Computer Science 2021

AlphaVue

by Hamish Anderson

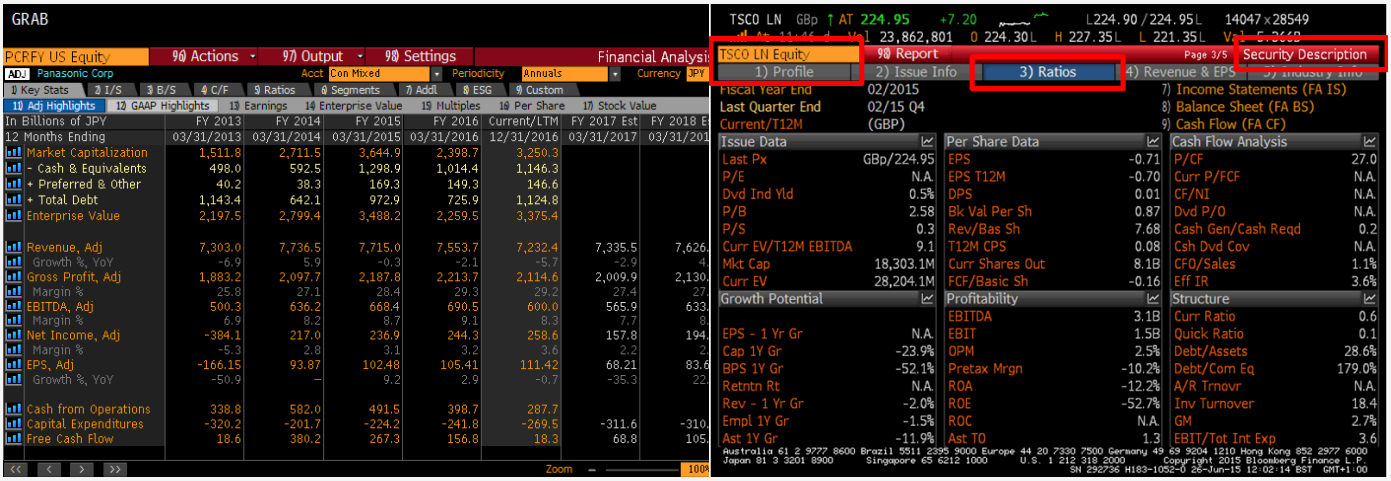
Table of Contents

Project brief

To design and program a securities analysis tool. A program that obtains information on a specific stock the user enters by the ticker code, and presents information such as what the company does, their financials, etc... in a well presented, easy to read, modern dashboard.

I am going to do this by using:

- IEXcloud API
- Python
- Visual Studio Community 2019 / PyCharm 2020.3
- PyQt5 & QtDesigner



(Bloomberg terminal above)

My idea for making a stock screener comes from the unappealing and outdated look of stock screeners such as Bloomberg terminal, which is used by almost all investment banks. I aim in my project to make this but provide a more modern look, and with far less functionality as time restrictions permit.

Algorithm

Initially:

- Display startup window.
- Display AlphaVue logo.
- Display text "Testing API connection…"
- Attempt to connect to API.
- If (API connection) == false
    - Display error message "an error occurred, check API key or Internet connection."
- else display "API connection successful."
    - pause for two seconds.
    - Program switch to terminal window.

When program switches to terminal window:

- Display terminal window.
- Display background
- Display logo, slogan, and version number in top left corner.
- Create and display text field "tickerinputTF".
- Create and display button "enter".
- Create and display button "help".

When user presses enter on keyboard or enter button on terminal window:

- Parse value of tickerinputTF to string and store in string variable tickercode.
- Try: Retrieve company overview using API call with tickercode string as args, store data to tickoverview string

  Catch: display error message "Error, cannot connect to API, check internet connection."

- If tickoverview = {}
    - Display "Error, invalid ticker code entered".
- Else
    - Program switch to stock overview window.

When program switches to stock overview window:

- Create and display label ticker code in top left corner
- Create and display button "close" below ticker label.
- Create and display button "help" below close button.
- Create and display label description along top of window.
    - Populate label description with tickoverview string.

- Fetch balance sheet data from API
    - Create label balance sheet and display following data from API using labels:
        - Current assets
        - Current liabilities
        - Non current assets

        - Non current liabilities
        - Owners equity

- Fetch Income statement data from AlphaVantage API
    - Create label Income statement "current year" and display following data from API using labels
        - Total revenue
        - Cost of revenue
        - Gross profit
        - Interest income
        - Selling general and admin
        - research and dev
            - subtotal operating expenses
        - pretax income
        - net income
    - repeat process for year before. (current year – 1)

- Fetch company overview data from API
    - Create label description and display company description from API using labels.
    - Create and display data associated with the labels:
        - Company name
        - Exchange
        - Country
        - Industry
        - CEO
        - Employees
- Fetch company logo from API
    - Display logo by creating and displaying graphic view.
- Fetch historical prices for stock dating back to past 5 years.
    - Display chart in window using matplotlib

