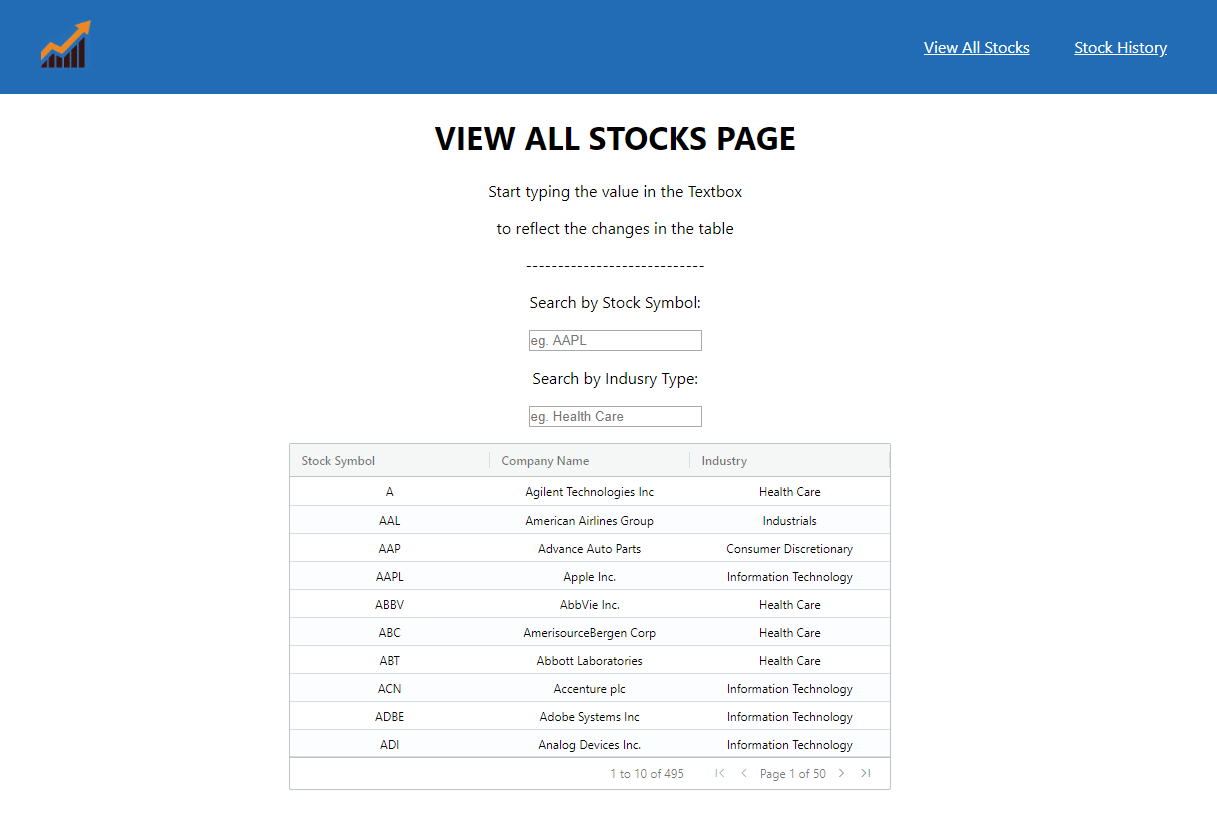
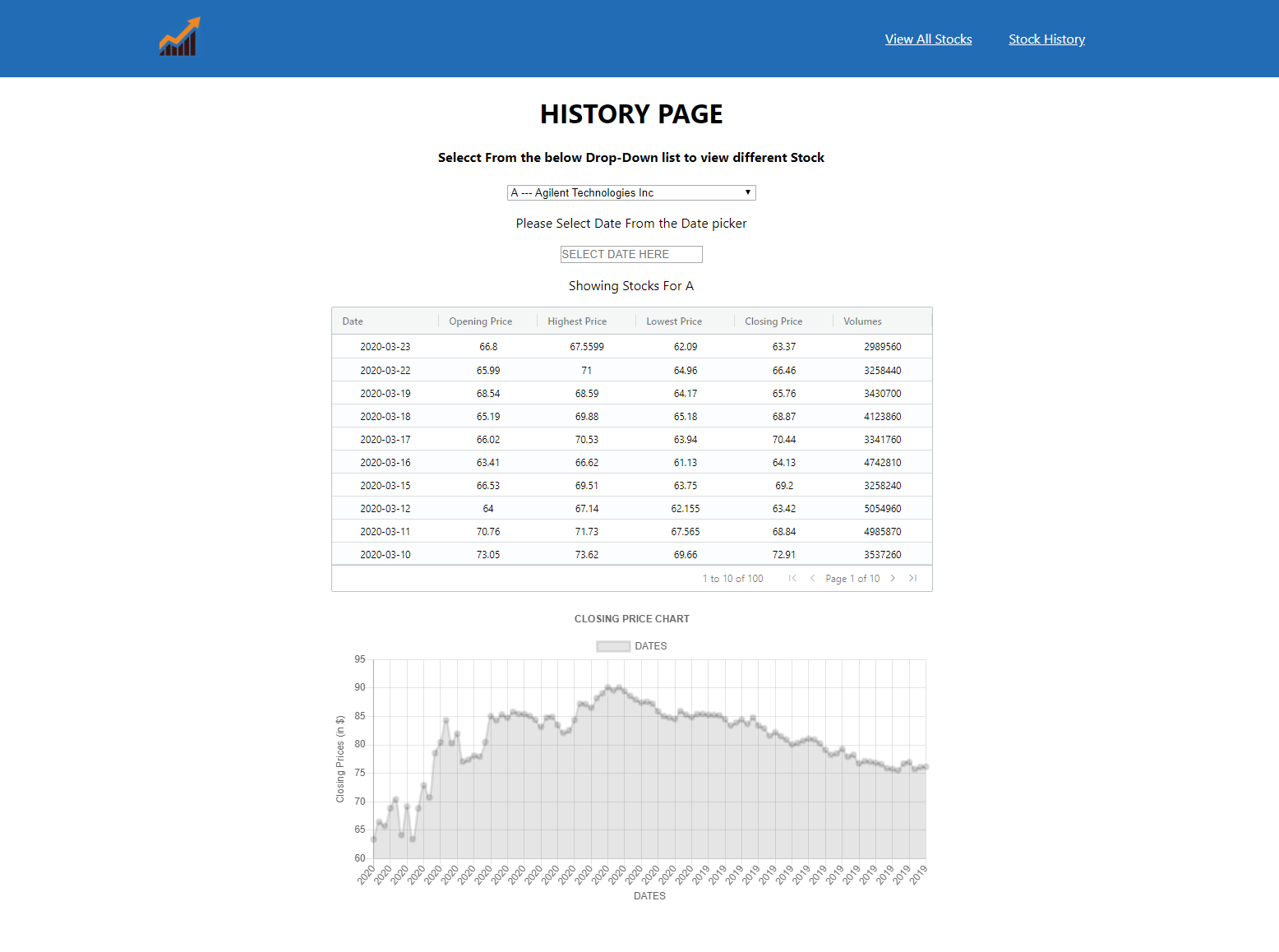
# **Introduction**

For this Assignment I have created a React App (WebApp) as required in the assignment. This is an app which fetches data from the given REST API and fetches data using React Hooks. This app also provides options to user such as searching the data without hitting the server eventually reducing the server Load. This app mainly contains three pages.

