WEB AND MOBILE APPLICATION DEVELOPMENT

table of contents

[1. INTRODUCTION 2](#_Toc62465726)

[1.1. BACKGROUND DESCRIPTION: 2](#_Toc62465727)

[1.1.1. Web technology descriptions: 2](#_Toc62465728)

[1.1.2. Application requirements: 2](#_Toc62465729)

[1.2. ‘STOCKMARKET EXPLORER’ design: 2](#_Toc62465730)

[1.2.1. Landing page: 2](#_Toc62465731)

[1.2.2. The Stocks Page: 3](#_Toc62465732)

[1.2.3. The Stock details page: 7](#_Toc62465733)

[2. TECHNICAL DESCRIPTION 11](#_Toc62465734)

[2.1. DEPLOYMENTS: 11](#_Toc62465735)

[2.1.1. Postman: 11](#_Toc62465736)

[2.1.2. Pre-requisites: 11](#_Toc62465737)

[2.1.3. Start the project: 12](#_Toc62465738)

[2.2. Code: 12](#_Toc62465739)

[2.2.1. Components: 12](#_Toc62465740)

[2.3. DATA FLOW: 15](#_Toc62465741)

[3. TESTING 15](#_Toc62465742)

[4. LIMITATIONS 17](#_Toc62465743)

[5. REFERENCES 18](#_Toc62465744)

[ The user initially views the landing page of the Application. 19](#_Toc62465745)

# INTRODUCTION

## BACKGROUND DESCRIPTION:

A “Stock Market Explorer” web application is been developed for the client side. This application allows the user to search and view the latest 100 days financial statistic information to be pertained with regards to the user’s choice of stock. This web application was built by using web technologies like React, Node and JS. Also the use of ag-grid and chartjs; have helped to build the application with the effective representation of stock data in Chart and table.

### Web technology descriptions:

* ReactJS – It is a javascript library which is used for building interactive user interfaces. **(Walke, 2013)**
* NodeJS – It is a free open source platform which performs all server related activities like open, read and modify data or files from the server or database **(Node.js Introduction, 2020)**

### Application requirements:

The applications which are used for building this web application were all listed below.

* The information about the stock is obtained from the server of REST API - http://131.181.190.87:3001
* Use postman software for fetching the data from REST-API
* Visual Studio Code – This is used for modifying and implementing the JS code for the application
* Mozilla Firefox (or) Google Chrome – used for viewing the “Stock Market Explorer” web application online.

## ‘STOCKMARKET EXPLORER’ design:

Based on the description provided in the assignment this web application maintains three web pages and they are

* Landing Page (1st page)
* Stock Page(2nd page)
* Stock Details Page (3rd page)

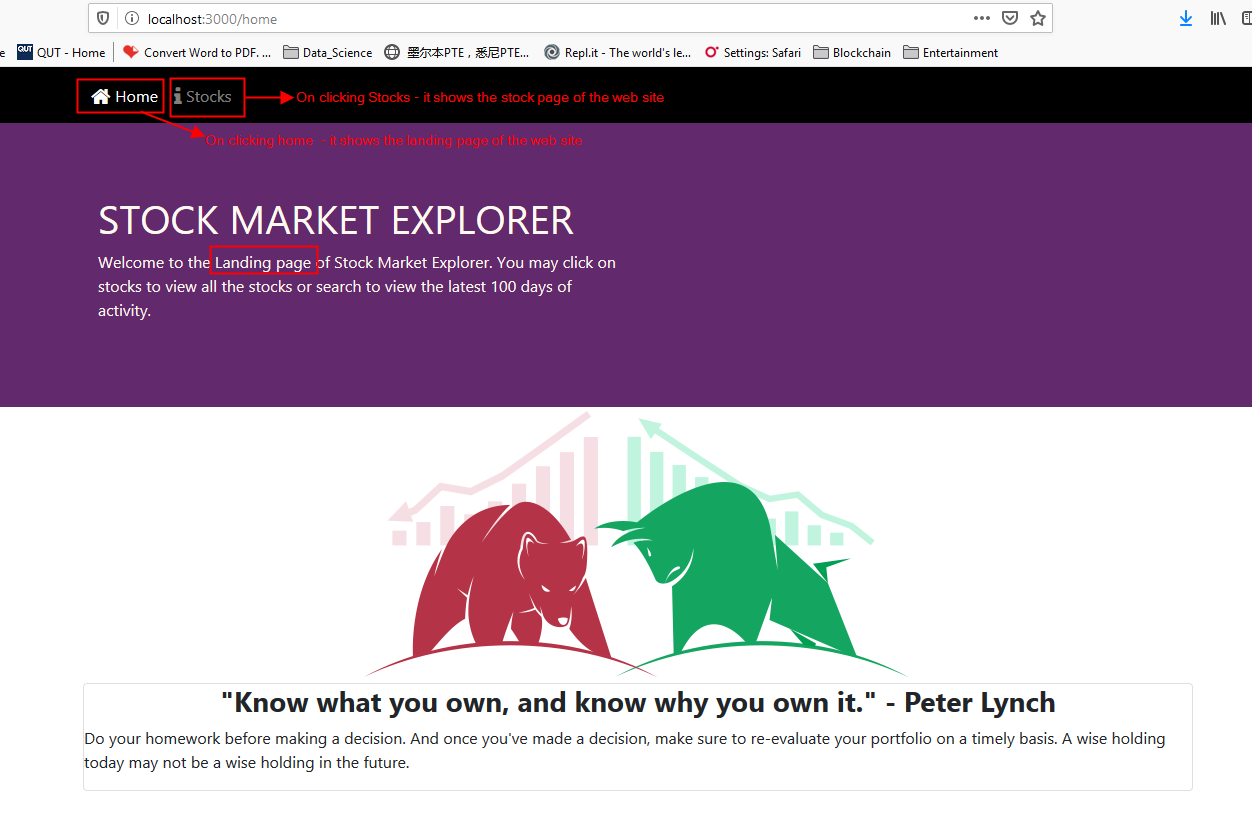
As specified in the assignment each web page represents different functionality. Let’s discuss in detail about the functionalities that each web page holds.

