Homework 4

“Angular and RxJS”

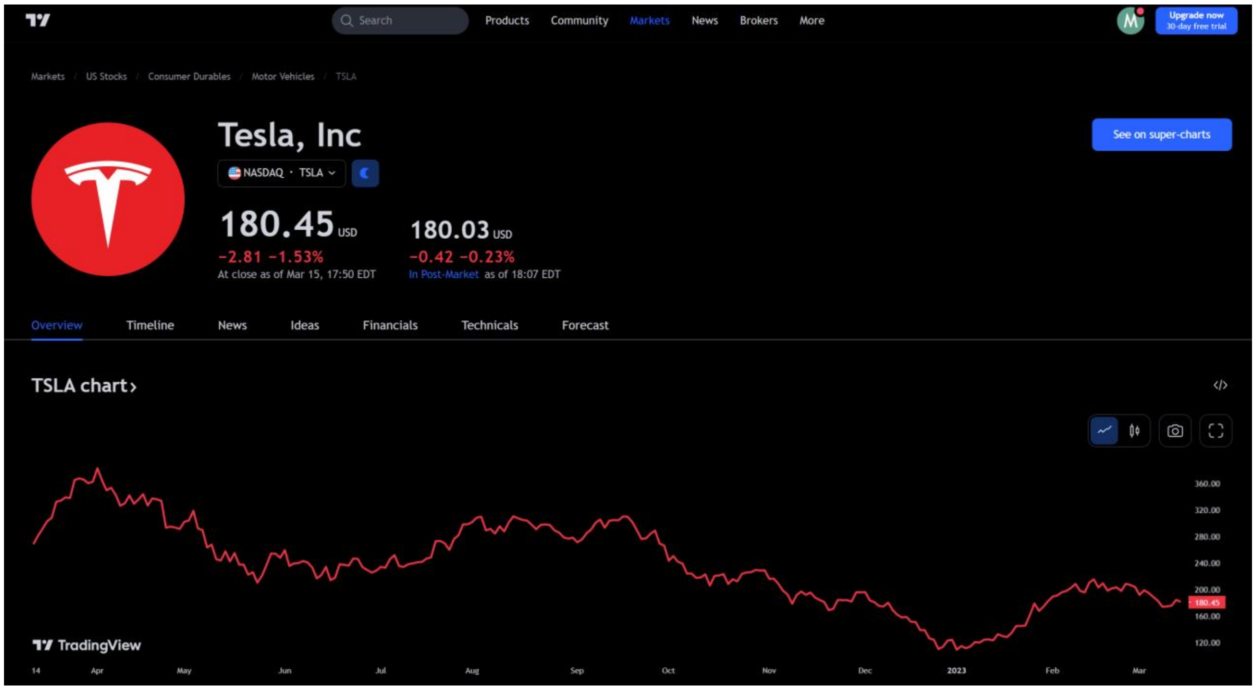
The following homework assignment is worth 100 points. Please submit all files needed for me to run your application using the Canvas drop box as a zip file. Since Angular applications are Node applications, please **do not include** the node_modules directory in your zip file.

Requirements

For this homework assignment, this will be the first assignment where **Node.js is required** to complete the assignment. That is assuming you used the Vue CDN for the third homework…

Stocks, Crypto, and Forex (oh my)

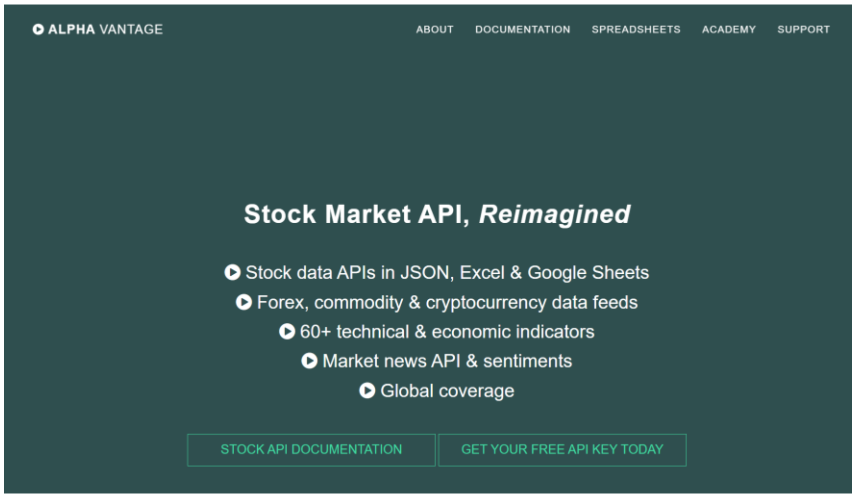
For this new homework assignment, you will be **building an Angular application** used to produce charts outlining financial data. As an example, the popular website Trading View shows stock data (using $TSLA) as follows:



Recall the second homework assignment where you were introduced to AJAX and querying an API for data for your application. We looked at approaches for querying with the Fetch API, jQuery and even the *XMLHTTPRequest* object. For Angular, we will be looking at a new approach in class called **RxJS**, which uses observables and fits well around the framework.

