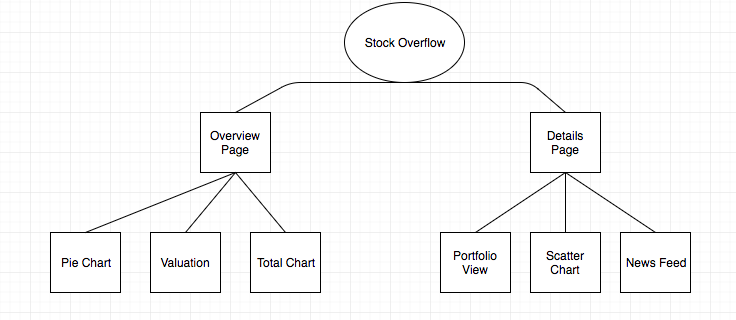
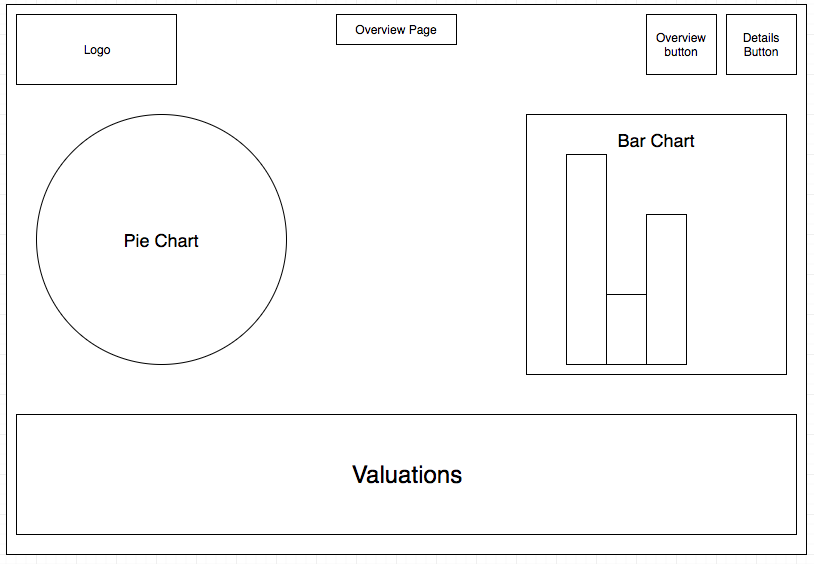
**Ben Ghirardani PDA**

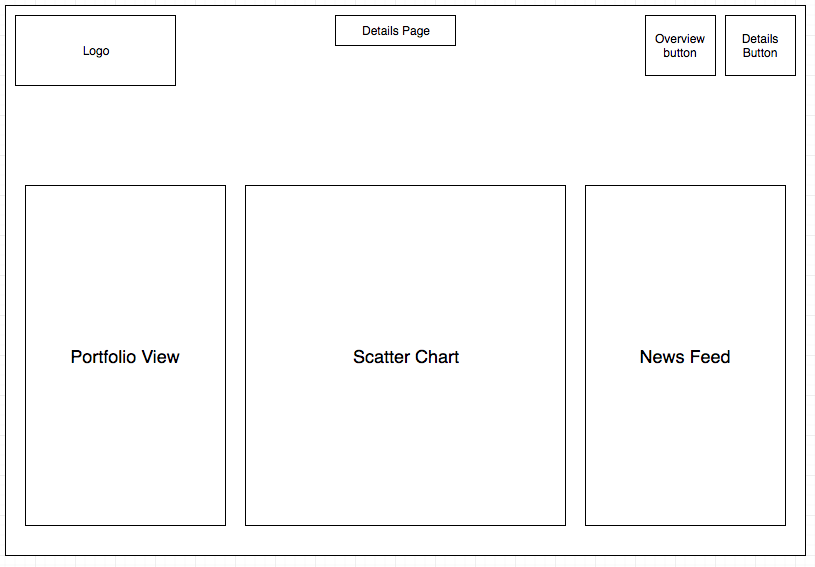
**Project Unit**

P 5 - Create a user sitemap

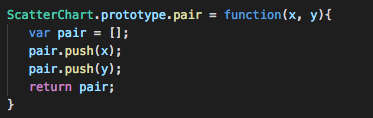


P 6 - Produce two wireframe designs





P 10 - Take a screenshot of an example of pseudocode for a function



Pseudocode for the above:

**function** pair (x, y)

create new array

push x to the end of the array

push y to the end of the array

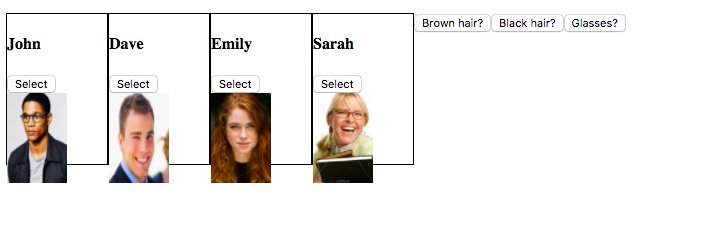
return the array

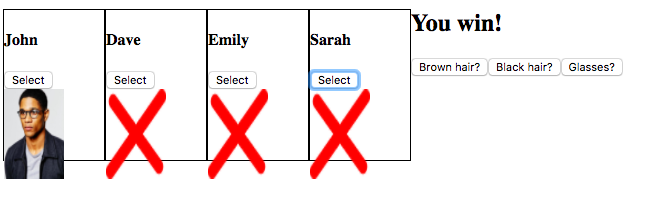
P 13 - Show user input being processed according to design requirements.

The user inputting something into the program:

The user input being saved in some way:

The user has selected 3 buttons, the data is saved by crossing out each entry. Finally, when only one option remains, the program will check whether remaining selection matches the winning selection or not and return either “You win!” or “You lose!”.

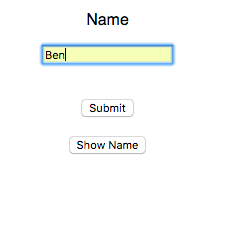




P 14 - Show an interaction with data persistence

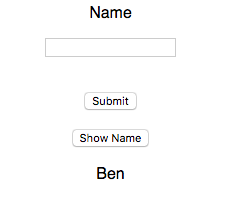
Data being inputted into your program:

A form takes in an input and the ‘Submit’ button saves it to state in React.



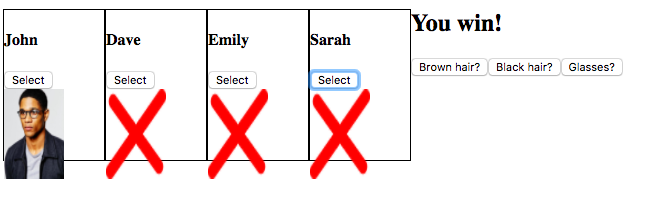
Confirmation of the data being saved:

The ‘Show Name’ button goes to this.state and displays what has been saved there.