1. The First one is the Landing Page often called as the Home page. This is a static page which only some information about the App.
2. The Second page is to view the details of the stocks. This page displays the table build with ag grid react library. This table displays the details of the stock in terms of stock symbol, Name of the company associated with the stock symbol and the Industry in which the company is based on. Additionally, it also allows users to filer the results as a search box is provided in this page. User can search the stocks by the Symbol name and Industry name as well.
3. The third and the final page is the Stock History page which shows various details of the stock according to the date in context of Opening Price, Highest Price of the stock as of the day, the lowest price of the stock in the day, the closing price of the stock at the end of the day and Volumes . This page also provides the Dropdown Menu which allows users to change the stock company which immediately effects the table. Moreover, the graph is also shown at the bottom which shows the Closing price of the stock in comparison to the date. This graph provides the overview of the share price of the company.

Below are the screenshots of some of the pages.





# **Technical Description.**

Since this is the first time, I am developing a react application I have learned a lot of new things in react. Now as we talk about the Technical Description lets start with the first Page – The Homepage. This is the static page which shows the general information of the WebApp. This page consists a bunch of Headings <H1>…<h6> tag and <p> tag as well.

The view all stocks page consists of input boxes and Ag-Grid in the form of table. This whenever this page loads, useEffect() method of react hooks is called, and it loads all the data in rawData variable of usestate() component. This data is then printed into the Ag-Grid table. There are two textboxes one for searching by symbol and other for search by industry in the table. When an input is made through this textbox onChange() method of textbox in triggered which fetches the value of the textbox and saves it to the symbol\_key variable of the use effect component. And then this key is used to change the content of the table according to the symbol key. And the main thing about the is that this all process is done without hitting the server. rawData is used as the dataset to search key value and when found the entire row consisting the symbol, name and industry is saved to the keydata variable of use state component. And then this key data is then filled in the Ag-grid. This same process is also applicable to search by industry textbox. Here also server hit is saved by processing the data which is already fetched once.

The search History page consist of some headings, dropdown menu, textbox, DatePicker, Ag-Grid table and Line graph based on ChartJs Library. First of all, when the table load, the default value of the dropdown list is kept to its first value and so the table and the graph are visible at the beginning. Now users are also provided the options to select other stock from the dropdown. When the user selects any other option from the menu the onChange() method of the dropdown component is called which sets the selected value to Selection variable of the Usestate component. And that value is used to fetch the data according to the symbol form the server. And then this fetch method of react hooks is used to fill data in the Ag-Grid table below. Below is the DatePicker which is the component of react-datepicker Library. This datepicker is used to filter the results which are only after the selected date. After that there is and Ag-Grid table below which displays all the data od the selected dropdown item. It changes itself as the value of the date-time picker is changed. This Ag-Grid table shows data in the terms of date, opening price, highest , lowest, closing and Volumes in terms of number of transactions on particular day. Below it the Line graph (component of Chartjs Library). This chart plots the dates on the x axis and closing price of the selected stock. This line graph also updates itself as the value of ag-grid table and dropdown menu changes.

**Other Libraries Used :**

1. Ag-Grid
2. react-chartjs-2
3. react-datepicker

**Endpoints:**

As the view all stocks page is loaded the data is fetched in background from the /all endpoint. And as we start searching the values in the textbox the data is filtered in the background which provides the exact same result as /all?symbol=xxx and /all?industry=xxx. Then this data is provided to the Ag-grid and then the table is displayed.

In the stock history page is loaded, the default data is loaded considering the symbol a and the data loaded into the Ag-Grid. The data which fetched in the background the uses the /history?symbol?symbol=xxx endpoint.

For Graphing I have used ChartJs library. This is an easy to use library which provides graph by just taking inputs in the form of arrays. But in this project that data which I was fetching was in the form of objects. So it was a bit tricky to convert that data into the form of array. So what I did was that at the time of fetching I stored the closing price and the dates in a separate array so that I have the required data for preparing the Graph. And when I want to print the graph I called the function and passed the data in the form of arrays and I got the graph…

# **Testing and Limitations.**

For testing I have checked the data at the beginning. Since the beginning I have planned to do all things as per the requirement including the graphs and Ag-grid. To be honest, Ag-grid was not that tough, but I took some time to get the grip on Graphs. For me that was also my learning curve since I have made my first react. One test was very crucial for me when I forgot to convert the string data of closing price to float value. Initially I converted them into the Int but by seeing all the graph values in integer I was bit concerned and found my mistake. That was another learning lesson for me. The results were came just as I was expecting. I completed almost all the parts, but one part didn’t go well which was providing the links to the symbol column in the Ag-Grid at the first page. I could have done that also but eventually I ran out of time.

I have successfully developed lot of components in this project. In the View All Stocks I have successfully added both the filters namely search by symbol and search by industry also. I didn’t add the buttons but instead I have changed the data of the table as you just start typing in the textboxes which I think is lot more cool than pressing the button. Moreover, the main thing is that the search did not even hit the server, it just processes the pre fetched data. And on top of that while the page loads the all the data is present in the Ag-Grid table. And then changes dynamically as the search keys are entered. In the Stock History Page, the default data and graph of stock symbol A which is of Agilent Technologies Inc is already present right at the page load. Here also I did not add any search button as I feel much better changing the data and the graph as the value of the dropdown list is changed. The graph at the changes the value of x-Axis according to the closing price of the stock which is much better.

Moreover, when I was checking the Dataset with the help of graph, I found that some of the data provided on the /history endpoint are missing. Below is the list of data variations which I found out:

* Link: <http://131.181.190.87:3001/history?symbol=SCG>

Symbol: SCG

The data provided here is I think not accurate as most of the opening price, Highest price, closing price are equal to 0.0045. And the Volumes are 0 for every entry.

* Link: <http://131.181.190.87:3001/history?symbol=HCN>

Symbol : HCG

The data provided here has the missing last entry for the date 23/03/2020

* Link: <http://131.181.190.87:3001/history?symbol=Xl>

Symbol : XL

The data provided here has the missing last entry for the date 23/03/2020.

Volumes are also missing in the last three date entries.

* Link: <http://131.181.190.87:3001/history?symbol=COL>

Symbol : COL

The data provided here has the missing last entry for the date 23/03/2020.

