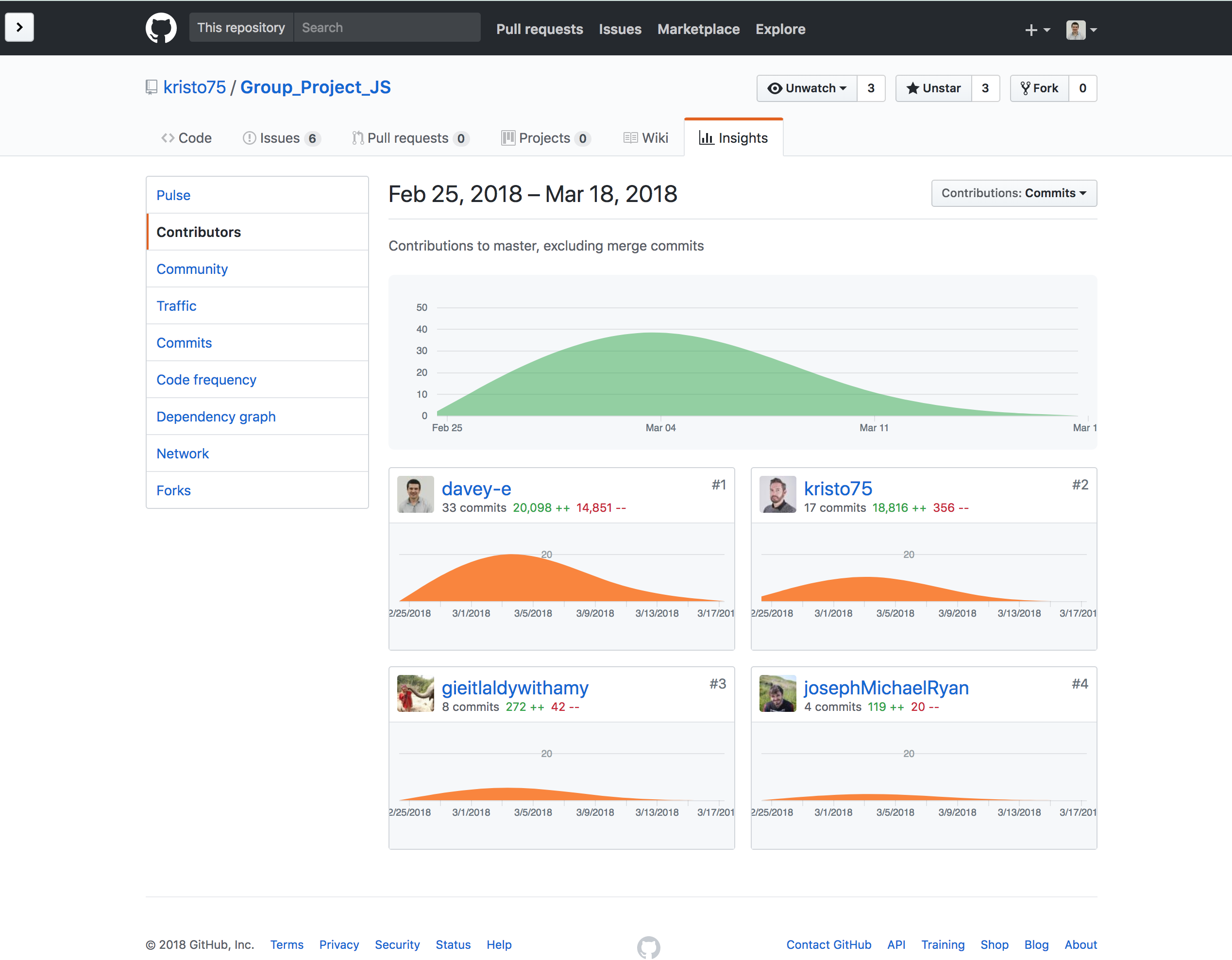
# Project Unit – Evidence

## **David Ellis**

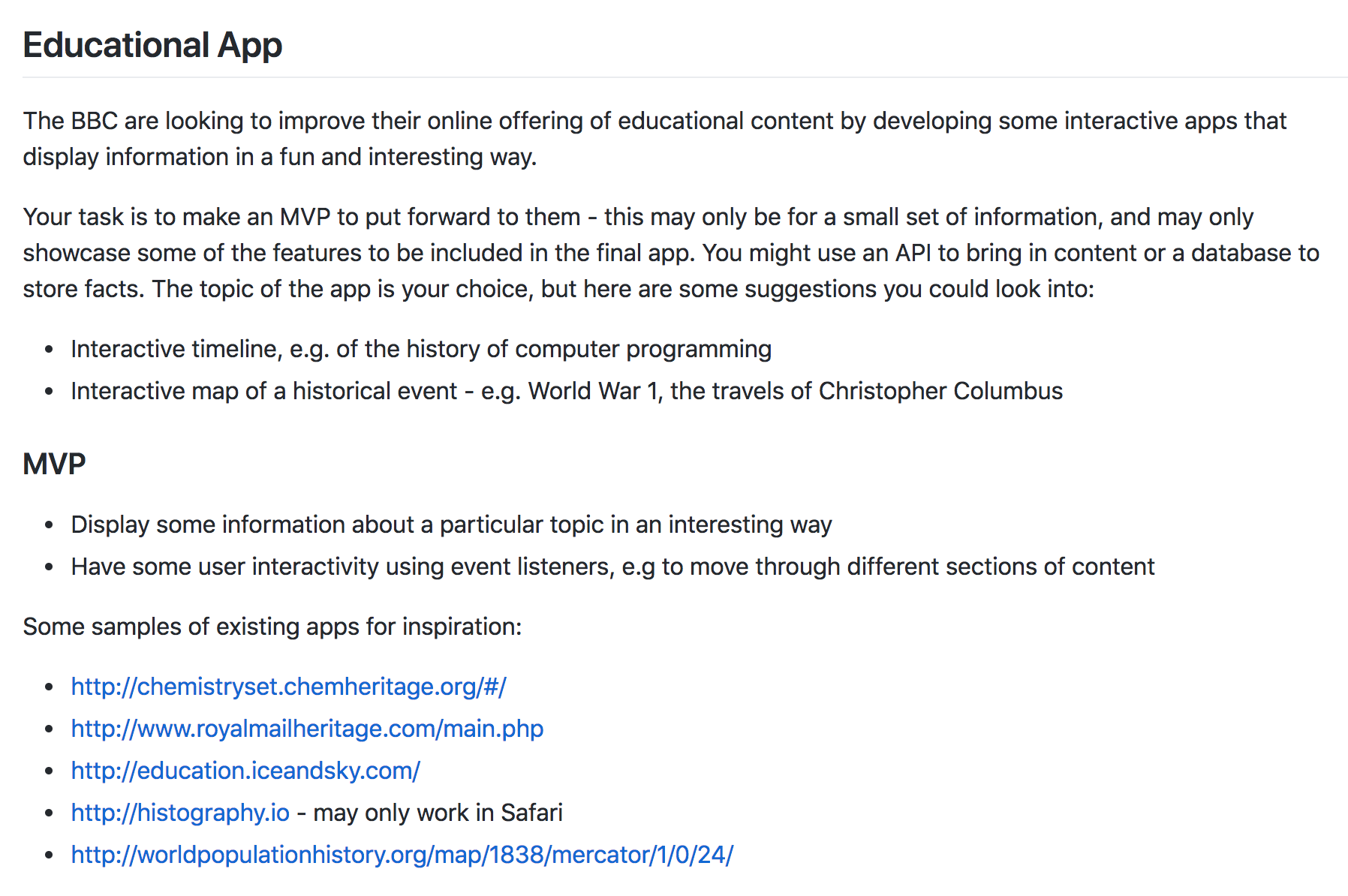
## **Cohort E18**

## P1 – Group Project – GitHub Contributors