William Ee

N10494448

CAB230 Stocks API – Client Side



CAB230

Stocks API – Client Side Application

Contents

[Introduction 2](#_Toc40782695)

[Purpose & description 2](#_Toc40782696)

[Completeness and Limitations 2](#_Toc40782697)

[Use of End Points 3](#_Toc40782698)

[/stocks/symbols 3](#_Toc40782699)

[/stocks/{symbol} 4](#_Toc40782700)

[/stocks/authed/{symbol} 4](#_Toc40782701)

[/user/register 6](#_Toc40782702)

[/user/login 6](#_Toc40782703)

[Modules used 7](#_Toc40782704)

[Ag-grid-react 7](#_Toc40782705)

[React Router Dom 7](#_Toc40782706)

[Chart.js 7](#_Toc40782707)

[Jwtwebtoken 7](#_Toc40782708)

[React Strap 7](#_Toc40782709)

[Application Design 7](#_Toc40782710)

[Navigation and Layout / User Guide 7](#_Toc40782711)

[Technical Description 11](#_Toc40782712)

[Architecture 11](#_Toc40782713)

[Test plan 12](#_Toc40782714)

[Difficulties 14](#_Toc40782715)

[References 15](#_Toc40782716)

## Introduction

### Purpose & description

The purpose of the stock client application is to allow users to view and filter a list of stocks, select a stock and view quote information, and select quote information from a series of dates. The application draws its data from a database through REST API queries.

The application was produced using React, and a series of modules which are specified further in the report.

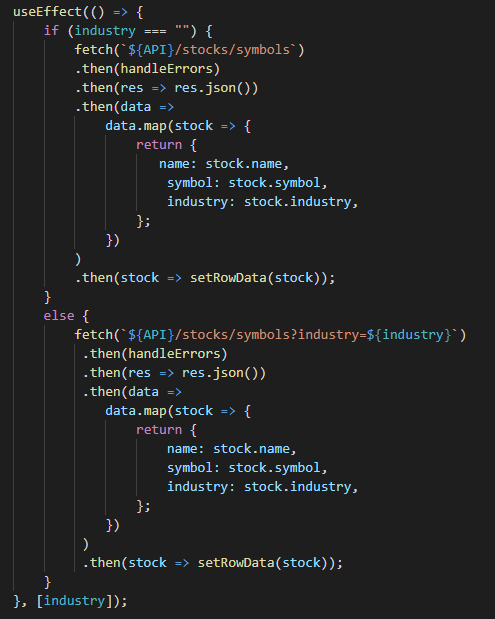
### Completeness and Limitations

|  |  |
| --- | --- |
| **Element** | **Completeness** |
| Navigation using React Router | Navigation implements React-Router successfully: Router, Switch, Route and Link for the primary navigation. UseHistory has been used for secondary navigation functionality e.g. redirecting to home page upon successful login or redirecting to quote page upon clicking on a specific stock. However, after a successful login, one weakness is that it not only redirects, but it has to refresh on that page in order to for changes to be seen on the page e.g. (login and register buttons 🡪 logout button), this is not significant, it is just slightly clunky. |
| Forms | Forms have been implemented successfully to register, login and logout in order to generate a jwt-token used for accessing price history functionality. |
| Charting | A line chart has been implemented using chart.js for open, close, high and low data. No chart has been created for volumes however. Chart becomes difficult to read when there are a significant amount of values being plotted. |
| Styling | Styling has been used to create a sufficiently visually appealing site, that follows basic principles of web design. The web application has some responsive design particularly the navigation bar that turns into a hamburger menu. |
| User endpoints | User endpoints including authenticated routes work as expected. When searching for an industry (filtering through search bar) it may query the server quite a lot (whenever the value inside the search changes). |
| Table | Data is displayed in an ag-grid react table. There is certain sorting features e.g. ascending descending order on tables. Tables are paged. |
| Application robustness | Application handles errors fine |
| Implementation of code | The implementation of code is somewhat inefficient and is not at a high quality. This is because the focus whilst implementing was solely to make all the desired functionalities work, rather than efficiency and code quality e.g. there are some components that are excessively long, with several functions and sub-components. |

## Use of End Points

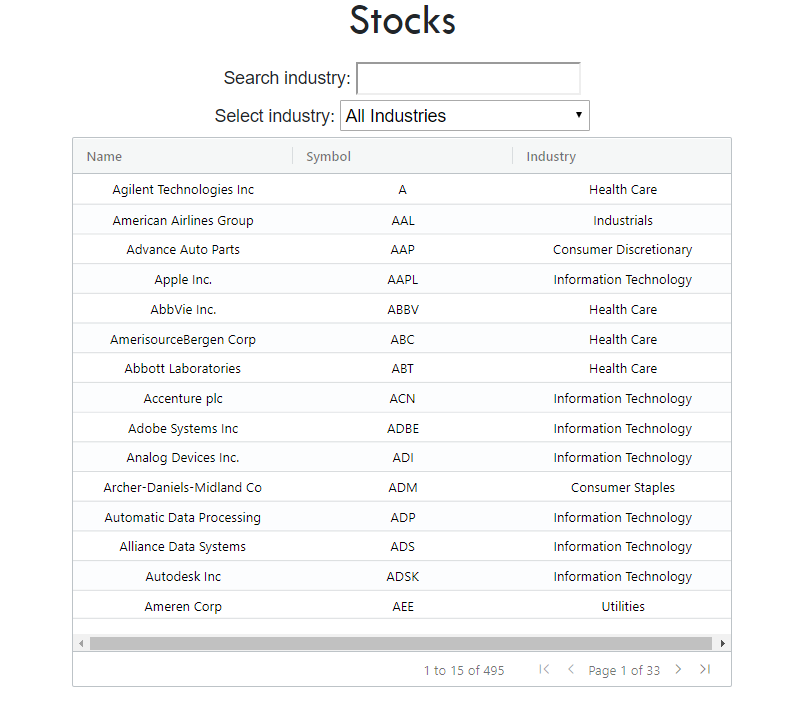
#### /stocks/symbols

This end point returns all available stocks from the database. This can be optionally filtered, by adding an industry to the fetch URL.



Whenever the industry is updated through a dropdown or search bar, or upon initial render, it uses the /stocks/symbols end point. If no industry has been specified, it returns the default end point without industry filtering. Otherwise, it adds industry to the URL endpoint to return data only of that industry.

Figure 1 Use of /stocks/symbols



Search bar and dropdown that update the industry filter

Figure 2 Result of /stocks/symbols

#### /stocks/{symbol}

This endpoint returns the latest entry for a stock. The stock selected is based on the symbol provided.



