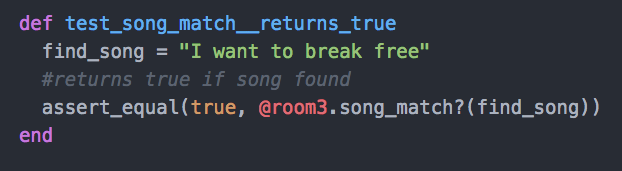
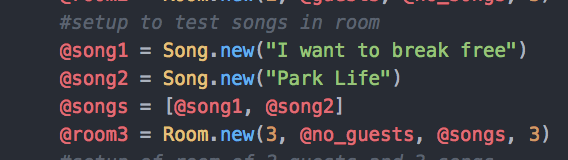
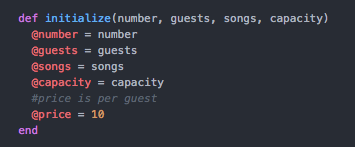
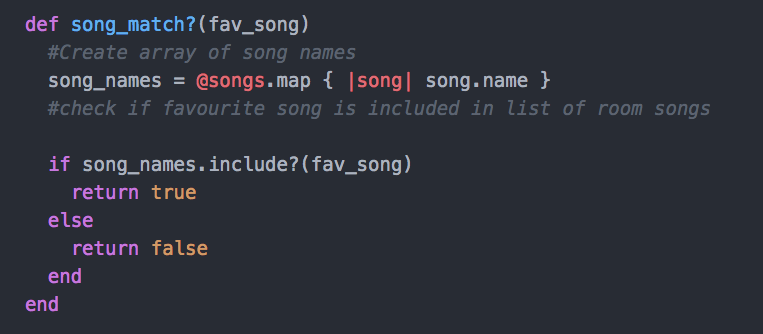
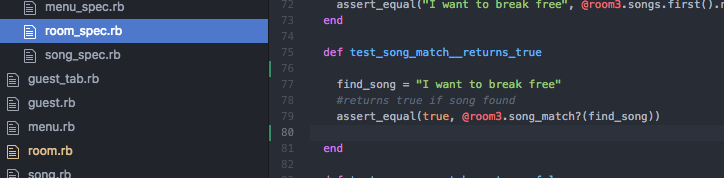
**Week 2 – I.T 5 – Use of an array in a program**

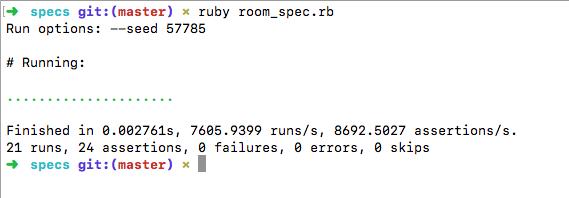






Result of function running

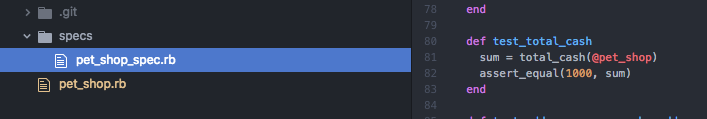


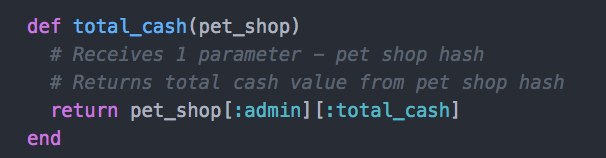


Week 2 – I.T 6 – Use of a hash in a program

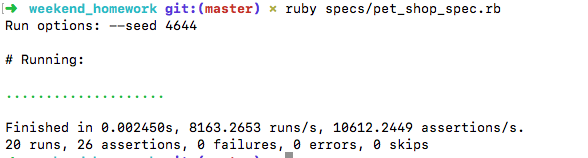
“admin” hash contained within “pet\_shop” hash



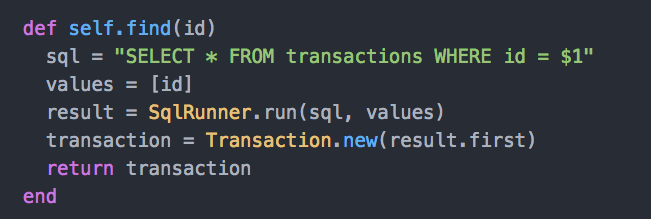




Result of function running

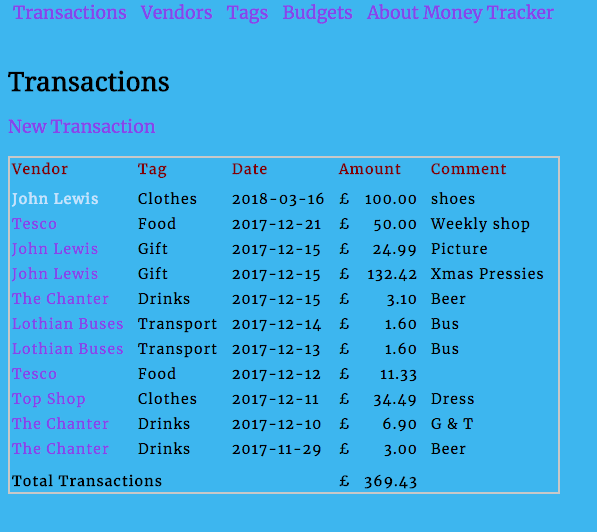


Week 3 – I.T.3 – Demonstrate searching data in a program

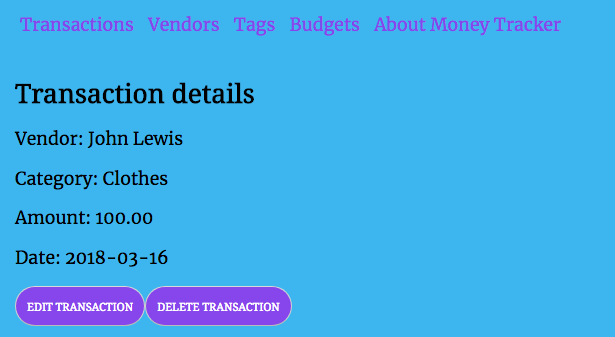


Result of search function running

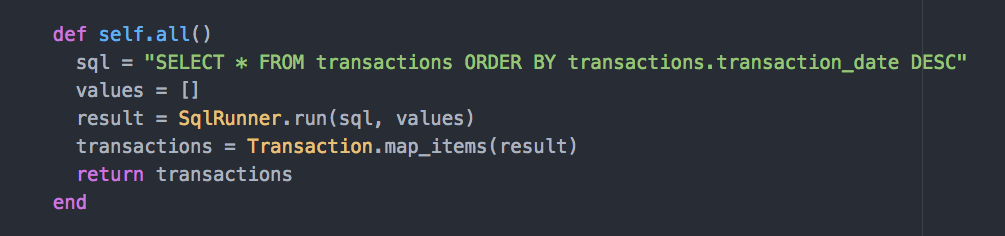
Screenshot of all transactions



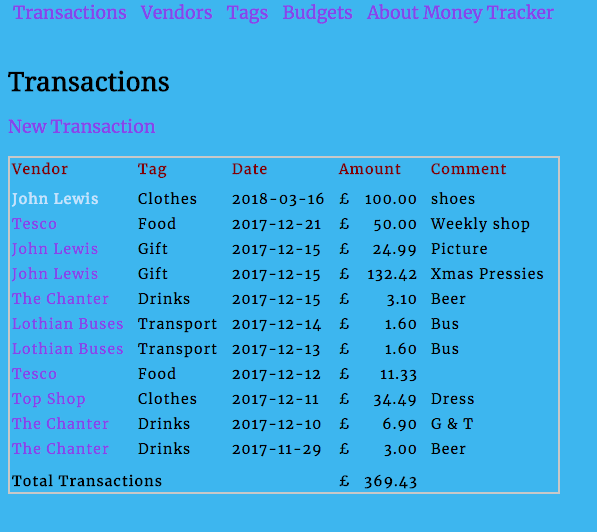
User selects to view 1 transaction and the above search function is then run to produce the result of transaction searched/ found and displayed



