2020

CAB230 Stocks API – Client Side

CAB230

Stocks API – Client Side Application

<student name/s>

<student number/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

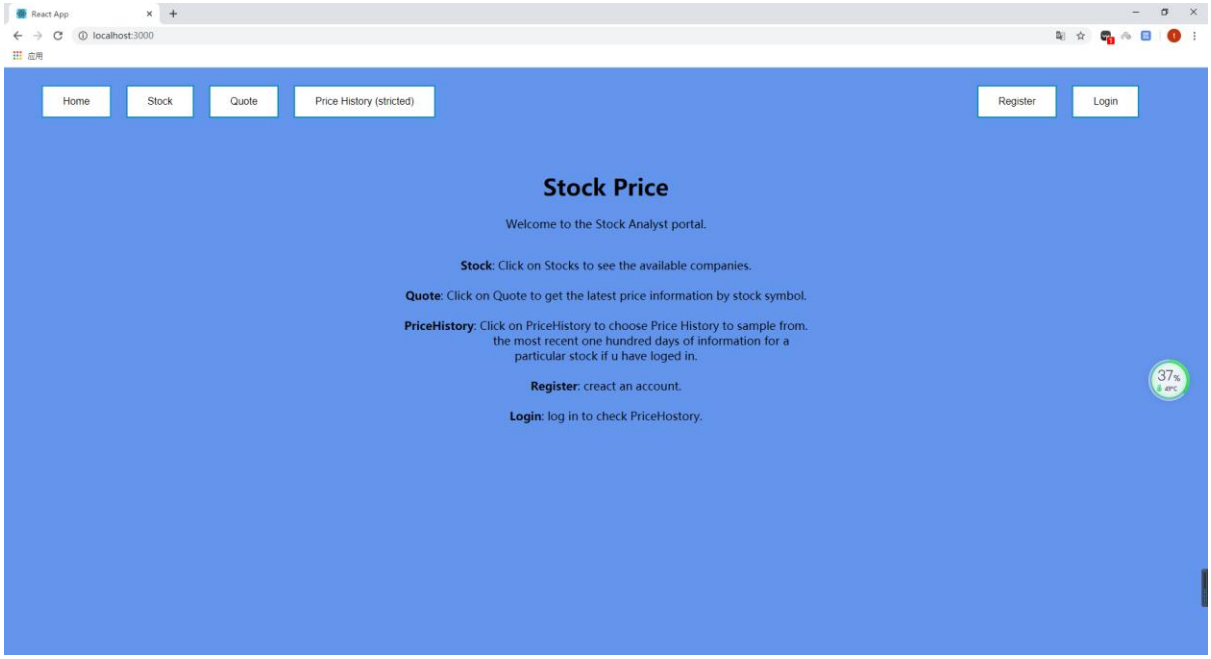
This report should be around 10 pages or so including screenshots

Introduction

Purpose & description

This is written in a high-level professional tone. Tell is in about a paragraph or so what the app is supposed to do. This should be in ***your*** words, though you should feel absolutely free to steal some of it from the assignment specification.

In my react App, I follow the requirement of assignment to basically achieve a stock system. Firstly, in my home page, there is a paragraph (copy from assignment specification) explain that the usage for every button. Please see below screenshot. As you can see, buttons on my upper left corner is look over stock price by different category (price history only check it after login), which is GET section in the assignment specification (basically finished all GET section). Buttons on my upper right corner is registering account or logging their own account (after they login they could check price history)., which is POST section in the assignment specification.



In a second (and maybe third) paragraph, go ahead and tell us what to look for in your app: ***What did you do that was different? Did you do something to provide the user with functionality beyond what was expected? Is there some special set of modules that you have used that make it look great? Is there some other module that you have used that makes it more efficient?*** At this point, this description is at a very high level still. You will list your modules below.

At this point you can show 1-2 basic screenshots of your application to illustrate the approach, but leave the more detailed screenshotting to the use cases below.

I have one difference with assignment requirement, which on my query section ("/query"). I add line chart on my query section. It will show that latest price following by 'open', 'high', 'low' and 'close' (objects). Please see photo 2. Now, I type on my search Bar to search one of company name by symbol, which is 'AAL' (please see photo 3 to see how is look like). In addition, I add all dependencies following assignment specification requirements and other resource from blackboard. (please see photo 4).

Photo 2

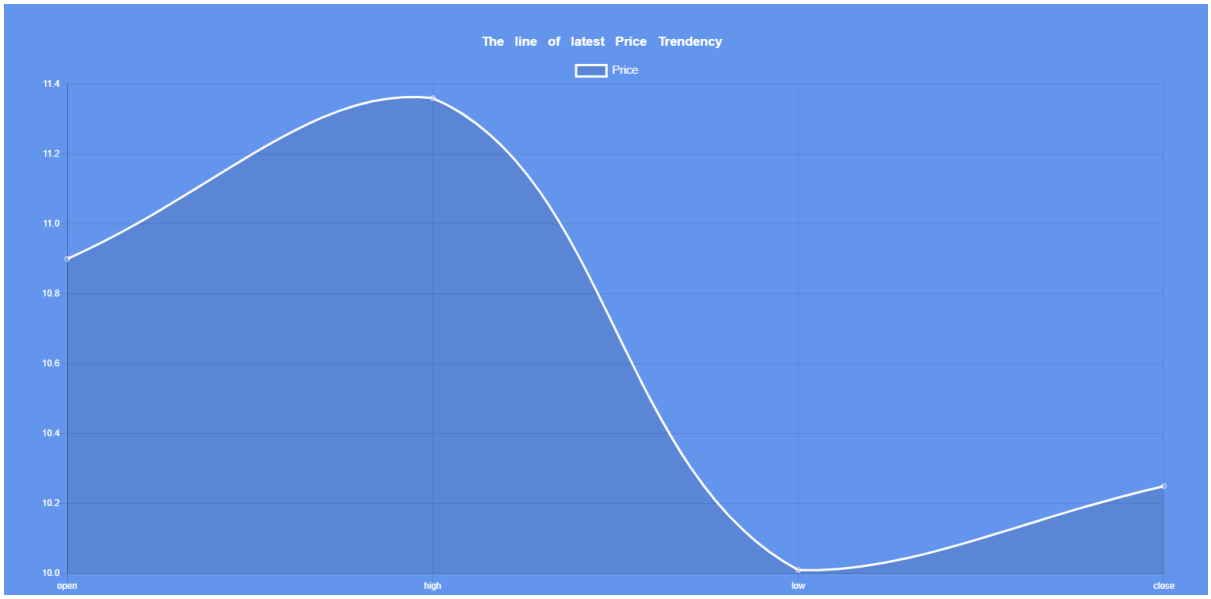


Photo 3

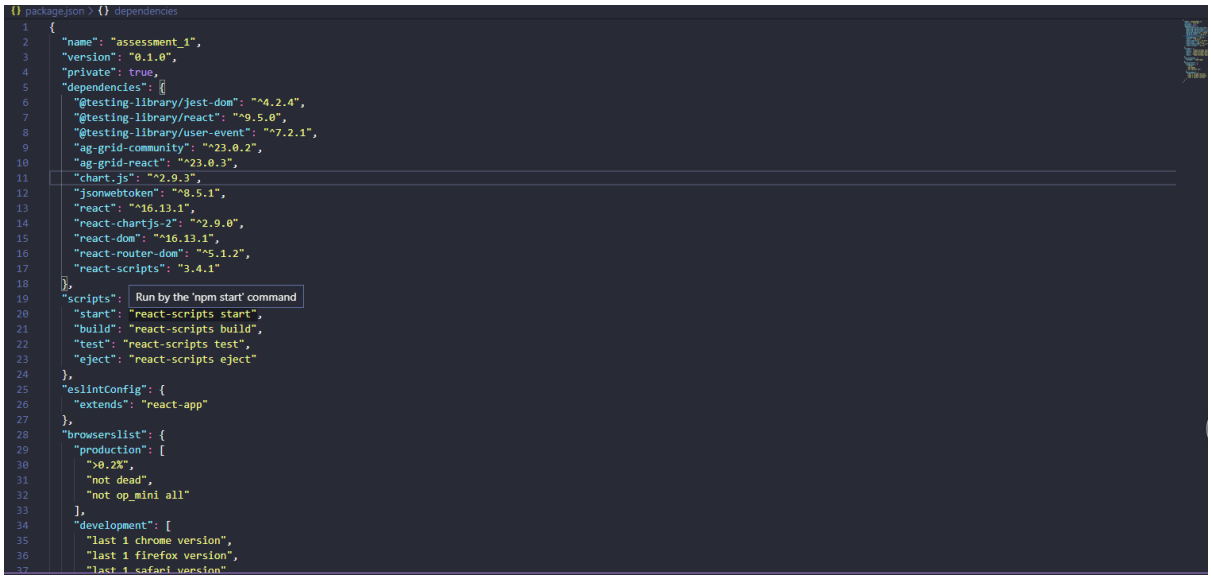


Photo 4

Completeness and Limitations

Here we want you to tell us in a couple of sentences what works and what doesn't. ***Make a claim against the standards we laid out in the assignment specification (see below) and briefly justify that claim.*** Don't give us deep details of the bugs here. Putting a positive spin on what you have achieved is fine – by all means focus on the stuff that works. But be realistic in your claim.

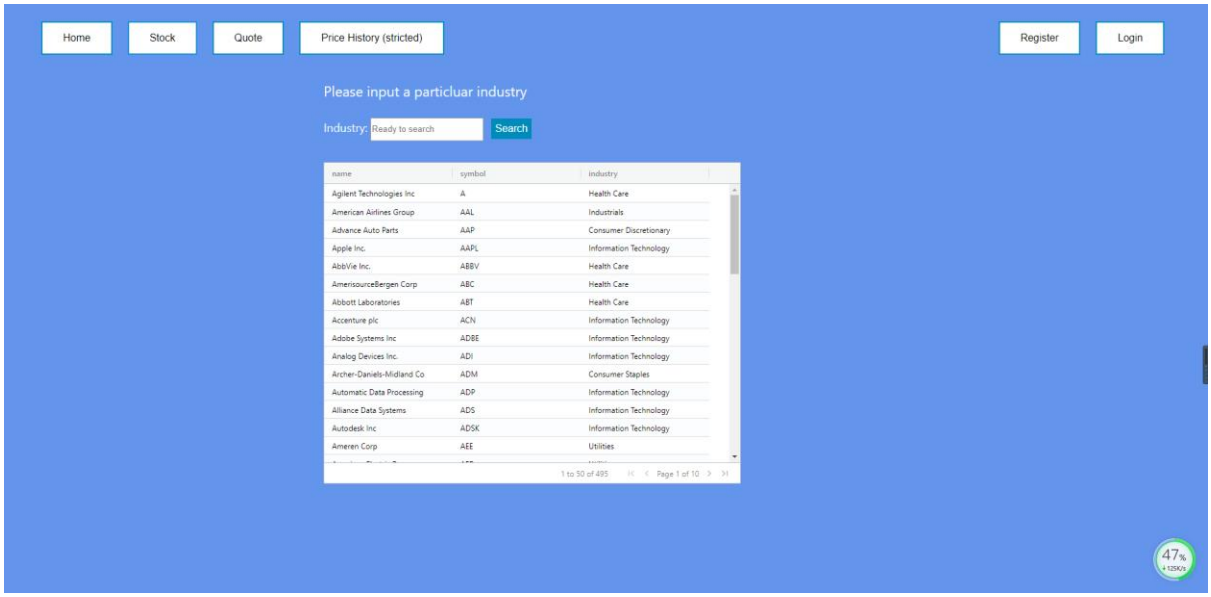
Basically, I finished all the functions, I could fetch data from res, then return the value into agrid-table and chart-line. In addition, I could type in search Bar, and then it will return exactly what I am looking for. Moreover, I could successfully login account form POST, also you could register account by yourself, it is allowed. Howevert, in my css file, I didn't make it well. I usually write my css file by className. Therefore, it will make my code more redundancy. More detail will show below using several screenshots and texts.

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

In the first GET, it links to my stock page ('/stock'). When I click 'stock' button, please see picture below.

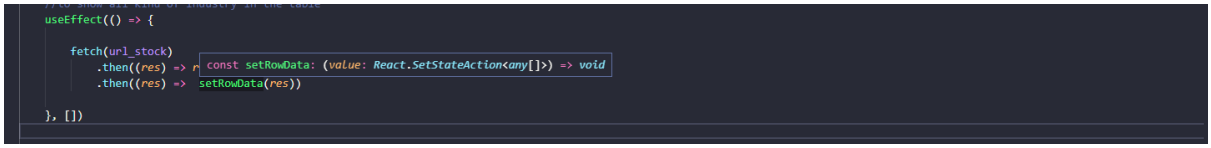


As you could see, it directly links to '/stock'. In stock page, there is a heading describe that user have to type specific industry name in the search Bar. it will show all stocks in the table, because I use useEffect to make that happen, which means that every time I npm start this project. it will fetch value from res, and then return it on table. Please see below code.

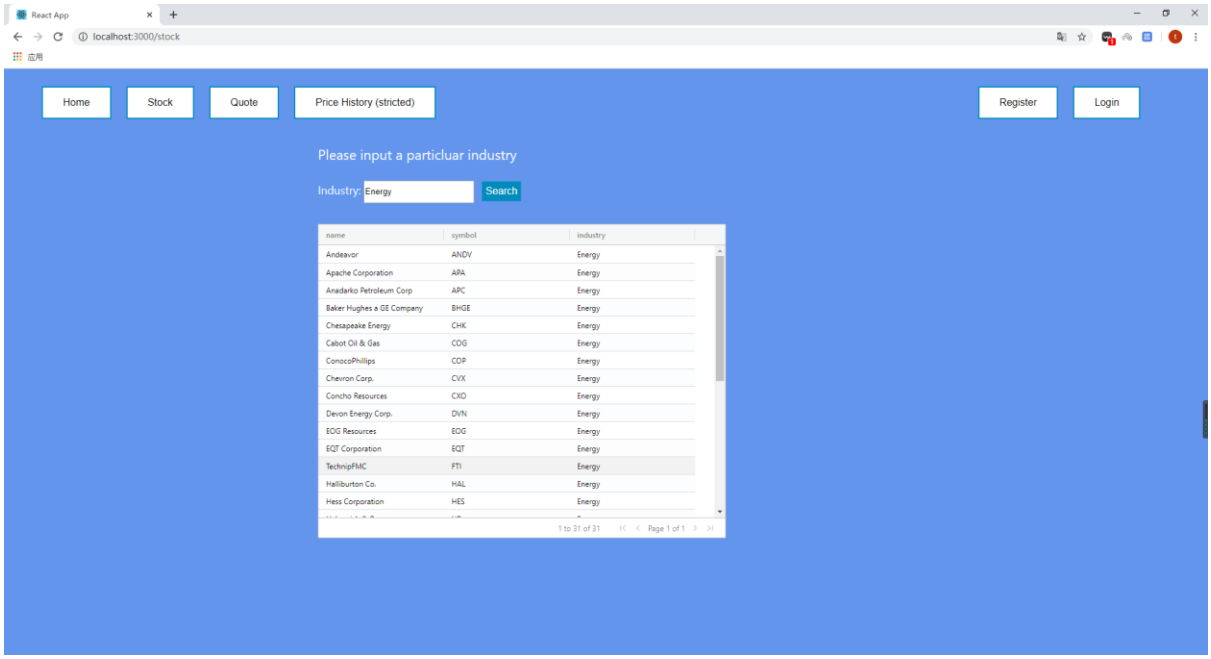
This is my first time to fetch values in the database. I used 'map' return each value.

```
//to show all kind of industry in the table
useEffect (() => {

    fetch(url_stock)
        .then((res) => res.json())
        .then((res) => {
            return res.map(data =>{
                return{
                    name: data.name,
                    symbol: data.symbol,
                    industry: data.industry,
                }
            })
        })

        .then ((res) => setRowData(res))
}, [])
```

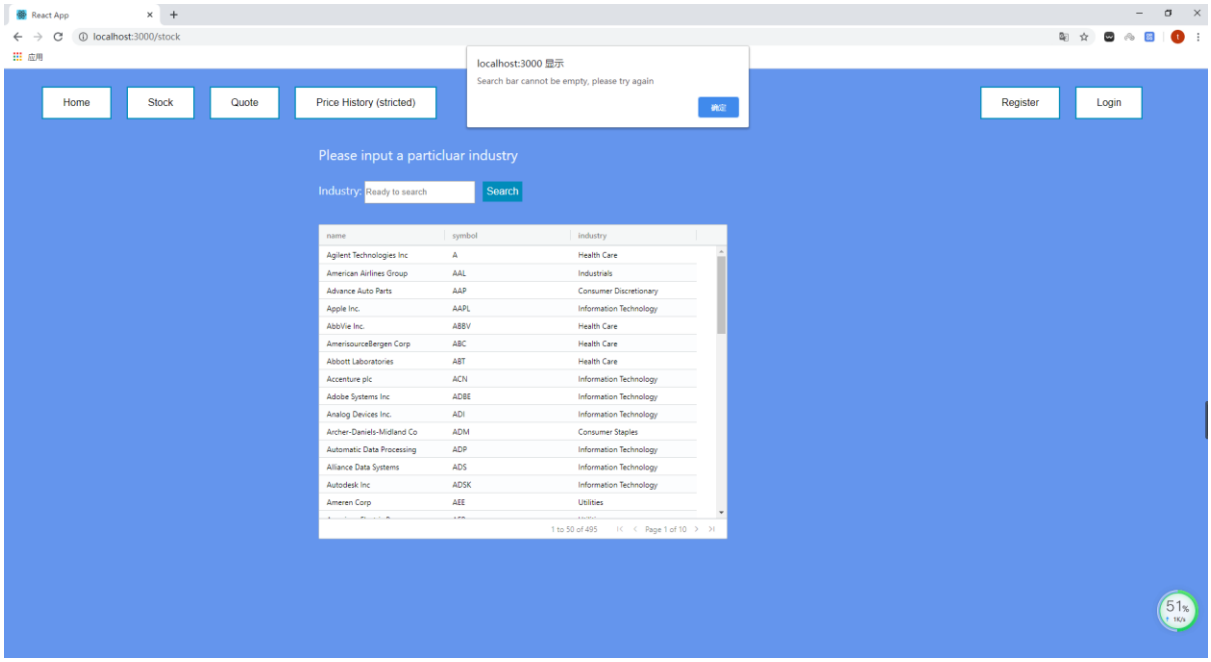
Now, I realized that I don't have to use 'map' return my value. I can just setRowData from res. See below. I just need to judge what type res returns.



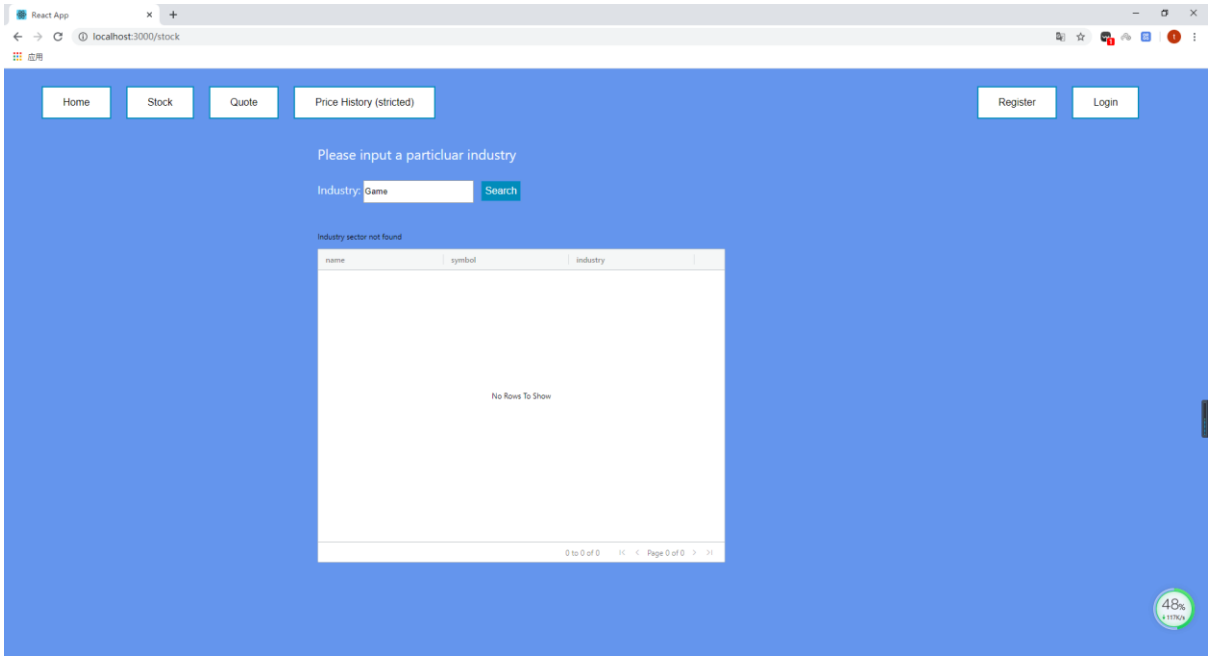
Now, moving on search Bar, I am going to type one of industry name in the search bar to see what is happen.



Now, you could see, I type ‘Energy’ (one of industry name), it returns value that I search for. Now, I am going to clear the search bar and then click the button to see what is happen.

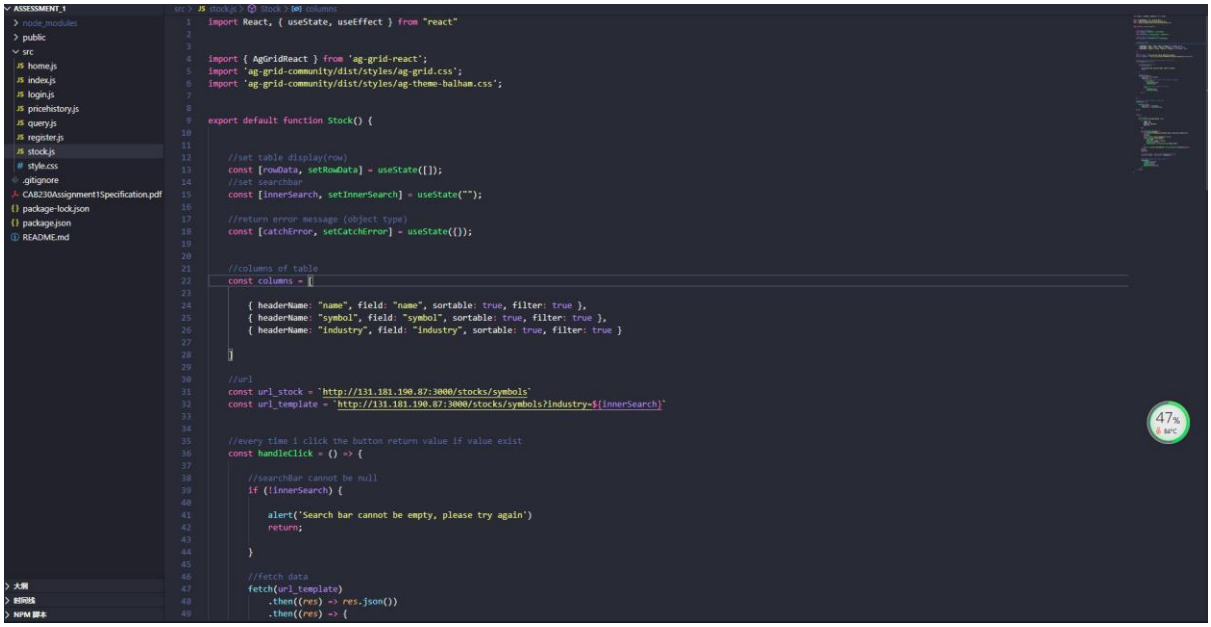


As you can see, I click the button without any character, it will involve a warning to tell us that search bar must have a value inside it. Otherwise involve alert. Now, I am going to type a random industry name which industry doesn’t exist in the res.



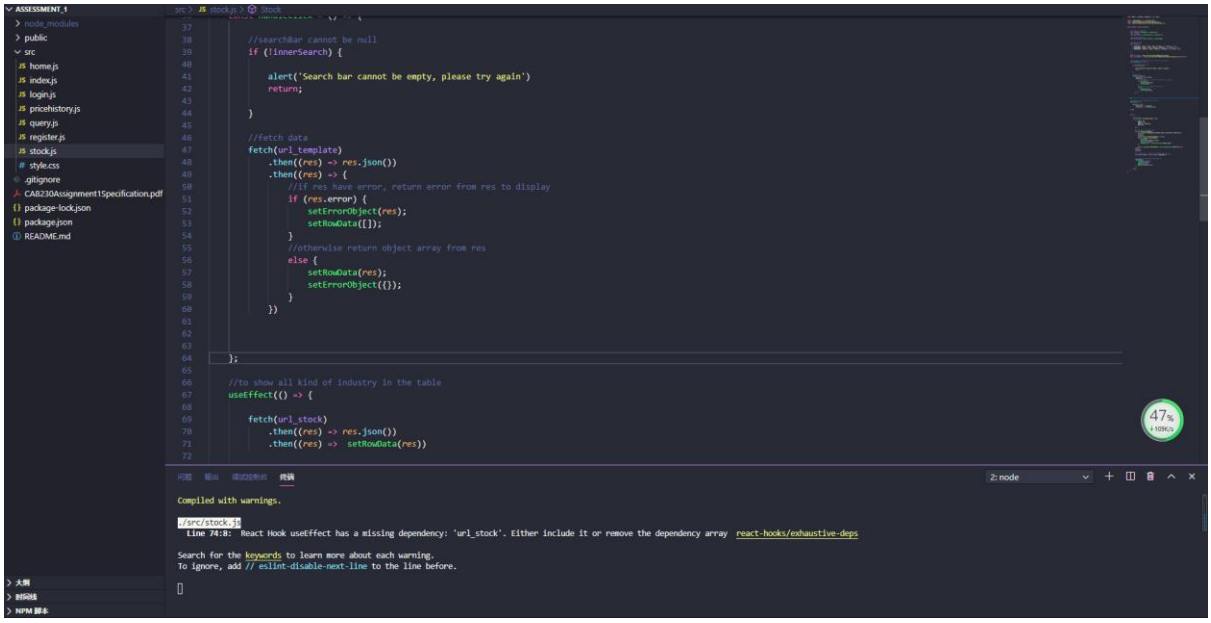
We could see, I type game inside it, it returns no data on the table. In addition, on the left upper corner of table, it displays that 'Industry sector not found' (return error message from res). Now, I screenshot my stock below.

Below picture, there are several declarations of useState, rowdata is putting value into table (array object). innerSearch is detect what I type in search Bar. errorObject is that if res doesn't have this industry name return error message from res on the left upper corner of table. (Please see above picture). And then, columns are table of columns. url_stock is fetching all stock data and return on the table. url_template is fetching data by innerSearch, which means that it depends on what I type on the search Bar. Moving handleClick section, handleclick is function depend on button, which means every time I click the button, it will run handleClick function. I basically learn how to fetch data depend on button from this video. A herf = https://www.youtube.com/watch?v=1tfd6ANaNRY



```
import React, { useState, useEffect } from "react"


import { AgGridReact } from 'ag-grid-react';
import 'ag-grid-community/dist/styles/ag-grid.css';
import 'ag-grid-community/dist/styles/ag-theme-balham.css';


export default function Stock() {


    //set table display(row)
    const [rowData, setRowData] = useState([]);
    //set searchbar
    const [innerSearch, setInnerSearch] = useState("");

    //return error message (object type)
    const [catchError, setCatchError] = useState({});


    //columns of table
    const columns = [

        { headerName: "name", field: "name", sortable: true, filter: true },
        { headerName: "symbol", field: "symbol", sortable: true, filter: true },
        { headerName: "industry", field: "industry", sortable: true, filter: true }

    ]


    //url
    const url_stock = `http://131.181.190.87:3000/stocks/symbols`
    const url_template = `http://131.181.190.87:3000/stocks/symbols?industry=${innerSearch}`


    //every time I click the button return value if value exist
    const handleClick = () => {

        //searchBar cannot be null
        if (!innerSearch) {

            alert('Search bar cannot be empty, please try again')
            return;

        }

        //fetch data
        fetch(url_template)
            .then((res) => res.json())
            .then((res) => {
```

