COP-290
Assignment - 1
Subtask_2

Orin Pao 2022CS11130 | Akash Meshram 2022CS11627

Part1: Stock Analysis

In this part, we have taken 3 inputs from the user

a) Stock to be analyzed:
In this, we have included the *NIFTY 50* stocks of *NSE* in the form of a dropdown menu

b) Criteria on which Stock needs to be analyzed:
In this, we have considered 3 criteria namely 'Market Price', 'Value', 'LTP'. We have considered only these parameters as these are the most important parameters involved while choosing any stock.

c) Time Scale:
In this, we have made it by adopting the idea similar to Groww. We have timescales like '1 Day', '1 Week' and so on. In this part, we have seen that when we select '1 Day' the graph is more or less linear, the reason being that jugaad-data records the data very few times in a single day.

Therefore, there is not a curve seen in the graph.
We have a maximum time scale of 5 years and a minimum of 1 day.

Here, we had to make a design decision as to use a calendar for taking input from the user or not, and we realized that selecting both the start date and end date individually and navigating through the

calendar to a date almost 5 years back is somewhat tedious, rather we have directly provided an option to choose the span over which the user want to see the data.

After all the inputs are taken from the user, and on clicking the ‘Go’ button, the page requests data from ‘jugaad-data’ and saves the corresponding data in a .csv file in the same directory where app.py is present, then this csv file is read and using ‘Plotly Express’ we have made a plot in the form of an html file in the ‘static’ folder and then using ‘iframe’ we have displayed this html file directly on our webpage.

Part2: Multiple Stocks Stats

This part was quite similar to part 1 of this subtask, the only difference was that now we had more than one stock which was to be plotted.

In this part we have again taken the list of NIFTY-FIFTY stocks as our sample space and the Criteria is also the same as the previous one, the difference is that now we have removed the ‘timescale’ section from this page.

We could have easily added it as well in this part but we felt that if someone wants to compare the stocks then he/she is most likely to look at the current trends, and so we have fixed the timescale from our side and it is roughly equal to half a year.

The overall functioning is similar to that of part 1, but now we are using a loop to extract data of every stock that is ‘checked’ and then its data is written into csv file and finally the data is clubbed together to form a combined ‘Dataframe’ which is used for plotting the final comparison graph.

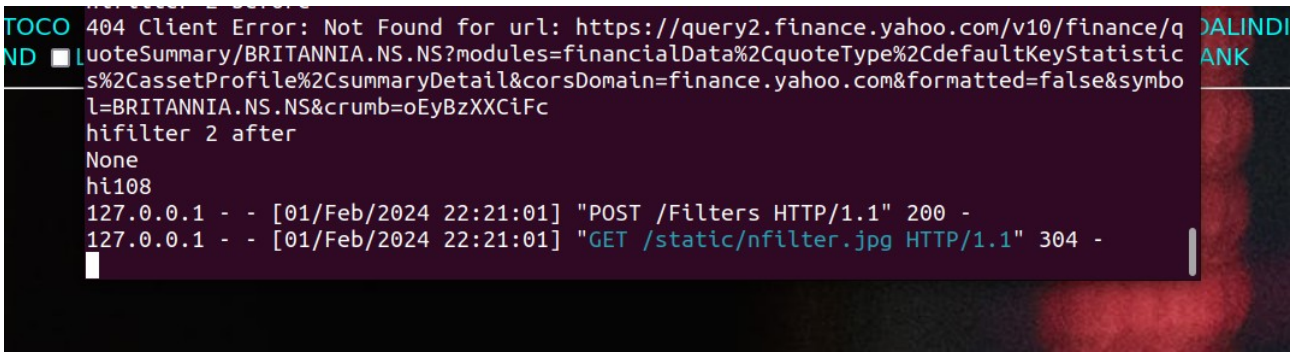
In both these parts we had initially plotted a graph similar to subtask 1 of this assignment i.e. a png file, but later we realized that png plots were not suitable for topic like Stocks where we want to know the exact values of the parameters, so we wanetd to generate an interactive graph, which was not possible with matplotlib, hence we shifted to ‘plotly’ for making the graphs interactive after some sort of research.

Part-3:

In this part I have used *Yahoo Finance* or *'yfinance'* as my primary stock library as jugaad-data did'nt contain the necessary filters which I wanted to use for the 3rd part.

