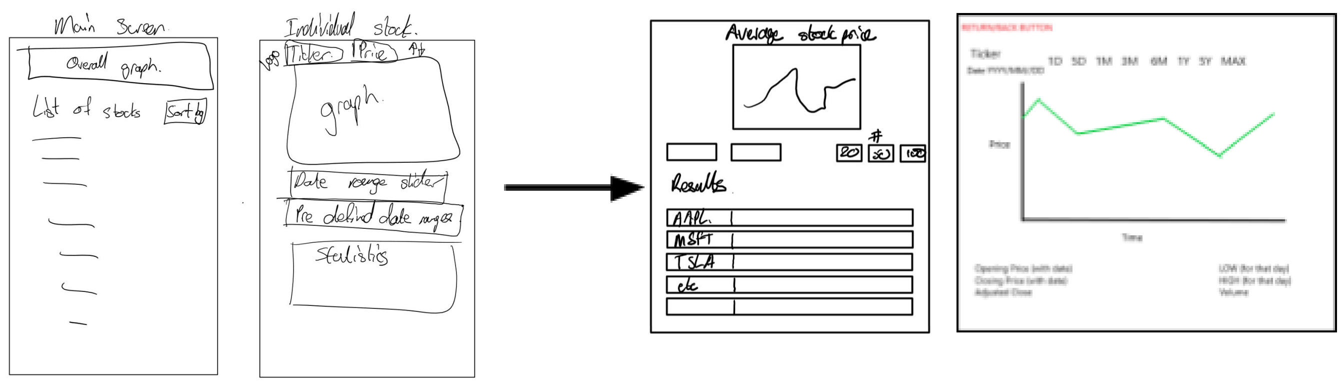
CS1013 Programming Project Report: Group 21

Team Organisation

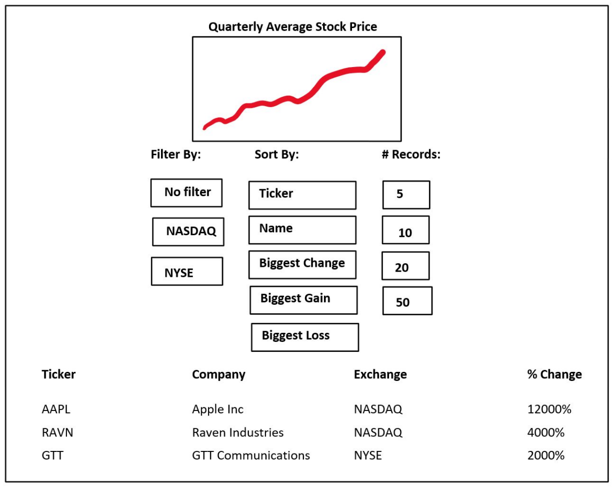
We first contacted each other via Facebook Messenger and made a group chat for our processing project. A few days after this, we had our first meeting where we sketched a few drafts. We decided to take the approach of splitting up the work on a week-by-week basis which resulted in us getting a lot of work done in a short period of time. Thanks to our team member Cian Mawhinney, we were able to write an email each week describing our list of things to do along with our distribution of the workload. This worked out really well because we knew exactly what we needed to do, and so there was no confusion.

How we drafted the program

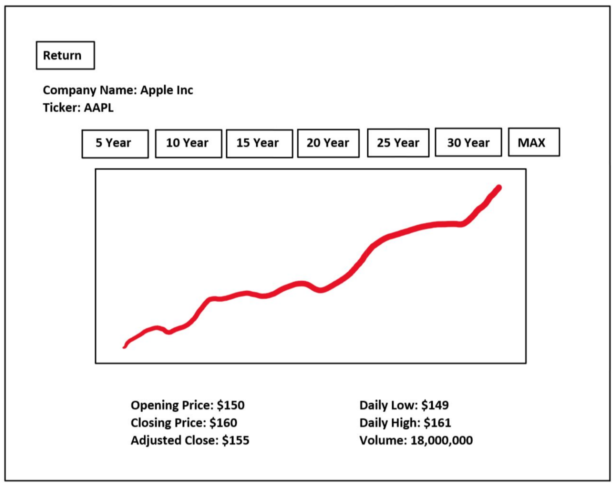
Being able to refer to drawings of our program was very useful during the project. During the first meeting we drew some quick sketches, before refining them in the lab afterwards.



