Nicholas Havilah-n10469231

Stocks API-Client Side

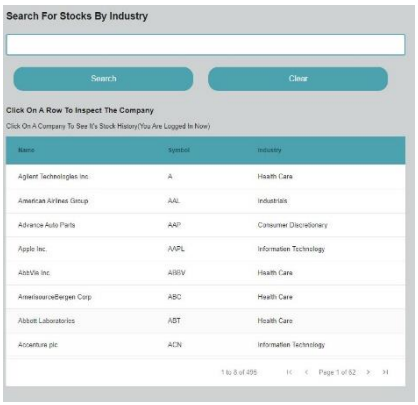
CAB230 Assignment 1 Report

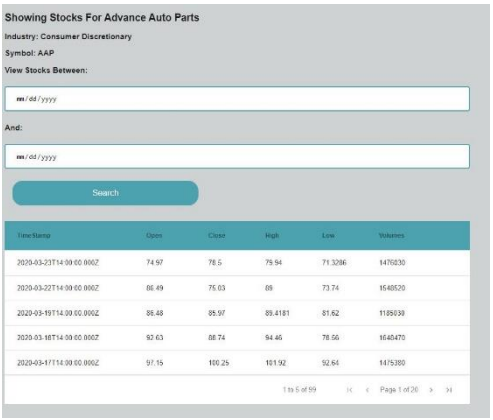
Introduction

Purpose And Description

The app was designed to allow users to view companies and review the performance of their stocks over time. User simplicity was key to this design, and to achieve this, the user can inspect all available companies and select one for closer inspection. This was achieved through interfacing with the Stocks API(a REST API) that retrieves stock market data from a specifically created database.

Key features in this app are centred around the company inspection page. On this page, the user can view a snapshot of the selected company's most recent stock data (if they are not logged in), or a history of the stock data for the given period from November 6, 2019 to March 24, 2020 (if they are logged in). The key highlight of this page is when the user is logged in, where it displays the company's stock history. They can view this graphically, and choose to view the stock's closing values, opening values, highest values, lowest values, or the volumes of stocks traded over time. The other key functionality that increases the simplicity of use is the Stocks By Industry page, where the user can choose what industries they want to see, and they can select a company and inspect it by clicking on a row in the table.





Completeness And Limitations

This app is almost complete but ideally needs some work still with some error handling and how the login functionality is handled. Therefore, this app is worth being awarded a grade of 6 or 7. This is because it completes all of the user endpoints, including the authenticated query routes and presents them in a user-friendly manner. It uses tables and client-side processing to handle the data (through search bars and column-based sorting in the Ag-Grid table), and does not excessively query the server, as the general loading times are under 3 seconds for all data. All navigation is handled through react router, and search parameters are passed into the URL, allowing values to be stored without using any extra memory.

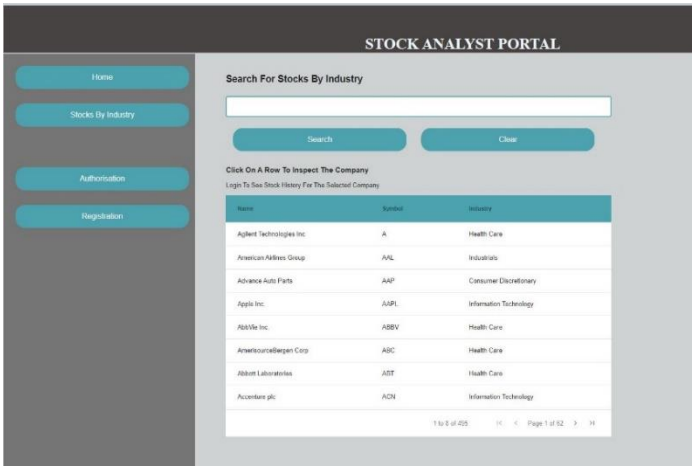
As described above, the app makes use of ChartJs to display data graphically (shown below), which improves the usability of the app, as it allows the user to better understand the data being returned through their searches.