As in *jugaad-data*, only *NIFTY-50* stocks were present, I have given the option of only using *NIFTY-50* stocks which are included in the yfinance library.

Note :- For *non-NIFTY* stocks, the input to the form has to be passed by suffixing " .NS " at the end of the stock symbol. Whereas, in case of NIFTY stocks suffixing " .NS " in the input of the form gives the following error.



Working and Design:

In this part 2 fieldsets have been provided to the user. Fieldset 1 contains the faminly of filteres which he/she wishes to apply. Fieldset 2 contains the list of available stocks to choose from.

I have used checkboxes to improve flexibility.
The inputs to the form are used by flask to extract information about tickers using *".info"* method. If a stock value passes the criteria of a filter value, then it is appended to the List.

The return type of *filtered_stocks* is List of Dictionaries.

All ticker don't work

- I have given freedom to the user in choosing on which stocks he would like to apply the filters on. (including a Select All checkbox)

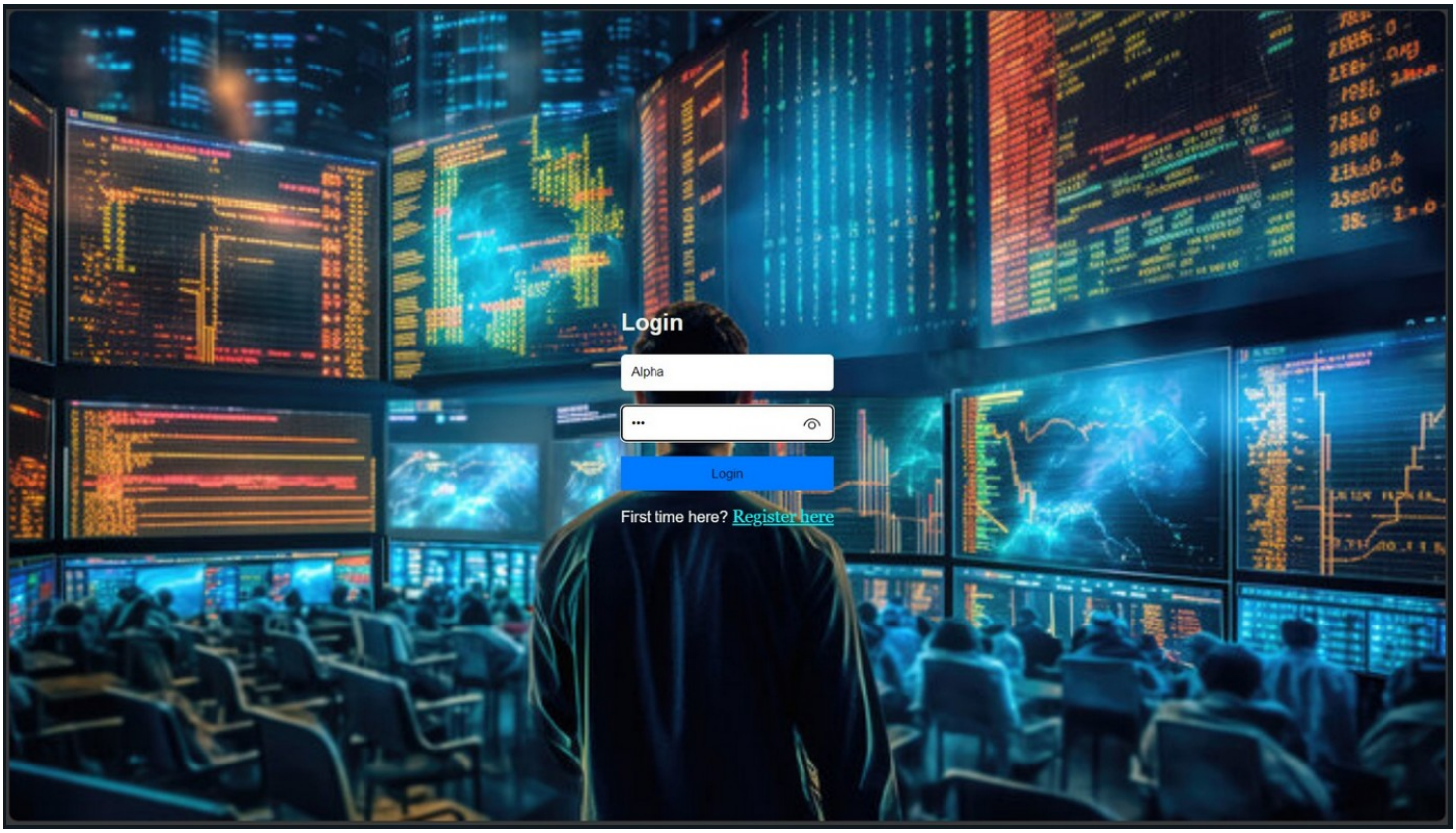
Using the *'.info'* method to get the ticker information such as -

