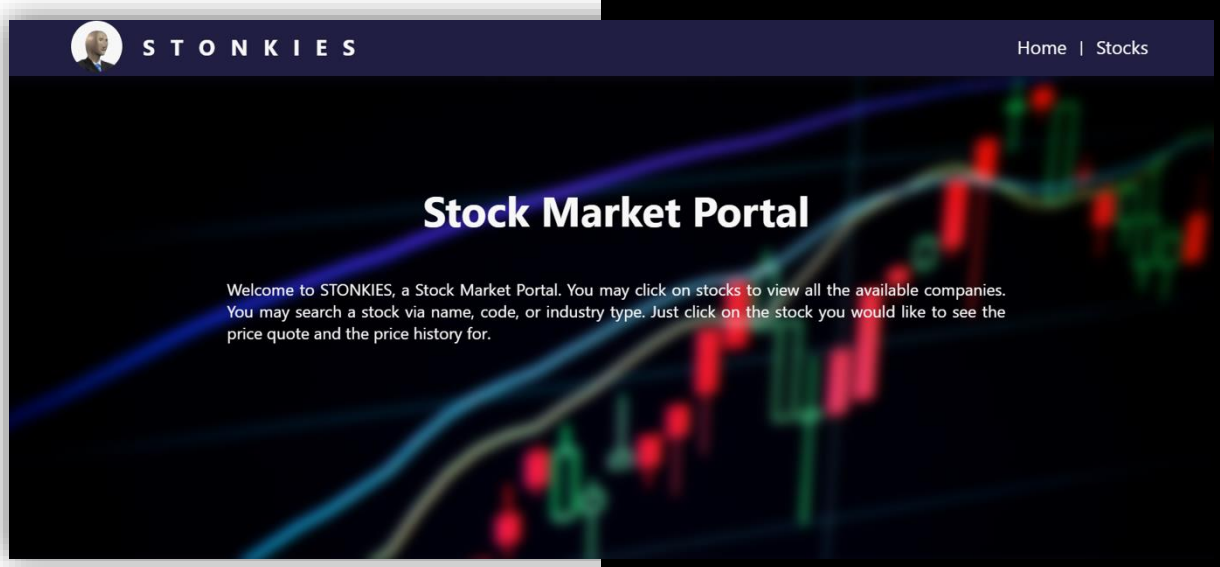
2022

IFN666 Stocks Web APP – Client Side



IFN666

Stocks Web APP – Client Side

Link to Demo video:
https://youtu.be/z9wirmpAukQ

Jaskaran Singh Sehmbey

N10911685

12/5/2022

Contents

Introduction

Purpose & description

The Web-app ‘Stonkies’, is a simple stock market portal, where users can view and analyse stock market data. It is a client-side web application which uses APIs to get the latest and correct information. This information is presented in a user-friendly manner, so that everyone can understand the data clearly and access it easily due to the simple navigation. The website is developed with the use of React.js with the additional help of CSS, JavaScript, and various modules (dependencies that will be discussed later in the document).

In the Web-app, first thing I did was to remove price history and quote from the navigation bar, as this was creating confusion on how the overall navigation was meant to be. Adding the home page and Stocks page makes it clearer for users, which can be seen on the ‘stocks’ page where the navigation becomes unidirectional. After a user filters a stock via code, name, or industry type and clicks on it, I choose to display Price Quote and Price History on the same page to make navigation for users easier. Another feature I chose to add was the description of the selected stock which makes the page full of important information which a user could be interested in.

The website has a dark contrast, and the colours are based on the 60/30/10 colour rule. The website also uses Ag grid for making the tables and Chart.js to display the line graph. I named the website ‘Stonkies’ based on the stocks meme ‘stonks’ to make the website trendy, which could also make it engaging for the younger users (approx. 16 – 25 years old) and set them as the target audience who want to get into stock market.