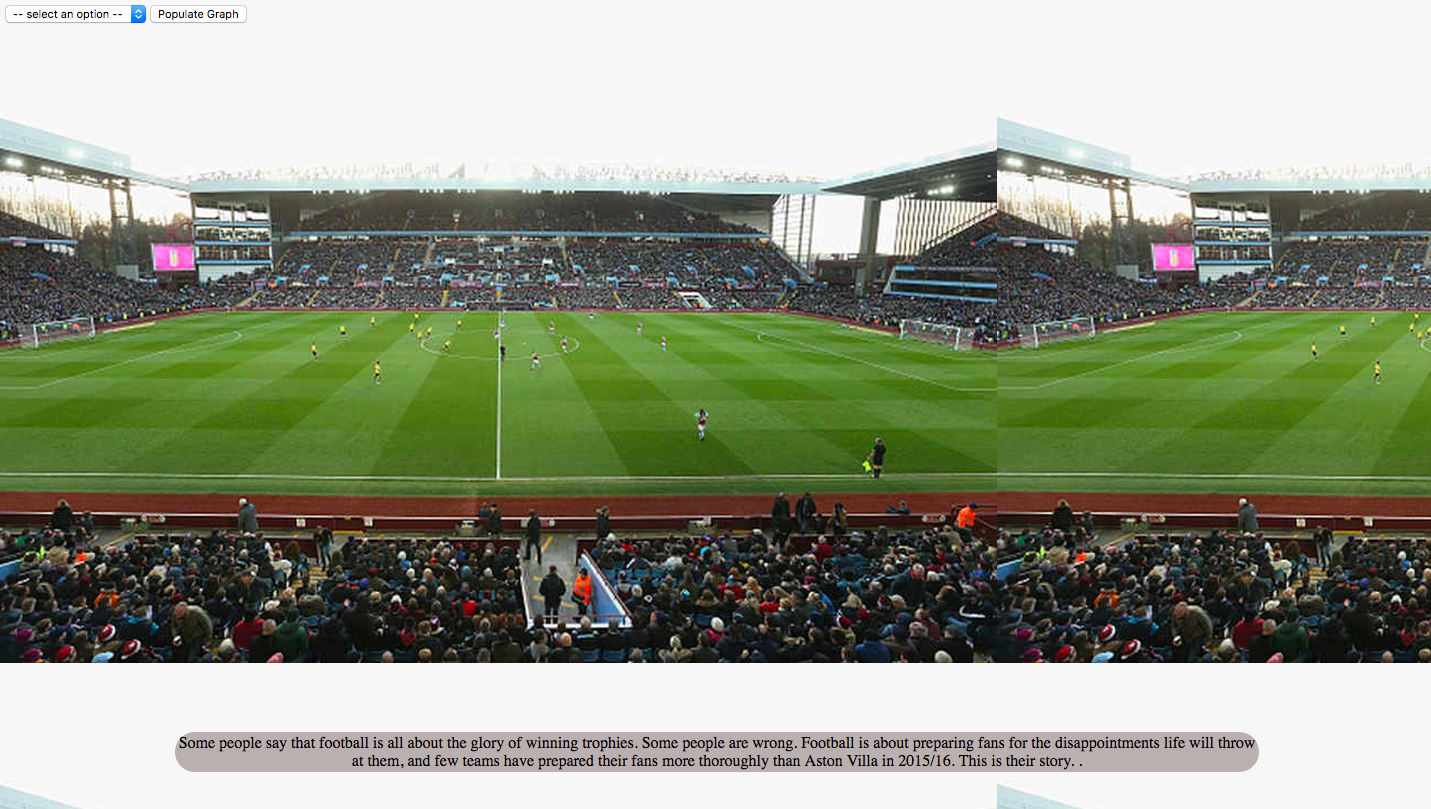
P 15 - Show the correct output of results and feedback to the user

The user selects a box and the action performed is that the picture is replaced with an ‘X’.

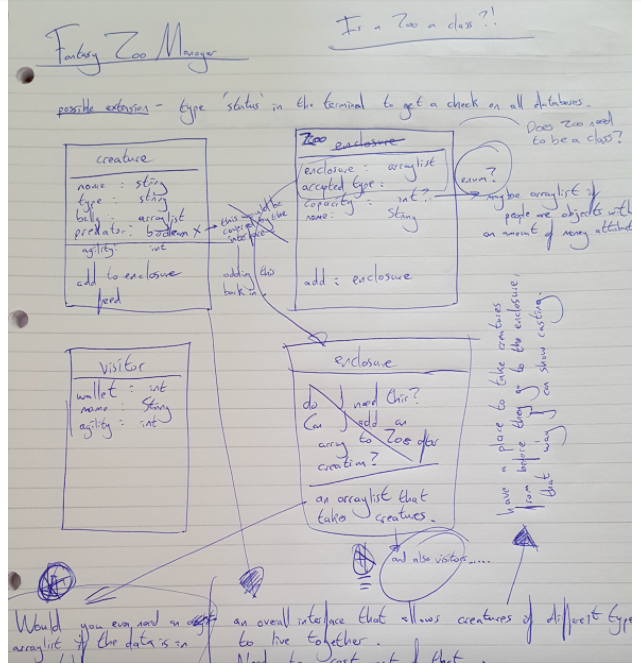


P 11 - Take a screenshot of one of your projects where you have worked alone and attach the Github link

https://github.com/ben-ghirardani/football\_api



P 12 - Take screenshots or photos of your planning and the different stages of development to show changes