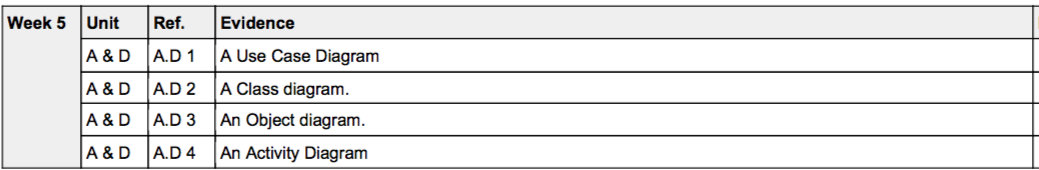
**Week 3 – I.T.4 – Demonstrate sorting data in a program**



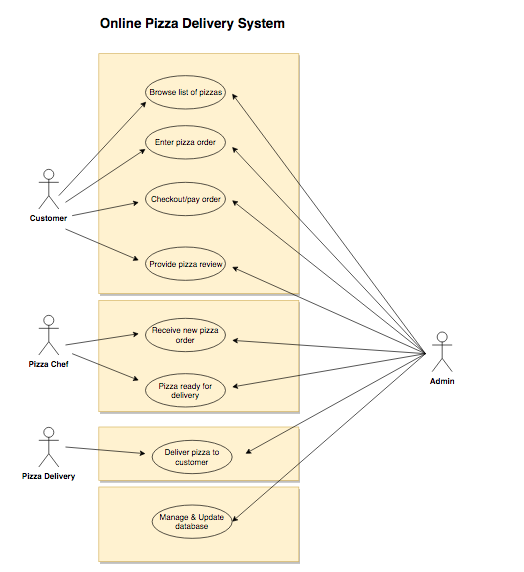
Transactions are retrieved and sorted in descending order to be displayed on the screen in descending order



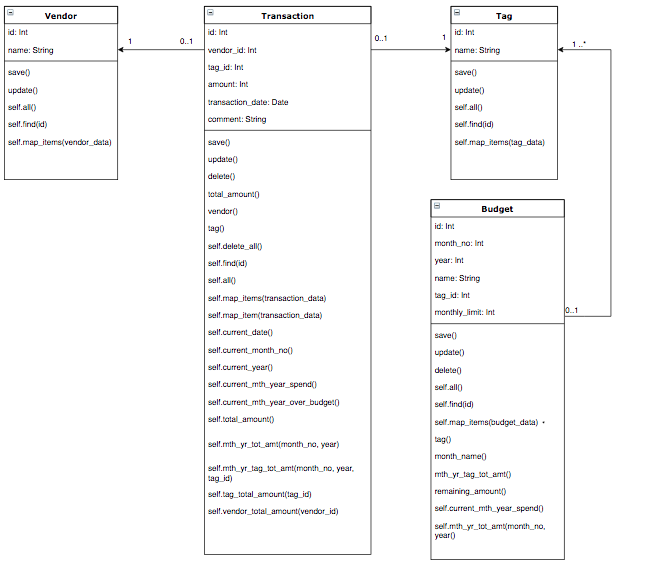
Week 5



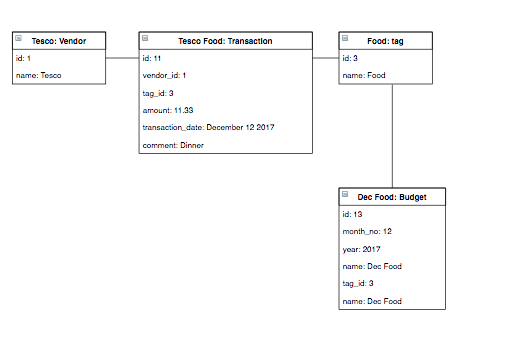
A.D 1 – A Use case diagram – Online Pizza Delivery System

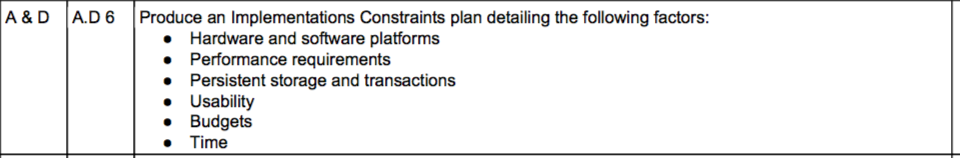


A.D 2 – Class Diagram – Money Tracker App



A.D 3 – Object Diagram – Money Tracker App





**Money Tracker Project Implementation Constraints**

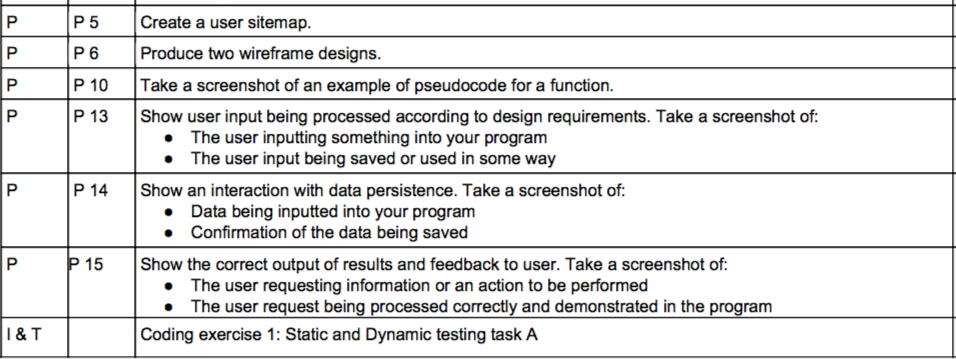
The project must be built using only:

* Sinatra
* PostGRES SQL
* HTML / CSS
* Ruby

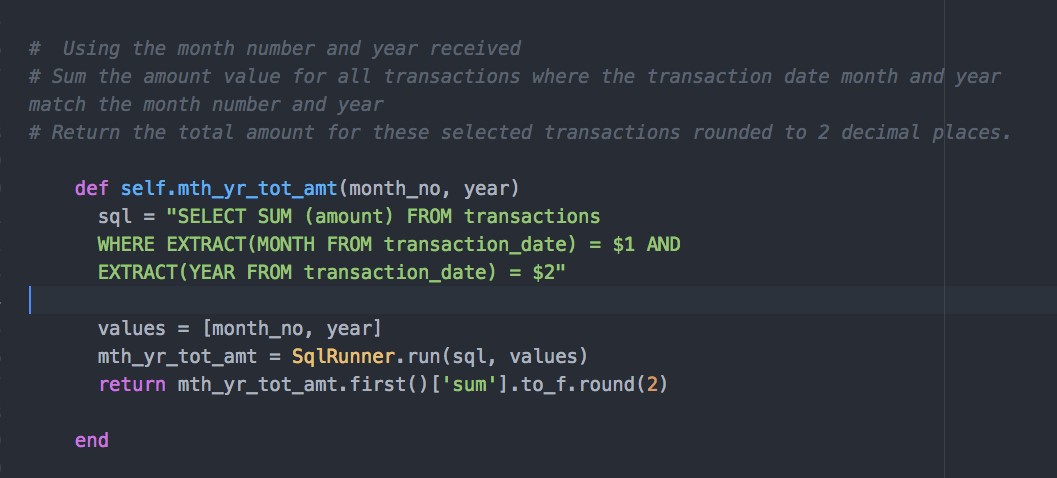
It must NOT use:

* Any Object Relational Mapper (e.g. ActiveRecord)
* JavaScript.
* Authentication / User login

