

2020

CAB230 Stocks API – Client Side



Replace image with one
with some relevance to
your application here

CAB230

Stocks API – Client Side Application

<student name/s>

<student number/s>

4/23/2020

Contents

Introduction	3
Purpose & description.....	3
Completeness and Limitations.....	4
Use of End Points	7
/stocks/symbols	6
/stocks/{symbol}	12
/stocks/authed/{symbol}	18
/user/register	24
/user/login	28
Modules used.....	32
Module 1: Ag-grid-react // ag-grid-community	32
Module 2: react-chartjs-2	33
Application Design	34
Navigation and Layout	34
Test Plan	44
Technical Description.....	44
Architecture	52
Difficulties / Exclusions / unresolved & persistent errors.....	55
User guide	65
References	67

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

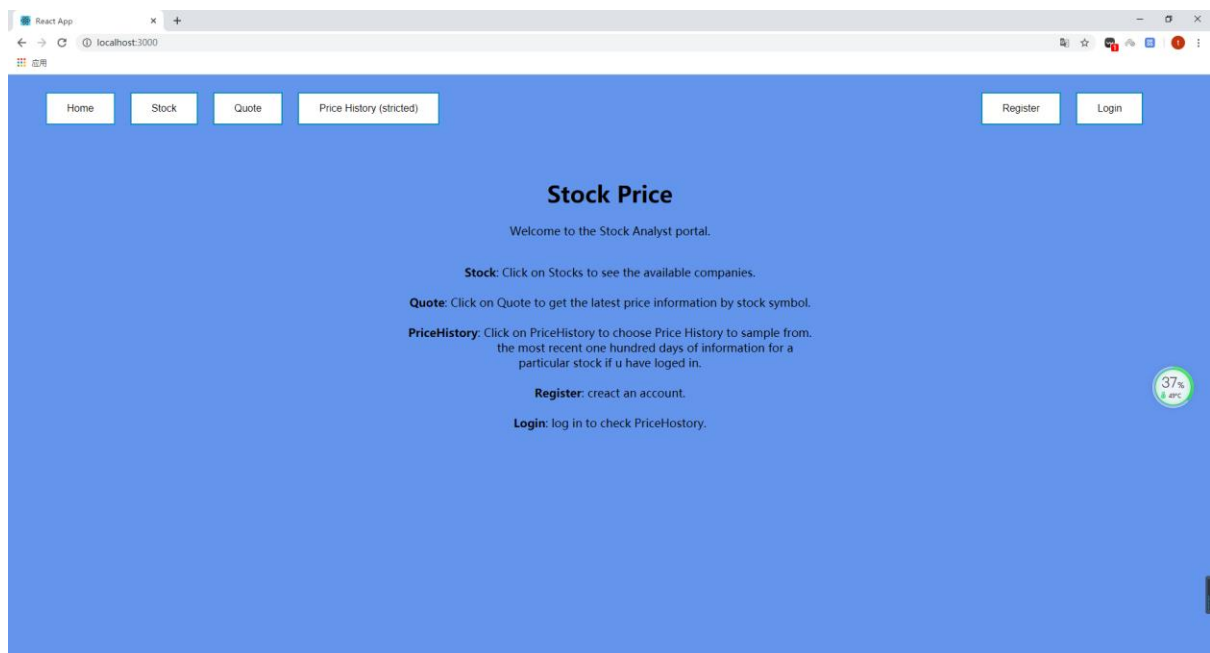
This report should be around 10 pages or so including screenshots

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).

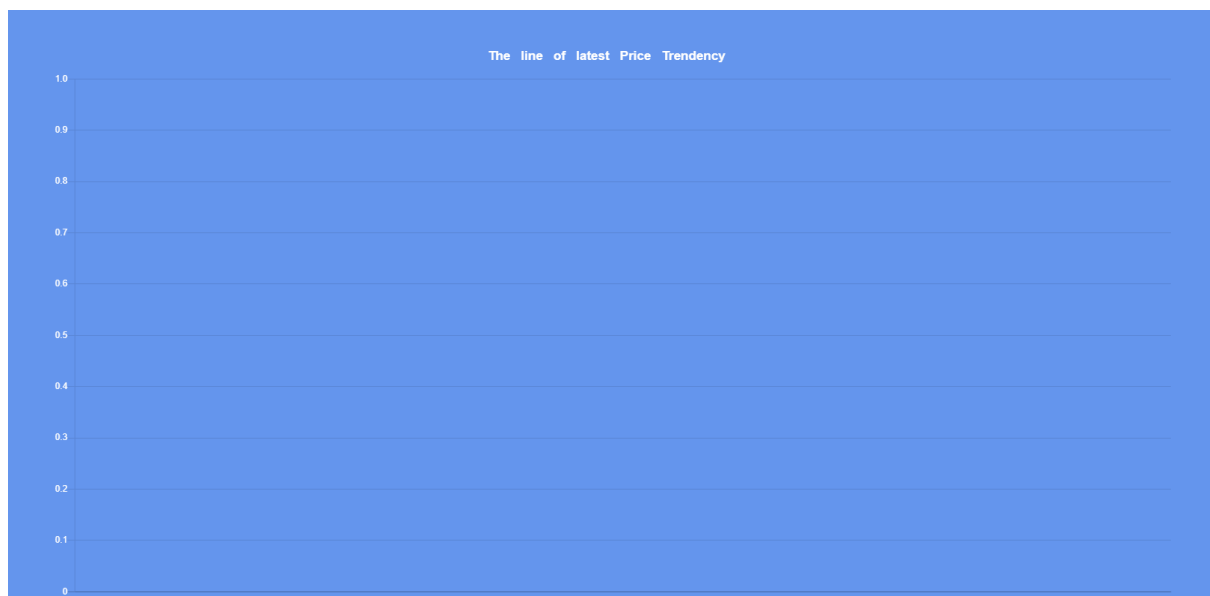


Photo 2

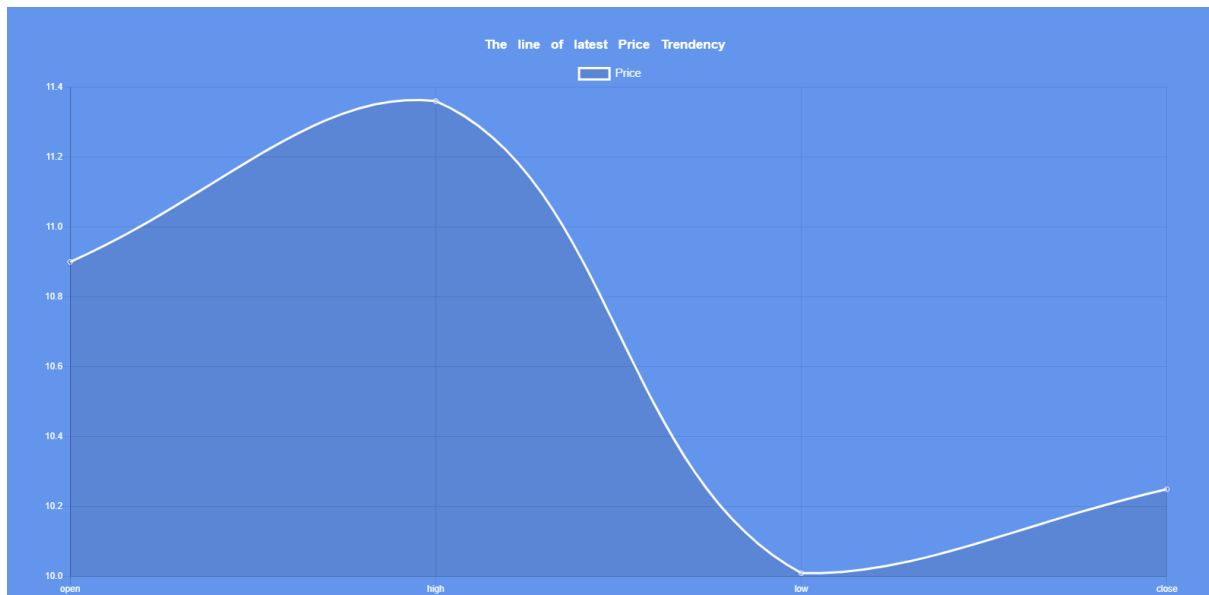


Photo 3

```
1 {
2   "name": "assessment_1",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^4.2.4",
7     "@testing-library/react": "^9.5.0",
8     "@testing-library/user-event": "^7.2.1",
9     "ag-grid-community": "^23.0.2",
10    "ag-grid-react": "^23.0.3",
11    "chart.js": "^2.9.3",
12    "jsonwebtoken": "^8.5.1",
13    "react": "^16.13.1",
14    "react-chartjs-2": "^2.9.0",
15    "react-dom": "^16.13.1",
16    "react-router-dom": "^5.1.2",
17    "react-scripts": "3.4.1"
18  },
19  "scripts": {
20    "start": "react-scripts start",
21    "build": "react-scripts build",
22    "test": "react-scripts test",
23    "eject": "react-scripts eject"
24  },
25  "eslintConfig": {
26    "extends": "react-app"
27  },
28  "browserslist": {
29    "production": [
30      ">0.2%",
31      "not dead",
32      "not op_mini all"
33    ],
34    "development": [
35      "last 1 chrome version",
36      "last 1 firefox version",
37      "last 1 safari version"
38    ]
39  }
40 }
```

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.

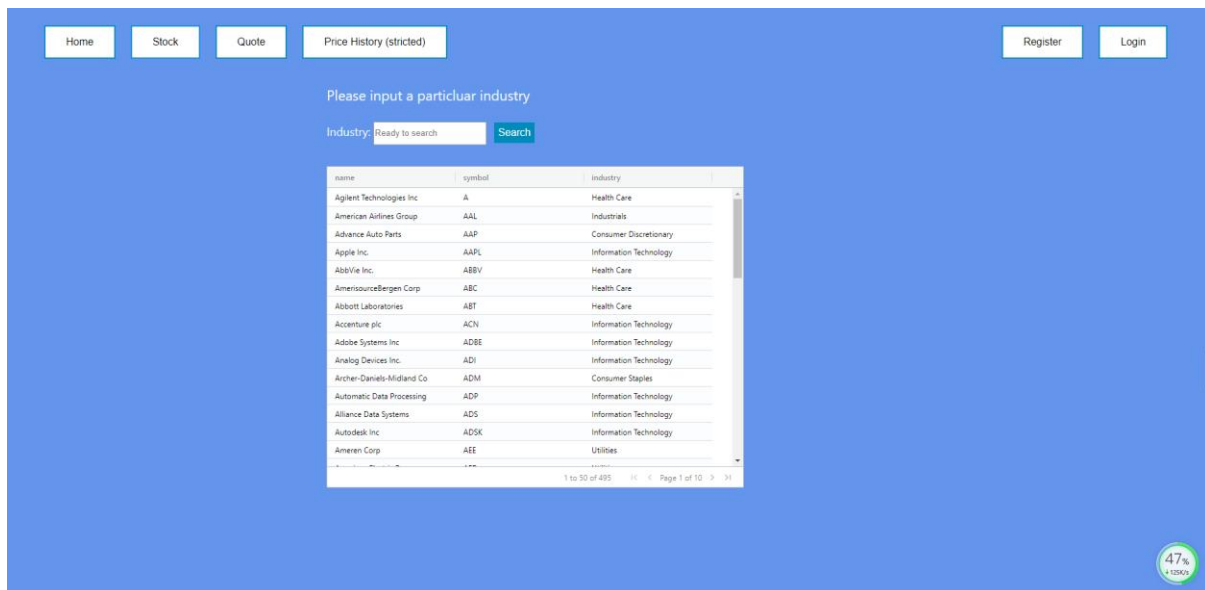
Basically, I finished all the functions, I could fetch data from res, then return the value into agrid-table and chart-line. In addition, I could type in search Bar, and then it will return exactly what I am looking for. Moreover, I could successfully login account form POST, also you could register account by yourself, it is allowed. However, 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 does.** (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))

}, [])
```

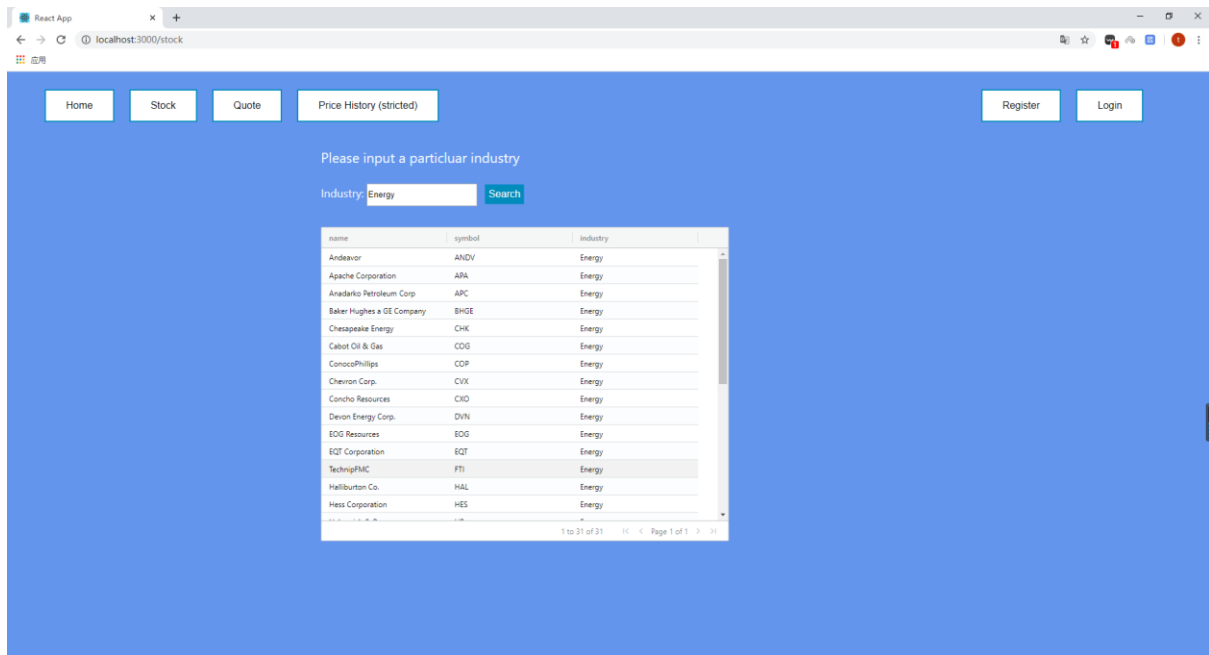
Now, I realized that I don't have to use 'map' return my value. I can just setRowData from res. See below. I just need to judge what type res returns.

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

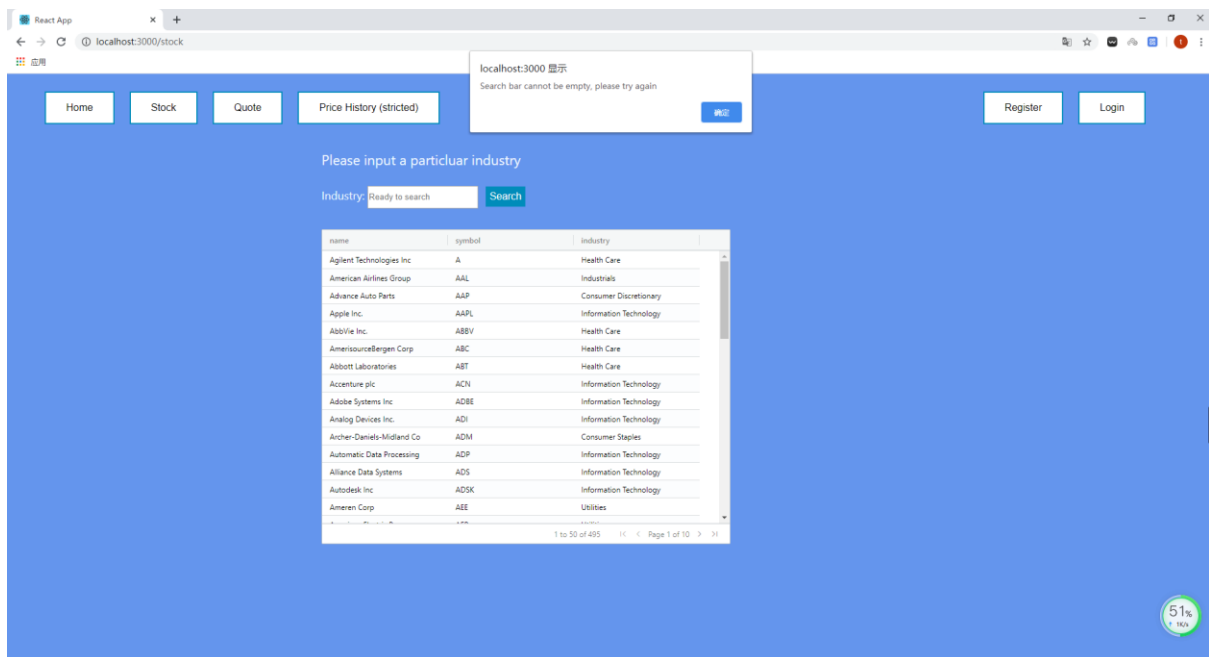
  fetch(url_stock)
    .then((res) => {
      const setRowData: (value: React.SetStateAction<any[]>) => void
      .then((res) => setRowData(res))
    })

}, [])
```

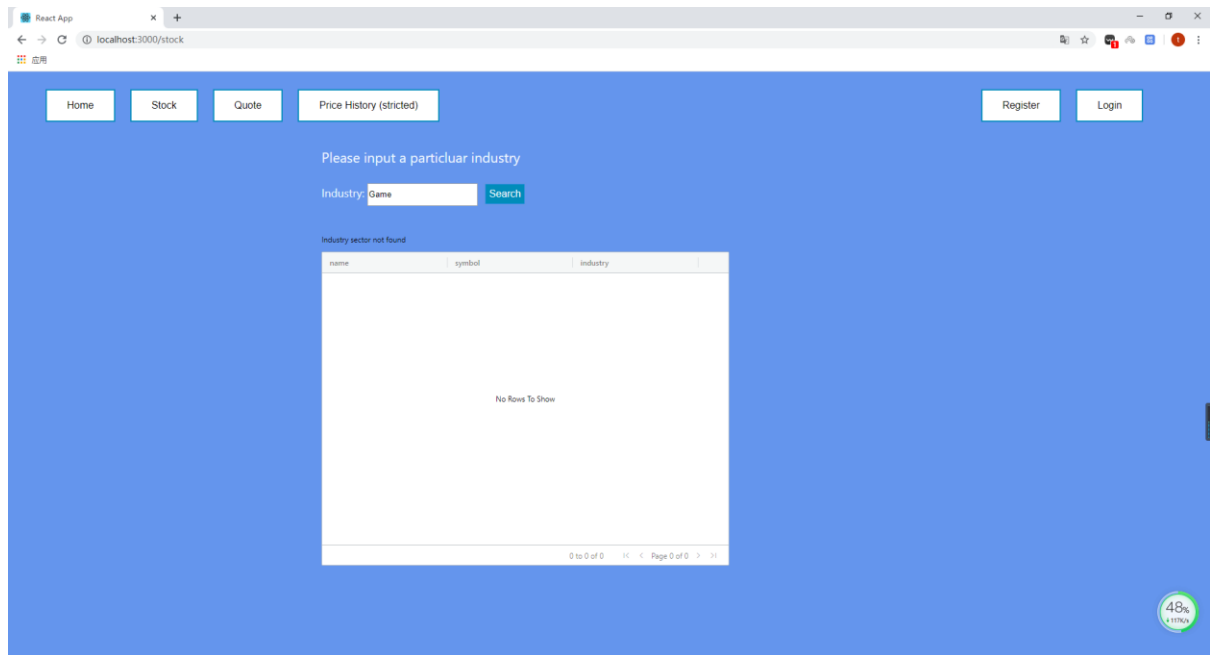
Now, moving on search Bar, I am going to type one of industry name in the search bar to see what is happen.



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 href = <https://www.youtube.com/watch?v=1tfd6ANaNRy>

```
src > JS stock.js 300 columns
1 import React, { useState, useEffect } from "react"
2
3
4 import { AgGridReact } from "ag-grid-react";
5 import "ag-grid-community/dist/styles/ag-grid.css";
6 import "ag-grid-community/dist/styles/ag-theme-balham.css";
7
8 export default function Stock() {
9
10
11
12 //set table display(row)
13 const [rowData, setRowData] = useState([]);
14 //set searchbar
15 const [innerSearch, setInnerSearch] = useState("");
16
17 //return error message (object type)
18 const [catchError, setCatchError] = useState({});
19
20
21 //columns of table
22 const columns = [
23
24   { headerName: "name", field: "name", sortable: true, filter: true },
25   { headerName: "symbol", field: "symbol", sortable: true, filter: true },
26   { headerName: "industry", field: "industry", sortable: true, filter: true }
27 ];
28
29
30 //url
31 const url_stock = "http://131.181.190.87:3000/stocks/symbols";
32 const url_template = "http://131.181.190.87:3000/stocks/symbols?industry=${innerSearch}";
33
34
35 //every time i click the button return value if value exist
36 const handleClick = () => {
37
38   //searchbar cannot be null
39   if (!innerSearch) {
40     alert("Search bar cannot be empty, please try again");
41     return;
42   }
43
44   //fetch data
45   fetch(url_template)
46     .then((res) => res.json())
47     .then((res) => {
48
49     });
50
51   };
52
53   //to show all kind of industry in the table
54   useEffect(() => {
55
56     fetch(url_stock)
57       .then((res) => res.json())
58       .then((res) => setRowData(res));
59
60   });
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

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).

```
src > JS stock.js 300 columns
37
38 //searchbar cannot be null
39 if (!innerSearch) {
40   alert("Search bar cannot be empty, please try again");
41   return;
42 }
43
44 //fetch data
45 fetch(url_template)
46   .then((res) => res.json())
47   .then((res) => {
48     //if res have error, return error from res to display
49     if (res.error) {
50       setErrorObject(res);
51       setRowData([]);
52     }
53     //otherwise return object array from res
54     else {
55       setRowData(res);
56       setErrorObject({});
57     }
58   });
59
60
61
62
63
64
65
66 //to show all kind of industry in the table
67 useEffect(() => {
68
69   fetch(url_stock)
70     .then((res) => res.json())
71     .then((res) => setRowData(res));
72
73 });
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Compiled with warnings.