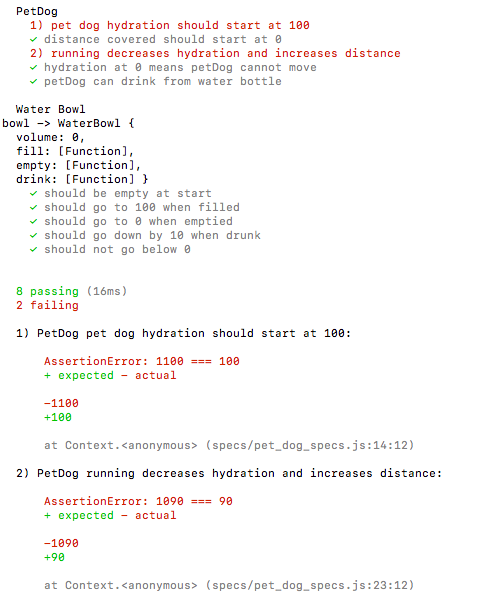


P 18 - Demonstrate testing in your program. Take screenshots of:

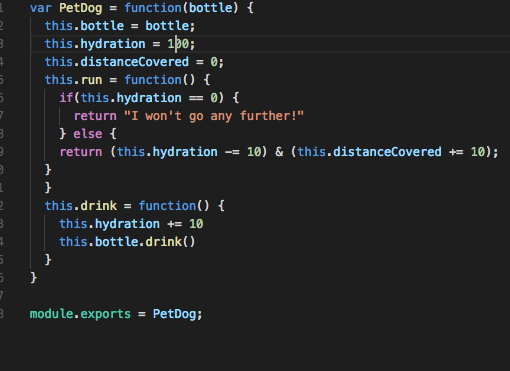
Example of test code:



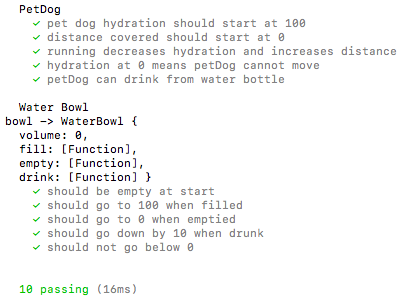
The test code failing to pass:



Example of test code once errors have been corrected:



The test code passing:



P 16 Show an API being used within your program. Take a screenshot of:

The code that uses or implements the API:

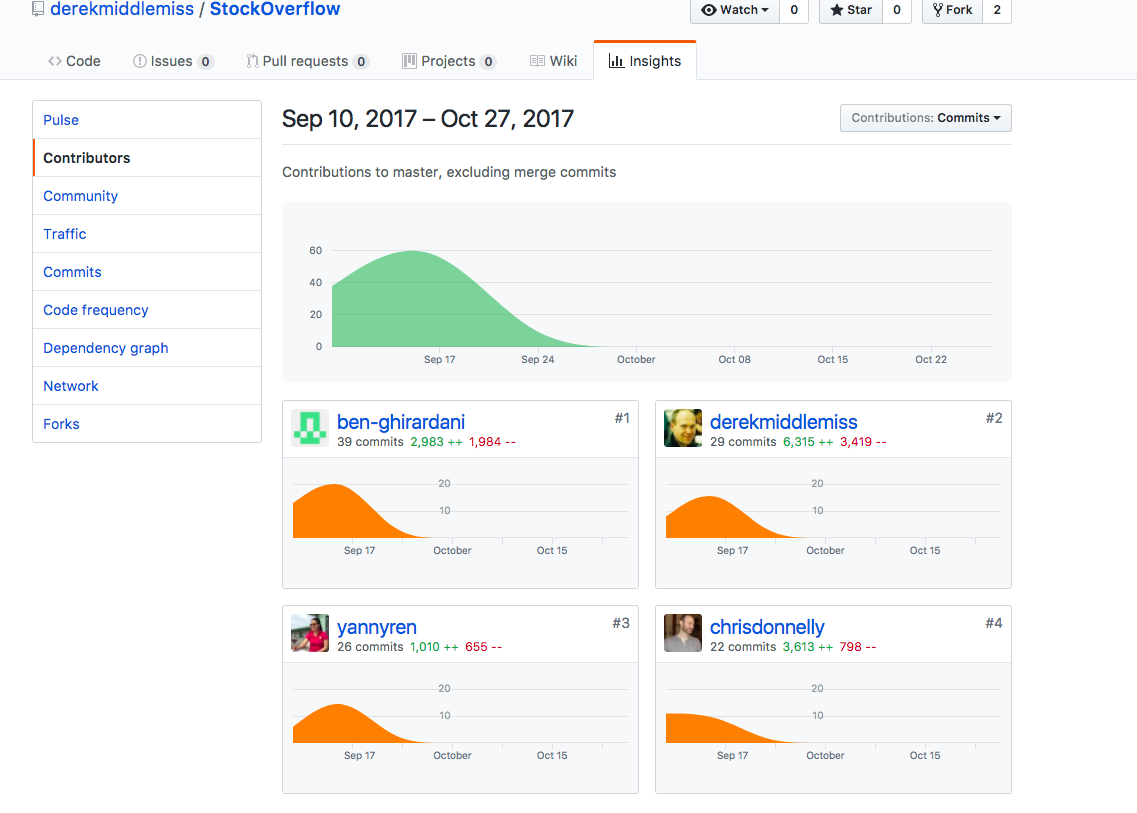


The API being used by the program:

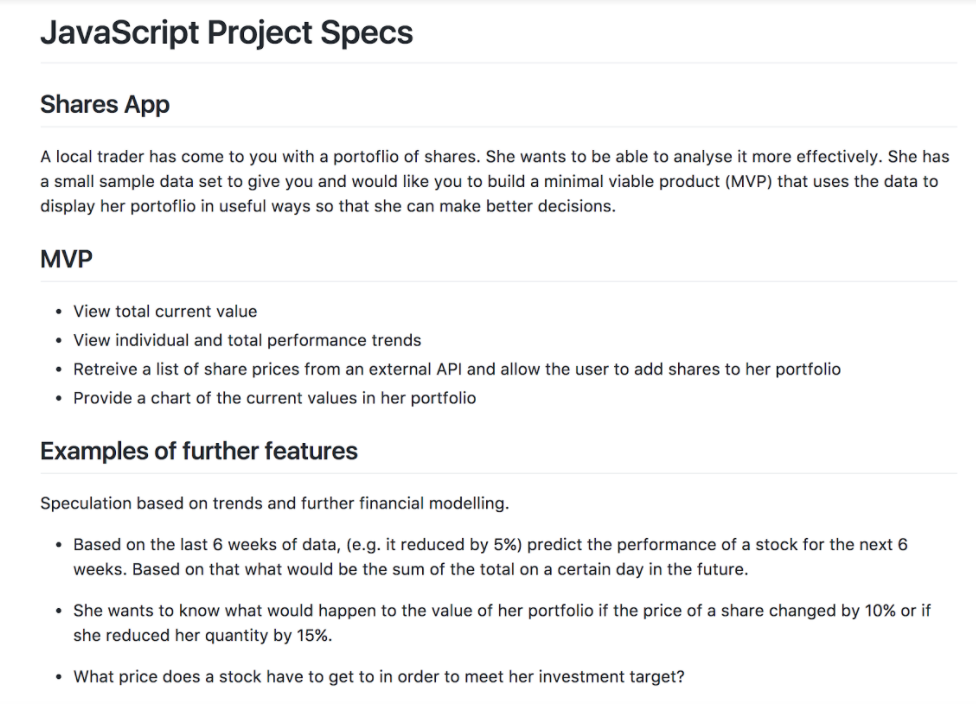
(originally my API would produce a list of fixtures in the drop-down below, but I’m getting an error 403 access denied, I’ve tried a new API key and still can’t get access)



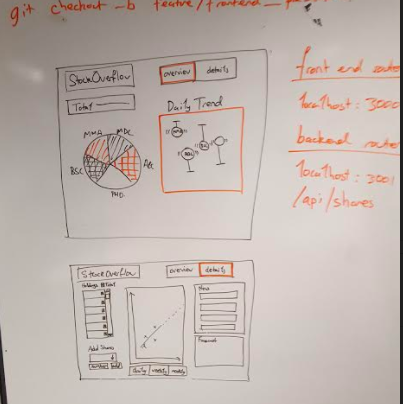
P 1 - Take a screenshot of the contributor’s page on Github from your group project to show the team you worked with

