Homework 8: Ajax, JSON, Responsive Design and Node.js

Stock Search

(AJAX/JSON/HTML5/Bootstrap/Angular /Node.js/Cloud Exercise)

1. Objectives

- Get familiar with the AJAX and JSON technologies
- Use a combination of HTML5, Bootstrap and Angular on client side
- Use Node.js on server side
- Get familiar with Bootstrap to enhance the user experience using responsive design
- Get hands-on experience of Cloud services hosting NodeJS/Express
- Learn to use popular APIs such as Tiingo API and NewsAPI

2. Background

2.1 AJAX and JSON

AJAX (Asynchronous JavaScript + XML) incorporates several technologies

- Standards-based presentation using XHTML and CSS
- Result display and interaction using the Document Object Model (DOM)
- Data interchange and manipulation using XML and JSON
- Asynchronous data retrieval using XMLHttpRequest
- JavaScript binding everything together

See the class slides at http://csci571.com/slides/ajax.pdf

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. Its main application is in AJAX web application programming, where it serves as an alternative to the use of the XML format for data exchange between client and server. See the class slides at:

http://csci571.com/slides/JSON1.pdf

2.2 Bootstrap

Bootstrap is a free collection of tools for creating responsive websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. To learn more details about Bootstrap please refer to the lecture material on Responsive Web Design (RWD). You should use **Bootstrap 4** in this homework. See the class slides at:

http://csci571.com/slides/Responsive.pdf

2.3 Cloud Services

2.3.1 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for Node.js visit this page:

https://cloud.google.com/appengine/docs/standard/nodejs/

2.3.2 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

To learn more about AWS support for Node.js visit this page:

https://aws.amazon.com/getting-started/projects/deploy-nodejs-web-app/

2.3.3 Microsoft Azure

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

To learn more about Azure support for Node.js visit this page:

https://docs.microsoft.com/en-us/javascript/azure/?view=azure-node-latest

2.4 Angular

Angular is a toolset for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

For this homework, Angular 7+ (Angular 7,8, 9 or 10) can be used, but **Angular 10 is recommended**. Please note Angular 7+ will need familiarity with Typescript and component-based programming.

To learn more about Angular 7+, visit this page:

https://angular.io/

2.5 Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

To learn more about Node.js, visit:

https://Node.js.org/en/

Also, **Express.js** is strongly recommended. Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is in fact the standard server framework for Node.js.

To learn more about Express.js, visit:

http://expressjs.com/

All APIs calls should be done through your Node.JS server

3. High Level Description

In this exercise you will create a webpage that allows users to search for stocks using the Tiingo API and display the results on the details page. The application evolves from previous homework.

A user will first open a page as shown below in **Figure 1**, where user can enter the stock ticker symbol and select from a list of matching stock symbols using "autocomplete." A quote on a matched stock symbol can be performed. The description of the Search Box is given in Section 3.1. Instructions on how to use the API are given in **Section 4**. All implementation details and requirements will be explained in the following sections.

There are 4 routes for this application:

a) Homepage/ Search Route ['/'] – It is a default route of this application.
b) Watchlist Route ['/watchlist'] – It displays the watchlist of the user.
c) Portfolio Route ['/portfolio'] – It displays the portfolio of the user.
d) Details Route ['/details/<ticker>'] – It shows the details of the <ticker>.

When a user initially opens your web page, the initial search page should look like in **Figure 1**.

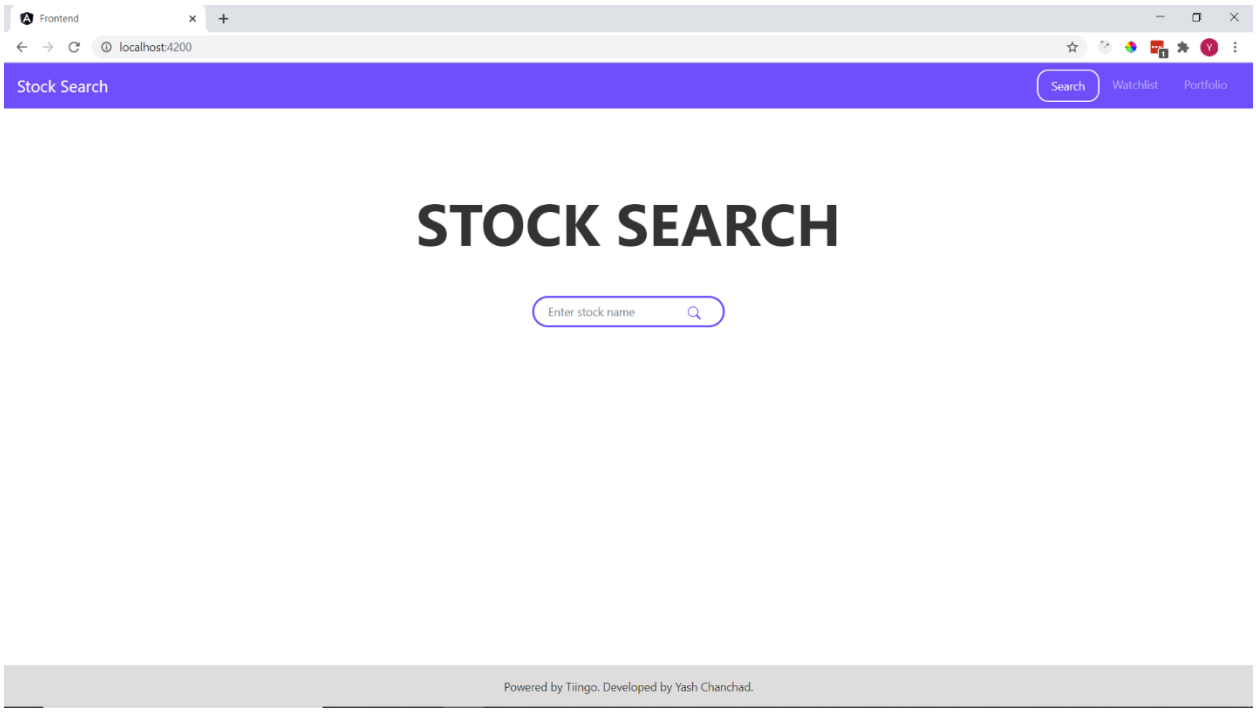


Figure 1: Initial Search Page

3.1 Search Page/ Homepage

3.1.1 Design

You must replicate the **Search Bar** displayed in **Figure 1** using a **Bootstrap form**. The Search Bar contains two components.

1. **Stock Ticker**: This is a text box, which enables the user to search for valid stocks by entering keywords and accept a suggestion of all possible tickers. The user needs to select one match before clicking on the search icon. Notice the "helper" text inside the search box.
2. **Search Button**: The "Search" button (which uses the widely used search icon), when clicked, will read the value from the textbox and send the request to the backend server. On a successful response, details for that stock will be displayed.

3.1.2 Search Execution

Once the user clicks on the "Search" button, your application should make an HTTP call to the Node.js script hosted on GA/AWS/Azure back end (the Cloud Services) . The Node.js script on Cloud Services will then make a request to the Tiingo API and NewsAPI services to get the detailed information and news.

3.1.3 Autocomplete

- A Search Bar allows a user to enter a keyword (Stock ticker symbol / company name) to retrieve information. Based on the user input, the text box should display a list of all the matching company's ticker symbols with company's name (see **Figure 2**). The autocomplete JSON data is retrieved from the ***Tiingo Search API*** (refer to Section 4.1.5). These are examples of calling this API:

```
https://api.tiingo.com/tiingo/utilities/search/<query>
# or
https://api.tiingo.com/tiingo/utilities/search?query=<query>
For example:
https://api.tiingo.com/tiingo/utilities/search/apple
```

- The autocomplete function should be implemented using **Angular Material**. Refer to Section 5.3 for more details.

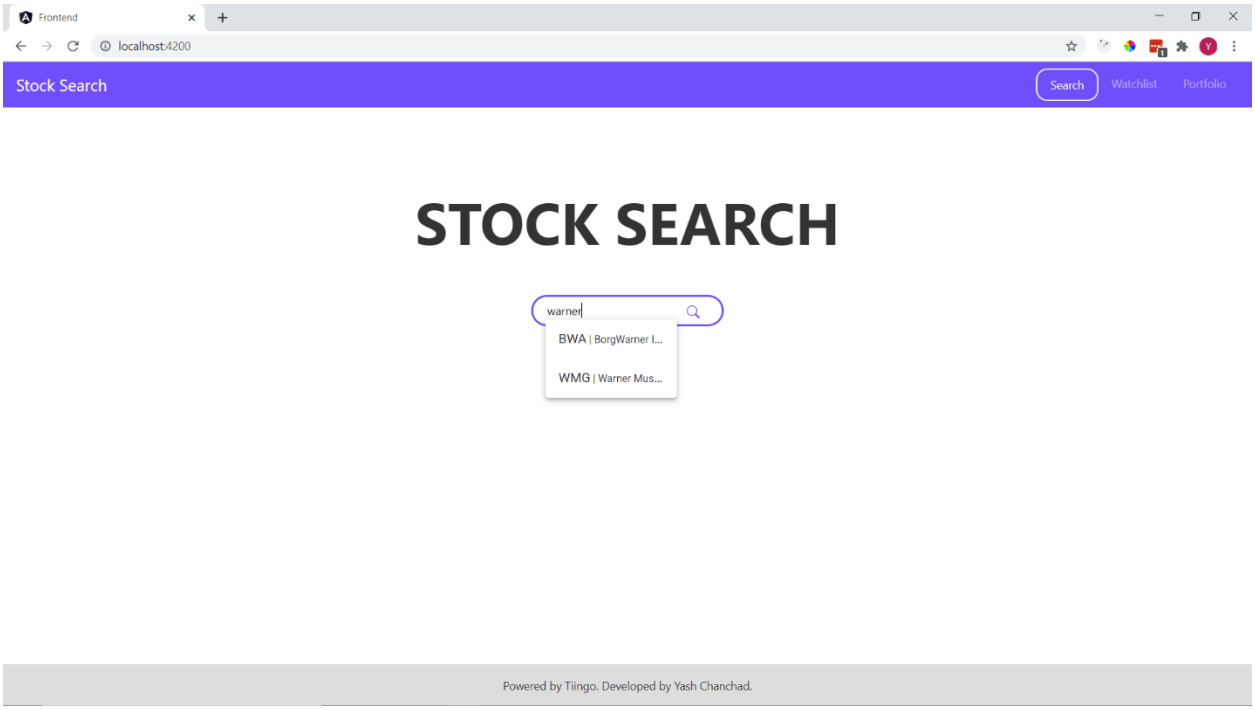


Figure 2: Autocomplete Suggestions

3.2 Details Page

3.2.1 Details of Searched Stock

After the user clicks the Search button, a page should route to /details/<ticker> path (example: /details/AMZN). The following components need to be displayed on successful search:

- Symbol, company name, trading Exchange (such as NASS+DAQ), and Buy button on top left;
- Last price, change, percent change, and date/time, on top right. The change items should be preceded by appropriately colored arrows);
- Indication of open / closed market;
- Summary, Top News and Charts tabs.

Please refer to **Figure 3.1** and **Figure 3.2** below.

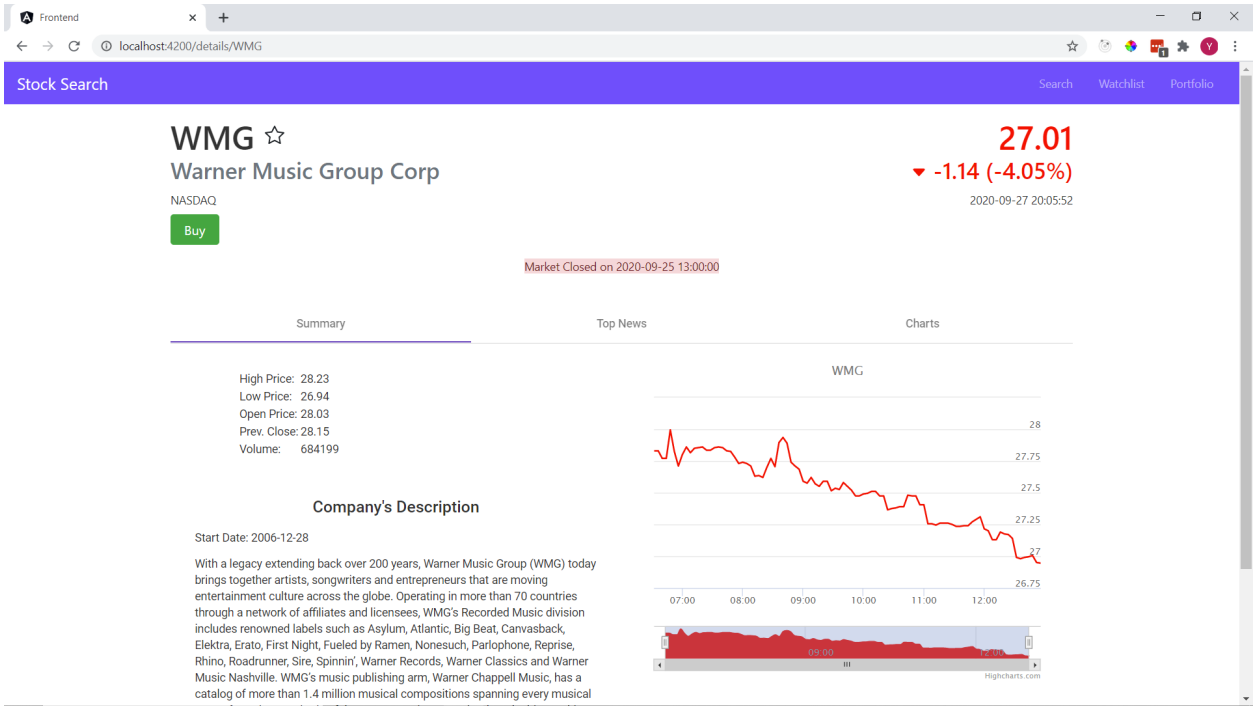


Figure 3.1: Detail page overview (Market is Closed)