### Landing page:

As specified in the Assignment the landing page (Figure: 1) of the web application has the 2 major functionality button and this function is replicated in all the pages of the web site. These two buttons are available in the Navigation bar header of the website, so that it allows the user to navigate back to stocks or home page based on their requirement.

* **Home button** – Whenever home button is selected, it automatically navigates the user to the Landing page of the website.
* **Stock button** – When this button is selected by the user; it navigates the user to the Stock page of the website.

Also, an image and a quote about the stock market is also provided in this page of the website.



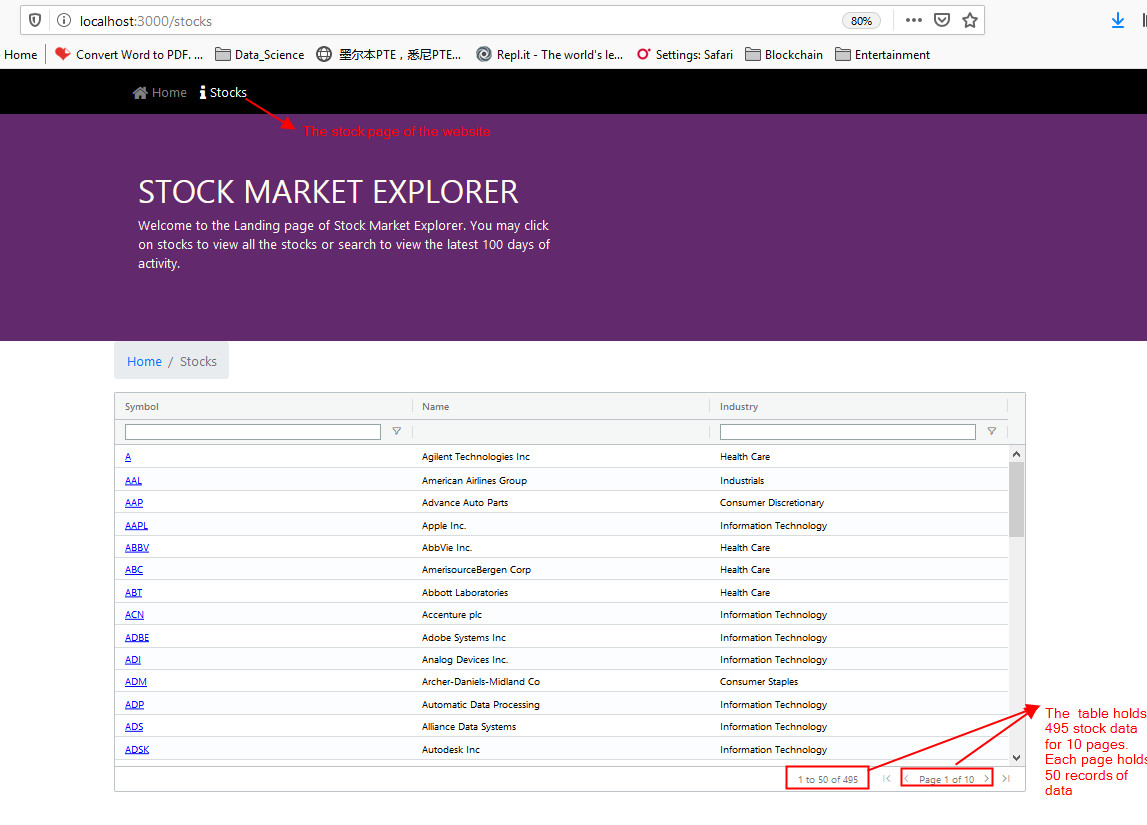
**Figure: 1:**  The Landing page

### The Stocks Page:

The actual appearance of the Stock page in the website is shown in- Figure: 2(a). On selecting the Stock button in the Navigation bar; the user is navigated to this page of the website. This page holds the stock information of 495 industries in the stock table. Each stock in the table is represented based on industry name type and stock code. The table holds 10 pages, and each page holds 50 stock data.

In this page the user can perform the following activity;

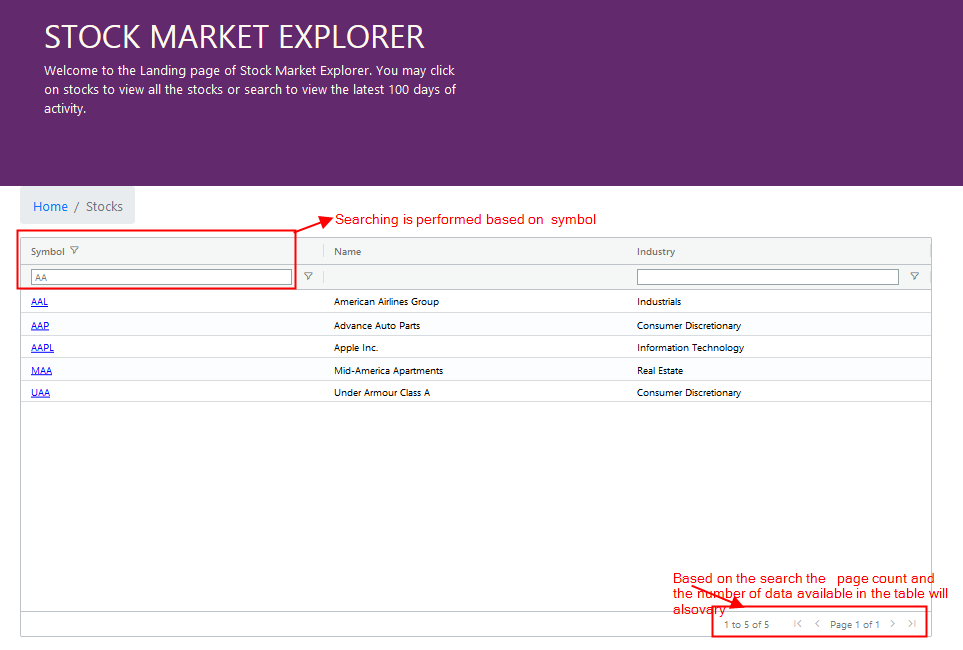
* Search the stock based on symbol
* Search the stock based on Industry name
* Sorting of stock values based on Industry name and symbol
* Navigate the user to the stock details page