Warning

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 [hooks](#) 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
  # ignore
  CAR20Assignment1Specification.pdf
  package-lock.json
  README.md

  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122

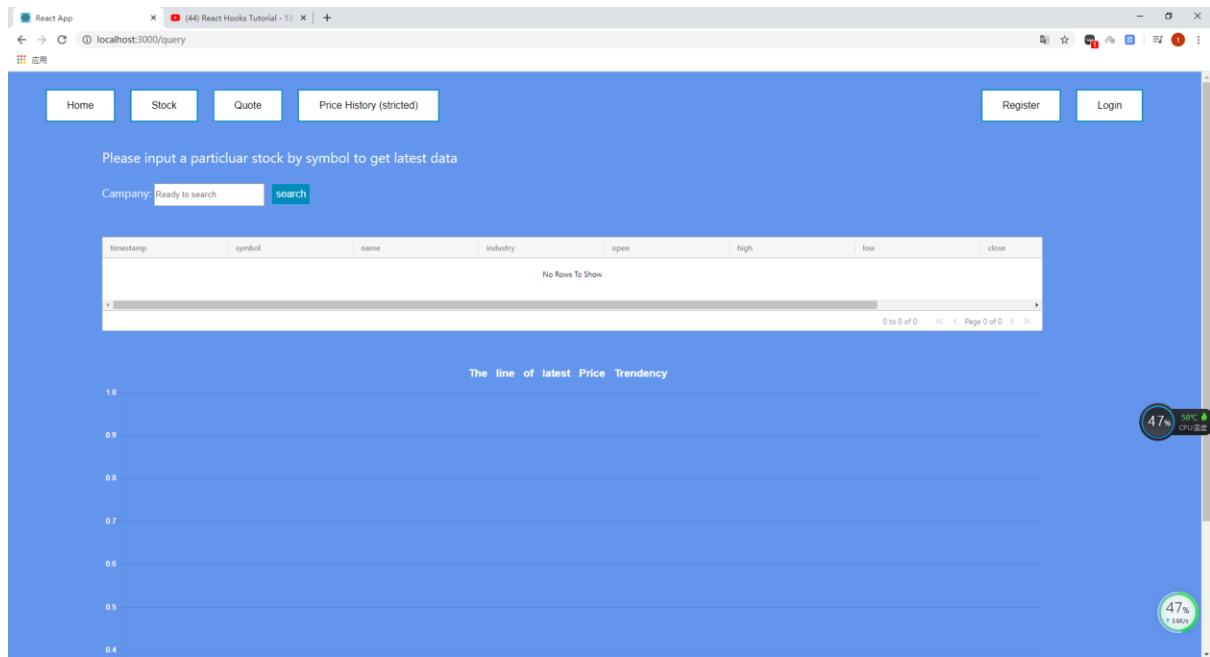
return (
  //table size
  <div className="ag-theme-balham" style={
    {
      width: 650,
      height: 500,
      position: "absolute",
      left: 500
    }
  }>
    /* search bar and button */
    <div className="searchBar">
      <p1 className="heading-size">Please input a particular industry</p1>
      <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>
    </div>
    <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}
    />
  </div>
)
```

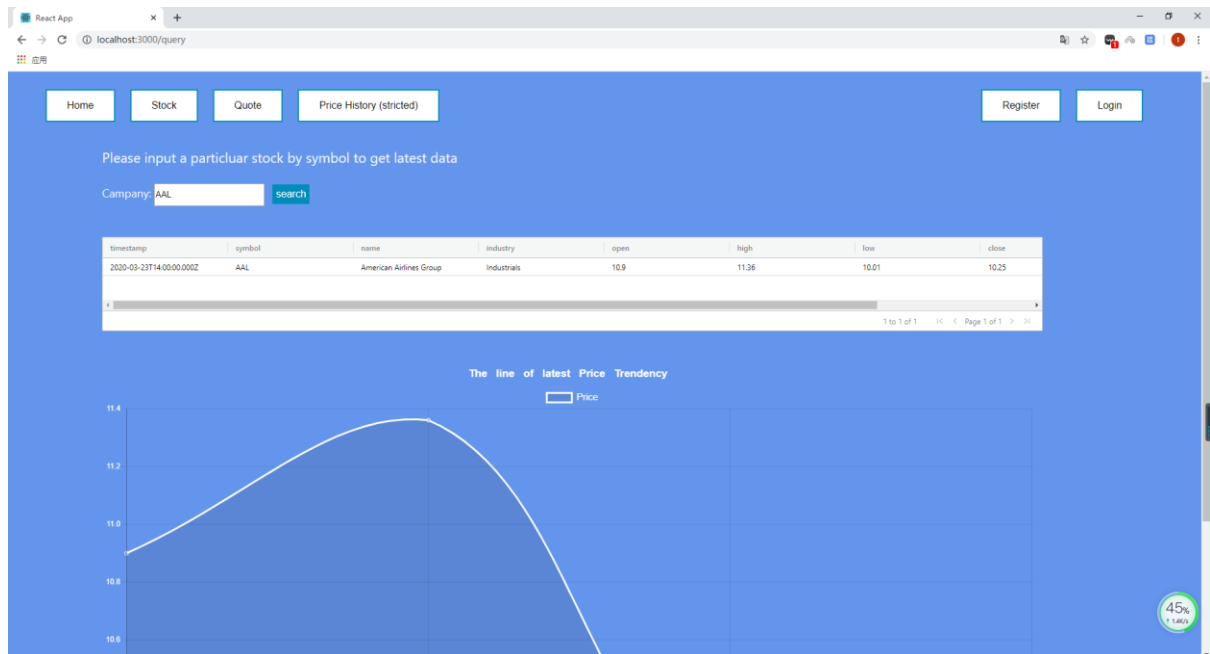
47%

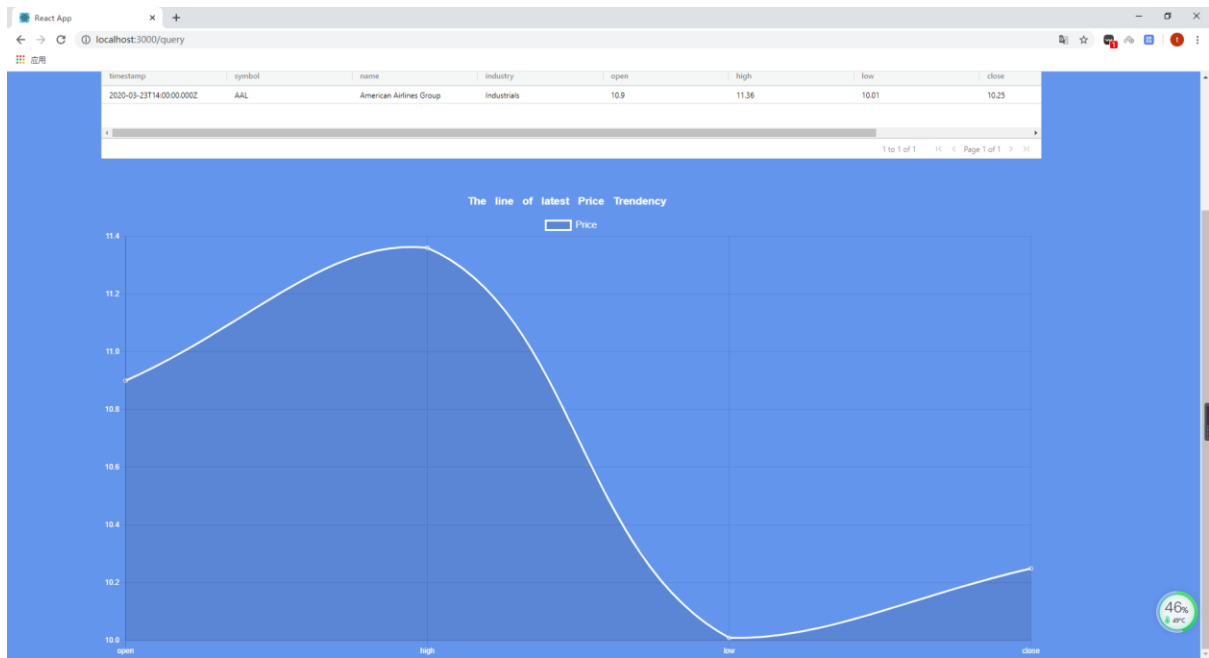
`/stocks/{symbol}`

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.

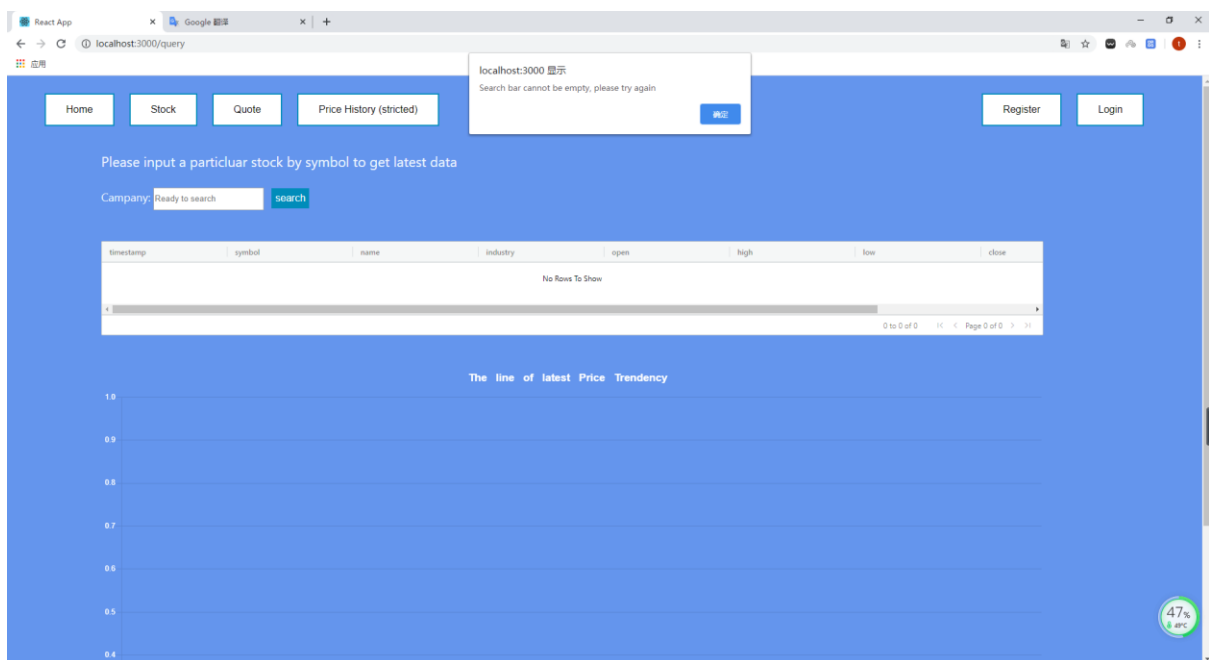


As we could see, it is similar with stock section. I add chartjs in the query to see price tendency on the latest day. I also write instruction on the top that provide user how to use it (only allow stock company name). now, I am going to type a symbol here to see what is happen.

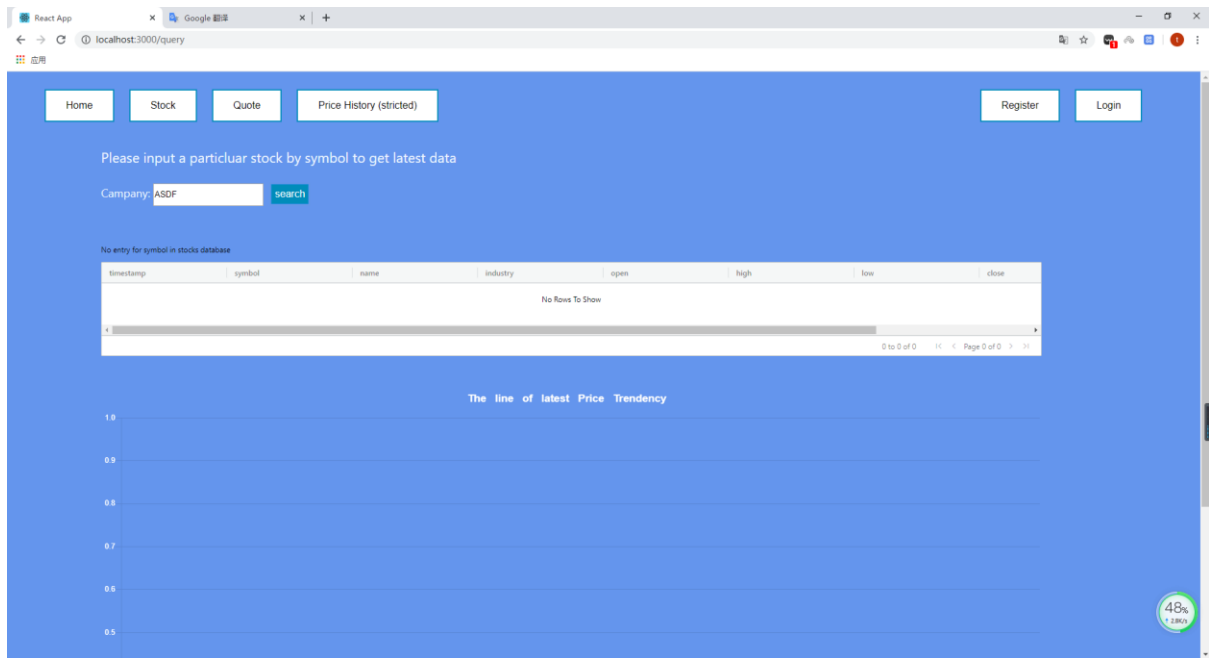




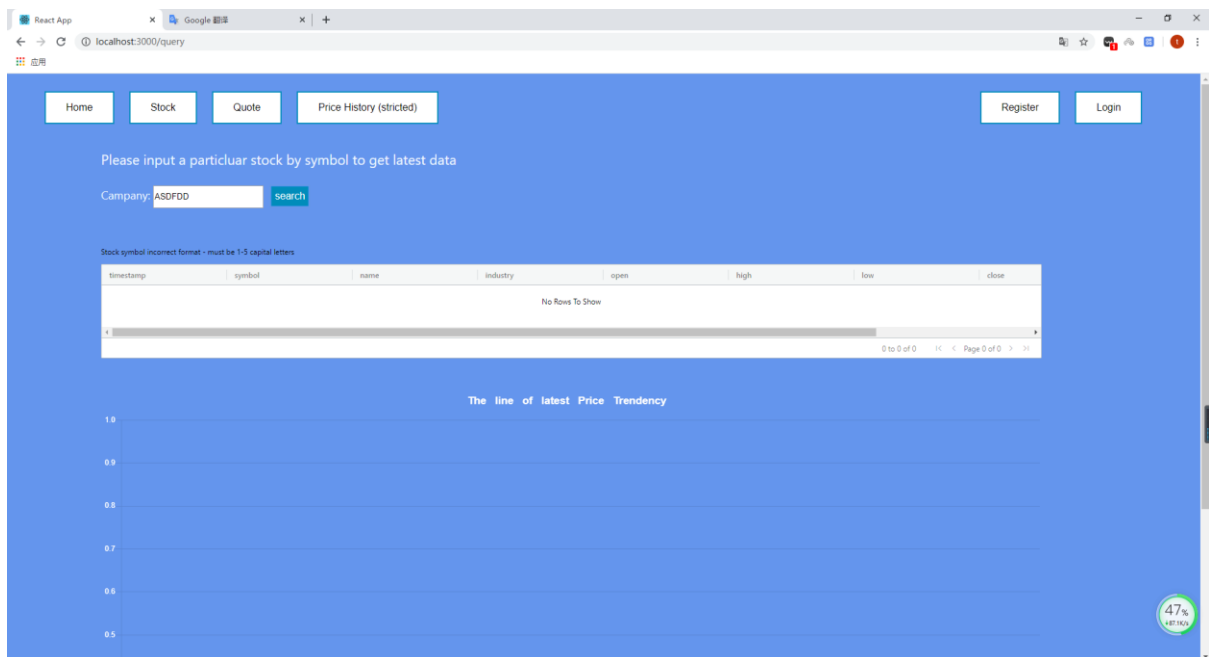
As we could see, I type one of company name by symbol (AAL). It returns the object from res (not array). That is why I have only one line in the table. In addition, how I default my chart is that I extract 4 objects type from res, which 'open', 'high', 'low' and 'close'. And then, let these objects become 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.

```
ASSESSMENT_1
> node_modules
> public
> src
  # home.js
  # index.js
  # login.js
  # pricehistory.js
  # query.js
  # register.js
  # stock.js
  # style.css
  # ignore
  # CAB230Assignment1Specification.pdf
  # package-lock.json
  # package.json
  # README.md

src > JS query.js > Query
1 import React, { useState } from "react";
2
3 import { AgGridReact } from "ag-grid-react";
4 import "ag-grid-community/dist/styles/ag-grid.css";
5 import "ag-grid-community/dist/styles/ag-theme-balham.css";
6 import { Line } from "react-chartjs-2";
7
8 export default function Query() {
9
10   //set table display(row)
11   const [rowData, setRowData] = useState([]);
12   //set chart display
13   const [lineData, setLineData] = useState({});
14   //return error message
15   const [catchError, setCatchError] = useState({});
16   //set searchbar
17   const [innerSearch, setInnerSearch] = useState("");
18
19   //columns of table
20   const columns = [
21     { headerName: "timestamp", field: "timestamp", sortable: true, filter: true },
22     { headerName: "symbol", field: "symbol", sortable: true, filter: true },
23     { headerName: "name", field: "name", sortable: true, filter: true },
24     { headerName: "industry", field: "industry", sortable: true, filter: true },
25     { headerName: "open", field: "open", sortable: true, filter: true },
26     { headerName: "high", field: "high", sortable: true, filter: true },
27     { headerName: "low", field: "low", sortable: true, filter: true },
28     { headerName: "close", field: "close", sortable: true, filter: true },
29     { headerName: "volumes", field: "volumes", sortable: true, filter: true }
30   ];
31
32   //every time i click the button return value if value exist
33   const handleClick = () => {
34
35     //searchbar cannot be null
36     if (!innerSearch) {
37       alert("Search bar cannot be empty, please try again");
38       return;
39     }
40
41     //url
42     const url = "http://131.181.190.87:3000/stocks/${innerSearch}";
43
44     fetch(url)
45       .then(res => res.json())
46       .then(res => {
47         //if res have error, return error from res to display, set rowData and lineData are null
48         if (res.error) {
49
50           48%
51           sec
```

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.

```
ASSESSMENT_1
> node_modules
> public
> src
  # home.js
  # index.js
  # login.js
  # pricehistory.js
  # query.js
  # register.js
  # stock.js
  # style.css
  # ignore
  # CAB230Assignment1Specification.pdf
  # package-lock.json
  # package.json
  # README.md

src > JS query.js > Query
36 //searchbar cannot be null
37 if (!innerSearch) {
38   alert("Search bar cannot be empty, please try again");
39   return;
40 }
41
42 //url
43 const url = "http://131.181.190.87:3000/stocks/${innerSearch}";
44
45 fetch(url)
46   .then(res => res.json())
47   .then(res => {
48     //if res have error, return error from res to display, set rowData and lineData are null
49     if (res.error) {
50       setCatchError(res)
51       setRowData({})
52       setLineData({})
53       //otherwise res return the put the data into table and line chart
54     } else {
55
56       //res return object type, then default object to array and display to table(row)
57       const tableArr = [res];
58       //check whether have error
59       setCatchError({})
60
61       //res return object and then let these objects be array objects to display on line chart
62       const { open, high, low, close } = res
63       const lineArr = [open, high, low, close]
64
65       //set line chart
66       const lineChart = {
67         labels: ['open', 'high', 'low', 'close'],
68         datasets: [
69           {
70             label: 'Price',
71             data: lineArr,
72             pointBorderColor: "ffff",
73             borderColor: "ffff",
74             pointHoverBorderColor: "ffff",
75             borderJoinStyle: "miter",
76           }
77         ],
78       };
79
80       //put the value into chart and table
81       setRowData(tableArr)
82       setLineData(lineChart)
83
84           49%
85           sec
```

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.


```

83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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.



ASSESSMENT_1

src > JS Query.js > Query

img

stockapp

stock2.jpg

stock_modules

public

src

home.js

index.js

login.js

pricehistory.js

query.js

register.js

stocks.js

style.css

.gitignore

CAB230Assignment1Specification.pdf

package-lock.json

package.json

README.md

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

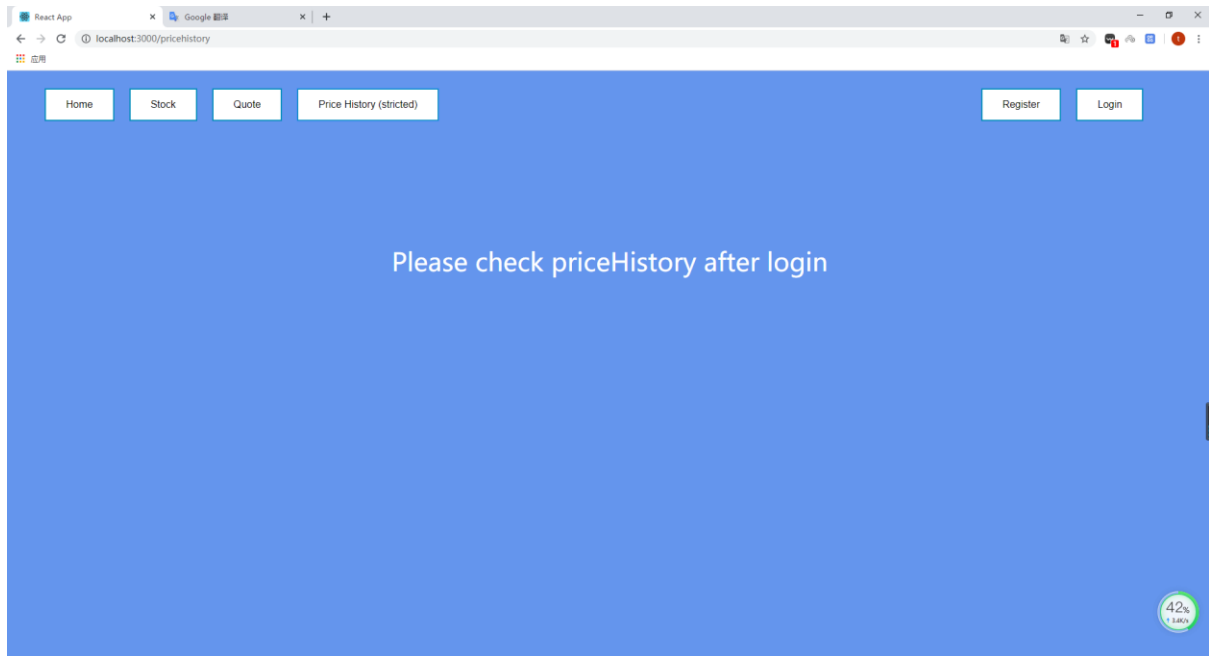
```
123  /* Display the table by follow code */
124
125  <AgGridReact
126    columnDefs={columns}
127    rowData={rowData}
128    pagination={true}
129    paginationPageSize={50}
130  />
131
132  <br></br>
133  <br></br>
134  <br></br>
135
136  /* display the line chart by foloww code */
137  <Line data={lineData}
138
139    //set title and legend and scales
140    options={{
141
142      //set title name and color
143      title:{
144        display: true,
145        text: "The line of latest Price Trendency ",
146        fontColor: "fff",
147        fontSize:17
148      },
149
150      //set legend size and color
151      legend:{
152        labels:{
153          fontColor: "fff",
154          fontSize: 15
155        }
156      },
157
158      //set axes and yAxes tick color
159      scales: {
160        yAxes: [{
161          ticks: {
162            fontColor: "fff",
163          }
164        }],
165        xAxes: [{
166          ticks: {
167            fontColor: "fff",
168          }
169        }],
170      }
171    }
172  />
```

49%

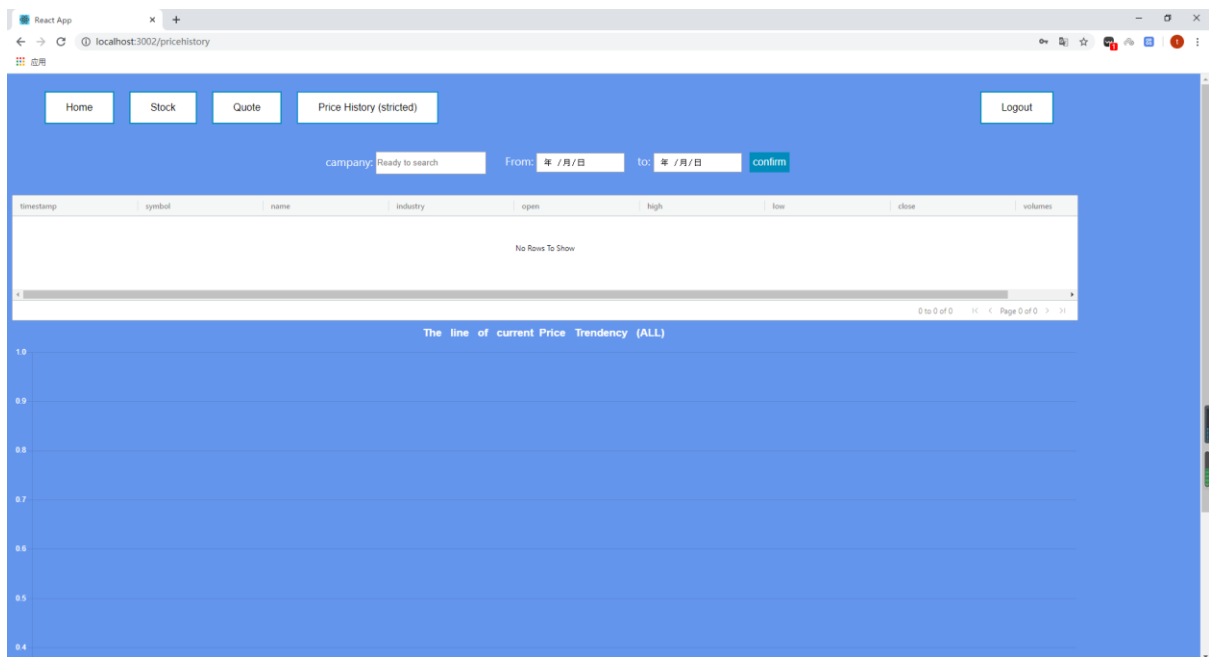
1/20/20

`/stocks/authed/{symbol}`

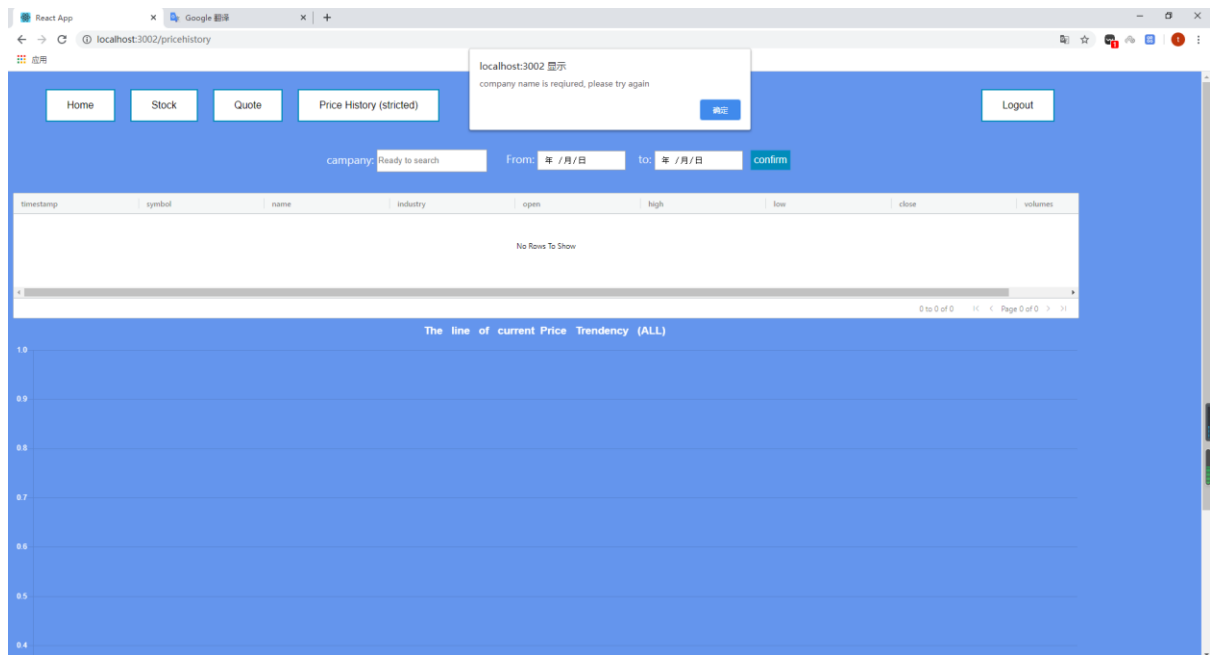
In the Third GET, it links to my price history page (`/pricehistory`). When I click 'price history (strict)' button, it will move on to 'price history' section. please see picture below. There is a heading to tell user if you want to check price history on my stock system, you have to login. Now, let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



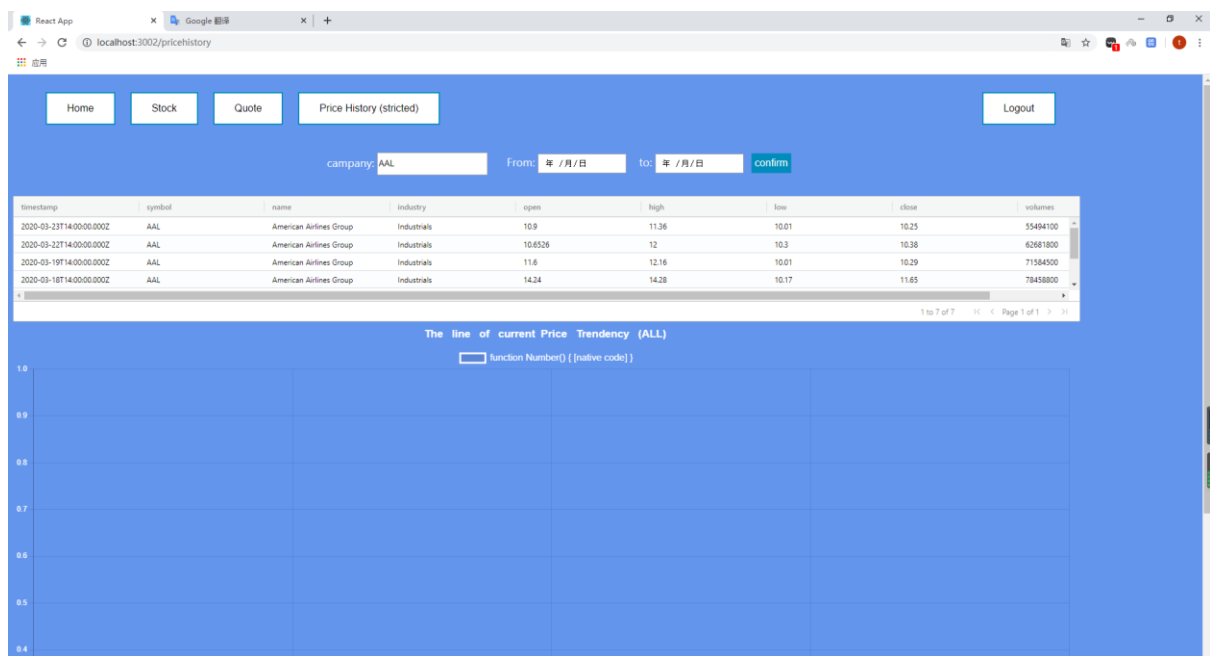
Now, we can see the difference. It has search bar, table and line char here. (in the date section, my computer system is not English, so detect my computer system). If I click without any character or date on the search bar. See what is happen.