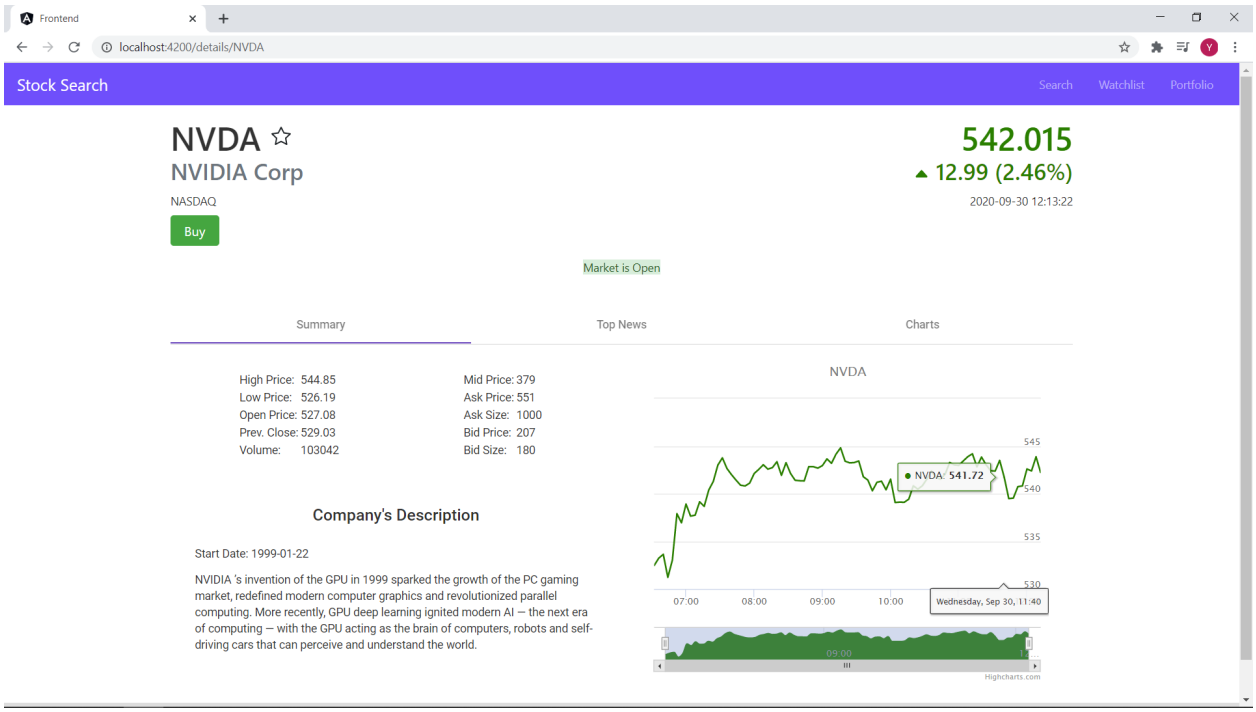


Figure 3.2: Detail page overview (Market is Open)

- When the user clicks on the star icon, the white star turns yellow, and that ticker should be stored in local storage (see HTML5 localstorage). A self-closing alert should be displayed at the top and that stock should be visible on the Watchlist Page (see section 3.3)
- When the user clicks on the **Buy** button, a modal window should open. which will display the details (stock symbol, current price, quantity of shares to buy and total price for the shares), as shown in Figure 3.3. Note that:
    - The **Buy** button should be disabled if the user inputs the quantity < 1 or the quantity field is empty. (see Figure 3.4);
    - The **Buy** button will be enabled once the user enters a number greater than 0 (see **Figure** 3.3).

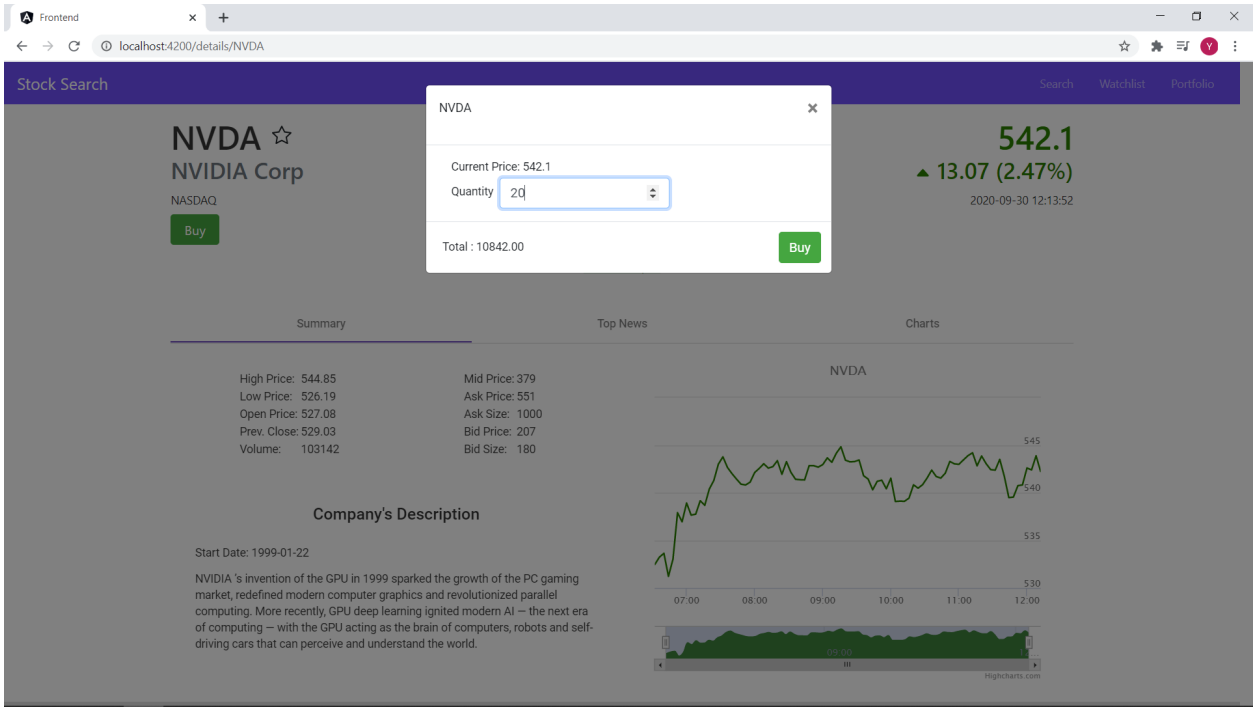


Figure 3.3: Buy Button enabled for valid input

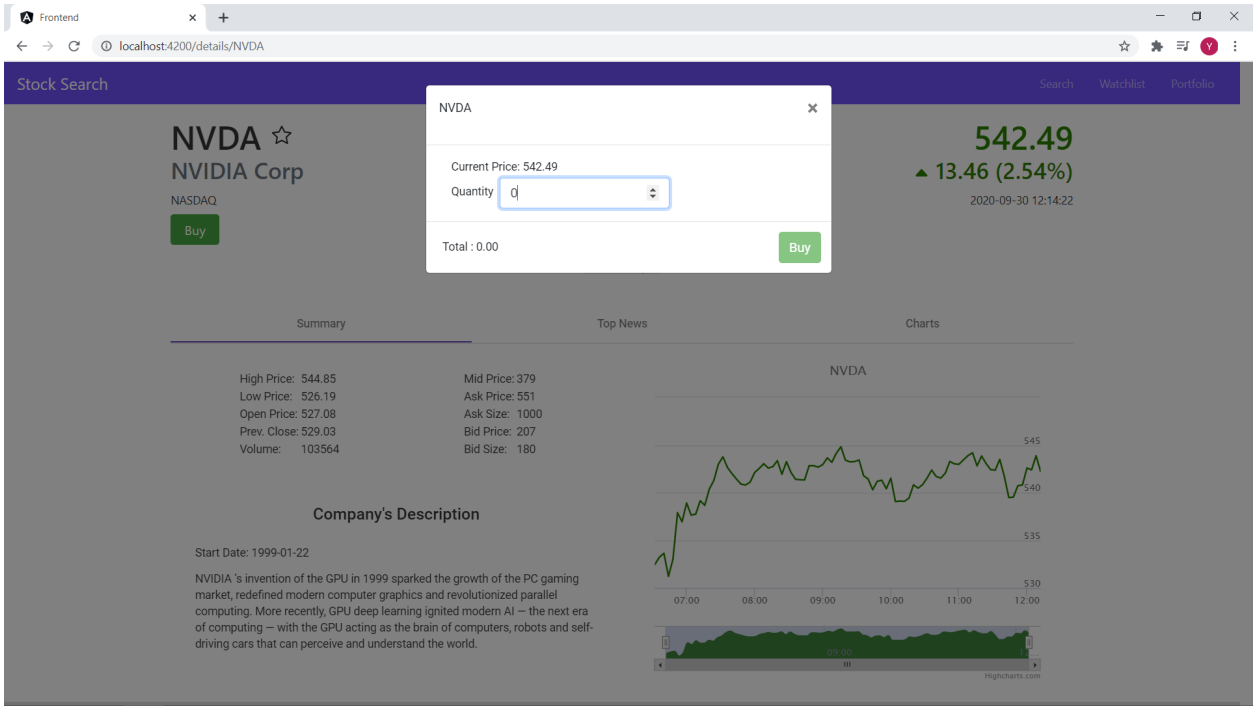


Figure 3.4: Buy button disabled for invalid input

As described, the page contains two major panels:

1) **Stock Details**: This panel displays all the values mentioned in **Table 3.1**. Last Timestamp should be only displayed beside the Market status if "Market is Close".

Fields	Sample Values	API reference
Ticker	NVDA	From table 4.1, use 'ticker' key
Company's Name	Nvidia	From table 4.1, use 'name' key
Exchange Code	NASDAQ	From table 4.1, use 'exchangeCode' key
Last Price	108.22	From table 4.2, use 'last' key
Change	1.1	Calculation is **'last' – 'prevClose'** from table 4.2
Change Percentage	1.03%	Calculation is **'Change' * 100 / 'prevClose'** from table 4.2
Current Timestamp	2020-09-24 16:26:27	Use the Date() function in JavaScript to display current time.
Market Status	Open/Close	Section 4.2
Last Timestamp	2020-09-24 13:00:00	From Table 4.2, use value available for key 'timestamp'.

Table 3.1 – Stock Details

2) **Material Tabs** – This component helps users see different stock data on the same page, and the tabs content is detailed in the following sections.

3.2.2 Summary Tab

It contains summary of the stock, which includes:

- Details are as following:
  - If market is Open: Display all the fields mentioned in **Table 3.2** and **Table 3.3**, as shown in **Figure 3.2**.
  - If market is Closed: Display all the fields mentioned in **Table 3.2**, as shown in **Figure 3.1**.
- Company's Description: values for 'startDate' and 'description' key from **Table 4.1.**
- Chart for the last Working Day:
  - If market is Open: Show stock chart available in Highcharts for that day.
  - If market is Closed: Show stock chart for the day when the market was closed. (i.e., last working day).

Fields	Sample Values	API reference
High Price	110.25	From table 4.2, use 'high' key
Low Price	105	From table 4.2, use 'low' key
Open Price	105.17	From table 4.2, use 'open' key
Prev. Close	107.12	From table 4.2, use 'close' key
Volume	165893169	From table 4.2, use 'volume' key

Table 3.2: Fields used inside Summary Tab (Part 1)

Fields	Sample Values	API reference
Mid Price	110.1	From Table 4.2, use 'mid' key
Ask Price	110.29	From Table 4.2, use 'askPrice' key
Ask Size	200	From Table 4.2, use 'askSize' key
Bid Price	110.35	From Table 4.2, use 'bidPrice' key
Bid Size	200	From Table 4.2, use 'bidSize' key

Table 3.3: Fields used inside Summary Tab (Part 2)

3.2.3 Top News Tab

This tab shows top news for the given stock ticker symbol from News API (see **Figure 3.5** and **Figure 3.6**).

- Show cards which contains Image and Title.
  - For Image use 'urlToImage' key from **Table 4.6**.
  - For Title use 'title' key from **Table 4.6**.

- When clicked on card, open a Modal window as shown in **Figure 3.6**. For details regarding Modal check section 5.3. Modal contains all the fields mentioned in **Table 3.4**.
- User can share the news articles on Twitter and Facebook. For details on how to use it, check **Section 4.1.3**. Twitter and Facebook should open in a new browser tab, if clicked.
    - In Twitter, it should create a post having following content:
        - Title of the news article
        - URL of the news article.
    - In Facebook, it should create a post, which contains URL of the news Article.
- Inside modal when user clicks on '***here***' in '*For more details click* ***here***', it should open the URL for the article in a new browser tab.

Fields	API reference
Publisher	From Table 4.6, use 'source' key.
Published Date	From Table 4.6, use 'publishedAt' key.
Title	From Table 4.6, use 'title' key.
Description	From Table 4.6, use 'description' key.
URL	From Table 4.6, use 'url' key.