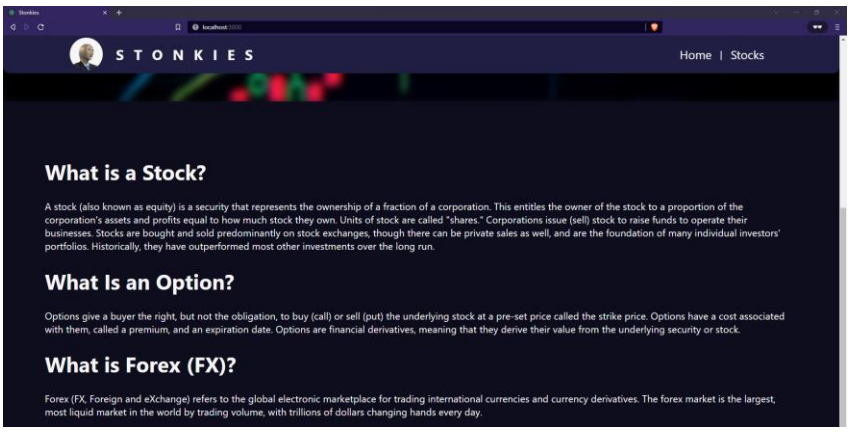


Figure 1: Home Page which also has information related to stocks.

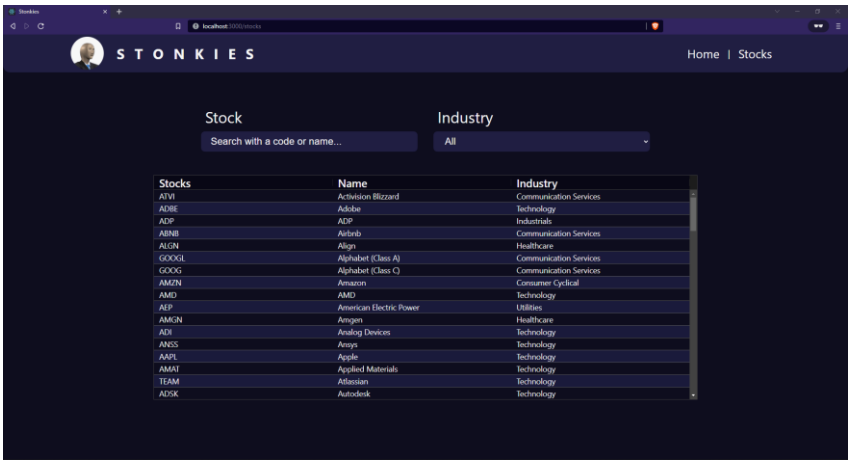


Figure 2: Stocks page showing all the stocks available to view more info for.

Completeness and Limitations

All the features in the brief document work on the Stonkies website. The features include various filters, tables, charts, APIs fetching information, navigation, and error handling. These features are made with the help of React Hooks (useState, useEffect, useCallback, useLocation, etc), React Router, Query REST APIs, third party components, and personalised components.

The Website runs smoothly with easy navigation (handled using React Router), with good components to help structure the website neatly. The APIs run smoothly with proper error handling and loading screens as well. The Table can be sorted according to the name, code, or industry in ascending or descending order as well on top of the filter functionality. Hence overall in my perspective it is a website with a potential of at least a grade 6 or even a grade 7. – idk if this is good to include or not

The only limitation of the website is that the 'Alphavantage' API runs out of API calls very quickly compared to the 'Financial Modeling Prep' API. But even so, there are error handling screens when it runs out of API calls. Other than that, there are no errors or limitation. The FMP API is called once when a user goes to the "Stocks page". The AV API has 3 endpoints - description, price history, and quote which all 3 are called together once when a user clicks on a stock from the stock list and the history page loads. There are no unnecessary API calls while searching or filtering.