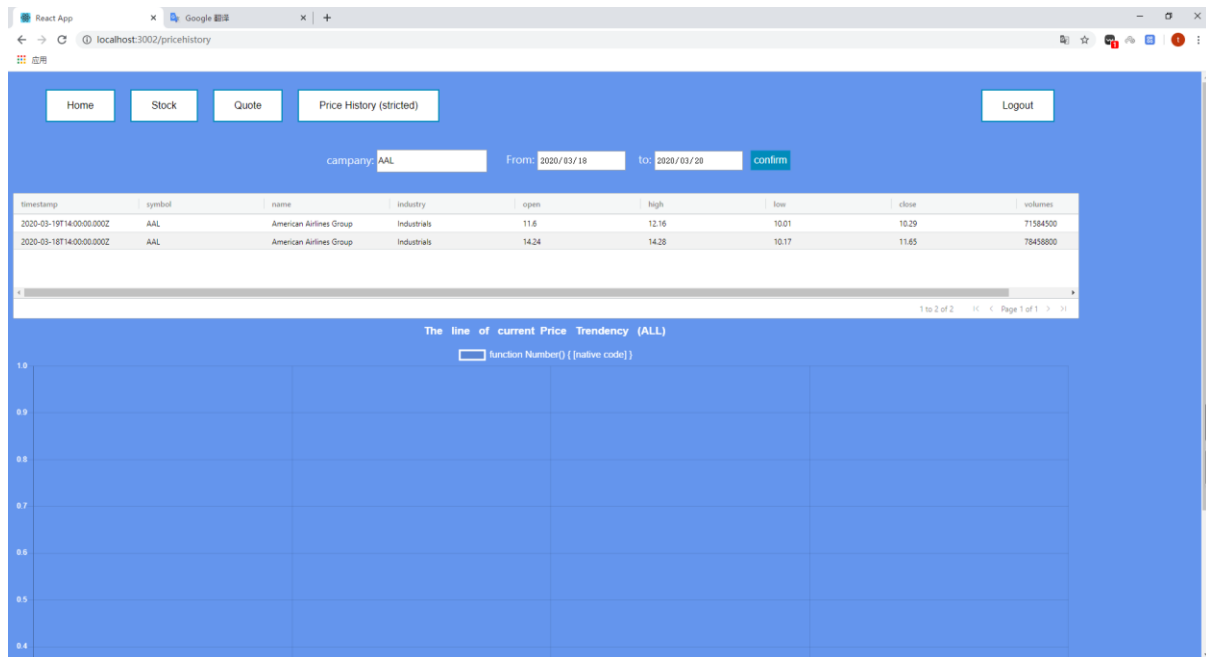
Now we could see that it involves an alert, which means that company name by symbol is required. Date is not required. Let me type a one of company name by symbol to see what is happen.



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.

```

1  import React, { useState, useEffect } from 'react'
2
3  import { AgGridReact } from 'ag-grid-react';
4  import 'ag-grid-community/dist/styles/ag-grid.css';
5  import 'ag-grid-community/dist/styles/ag-theme-balham.css';
6
7
8  import { Line } from 'react-chartjs-2'
9  //import { Login } from 'login.js';
10
11
12  export default function PriceHistory() {
13
14
15
16    //return error message from res
17    const [catchError, setCatchError] = useState({});
18    //set table row
19    const [rowData, setRowData] = useState([]);
20    //set line dataset
21    const [lineData, setlineData] = useState({});
22    //whether login
23    const [isLogin, setIsLogin] = useState(false);
24    //set search bar
25    const [innerSearch, setInnerSearch] = useState('');
26    //set previous date
27    const [from, setfrom] = useState('2020-03-15T00:00:00.000Z');
28    //set current date
29    const [to, setto] = useState('');
30
31
32    //const [lineData, setlineData] = useState({})
33
34    //check login state
35    useEffect(() => {
36      let token = localStorage.getItem('token')
37      if (!token) {
38        setIsLogin(false)
39      } else {
40        setIsLogin(true)
41      }
42    }, []);
43
44
45    //set searchbar by symbol
46    const handleClick = () => {
47
48      //only login
49      var token = localStorage.getItem('token')
50

```

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.

```

46 const handleClick = () => {
47   //only login
48   var token = localStorage.getItem('token')
49
50   //searchbar cannot be null
51   if (!innerSearch) {
52     alert('company name is required, please try again')
53     return
54   }
55
56   //url
57   const url = "http://131.190.87:3000/stocks/authed/${innerSearch}?from=${from}&to=${to}"
58
59   //fetch from res(only login)
60   return fetch(url, {
61     headers: {
62       accept: "application/json",
63       "Content-Type": "application/json",
64       Authorization: `Bearer ${token}`
65     }
66   })
67   .then((res) => res.json())
68   .then((res) => {
69
70     //if res return array, then put value into rowdata, error is null,
71     if (Array.isArray(res)) {
72       setRowData(res)
73       setErrorObject({})
74
75       const { open, high, low, close } = res
76       const lineArr = [ open, high, low, close ]
77       const lineHistory = {
78         labels: [ 'open', 'high', 'low', 'close' ],
79         datasets: [
80           {
81             label: 'Price',
82             data: lineArr,
83             pointBorderColor: "#fff",
84             borderColor: "#fff",
85             pointHoverBorderColor: "#fff",
86             borderJoinStyle: "miter",
87           }
88         ]
89       }
90     }
91   })
92   // i probably need second datasets i think
93 }

```

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

```

95 //otherwise rowdata is null, involve error message from res
96 } else {
97   setRowData({})
98   setErrorObject(res)
99 }
100
101 //columns of table
102 const columns = [
103   { headerName: "timestamp", field: "timestamp", sortable: true, filter: true },
104   { headerName: "symbol", field: "symbol", sortable: true, filter: true },
105   { headerName: "name", field: "name", sortable: true, filter: true },
106   { headerName: "industry", field: "industry", sortable: true, filter: true },
107   { headerName: "open", field: "open", sortable: true, filter: true },
108   { headerName: "high", field: "high", sortable: true, filter: true },
109   { headerName: "low", field: "low", sortable: true, filter: true },
110   { headerName: "close", field: "close", sortable: true, filter: true },
111   { headerName: "volumes", field: "volumes", sortable: true, filter: true }
112 ]
113
114 return (
115   <div>
116     //if login, then return below
117     {islogin ?
118       // table div
119       <div className="ag-theme-balham" style={{
120         width: 700,
121         height: 200,
122       }}>
123         {` companyname by symbol (it is compulsory) `}
124         <div className="searchbar-history">
125           <label htmlFor="company-name"> company: </label>
126           <input className="searchoption" type="text">
127         </div>
128       </div>
129     } :
130   )
131 )

```

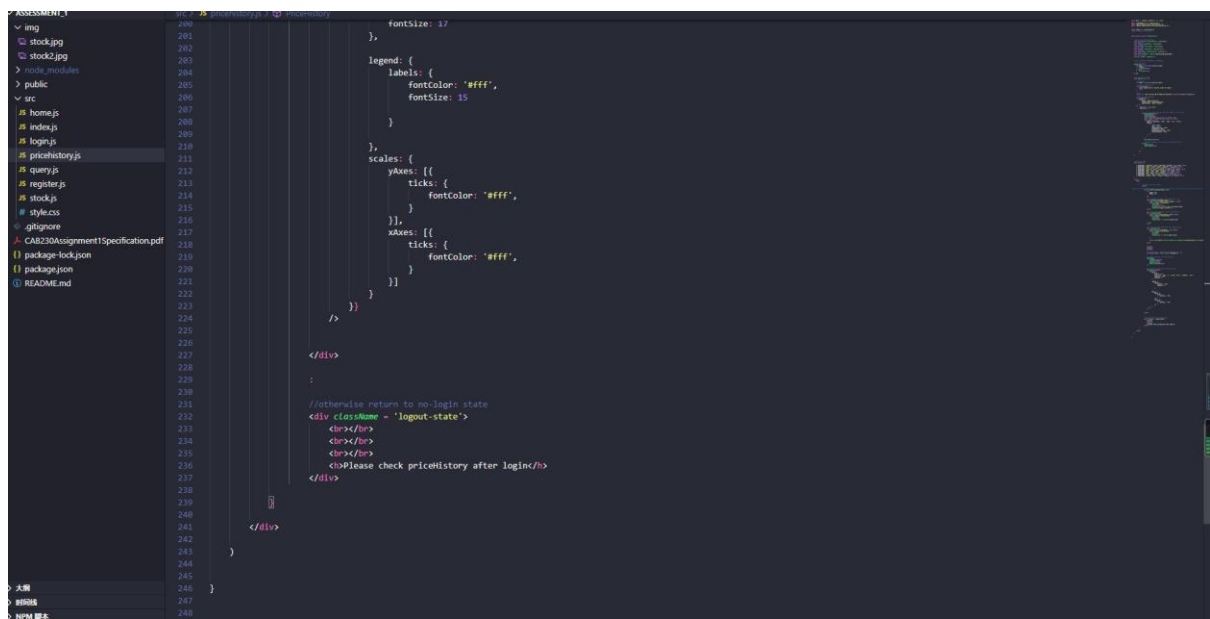
The layout code

```
126 //if login, then return below
127 islogin ?
128
129 // table size
130 <div className="ag-theme-balham" style=
131 {
132   width: 1700,
133   height: 200,
134 } >
135
136 /* companyname by symbol (it is compulsory) */
137 <div className="searchbar-history">
138   <label htmlFor="company-name"> company: </label>
139   <input className="searchoption"
140     type="text"
141     value={innerSearch}
142     onChange={e => setInnerSearch(e.target.value)}
143     placeholder="Ready to search" />
144 </div>
145
146 /* Initial date requirement (not the compulsory)*/
147 <div className="from">
148   <label htmlFor="previousdate"> From: </label>
149   <input className="previous-date"
150     type="date"
151     value={from}
152     onChange={e => setFrom(e.target.value)}
153   />
154 </div>
155
156 /* second date requirement (not the compulsory) */
157 <div className="to">
158   <label htmlFor="currentdate"> to: </label>
159   <input className="current-date"
160     type="date"
161     value={to}
162     onChange={e => setTo(e.target.value)}
163   />
164 </div>
165
166 <button type="button" onClick={handleClick} className="currentdate-button" id="currentdate">confirm/button>
167
168 </div>
169
170 </div>
```

Layout code of table and line-chart

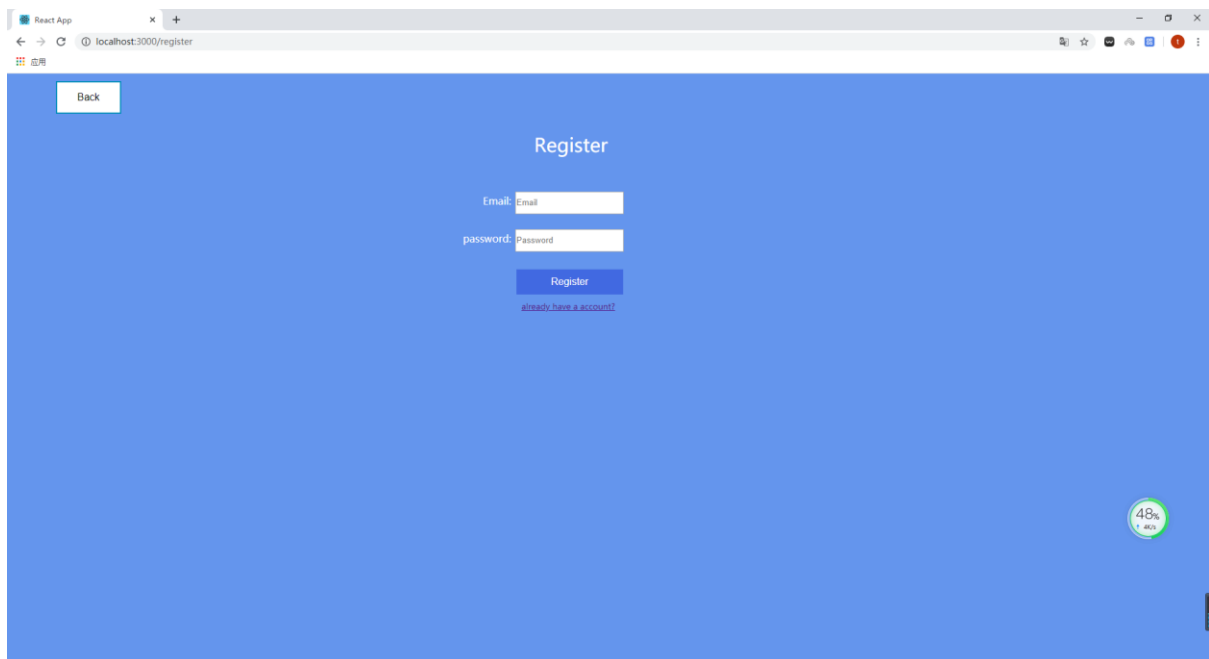
```
101 <div></div>
102 <div></div>
103 <div></div>
104
105 /* If have error , return error message from res */
106 (catchError.error ? <p>(catchError.message)</p> : '')
107
108 /* display the table by folloow code */
109 <AgGridReact
110   columnDefs={columns}
111   rowData={rowData}
112   pagination={true}
113   paginationPageSize={50}
114 />
115
116 /* display the line chart by folloow code */
117 <Line data={lineData}
118   options={{
119     title: {
120       display: true,
121       text: "The line of current Price Trendency (ALL) ",
122       fontColor: 'fff',
123       fontSize: 17
124     },
125     legend: {
126       label: (property) Chart.ChartLegendLabelOptions.fontColor?: Chart.ChartColor
127       fontColor: 'fff',
128       fontSize: 15
129     },
130     scales: {
131       yAxes: [{
132         ticks: {
133           fontColor: 'fff',
134         },
135       }],
136       xAxes: [{
137         ticks: {
138           fontColor: 'fff',
139         },
140       }],
141     },
142   }}
143 />
```

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

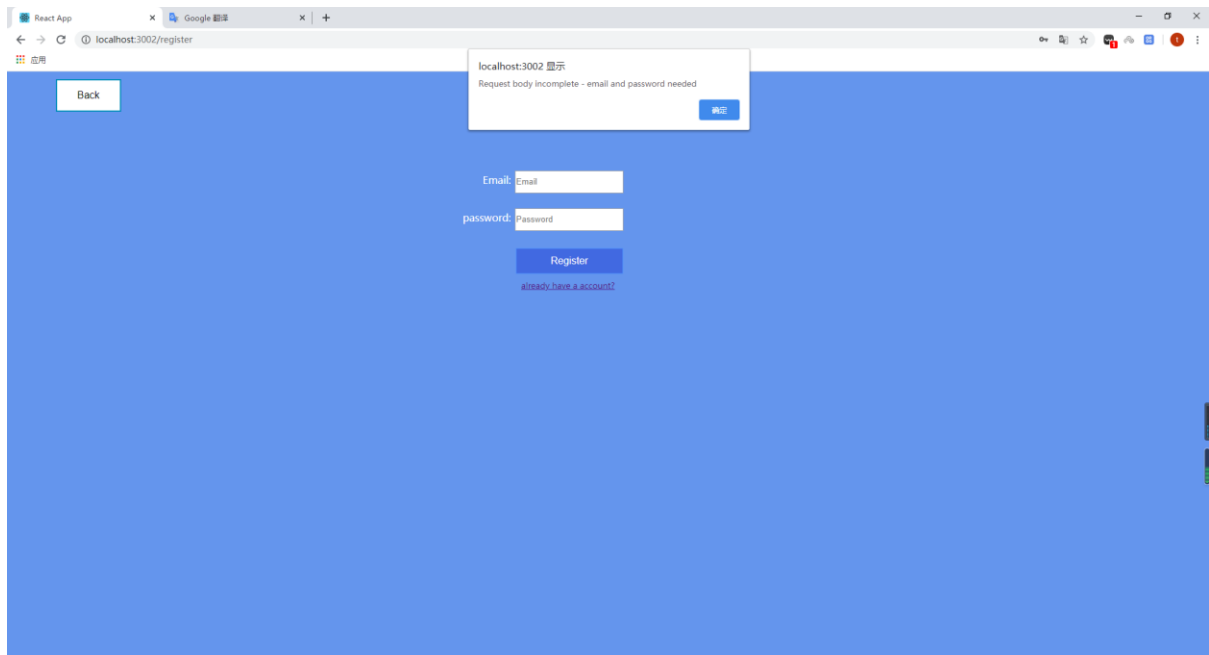


</user/register>

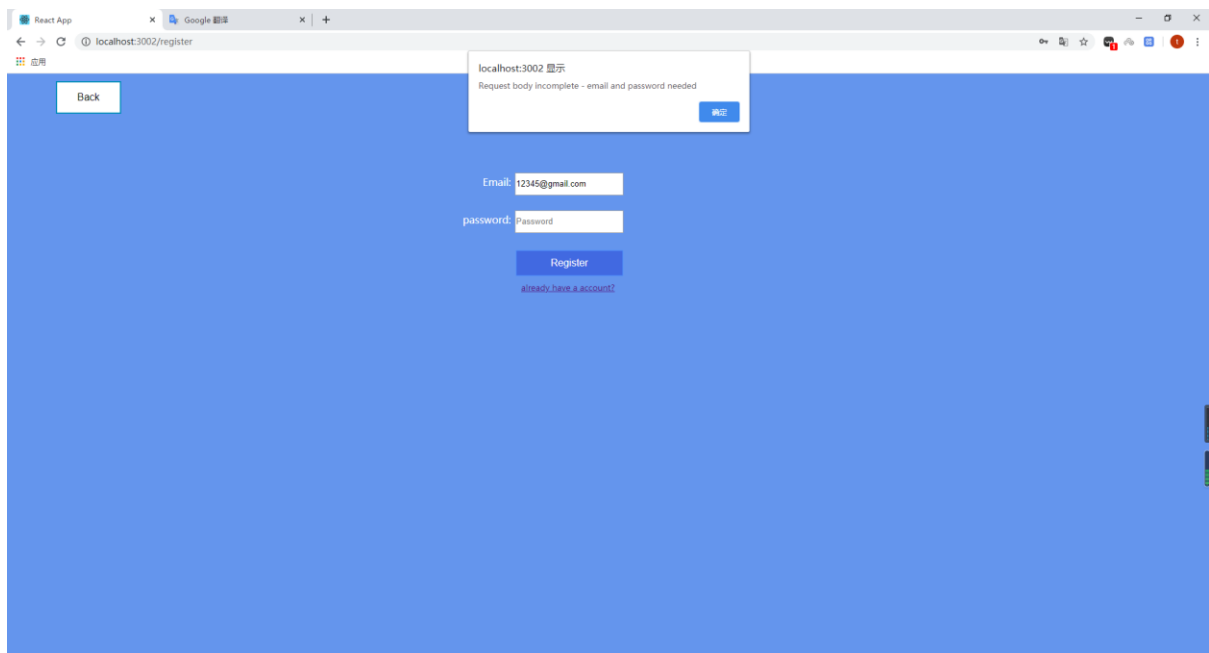
In the register POST, it links to my register page ('/register'). When I click 'Register' button, it will move on to 'register' section. please see picture below. We could see it is moving on a new page, which is '/register'. That is my register layout. It is simple but useful. On the left upper corner, 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.



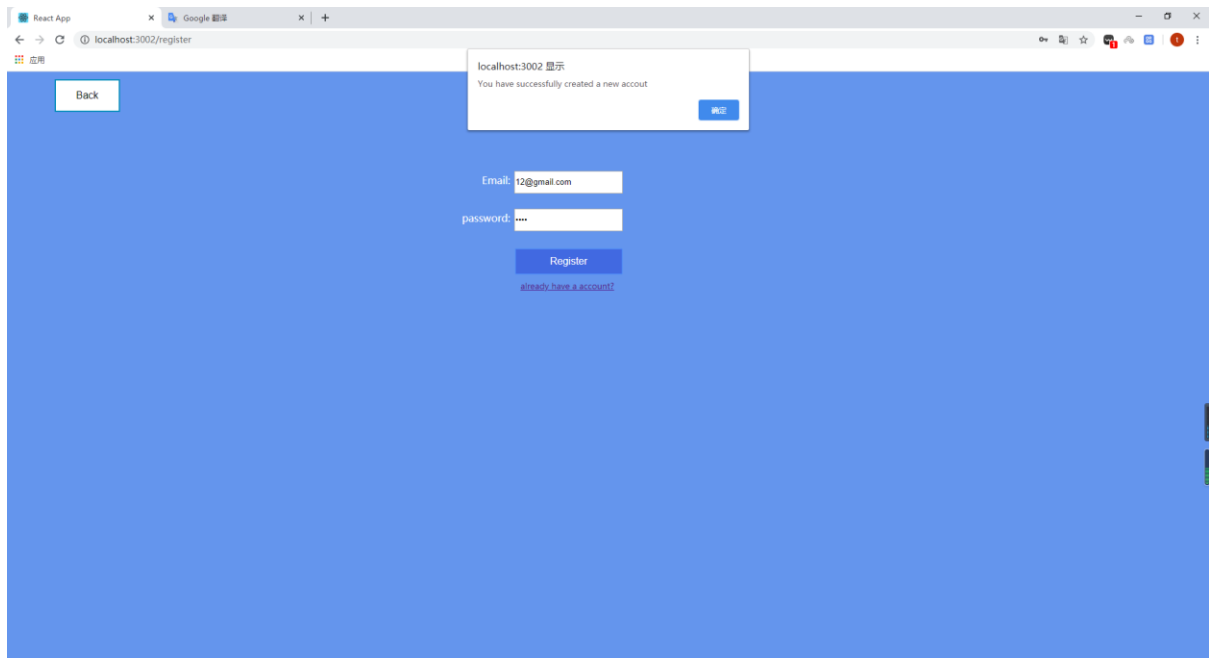
As we could see that it involves an alert, because I return error message from res as an alert. Now, I am going to type email without password to see what is happen.