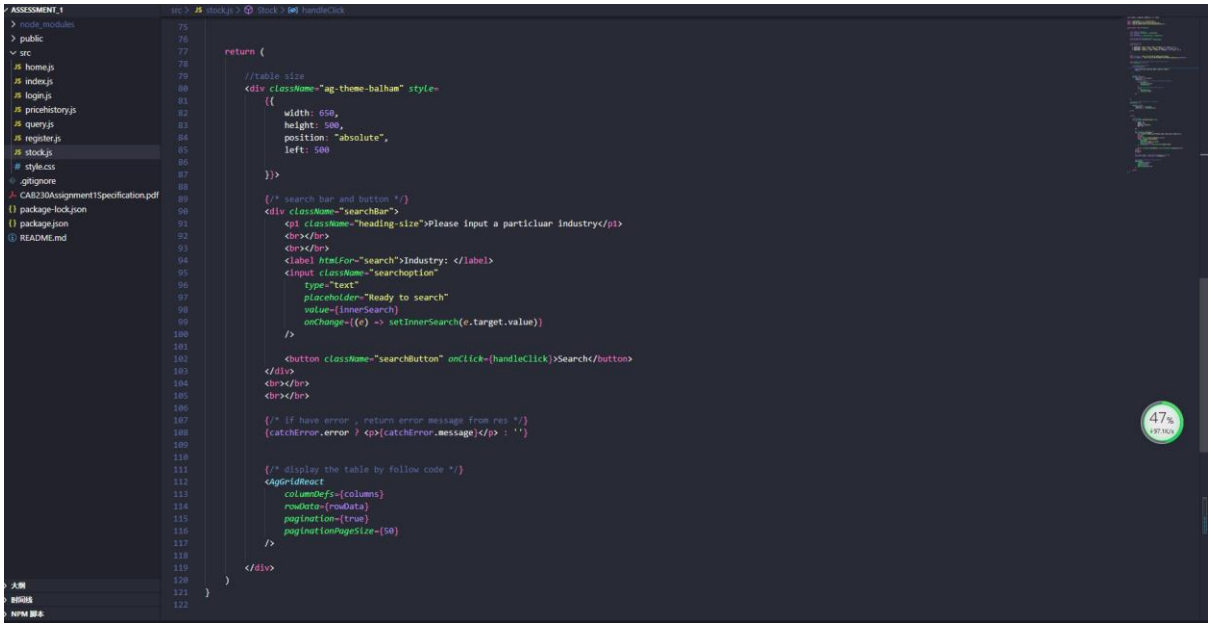
Now moving on handleClick, as we could see, if innersearch empty, it will alert message on the screen, otherwise, it will fetch data (depend on innerSearch). If res has an error, setErrorObject from res, setRowData is null, otherwise return value from res to setRowData, setErrorObject is null. useEffect is returning the value to table every time I start (Please above picture).



```
    //searchBar cannot be null
    if (!innerSearch) {

        alert('Search bar cannot be empty, please try again')
        return;

    }

    //fetch data
    fetch(url_template)
        .then((res) => res.json())
        .then((res) => {
            //if res have error, return error from res to display
            if (res.error) {
                setErrorObject(res);
                setRowData([]);
            }
            //otherwise return object array from res
            else {
                setRowData(res);
                setErrorObject({});
            }
        })


};

//to show all kind of industry in the table
useEffect(() => {

    fetch(url_stock)
        .then((res) => res.json())
        .then((res) => setRowData(res))
```

```
Compiled with warnings.

./src/stock.js
  Line 74:8:  React Hook useEffect has a missing dependency: 'url_stock'. Either include it or remove the dependency array  react-hooks/exhaustive-deps

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

Below code just return section on Stock function. I probably explain what my errorObject exactly do. It means that if errorObject have an error then return error message otherwise return null.


ASSESSMENT_1
node_modules
public
src
home.js
index.js
login.js
pricehistory.js
query.js
register.js
stock.js
style.css
.gitignore
CAB230Assignment1Specification.pdf
package-lock.json
package.json
README.md
大纲
时间线
NPM 脚本
src > stock.js > Stock > handleClick
return (
//table size
<div className="ag-theme-balham" style=
{{
width: 650,
height: 500,
position: "absolute",
left: 500
}}>
{/* search bar and button */}

<p1 className="heading-size">Please input a particluar industry</p1>

</br>

</br>

</br>
<label htmlFor="search">Industry: </label>
<input className="searchoption"
type="text"
placeholder="Ready to search"
value={innerSearch}
onChange={(e) => setInnerSearch(e.target.value)}
/>
<button className="searchButton" onClick={handleClick}>Search</button>

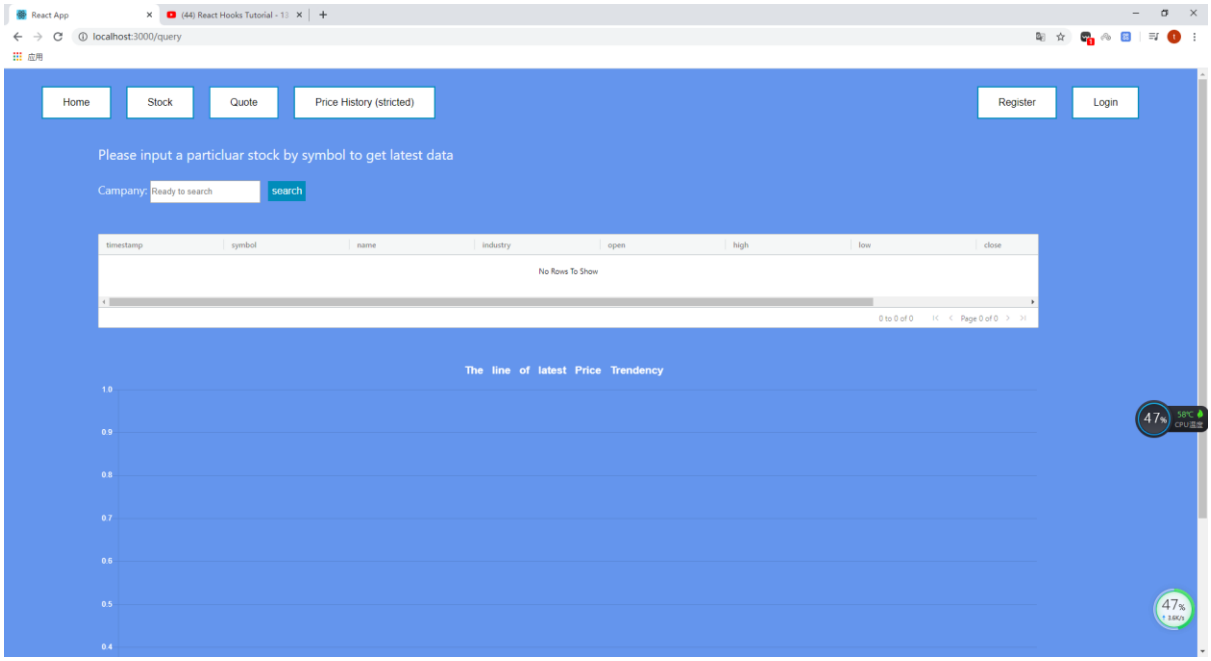
</br>

</br>
{/* if have error , return error message from res */}
{catchError.error ? {catchError.message} : ''}
{/* display the table by follow code */}
<AgGridReact
columnDefs={columns}
rowData={rowData}
pagination={true}
paginationPageSize={50}
/>

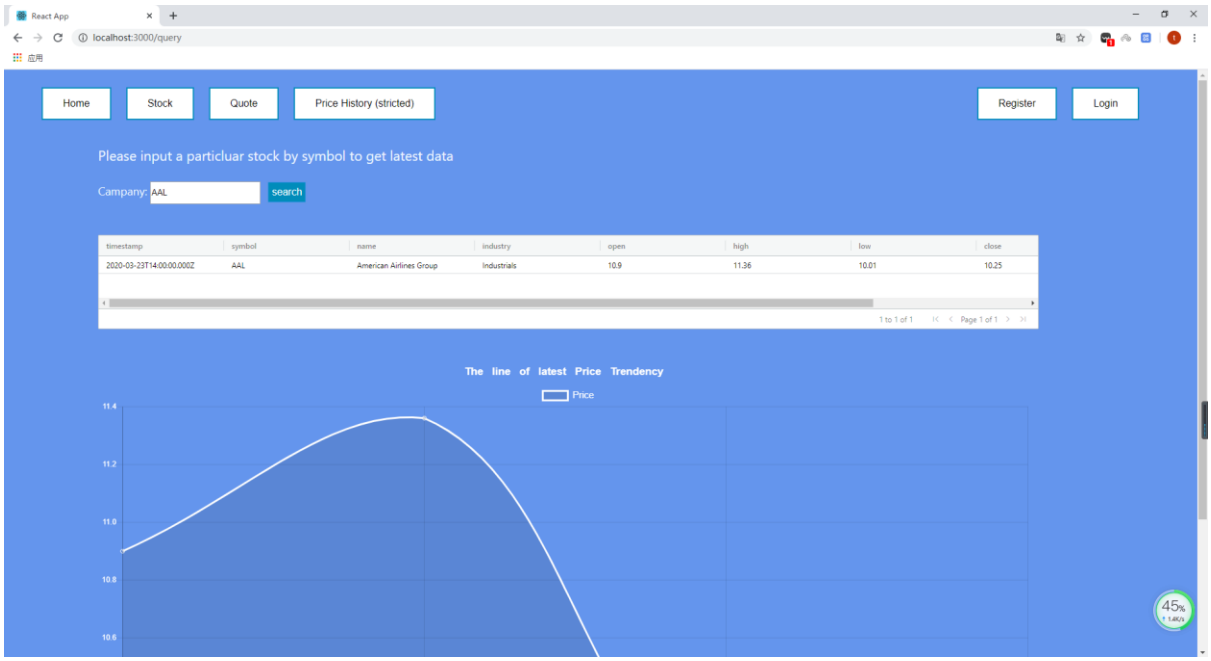
)
}
47%

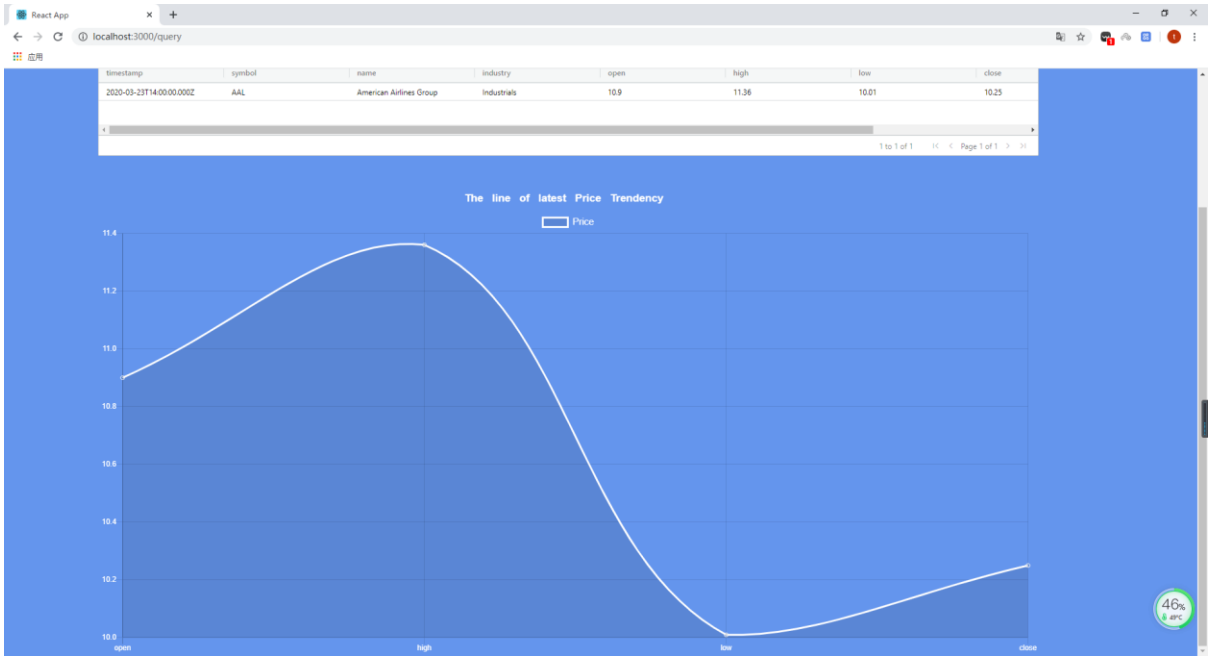
/stocks/{symbol}

In the second GET, it links to my query page (‘/query’). When I click ‘query’ button, it will move on to ‘query’ section. please see picture below.

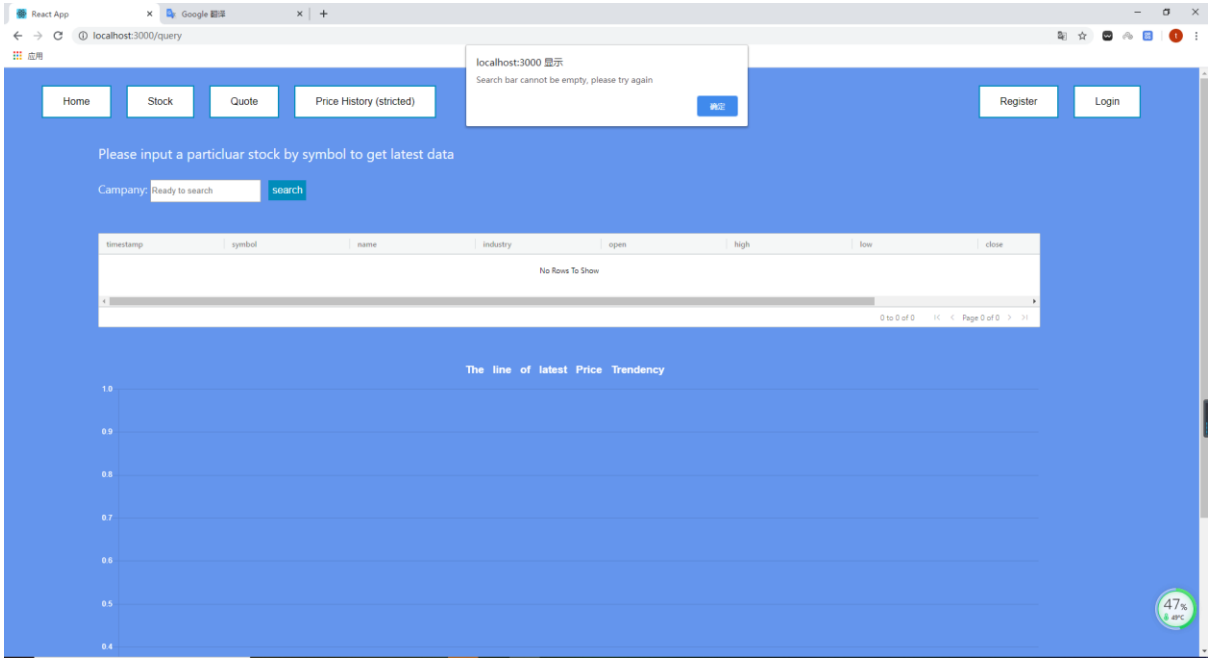


As we could see, it is similar with stock section. I add chartjs in the query to see price tendency on the latest day. I also write instruction on the top that provide user how to use it (only allow stock company name). now, I am going to type a symbol here to see what is happen.

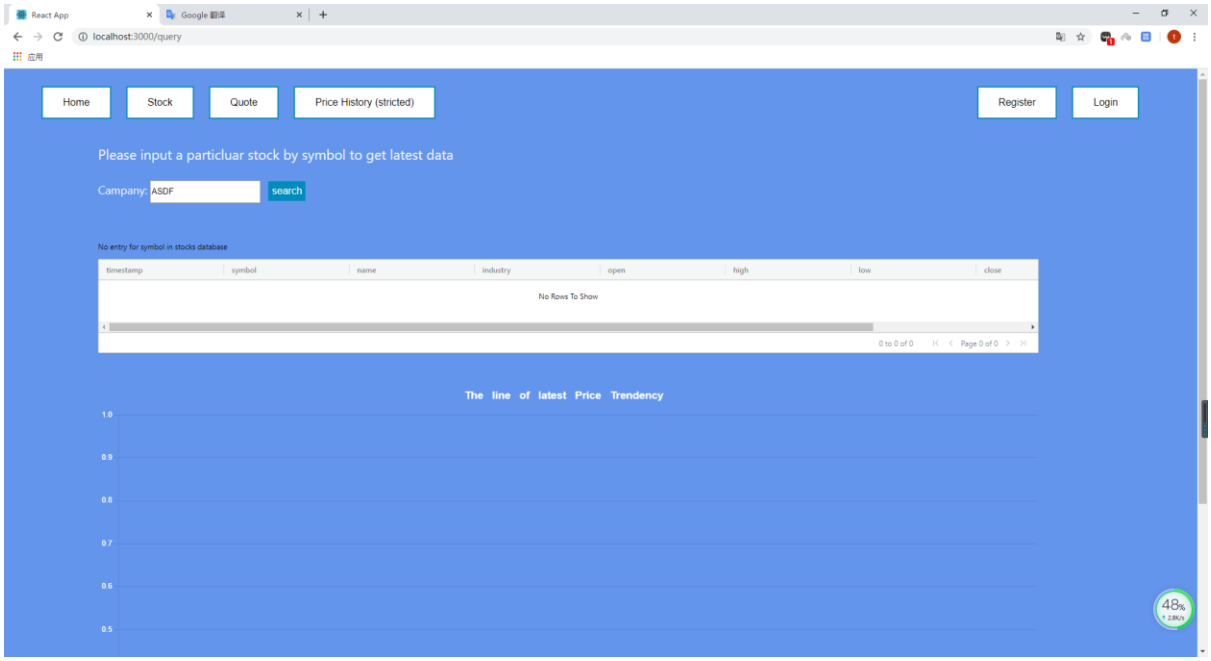




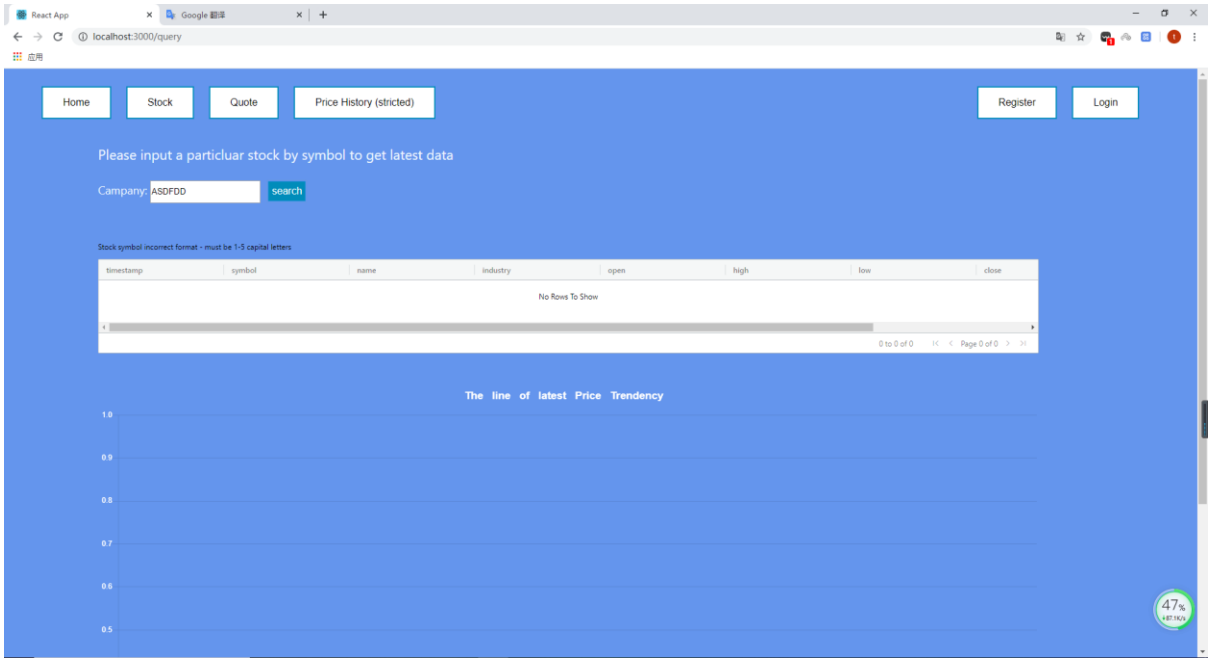
As we could see, I type one of company name by symbol (AAL). It returns the object from res (not array). That is why I have only one line in the table. In addition, how I default my chart is that I extract 4 objects type from res, which ‘open’, ‘high’, ‘low’ and ‘close’. And then, let these objects become array to display on the line chart. More detail I will explain below with my code. Now I am trying to clear my search bar and click it to what is happen.



We could see that if search bar is empty, then it will involve a warning on the top (alert), which means search bar cannot not be null. Now, I am going to type the symbol which doesn’t exist in the res to see what is happen.

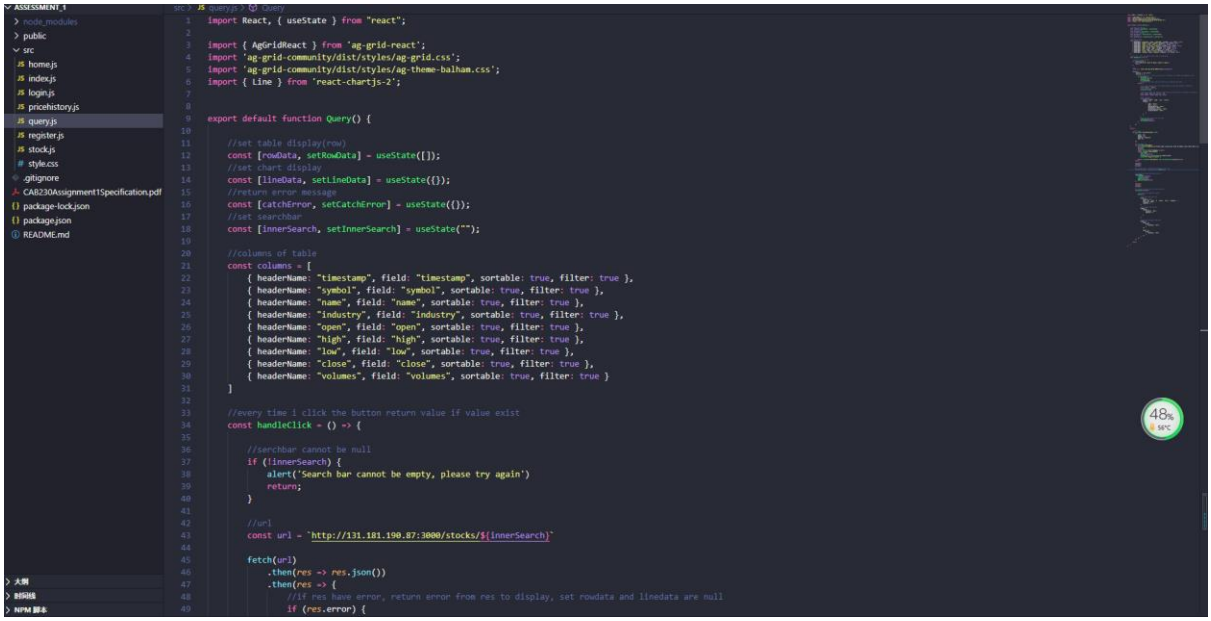


We could see that it involves error here on the left upper corner of table (doesn't have that company). Error message from res. I didn't code any error message. Now if I type more then 5 upper case letters to see what is happen.



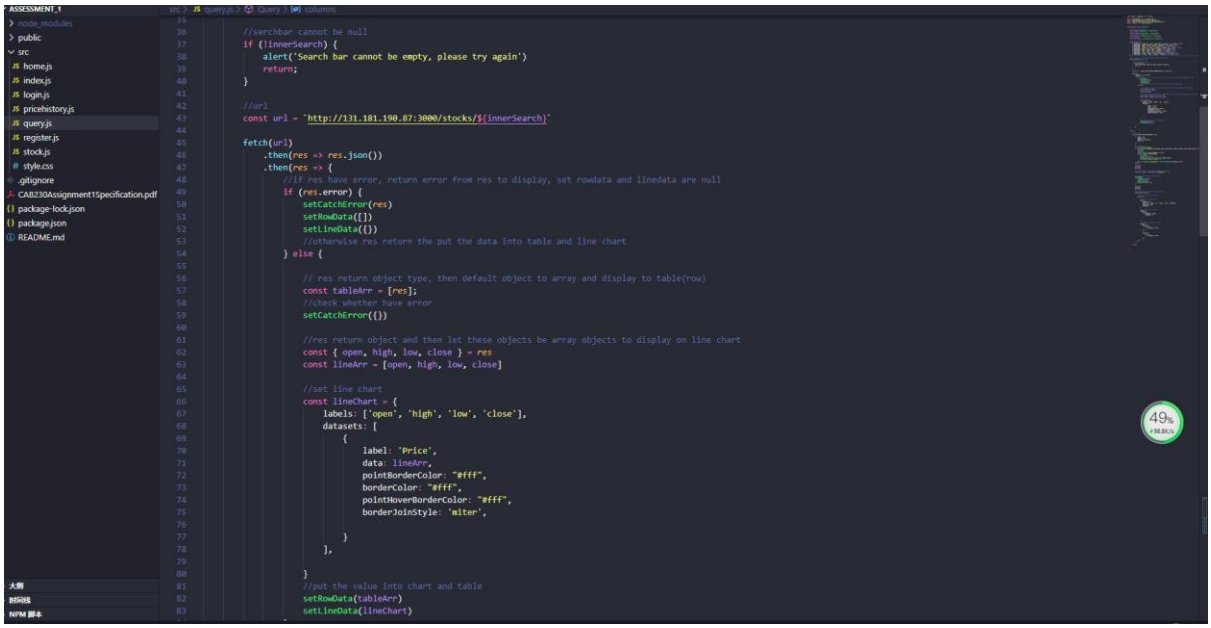
Error message is different with last one. The res gives us different error message depending on what you search for. Now I am going to explain my 'query' part code.

Firstly, I declare several useState on the top. Rowdata is setting row of table. Line data is setting dataset in the line chart. ErrorObject is returning error message from res. innerSearch is basically looking for what I type in search bar. Columns is columns of table.



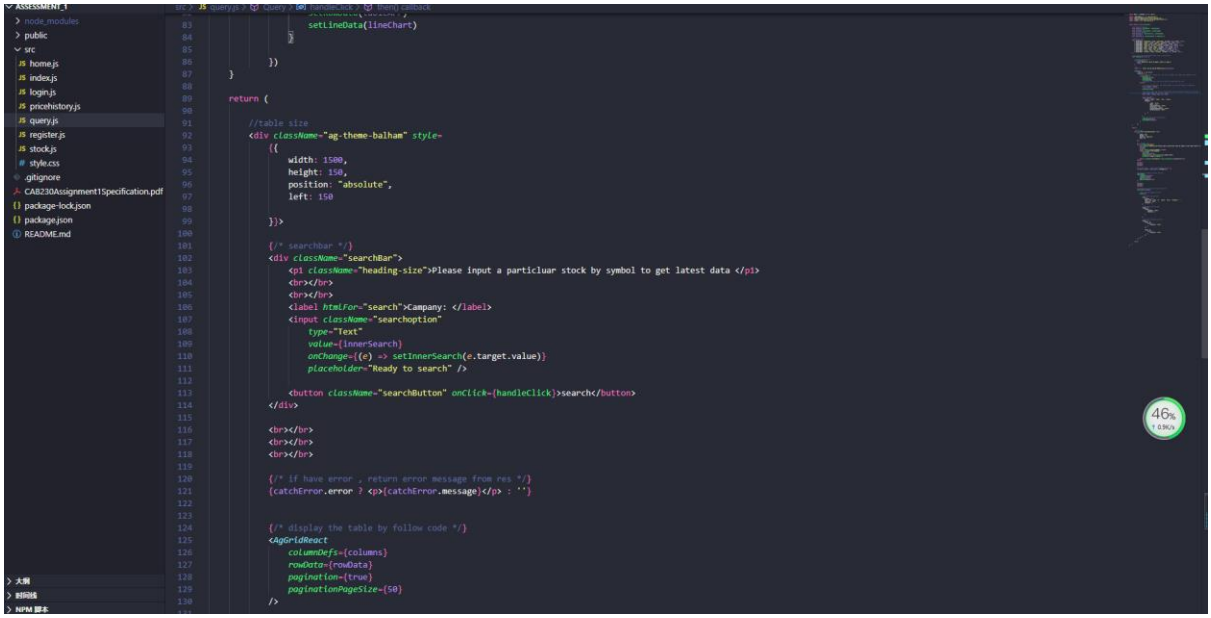
```
import React, { useState } from "react";

import { AgGridReact } from 'ag-grid-react';
import 'ag-grid-community/dist/styles/ag-grid.css';
import 'ag-grid-community/dist/styles/ag-theme-balham.css';
import { Line } from 'react-chartjs-2';


export default function Query() {

    //set table display(row)
    const [rowData, setRowData] = useState([]);
    //set chart display
    const [lineData, setLineData] = useState({});
    //return error message
    const [catchError, setCatchError] = useState({});
    //set searchbar
    const [innerSearch, setInnerSearch] = useState("");

    //columns of table
    const columns = [
        { headerName: "timestamp", field: "timestamp", sortable: true, filter: true },
        { headerName: "symbol", field: "symbol", sortable: true, filter: true },
        { headerName: "name", field: "name", sortable: true, filter: true },
        { headerName: "industry", field: "industry", sortable: true, filter: true },
        { headerName: "open", field: "open", sortable: true, filter: true },
        { headerName: "high", field: "high", sortable: true, filter: true },
        { headerName: "low", field: "low", sortable: true, filter: true },
        { headerName: "close", field: "close", sortable: true, filter: true },
        { headerName: "volumes", field: "volumes", sortable: true, filter: true }
    ]

    //every time i click the button return value if value exist
    const handleClick = () => {

        //serchbar cannot be null
        if (!innerSearch) {
            alert('Search bar cannot be empty, please try again')
            return;
        }

        //url
        const url = `http://131.181.190.87:3000/stocks/${innerSearch}`

        fetch(url)
            .then(res => res.json())
            .then(res => {
                //if res have error, return error from res to display, set rowdata and linedata are null
                if (res.error) {
```

The usage of handleClick is similar with my ‘stock’ section. It also fetches data depend on ‘search’ button. If search bar is empty, then involve alert. If not, then go to fetch data. If res return error. Then set error to setErrorObject, setRowData and setLineData are null. Otherwise, [res] means res return object to me, I need to let it become array so that it could display on the table. And then, set setErrorObject is null, then, extracting 4 objects from res and let them be array so that we could put them into dataset. All the values will display on the line chart.

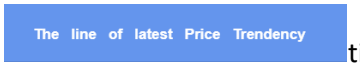


```
        //serchbar cannot be null
        if (!innerSearch) {
            alert('Search bar cannot be empty, please try again')
            return;
        }

        //url
        const url = `http://131.181.190.87:3000/stocks/${innerSearch}`

        fetch(url)
            .then(res => res.json())
            .then(res => {
                //if res have error, return error from res to display, set rowdata and linedata are null
                if (res.error) {
                    setCatchError(res)
                    setRowData([])
                    setLineData({})
                    //otherwise res return the put the data into table and line chart
                } else {

                    // res return object type, then default object to array and display to table(row)
                    const tableArr = [res];
                    //check whether have error
                    setCatchError({})

                    //res return object and then let these objects be array objects to display on line chart
                    const { open, high, low, close } = res
                    const lineArr = [open, high, low, close]

                    //set line chart
                    const lineChart = {
                        labels: ['open', 'high', 'low', 'close'],
                        datasets: [
                            {
                                label: 'Price',
                                data: lineArr,
                                pointBorderColor: "#fff",
                                borderColor: "#fff",
                                pointHoverBorderColor: "#fff",
                                borderJoinStyle: 'miter',