- sector: Industry sector
- industry: Industry category
- previousClose: Previous day's closing price
- regularMarketPrice: Current market price
- marketCap: Market capitalization
- trailingPE: Trailing price-to-earnings ratio
- forwardPE: Forward price-to-earnings ratio
- beta: Beta value

return a *scalar* (single float value) whereas using *Historical method* to get historical data such as `Open`, `High`, `Low`, etc return a *vector*.

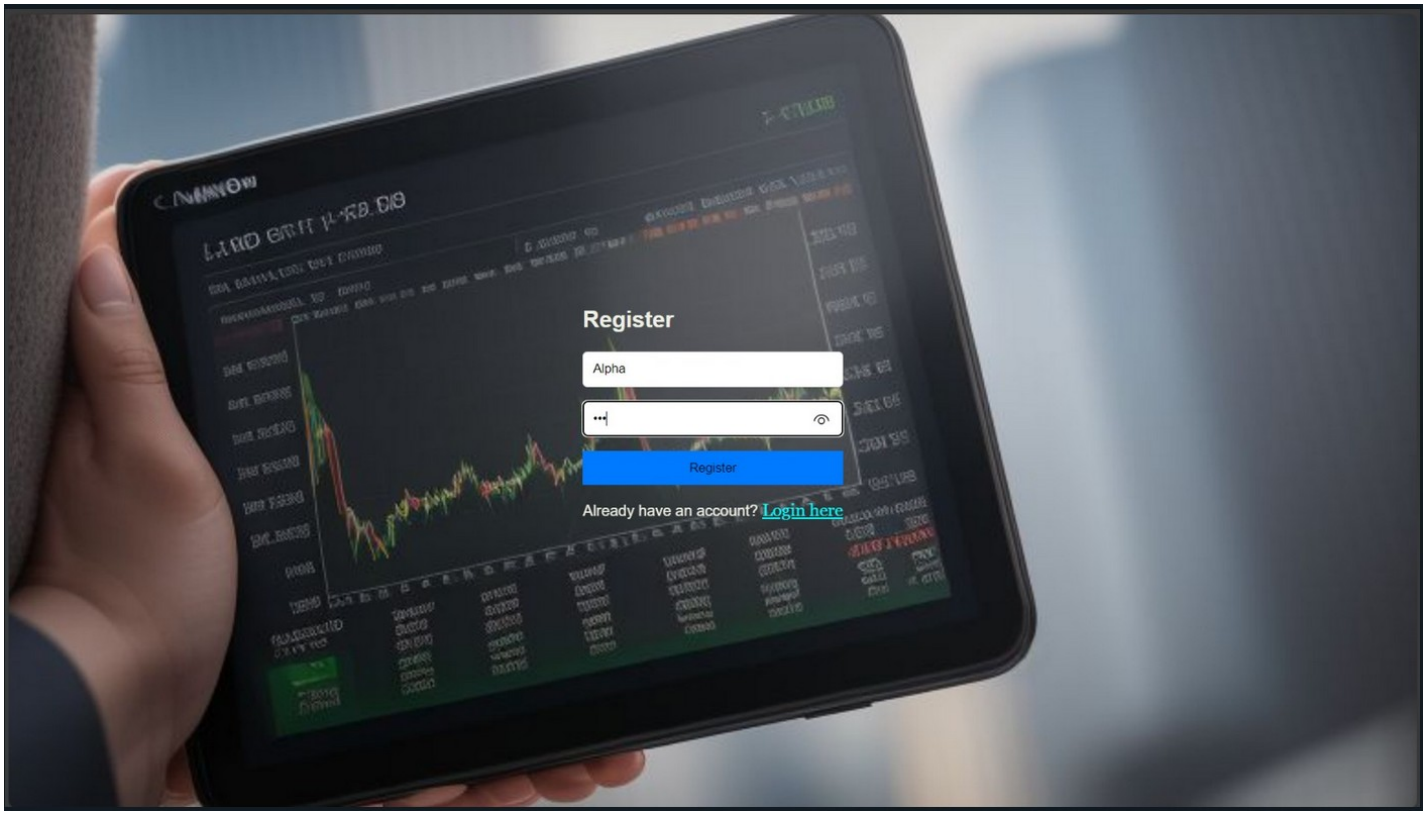
This caused problems as when I was returning value and trying to display it on the html page.