Use of APIs

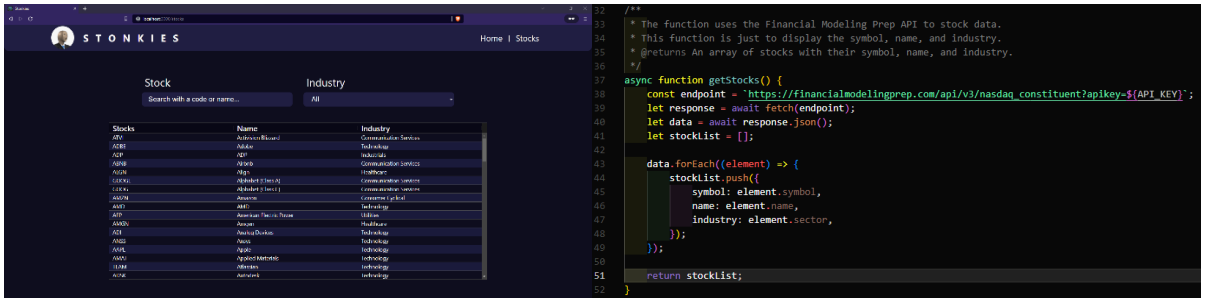
The following APIs are used in the Stonkies web-app:

- Financial Modeling Prep (FMP)
- Alpha Vantage (AV)

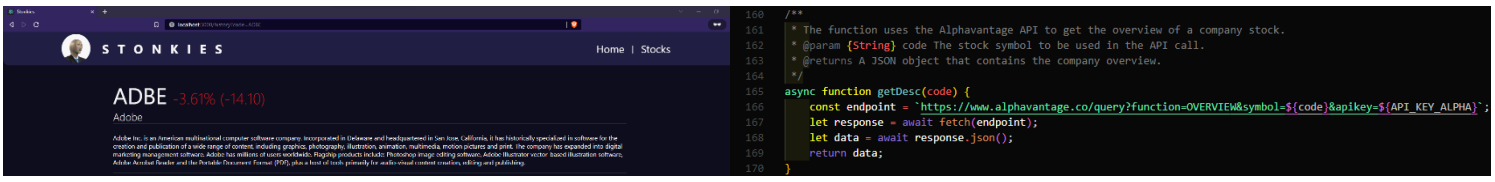
Use of End Points of the chosen Stock API

I chose to use two APIs – FMP and AV, to avoid running out of API calls too quickly. The FMP API is used to get the NASDAQ 100 stock list and just display the list of stocks available to view more information for. Whereas the AV API is used to do the detailed work – getting stock history, price quote and description of a single selected stock. This way if we run out of AV API calls to see the data of a particular stock, we can go back to the list of stocks and wait before clicking on another stock for its details. Now we will look at the APIs in more detail.

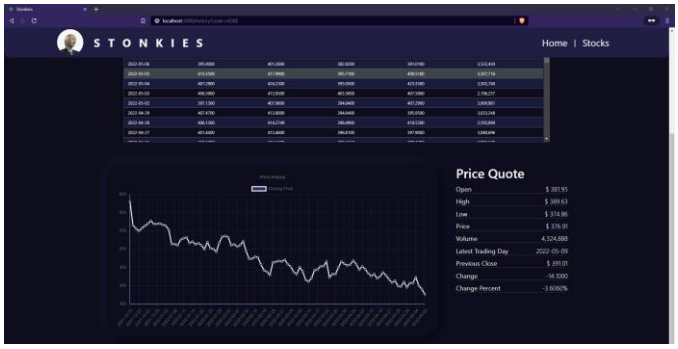
1. FMP - **List of Nasdaq 100 companies**

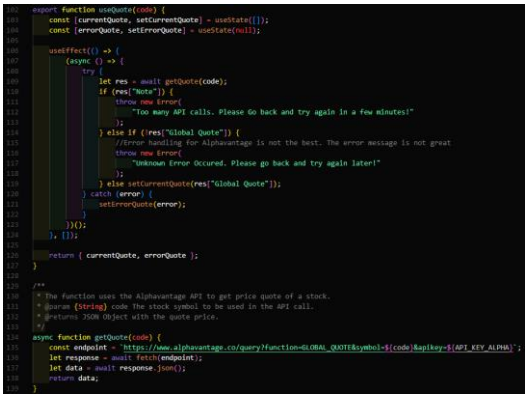


2. AV - **Company Overview** – Description Only (*https://www.alphavantage.co/query?function=OVERVIEW&symbol=IBM&apikey=demo*)