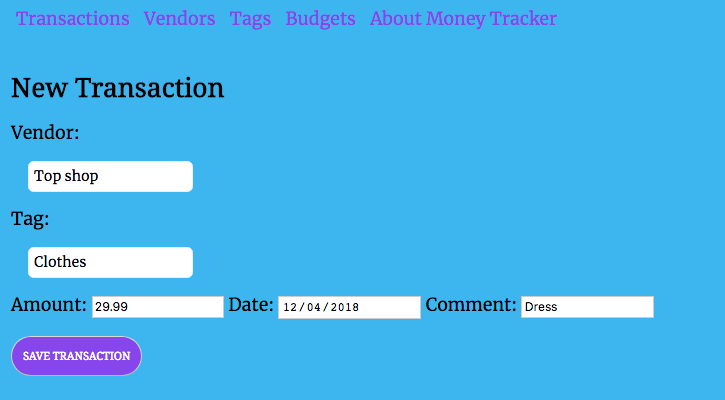
Time Constraint – 1 week



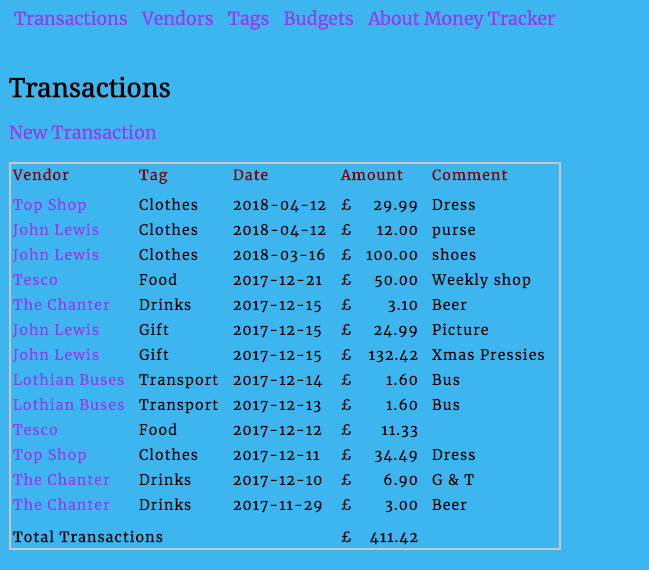
P 10 - An example of pseudocode for a function



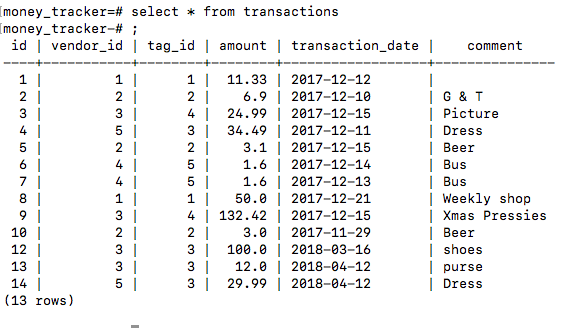
P13 – User input being processed to design requirements.



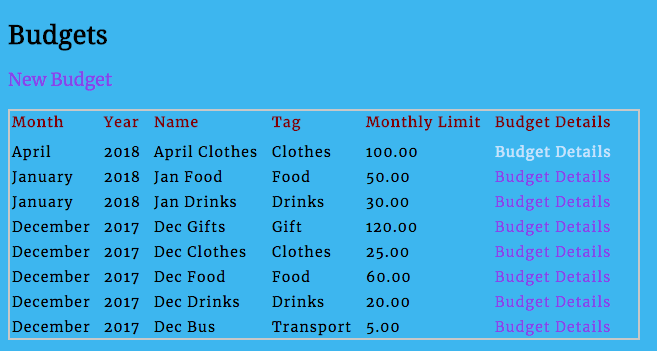
User input being saved – shows transaction has been saved and added to list of transactions – most recent “Dress” transaction at the top of the list.



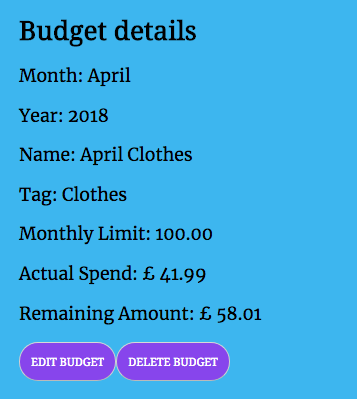
P14 – Interaction with data persistence – data input in P13 saved to database



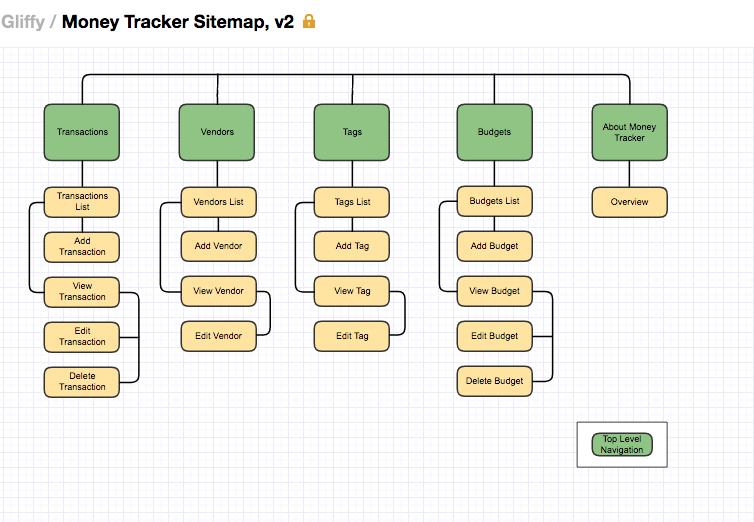
P15 - User wishes to view budget details for April 2018 and clicks the “Budget details” link for that month



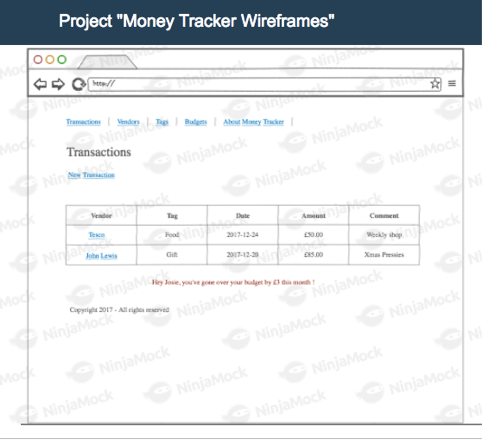
**Result** – user request being processed correctly – budget details for April 2018 displayed

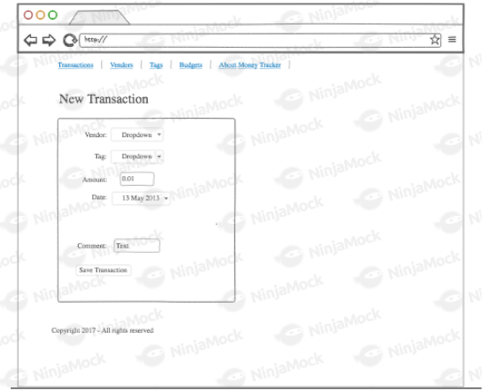


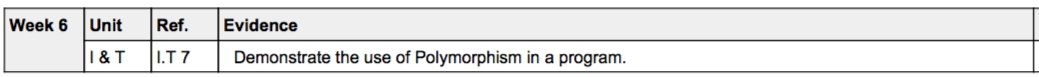
Week 5 – P5 – User Site Map – Money Tracker App



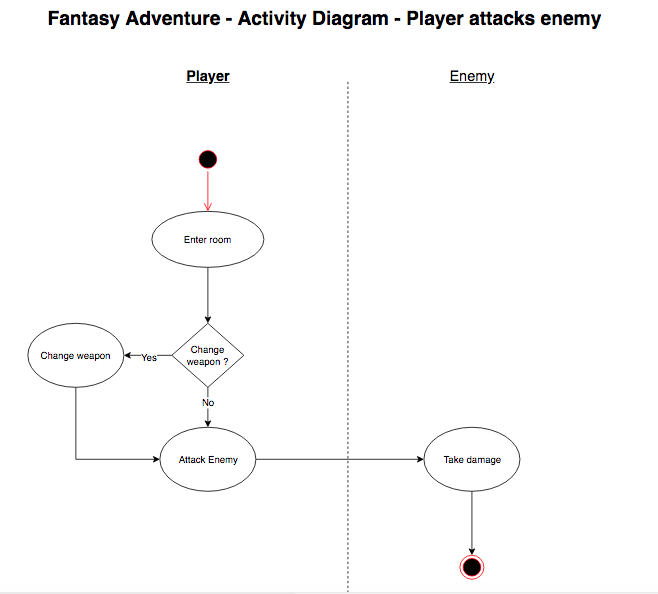
Week 5 – P6 – Two Wireframe Designs – Money Tracker App