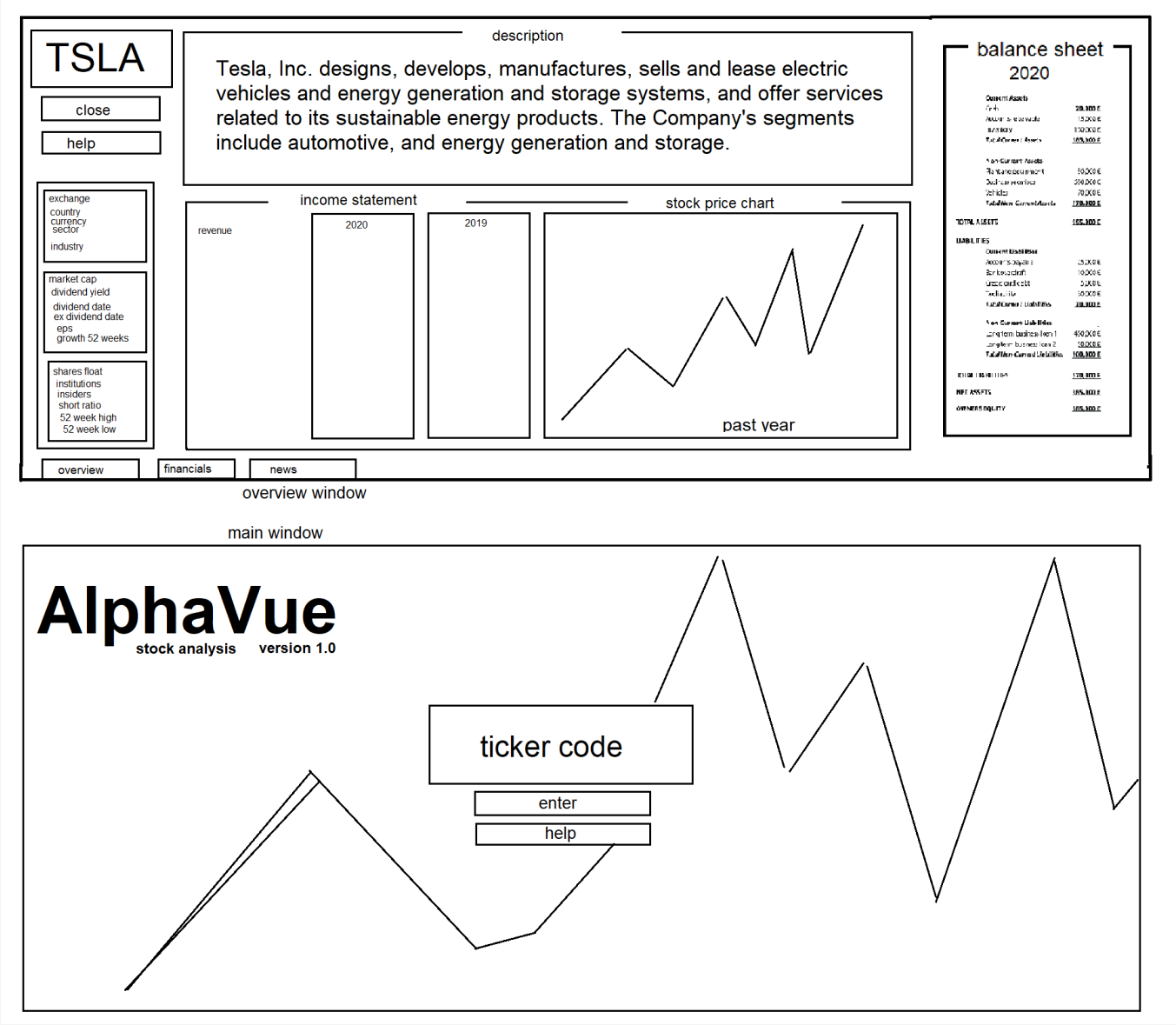
When user presses close button on stock overview window:

- Close stock overview window.
- Refocus onto terminal window.

When user presses help button:

- Display message box "This is the stock overview window, this shows the general overview and financials of the stock you have entered."

Project design



I intend to go with a simple design approach for this project, displaying only the necessary information needed with no distractions.

Project Design

The design of this program is intended to be simple with all the useful information needed to you presented infront of you. Such as the income figures and balance sheet figures, a key part of information used in evaluating whether or not to invest in a stock.The key libraries/classes I will be using in this program are PyEX and PyQT5

PyEX is a library that makes it simple to retrieve information from the IEXcloud API in a single command, otherwise I would have to write out a lot of code in order to do this.

PyQT5 is a GUI development framework for python, that when used in conjunction with QT Designer makes it very simple to develop GUI applications, by using a drag and drop approach and then it does all the hard work for you in converting that layout to code.

The analysis window for this project has over 3000 lines of code in its GUI file, which I am glad I didn't have to code it by hand.

PyEX : https://pyex.readthedocs.io/
PyQT5 : https://doc.qt.io/qtforpython/

Project design

Methods used:

- millify()
  - Make big numbers more human readable, i.e 36980000 is turned into 36.98 Million

- search_btn_clicked()
  - Retrieve all necessary data from the API and display it.

- help_btn_clicked()
  - Display a dialogue box showing information on how to use the program

- help_btn_clicked_analysis()
  - Display a dialogue showing information on what it is and how to use it.

- on_exit_btn_clicked_analysis()
  - Close the analysis window.

- on_exit_btn_clicked()
  - Close and exit the program

- loadingFunc()
  - Logic to see whether there is an internet connection or ticker code is invalid, otherwise print an error in a dialogue box.
- __init__()
  - Initiated when the program is started, goes through all the startup procedures; displaying loading screen... and switching to the search window.

Project implementation

11th August:

- Finding Financial market API's was quite difficult, as everyone wants to sell their data, free API's were dodgy and quite limited in their coverage, functionality and reliability, and other API's were just far too out of budget, with some costing over $80,000 a month.

- I initially thought I was going to use AlphaVantage but this had issues.

- AlphaVantage API financial statements were not completely standardized and were mixed between GAAP and IFRS taxonomy form, which will make development more difficult than it should be to account for these two standards used.

- Financial data on AlphaVantage was quite dodgy as figures would not add up correctly, e.g the gross profit was off by a million in the most recent data when calculating using **total revenue – cost of revenue,** although this issue wasn't apparent in previous years data.

- AlphaVantage although it was free provided a lot of functionality, and was very accurate with all of its other functions. However looking on internet forums showed many people frustrated with their inconsistent data on financials, getting vastly different figures from different, more reliable sources like Bloomberg.

- To get access to the Bloomberg API, used in the famous Bloomberg terminal. Costs over $80,000 per month so that was definitely off the list.

- I looked at FinnHub.io, it provided all of the functionality and plenty more than I needed, providing coverage for almost all financial exchanges in the world. However the free plan was severely limited and didn't let you get access to standardized financial reports. The paid plan costs $1000 per month so that was also off the list.

- IEX cloud was found to be a suitable alternative, it provides:
  - Accurate verified data by humans.
  - Standardized financial reports.
  - many python libraries.
  - Tons more functionality & features.
  - More API calls per minute.
  - Used by investment banks like JP Morgan & StoneX.

- However IEXcloud costs $9 per month for personal use, it is not too much compared to the others and I am happy to cover this for the project, as it will save me a lot of time for this project, and I will have more time to implement cooler features that the API offers.

12th August:

- Today I'm looking at how I can use the IEX cloud API and parsing the JSON data from the API calls to useable data in defined variables.

- The pyEX library returns the values of all API calls to a single pandas dataframe entry, indexed at 0. It took me a while to figure out why I couldn't retrieve a specific variable back after I tried sorting it, until I realized that everything was enclosed with [], and it wasn't all individually [], so no data existed beyond the 0 position.