Table 3.4: Fields used inside modal of Top News Tab

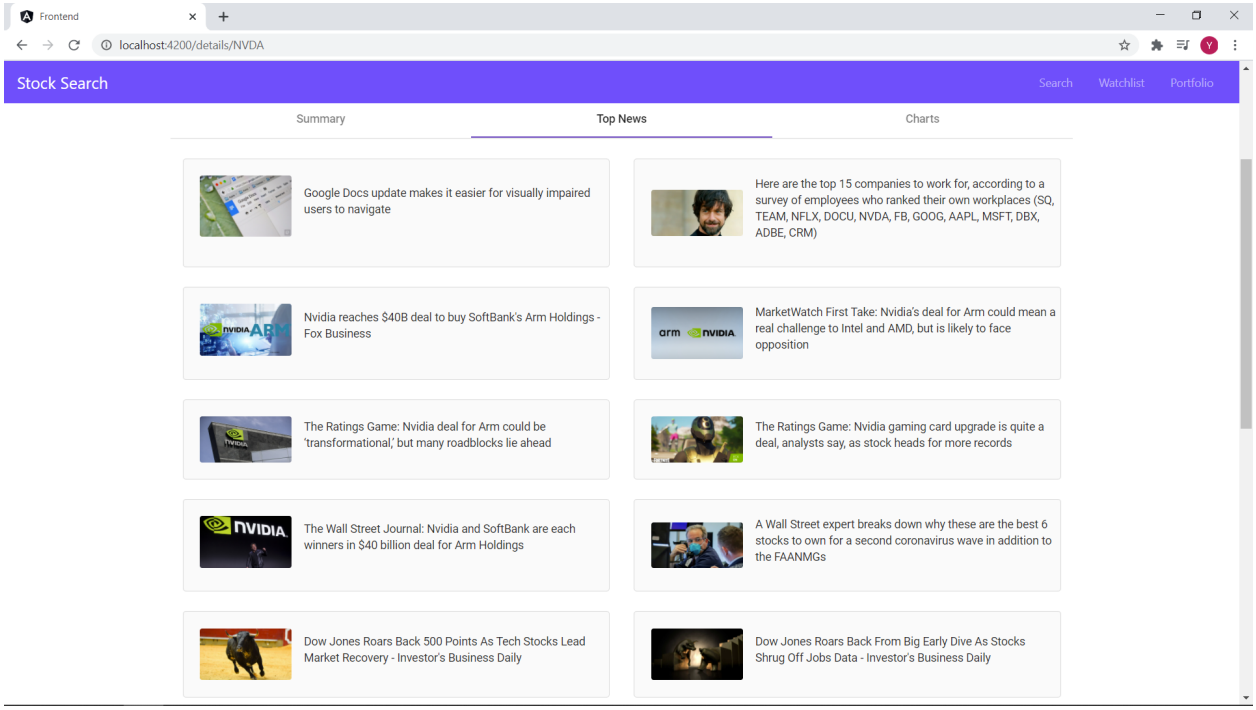


Figure 3.5: Top News Tab overview

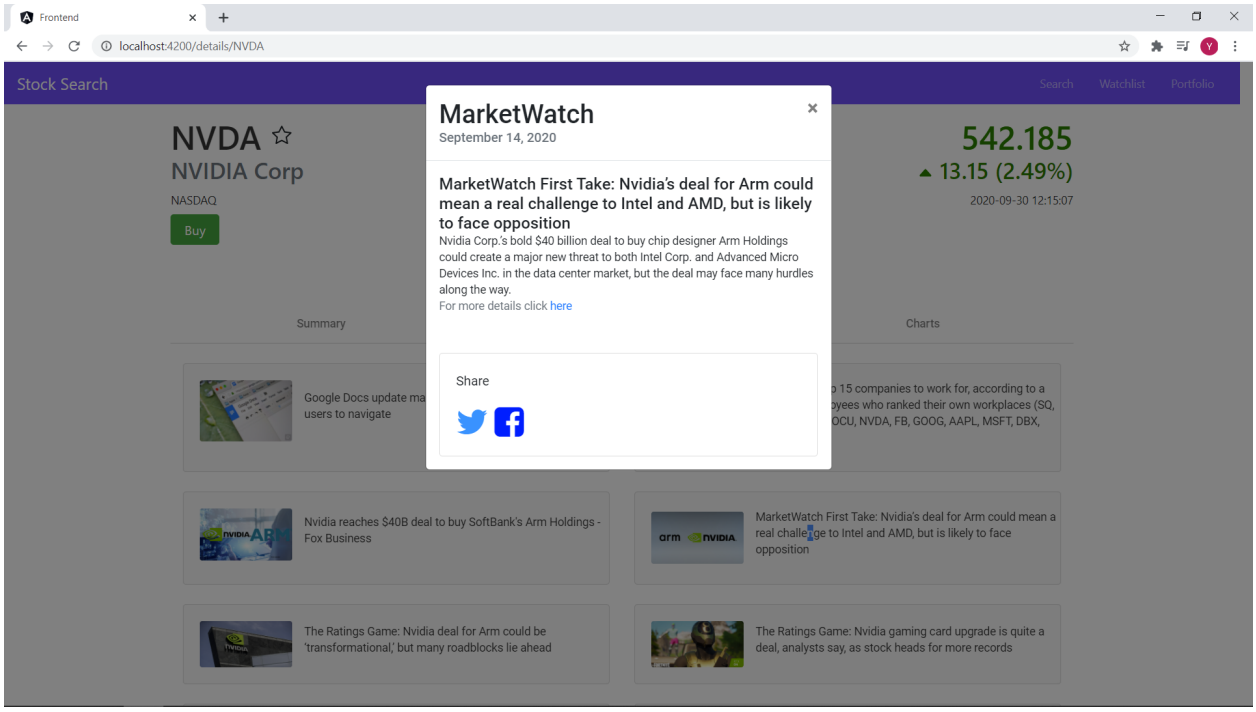


Figure 3.6: Top News Detailed Modal overview.

3.2.4 Charts Tab

This tab uses HighCharts to display historical stock market data on the related stock.

- See **Figure 3.7** for reference. For more details regarding Highcharts see section 5.5.
- Display SMA and Volume by Price chart for data of the last 2 years.

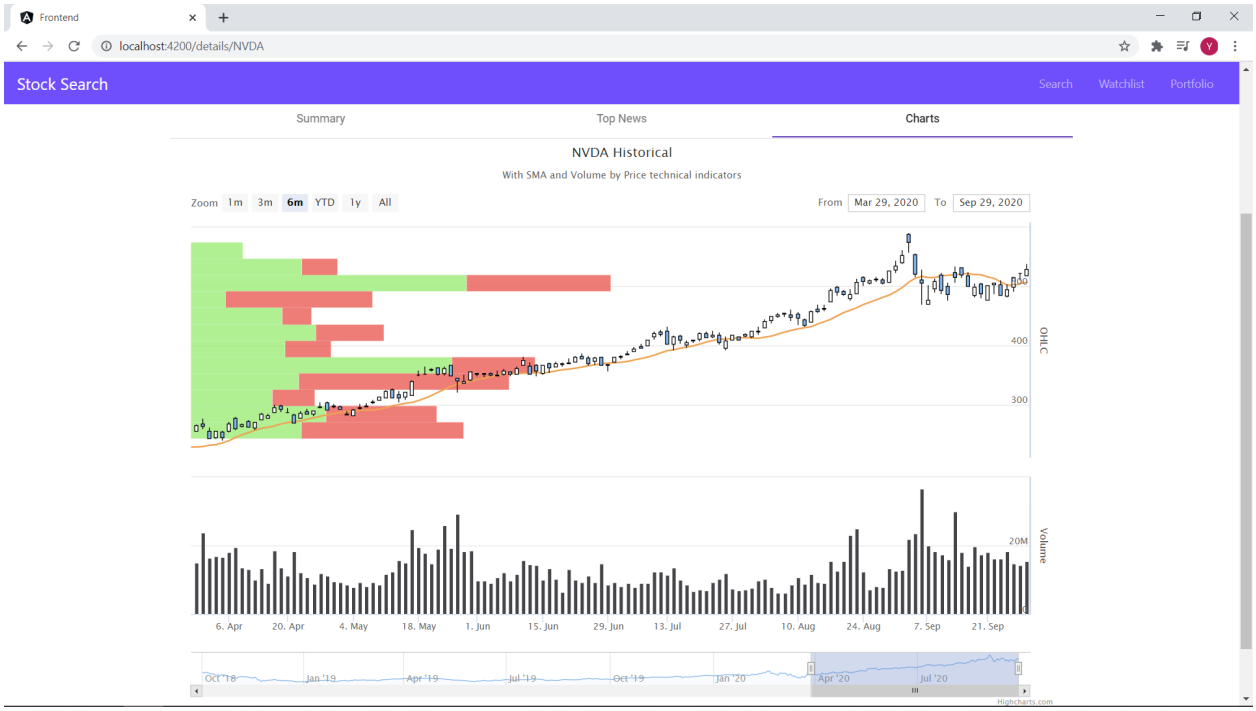


Figure 3.7: Charts tab overview.

Note: As user mentioned before, that user need to select one from the suggestions before clicking the search button. If user manually edits the /details/<ticker> route and if the <ticker> entered by user is invalid, then you need to display an alert on that page instead of details.

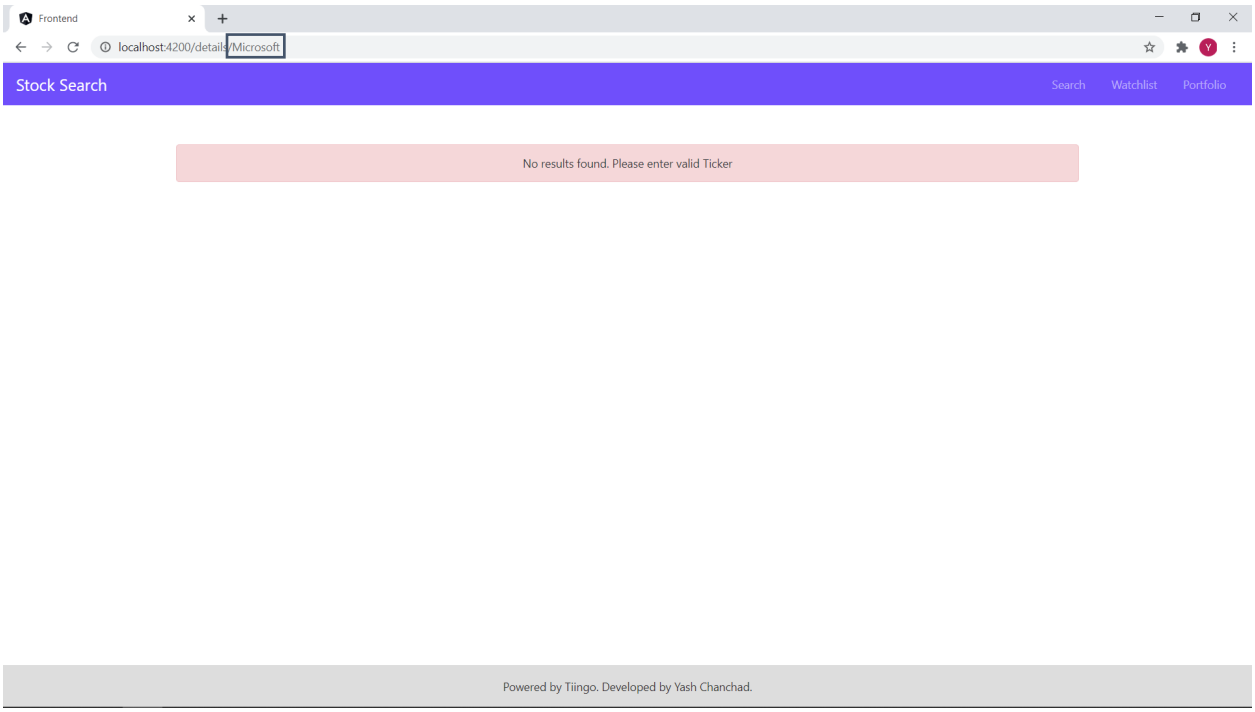


Figure 3.8: Error Alert.

Important note: Data mentioned in Stock Details section and Summary Tab, should auto-update every 15 seconds.

3.3 Watchlist Menu

This menu will display all the stocks that are added to the watchlist by the user. This watchlist will be maintained in local storage of the application. For more details on local storage, see section 5.4. (see **Figure 3.9**)

- If the change is positive, the color of the 'last', 'change' and 'changePercentage' keys should be green
- If the change is negative, the color of the 'last', 'change' and 'changePercentage' keys should be red
- If there is no change, the color of the 'last', 'change' and 'changePercentage' keys should be black.
- When clicked on close button at present on the right-top corner of the card, it should remove it from the watchlist and display an updated watchlist.
- When clicked on card, it should open the details route of that ticker (stock).
- If watchlist is empty, it should display the alert as shown in **Figure 3.10**.

'last' key should be used from **Table 4.2**. 'change' and 'changePercentage' should be calculated in a similar fashion as to the one shown in **Table 3.1**.

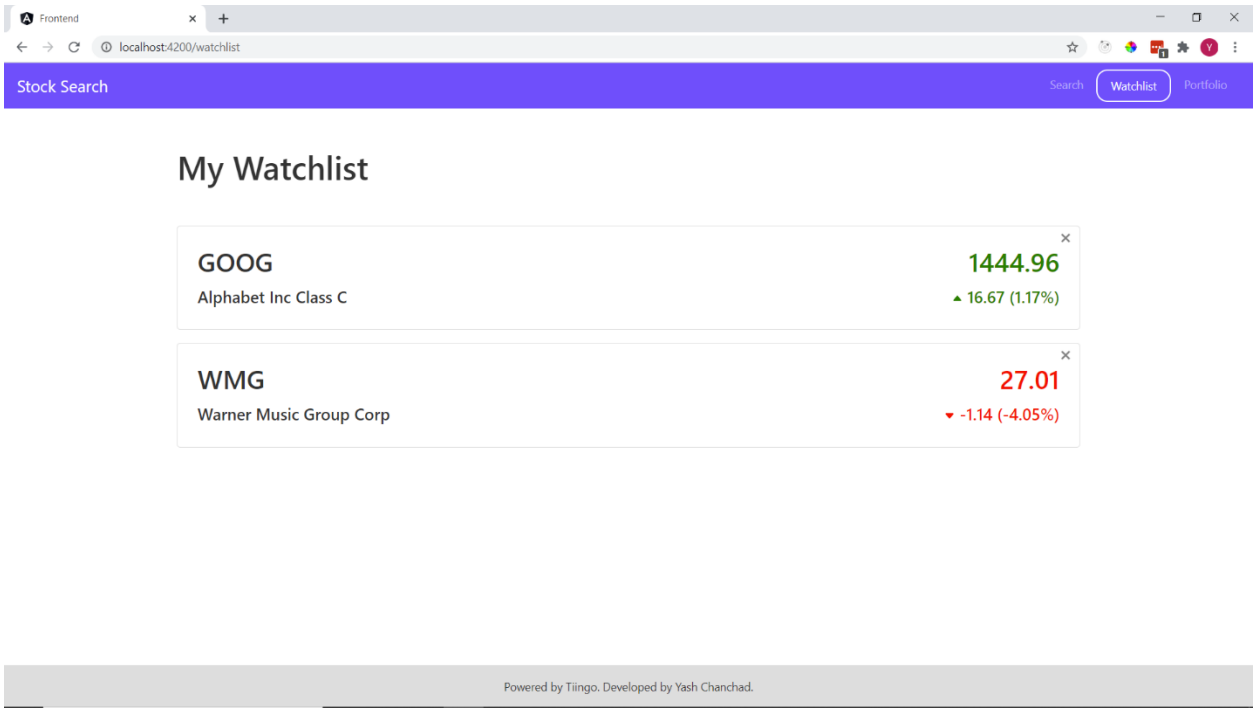


Figure 3.9: Watchlist menu page

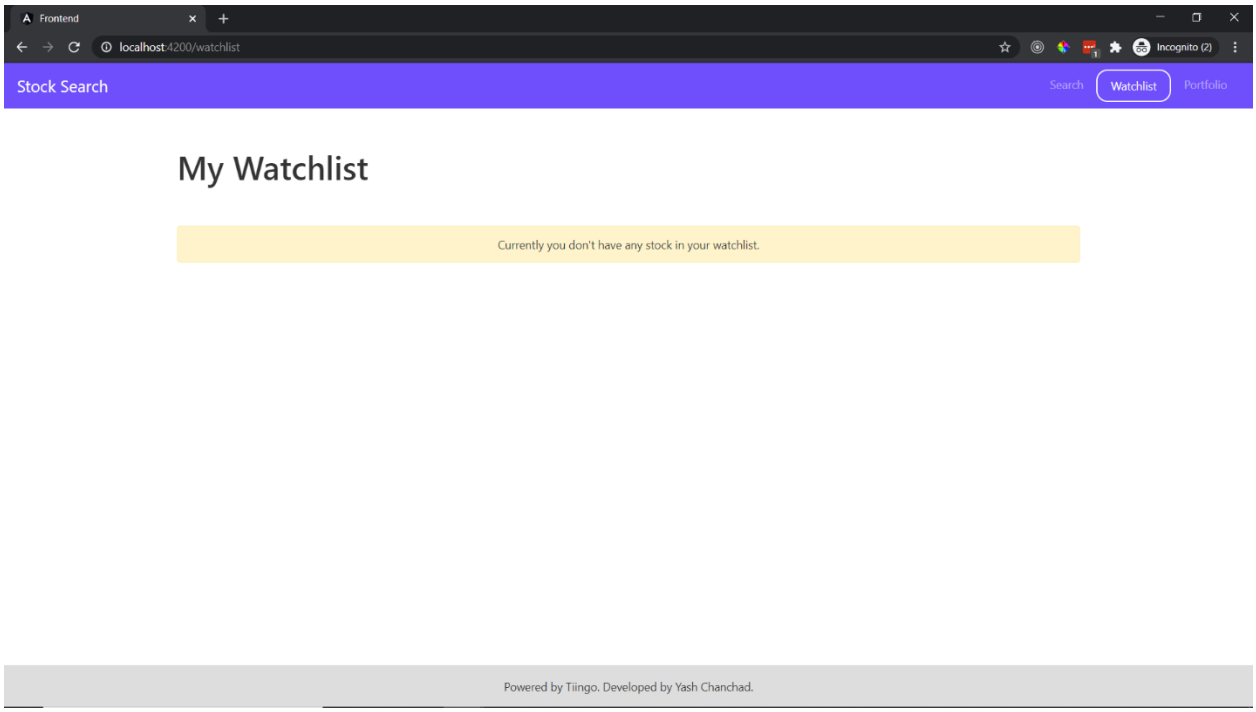


Figure 3.10: Watchlist Empty Alert

3.4 Portfolio Menu

This menu will display all the stocks that have been bought by the user (i.e. the current portfolio of the user). This portfolio will be maintained in local storage of the application. For more details on local storage, see section 5.4 and **Figure 3.11**. In particular:

- If the current rate is greater than the rate at which user bought it, then color of the 'Change', 'Current Price' and 'Current Total' keys should be green;
- If the change is negative, the color of the 'Change', 'Current Price' and 'Current Total' keys should be red;
- If there is no change, the color of the 'Change', 'Current Price' and 'Current Total' keys should be black;
- When clicking on **Buy** button, modal should open as shown in **Figure 3.13**. The Buy button inside modal should be disabled if the quantity entered by user is not valid as shown in **Figure 3.14**. Valid input should be greater than 0 and must be non-empty;
- When clicking on **Sell** button, modal should open as shown in **Figure 3.15 and 3.16**. Sell button inside modal should be disabled if the quantity entered by user is not valid. Input is Valid if, 0<input<=Quantity and must be non-empty. Quantity is described in **Table 3.5;**
- When clicked on card's header part, it should open the details route of that ticker (stock);
- If portfolio is empty, it should display the alert as shown in **Figure 3.12**.