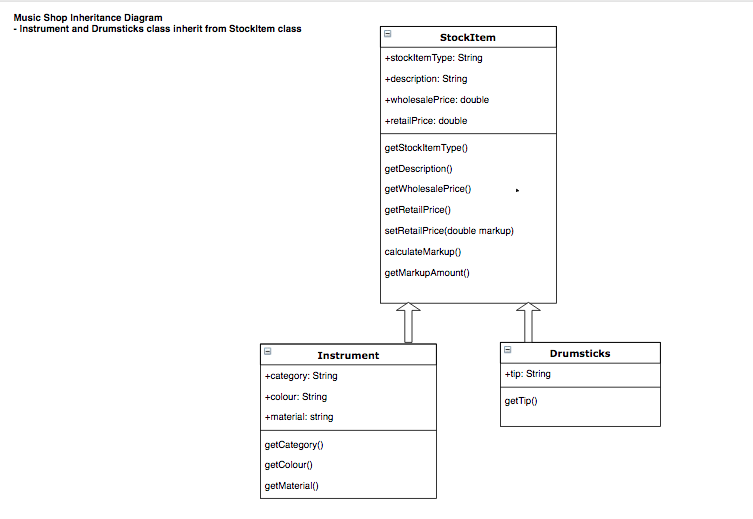




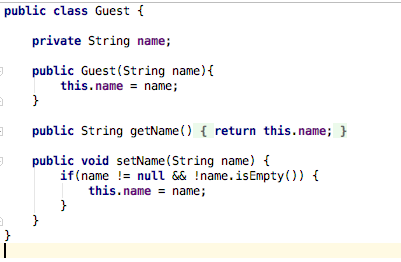
A.D 4 – Activity Diagram

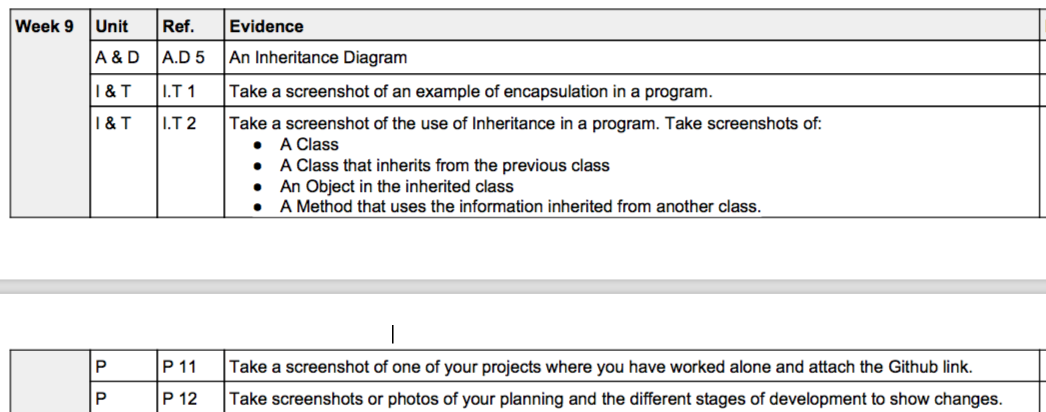


Week 9 – A.D 5 - Inheritance Diagram



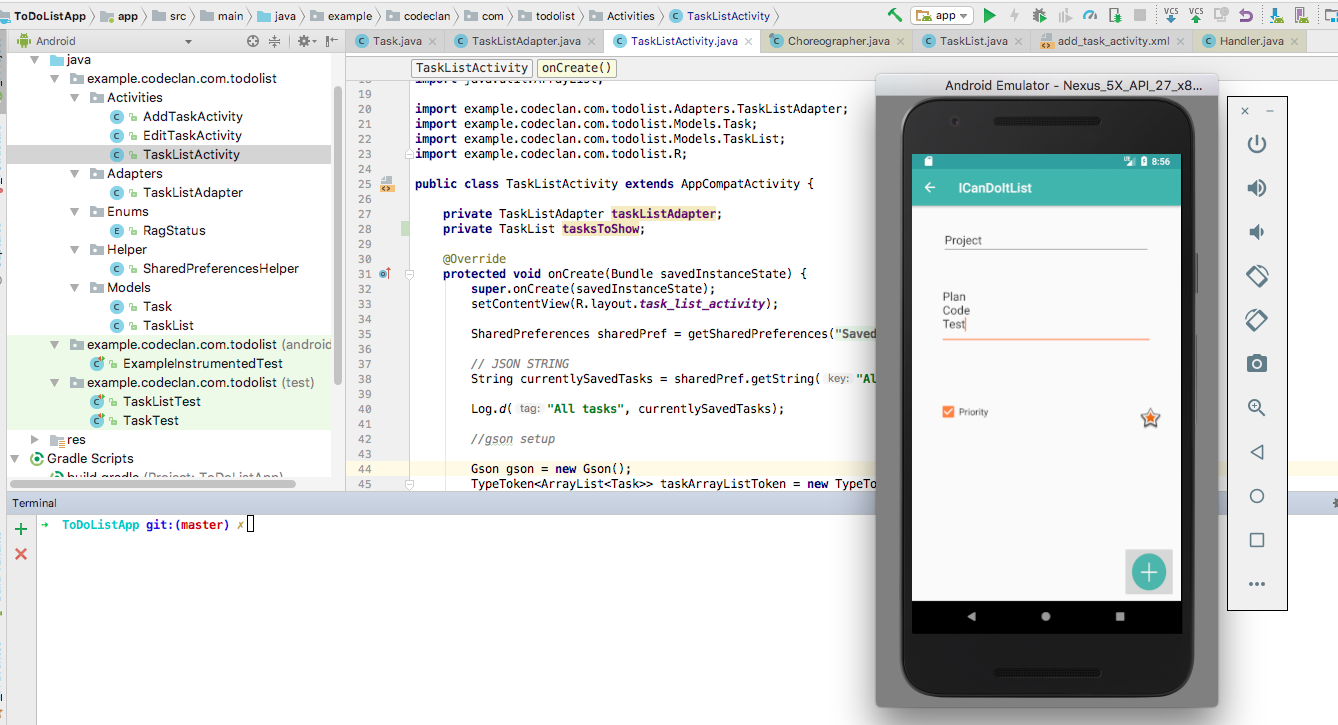
week 9 – I.T. 1 Encapsulation in a program