This endpoint is used whenever the symbol is changed or when the page is rendered, for this application these occur at the same time (when the user selects a particular stock, it redirects to the quote page and stores the symbol to be used). This endpoint is only used when the user is not logged in. The data returned is a single object of the latest entry for the stock.

Figure 3 Use of /stocks/{symbol}

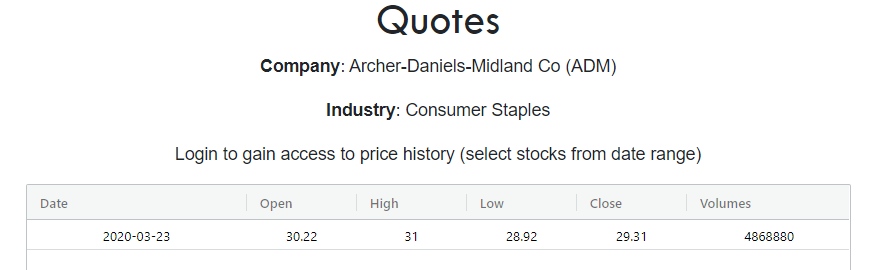


Figure 4 Result of /stocks/{symbol}

#### /stocks/authed/{symbol}



This endpoint returns the entries of stock searched by symbol.

This route is an authenticated route. The authentication of this route is completed here, and will only return values if the jwt-token is valid.

The route displayed only returns the most recent value similar to /stocks/{symbol} as seen below in figure 6. This is used when no dates have been selected yet to filter by.

Figure 5 Use of /stocks/authed/{symbol} no dates selected

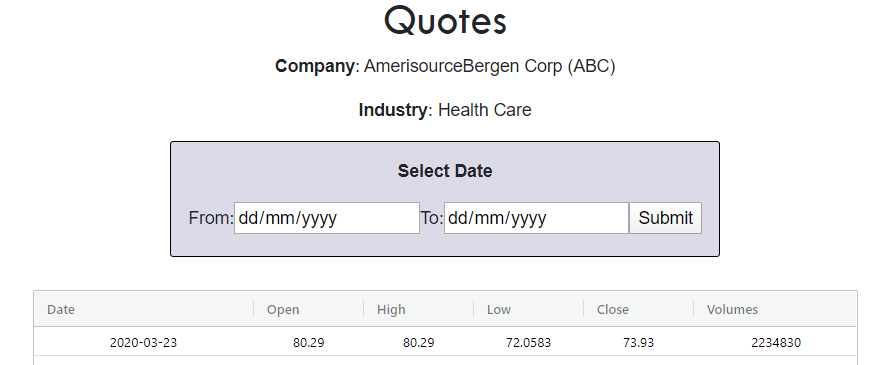


Figure 6 Result of /stocks/authed/{symbol} no dates selected



This route does filter by dates returning values between the from and to date.

Likewise it is an authenticated route which can only be used with a valid jwt-token.

This end point is only used when the user is logged in and has selected at least one date.

Figure 7 Use of /stocks/authed/{symbol} with dates selected

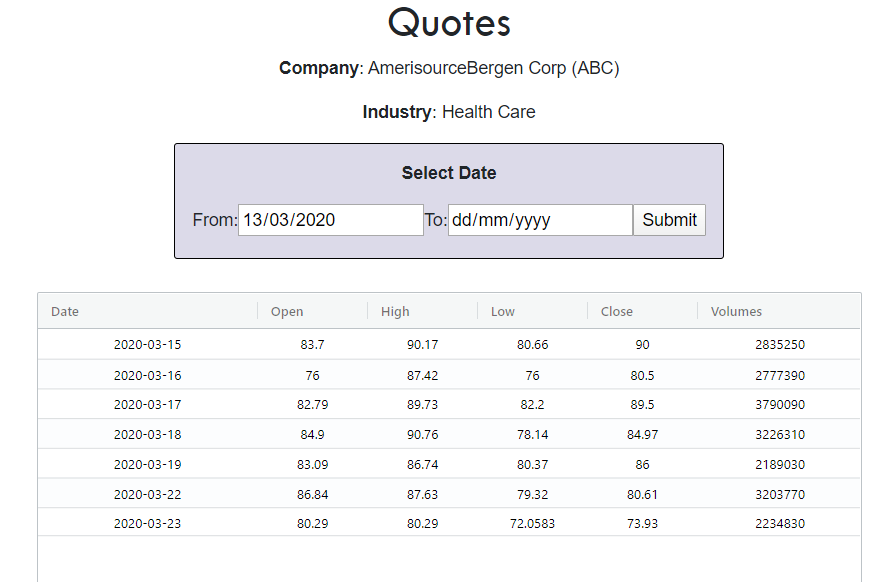


Figure 8 Result of /stocks/authed/{symbol} with dates selected

#### /user/register



This route is used to create a new user.

This is done through posting an email and password.

Data returned is whether or not the registration was successful.

Figure 9 Use of /user/register

#### /user/login



This route is used to login to an existing account by submitting an email and password.

Once successfully logged in, stores the the jwt-token the server provides.

This is done through posting an email and password.

This application considers a user to be logged in if a valid jwt-token has been stored in localStorage.

Figure 10 Use of /user/login

### 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 establishes a router that handles all the pages, each page with a specified route. This enables navigation to a certain page through a route. Used for navigating between pages, redirecting and storing symbol as a param.

<https://www.npmjs.com/package/react-router-dom>

#### Chart.js

Module to create a graph using data.

<https://github.com/chartjs>

#### Jwtwebtoken

Module used to decode web-token to extract token expiry date.

<https://www.npmjs.com/package/jsonwebtoken>

#### React Strap

Contains pre-made components. In this application its responsive navigation bar was utilized.