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'.

```

src > JS register.js
1  import React, { useState } from "react";
2  import { useHistory } from "react-router-dom";
3  import { Link } from "react-router-dom";
4
5  function Register() {
6
7      //set email and password state
8      const [email, setEmail] = useState("");
9      const [password, setPassword] = useState("");
10
11      //if registered, then go to '/login'
12      const history = useHistory();
13
14
15      //get token
16      function getRegister() {
17
18          const url = "http://131.181.198.87:3000/user/register"
19
20          //get token
21          return fetch(url, {
22              method: "POST",
23              headers: { accept: "application/json", "Content-Type": "application/json" },
24              body: JSON.stringify({ email: email, password: password })
25          })
26          .then((res) => res.json())
27          .then((res) => {
28              //if have a error, then return alert on screen
29              if (res.error) {
30                  alert(res.message)
31              } else {
32                  //otherwise push to '/login'
33                  alert("You have successfully created a new account")
34                  history.push('/login')
35              }
36          })
37      }
38
39
40      return (
41
42          //email and password display and style
43          <div className="login-and-register">
44
45              { /* back to home page(link to '/') */ }
46              <div>
47                  <link to="/"><button className="button back-to-home">Back</button></link>
48              </div>
49

```

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

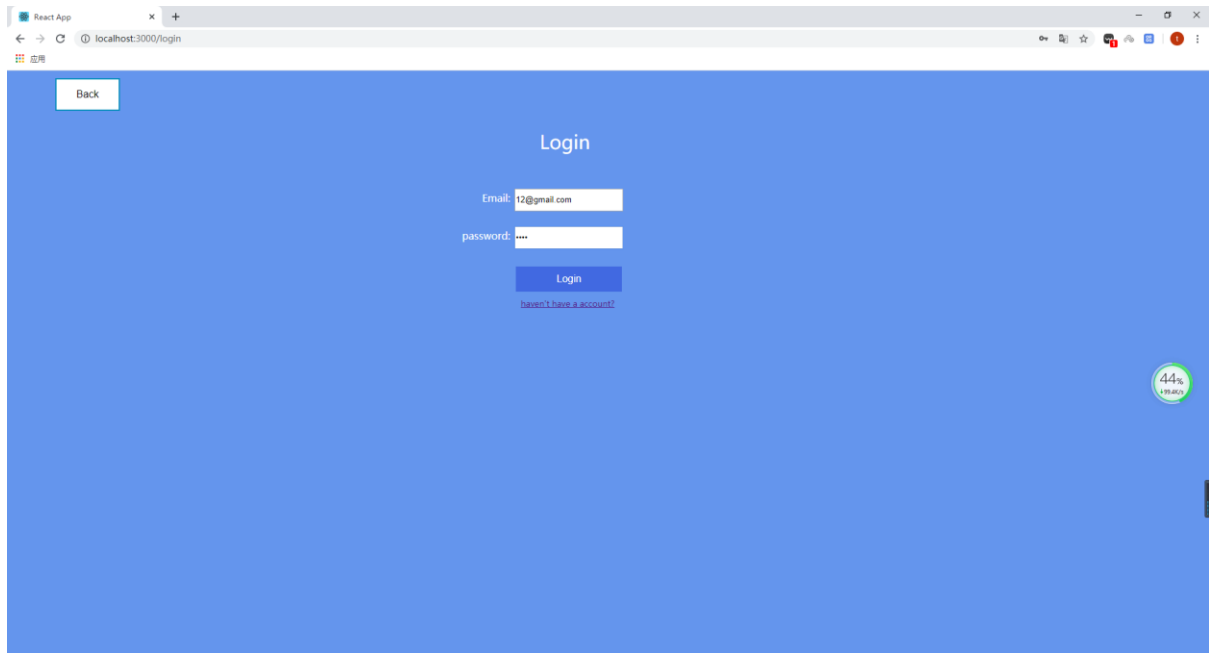
The screenshot shows a code editor interface with three main panels:

- Left Panel (File Explorer):** Displays a project structure. The root directory is "ASSIGNMENT_1". It contains several folders like "node_modules", "public", "src", and "dist". Under "public", there are files like "favicon.ico", "index.html", "manifest.json", "robots.txt", "home.js", "index.js", "login.js", "pricehistory.js", "query.js", "register.js", "stock.js", "style.css", and ".gitignore". There's also a PDF file named "CAB230Assignment1Specification.pdf". Under "src", there are files like "package-lock.json", "package.json", and "README.md".
- Main Panel (Code Editor):** Shows the content of a file named "register.js". The code is as follows:

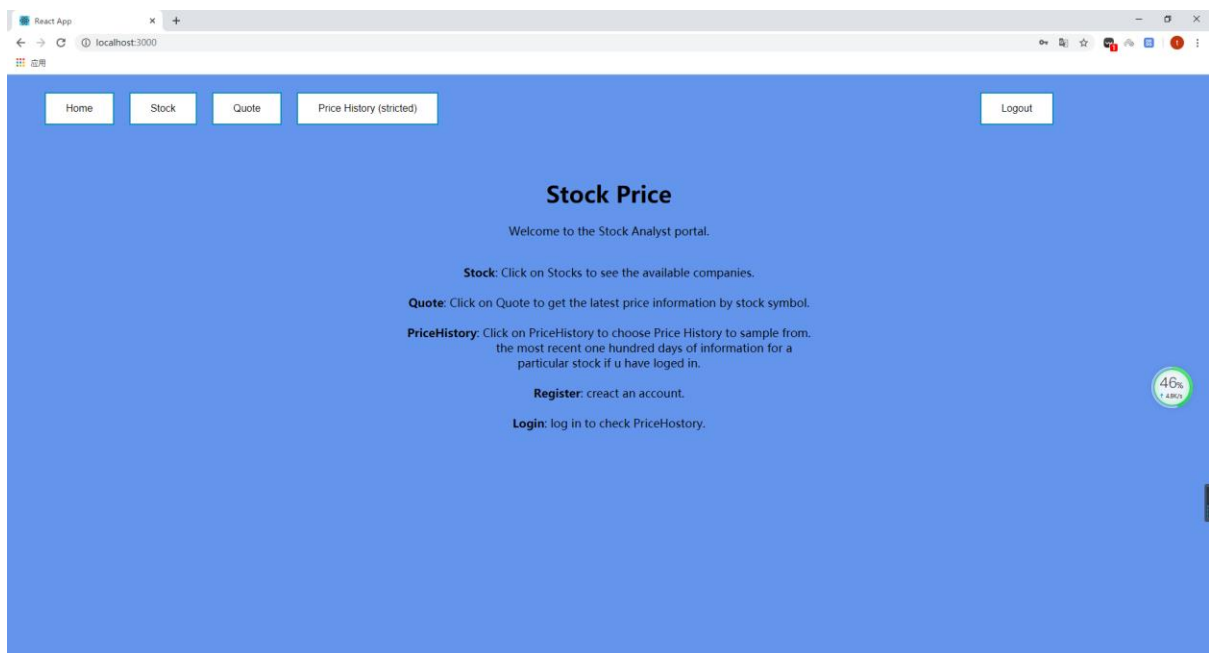
```
// register.js
<!-- Register -->
44 // mode modules
45 </script>
46 <script>
47 {/* back to home page(link to '/') */}
48 <div>
49 <link to="/" /><button className='button back-to-home'>Back</button></link>
50 </div>
51 <br/>
52 <br/>
53 <br/>
54
55 <div className='heading-register'>
56 <h2>Register</h2>
57 </div>
58
59 <br/>
60 <br/>
61
62 <!-- Email -->
63 <div className='username'>
64 <label className='login-register-label-register'>Email: </label>
65 <input className='username-txt'
66 type='text'
67 value={email}
68 onChange={e => setEmail(e.target.value)}
69 placeholder='email'
70 />
71 </div>
72
73
74 <br/>
75 <br/>
76
77 <!-- password -->
78 <div className='password'>
79 <label className='login-register-label-register'>password: </label>
80 <input className='password-txt'
81 type='password'
82 value={password}
83 onChange={e => setPassword(e.target.value)}
84 placeholder='Password'
85 />
86
87 </div>
88
89 <br/>
90 <br/>
91
92 {/* register button */}
```
- Right Panel (Search Bar):** Contains a search bar with the text "search" and a magnifying glass icon. Below it, there are search results, including a file named "register.js" with a snippet of code.

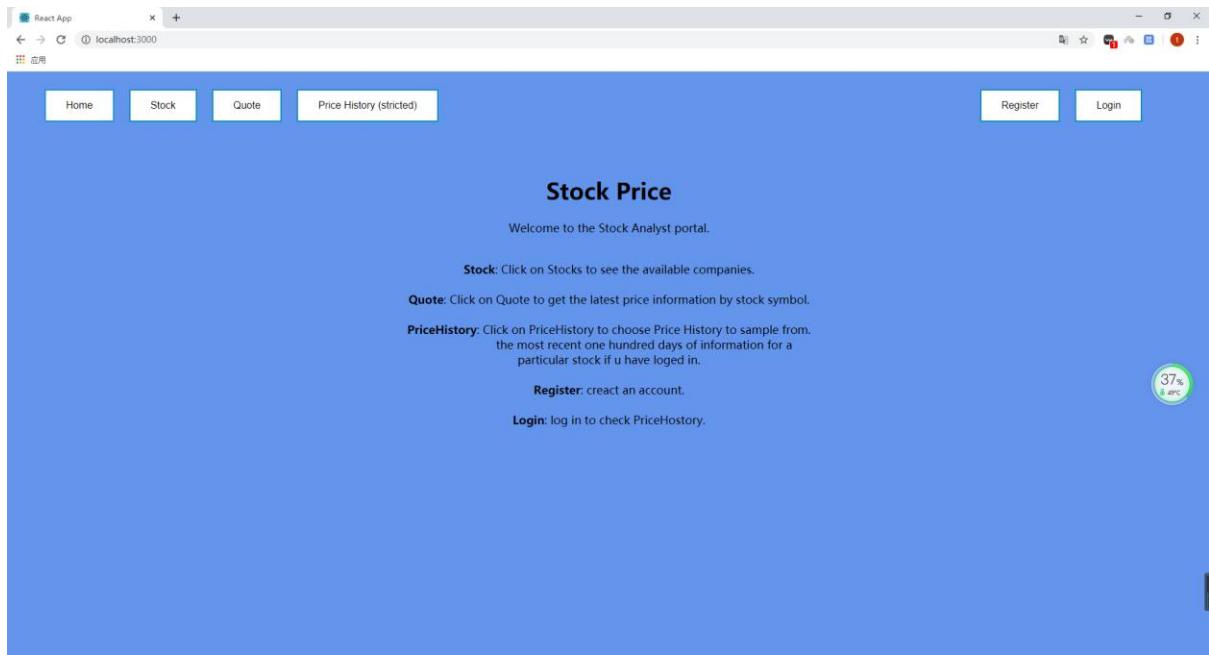
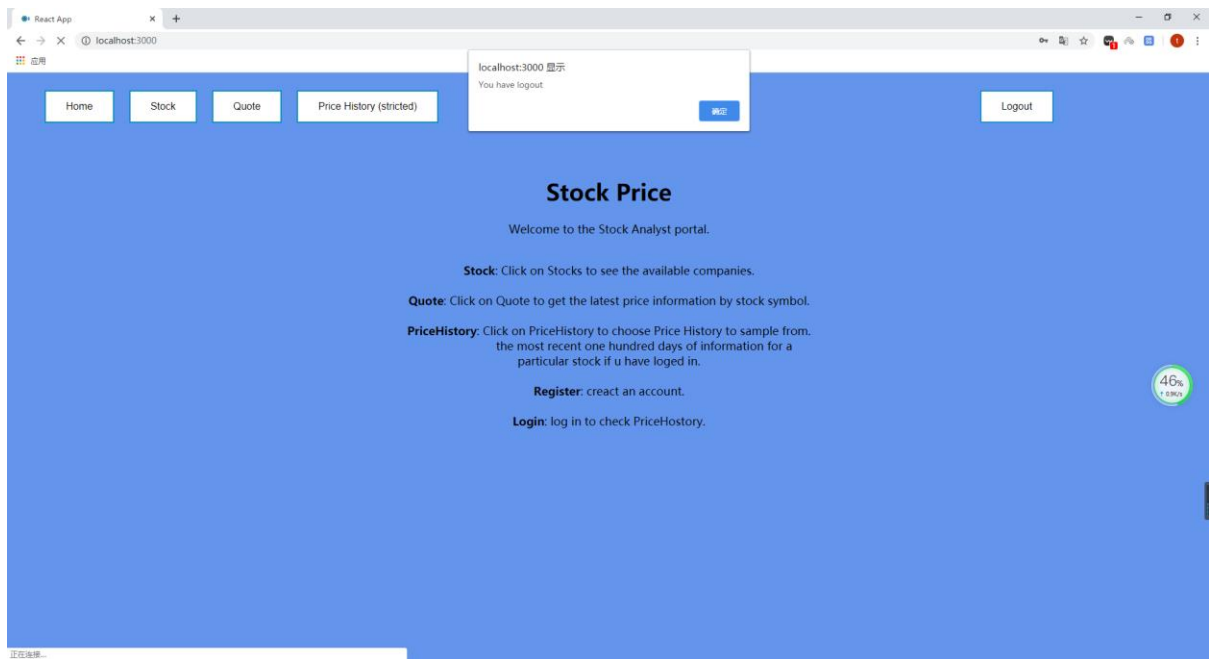
</user/login>

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'.

```

1 import React from "react";
2 import { useState } from "react";
3 //import jwt from "jsonwebtoken";
4 import { useHistory, withRouter } from "react-router-dom";
5
6 import { Link } from "react-router-dom";
7
8 export default function Login() {
9
10   //set email and password
11   const [emailState, setEmailState] = useState("");
12   const [passwordState, setPasswordState] = useState("");
13
14   //If login, click to '/' page(home)
15   const history = useHistory();
16
17
18
19   //exampleapi.com
20   //andikfjl
21
22   //set login, get token
23   function setLogin() {
24
25     //url
26     const url = "http://131.181.190.87:3000/user/login";
27
28     //get token
29     return fetch(url, {
30       method: "POST",
31       headers: { accept: "application/json", "Content-Type": "application/json" },
32       body: JSON.stringify({ email: emailState, password: passwordState })
33     })
34       .then((res) => res.json())
35       .then((res) => {
36         //if have a error, then return alert on display
37         if (res.error) {
38           alert(res.message)
39           //otherwise set token in localStorage, then declare token in anywhere
40         } else {
41           localStorage.setItem("token", res.token)
42           history.push("/")
43         }
44       })
45     )
46   }
47 }

```

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

```

57 <div className="login-and-register">
58   <div>
59     { /* back to home page(link to '/') */ }
60     <link to="/"><button className="button back-to-home">Back</button></Link>
61   </div>
62
63   <div>
64     <div className="heading-login">
65       <h2>Login</h2>
66     </div>
67
68     { /* email */ }
69     <div className="username">
70       <label className="login-register-label">Email: </label>
71       <input className="username txt"
72         type="email"
73         value={emailState}
74         onChange={e => setEmailState(e.target.value)}
75         placeholder="Email"
76       />
77     </div>
78
79     { /* password */ }
80     <div className="password">
81       <label className="login-register-label">password: </label>
82       <input className="password txt"
83         type="password"
84         value={passwordState}
85         onChange={e => setPasswordState(e.target.value)}
86         placeholder="Password"
87       />
88     </div>
89
90     <div>
91       <button type="button" className="button login">Login</button>
92       <button type="button" className="button register">Register</button>
93     </div>
94   </div>
95 </div>

```

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.

```

15  // Home.js : Qr Header
16
17  function Header() {
18
19    //If get token, then login
20    let token = localStorage.getItem('token');
21
22    //logout, clear token
23    function Logout() {
24      localStorage.clear();
25      window.location.href = '/';
26      alert('You have logout')
27    }
28
29    return (
30      //Header
31      <div className="Header">
32
33        <div className="LeftHeader">
34          <ul>
35            <li><a href="/"/><button className="button Home">Home</button></a></li>
36            <li><a href="/stock"><button className="button Stock">Stock</button></a></li>
37            <li><a href="/query"><button className="button Quote">Quote</button></a></li>
38            <li><a href="/pricehistory"><button className="button PriceHistory">Price History (stricted)</button></a></li>
39          </ul>
40        </div>
41
42        { /* If get token from res, then display logout button, otherwise display register and login */
43        token ?
44          <div className="rightHeader">
45            <ul>
46              <li><a href="#" onClick={Logout} className="button login">logout</a></li>
47            </ul>
48          </div>
49          :
50          <div className="rightHeader">
51            <ul>
52              <li><a href="/register"><button className="button Register">Register</button></a></li>
53              <li><a href="/login"><button className="button login">login</button></a></li>
54            </ul>
55          </div>
56        }
57      </div>
58    )
59  }
60
```

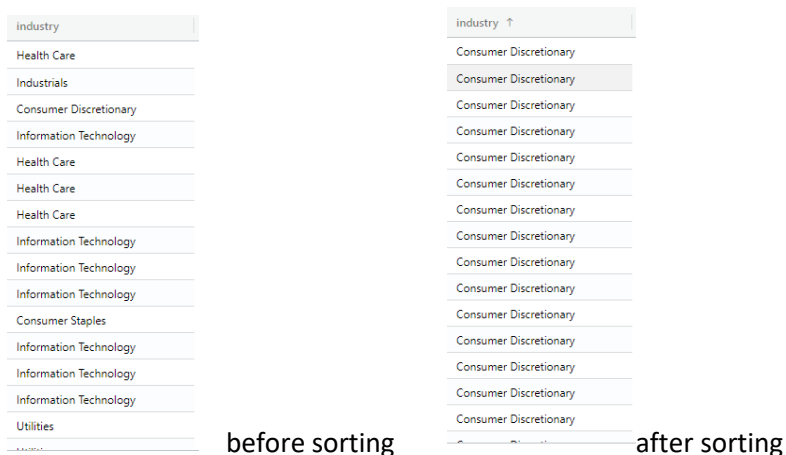
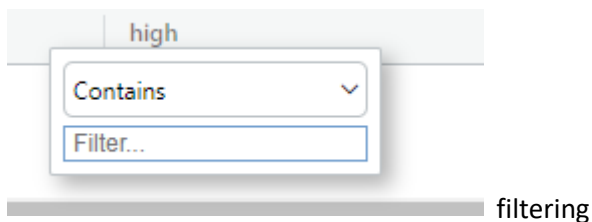

Modules used

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.



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.



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.

[Home](#) [stock](#) [Query](#) [Pricehistory](#)

[Register](#) [Login](#)

Insturction of stock system

stock

Query

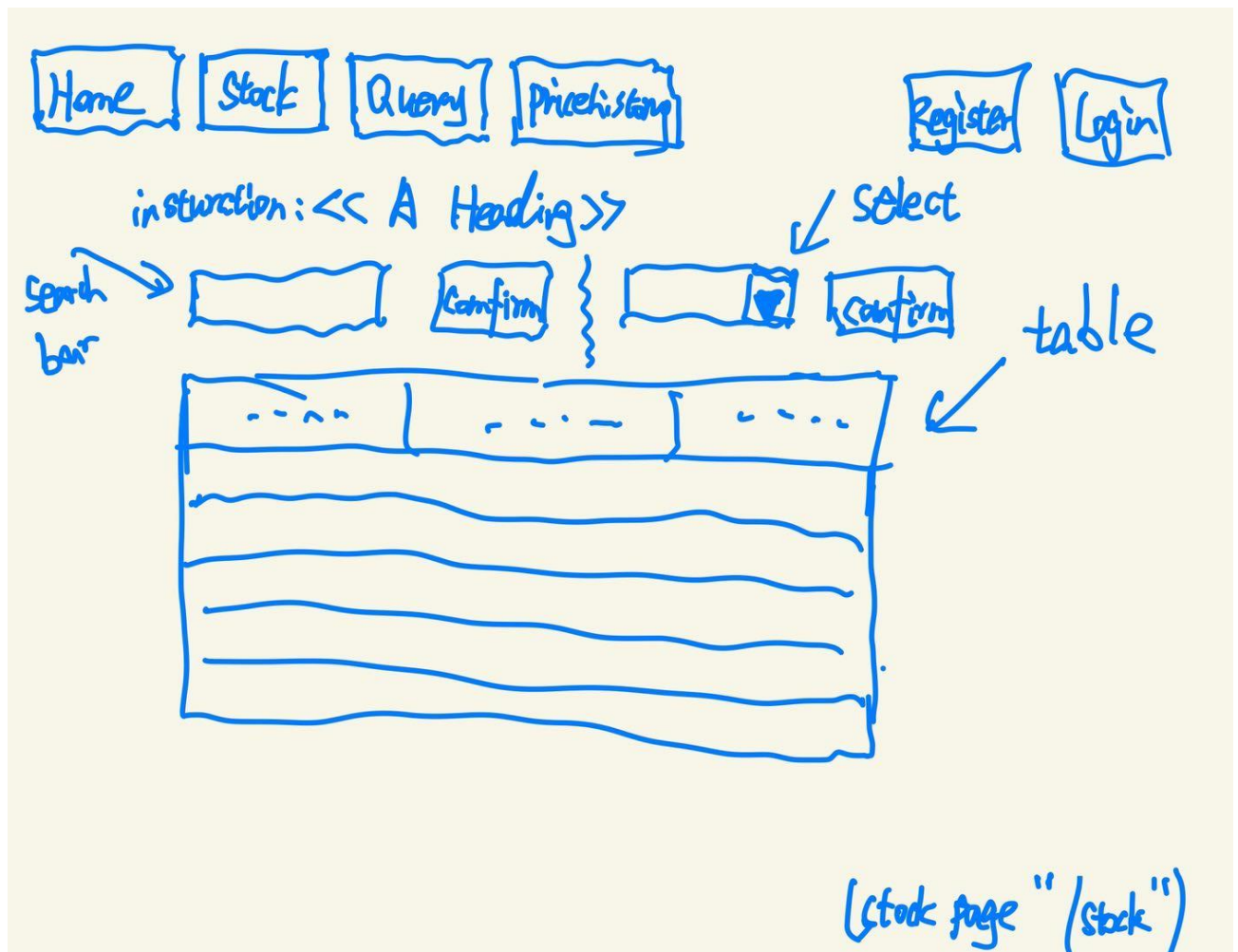
pricehistory

Register

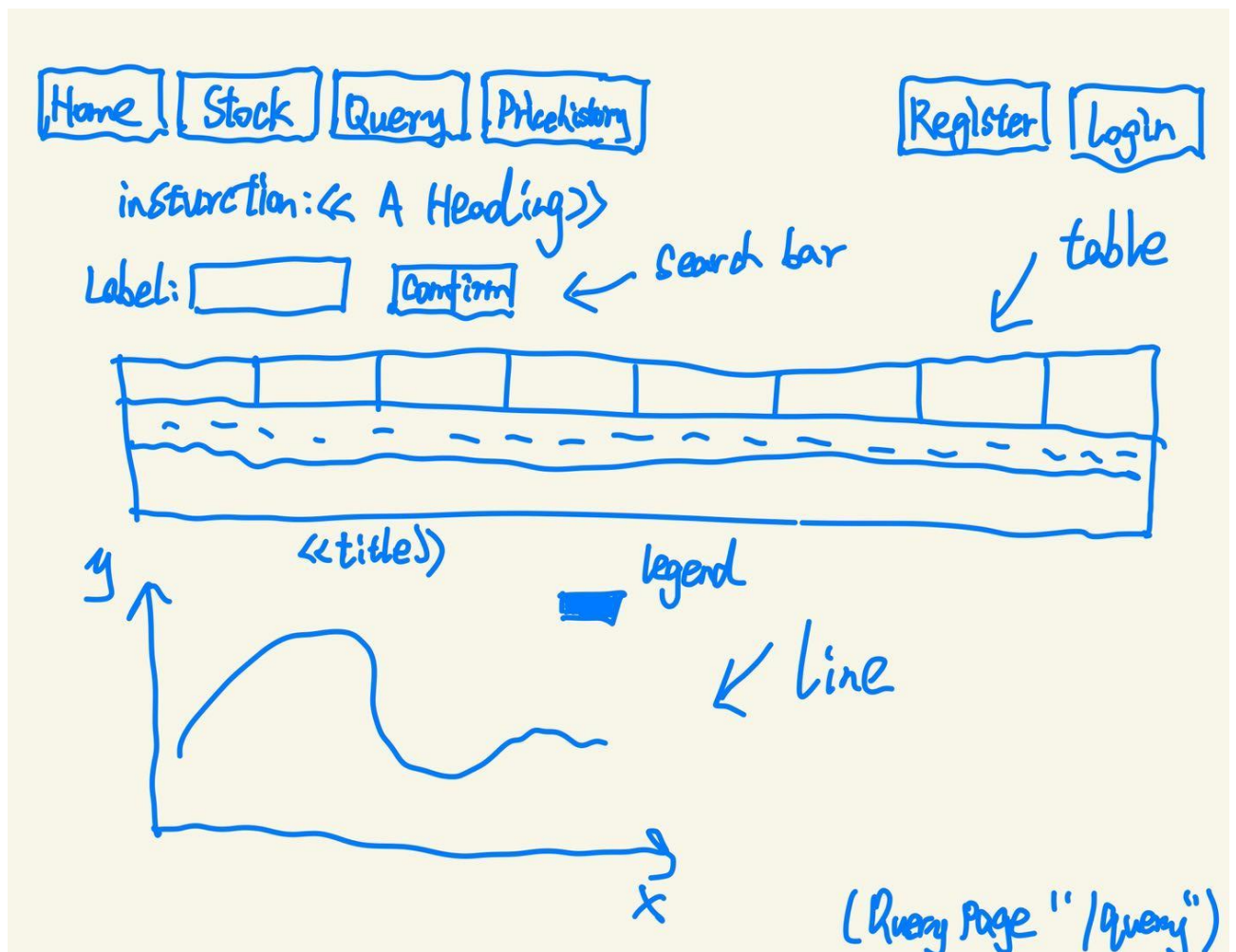
Login

(Home page "/")

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.