                            }
                        ],

                    }
                    //put the value into chart and table
                    setRowData(tableArr)
                    setLineData(lineChart)
```

Now, it will return jsx. How I default or design all the layout. errorObject is similar meaning with stock errorObject. It will return error message from res.



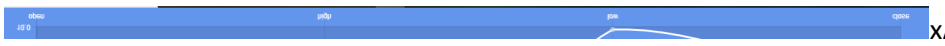
Below code is how I default line and table. I will focus on ‘Line’ chart section. What ‘option’ use for is that Encapsulating layout of line chart. Title mean it will have a heading or title on the top of line chart. Label is what value that I user for (Please see below blue picture). Scales is default xAxes and yAxes layout. I just change them to white color. I basically learn how to chart.js from several resource. I will put it on module part and reference.

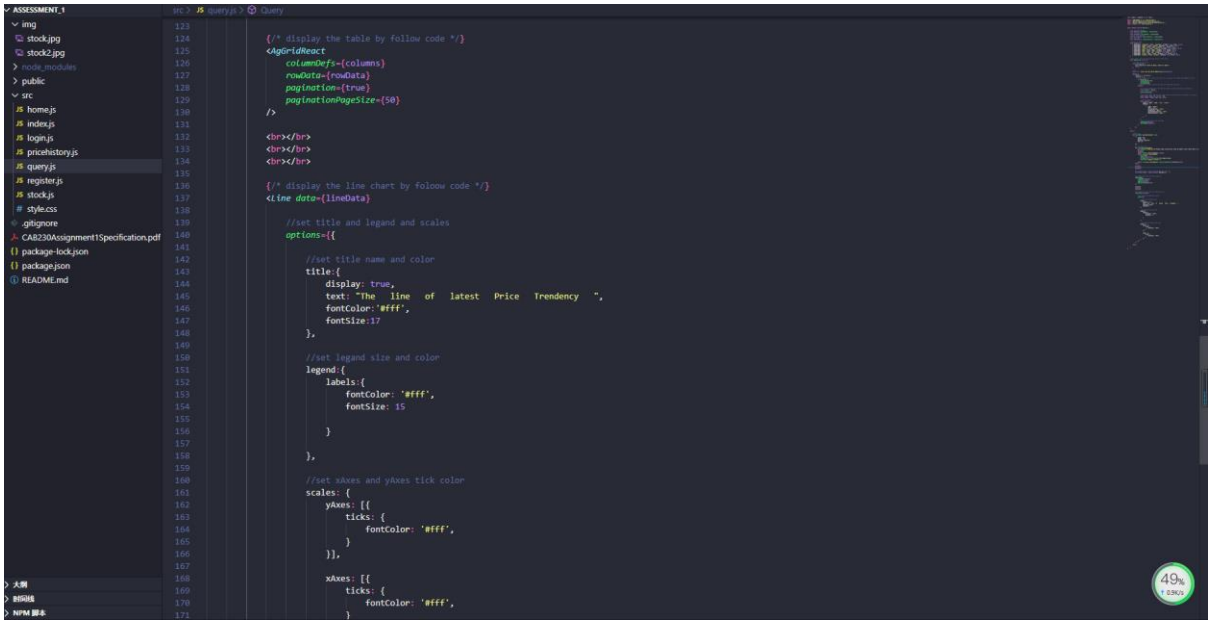


title

Label


yAxes


xAxes


ASSESSMENT_1
img
stock.jpg
stock2.jpg
node_modules
public
src
home.js
index.js
login.js
pricehistory.js
query.js
register.js
stock.js
style.css
.gitignore
CAB230Assignment1Specification.pdf
package-lock.json
package.json
README.md
大纲
时间线
NPM 脚本
src > JS query.js > Query
{/* display the table by follow code */}
<AgGridReact
columnDefs={columns}
rowData={rowData}
pagination={true}
paginationPageSize={50}
/>

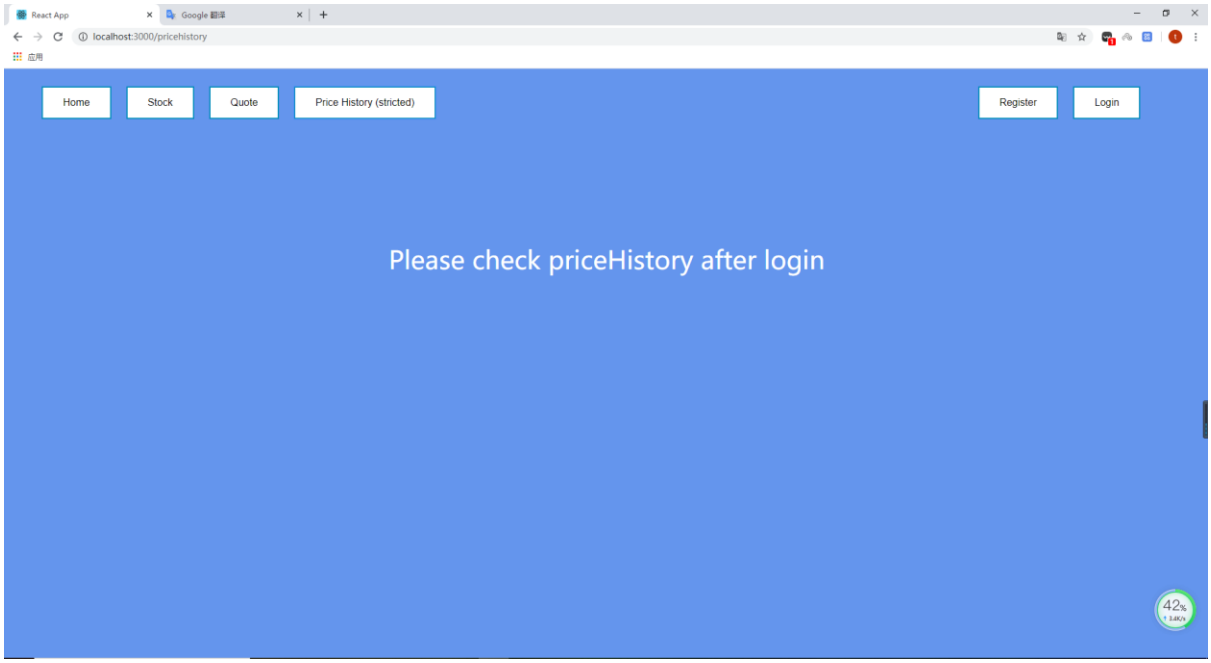
</br>

</br>

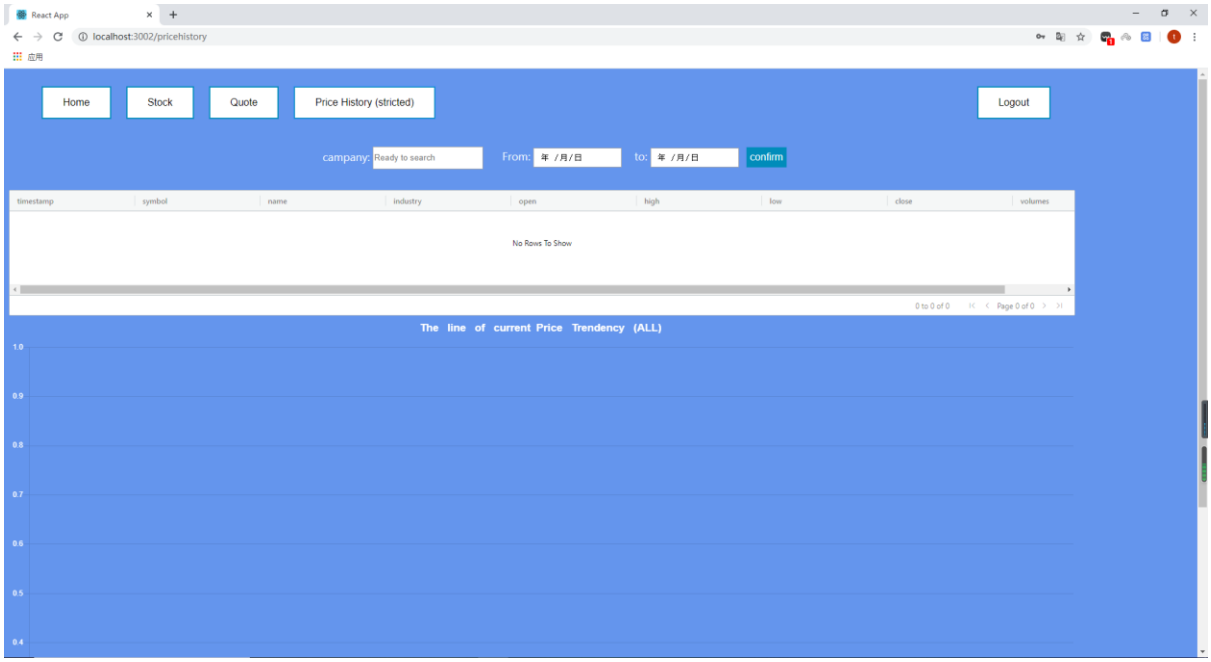
</br>
{/* display the line chart by foloow code */}
<Line data={lineData}
//set title and legand and scales
options={{
//set title name and color
title:{
display: true,
text: "The line of latest Price Trendency ",
fontColor:"#fff",
fontSize:17
},
//set legand size and color
legend:{
labels:{
fontColor: "#fff",
fontSize: 15
}
},
//set xAxes and yAxes tick color
scales: {
yAxes: [{
ticks: {
fontColor: "#fff",
}
}],
xAxes: [{
ticks: {
fontColor: "#fff",
}
49%
0.5K/s

/stocks/authed/{symbol}

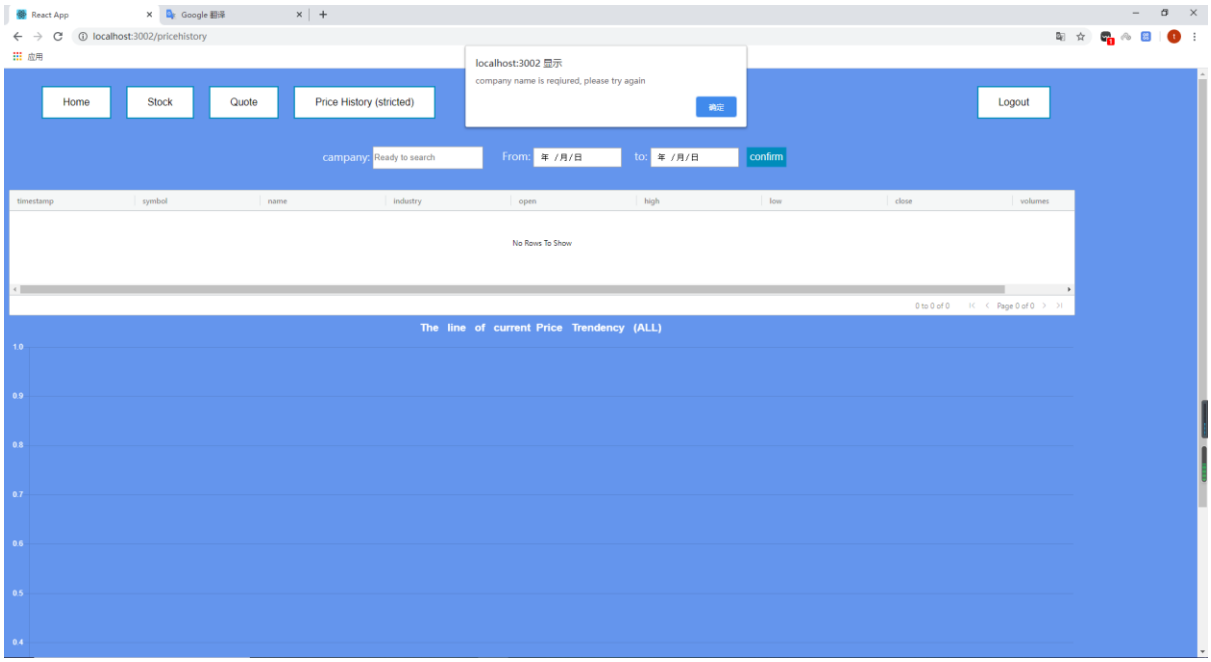
In the Third GET, it links to my price history page (‘/pricehistory’). When I click ‘price history (strict)’ button, it will move on to ‘price history’ section. please see picture below. There is a heading to tell user if you want to check price history on my stock system, you have to login. Now, let me login to see what is happen.



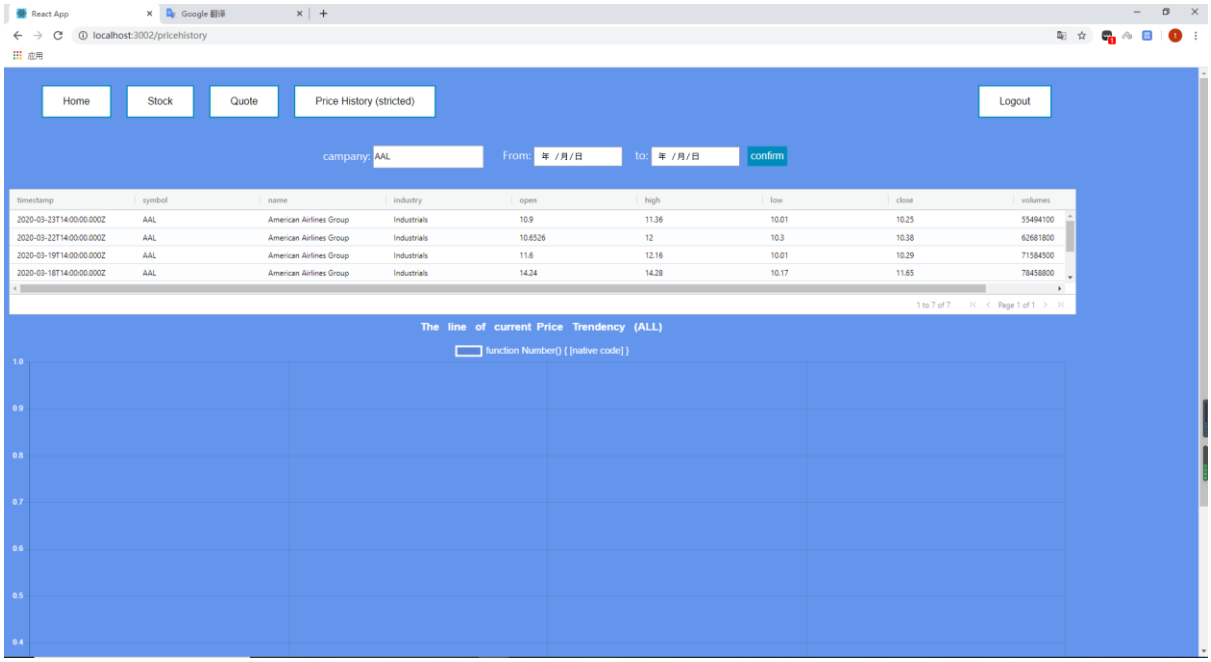
Now, we can see the difference. It has search bar, table and line char here. (in the date section, my computer system is not English, so detect my computer system). If I click without any character or date on the search bar. See what is happen.



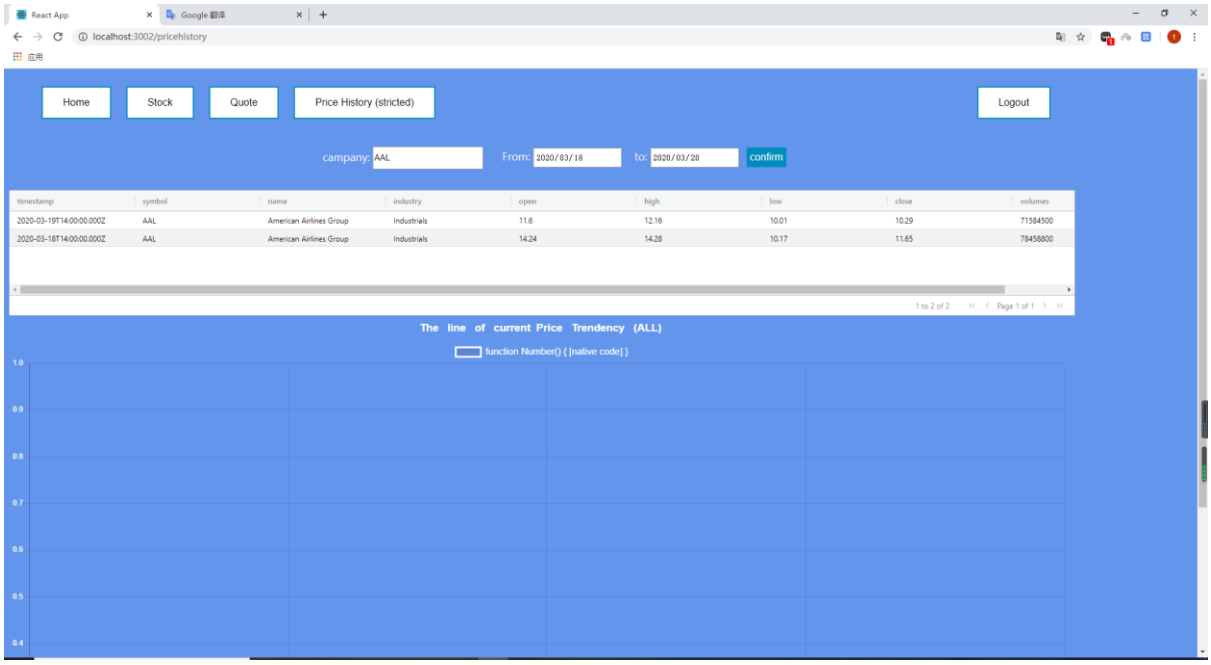
Now we could see that it involves an alert, which means that company name by symbol is required. Date is not required. Let me type a one of company name by symbol to see what is happen.



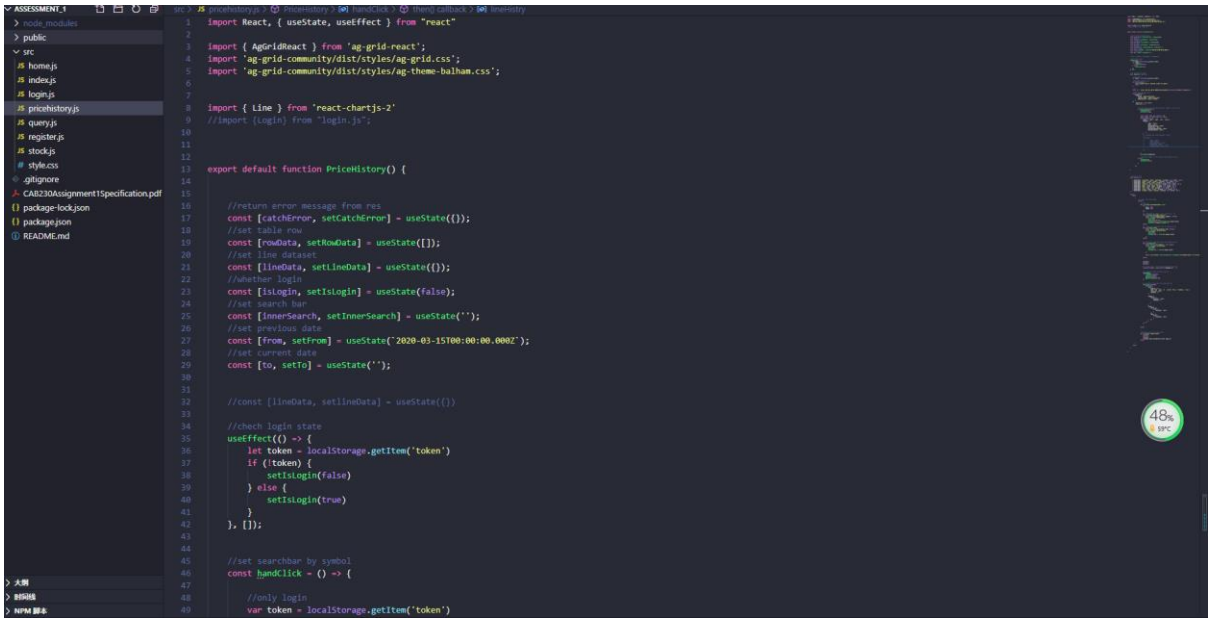
If I type ‘AAL’ (one of companies) in the search bar. It returns all values from res. If I add date to see what is happen.



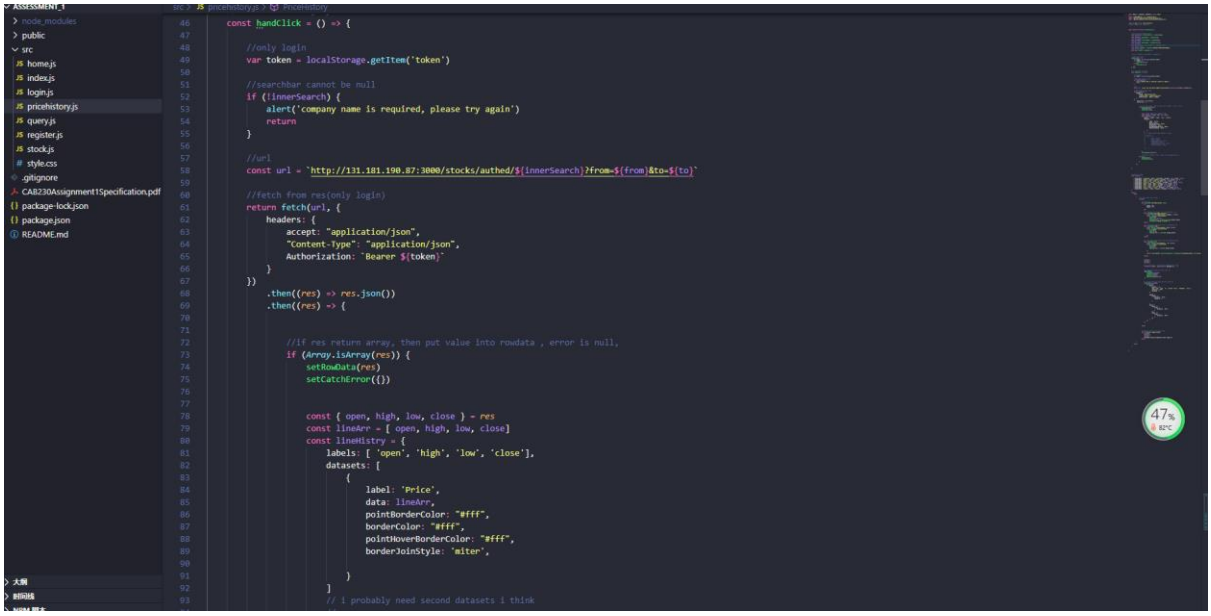
Now I add company and data from 2020/3/18 to 2020/3/20. It returns exactly value one the table that I search for. However, I try to make line chart happen. But I stuck. I don't know how to default the date. This is one of my limitation. Let's move on to my code.



I import agrid, react, useState, useEffect and chartjs on the top. Moving down, I declare several useState in the function. It is similar usage with my stock.js and query.js. useEffect is checking login state. If log in, it will return token form res and set setIsLogin is true. Otherwise, false.

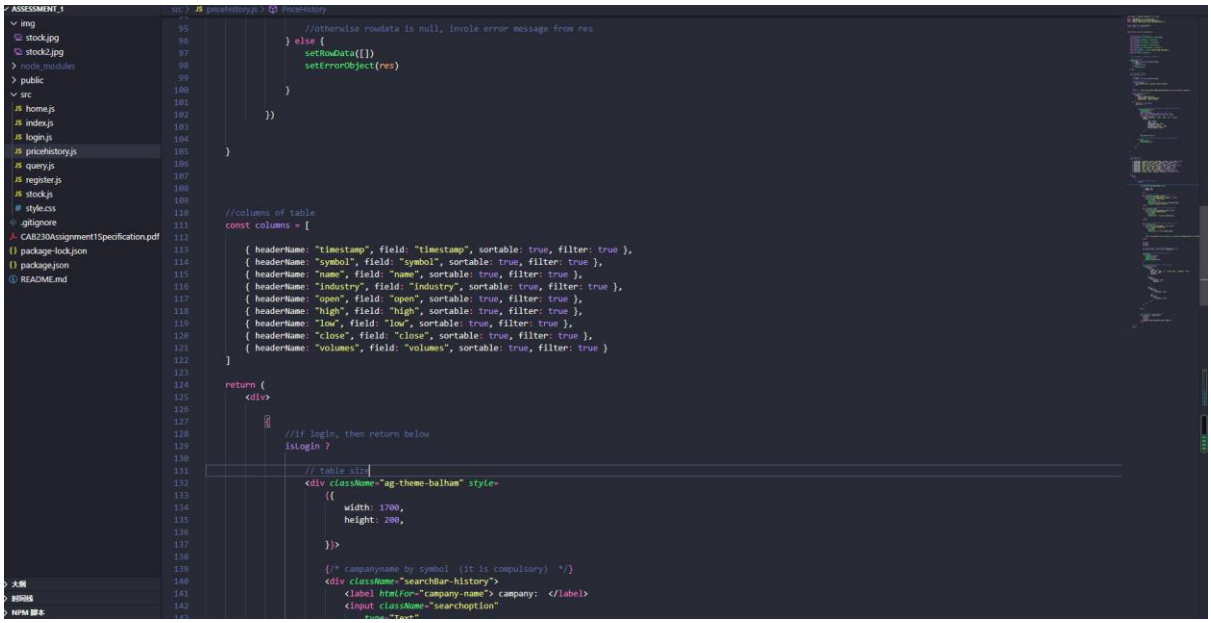


The usage of handleClick is similar with my ‘stock’ section. It also fetches data depend on ‘search’ button. If search bar is empty, then involve alert. If not, then go to fetch data. If res return error. Then set error to setErrorObject, setRowData and setLineData are null. Otherwise, [res] means res return object to me, I need to let it become array so that it could display on the table. And then, set setErrorObject is null, then, extracting 4 objects from res and let them be array so that we could put them into dataset. All the values will display on the line chart.



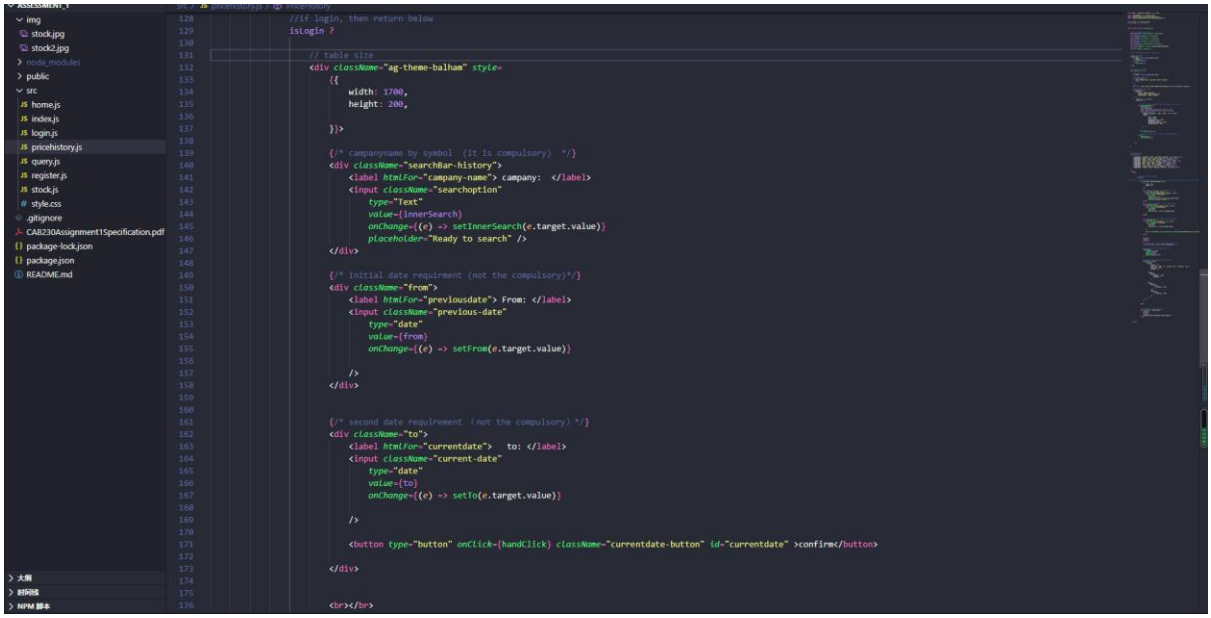
```
const handClick = () => {

    //only login
    var token = localStorage.getItem('token')

    //searchbar cannot be null
    if (!innerSearch) {
        alert('company name is required, please try again')
        return
    }

    //url
    const url = `http://131.181.190.87:3000/stocks/authed/${innerSearch}?from=${from}&to=${to}`

    //fetch from res(only login)
    return fetch(url, {
        headers: {
            accept: "application/json",
            "Content-Type": "application/json",
            Authorization: `Bearer ${token}`
        }
    })
        .then((res) => res.json())
        .then((res) => {


            //if res return array, then put value into rowdata , error is null,
            if (Array.isArray(res)) {
                setRowData(res)
                setCatchError({})


                const { open, high, low, close } = res
                const lineArr = [ open, high, low, close]
                const lineHistry = {
                    labels: [ 'open', 'high', 'low', 'close'],
                    datasets: [
                        {
                            label: 'Price',
                            data: lineArr,
                            pointBorderColor: "#fff",
                            borderColor: "#fff",
                            pointHoverBorderColor: "#fff",
                            borderJoinStyle: 'miter',


                        }
                    ]
                    // i probably need second datasets i think
```

The table columns. Moving on if islogin is true It will return below code.



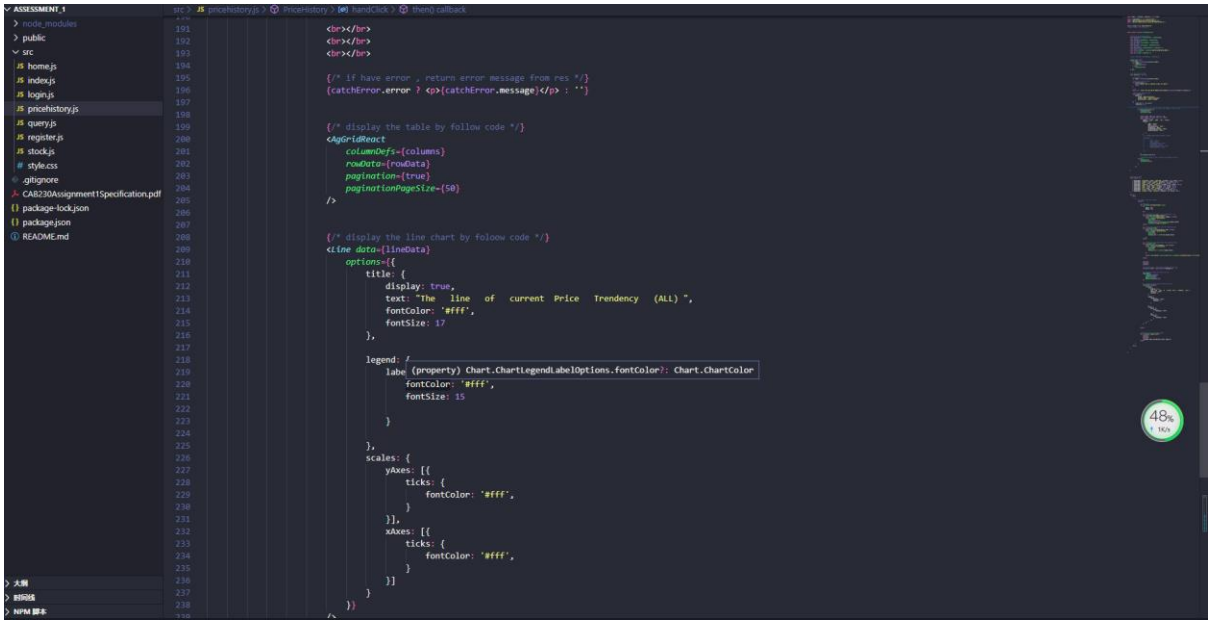
```
                //otherwise rowdata is null, invole error message from res
            } else {
                setRowData([])
                setErrorObject(res)


            }


        })


}



//columns of table
const columns = [

    { headerName: "timestamp", field: "timestamp", sortable: true, filter: true },
    { headerName: "symbol", field: "symbol", sortable: true, filter: true },
    { headerName: "name", field: "name", sortable: true, filter: true },
    { headerName: "industry", field: "industry", sortable: true, filter: true },
    { headerName: "open", field: "open", sortable: true, filter: true },
    { headerName: "high", field: "high", sortable: true, filter: true },
    { headerName: "low", field: "low", sortable: true, filter: true },
    { headerName: "close", field: "close", sortable: true, filter: true },
    { headerName: "volumes", field: "volumes", sortable: true, filter: true }
]

return (
    <div>

        {

            //if login, then return below
            isLogin ?

                // table size
                <div className="ag-theme-balham" style=
                    {{
                        width: 1700,
                        height: 200,

                    }}>

                    {/* companyname by symbol  (it is compulsory)  */}
                    <div className="searchBar-history">
                        <label htmlFor="campany-name"> campany:  </label>
                        <input className="searchoption"
```

The layout code



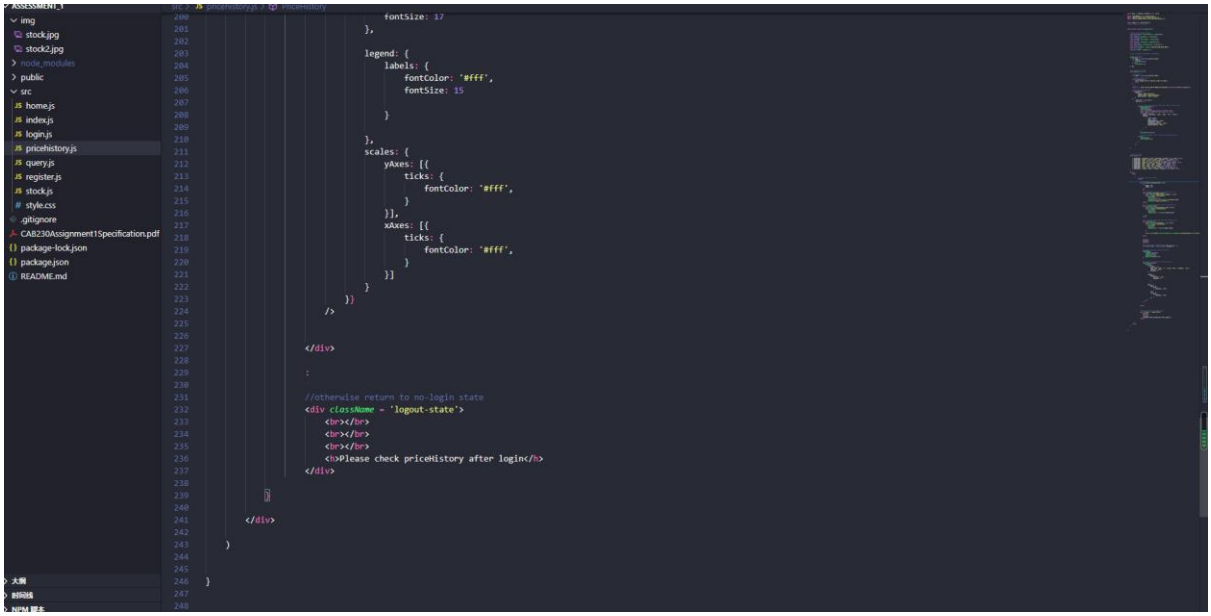
```
//if login, then return below
islogin ?

    // table size
    <div className="ag-theme-balham" style=
        {{
            width: 1700,
            height: 200,

        }}>

        {/* campanyname by symbol  (it is compulsory)  */}
        <div className="searchBar-history">
            <label htmlFor="company-name"> campany:  </label>
            <input className="searchoption"
                type="Text"
                value={innerSearch}
                onChange={(e) => setInnerSearch(e.target.value)}
                placeholder="Ready to search" />
        </div>

        {/* initial date requirment (not the compulsory)*/}
        <div className="from">
            <label htmlFor="previousdate"> From: </label>
            <input className="previous-date"
                type="date"
                value={from}
                onChange={(e) => setFrom(e.target.value)}

            />
        </div>


        {/* second date requirement  (not the compulsory) */}
        <div className="to">
            <label htmlFor="currentdate">   to: </label>
            <input className="current-date"
                type="date"
                value={to}
                onChange={(e) => setTo(e.target.value)}

            />

            <button type="button" onClick={handClick} className="currentdate-button" id="currentdate" >confirm</button>

        </div>


        <br></br>
```

Layout code of table and line-chart



```
            <br></br>
            <br></br>
            <br></br>

            {/* if have error , return error message from res */}
            {catchError.error ? <p>{catchError.message}</p> : ''}


            {/* display the table by follow code */}
            <AgGridReact
                columnDefs={columns}
                rowData={rowData}
                pagination={true}
                paginationPageSize={50}
            />


            {/* display the line chart by foloow code */}
            <Line data={lineData}
                options={{
                    title: {
                        display: true,
                        text: "The  line  of  current  Price  Trendency  (ALL) ",
                        fontColor: '#fff',
                        fontSize: 17
                    },

                    legend: {
                        labe (property) Chart.ChartLegendLabelOptions.fontColor?: Chart.ChartColor
                            fontColor: '#fff',
                            fontSize: 15