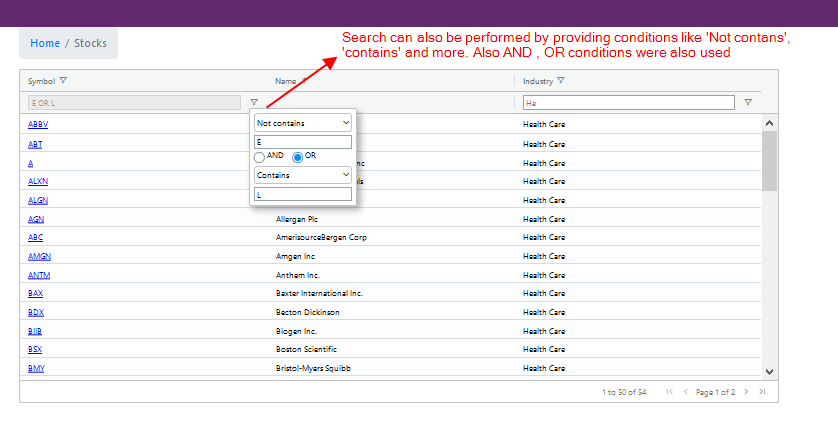
**Figure: 2 (a):** The stock page of the Application

#### Stock code search:

This search is performed in the Symbol column of the Stock table. Figure 2(b) and 2(c) gives a clear idea about how the search is been performed in the web site.



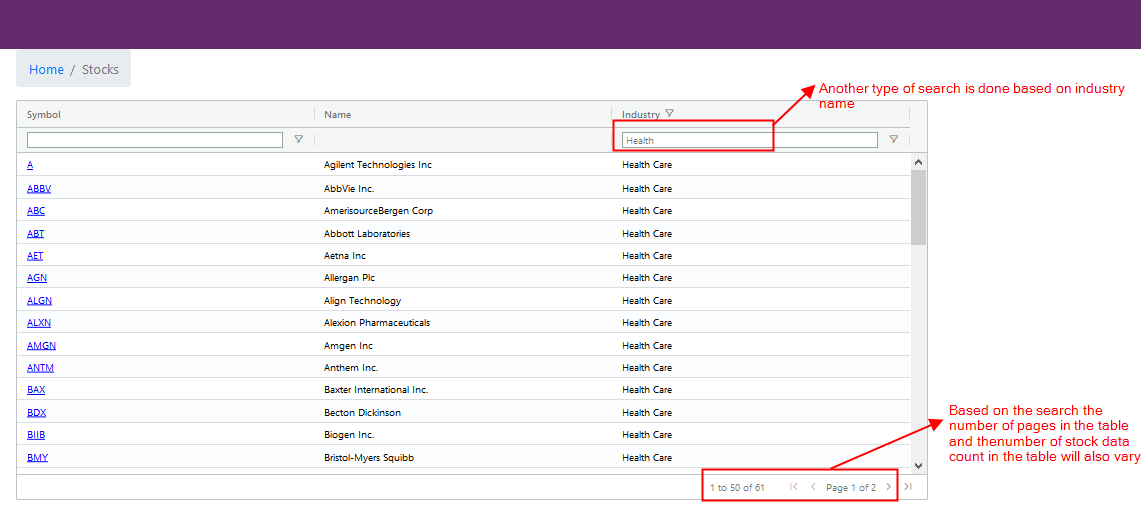
**Figure: 2 (b):** Search activity of symbol column through Textbox



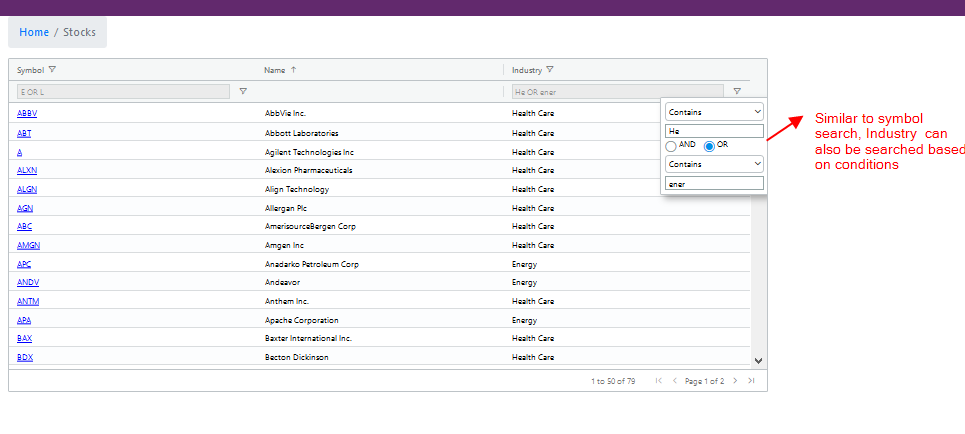
**Figure: 2 (c):** Search activity of symbol column through dropbox

#### Industry name search:

This search is performed in the Industry column of the Stock table. Figure 2(d) and 2(e) gives a clear idea about how the search is been performed in the web site.



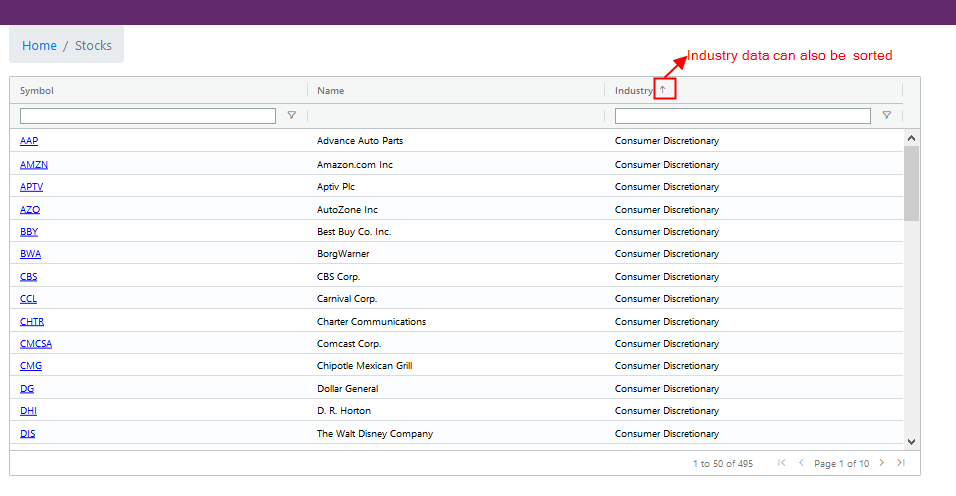
**Figure: 2 (d):** Search activity of industry column through Textbox