<https://www.npmjs.com/package/reactstrap>

<https://reactstrap.github.io/components/navbar/>

## Application Design

### Navigation and Layout / User Guide

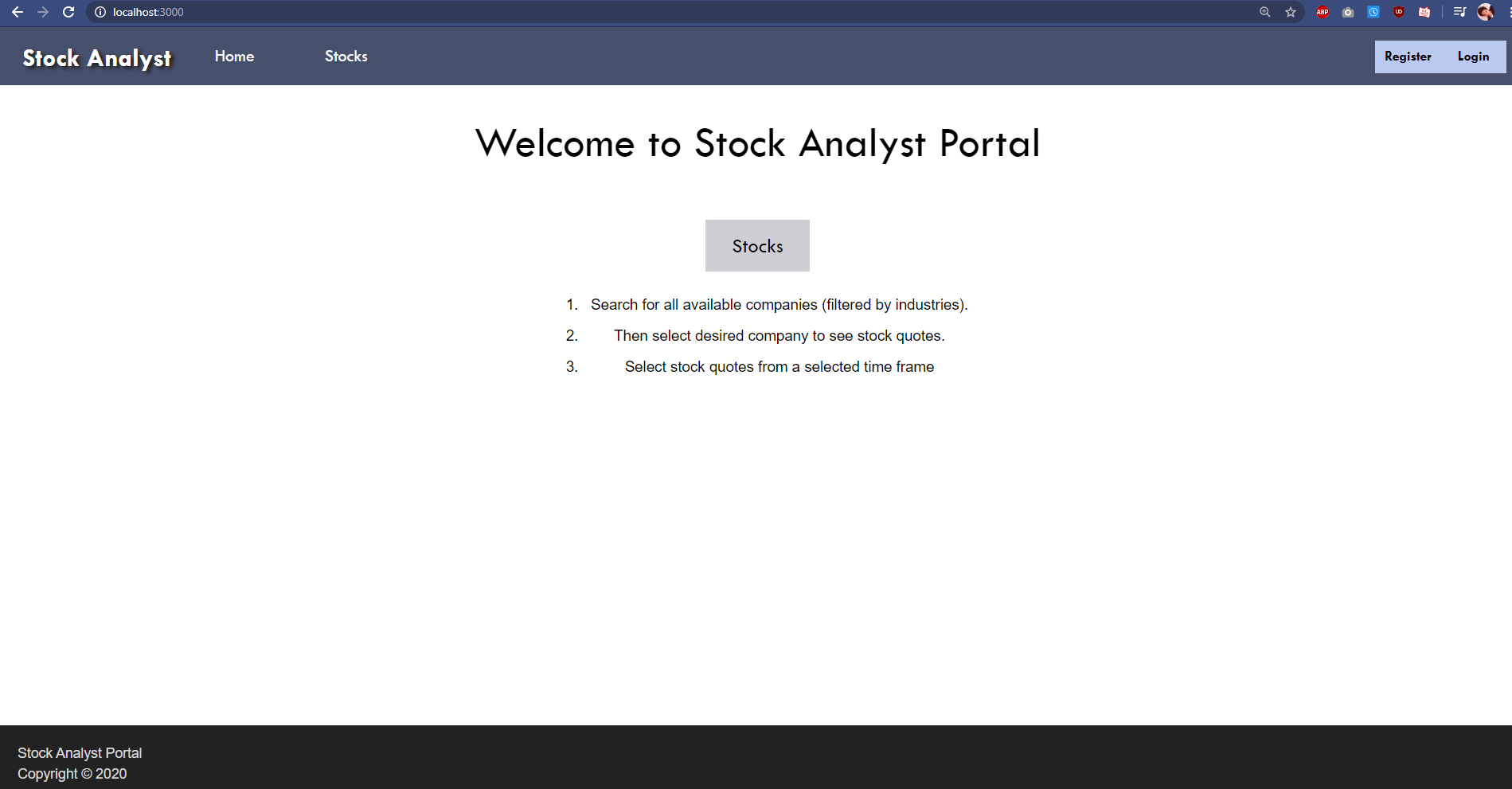


Figure 11 Landing Page

After running npm start, the browser opens up the landing page seen above. This has a button linking to the stocks page, and general info about functionality of the application. The home page although not particularly important as it does not have any primary functionalities, however it was included as simply an introduction to the application, similar to a hotel lobby for a hotel (Babich, 2017). For this purpose, the stocks page would not be suitable as it simply presents a bunch of information from the get-go. Furthermore, if the application were to be extended to have more functionalities outside of the stocks and quotes, it would not be suitable for it to lead the user to the stock functionality immediately, as the user may not wish to use the stock functionality.