                        }

                    },
                    scales: {
                        yAxes: [{
                            ticks: {
                                fontColor: '#fff',
                            }
                        }],
                        xAxes: [{
                            ticks: {
                                fontColor: '#fff',
                            }
                        }]
                    }
                }}
```

Otherwise, islogin is false, it will return a heading on the screen.


ASSESSMENT_1
img
stock.jpg
stock2.jpg
node_modules
public
src
home.js
index.js
login.js
pricehistory.js
query.js
register.js
stock.js
style.css
.gitignore
CAB230Assignment1Specification.pdf
package-lock.json
package.json
README.md
大纲
时间线
NPM 脚本
src > pricehistory.js > PriceHistory
fontSize: 17
},
legend: {
labels: {
fontColor: '#fff',
fontSize: 15
}
},
scales: {
yAxes: [{
ticks: {
fontColor: '#fff',
}
}],
xAxes: [{
ticks: {
fontColor: '#fff',
}
}]
}
})
/>

:
//otherwise return to no-login state

</br>

</br>

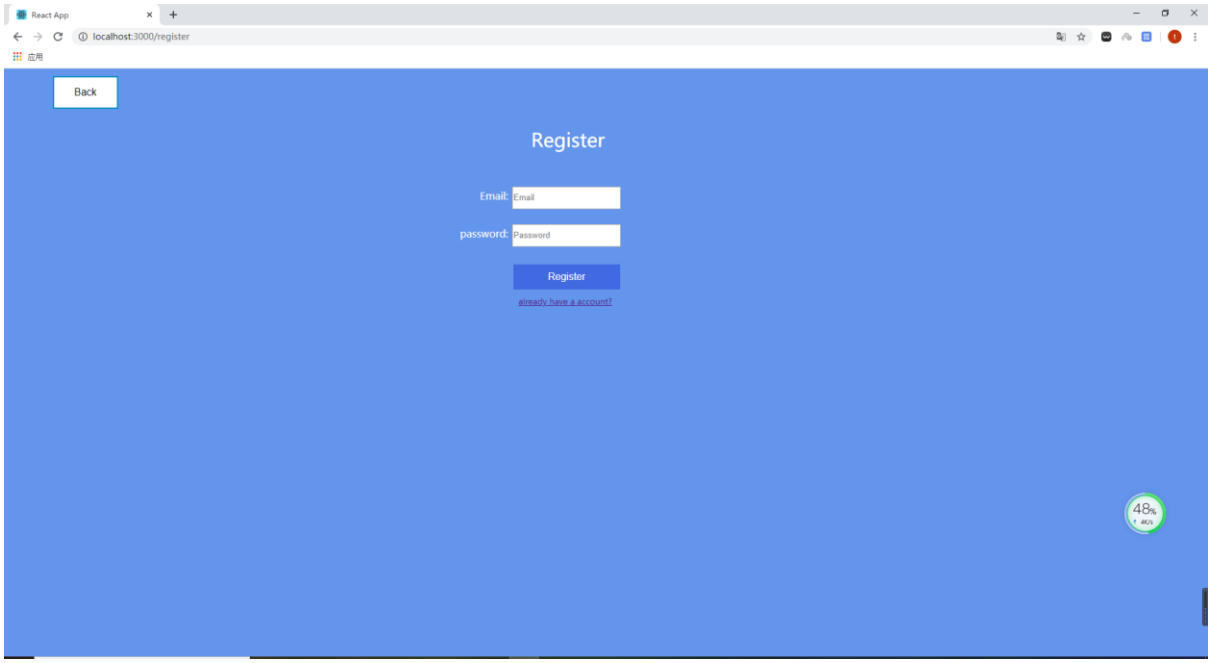
</br>
<h>Please check priceHistory after login</h>

}

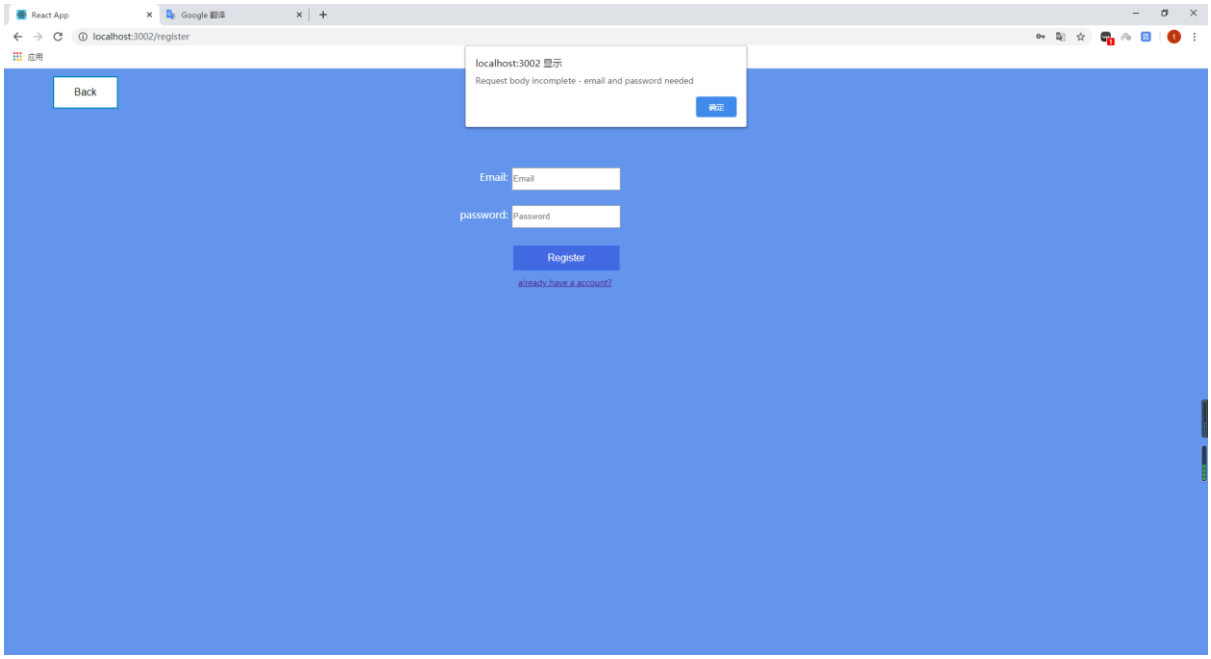
)
}

/user/register

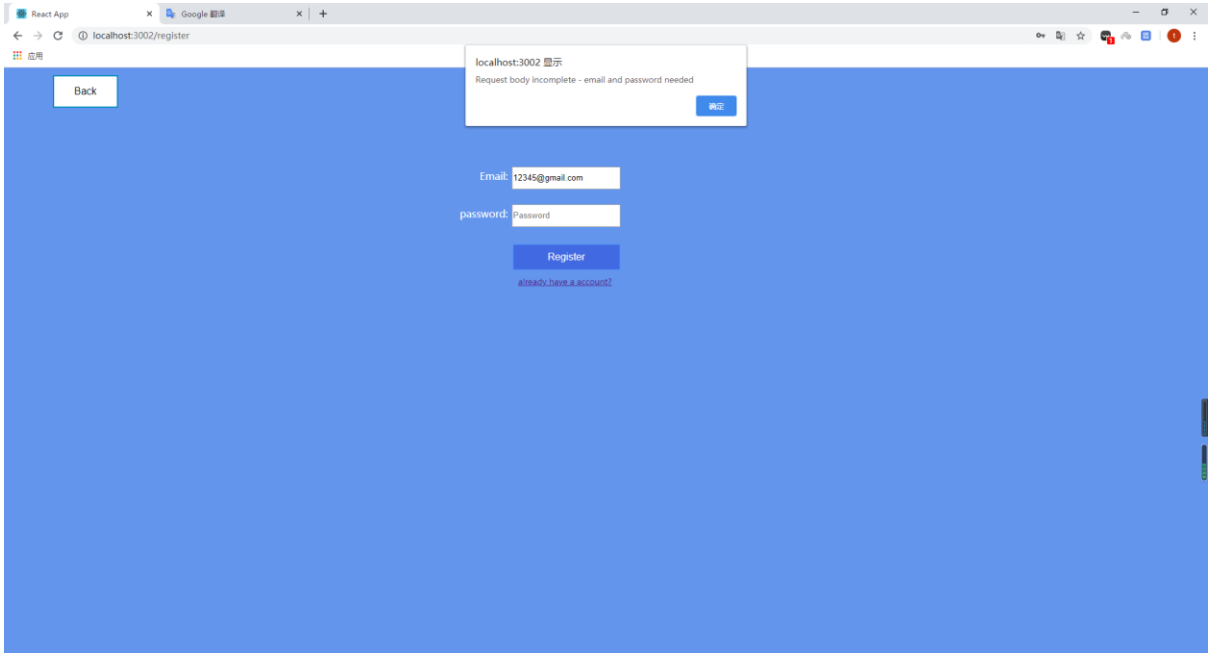
In the register POST, it links to my register page (‘/register’). When I click ‘Register’ button, it will move on to ‘register’ section. please see picture below. We could see it is moving on a new page, which is ‘/register’. That is my register layout. It is simple but useful. On the left upper corner, there is button. Every time click it, it will return home page (‘/’). In addition, bellowing register button there is a link. If I click it, it will move on login section (‘login’). Now, i am going to create an account without email and password to see what is happen.



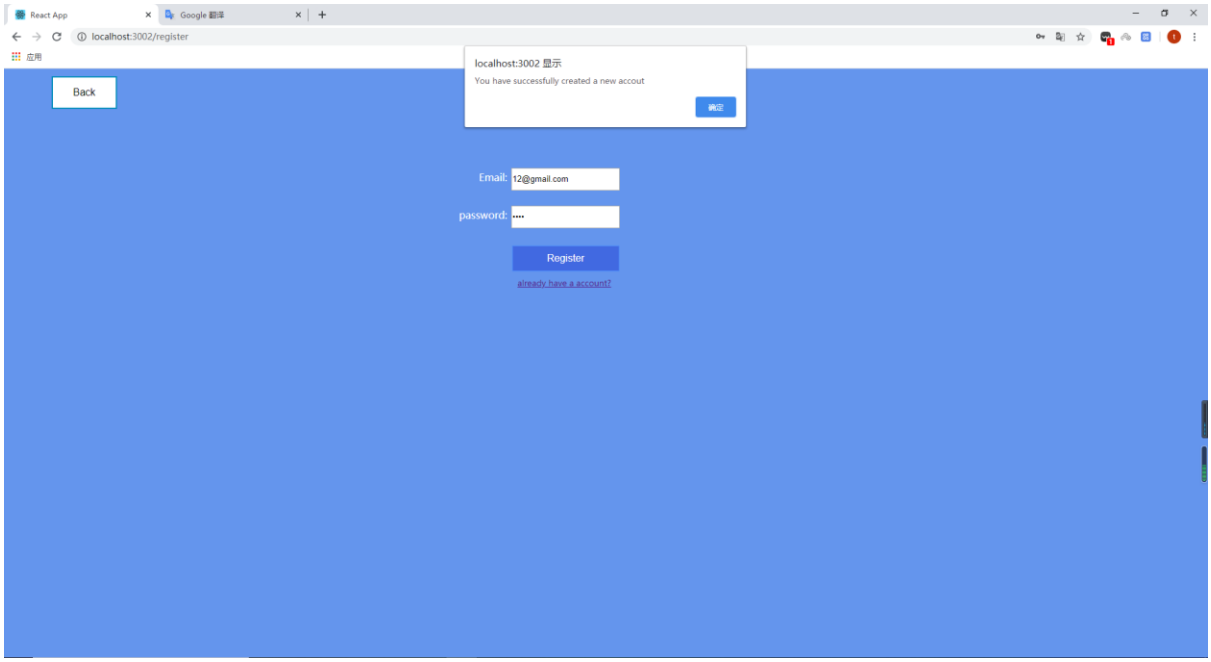
As we could see that it involves an alert, because I return error message from res as an alert. Now, I am going to type email without password to see what is happen.



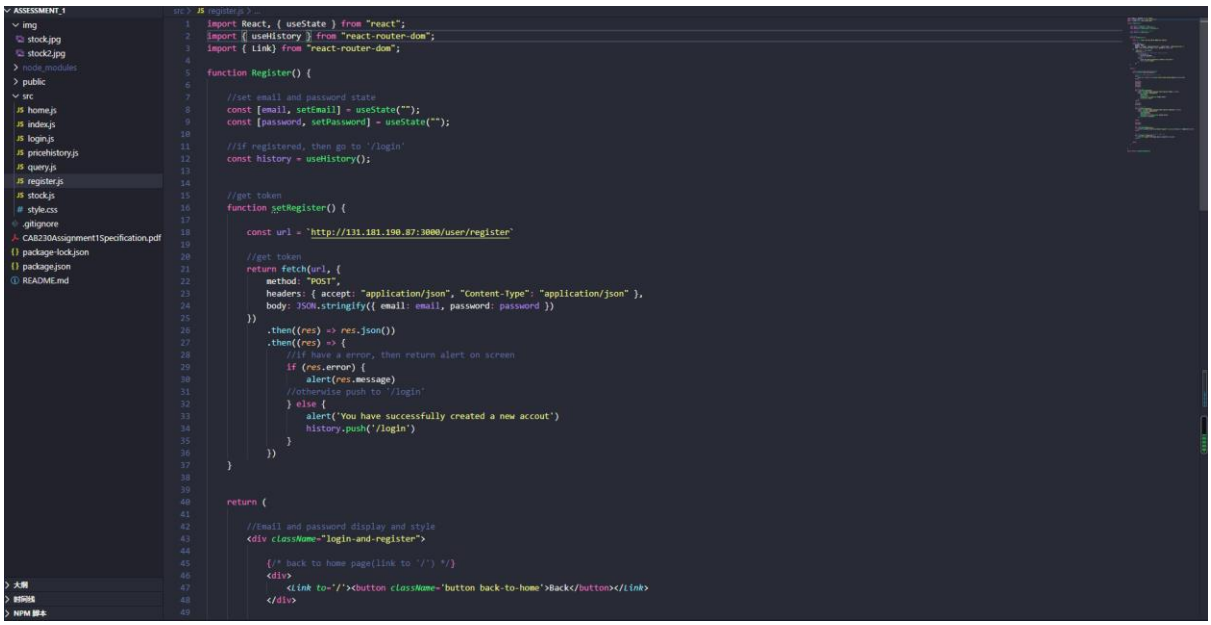
We could see that it involves another alert. It basically depends on res. Now I am going to create a real account to see what is happen.



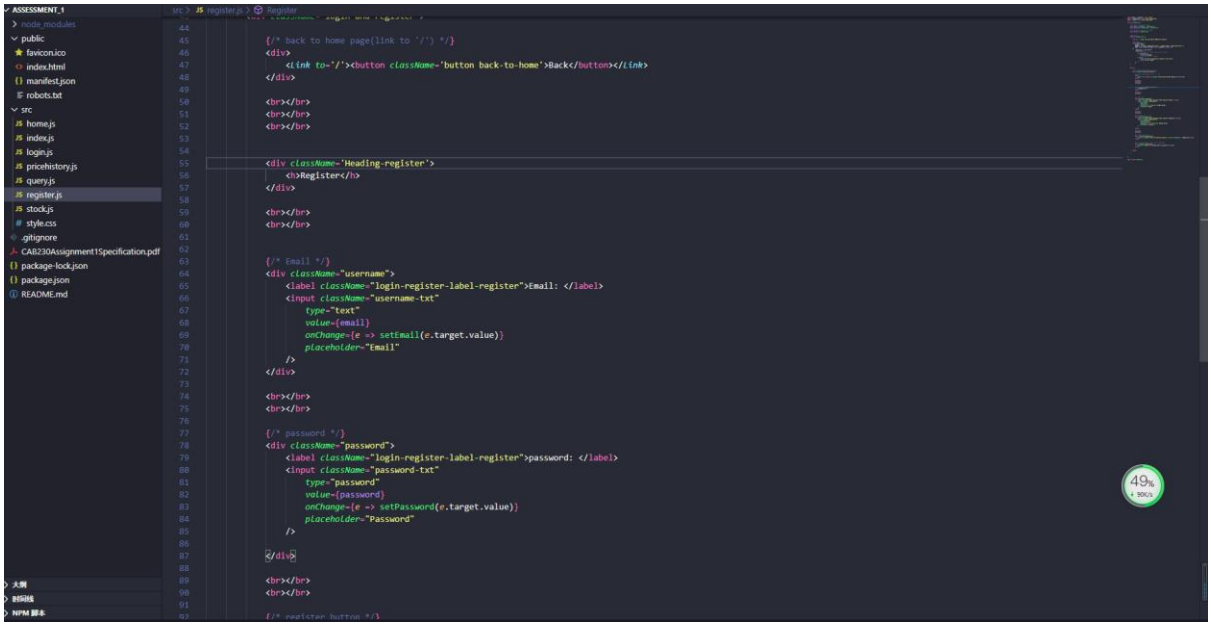
Now we could see that it involves another alert. I added this an alert on my code (not res return). When click the alert button, it will move on login section.



Moving on my register code, it is quite simple. I basically follow JWT worksheet and watch several useful videos on the YouTube to achieve that. useHistory is if user registered, it will move on login section. It is similar with ‘Link’.



Below code it is register layout. I wrote css code in my style.css.


ASSESSMENT_1
node_modules
public
favicon.ico
index.html
manifest.json
robots.txt
src
home.js
index.js
login.js
pricehistory.js
query.js
register.js
stock.js
style.css
.gitignore
CAB230Assignment1Specification.pdf
package-lock.json
package.json
README.md
大纲
时间线
NPM 脚本
src > JS register.js > Register
{/* back to home page(link to '/') */}

<Link to='/'><button className='button back-to-home'>Back</button></Link>

</br>

</br>

</br>

<h>Register</h>

</br>

</br>
{/* Email */}

<label className="login-register-label-register">Email: </label>
<input className="username-txt"
type="text"
value={email}
onChange={e => setEmail(e.target.value)}
placeholder="Email"
/>

</br>

</br>
{/* password */}

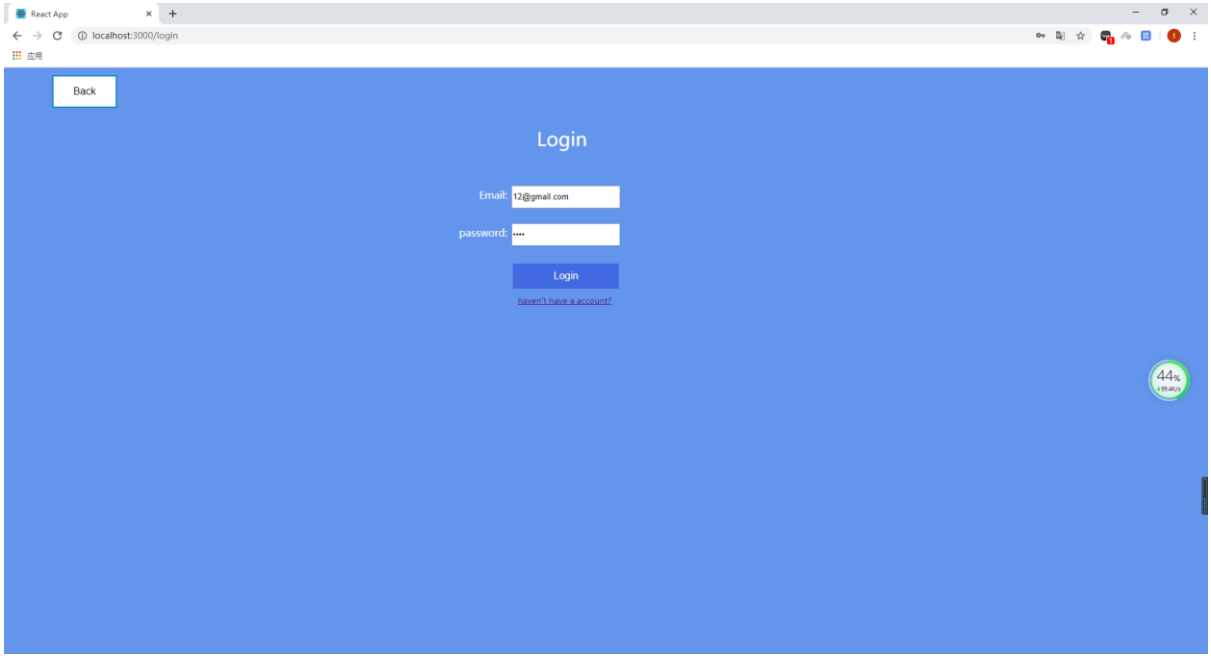
<label className="login-register-label-register">password: </label>
<input className="password-txt"
type="password"
value={password}
onChange={e => setPassword(e.target.value)}
placeholder="Password"
/>

</br>

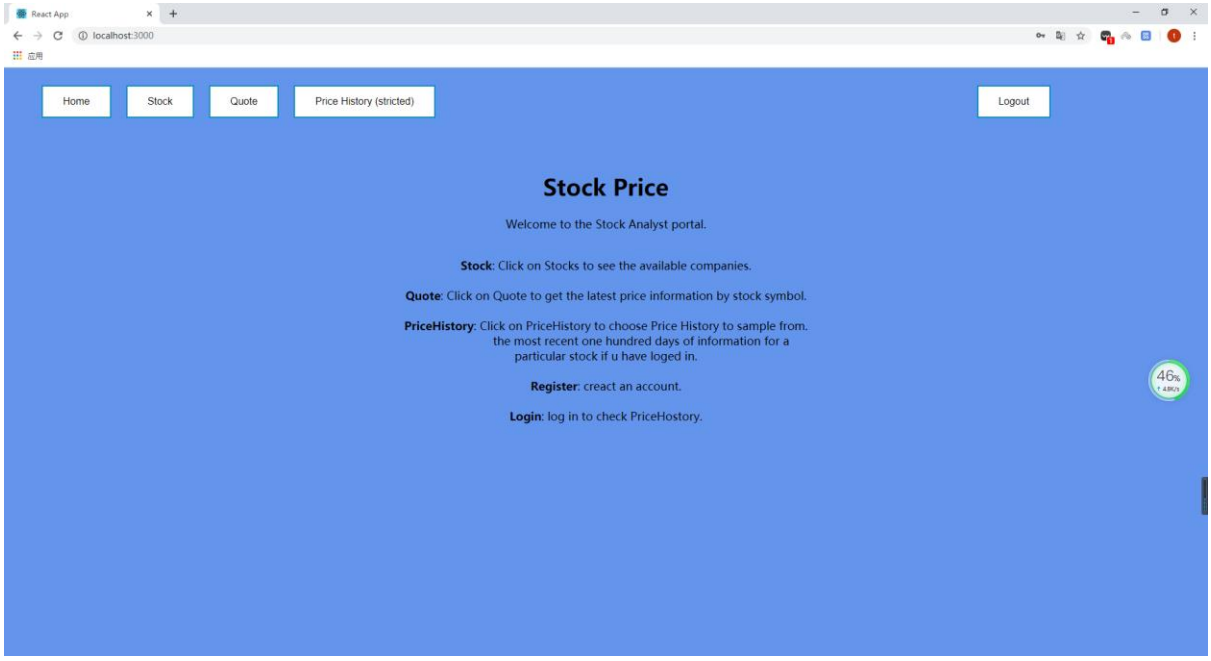
</br>
49%
+ 90Kb

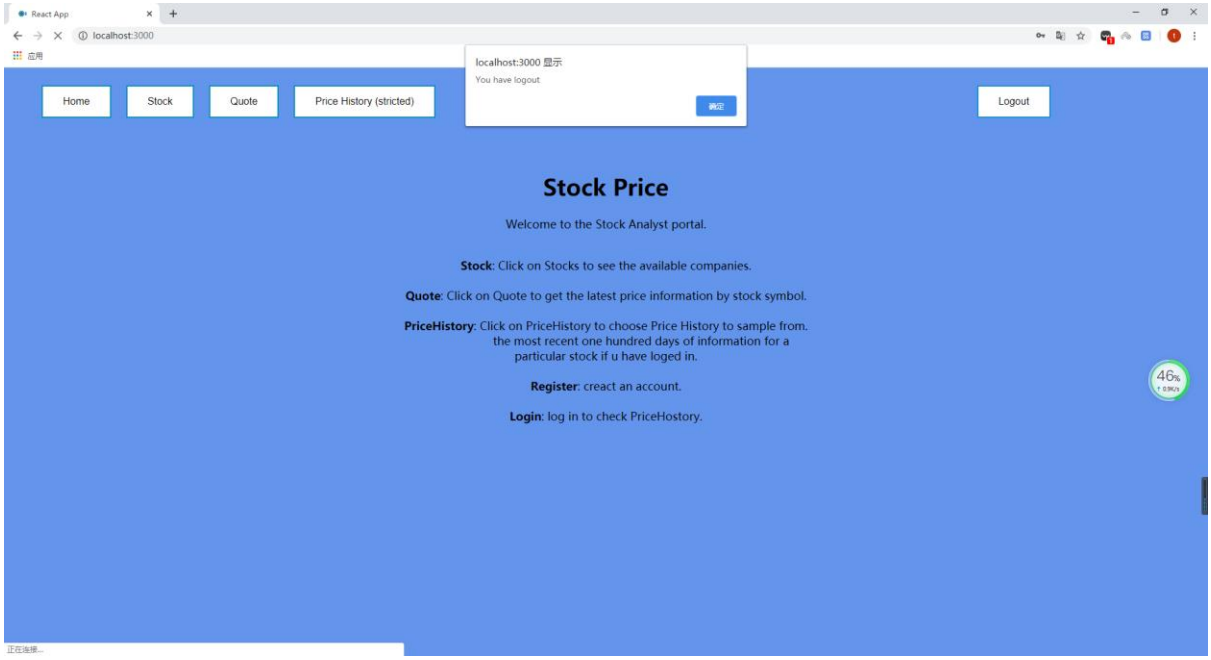
/user/login

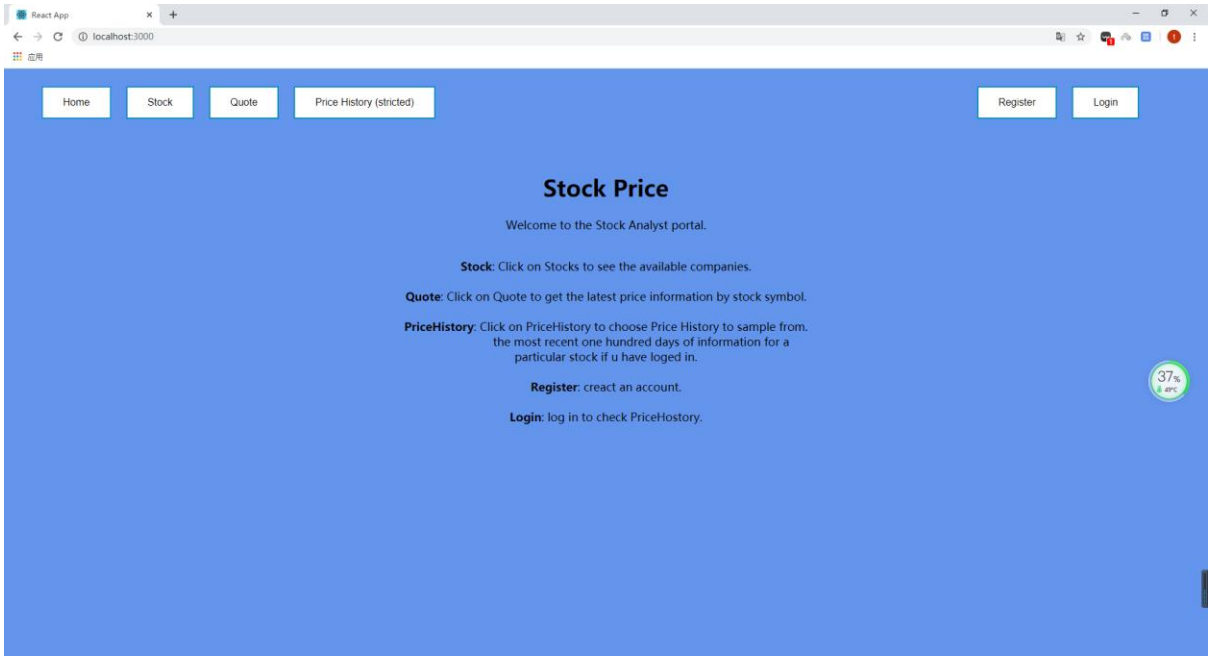
In the login POST, it links to my login page (‘/login’). When I click ‘login’ button, it will move on to ‘login’ section. please see picture below. We could see it is moving on a new page, which is ‘/login’. That is my login layout. It is simple but useful. On the left upper corner, there is button. Every time click it, it will return home page (‘/’). In addition, bellowing register button there is a link. If I click it, it will move on register section (‘/register’ ----if user haven’t had an account). Now, I am going to insert email and password that I just registered.



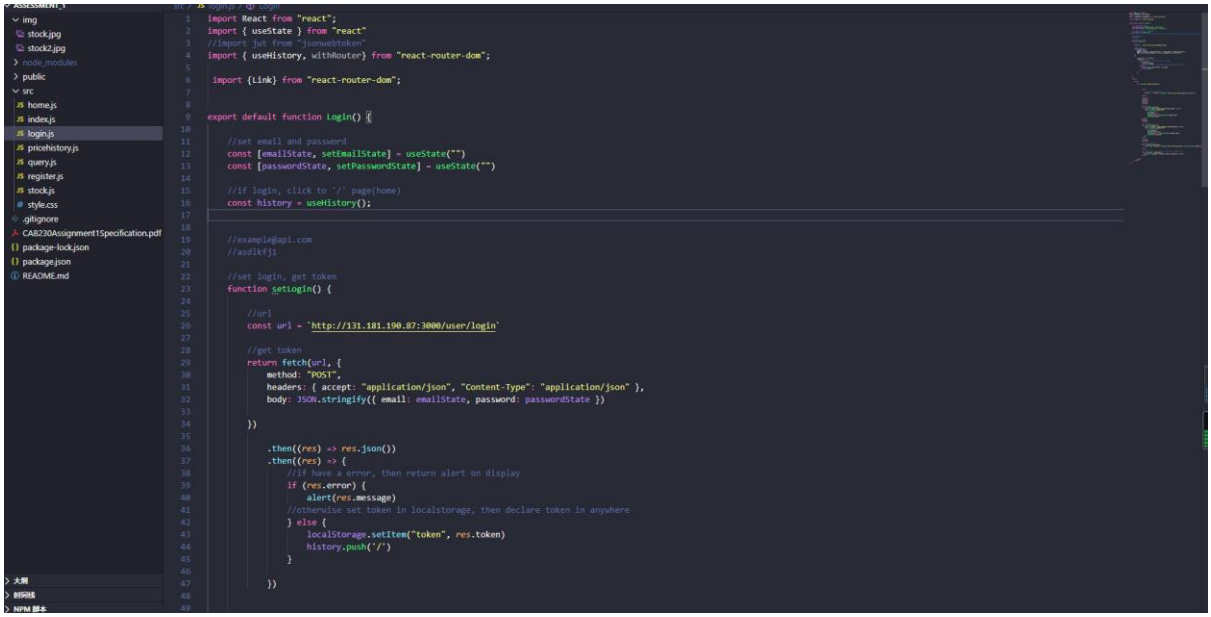
We could see that it is moving on home page (‘/’). It also has a new button on the right upper corner (Logout). When I click the logout. It will return home page with login and register.





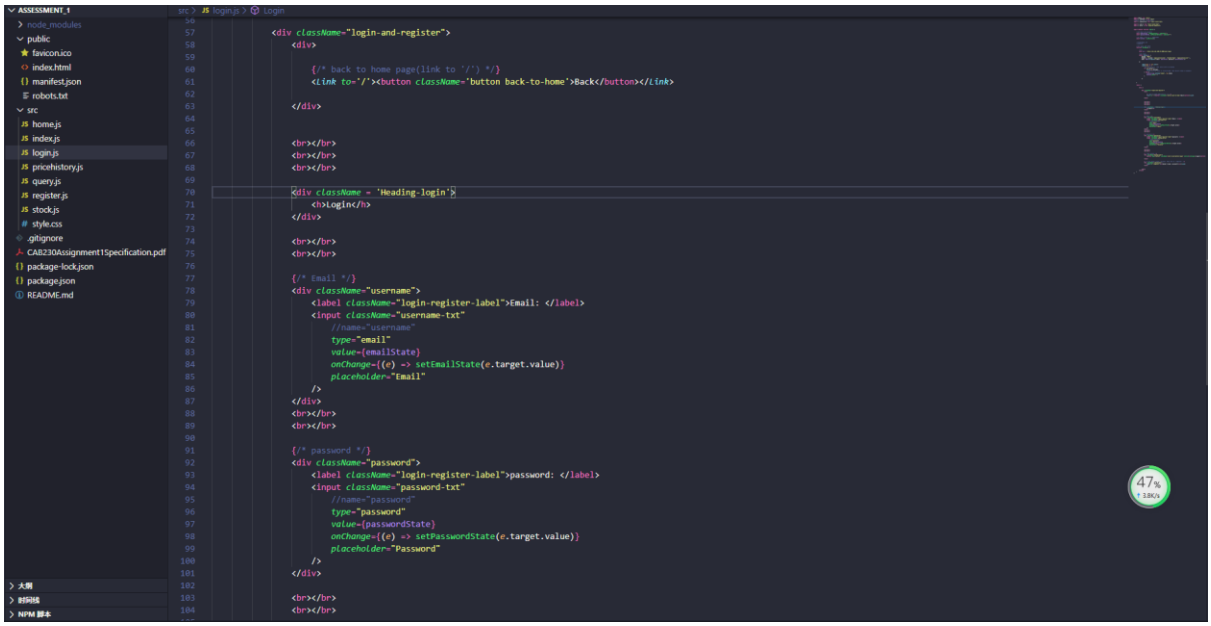


Moving on my login code, it is quite simple. I basically follow JWT worksheet and watch several useful videos on the YouTube to achieve that. useHistory is if user registered, it will move on to home page. It is similar with ‘Link’.



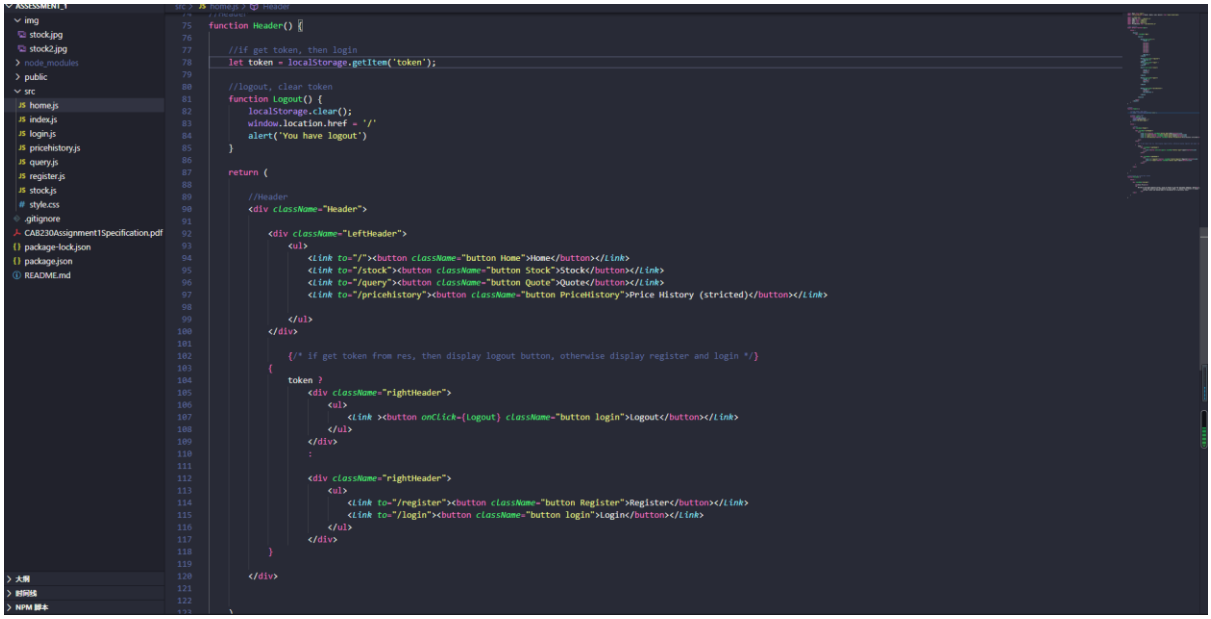
```
import React from "react";
import { useState } from "react"
//import jwt from "jsonwebtoken"
import { useHistory, withRouter} from "react-router-dom";

 import {Link} from "react-router-dom";


export default function Login() {

    //set email and password
    const [emailState, setEmailState] = useState("")
    const [passwordState, setPasswordState] = useState("")

    //if login, click to '/' page(home)
    const history = useHistory();


    //example@api.com
    //asdlkfj1

    //set login, get token
    function setLogin() {

        //url
        const url = `http://131.181.190.87:3000/user/login`

        //get token
        return fetch(url, {
            method: "POST",
            headers: { accept: "application/json", "Content-Type": "application/json" },
            body: JSON.stringify({ email: emailState, password: passwordState })

        })

            .then((res) => res.json())
            .then((res) => {
                //if have a error, then return alert on display
                if (res.error) {
                    alert(res.message)
                //otherwise set token in localstorage, then declare token in anywhere
                } else {
                    localStorage.setItem("token", res.token)
                    history.push('/')
                }

            })
```

Below code it is login layout. I wrote css code my style.css.



```
            <div className="login-and-register">
                <div>

                    {/* back to home page(link to '/') */}
                    <Link to='/'><button className='button back-to-home'>Back</button></Link>

                </div>


                <br></br>
                <br></br>
                <br></br>

                <div className = 'Heading-login'>
                    <h>Login</h>
                </div>

                <br></br>
                <br></br>

                {/* Email */}
                <div className="username">
                    <label className="login-register-label">Email: </label>
                    <input className="username-txt"
                        //name="username"
                        type="email"
                        value={emailState}
                        onChange={(e) => setEmailState(e.target.value)}
                        placeholder="Email"
                    />
                </div>
                <br></br>
                <br></br>

                {/* password */}
                <div className="password">
                    <label className="login-register-label">password: </label>
                    <input className="password-txt"
                        //name="password"
                        type="password"
                        value={passwordState}
                        onChange={(e) => setPasswordState(e.target.value)}
                        placeholder="Password"
                    />
                </div>