- This was fixed very simply by adding [0] to the end of it, and then pandas and pyEX does all the work in retrieving the value I want in the second line.

```
df = c.incomeStatement("AAPL")[0]
print(df['currency'])
```

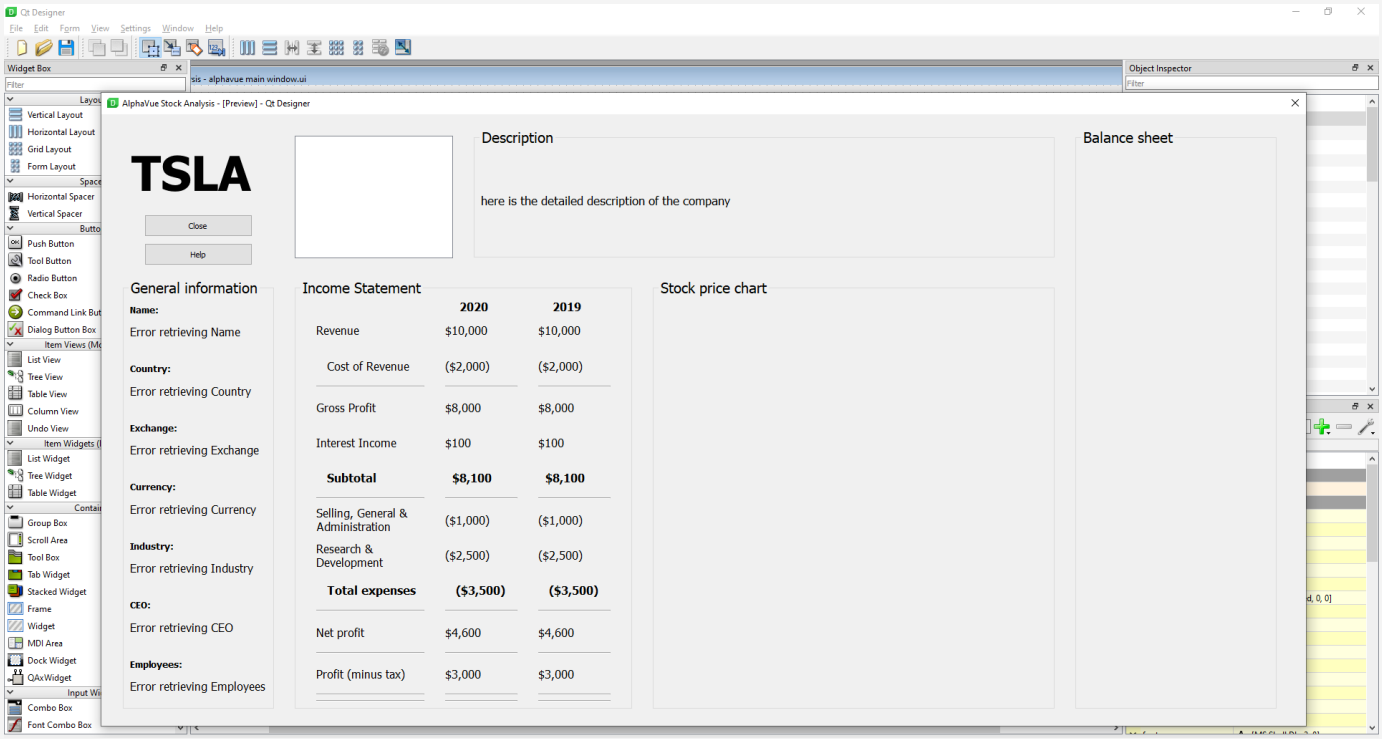
(This is how simple it is to get income data with PyEX)

17th August:

Today I finished the making of the algorithm, and project description and other necessary tasks to complete before getting onto coding. I used Canva to create this as it makes reports look really nice.

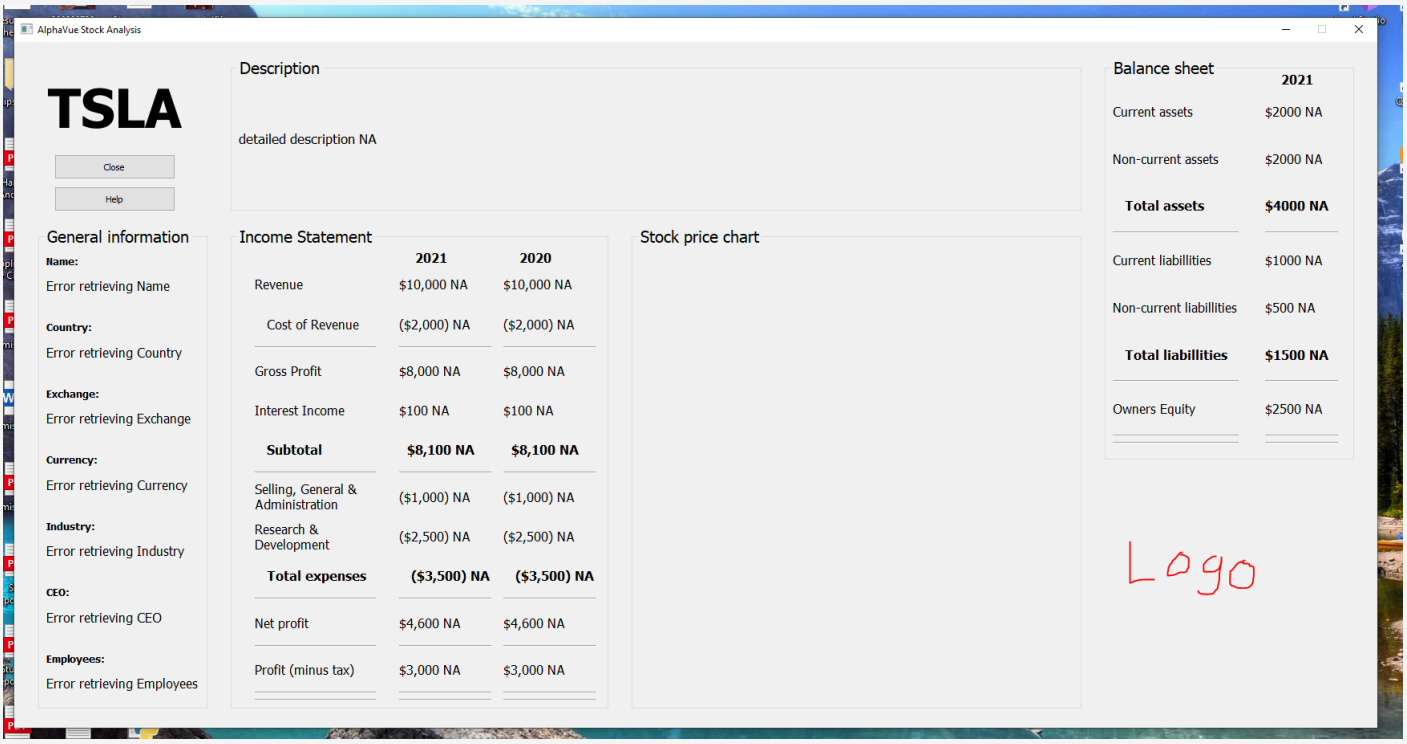
I also learned how to use GitHub with Visual Studio, which is what I'll be using to backup my code.