There are some limitations to the app. The way the app shows the user they are logged in could be improved, as it currently pops an alert to notify them, and on some of the other pages, it also indicates whether the user is logged in or not. This could be much more streamlined through having the authorisation button become a login/logout button (depending on whether the user is logged in or not). This would prevent unnecessary code from being developed, which could make it perform better. Secondly, the charts used have tooltips that flicker if the user stays hovering over one area, and they do not disappear when the mouse is not over the chart, which could annoy the user. However, these are only minor issues that could be removed with further time spent on development, and the core functionality has been achieved and extended such that this app is worth being graded as a 6/7.

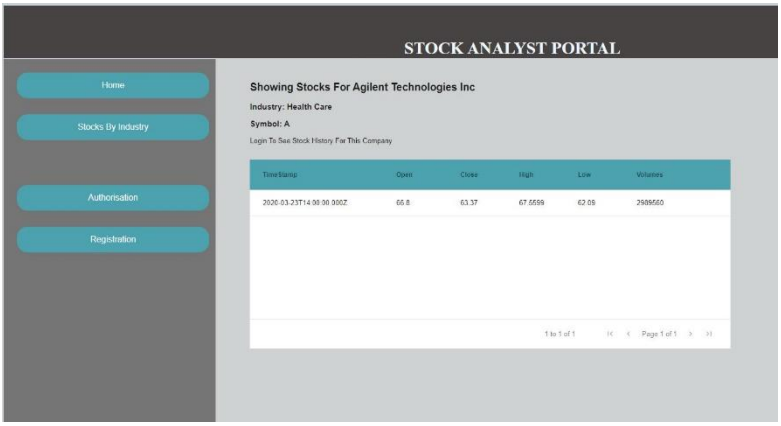
Use Of Endpoints

/stocks/symbols

Here is the page that shows the stocks for all companies of the available industries. (With a drop-down menu for all option in case the user does not know what industries are available, and a search bar for custom search). The table columns can be sorted and the first two can be filtered by name as well. Each row is a link that allows the user to inspect the selected company further.



/stocks/{symbol}

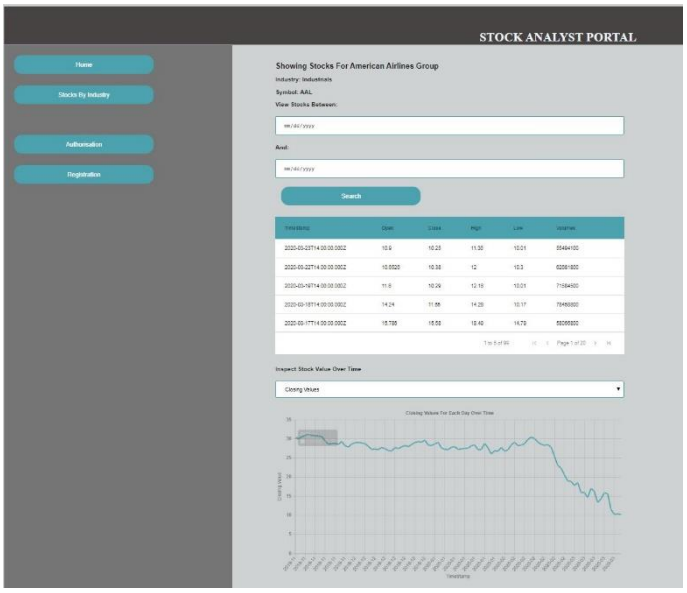


Here is the page that shows a company's most recent stock values. It shows everything presented as a simple table for ease of viewing, and to maintain design consistency through the rest of the website. It also informs the user of the opportunity to see the company's stock history if they login too.