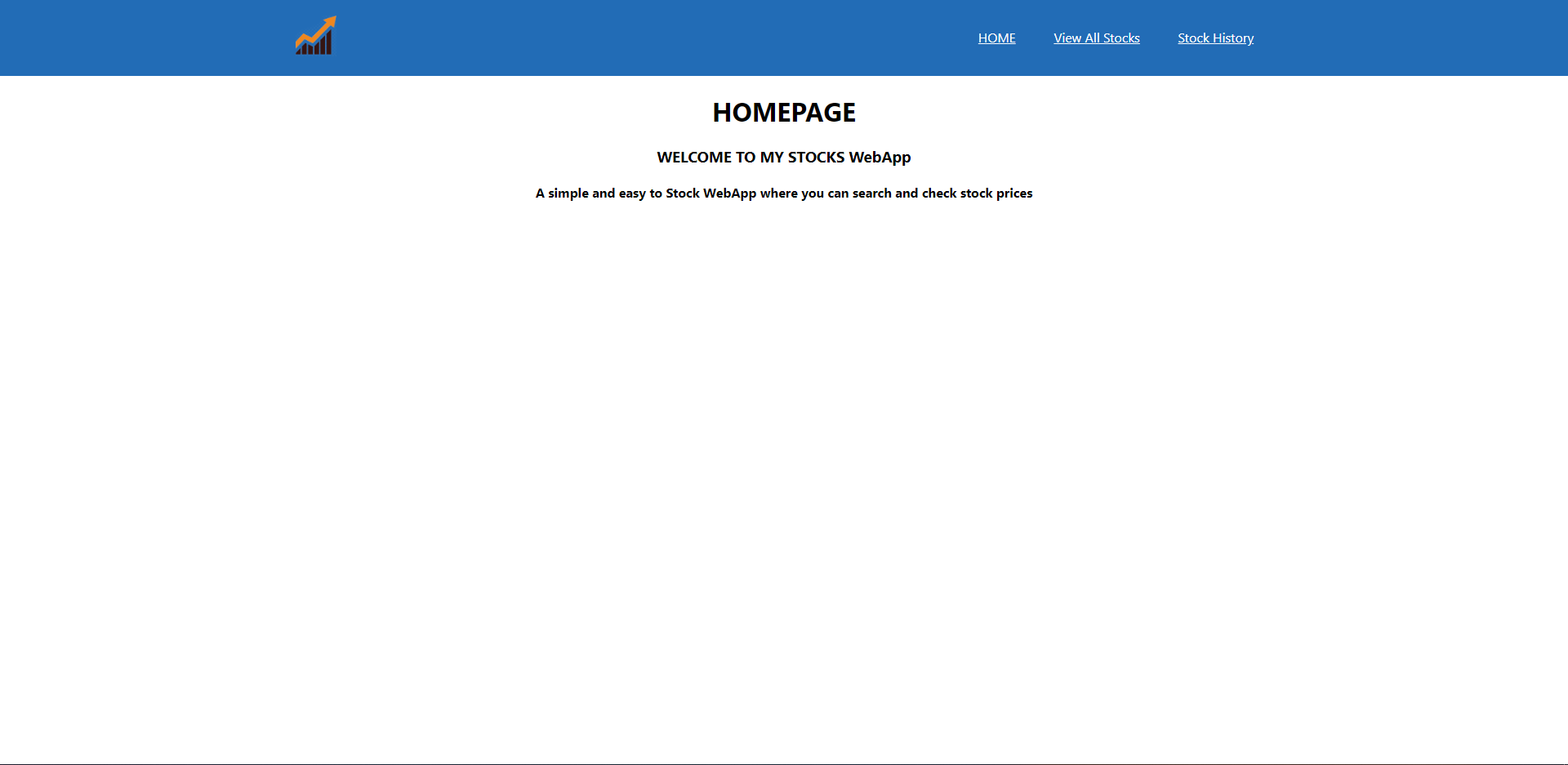
* Link: <http://131.181.190.87:3001/history?symbol=EHC>

Symbol : EVHC

The data provided here is not accurate as all the prices are same at 45.99 every time since the beginning of 2020

# **Appendix: USER GUIDE**

As we open the WebApp we will be on the Homepage which looks something like this:

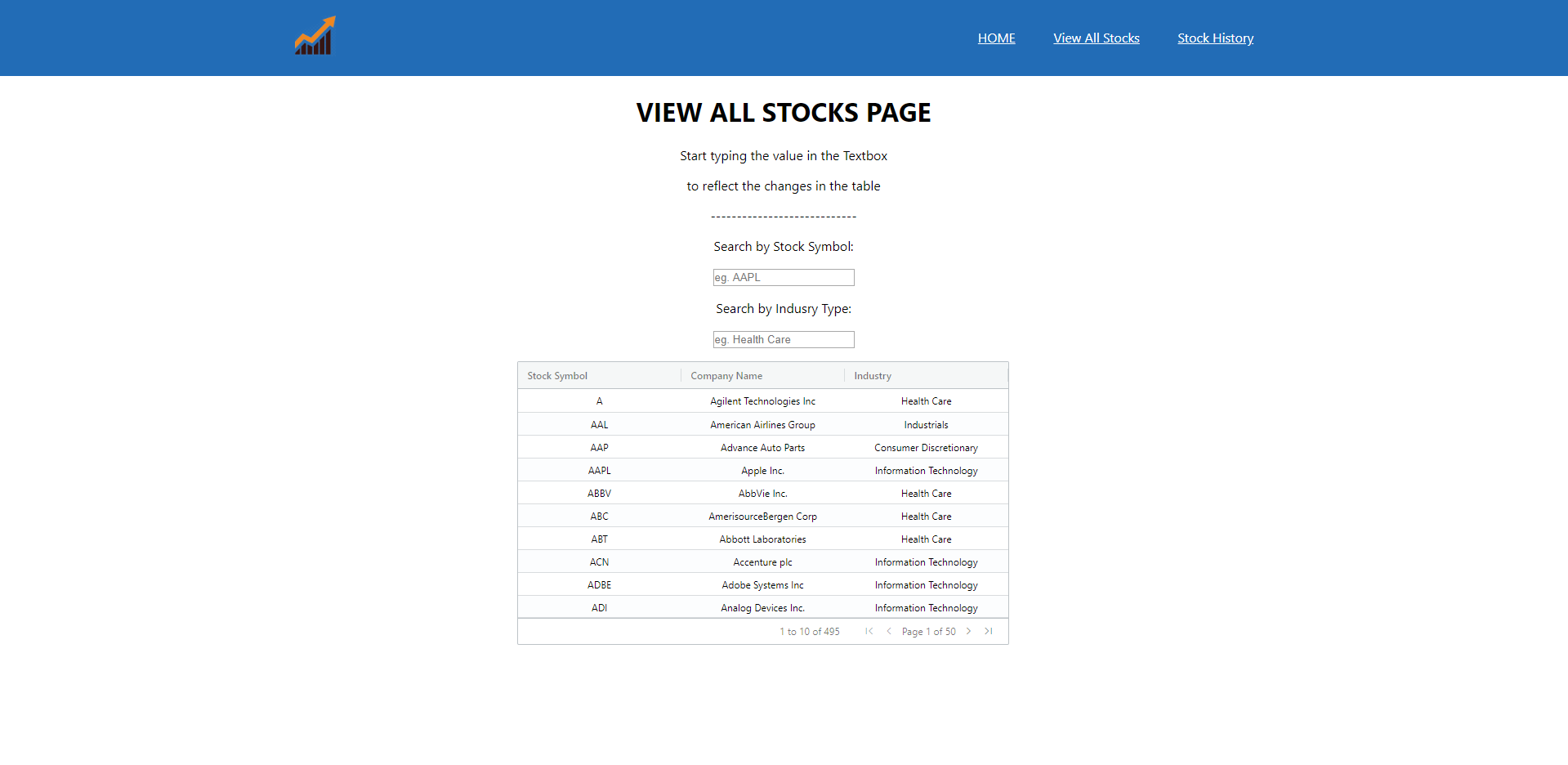


Now we have the Navigation menu in the Header with the options such as Home, View All Stocks and Stock History. This Header will remain on all the pages of this WebApp. The Header with the navigation links will look as below:

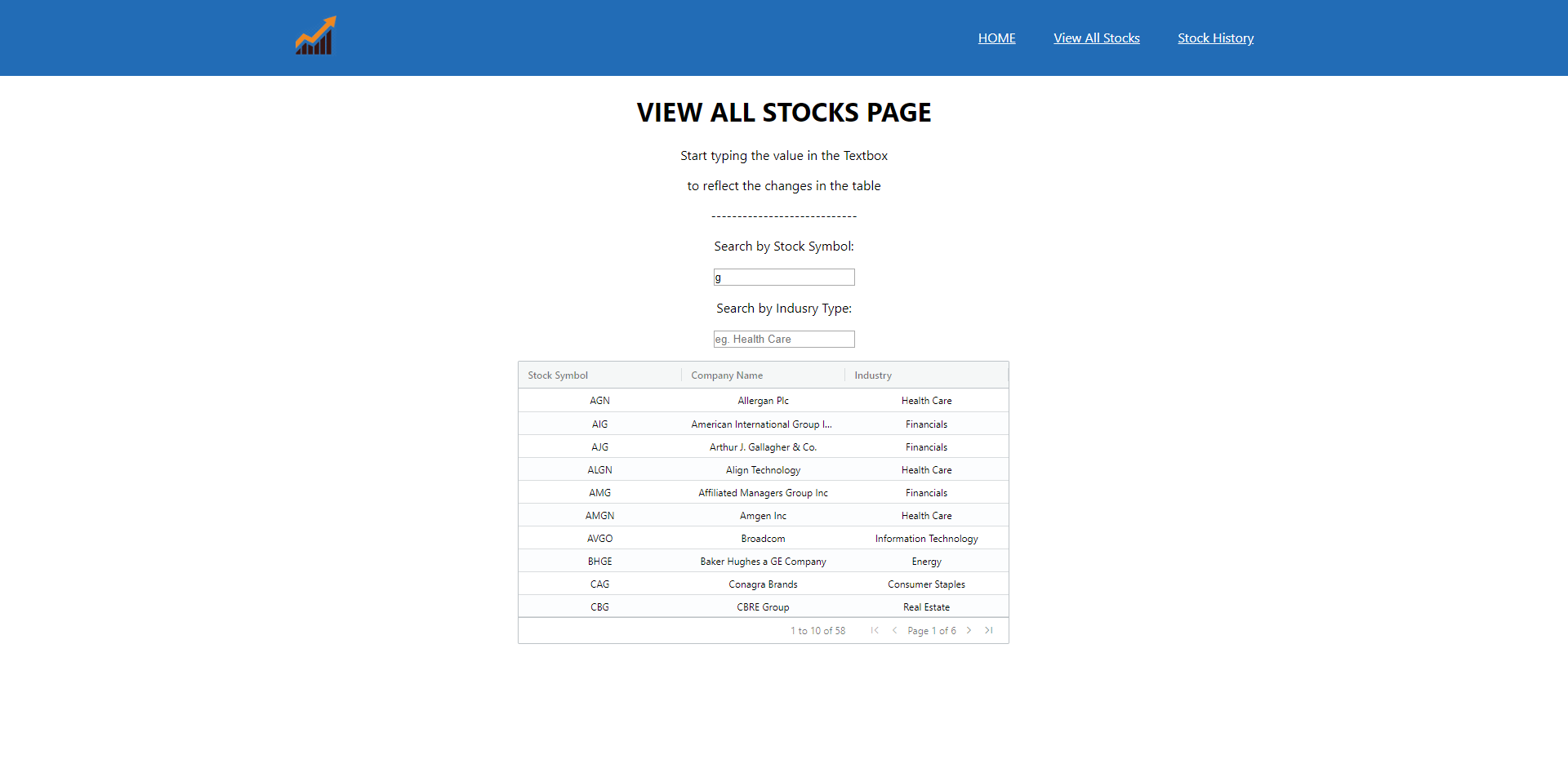


Now as we click any of the navigation Items, we will be forwarded with the destination page.