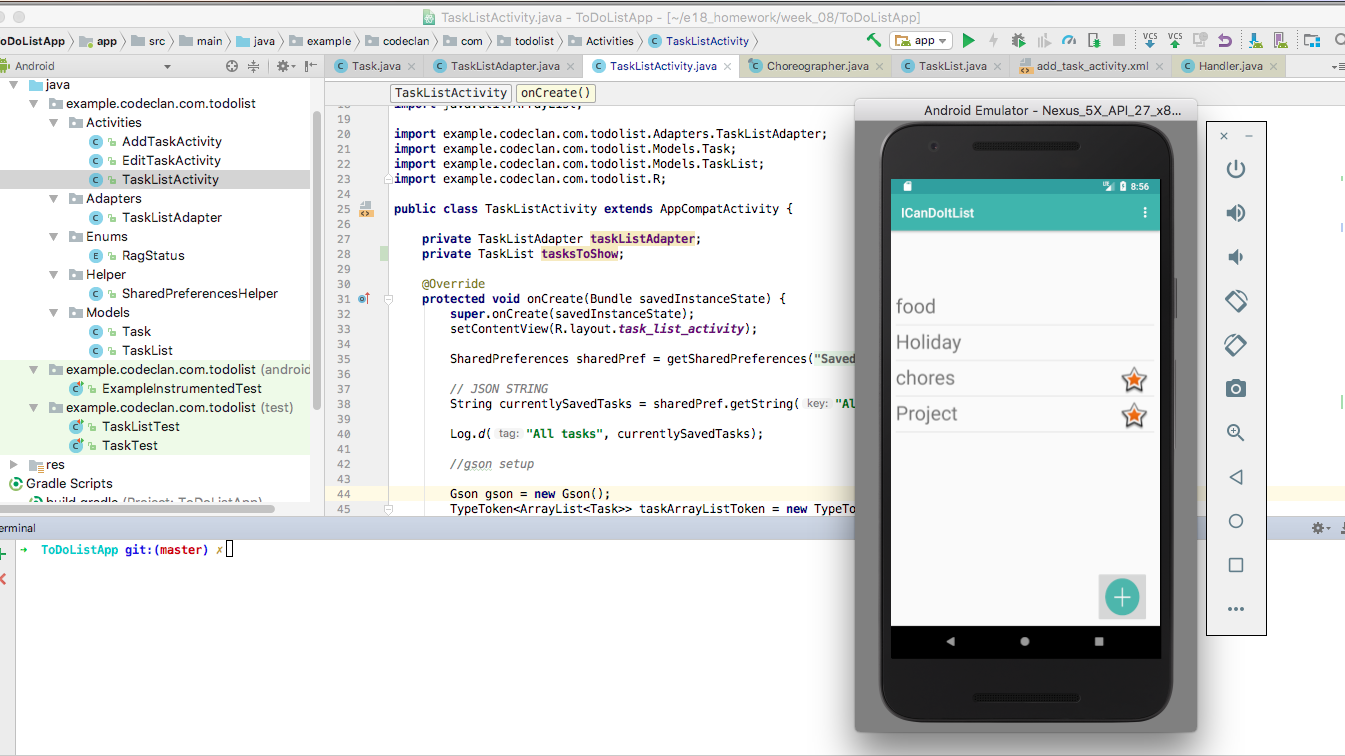


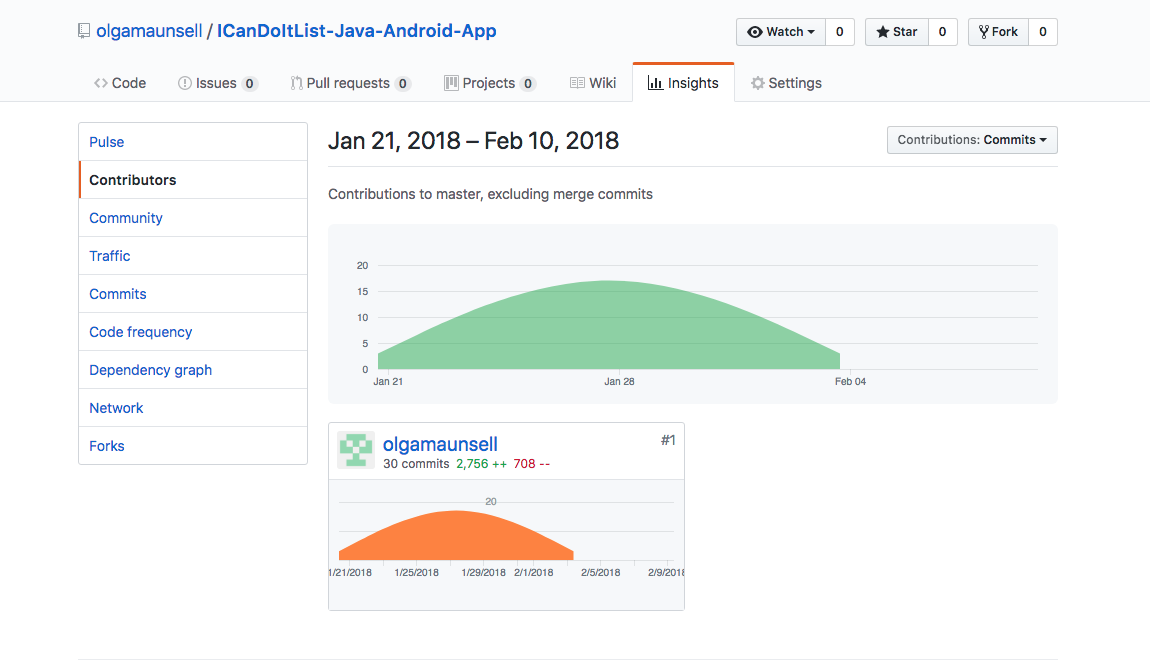


**Week 9 – P11** – Screenshots of my 2nd Individual project - a to do list app I called “ICanDoItList”

https://github.com/olgamaunsell/ICanDoItList-Java-Android-App

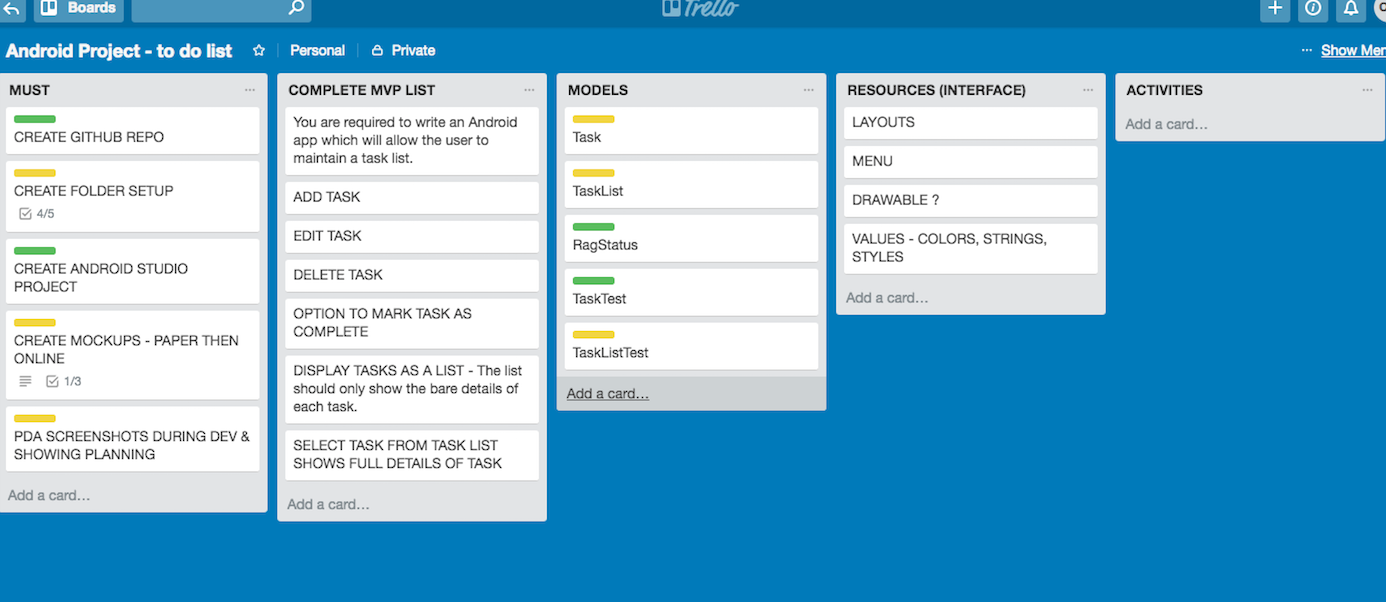


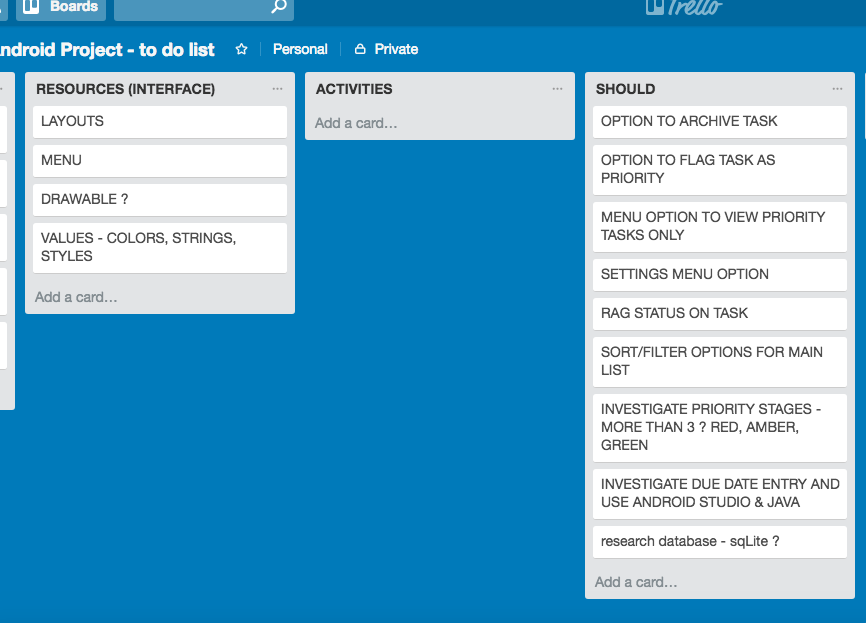




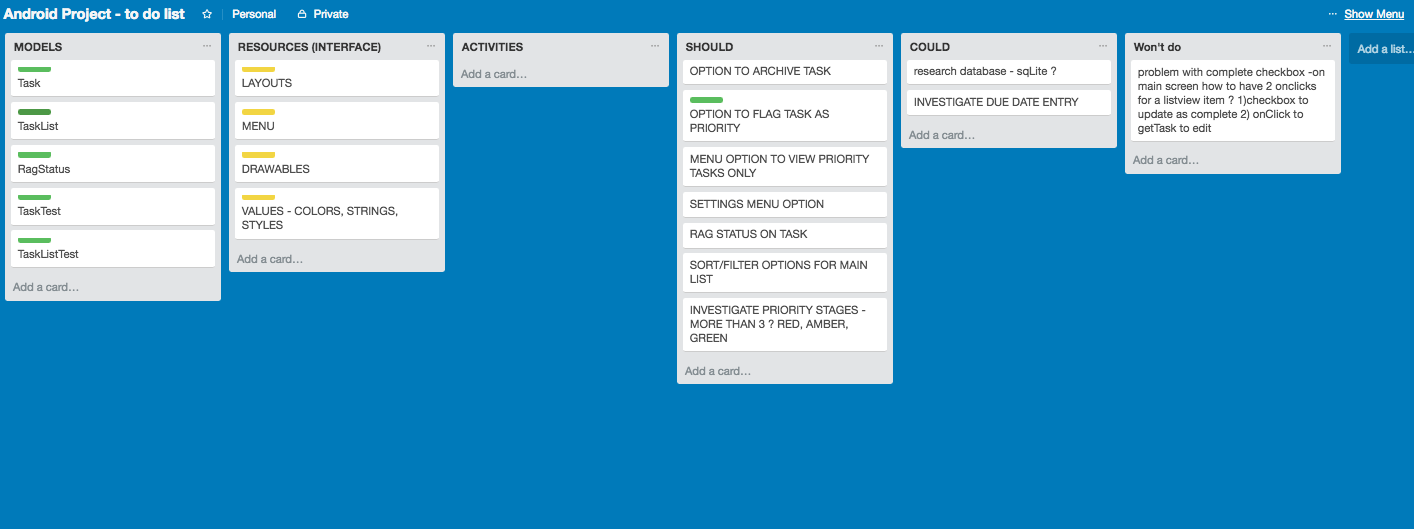