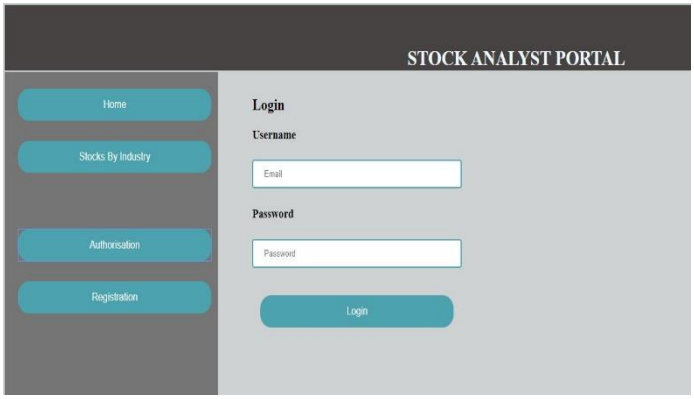
Nicholas Havilah-n10469231

/stocks/authed/{symbol}

Here is the page that shows a company's stock history over time. It shows the opportunity to filter stocks between selected dates, which updates the table and chart below. All the columns apart from the timestamp can be filtered through a number search, and all can be filtered by highest to lowest and vice versa. Beneath the table is the chart, where the user can select the stock data they want to see over time.



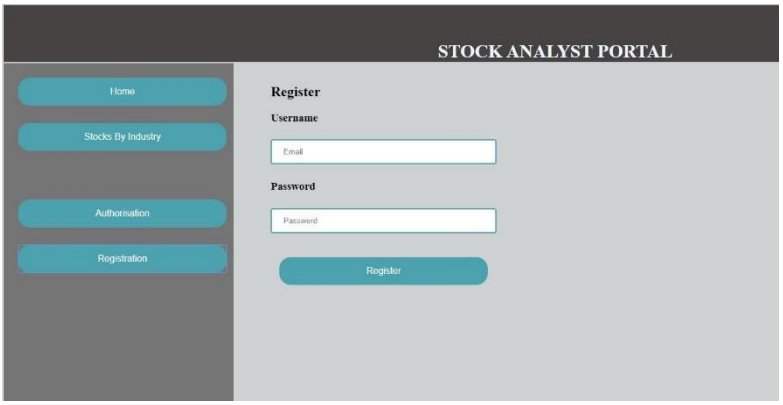
/user/login



Here is the login page. If it does not successfully login, it prompts the user to check their username and/or password. This design was chosen instead of a react form in order to maintain the same styling used by other components throughout the app, creating a more cohesive theme for a better-looking website. On login it also redirects to the homepage and notifies the user that the login is successful through an alert.

/user/register

This uses the same styling as the login page to help maintain a consistent design style throughout the app. However, instead of redirecting after a successful push request(login in the previous page) this presents a message at the bottom of the page that shows whether the user was created successfully or not.



Nicholas Havilah-n10469231

Modules Used

Ag-grid-react

Module to provide fully-featured table components, including sorting and filtering.

https://www.ag-grid.com/react-grid/

React-router-dom

Module to provide smooth navigation between "pages" of the app, by creating declarative routing between components.

npmjs.com/package/react-router

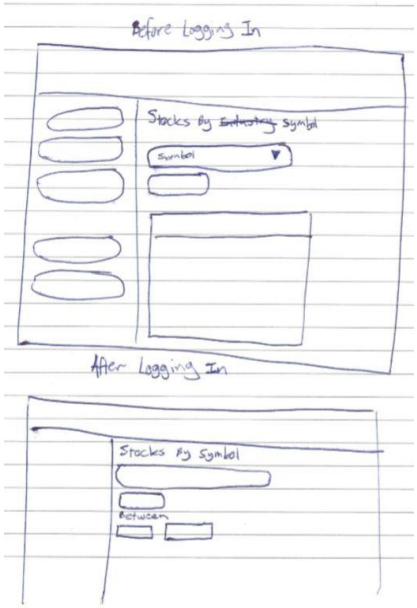
ChartJs-2

Module to display data graphically on the app.

https://www.chartjs.org/docs/latest/

Application Design

Navigation And Layout

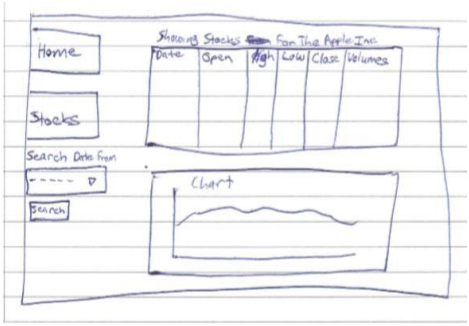


Design for the app was kept relatively simple, in order to make the user experience streamlined and easy to perform. Instead of the traditional routing bar where all navigation is across the header bar, all navigation buttons were laid out along the left-hand border to give the app more of a stocks-dashboard feel. Initially, there was a navigation button to view a particular company by their symbol (see left), with a drop-down menu that allowed the user to change their selected company. Then after selecting a company, it could redirect to a results page (if they were logged in, otherwise it would remain on the same page as the top sketch) to display the stock history.

