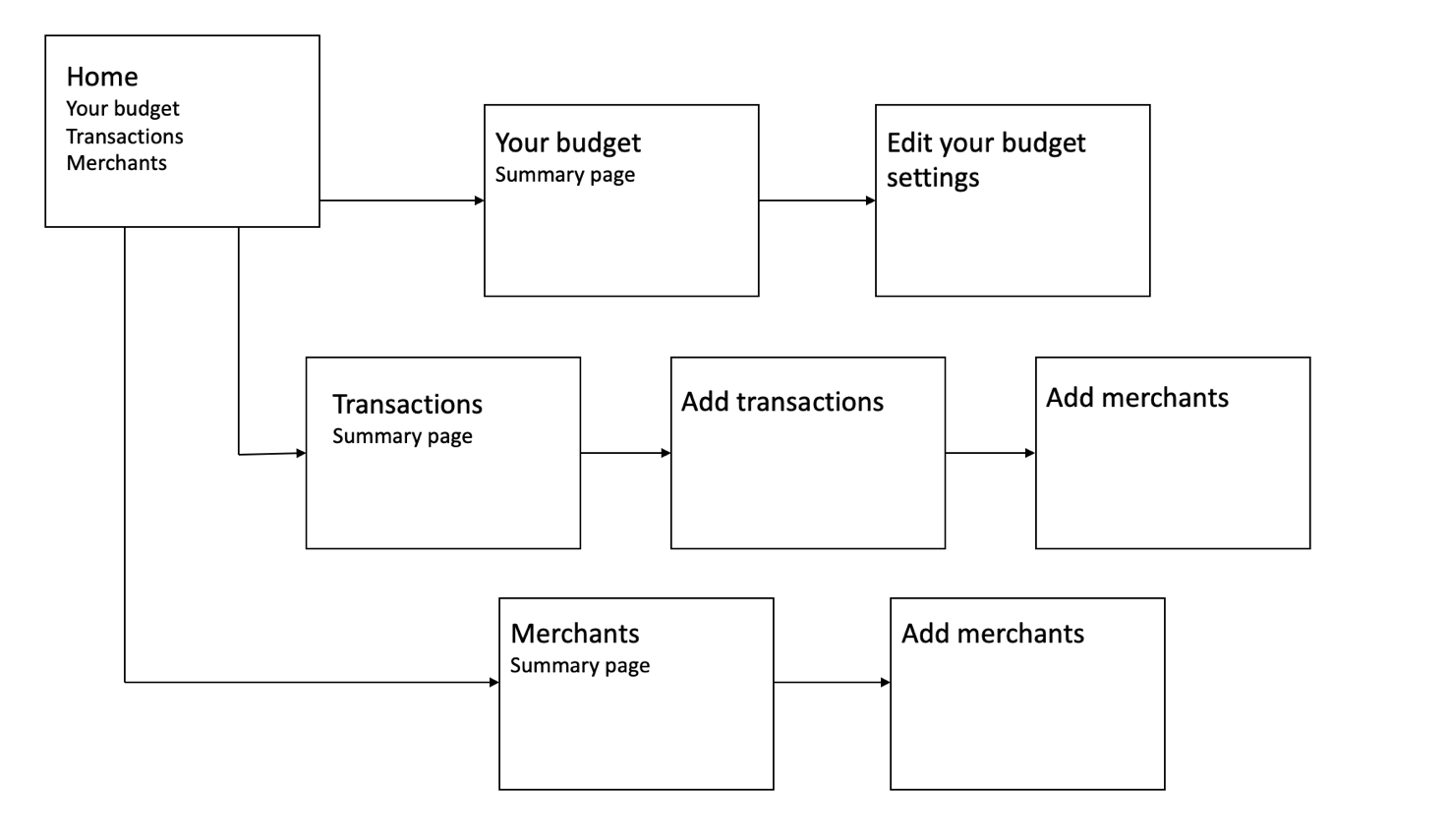


The Activity Diagram shows the process for adding a new transaction to the budget tracker. It includes a decision point when the user has to use an existing merchant or decide to add a new merchant to the system.

| 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:** | |

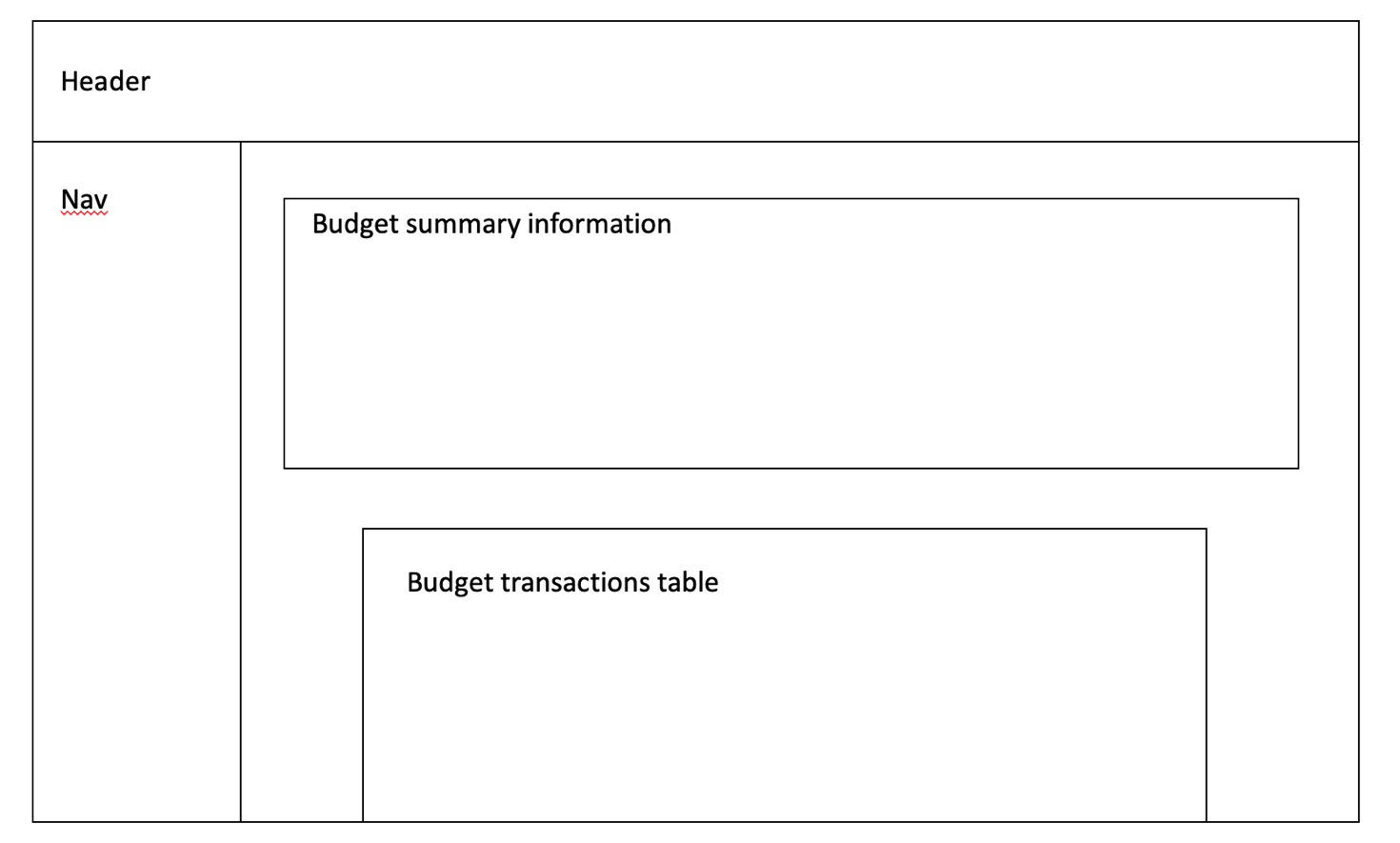
****

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.5 | User Site Map | |
|  |  | **Description:** | |