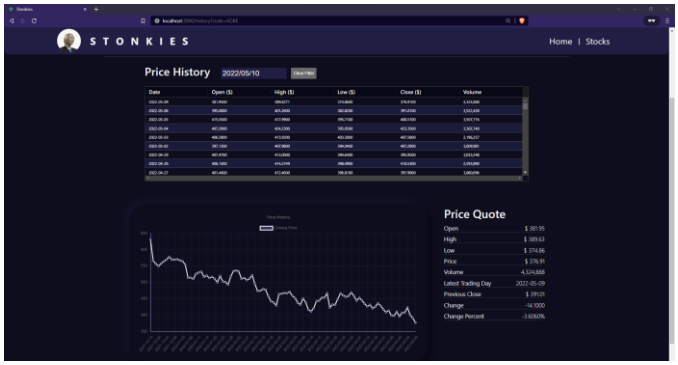


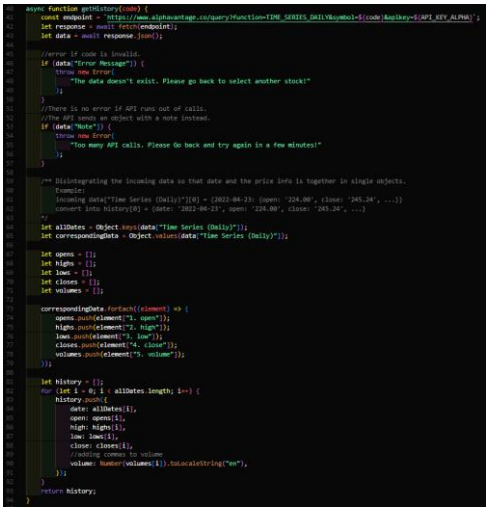
3. AV - **Quote Endpoint** – Price Quote Section, heading price change, and change percentage (*https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=IBM&apikey=demo*)





4. AV - **Time_SERIES_DAILY** – Price History Table and Chart (*https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=IBM&apikey=demo*)





Modules used

1. *Ag-grid-react*
   Module to provide fully featured table components, including sorting and filtering.
   https://www.ag-grid.com/react-grid/
2. *React-chartjs-2*
   React components for Chart.js.
   https://www.npmjs.com/package/react-chartjs-2
3. Chart.js
   Simple JavaScript charting module
   https://www.chartjs.org/
4. React-datepicker
   A simple and reusable Datepicker component for React
   https://www.npmjs.com/package/react-datepicker
5. React-parallax
   Module to create a parallax scroll effects for images.

https://www.npmjs.com/package/react-parallax

6. React Router Dom
   Module used to make routing in the web-app.
   https://reactrouter.com/

Application Design

Navigation and Layout

On the website, there are 3 main pages that a user will see. The navigation only has 2 options – 'Home' or 'Stocks'. My idea was to keep the navigation as simple as possible, which meant the removal of price history and quote from it. This is because if price history and quote were included in the nav bar and the user searches for a stock in the 'Stocks' page, to view history or quote they will again need to switch to that page and search for the stock again. Therefore, to eliminate the repetitive process this design choice was made. The 'Home' page is simple and provides an introduction to the website with some definitions if a user scrolls down.