**Week 9 – P12** – Screenshots of planning and the different stages of development to show changes

Day 2 of individual Java Android Studio project (27th Jan)

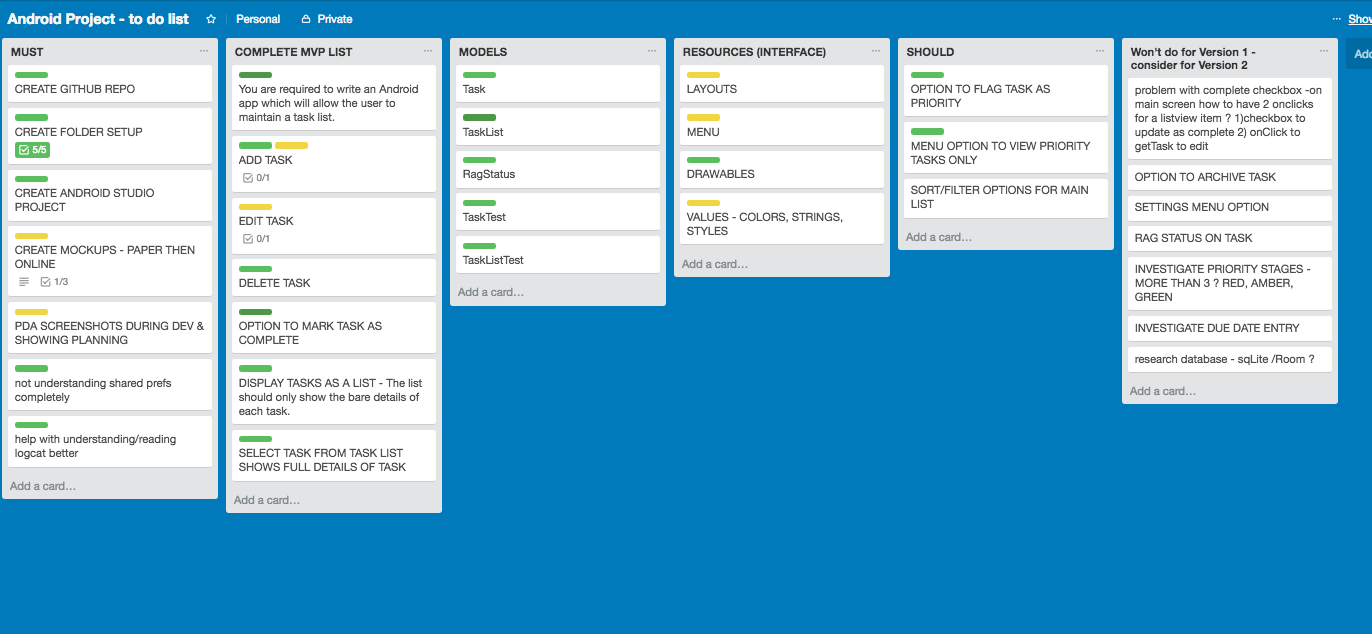




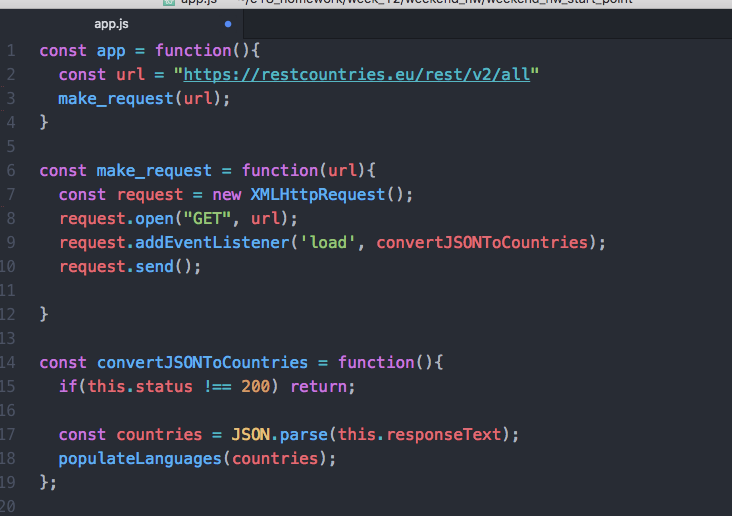
During the project week I decided after research not to include “completed task” checkbox on Main List screen – too complicated/difficult to do with Shared preferences – this would be better with a database as a future extension.



