

2020

# CAB230 Stocks API – Client Side



Scott Jacob>

n9013695

2020/05/13

## Contents

Introduction .....	2
Purpose & description.....	2
Completeness and Limitations.....	3
Use of End Points .....	4
/stocks/symbols .....	4
/stocks/{symbol} .....	4
/stocks/authed/{symbol} .....	5
/user/register .....	5
/user/login .....	5
Modules used.....	6
Ag-grid-react .....	6
Ag-grid-charts.....	6
React-bootstrap .....	6
React-bootstrap-typeahead.....	6
React-Datepicker.....	6
Axios.....	6
Application Design .....	7
Navigation and Layout .....	7
Technical Description.....	8
Architecture .....	8
Test plan.....	9
Difficulties / Exclusions / unresolved & persistent errors.....	10
Extensions (Optional).....	11
User guide .....	12
References .....	14
Appendices .....	15

### Introduction

#### Purpose & description

This application name Stocky has been designed to display and gather stock data from a REST API that has been externally hosted. The application consists of many pages to provide functionality such as: Login, Sign Up (Registration), Search functionality in the case of: Industries – listing all stocks for a specific industry, or all industries in the data. Symbol data – retrieving snapshot data from the API for a specified stock Symbol. Price History – historical data retrieved by searching for a Stock Symbol and a data range, also providing limited trend data charting.

This app has been designed to be sleek while containing core features and functionality in the center of the screen for maximum readability. For the purposes of clean design as well as features available the package react-bootstrap was selected. This package supports the functional component development style as well as large amounts of customization. Other modules such as axios have been used to simplify GET and POST requests to the endpoint allowing for cleaner code and ease of use. Other third-party libraries have also been used such as react-bootstrap-typeahead for autocomplete data - this data is rendered and processed client side from an array of options. React-datepicker has also been used for its simple modular design with large amounts of customization.

Ag-Grid was also used to render and display table data to the user. This library provides client-side filtering and sorting through simple option specification. Ag-grid-charts was also used to directly translate the data to a chart from the same request. This aids in performance as the user makes one request to the server and can have table data displayed as well as chart data from the same dataset.

*\*A higher quality demo video link can be found under references.*

### Completeness and Limitations

This application has implemented the ability to query, retrieve and display data for all provided API endpoints. The user can Login and Register with errors being promptly displayed to the user when a query fails as well as notifications for successful queries i.e. successful registration. The user can also logout with the view changing to reflect this change from any page. The application makes use of React Router to handle page navigation including presenting “404” or not found to any pages manually inputted by the user into the search bar. The user can query the data endpoints with the use of the following:

- Industry: Dropdown menu with instructions at the top of the page.
- Quote: Search bar with autocomplete (client hosted as to not query the server unnecessarily with complementing instructions and error handling)
- Price History: Search bar same as implemented above with Date Range selectors including dynamic changing of dates to prevent wrongful queries. Error handling is also present to handle unwanted and unexpected queries and server responses. Charting in the form of two charts has also been provided for Closing values as well as Volume sale trends.

This application also features an adaptive Navbar which changes based on whether a token is stored on local storage, this code can easily be adapted to check for more secure options in the future. In the case that a token is present but has expired the user will be prompted to login again in the event of a new request for data via the authenticated route. One feature I am particularly proud of implementing is disabling links for users which are not authenticated and providing a tooltip on hover to guide them in how to access the link.

Limitations of the application can be described as some errors may not explain the full reason the request has failed to retrieve data. Multiple methods of error detection have been utilized to catch all errors possible even if displaying an error that is not quite correct. Charting data is also not presenting in an appropriate way, chart data is rendering from right to left instead of the usual and more appealing left to right. The data however is still able to be interpreted and displays tooltips with the correct timestamp data for the value. The design is also not fully optimized to seamlessly change pages.

### Use of End Points

*/stocks/symbols*

Stocks

Selecting Blank and searching will search for all industries.

Select Industry:

Submit

Name	Symbol	Industry
No Rows To Show		

This endpoint displays stock data filtered by industry. (blank for all industries)

*/stocks/{symbol}*

Quotes

Search the Stock Symbol to see the latest snapshot. Must be 1-5 Capital Letters

Please select the stock after typing to ensure the stock is retrieved

Search a symbol e.g. AAL

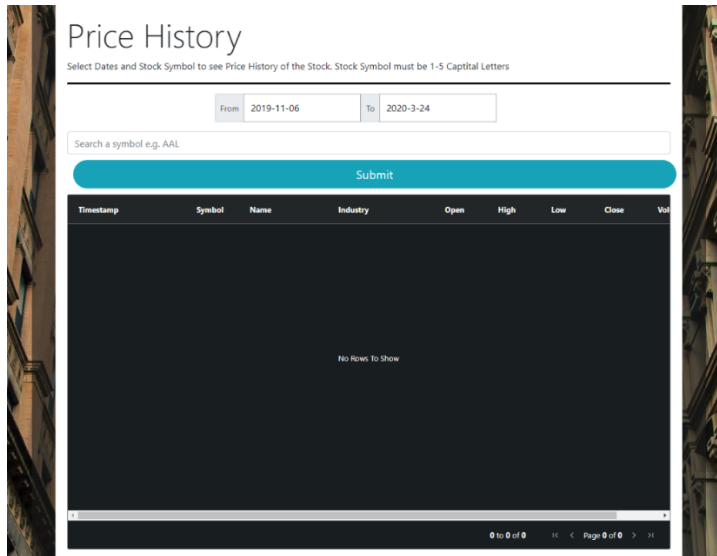
Submit

Timestamp	Symbol	Name	Industry	Open	High	Low	Close
No Rows To Show							

0 to 0 of 0 | Page 0 of 0

This endpoint displays a single snapshot of a stock and returns it. Searched by symbol.

