



# HOMework ASSIGNMENT 2

CSCI 571 – Spring 2024

[Abstract](#)

Server-side Scripting using Python, Flask, JSON, AJAX, and the Finnhub Stock API

This content is protected and may not be shared, uploaded, or distributed.

Marco Papa  
papa@usc.edu

# Assignment 2: Server-side Scripting using Python

## Flask, JSON and Finnhub Stock API

### 1. Objectives

- Get experience with Python programming language and Flask framework.
- Get experience creating web pages using HTML, CSS, JavaScript, DOM, JSON format and XMLHttpRequest object.
- Get experience with Finnhub Stock API, Polygon.io API and HighCharts.
- Getting hands-on experience in GCP, AWS or Azure.

### 1.1 Directions

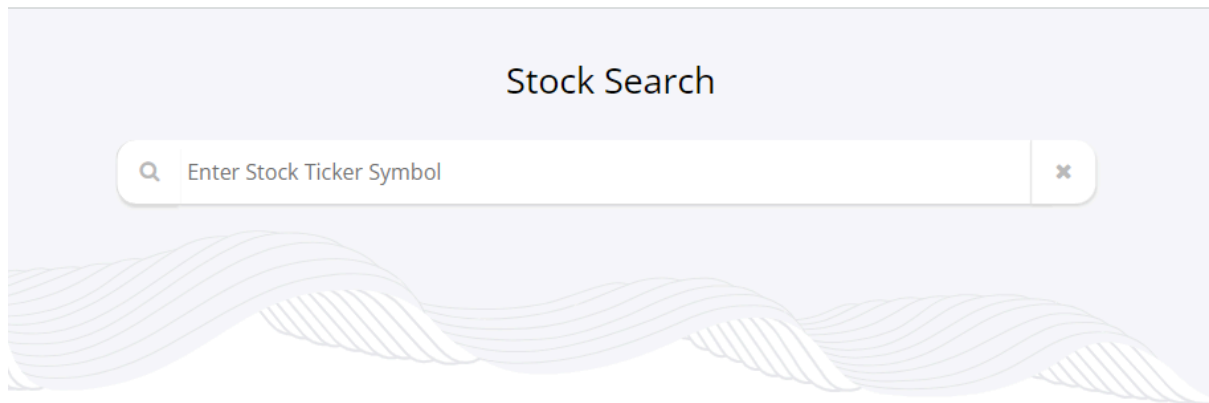
- The backend of this homework must be implemented in the cloud on Google Cloud App Engine, AWS or Azure, using Python.
- See the ungraded assignment **Cloud Setup (Python)** for deployment of your project on GCP, AWS or Azure.
- See the hints in section 3; a lot of reference material is provided to you.
- For Python and Flask kick-start, please refer to the Lecture slides on the class website.
- YouTube Link: <https://www.youtube.com/watch?v=8Dze7tFvcUM>
- You must refer to the **grading guidelines, the video, these specs and Piazza**. Styling will be graded and the point's breakup is mentioned in the grading guidelines.

### 2. Description

In this exercise, you are asked to create a webpage that allows you to search for stock information using the [Finnhub Stock API](#), and the results will be displayed in both tabular format and charts format using [HighCharts](#). You will also provide news articles for the selected stock using the Finnhub API. Chart historical data will be provided by the [Polygon.io](#) API and displayed using [HighCharts](#).

### 2.1. Description of the Search Form

The user first opens a web page as shown below in **Figure 1**, the initial search page, where he/she can enter a stock ticker symbol. Providing a value for the "Stock Ticker Symbol" field is mandatory.