However, this approach was discarded, as it made the user experience more complex (especially if they did not know what the symbol of their desired company was, and the drop-down menu was creating performance issues and excessively querying the server.

The app flows from one screen to another almost instantly. Throughout testing, it was noted that there were some slight delays with loading data, so a loading wheel was introduced to play until the fetch requests were completed(see right), to be used a basic indicator to the user that the website was not broken, as shown below. This was used for showing the stocks by industry page and the company's stock details pages, as these took the longest to load.

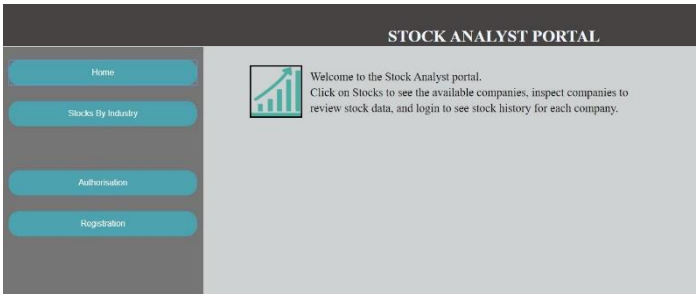
While the design for the company's stock history page conveyed its purpose, the design could be considered slightly awkward. In the current design, everything is laid out underneath each other in a



stack, which is different to many other websites. This is particularly prevalent in the date selectors (right), which would have been better to have next to each other, using the design style shown left. This was not used though, in order to maintain a consistent design style to the other inputs used throughout the app. The chart being positioned under the table may be considered awkward too. Instead, a future iteration of the app could have the chart and table next to each other, or the website could have an option to display one, the other, or both. This would create a better user experience.

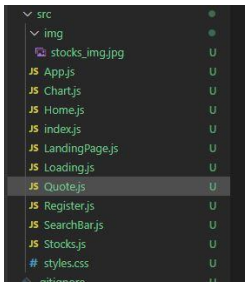
The other awkward design implementation has been discussed above, and that is the login indication. Instead of the current design, the login button could disappear or change to logout when the user successfully logs in.

The rest of the design could be considered a success as it removes the chances of the user pressing a wrong button and breaking the website, and the navigation pipeline is simple to follow. This is also conveyed through the home page, as seen right.



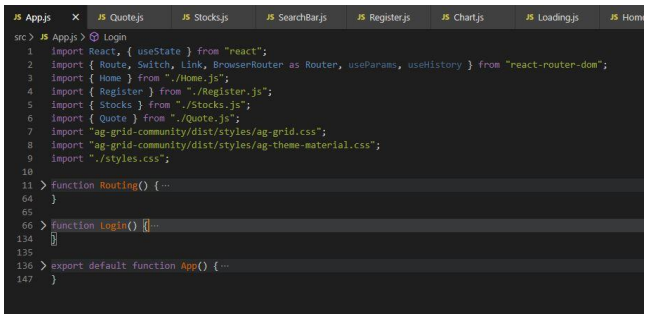
Technical Description

Architecture



The architecture of the project was designed with development in mind, and as such, there are many files used for specific purposes in order to isolate any errors encountered during development. This also meant that certain features could be disabled for testing in order to isolate bugs from the working code, in order to fix them more effectively. Login was the exception here, as trying to move the functionality to another file created issues that were unable to be fixed in time. Meanwhile the routing functionality was also kept in App.js as it a core functionality that worked best when stored inside the main file.

A map for how the App code interacts with the other components throughout the application can be described as such:



The other components call other components in order to create the functional app as needed.