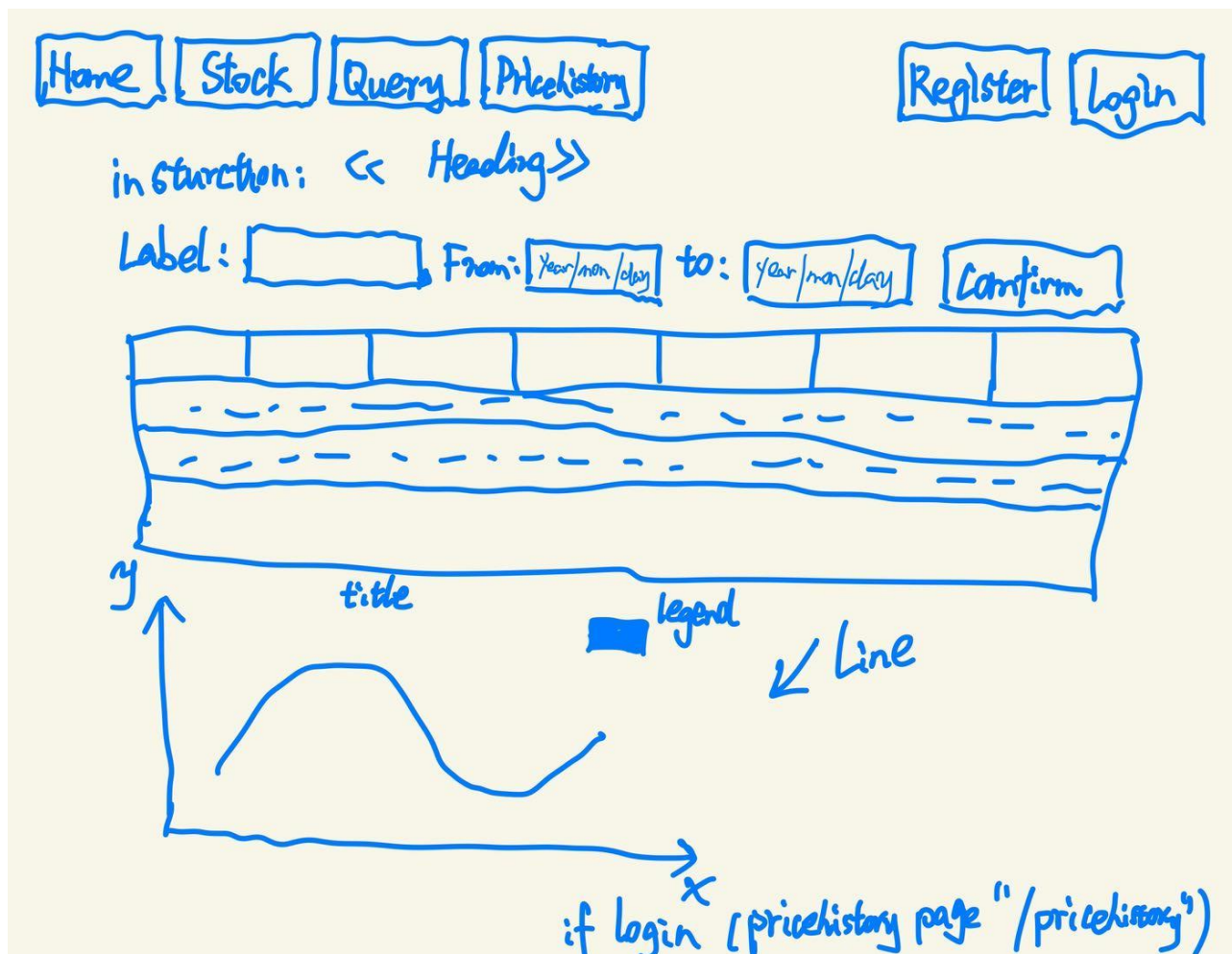
Home Stock Query Pricehistory

Register Login

A Heading → check price history after log in

if log out (price history page "/pricehistory")

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.

Register

Email:

Password:

(Register page "/register")

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.

Back

Register

Email:

Password:

Already have an account?

← click it

(Register page "/register")

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.

Log'in

Email:

Password:

(Log'in page "/>

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

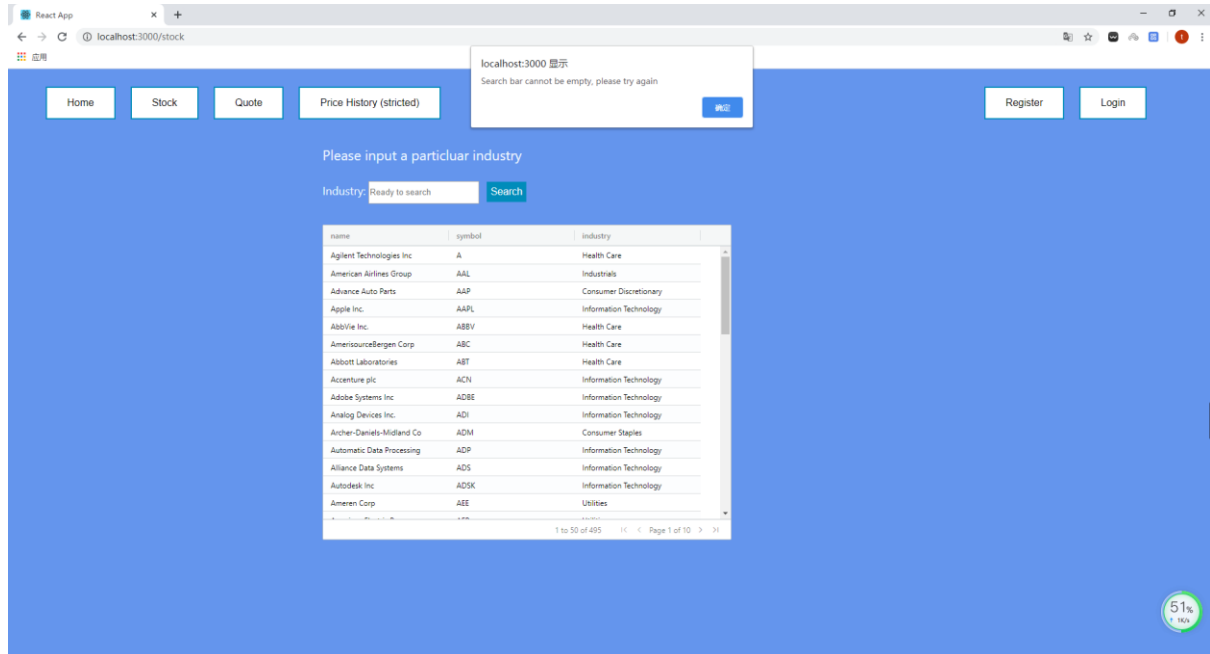
← click it

(login page "/login")

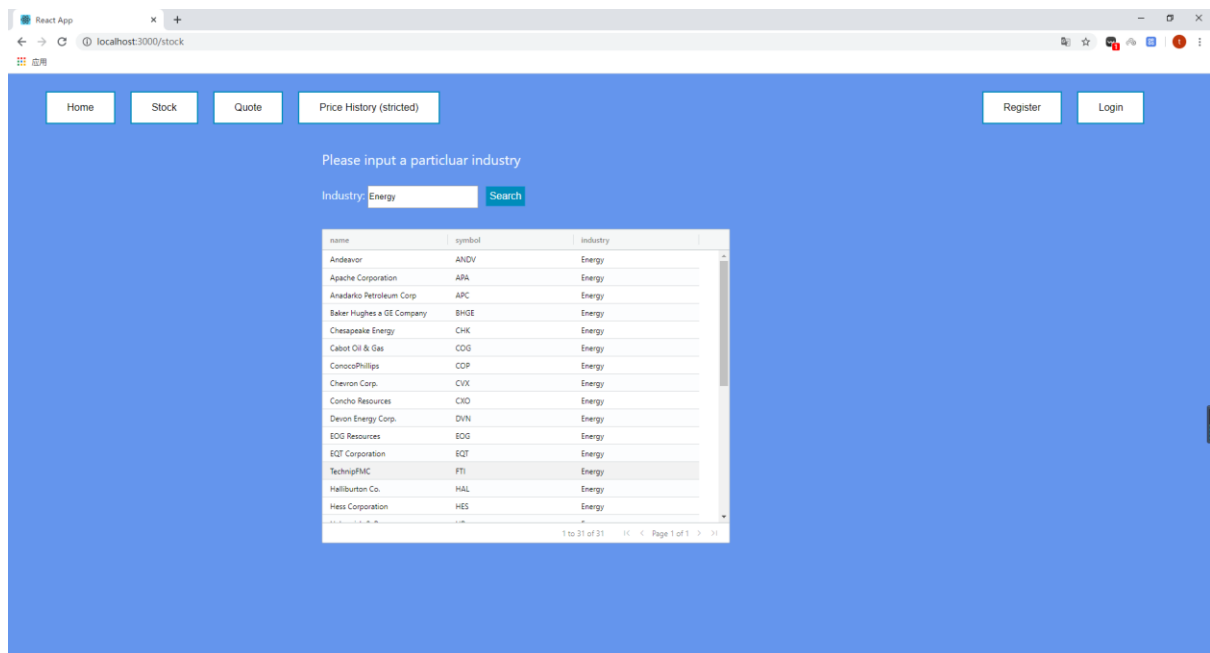
Test Plan

Stock:

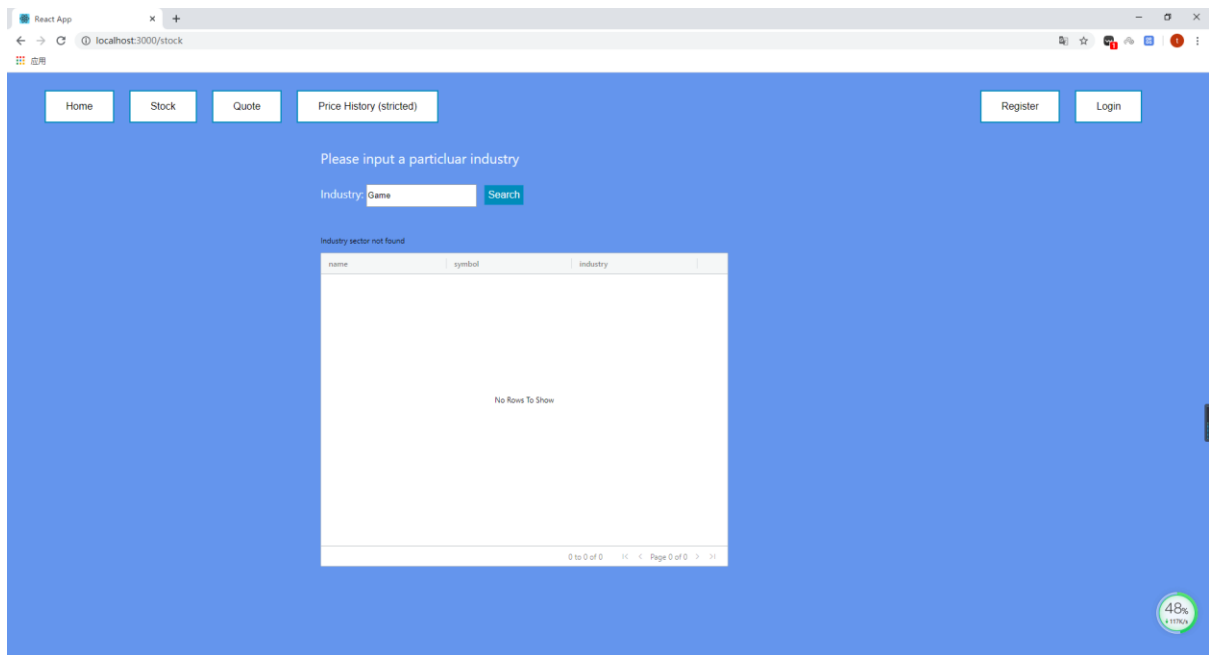
Test empty value pass



Test Industry name Pass

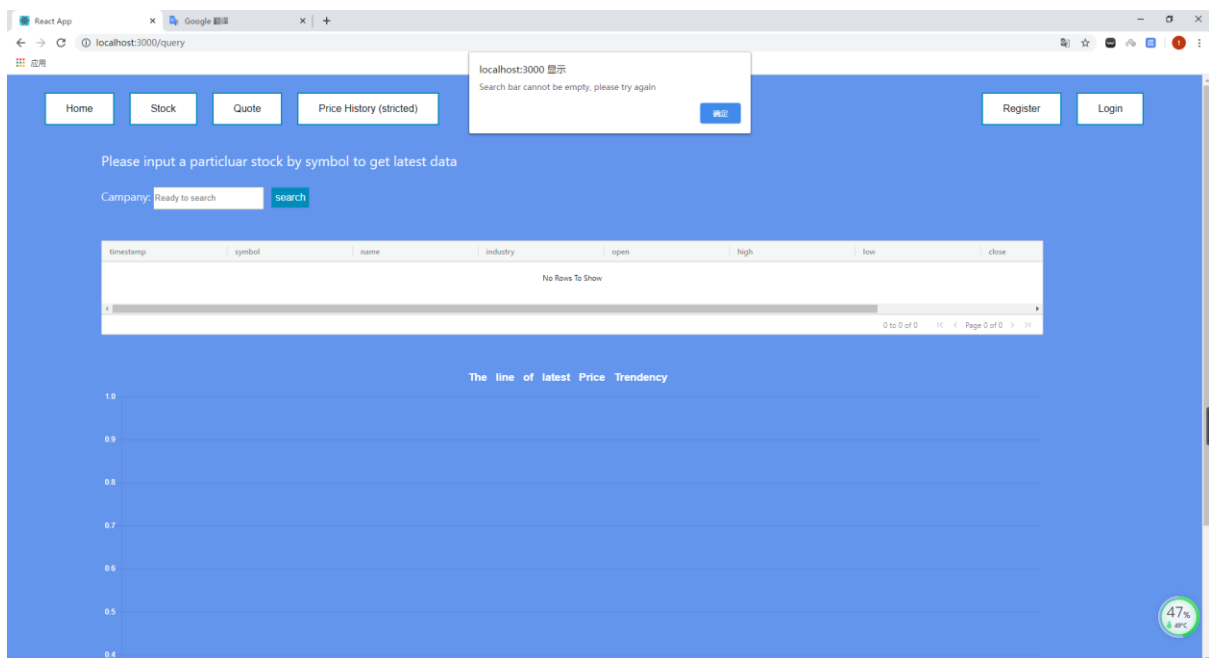


Test industry name doesn't exist Pass



Query

Test empty value Pass



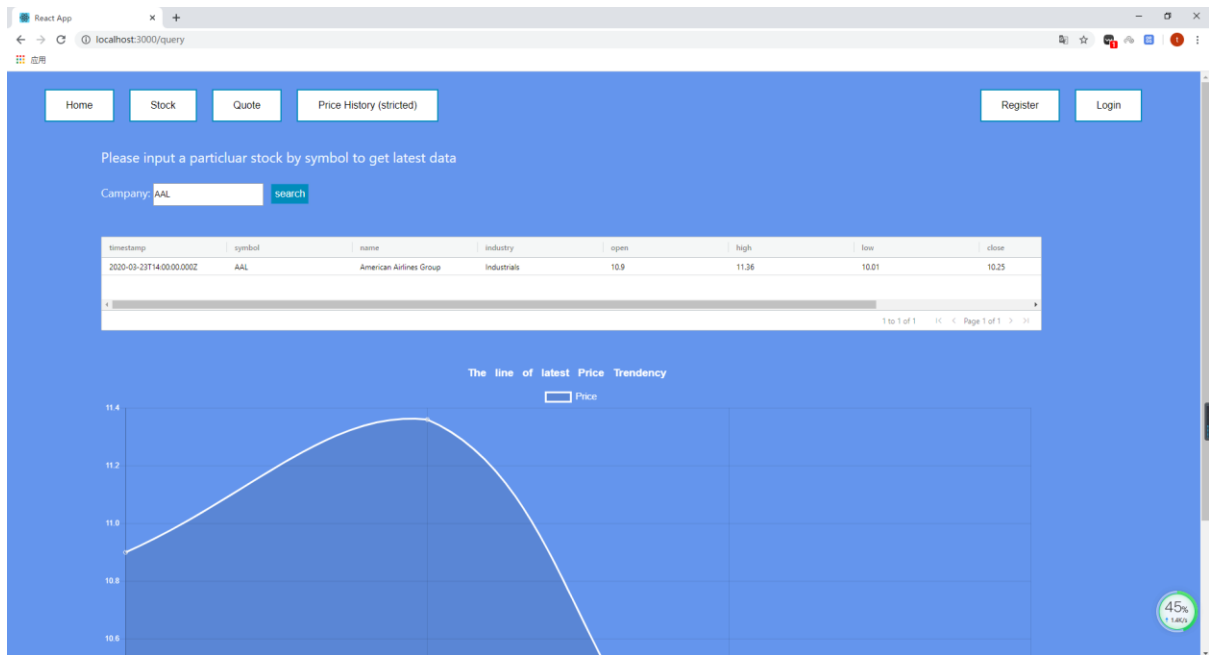
Test doesn't exist value Pass

The screenshot shows a web application interface with a blue background. At the top, there are navigation buttons: 'Home', 'Stock', 'Quote', 'Price History (stricted)', 'Register', and 'Login'. Below these, a prompt says 'Please input a particular stock by symbol to get latest data'. A search bar contains the text 'ASDF' and a 'search' button. Below the search bar, a message states 'No entry for symbol in stocks database'. Underneath this message is a table with columns: 'timestamp', 'symbol', 'name', 'industry', 'open', 'high', 'low', and 'close'. The table body is empty, with the text 'No Rows To Show' in the center. Below the table is a line chart titled 'The line of latest Price Trendency'. The y-axis of the chart ranges from 0.5 to 1.0. In the bottom right corner, there is a green circular badge with '48%' and '¥ 2.81M'.

Test more than 5 letters Pass

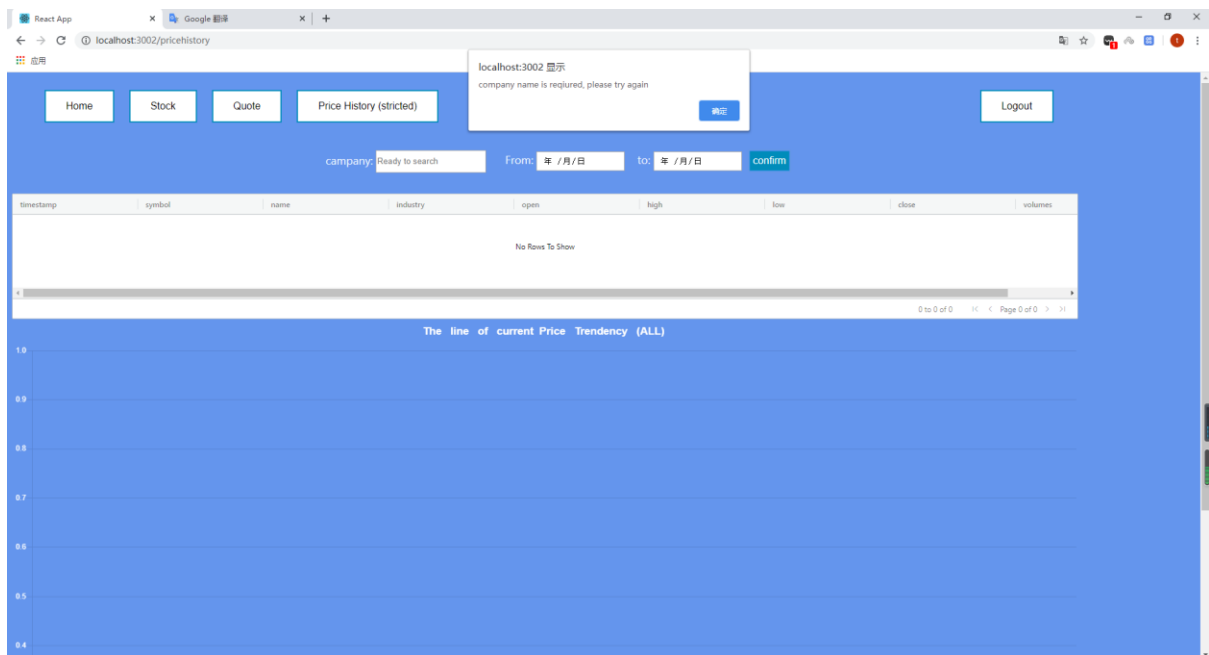
The screenshot shows the same web application interface as the previous one. The search bar now contains the text 'ASDFDD'. Below the search bar, a message states 'Stock symbol incorrect format - must be 1-5 capital letters'. The table below this message is also empty, with 'No Rows To Show' in the center. The line chart titled 'The line of latest Price Trendency' is still present, with the y-axis ranging from 0.5 to 1.0. In the bottom right corner, there is a green circular badge with '47%' and '¥ 2.81M'.