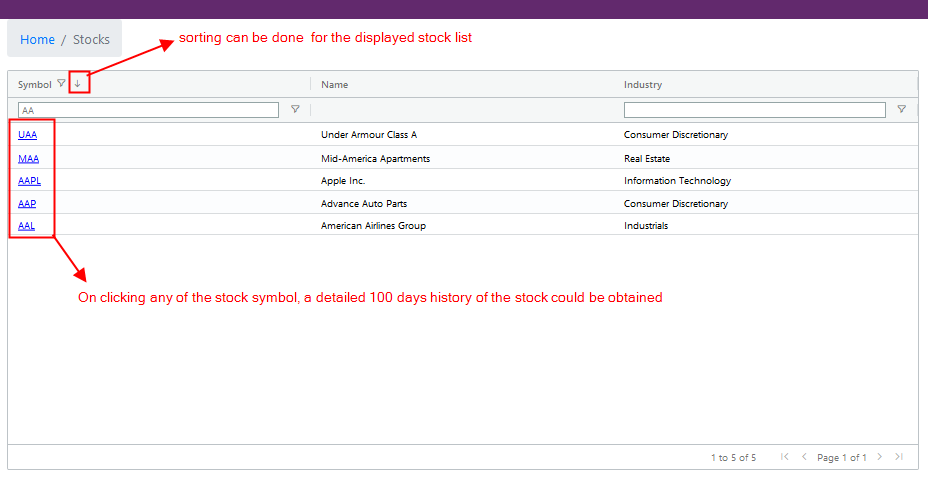
**Figure: 2 (e):** Search activity of Industry column through dropbox

#### Industry and Stock code sorting:

Sorting is performed based on alphabetic arrangement of data for the industry and Symbol column of the stock table. Figure: 2(f) and 2(g) gives a clear idea about how this activity is been performed in the website.



**Figure: 2 (f):** Sorting performed in industry column



**Figure: 2 (g):** Sorting performed in Symbol column and also navigation activity performed in symbol column

#### Navigate to Stock details page:

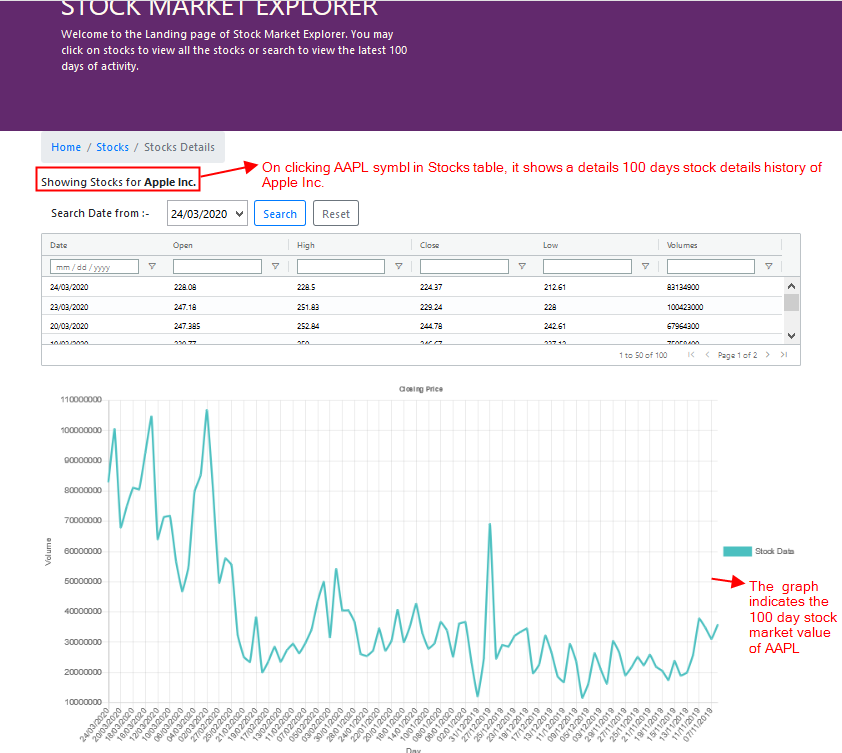
In figure: 2(g); in-order to navigate to the next page of the website, the user needs to select any of the stock code of which they want to know about the 100 day stock history of that particular stock.

### The Stock details page:

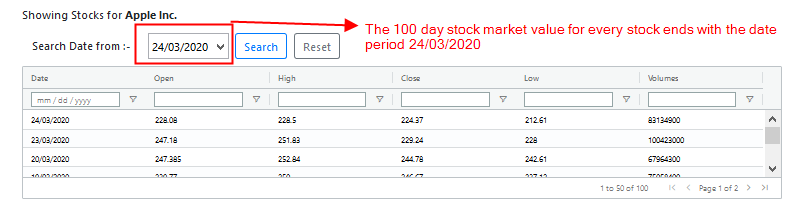
This page holds the 100 day stock history about the stock of which the user has selected from the Stock table of Stocks page. This page would also provide a graph of which it clearly indicates how the stock was invested in the market. Figure: 3(a) and 3(b): gives a clear image of how this appears in the website.

Based on the Assignment specification, this page allows the user to perform the following activity.