Test Plan

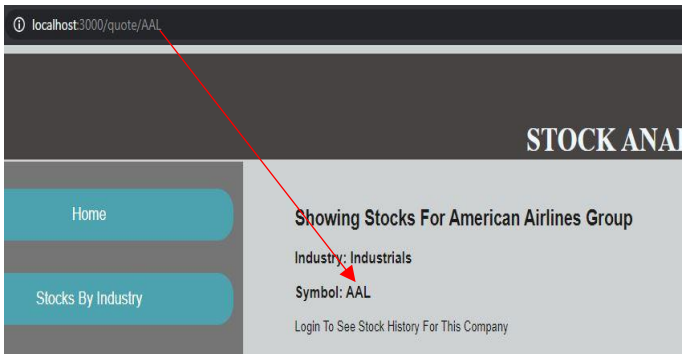
Task	Expected Outcome	Result	Screenshot/s(Appendix A)
View all stocks without having searched for an industry	Search bar, table with all industries displayed	PASS	01
Search for existing industry	Table displays only results belonging to that industry	PASS	02
Search for non-existent industry	Table displays all industries; error message below search bar	PASS	03
Clear search bar using clear button	Table displays all industries	PASS	04
Clicking on row to inspect company data	If not logged in, shows a table with a single set of results. Above the table are some other key characteristics of the company	PASS	05
Register a user	If user does not exist, message below registration confirming registration	PASS	06
Try to register existing user	Message confirming user already exists	PASS	07
Try to register without completing form	Message below form requesting email and password	PASS	08
Login	Login form disappears and is replaced with a logout button	PASS	09
Try to login with wrong username or password	Message below form telling user to check their username and/or password	PASS	10
Clicking on a row to inspect company data while logged in(from the stocks by industry page)	Shows the company name and some information, a pair of date selectors, a table with the rest of the stock data (with more rows now as the history is accessed), and a chart displaying the table information graphically	PASS	11
Use the date selectors to filter the results by date	Results are reduced to match the search parameters passed from the date pickers. When selecting the final date, it automatically forces the minimum date to be the value of the other picker	PASS	12
Change the content displayed on the chart	Name of chart, axis titles, and data all changes on the chart when the drop-down selection is changed	PASS	13
Logout	On button press, the login form returns	PASS	14

Difficulties/Exclusions/Unresolved & Persistent Errors

One of the major roadblocks that came with getting the current iteration of the app to work was in the way data was passed through to the stock information page. Initially, states were passed but this proved to be unreliable, so another approach was used. React router has a feature called useParams, which can be used to pass information through the url:

Nicholas Havilah-n10469231

Localhost:3000/stocks/Industrials

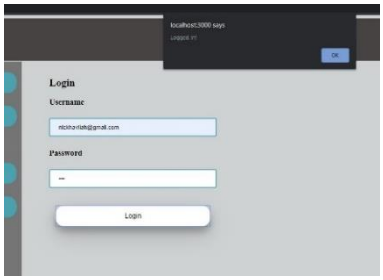


The last part of the URL was treated as a variable, so when the user searched for something the page would re-render, with the new search term being used, as shown left.

This also was effective for allowing the user to search through the URL as an alternative, to increase the usability of the app.

Another major roadblock in the development process, was in trying to convey to the user that they had successfully logged in. Trying to switch the login button to a logout button was unsuccessful, as it never changed the button.

So instead of continuing trying to fix it, an alternative solution was implemented to make progress and save time, using an alert to inform the user they had logged in successfully, and then some minor text prompts on some of the other pages to help inform the user if they weren't logged in, as seen below:



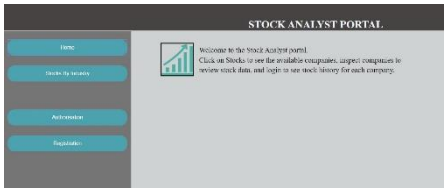
There appears to be only one bug, as the rest were aesthetic features that could be improved with more development. This bug involved the tooltip on the chart. When holding the mouse still, the tooltip flickers constantly, though this flickering disappears when the mouse moves around on the chart. The final part of the tooltips unusual behaviour is when the mouse goes off the chart, the tooltip does not disappear, and continues with the flickering behaviour as described above.