'last' key should be used from Table 4.2 for 'Current Price'. 'Current Change' and 'Current Total' should be calculated as shown in **Table 3.5**.

Quantity	Total Number of stocks bought by user. It should be more than 0, else remove it from the portfolio local storage.
Total Cost	Total cost is sum of the total cost paid for all the purchases of the stock. For Example, if user has bought 10 stocks of AAPL in past, at the rate of 200, and today if user buys another 10 stocks of AAPL at the current price, i.e. 300, then the Total Cost for the user will be (10*200) + (10*300) = 5000. So Quantity is 20 and Total Amount is 5000.
Average Cost per Share	(Total Cost / Quantity)
Current Price	'last' key from the table 4.2
Change	(Average Cost per Share – Current Price) of the stock. Here, Current Price is 'last' key from the table 4.2
Market Value	(Current Price * Qty), here Current Price is 'last' key from **Table 4.2** and Qty is the number of stocks present in user's portfolio.

Table 3.5: Fields used in Portfolio Cards.

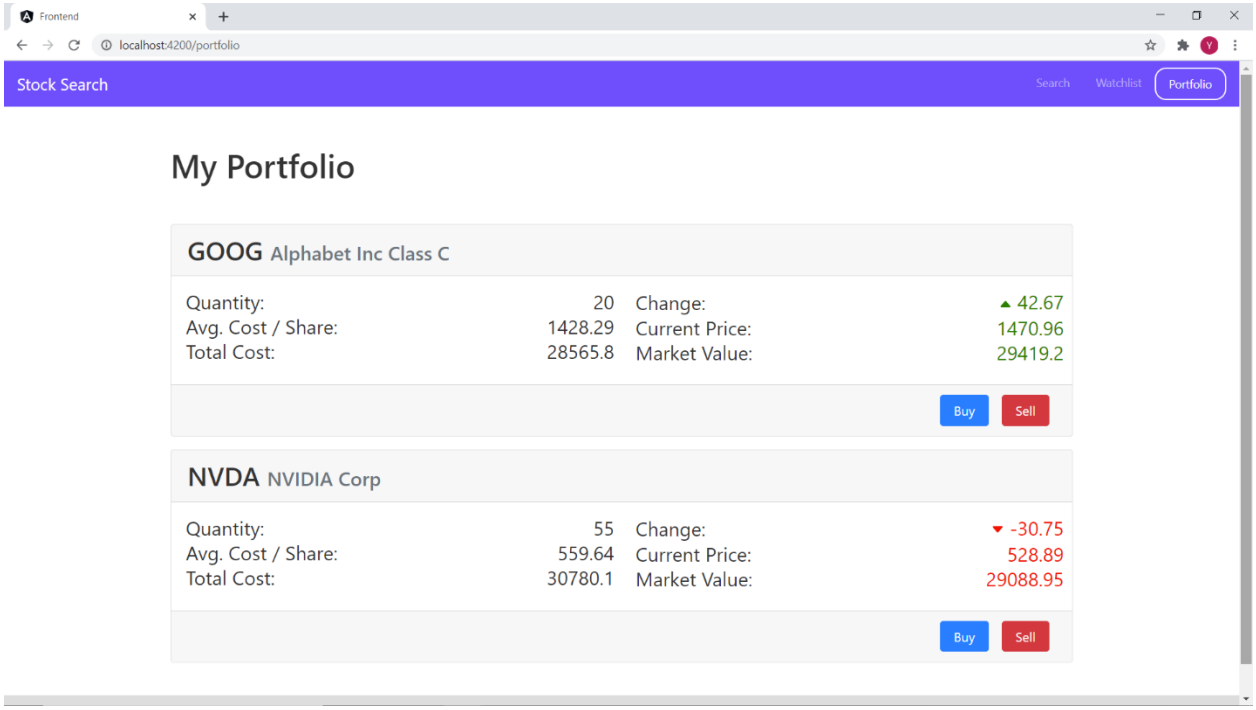


Figure 3.11: Portfolio