The 'Stocks' page includes a table with a list of all the stocks (Nasdaq 100) and 2 filters to search for a stock that a user may want to know further price history and quote for. The user can either search via name or the code/symbol of the stock. The second filter is to categorise the industry type of the different stocks. Once the user clicks on a particular stock, they will now be taken to the third page, which is not on the nav bar – 'History' page (localhost:3000/history?code=xxxx). This page shows the price history on a table and a chart. The price history can be filtered as well with the date picker component so that the history will be showed from the picked date in the past till the current date. This page also shows the stock quote and the description of the company that the stock is for. I decided to keep quote in the same page as you can see that the quote doesn't have too much information, so it would be hard to design a separate page with just the quote as it will look empty. The layout of the 'History' page is also simple with big headers, also showing the stock price change next to the symbol (green if it is positive, red if it is negative). This was seen in many other stock market websites and is a good way of presenting the price change and change percentage. The table is on top of the chart as the history table is quite wide, so the chart and quote can fit in one line very well.

After a user has viewed the particular stock in the 'History' page, they can simply press 'Stocks' on the nav bar to go back to the stock list.

The following shows the paper prototype of the website. As you can see a lot of design changes were made to the prototype compared to the final result.

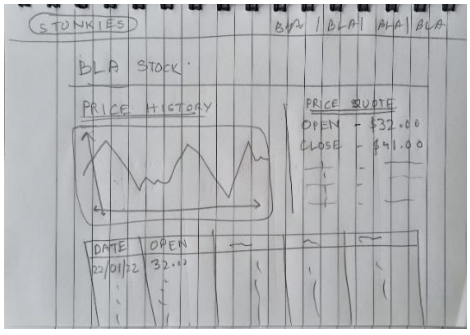


Figure 4: History page

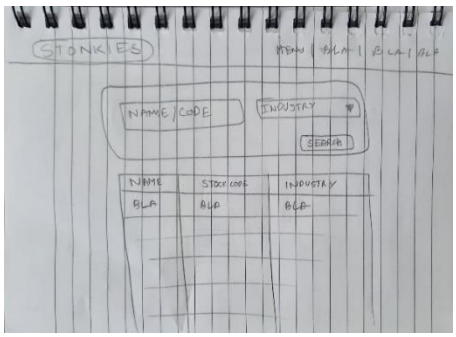


Figure 3 Stocks page

There is a big compromise that makes the website awkward to use – that is the price quote is on the same page as price history. As explained before, quote doesn't have too much information, hence

giving it a separate page would have too much white space. Having it together with the price history makes the information easily accessible but without proper layout and spacing the page can look congested. One way of fixing this could be adding an overview page of the selected stock which has the quote, description, and latest news related to the company. The screenshot of wall street journal's stock market portal (figure 29 in appendix- C) shows a good example of Tesla's stock page overview.

Technical Description

Architecture

On the source code level, there are 3 folders and 4 files:

1. node_modules – Used to store all the npm packages.
2. public – Contains all the static files which are not processed by the webpack (react javascript compiler)
3. src – contains all the files that needs to be processed by the webpack to run the react application.
4. .gitignore – text file which tells Git to ignore files and directories mentioned in it.
5. Package-lock.json – keeps track of exact version of all the packages (dependencies)
6. package.json – JSON file that holds the metadata relevant to the project and is used to manage the project dependencies like installations from npm, scripts such as 'npm start', version, etc.
7. README.md – markdown file that describes what the directory is about and has a few instructions.

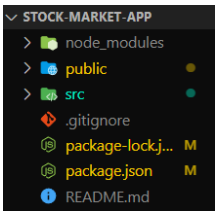


Figure 5

The src directory has 5 main folders inside it. These folders have been divided according to the services that they provide.