Wed 31st Jan 2018 - Updated plan on last Day



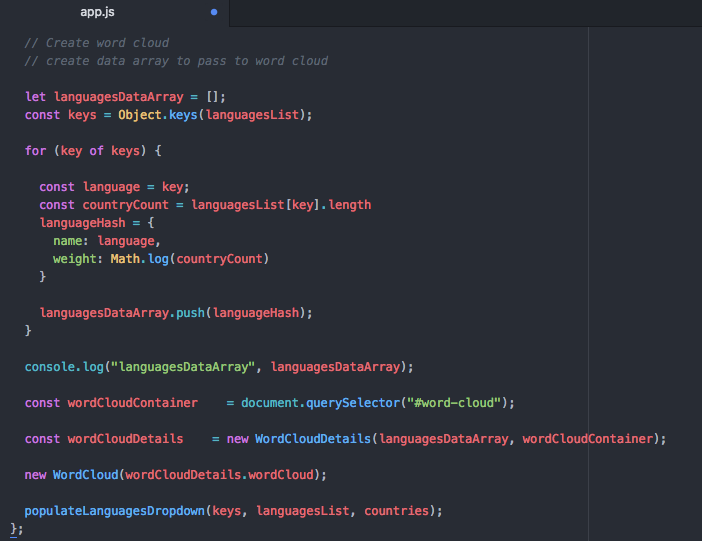
Week 12 – P16 – Show an API being used within a program



Part 1 of method populateLanguages



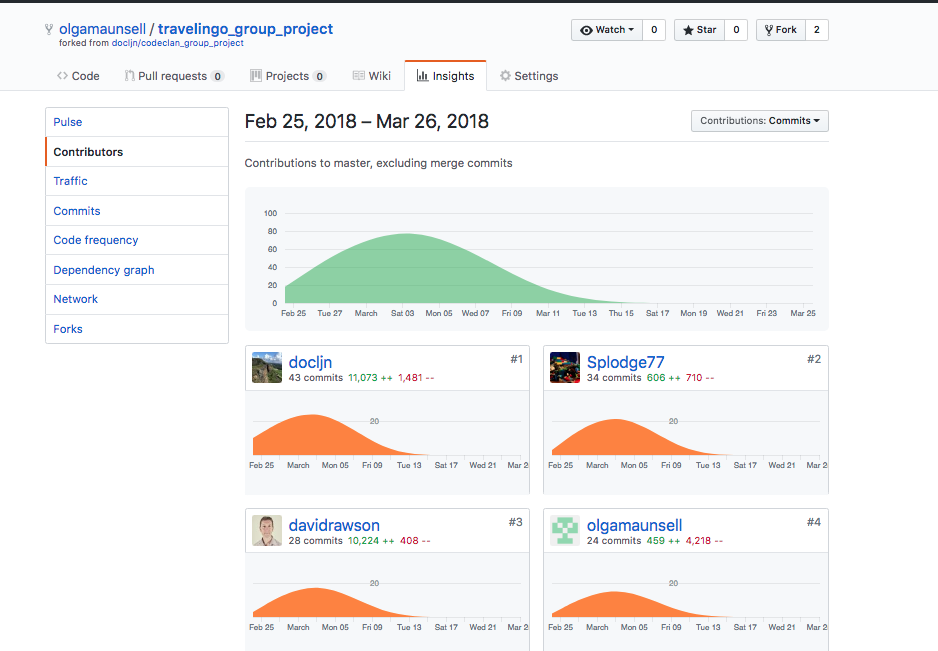
Part 2 of method populateLanguages



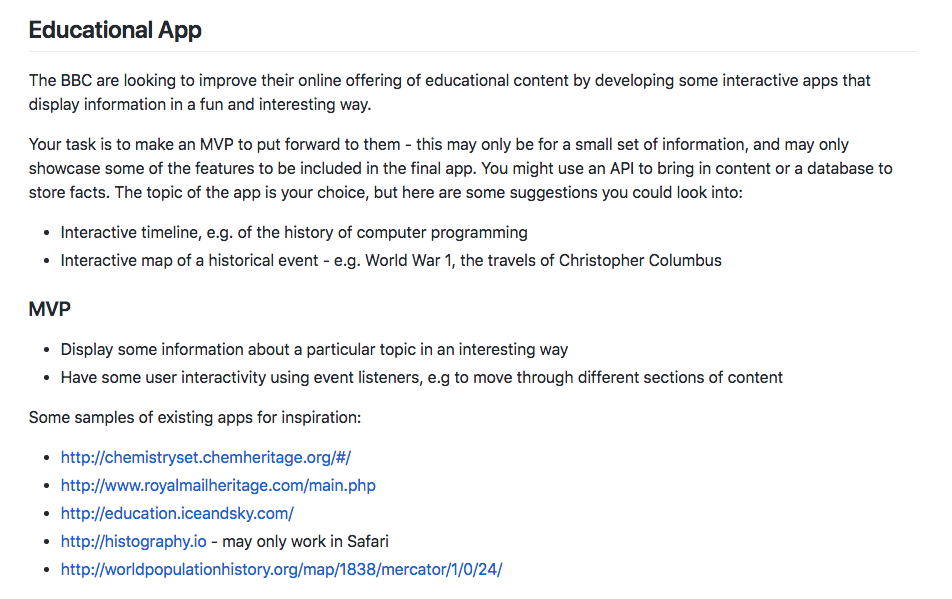
The API being used by the program whilst running