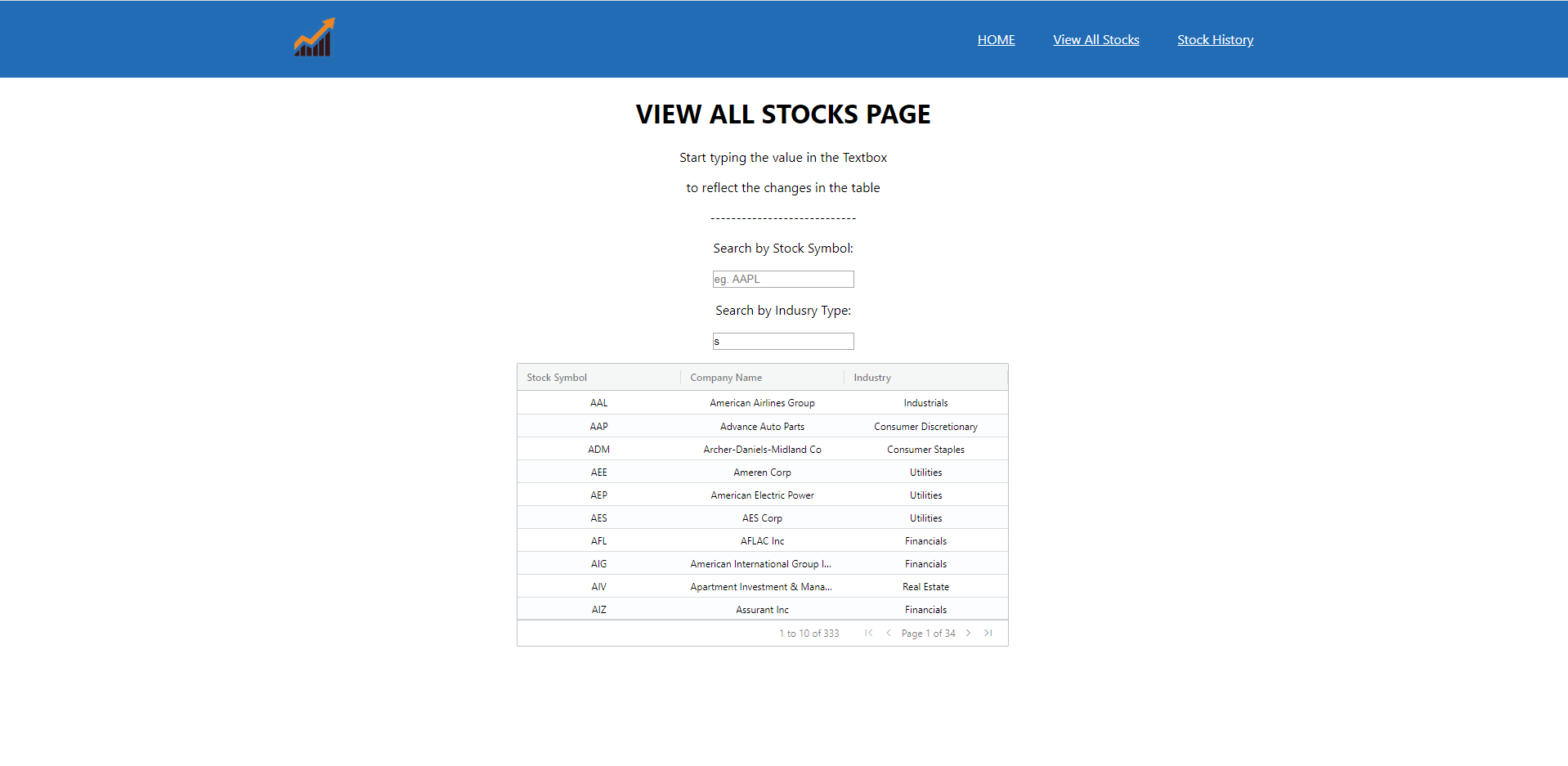
Now if we click on the View All Stocks link, we will be forwarded to the following page:

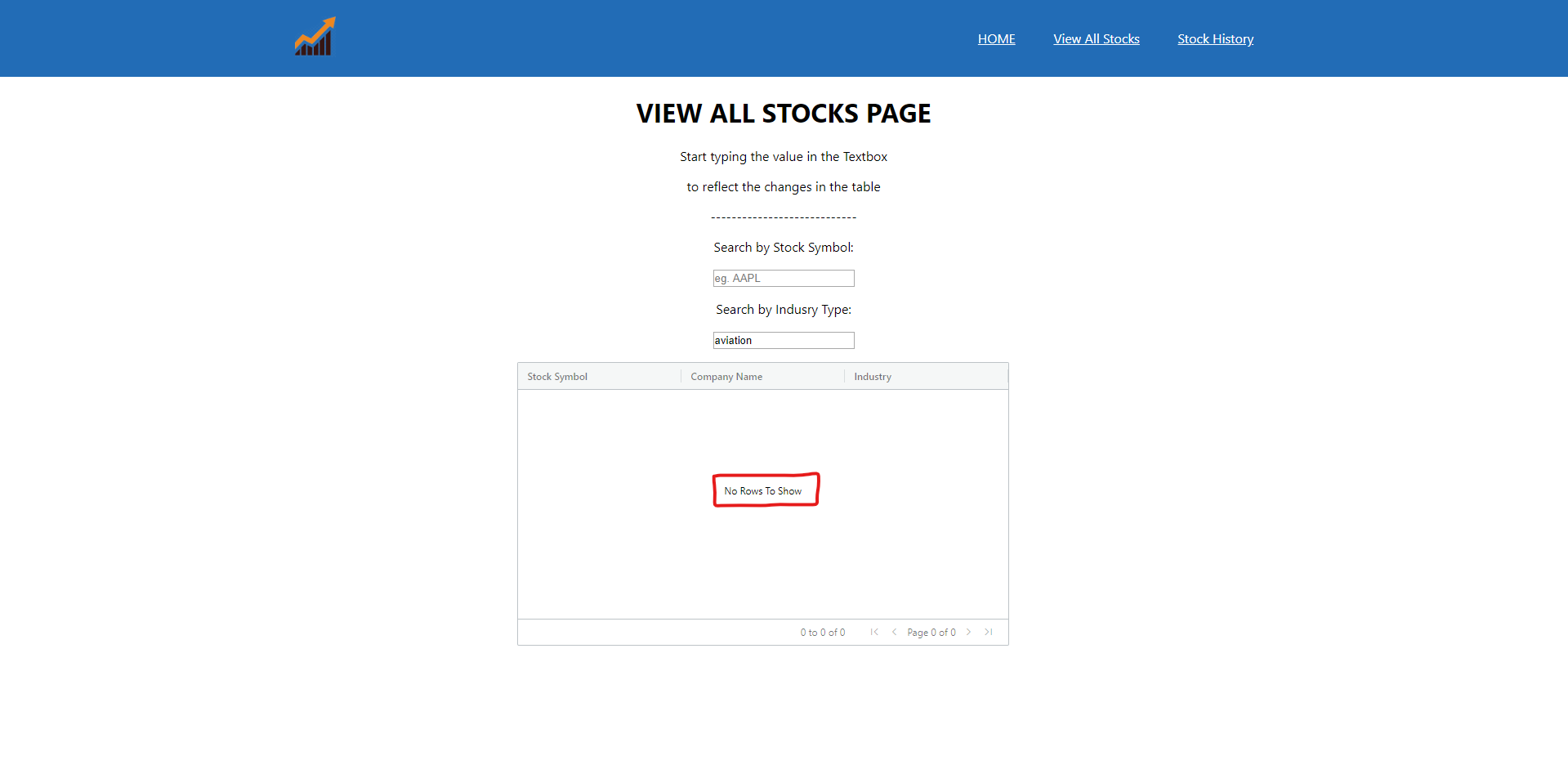


As we can see here that we have two textboxes for searching the values according to the provided input. The search is not Case-sensitive as it will automatically search for all values which have that character either in the beginning or somewhere in between or even at the end. For example, if we want to search stocks which have G in their symbol name we will get the following output:

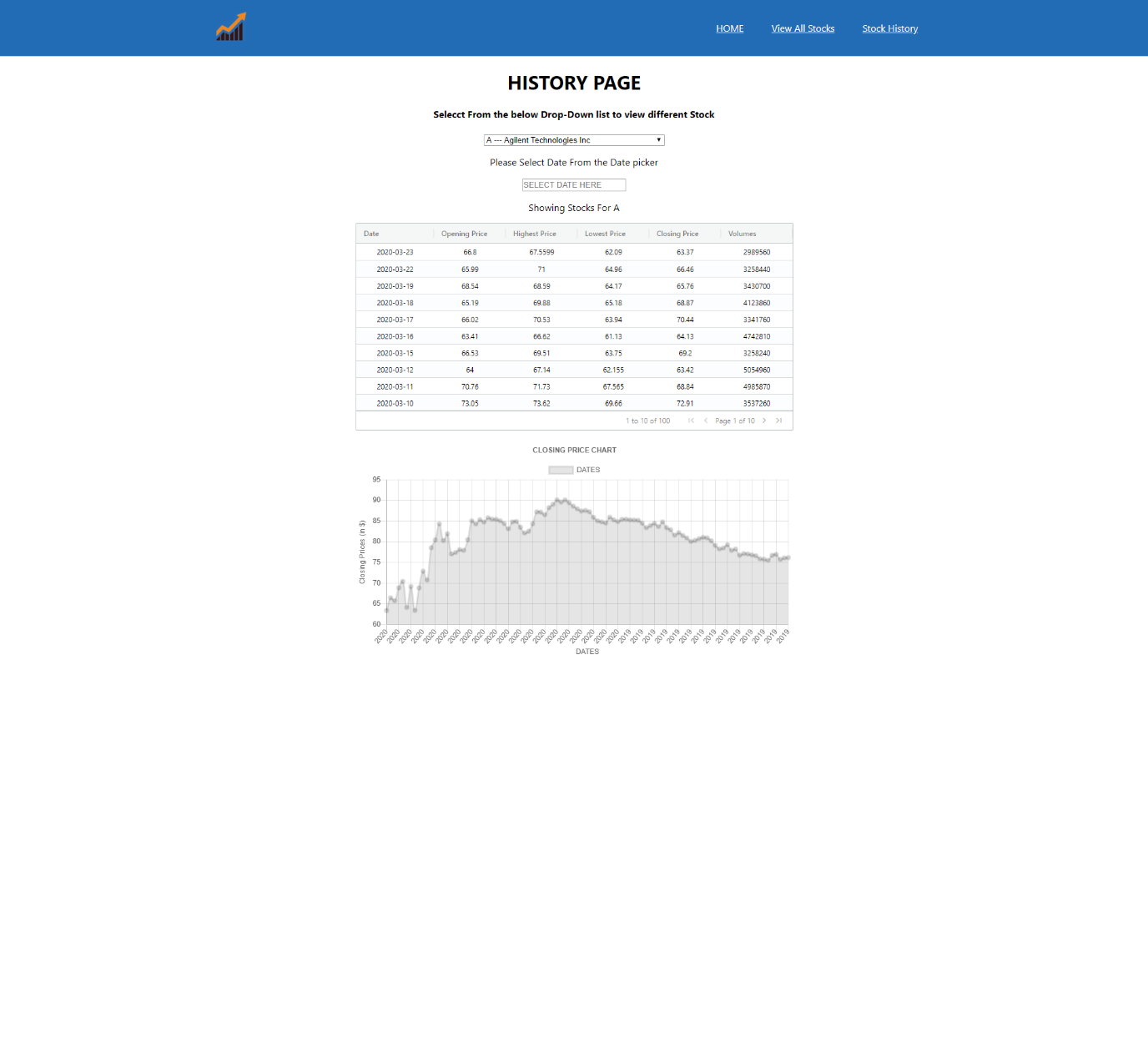


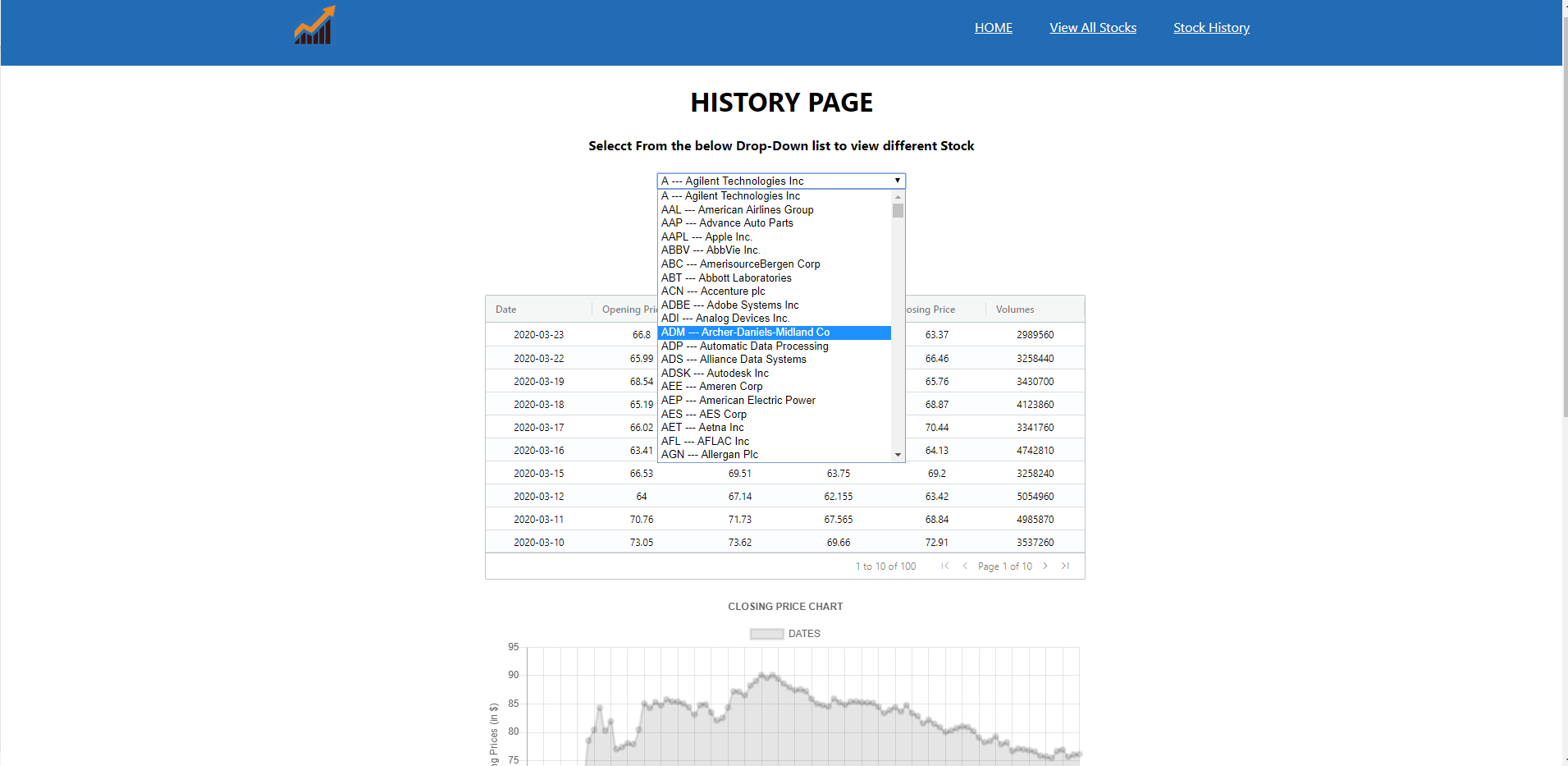
The Ag-grid also provides us the pagination option which is enabled here and as we press the [>] or [< ]symbol we can navigate to the next/previous table page. >| and |< symbols are used to navigate to the last and the first page, respectively.