Financial Data API

For working on this homework assignment, you will need an API for querying financial data. A great free API that you can use is **Alpha Vantage** (https://www.alphavantage.co/):

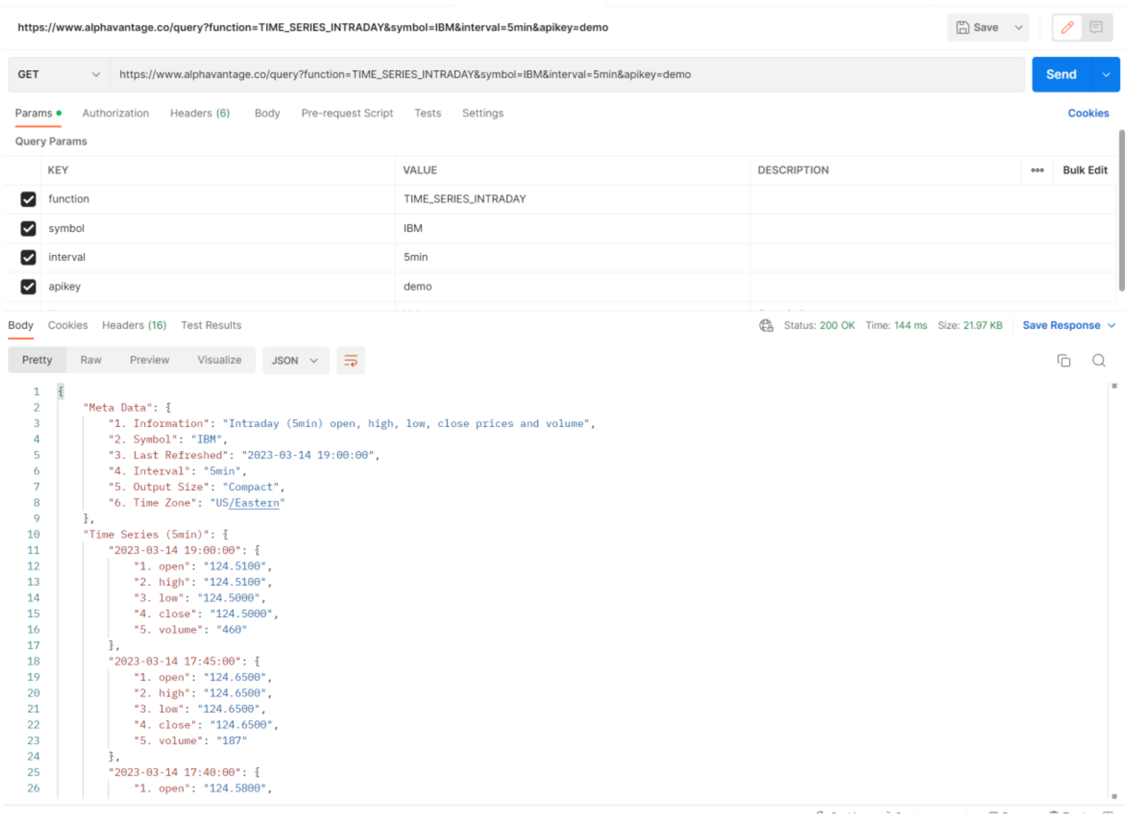


Stock Market API, *Reimagined*

- Stock data APIs in JSON, Excel & Google Sheets
- Forex, commodity & cryptocurrency data feeds
- 60+ technical & economic indicators
- Market news API & sentiments
- Global coverage

This API is free to use, although you will need to create a free API key. With your API key, you can access several of their free API calls. Their API is documented here: https://www.alphavantage.co/documentation/.

As an example, using IBM as an example, the following API call can be used to obtain price and volume data throughout the day in JSON format:



GET https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=IBM&interval=5min&apikey=demo