**Figure 1: Initial search page**

The search form has two buttons:

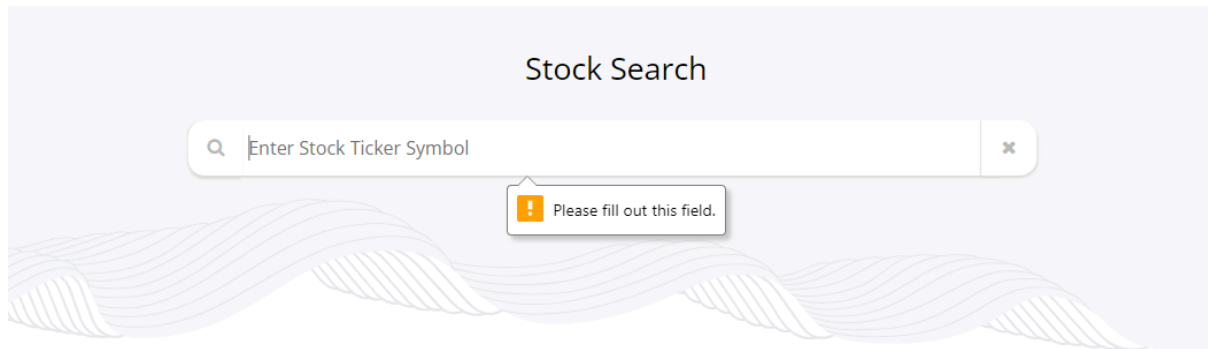
**1. Search button (Magnifying glass symbol):**

Once the user has provided valid data (a stock ticker is required), and clicked on the **Search** button or when the enter/return key is hit, your front-end JavaScript script will make a request to your web server providing it with the form data that was entered (the ticker symbol). You must use GET to transfer the form data to your web server (do not use POST, as you would be unable to provide a sample link to your cloud services). A Python script using Flask will retrieve the data and send it to the FintHub Stock API. You need to use the *Flask* Python framework to make all the API calls.

**Using XMLHttpRequest or any other JavaScript calls for anything other than calling your own “cloud” backend will lead to a 4-point penalty. Do not call the FintHub Stock API directly from JavaScript.**

Define routing endpoints and make your API call from the Python backend. The recommended tutorial for *Flask* and more importantly, routing, can be found at the following link: <https://flask.palletsprojects.com/en/1.1.x/>

If the user clicks on the **Search** button without providing a value in the field, an alert should pop up “Please fill out this field” (an example is shown in **Figure 2**).



**Figure 2: An Example of Empty Input alert**

## **2. Clear button (Cross symbol):**

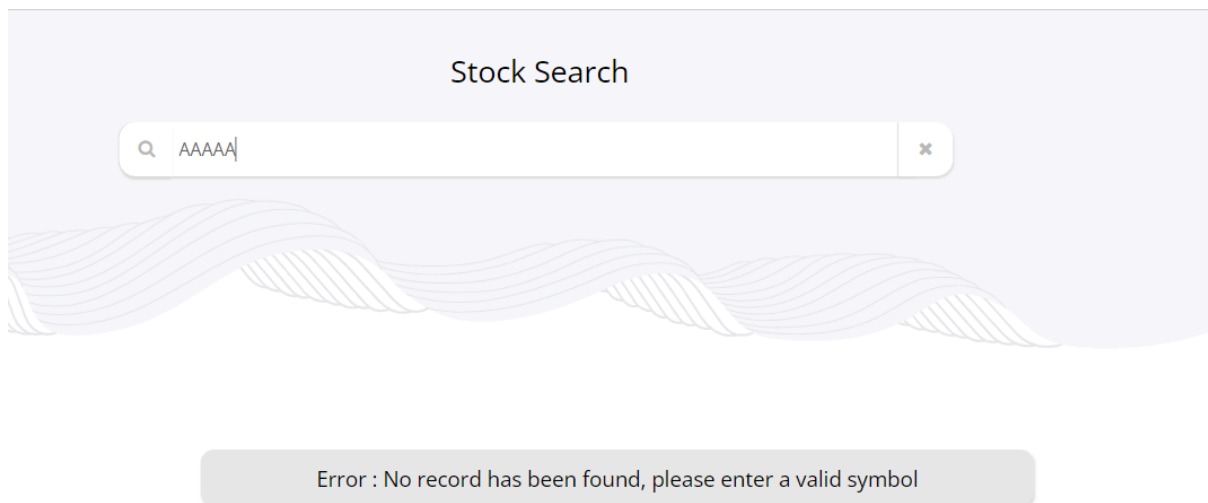
This button must clear the result area (below the search area) and the text field (stock ticker symbol).