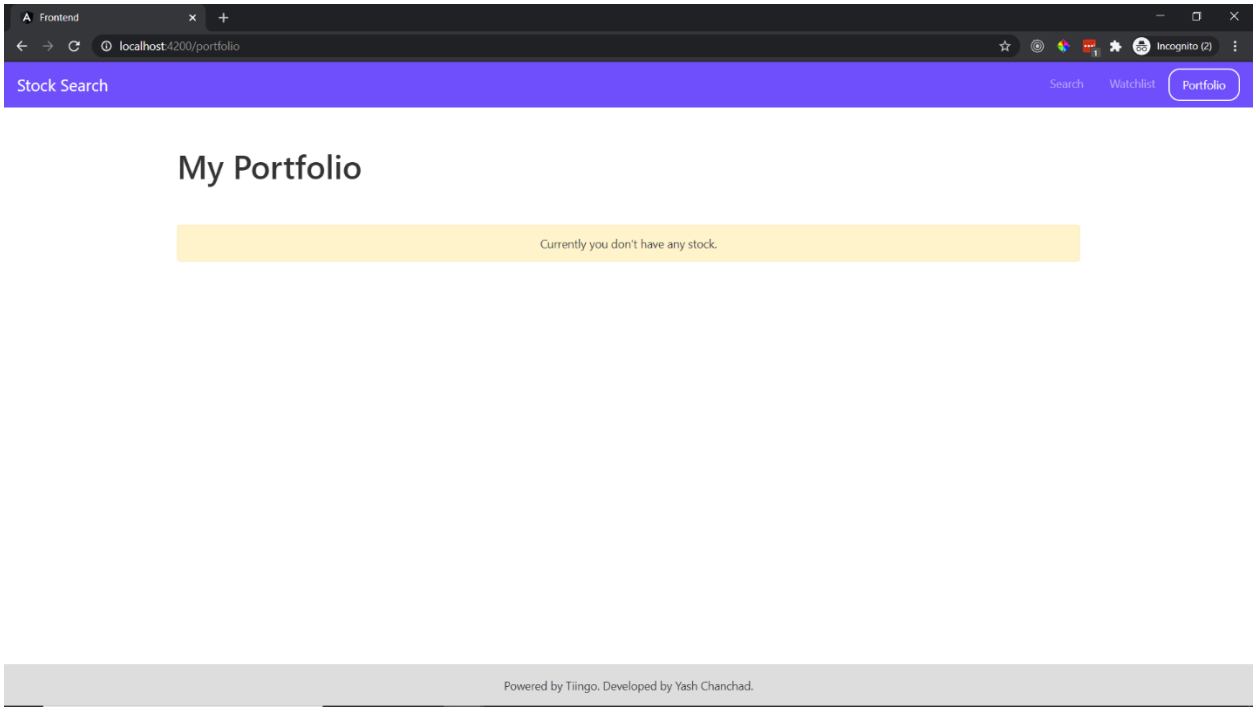


Figure 3.12: Portfolio Empty Alert

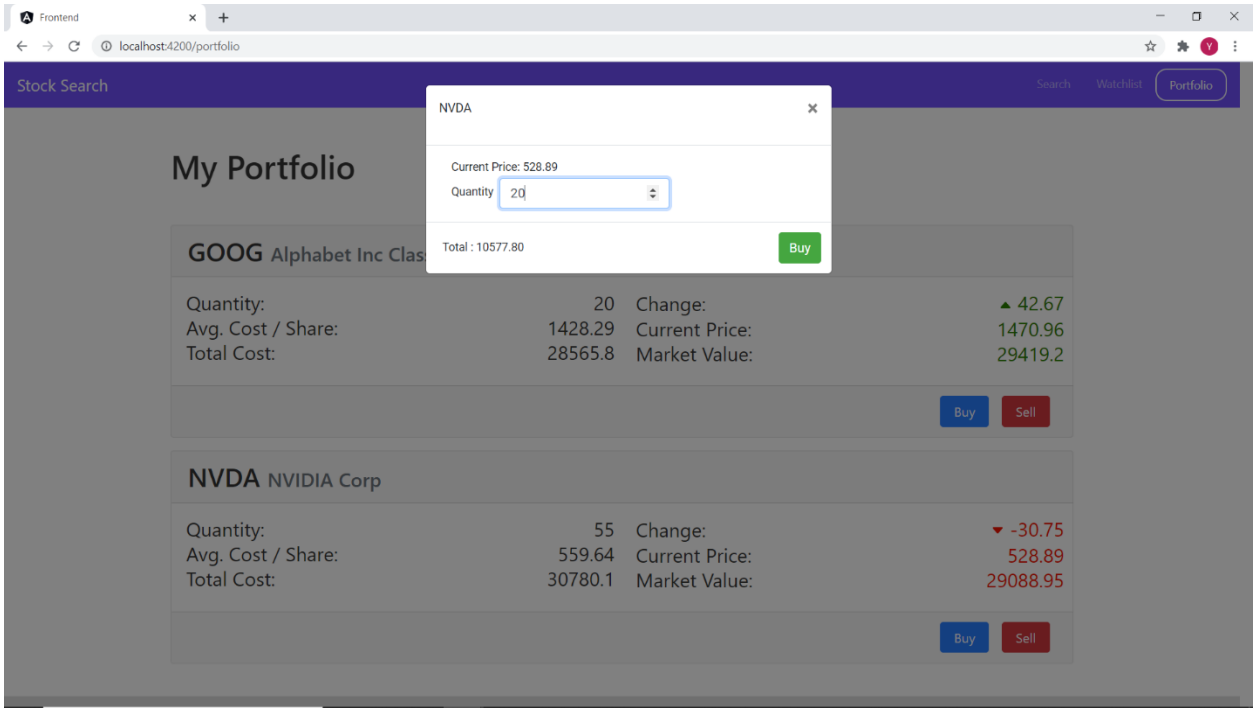


Figure 3.13: Modal for Buying Stock

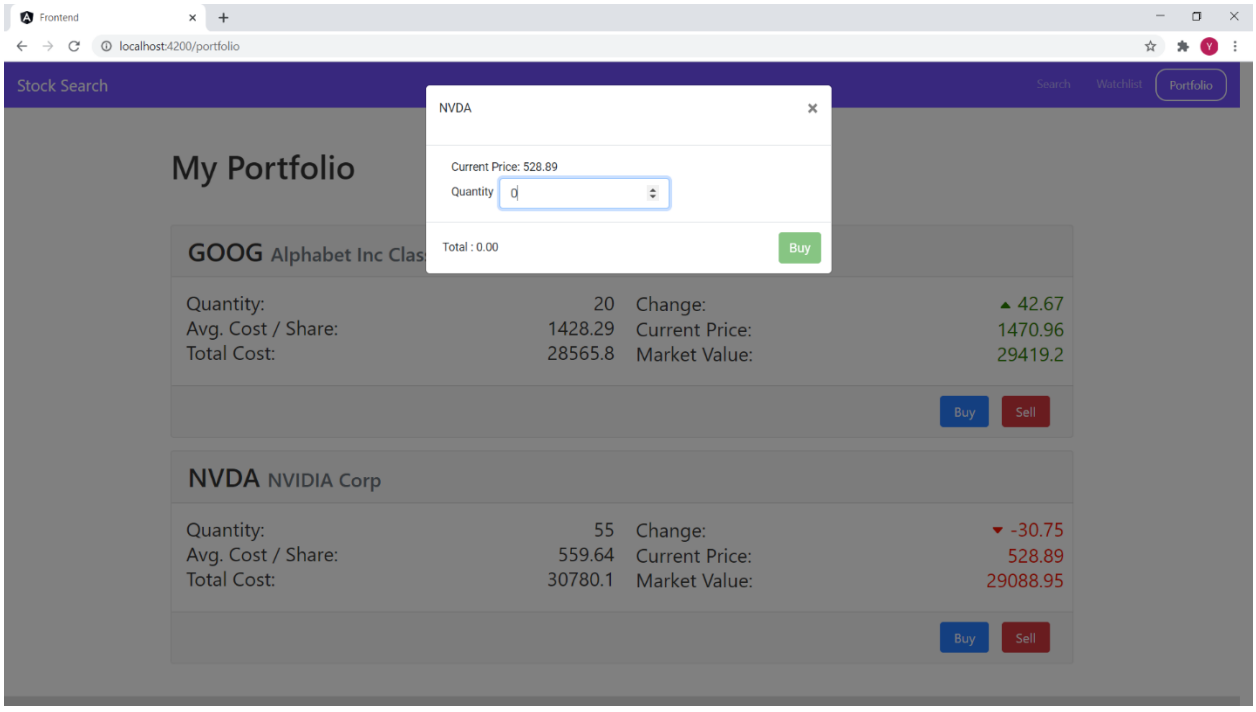


Figure 3.14: Input is invalid in Modal for Buying Stock

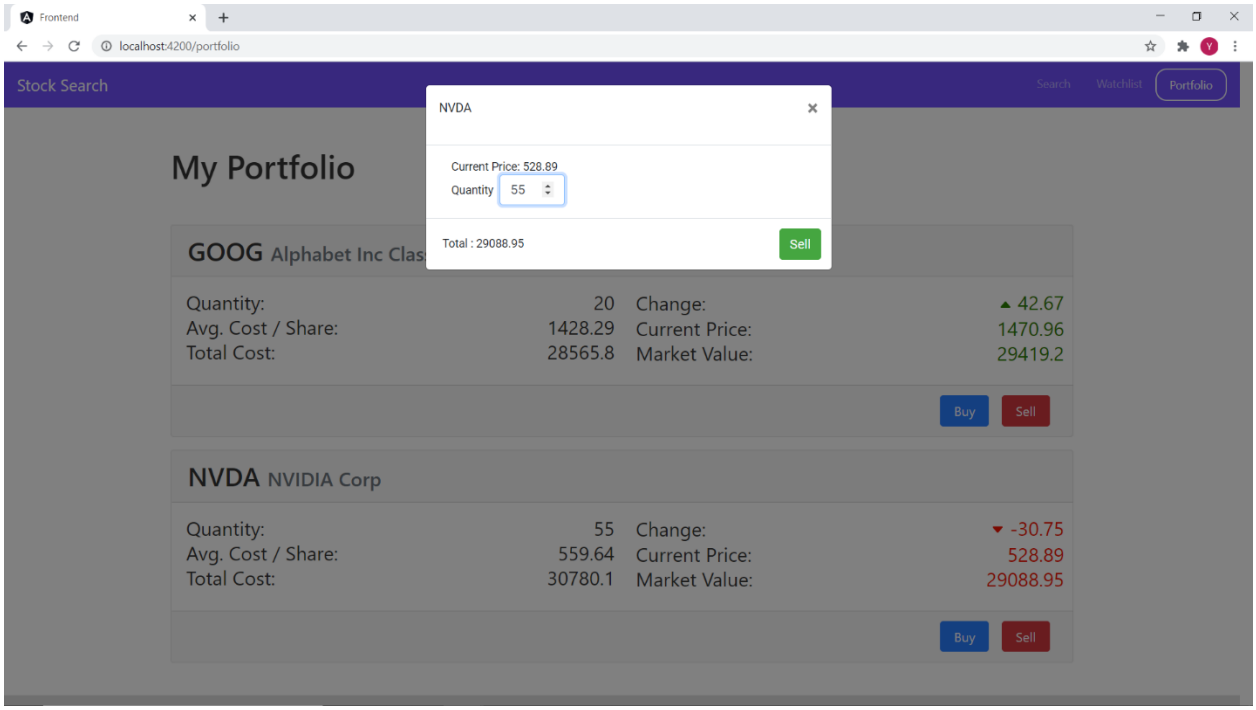


Figure 3.15: Modal for Selling Stock

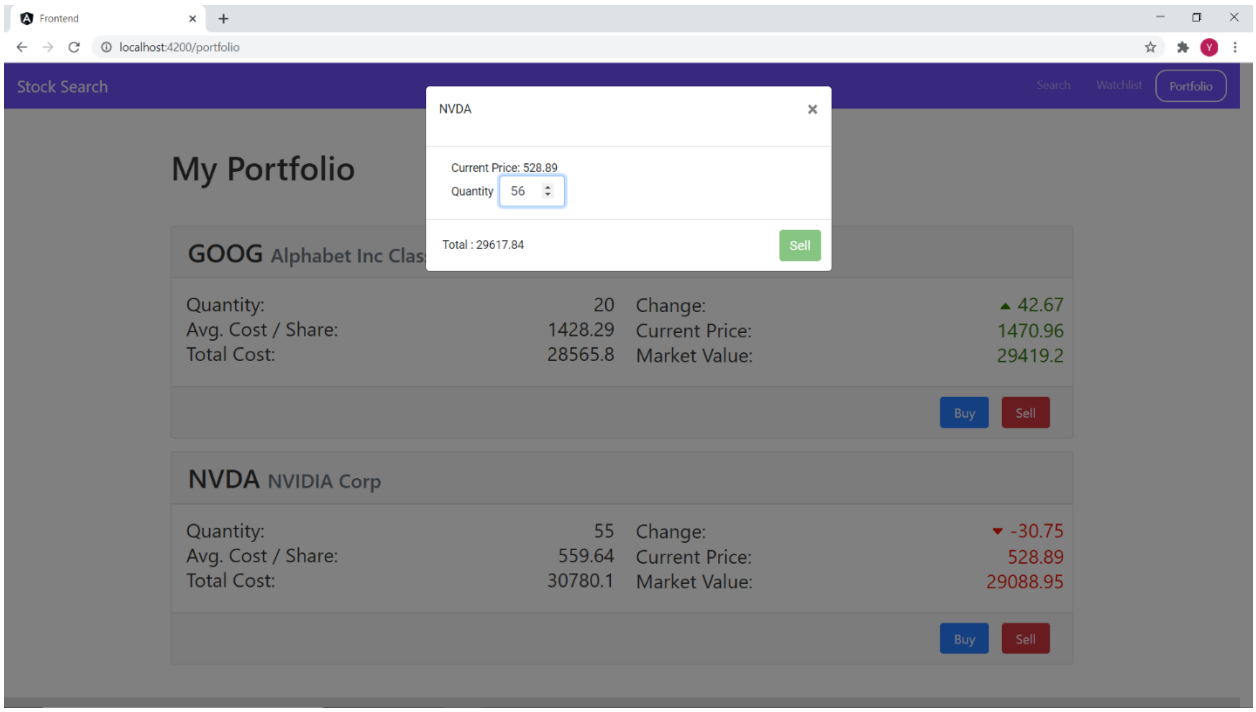
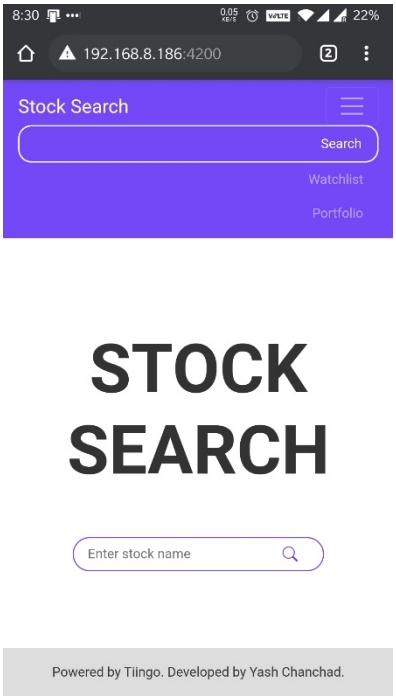


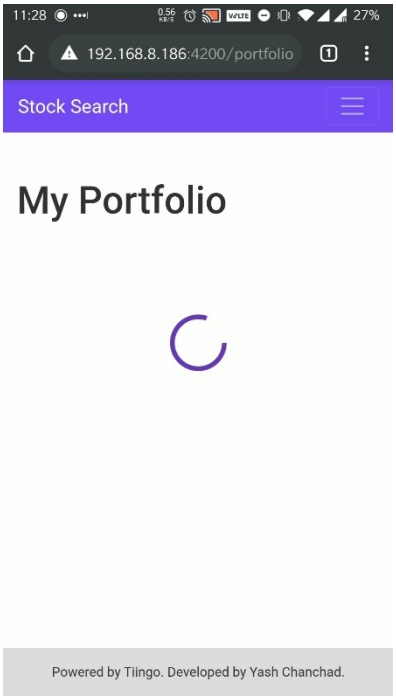
Figure 3.16: Input is invalid in Modal for Selling Stock

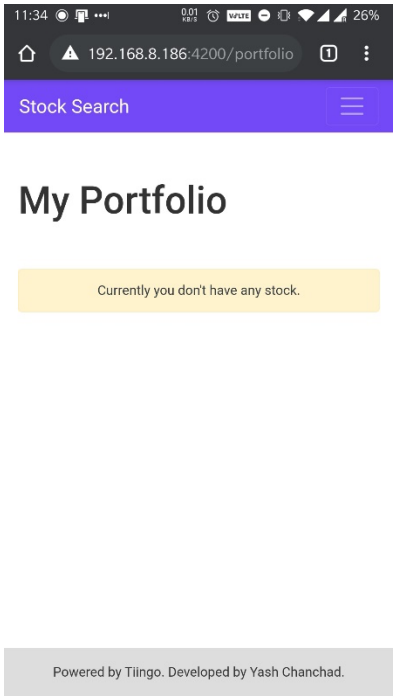
Note: Portfolio and Watchlist must be sorted in ascending order based on Ticker.

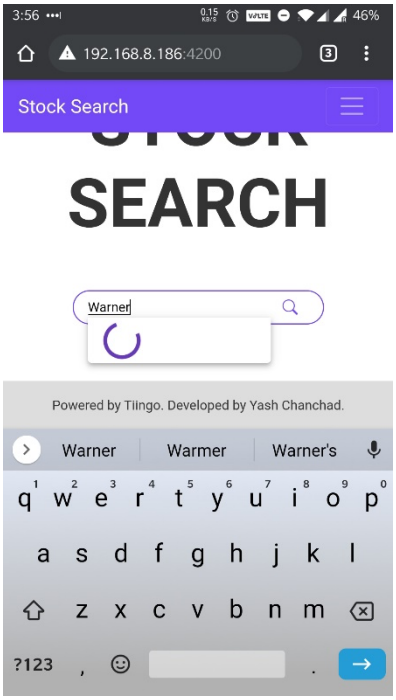
3.5 Responsive Design

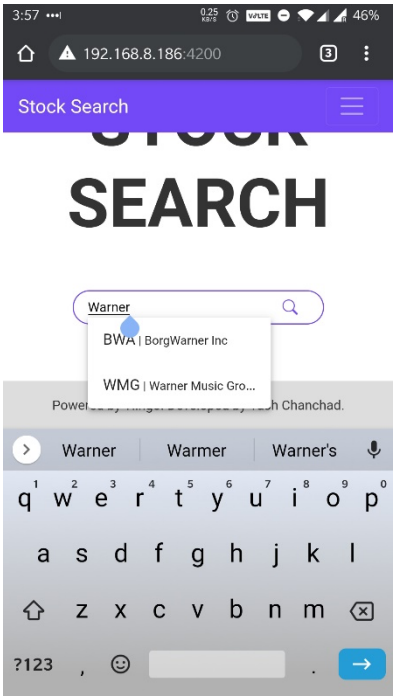
The following are a few snapshots of the web app opened with Google Chrome on a OnePlus mobile device.

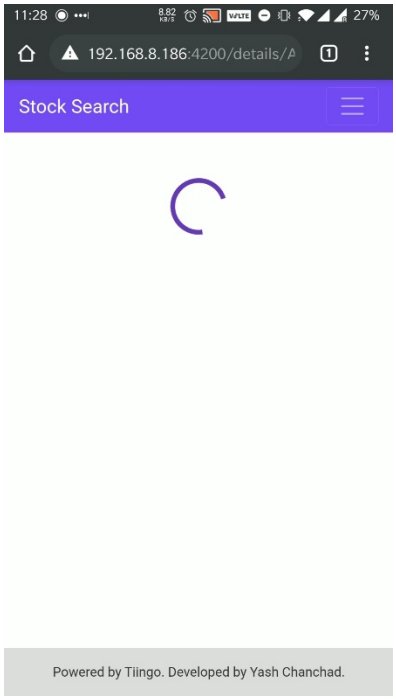






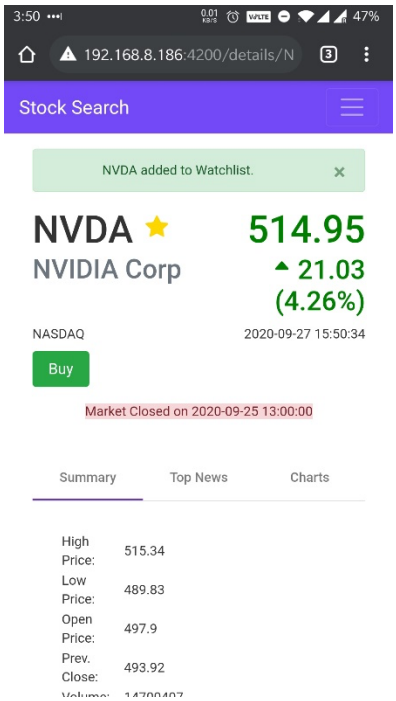


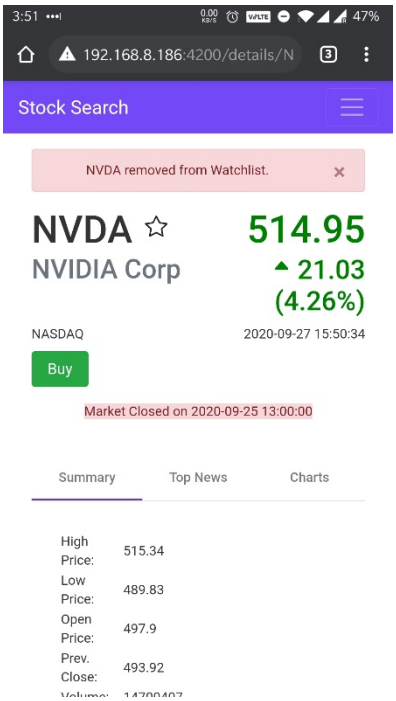



11:28
192.168.8.186:4200/details/A
Stock Search
Powered by Tiingo. Developed by Yash Chanchad.