                <br></br>
                <br></br>
```

Now, it is my home page. In the function ‘Header’. firstly, it will check whether return token from res. If it gets token from res. Then it will display the first ‘rightHeader’ (Logout button). Otherwise it will return second ‘rightHeader’ (Register and Login buttons). ‘Logout’ function is going to clear the token and then link home page without token.


ASSESSMENT_1
img
stock.jpg
stock2.jpg
node_modules
public
src
home.js
index.js
login.js
pricehistory.js
query.js
register.js
stock.js
style.css
.gitignore
CAB230Assignment1Specification.pdf
package-lock.json
package.json
README.md
大纲
时间线
NPM 脚本
function Header() {
//if get token, then login
let token = localStorage.getItem('token');
//logout, clear token
function Logout() {
localStorage.clear();
window.location.href = '/'
alert('You have logout')
}
return (
//Header

<Link to="/"><button className="button Home">Home</button></Link>
<Link to="/stock"><button className="button Stock">Stock</button></Link>
<Link to="/query"><button className="button Quote">Quote</button></Link>
<Link to="/pricehistory"><button className="button PriceHistory">Price History (stricted)</button></Link>

{/* if get token from res, then display logout button, otherwise display register and login */}
{
token ?

<Link ><button onClick={Logout} className="button login">Logout</button></Link>

:

<Link to="/register"><button className="button Register">Register</button></Link>
<Link to="/login"><button className="button login">Login</button></Link>

}

)

Modules used

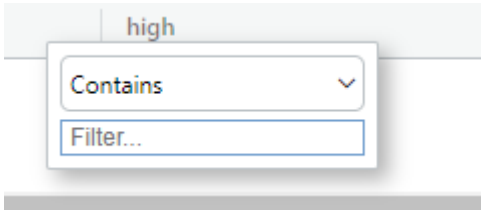
This is just a list of the external modules that you have used. You need not specify core React modules. In each case, ***we want the name, a brief description, and a link to the docs at npm or github or wherever.*** The first one is ag-grid-react as most people will be using this. Just copy that style and add more as necessary.

Module 1: Ag-grid-react // ag-grid-community

Link: 1. https://www.ag-grid.com/react-grid/ (official website)

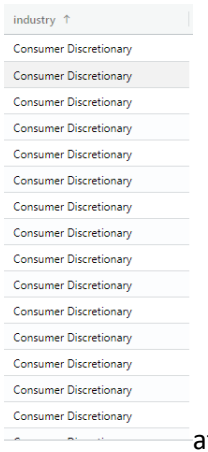
2. https://www.npmjs.com/package/ag-grid-react (installation)
3. https://www.youtube.com/watch?v=6PA45adHun8 (YouTube resource (installation) and Sorting and filtering usage)
4. sorting and filtering also learn from ---- BlackBoard ---- (DataTablesinReact)

Basically, I designed my table layout from resource above. They are very good resource on me. Now, I list several features that I used in my table.



filtering

before sorting



after sorting

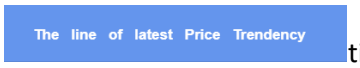
Module 2: react-chartjs-2

Link: 1. https://www.npmjs.com/package/react-chartjs-2 (installation)

2. https://www.youtube.com/watch?v=Ly-9VTXJlnA&t=735s (YouTube resource (beginning and style))
3. https://www.youtube.com/watch?v=AcoUu3bgKgM (YouTube resource (Option and style))
4. https://www.chartjs.org/docs/latest/getting-started/usage.html (chartjs---usage and Introduction)
5. https://www.chartjs.org/docs/latest/axes/cartesian/linear.html (chartjs----the usage of Linear scale)

Basically, I designed my line-chart layout from resource above. They are very good resource on me.

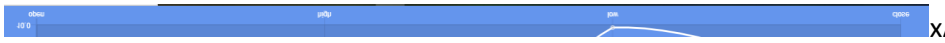
I list several features that I used in my line-chart below. I usually change their font colour to match my background colour.



title

Label


yAxes


xAxes

Application Design

Navigation and Layout

Here we want you tell us – again in a paragraph or two - about the design process for the site, the choices that you have made and any alternatives considered. Tell us about the choices you have made for navigation – the menu items and the flow between the screens – and the layout. The idea here is that you will tell us in this section how the application is used and we can then use this when we are looking at the technical description describing how it is built.