## **2.2 Displaying Results**

In this section, we outline how to use the form data to construct the calls to the RESTful web services of the *Finnhub APIs* and display the results in the web page.

If the ticker symbol is invalid (if your API call returns no data) then your webpage should display an error message which says "No record has been found".

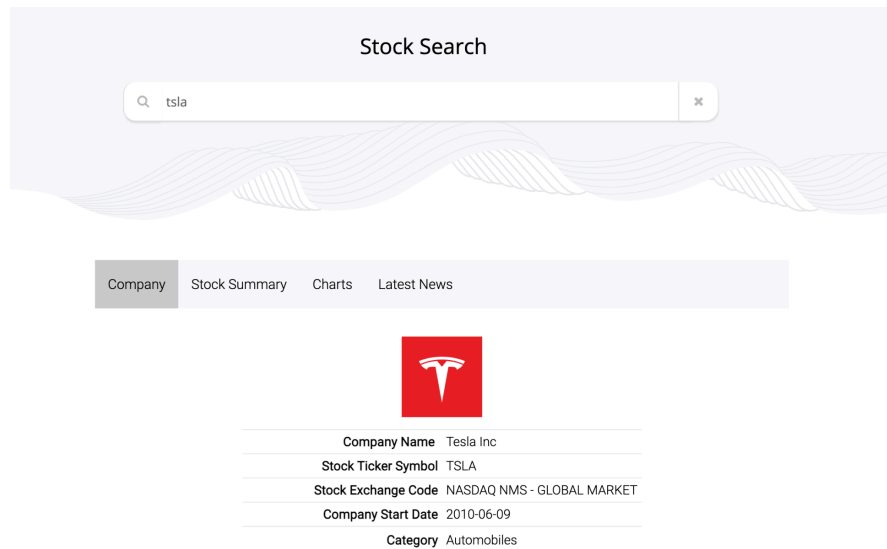
A sample result is shown in **Figure 3**.



**Figure 3: A Sample Search Result for Invalid ticker symbol**

There are **four** components to display in “tabs” on the web page: **Company**, **Stock Summary**, **Charts** and **Latest News**.

A sample result is shown in **Figure 4**.



**Figure 4: A Sample Search Result for Tesla Inc (TSLA)**

### 2.2.1 Displaying Company Tab

We use the *Finnhub “Company Profile 2” Services* to get general information about a company. Please refer to

<https://finnhub.io/docs/api/company-profile2>

for more details on the API.

API Sample: [https://finnhub.io/api/v1/stock/profile2?symbol=TSLA&token=API\\_KEY](https://finnhub.io/api/v1/stock/profile2?symbol=TSLA&token=API_KEY)

The process to create your API key is explained in section 3. When constructing the Python request required to obtain the Company Details, you need to provide two values in the URL:

1. The first value, the **symbol**, is the **stock ticker** the user entered in the form.
2. The second value, the **token** is your **API\_KEY** that you create as described in section 3.

The **response** of this Python request is a **JSON-formatted object**. **Figure 5** shows a sample response from the request for the TSLA symbol. You need to parse this JSON object and extract some fields as required. The mapping of the JSON response to the tabular data to be displayed on the screen is shown in **Table 1**.

```

{
  "country": "US",
  "currency": "USD",
  "exchange": "NASDAQ NMS - GLOBAL MARKET",
  "finnhubIndustry": "Automobiles",
  "ipo": "2010-06-09",
  "logo": "https://finnhub.io/api/logo?symbol=TSLA",
  "marketCapitalization": 1000519,
  "name": "Tesla Inc",
  "phone": "16506815000.0",
  "shareOutstanding": 1004.26,
  "ticker": "TSLA",
  "weburl": "https://www.tesla.com/"
}

```


**Figure 5: Sample JSON response from Finnhub API's Company Profile 2 endpoint**

Tabular Data	Data from Finnhub company profile 2 service
Company Logo (At the top)	The value of key "logo"
Company Name	The value of key "name"
Stock Ticker Symbol	The value of key "ticker"
Stock Exchange Code	The value of key "exchange"
Company Start Date	The value of key "ipo"
Category	The value of key "finnhubIndustry"

**Table 1: Mapping JSON data and company tab display**

After extracting the data, the data should be displayed in a tabular format with the logo at the top of the table. A sample output is shown in **Figure 6**.