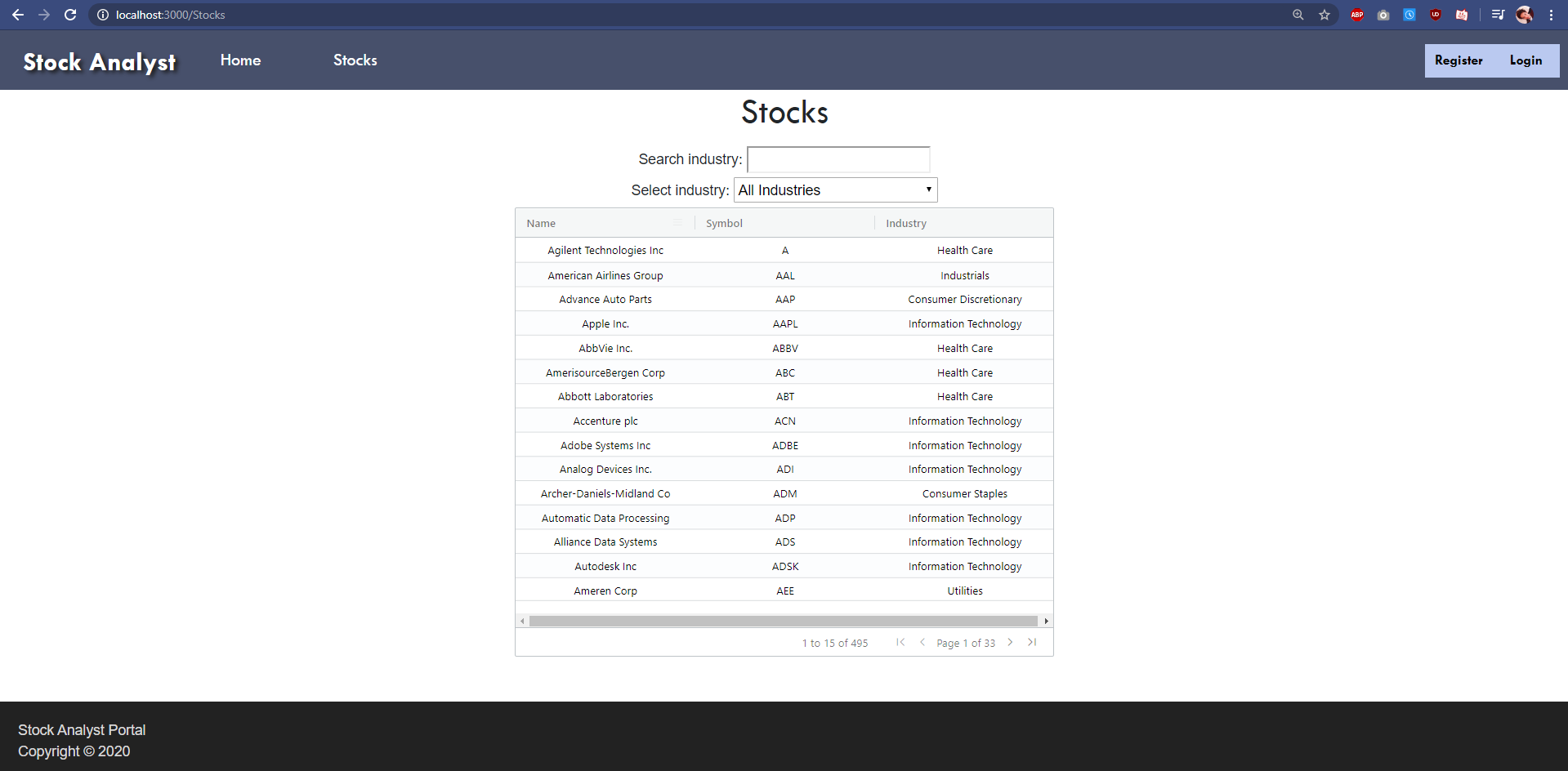


Figure 12 Stocks Page

To access the stock page, either the stocks button in the home page, or the stocks navigation on all pages in the nav bar could be clicked. Initially, it was planned to be able to filter by more than one industry e.g. selecting Utilities, Health Care and Information Technology, however upon looking at the end point more, this was not convenient to implement. The search bar is slightly choppy, as it updates the state of the industry every time its value is changed, causing the table to be changed frequently or having an error message pop up. A timer was considered to be placed on the useEffect which updates the table whenever the industry changes, or the use of a submit button for the submitting a search query (which will result in less updates), however ultimately it was decided that the choppiness of updates giving instantaneous feedback, was preferred over a smoother experience that had less updates that did not provide feedback instantly. Clicking on a particular stock in the table will redirect to the quotes page.

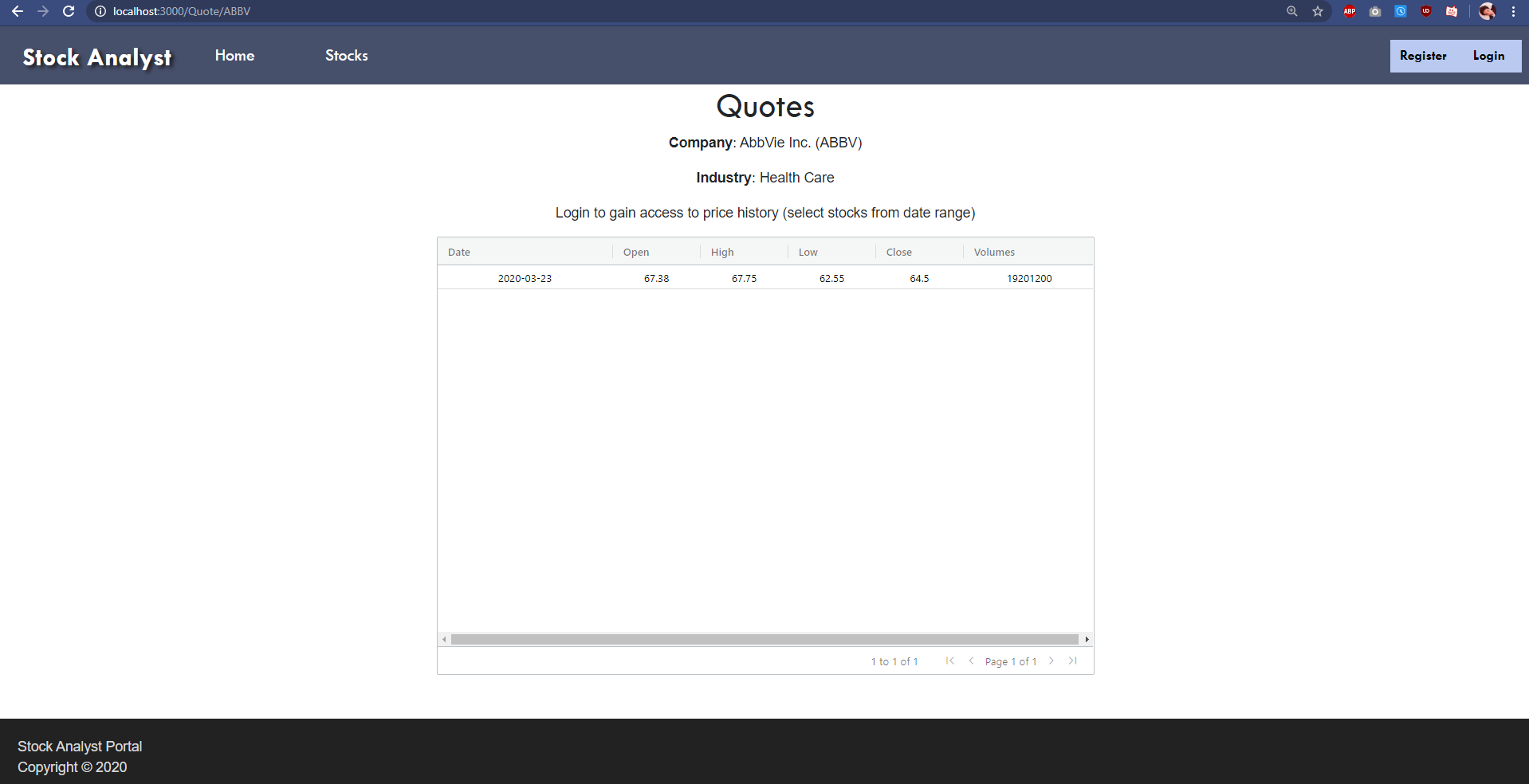


Figure 13 Quotes Page

After clicking on a particular stock, the user is redirected to the quotes page, already displaying information for that particular stock. This was done by accessing the symbol through useParams. Initially it was planned to have a search feature and dropdown selection like the stock page, and have the stock and quotes page independent of each other. However, as there is an extremely long list of potential symbols compared to industries, it was not reasonable to implement a dropdown. Additionally, stock symbols are fairly abstract, unlike industries e.g. AAL vs Health Care, and therefore a normal user would not search for a stock symbol or be aware of the stock symbol they are looking for. As the quotes page is dependent on the results from the stocks page, the link to the quotes page was removed from the nav bar, as it would not return any results as no symbol would be inputted yet. One disadvantage of removing this link however, is that the only way back to the page is through the stocks page. As the application currently is simplistic in nature without many pages and functionalities this is not an issue, however if it were to be extended upon, the user may forget how to reach the quotes page.