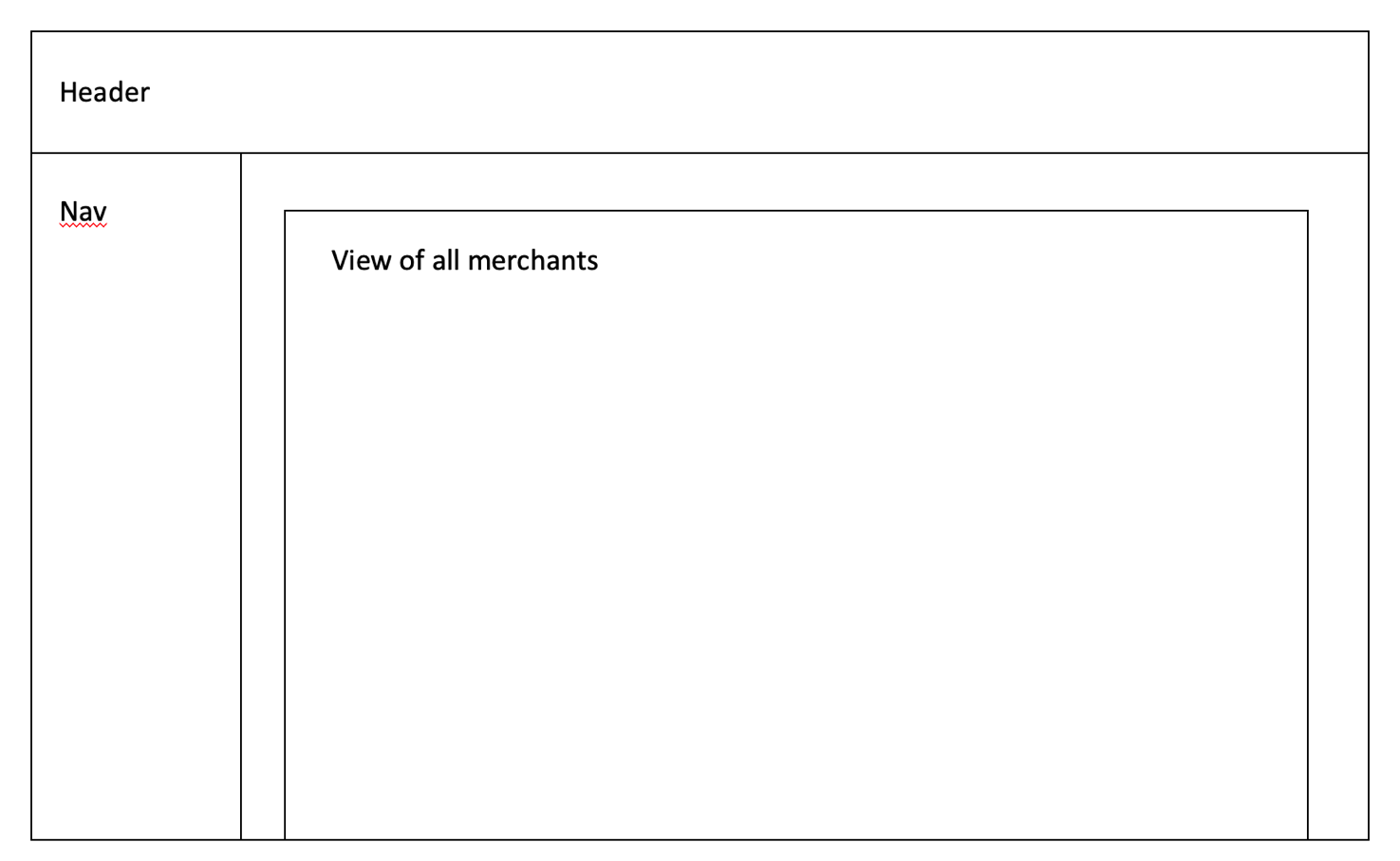


User site map showing the paths available to users of the budget tracker application.

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.6 | 2 Wireframe Diagrams | |
|  |  | **Description:** | |

****

Budget tracker application’s budget summary information page.

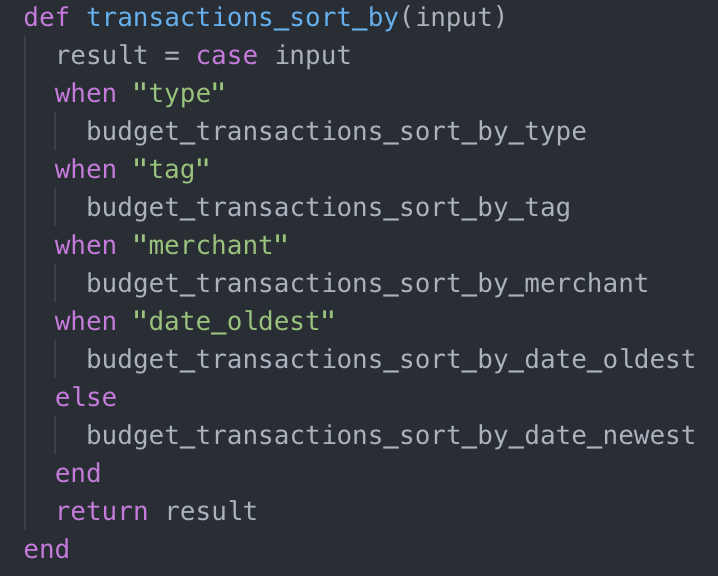
****

Budget tracker application’s merchants page which lists all merchants added to the application.

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

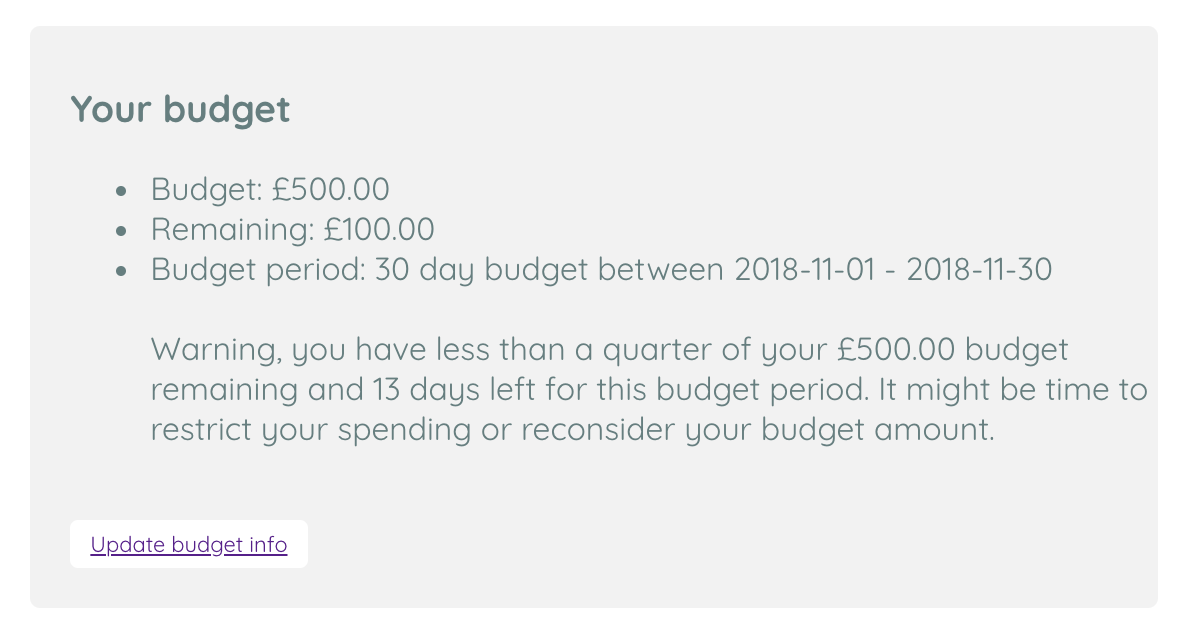
For the budget tracker application, I need to be able to sort a table of data by different fields.

I will need to take in how the user wants the data to be sorted and based on that information direct the systems to the appropriate sort method.

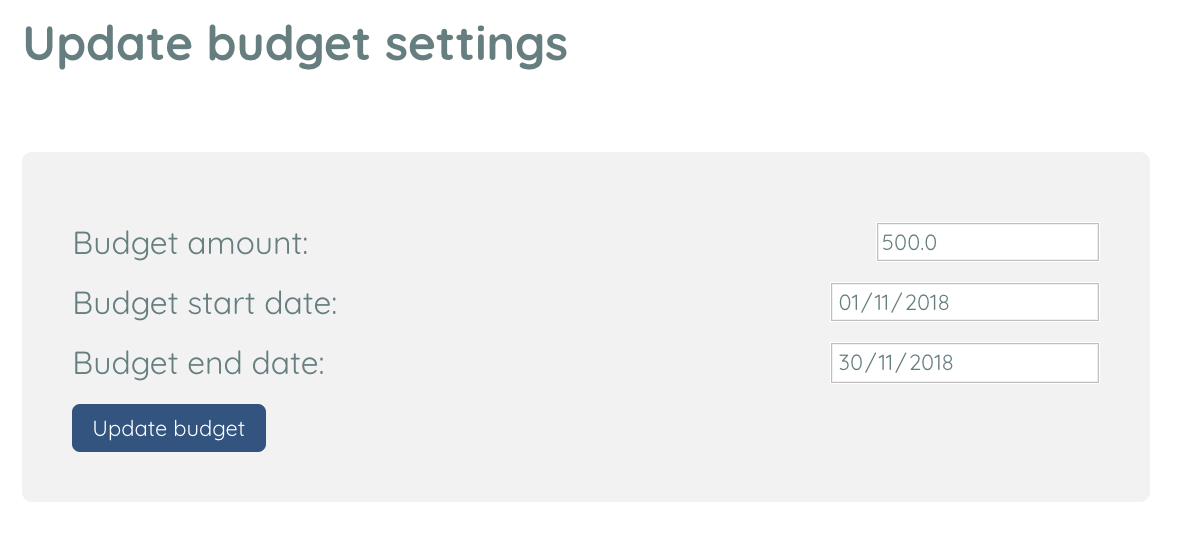


| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.13 | Show user input being processed according to design requirements. Take a screenshot of:  \* The user inputting something into your program  \* The user input being saved or used in some way | |
|  |  | **Description:** | |