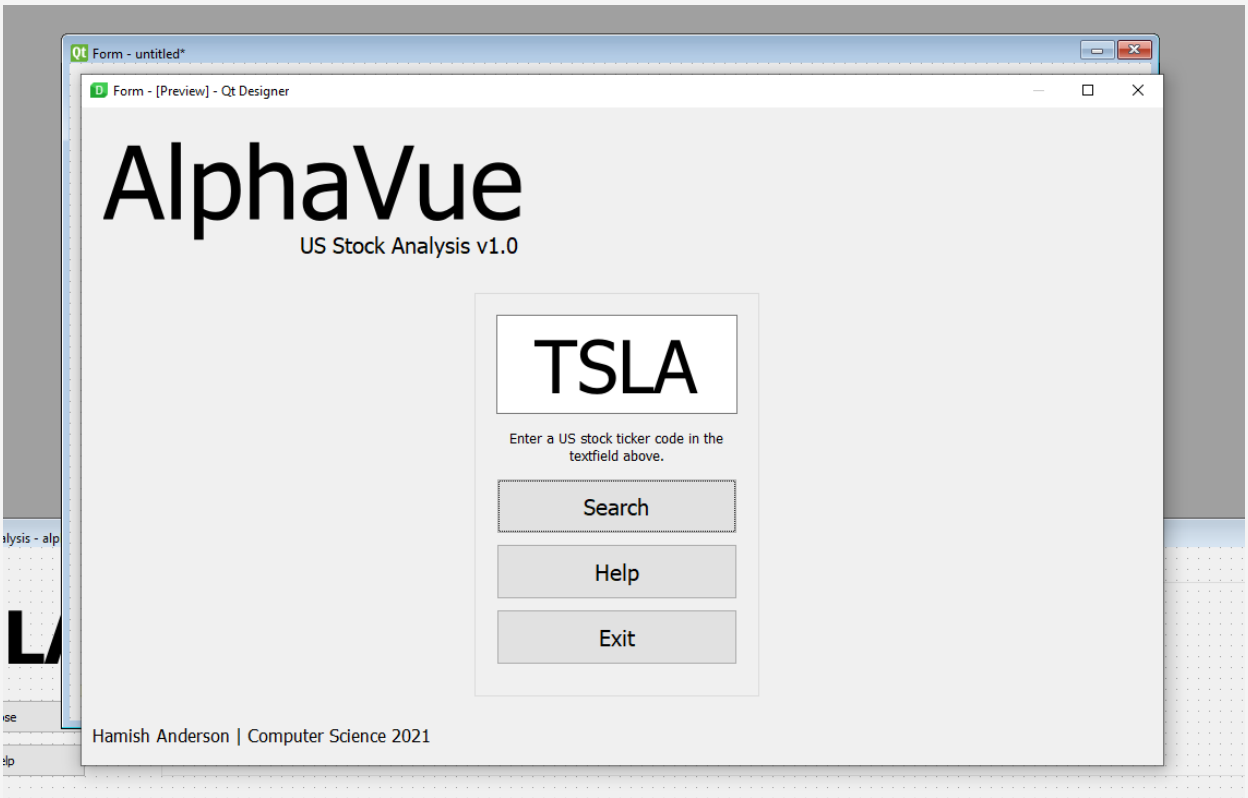
23rd August:



Today I started work on making the GUI using Qt Designer, to do this I watched a few YouTube videos to learn how to do it properly.

31st August:





Finished making the GUI's for the project v1.0

I also learned today how to implement the code QtDesigner creates.

September 1:

Today I implemented all the GUI into the same program and made the buttons usable such as the help, exit, etc...

It took a while to get the loading function to work as I thought I could use the time.delay function, but this caused the whole program and its processes to delay for a while which didn't let the loading window load.

To fix this issue I found the Qtimer function, which basically waits until the timer is up and then executes a function, this allows the user to see the loading screen and give it time for it to appear as the GUI takes a little bit to load.

September 6:

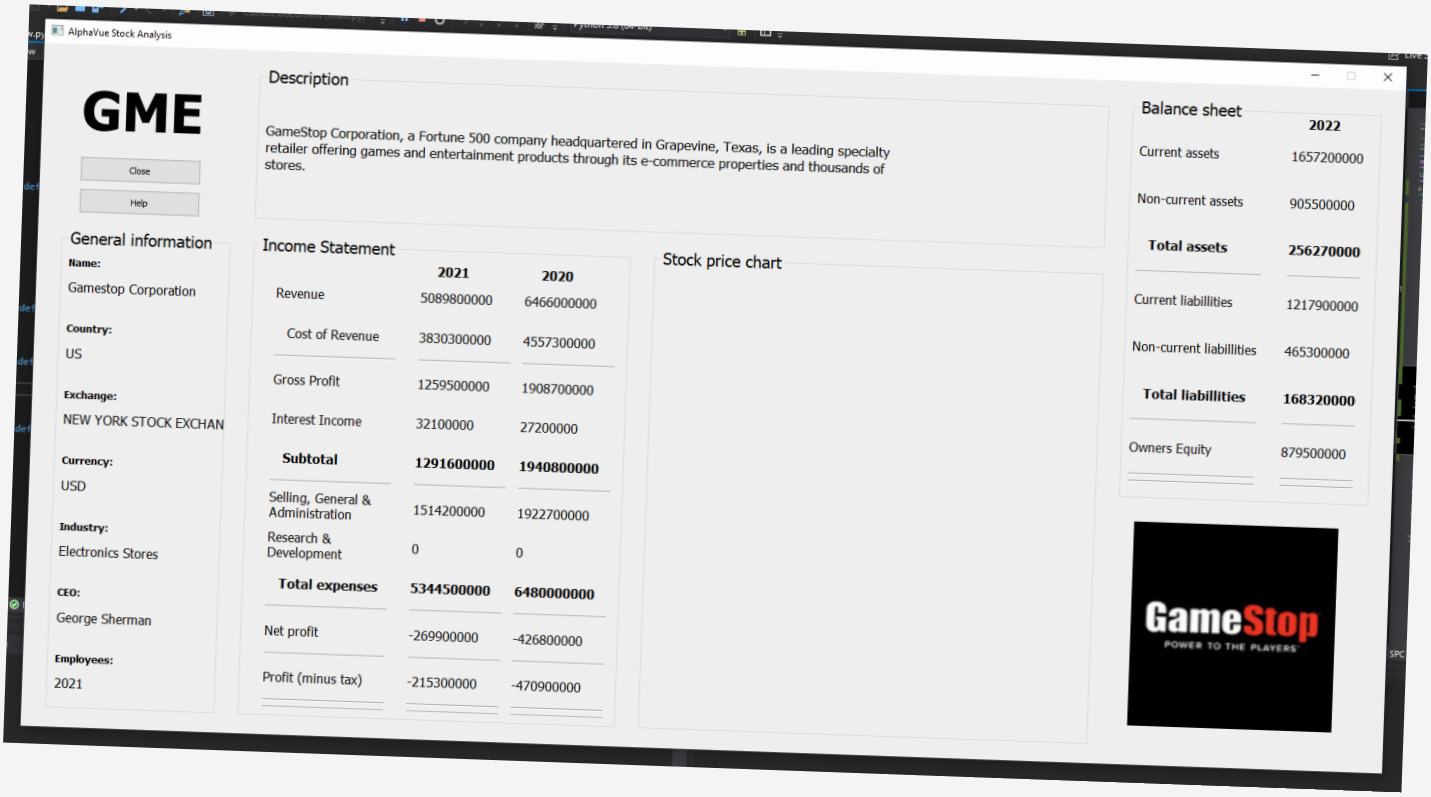
Today I started work on retrieving the data from the API's and changing the labels, it took quite a while to figure out why the company function wouldn't work however, this was as I had enclosed the command with [0] at the end, in which for some reason this command already does it for you unlike the income and balance sheet functions which is required in order to call specific data.

I also added some error catching to invalid ticker codes with a new error dialogue box shown.

September 8:

Today I put in the setText label stuff for the current year and past year income statement field ,and the balance sheet field,. Although I have found an issue in that the balance sheet heading says 2022 values? this was a simple fix as it was using quarterly data so I changed it to annual to match the Income statement.

Version 1.0 is nearly complete, just have to put in the stock price chart and make the buttons work, then I can add extra features after.



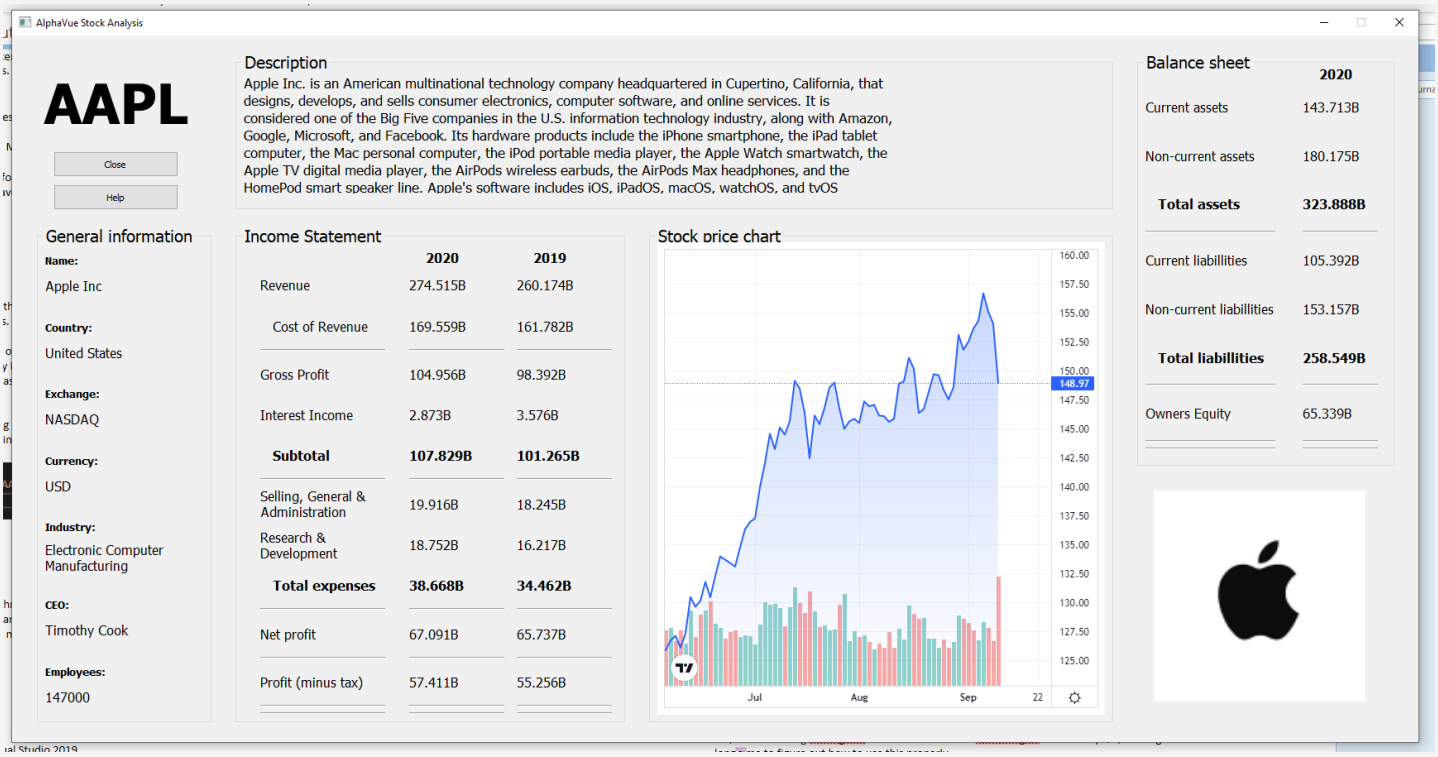
Starting to display some data...

September 11/12:

I have managed to get the stock charts to get implemented and after this that was the finish of version 1.0. I did this using TradingView HTML charts and the QWebEngine that is in PyQt5, although it took a long time to figure out how to use this properly. I learned how to do this on YouTube.

I also changed my IDE to PyCharm as Visual studio was being too slow and buggy for me.