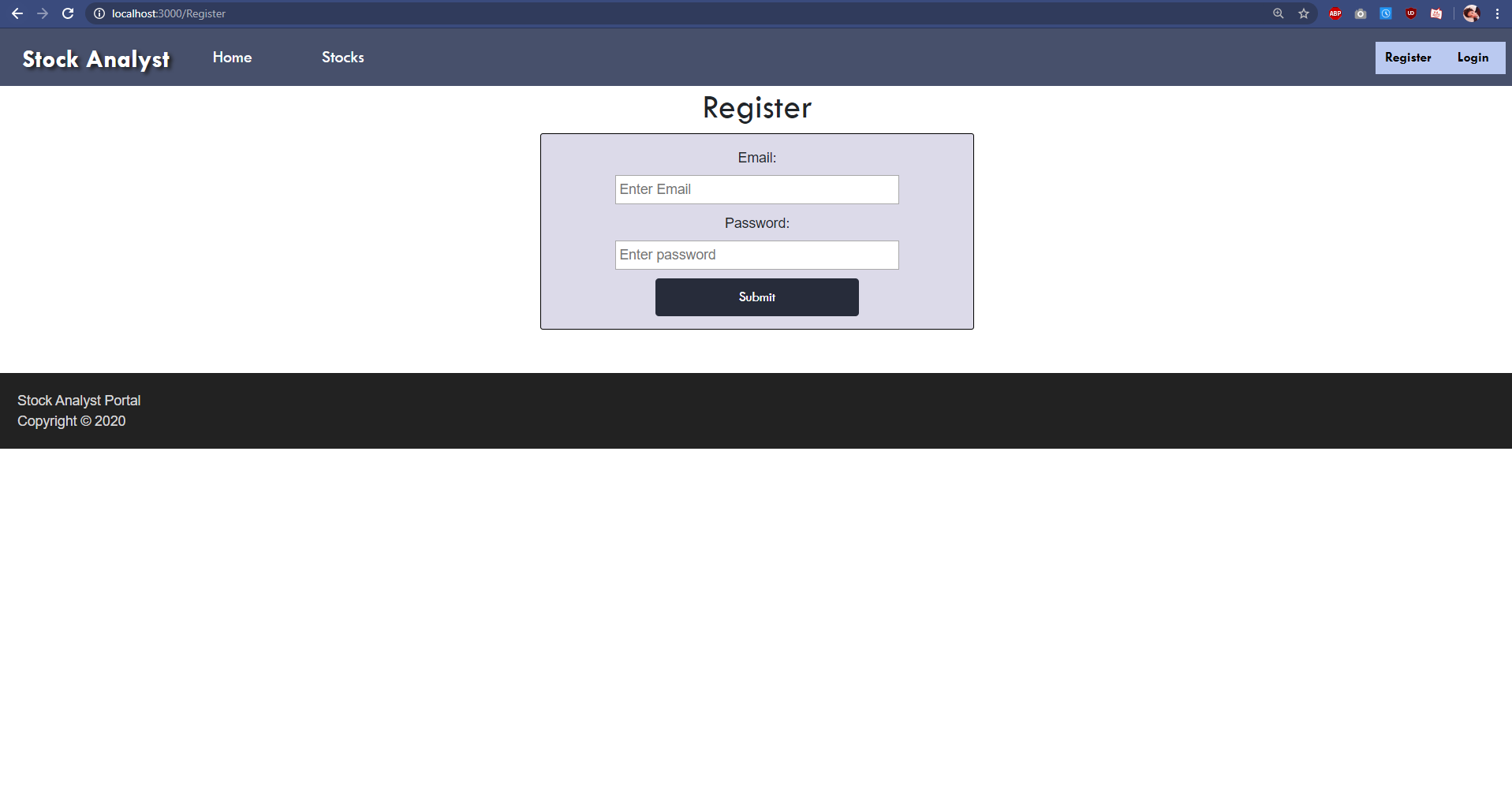


Figure 14 Register Page

The register page can be accessed when the user is not logged in, through clicking on the register button in the nav bar. After the user has entered an unused email, and a password they will have created an account and will be redirected to the login page to login.

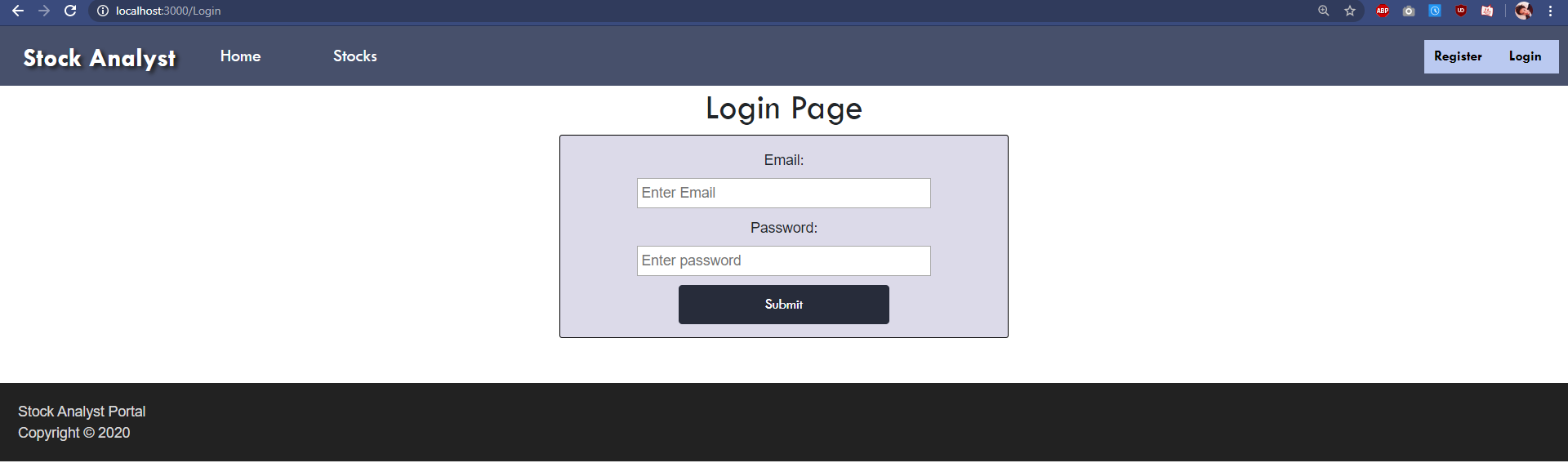


Figure 15 Login Page

The login page can be accessed when the user is not logged in, through clicking on the login button in the nav bar, or being redirected after registering. After the user has entered a valid email a password they will be redirected to the home page and logged in.

Register and login were planned to be pop up forms rather than their own individual pages, however it was simpler to just create pages rather than pop-ups. There is a lot of white space below the register and login page’s footer which could be fixed with adding a margin to the bottom of the form.