* Search the stock details based on date
* Search the stock details based on conditions
* Sorting of Stock details values



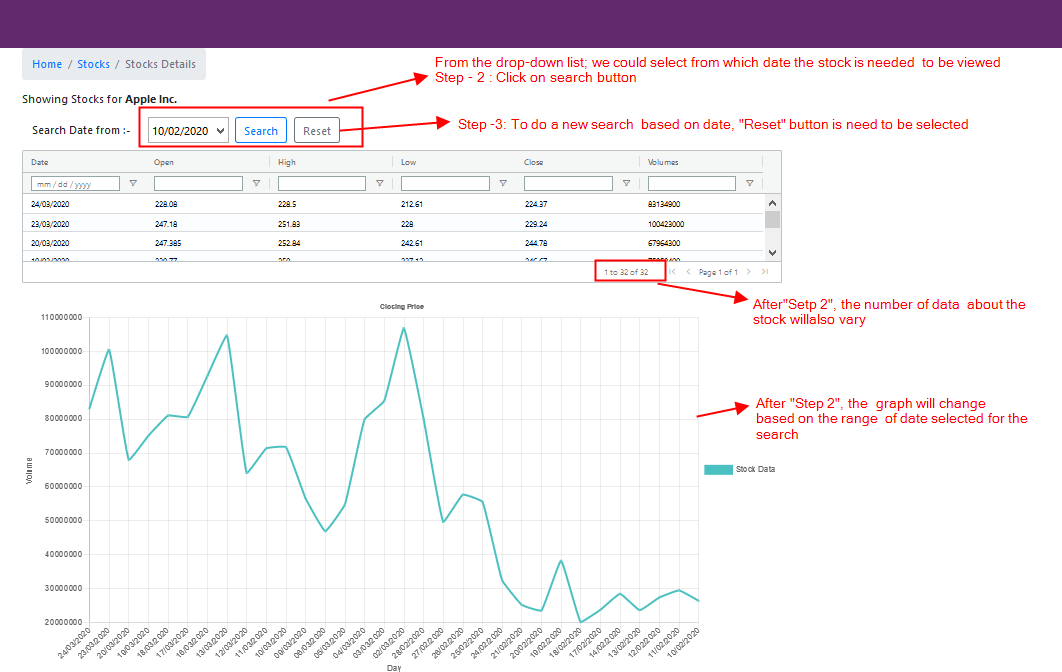
**Figure: 3(a):** Stock details Page for AAPL



**Figure: 3(b):** The date dropbox specifies the 100 day history details of AAPL stock

#### Date Search:

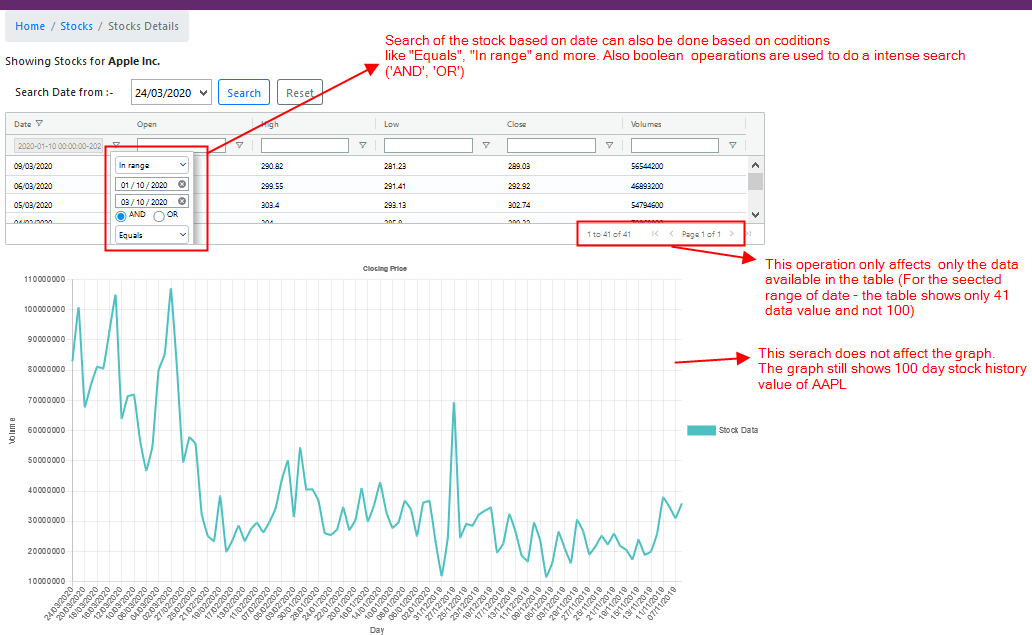
This allows the user to search the stock based on date by following the three steps of which it’s been specified in Figure: 3(c). Based on the search the table and graph representation of the stock will also change. The value of the table and graph will always ends with the date period 24/03/2020.



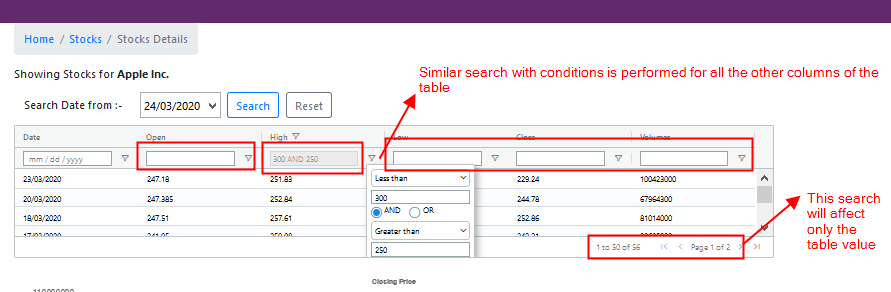
**Figure: 3(c):** Search activity performed in “Search date From” dropbox

#### Condition Search:

This allows the users to search the stock values in the Stock details table by applying conditions. This condition search can be performed for all the columns of the table. Figure: 3(d) and 3(e) gives a clear idea about how this activity is performed in the website.