After this I added some extra features for better readability such as millified numbers.



Displaying some usable data now. version 1.0 is finished...

September 15:

I let Noah test out my program to try and break it and he did exactly that, he searched up a company that had no financial information available in the API's database and so the program threw errors as it was trying to retrieve data to populate text labels with nothing, so I added a ton of try catch's to prevent the error from occurring and print out the error so that the user knows what happened.

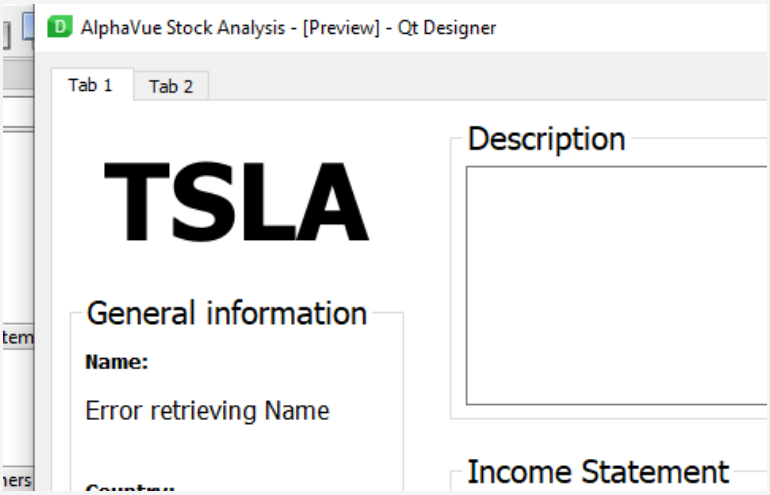
Now for Version 2.0

For version 2.0 I looked at the IEXcloud API catalogue and decided I wanted:

- Full window charts
- Insider transaction information
- Institutional Ownership information
- Analyst price predictions
- News about the stock.

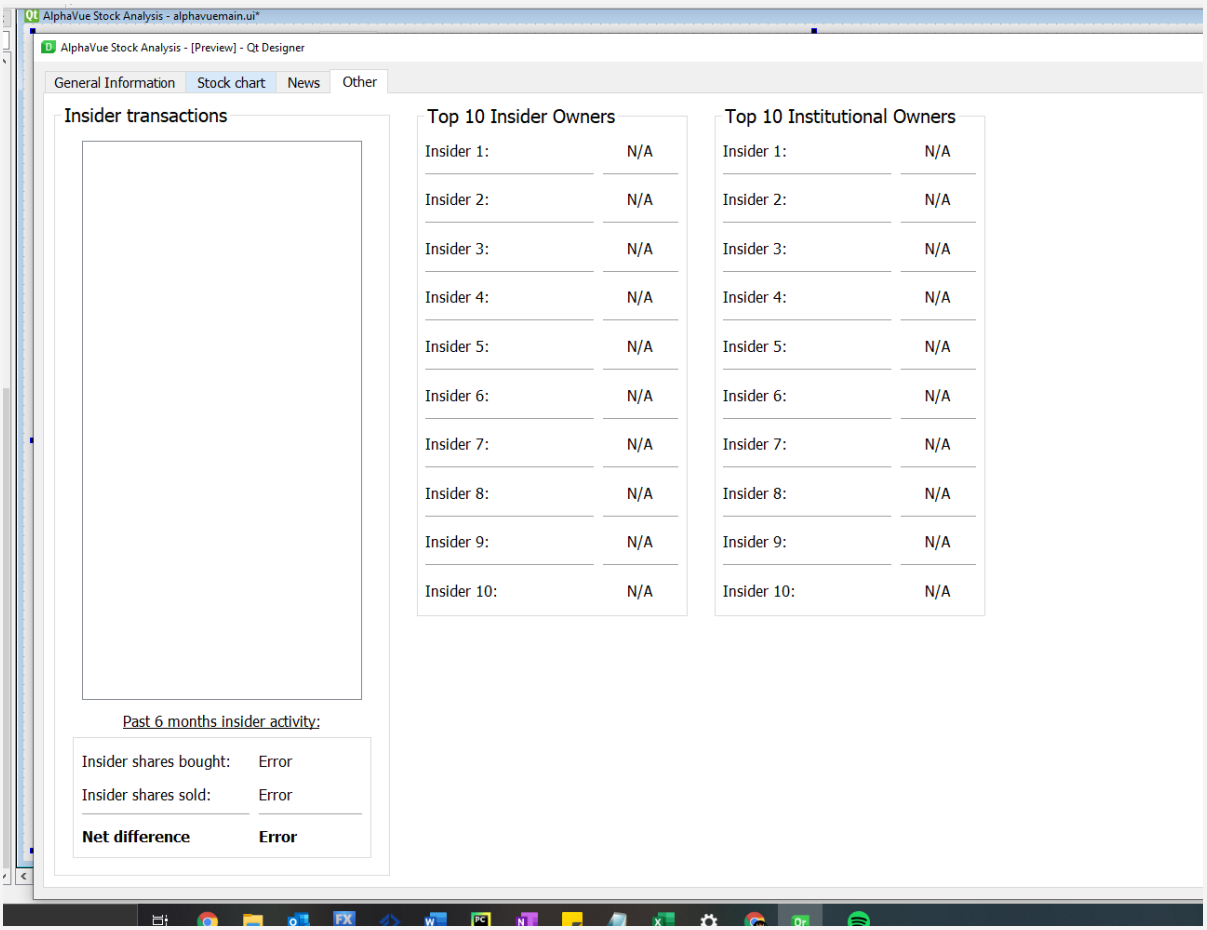
September 17:

Today I am rebuilding the GUI and adding more features, adding tabbed windows.