Week 14 – P1 – Contributors page – Group project

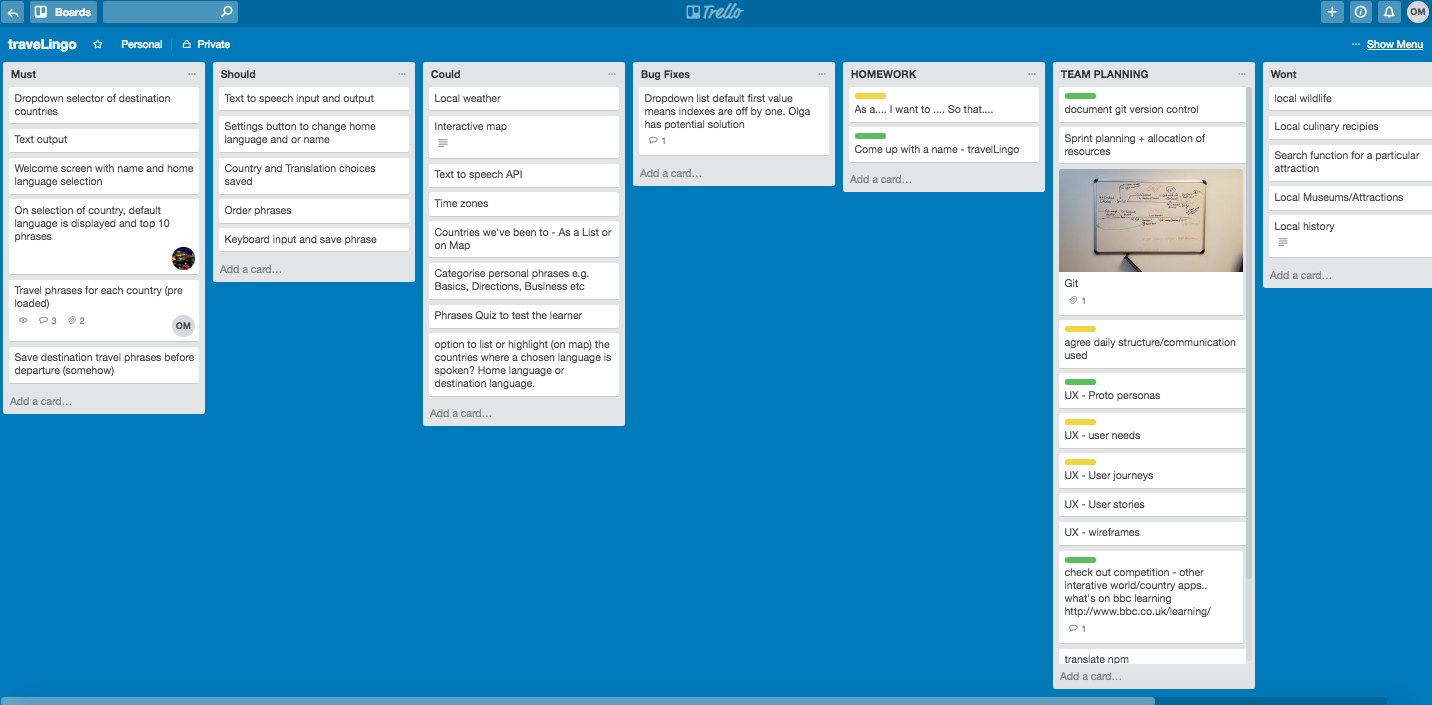


P2 – Screenshot of project brief – Group Project

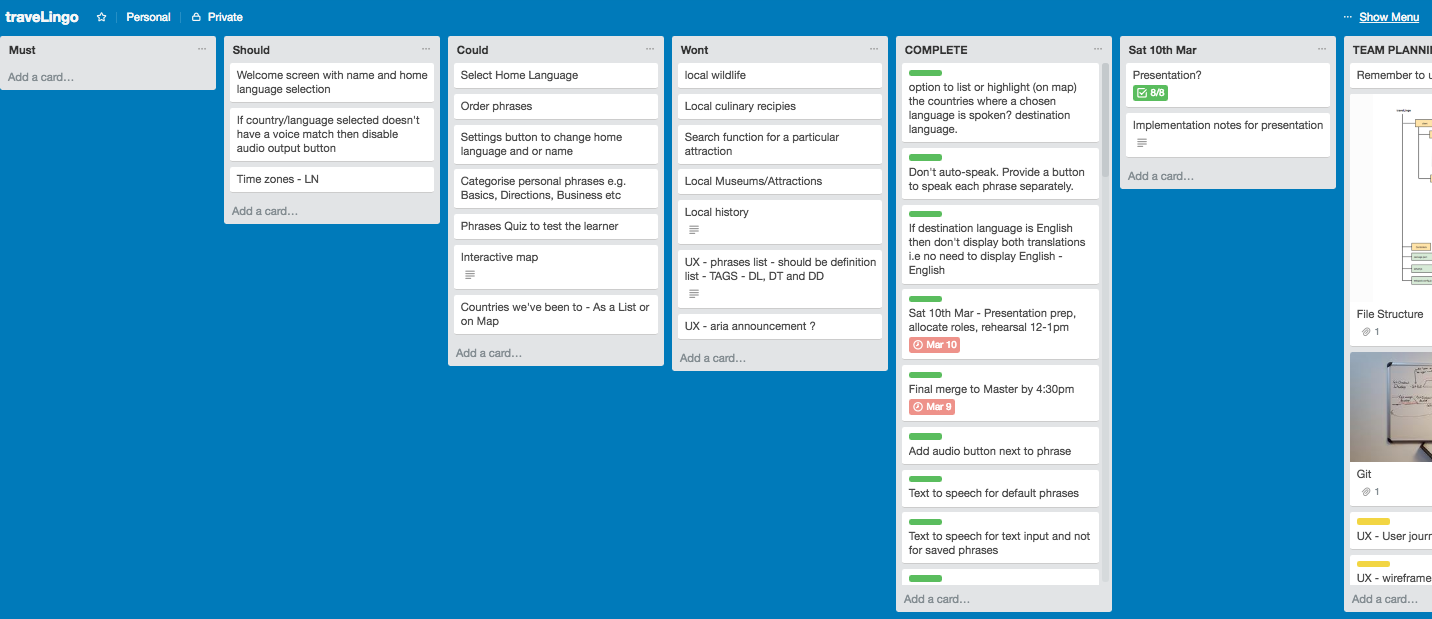


P3 – Planning – Group Project

Start of group project



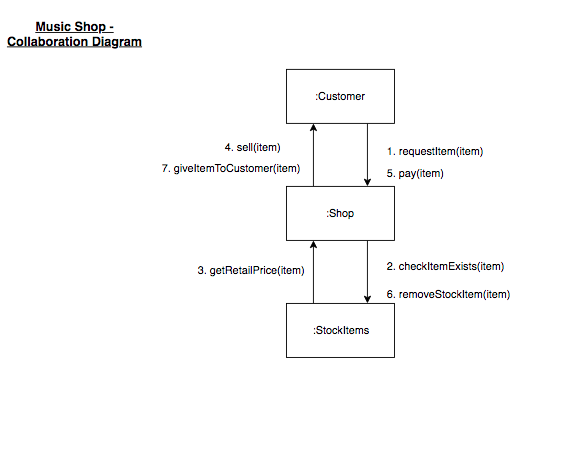
End of group project

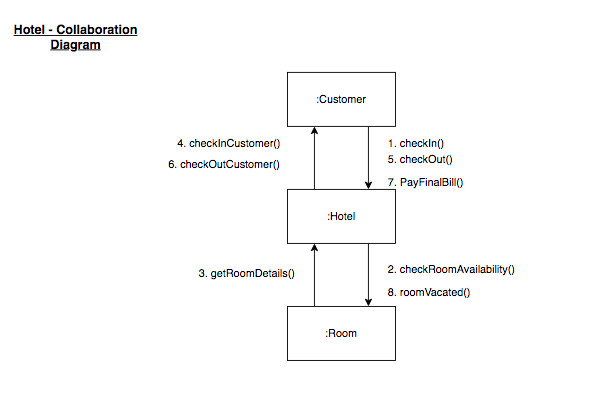
****

**P4 – Money Tracker App – Acceptance Criteria and Testplan**

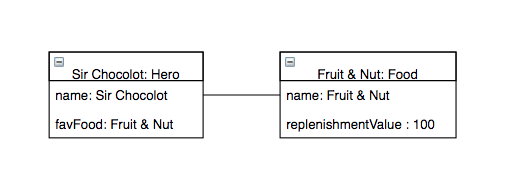
|  |  |  |
| --- | --- | --- |
| **Acceptance Criteria** | **Expected Result/Output** | **Pass/Fail** |
| An option is provided to add a new transaction | When a user selects “New Transaction” a new screen will appear allowing user to enter Vendor, Tag, Amount, Date, Comment and save the transaction | Pass |
| When a user enters and saves a transaction, transaction is created and can be viewed in the transaction list. | When user enters transaction details and saves transaction a message “Transaction created” is displayed. The new transaction is now also displayed on the main transactions screen. | Pass |
| When a user enters a transaction but does not select the save button, the transaction is not saved | Transaction is not saved.  Transaction is not displayed on the main transactions screen. | Pass |
| User can view all transactions added | The main transactions screen lists all transactions added by the user in descending date order | Pass |
| User can view total amount of all transactions | The total amount of all transactions is displayed to 2 decimal places at the bottom of the transactions list. | Pass |
| User can select an individual transaction | User can select a transaction from the main transactions screen , a new screen will appear displaying the transaction details with options to edit or delete | Pass |
| User can edit a transaction | When user selects to edit a transaction a new screen is displayed allowing user to edit and update transaction details. On update the main transaction screen will display the new details. | Pass |
| User can delete a transaction | When user selects to delete a transaction, the transaction is deleted. The transaction can no longer be viewed on the main transactions screen | Pass |

P7 – 2 System Interaction Diagrams





P8 – Object Diagram



**week 6 – I.T. 7 -** polymorphism

Polymorphism (I&T.7): I’m looking for about four screenshots and the best example to look at is if you were to do the bear, river and fish example you did as part of the Java unit. You need to use a different example, as this was a code-along. However, to use this as an example we are looking for the following:

- Screenshot 1: The **bear class.** In the bear class you should have an **edible** arraylist (called belly), which should take in different types of food that **implement edible**, such as a fish or human. This screenshot should include this, plus a method that populates the arraylist by adding different types of food to the arraylist.

- Screenshots 2 & 3: Two different classes/types that **implement** the **edible interface**. For example, if you have **food** then populating the array list with **humans, fish,** etc.

- Screenshot 4: The edible interface.

<End of Document>