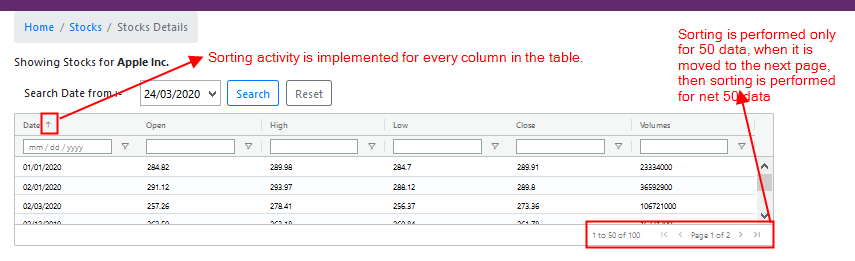
**Figure: 3(d):**  Search activity performed in the date of Stock details table drop box



**Figure: 3(e):**  Search activity performed in all columns of Stock details table drop box

#### Sorting of Stock values:

This activity allows the user to sort the values of all the columns in the stock details table and the sorting changes which are performed in the table will not reflect any graph in the Graph. Figure: 3(f) gives a clear idea about how the sorting activity is performed in the website.



**Figure: 3(f):**  Sorting is performed for all columns of Stock details table

# TECHNICAL DESCRIPTION

## DEPLOYMENTS:

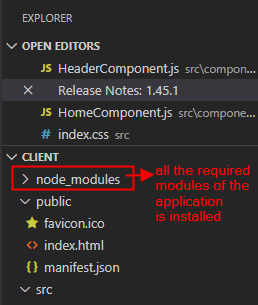
### Postman:

This software is used for getting the data from the REST API server - <http://131.181.190.87:3001>.

### Pre-requisites:

Install ‘npm’ in-order to deploy all backend dependencies of the application. This will help install all the required node modules for the application.

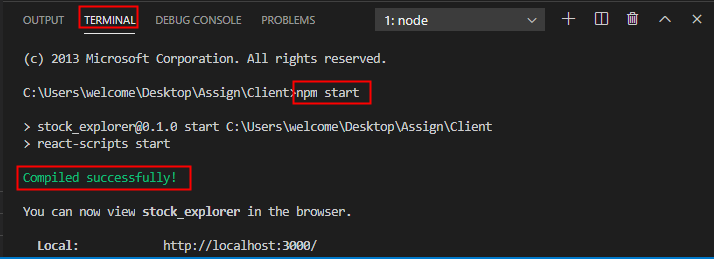




**Figure: 5:** All the node modules are installed

### Start the project:

To start the application in the client side, then start npm in the terminal (Figure:6). This helps to compile the application code successfully and displays the web application in the client side.

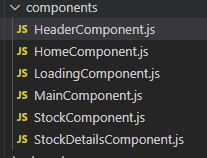


**Figure: 6:** To start the application

## Code:

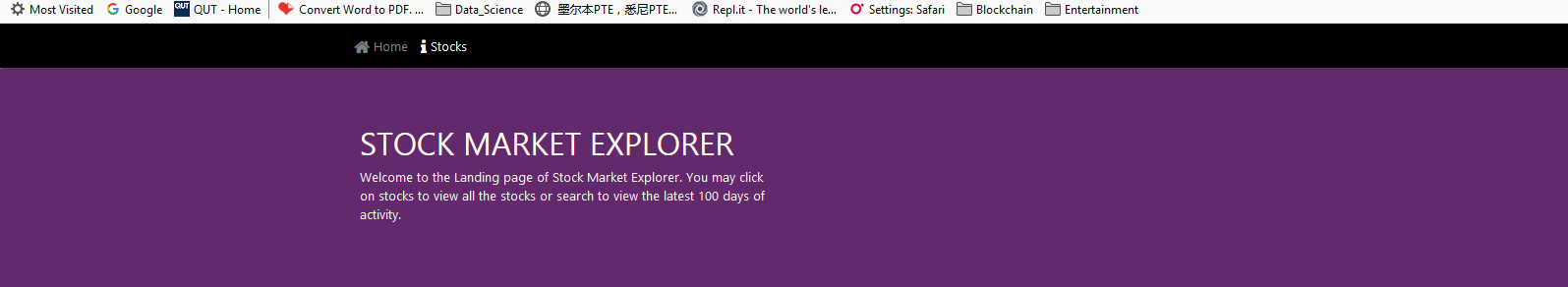
### Components:

There are 6 components used for this web application, they are.



* **HeaderComponents:** Handles the header section of the web application Figure: 7:. This section remains the same in all the web pages of the application. It maintains the navigation bar (Home and Stocks), the web application name “STOCK MARKET EXPLORER” and the content below the application name.

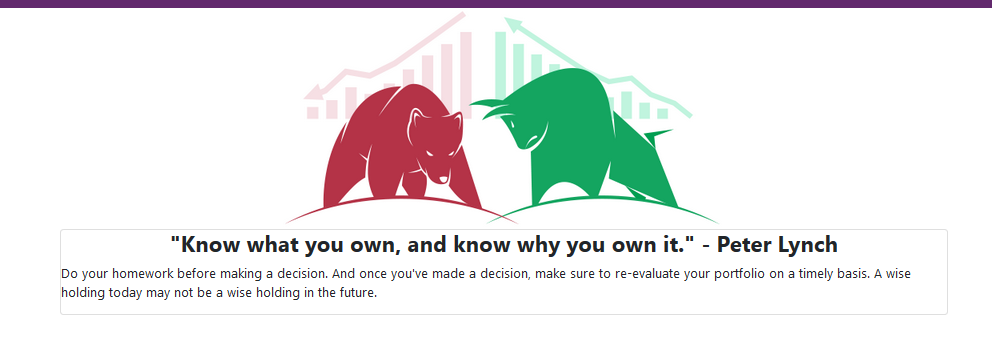
Navbar components are used for providing navigation activity in the web application.



**Figure: 7:** HomeComponent.js

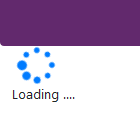
* **HomeComponents:** The Landing page body content is handled by this component. It maintains the image and the quote which is written below the image.

As for the Classname : container (image) and post card is used for the ‘quote’



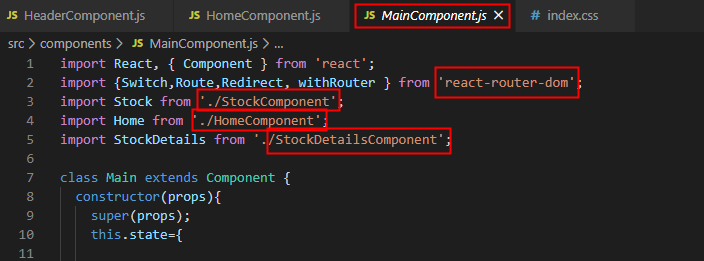
**Figure: 8:** HomeComponent.js

* **LoadingComponent:**  In the server side if rendering is happening then the page will display a loading function. So that activity is handled by this component.