Towards the end of the project, we settled on the following design:



Ticker	Company	Exchange	% Change
AAPL	Apple Inc	NASDAQ	12000%
RAVN	Raven Industries	NASDAQ	4000%
GTT	GTT Communications	NYSE	2000%



Company Name: Apple Inc
Ticker: AAPL

Opening Price: $150
Closing Price: $160
Adjusted Close: $155

Daily Low: $149
Daily High: $161
Volume: 18,000,000

Group Members:

Cian Mawhinney

I was responsible for **organising the group**, and ensuring that each deadline was met. As part of this, I wrote an email describing the tasks to be completed by each person during the week. During the labs we would look at the next goal together, and decide as a team how it should be implemented. An example of this was how we decided that we would **create our own UI kit**, and use that to create the visual elements of our program. We also decided that it would look better and be faster to utilise a graphing library to draw our graphs and wrap our own code around it so that it could be integrated into our own codebase effectively.

Another of my responsibilities was **designing how our UI kit would work** and implementing parts of it, such as the screen class, the scrollbars, and an abstract widget that could be extended to create specific widgets (ie. buttons, text and graphs).

Matthew Dowse

I made the DailyPrice class. This class was created to get each specific company's opening price, closing price, adjusted closing price, lowest price, highest price and total volume of shares traded that day. As well as the date in the YYYY-MM-DD format. This class holds a specific price record from the daily_pricesXX.csv file.

I also made the Graph class using the Grafica library. This class was created to visualise the data using a line graph from my initial sketch. This class is a wrapper class for the grafica library, so that it can be stored as a Widget.

In the Company class I created a function that takes in two dates, and returns an array with all the daily stock prices between the two dates.

Inside the Main I built the second screen which entailed bringing all the other classes together like the Button class for the interactive buttons, the Text class for the displaying of text, the Screen class to actually display a screen, the Company class to get each company, the DailyPrice class to get each price and of course the Graph class to display the graph.
For the interactive buttons I extended the switch statement in the main for the second screen so that the graph would change for each selected time frame.

One problem I ran into was that the code I had initially made to display the graph was designed to take in more than 1000 entries. I had designed it to display many points so that the user could see the change in one day, one week, one month, three months, six months and one year. Instead I had to change the functionality of my code so that the graph would display more effectively by changing the time frame to one year, five year, ten year and so on until it reached max points. The code can be quickly altered to a more convenient time frame depending on the specific data entry.

Prathamesh Sai
I made the **text class** with Subrahmanyam to output text onto the screen. We could then make objects of this class wherever we needed text to be displayed. This was an essential idea which was used throughout the program. It was used to display various types of data and information throughout the program.

In the main, I made functions such as **getCompaniesOnStockExchange().** In this function, it filters through the array of companies and returns a new array of all the companies that match the exchange passed into the function.
This is a useful function as it allows the user to quickly differentiate between different companies on different stock exchanges.

Also, I constructed the function **filterCompanies()**. This was also an essential function in our program which filters through companies and returns the companies which fit the query that matched in the switch statement; for example, you could filter companies by nothing or by stock exchange. This was an essential part of the user interface so that the user can differentiate between different queries.

Likewise, I developed the function **getCompanyByTicker().** This was a simple yet handy function to return the company that matches the ticker passed into the function and it proved to be useful in the setup().

Similarly, throughout the duration of our time working on the program, I tried to fix little things such as **cleaning up the user interface** by adjusting the positions of the query boxes and the text on both the first and second screen to make the program look clean and aesthetically pleasing. Little additions such as this proved to be very helpful for our overall program.

Subrahmanyam Rayanapati
I initially added the code to read and store the data from the CSV file in **setup()** by using **loadStrings()**. Later we used BufferedReader to read the files. Sai and I created the **text class** which helps display the text. Then he helped me create objects of Text class to print the data on the screen. I worked on the function **sortCompanies()** which sorts and returns the companies according to user entry. I also worked on the function **addCompanies()** which adds the filtered/ sorted companies to the home screen using the objects of text class.

What we think our program does well:

Code

- We designed our own UI kit, making use of only one external library.
- Clear naming of method and variable names
- Code is thoroughly commented

Runnable Program

- Straightforward and easy-to-use
- Elements are unambiguous
- No confusion involved in the interface
- Simple things such as the text, scrollbar, widgets, and the graphs are presented really well