This could be the result of the data being constantly reloaded, or one of the options for the tooltip was not enabled. It can only be determined through further testing and development.

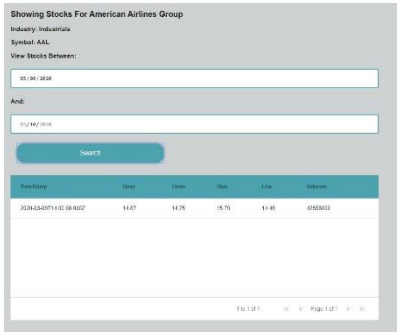
Nicholas Havilah-n10469231

User Guide

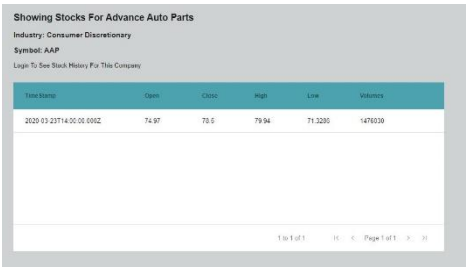
You will open to the home screen:



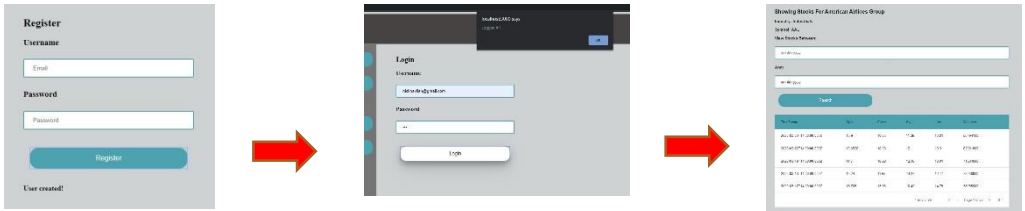
Click on the “Stocks By Industry” button to view the industries and the companies associated with them. You can then search for specific industries in the search bar:



Click on a row of the table to inspect the company’s stock data (optional filters are in the column headers):



To see the company’s stock history, register, then login (or if you are already registered, just login). After that return to Stocks By Industry and find the company you want to see the stock history for (optional filters for the table are in the column headers): (The user can also filter stock history between dates using the date pickers on this page too)

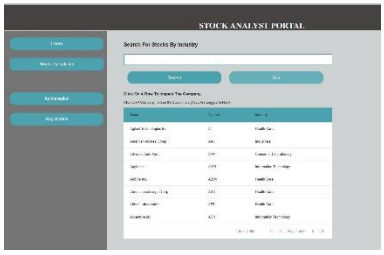


For the graph of the company’s stock history, choose what information you want to see over time:

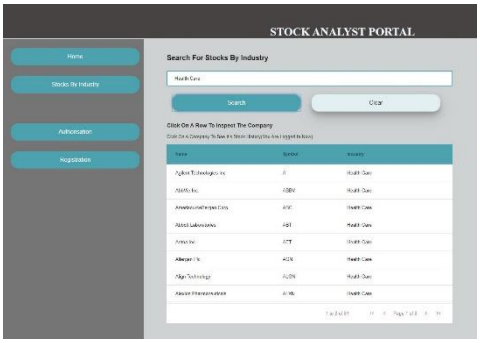
Nicholas Havilah-n10469231

Appendix A-Screenshots For The Test Plan

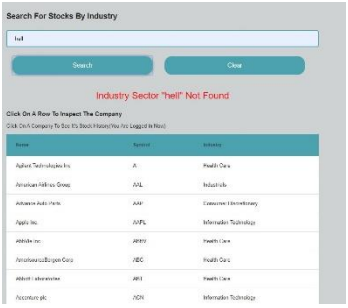
01



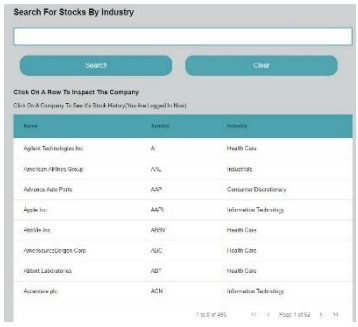
02



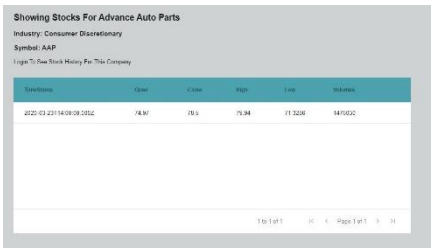
03



04

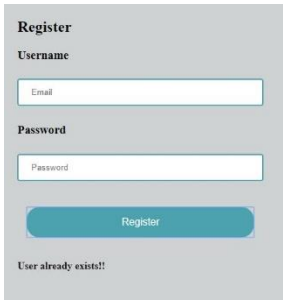


05



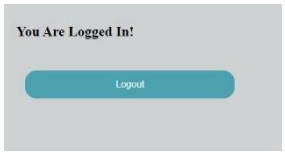
06

07



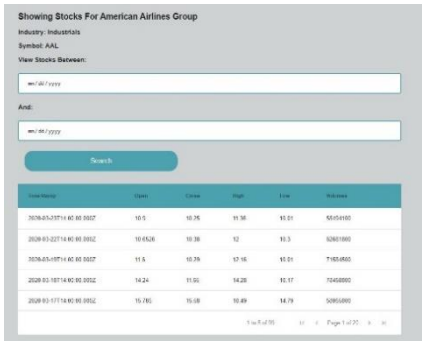
08

09



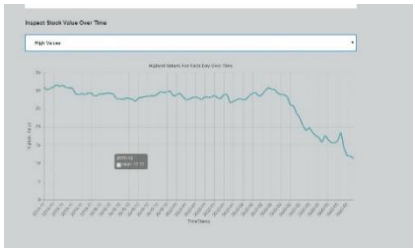
10

11



12

13

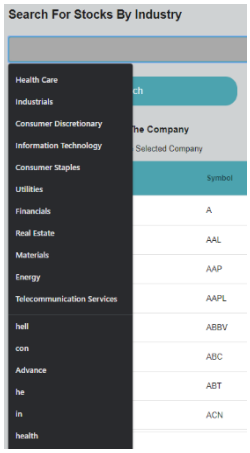


14

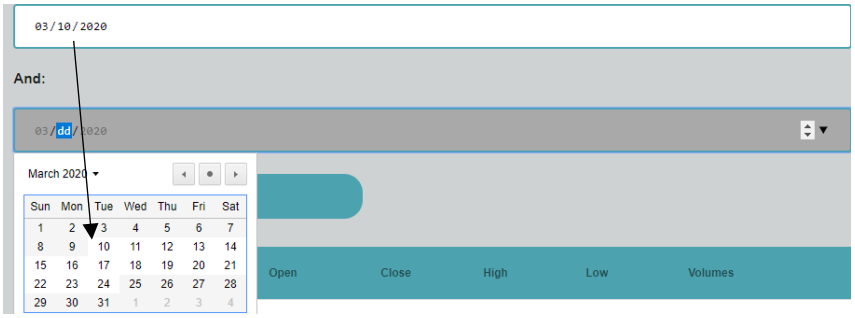
Appendix B-Drop Down Menus

In the screen recordings, it was unable to show the drop-down menus being used, so here is the snapshots of the menus used.

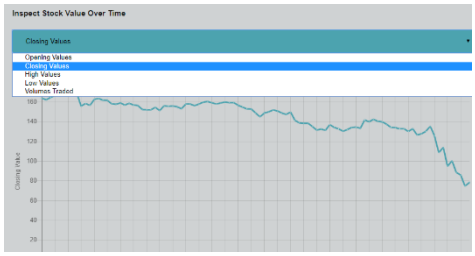
1-Search bar drop-down menu



2-Date selector drop down-note there is a limit on the minimum date, based on the value of the first date selector



3-The ChartJs data type selector



4-Ag-grid-react table filtering example