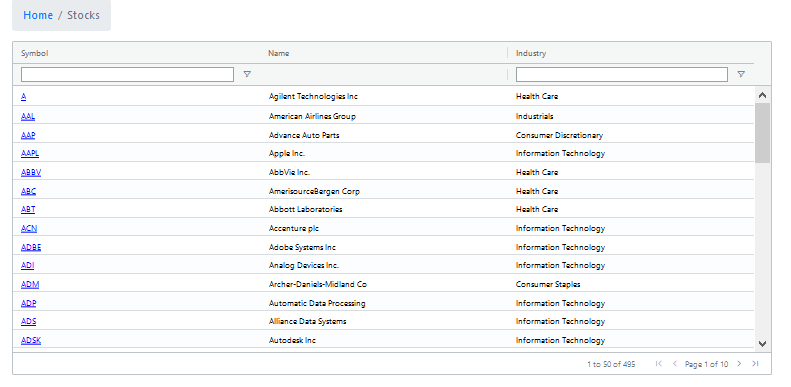
**Figure: 9: LoadingComponent.js**

* **MainComponent:** In this component the react router is deployed and it act as the skeleton component that helps to connect all the other components of this application.



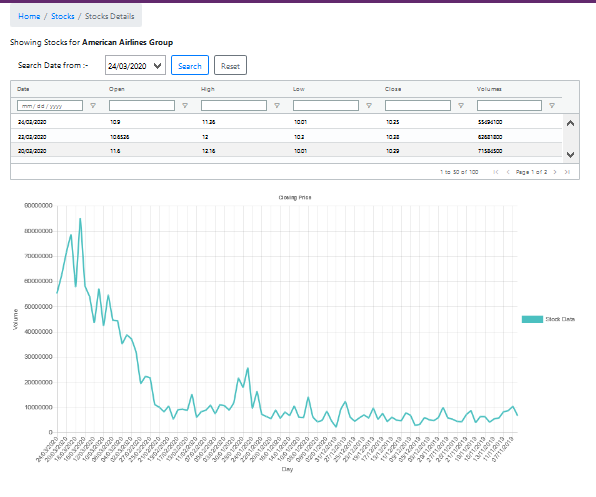
**Figure: 10: MainComponent.js**

* **StockComponent:**  It handles the stock page body content of the web application and it uses aggrid for handling table activities.



**Figure: 11: StockComponent.js**

* **StockDetailsCompnent:** It handles the Stock details body content of the web application. It uses aggrid for handling table activities and chartjs for handling the graph



**Figure: 12: StockDetailsComponent.js**

## DATA FLOW:

The project always starts with the **“Index.js” 🡪 “App.js” is initiated 🡪 calls the “MainComponent.js”. This helps to call internally all the other related component of the web application.**

**App.css –** The presentation styles which were not obtained from the bootstrap, there was a need to imply some extra customization for the application. So these extra customize styles of which it’s been used for the project is all added in this section