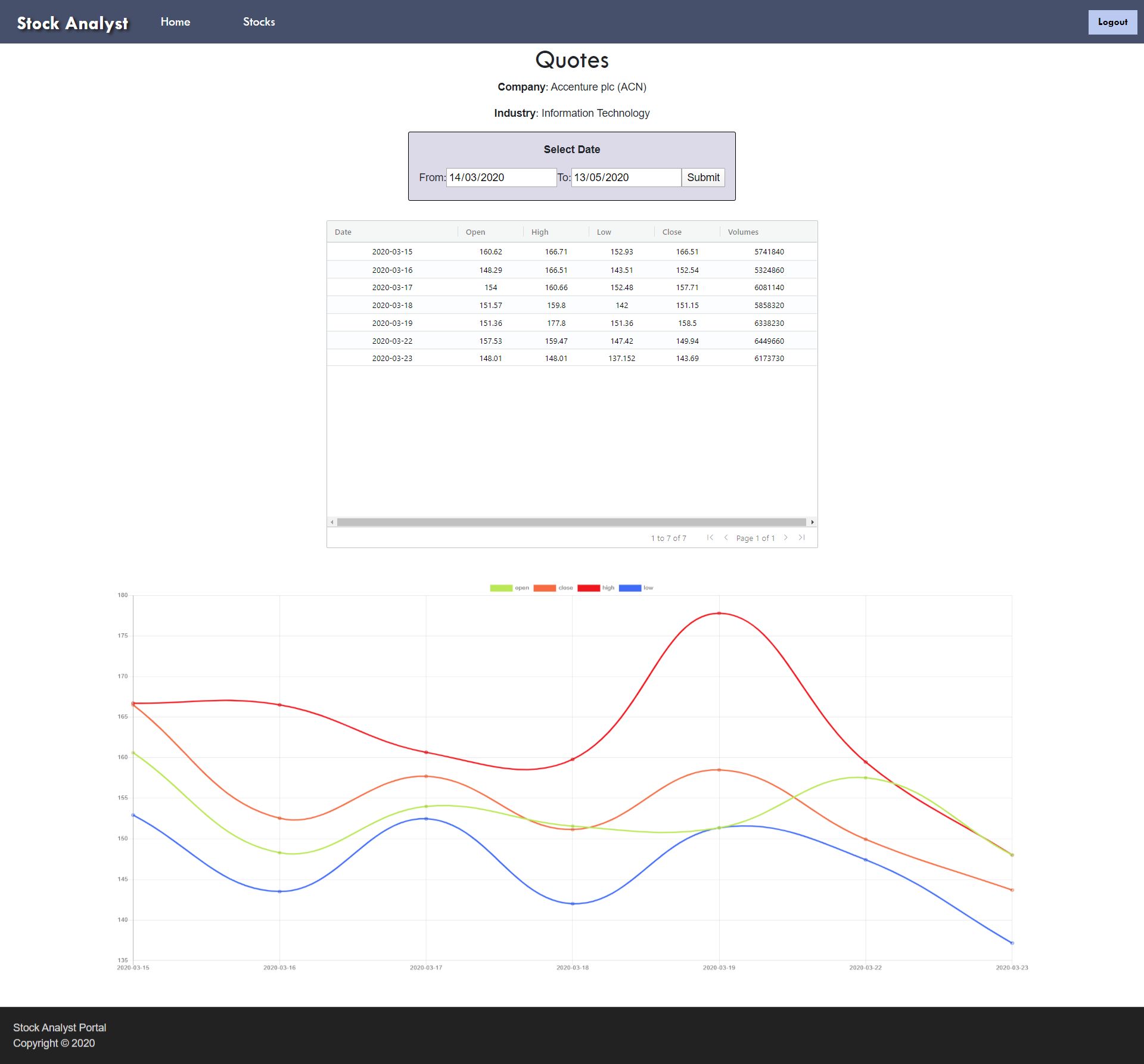


Figure 16 Logged-in Quotes Page

The logged in quotes page can be accessed through the same way the quotes page is accessed from except the user is required to log-in. On this page the user initially sees the current stock, however can find a range of results based on selected dates. They can remove both dates to see the current date again. One issue which was found, was that selecting of dates was quite annoying, as initially the both dates were empty. Furthermore, it would have been beneficial if the current date was the maximum date that could be selected, to pre-emptively prevent errors, rather than display a message once the error has occurred.

Another issue is that if the user was not logged in but had already selected a particular stock to examine but wanted to access the price history, after logging in the user is redirected to the home page, rather than the price page with the current stock. This is inconvenient as the user has to search for that particular stock again.

## Technical Description

### Architecture

**Application source code**:

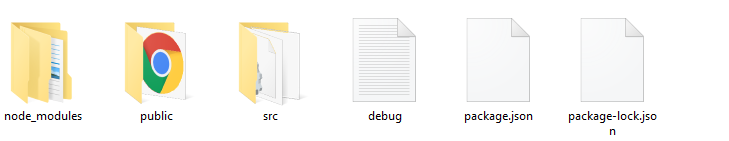


Figure 17 Application Source Code

**Src Directory**: Index renders application to the ReactDOM. Styles, applies styling to everything in App. App holds the primary application, returning a router with all components

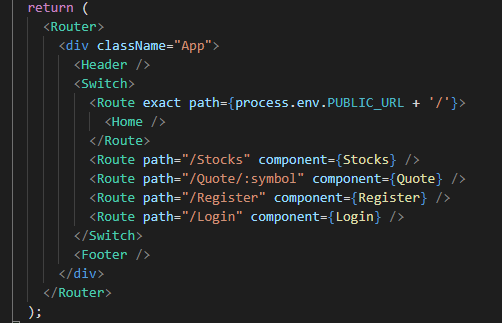


Figure 18 App.js



Figure 19 Src Directory

**Webpage Directory**: Contains the webpages of the application. These are imported into the App.js to be used as components in Routes.

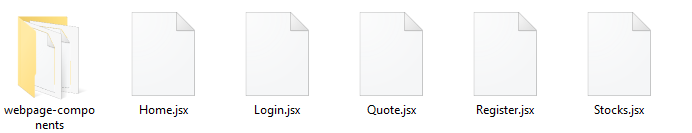


Figure 20 Webpage Directory

**Webpage Components Directory**: Contains components found in the webpages such as the footer, navigation bar, and components to facilitate the retrieval of data from the API, and displaying of that data into tables for the stock and quote pages.