Screenshots



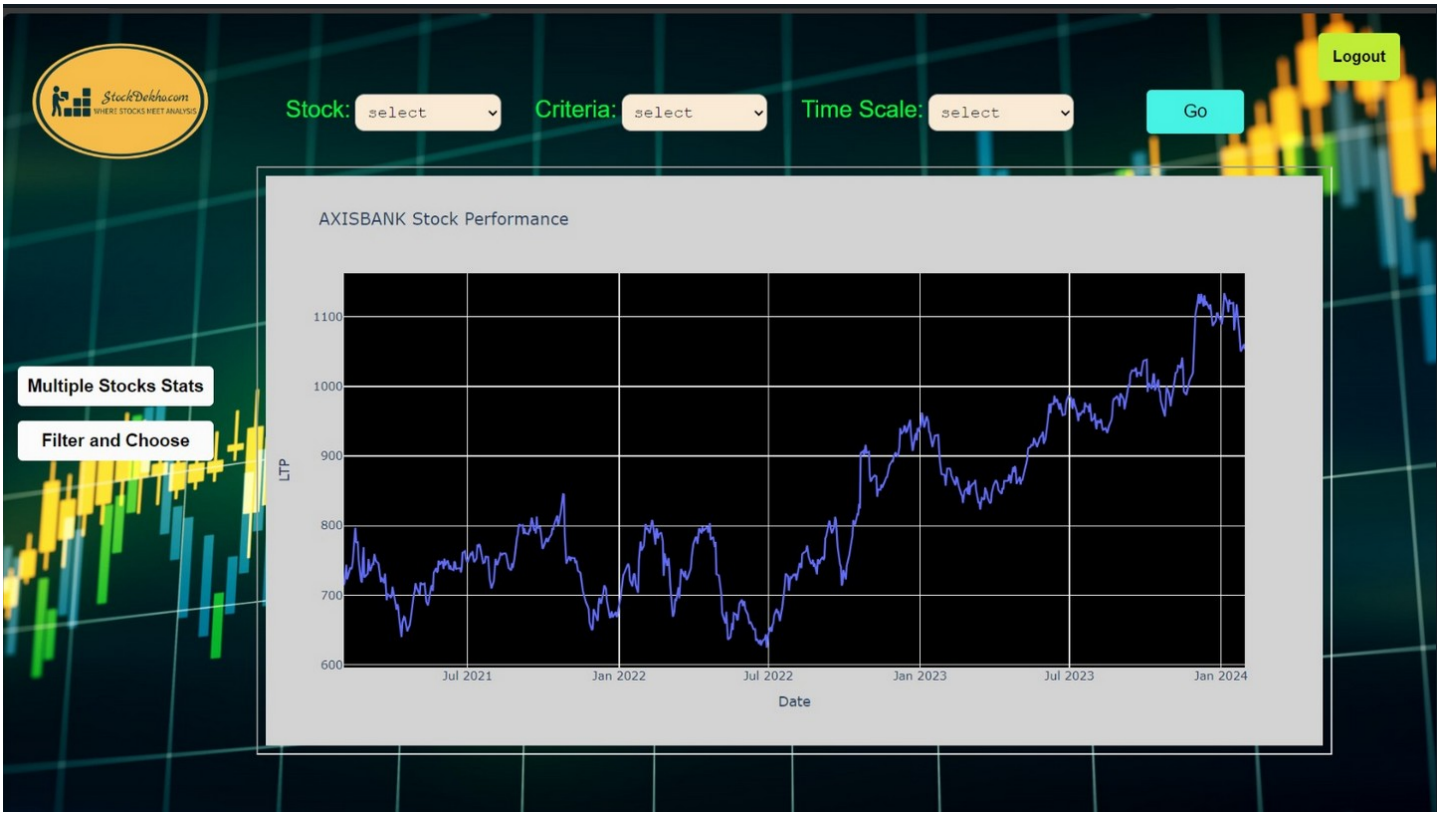
LOGIN

REGISTER

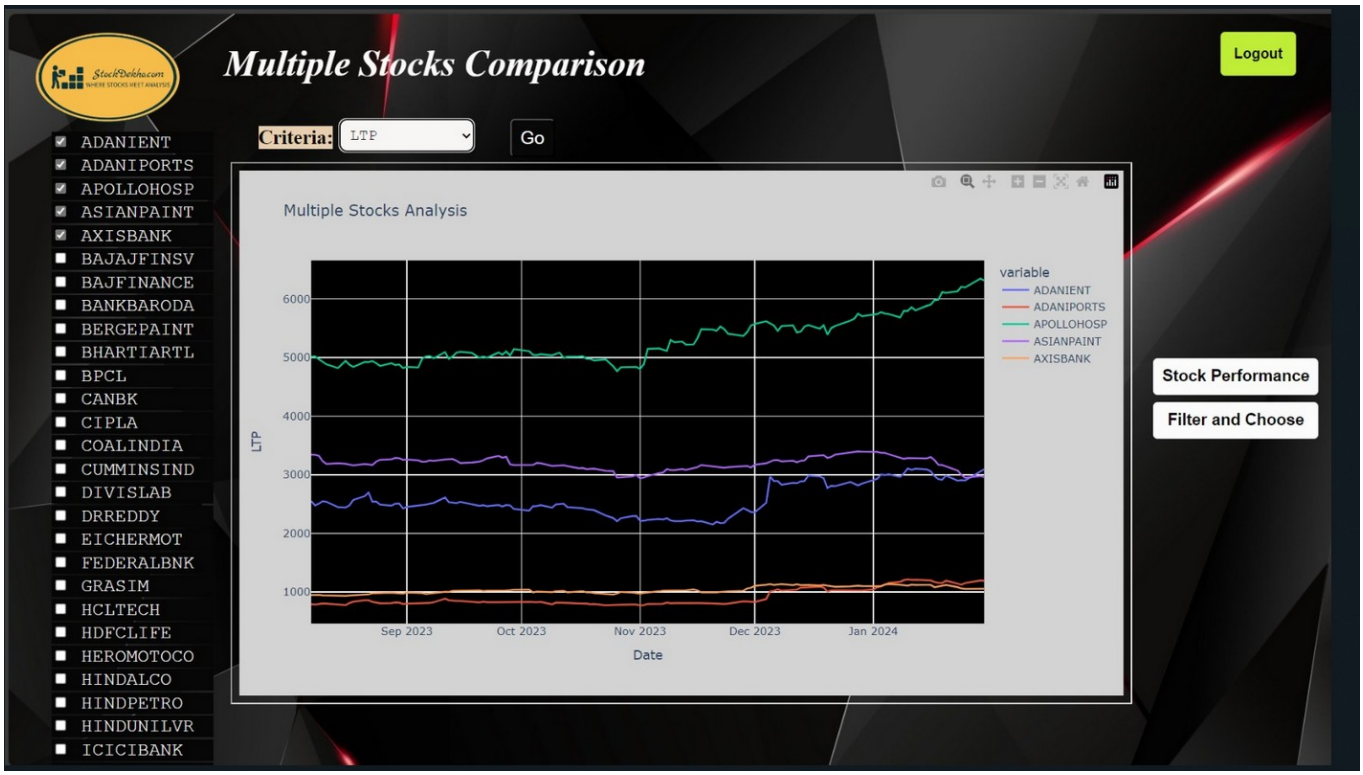

Register
Alpha
Register
Already have an account? Login here

WELCOME

SINGLE STOCK PERFORMANCE



MULTI-STOCK PERFORMANCE



FILTER