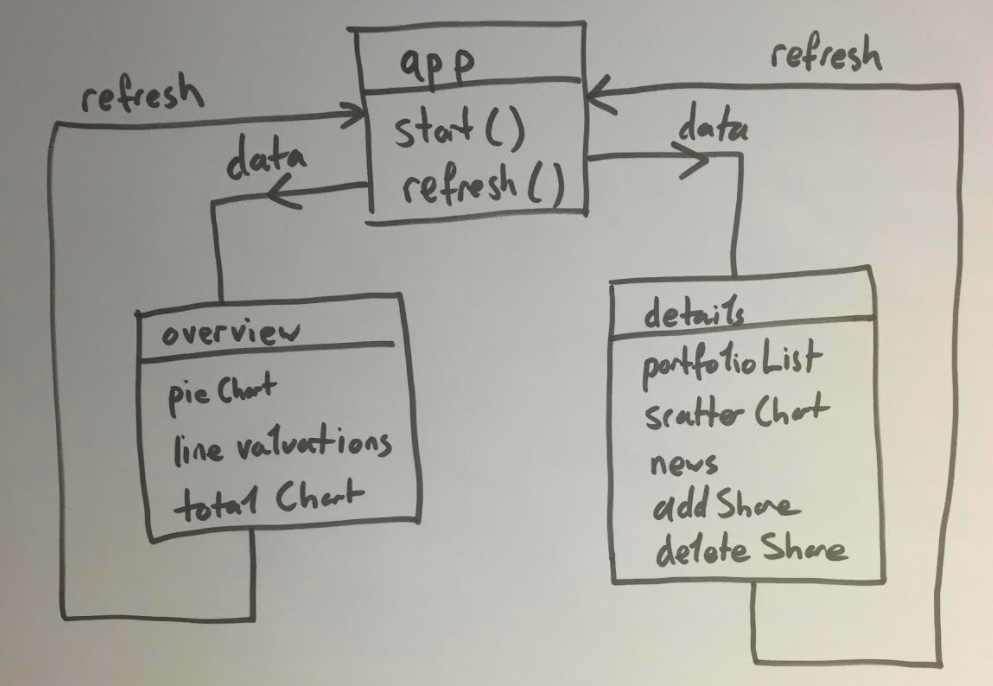


P 2 - Take a screenshot of the project brief from your group project



P 3 - Provide a screenshot of the planning you completed during your group project



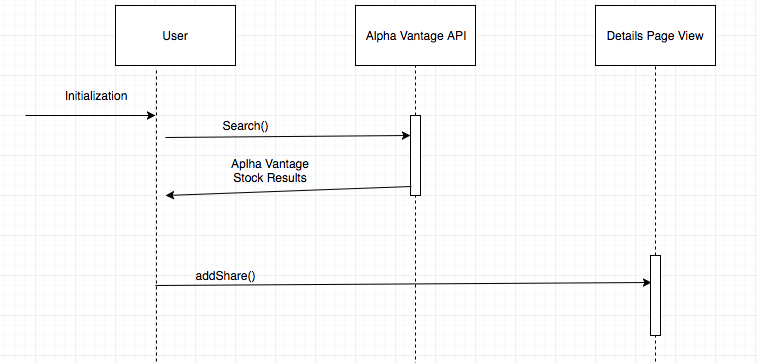


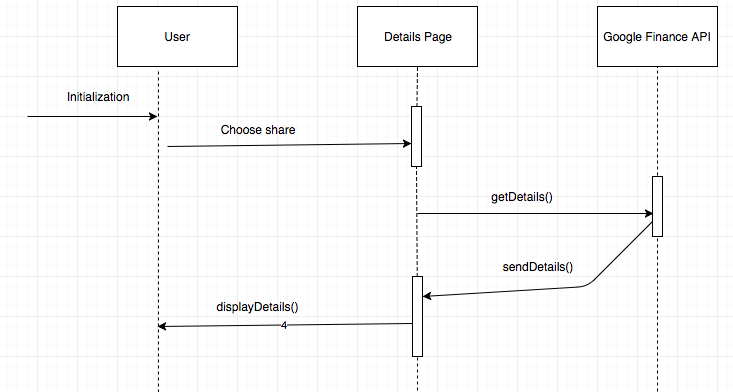
P 4 - Write an acceptance criteria and test plan