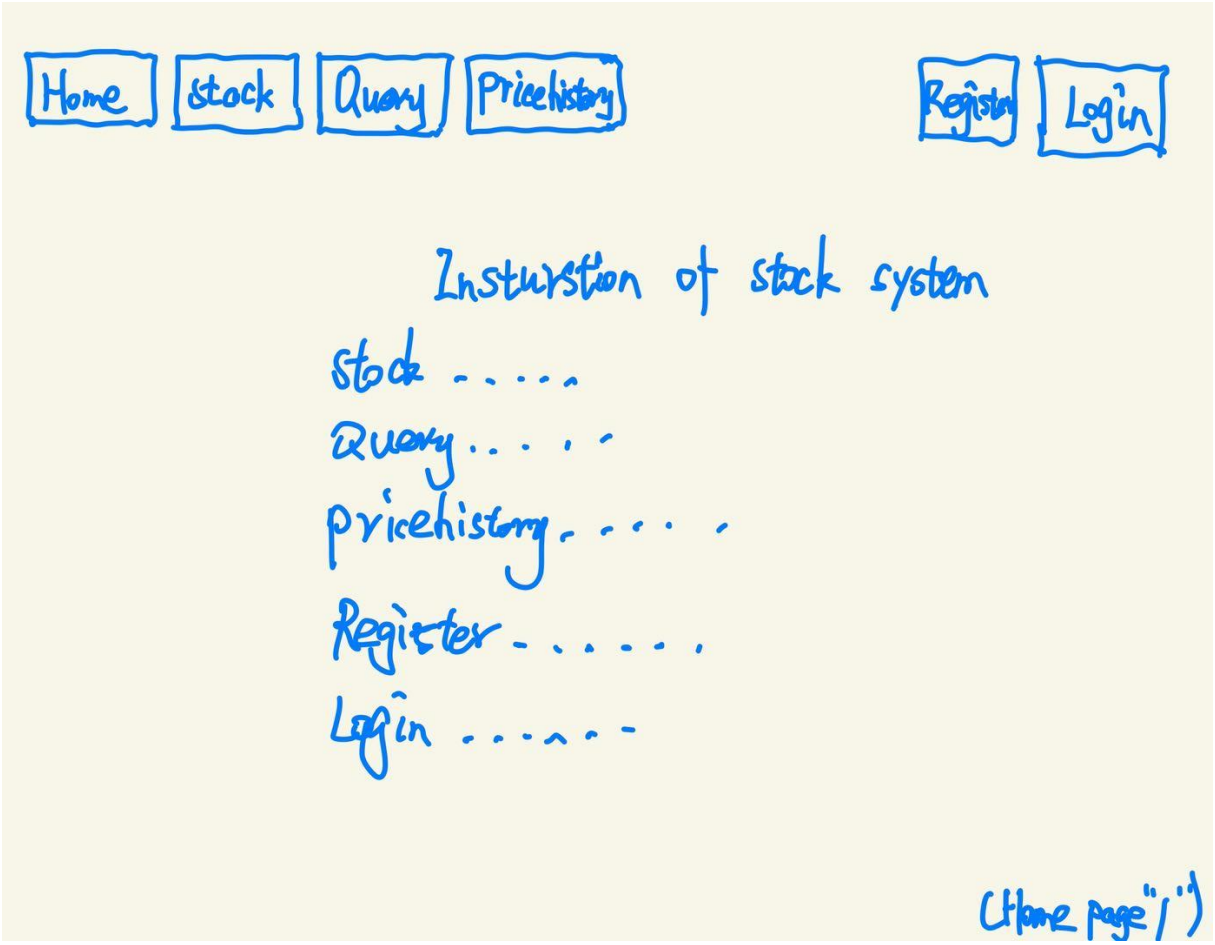
Basically, the layout of my stock system is similar with layout of assignment specification. Clicking different button will link to relatively page. More detail please see my sketches.

Show us some of your design mock-ups – scan some of your sketches – and show us how your application flows from one screen to another. ***Comment on the usability of your design – are there compromises that make it awkward to use? How might you improve those? Note that our expectations in respect of usability are basic – you can be critical here without losing lots of marks.*** We are looking for you to highlight the good and the bad aspects of your design and layout.

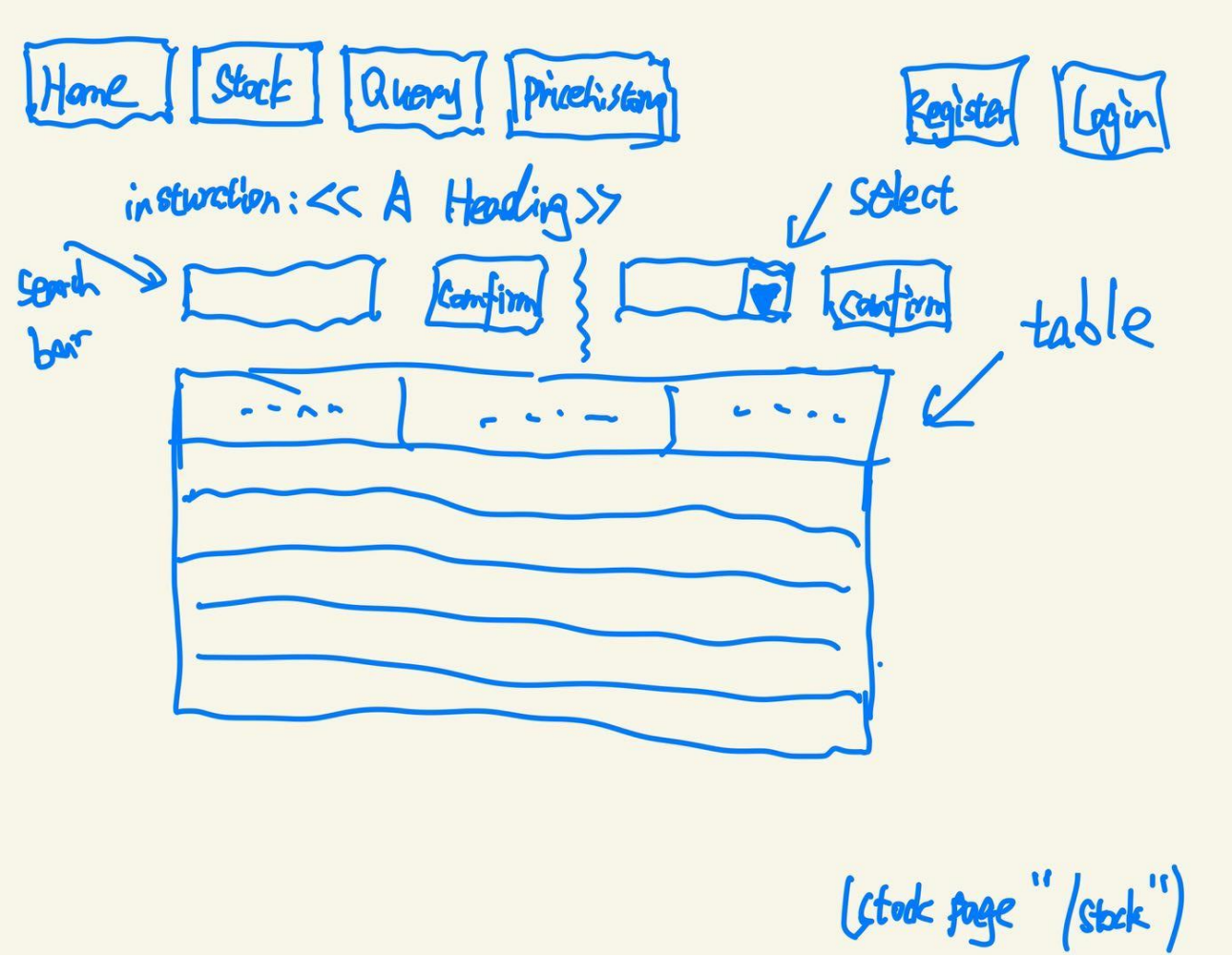
Please see the CRA and podcast 2 for more guidance, but your application should be logically laid out and the widgets should be well chosen to suit the data that they control. This will ***not*** be a high bar to clear.

Sketches:

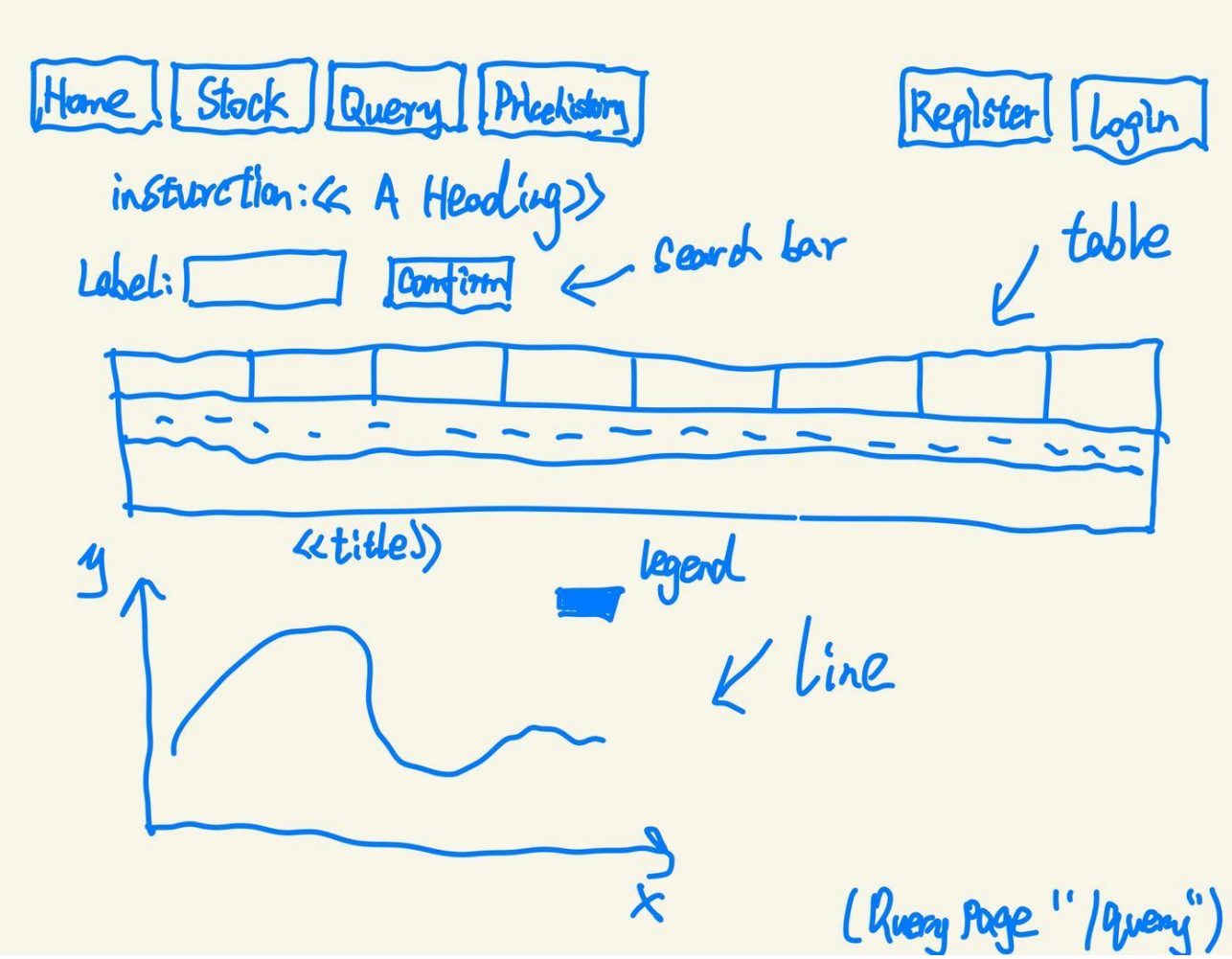
This my home page, just briefly instruction below how to use my stock system.



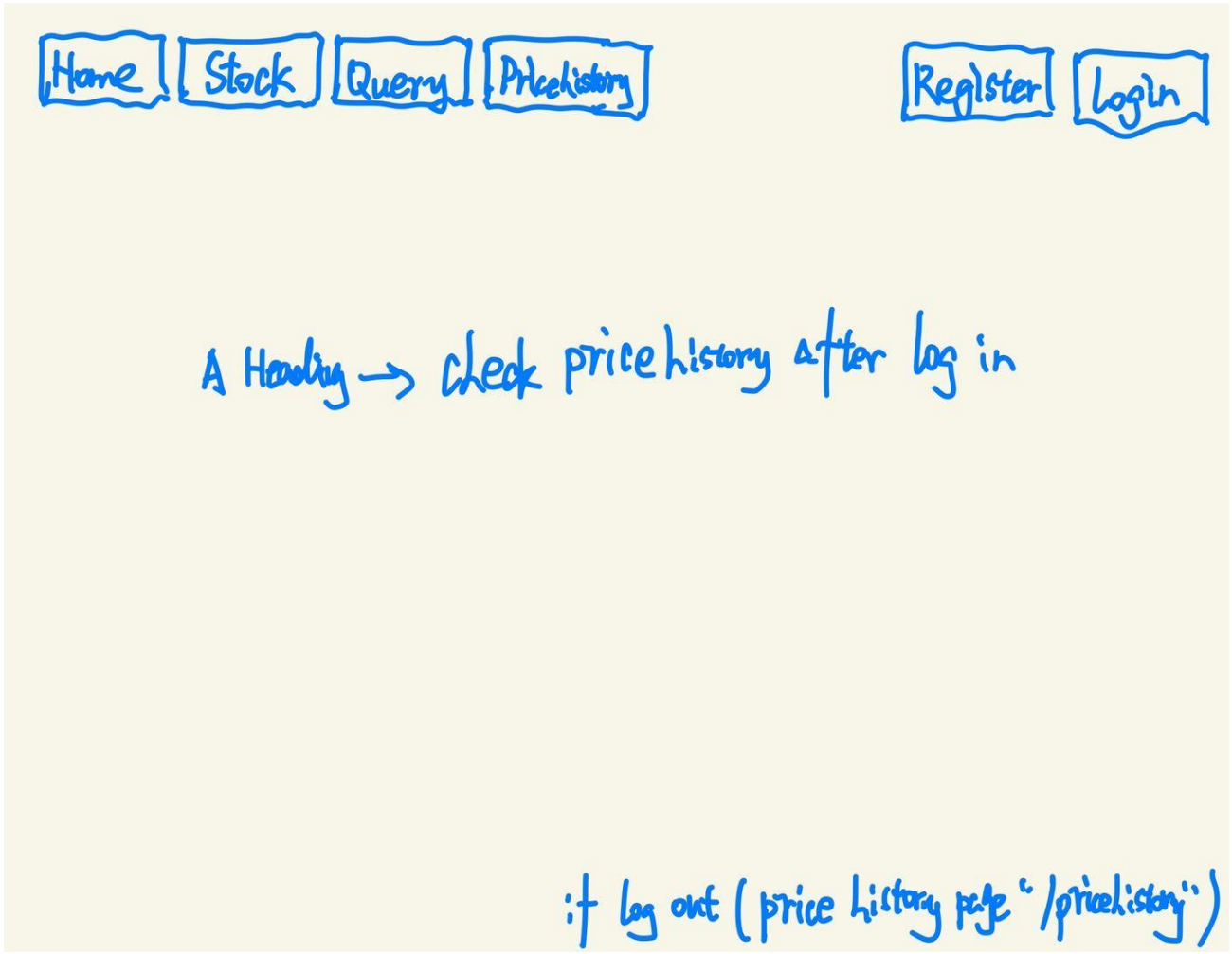
This is my stock (first Get), I want to make it happen with select. But I don't have too much time to finish it. so, I only do the search bar here.



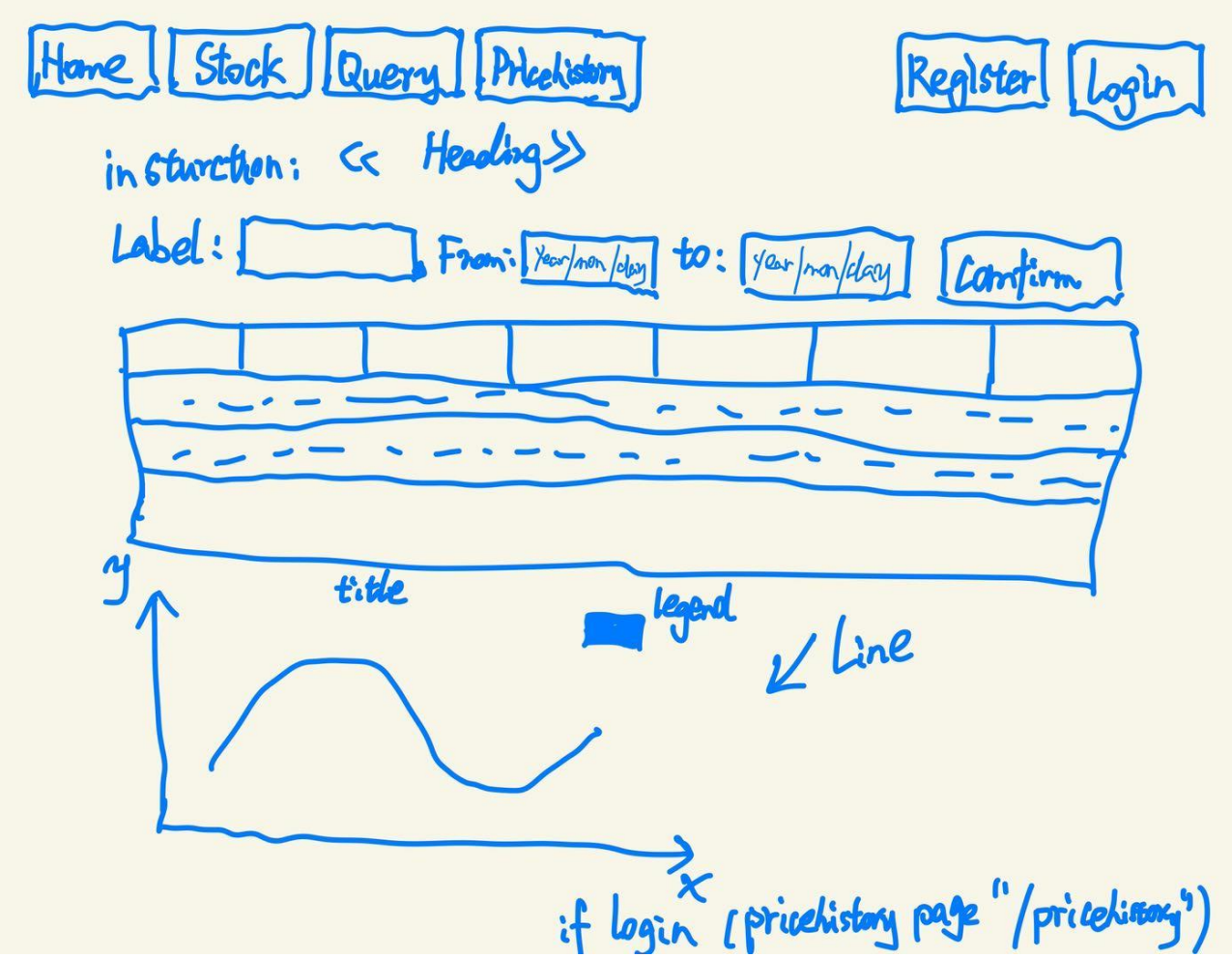
This my query page (second Get) there a search bar here. We could type there to search what I search and display on the table and line-chart.



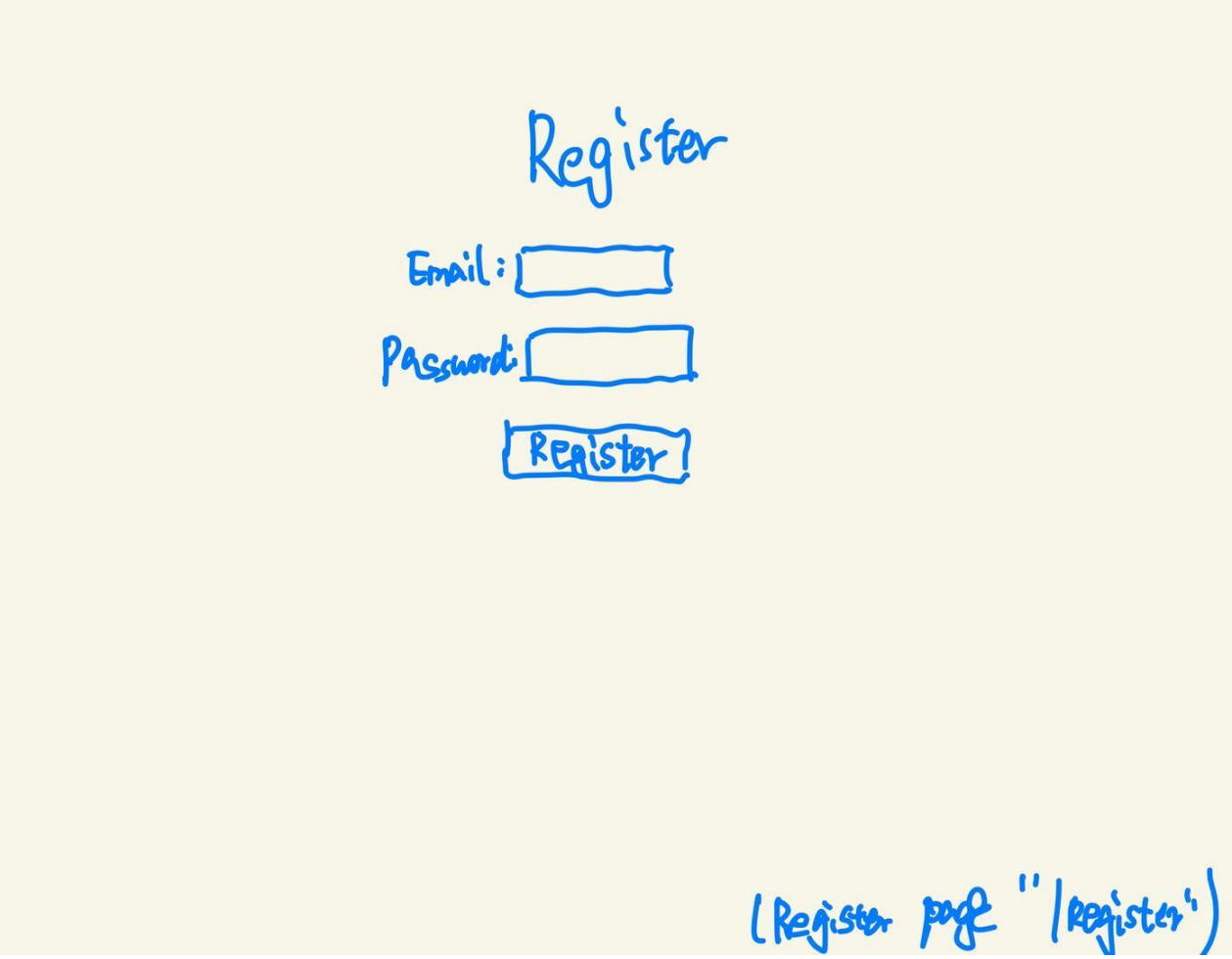
This is my price history (third Get----logout) just tell user have to login to check price history.



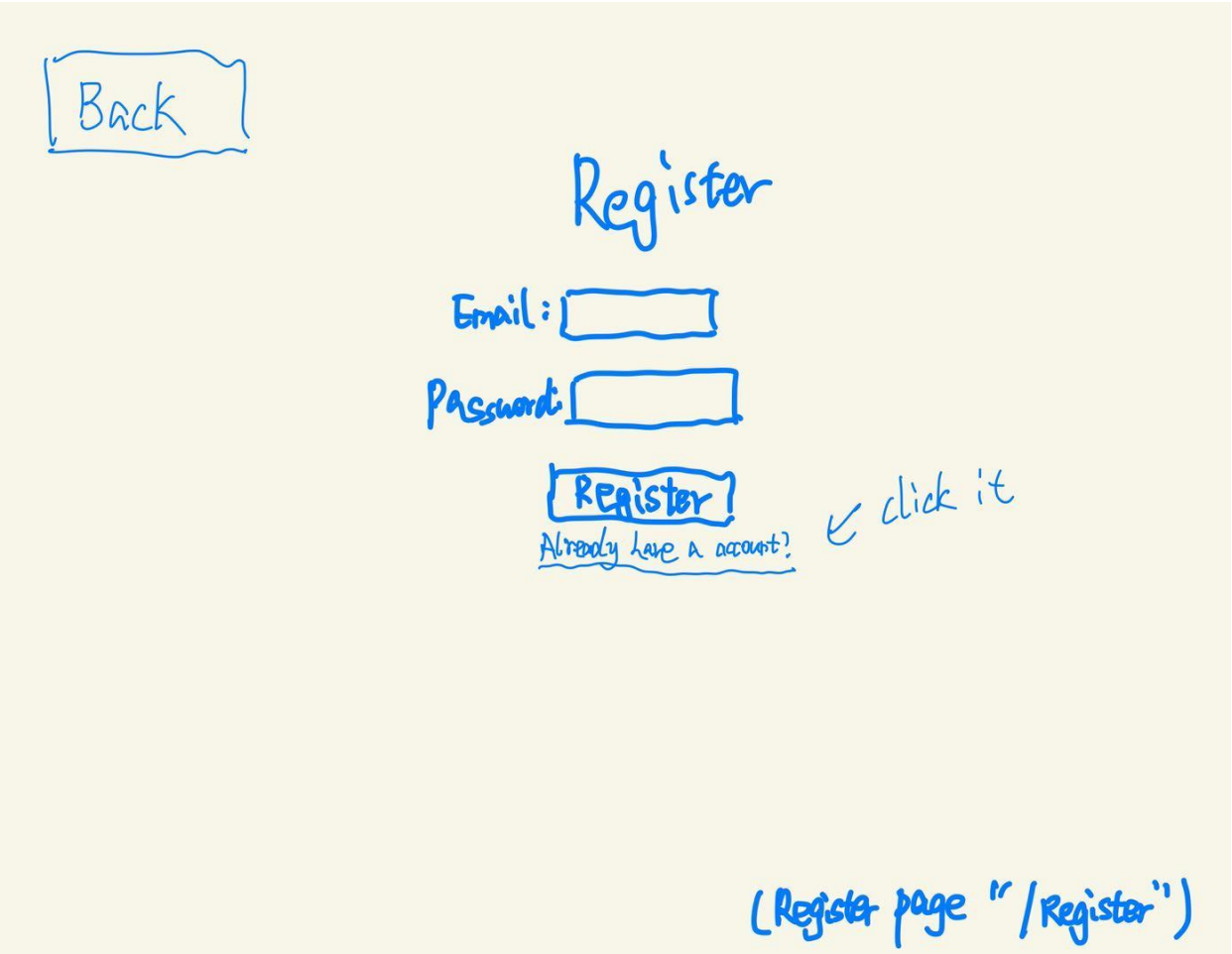
This is my price history (third Get----login) there is search bar here. Search bar is required. Date is not required. If I type on search bar it will display on table and line-chart.



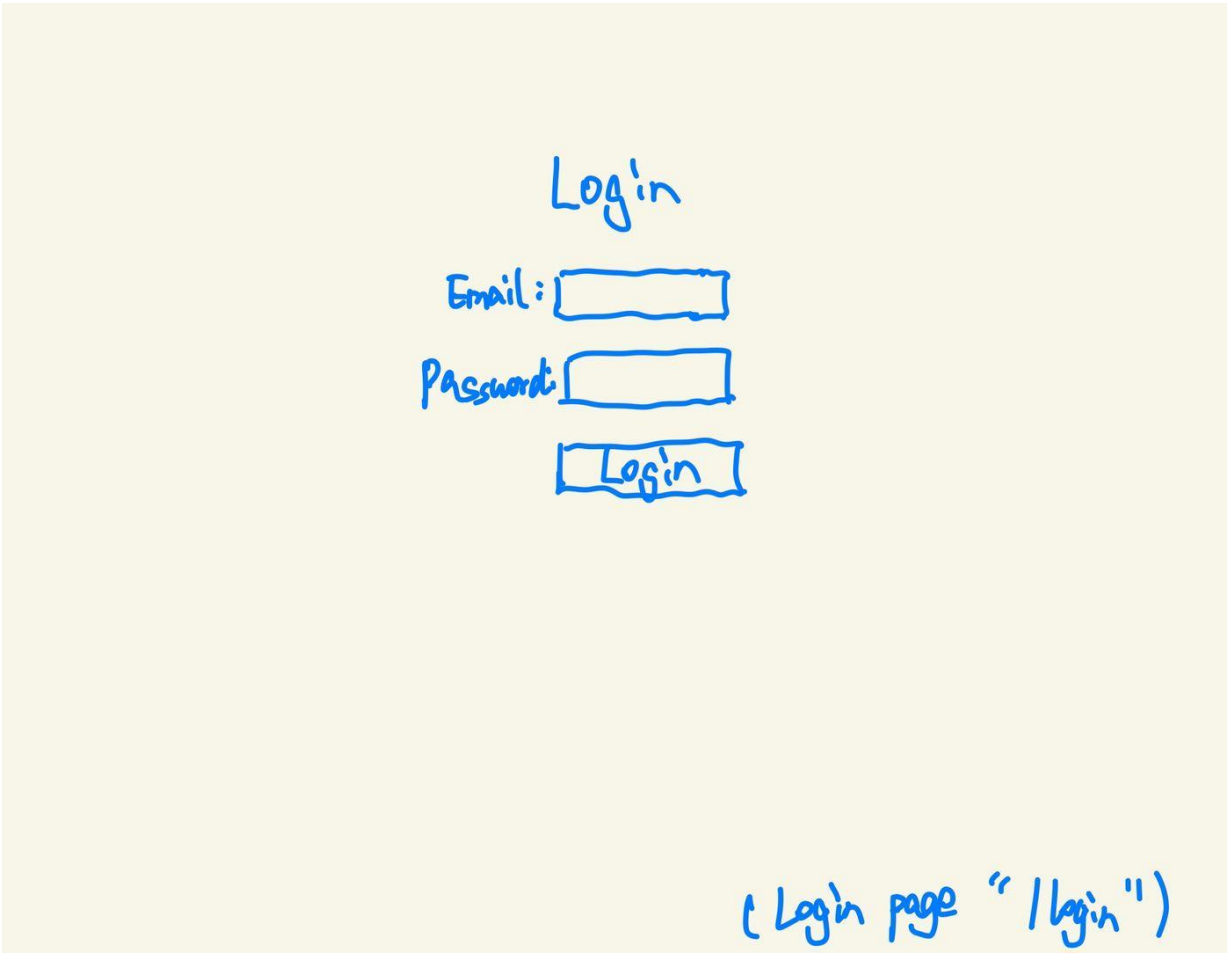
This is prototype of my register. Because when I click register button from home page, I can't go back. So I added 'back' button on the left upper corner in register page. Please see below.



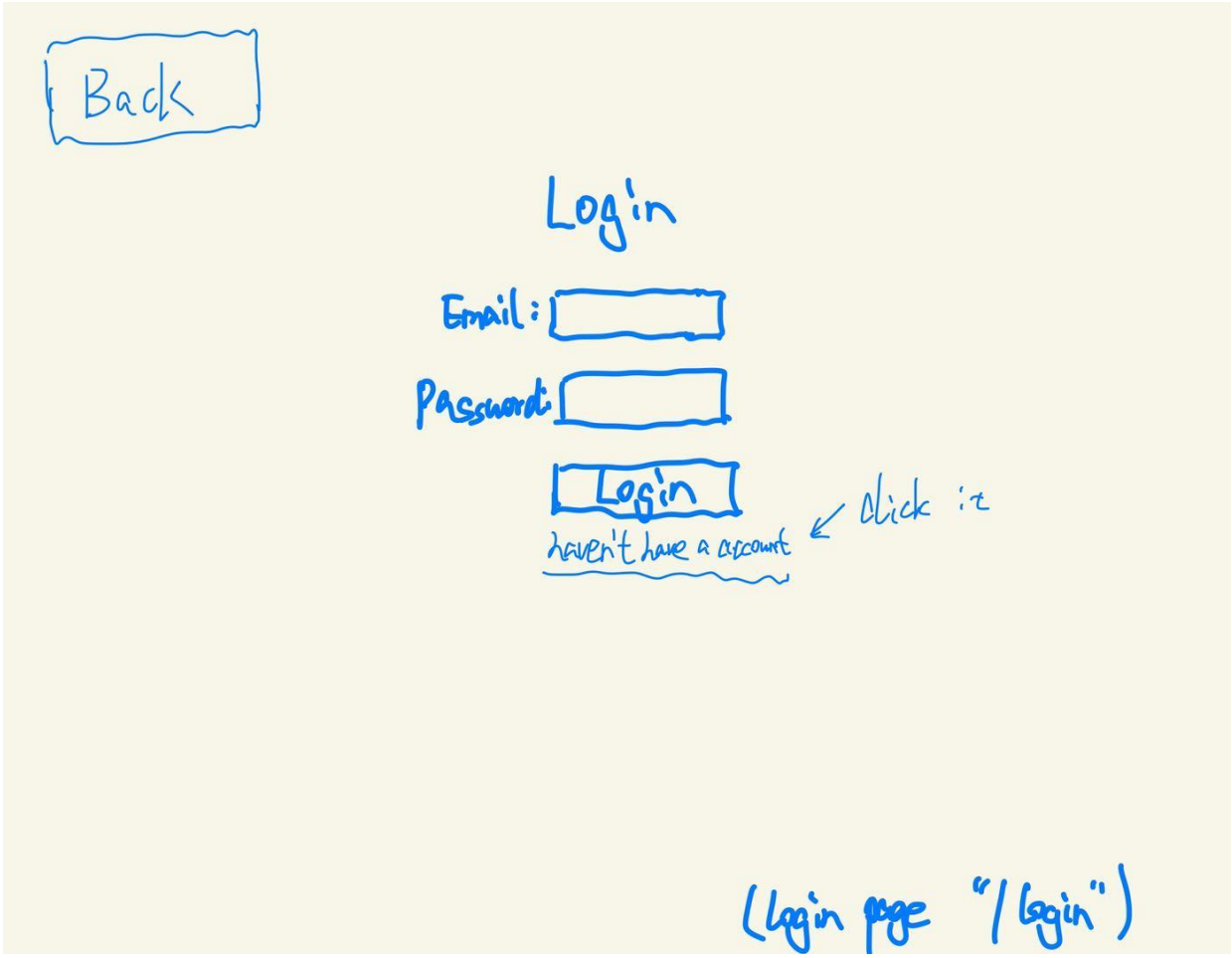
This is my final layout of register. In addition, I also add ‘already have an account?’ below register button. If I click it, it will link to login section.



This is prototype of my login. Because when I click login button from home page, I can't go back. So, I added 'back' button on the left upper corner in login page. Please see below.



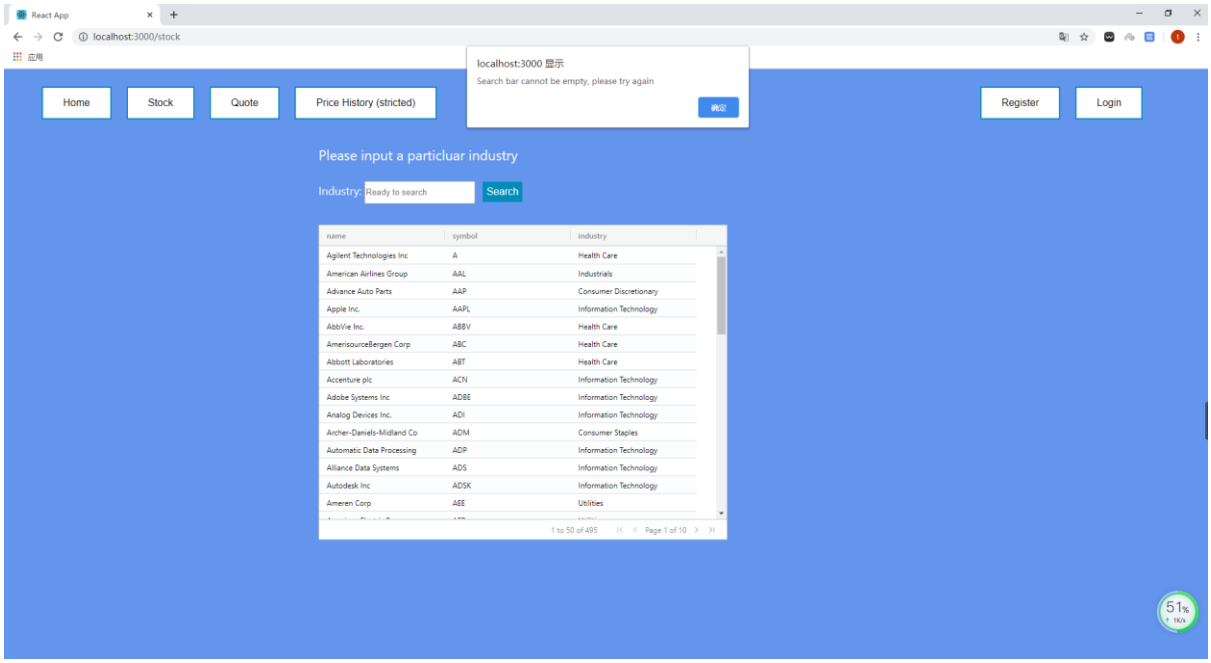
This is my final layout of login. In addition, I also add ‘haven’t have an account?’ below login button. If I click it, it will link to register section.


Back
Login
Email:
Password:
Login
haven't have a account
click it
(Login page "/login")

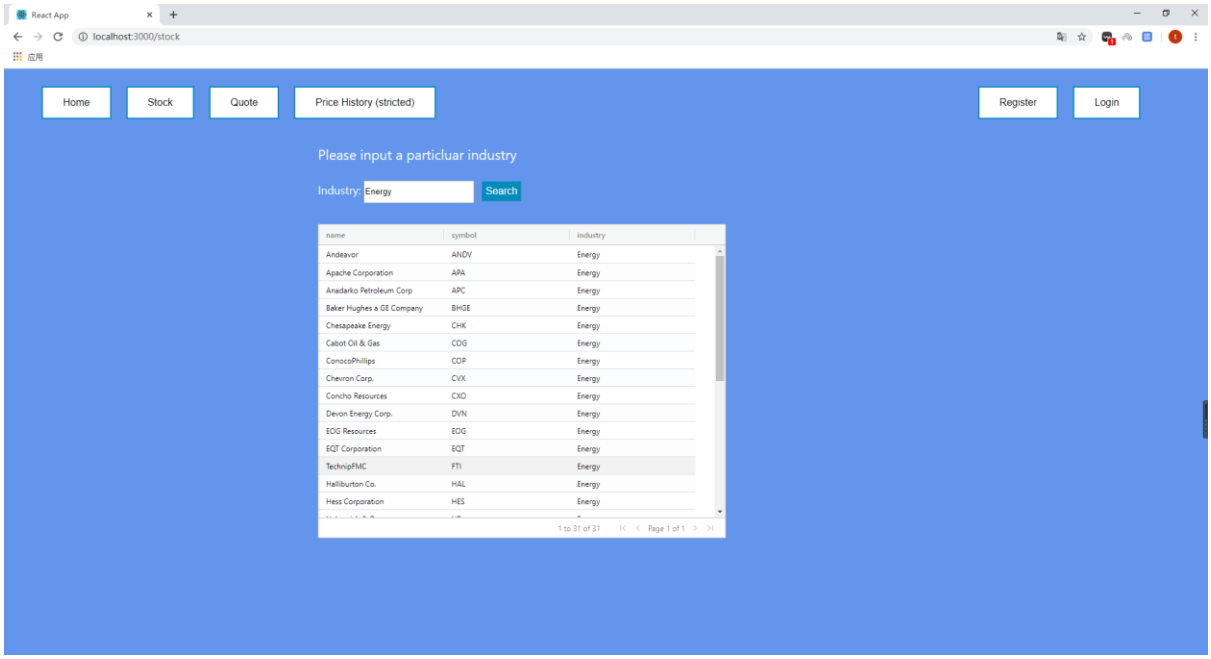
Test Plan

Stock:

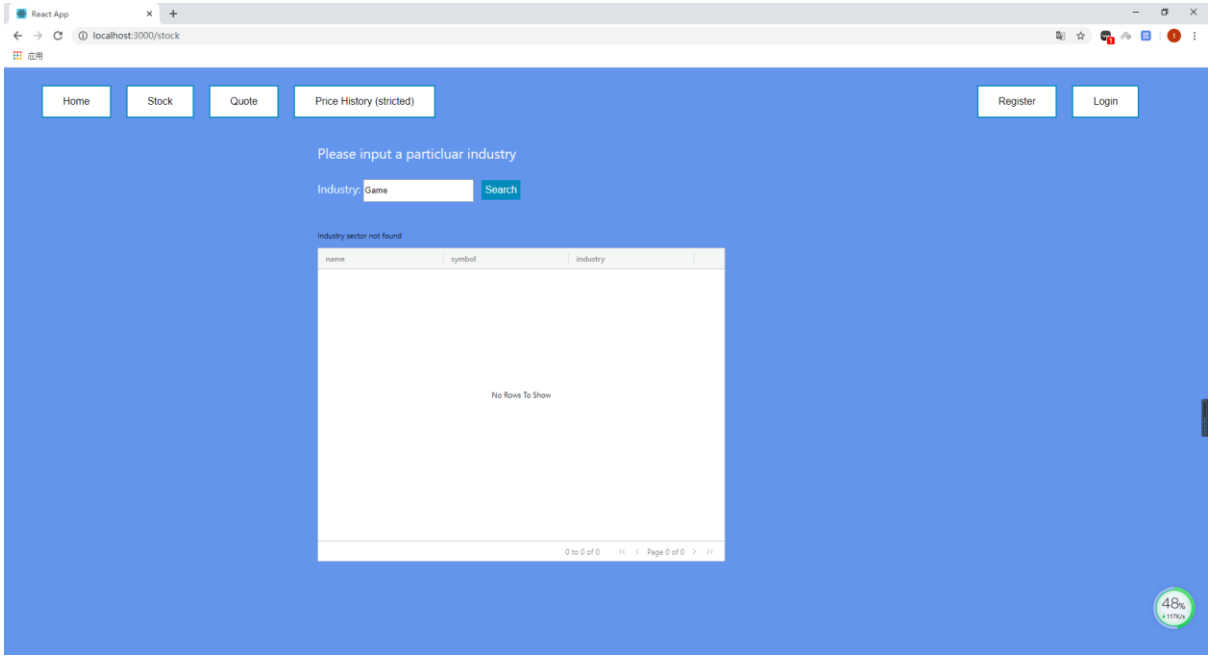
Test empty value pass



Test Industry name Pass

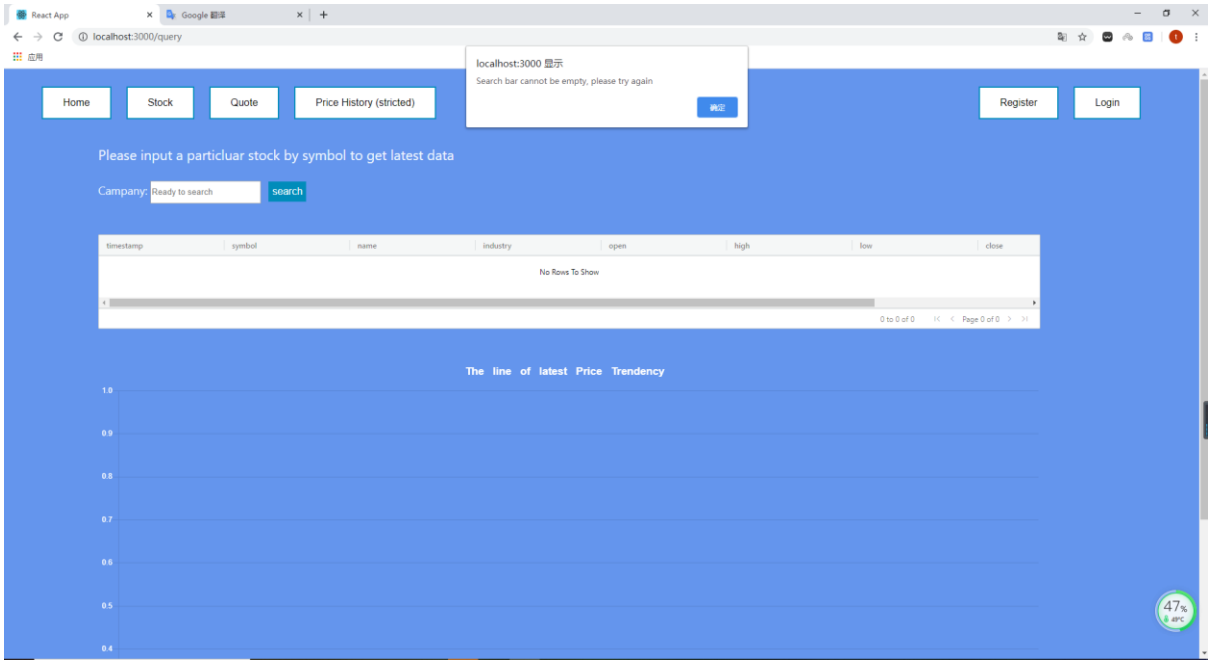


Test industry name doesn't exist Pass

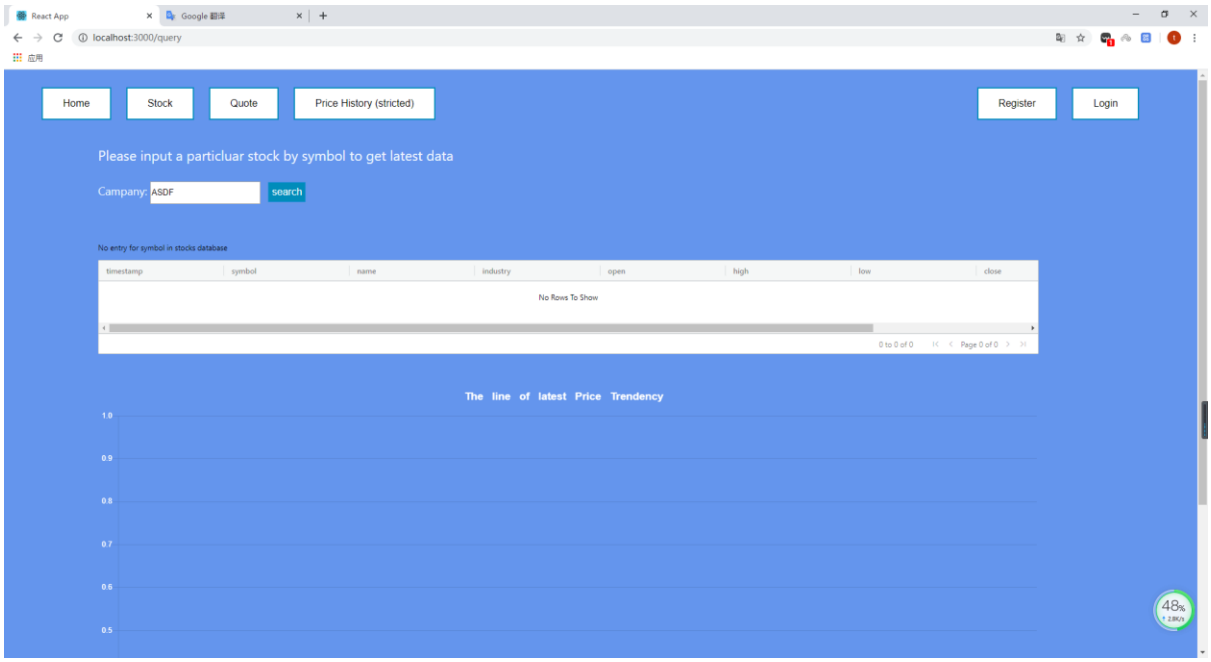


Query

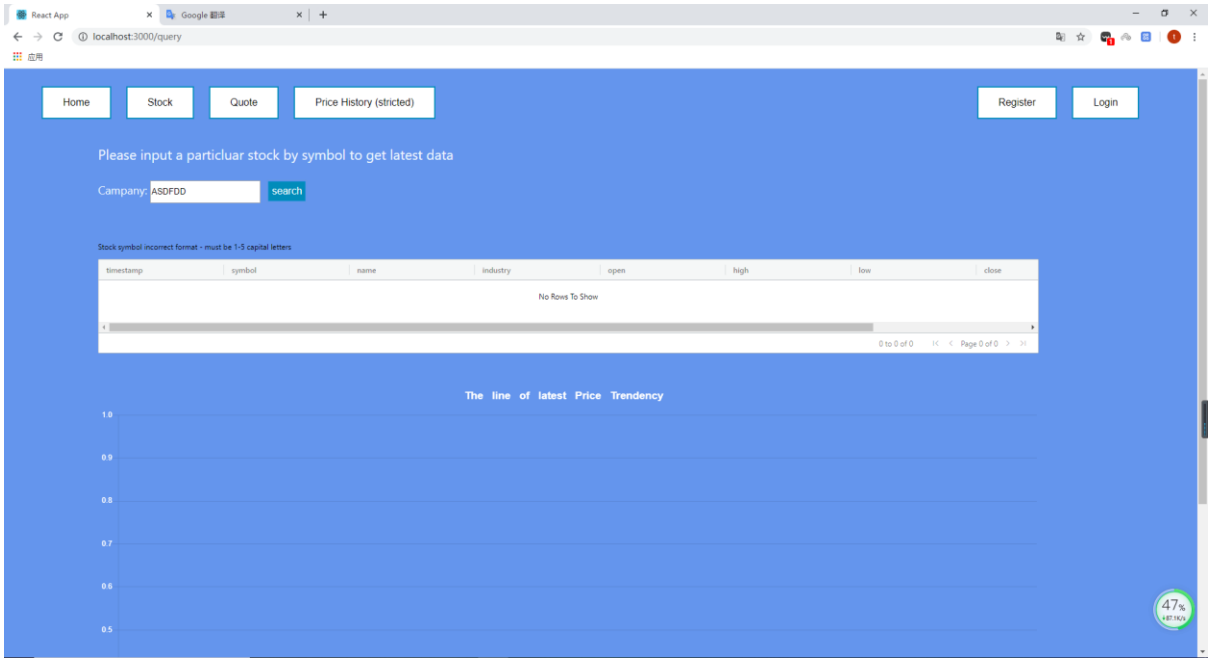
Test empty value Pass



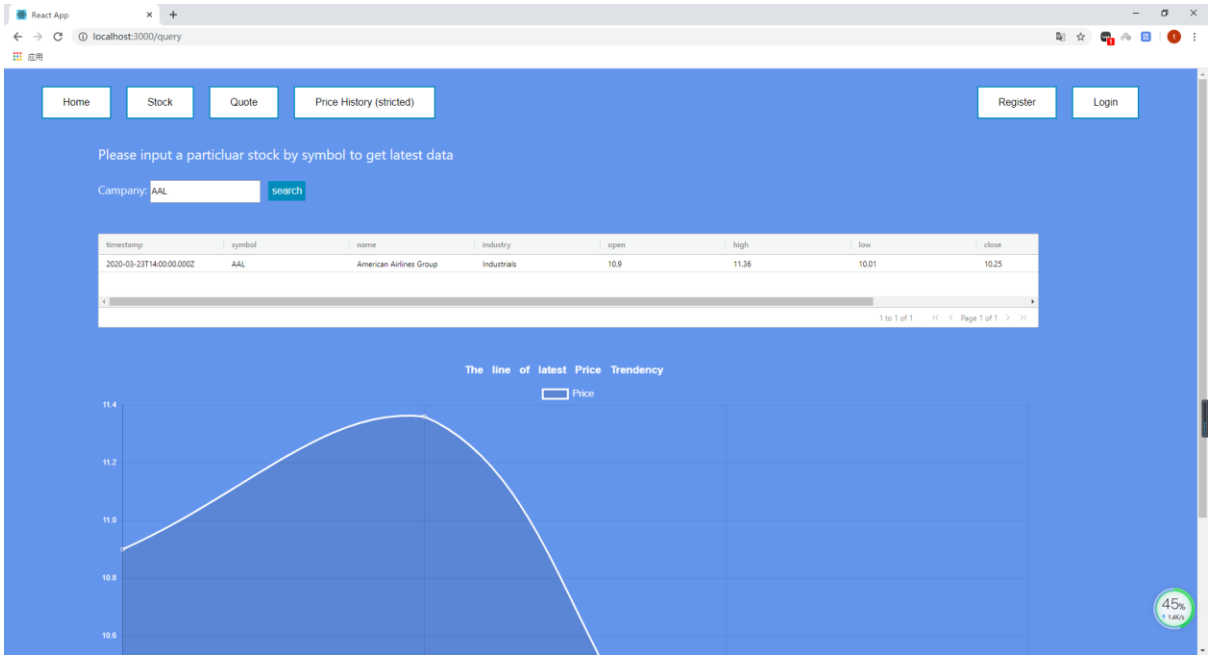
Test doesn’t exist value Pass



Test more than 5 letters Pass

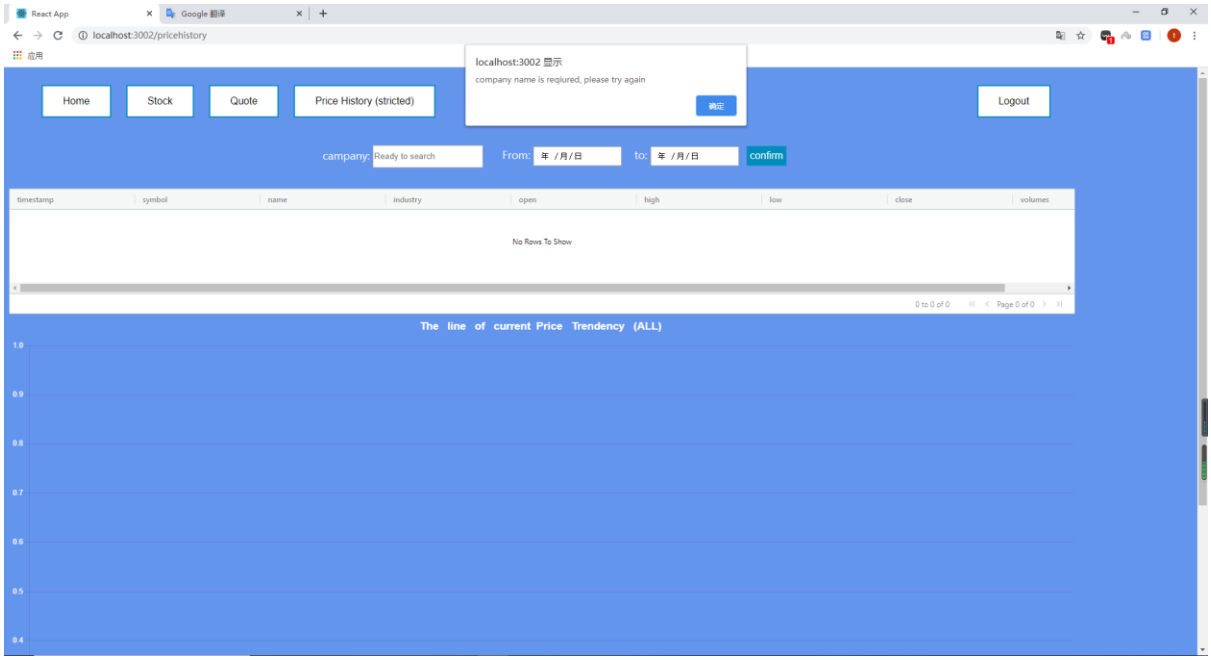


Test correct name return correct value in table and chart Pass

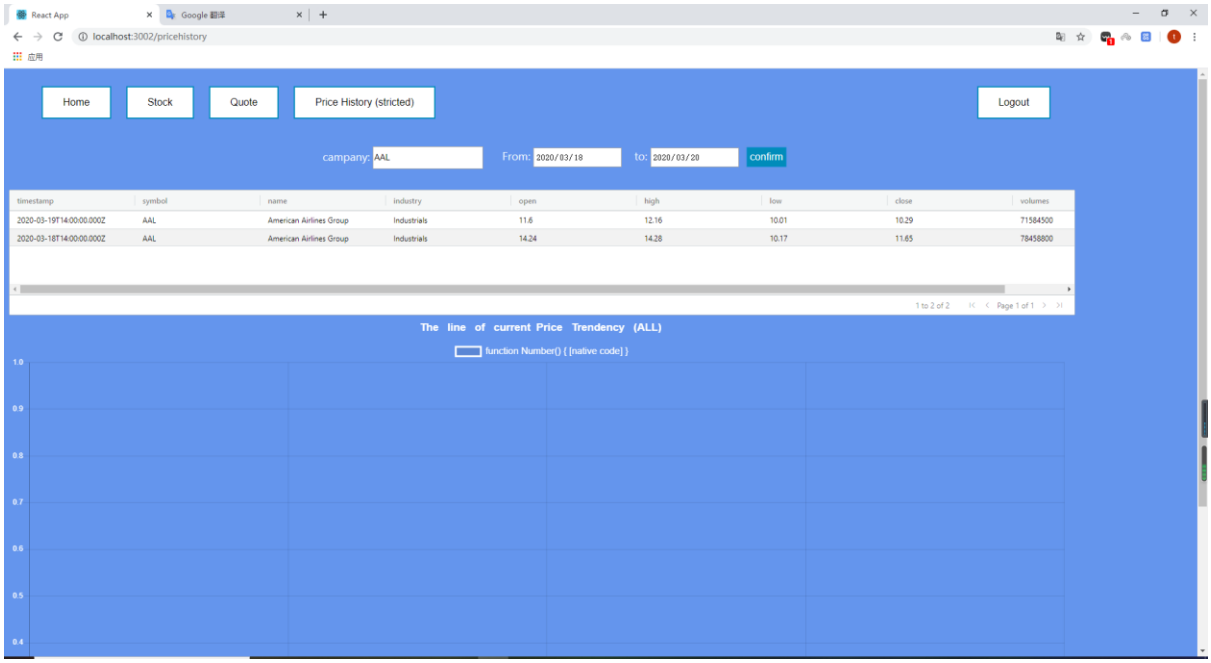


Price History:

Test empty value Pass

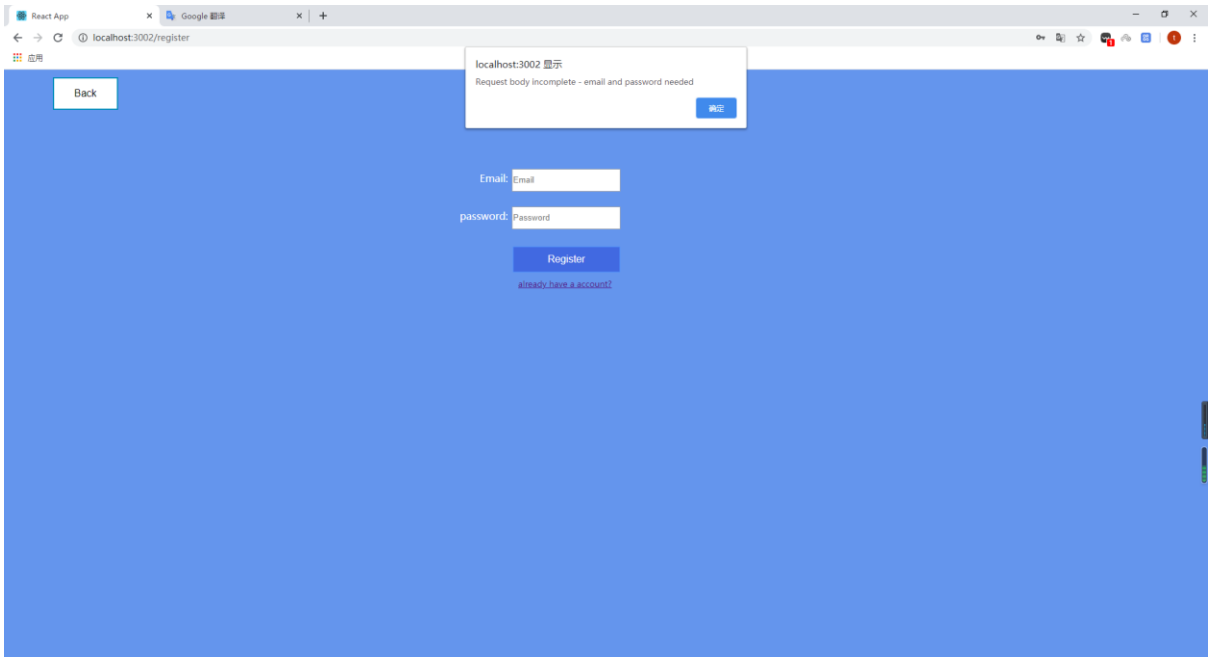


Test company with data return value in table Pass return in line chart decline

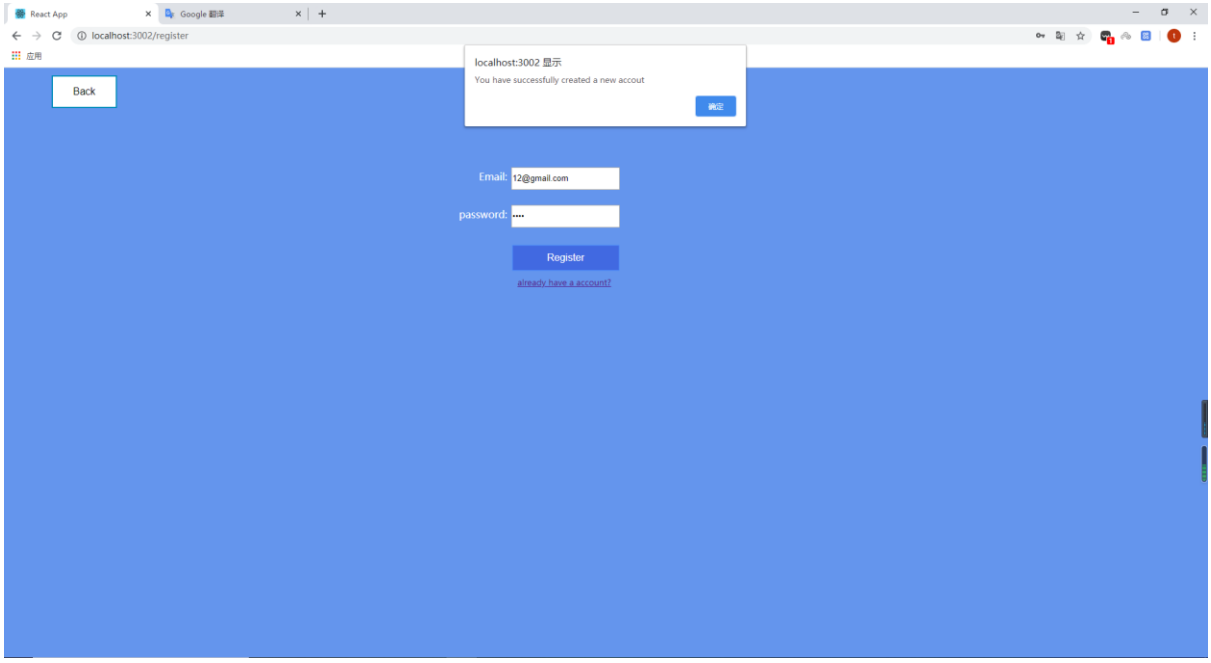


Register:

Test empty value pass

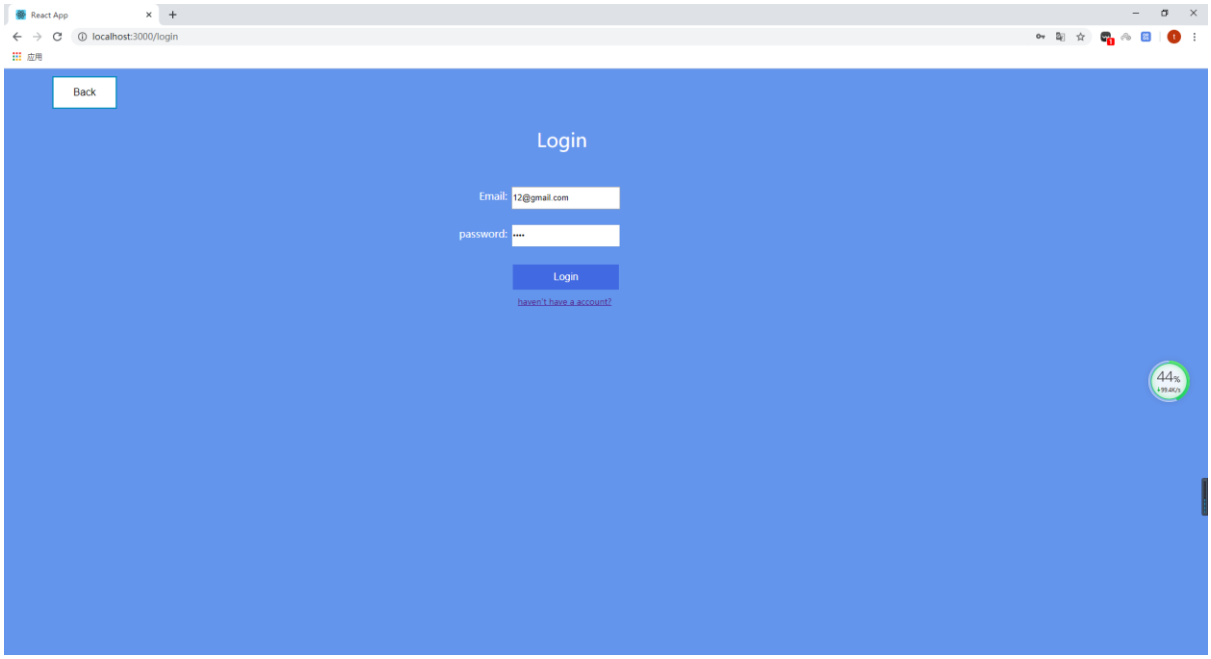


Register a real account pass

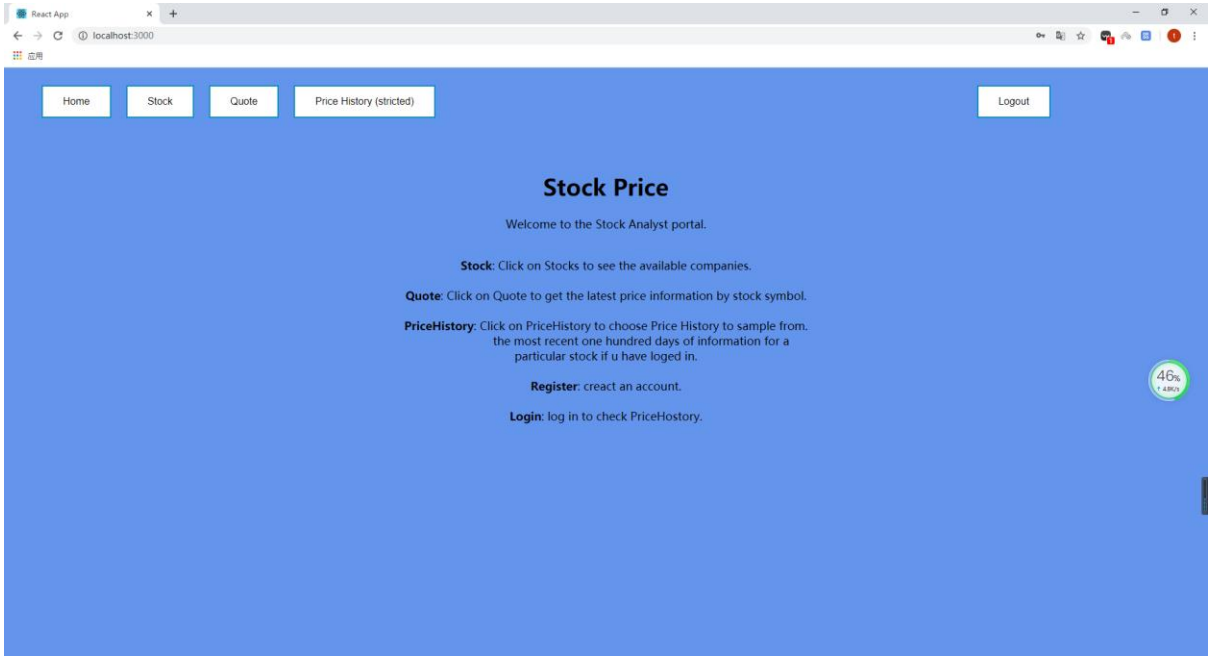


Login:

Successfully login an account just registered Pass