Company
Stock Summary
Charts
Latest News



Company Name	Tesla Inc
Stock Ticker Symbol	TSLA
Stock Exchange Code	NASDAQ NMS - GLOBAL MARKET
Company IPO Date	2010-06-09
Category	Automobiles

**Figure 6: Example of Company Tab**

## 2.2.2 Displaying Stock Summary Tab

We use the **Finnhub “Quote” Services** to fetch the stock information for the ticker. Please refer to <https://finnhub.io/docs/api/quote> for more details on the Service.

API Sample: [https://finnhub.io/api/v1/quote?symbol=TESLA&token=API\\_KEY](https://finnhub.io/api/v1/quote?symbol=TESLA&token=API_KEY)

When constructing the Python request required to obtain the Company Details, you need to provide two values in the URL:

1. The first value, the **symbol**, is the **stock ticker** the user entered in the form.
2. The second value, the **token** is your **API\_KEY** that you create as described in section 3.

The **response** of this Python request is a **JSON-formatted object**. **Figure 7** shows a sample response from the request. You need to parse this JSON object and extract some fields as required. The mapping of the JSON response to the tabular data to be displayed on the screen is shown in **Table 2**.

```
{
  "c": 162.41,
  "d": -2.1,
  "dp": -1.2765,
  "h": 166.32,
  "l": 162.3,
  "o": 164.415,
  "pc": 164.51,
  "t": 1642798803
}
```

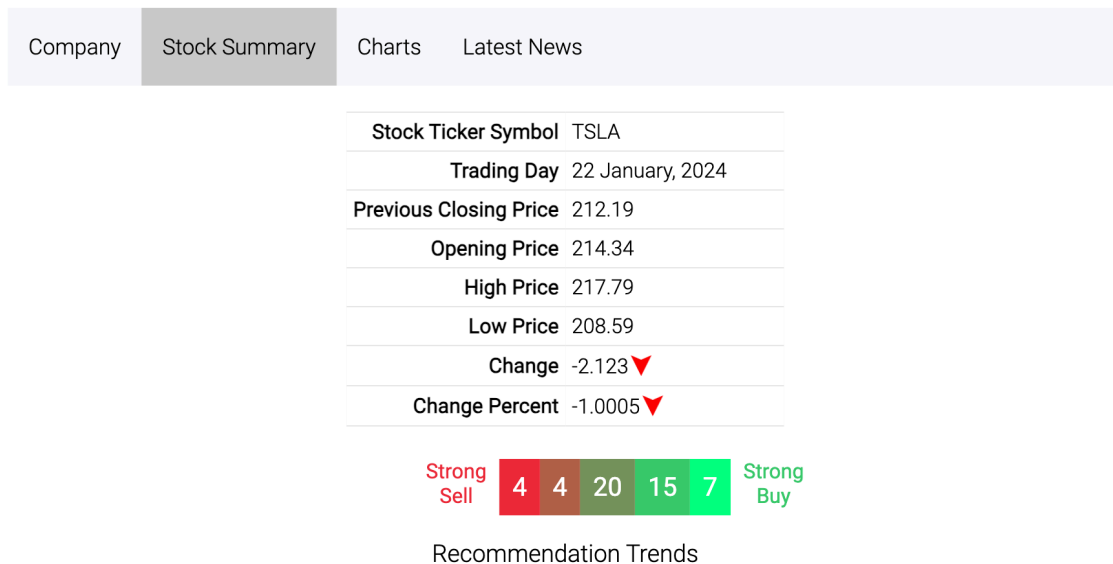
Figure 7: Sample JSON response from Finnhub API’s Quote endpoint