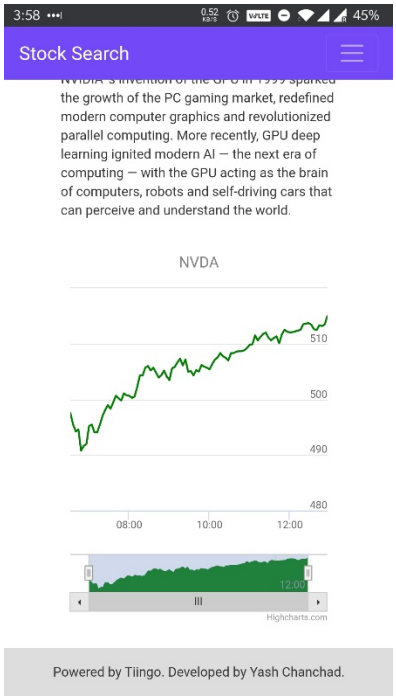

9:37
192.168.8.186:4200/details/M
Stock Search
MSFT
Microsoft Corp
208.65
0.83 (0.4%)
NASDAQ
2020-09-28 09:37:18
Buy
Market is Open
Summary
Top News
Charts
High Price: 212.55
Low Price: 208.14
Open Price: 211.02
Prev. Close: 207.82
Volume: 197007
Mid Price: 209.445
Ask Price: 210.38
Ask Size: 100
Bid Price: 208.51
Bid Size: 100
Company's Description
Start Date: 1986-03-13

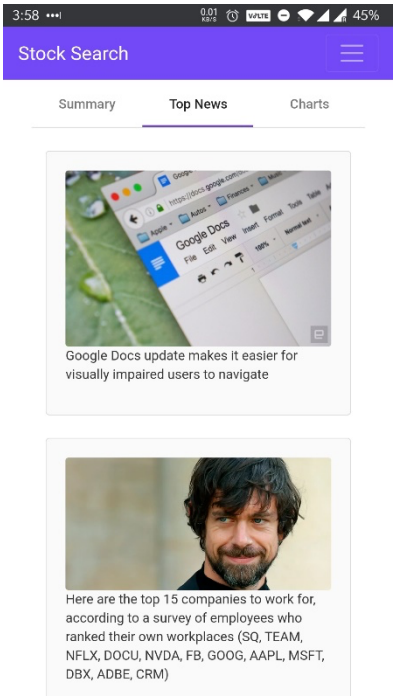

3:50
192.168.8.186:4200/details/N
Stock Search
NVDA
NVIDIA Corp
514.95
21.03 (4.26%)
NASDAQ
2020-09-27 15:50:34
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts
High Price: 515.34
Low Price: 489.83
Open Price: 497.9
Prev. Close: 493.92
Volume: 14700407

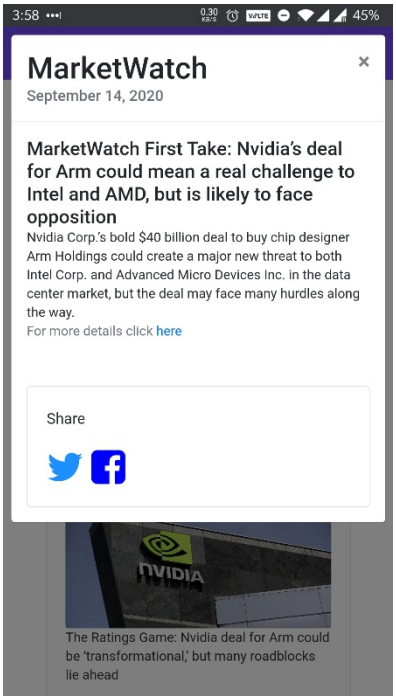

3:50
192.168.8.186:4200/details/N
Stock Search
NVDA added to Watchlist.
NVDA
NVIDIA Corp
514.95
21.03 (4.26%)
NASDAQ
2020-09-27 15:50:34
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts
High Price: 515.34
Low Price: 489.83
Open Price: 497.9
Prev. Close: 493.92


3:51
192.168.8.186:4200/details/N
Stock Search
NVDA removed from Watchlist.
NVDA
NVIDIA Corp
514.95
21.03 (4.26%)
NASDAQ
2020-09-27 15:50:34
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts
High Price: 515.34
Low Price: 489.83
Open Price: 497.9
Prev. Close: 493.92

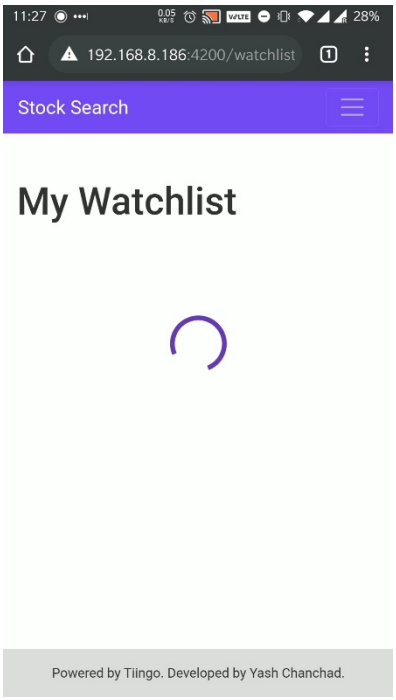

3:58
Stock Search
Company's Description
Start Date: 1999-01-22
NVIDIA 's invention of the GPU in 1999 sparked the growth of the PC gaming market, redefined modern computer graphics and revolutionized parallel computing. More recently, GPU deep learning ignited modern AI — the next era of computing — with the GPU acting as the brain of computers, robots and self-driving cars that can perceive and understand the world.
NVDA
510
500
490
480
08:00
10:00
12:00


3:58
45%
Stock Search
the growth of the PC gaming market, redefined modern computer graphics and revolutionized parallel computing. More recently, GPU deep learning ignited modern AI — the next era of computing — with the GPU acting as the brain of computers, robots and self-driving cars that can perceive and understand the world.
NVDA
510
500
490
480
08:00
10:00
12:00
Highcharts.com
Powered by Tiingo. Developed by Yash Chanchad.

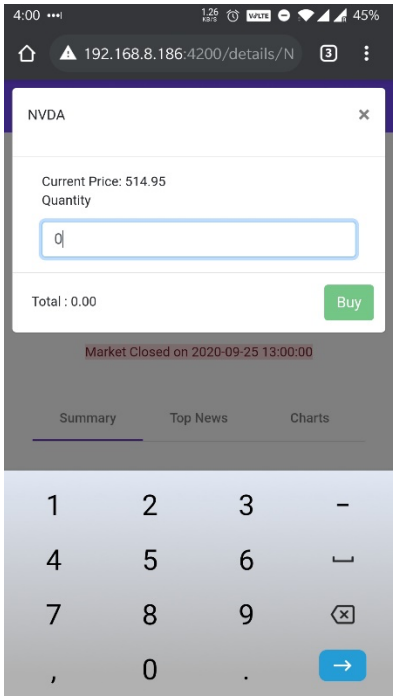

3:58
45%
Stock Search
Summary
Top News
Charts
Google Docs
Google Docs update makes it easier for visually impaired users to navigate
Here are the top 15 companies to work for, according to a survey of employees who ranked their own workplaces (SQ, TEAM, NFLX, DOCU, NVDA, FB, GOOG, AAPL, MSFT, DBX, ADBE, CRM)

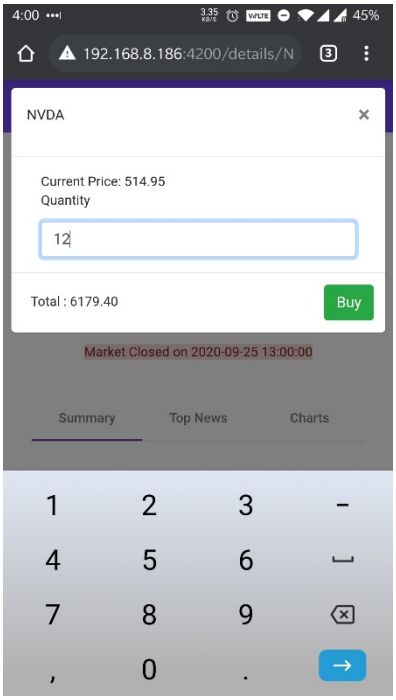

3:58
45%
MarketWatch
September 14, 2020
MarketWatch First Take: Nvidia's deal for Arm could mean a real challenge to Intel and AMD, but is likely to face opposition
Nvidia Corp.'s bold $40 billion deal to buy chip designer Arm Holdings could create a major new threat to both Intel Corp. and Advanced Micro Devices Inc. in the data center market, but the deal may face many hurdles along the way.
For more details click here
Share
NVIDIA
The Ratings Game: Nvidia deal for Arm could be 'transformational,' but many roadblocks lie ahead

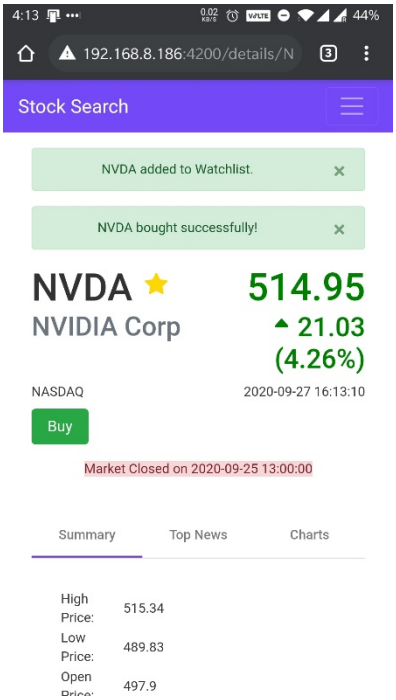

3:51
47%
Stock Search
Summary
Top News
Charts
NVDA Historical
With SMA and Volume by Price technical indicators
Zoom 1m 3m 6m YTD 1y All
From Mar 25, 2020 To Sep 25, 2020
500
400
300
OHLC
100M
0M
Volume
May '20
Jul '20
Sep '20

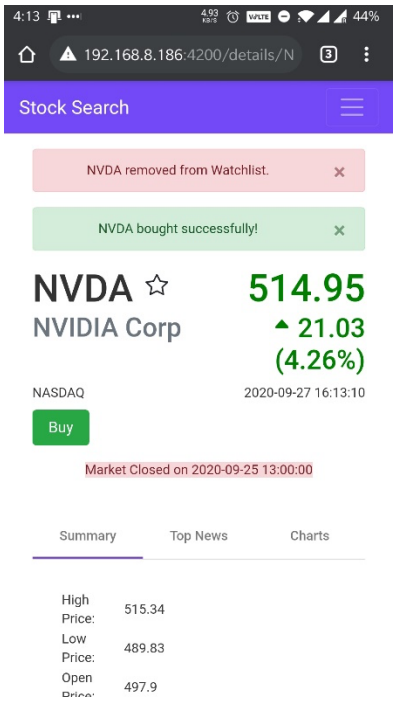

11:27
28%
192.168.8.186:4200/watchlist
Stock Search
My Watchlist
Powered by Tiingo. Developed by Yash Chanchad.


3:57
46%
192.168.8.186:4200/watchlist
Stock Search
My Watchlist
NVDA
514.95
NVIDIA Corp
21.03 (4.26%)
WMG
27.01
Warner Music Group Corp
-1.14 (-4.05%)
Powered by Tiingo. Developed by Yash Chanchad.

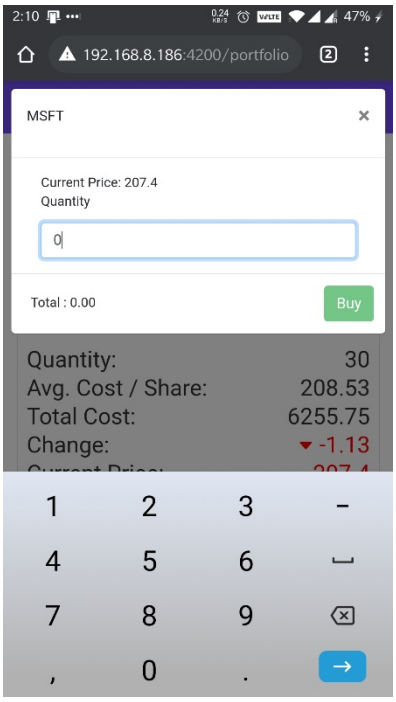

4:00
192.168.8.186:4200/details/N
NVDA
Current Price: 514.95
Quantity
0
Total : 0.00
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts

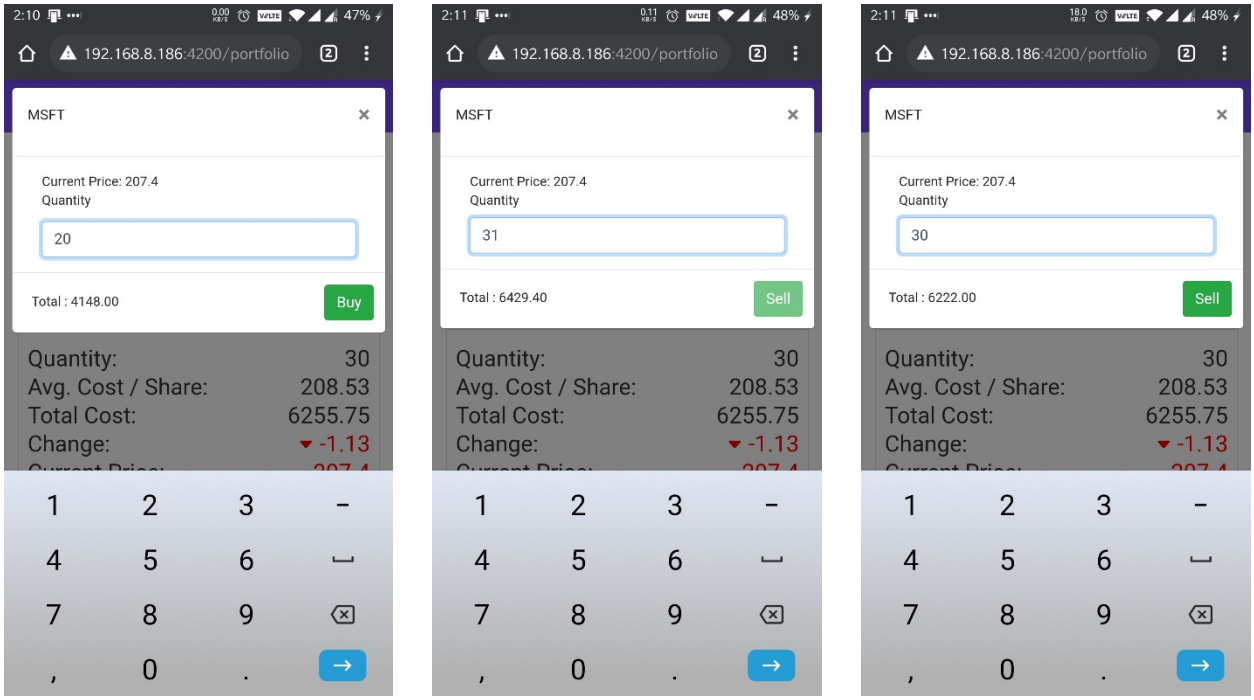

4:00
192.168.8.186:4200/details/N
NVDA
Current Price: 514.95
Quantity
12
Total : 6179.40
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts


4:13
192.168.8.186:4200/details/N
Stock Search
NVDA added to Watchlist.
NVDA bought successfully!
NVDA
NVIDIA Corp
514.95
21.03
(4.26%)
NASDAQ
2020-09-27 16:13:10
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts
High Price: 515.34
Low Price: 489.83
Open Price: 497.9


4:13
192.168.8.186:4200/details/N
Stock Search
NVDA removed from Watchlist.
NVDA bought successfully!
NVDA
NVIDIA Corp
514.95
21.03
(4.26%)
NASDAQ
2020-09-27 16:13:10
Buy
Market Closed on 2020-09-25 13:00:00
Summary
Top News
Charts
High Price: 515.34
Low Price: 489.83
Open Price: 497.9


2:10
192.168.8.186:4200/portfolio
Stock Search
My Portfolio
MSFT Microsoft Corp
Quantity: 30
Avg. Cost / Share: 208.53
Total Cost: 6255.75
Change: -1.13
Current Price: 207.4
Market Value: 6222
Buy
Sell
Powered by Tiingo. Developed by Yash Chanchad.


2:10
192.168.8.186:4200/portfolio
MSFT
Current Price: 207.4
Quantity
0
Total : 0.00
Buy
Quantity: 30
Avg. Cost / Share: 208.53
Total Cost: 6255.75
Change: -1.13



You must watch the video carefully to see how the page looks like on mobile devices. All functions must work on mobile devices.