Page Screenshot

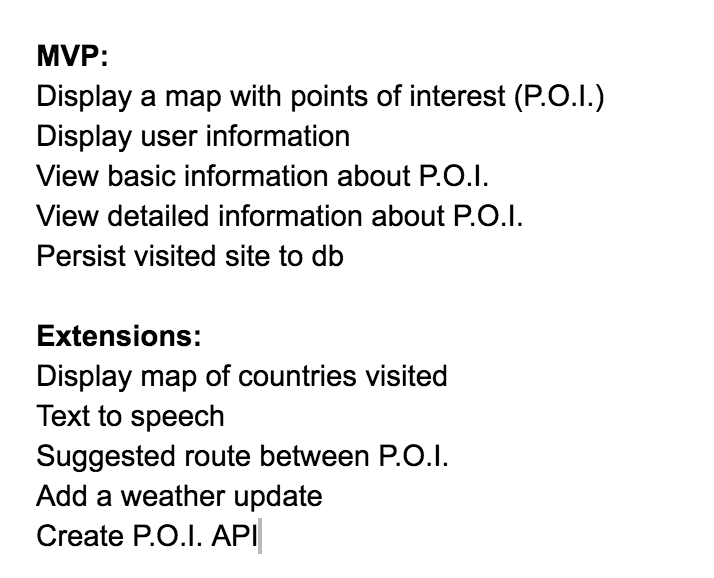


## P2 – Group Project – Project Brief Screenshots

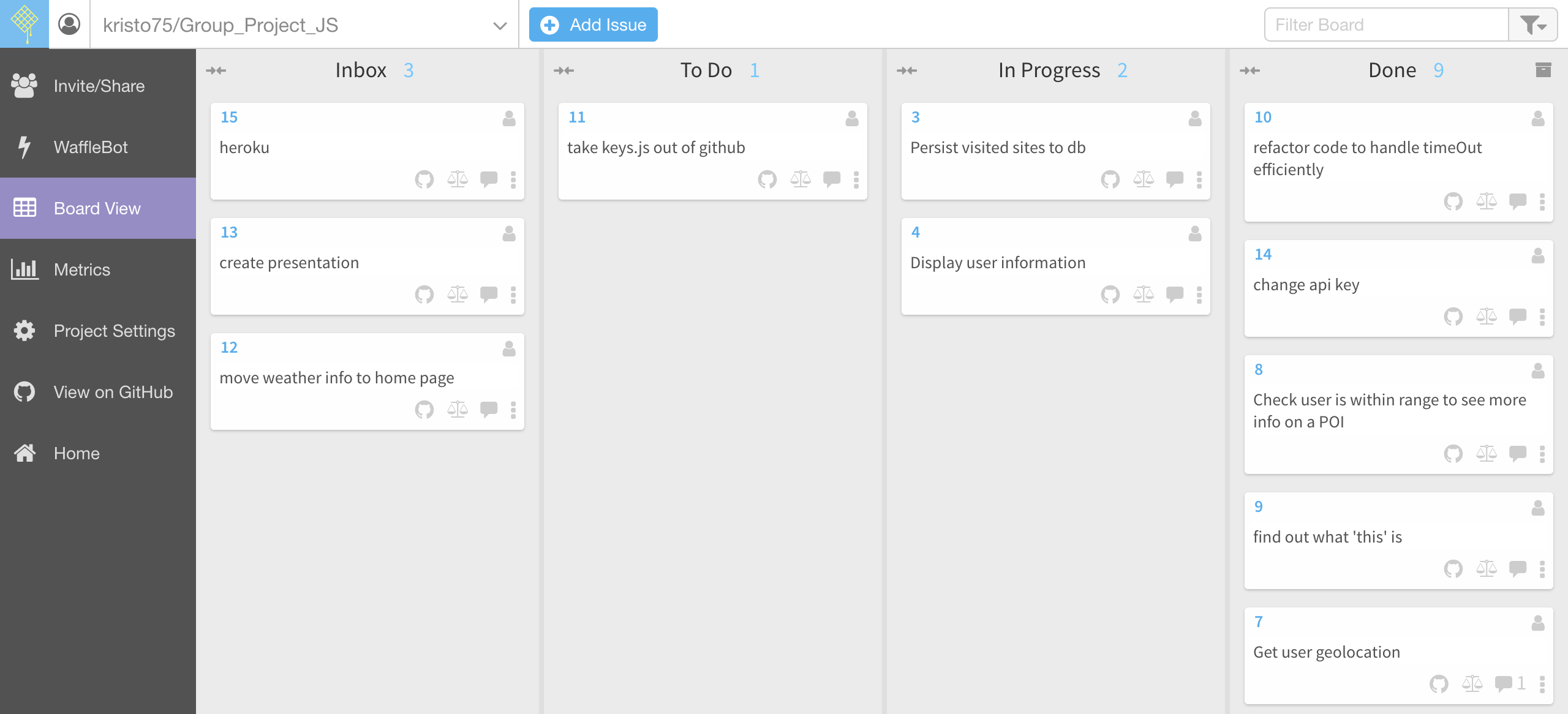
### Project Brief From Classnotes:



### Project brief that we created for our specific app:



## P3 – Group Project – Planning Screenshots



## P4 – Acceptance Criteria and Test Plan

|  |  |  |
| --- | --- | --- |
| Acceptance Criteria | Expected Result/Output | Pass/Fail |
| A user is able to see a list of all transactions in the system for all tags | The main page of the app displays a table with all of the transactions for all tags fetched from the database shown. The table can be scrolled to shown transactions that are at the bottom of the table | Pass |
| A user is able to see the total amount spent for all transactions for all tags | The main page of the app displays the total amount of all transactions in the header bar | Pass |
| A user is able to see a list of the available tags that can be selected to filter the data | The main page of the app shows buttons for each of the tags that are in the system in the navigation bar on the left-hand side. | Pass |
| A user is able to filter the data by selecting one of the available tags | The user can click on each tag button in the navigation bar on the left-hand side to filter the data in the table by tag |  |
| A user is able to add a new transaction to the system | A New Transaction button is displayed on the navigation bar on the left-hand side of the main page of the app. When the user clicks on this button a form is displayed to allow the user to ender the transaction data | Pass |
| A user is able to select the date, amount, tag and vendor when adding a new transaction | The New Transaction form displays input elements for date, amount, tag and vendor and also displays a Submit button | Pass |
| A user is able to add a new tag to the system | A New Tag button is displayed on the navigation bar on the left-hand side of the main page of the app. When the user clicks on this button a form is displayed to allow the user to ender the new tag | Pass |
| A user is able to add a new vendor to the system | A New Vendor button is displayed on the navigation bar on the left-hand side of the main page of the app. When the user clicks on this button a form is displayed to allow the user to ender the new vendor | Pass |

## P5 – User Sitemap



## P6 – Wireframe Diagrams





## P7 – System Interaction Diagrams \*W14

Produce two system interaction diagrams (sequence and/or collaboration diagrams: 2 sequence diagrams, 2 collaboration diagrams or 1 of each).

## P8 – Object Diagrams

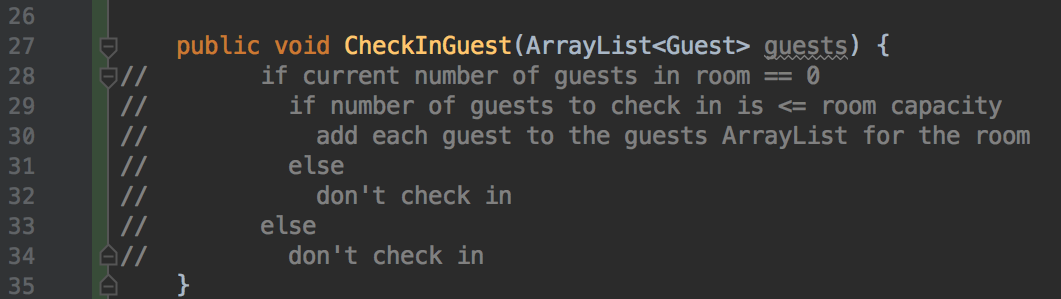




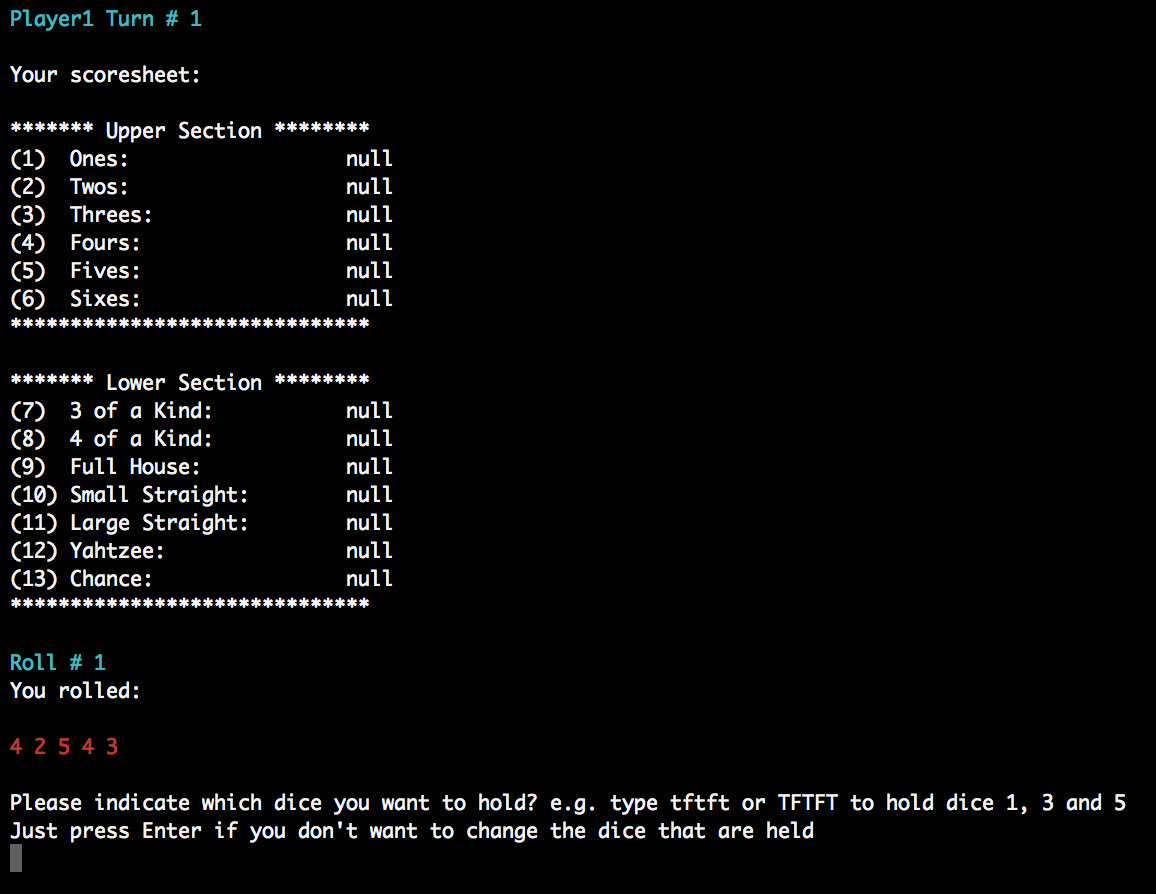
## P9 – Algorithm Example Screenshots and Explanations\*W14