Tabular Data	Data from Finnhub Quote service JSON response
Stock Ticker Symbol	The value of key “ <i>ticker</i> ” from company profile 2 service
Trading Day	The value of key “ <i>t</i> ”. The time stamp is given in the Unix epoch time. Convert it to a human readable date as given in the screenshot below.
Previous Closing Price	The value of key “ <i>pc</i> ”
Opening Price	The value of key “ <i>o</i> ”
High Price	The value of key “ <i>h</i> ”
Low Price	The value of key “ <i>l</i> ”
Change	The value of key “ <i>d</i> ”. Also display the upward arrow or downward arrow depending on whether the difference value is positive or negative.

Change Percent	The value of key “dp”. Also display the upward arrow or downward arrow depending on whether the difference value is positive or negative.
----------------	---

**Table 2: Mapping JSON data and stock summary display**

After extracting the data, the data should be displayed in a tabular format. A sample output is shown in **Figure 8**.



**Figure 8: Example of Stock Summary Tab**

As per the figure 8. The last buy/sell indicator is made using “**Recommendation Trends**” **Services of Finnhub API**. Please refer to <https://finnhub.io/docs/api/recommendation-trends> for more details on the Service.

API Sample: [https://finnhub.io/api/v1/stock/recommendation?symbol=TSLA&token=API\\_KEY](https://finnhub.io/api/v1/stock/recommendation?symbol=TSLA&token=API_KEY)

When constructing the python request required obtaining the Recommendation Trends, you need to provide two values in the URL:

1. The first value, the **symbol**, is the **stock ticker** the user entered in the form.
2. The second value, the **token** is your **API\_KEY** that you create as described in section 3.

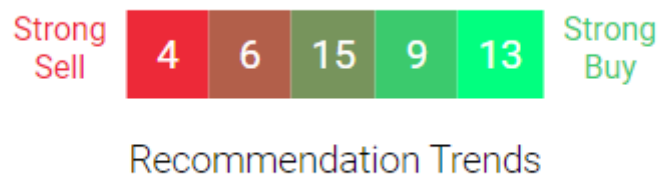
The **response** of this Python request is a **JSON-formatted object**. **Figure 9** shows a sample response from the request. You need to parse this JSON object and extract some fields as required.



```
[{
  "buy": 9,
  "hold": 15,
  "period": "2022-01-01",
  "sell": 6,
  "strongBuy": 13,
  "strongSell": 4,
  "symbol": "TSLA"
}, {
  "buy": 9,
  "hold": 14,
  "period": "2021-12-01",
  "sell": 6,
  "strongBuy": 12,
  "strongSell": 4,
  "symbol": "TSLA"
}, {
  "buy": 10,
  "hold": 13,
  "period": "2021-11-01",
  "sell": 6,
  "strongBuy": 13,
  "strongSell": 4,
```

**Figure 9: Sample JSON response from Finnhub API's Recommendation Trends endpoint**

Use the recommendation values for the latest date for building the indicator as given in **figure 10**. You will need 5 values from the latest response and display them in this respective order: **strongSell**, **sell**, **hold**, **buy**, **strongBuy**.



**Figure 10: Example of Recommendation Trends**

### 2.2.3 Displaying Charts Tab

You should extract the content of Time Series Data from the returned JSON object to construct a chart which is responsible for displaying (close) price and volume. The chart is provided by **HighCharts**. Find more information about HighCharts at:

<https://www.highcharts.com/demo>

<https://www.highcharts.com/demo/stock/area>

<https://www.highcharts.com/demo/stock/candlestick-and-volume>

The historical data for the ticker can be obtained from **Polygon.io “Aggregate (Bars)” Service**.

Please refer to the API documentation at:

[https://polygon.io/docs/stocks/get\\_v2\\_aggs\\_ticker\\_stockticker\\_range\\_multiplier\\_timespan\\_from\\_to](https://polygon.io/docs/stocks/get_v2_aggs_ticker_stockticker_range_multiplier_timespan_from_to)

for more details on the Service.