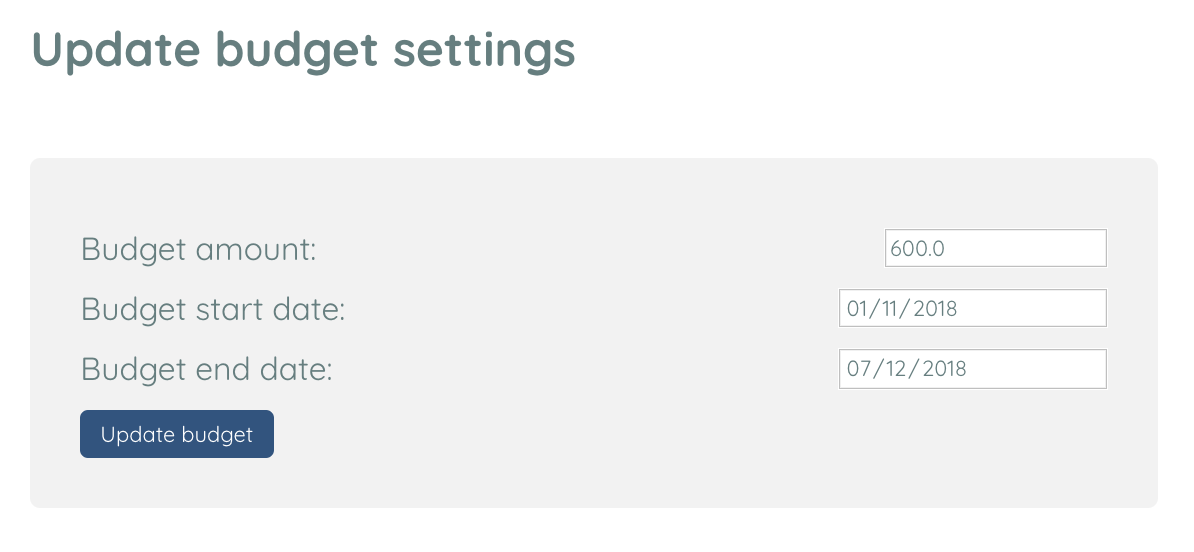
Original budget information available to view and be amended if required:



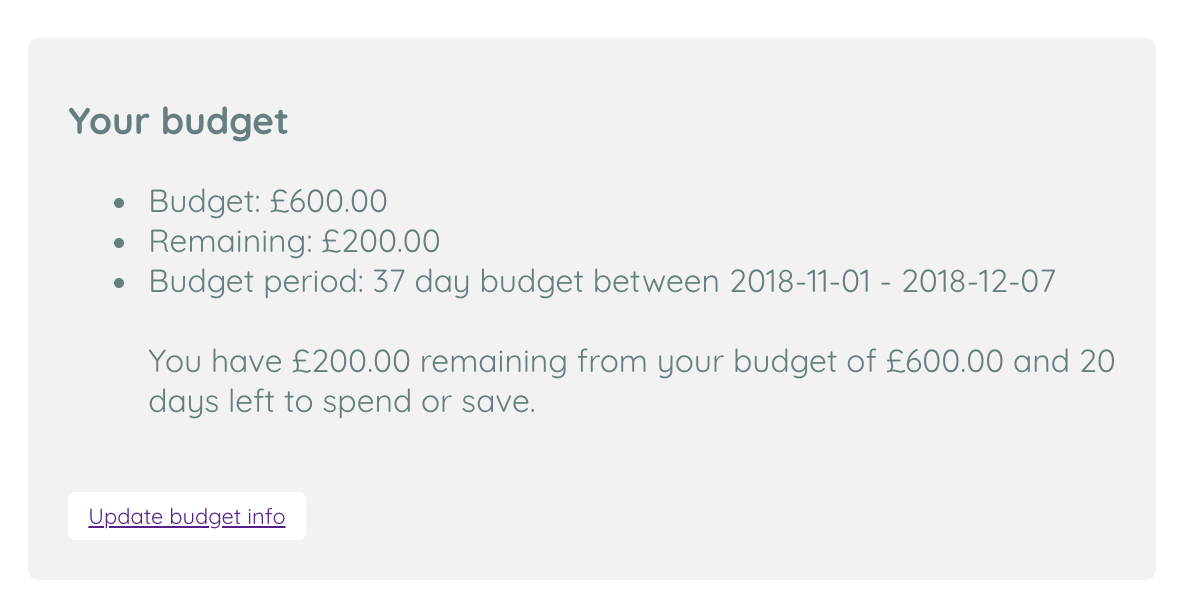
User selects ‘Update budget info’ and is taken to the following screen which still holds the original information:



Budget settings are amended as follows:

****

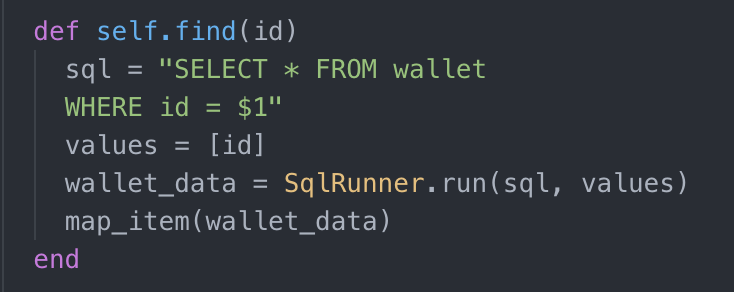
After saving these details the application updates the budget summary to display the following:



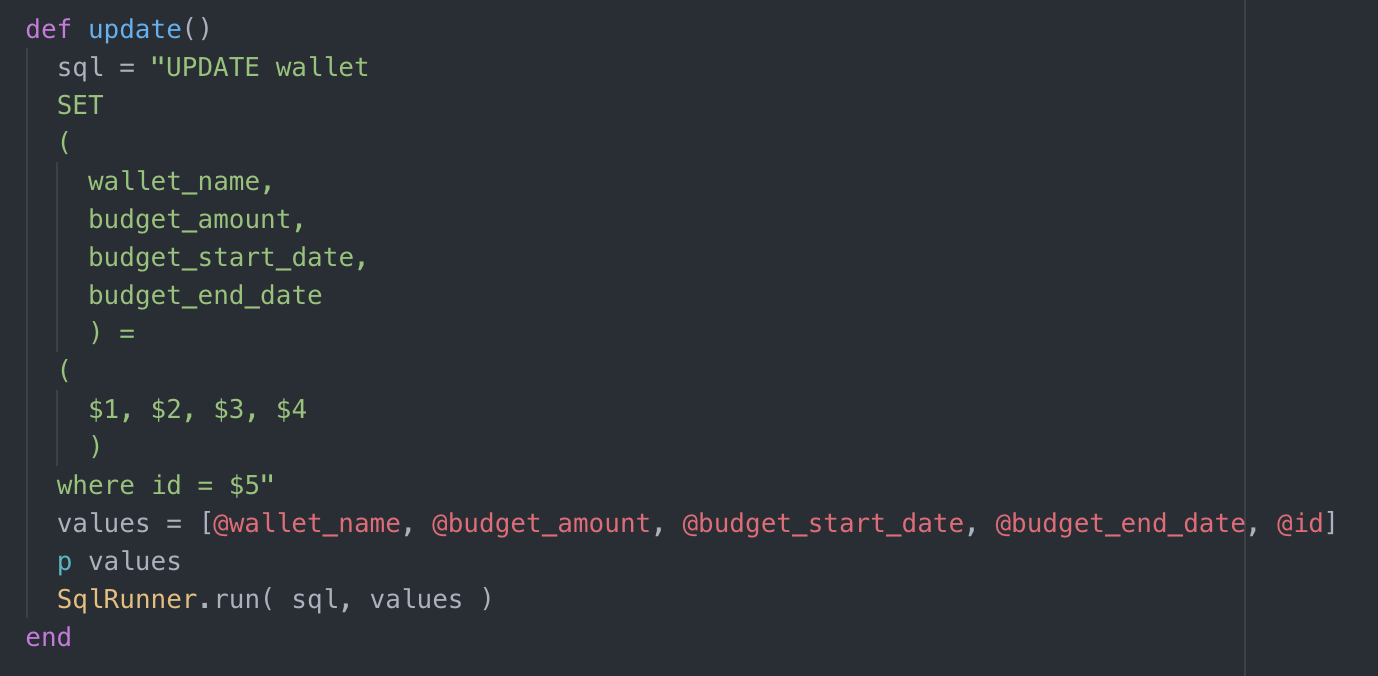
Behind the scenes the controller in Sinatra is calling on methods from the ‘wallet’ class in order to update the information:



The methods being used are the class method find with an id input:

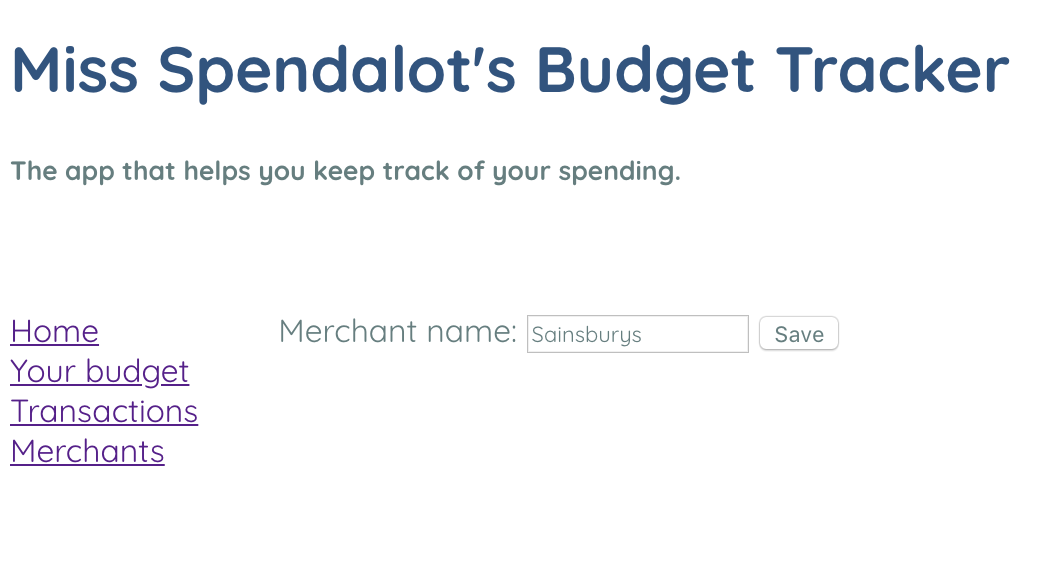


And then the update method:

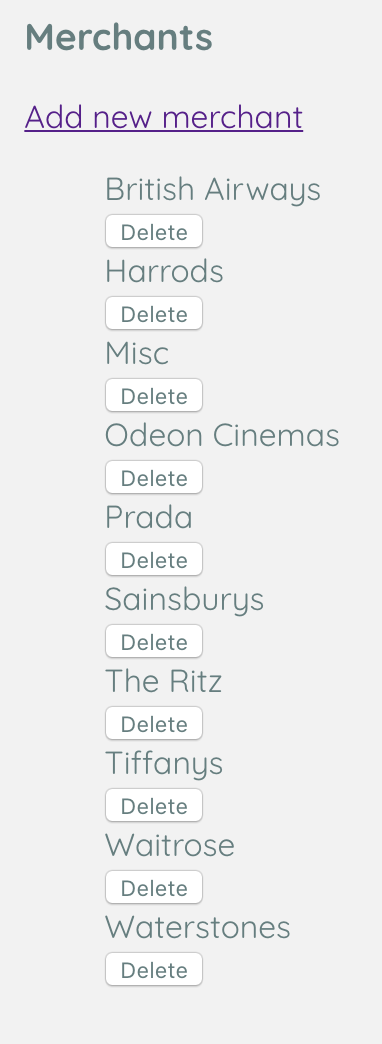


| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.14 | Show an interaction with data persistence. Take a screenshot of:  \* Data being inputted into your program  \* Confirmation of the data being saved | |
|  |  | **Description:** | |

A new merchant being added to the budget tracker:

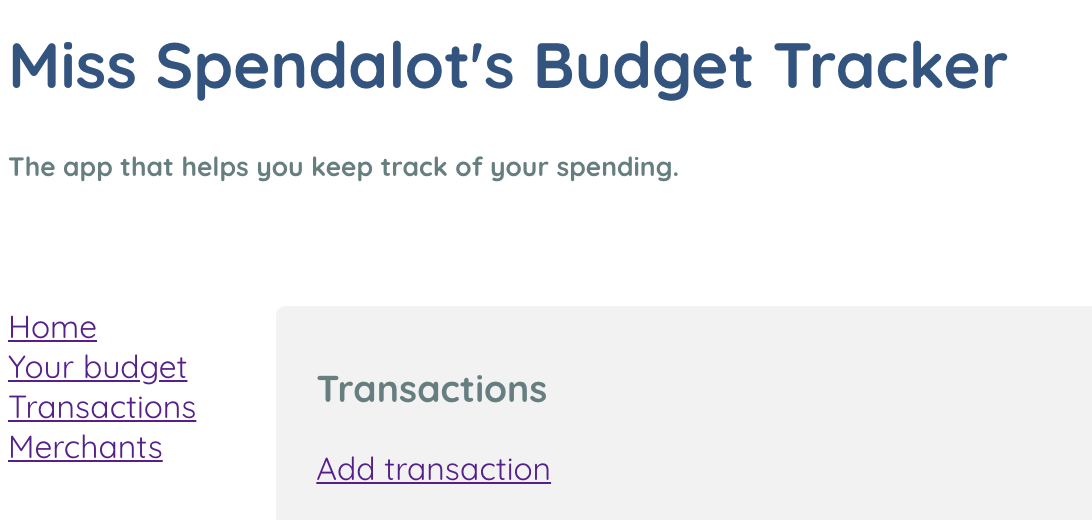
****

Confirmation this has been added is given when the new merchant ‘Sainsburys’ appears in the list of merchants:

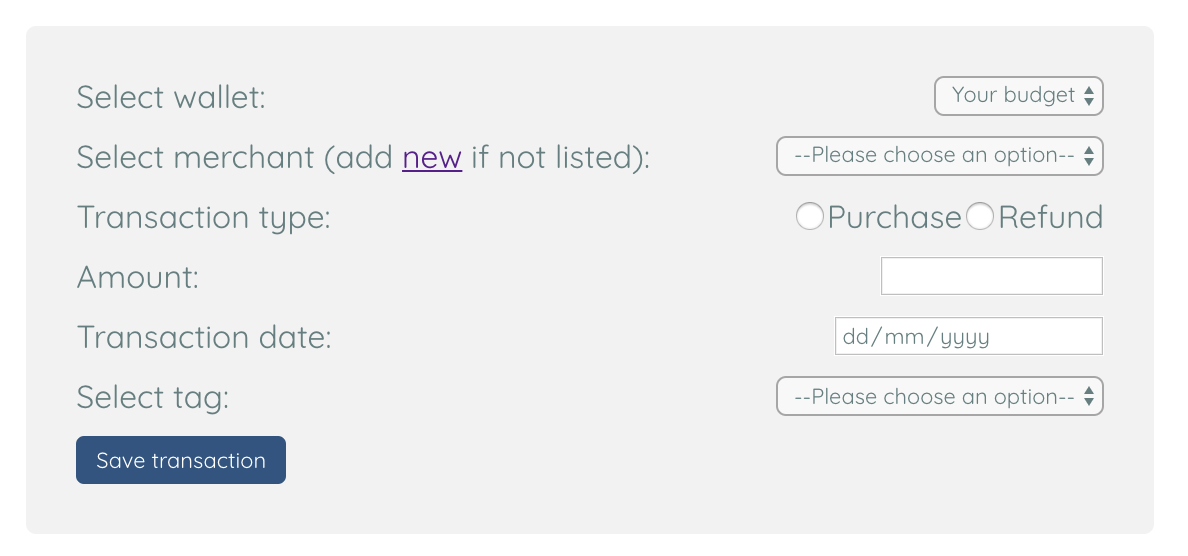


| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.15 | Show the correct output of results and feedback to user. Take a screenshot of:  \* The user requesting information or an action to be performed  \* The user request being processed correctly and demonstrated in the program | |
|  |  | **Description:** | |

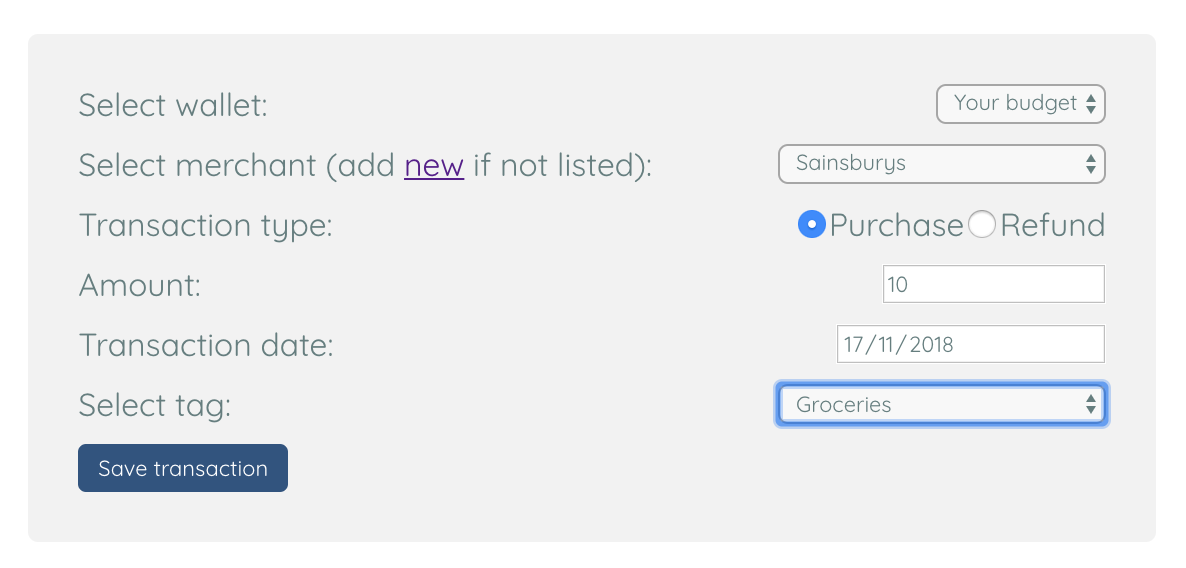
In the budget tracker app the user can request to add a new transaction:



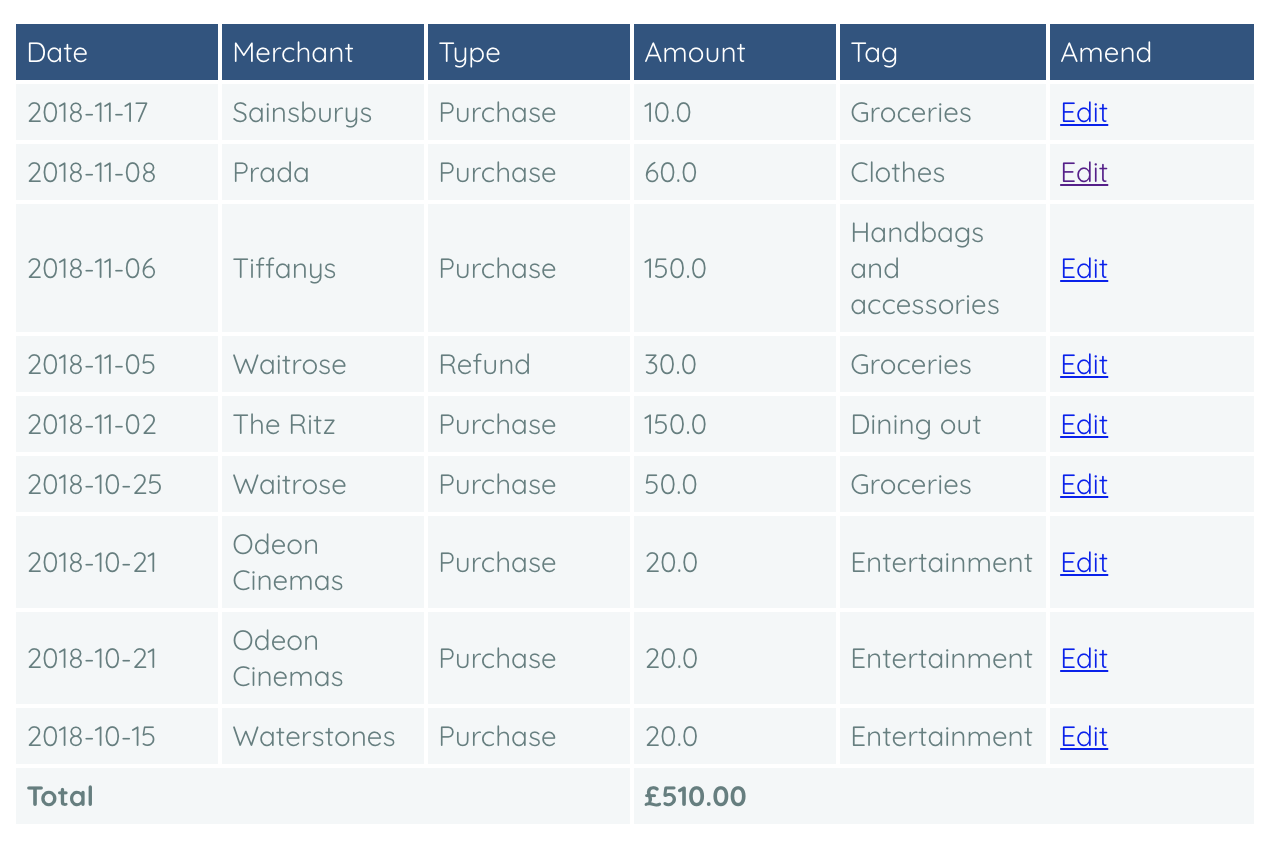
When they select ‘Add transaction’ they are taken to the following form:



This can be filled out as follows:

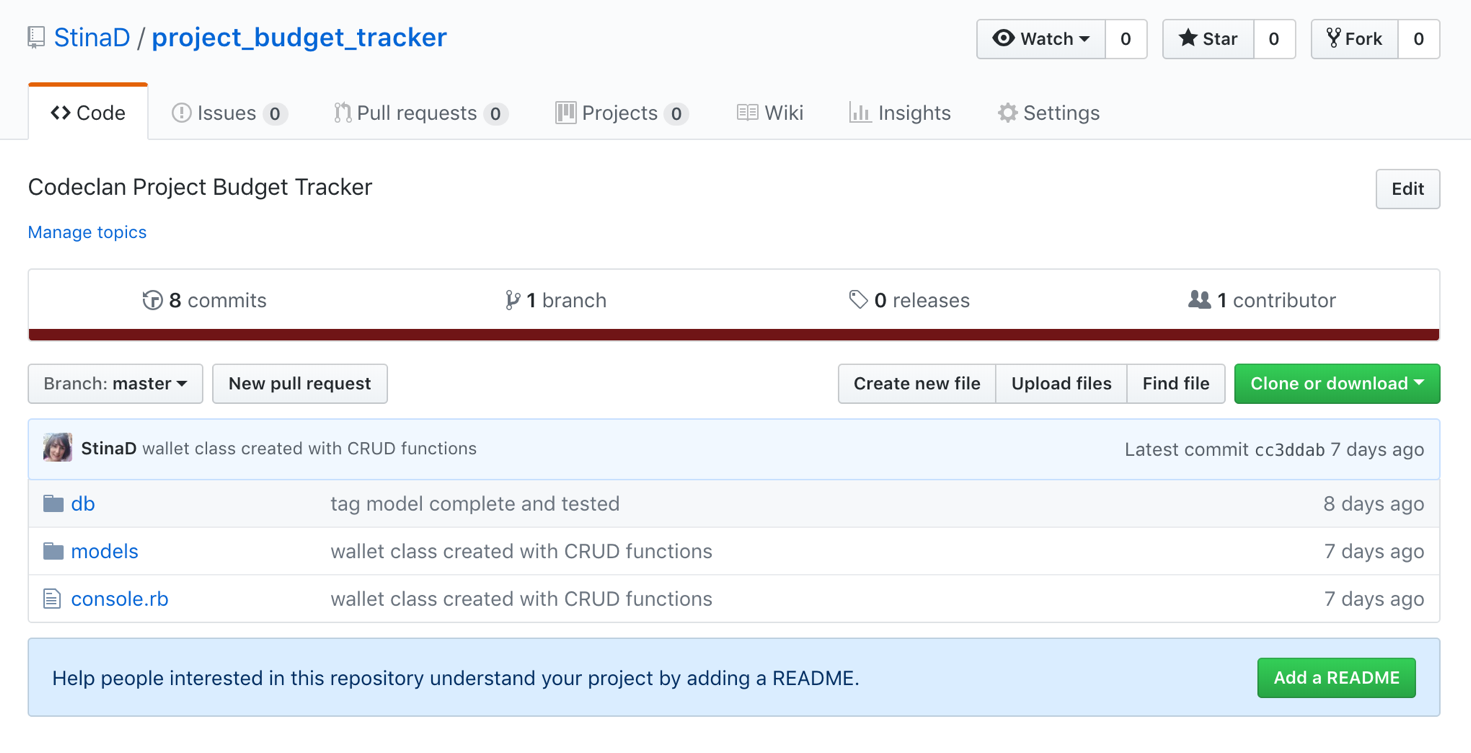
****

The new transaction is then visible in the list of transactions:

****

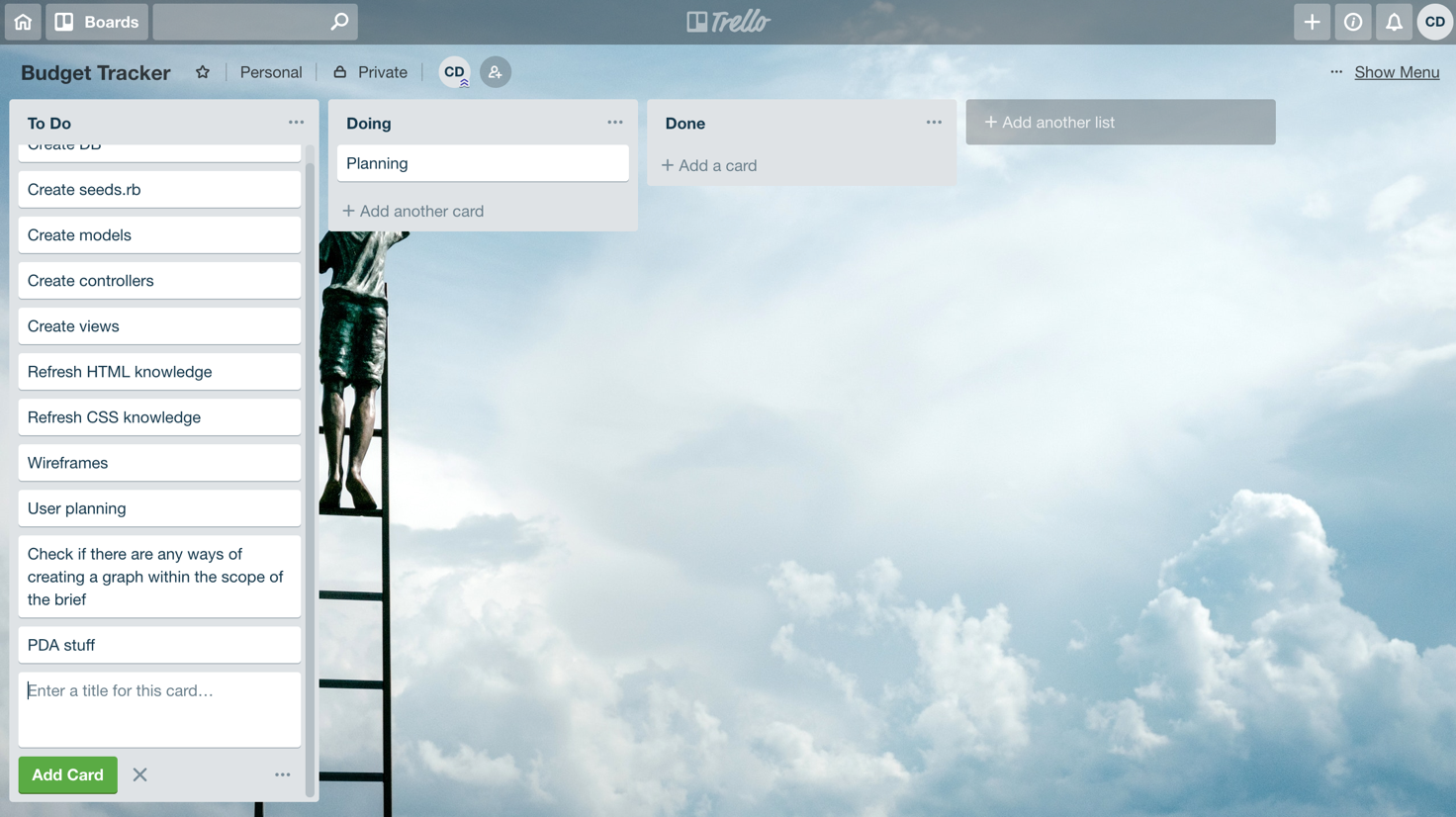
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.11 | Take a screenshot of one of your projects where you have worked alone and attach the Github link. | |
|  |  | **Description:** | |

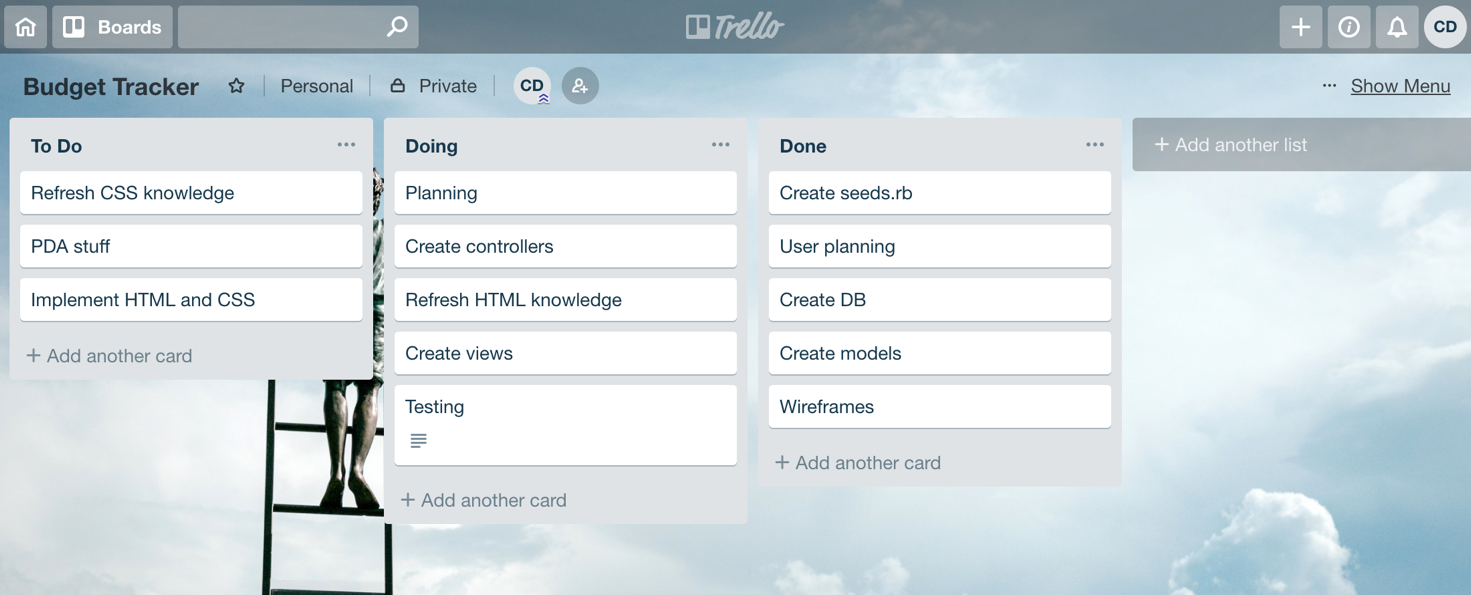
<https://github.com/StinaD/project_budget_tracker>



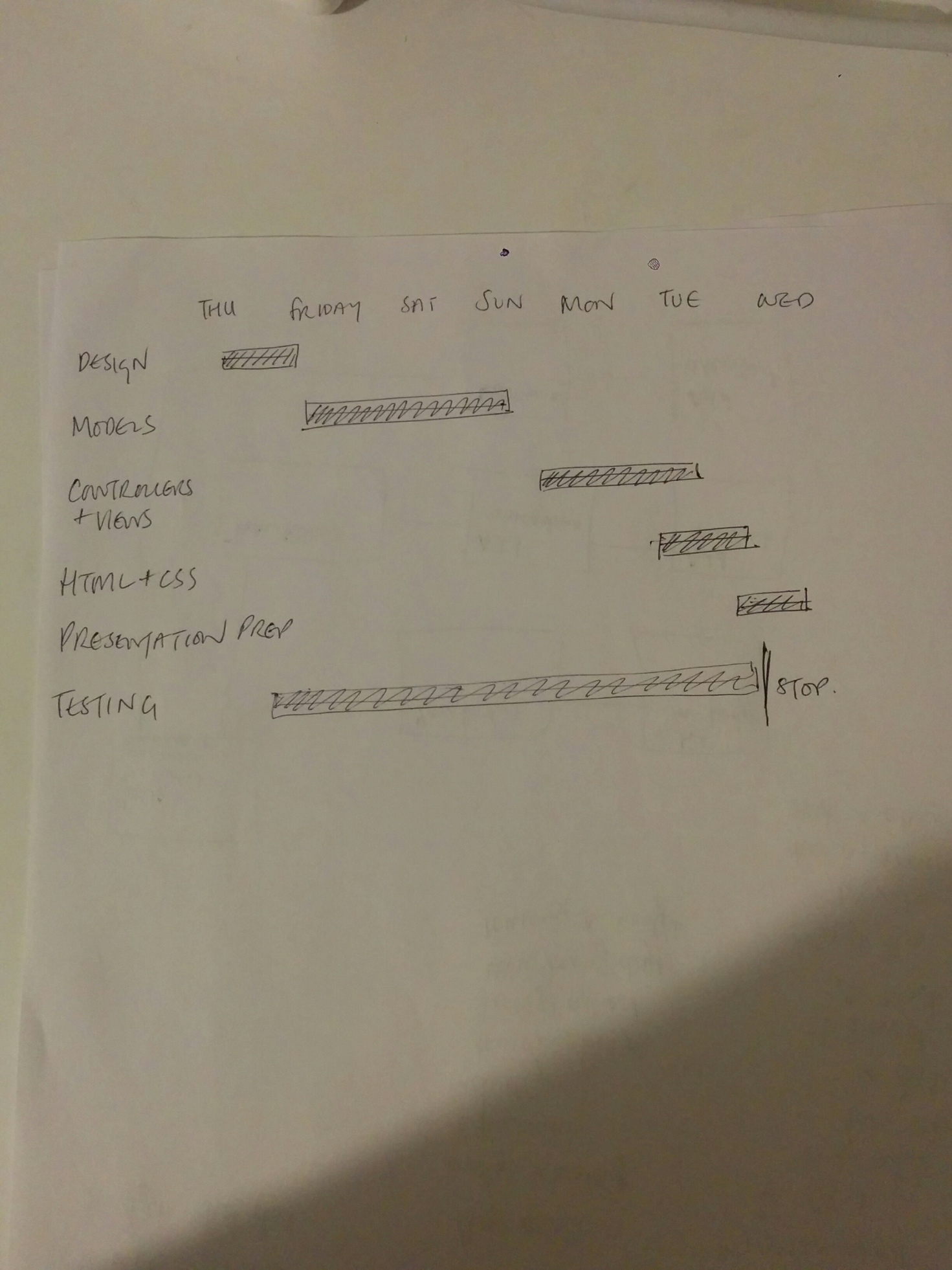
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.12 | Take screenshots or photos of your planning and the different stages of development to show changes. | |
|  |  | **Description:** | |

Initial Kanban board used for activities to be done using Trello board:

****



This was used concurrently with a gantt chart indicating when things will be done during the project time period:

****

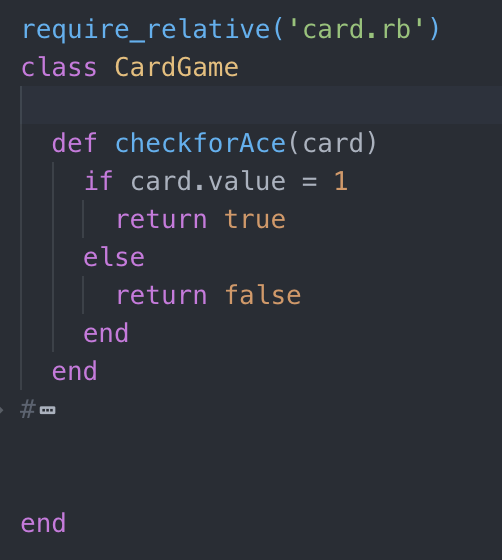
**Week 7**

| 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**

**Description here**

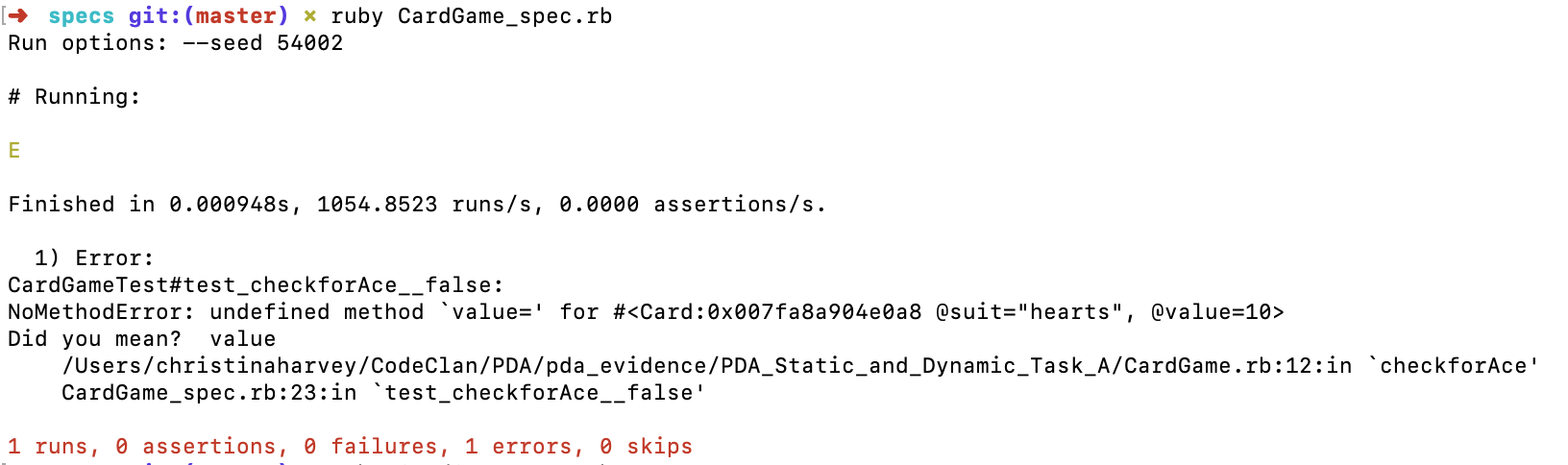
| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.18 | Demonstrate testing in your program. Take screenshots of:  \* Example of test code  \* The test code failing to pass  \* Example of the test code once errors have been corrected  \* The test code passing | |
|  |  | **Description:**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

1. Original function:

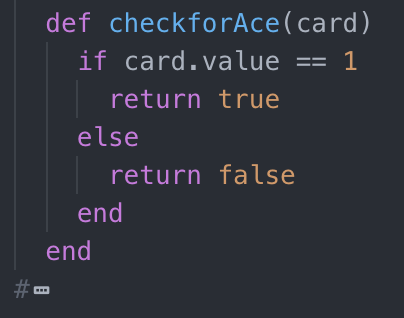
Test:

****

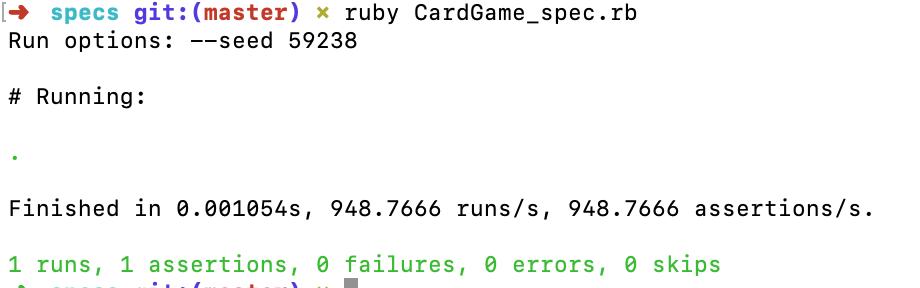
Test fail:

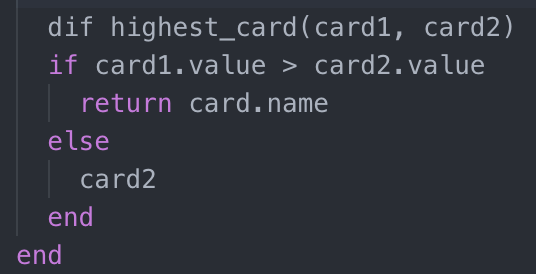


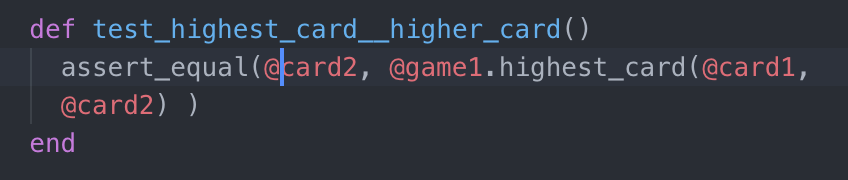
To fix the error the function needs to use ‘==‘ rather than just ‘=‘:



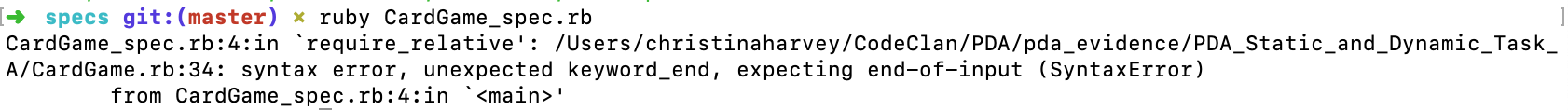
Test then passes:



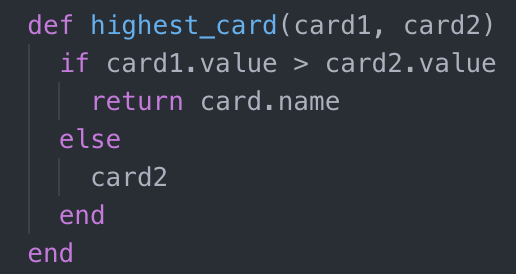
2. Original function:

Test:

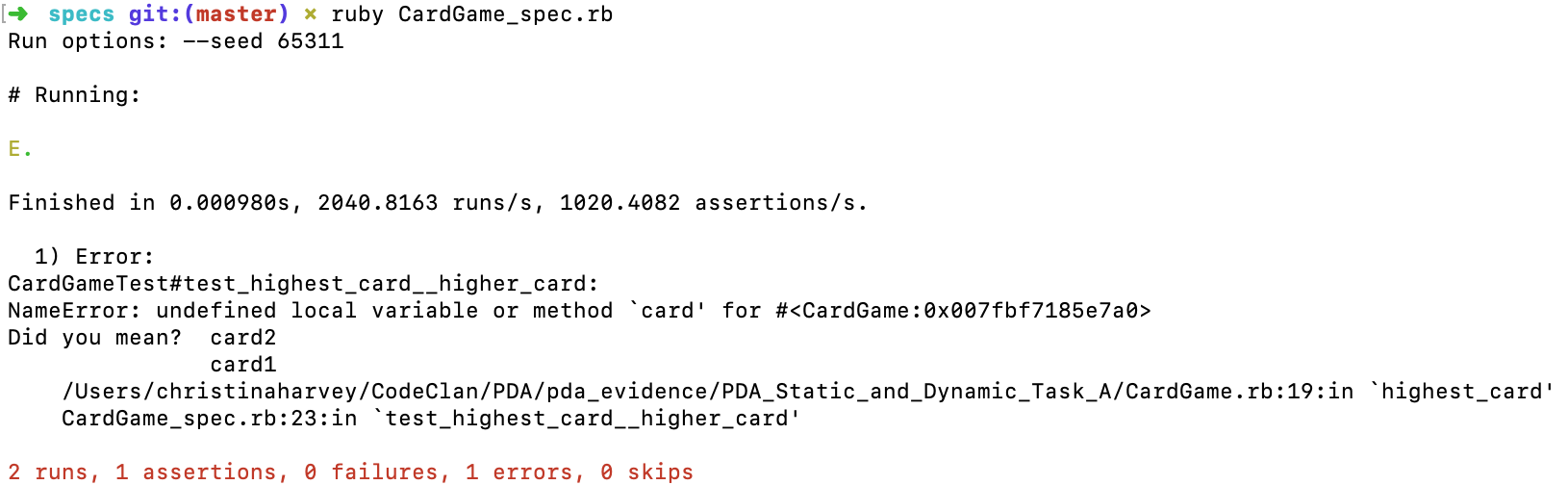
First error:



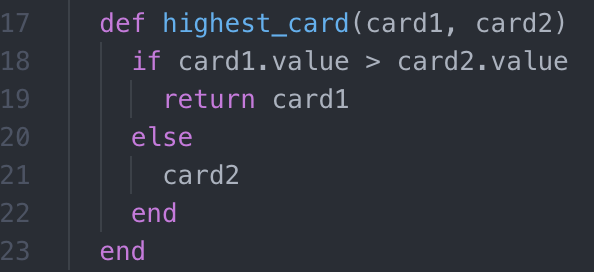
Function requires ‘def’ at the beginning rather than ‘dif’:



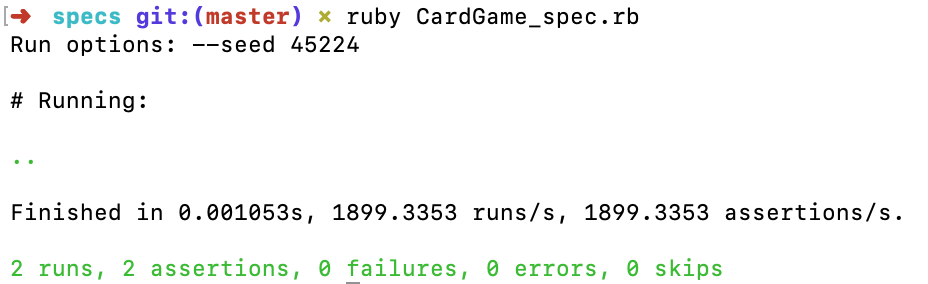
The test is then re-run which gives this error message:



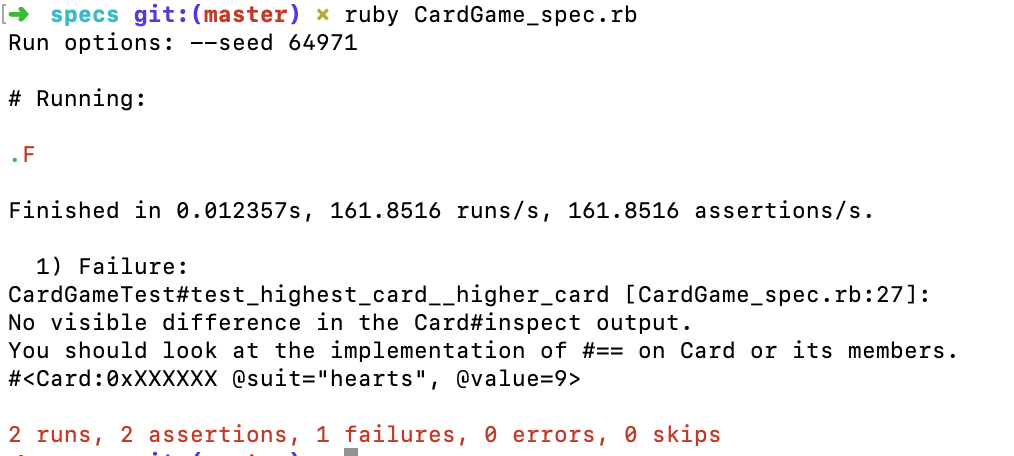
‘card.name’ needs to be changed to card1 on line 19:



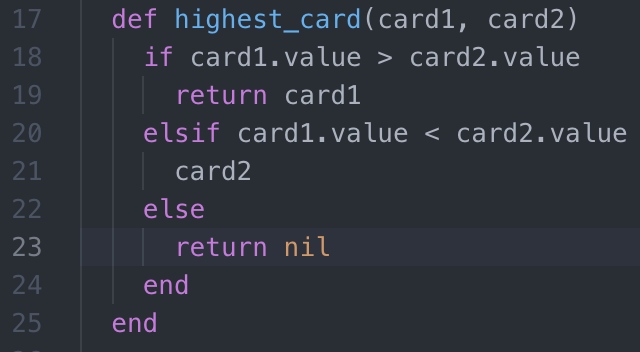
The test is then rerun and passes:



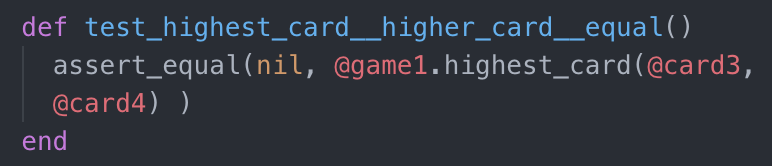
However the test then fails if you put two cards in with the same value.



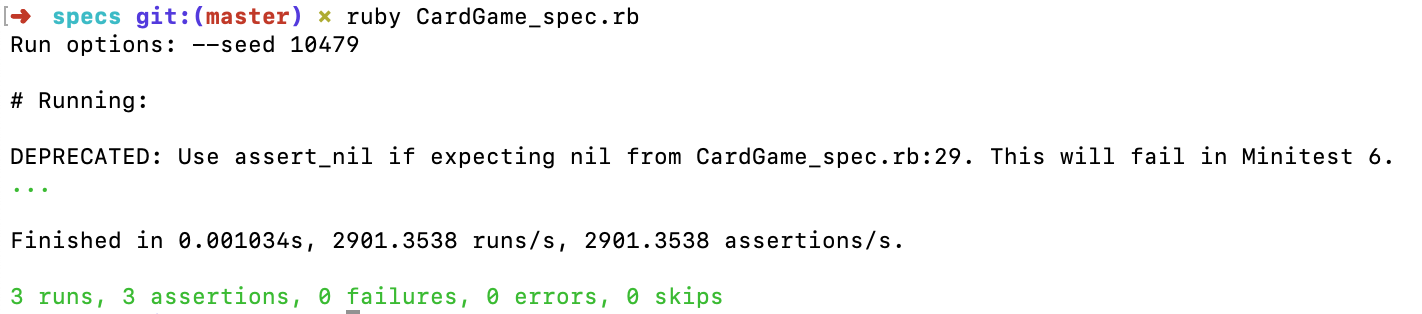
I amended the function to an elsif function:



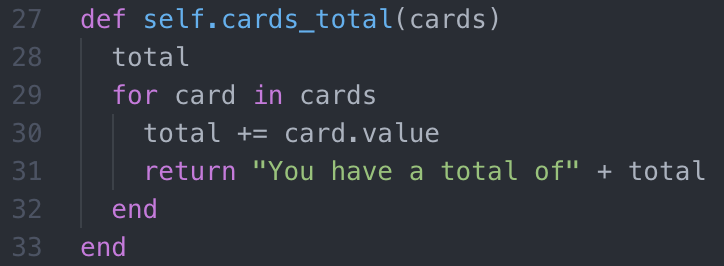
An additional test is needed for this alternative outcome:



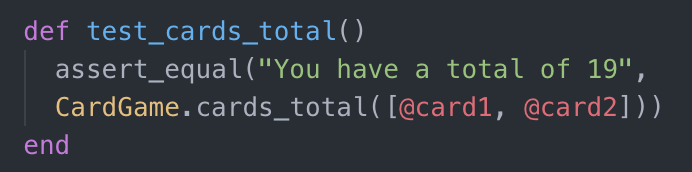
The tests then pass:



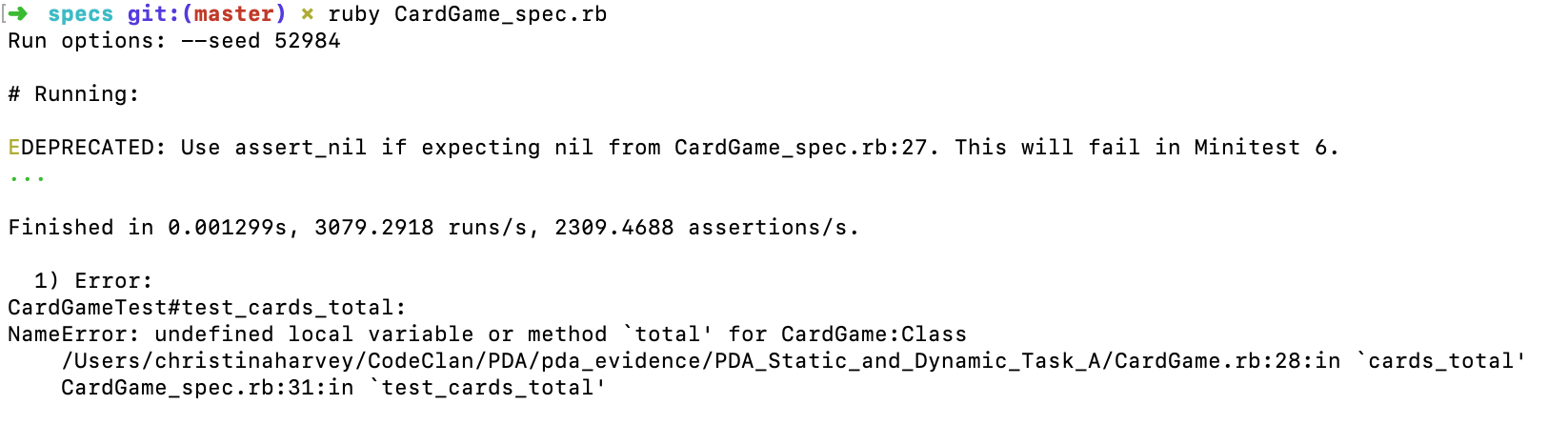
3. Original function:



Test:



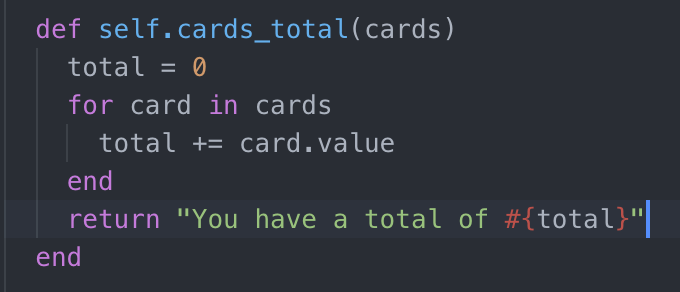
The test fails:



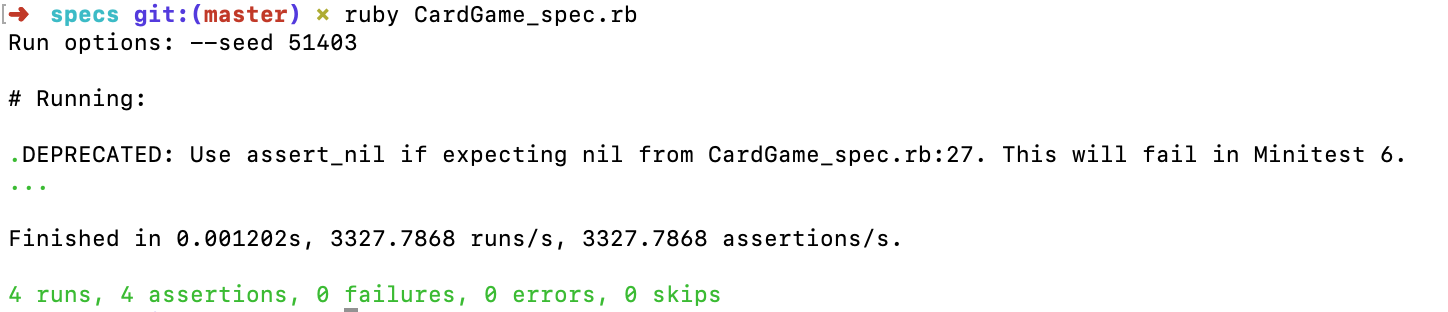
Reasons for failure:

* An end was missing to close the class.
* Total is undefined.
* Returning the string needs to follow the for loop (ie sit outside the for loop).
* String interpolation needs to be used.

Revised function:



Test pass result:



**Week 9**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.1 | Take a screenshot of the contributor’s page on Github from your group project to show the team you worked with. | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

| 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**

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

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

**Week 12**

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

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

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

**Paste Screenshot here**

**Description here**

| 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**

**Description here**

| Unit | Ref | Evidence |  |
| --- | --- | --- | --- |
| P | P.9 | Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms. | |
|  |  | **Description:** | |

**Paste Screenshot here**

**Description here**