The second textbox is to search for the stocks belonging to the specific industry. Below is the example which shows the industry which have the character (s) in it. 

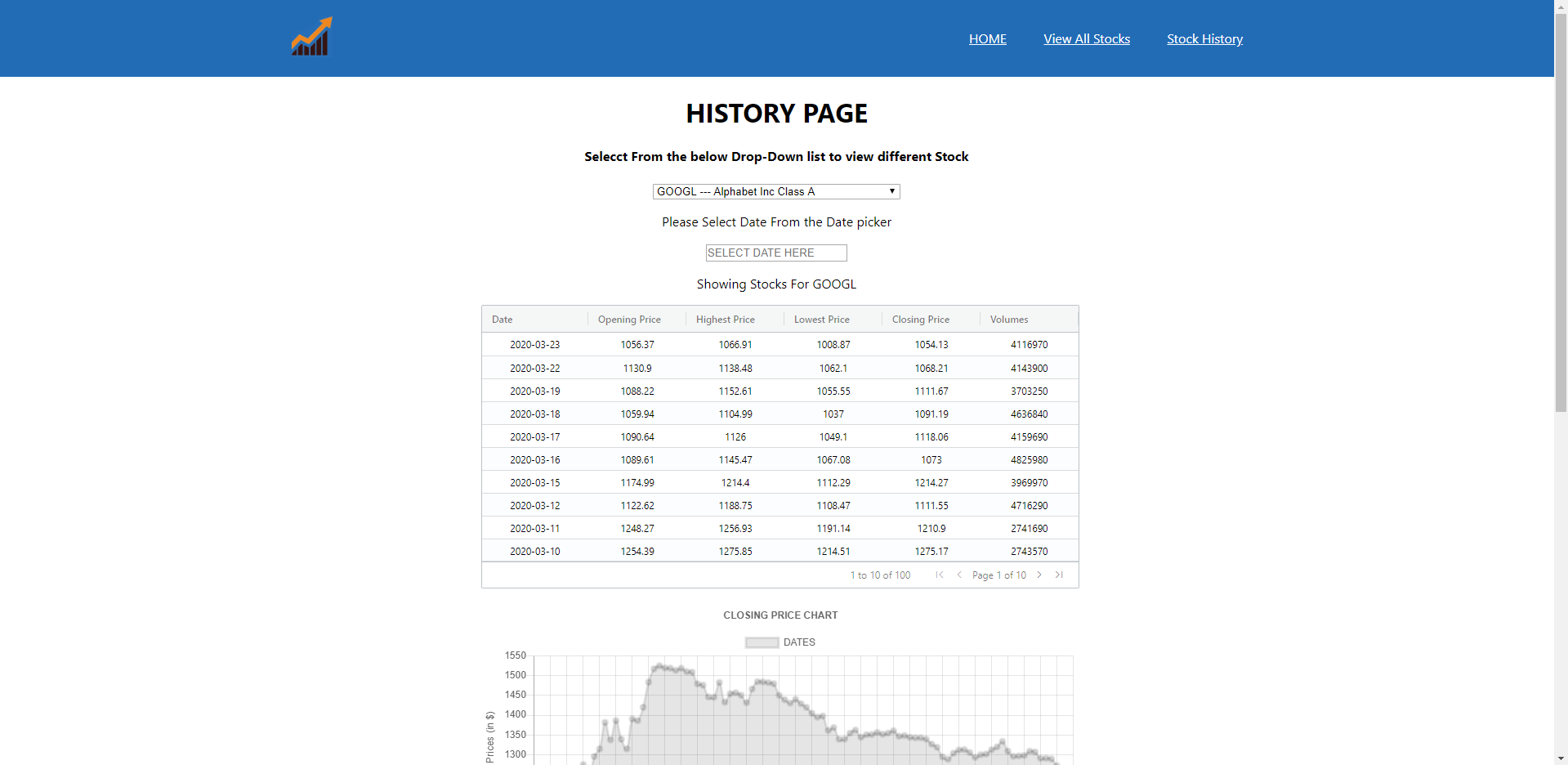
The Ag-Grid will show No Rows To Scow if the extered symbol or industry name you entered is not present in our dataset. Below is the example to that: 