Test correct name return correct value in table and chart Pass



Price History:

Test empty value Pass

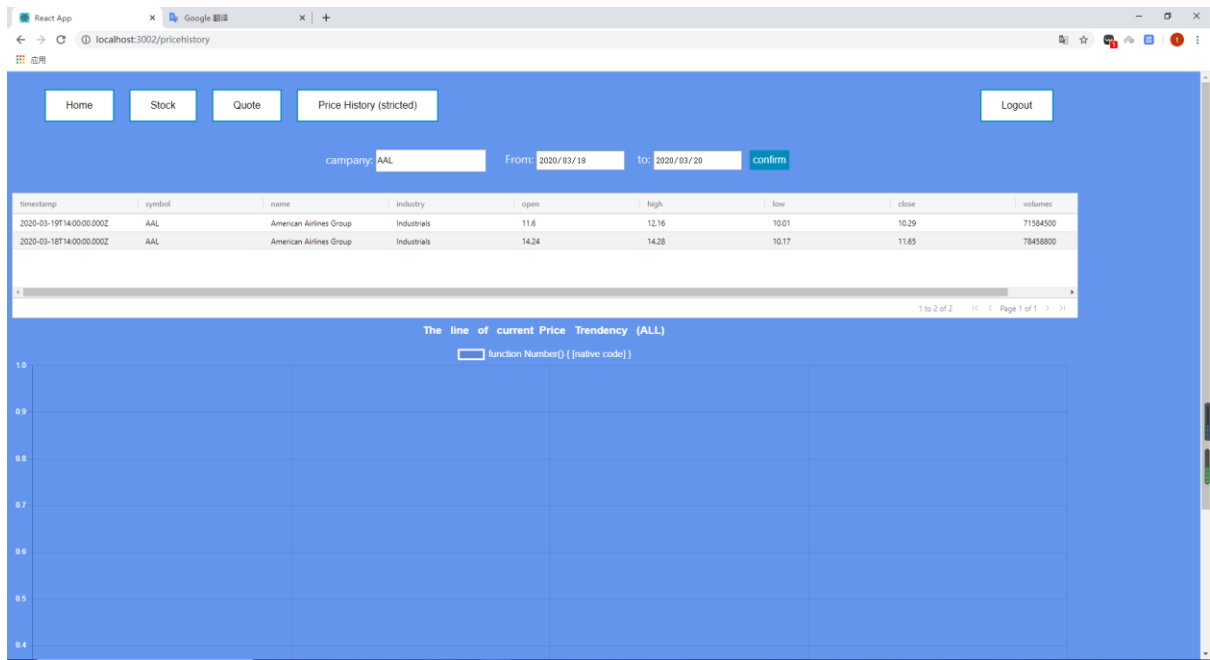


Test company with data return value in table

Pass

return in line chart

decline

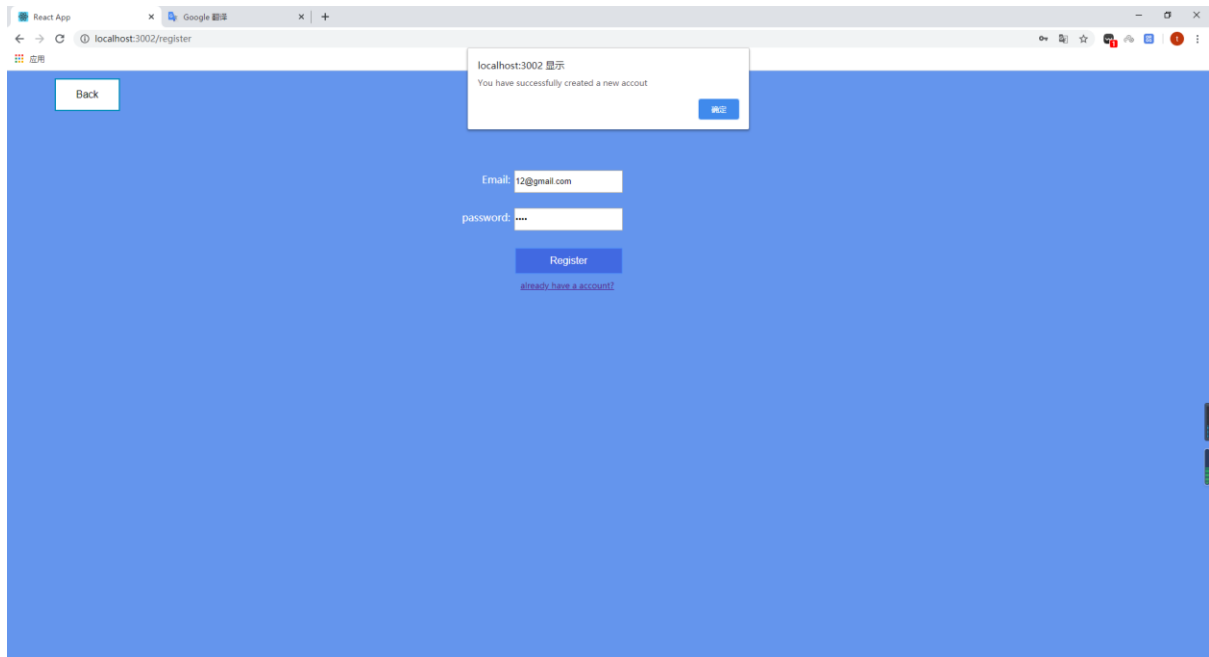


Register:

Test empty value pass

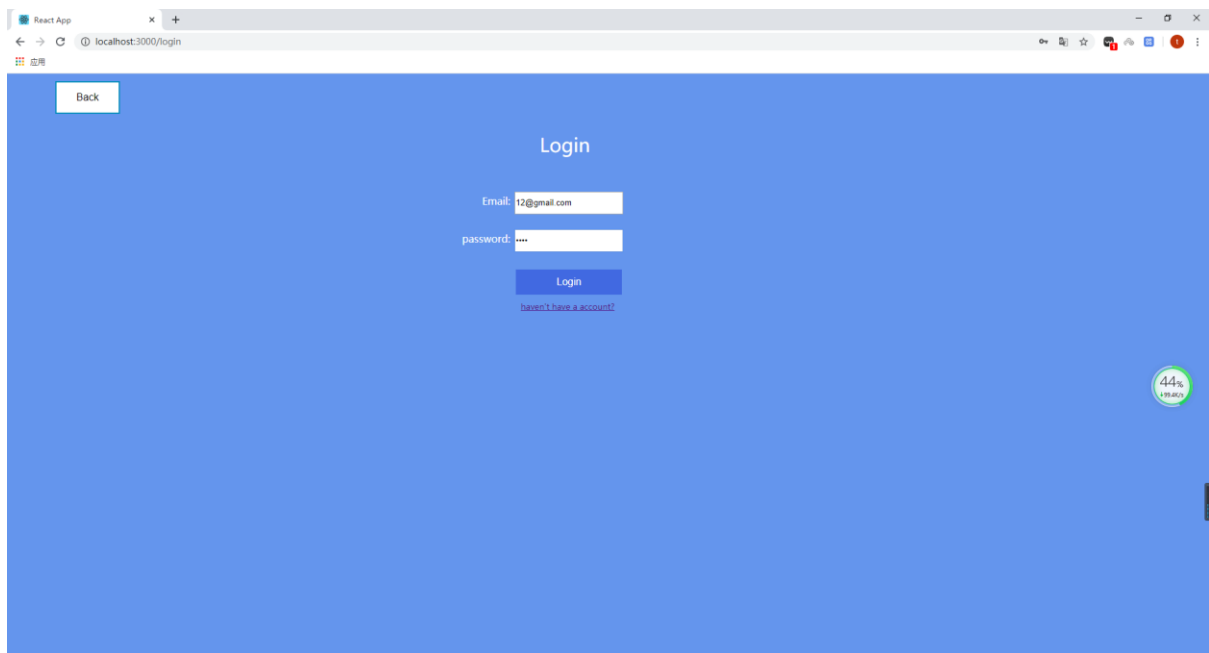
The screenshot shows a web application interface for a registration form. At the top, there is a 'Back' button. The form has two input fields: 'Email: Email' and 'password: Password'. Below these fields is a 'Register' button and a link 'already have a account?'. An error message is displayed in a white box with a blue border: 'localhost:3002 显示' and 'Request body incomplete - email and password needed'. The background of the page is blue.

Register a real account [pass](#)

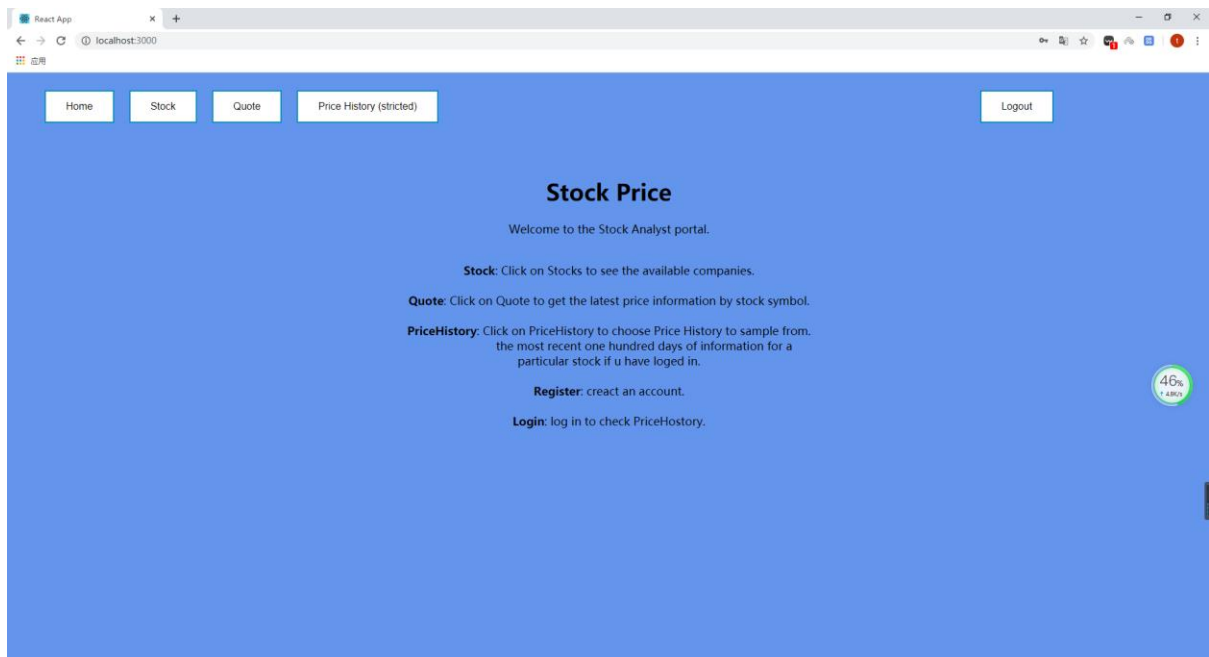


Login:

Successfully login an account just registered [Pass](#)



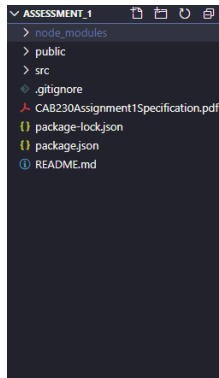
Login In Pass



Technical Description

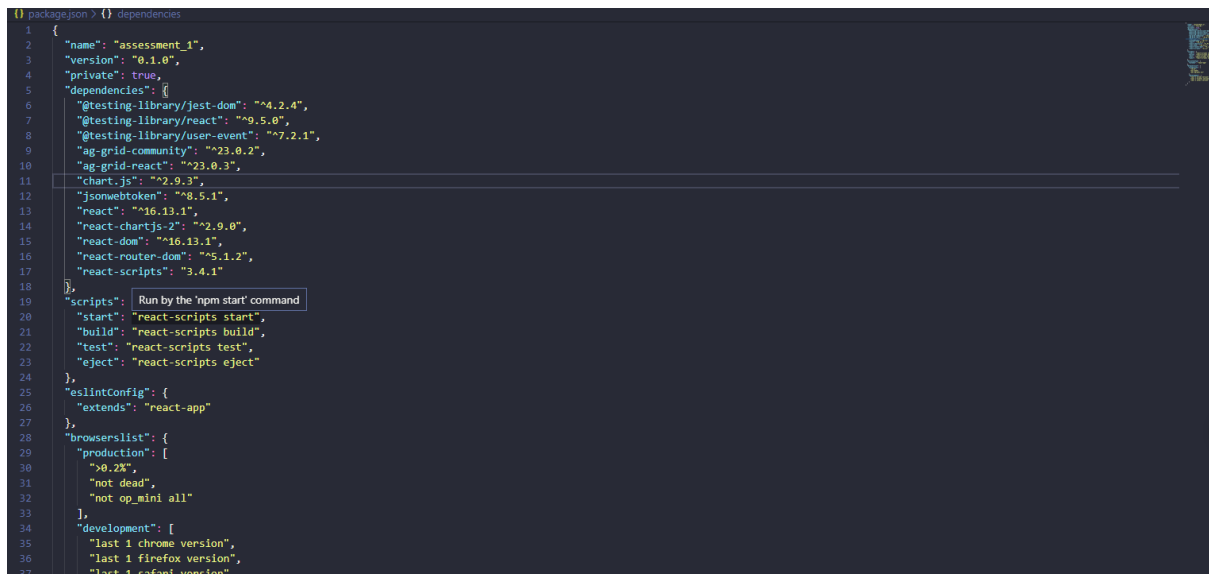
Architecture

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:

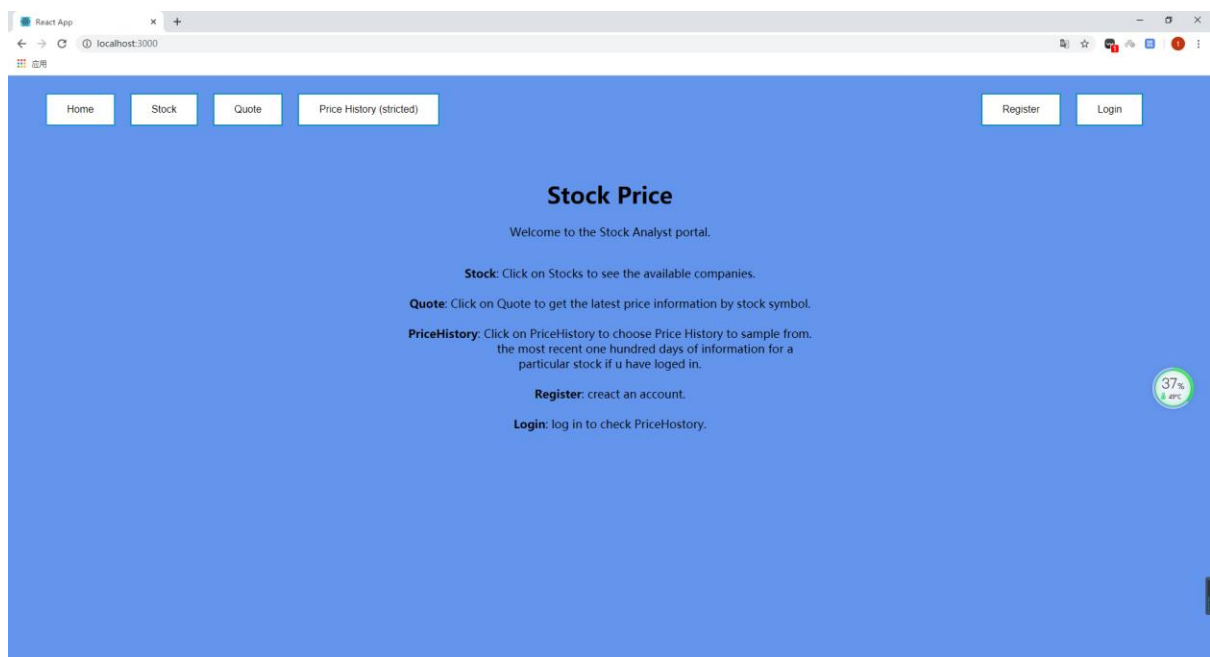
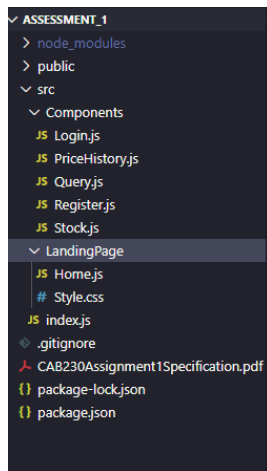


Assessment is the main folder, it including node-modules, public and src folders and assignment specification and package.json.

Package.json 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.

- 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?

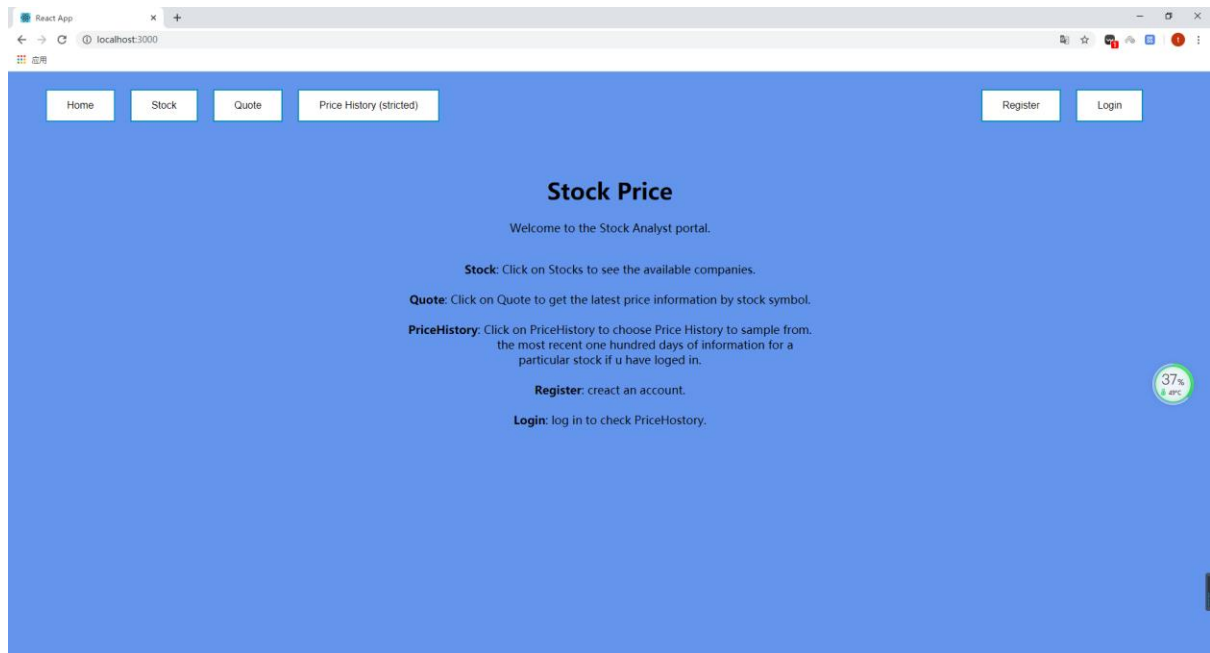
Not too much, I have already talk on other topic or question.

User guide

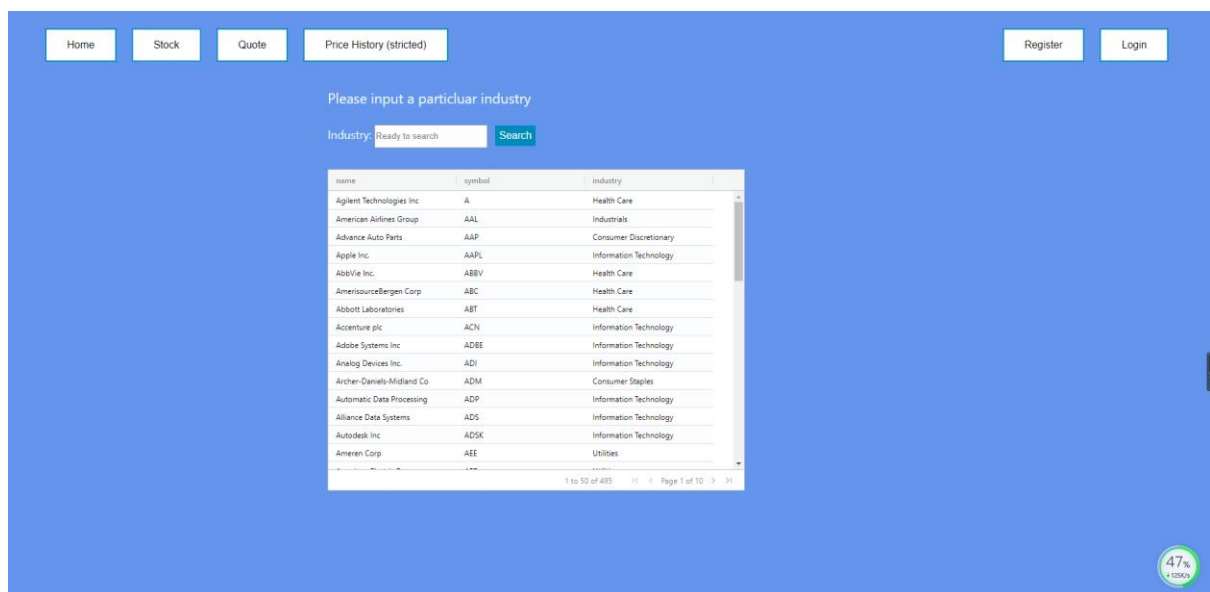
Tell us how to use your application

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

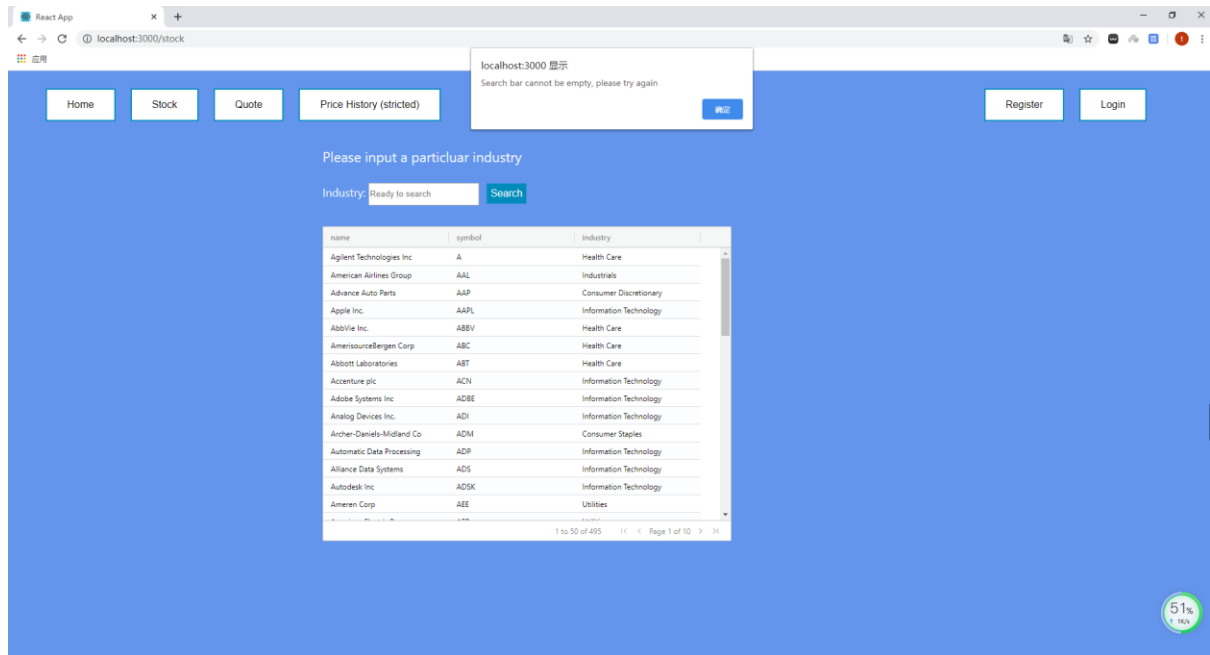
This is my home page layout. If you start from vs-code. It will link that page. In addition, I added in instruction in the center to explain each button usage.



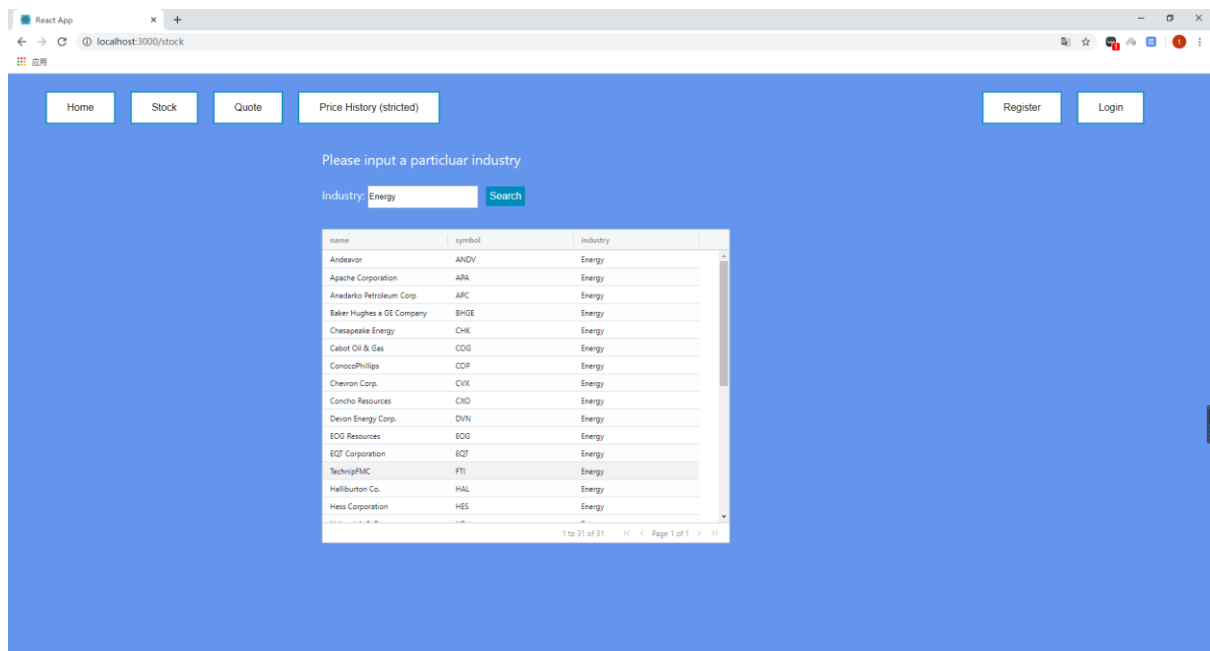
Now, clicking 'stock' button will move on a new page below. User could search industry in the search bar and has display all industries in the table.



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

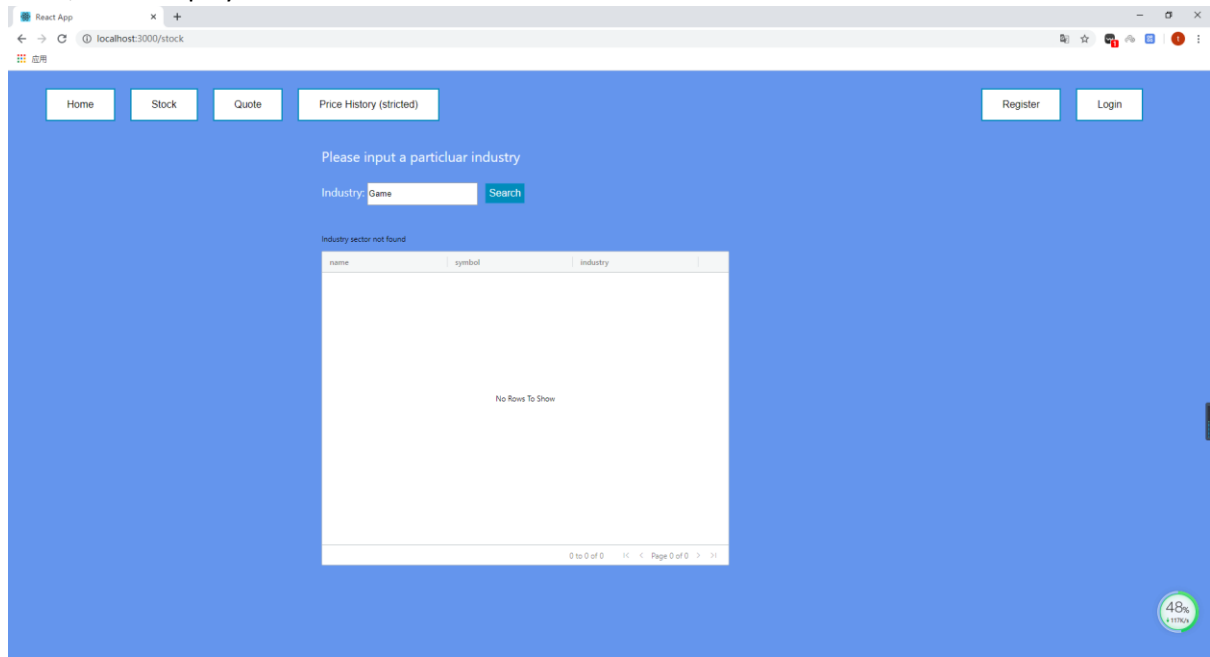


Now, I am going to add one of industry (Energy) in the search bar, the table has change. Industry has become to 'Energy' on the top.