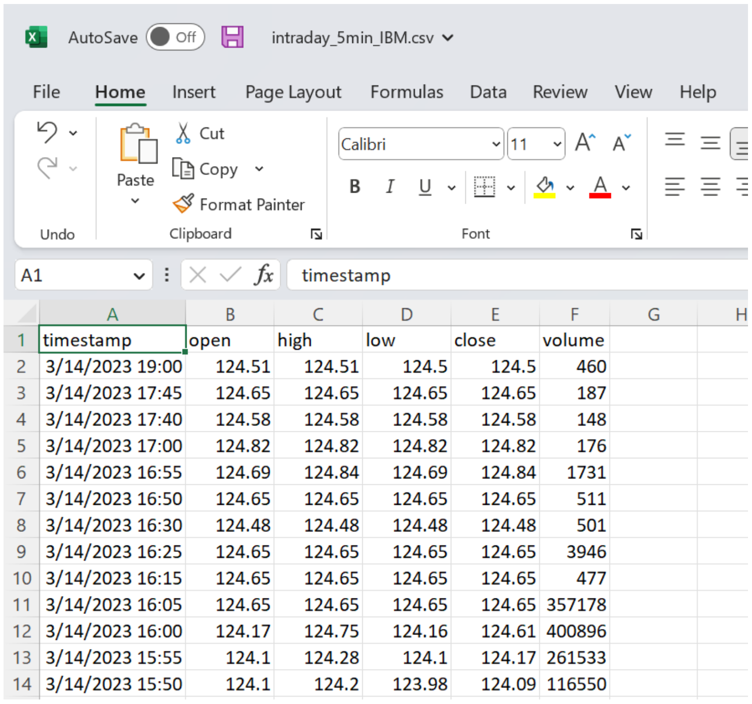
KEY	VALUE	DESCRIPTION
function	TIME_SERIES_INTRADAY	
symbol	IBM	
interval	5min	
apikey	demo	

Status: 200 OK Time: 144 ms Size: 21.97 KB

```
{
    "Meta Data": {
        "1. Information": "Intraday (5min) open, high, low, close prices and volume",
        "2. Symbol": "IBM",
        "3. Last Refreshed": "2023-03-14 19:00:00",
        "4. Interval": "5min",
        "5. Output Size": "Compact",
        "6. Time Zone": "US/Eastern"
    },
    "Time Series (5min)": {
        "2023-03-14 19:00:00": {
            "1. open": "124.5100",
            "2. high": "124.5100",
            "3. low": "124.5000",
            "4. close": "124.5000",
            "5. volume": "460"
        },
        "2023-03-14 17:45:00": {
            "1. open": "124.6500",
            "2. high": "124.6500",
            "3. low": "124.6500",
            "4. close": "124.6500",
            "5. volume": "187"
        },
        "2023-03-14 17:40:00": {
            "1. open": "124.5800",
```

If the data set is too confusing to work with, you can use their same API to retrieve the data set as comma separated values (CSV), which can be viewed in Excel:

`https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=IBM&interval=5min&apikey=demo&datatype=csv`



	A	B	C	D	E	F
1	timestamp	open	high	low	close	volume
2	3/14/2023 19:00	124.51	124.51	124.5	124.5	460
3	3/14/2023 17:45	124.65	124.65	124.65	124.65	187
4	3/14/2023 17:40	124.58	124.58	124.58	124.58	148
5	3/14/2023 17:00	124.82	124.82	124.82	124.82	176
6	3/14/2023 16:55	124.69	124.84	124.69	124.84	1731
7	3/14/2023 16:50	124.65	124.65	124.65	124.65	511
8	3/14/2023 16:30	124.48	124.48	124.48	124.48	501
9	3/14/2023 16:25	124.65	124.65	124.65	124.65	3946
10	3/14/2023 16:15	124.65	124.65	124.65	124.65	477
11	3/14/2023 16:05	124.65	124.65	124.65	124.65	357178
12	3/14/2023 16:00	124.17	124.75	124.16	124.61	400896
13	3/14/2023 15:55	124.1	124.28	124.1	124.17	261533
14	3/14/2023 15:50	124.1	124.2	123.98	124.09	116550

For working with CSV data, there are Node packages you can use (from NPM) that can be added to your application. As an example, the most popular package is called **csv** -> https://www.npmjs.com/package/csv.

Implementation Details

For this assignment you will create an Angular application to produce a way for a user to be able to view price data for any symbol the user wishes to query on. You will need to use **Angular Router** and handle the following two routes:

Route Name	Description
/	The home page where you will present the user a search option where the user can search for symbols to view price data (stocks, crypto, forex, etc.)
/data/{symbol}	The detailed page where the user is shown a chart showing the price/volume data for the symbol in the **URL query string**. For instance if the user navigates to /data/PTON, you should query the API for *Peloton* and produce a chart with that symbol's data.

Additional Notes

1. If you wish to use a different API than the one provided (Alpha Vantage), feel free to do so.
2. Feel free to be as creative as possible on your search and detail pages in the application.
3. This assignment is recommending you use **RxJS** for querying the API. If you wish to use an additional HTTP client (such as **Axios**), feel free to do so.
4. For charting data, there are a number of Angular packages available for you to use. One of the most popular is **ngx-echarts** -> https://github.com/xieziyu/ngx-echarts.

Submission

Please submit a zip file of all files needed for me to run your application before close of the Canvas drop box. Do not include the **node_modules** directory in your submission.