One easy way to test for mobile devices is to use Google Chrome Responsive Design Mode and Safari Develop – User Agent menu.

3.5 Navbar

The Navigation bar must be present on top of the page, and visible at all times as shown in all the figures above. You can use Bootstrap to create a navbar. It consists of following menu options:

1. Search
2. Watchlist
3. Portfolio

3.6 Footer

The Footer must be present at the end of each page, as shown in above figures. It should contain following line:

"Powered by Tiingo. Developed by <student's name>"

Here <student's name> must be replace with your name.

4. API's description

4.1 Tiingo API calls, similar to Homework 6

In this homework, we will use the Tiingo API. A comprehensive reference about this API is available at:

1) https://api.tiingo.com/documentation/end-of-day
2) https://api.tiingo.com/documentation/iex
3) https://api.tiingo.com/documentation/utilities/search

4.1.1 Company's Description

For **Company's Description**, use the following API. For more details refer **Figure 4.1**:
https://api.tiingo.com/tiingo/daily/<ticker>?token=<APIKeyTiingo>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: MSFT
- Token: The API access Token. It is private, please do not share with anyone. See Homework 6.

An example URL constructed from the parameters will look similar to this:

https://api.tiingo.com/tiingo/daily/AAPL?token=12PrIvA32tEmYtEmpToKeN23

Response:

Response Keys	**Details**
ticker	Ticker symbol.
name	Company's Name
description	Company's Description
startDate	Company's Start Date
exchangeCode	Company's Exchange Code

Table 4.1: Response received for Company's Meta Data API call.

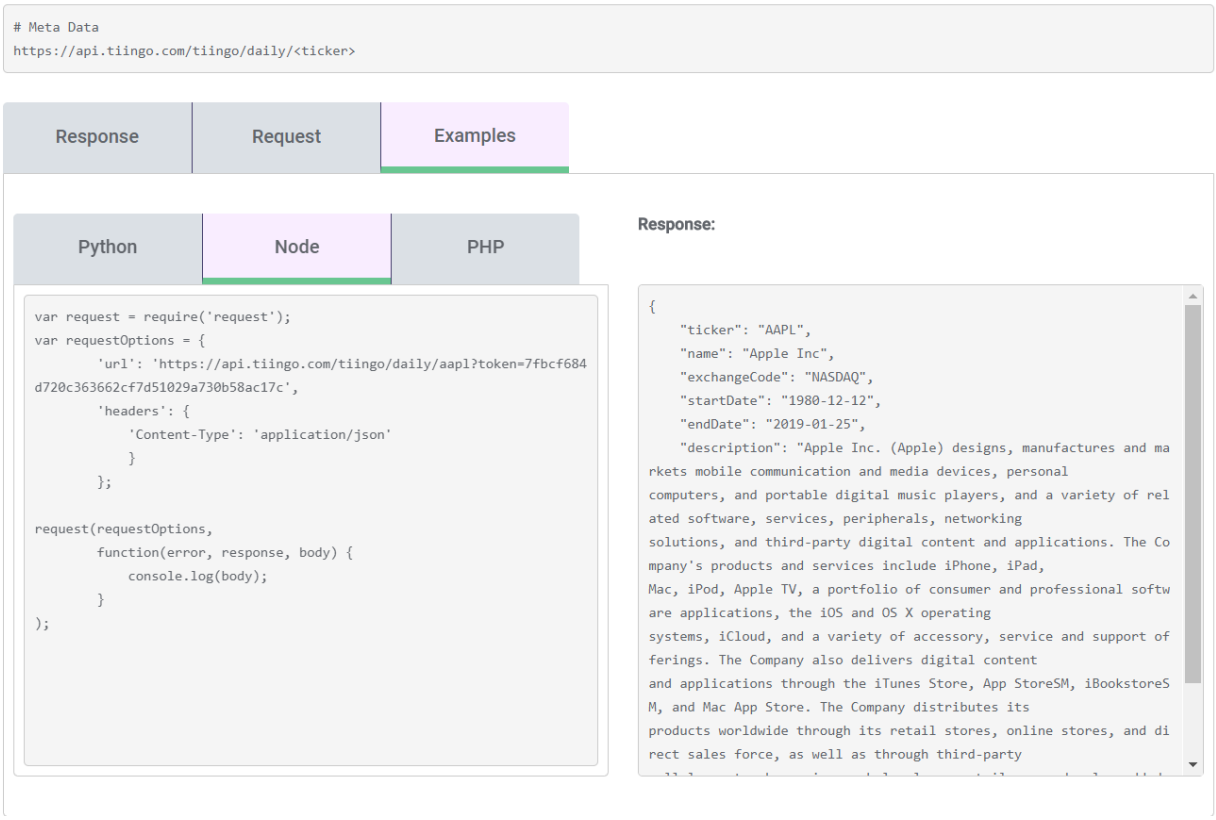


Figure 4.1: Details regarding Company's Meta Data

4.1.2 Company's Latest Price of the stock

For **Company's Latest Price**, use the following API. For more details refer Figure 4.2:
https://api.tiingo.com/iex/?tickers=<ticker>&token=<APIKeyTiingo>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: AAPL
- Token: The API access Token. It is private, please do not share with anyone.

An example URL constructed from the parameters will look similar to:

https://api.tiingo.com/iex?tickers=AAPL&token=12PrIvA32tEmYtEmpToKeN23

Response:

Response Keys	Details
ticker	Ticker symbol.
timestamp	Timestamp of the data.
last	Company's latest price based on timestamp
prevClose	Company's previous closing based on timestamp
open	Company's open price based on timestamp
high	Company's high price based on timestamp
low	Company's low price based on timestamp
mid	Company's mid price based on timestamp
volume	Company's volume based on timestamp
bidSize	Company's bid size based on timestamp
bidPrice	Company's bid price based on timestamp
askSize	Company's ask size based on timestamp
askPrice	Company's ask price based on timestamp

Table 4.2: Response received for Company's Latest Price of the Stock.

Value of 'mid', 'bidPrice', 'bidSize', 'askPrice', 'askSize' will be null when market is closed. Value of mid can be null even when the market is open, if this happens then you should display '-' instead of null. Market Status must be open if the difference between current Timestamp (current Timestamp will be of the created using new Date() in javascript) and 'timestamp' key is less than 60 seconds.

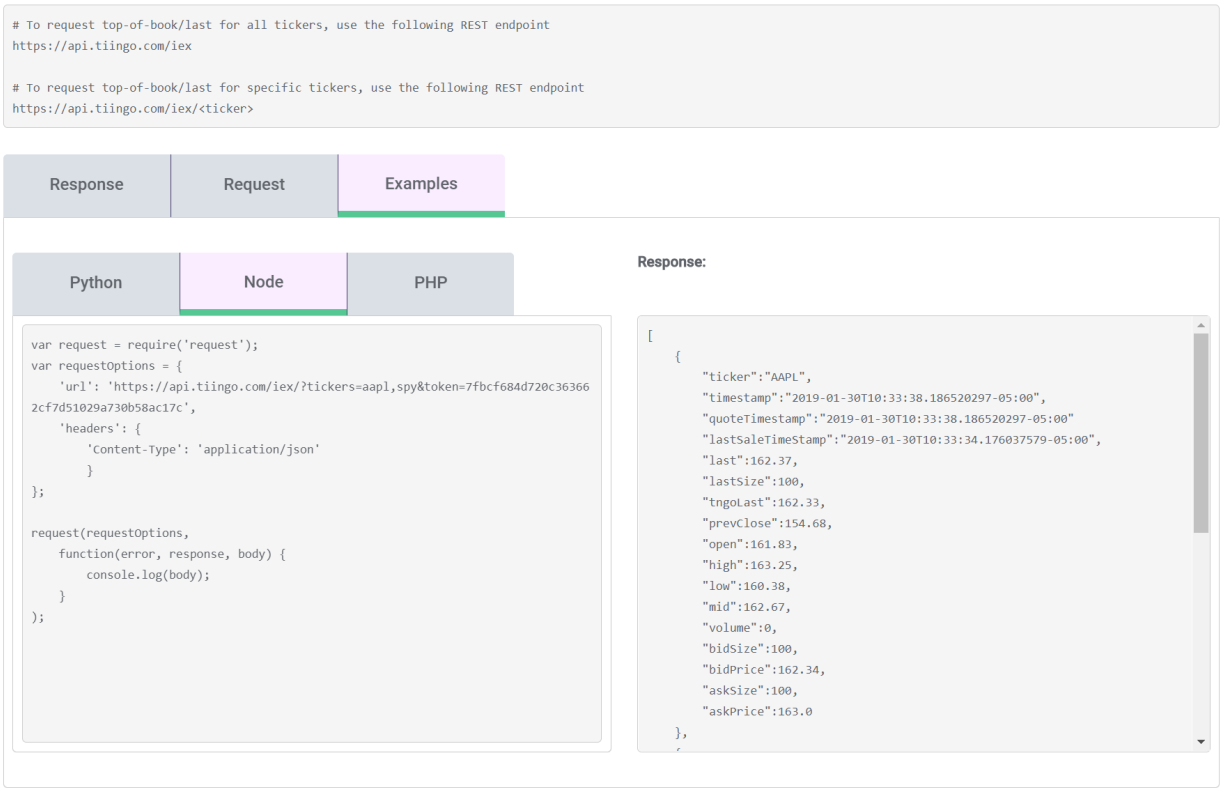


Figure 4.2: Details regarding Company's Meta Data

4.1.3 Company's Historical Data:

For **Company's Historical data**, use the following API. For more details refer to **Figure 4.3**:
https://api.tiingo.com/tiingo/daily/<ticker>/prices?startDate=<startDate>&resampleFreq=<resampleFreq>&token=<APIKeyTiingo>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: AAPL
- startDate: the date from which we need data.
- resampleFreq: the interval between 2 data lists/ arrays.
- Token: The API access Token. It is private, please do not share with anyone.

An example URL constructed from the parameters will look similar to this:

https://api.tiingo.com/tiingo/daily/AAPL/prices?startDate=2019-09-10&resampleFreq=4min&token=12PrIvA32tEmYtEmpToKeN23

Response: We receive an array of objects, where each object contains following keys. We only need the following keys from the response.

Response Keys	Details
date	Date and time of the data.
open	Open Price at the specific date and time.
high	High Price at the specific date and time.
low	Low Price at the specific date and time.
close	Close Price at the specific date and time.
volume	Volume at the specific date and time.

Table 4.3: Details regarding Company's Historical Data.

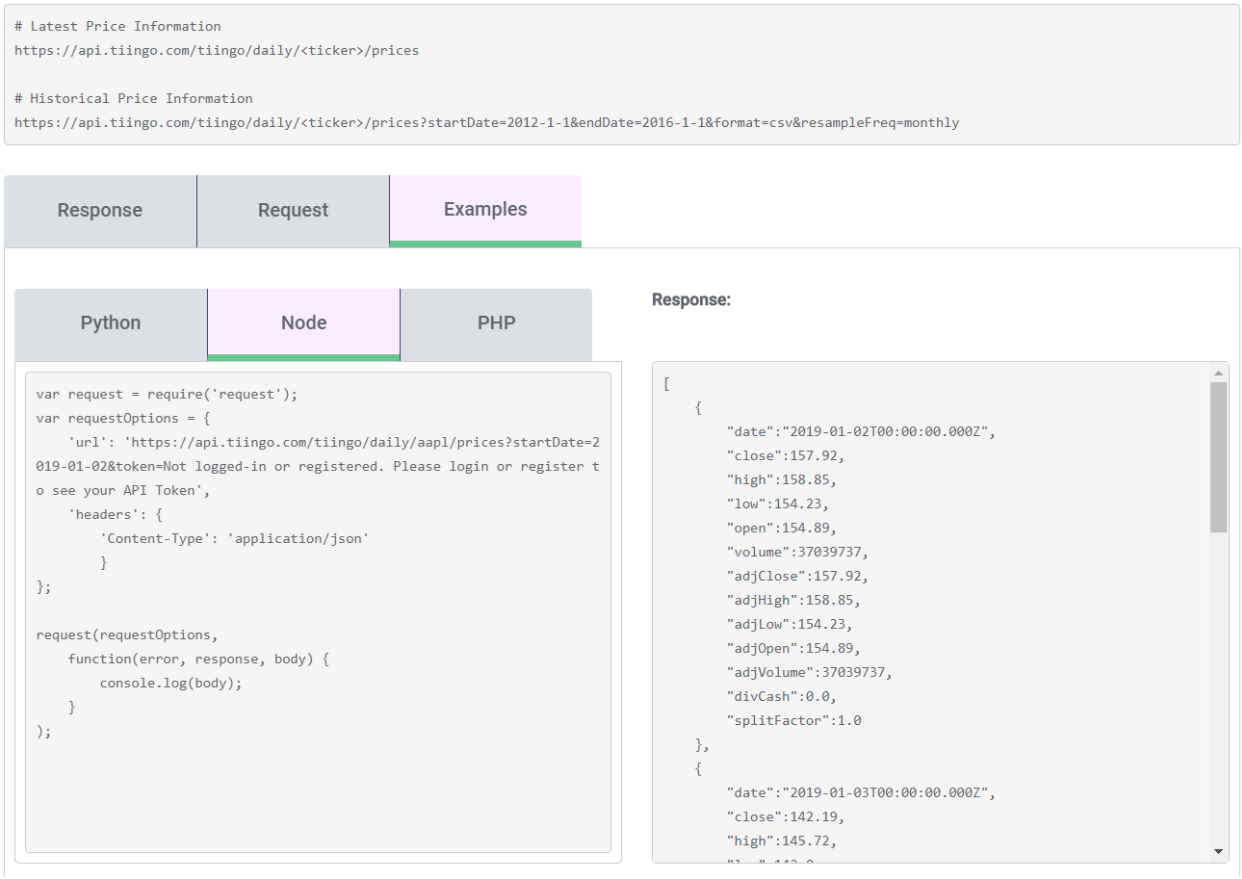


Figure 4.3: Details regarding Company's Historical Data

4.1.4 Company's Daily Chart Data:

For **Company's Last day's chart data**, use the following API. For more details refer to **Figure 4.4**:

https://api.tiingo.com/iex/<ticker>/prices?startDate=<startDate>&resampleFreq=<resampleFreq>&token=<APIKeyTiingo>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: AAPL
- startDate: the date from which we need data.
- resampleFreq: the interval at which you need data.
- Token: The API access Token. It is private, please do not share with anyone.

An example URL constructed from the parameters will look similar to this:

https://api.tiingo.com/iex/AAPL/prices?startDate=2019-09-10&resampleFreq=4min&token=12PrIvA32tEmYtEmpToKeN23

Response: We receive array of objects, where each object contains following keys

Response Keys	**Details**
date	Date and time of the data.
open	Open Price at the specific date and time.
high	High Price at the specific date and time.
low	Low Price at the specific date and time.
close	Close Price at the specific date and time.
volume	Volume at the specific date and time.