API Sample:

*GET: /v2/aggs/ticker/{stockTicker}/range/{multiplier}/{timespan}/{from}/{to}*

*[https://api.polygon.io/v2/aggs/ticker/AAPL/range/1/day/2023-01-09/2023-07-09?adjusted=true&sort=asc&apiKey=ctO8iVF\\_Gi19afBovU1ZSr6Ulxqt8Fr3](https://api.polygon.io/v2/aggs/ticker/AAPL/range/1/day/2023-01-09/2023-07-09?adjusted=true&sort=asc&apiKey=ctO8iVF_Gi19afBovU1ZSr6Ulxqt8Fr3)*

When constructing the Python request required to obtain the Company Stock Chart, you need to provide 6 values:

1. The first value, the **stockTicker** the user entered in the form.
2. The second value, the **multiplier**, is the size of the timespan multiplier. Should be a **1**.
3. The third value, the **timespan**, should be **day**.
4. The fourth value, the **from** date, is the start of the aggregate time window. Either a date with the format YYYY-MM-DD or a millisecond timestamp. This should be **6 months and 1 day** prior to the current date. You can use Python DateUtils’s “relativedelta” to calculate the date. Please refer to <https://dateutil.readthedocs.io/en/stable/relativedelta.html> for more details.
5. The fifth value, **to**, is the end of the aggregate time window. Either a date with the format YYYY-MM-DD or a millisecond timestamp. It should be the current date (**Today’s date**).
6. The sixth value, the **query string**, should contain “adjusted=true&sort=asc&apiKey=YOUR\_KEY”, where the **apiKey** is your **Api Key** that you created as described in section 3.2. Do **not** use the **limit** parameter, as it will conflict with the **from/to** dates. Limit will default to 5,000, which is enough for 6 months of data.

**Note:** The Service only accepts dates in UNIX timestamps so you will need to convert dates to UNIX epoch time.

The **response** of this Python request is a **JSON-formatted object**. **Figure 11** shows a sample response from the request. You need to parse this JSON object and extract some fields as required.

```
{
  "c": 75.0875,
  "h": 75.15,
  "l": 73.7975,
  "n": 1,
  "o": 74.06,
  "t": 1577941200000,
  "v": 135647456,
  "vw": 74.6099
},
```

**Figure 11: Sample JSON response from Polygon.io API's Aggregate (Bars) Endpoint**

The data obtained from the API can then be mapped to the **HighCharts** dataset using the mapping below.

Chart Data	Data from polygon.io Aggregate (Bars) service JSON response
Date	The value of key "t". The time stamp is given in the Unix epoch time.
Stock Price	The value of key "c"
Volume	The value of key "v"

**Table 3: Mapping JSON data and HighCharts data**

**Note:** Map each returned array element for each attribute, by its index. For example, for date (timestamp) t[0], close price is c[0] and volume is v[0].

For mapping the Stock price data to data for HighCharts, create an array of data points (x1, y1) where x1 will be the date and y1 will be the corresponding close stock price for that day. This array will then act as an input dataset for your HighCharts. Please refer to links in **Section 3** for more details.

Similarly create another array of points (x2, y2) where x2 will be the date and y1 will be the volume for that day. This array will be the second input for your HighCharts. Since you will be plotting Stock Price vs Date and Volume vs Date you will have two different datasets and two y-axis and a single x-axis.

Initially, the chart shows the historical stock price (in blue line with filling the area below, two digits after decimal) and volume (in grey bar) for the **past six months** by an interval of **one day**. **Figure 12** shows an example of the Stock Price/Volume chart.



**Figure 12: An Example of Chart showing Stock Price/Volume for 6 months**

The title of the chart for showing price/volume is “**Stock Price <Ticker> (YYYY-MM-DD)**”, where “YYYY-MM-DD” is today’s date. The subtitle of the chart should be “Source: Polygon.io” and should hyperlink to the Polygon.io website: <https://polygon.io/>. The title of the Y-axis is “**Stock Price**” when showing the stock price and the other Y-axis is “**Volume**”.