September 18

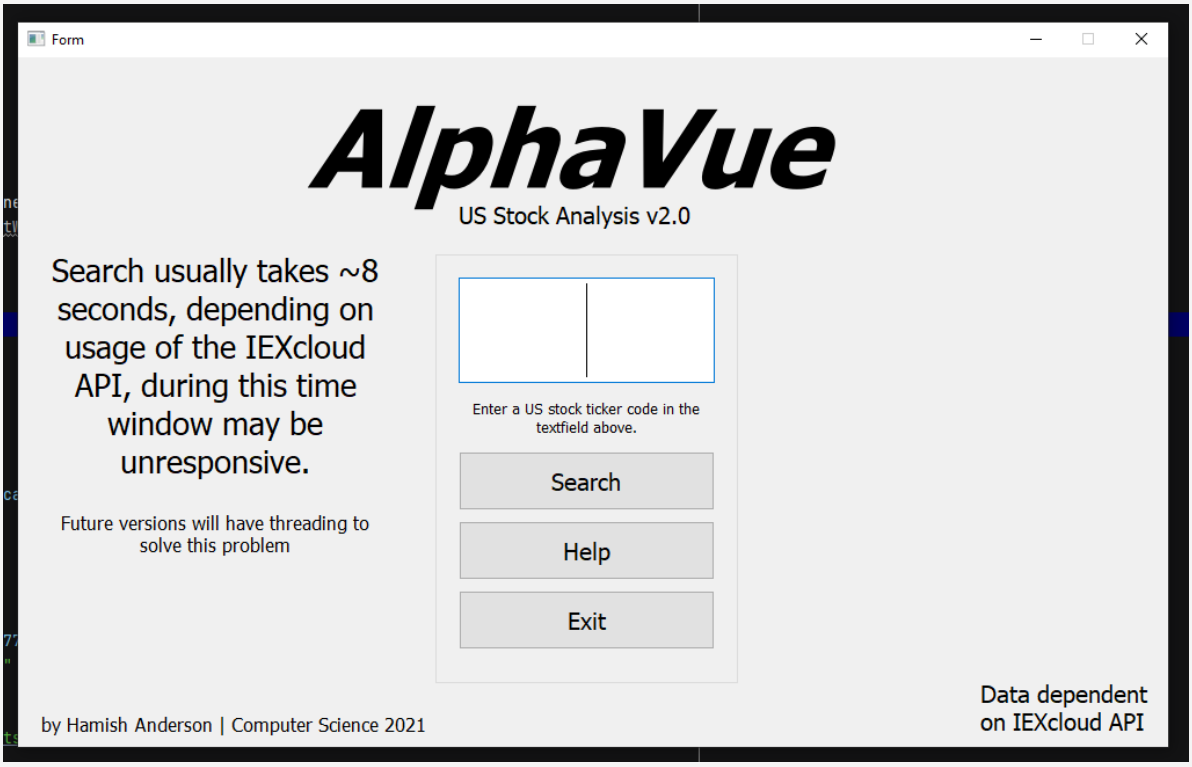
Today I'm currently making the insider transaction and other information page, and full window stock chart, all in QtDesigner.



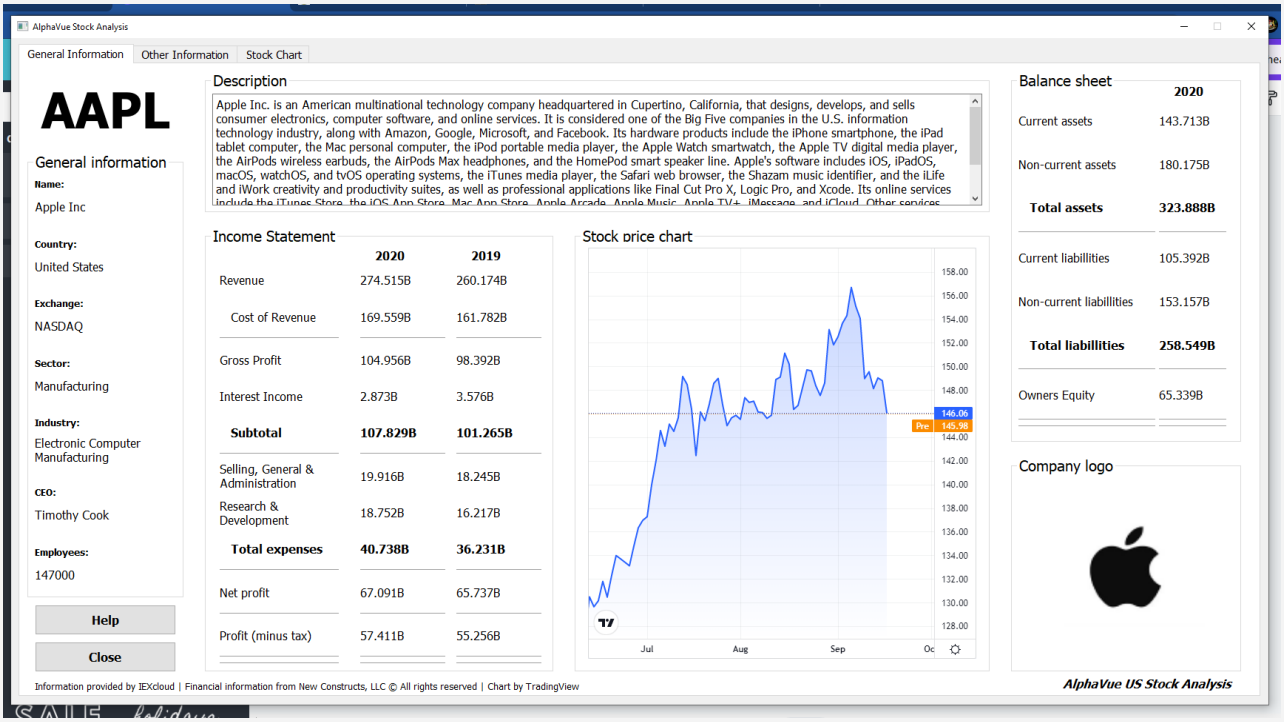
After I designed this I quickly exported the code and went to work on coding the backend of these new features, getting all the functionality up and running.

September 20:

Today I finished version 2.0 and the period of working on my project has concluded. After this I went and did a bit of work on my journal, after adding most of my comments to my code. Below are screenshots of the finished project:

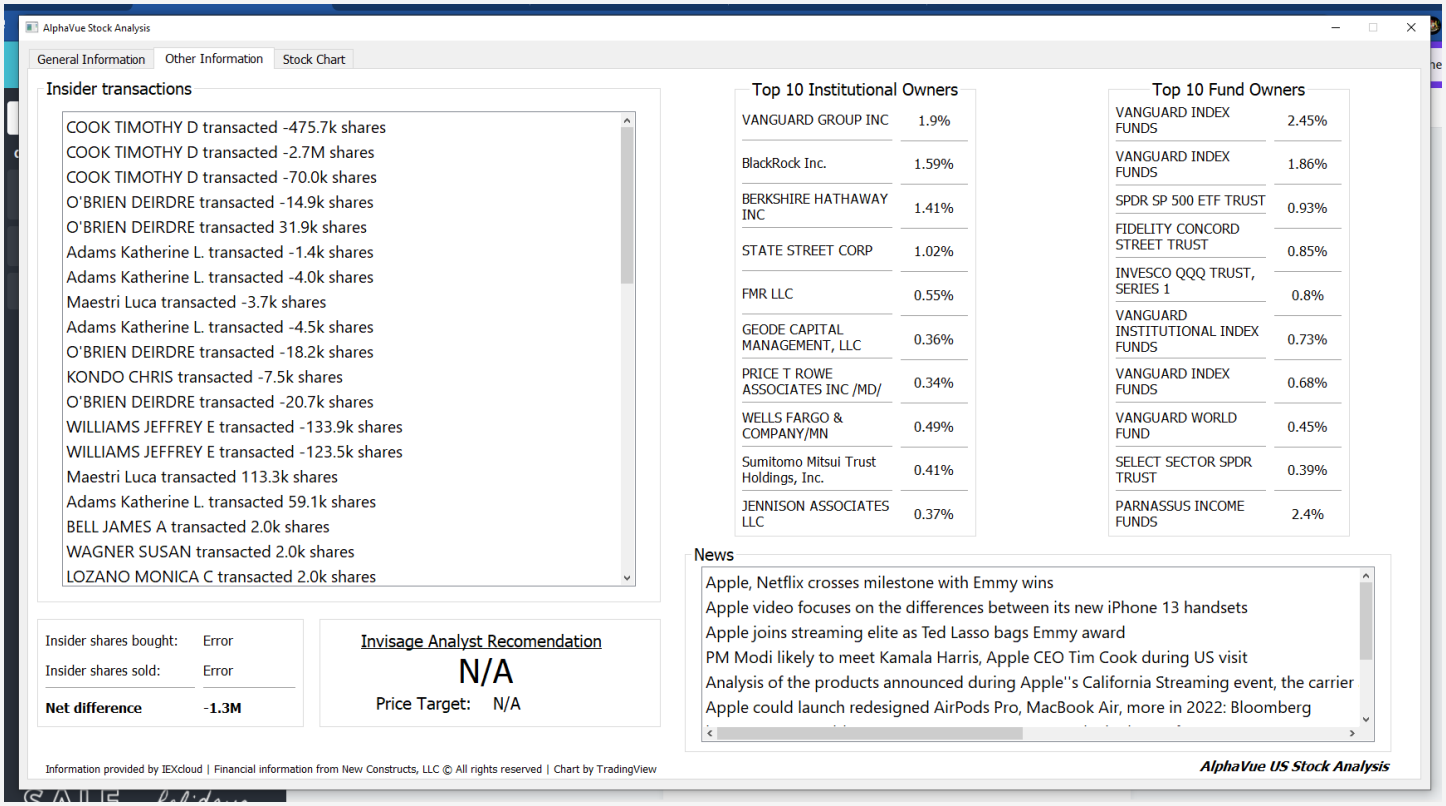


Search window



Analysis tab

Screenshots... continued



Other information tab

Full window stock chart tab

Project testing

For project testing I got Angus Foster to test out my program, he tested it out very thoroughly not knowing much and he was very satisfied with the program.

Angus tried a lot of things, entering jibberish, math operators and weird looking stock ticker codes such as BRK.A (Berkshire Hathaway Class A)

There were a few minor issues found, but these issues are easy to fix.

This included:

- Data from the previous stock searched displaying for stocks that don't have that data.
    - Need to clear all the variables after each search.

- Takes a while to load
    - Could be fixed by implementing threading in the application, this means it can run multiple processes at the same time, instead of waiting for one command to finish, then going on the next...

However after all this testing what we found is that pretty much all S&P 500 stocks will work just fine, its just when you go and search for obscure, low cap stocks that the IEX-cloud API usually doesn't have enough information to display something suitable, and so the program displays only partial information. However if this is the case most values are default set to N/A or Error.

Evaluation and reflection

This project was especially **fun** for me to do and I am **very satisfied** with the end product that I have created.

Although I didn't complete every aspect of the program such as the insider bought and insider sold totals, and double clicking on news to bring up link in web browser. I am very satisfied with how far I've come with this project.

At the start of the project I didn't really have a **clue** how I was going to make this. I knew that there were lots of tools available to do it, such as using something called an **API** (Application Programming Interface) but I didn't know how to use them, I only heard about them from watching some YouTube videos in the past. **Algorithms** and **proper planning** helped me achieve my goals for this project.

I have learned many skills throughout this project, including using Object Oriented Programming in Python, the PyQt5 GUI framework, QtDesigner, GitHub, working with API's, and of course time management.

This project **exceeds** the initial expectations I set out in the beginning, initially I thought I wouldn't be able to get past version 1.0, however I proved myself **wrong** and added many more features in version 2.0 that I didn't originally expect, such as news and insider transaction history.

Evaluation and reflection

There were a few changes made in development, this included changing my IDE to PyCharm, this was as Visual Studio Community 2019 was very buggy with Python and was slow, as soon as I switched to PyCharm everything worked a lot more faster.

Also originally I was expecting to use the AlphaVantage API, however after trying to work with it it turns out that free data is often free for a reason, it isn't very reliable and the figures are a bit hit and miss when comparing to other websites. I wanted the user to have accurate information.

To improve this project I think you could add a dark theme to the program, this would make the user experience alot more nicer and it is easier on the eyes, preventing fatigue for people who may have to use the program for extended periods of time.

Another tabbed window showing financial statements (especially balance sheet, as this program only displays the most recent) for further than 2 years and displaying the financial ratio's accompanying the data would be another cool addition too. Showing the percentage change from year to year in each category.

Evaluation and reflection cont...

The **Iterative SDLC** model I used with this project was satisfactory, however, if I were to complete another project, I would probably break the program down into even smaller steps (i.e have more versions)

This allows are more **gradual production process** to the program, which would be more useful to get user feedback quickly, instead of adding a ton of features and then releasing that version and having to make a lot changes based on feedback, lightening the workload a bit and getting to really understand what needs to be made. Although the trade off here is that you have to meet and talk a lot more to the client who you are making the program for. Luckily with this project it didn't really matter as I was more the client.

Requirements to run:

- PyQt5 library
- PyEx (5.0) library
- Requests library
- IEXcloud API Key (paid plan)
- Python 3.8

Program has only been tested on **Windows 10** systems.

All the code and history can be viewed **online** on GitHub:

- https://github.com/HamfishO/AlphaVue

AlphaVue

END PAGE

report made with canva.com

by hamish anderson