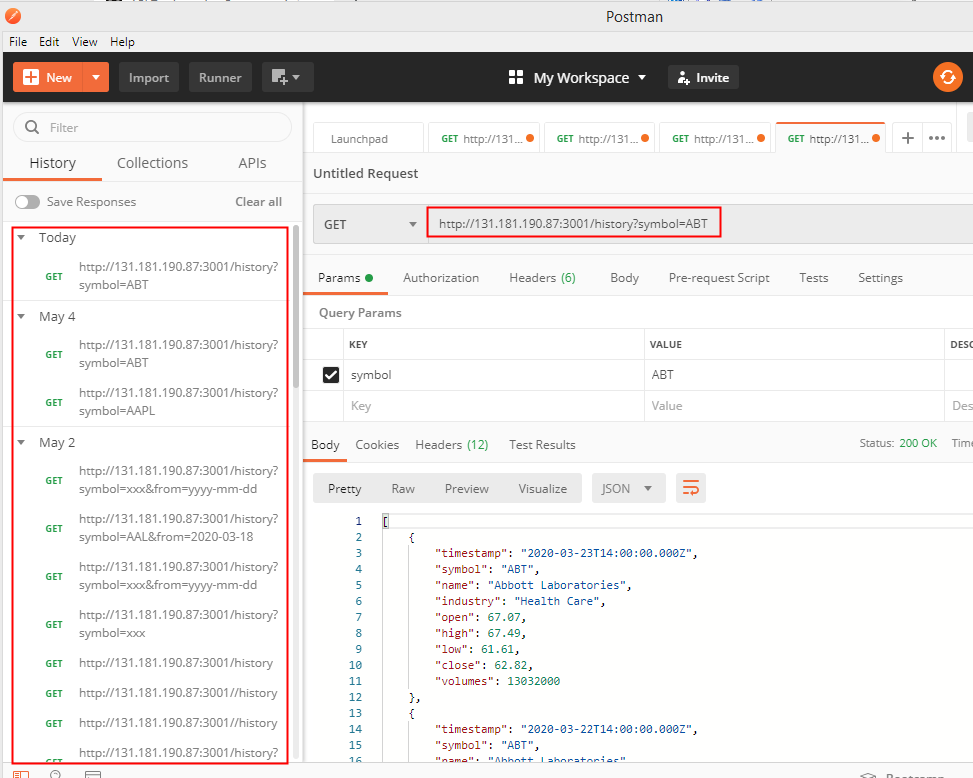
# TESTING

Some of the testing which is performed for this Web application is listed below

* **API Testing using Postman:**

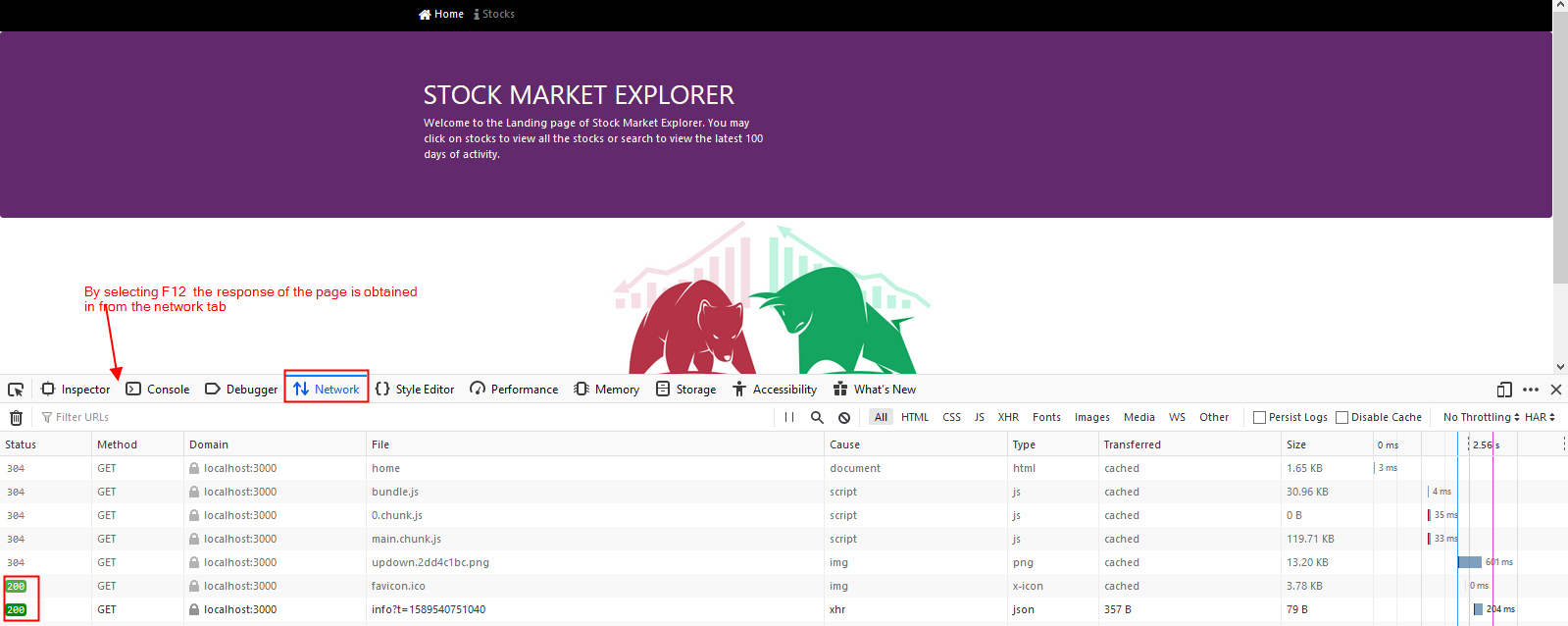
For the below listed API it is important to check if they meet functionality expectations, performance, security and provides correct responses. So postman application is used for checking the API responses by sending access request to the web server. For performing this activity VPN is not required. So using postman Unit testing activity is performed. Some of the testing activity performed using postman is shown in Figure: 13 **(Kotecha, 2020)**

* + /all (GET)
  + /all?symbol=xxx (GET)
  + /industry?industry=xxx (GET)
  + /history?symbol=xxx (GET)
  + /history?symbol=xxx&from=yyyy-mm-dd (GET)



**Figure: 13: Postman API test**

* **Console log and console debug is checked:** It is checked in the browser end by pressing F12 option. Figure: 14: gives a good understanding about how this analysis is performed. When the response status is 200 – then it is indicated as successful request. **(Kotecha, 2020)**



**Figure: 14: Console response check**

* **Component testing:** By using React Devtools (Chrome browser extension) two main features are analyzed **(Moriarty, 2018)**
* View of the component tree
* The current state and props of each component of which was been selected for preview.

# LIMITATIONS

* For the table in Stock Page ag-grid features were used and this helped to perform client side conditional search. In this search data which is available in client side is been filtered and searched. In the later part of the search, the filtered data is obtained back to the user only by calling the server through refreshing the Stocks page.
* Similarly the conditional search which is performed in the Stock-Details page will not affect the graph content as the Chartjs completely links with the server side and not on the client side. Whereas the table is completely links with client side data. So to get back the filtered data, the server needs to call by refreshing the Stock details page of the application.
* For implementing in the production environment, where the number of users who access the application would be 200+, then this application could use MongDB database, AWS and more.

# REFERENCES

* Kotecha, K., 2018. *API Testing Using Postman*. [online] Medium. Available at: <<https://medium.com/aubergine-solutions/api-testing-using-postman-323670c89f6d>> [Accessed 6 February 2018]. (Kotecha, 2018)
* Moriarty, P., 2018. *React Devtools: A Brief Introduction | Digitalocean*. [online] Digitalocean.com. Available at: <<https://www.digitalocean.com/community/tutorials/react-react-devtools-intro>> [Accessed 21 May 2018]. (Moriarty, 2018)
* W3schools.com. 2020. *Node.Js Introduction*. [online] Available at: <<https://www.w3schools.com/nodejs/nodejs_intro.asp>> [Accessed 15 May 2020]. (Node.js Introduction, 2020)
* Walke, J., 2013. *React (Web Framework)*. [online] En.wikipedia.org. Available at: <[https://en.wikipedia.org/wiki/React\_(web\_framework)>](https://en.wikipedia.org/wiki/React_(web_framework)%3e%20) [Accessed 29 May 2013]. (Walke, 2013)

APPENDIX – User Manual

# The user initially views the landing page of the Application.

* In this page user has the option to navigate to the stock page. As well as home page (Landing page) is also available for the user to come back to the landing page from Stocks or Stock details page.
* By selecting the Stock page, the user has the option to perform intense search on Stock code and industry name. This is performed by using the two proposed search functionalities
* Search by using text box
* Search by applying conditions in the dropbox
* User also has the option to move to next page of the application “Stock details” by selecting any of the stock code links.
* In Stock details page user has the option to view (by chart and table) the 100 days stock value of the respective selected stock.
* The stock information can also be searched by two concepts
* Search by “Search From Date” option
* Search by applying conditions in the table