

2020

# CAB230 Stocks API – Client Side



CAB230

Stocks API – Client Side Application

<John Nguyen /s>

<N10132767 /s>

5/13/2020

## Contents

Introduction .....	2
Purpose & description .....	2
.....	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 .....	6
Modules used .....	6
Ag-grid-react .....	6
React-bootstrap .....	6
Reactstrap .....	6
React-chartjs-2 .....	6
React-hook-form .....	7
Axios.....	7
Application Design .....	7
Navigation and Layout .....	7
Technical Description.....	8
Architecture .....	8
Test plan .....	9
Difficulties / Exclusions / unresolved & persistent errors.....	17
User guide .....	17
Appendices .....	22
JWT error catch problem using fetch: (resolved using Axios).....	22

## Introduction

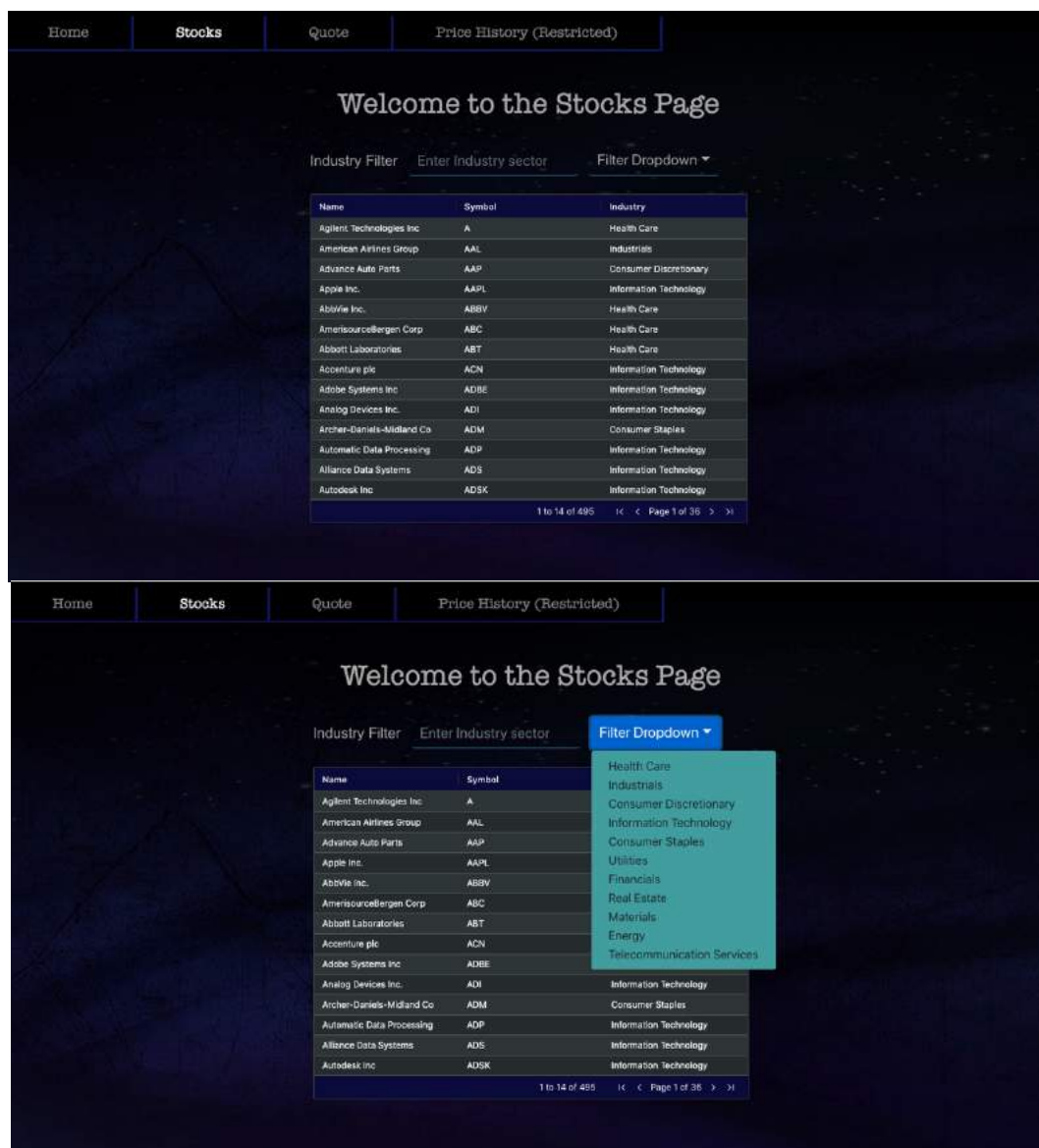
### Purpose & description

This assignment is a React-based web application which allows users to view and analyse stock market statistics drawn from a database that QUT have created for this purpose and exposed to us to via a REST API.

The aims of this assignment are to:

- build a sophisticated client web application using modern approaches
- Provide experience in querying REST APIs and presenting the results for your users
- Provide experience with modern web technologies including node.js and React

There is one specific functionality that would be considered a little unique. Regarding the querying the list of stocks and the stocks page, there includes a text search bar and a dropdown bar to filter the stock list by industry. These two functionalities are superfluous as the a-grid already has filter options for the industry sector. Although, the dropdown and search bar were included as it provides extra options to sort the stock list by industry sector.



## Completeness and Limitations

- **[Grade of 4 level]:** A simple React app with limited styling which implements the unauthenticated Query endpoints and presents the data cleanly using table components. User endpoints and the authenticated queries may not have been implemented successfully and the client side processing in the table components is very limited. A react-strap table component or similar would suffice here.
- **[Grade of 5 level]:** At this level we would expect successful implementation of the User endpoints and the authenticated Query route. Table component usage and client side processing would be expected to use the standard functionality provided by a component such as ag-Grid-react, and there should not be excessive querying of the server.
- **[Grade of 6 and 7 level]:** Here the expectation is that you have exceeded the grade of 5 level in that all of the basics are there and working smoothly. Navigation is handled using React Router, React forms are used for the data entry and there is evidence of a really good match between your components and the services that they are using. We would expect some use of charting or other information graphics to show how the stock prices are varying and advanced use of the client side processing. The split between the grade of 6 and grade of 7 will involve a tradeoff between the features and the quality of the execution, and we will happily give you an opinion on your proposed application. For charting we recommend the use of chartJS (<https://www.chartjs.org/>), especially via the widely used React wrapper you can find here: <https://www.npmjs.com/package/react-chartjs-2>. d3 (<https://d3js.org/>) is a popular choice, but it is an advanced library and you shouldn't attempt it unless you have prior experience.

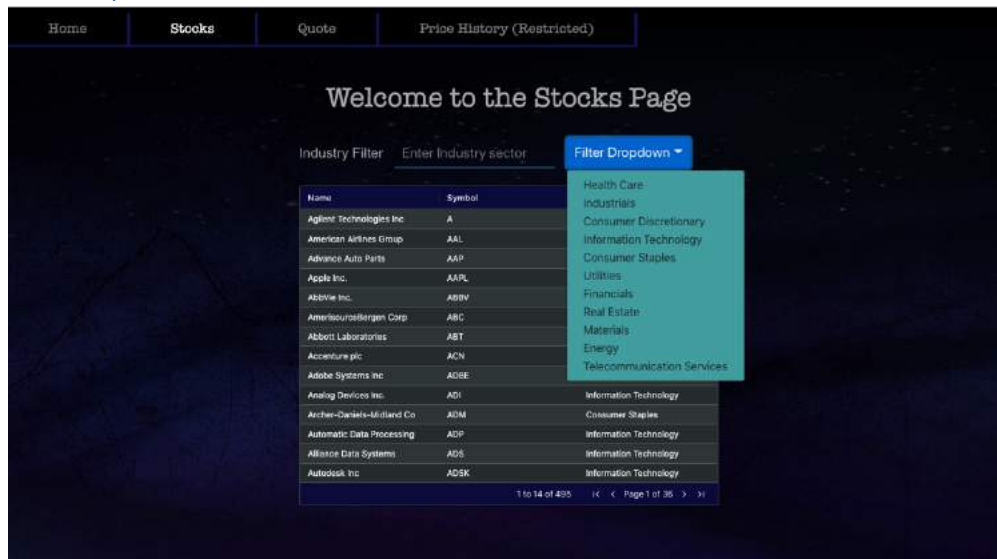
### **Claim: standards of this react application has met the criteria of Grade 6 and 7 level**

The react application does have all the basic functionalities working smoothly. Navigation is handled using React Router, React forms have been used for filter search querying from the endpoints and user authentication. Charting has been included for the display of the closing price history over the last 100 days with a date query.

There are no limitations with the application.