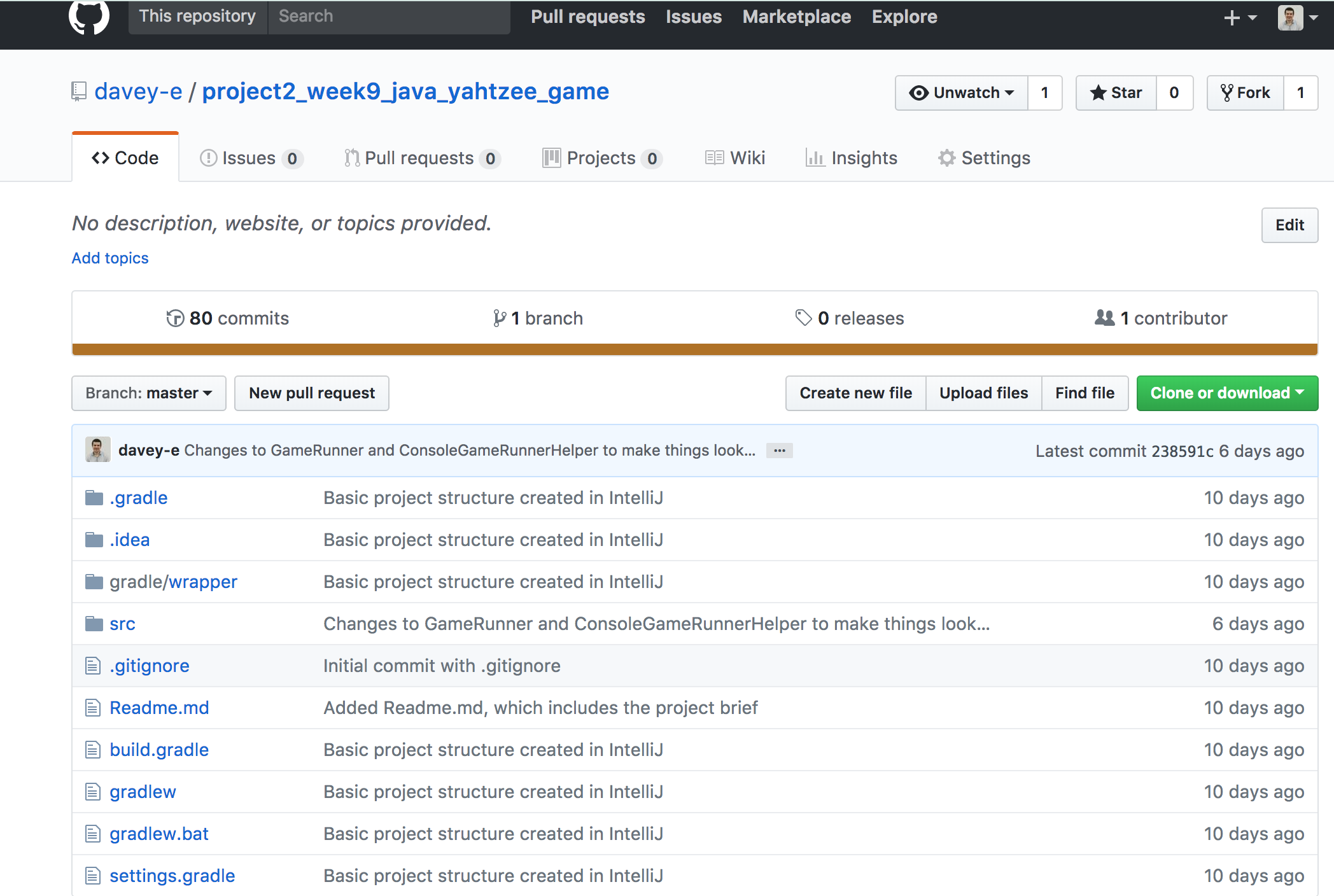
Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.

## P10 – Pseudocode Example Screenshot



## P11 – Individual Project Screenshot and GitHub Link





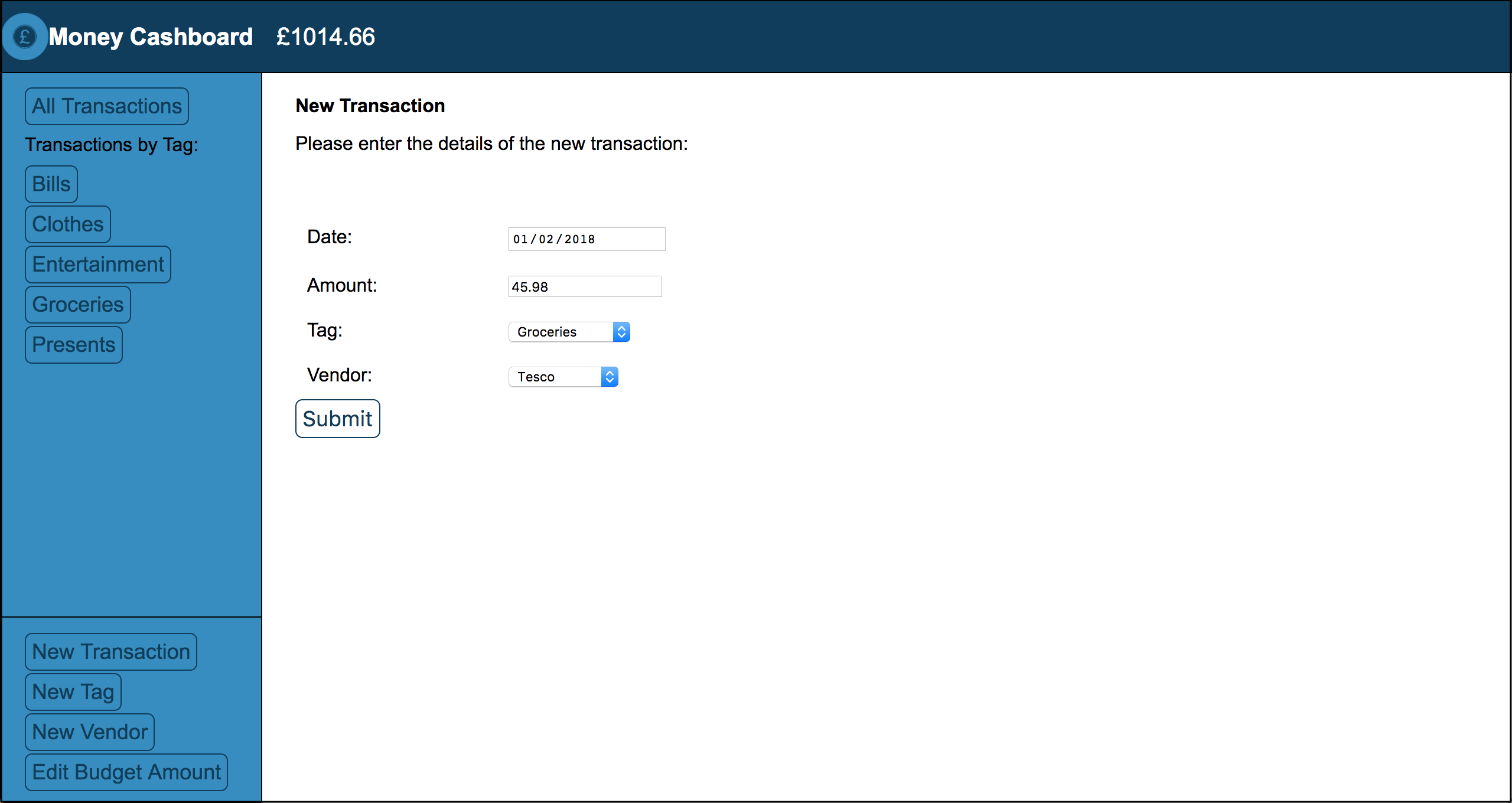
<https://github.com/davey-e/project2_week9_java_yahtzee_game>