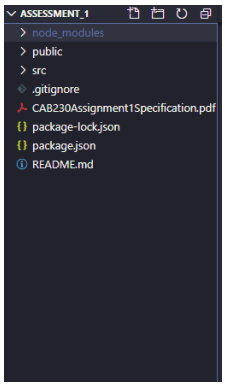
Login In Pass



Technical Description

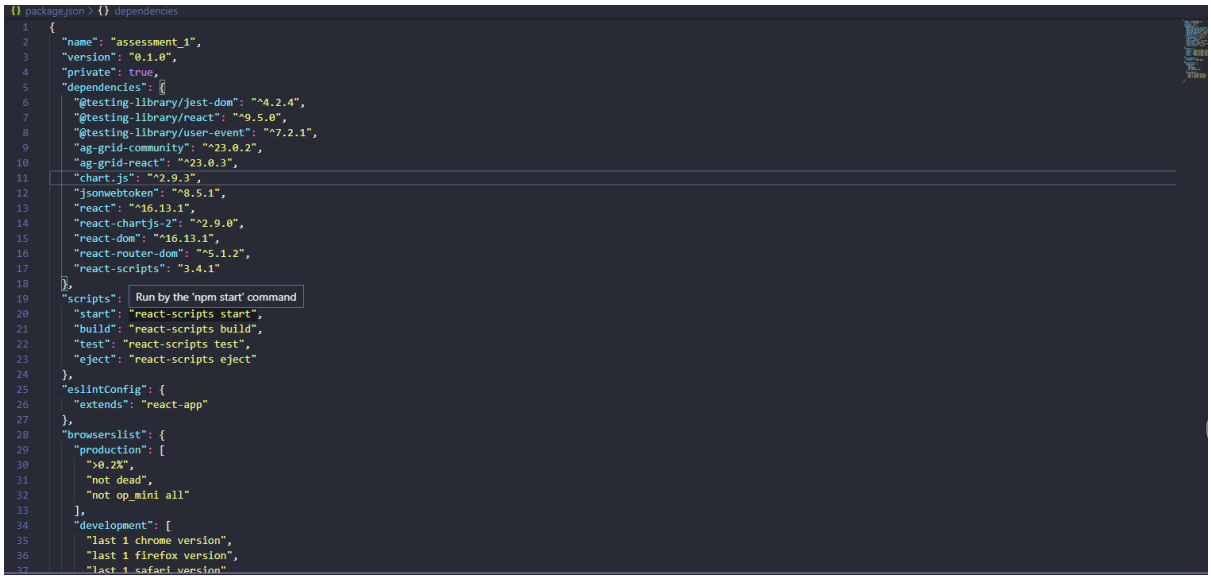
Architecture

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:

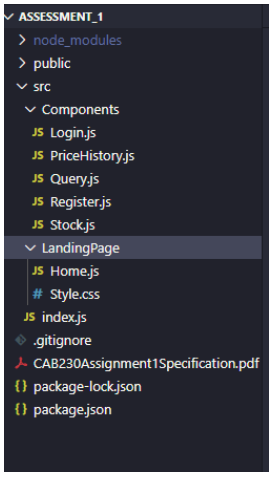


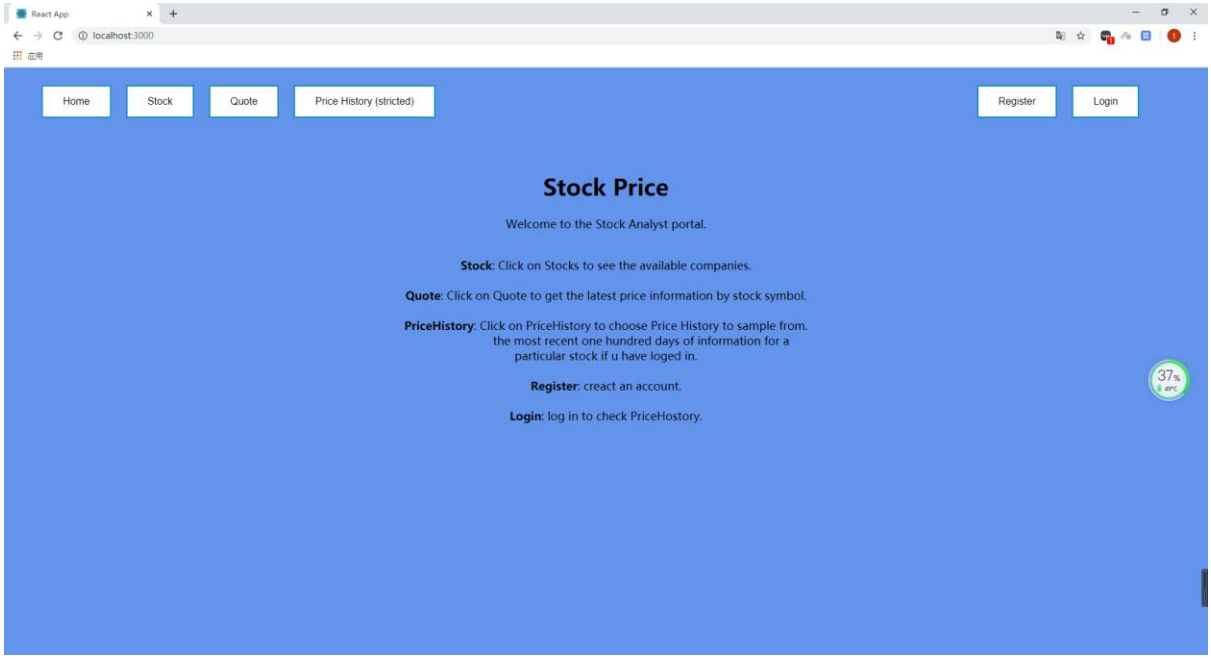
Assessment is the main folder, it including node-modules, public and src folders and assignment specification and package.json.

Package.jason is what modules that I added in there.



Dig into the src directory and tell us how you have split the responsibilities. Tell us in a sentence or two how the application is controlled, and the services supported. In the image below we see the organization for Michael's demo app. Tell us briefly about the split across these folders and why you chose it that way.





Now, we click ‘src’ folder, it includes my main js files and css file. Each js files is correspond with each menu button on the screen.

Difficulties / Exclusions / unresolved & persistent errors /

- What were your major roadblocks / how did you resolve them?

The first big roadblock is that when I am doing second GET, I just code similar with my first one (stock). but it doesn't work. So, I google it why it doesn't work. I stuck amount of time on this issue. Now, I fixed it already, I realize that the second one it returns object not array, I have to make them be array. It will display on table. secondly, login part is my second stuck part, I just follow JWT worksheet and console it, it does return the token to me. But I don't know how to link to my home page, after checking several resources on google. I use history.push make that happen. Finally, line chart is third part I stuck for a long time. I watched a lot of tutorial on Youtube for chart.js. for example: how to set dataset and how to design the layout or color etc.

- Any functionality you didn't or couldn't finish and the technical issues encountered

The function that I couldn't finish Probably is the chartjs on price history. If I select date more than 1 day, it will return an array to me. It that mean I need different line to show different day? i didn't make it happen. If I have more time, I could make that happen. So, I only make the second GET line-chart.

- Are there any outstanding bugs?

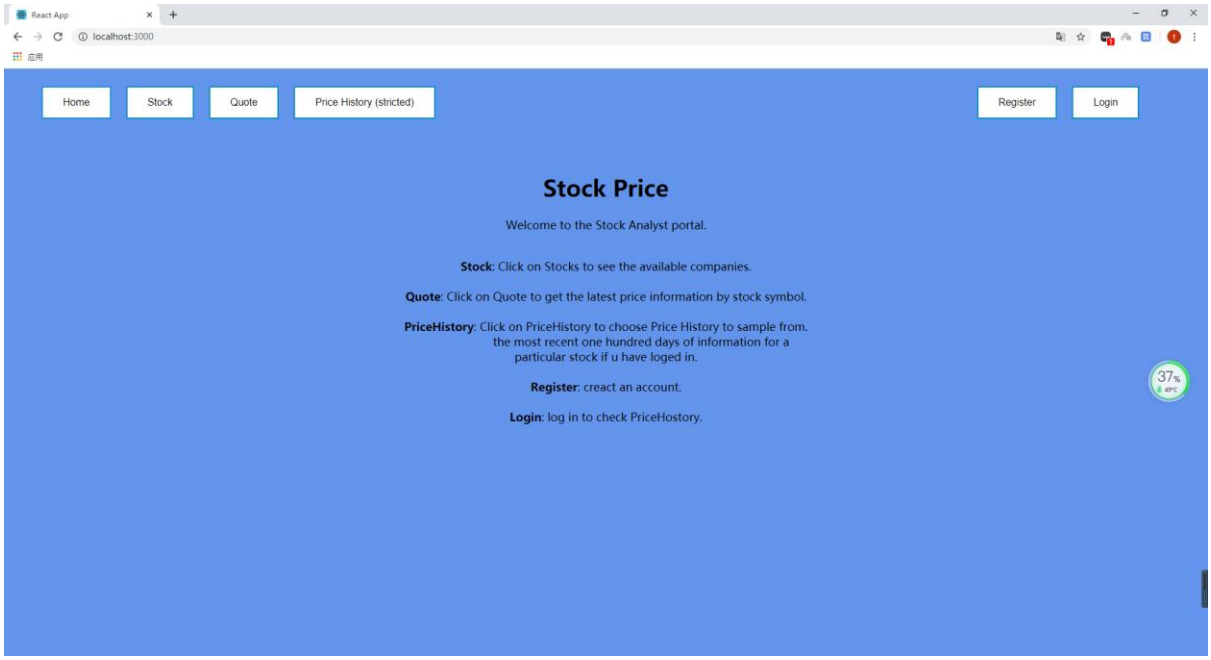
Not too much, I have already talk on other topic or question.

User guide

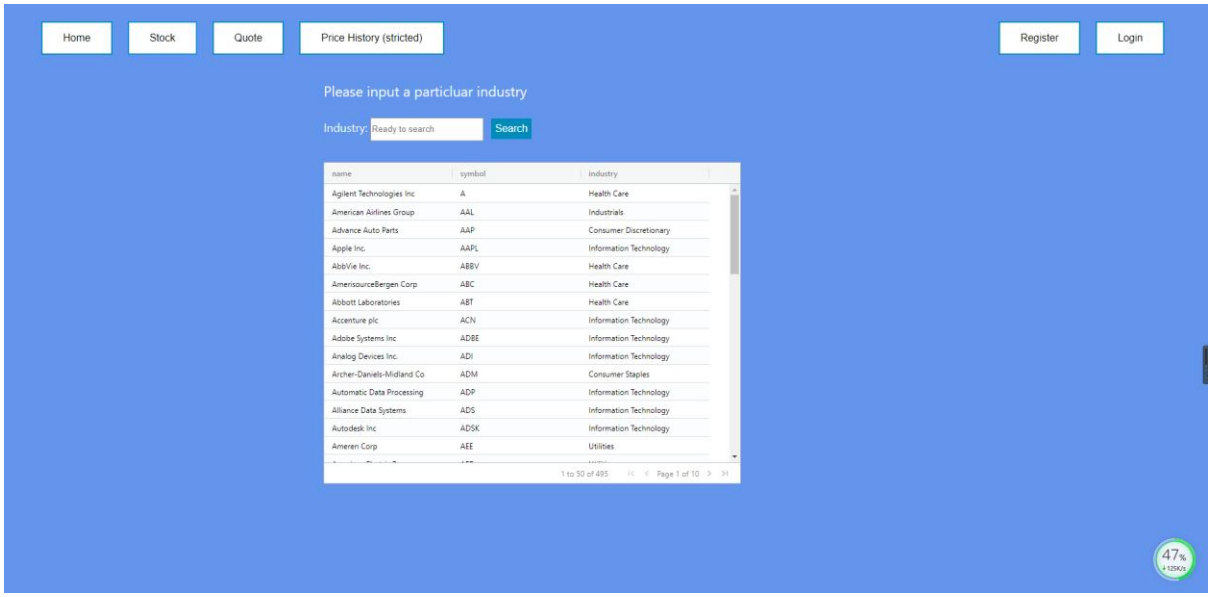
Tell us how to use your application

Use screenshots liberally here. You may re-use screenshots from above.

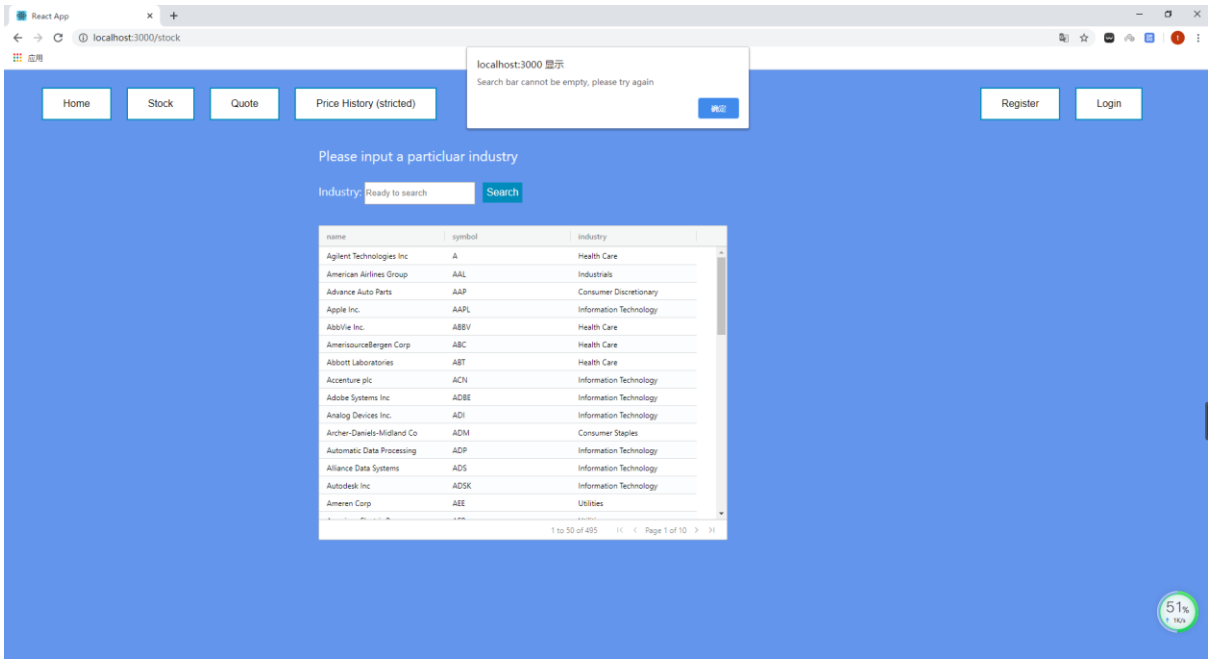
This is my home page layout. If you start from vs-code. It will link that page. In addition, I added in instruction in the center to explain each button usage.



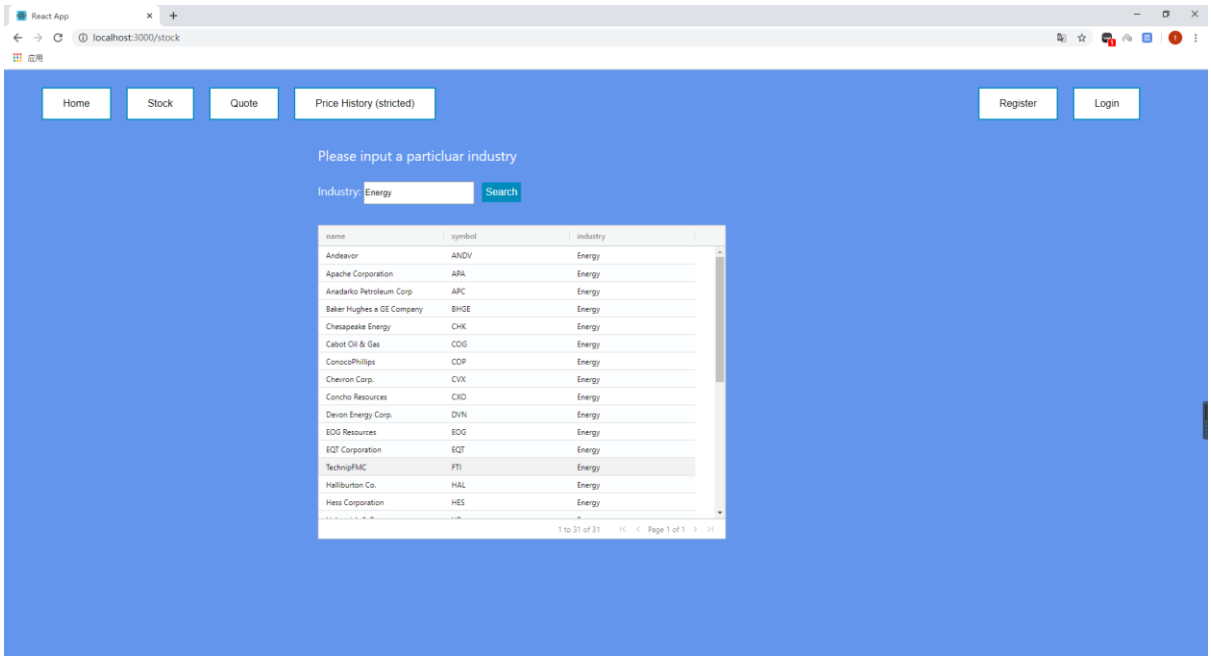
Now, clicking ‘stock’ button will move on a new page below. User could search industry in the search bar and has display all industries in the table.



Now, I click search button with empty word. It will involve an alert. Search bar cannot be null.

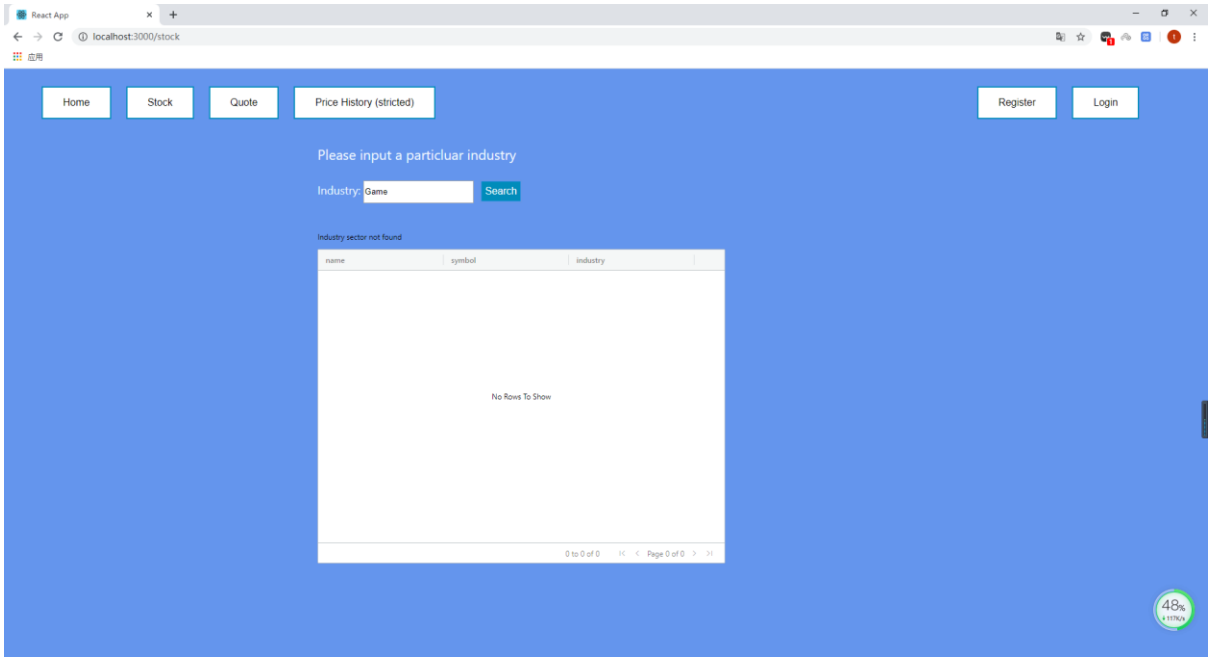


Now, I am going to add one of industry (Energy) in the search bar, the table has change. Industry has become to ‘Energy’ on the top.

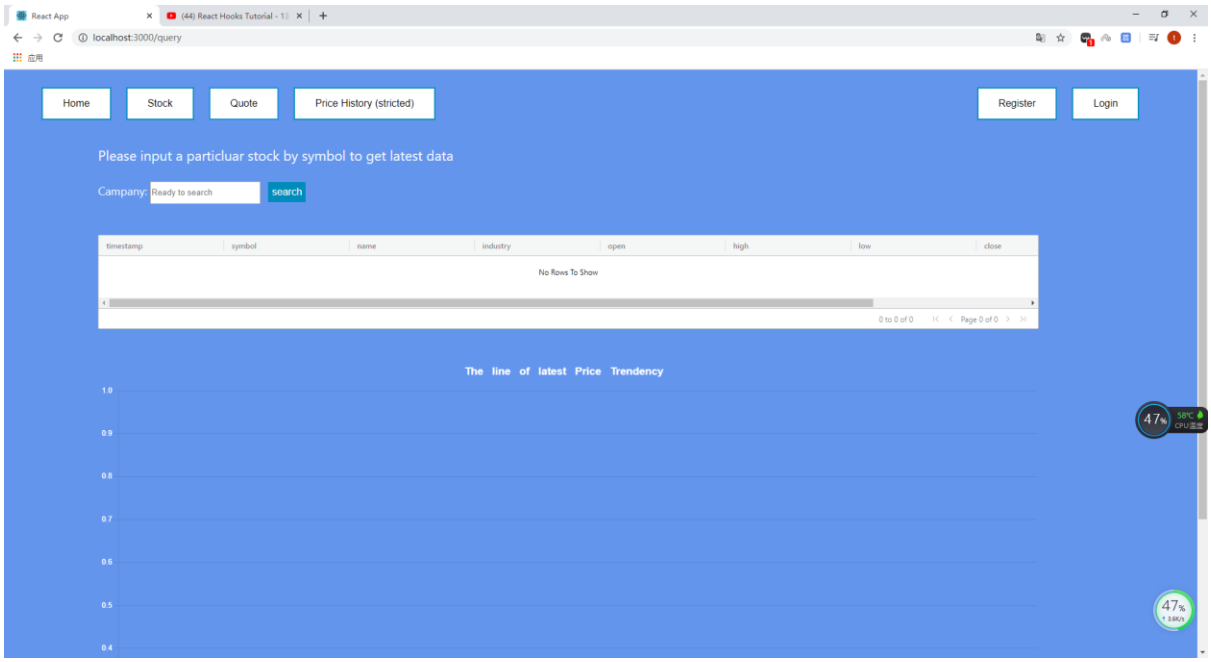


Now I am going to add industry name that doesn’t exist. It returns nothing in the table. in addition, there is a sentence on the left upper corner of table. I basically catch error from res. If res return

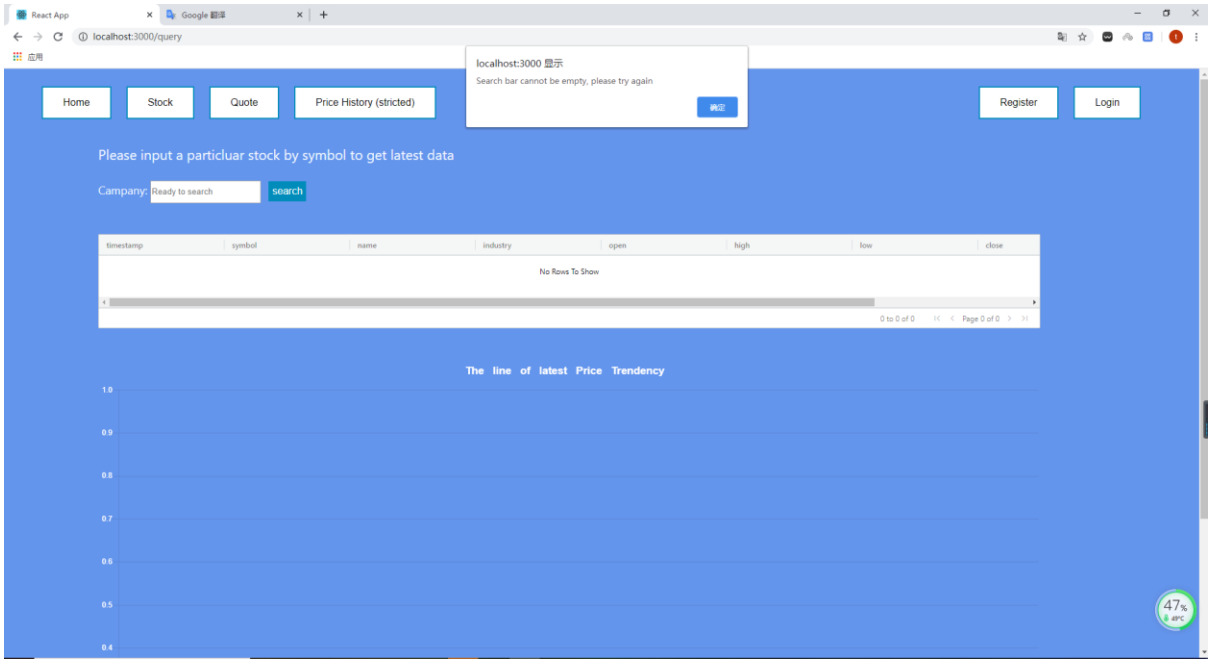
error, it will display on the screen.



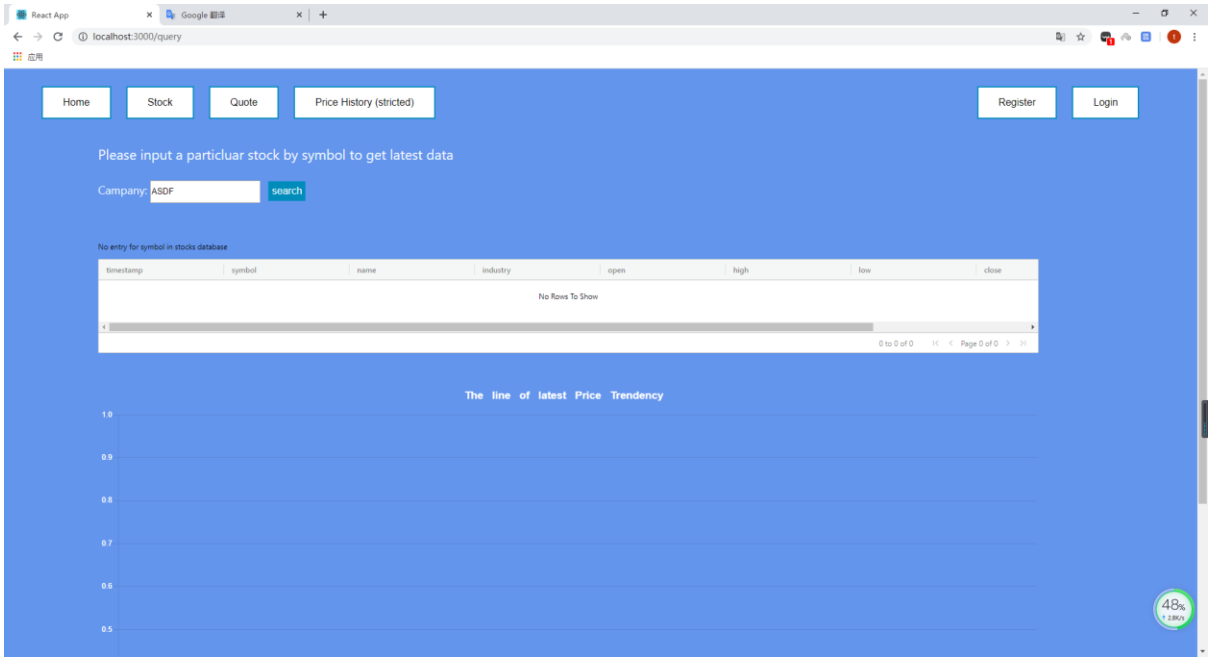
Now, I am going to click ‘Query’ button. It will move on a new page (‘/query’). This is my query layout including search bar, table and line-chart.



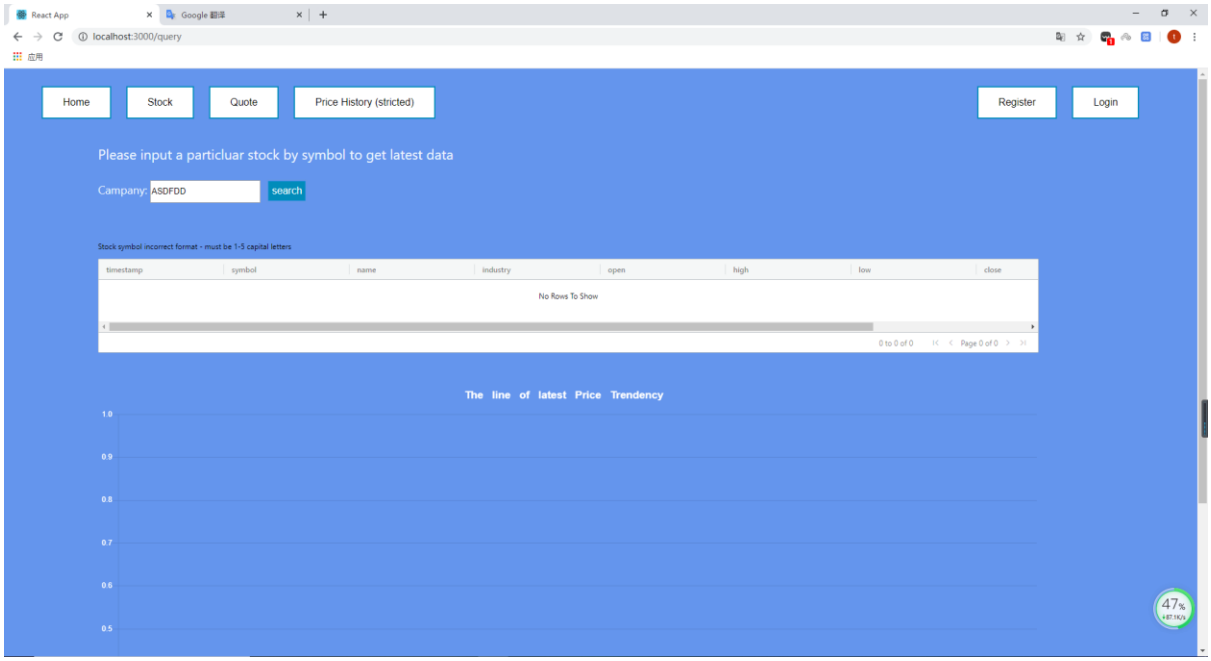