1. apis – The JavaScript files with functions related to the APIs being used.
2. assets – The images used on the website and room for other possible assets in the future.
3. pages – The different pages that can be navigated to.
4. components – Includes the components that can either be re-used (loading, nav-bar), or helps the code in the pages folder become cleaner, easier to understand, and easier to change layout with (for instance, if the quote box needs to be moved to another location on the page, that can be done by just moving <QuoteBox /> (figure 6).
5. styles – all the CSS files in one place instead of mixing them with the pages and components folders for easier access

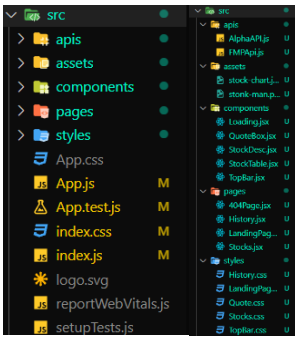


Figure 7

Figure 6

Figure 8: Components used on the 'History'

Keeping the pages folder separate from components made the architecture cleaner so that there was no confusion between where the navigation pages were (the ones to be displayed) and were the components (used inside the pages). All the CSS files names also relate to the pages they are being used in. Some components inside the pages files also use the CSS directly from the parent file (such as StockTable.jsx component is called inside Stocks.jsx page but the CSS Stocks.css is only imported inside Stocks.jsx)

Test plan

The following are the manual tests conducted on the finished website. The screenshots mentioned are available in the appendix -@

Task	Expected Outcome	Result	Screenshot(s)
Checking Home page			
Landing page show on http://localhost:3000/	Home page displayed with introduction and some basic info about stocks.	PASS	Figure 10
Scroll up and down to see if the parallax image in the background is working.	The parallax works when scrolled up or down.	PASS	N/a
Checking Stocks page			
Go to Stocks page (http://localhost:3000/stocks) from the navbar and check the list of all the stocks	Stocks page is displayed with the 2 input options and a table with the filter.	PASS	Figure 11
Type a company name in the Stock filter to filter by name.	Stocks filtered by name.	PASS	Figure 12
Start typing a company code in the Stock filter to filter by code.	All Stocks starting with the letters in the search field are filtered and displayed.	PASS	Figure 13
Use the industry filter and select an industry from the available dropdown options.	Stocks filtered by industry.	PASS	Figure 14 – a Figure 14 – b
Type into Stock and Industry to filter by both.	Stocks filtered by both name and industry.	PASS	Figure 15
Search by “invalid” name.	Message “No Results Found” shown in the Table.	PASS	Figure 16
Search by "Tesla” name in the wrong industry type.	Message “No Results Found” shown in the Table.	PASS	Figure 17
Refresh to see the loading screen.	“Loading…” message shown.	PASS	Figure 18
Click on a row to navigate to the detailed stock dashboard (History page).	Stock details shown on the new a new page.	PASS	Figure 19
Intentionally throw a new error in the FMPApi.js file in the fetch function to see if the try and catch detects the error.	The error message is shown on the stocks page with the nav-bar available on the top.	PASS	Figure 20 – a Figure 20 – b
Checking the detailed stock information (History page)			

Detailed stock details shown at http://localhost:3000/history?code=ADBE	Details about the ADBE (adobe) stock is shown.	PASS	Figure 19
Check the description, price history table, price history chart, and the quote of the selected stock.	All the data is available without errors.	PASS	Figure 21
Change the Date filter.	The price history table and the price history chart updates according to the picked date.	PASS	Figure 22
Use the 'Clear Filter' button.	The filter is cleared, and the table and the chart show the entire data.	PASS	Figure 23
Change the stock code in the URL http://localhost:3000/history?code=invalid	A data not available message is displayed, and user is asked to go back to pick another stock. Nav-bar still available.	PASS	Figure 24
Pick a valid stock and refresh the page a couple of times to run out of API calls.	Too many API calls error message is displayed, and user is asked to go back and try again in a few minutes. Nav-bar still available.	PASS	Figure 25
Try picking a date in the future.	Unable to pick it as the dates are greyed out and disabled.	PASS	Figure 26
Change the stock code in the URL http://localhost:3000/history?code=gold which is not part of the Nasdaq 100 list but is a company outside the list with the stock code gold	The page opens and data is shown but there is a message underneath the code stating that this code is not a part of the Nasdaq 100 list.	PASS	Figure 28
	404 error		
Change the URL to http://localhost:3000/random	A 404-error message is displayed as this page does not exist. A link to go back to the home page is provided and it works.	PASS	Figure 27