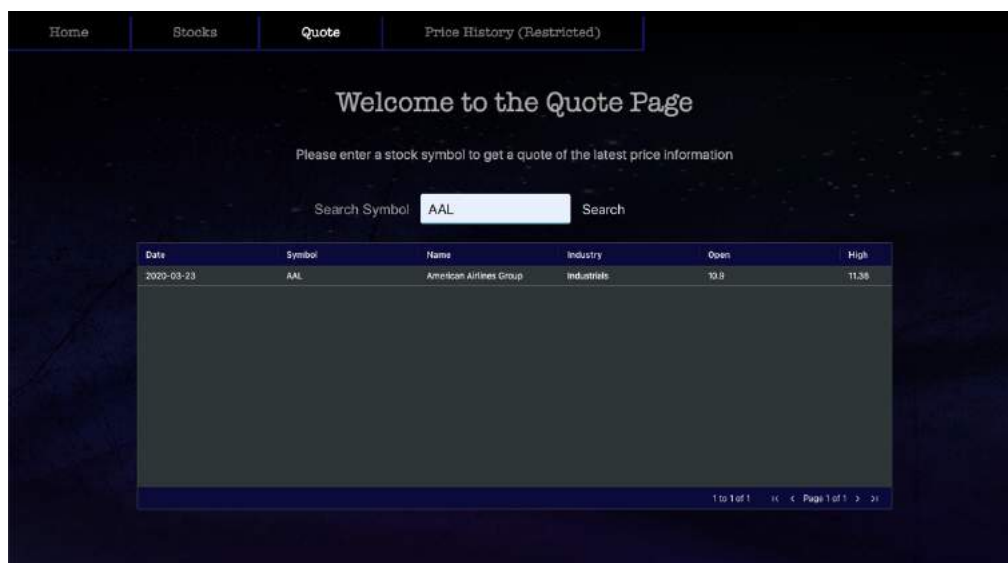
## Use of End Points

/stocks/symbols



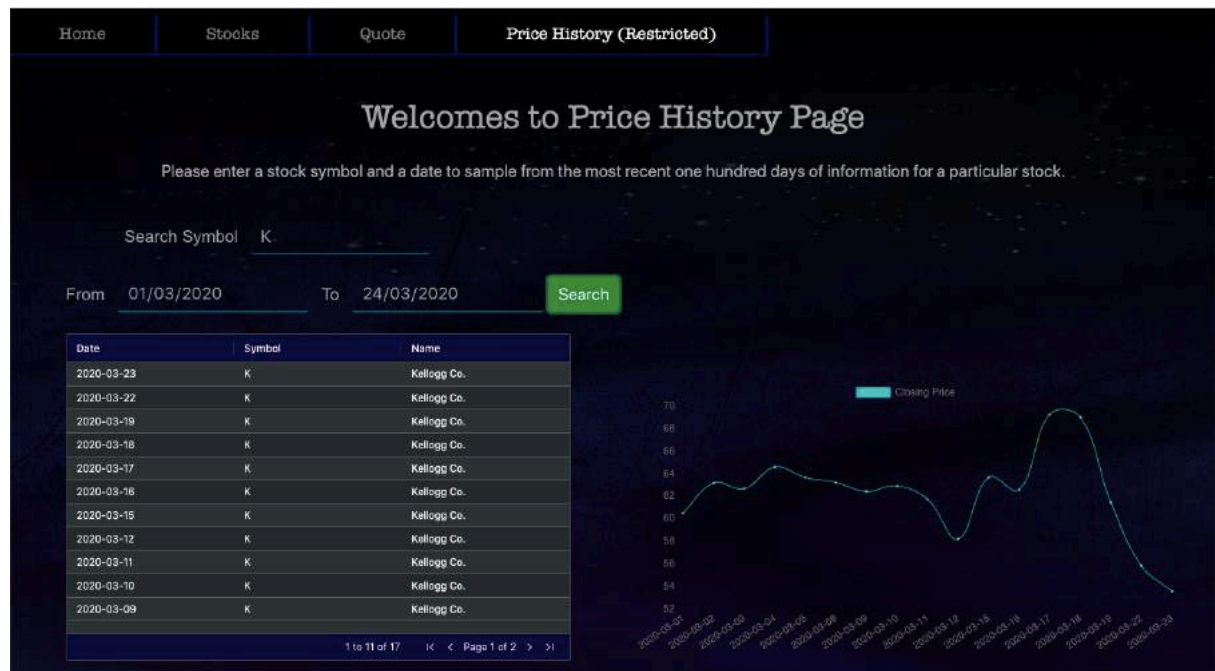
This is the Stocks page covers this section endpoint. Search bar form and a dropdown menu is provided so that the user can filter the list by industry sector.

/stocks/{symbol}



This is the Quote page covers this section endpoint. Search bar form is provided so that the user can get the latest entry for a particular stock, searched by symbol (1-5 characters)

</stocks/authed/{symbol}>




This is the Price-History page covers this section endpoint. Search bar forms including type text and date are provided so that users can get all the entries of the stock searched by symbol, optionally filtered by date. This page also requires user authentication to view contents of this page.

</user/register>

Home Stocks Quote Price History (Restricted)

## User Registration



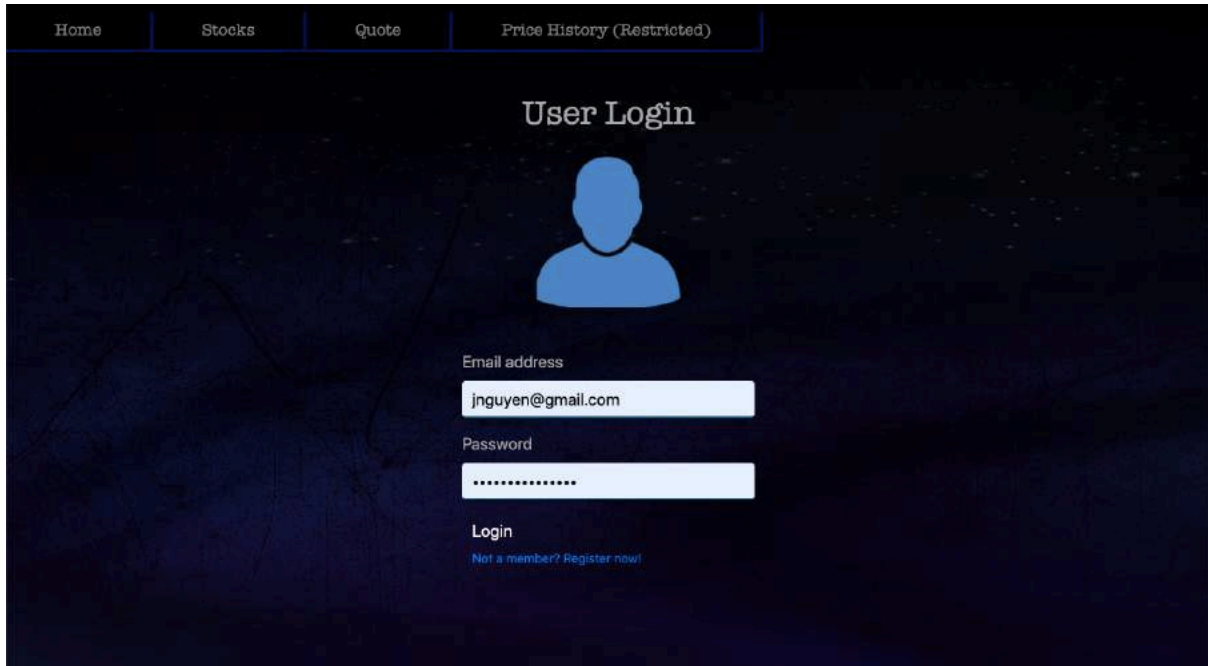
Email address

Password

[Already a member? Login now!](#)

This is the Register page covers this section endpoint. Email are provided so that users can register their account by entering an email and password.

*/user/login*



Home Stocks Quote Price History (Restricted)

## User Login

Email address

jnguyen@gmail.com

Password

.....

Login

[Not a member? Register now!](#)

This is the Login page covers this section endpoint. Email are provided so that users can login their account by entering an email and password.

Modules used

*Ag-grid-react*

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

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

*React-bootstrap*

Module to provide mainly for the navigation bar, forms, onclick buttons and the application CSS.

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

*Reactstrap*

Module to provide the main dropdown menu features such as button and dropdown item etc.

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

*React-chartjs-2*



Module to provide charting, namely a line graph for the closing price of the stocks.

<https://www.npmjs.com/package/react-chartjs-2>

### *React-hook-form*

Module to handle the on submit process regarding user registration and login validation.

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

### *Axios*

Module to handle the JWT authentication processes via Fetch POST for registration and login for the REST API. Reasoning and justification of the use of axios is listed in the appendices.

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

## Application Design

### Navigation and Layout

The design of the website has a dark midnight blue color theme which compliments the arid-Balham-dark theme for the graph. Alternatives were to use a brighter theme although a dark theme gives contrast to the vibrant colors of the graph, which is more appealing for users to view.

The navigation of the site relies mainly on a navigation bar with header links that take a user to a specific page that queries one of the endpoints from the API. Although there are important two navigation links not listed on the navigation bar, which are login and registration links. There's a login button on the main home page, that once clicked, redirects the user to the login page. User can then authenticate on the page or redirect to the registration page if they need to register.

Upon login authentication, the user is redirected to the home page where the login button is replaced with a logout button. Logging out removes authentication; thus, prevents access to the price history navigation link header.

Clicking the price history navigation link header without user authentication will reject access with an alert – subsequently, the user is redirected to the home page.

There was no critical compromising on the design. Although initially, the navigation links included a register and a login page, and the styling was difficult to float the two-header links to the far-right end of the screen without using margin styling. Margin styling worked but the design would look a little different on a different browser, and it was a little difficult to replace the header links with a logout link when a user authenticates. Even if it is possible to achieve a replacement, the logout link would need to redirect to a page – which is inconvenient as it would be more efficient to logout upon press. Instead, a login button is installed on the home page upon accessing the website. The login button redirects a user to a login page which includes a link to the registration page.



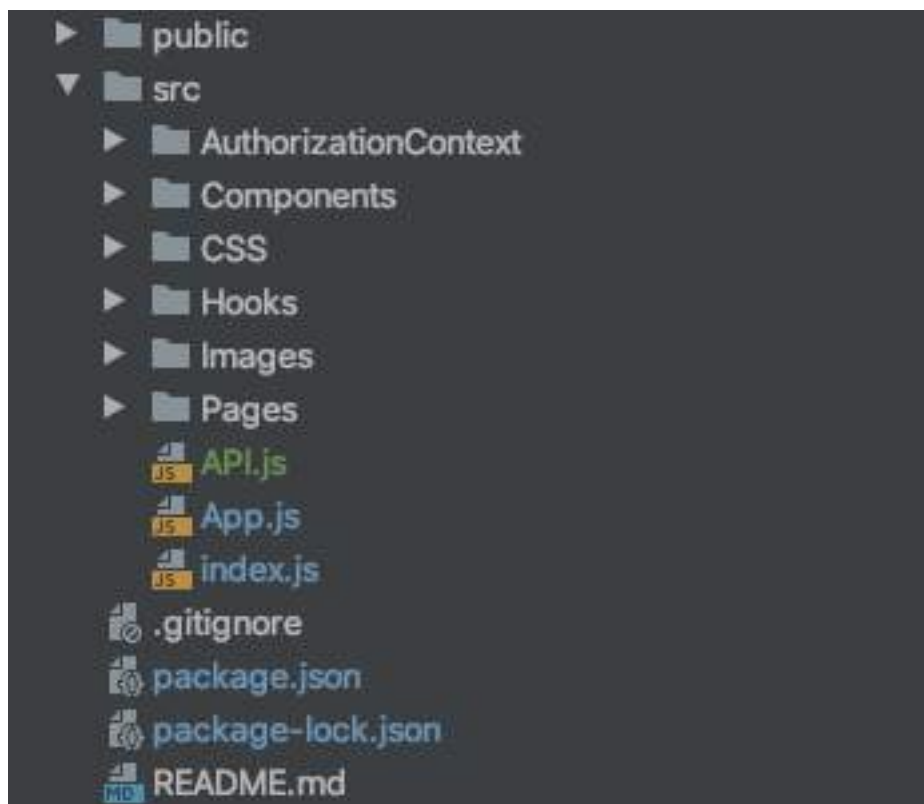
## Technical Description

### Architecture

The architecture of this application at source code level includes a main folder which is the application. Inside that folder includes a public and src folder. Other files include package.json and a readme.md



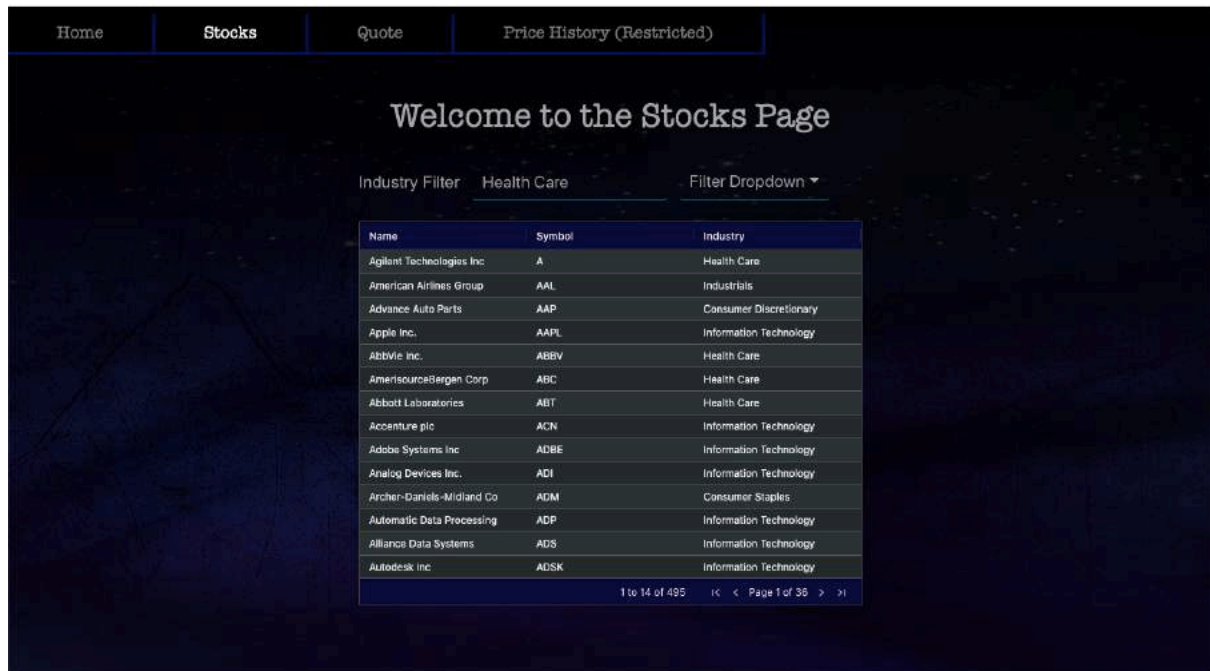
In the main src directory contains the main application files. The responsibilities of each section are allocated within folders. For example, the AuthorizationContext folder provides a main file which uses a context for main APP within index.js. This can provide information of the authentication process to do with the state of login, logout and registration. The components include the main components of app, such as forms and graphing. The CSS folder has the styling sheets for the components and pages. Hooks folder contains hooks for updating the state of the pages from querying an endpoint. Images folder contains images such as the hero image, login, logout and registration.