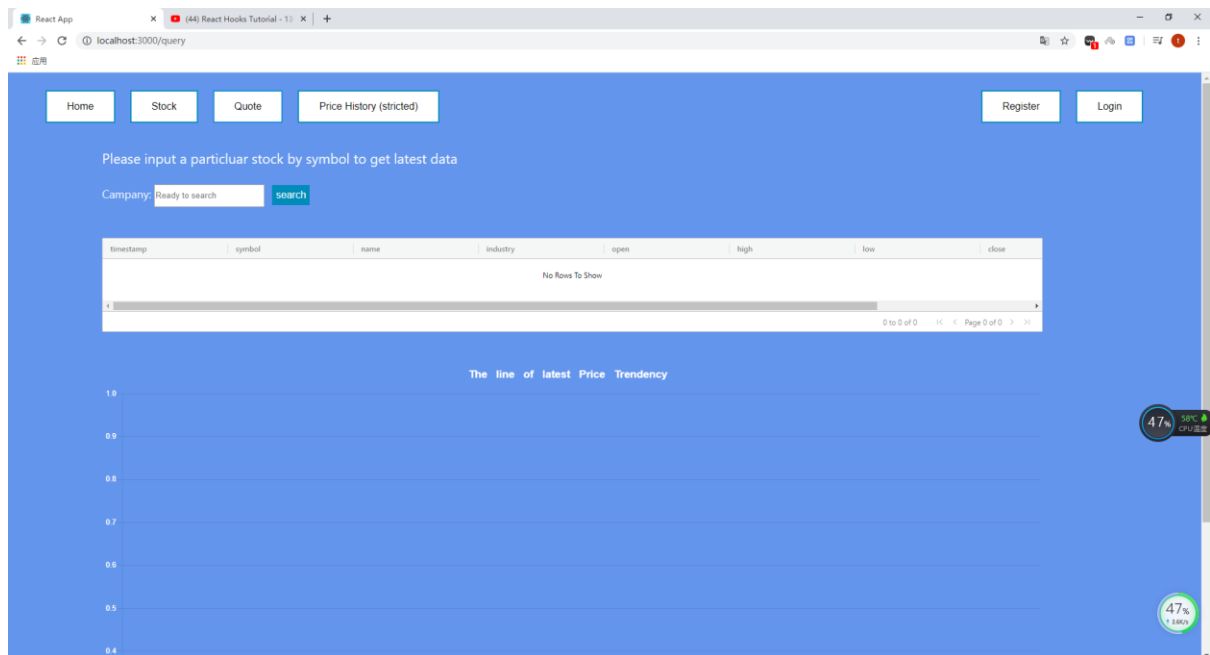


Now I am going to add industry name that doesn't exist. It returns nothing in the table. in addition, there is a sentence on the left upper corner of table. I basically catch error from res. If res return

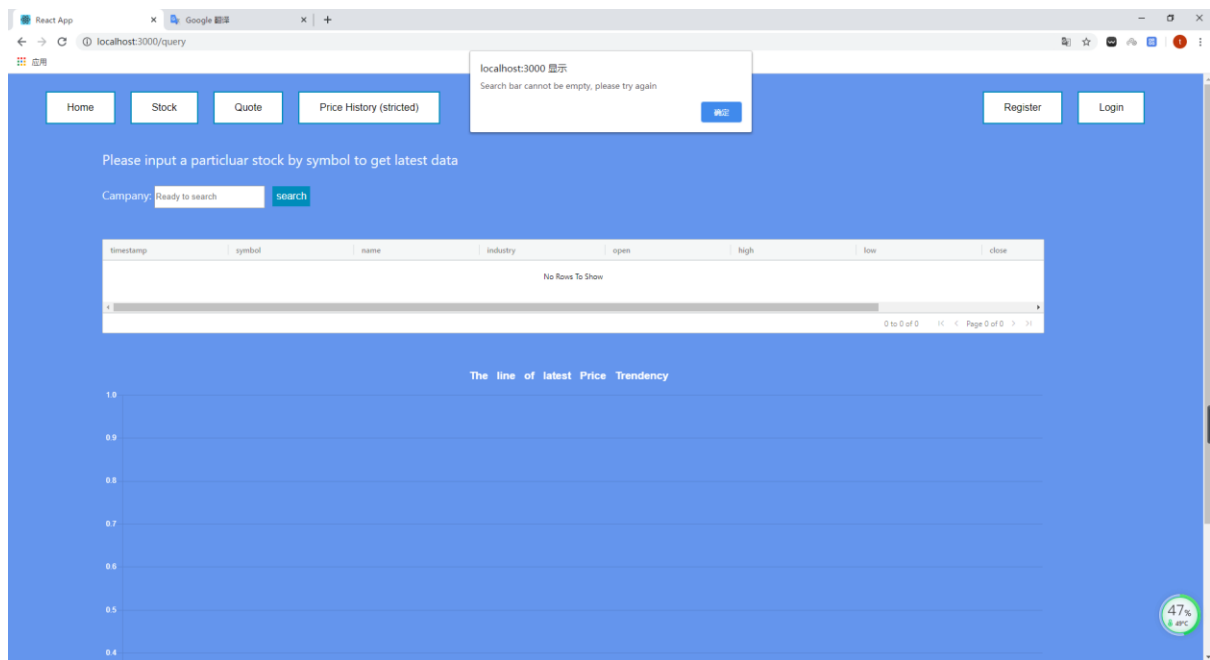
error, it will display on the screen.



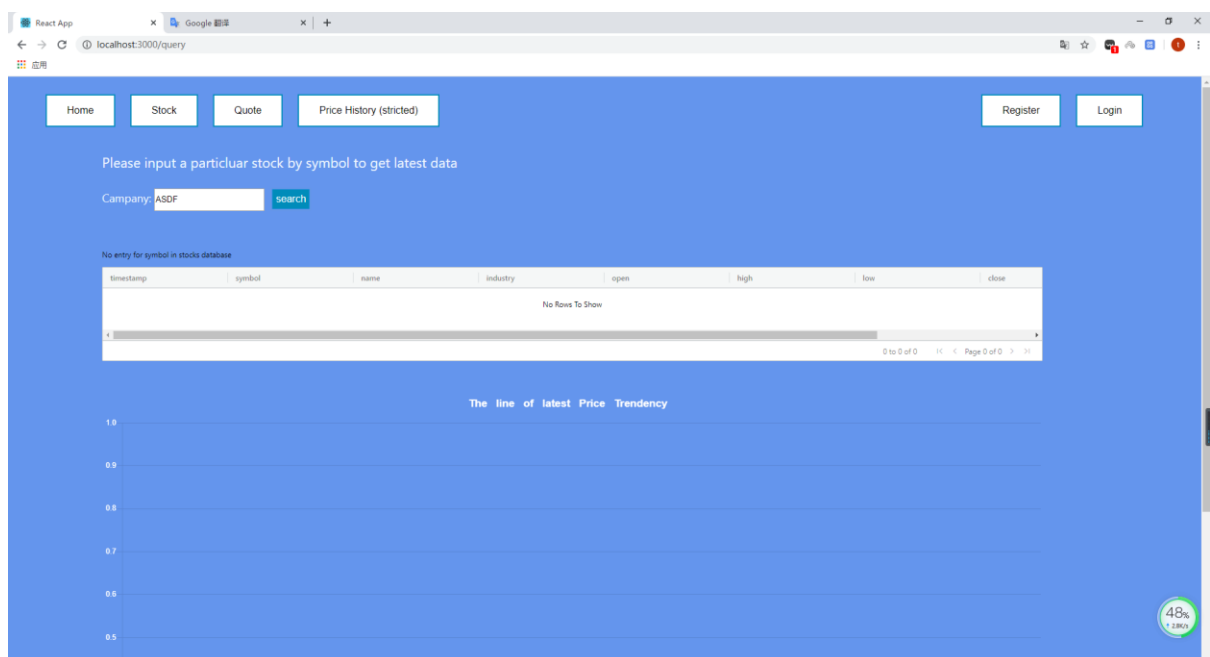
Now, I am going to click 'Query' button. It will move on a new page ('/query'). This is my query layout including search bar, table and line-chart.



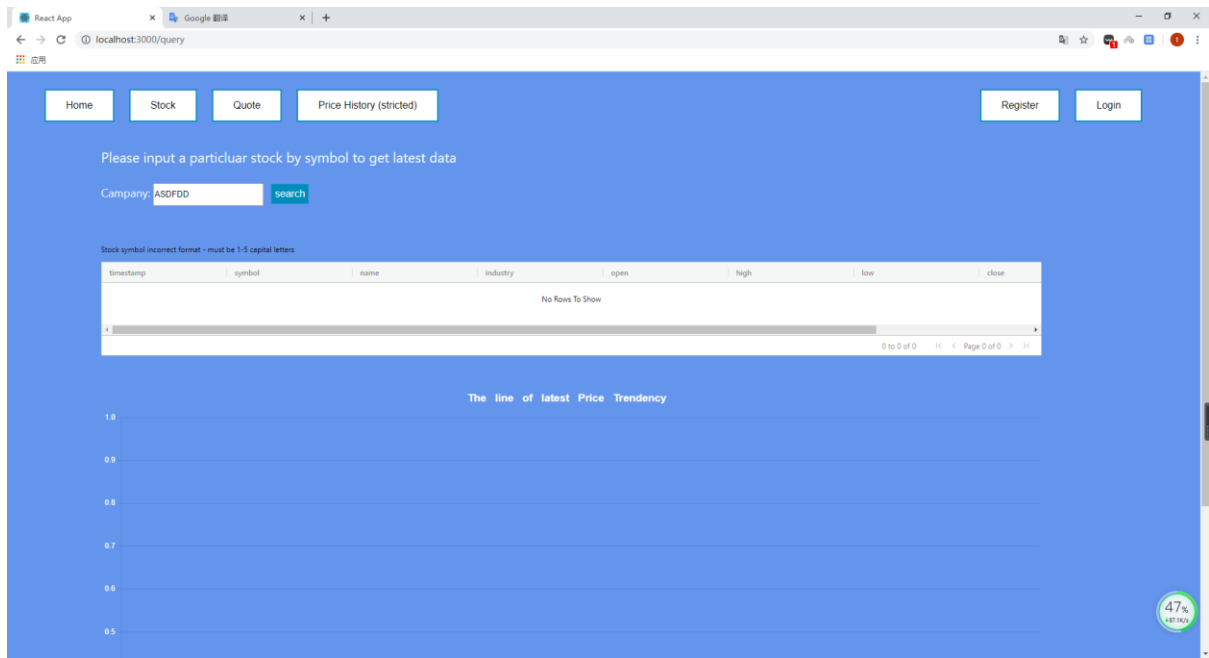
I am going to type without company by symbol. It involves an alert, which means that search bar cannot be null.



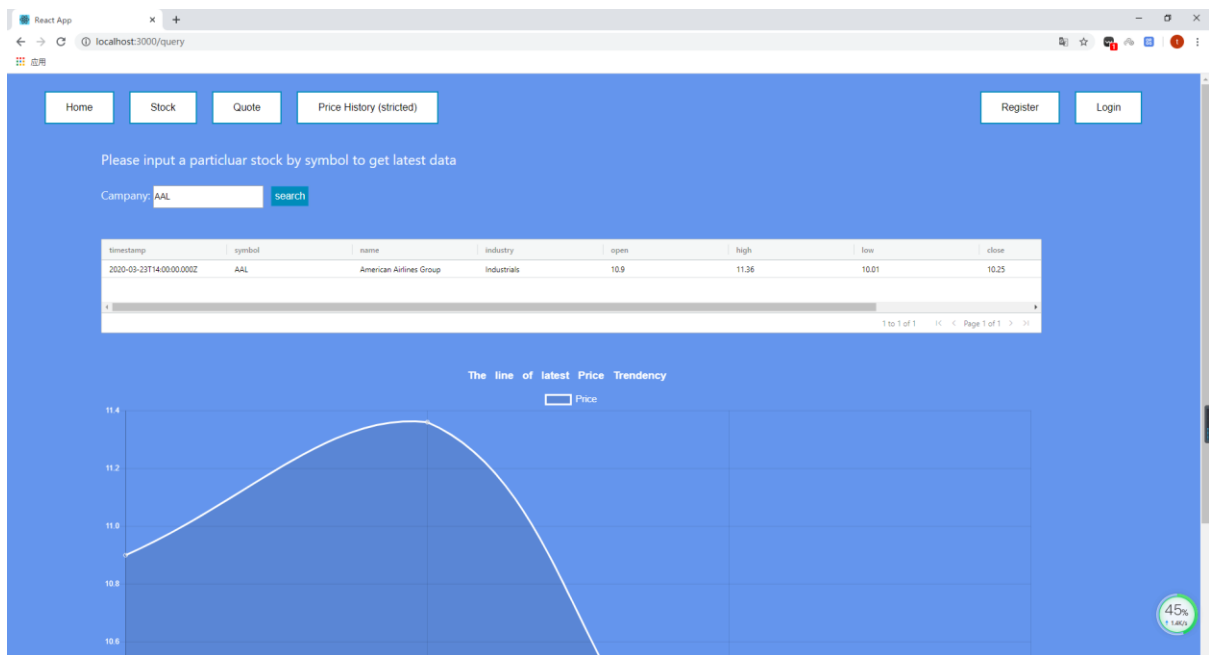
If I type company name doesn't exist, it will catch error from res and display error on the left upper corner of table.

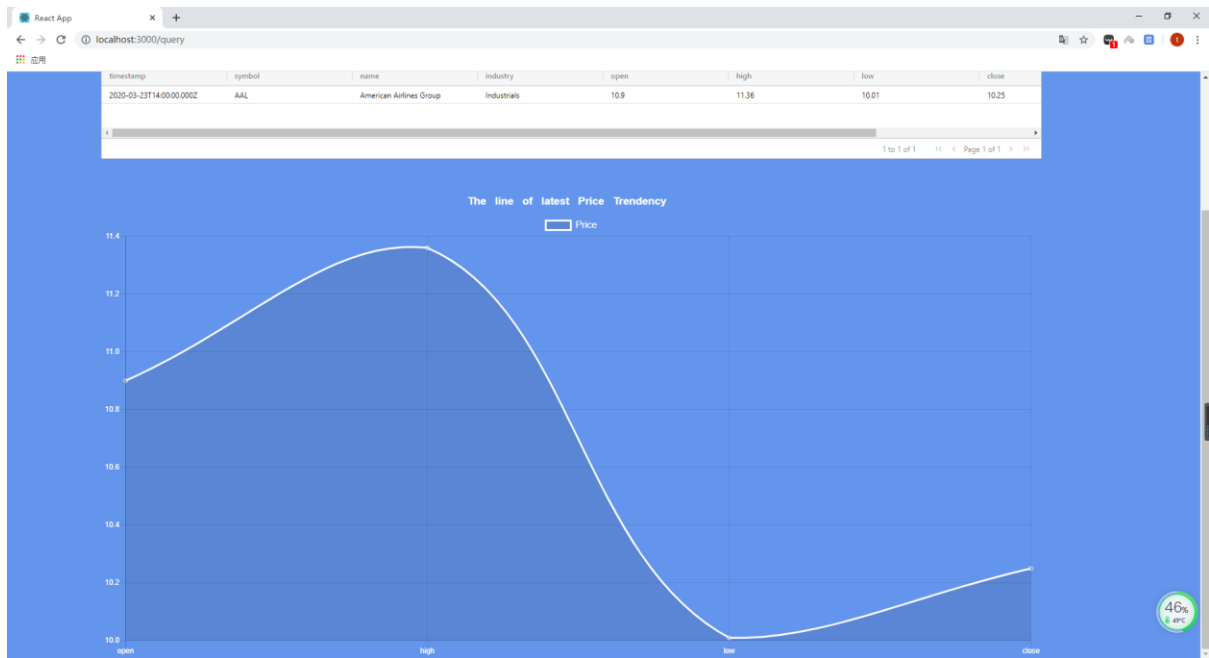


If I type company name more than 5 letters, it also will catch error from res and display error on the left upper corner of table.

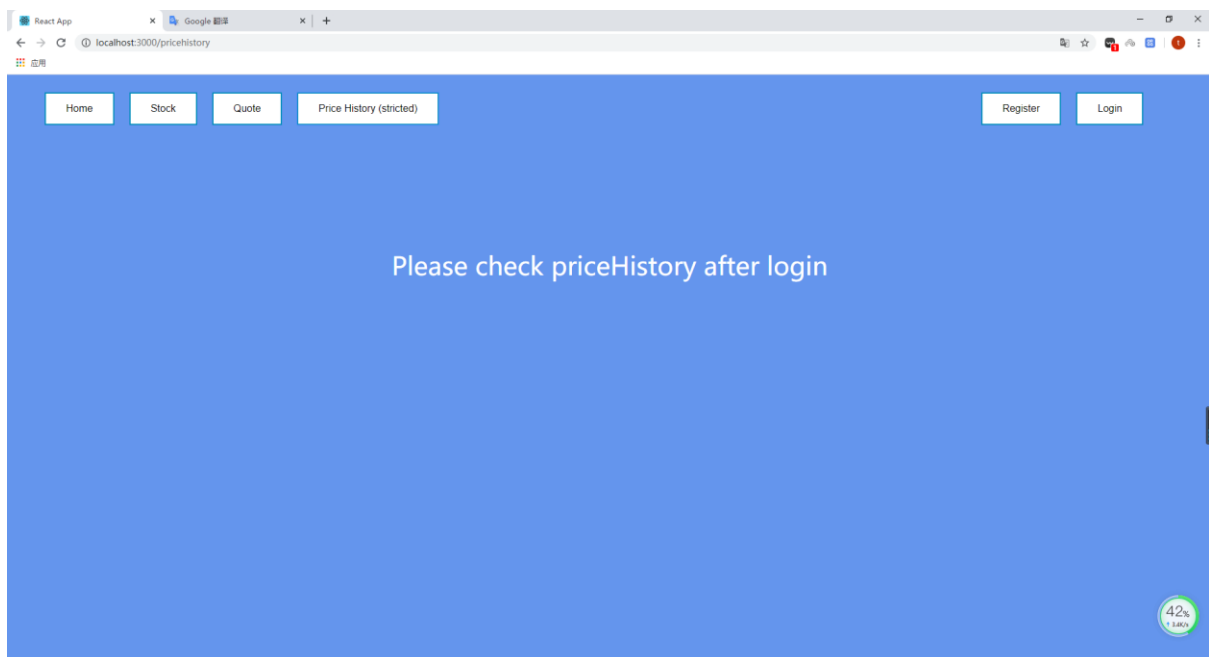


Now, I am going to type one of company by symbol. As you cloud see it return value in the table and chart.

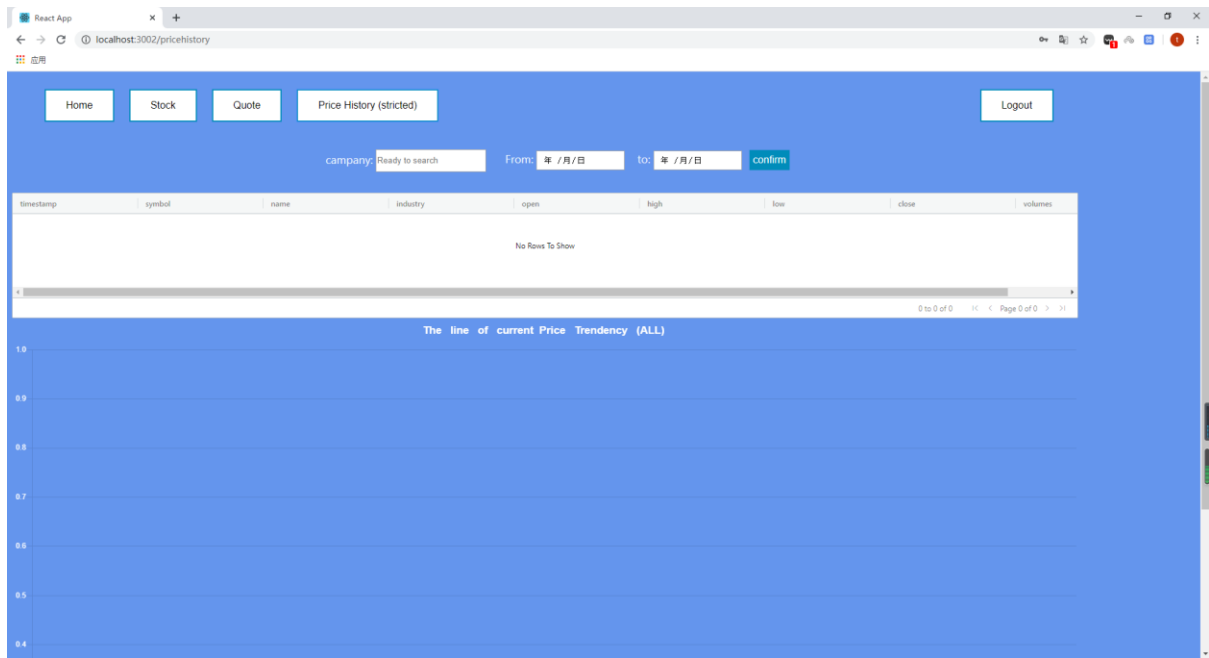




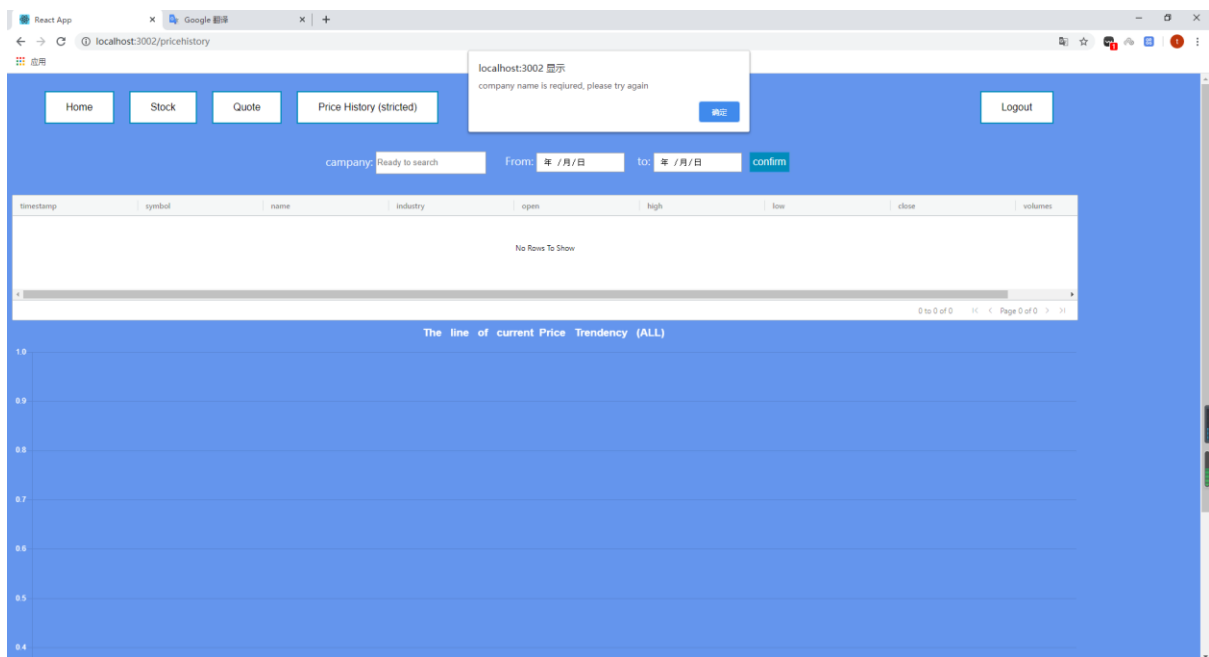
Now, I am going to click 'price history' button without login, there is instruction on the center. It tells user if you want to check history, you have to login.