Difficulties / Exclusions / unresolved & persistent errors /

My biggest roadblock was dealing with promises and learning async/await functions. The logic was quite clear from the beginning, but I still made some smaller examples in a separate file to understand how it works. I was able to use the practical example to make similar functions in my project 'stonkies' and was able to fetch the data very easily. Displaying the fetched data was also very simple. One big issue that came up was when I tried to fetch the price history from the AV API and display it in the AgGrid table. The data was coming through perfectly as I was able to console.log it but for some reason, that I didn't understand yet, it wasn't showing in the table and using useState I was unable to put the data using the setRowData. I had previously used the FMP API to display the list of Nasdaq 100 list in the AgGrid table, so the logic should have been the same, but I knew because of different APIs, it might be the issue. After a few hours of trying to debug the issue, I finally approached Benjamin, my tutor, and showed him the issue. He tried to solve the issue using useEffect hook, but we could see the data in console.log that it would set the data using useState but then unset it. Finally, I pointed out that the data is also being set in the onGridReady function of the AgGrid. Once we removed that, the table was populated. The solution was basically set the useState inside useEffect only when stockHistory was being populated. Figure 9 shows the solution from line 98 to 100 in History.jsx

Figure 9

Other than that, I was able to implement all the functionalities successfully. The only other issue was to make the website responsive to mobile browsers. I tried to make most of the components responsive to smaller screen sizes (phones, ipads) but some other 3rd party components such as ag-grid and chart.js were hard to make responsive and currently doesn't look very good compared to the web version. The demo doesn't show the responsiveness as it was implemented at a very late stage, but it has been implemented as you can see in Figures XXX in appendix – C.

Extensions

There are many potential future possibilities for this web-app. For me, I would like to see this app as a learning tool for people who want to get into trading stocks. I have used the Binance API before with python to make a trading bot, and it would be good to combine the knowledge from the Binance API to make it a trading platform for beginners. It would have latest news about the companies, tutorial links, step by step guide on what and how to do (short-term and long-term) trading, and finally monetising the app so that we can make as many API calls as we want.

User guide

The website is very simple to use.

1. Once you go to the home page (localhost:3000), you can scroll down to view info about what stocks are. Next you can simply click on "Stocks" on the navigation bar.
2. In the Navigation bar, feel free to search a stock by its name or code and even filter the stocks using the industry type.
3. Click on a stock that you want more information about

4. Once the page loads, you can now look at the stock price change and percentage change on the top next to the symbol. If it is positive the numbers will be green and if it is in negative the red. You can read the description of the company, view the price history on either the table or the chart. To view history from a particular period, just change the date next to the title “Price History”, the table and the chart will update instantly. You can also view the price quote at the bottom right side, next to the price history chart.

References

Chart.js | Open source HTML5 Charts for your website. (n.d.). Retrieved 12 May 2022, from https://www.chartjs.org/

Free Stock API and Financial Statements API - FMP API. (n.d.). Financial Modeling Prep. Retrieved 12 May 2022, from https://financialmodelingprep.com/developer/docs

Free Stock APIs in JSON & Excel | Alpha Vantage. (n.d.). Retrieved 12 May 2022, from https://www.alphavantage.co/

React Data Grid: Documentation. (n.d.). Retrieved 12 May 2022, from https://www.ag-grid.com/react-data-grid/

React-chartjs-2. (n.d.). Npm. Retrieved 12 May 2022, from https://www.npmjs.com/package/react-chartjs-2

React-datepicker. (n.d.). Npm. Retrieved 12 May 2022, from https://www.npmjs.com/package/react-datepicker

React-parallax. (n.d.). Npm. Retrieved 12 May 2022, from https://www.npmjs.com/package/react-parallax

TSLA | Tesla Inc. Stock Price & News. (n.d.). WSJ. Retrieved 12 May 2022, from https://www.wsj.com/market-data/quotes/TSLA