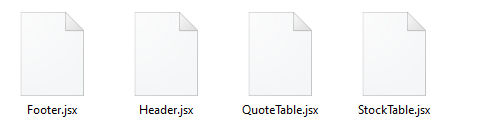


Figure 21 Webpage Components Directory

### Test plan

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Result** | **Source (screenshots are only shown for functionalities not show in the demonstration video)** |
| Display stocks | PASS | see demo.mp4 |
| Filter stocks (search and dropdown) | PASS | see demo.mp4 |
| Deal with error when stock not found | PASS | see demo.mp4 |
| Select stock to view quote of | PASS | see demo.mp4 |
| Display current quote of stock [auth route] | PASS | see demo.mp4 |
| Display current quote of stock & do not display date selection functionality | PASS | see demo.mp4 |
| Display a range of results for a stock | PASS | see demo.mp4 |
| Display a graph depicting said results | PASS | see demo.mp4 |
| Provide error response for when user selects date out of range | PASS | see demo.mp4 |
| Display current results again, after selecting a range of dates | PASS | see demo.mp4 |
| Register user | PASS | see demo.mp4 |
| Handle registering same user or incomplete form | PASS | see demo.mp4 |
| Login user generating token | PASS | see demo.mp4 |
| Logout user | PASS | Figure 22 & 23 |
| Responsive Navigation | PASS | Figure 24 & 25 |
| Result sorting through tables | PASS | see demo.mp4 |
| Deal with expiry of token (logout) | PASS | User is not notified however that they are logged out upon expiry.  Figure 26 & 27 |
| Indicate not connected to server | NOT IMPLEMENTED |  |