I am going to type without company by symbol. It involves an alert, which means that search bar cannot be null.



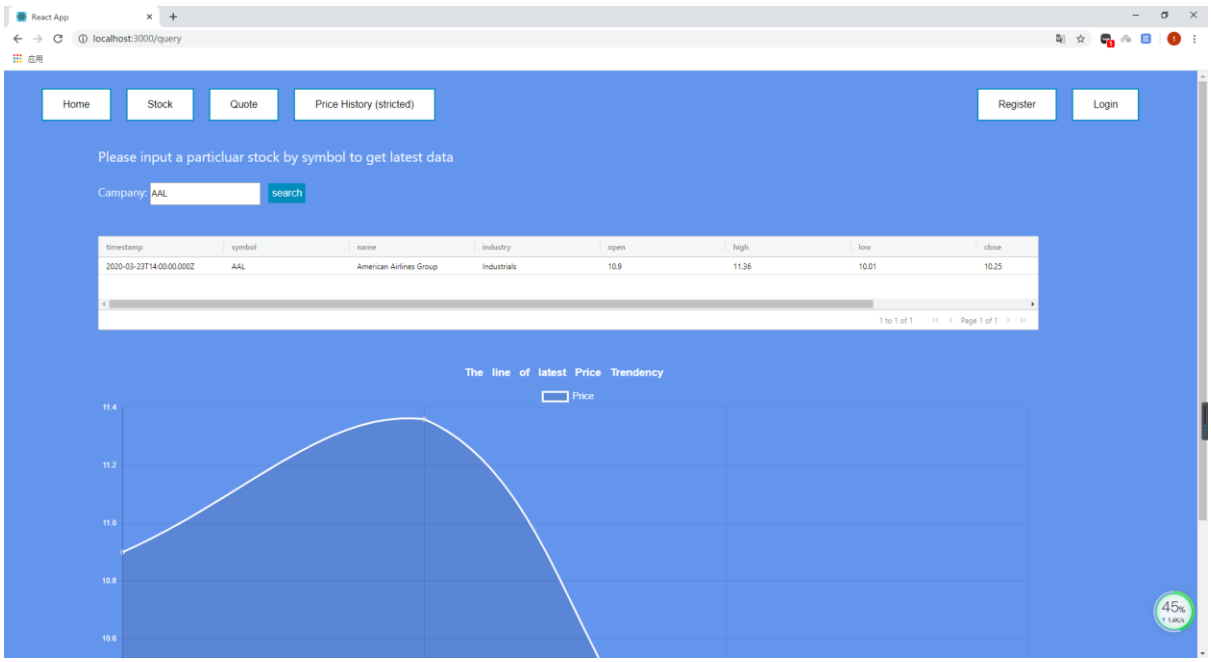
If I type company name doesn't exist, it will catch error from res and display error on the left upper corner of table.

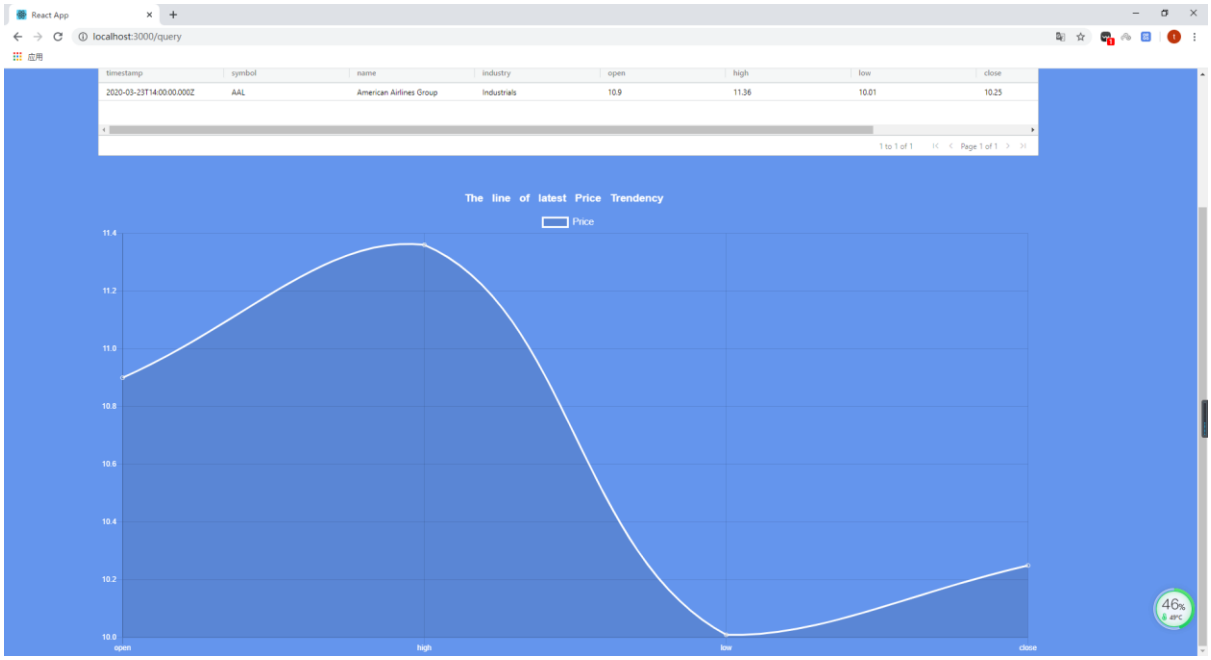


If I type company name more than 5 letters, it also will catch error from res and display error on the left upper corner of table.

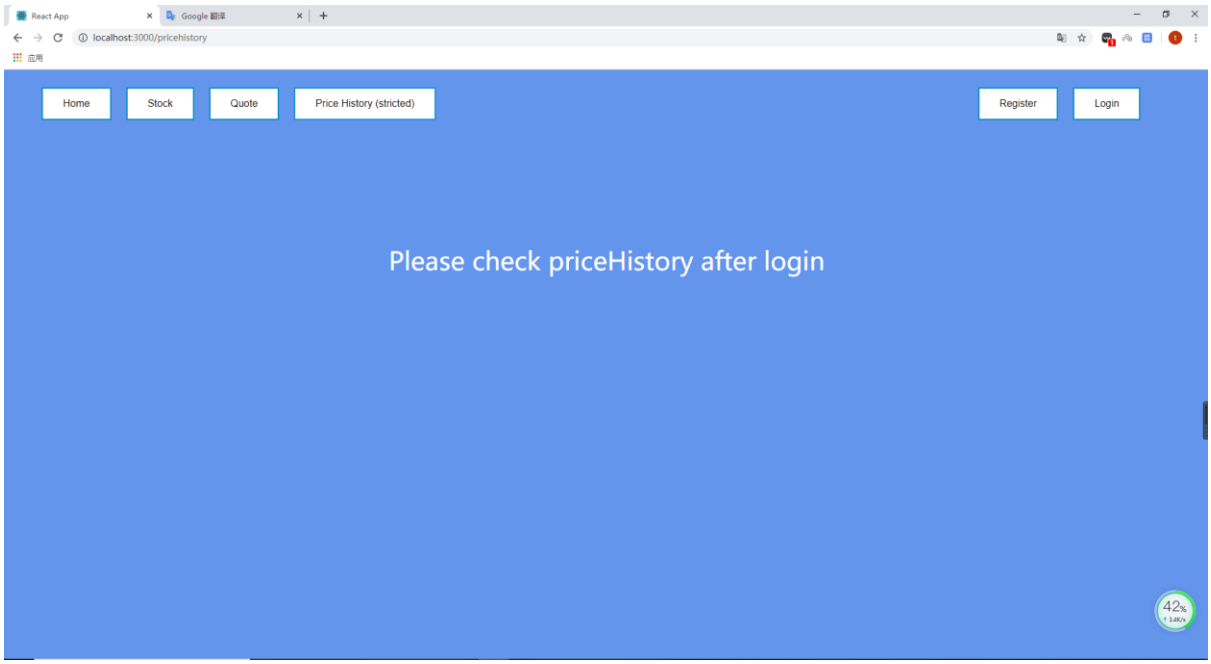


Now, I am going to type one of company by symbol. As you cloud see it return value in the table and chart.

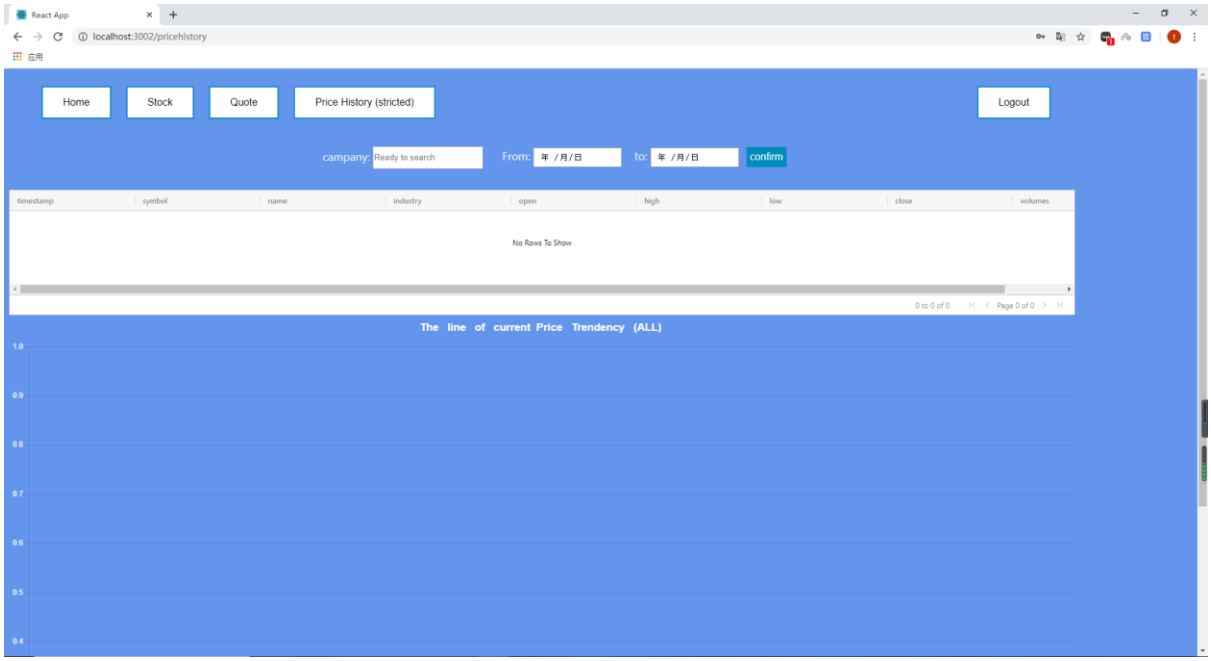




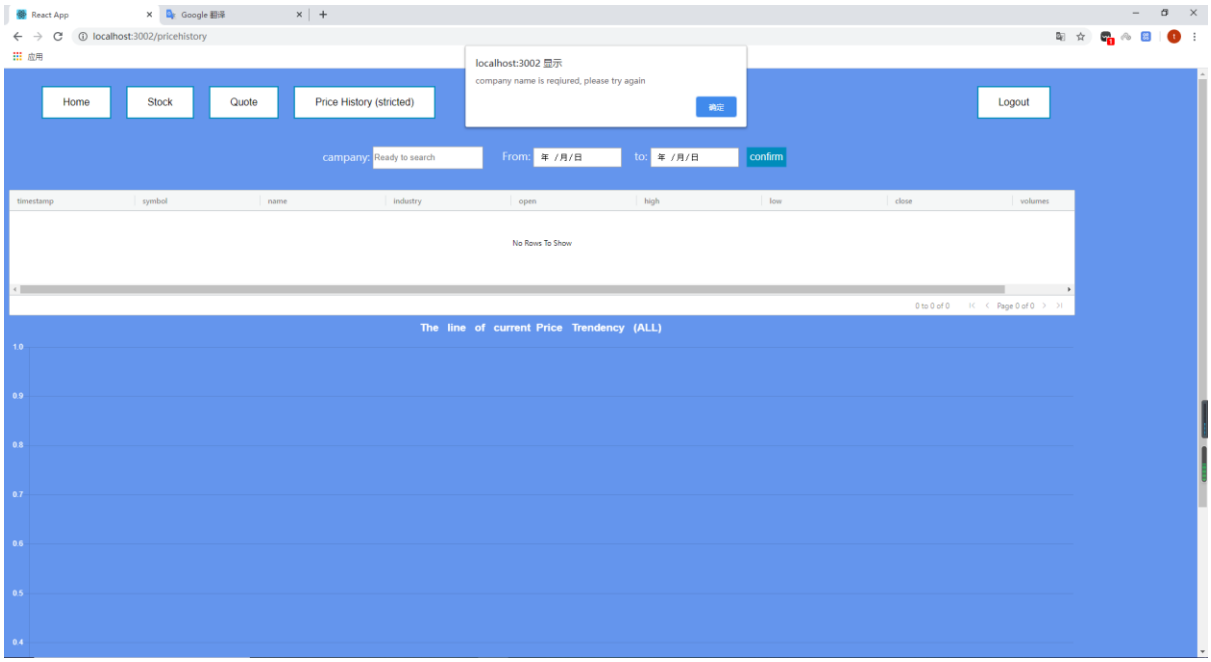
Now, I am going to click ‘price history’ button without login, there is instruction on the center. It tells user if you want to check history, you have to login.



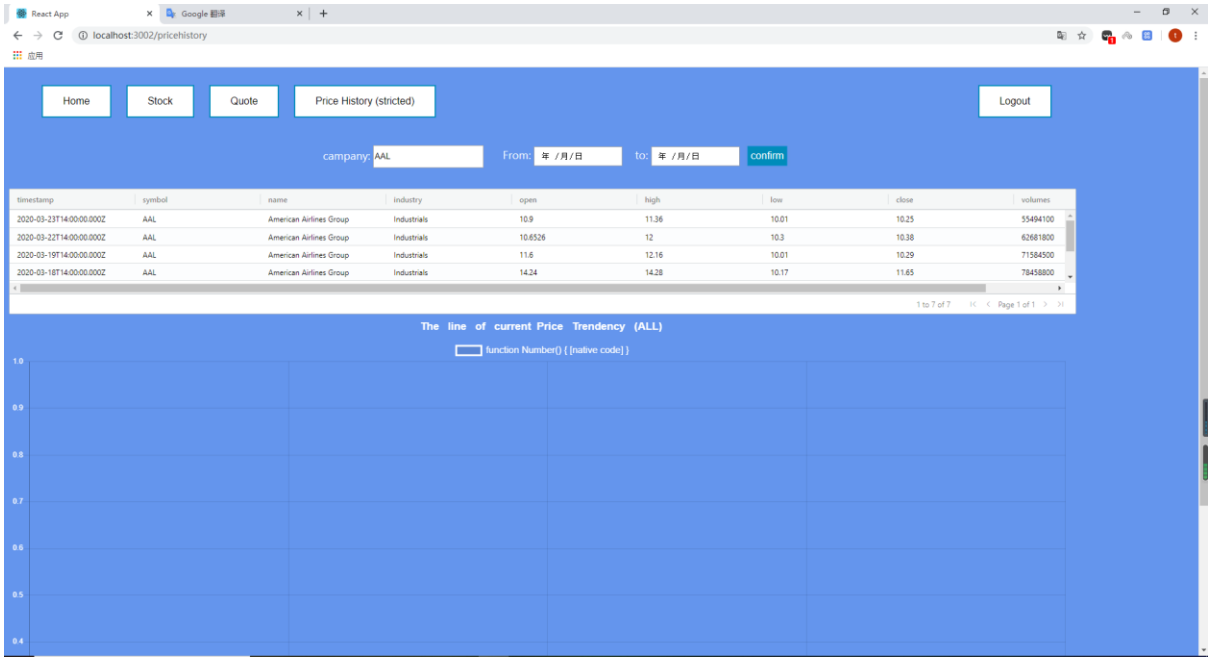
Now, I have logged in, there are search bar, date, table and line-chart in this page.



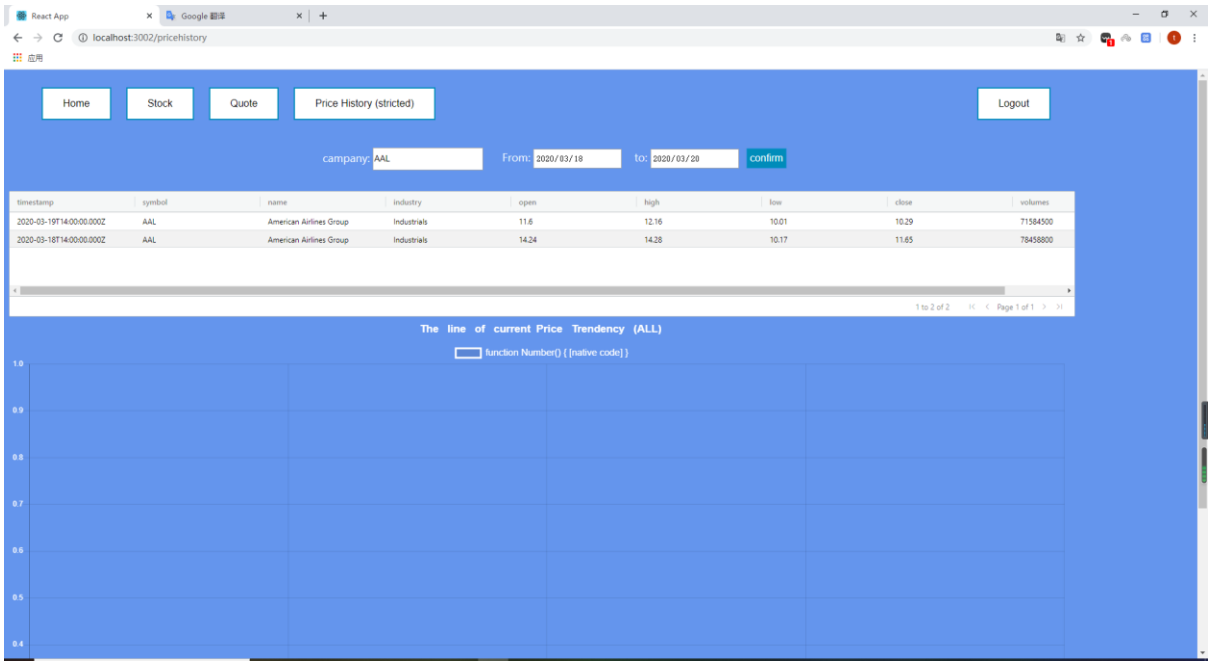
Now, I am going to type without any word. It involves an alert on the top, which means search bar cannot be null, because company name is required, date is not required.



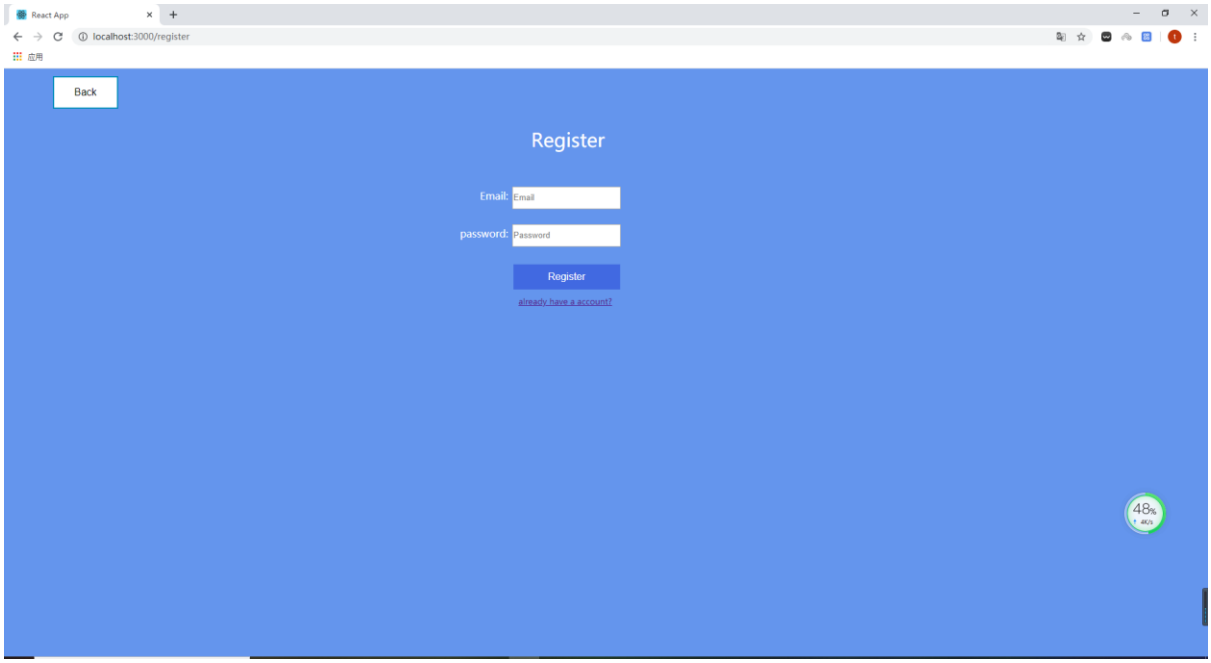
Now, I am going to type one of company name by symbol. It does return values in the table.



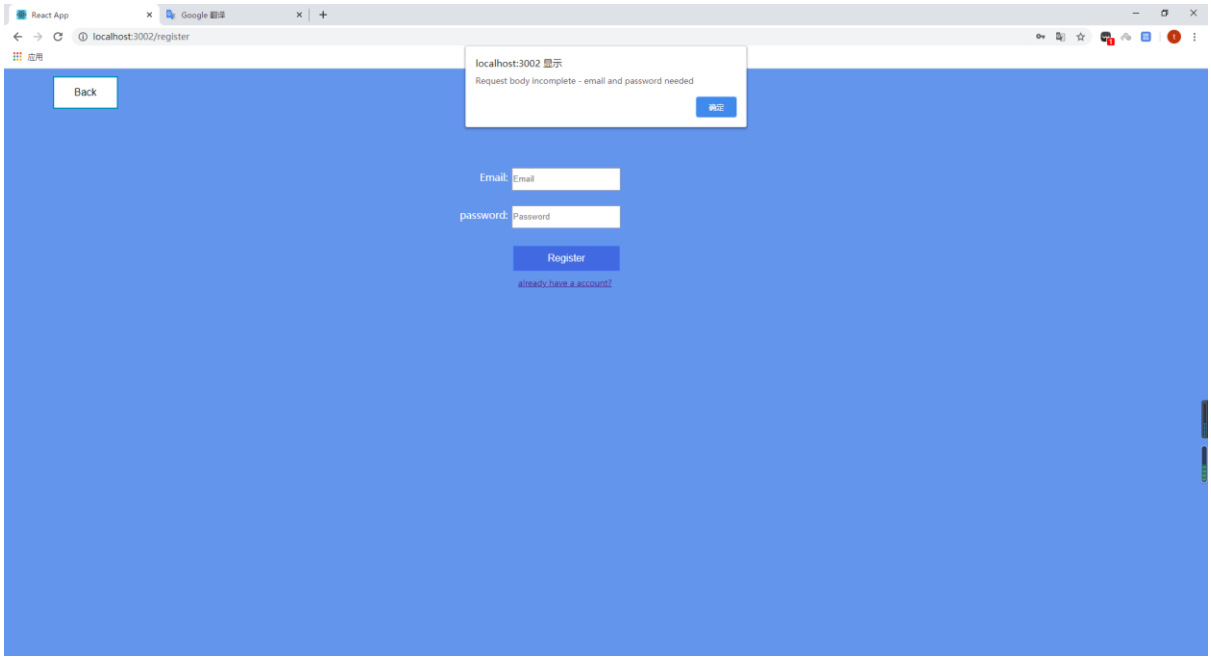
Now I am going to type one of company and the date. you could see It returns the values in the table with correct time.



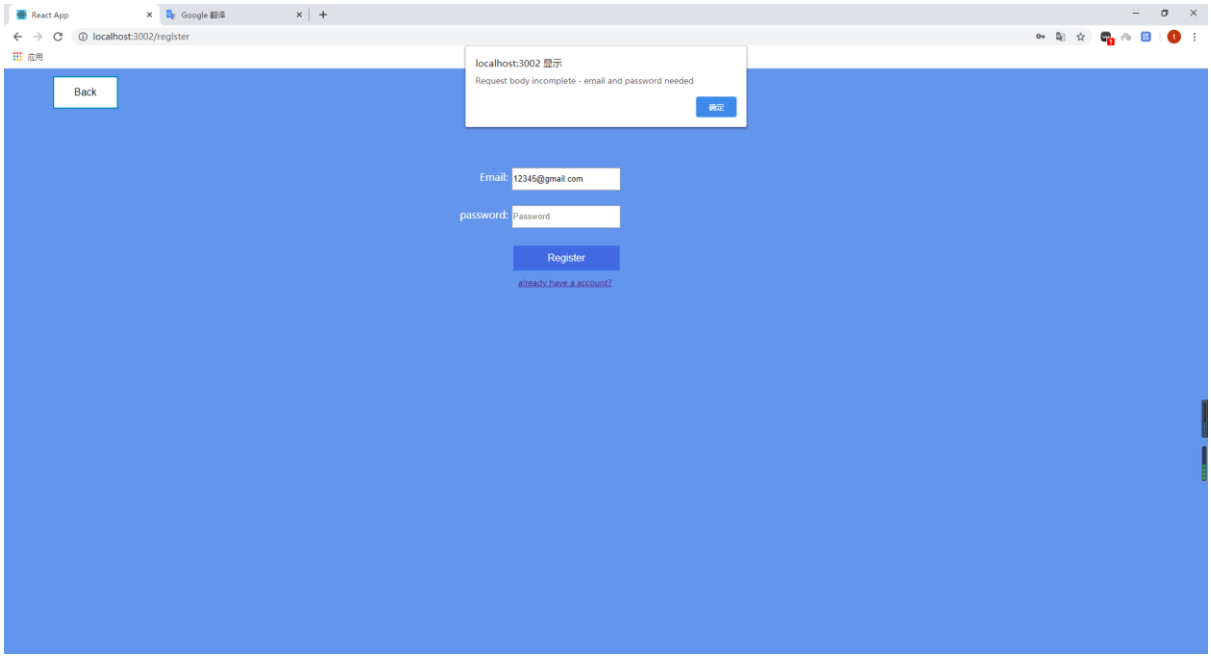
Now, moving on register section. this is my register layout below. ‘Back’ button is backing to home page. In addition, there is a sentence below ‘register’ button, which mean if you have an account already, you could click it. it will move on login page.



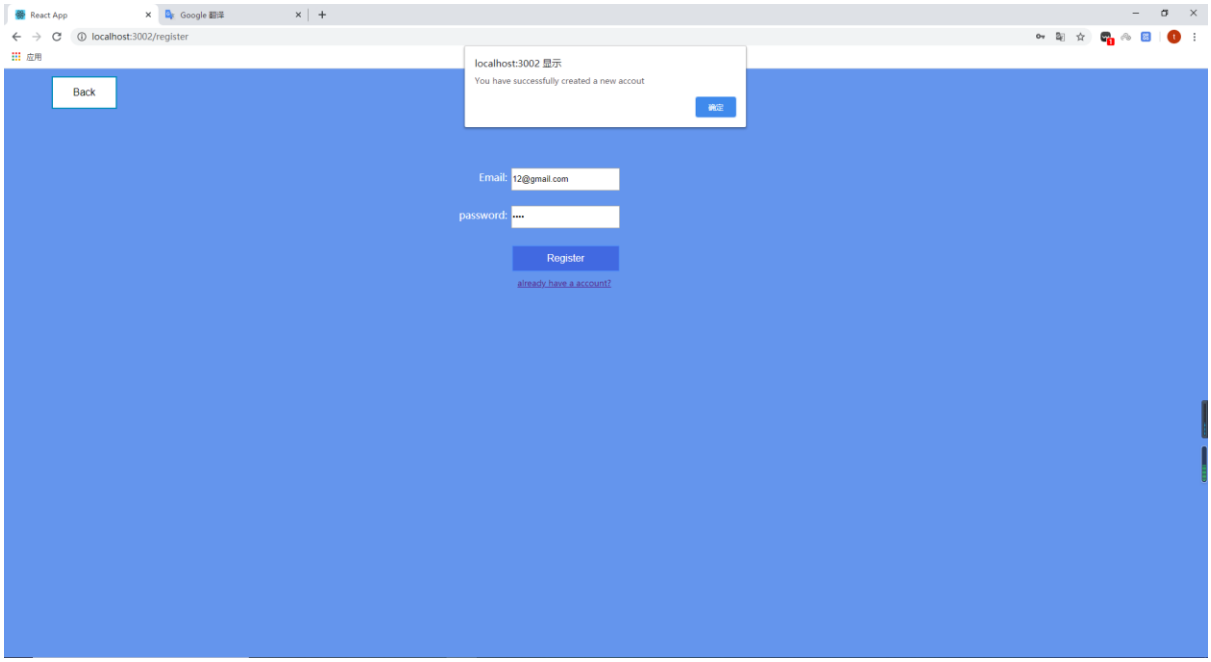
Now, I am going to type without email and password. It involves an alert here, because I catch error from res and display on the screen, which means that email and password cannot be null.



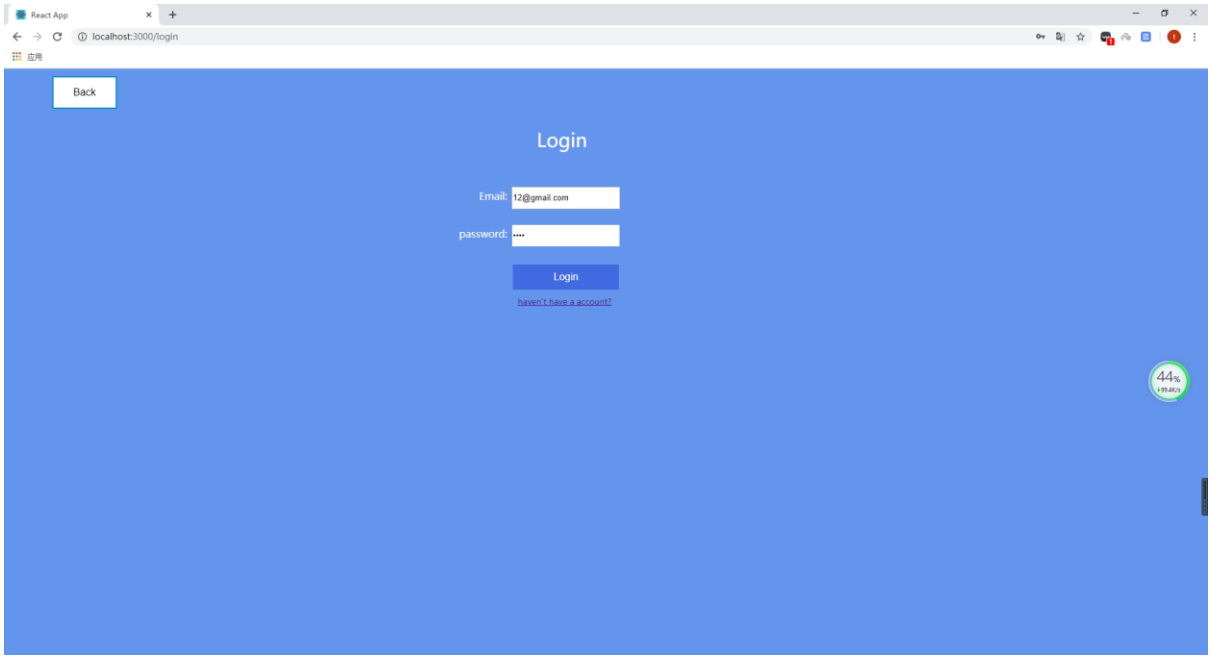
Now, I am going to type email address without password. It also involves an alert on the top, because I catch error form res as alert.



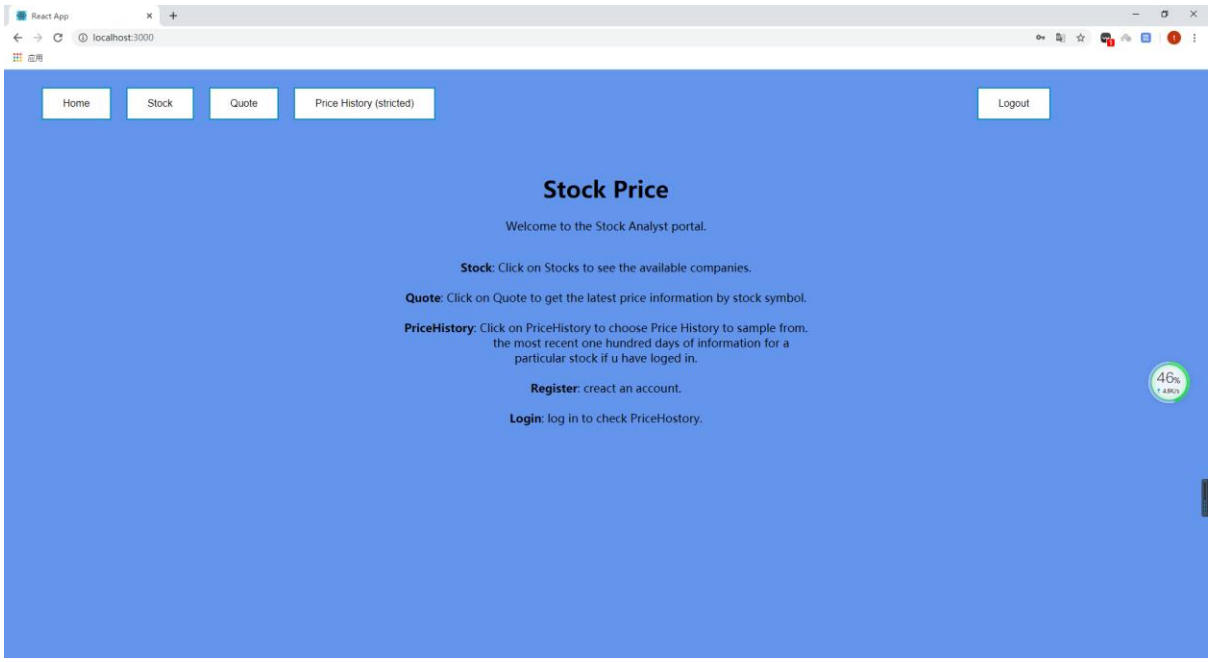
Now, I am going to create a real account. You can see if you successfully create a new account, it also has an alert on the top. This alert is added by me. Not from res.



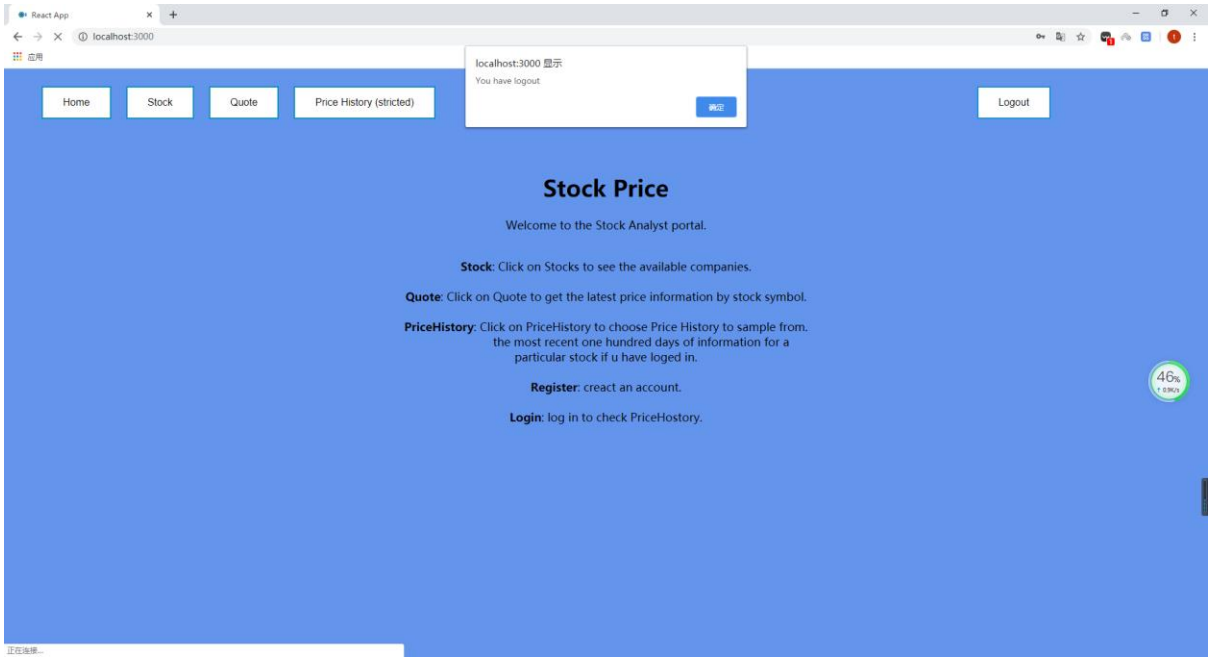
Now, we have already an account. Just log in to see what is happen.



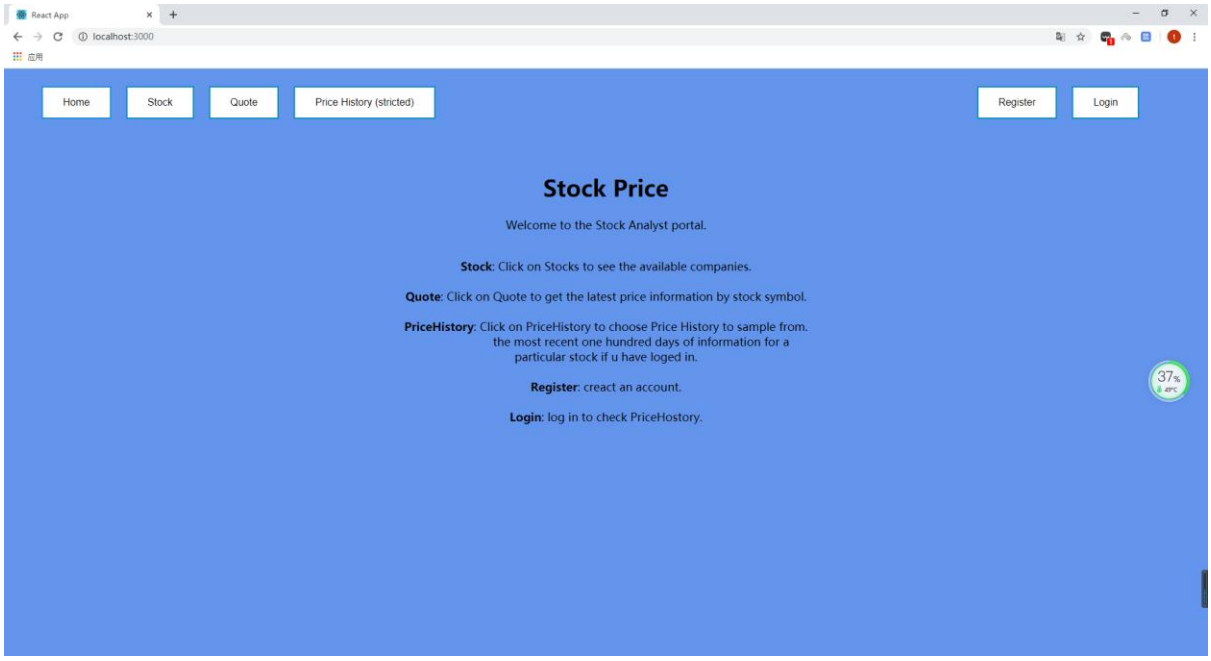
Now, if we log in, it will move on home page. The difference is right upper corner has changed to ‘logout’ button.



Now, we logout, it also involves an alert on the top. This alert is added on by me. Not from res.



Now, it moves on home page without login.



References

ag-Grid (2020). ag-Grid: Getting Started with React. Retrieved from: https://www.youtube.com/watch?v=6PA45adHun8

ag-Grid (2020). ag-Grid React Component. Retrieved from: https://www.npmjs.com/package/ag-grid-react

ag-Grid. React Grid | Get Started with ag-Grid and React. Retrieved from: https://www.ag-grid.com/react-grid/

Chartjs. Linear Cartesian Axis. Retrieved from: https://www.chartjs.org/docs/latest/axes/cartesian/linear.html

Codevolution (2019). React Hooks Tutorial - 14 - Fetching data with useEffect Part 3. Retrieved from: https://www.youtube.com/watch?v=1tfd6ANaNRY

Chartjs. The Usage of Chaertjs. Retrieved from: https://www.chartjs.org/docs/latest/getting-started/usage.html

Media, T. (2017). Using Chart.js With React. Retrieved from: https://www.youtube.com/watch?v=Ly-9VTXJlnA&t=735s

react-chartjs-2 (2020). react-chartjs-2. Retrieved from: https://www.npmjs.com/package/react-chartjs-2

Tuts, L, U. (2016). ChartJS Tutorials #4 - Chart Options. Retrieved from: https://www.youtube.com/watch?v=AcoUu3bgKgM