## P12 – Project Planning Screenshots/Photos at Different Stages \*W9

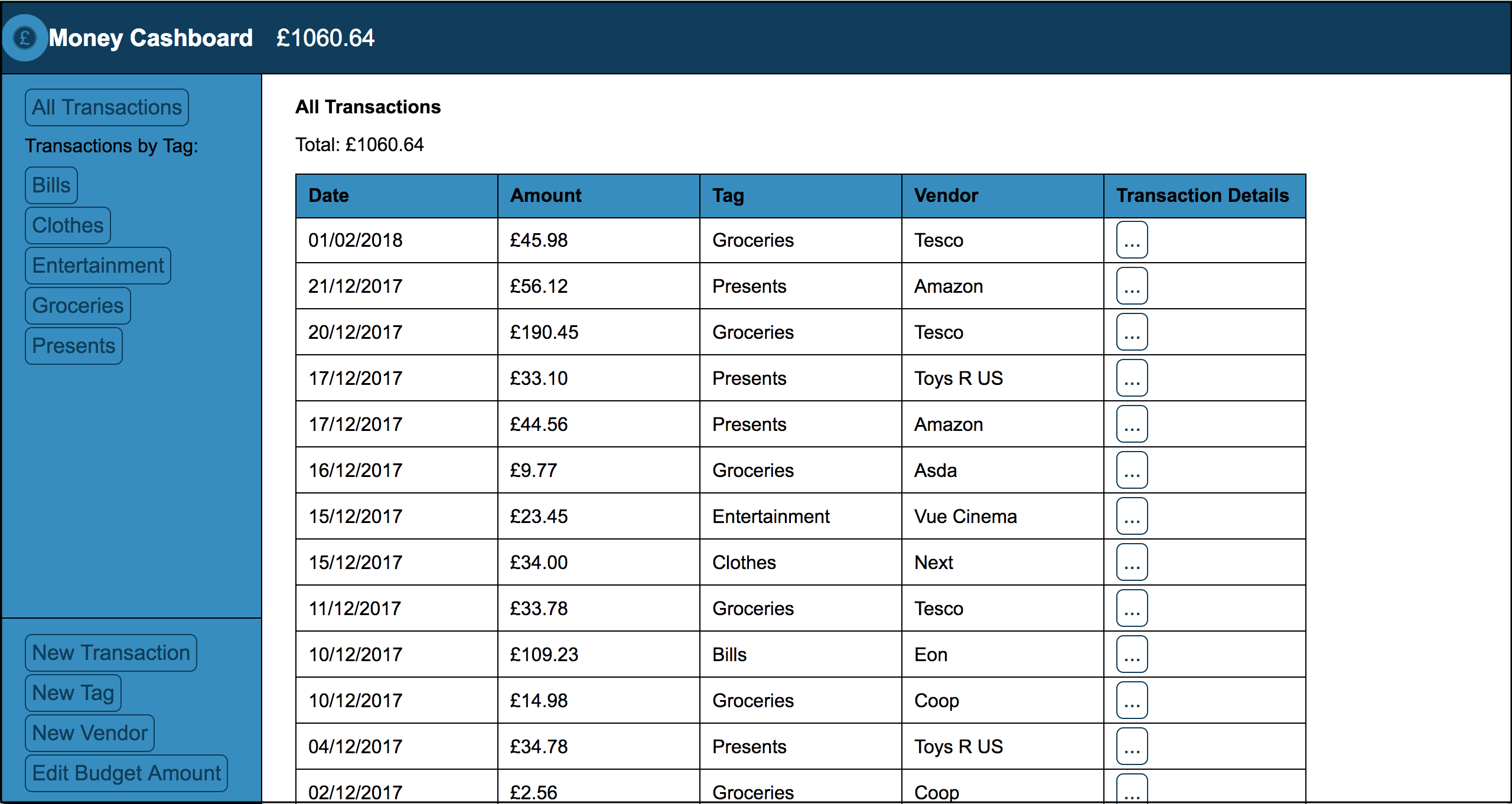
Take screenshots or photos of your planning and the different stages of development to show changes.