## Test plan

/stocks/symbols endpoint

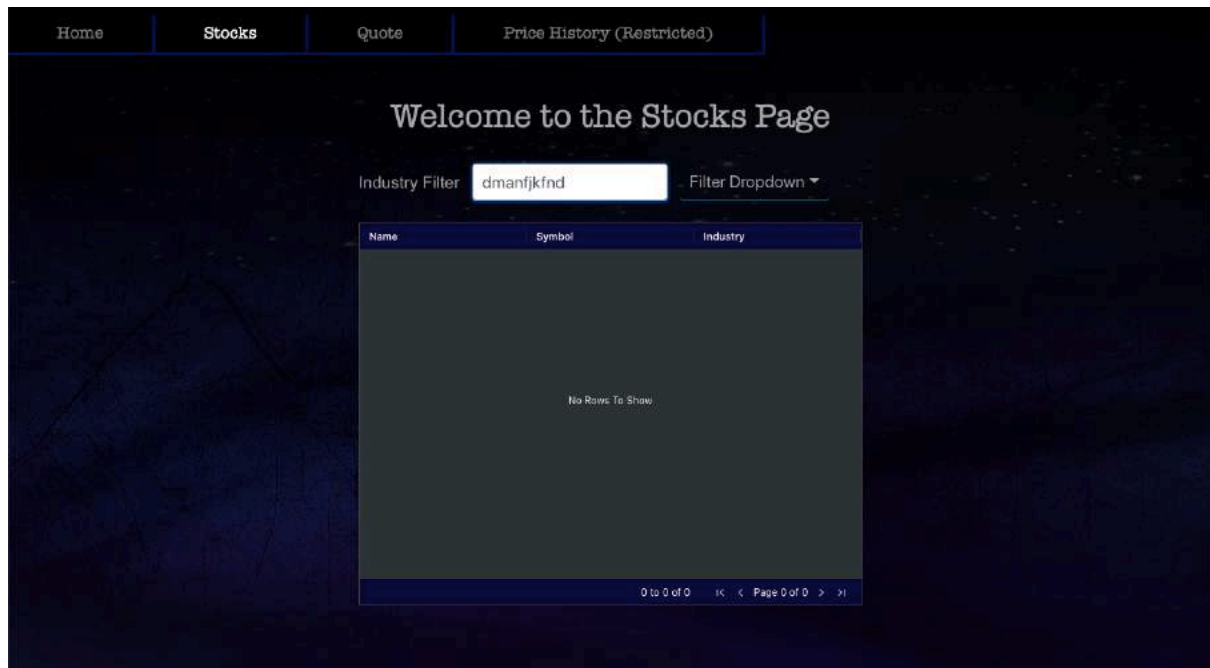
Positive case:



The screenshot shows the 'Stocks' page with a navigation bar at the top containing 'Home', 'Stocks', 'Quote', and 'Price History (Restricted)'. The main heading is 'Welcome to the Stocks Page'. Below this, there is an 'Industry Filter' set to 'Health Care' and a 'Filter Dropdown' menu. A table displays a list of companies in the Health Care industry. The table has three columns: 'Name', 'Symbol', and 'Industry'. The companies listed are Agilent Technologies Inc., American Airlines Group, Advance Auto Parts, Apple Inc., Abbvie Inc., AmersourceBergen Corp., Abbott Laboratories, Accenture plc, Adobe Systems Inc., Analog Devices Inc., Archer-Daniels-Midland Co., Automatic Data Processing, Alliance Data Systems, and Autodesk Inc. The table is paginated, showing '1 to 14 of 495' results, with navigation links for '1', '<', 'Page 1 of 36', '>', and '36'.

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
AmersourceBergen 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
Automatic Data Processing	ADP	Information Technology
Alliance Data Systems	ADS	Information Technology
Autodesk Inc.	ADSK	Information Technology

Negative case:

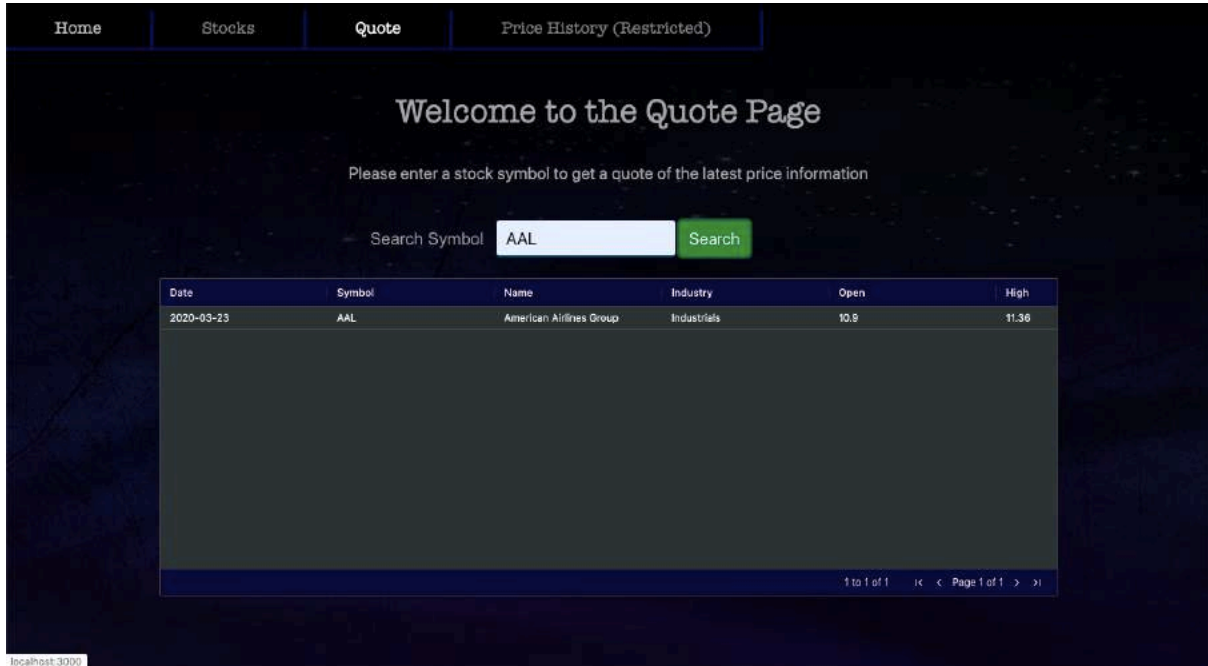


The screenshot shows the 'Stocks' page with the same navigation bar. The 'Industry Filter' is set to 'dmanfjkfnd', which is an invalid filter. The 'Filter Dropdown' menu is open. The table below the filter shows 'No Rows To Show'. The pagination at the bottom indicates '0 to 0 of 0' results, with navigation links for '1', '<', 'Page 0 of 0', '>', and '0'.

Name	Symbol	Industry
No Rows To Show		

/stocks/{symbols} endpoint

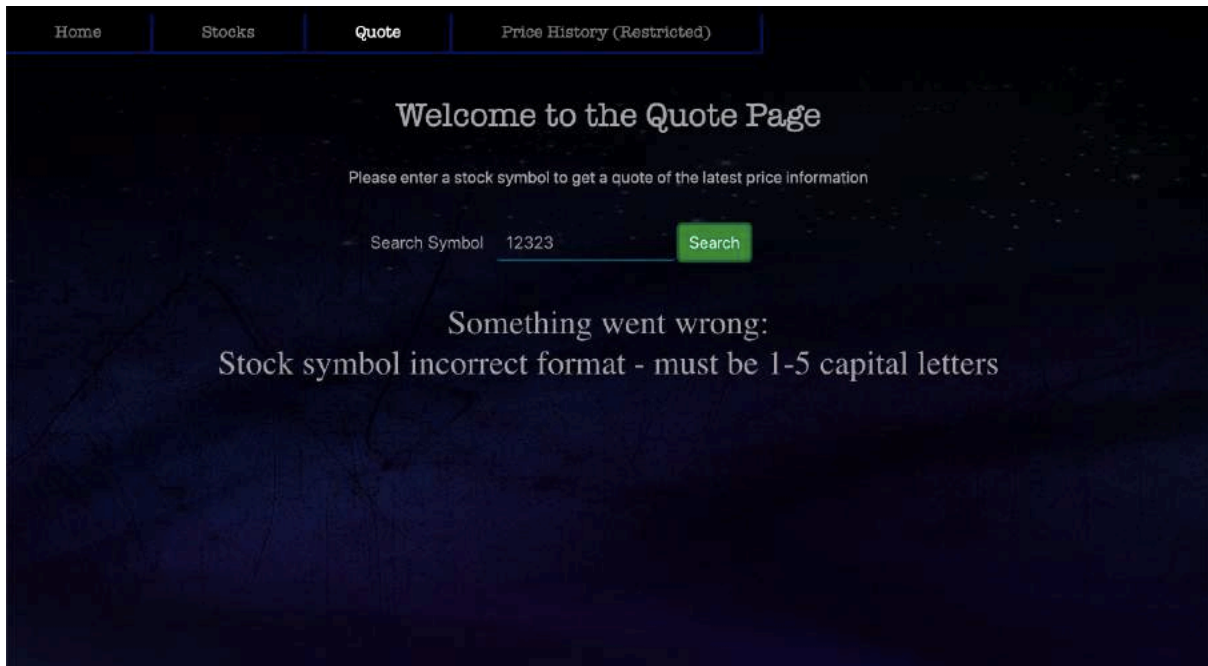
Positive case:



The screenshot shows a web application interface for stock quotes. At the top, there is a navigation bar with four tabs: Home, Stocks, Quote (which is active), and Price History (Restricted). Below the navigation bar, the page is titled "Welcome to the Quote Page". A message prompts the user to "Please enter a stock symbol to get a quote of the latest price information". A search bar labeled "Search Symbol" contains the text "AAL", and a green "Search" button is next to it. Below the search bar, a table displays the stock information for AAL. The table has six columns: Date, Symbol, Name, Industry, Open, and High. The data row shows the date 2020-03-23, symbol AAL, name American Airlines Group, industry Industrials, an open price of 10.9, and a high price of 11.36. At the bottom right of the table, there is a pagination control showing "1 to 1 of 1" and navigation arrows. A small "localhost:3000" label is visible in the bottom left corner of the browser window.