Plan for Stock Overflow group project

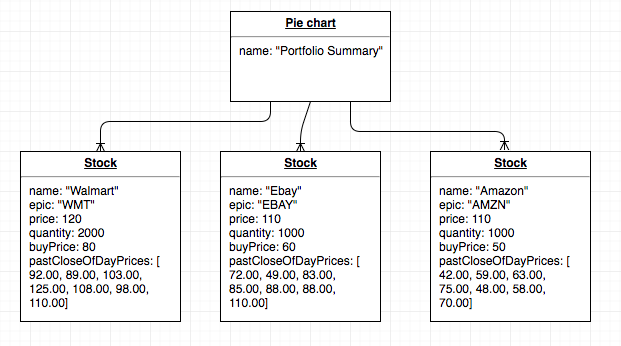
|  |  |  |
| --- | --- | --- |
| Acceptance Criteria | Expected Result/Output | Pass/Fail |
| A user is able to view their stocks and how they are performing. | Navigate to the details page and view the chart and data. | Pass. |
| A user is able to add new stocks to their portfolio. | On details page, search for a stock and add. | Pass. |
| A user can see recent financial news for the stock they have selected. | On the details page, highlight a stock, news feed view should automatically update with articles. | Pass. |

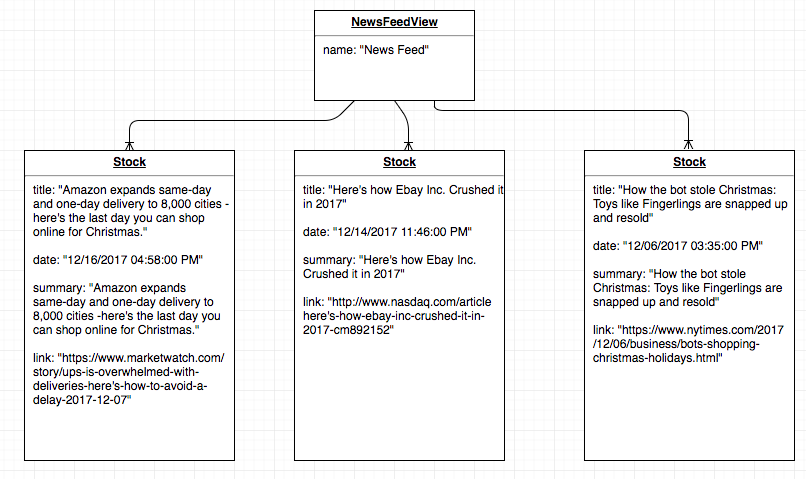
P 7 - Produce 2 system interaction diagrams (sequence and/or collaboration diagrams)





P 8 - Produce two object diagrams



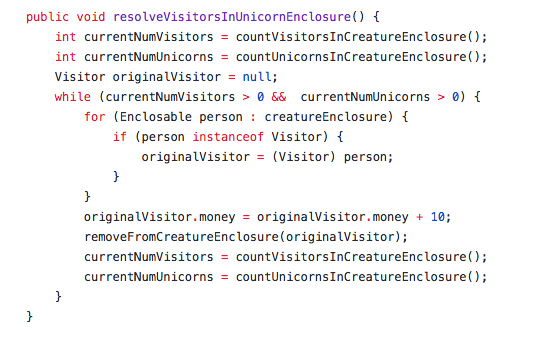


P 9 Select two algorithms you have written (NOT from the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms

This algorithm was included in my fantasy zoo keeper project. It compares the agility variable of dragons and visitors and either allows visitors to leave or get eaten depending on who’s agility is highest.



This algorithm was also included in my fantasy zoo keeper project. It loops through an arraylist, and if there are both unicorns and visitors in the arraylist, the visitors are provided with funds and removed from the arraylist.



P 17 Produce a bug tracking

|  |  |  |  |
| --- | --- | --- | --- |
| NewsFeed items are not populating in the NewsFeedView. | Failed | Added an appendChild() method to the render method to bring the data in. | Passed |
| NewsFeedView gave an error when there were more than one news options. | Failed | Created a ‘for loop’ to loop through all available news items and display to screen. | Passed |
| NewsFeedView treated companies with no current news stories as an error. | Failed | Added an option to display “no current news” to allow the function to continue. | Passed |
| NewsFeedView could not display news articles. | Failed | API was only sending data as XML. Switched to Google Finance to get JSON. | Passed |
| PieChart was returning unexpected results. | Failed | Seed data and live stock data was coming in slightly different formats, adapted seed data to match the format live data was received in. | Passed |
| Data from the Overview and Details Page was merged on one page | Failed | The code which determines which view is currently visible was corrected to switch views rather than display both at the same time. | Passed |