Table 4.4: Details regarding Company's Historical Data.

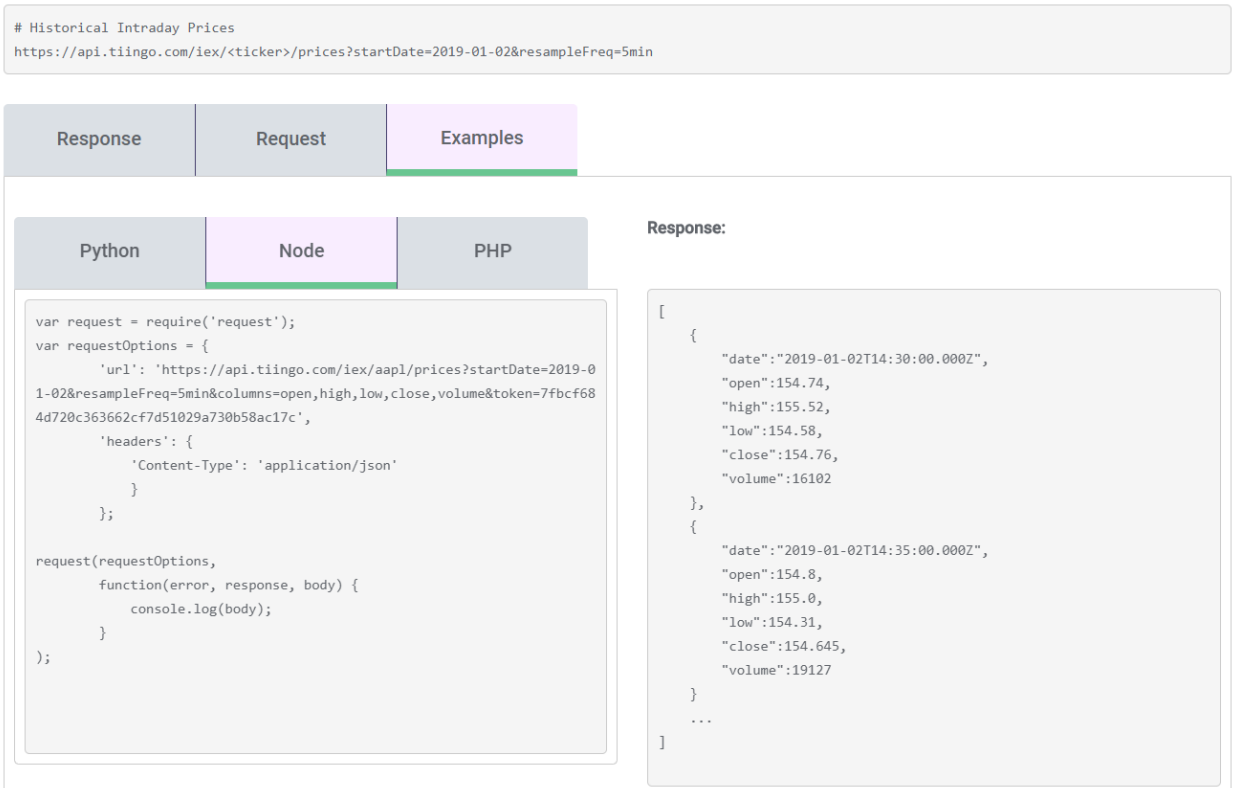


Figure 4.4: Details regarding Company's Historical Data for the latest working day.

4.1.5 Autocomplete:

For **Autocomplete**, use the following API. For more details refer Figure 4.5:
https://api.tiingo.com/tiingo/utilities/search?query=<query>&token=<APIKeyTiingo>

URL parameter in API Call:

- Query: query means the combination of letters for which you want results for autocomplete. E.g.: AA
- Token: The API access Token. It is private, please do not share with anyone.

An example URL constructed from the parameters will look similar like:

https://api.tiingo.com/tiingo/utilities/search?query=AA&token=12PrIvA32tEmYtEmpToKeN23

Response: We receive a response in the form of array of objects. From those objects we only need the following keys:

Response Keys	**Details**
Ticker	Ticker symbol.
Name	Company's Name

Table 4.5: Response received for autocomplete API call.

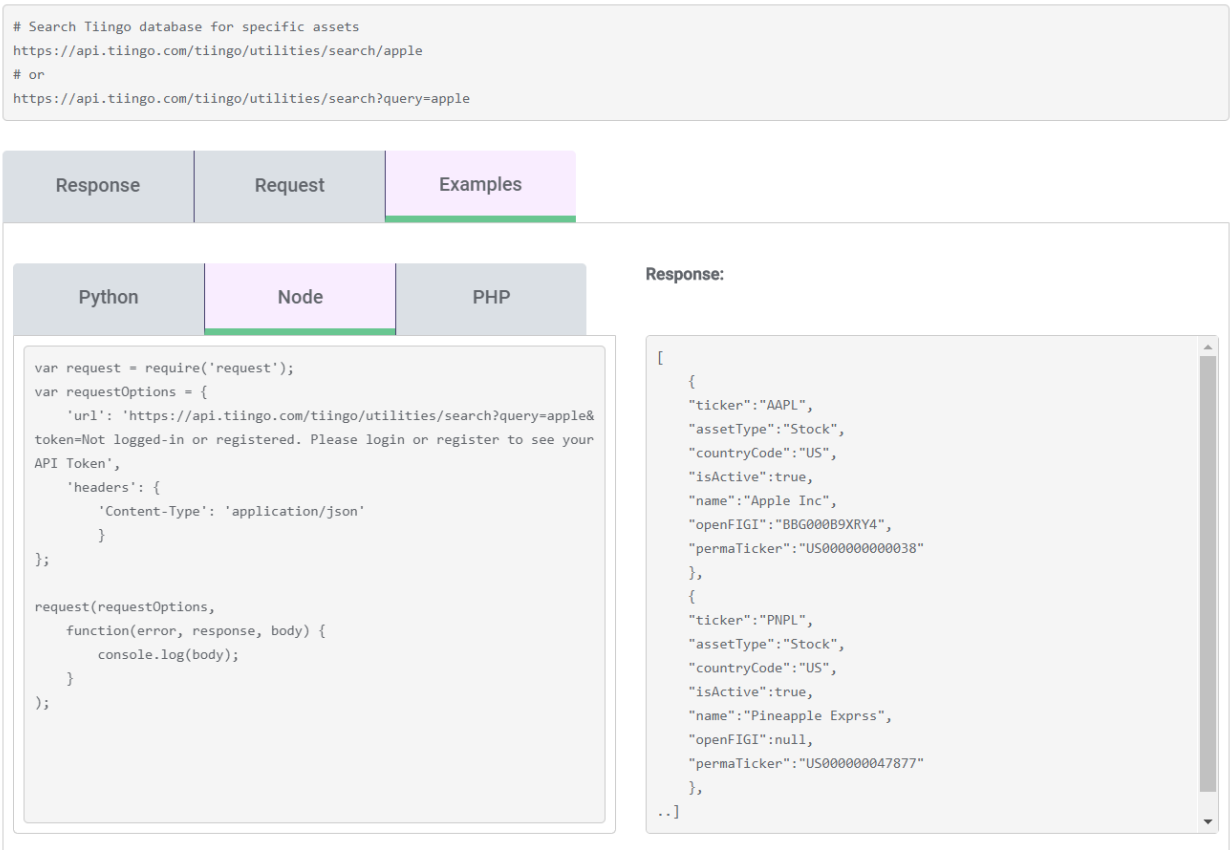


Figure 4.5: Details regarding Autocomplete Response.

4.2 News API:

For reference, check the following link:

https://newsapi.org/docs/endpoints/everything

For News related to Company, use the following API. For more details refer Figure 4.6:

https://newsapi.org/v2/everything?apiKey=<APIkey>&q=<Query>

URL parameter in API Call:

- Query: Ticker symbol of the stock. E.g.: AAPL
- APIkey: The API access Key. It is private, please do not share with anyone.

An example URL constructed from the parameters will look similar to:

https://newsapi.org/v2/everything?apiKey=12PrIvA32tEmYtEmpToKeN23&q=app

Response: We receive the response in the form of array of objects. From those objects we only need the following keys:

Response Keys	Details
url	url for the article.
title	Article's Title
description	Article's brief description
source	Source of the Article. E.g. Business Insider
urlToImage	Image URL for the article.
publishedAt	Date when the article was published

Table 4.6: Response received from News API call.

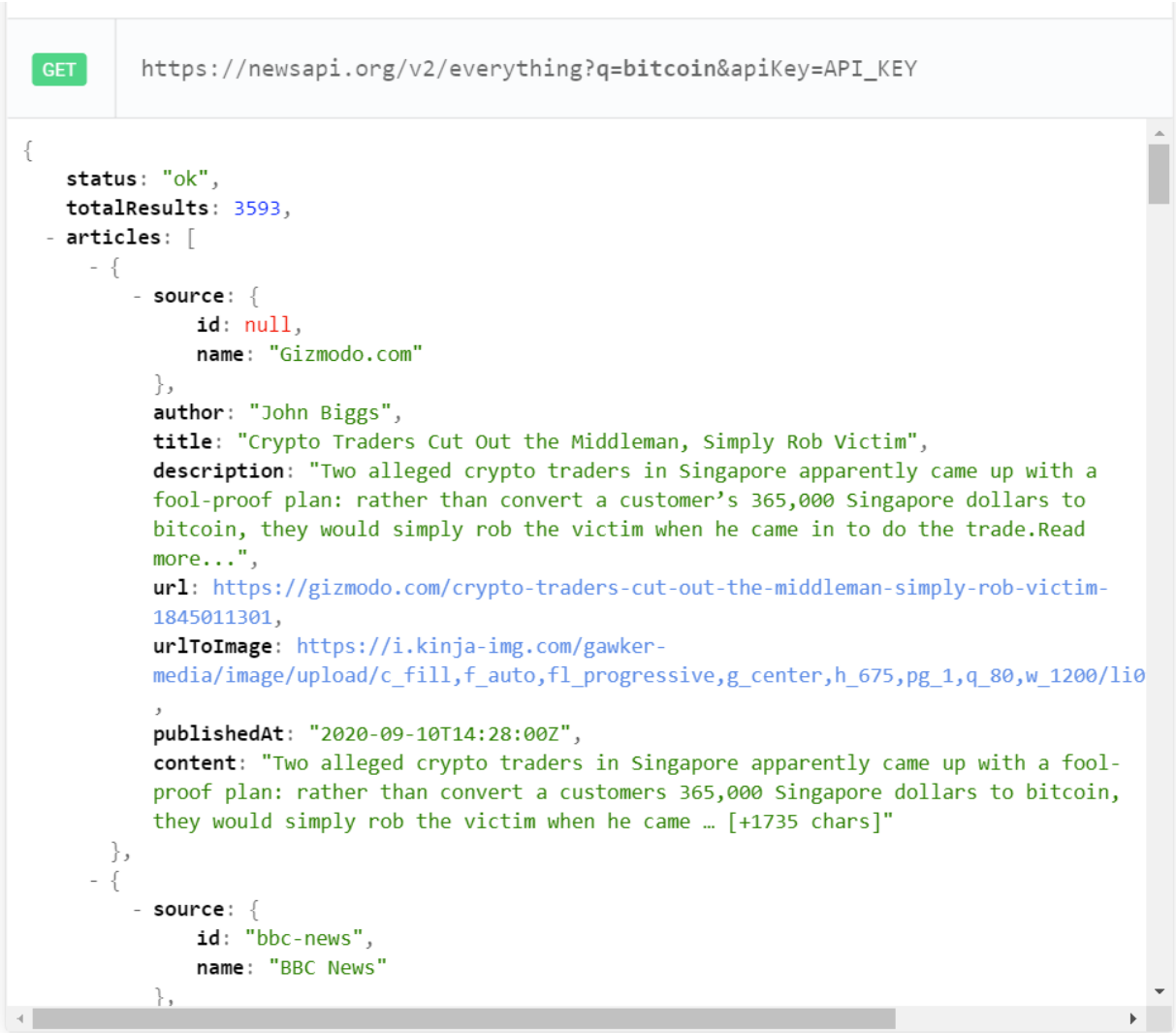


Figure 4.6: Details regarding Company News

4.3 Socials

4.3.1 Twitter

Refer the following link for details:

https://developer.twitter.com/en/docs/twitter-for-websites/tweet-button/overview

4.3.2 Facebook
Refer the following link for details:
https://developers.facebook.com/docs/plugins/share-button/

5. Implementation Hints

5.1 Bootstrap Library

To get started with the Bootstrap toolkit, please see:
https://getbootstrap.com/docs/4.0/getting-started/introduction/.

Bootstrap can be imported to Angular in couple of ways. See:
https://www.techiediaries.com/angular-bootstrap/

5.2 Bootstrap UI Components

Bootstrap provides a complete mechanism to make Web pages responsive for different mobile devices. In this exercise, you will get hands-on experience with responsive design using the Bootstrap Grid System.

https://getbootstrap.com/docs/4.0/layout/grid/

At a minimum, you will need to use Bootstrap Forms, Alerts, Cards and Buttons to implement the required functionality.

Bootstrap Forms	https://getbootstrap.com/docs/4.0/components/forms/
Bootstrap Alerts	https://getbootstrap.com/docs/4.0/components/alerts/
Bootstrap Cards	https://getbootstrap.com/docs/4.0/components/card/
Bootstrap Buttons	https://getbootstrap.com/docs/4.0/components/buttons/

5.3 Angular

- Angular Set up –
  https://angular.io/

- Angular Material Installation -
  https://material.angular.io/guide/getting-started

- Angular Material Tabs -
  https://material.angular.io/components/tabs/overview

- Angular Material Spinner –

https://material.angular.io/components/progress-spinner/overview

- Angular Bootstrap Modal - https://ng-bootstrap.github.io/#/components/modal/examples

- Angular Material Autocomplete: https://material.angular.io/components/autocomplete/overview

5.4 Local Storage

Refer the following documentation for detailed understanding on Local Storage and how to use it.

https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

5.5 Highcharts

Refer the following documentation on Highcharts and how to use it.

https://api.highcharts.com/highstock/

For using Highchart in Angular you can use highcharts-angular package. Refer following link for more details:

https://www.npmjs.com/package/highcharts-angular

5.6 Icons

- https://icons.getbootstrap.com/icons/caret-up-fill/
- https://icons.getbootstrap.com/icons/caret-down-fill/
- https://icons.getbootstrap.com/icons/star/
- https://icons.getbootstrap.com/icons/star-fill/

5.6 Deploy Node.js application on Cloud Services

Since Homework #8 is implemented with Node.js on Cloud Services, you should **select Nginx as your proxy server (if available)**, which should be the default option.

6. Files to Submit

In your course homework page, you should update the Homework #8 link to refer to your new initial web page for this exercise. Additionally, you need to provide **an additional link** to the URL of the cloud service where the AJAX call is made with sample parameter values.

****IMPORTANT**:**
All videos are part of the homework description. All discussions and explanations in Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.