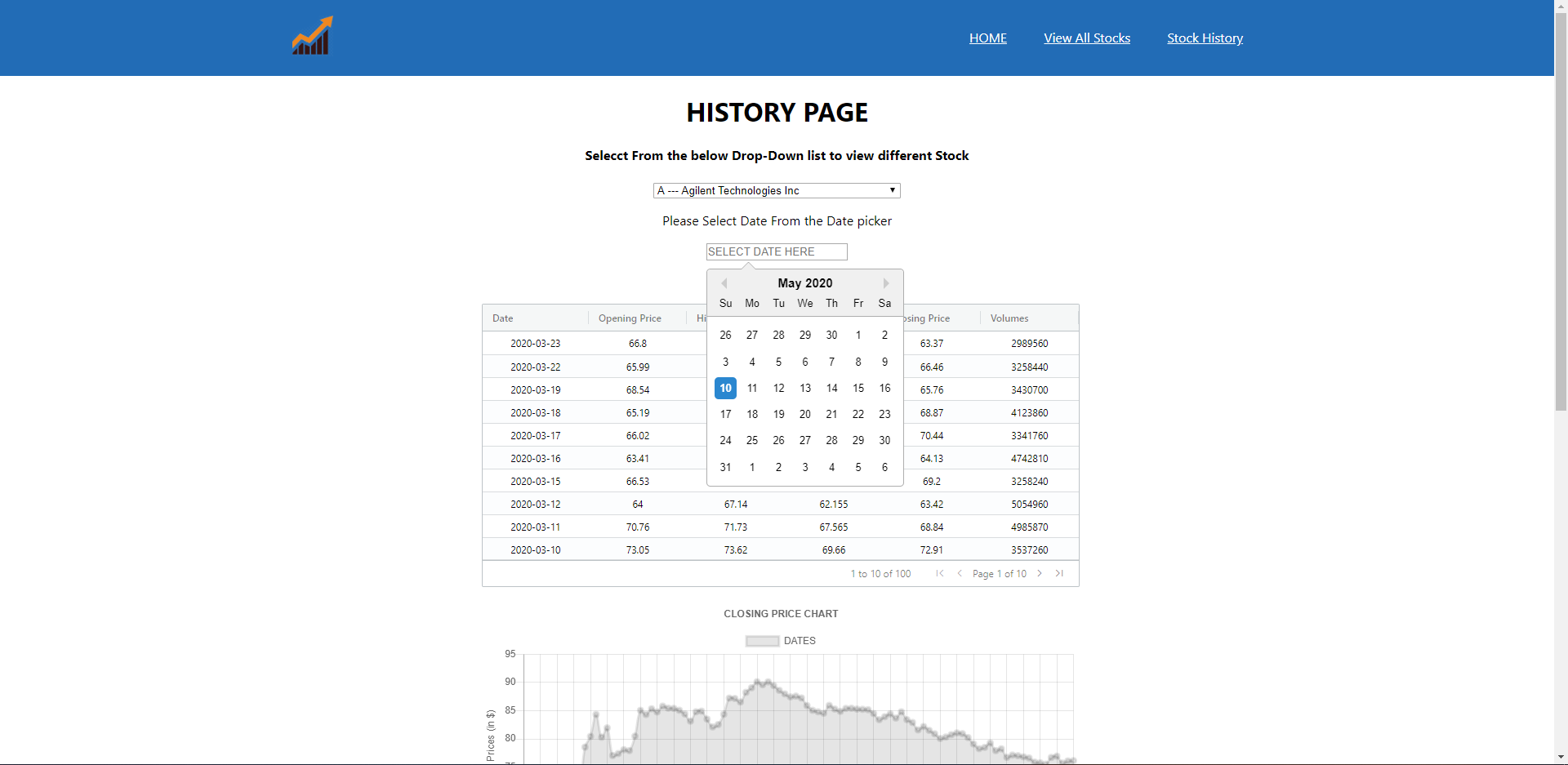
Now we will move to the next link in the Header Stock History Page:

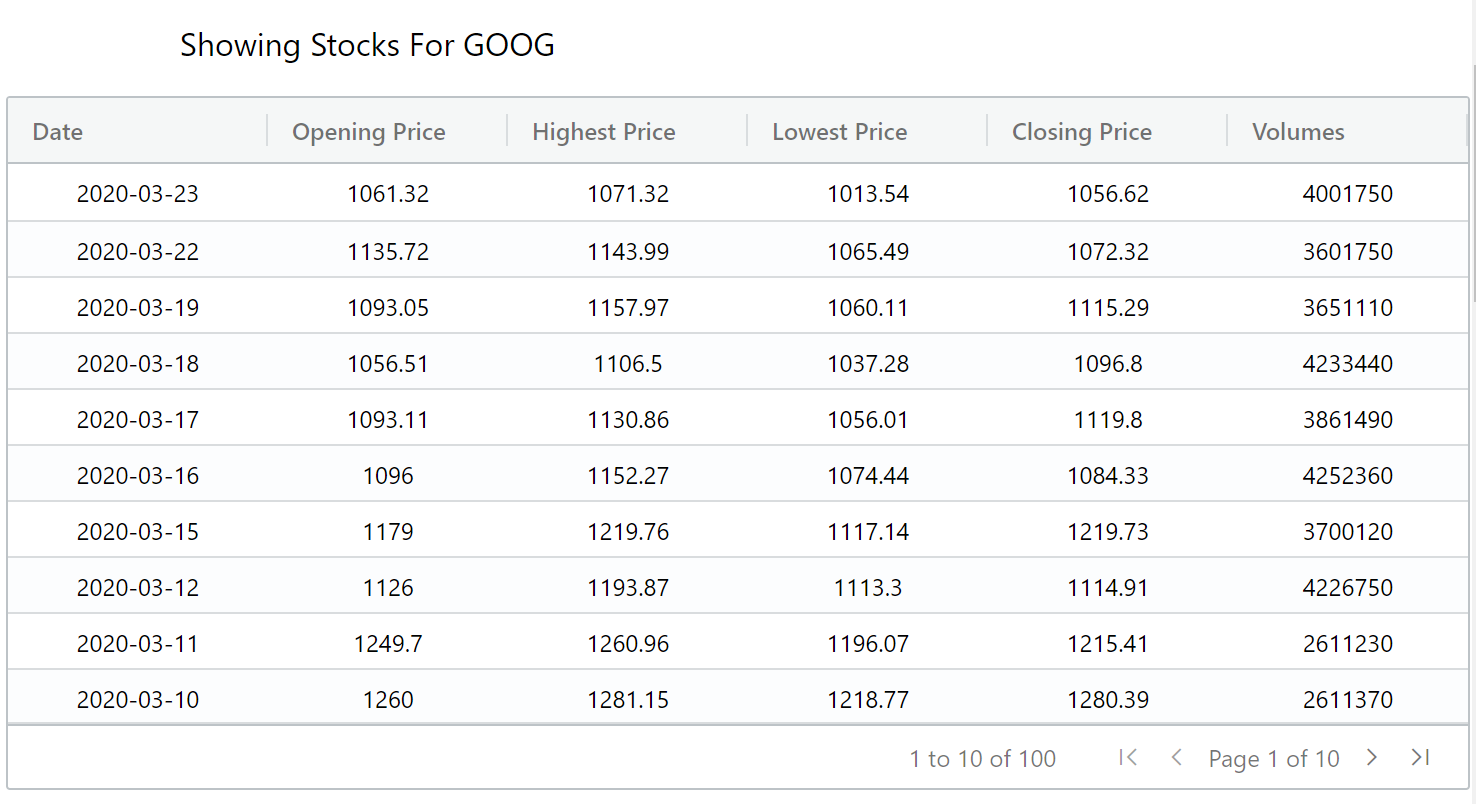
The default view of the Stock History Page is as below: 

This page has the Dropdown list which allows us to select the stock whose history we want to check:

Now if we select any menu item we would get the details of that stock which includes changes in the table and graph as well. For example I have selected the GOOGL stock in dropdown below is the result of that menu item.



Below the drop down menu we have the date picker textbox which allows us to select dace graphically instead of typing the date manually. Below is the screenshot of the date picker 

Below is the Ag-Grid which shows us the data in the table form. This data is changed every time as we change the value of the drop down menu. Below is the look of the stock of GOOG: 

As the value of the Ag-Grid table changes the value of the graph below also change. This graph is the component fo ChartJs library. Below is the screenshot of the same stock symbol GOOG 