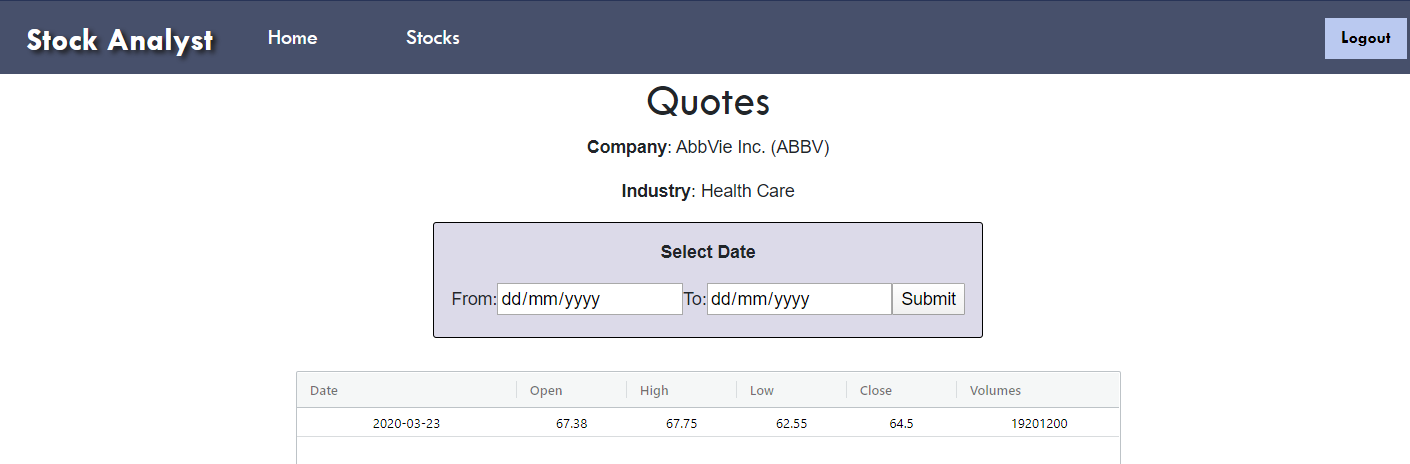


Figure 22 Logged In

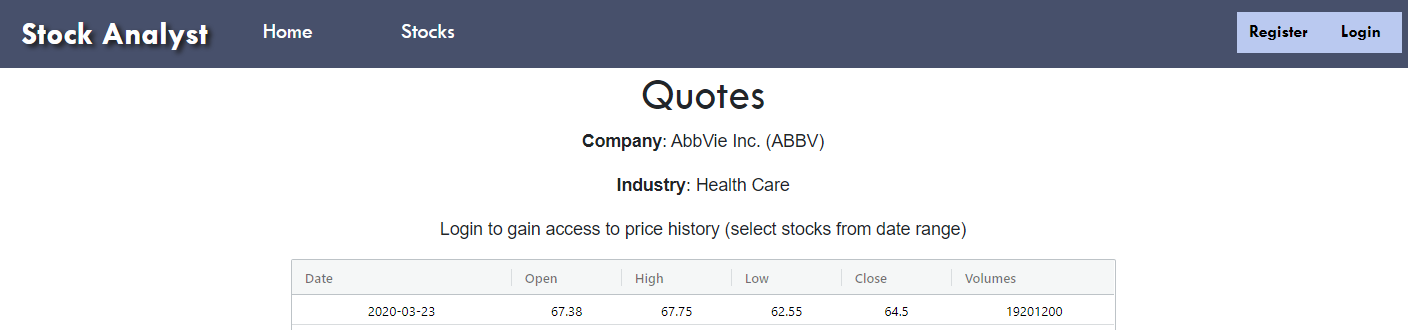


Figure 23 After Logged Out

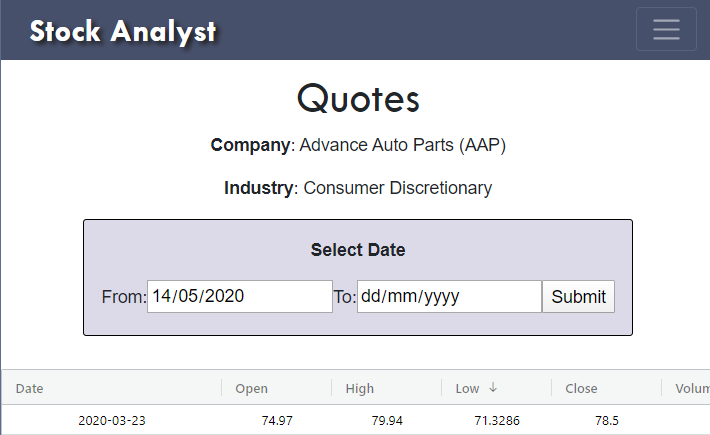


Figure 24 Nav bar closed

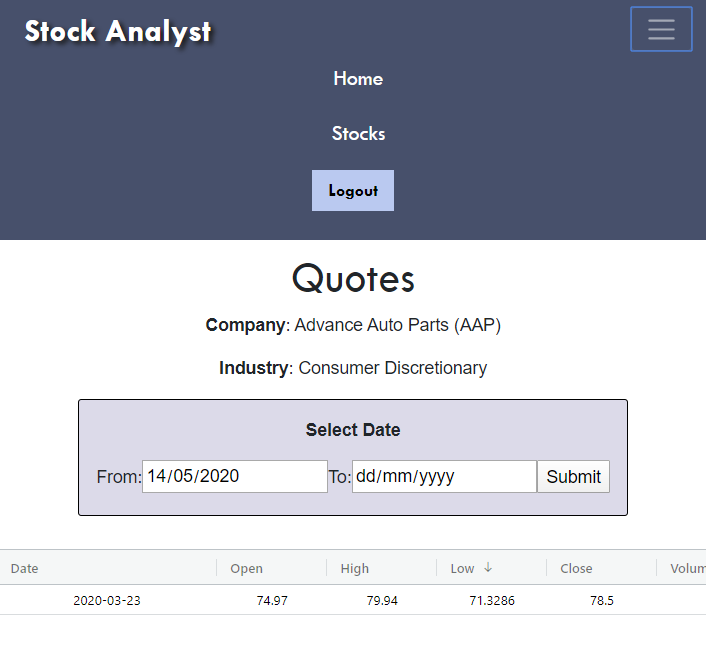


Figure 25 Nav bar opened

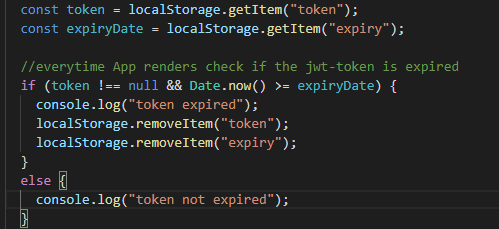


Figure 26 Code dealing with expiry of jwt-token

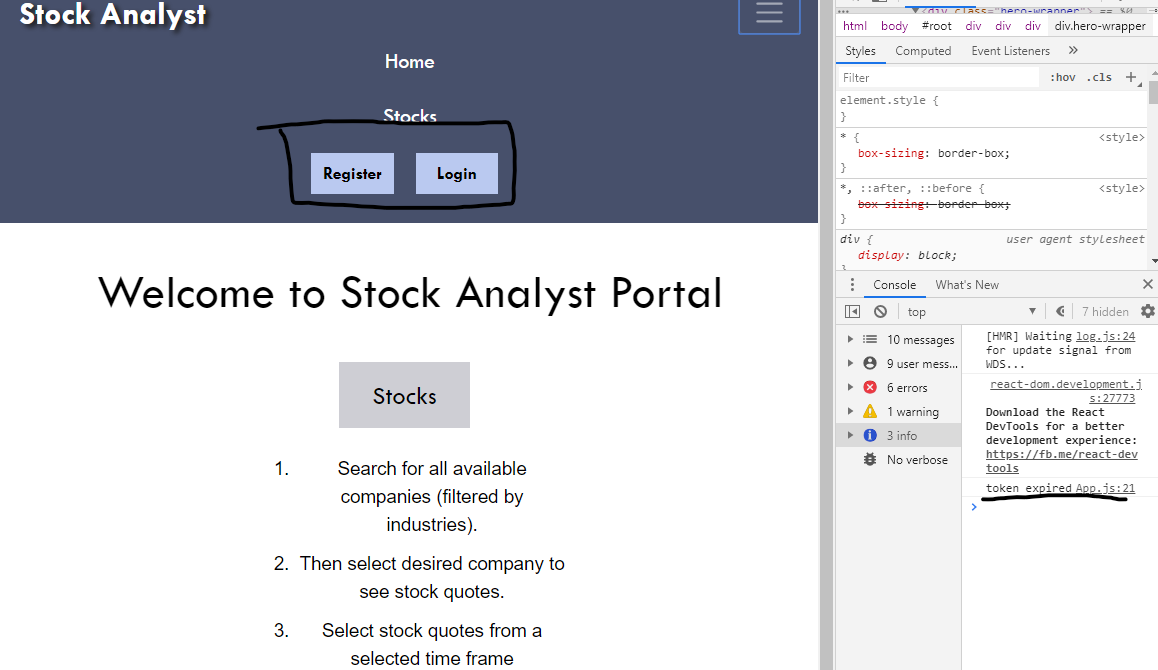


Figure 27 Expiry of jwt-token

Difficulties

After logging in, retrieving the token and decoded token, and redirecting back to the home page, the page showed no indication that it was logged in e.g. log in and register buttons would not disappear. It would only show an indication after being refreshed. To resolve this, a history.go function was added after redirecting to refresh the page once it has been redirected, alternatively could change useState to trigger a re-rendering.

When adding data to the graph it was realized that the dates were in descending order rather than ascending, as this was the way the data was provided and mapped. Initially to resolve this, the array containing the each individual data array e.g. an array of dates was reversed, then set as a useState. However, this did not work as .reverse(), reverses the order of the array in place and therefore it is reversed after call (MDN, 2020). To resolve this issue as seen below in figure 28, the entire dataset returned was reversed, then pushed into arrays.

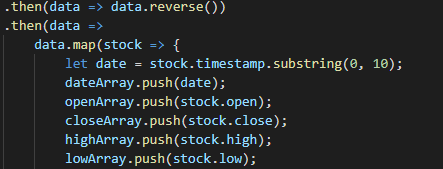


Figure 28 Reversing data

When using the search bar to filter stocks via industry, there was an issue where the search input field would lose focus after typing a single character. This was because it was re-rendering the search input component every time the industry state value was updated (Golubenko, 2019). This was resolved by removing the search bar component and putting its contents directly into the render as seen below in figure 29.

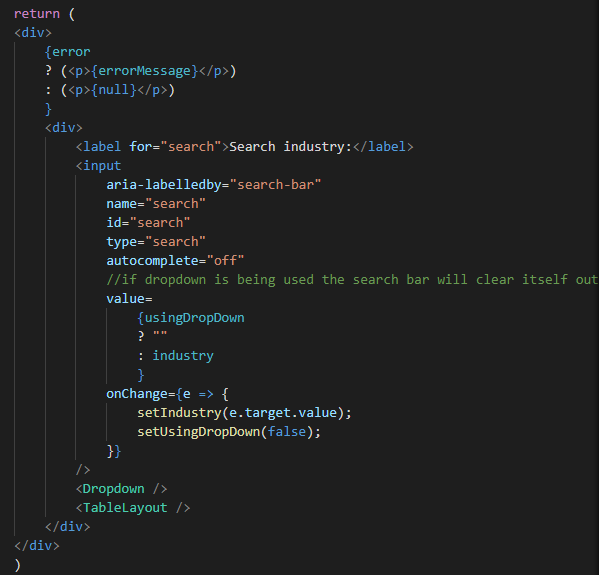


Figure 29 Search bar in return statement

# References

Babich, N. (2017, November 23). *UX Mythbusting: Is The Homepage Really The Most Important Part of Your Website?* Retrieved from Adobe Blog: https://theblog.adobe.com/ux-mythbusting-is-the-homepage-really-the-most-important-part-of-your-website/

Golubenko, A. (2019, November). *React - input loses focus on keypress*. Retrieved from Stack Overflow: https://stackoverflow.com/questions/58778631/react-input-loses-focus-on-keypress

MDN. (2020, January 13). *Array.prototype.reverse()*. Retrieved from MDN web docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/reverse