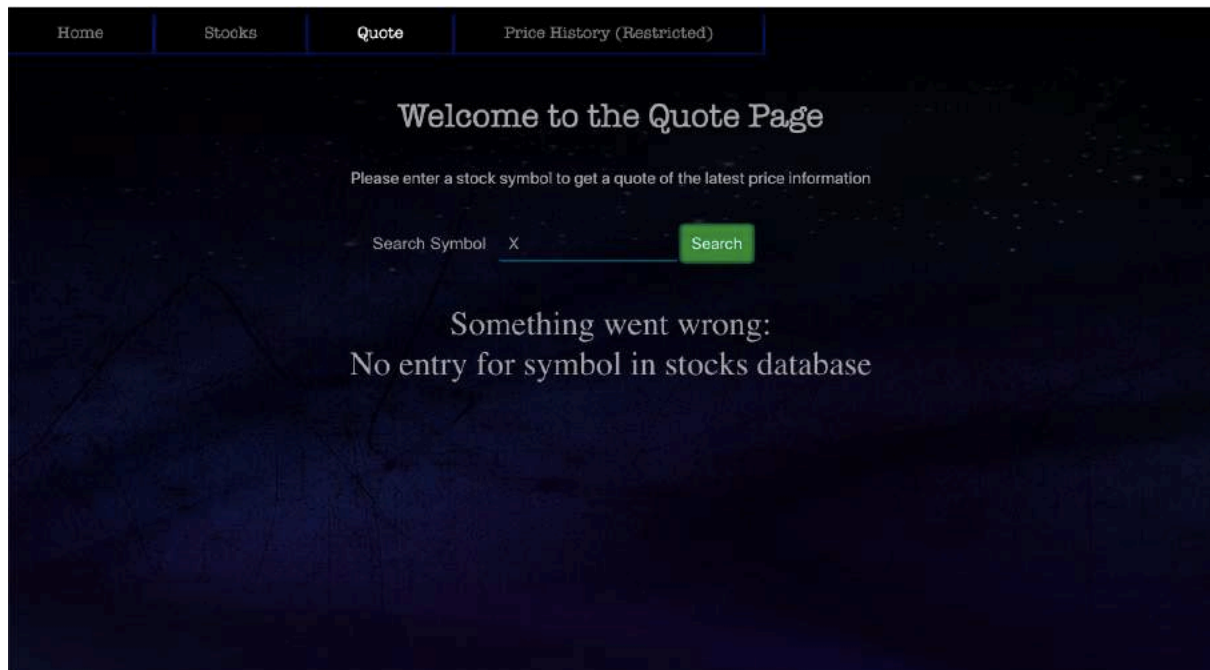
Date	Symbol	Name	Industry	Open	High
2020-03-23	AAL	American Airlines Group	Industrials	10.9	11.36

Negative case: When user enters an incorrect format/length of the stock symbol



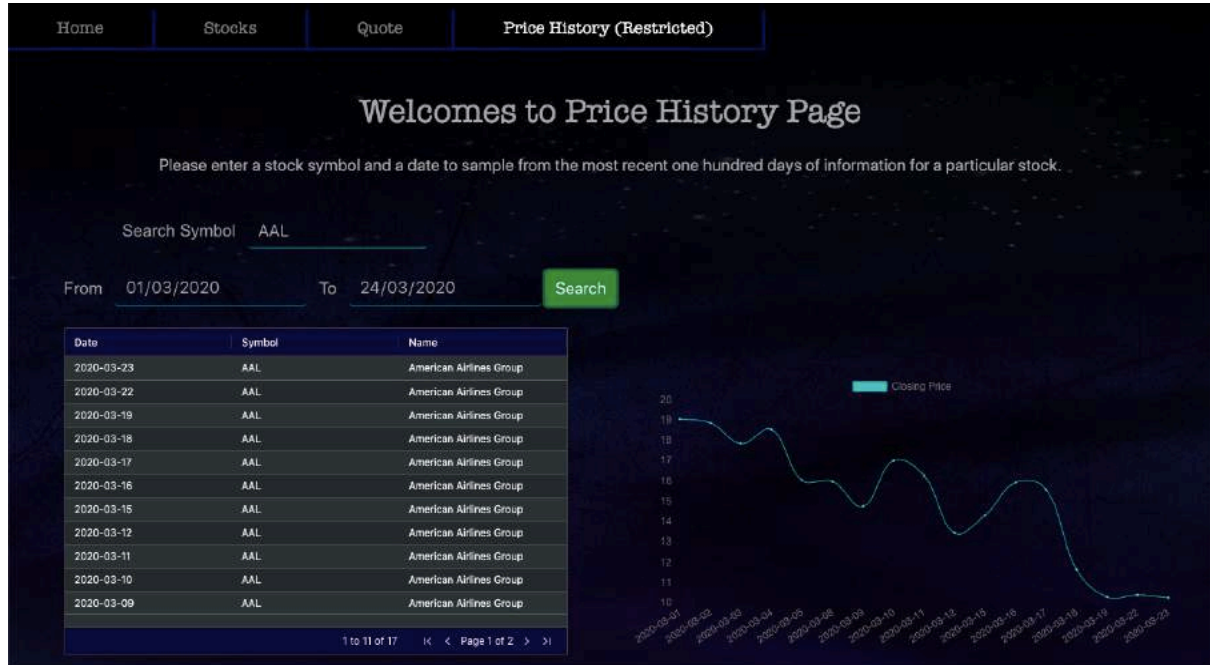
The screenshot shows the same web application interface as the previous one, but with an error message. The navigation bar and "Welcome to the Quote Page" header are the same. The search bar now contains the text "12323" instead of "AAL". The green "Search" button is still present. Below the search bar, a message states: "Something went wrong: Stock symbol incorrect format - must be 1-5 capital letters".

Negative case: When user presses incorrect symbol

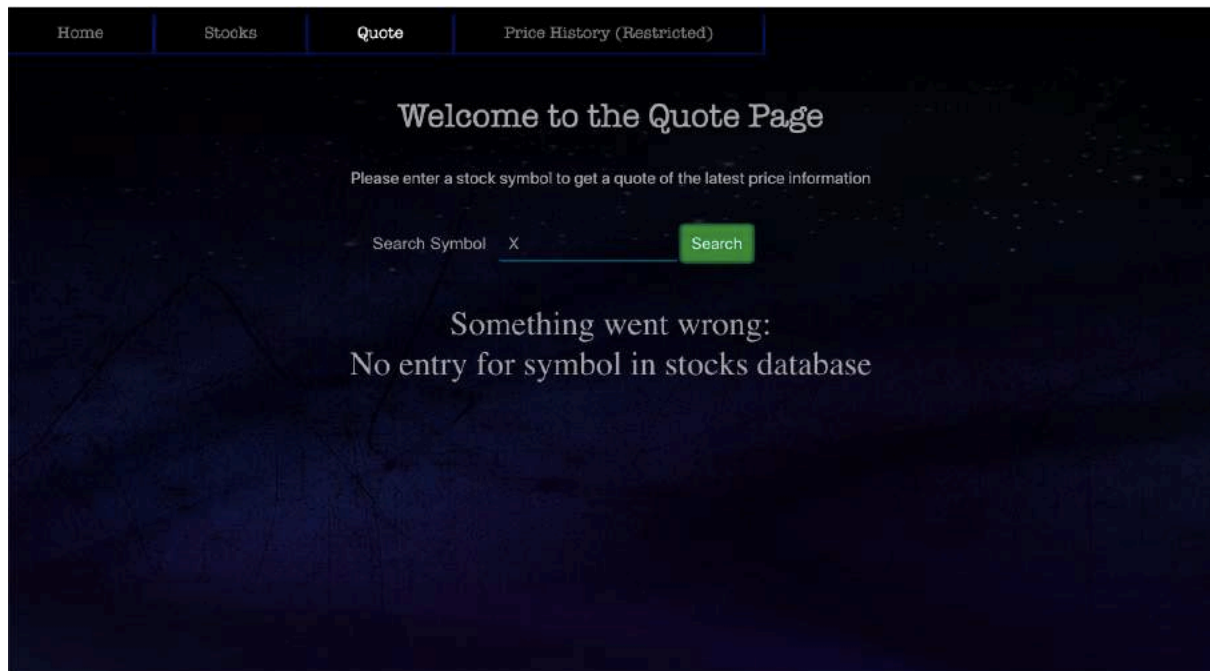


/stocks/authed/{symbols} endpoint

Positive case:

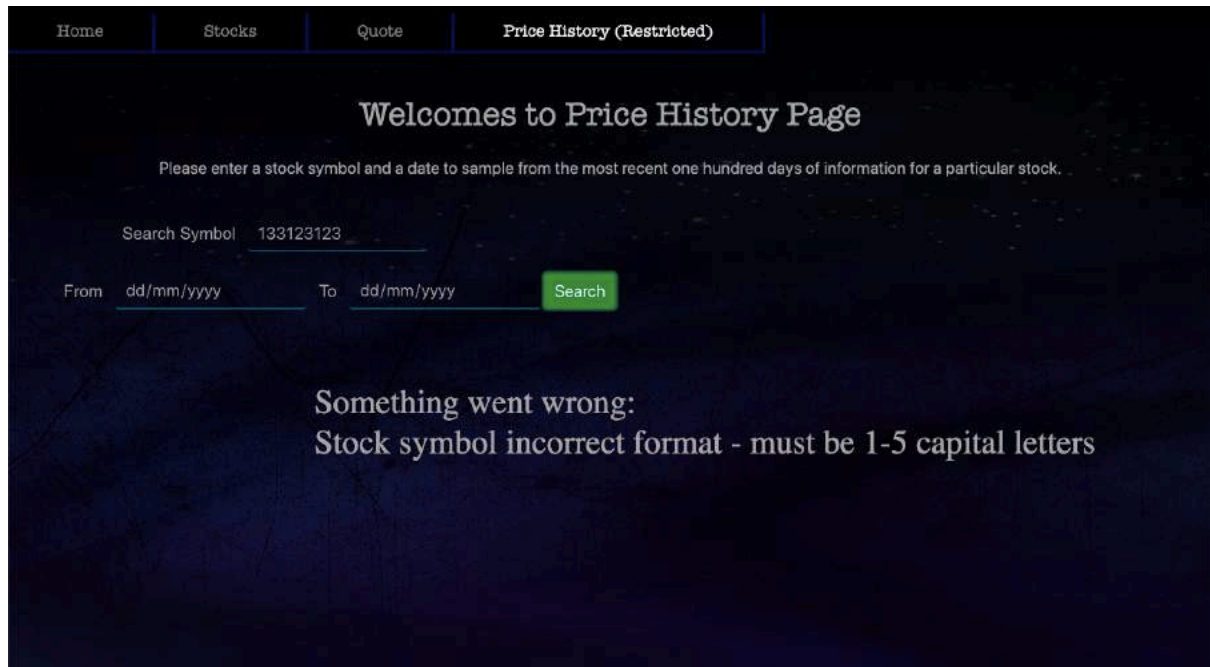


Negative case: When user presses incorrect symbol



The screenshot shows a web application with a dark blue background and a light blue navigation bar at the top. The navigation bar contains four links: Home, Stocks, Quote, and Price History (Restricted). The main content area is titled "Welcome to the Quote Page" and contains a prompt: "Please enter a stock symbol to get a quote of the latest price information". Below this prompt is a search form with a label "Search Symbol", a text input field containing the letter "X", and a green "Search" button. Below the search form, a message states: "Something went wrong: No entry for symbol in stocks database".

Negative case: When user enters an incorrect format/length of the stock symbol



The screenshot shows a web application with a dark blue background and a light blue navigation bar at the top. The navigation bar contains four links: Home, Stocks, Quote, and Price History (Restricted). The main content area is titled "Welcomes to Price History Page" and contains a prompt: "Please enter a stock symbol and a date to sample from the most recent one hundred days of information for a particular stock.". Below this prompt is a search form with a label "Search Symbol", a text input field containing the string "133123123", and a green "Search" button. Below the search form, there are two date input fields labeled "From" and "To", both with placeholder text "dd/mm/yyyy", and a green "Search" button. Below the search form, a message states: "Something went wrong: Stock symbol incorrect format - must be 1-5 capital letters".



Negative case: When user provides date and time query outside the scope of the stock dates

The screenshot shows a web application with a dark blue background and a light blue navigation bar at the top. The navigation bar contains four links: "Home", "Stocks", "Quote", and "Price History (Restricted)". Below the navigation bar, the page title "Welcomes to Price History Page" is displayed in a large, bold, white font. Underneath the title, a message in a smaller white font reads: "Please enter a stock symbol and a date to sample from the most recent one hundred days of information for a particular stock." Below this message, there is a search form. The form has a "Search Symbol" label followed by a text input field containing "AAL". Below the symbol field, there are two date input fields labeled "From" and "To". The "From" field contains "01/07/2019" and the "To" field contains "05/09/2019". To the right of the "To" field is a green "Search" button. Below the search form, a message in a large, bold, white font reads: "Something went wrong: No entries available for query symbol for supplied date range".

Edge case: Unlike chrome, on a safari browser the date forms don't provide a strict layout for inputting the date, nor does it provide a date picker. Thus, entering an incorrect date will result this:

The screenshot shows the same web application as the first image. The navigation bar and page title are identical. The message below the title is also the same. However, the search form has different input values. The "Search Symbol" field contains "A". The "From" date field contains "123122133213213" and the "To" date field contains "1231231212213". The green "Search" button is still present. Below the search form, a message in a large, bold, white font reads: "Something went wrong: From date cannot be parsed by Date.parse()".

/user/login

Positive case:

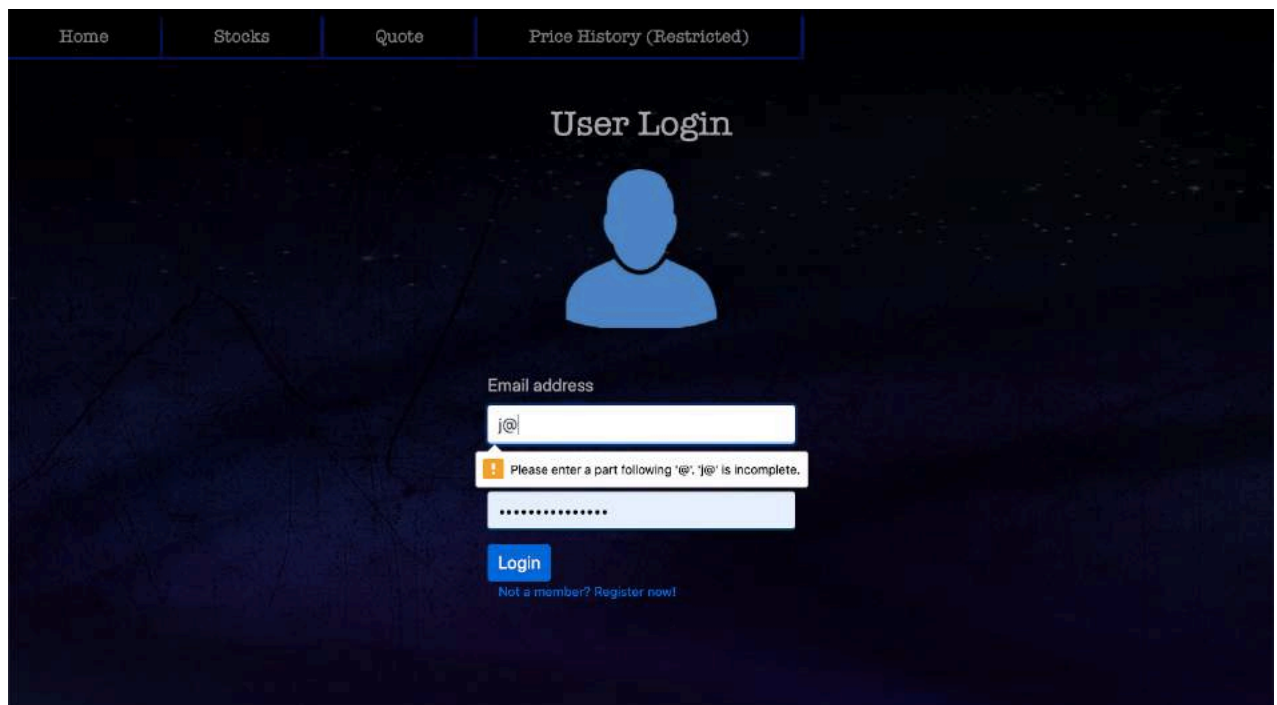
The screenshot shows a web application with a dark blue background and a navigation bar at the top with links for 'Home', 'Stocks', and 'Quote'. A white notification box in the top right corner displays the message: 'localhost:3000 says You successfully login' with an 'OK' button. The main content area is titled 'User Login' and features a blue silhouette of a person. Below the silhouette are two input fields: 'Email address' containing 'jnguyen911@gmail.com' and 'Password' with masked characters. A blue 'Login' button is positioned below the password field, followed by the text 'Not a member? Register now!'.

Negative case: when user types an invalid email account – does not exist or incorrect password

This screenshot shows the same login page as the previous one, but with an error message. The notification box now displays: 'localhost:3000 says Error: Request failed with status code 401 Invalid Account. Try again.' The 'Email address' field contains 'jnguyen123911@gmail.com', which is underlined in red to indicate it is invalid. The 'Password' field is masked, and the 'Login' button remains visible below it, along with the 'Not a member? Register now!' link.



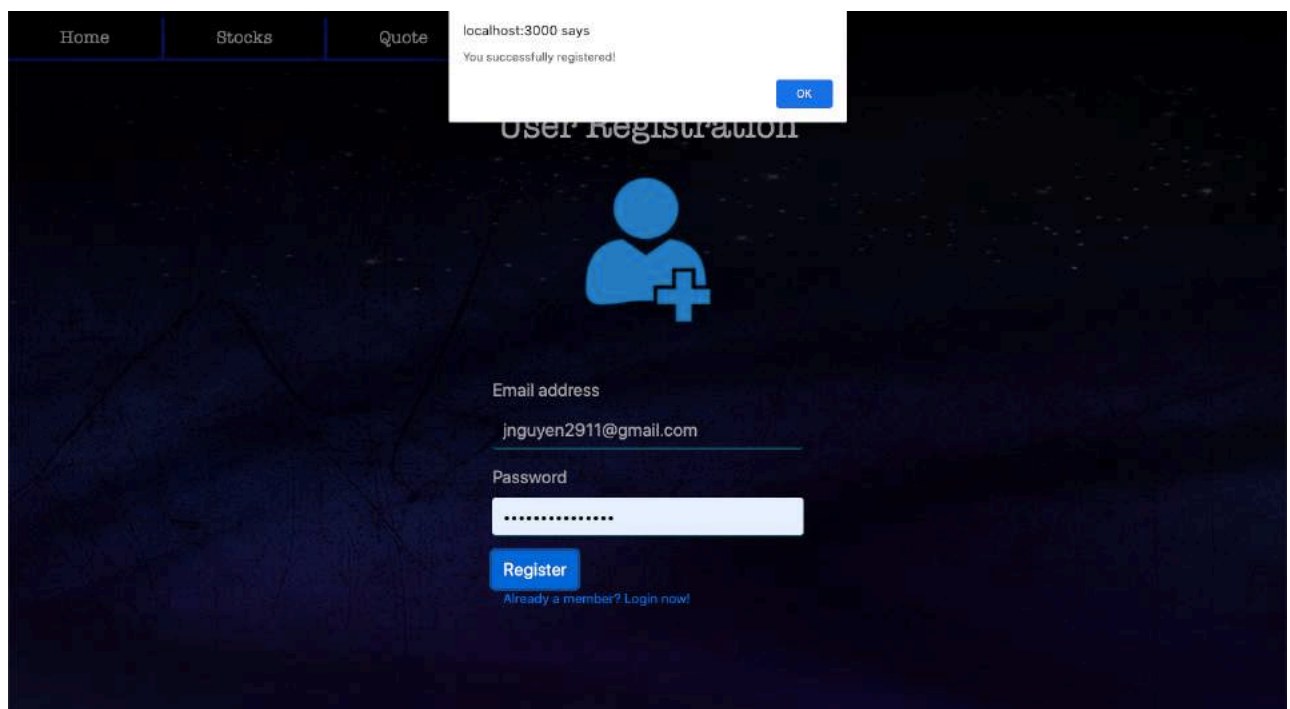
Negative case: when user enters incorrect format of email



The screenshot shows the 'User Login' page of a web application. At the top, there is a navigation bar with links: Home, Stocks, Quote, and Price History (Restricted). The main heading is 'User Login' with a blue silhouette icon of a person. Below the icon, there is a form with two input fields. The first field is labeled 'Email address' and contains the text 'j@'. A yellow error message box is displayed below this field, stating: 'Please enter a part following '@'. 'j@' is incomplete.' The second field is a password field with masked characters '.....'. Below the password field is a blue 'Login' button. At the bottom, there is a link that says 'Not a member? Register now!'.

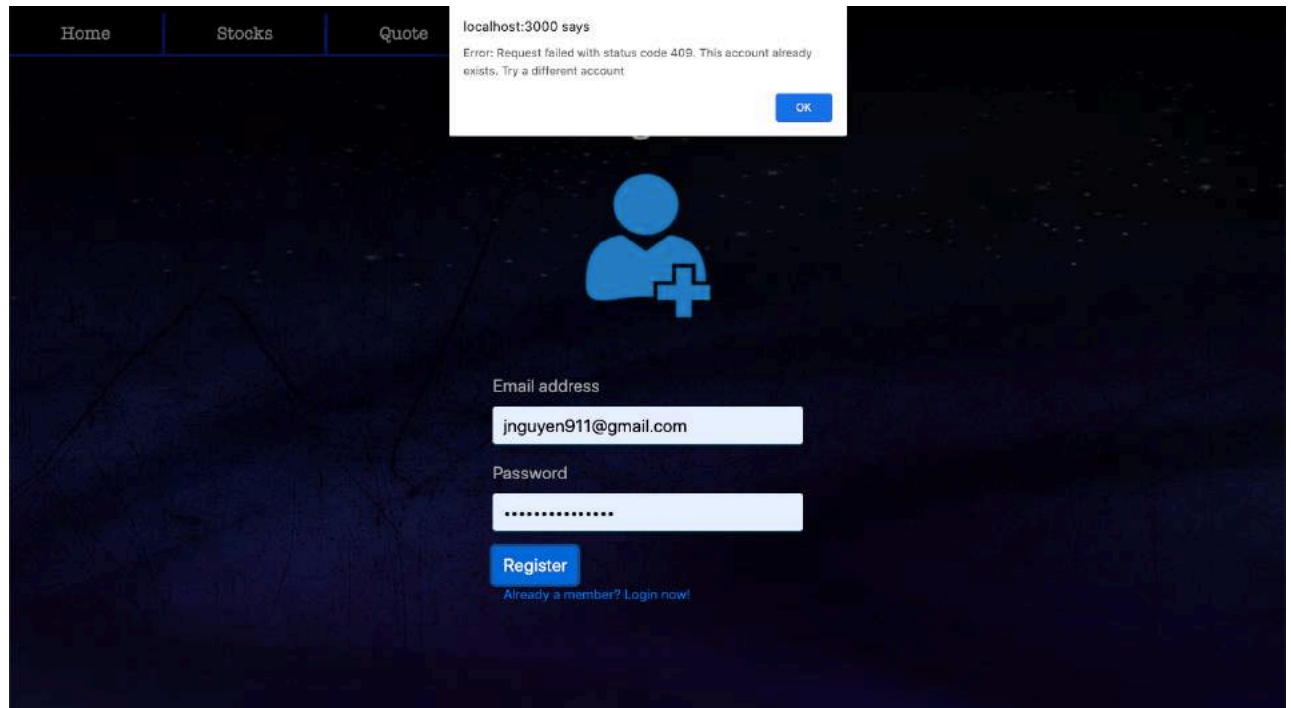
/user/register

Positive case:



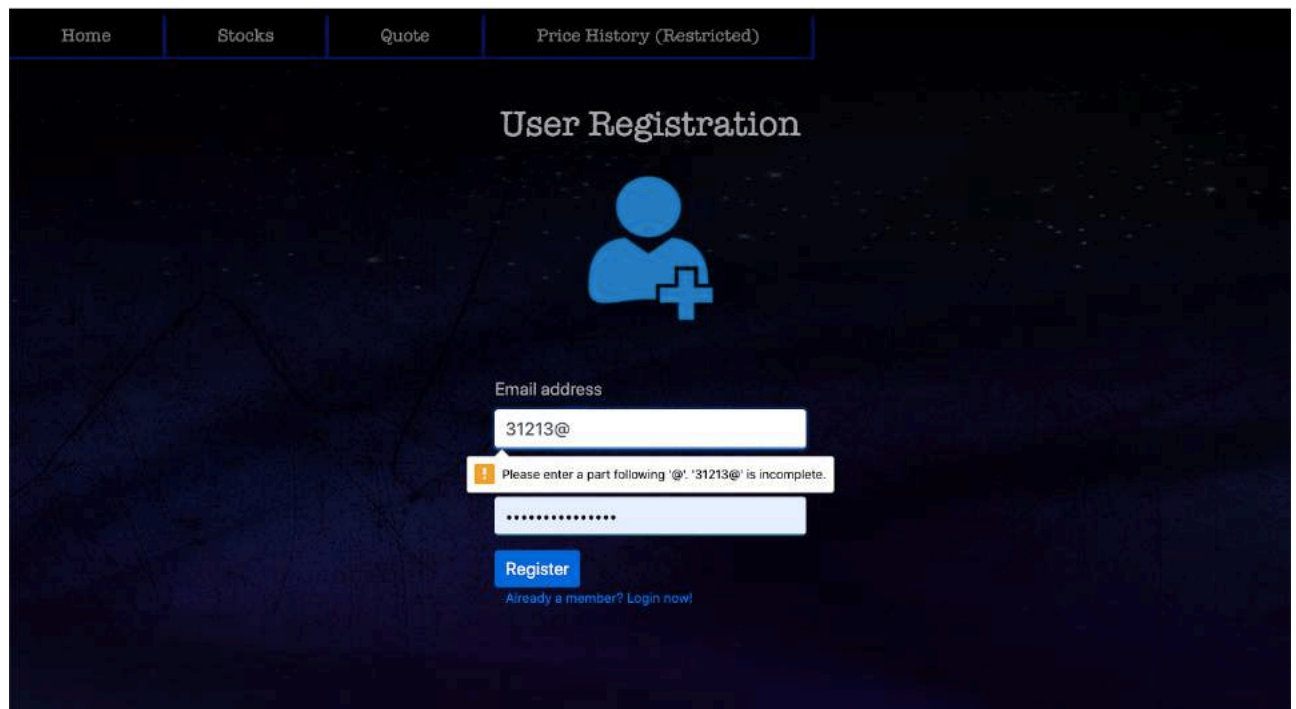
The screenshot shows the 'User Registration' page of a web application. At the top, there is a navigation bar with links: Home, Stocks, and Quote. A white toast notification is displayed at the top center, stating: 'localhost:3000 says You successfully registered!' with an 'OK' button. The main heading is 'User Registration' with a blue silhouette icon of a person with a plus sign. Below the icon, there is a form with two input fields. The first field is labeled 'Email address' and contains the text 'jnguyen2911@gmail.com'. The second field is labeled 'Password' and contains masked characters '.....'. Below the password field is a blue 'Register' button. At the bottom, there is a link that says 'Already a member? Login now!'.

Negative case: When user enters an email that already exists



The screenshot shows a web application with a dark blue background and a light blue navigation bar at the top containing the links "Home", "Stocks", and "Quote". A white error message box is displayed in the upper right, stating "localhost:3000 says" and "Error: Request failed with status code 409. This account already exists. Try a different account", with an "OK" button. Below the navigation bar, there is a large light blue icon of a person with a plus sign. Underneath the icon, the text "Email address" is followed by a text input field containing "jnguyen911@gmail.com". Below this, the text "Password" is followed by a password input field with masked characters. A blue "Register" button is positioned below the password field, and a link "Already a member? Login now!" is located directly beneath the button.

Negative case: When user enters an email by incorrect format



The screenshot shows a web application with a dark blue background and a light blue navigation bar at the top containing the links "Home", "Stocks", "Quote", and "Price History (Restricted)". The main heading "User Registration" is centered above a large light blue icon of a person with a plus sign. Below the icon, the text "Email address" is followed by a text input field containing "31213@". A yellow error message box with an exclamation mark icon is displayed below the input field, stating "Please enter a part following '@'. '31213@' is incomplete." Below the error message, there is a password input field with masked characters. A blue "Register" button is positioned below the password field, and a link "Already a member? Login now!" is located directly beneath the button.

[Difficulties / Exclusions / unresolved & persistent errors /](#)

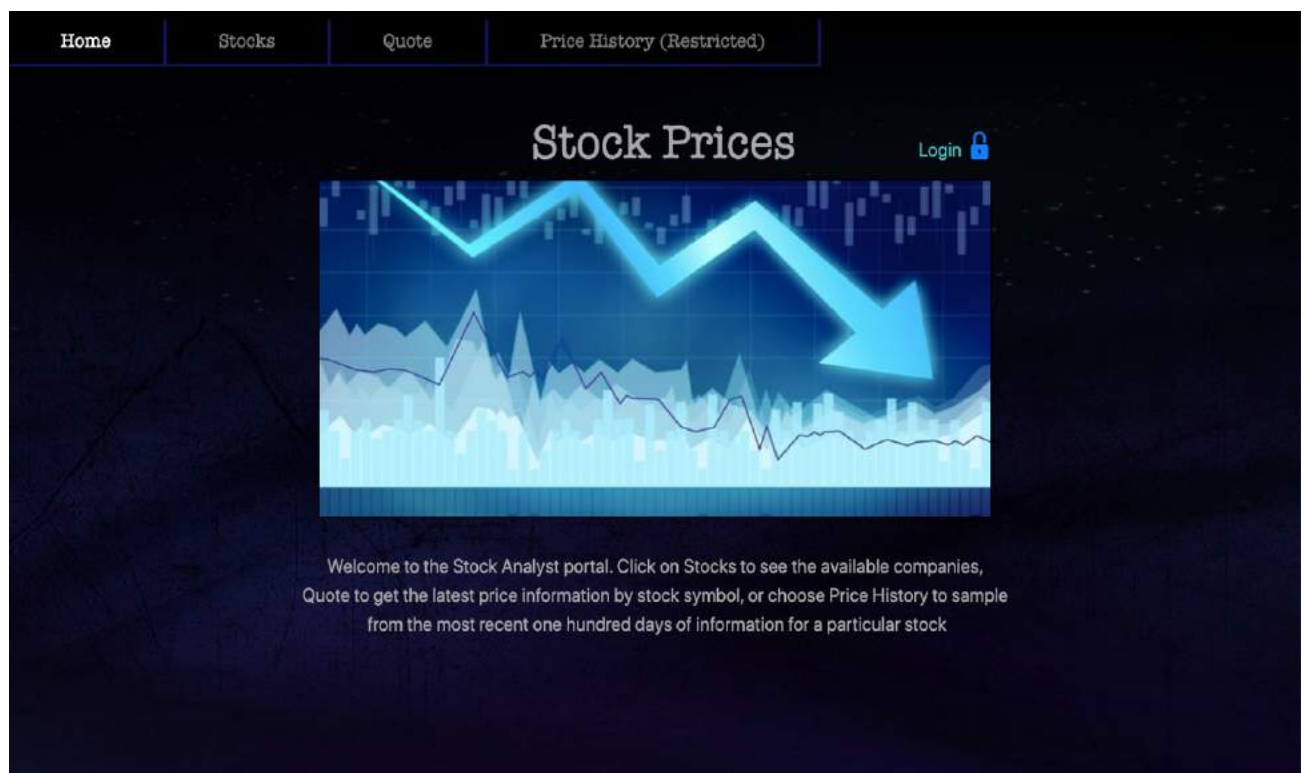
Some difficulties met were working with the JWT authentication forms, specifically for post querying for the user/login endpoint. When using the standard fetch Post method to convert the json data and returning the token value, I could not find a way to catch the error using standard fetch. See appendices for screenshots of the results.

The issue was resolved using axios with the async and await methodology. Using axios to catch using the error post query was well regarded and successful.

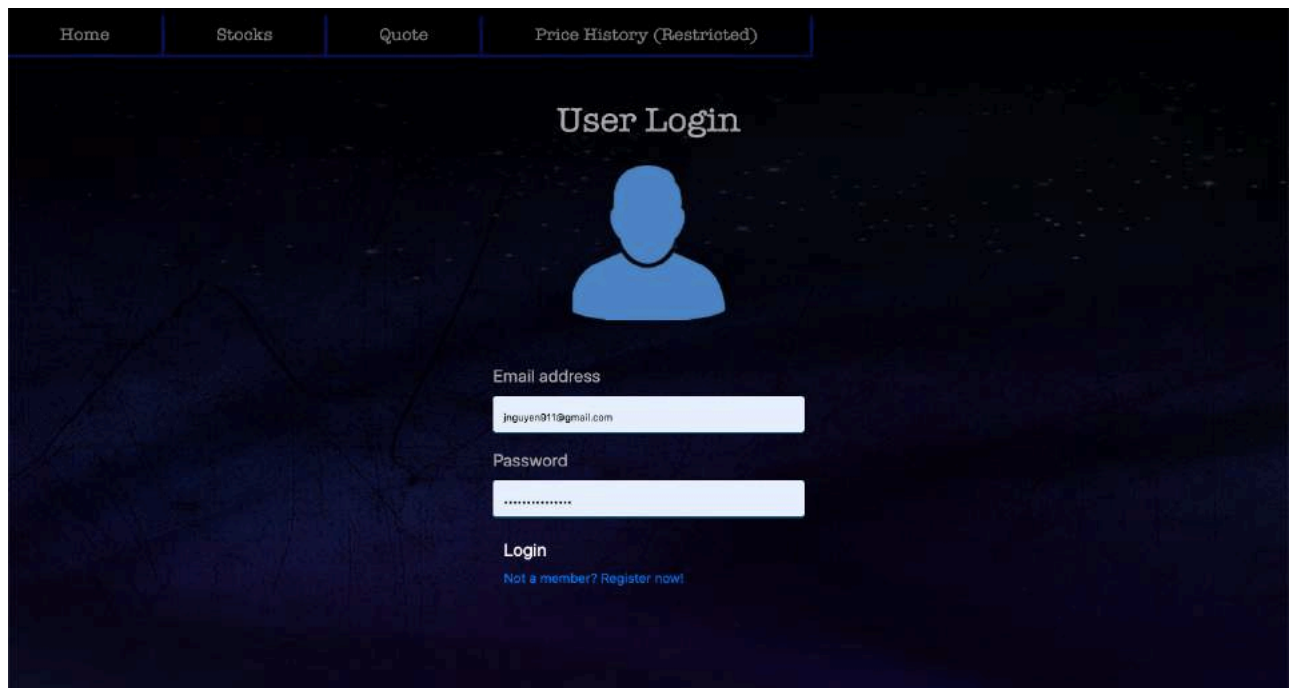
Most of the errors for this program other than the JWT forms have been quite trivial to be worthy of mention.

## [User guide](#)

Upon visit, user lands onto the home page



User can then press the login button to the login page



The image shows a 'User Login' page with a dark blue background and a light blue mountain silhouette. At the top, there is a navigation bar with four items: 'Home', 'Stocks', 'Quote', and 'Price History (Restricted)'. The main heading is 'User Login' in a light blue serif font. Below the heading is a light blue icon of a person's head and shoulders. The form contains two input fields: 'Email address' with the value 'jnguyen911@gmail.com' and 'Password' with masked characters. Below the password field is a 'Login' button and a link that says 'Not a member? Register now!'.

Home Stocks Quote Price History (Restricted)

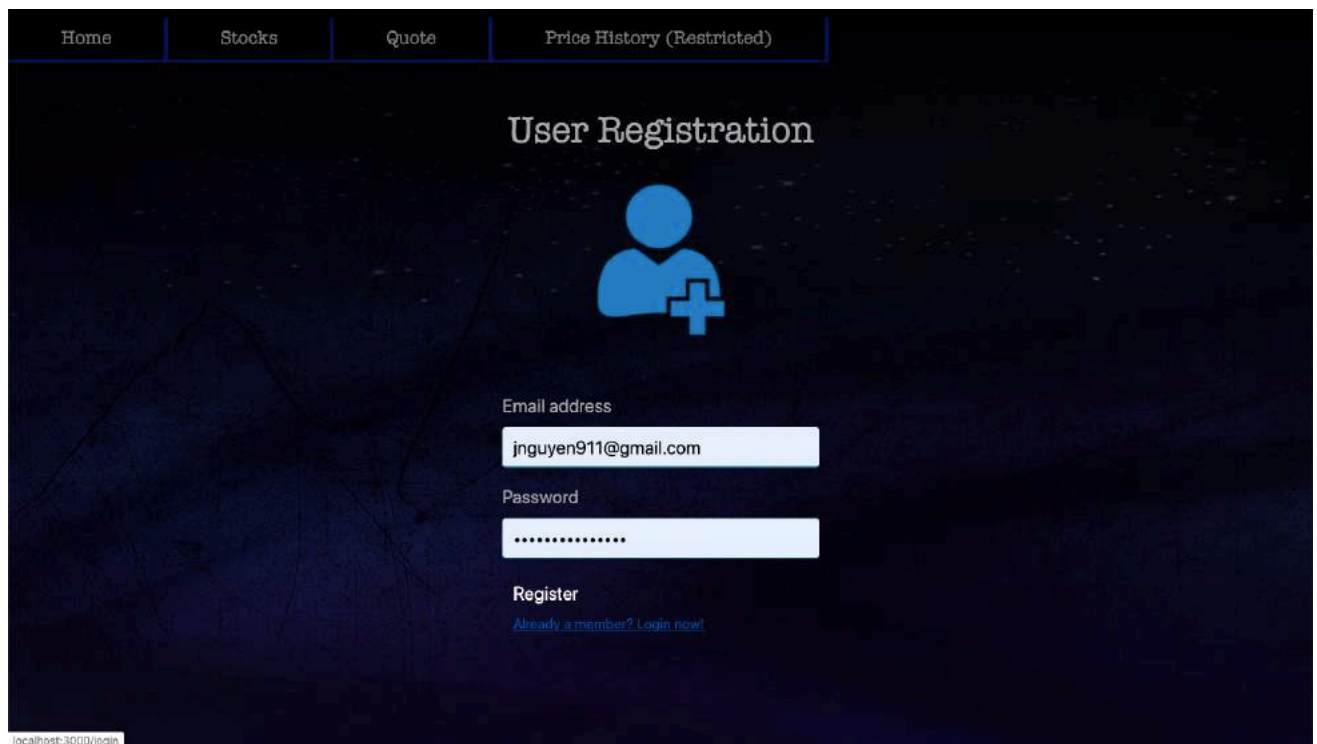
## User Login

Email address  
jnguyen911@gmail.com

Password  
\*\*\*\*\*

Login  
[Not a member? Register now!](#)

User can decide to register if their account is not registered, or they can login which they will be redirected back to the home page.



The image shows a 'User Registration' page with a dark blue background and a light blue mountain silhouette. At the top, there is a navigation bar with four items: 'Home', 'Stocks', 'Quote', and 'Price History (Restricted)'. The main heading is 'User Registration' in a light blue serif font. Below the heading is a light blue icon of a person's head and shoulders with a plus sign. The form contains two input fields: 'Email address' with the value 'jnguyen911@gmail.com' and 'Password' with masked characters. Below the password field is a 'Register' button and a link that says 'Already a member? Login now!'.

Home Stocks Quote Price History (Restricted)

## User Registration

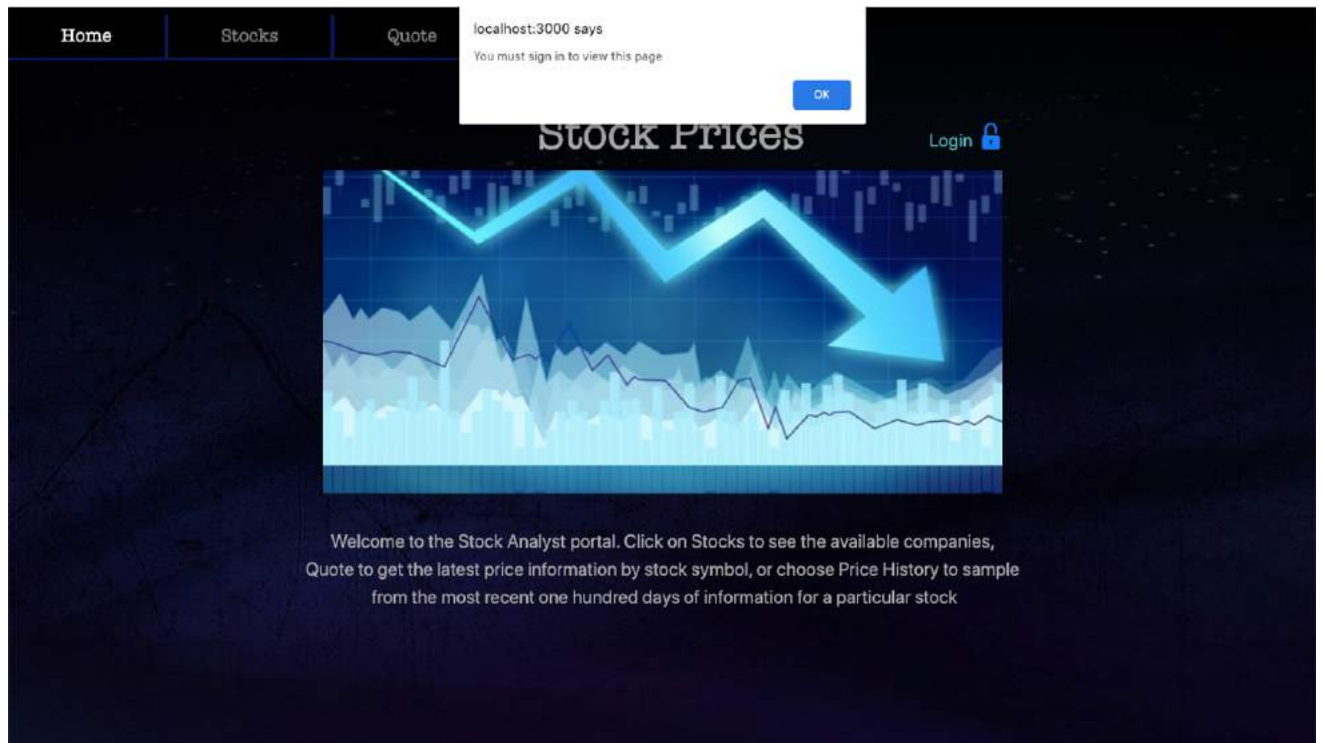
Email address  
jnguyen911@gmail.com

Password  
\*\*\*\*\*

Register  
[Already a member? Login now!](#)

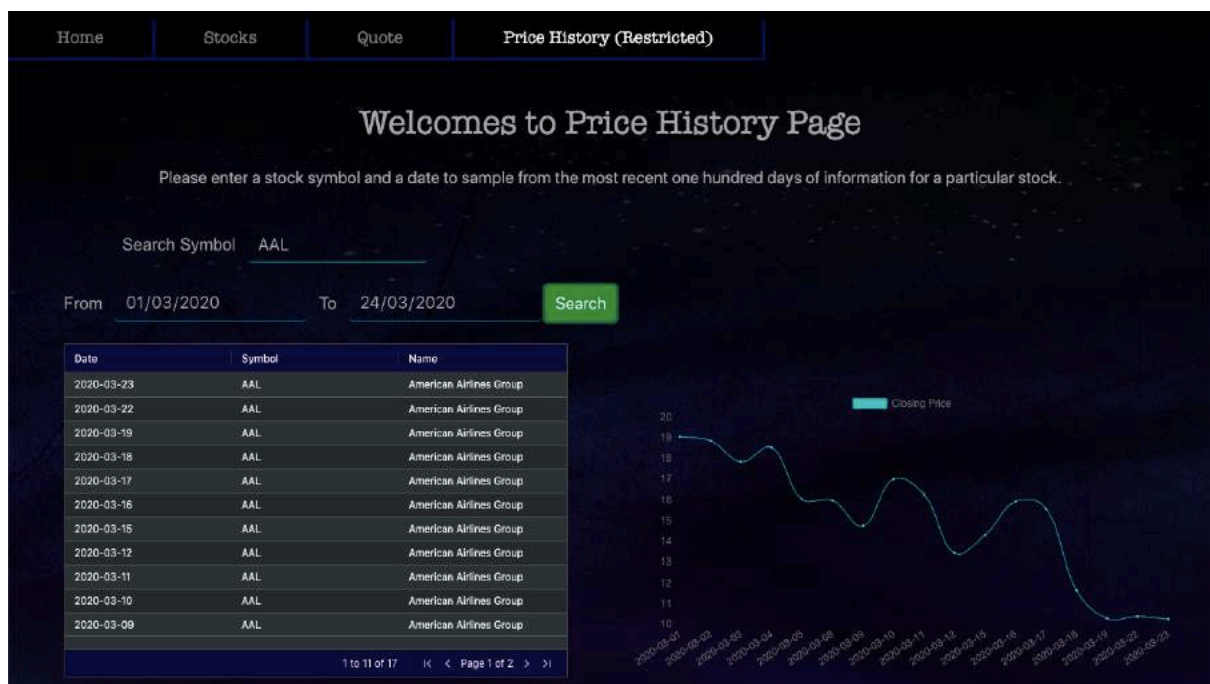
localhost:3000/login

If the user is not logged in, they cannot access the price history page



Otherwise if the user is logged in, they have access to the price history page.