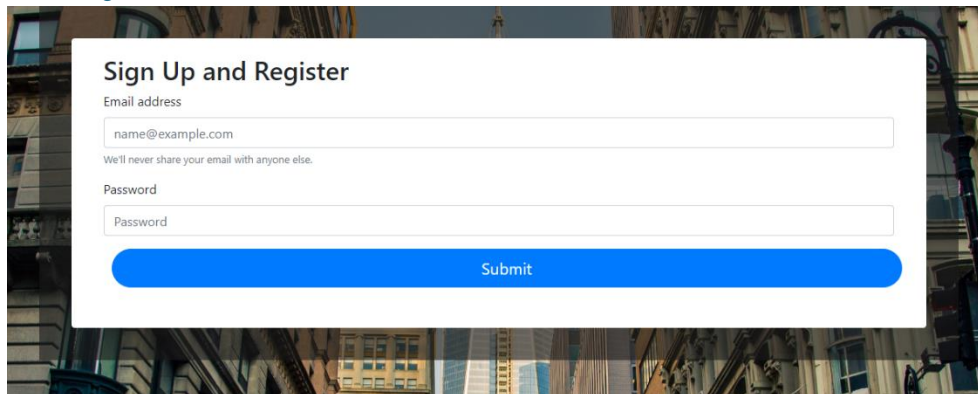
*/stocks/authed/{symbol}*



The screenshot shows a web application titled "Price History". Below the title is a subtitle: "Select Dates and Stock Symbol to see Price History of the Stock. Stock Symbol must be 1-5 Capital Letters". There are two date input fields: "From" with the value "2019-11-06" and "To" with the value "2020-3-24". Below these is a search bar with the placeholder text "Search a symbol e.g. AAL" and a blue "Submit" button. Underneath the button is a table with the following headers: "Timestamp", "Symbol", "Name", "Industry", "Open", "High", "Low", "Close", and "Vol". The table body is currently empty, displaying "No Rows To Show". At the bottom of the page, there is a pagination bar showing "0 to 0 of 0" and "Page 0 of 0".

This endpoint requires authentication and displays all stock entries between the dates selected, this endpoint also charts specific data such as close price and volumes sold.

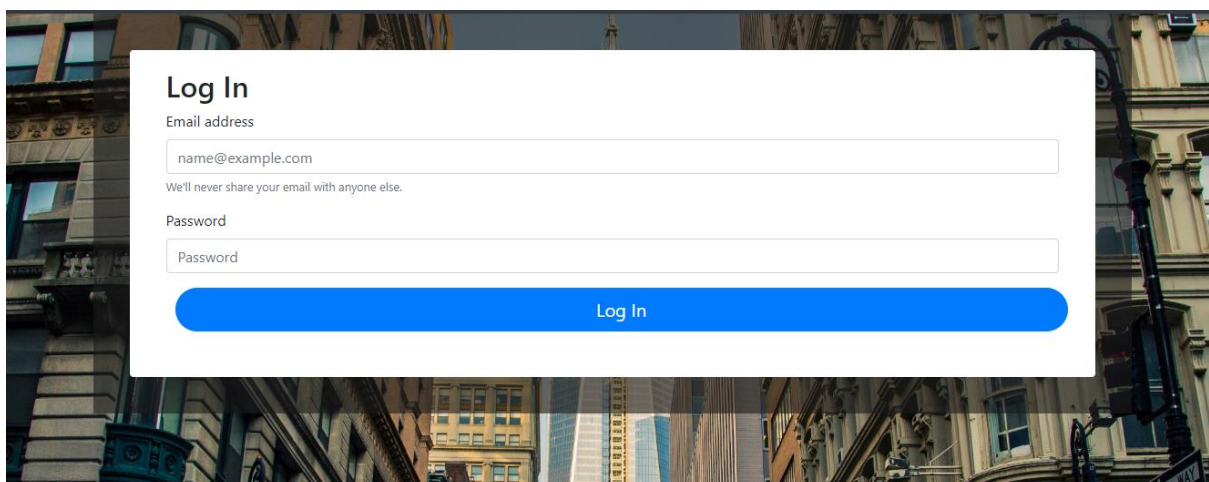
*/user/register*



The screenshot shows a web application titled "Sign Up and Register". It features two input fields: "Email address" with the value "name@example.com" and "Password" with the value "Password". Below the email field is a small text note: "We'll never share your email with anyone else.". A large blue "Submit" button is positioned at the bottom of the form.

Register a user to the site.

*/user/login*



The screenshot shows a web application titled "Log In". It features two input fields: "Email address" with the value "name@example.com" and "Password" with the value "Password". Below the email field is a small text note: "We'll never share your email with anyone else.". A large blue "Log In" button is positioned at the bottom of the form.

Login to the site.

### Modules used

#### *Ag-grid-react*

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

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

#### *Ag-grid-charts*

Module to provide charting functionality for existing Ag-grid table data.

<https://www.ag-grid.com/react-charts/>

#### *React-bootstrap*

Module to provide increased compatibility between standard Bootstrap styling and the React Framework.

<https://react-bootstrap.github.io/>

#### *React-bootstrap-typeahead*

Module to provide built in autocomplete to react forms.

<https://www.npmjs.com/package/react-bootstrap-typeahead>

<https://ericgio.github.io/react-bootstrap-typeahead/>

#### *React-Datepicker*

Module used to provide a modular and re-usable datepicker component for React.

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

<https://reactdatepicker.com/>

#### *Axios*

Module to provide simplified GET and POST requests for React.

<https://github.com/axios/axios>



### Application Design

#### Navigation and Layout

The design inspiration was that each page of the application should be accessible from each other page, the user should not have to navigate multiple menus to reach the page that they want.

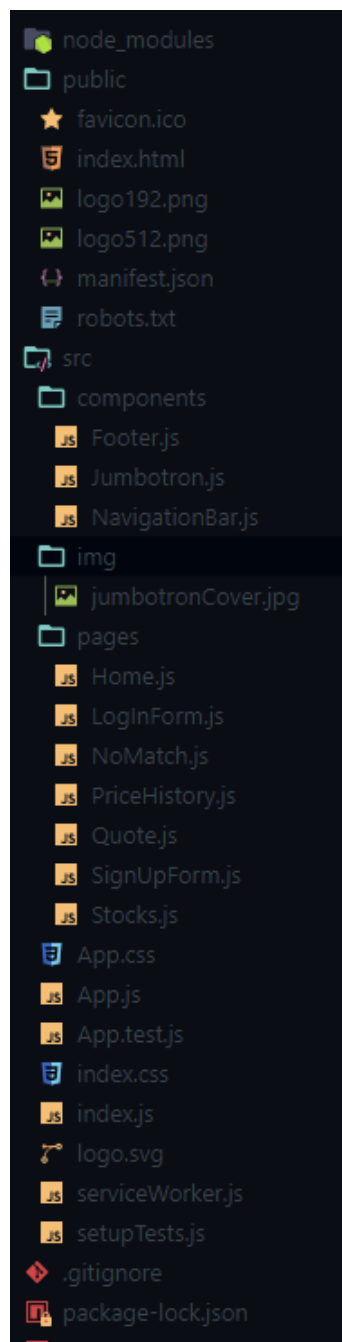
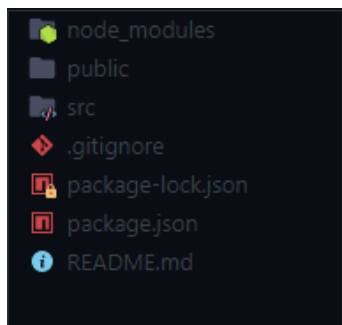
The major design factor as stated at the beginning of this report was minimizing the effort of the user in navigating and using the application. Apart from the navbar all page data is centered in the screen with a container-based design being used to pull the users attention towards it. The navbar design was chosen to be neutral in color with only a few links as well as two buttons for login and registration, replaced with a logout button once the user is authenticated. The main interface consists of a full-page background image covered by a transparent outer box and finally a white opaque container inside with the page content in contrasting colors. The landing page contains a full-page image on initial load with a welcome banner and some cards underneath all containing descriptions of features and navigational aids (buttons). The layout has been kept consistent between all functional pages to keep navigation simple.

Different styling frameworks were considered for this assignment, the original thought was to use reactstrap however some of the components did not offer enough customization off the bat without more effort being used to change them. The decision was made early on in development to switch to react-bootstrap which was found to be a better suite and easier to develop with. Each form for each page was setup to be roughly the same design with different inputs based on what the user was to achieve on that page. The idea was that the user selects the page based on their needs and uses the form to query the API and see the results on that same page. In the event of an error an alert is displayed at the top of the page to assist the user.



## Technical Description

### Architecture



## Code Structure

The application architecture is set up by the standard React-App install from nodejs. This structure allows a simple and manageable code structure.

From the standard install source code is organised into different categories: components, pages and images.

Components are made up from drop and play features of the application such as the navigation bar, the footer and the jumbotron.

All images used are stored in img with an appropriate name to recognise the image being used.

Pages consists of the functional pages the application will display. All core logic is contained in these pages and is broken up with clear commenting and descriptions of the sections.

Finally, on the outer most level is the main app.js and index.js files.

This format was chosen to manage the files in a way that is not only recognisable for the function of the code but also to aid in the development process.

## Test plan

Task	Expected Outcome	Result	Screenshot/s (Appendix A)
Search for Industries (all)	Table data is displayed (all industries)	PASS	01
Search specific industry (health care)	Table data is displayed (health care industry)	PASS	02
Search Symbol (AAL)	Table data is displayed (symbol AAL)	PASS	03
Search Symbol (Fail)	Error Message (symbol does not exist)	PASS	04
Search Price History (min date to max date; symbol BRK.B)	Table data displayed (BRK.B) from within selected date ranges, charting data also displayed	PASS	05-07
Search Price History (valid symbol, invalid dates)	Error Message (invalid date range)	PASS	08
Search Price History (invalid symbol, valid dates)	Error Message (symbol does not exist)	PASS	09
Register User	User is successfully registered	PASS	10
Register User (Fail)	Error message is shown (already registered)	PASS	11
Register User (Fail)	Error Message (form incomplete)	PASS	12
Login	Page is redirected, navbar changes (login and logout buttons replaced by logout button, price history tab accessible)	PASS	13
Login (Fail)	Error Message (password incorrect)	PASS	14
Logout	Navbar changes (price history tab is disabled, login and register buttons shown)	PASS	15

### Difficulties / Exclusions / unresolved & persistent errors /

Difficulties for building this application were mostly focused around the querying of the API and the displaying of the returned data. Some of the API endpoints required different methods of hitting them in order to successfully query them. This however was not the most difficult aspect faced. Charting for the price history page proved to be the most difficult with multiple hours being spent on trying different packages and approaches to charting the data. For example, the beginning thought was to use some integrated charting through Ag-grid-enterprise charting functionality however this approach proved troublesome and requiring large amounts of configuration and restructuring of the application. The next option was to use Chart.js to chart the data. This package also proved difficult to coerce into displaying any data from an external API and would have required a separate request to the server to be conducted simultaneously. Ag-grid-charts standalone proved to be the simplest option with no restructuring or reconfiguration required. This package also was able to use the original retrieved dataset and filter it through specific x and y values.

Other hindrances to the building of the application were also due to the functionality that Ag-grid provides with table data. For single returns such as the {symbol} endpoint where data is returned in a single JSON object and not an array, Ag-grid would not render the data unless pushed to an array. This was overcome later in the development process after finding a method to push data from a React Hook State to an array using the Concat() method. This data was buried inside a stack overflow answer requiring precise searching. Another method was since found later in the development process by wrapping the returned data in the Ag-grid component with an array however this method has proven to be more reliable.

Some persistent errors for the project are most notably the error values being returned by the API. Some errors come in an error response and some as a simple JSON object with a message. As a result, the way in which error handling is conducted underneath is not the most optimal. Multiple methods have been used to ensure that all errors are captured regardless of the method that the API used to return them. This includes logic for assessing the normal return value of an error as well as whether the error flag has been set in a returning response consequently throwing that error to the error handling logic.

Another outstanding bug is the background image that is used for the application. The image can and will distort based on not only device viewport but also the component rendered on the screen. The background will also appear zoomed for no reason. CSS is used to fit the image to the screen for various pages, most pages sharing the same CSS properties and formatting, yet the issue persists. Although not a major bug in functionality it is quite noticeable on use of the application. This issue is mostly due to the different ways which the components must be displayed, as some of the components such as the login and register forms are not long enough to take up full-page height on a standard 1920x1080p display. Other pages are longer thus stretching the image.

Other bugs which have been found and not fixed currently are due to the use of the autocomplete form. The form requires that the option is selected by the user as just typing and selecting submit does not actually change the state. A minor inconvenience to the use of the feature however I believe that incorporating an autocomplete feature for the stock symbols is necessary in conveying the type of input that the form needs.

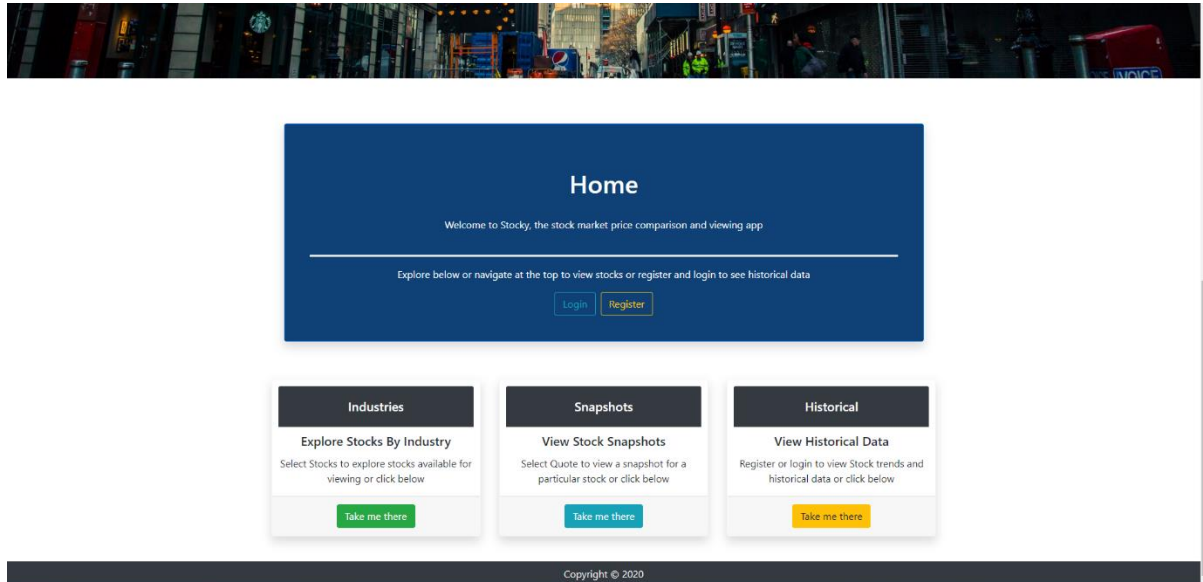
### Extensions (Optional)

An extension to this application that I would like to do would be a redesign of the core functionality. That is moving from multiple pages to a seamless single page design. Not only would this make the design of the application more fluid, it would also fix some of the issues surrounding the conditional rendering of components for Login and Logout. Components could be changed by state instead of having to utilize page redirects and refreshing. This design could also accommodate more complex API's and charting for data.

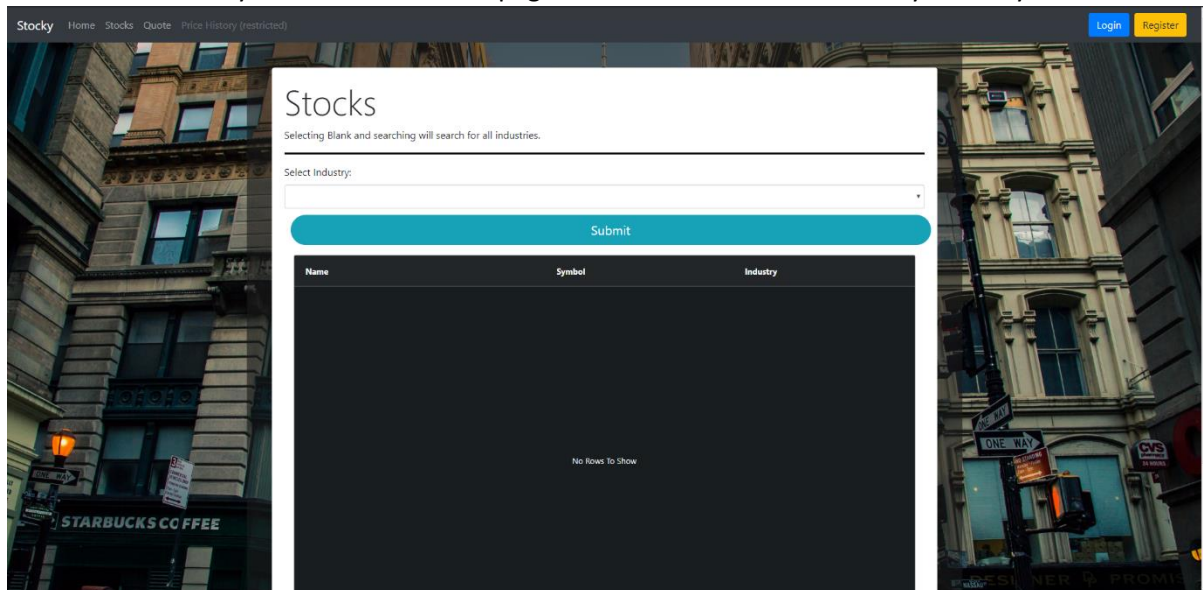
### User guide

#### How to use Stocky:

Greeted on the homepage you will find an abundance of links and helpful descriptions to navigate the site. On the bottom of the homepage information is displayed.



The navigation bar is also displayed on every page. Clicking on the Stocks link in the nav bar or from the industries card you will be taken to a page to search for stocks filtered by industry.



On this page you can select an industry from the dropdown menu or leave blank and hit submit. The stocks will then be retrieved and shown. Clicking on the Symbol and Name tabs will sort the list alphabetically.

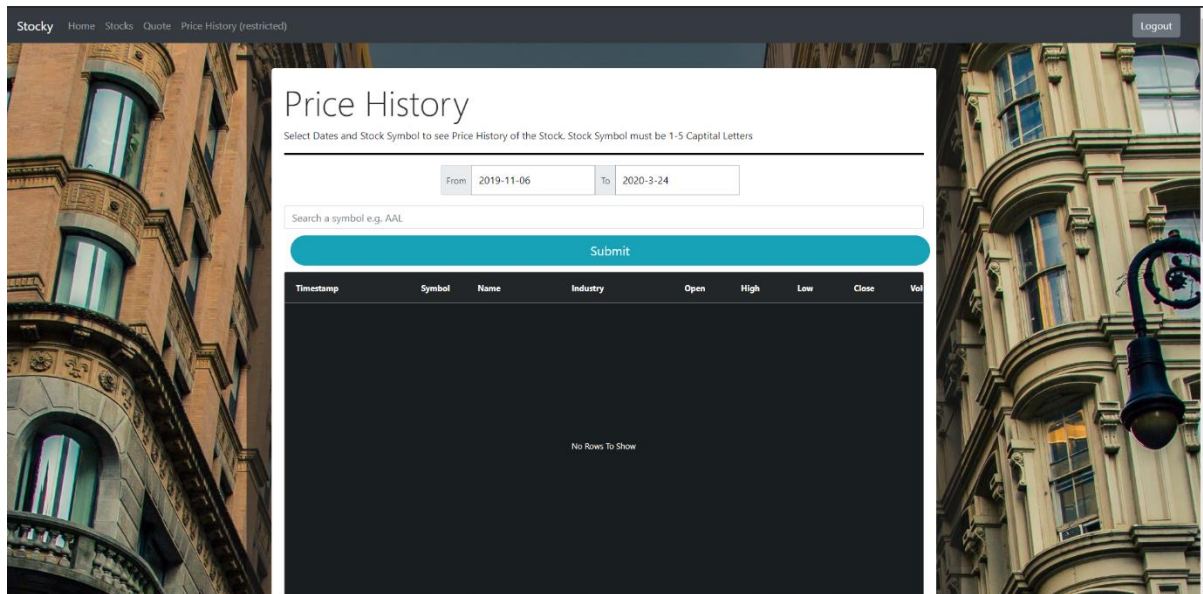
After finding a stock to search or if you already know the stock symbol you can navigate to the Quote tab or through the homepage again to view a single snapshot of that stock.

The Quote page features autocomplete for all stocks available for search at the time of developing this application. Clicking on a stock after searching or using the menu, then hitting submit will search for the stock [see Appendix A03].

To view the price history of a stock you will first need to login; or register if you do not have an account. You can do this by clicking on Login or Register on the navigation bar or through the homepage. Errors will be displayed alerting the user to failed attempts and logging in and registering.

After successful login the page will be redirected to the homepage. You will notice that the navigation bar has changed with the Price History tab now accessible and the login, register buttons replaced with a button to logout [see Appendix A13, A14].

Upon accessing the Price History tab you will be greeted with this screen.



The screenshot shows a web application titled 'Price History' with a navigation bar at the top containing 'Stocky', 'Home', 'Stocks', 'Quote', and 'Price History (restricted)'. A 'Logout' button is in the top right corner. The main content area has a title 'Price History' and a subtitle 'Select Dates and Stock Symbol to see Price History of the Stock. Stock Symbol must be 1-5 Capital Letters'. Below this is a form with two date pickers: 'From' (2019-11-06) and 'To' (2020-3-24). A search bar with the placeholder 'Search a symbol e.g. AAL' is present, followed by a blue 'Submit' button. Below the form is a table with the following headers: 'Timestamp', 'Symbol', 'Name', 'Industry', 'Open', 'High', 'Low', 'Close', and 'Vol'. The table body is empty, displaying 'No Rows To Show'.

After selecting a symbol as before and altering the dates if you so wish you will see the table fill and two charts below populate if the request was successful. Error messages will be shown under the navigation bar if unsuccessful.

After the user is finished, they may logout of the application.

## References

Higher quality demo video located at:

<https://drive.google.com/open?id=1AtmPuH1vUSOLEXGfWf0M5nVIRz3UdaOp>



## Appendices

A:

01

The screenshot shows a web application titled "Stocks". Below the title is a message: "Selecting Blank and searching will search for all industries." There is a dropdown menu labeled "Select Industry:" which is currently empty. Below the dropdown is a teal "Submit" button. Below the button is a table with three columns: "Name", "Symbol", and "Industry". The table lists various companies and their respective symbols and industries.

Name	Symbol	Industry
Agilent Technologies Inc	A	Health Care
American Airlines Group	AAL	Industrials
Advance Auto Parts	AAP	Consumer Discretionary
Apple Inc.	AAPL	Information Technology
AbbVie Inc.	ABBV	Health Care
AmerisourceBergen Corp	ABC	Health Care
Abbott Laboratories	ABT	Health Care
Accenture plc	ACN	Information Technology
Adobe Systems Inc	ADBE	Information Technology
Analog Devices Inc.	ADI	Information Technology
Archer-Daniels-Midland Co	ADM	Consumer Staples

02

The screenshot shows the same web application as in the previous image, but the "Select Industry:" dropdown menu is now set to "Health Care". The "Submit" button is still visible. The table below shows a filtered list of companies, all of which are in the "Health Care" industry.

Name	Symbol	Industry
Agilent Technologies Inc	A	Health Care
AbbVie Inc.	ABBV	Health Care
AmerisourceBergen Corp	ABC	Health Care
Abbott Laboratories	ABT	Health Care
Aetna Inc	AET	Health Care
Allergan Plc	AGN	Health Care
Align Technology	ALGN	Health Care
Alexion Pharmaceuticals	ALXN	Health Care
Amgen Inc	AMGN	Health Care
Anthem Inc.	ANTM	Health Care
Baxter International Inc.	BAX	Health Care
Becton Dickinson	BDX	Health Care

03

## Quotes

Search the Stock Symbol to see the latest snapshot. Must be 1-5 Capital Letters

Please select the stock after typing to ensure the stock is retrieved

Submit

Timestamp	Symbol	Name	Industry	Open	High	Low	Close
2020-03-23T14:00:00.000Z	AAL	American Airline...	Industrials	10.9	11.36	10.01	10.25

04

Stocky Home Stocks Quote Price History (restricted) Logout

No entry for symbol in stocks database

## Quotes

Search the Stock Symbol to see the latest snapshot. Must be 1-5 Capital Letters

Please select the stock after typing to ensure the stock is retrieved

Submit

Timestamp	Symbol	Name	Industry	Open	High	Low	Close
-----------	--------	------	----------	------	------	-----	-------

05

## Price History

Select Dates and Stock Symbol to see Price History of the Stock. Stock Symbol must be 1-5 Capital Letters

From 2019-11-06 To 2020-3-24

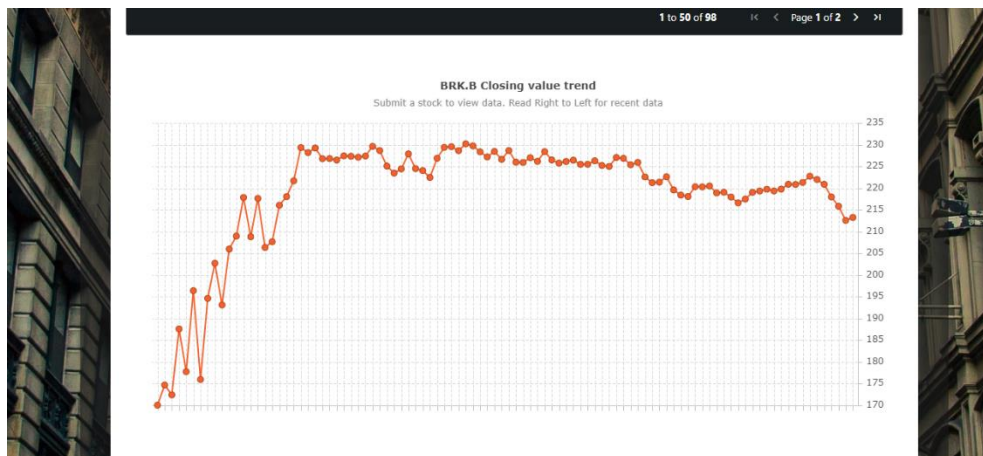
BRK.B

Submit

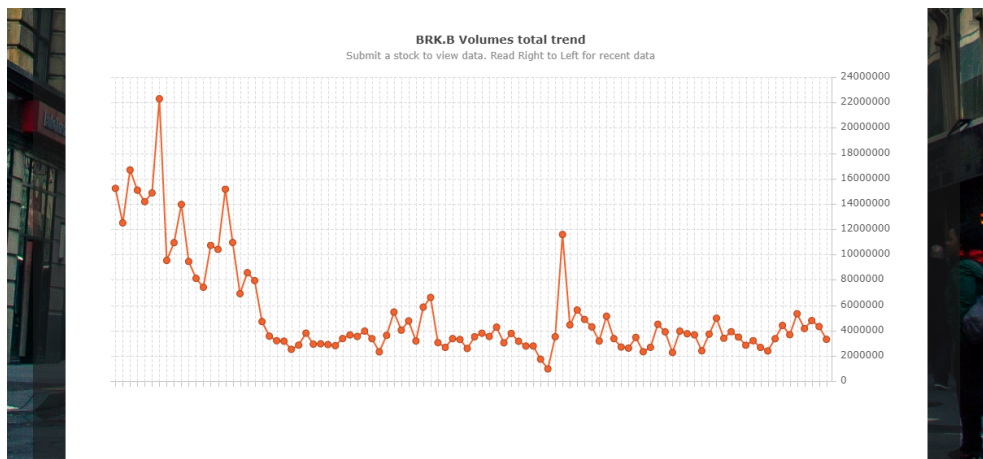
Timestamp	Symbol	Name	Industry	Open	High	Low	Close	Vol
2020-03-22T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	176.3	177.18	167.294	170.06	
2020-03-19T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	170.21	176.18	166	174.68	
2020-03-18T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	177.29	181.39	167	172.44	
2020-03-17T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	181.88	187.81	177	187.6	
2020-03-16T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	174	188.38	173.62	177.77	
2020-03-15T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	187.45	196.5	178.41	196.4	
2020-03-12T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	183.6	187.67	175	175.97	
2020-03-11T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	197.32	198.13	192.87	194.64	
2020-03-10T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	201.35	202.86	195.052	202.73	
2020-03-09T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	195	198.479	192.26	193.13	
2020-03-08T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	203.48	207.41	202.35	205.98	
2020-03-05T14:00:00.000Z	BRK.B	Berkshire Hatha...	Financials	213.55	213.75	207.01	208.06	

1 to 50 of 98 Page 1 of 2

06



07



08

Stocky Home Stocks Quote Price History (restricted) Logout

No entries available for query symbol for supplied date range

**Price History**  
Select Dates and Stock Symbol to see Price History of the Stock. Stock Symbol must be 1-5 Capital Letters

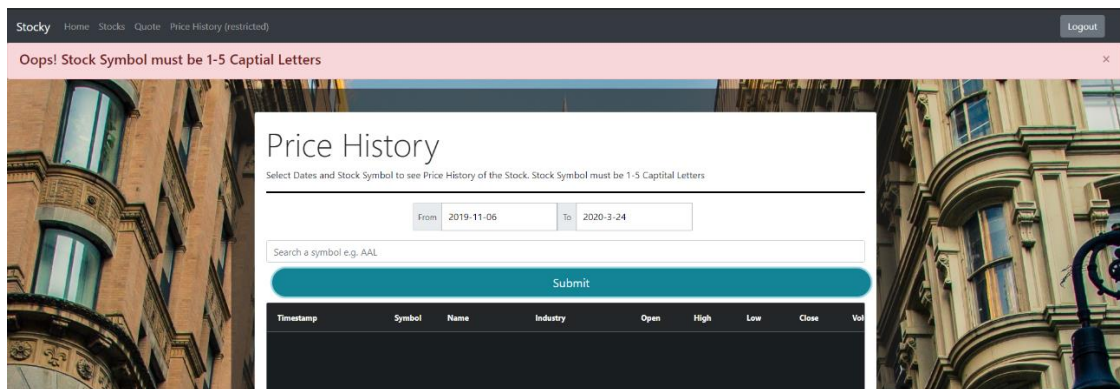
From 2019-11-06 To 2019-11-06

BRK.B

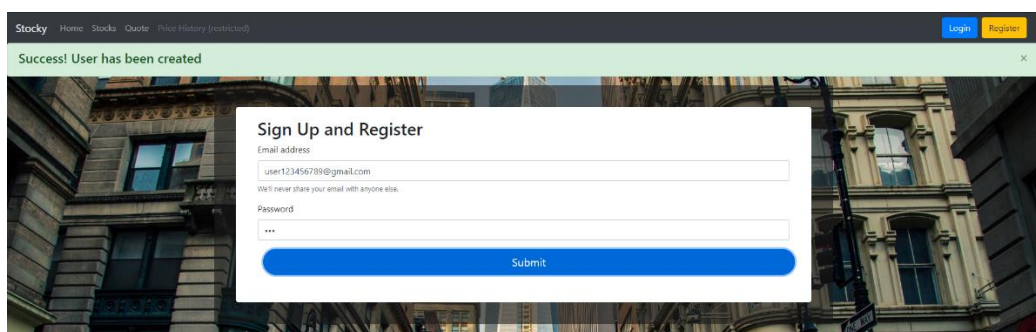
Submit

Timestamp	Symbol	Name	Industry	Open	High	Low	Close	Vol
-----------	--------	------	----------	------	------	-----	-------	-----

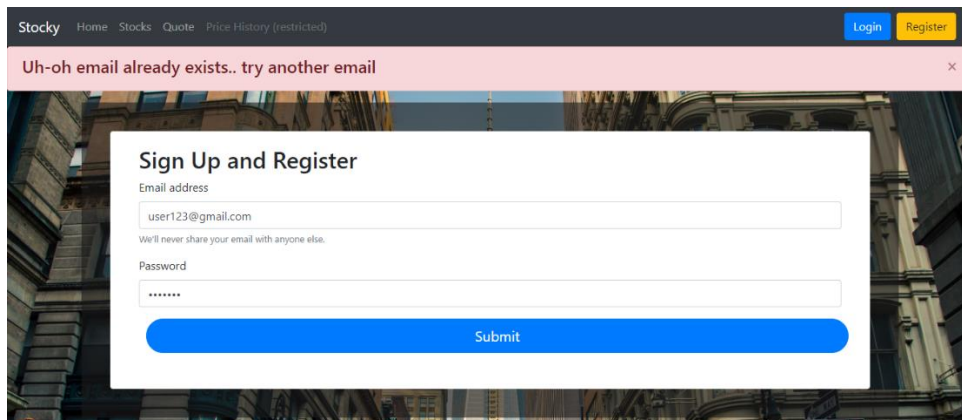
09



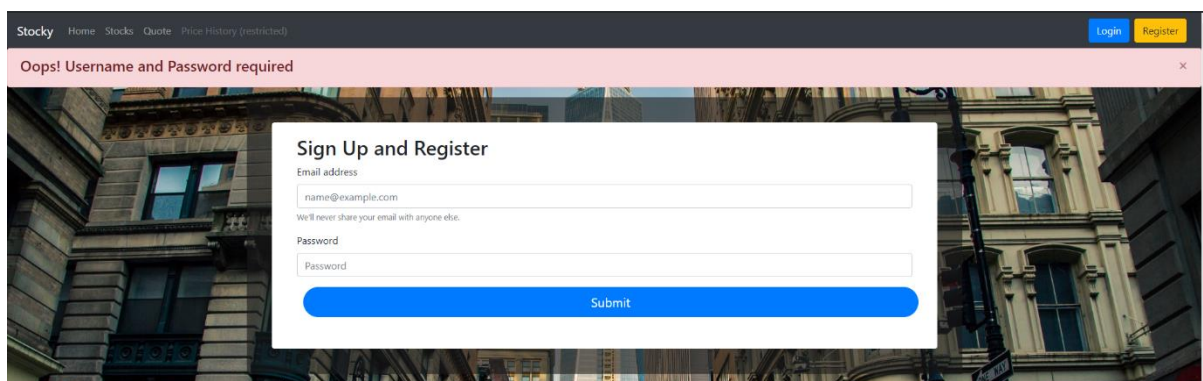
10



11



12





13



14



15

