

# CAB230 Stocks API – Client Side

CAB230 | 2020 S1

Stocks API – Client-Side Application

Max O'Dell

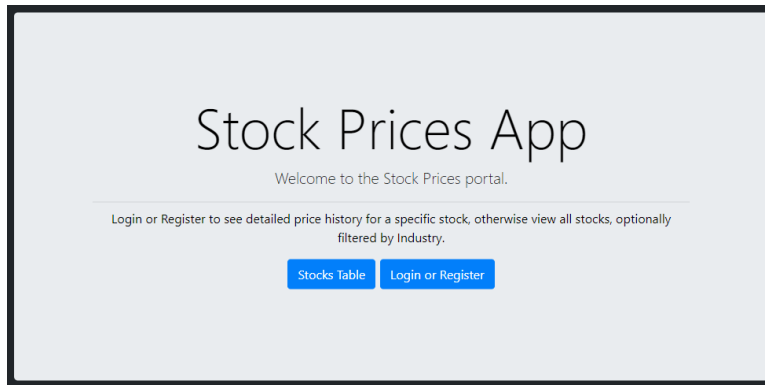
n10336516

## Contents

Introduction .....	2
Purpose & Description .....	2
Completeness and Limitations.....	2
Use of End Points .....	3
/stocks/symbols .....	3
/stocks/{symbol} .....	3
/stocks/authed/{symbol} .....	4
/user/register .....	5
/user/login .....	5
Modules Used .....	7
<b>ag-grid-react</b> .....	7
<b>reactstrap</b> .....	7
<b>react-chartjs-2</b> .....	7
<b>react-datepicker</b> .....	7
Application Design .....	8
Navigation and Layout .....	8
Technical Description .....	8
Architecture .....	8
Test Plan.....	10
Difficulties, Exclusions and Errors .....	11
Extensions .....	12
User Guide .....	12
Appendix A.....	15

## Introduction

### Purpose & Description



*The landing page of the application.*

Stocks Table		
Click on a stock to view price data, or filter by Industry		
Industry	Search	All 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
Automatic Data Processing	ADP	Information Technology
Alliance Data Systems	ADS	Information Technology

*The main component of the app, the Stocks Table.*

This application's main purpose and function is to allow its end users to view and analyse stock market statistics, and by extension manipulate this data through tools such as sorting, filtering, searching and charting. This app interacts with an associated database exposed through a REST API and allows users to seamlessly make requests to various endpoints, producing clean and detailed display of stock information as well as price history and charting.

This app uses a deliberate combination of modules to ensure a number of thought out features are evident throughout, such as a consistent colour theme and visually appealing elements. The layout and positioning of the delivered content is also held as a high priority, aiming to ensure an intuitive, responsive and professional feel.

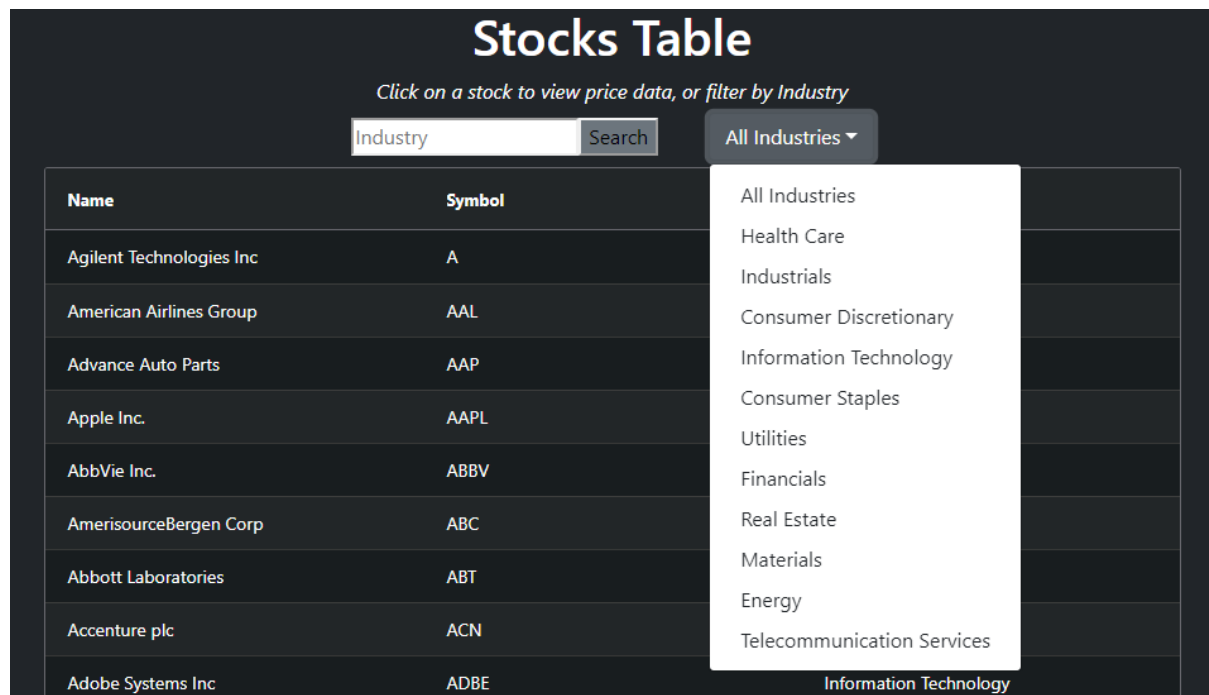
### Completeness and Limitations

It is believed that this application has been completed to a very high level, based on a number of reasons. Firstly, all features of a 6 or 7 level web application have been implemented with very little issues or shortfalls – any of which will be documented and described in detail. The favored 'bonus' feature of a chart element has been included and reacts to changes in the date range from date picking elements. These features combined with the clean and organized design make up my claim to a high standard result.

## Use of End Points

### </stocks/symbols>

This is the first endpoint used and returns all 495 companies, consisting of the stock's name, symbol and Industry. This endpoint has an optional Industry parameter where a partial or full industry string will return stocks of only companies in that Industry.

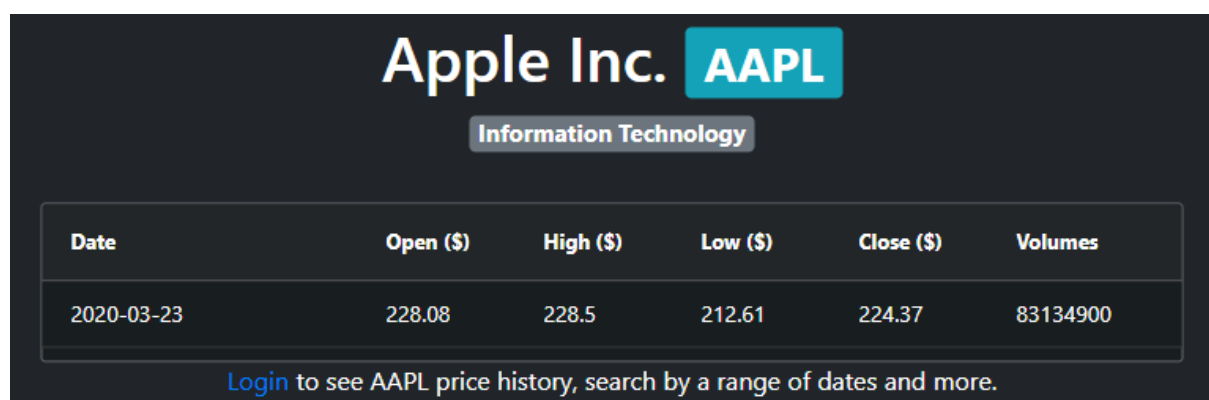


Name	Symbol	Industry
Agilent Technologies Inc	A	
American Airlines Group	AAL	
Advance Auto Parts	AAP	
Apple Inc.	AAPL	
AbbVie Inc.	ABBV	
AmerisourceBergen Corp	ABC	
Abbott Laboratories	ABT	
Accenture plc	ACN	
Adobe Systems Inc	ADBE	Information Technology

The user can either use a dropdown menu to select the industry they want to view or type the name of that industry into the search bar. Should an invalid industry be entered, the table will revert to displaying all stocks and discretely print the API's error response to the console to avoid screen clutter.

### </stocks/{symbol}>

This endpoint returns an object containing the details of the most recent entry for a particular stock, specified by its symbol.



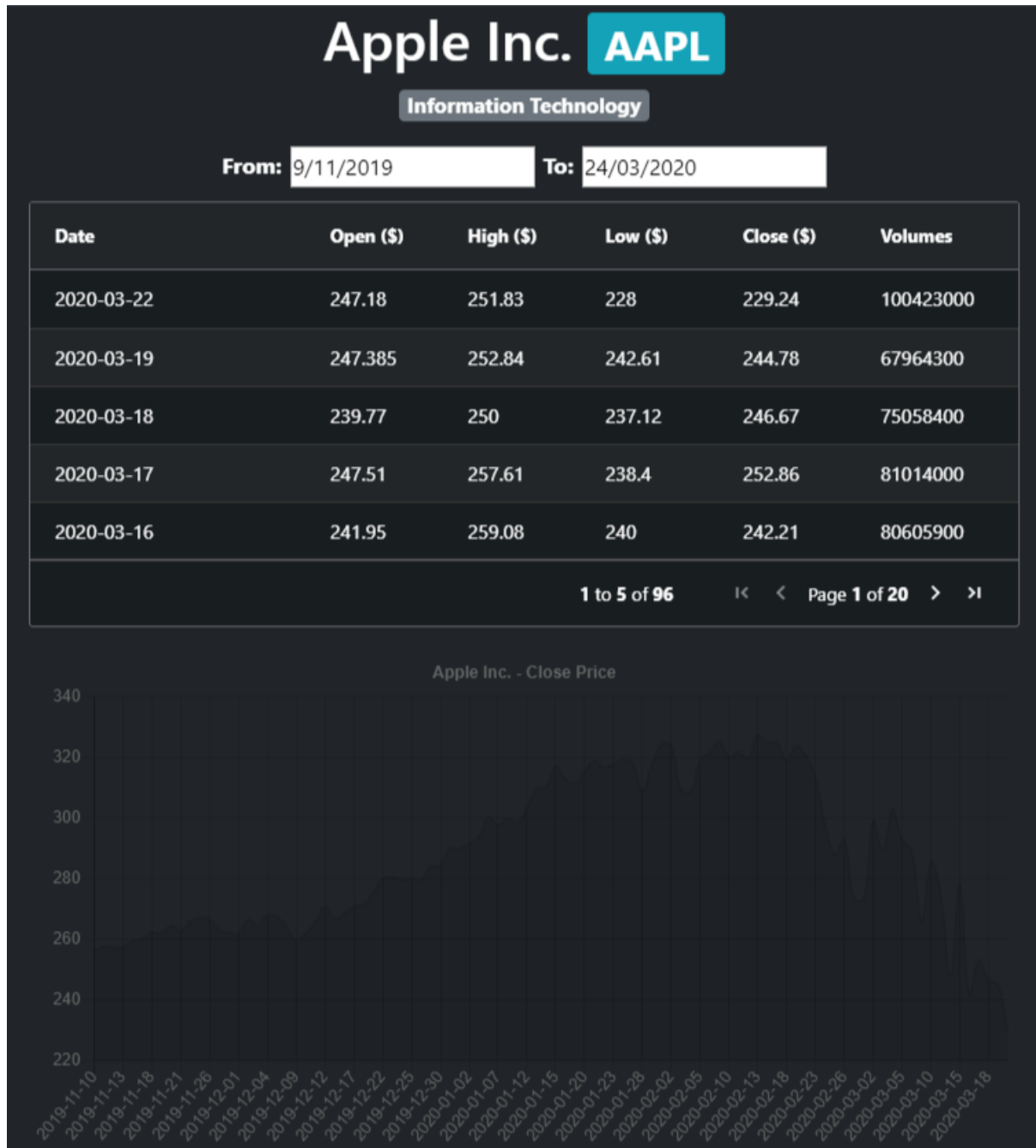
Date	Open (\$)	High (\$)	Low (\$)	Close (\$)	Volumes
2020-03-23	228.08	228.5	212.61	224.37	83134900

[Login](#) to see AAPL price history, search by a range of dates and more.

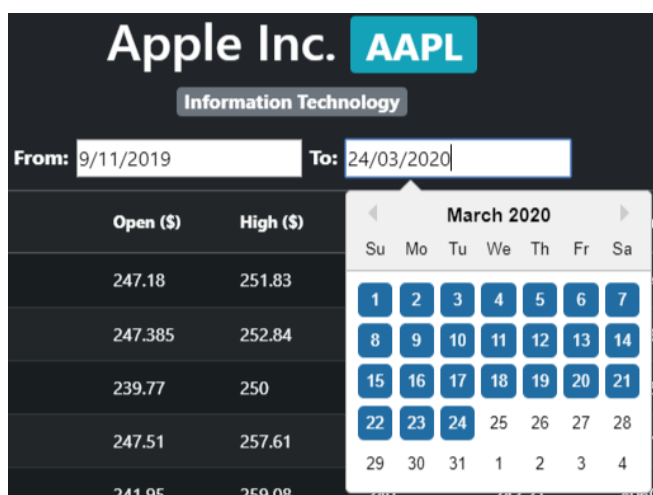
Clicking Login from the bottom message will redirect the user to the login page with a redirect URL that returns them to the stock page they were originally viewing.

/stocks/authed/{symbol}

This endpoint is only accessible once authenticated and allows the user to return data for a stock between two specified dates.



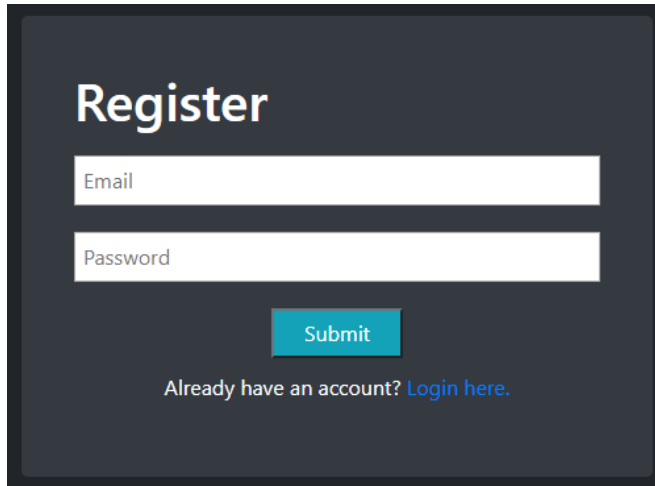
This page shows and charts all entries in the database by default and reacts dynamically to changes in the range of dates to be displayed, as specified by the two date pickers.



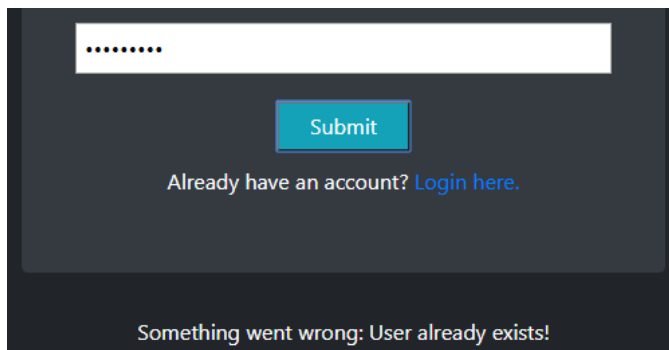
The highlighted blue dates represent the range of entries that are present in the database, as per the task specification.

</user/register>

This endpoint is used for the generation of the JWT (JSON Web Token), used as authentication for the 'authenticated' endpoint discussed previously.

A dark-themed registration form titled "Register" in white. It features two white input fields: "Email" and "Password". Below the fields is a teal "Submit" button. At the bottom, there is a link: "Already have an account? [Login here.](#)"

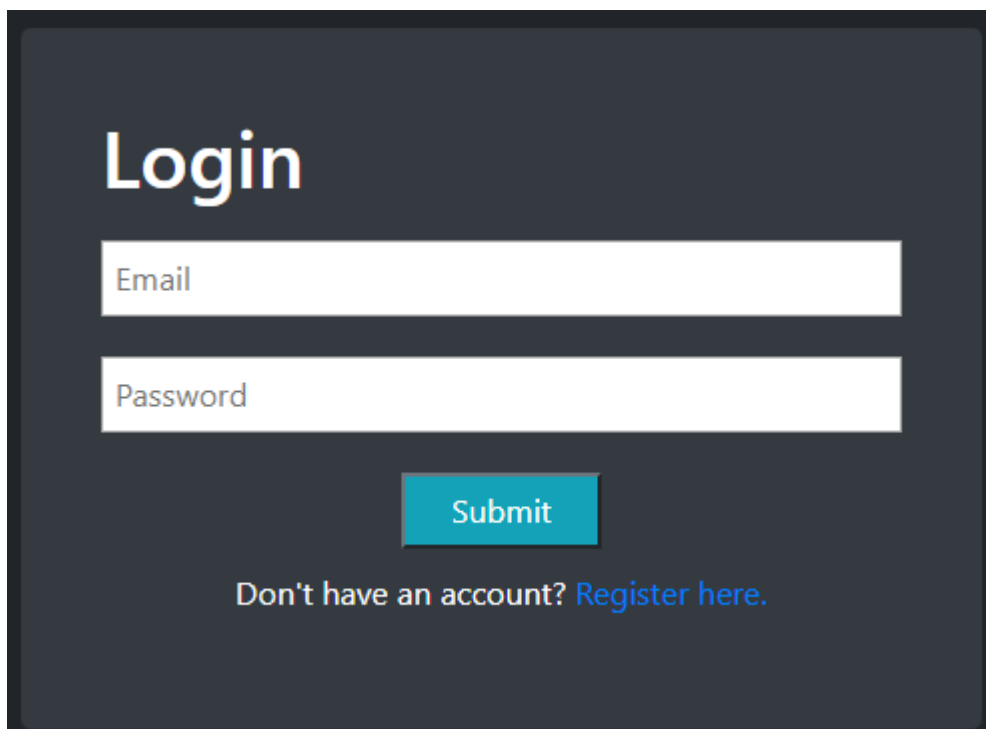
This box element is where the user enters their details which are passed through to the API in order to create their account.

A dark-themed registration form titled "Register" in white. The "Email" input field is filled with dots, indicating a password or masked text. Below the fields is a teal "Submit" button. At the bottom, there is a link: "Already have an account? [Login here.](#)" Below the form, a message is displayed: "Something went wrong: User already exists!"

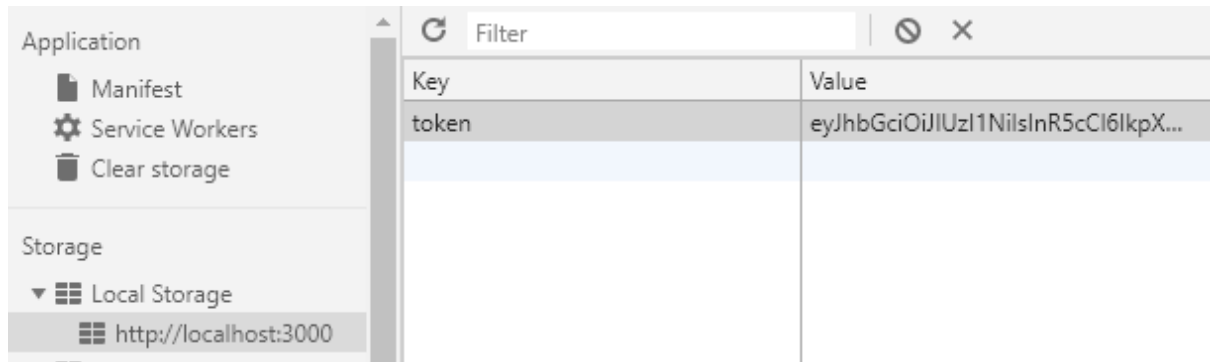
Any response that isn't successful is directly displayed on the screen for the user.

</user/login>

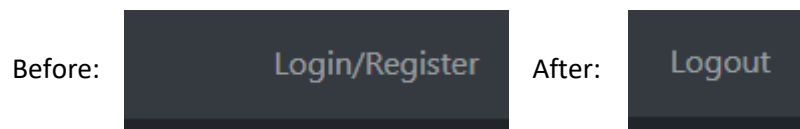
Finally, this endpoint is used to verify login details to authenticate the user with the API, comparing them with an existing JWT.

A dark-themed login form titled "Login" in white. It features two white input fields: "Email" and "Password". Below the fields is a teal "Submit" button. At the bottom, there is a link: "Don't have an account? [Register here.](#)"

Upon successful login, the user is redirected to the stock page they came from (if applicable), otherwise returning to the main table. The JWT response from the API is stored in localStorage and is accessible throughout the application for other requests.



The 'Login/Register' link in the navigation bar turns into a 'Logout' link and will log the user out and destroy the now-obsolete JWT.



## Modules Used

### *ag-grid-react*

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

[Website](#)

### *reactstrap*

Module providing Bootstrap components to JavaScript.

[GitHub](#)

### *react-chartjs-2*

React wrapper for JavaScript charting tool Chart.js 2.

[GitHub](#)

### *react-datepicker*

A simple and reusable Datepicker component for React.

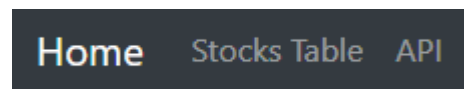
[npm](#)



## Application Design

### Navigation and Layout

The layout of the site is one that is clean and simple, with an emphasis on as little clutter as possible. The app is centered around the stocks table – that is to say that the individual stock view pages are generated solely off clicking that stock from the main table. This decision of layout was made in order to ensure the navigation and user experience was simple and required as little actual interaction as possible to achieve the desired feature of the app. All user API routes are handled neatly in the Login/Register page, which changes dynamically depending on whether the user is logging in or registering for the first time, with a link to switch between both.

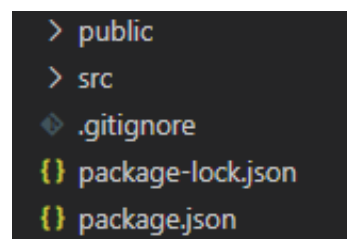


These layout choices allowed for a neat and concise navigation bar spanning the top of the site. This header is deliberately outside of the scope of the React Router system to ensure it is always visible no matter the page being displayed. At any point the main table can be navigated back to, as well as the Swagger Docs page with the API information and endpoints that opens in a new tab. I also opted to center everything in the middle of the screen to assist readability and visual appeal. I opted for a consistent dark colour scheme with contrasting blue elements throughout the app out of personal preference to give it a professional feel.

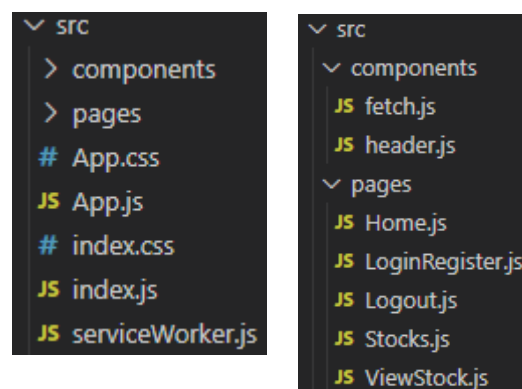
## Technical Description

### Architecture

At the source code level, the application has been organised in a concise manner with folders splitting responsibilities appropriately. Below is the top-level structure of the app files as seen in Microsoft VS Code.



Going deeper into the actual performance of the application and how the modules are organised, the following structure shows the delegation of processes.



The application uses a separate export module for every functionality, organised into pages and components. This allows for a minimal and organised main App.js file.

```
// written components
import Header from './components/header';

// pages
import Stocks from './pages/Stocks';
import Home from './pages/Home';
import ViewStock from './pages/ViewStock';
import LoginRegister from './pages/LoginRegister';
import Logout from './pages/Logout';
```

*App.js - Imports of all the modules*

```
<Router>

  <Header />

  <Switch>

    <Route exact path="/">
      <Home />
    </Route>

    <Route path="/table">
      <Stocks />
    </Route>

    <Route path="/stock">
      <ViewStock symbol={window.location.pathname} />
    </Route>

    <Route path="/login">
      <LoginRegister />
    </Route>

    <Route path="/register">
      <LoginRegister />
    </Route>

    <Route path="/logout">
      <Logout />
    </Route>

  </Switch>

</Router>
```

These modules are then called upon by a React Router system which routes the user through the website and passes the modules parameters if and when required.

## Test Plan

Test	Expected Outcome	Result	Screenshots – Appendix A
<b>Stocks Table</b>			
Sort stocks columns	Every column sortable	PASS	S1
Filter by industry using dropdown	Table displays stocks for that industry	PASS	S2
Filter by industry using search bar	Table displays stocks fully or partially matching that industry string	PASS	S3
Filter by industry using search bar – invalid industry	Error handled, logged to console, <i>display message</i>	PARTIAL	S4
<b>Stock View – not logged in</b>			
View company name, symbol, industry	Company information available	PASS	S5
View latest price and volumes details	Recent price data available	PASS	S6
Login and get redirected back to stock page	Return to the authentication version of the stock page after login	PASS	S7
<b>Stock View – logged in</b>			
View all stock price history data	Table populated with price data	PASS	S8
Sort columns	All columns sortable	PASS	S9
Choose date range with date pickers	Table updates with new price range	PASS	S10
View charting data	Line chart populated	PASS	S11
Provide invalid date range	Error handled and date ranged ignored, <i>display message</i>	PARTIAL	S12
<b>Login, Register, Logout</b>			
Create an account	New account created	PASS	S13
Log into an existing account	Account logged in to, token created	PASS	S14
Provide incorrect login details	Display API response	PASS	S15
Provide invalid registration details	Display API response	PASS	S16
Navigate between login and register	Switch between modes with link, text changes	PASS	S17
Log out of account	Token destroyed, account logged out of	PASS	S18

## Difficulties, Exclusions and Errors

During the development of this application there were a handful of difficulties and errors encountered. Most of these errors were dealt with simply enough - changing my approach.

An example of one of these roadblocks is when I was handling the submitted details from the login inputs in the Login/Register page. I attempted to throw custom errors based on the data or omission or data entered, using regular expressions to ensure a valid email address, for example. While this worked successfully it produced a plethora of console warnings and a host of other errors and eventually I admitted defeat and elected to simply pass through the error message from the API response if the input wasn't valid.

There is a functionality not **fully** represented in the application and that is full error handling. As previously stated, I have implemented a displayed message straight from the API should the input when logging in or registering be invalid, however this is not present in other areas of the site. I made an executive decision to neglect displaying an error message in the stocks table or in the 'view stock' page when an invalid industry or date range, instead catching the react error, extracting the error message from the API response and logging both to the console. I felt that this reduces the amount of information on the screen and therefore upholds the minimalistic look I was going for with this website.

There is one persistent bug present in the application. For some reason or another the introduction of the line graph from *react-chartjs2* made the contents of the stock view page slightly blurry. Upon research I discovered that this is caused by the *transform* style property that is used in my CSS file to center the contents of the page, as well as various other elements throughout the site.

```
.centered {  
  position: fixed;  
  top: 50%;  
  left: 50%;  
  transform: translate(-50%, -50%);  
}
```

I therefore had a decision to make, either hardcode the position of the page or accept the reality that the page was going to be slightly blurry. I ultimately decided that the display of the content if the Chrome window was resized rendering correctly was more important.

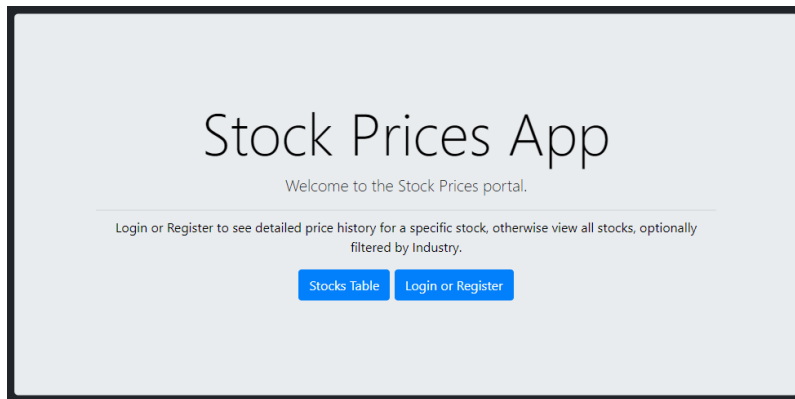
These were the only real hardships I encountered during development, and the rest of the app remains relatively issue-free.

## Extensions

This application has the potential to be extended further through the incorporation of various external APIs that give more information about the companies, or simply their own websites being linked.

## User Guide

The home jumbotron is the landing screen of the site, and from there two options are available – **Login or Register** or view the **Stocks Table**.



Should you elect to view the stocks table, you're greeted by the main component of the application initially displaying all stocks in the database.

Stocks Table		
Click on a stock to view price data, or filter by Industry		
Industry	Search	All 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
Automatic Data Processing	ADP	Information Technology
Alliance Data Systems	ADS	Information Technology

This table can be sorted in ascending or descending order by every column or filtered to only include stocks from a particular industry through the search bar or the drop-down menu.

Health Care ▾	utilites	Search	All Industries ▾
Industry	Symbol	Industry	
Health Care	AEE	Utilities	
Health Care	AEP	Utilities	
Health Care	AES	Utilities	
Health Care	AWK	Utilities	

From here, you are able to click on any stock to view details about the most recent entry of that company in the database.

<div> <div>Apple Inc.</div> <div>AAPL</div> </div>					
Information Technology					
Date	Open (\$)	High (\$)	Low (\$)	Close (\$)	Volumes
2020-03-23	228.08	228.5	212.61	224.37	83134900

[Login](#) to see AAPL price history, search by a range of dates and more.

Clicking the Login link at the bottom of this display will lead you to the Login page, also keeping note of the stock you were viewing at the time.

localhost:3000/login?redirect=AAPL

## Login

Submit

Don't have an account? [Register here.](#)

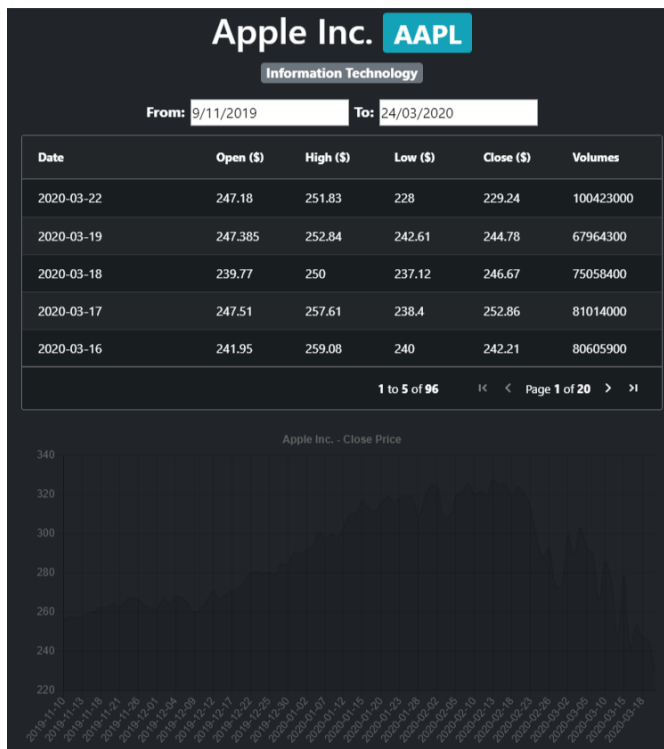
If you don't have an account already, clicking the click forwards you to the registration page where you can enter your details and create your account.

## Register

Submit

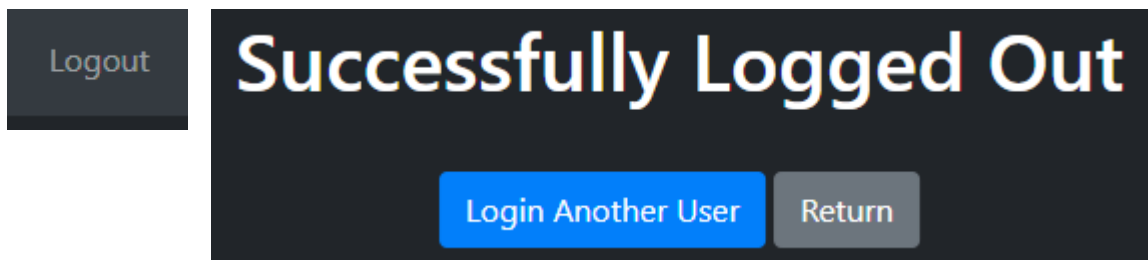
Already have an account? [Login here.](#)

After logging in to your new or existing account, the app will redirect you to the authenticated version of the stock view page you came from, if applicable.



This display allows you to choose a range of dates to display from the date pickers and updates the table and line chart as required. All columns are sortable. At any point it is possible to return to the stocks table and select another stock to view its price history in this detailed view.

Logging out is achieved by simply clicking Logout link on the navigation bar.



## Appendix A

### Test Plan Results

#### S1 - Sort stocks columns

Industry ↑
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary
Consumer Discretionary

#### S2 - Filter by industry using dropdown

Health Care ▾
Industry
Health Care
Health Care
Health Care
Health Care
Health Care
Health Care
Health Care
Health Care
Health Care

#### S3 - Filter by industry using search bar

utilites <span>×</span> Search		All Industries ▾
Symbol	Industry	
AEE	Utilities	
AEP	Utilities	
AES	Utilities	
AWK	Utilities	



#### S4 - Filter by industry using search bar – invalid industry

### Stocks Table

Click on a stock to view price data, or filter by Industry

All Industries ▾

```
[HMR] Waiting for update signal from WDS... log.js:24
Failed to load resource: the server responded with a status of 404 (Not Found) 131.181.190.87:3000/_bols?industry=aaa:1
Industry sector not found fetch.js:15
TypeError: data.map is not a function fetch.js:25
    at fetch.js:16
```

#### S5 - View company name, symbol, industry

# Apple Inc.

## AAPL

Information Technology

#### S6 - View latest price and volumes details

Date	Open (\$)	High (\$)	Low (\$)	Close (\$)	Volumes
2020-03-23	228.08	228.5	212.61	224.37	83134900

#### S7 – Login and get redirected back to stock page

localhost:3000/login?redirect=AAPL

#### S8 - View all stock price history data

Date	Open (\$)	High (\$)	Low (\$)	Close (\$)	Volumes
2020-03-22	247.18	251.83	228	229.24	100423000
2020-03-19	247.385	252.84	242.61	244.78	67964300
2020-03-18	239.77	250	237.12	246.67	75058400
2020-03-17	247.51	257.61	238.4	252.86	81014000
2020-03-16	241.95	259.08	240	242.21	80605900
1 to 5 of 96 < < Page 1 of 20 > >I					

#### S9 – Sort columns

Volumes ↓
106721000
104619000
100423000
92683000
85349300

### S10 - Choose date range with date pickers

**From:** 9/11/2019 **To:** 14/11/2019

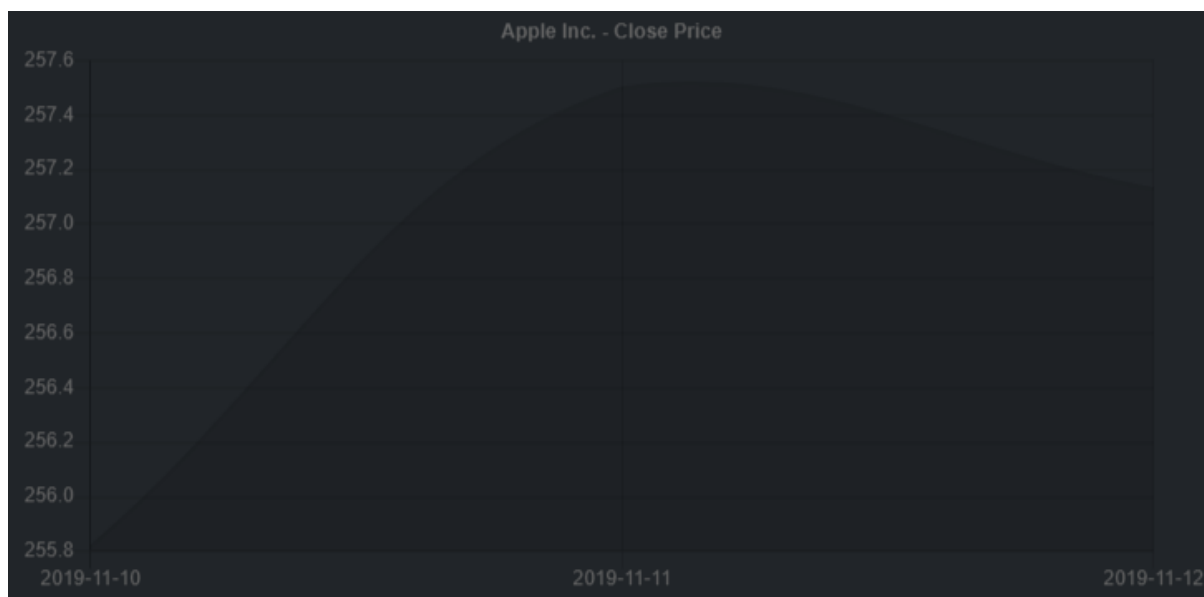
Open (\$)	High (\$)
257.05	258.19
257.33	257.845
249.54	255.93

**November 2019**  
Su Mo Tu We Th Fr Sa  
27 28 29 30 31 1 2  
3 4 5 6 7 8 9  
10 11 12 13 14 15 16  
17 18 19 20 21 22 23  
24 25 26 27 28 29 30

Date	Open (\$)	High (\$)	Low (\$)	Close (\$)	Volumes
2019-11-12	257.05	258.19	256.32	257.13	19974400
2019-11-11	257.33	257.845	255.38	257.5	25818000
2019-11-10	249.54	255.93	249.16	255.82	37781300

1 to 3 of 3 < < Page 1 of 1 > >

### S11 – View charting data



### S12 - Provide invalid date range

**Apple Inc.** AAPL

Information Technology

**From:** 3/04/2020 **To:** 14/11/2019

Failed to load resource: the server responded with a status of 404 (Not Found) 131.181.190.87:3000/-/-13T14:00:00.000Z:1

TypeError: data.map is not a function ViewStock.js:95

at ViewStock.js:83

### S13 - Create an account

# Register

Submit

Already have an account? [Login here.](#)

### S14 - Log into an existing account

You're already logged in! [Logout here.](#)

token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX...
-------	-------------------------------------

### S15 - Provide incorrect login details

Something went wrong: Incorrect password

### S16 - Provide invalid registration details

Something went wrong: User already exists!

### S17 - Navigate between login and register

Already have an account? [Login here.](#)

Don't have an account? [Register here.](#)

### S18 - Log out of account

# Successfully Logged Out

Login Another UserReturn

<div><div>Manifest</div><div>Service Workers</div><div>Clear storage</div></div> <div>Storage</div> <div><div>Local Storage</div><div>http://localhost:3000</div></div>	<table><tr><th>Key</th><th>Value</th></tr><tr><td></td><td></td></tr></table>	Key	Value		
Key	Value				