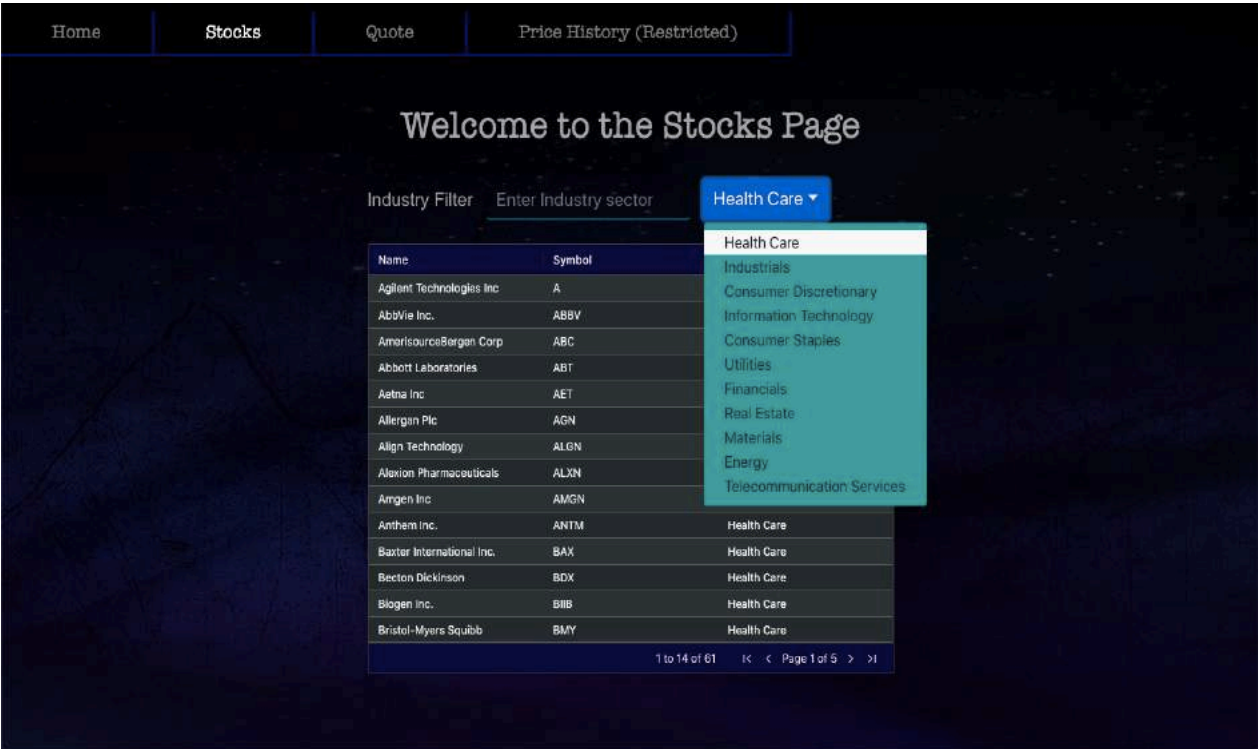
User can make a query to display the price history of the stocks which includes a grid table and graph.



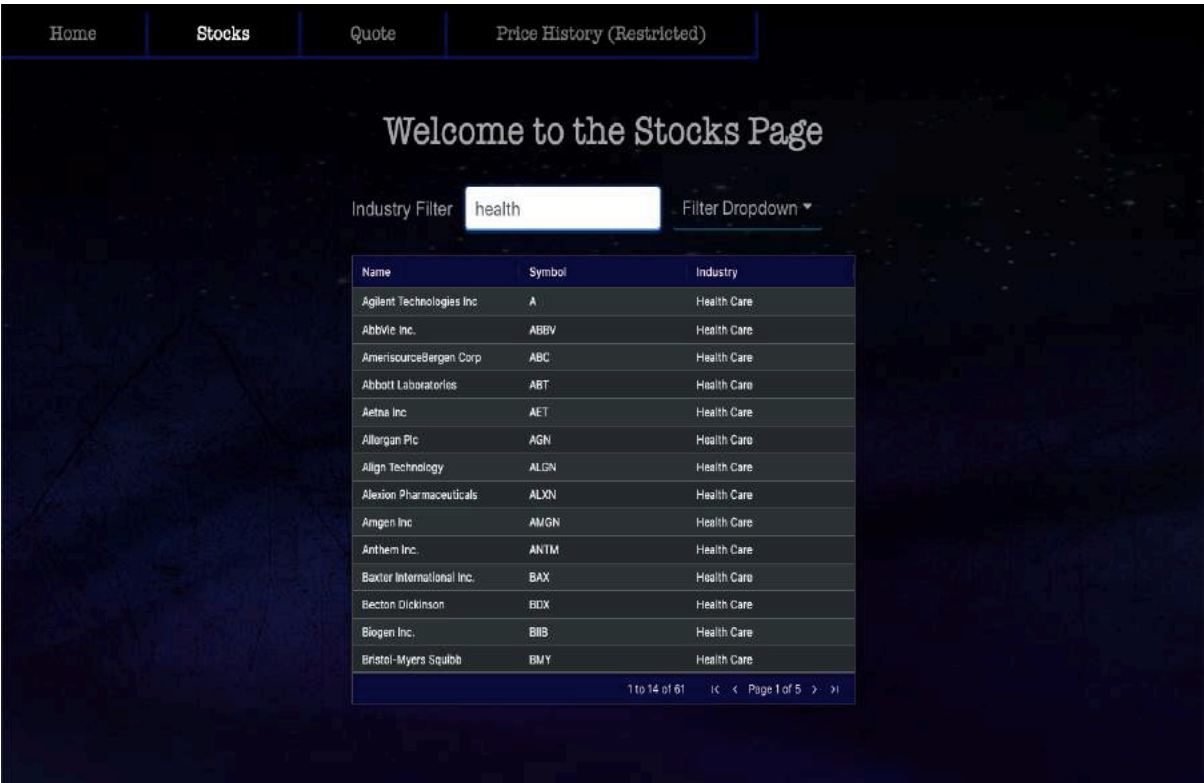


Clicking the stocks navigation link, user is redirected to the stocks page. User can then filter the stocks list by either inputting a search term through the search bar or using the dropdown button.

Filter by dropdown:



Filter by search bar:



When the user pressed on the quote navigation link, it should display a search form and a grid (a grid is displayed providing the user enters a valid search symbol

[Home](#)[Stocks](#)[Quote](#)[Price History \(Restricted\)](#)

## Welcome to the Quote Page

Please enter a stock symbol to get a quote of the latest price information

Search Symbol

Date	Symbol	Name	Industry	Open	High
2020-03-23	AAL	American Airlines Group	Industrials	10.9	11.36

1 to 1 of 1    < > Page 1 of 1



## Appendices

*JWT error catch problem using fetch: (resolved using Axios)*

Using Fetch:

This particular fetch function fails to catch an error when post querying an invalid account to the API

```
//-----POST-----  
  
// ValidateLogin uses Post method to login user through the API (retrieving token)  
export const ValidateLogin = (email, password) => {  
  const url = baseUrl + '/user/login';  
  
  // Using fetch to post user login email and password  
  return (fetch(url, {  
    method: 'POST',  
    headers: { accept: "application/json", "Content-type":"application/json"},  
    body: JSON.stringify({ email: email, password: password })  
  })  
    .then((res) => res.json())  
    .then((res) => res.token)  
  )  
}
```

Using Axios:

This axios function successfully catches an error when post querying an invalid account to the API.

```
//-----POST-----  
  
// ValidateLogin uses Axios Post method to login user through the API (retrieving token)  
export const ValidateLogin = async (email, password) => {  
  
  const MessageResponse = await axios({  
    method: 'POST',  
    baseUrl,  
    url: '/user/login',  
    data: {  
      email,  
      password  
    }  
  });  
  
  return MessageResponse.data.token;  
};
```

Here is the hook which is used for the function mentioned above. I have implemented a state variable for the token and error value, but let's focus our attention to the error state, which is returned from this hook. If an error is caught, then we can assume the value of error is not null and changes to the error value. However, if the value remains null, then in this context we will assume that error has failed to be caught by the hook – since our query is set up to fail by setting an invalid account.

Notice that the values passed through to the ValidateLogin function are an unregistered email and password. So, the post query to the API endpoint should guarantee a fail.

```
import {ValidateLogin} from "../API";

function useLogin(){
  const [token, setToken] = useState([]);
  const [error, setError] = useState(null);

  const unregisteredEmail = "jnguyen1236699@gmail.com";
  const unregisteredPassword = "chucksoyandr911";
  useEffect(() => {
    ValidateLogin(unregisteredEmail, unregisteredPassword)
      .then((tokenValue) => {
        setToken(tokenValue)
      })
      .catch((err) => {
        setError(err)
      });
  }, []);
  return {
    token,
    error
  }
}
```

To access the error variable, we simply call the hook and grab it. Then we can print to console.

```
const { error } = useLogin()

console.log(error + "",)
```

Regarding the Fetch function, when observing the console IDE in browser developer tools, it is unsuccessful in catching the state of the error:

```
[HMR] Waiting for update signal from WDS... log.js:24
Download the React DevTools for a better development experience: https://fb.me/react-devtools
react-dom.development.js:24994
null HomePage.js:36
❌ ▶ POST http://131.181.190.87:3000/user/login 401 (Unauthorized) API.js:11
null HomePage.js:36
>
```

First the value of the error state is null, then after the post query to the login endpoint failed, it is still null.

Therefore, the error state could not be caught.

Regarding the Axios function, when observing the console IDE in browser developer tools, it is successful in catching the state of the error:

```
[HMR] Waiting for update signal from WDS... log.js:24
Download the React DevTools for a better development experience: https://fb.me/react-devtools
react-dom.development.js:24994
null HomePage.js:36
❌ ▶ POST http://131.181.190.87:3000/user/login 401 (Unauthorized) xhr.js:178
Error: Request failed with status code 401 HomePage.js:36
> |
```

First the value of the error state null, then after the post query to the login endpoint failed, the value is changed

Therefore, the error state has been successfully caught.