## P13 – User Input Processing Screenshots

User input:

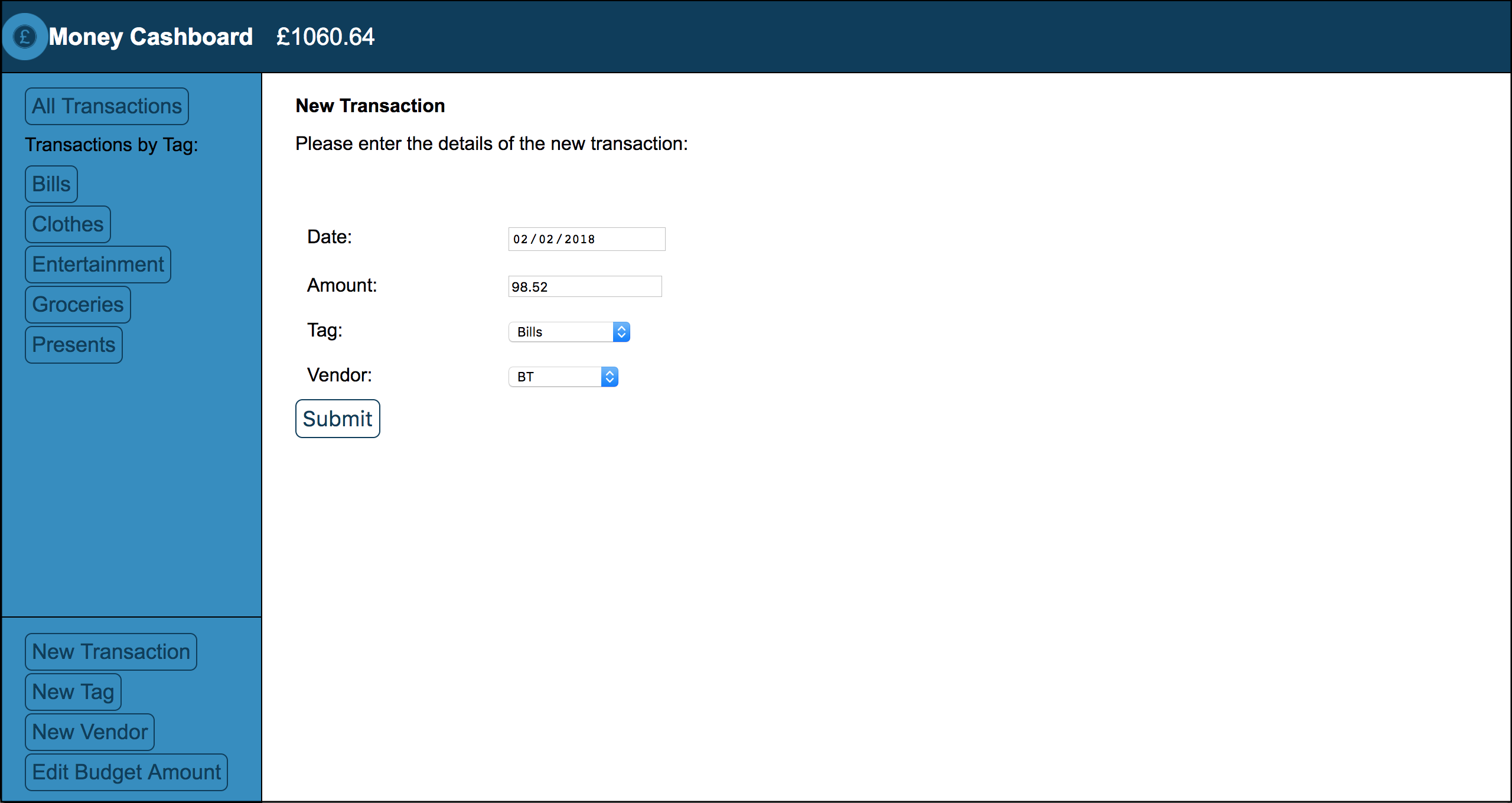


User input saved:



## P14 – Data Interaction and Persistence Screenshots

User Input:

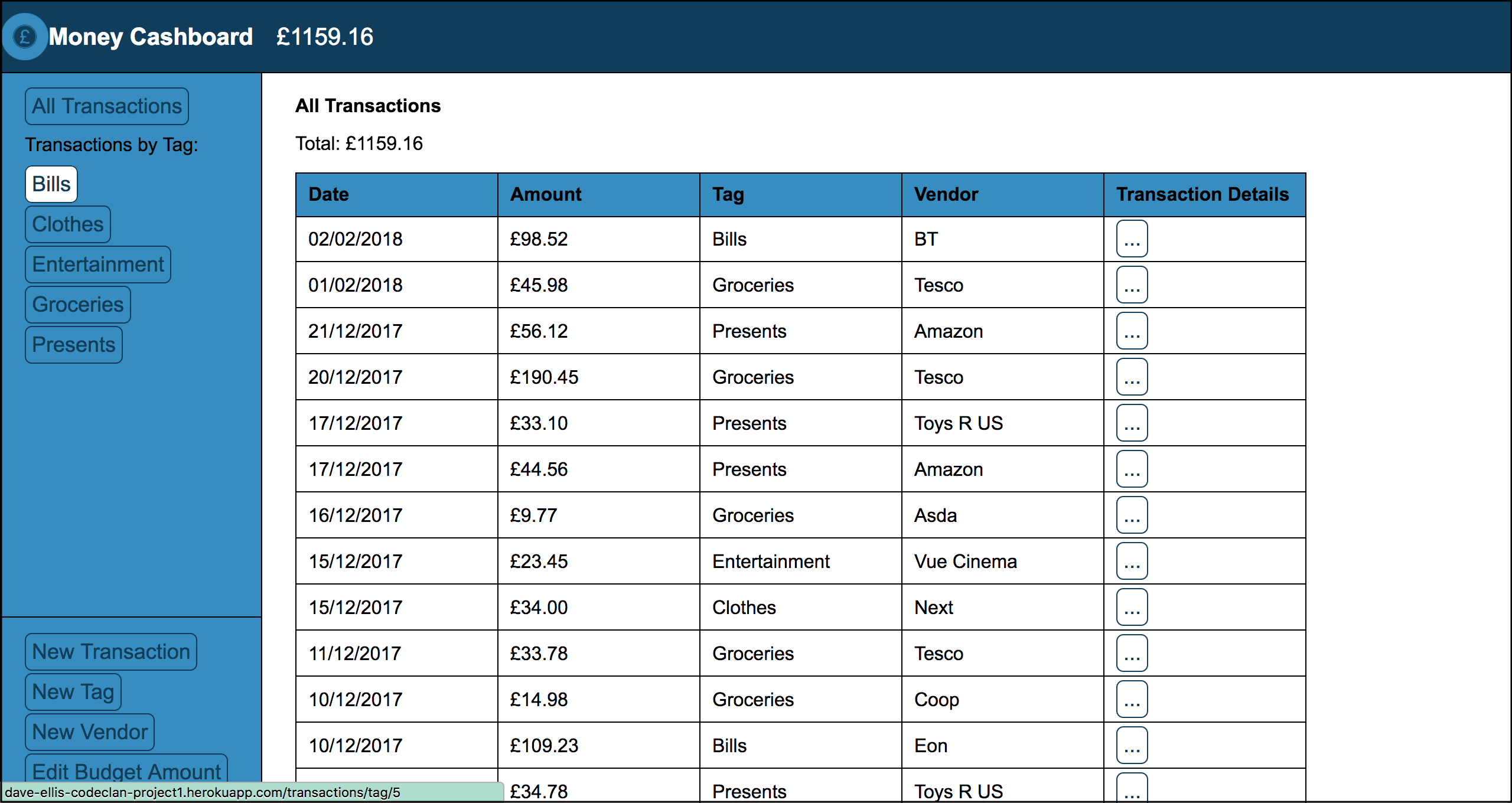


Data saved in database:

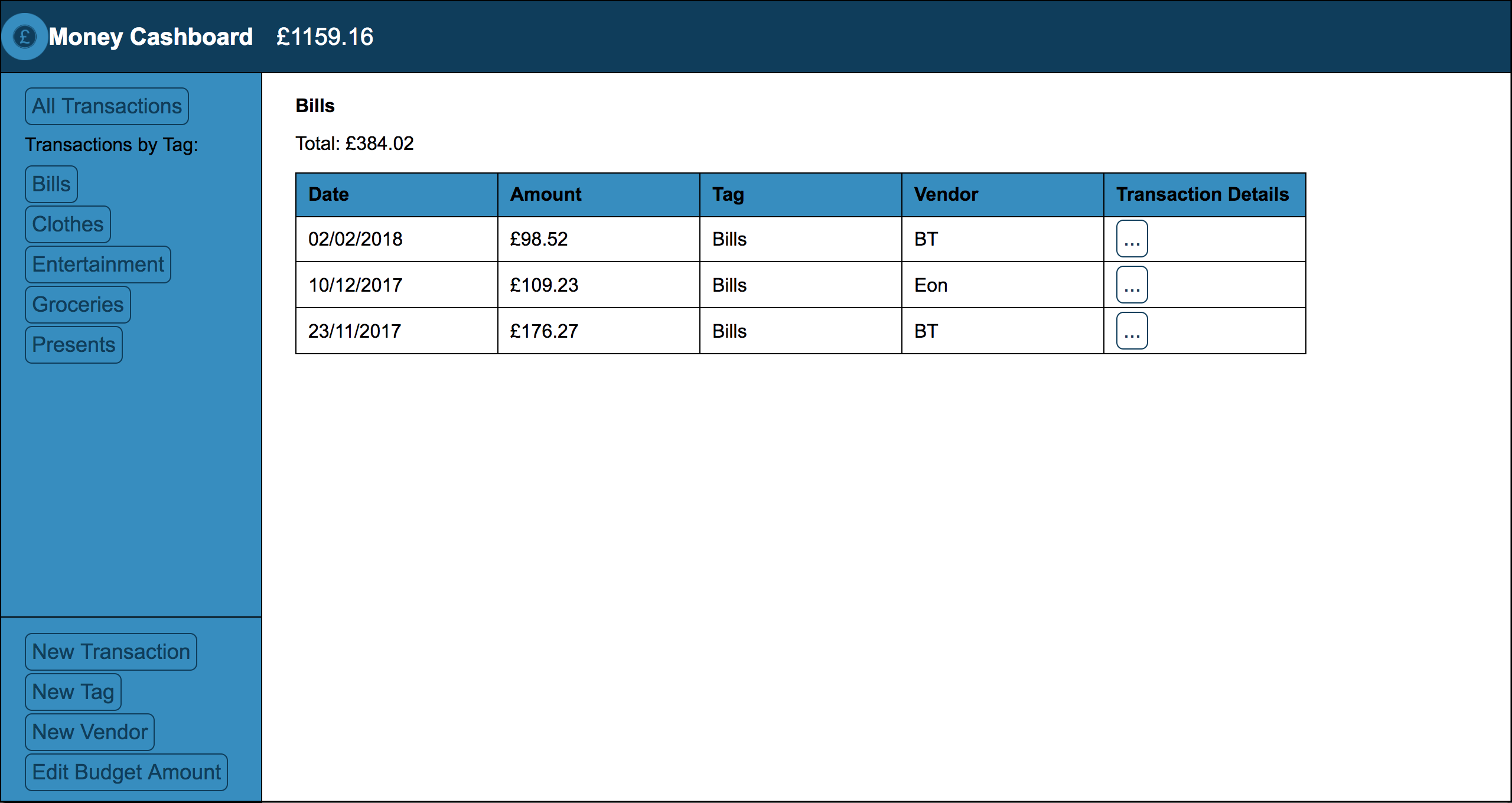


## P15 – User Feedback/Output Screenshots

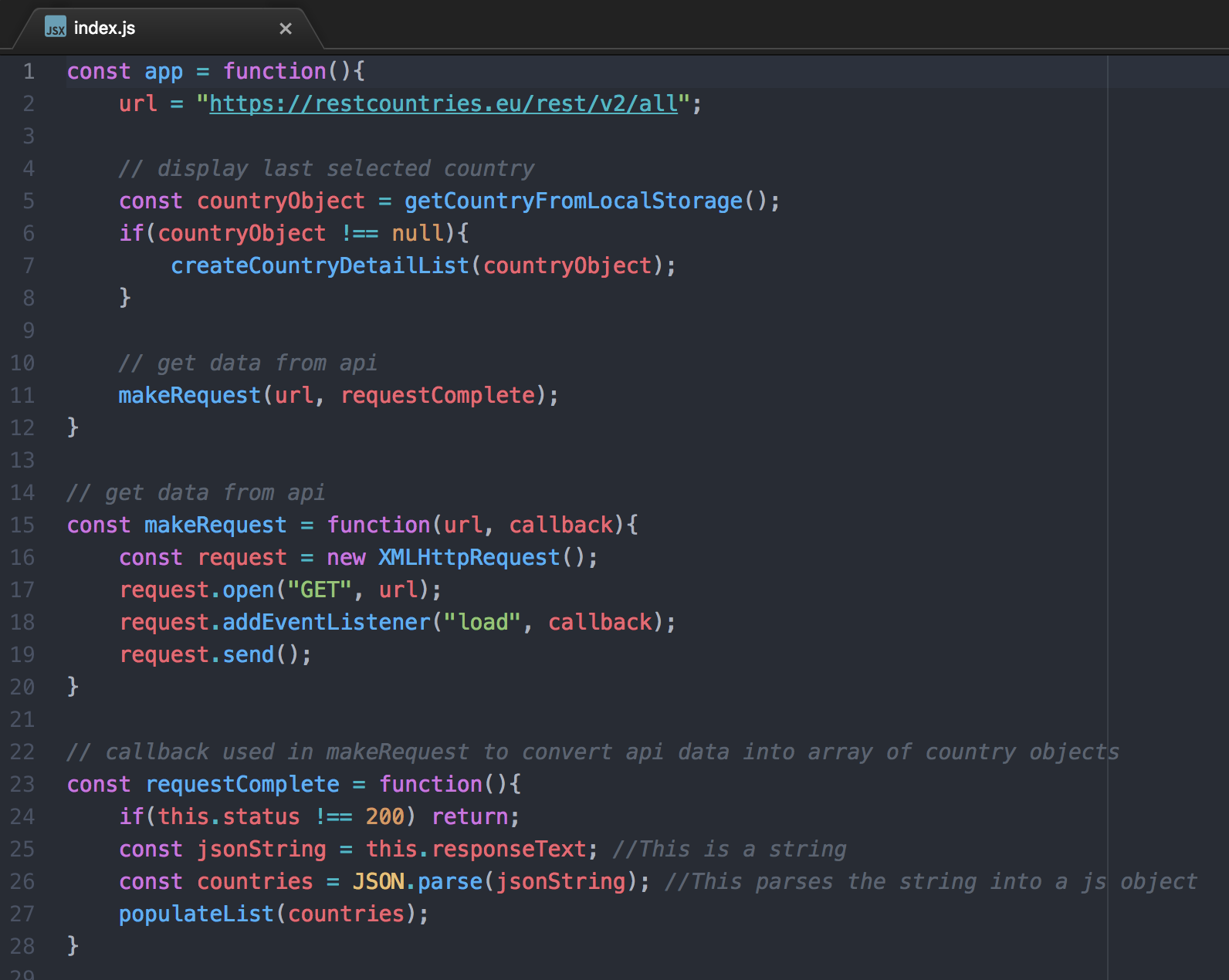
The user wants to see only the transactions with the “Bills” tag, so they click on the Bills button under Transactions by Tag:



The program responds by showing only the transactions with the Bills tag in the table:



## P16 – API Usage Screenshots





## P17 – Bug Tracking Report

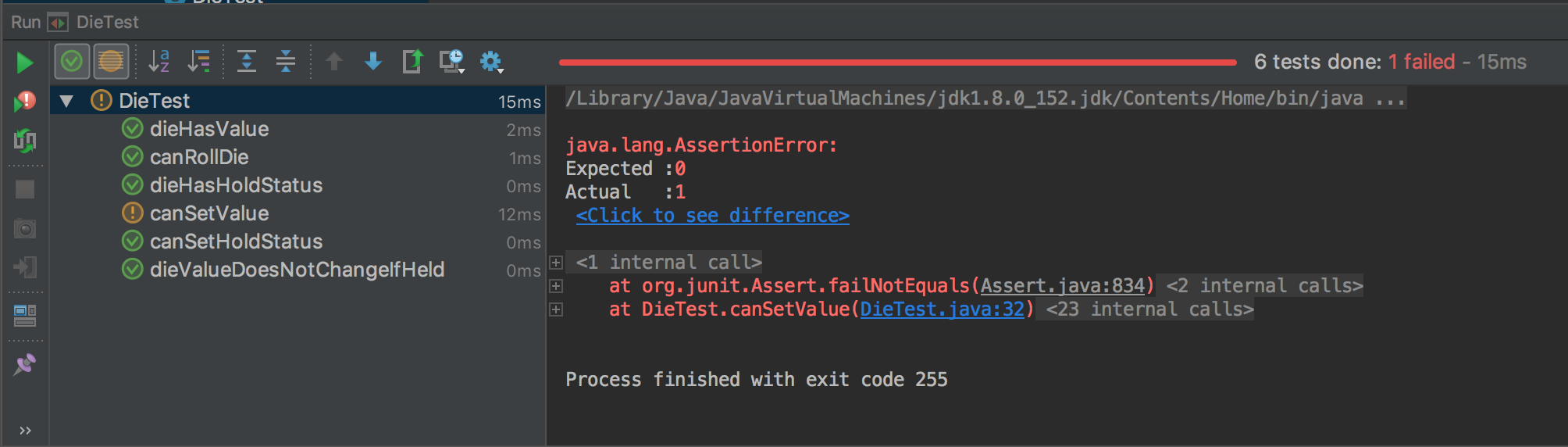
|  |  |  |  |
| --- | --- | --- | --- |
| Issue | Status | Bug Fix | Status After Fix |
| User can see 5 random points of interest from a 500m radius of their current location | Failed | Added a callback to the javascript code to make sure that points of interest are retrieved from the Sygic Travel api before randomly selecting and displaying 5 points of interest | Passed |
| User can click on the marker for a point of interest to see basic information about the selected place | Failed | Fixed a bug which was causing the information for the wrong point of interest to be displayed due to the incorrect POI id being passed to the Sygic Travel api | Passed |
| User can see detailed information for a point of interest once they are within 50m of the location | Failed | Added code to calculate the distance between the users current position and the point of interest position and then if the distance is less than 50m detailed information is retrieved from the Sygic Travel api and displayed in a modal | Passed |
| User can save a point of interest to their list of visited places once they are within 50m of the location | Failed | Added code to allow locations to be added to a Mongo database, but saving the locations only happens once the user is within 50m of the location and is triggered when they click on the marker for the location | Passed |
| User can see a list of the points of interest that they have visited | Failed | Fixed a bug which was causing none of the points of interest to by displayed due to the fact that the id of the database entries was not being passed in the call to get data from the database | Passed |
| User can see the weather for their current location | Failed | Fixed a bug which was causing the weather to be displayed multiple times due to the fact that the area where the weather data is displayed was not being cleared before adding the weather data to the DOM | Passed |

## P18 – Testing Example Screenshots

Example of test code:



Test failing to pass:



Test code with errors corrected:



Tests passing:

