2020

CAB230 Stocks API – Client Side

CAB230

Stocks API – Client-Side Application

<Sharafat Mir/s>

<n10015957/s>

4/23/2020

Contents

This template is adapted from one created for a more elaborate application. The original author spends most of his professional life talking to clients and producing architecture and services reports. You may find this a bit more elaborate than you are used to, but it is there to help you get a better mark

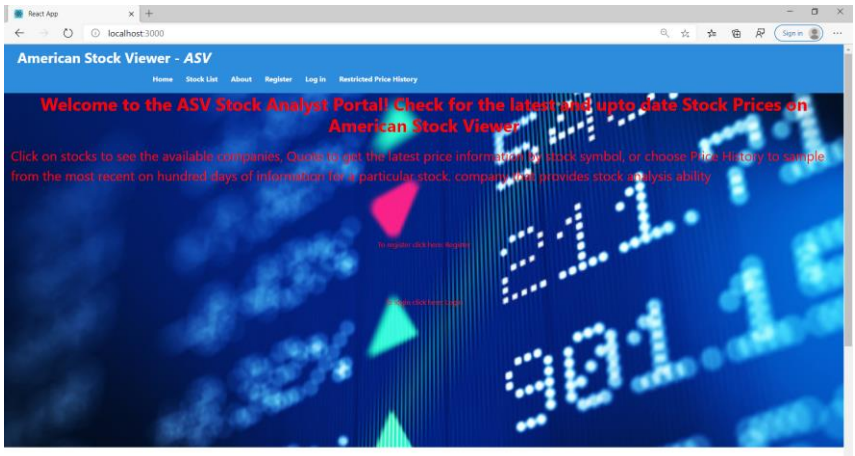
This report should be around 10 pages or so including screenshots

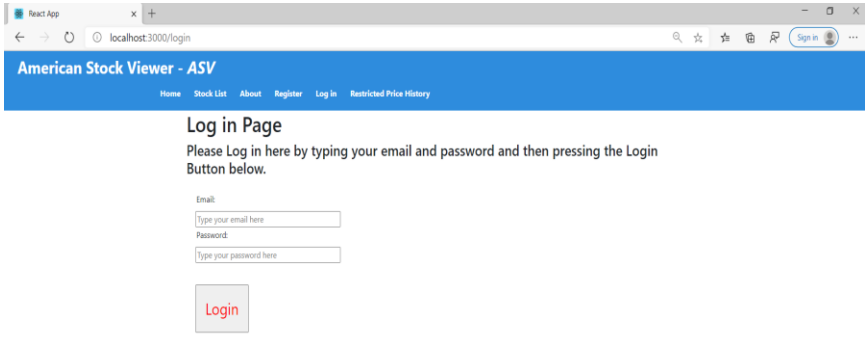
Introduction

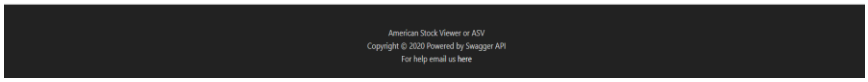
Purpose & description

This app is developed based on react web application to allow users to view and analyse stock market statistics drawn from a database via the rest API. The purpose of this web app is to allow everyone to use a sophisticated client web application using which uses modern approaches. This will people to view stocks hundreds and hundreds of different stocks and allow them to filter that information via sortable functionality and filtering functionality. This app also allows people to click on any stock of their choice and view the latest price information on that stock. Moreover, the app allows people to register, and login. The app handles errors in case some do not enter their password, or incorrect email, or if they're not registered within the system appropriate error messages will be displayed.

I have not done any other functionality that is beyond the expectation of this assessment. I have not used any special set of libraries that were not recommended by the CAB230 teaching unit. I have used the following libraries and packages: AgGridReact, React, ReactDOM, React Routers, Bootstrap for styling, reactstrap for styling, and moment for date etc. Some of the photos of the application.







Completeness and Limitations

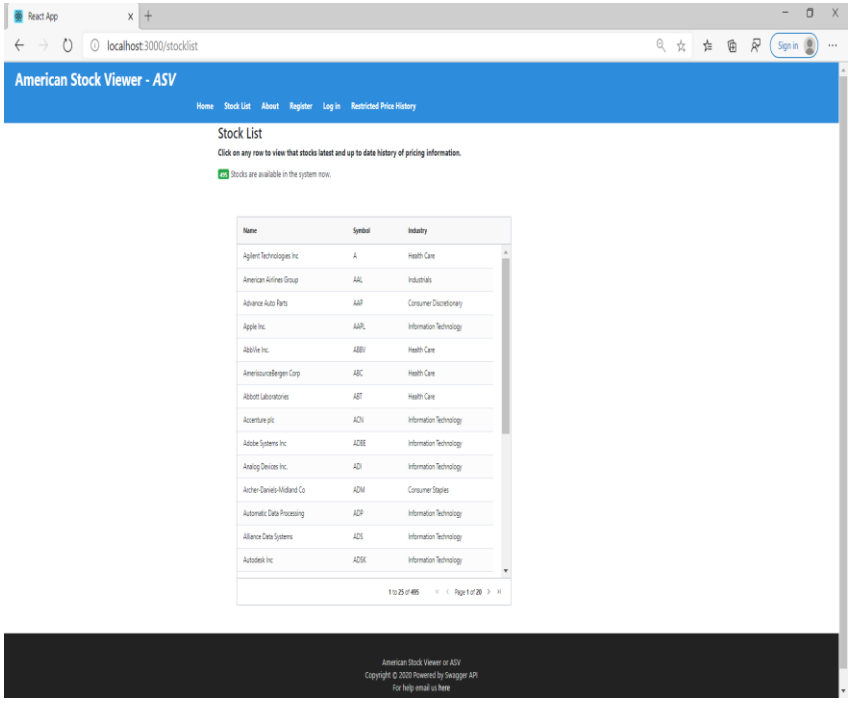
All of the functionality as per the Assignment CRA were not fully done due to faced errors in code and was not able to get help due time limitations. The following functionality worked: login process, and login error handling, register process and it's errors handling, retrieve stock name, symbol, and industry and it's errors handling, stock latest information query is retrieved and it's errors handling, navigation between pages, and navigation bar were all worked and were fully completed. What didn't work is are as follows: the chart.js to display all the data in a chart, the ability to retrieve authenticated stocks by date.

Use of End Points

In this section we want you to show us the facilities that you have provided in the app. Here you should ***organize the discussion around the endpoints of the API, showing the screen corresponding to that endpoint and providing a brief discussion of what it doe***s. (A couple of sentences is fine here – the screen shot tells the story. Write more if there is something you want to tell us. But otherwise just keep it short.)

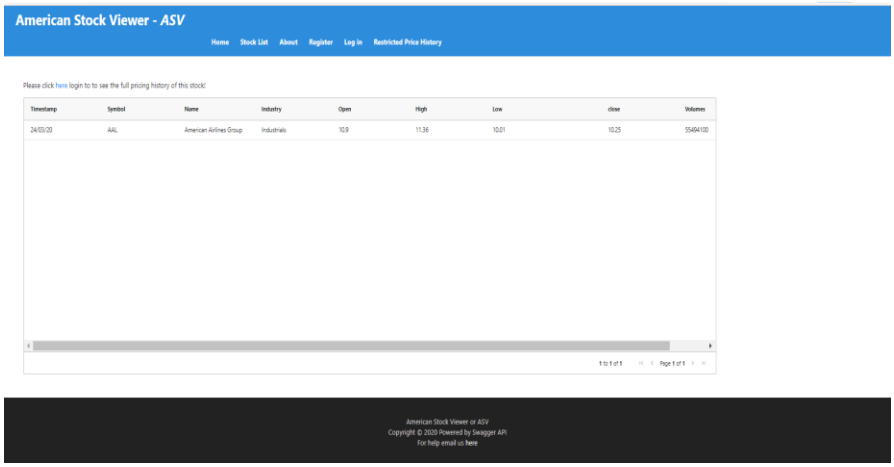
/stocks/symbols

This endpoint returns stocks information such as name, symbol, and industry. So, if you're logged in not and you click on the stock list button you will be taken to this page where you will see the returned results of this query.



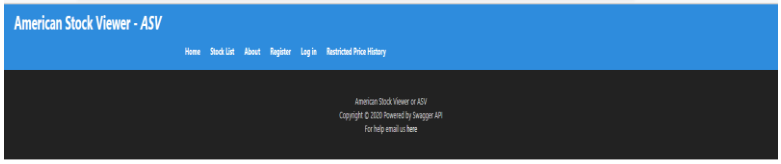
/stocks/{symbol}

This endpoint returns stocks information such as name, symbol, industry, open, high, low, close, volumes, timestamp, . So, when you're logged in or not and you click on the stock list button you will be taken to the stock information table where you will be able to click on any stock and it will hit this endpoint and return information.



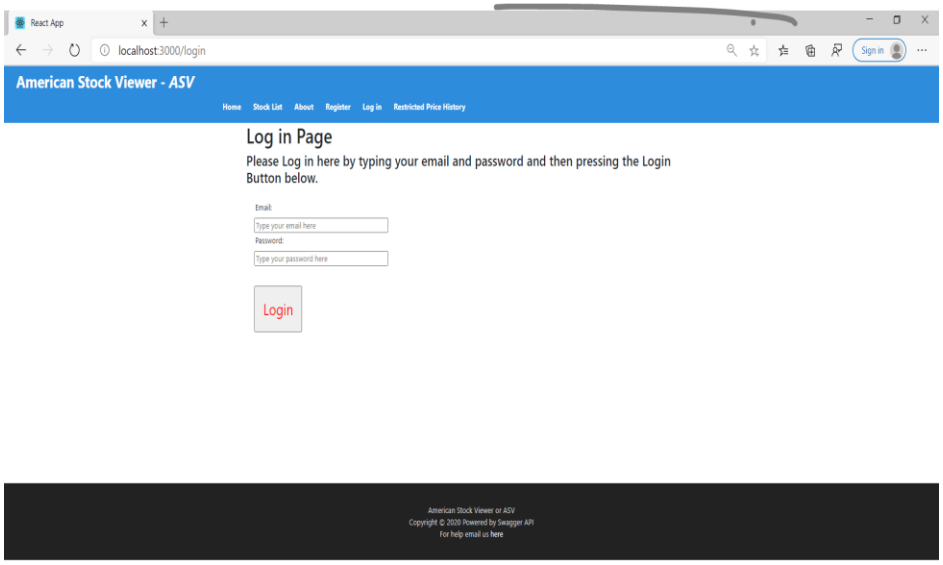
/stocks/authed/{symbol}

This endpoint returns stocks information from a certain date to a certain date such as name, symbol, industry, open, high, low, close, volumes, timestamp, for over hundreds of records. So, when you're logged in and you click on the stock list button you will be taken to the stock information table where you will be able to click on any stock and you will be asked to put a from date and a to date to return the queries for you. If there error the error is handled. I have implemented the functionality but for some reasons it doesn't work or it stopped working.



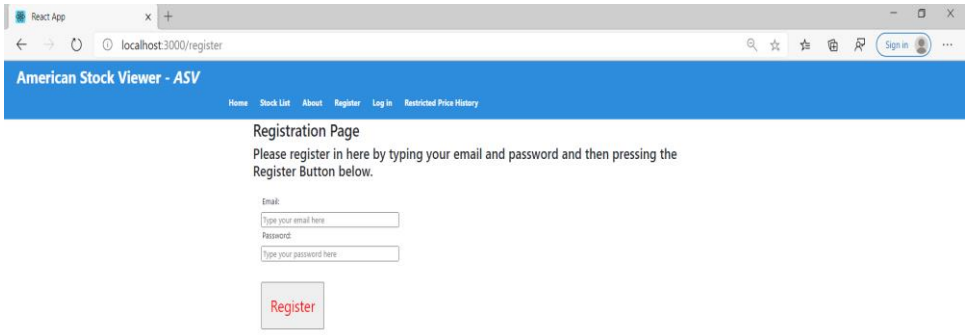
/user/register

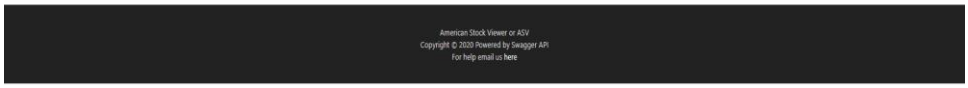
This endpoint check for login information such as email and password, when the user enters the password and email then those details are checked with the api using a post request. If it is correct it returns an appropriate response. If not, it will display appropriate error messages.



/user/login

This endpoint register users and check for users already registered using email and password, when the user enters the password and email then those details are checked with the api using a post request. If it is correct it returns an appropriate response. If not, it will display appropriate error messages.





Modules and packages used

Ag-grid-react

Module to provide fully featured table components, including sorting and filtering ability. AgGridReact is a very good module which provides good functionality but with less code.

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

React

React is a java script library for building user interfaces and allow you to design simple views for each state and has the ability to render components.

https://reactjs.org

React routers

This package or module provides the main functionality for react routers.

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

ReactDOM

React is a java script library for building user interfaces and allow you to design simple views for each state and has the ability to render components.

https://reactjs.org/docs/react-dom.html

Boostrap

Bootsrap is a powerful front-end framework for faster and easier web development and designing and styling.

https://getbootstrap.com/

Reactstrap

React is a java script library for building user interfaces and allow you to design simple views for each stat.

Moment

Moment allows formatting of date and time.

https://momentjs.com/docs/

Npm

Npm is node package manager and it can be used for JavaScript programing.

https://www.npmjs.com/

Jsonweb token

Json web token is internet standard for making json web access keys or tokens for information over the network.

https://jwt.io/introduction/

Application Design

Navigation and Layout

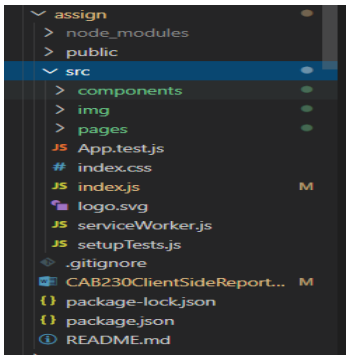
The design process for this application involved some sketching in a power point. After the sketching were done, I then tried to experiment with those sketches and made the application better from my feedback on the application design. The application navigation and layout are displayed in the header of the home page. There are buttons each button represent at the appropriate page. When you click on, any button it will take you to that page. The choice for this type of layout design was made after some research and looking at various type of good and bad websites examples. When the program compiles the UI opens and you're taken to the home page. There you're you will be able to see the differen buttons. Each button will take you to the appropriate page. Each button has a descriptive name to it to help the users know what button they're pressing before they press any buttons.

The below picture shows the sketches that were done using the Power point. These sketches were done, and experiment was done with these to achieve the aim of creating a user-friendly UI. This high-Fidelity prototype was done, and it enabled me to create a better user interface. I made certain changes during the process such as creating the navigation system in the top of the screen. This experimentation with the high-fidelity prototype allowed me to build a logically laid out application and the buttons widgets for navigations which suits the data entry and what they control.

Technical Description

Architecture

Briefly describe the overall architecture of your application at a source code level. The description above tells us something of the application's use. Now we want to see how that maps to the code organization. Your application source code will be organized something like this: The architecture of the application is organized and laid out professionally. There is folder called assign which is the project inside it as a folder called client. Inside the client folder there is a source code folder, the json packages, the gitignore file, and the public folder. Inside the source folder there are two folder one which is named component and one which is named pages and inside source there is the index js file and other css files which are relevant to the project styling part. A snippet of the code can be seen in the picture below.



The application is controlled from the source folder. The final </app> function is called here in this directory. I have chosen this method of structuring my directory as it is a good way of doing it based.

Overall, including the images, this section so far should still be under a page. We should get some sense of how the application works and how the data and control flow around. You may also find it helpful to show us screen grabs of code if that makes your points clearer. Tell us anything you think we need to know about how you have structured the application and made it work, but there also a section below to describe problems.

The component folder has three files are; Footer.js, Header.js, and Nav.js. These files contain the code for the navigation on the home screen between buttons. When each button is pressed it takes you to an appropriate page and I have react routing for that. As the code below shows.

```
<li>
  <Link key="stocklist" to="/stocklist"><b>Stock List</b>
</li>
```

The Header.js and Footer.js contains the code for the applications header and footer functionality. The img folder has a hero image that I have used in my assignment. The pages directory has the About.js file this file has the code for the about page and its functioning. The Authed.js file has code which displays an agGridReact table with data from an authenticated get request form the api. Stocklist js file show the agGridReact table with the data from a stock's directory from the api. The Stock js file also have a get request getting latest information for the user about the stock they have clicked on in page stocklist. There is also the login and register js file these files have the code for registration and login process. There is other css and other files such as json package, gitignore files related to the project inside source. There is also index js file which is calling the main for the entire code. All files have codes for error handling.

Test plan

For the manual testing please see the user's guide there are a lot of screenshots for a lot of different scenarios. All the test that are done here can be seen in the user's guide with screen shots tailor to each of these scenarios.

Task	Expected Outcome	Result	Screenshot/s (Appendix B)
Error Handling	Error Handled	PASS	3
Click arrow left *& right	Different pages can be visitted	PASS	1
Login Error	Login Error handled	PASS	1
Register	Register error handled	PASS	1
Stock retrieved	Stocks are successfully retrieved	PASS	1
Stock by date	from to date by stocks failed	FAIL	1
Search stocks	Stocks can be searched	PASS	1
Sort Stocks	Stocks can be sorted	PASS	1
Stocks by symbol	Return stock by symbol	PASS	1
After login go to restricted page	Goes to restricted page	PASS	1
Navigation	Goes from one tab to another	PASS	1
Chartjs	Displaying results in chart	FAIL	

Difficulties / Exclusions / unresolved & persistent errors /

I was not able to get the authenticated get requests working. I have implemented the code for it, but it doesn't work and was giving me errors. This didn't allow me to create the js chart as I wasn't able to get that data for the chart, and the get request wasn't working I don't know why. The file that has this code is named Authed.js. The server-side search functionality is implemented but doesn't work I don't know why that is.

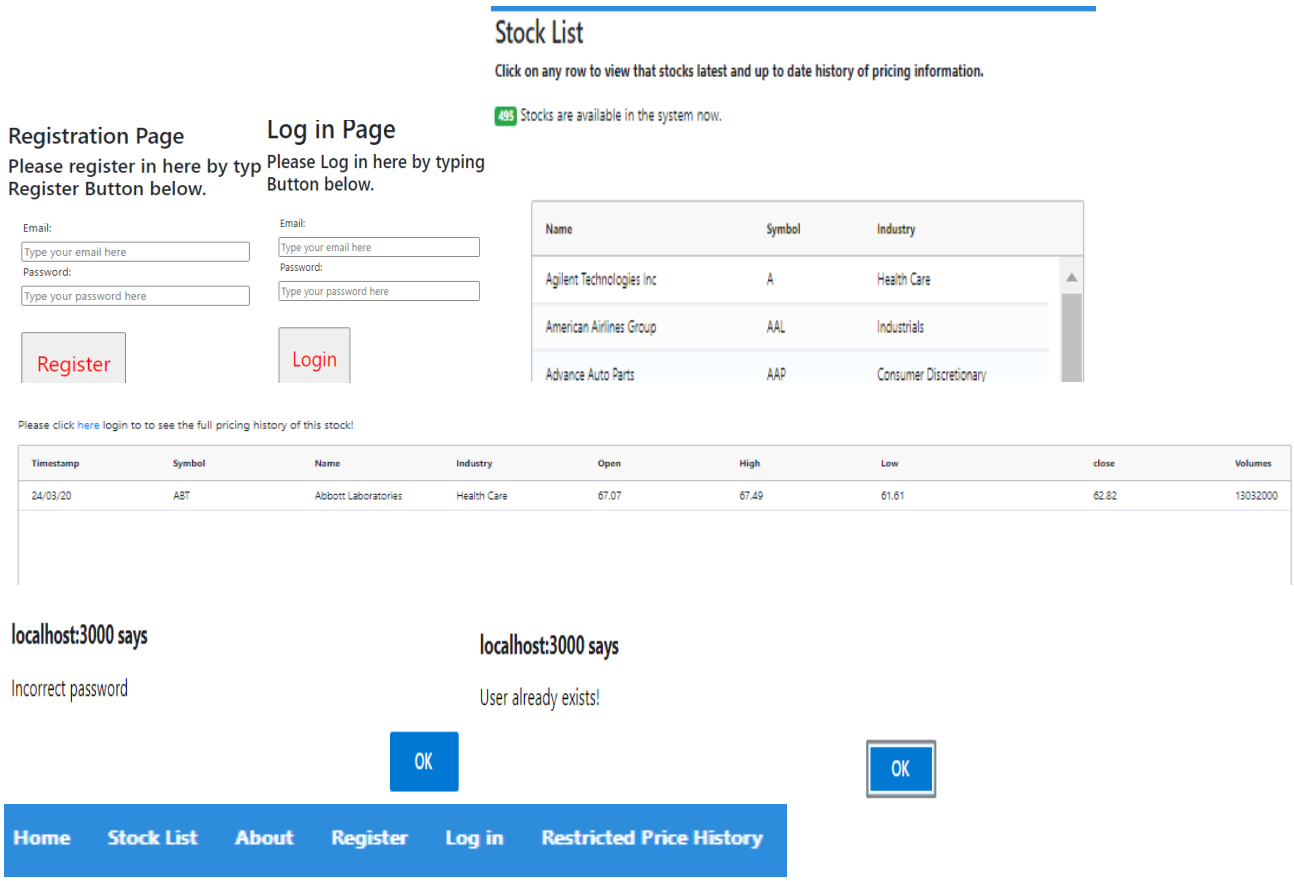
User guide

Download the .zip file from the api, then install all packages by running npm install inside project client file. After all the packages are installed run application by npm start. Then you're take to the web browser and application is running on your local host on port 3000. You will be landed in the below home page as can be seen in the picture.

Step1: To register click on register button, enter your password and email press enter. Then you're taken to the login page.

Step2: Enter your password to login.

Step3: You're taken to the authenticated page. Step4: if you don't want to login you can still use the application, you're only shown non-authenticated parts of the apps. So, click on stock list to view some stock for free. Step 4: to view more the latest information about the stock click on it. Then you're taken to the stock where it has more information about that specific stock.



References

Not any specific references were made to anything. But have used various resources for this assignment and some of the links are mentioned below.

https://jwt.io/introduction/

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

https://reactjs.org

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

https://reactjs.org/docs/react-dom.html

https://getbootstrap.com/

https://momentjs.com/docs/

https://www.npmjs.com/