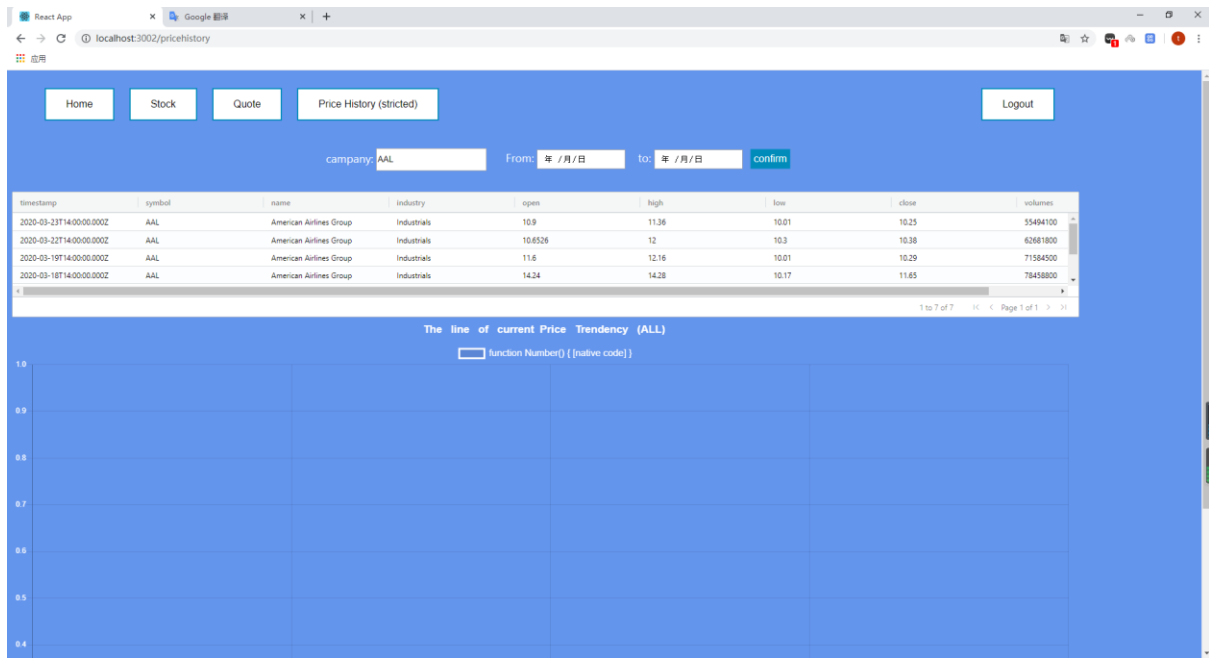
Now, I have logged in, there are search bar, date, table and line-chart in this page.



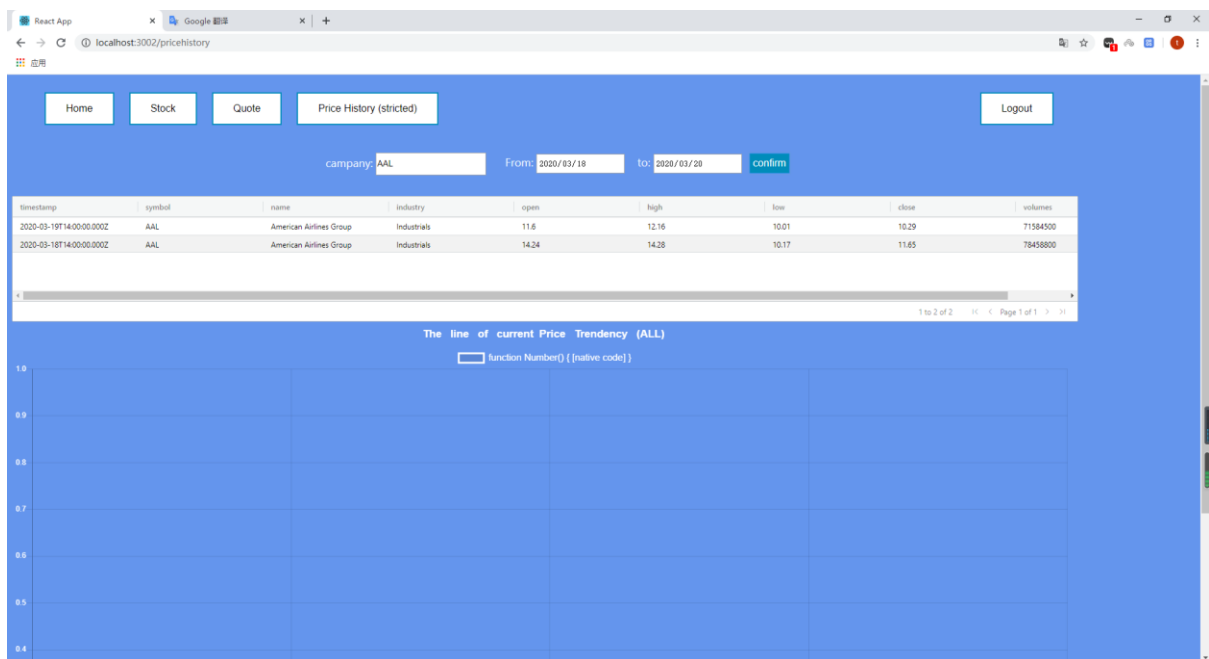
Now, I am going to type without any word. It involves an alert on the top, which means search bar cannot be null, because company name is required, date is not required.



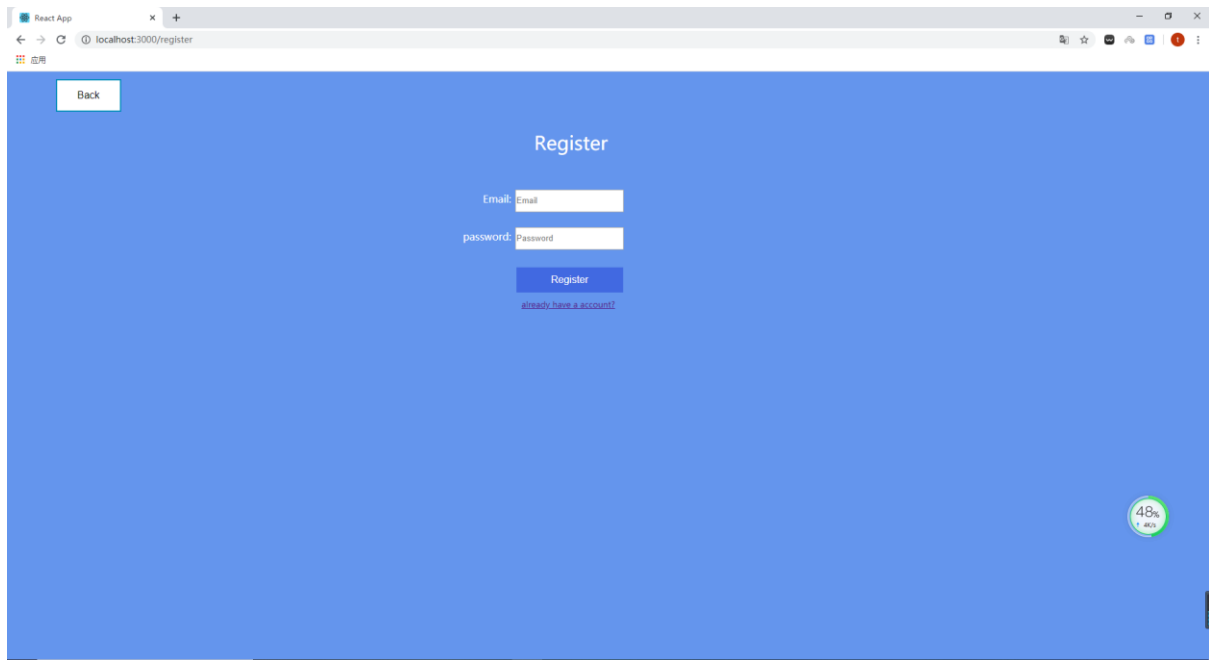
Now, I am going to type one of company name by symbol. It does return values in the table.



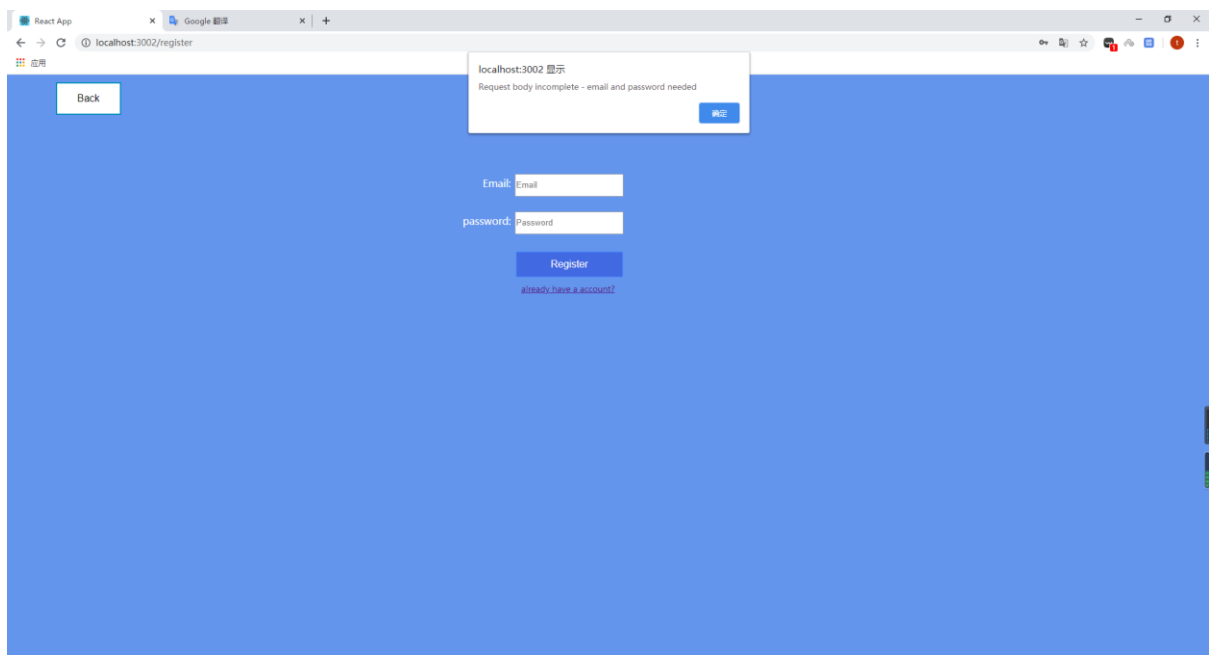
Now I am going to type one of company and the date. you could see It returns the values in the table with correct time.



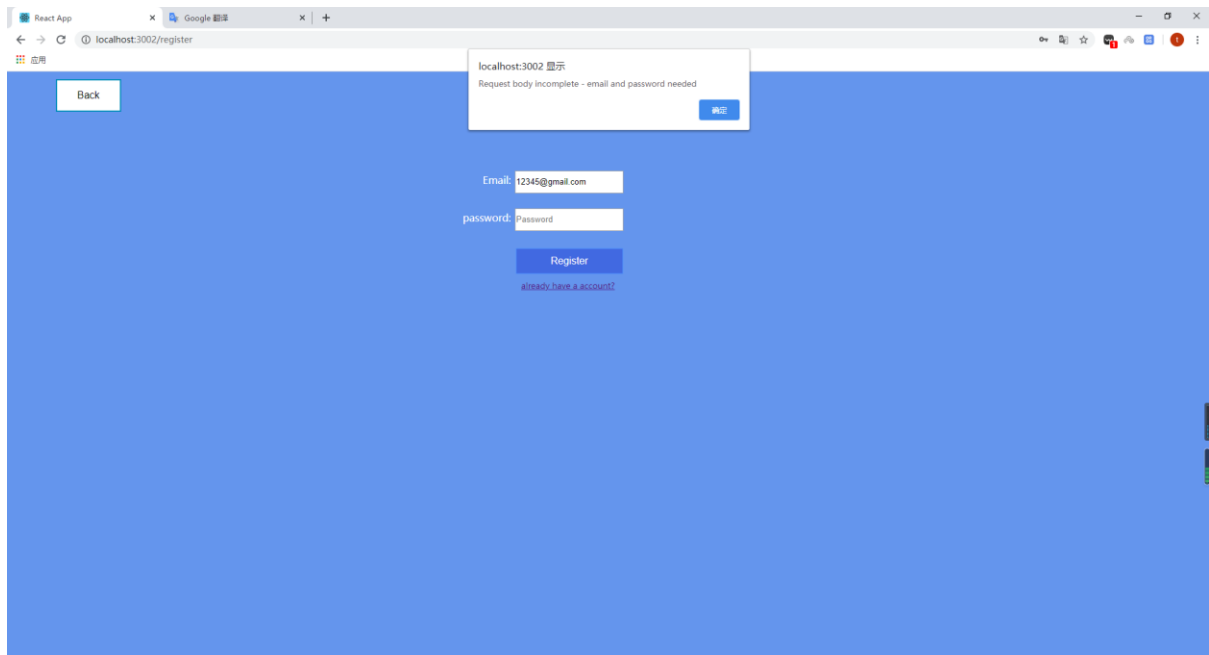
Now, moving on register section. this is my register layout below. 'Back' button is backing to home page. In addition, there is a sentence below 'register' button, which mean if you have an account already, you could click it. it will move on login page.



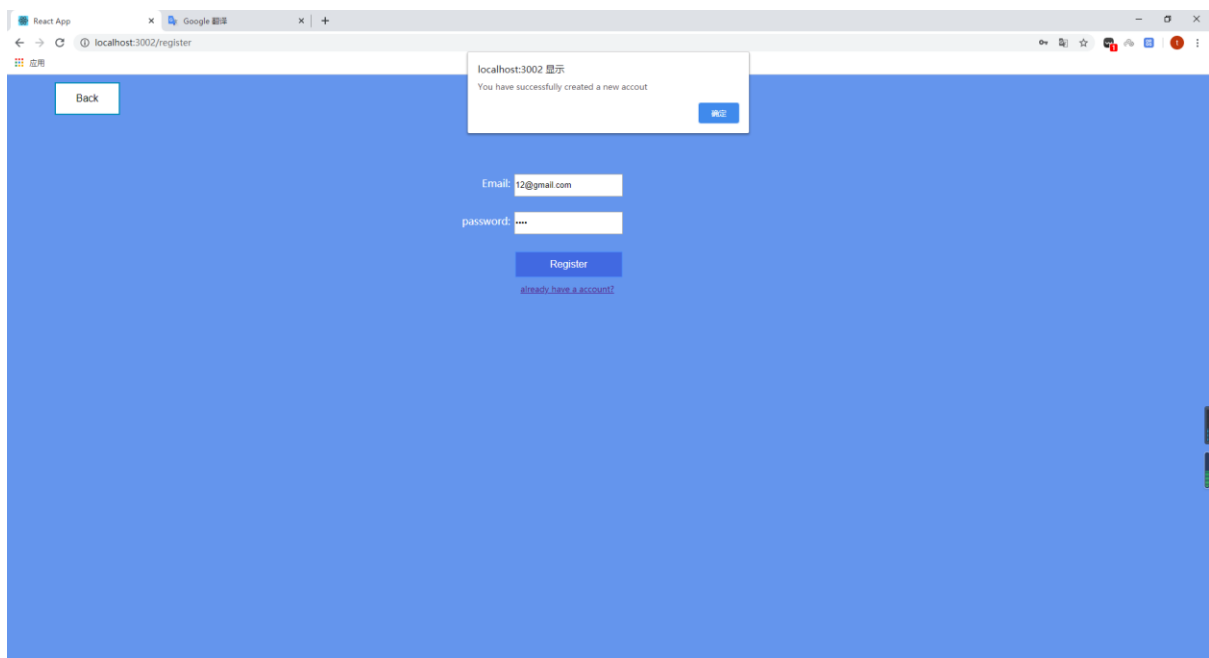
Now, I am going to type without email and password. It involves an alert here, because I catch error from res and display on the screen, which means that email and password cannot be null.



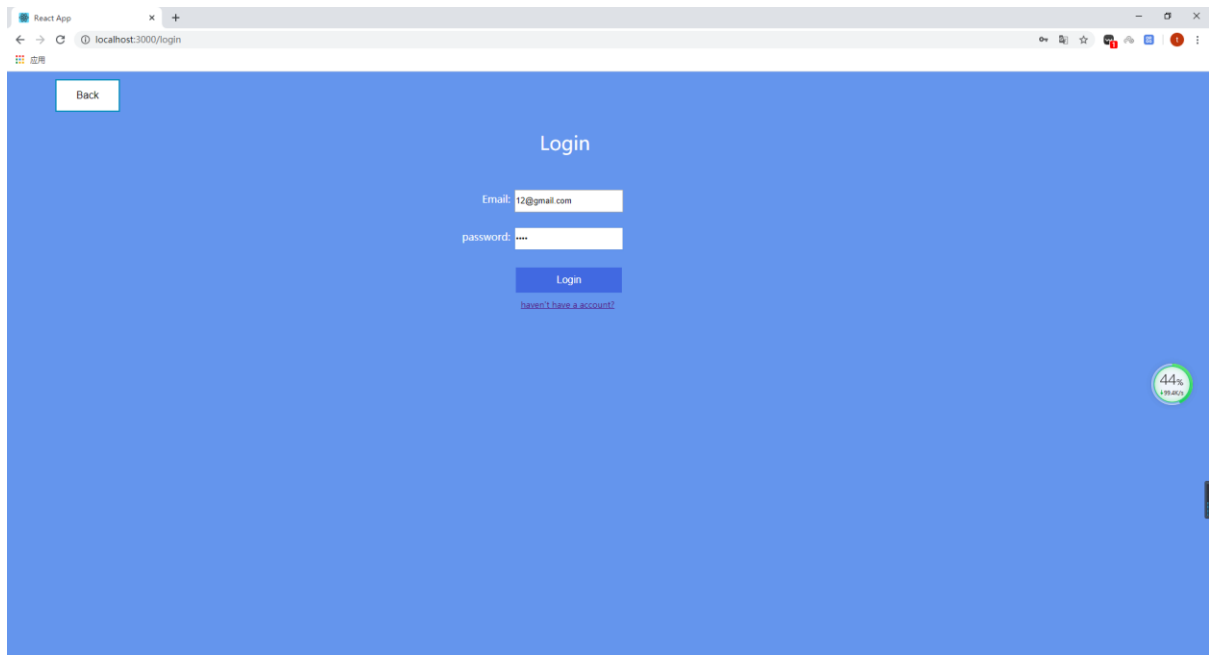
Now, I am going to type email address without password. It also involves an alert on the top, because I catch error from res as alert.



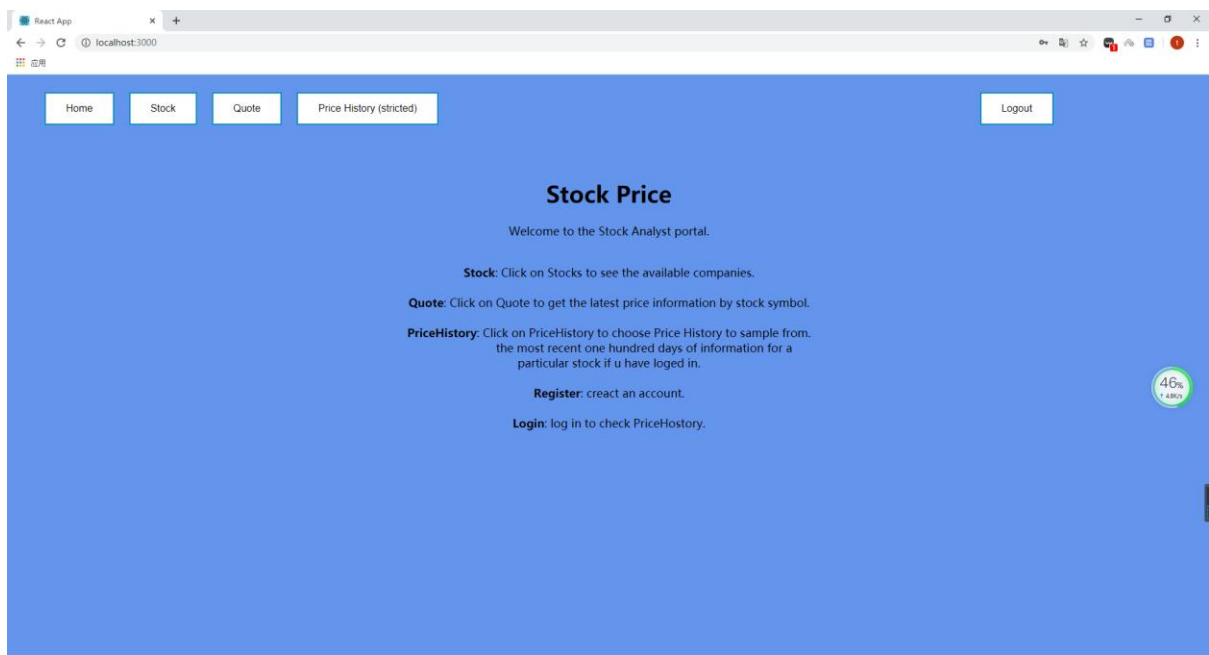
Now, I am going to create a real account. You can see if you successfully create a new account, it also has an alert on the top. This alert is added by me. Not from res.



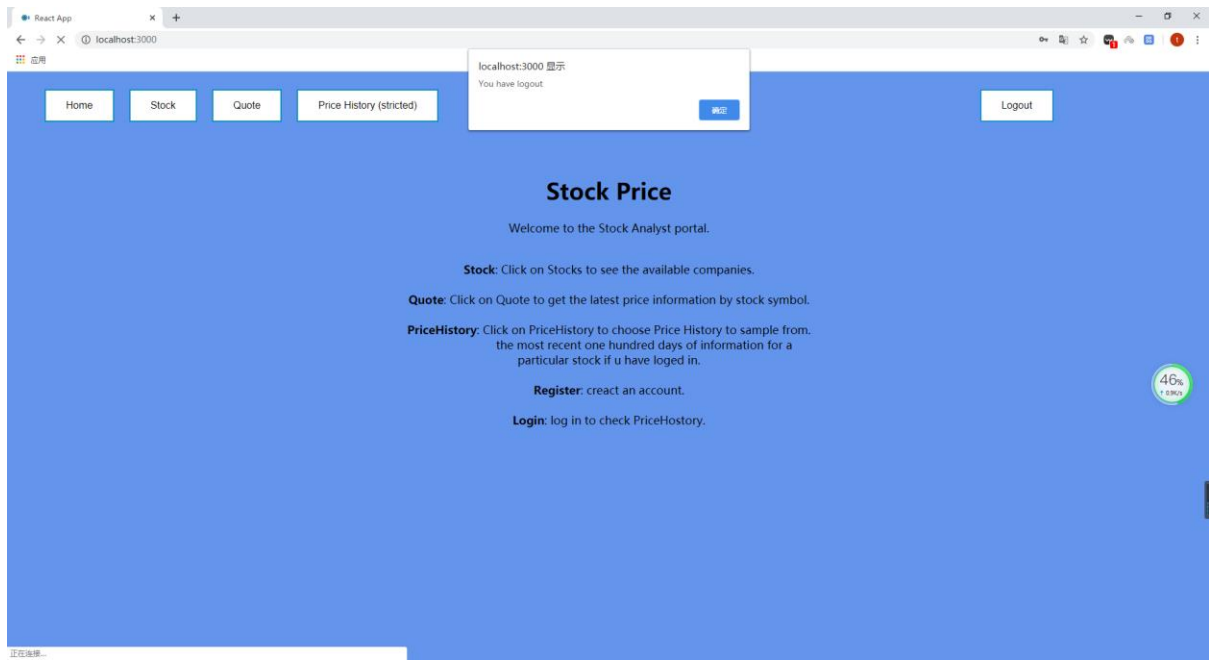
Now, we have already an account. Just log in to see what is happen.



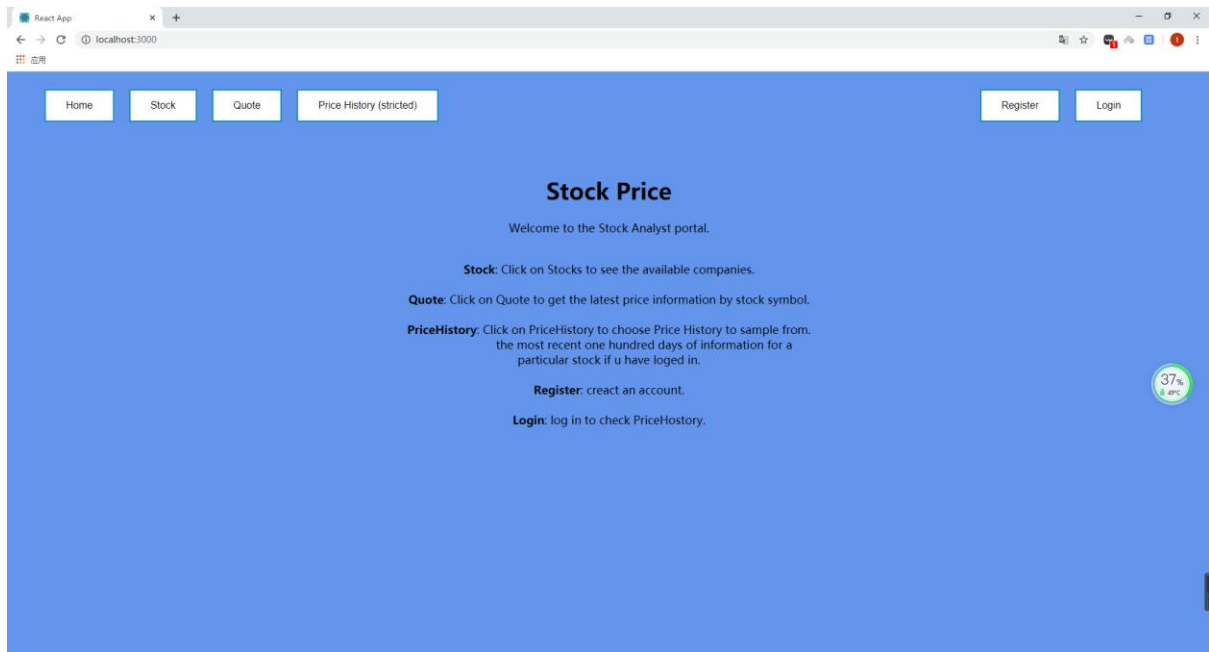
Now, if we log in, it will move on home page. The difference is right upper corner has changed to 'logout' button.



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 Chartjs. 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>