The X-axis changes on the basis of the zoom level 6 months, 3 months, 1 month, 15 days, and 7 days. Please refer to **Section 3** for references on how to change the chart data on the basis of the zoom level. **Figure 13** shows an example of the Stock Price/Volume chart for 15 days zoom level. Ensure that the volume bars are small enough to not overlap the quote line chart.



**Figure 13: An Example of Chart showing Stock Price/Volume for 15 days**

## 2.2.4 Displaying Latest News Tab

We use the **Finnhub “Company News” Services** to fetch the stock information for the ticker. Please refer to <https://finnhub.io/docs/api/company-news> for more details on the Service.

API Sample: [https://finnhub.io/api/v1/company-news?symbol=TESLA&from=BEFORE\\_30&to=TODAY&token=API\\_KEY](https://finnhub.io/api/v1/company-news?symbol=TESLA&from=BEFORE_30&to=TODAY&token=API_KEY)

When constructing the Python request required to obtain the Company News, you need to provide four values in the URL:

1. The first value, the **symbol**, is the **stock ticker** the user entered in the form.
2. The second value is the **from**, a prior date which should be 30 days prior to the current date. You can use Python DateUtils’s “**relativedelta**” to calculate the date. Please refer to <https://dateutil.readthedocs.io/en/stable/relativedelta.html> for more details.
3. The third value, **to**, is the current date (Today’s date).
4. The fourth value, the **token** is your **API\_KEY** that you create as described in section 3.

**Note:** The Service only accepts dates in YYYY-MM-DD format so you will need to convert dates to this format.

The **response** of this Python request is a **JSON-formatted object**. **Figure 14** shows a sample response from the request. You need to parse this JSON object and extract some fields as required. The mapping of the JSON response to the tabular data to be displayed on the screen is shown in **Table 4**. Only the articles that have **image**, **url**, **headline** and **datetime** keys present with a non-empty value should be displayed. If any attribute is missing, empty or blank, do not display it in the results. Display the **first five** articles from the response which have all the keys and values present as mentioned.

```
[{
  "category": "company",
  "datetime": 1642867200,
  "headline": "Bitcoin drops below $35,000 Saturday as global market selloff spreads",
  "id": 95112175,
  "image": "https://s.yimg.com/ny/api/res/1.2/QSz447Gxee8mVgnWMtQX3Q--/YXBwaWQ9aGlna",
  "related": "TSLA",
  "source": "Yahoo",
  "summary": "Cryptocurrency prices tumbled Friday night, with bitcoin hitting its 1",
  "url": "https://finnhub.io/api/news?id=a889402cc5e86397f7ec1173fbef770fb5abb6b4621"
}, {
  "category": "company",
  "datetime": 1642859520,
  "headline": "Why Does Bitcoin's Price Rise and Fall?",
  "id": 95112199,
  "image": "https://s.yimg.com/uu/api/res/1.2/AnkmgjYs51ST56wIv_P5ig--~B/aD0yNzU7dz0",
  "related": "TSLA",
  "source": "Yahoo",
  "summary": "Bitcoin's price has gone from $32,983 on Jan. 22, 2021 to $35,811 on t",
  "url": "https://finnhub.io/api/news?id=2809b72a2a1ea11972dbe1424dd85a0340beaa73f2e"
}, {
  "category": "company",
  "datetime": 1642852800,
```

**Figure 14: Sample JSON response from Finnhub API's Company News Endpoint**

The mapping between the information populated in the results and the JSON object is shown in **Table 4**.

Card Data	Data from Finnhub company-news service JSON response
Image	The value of key <i>"image"</i>
Title	The value of key <i>"headline"</i>
Date	The value of key <i>"datetime"</i> . The time stamp is given in the Unix epoch time. Convert it to human readable date as given in the screenshot below.
Link to Original Post	The value of key <i>"url"</i>

**Table 4: Mapping between result card and JSON object**

After extracting the data, it should be displayed as shown in **Figure 15**. The **"See Original Post"** hyperlink opens the original post in a new tab in the browser.

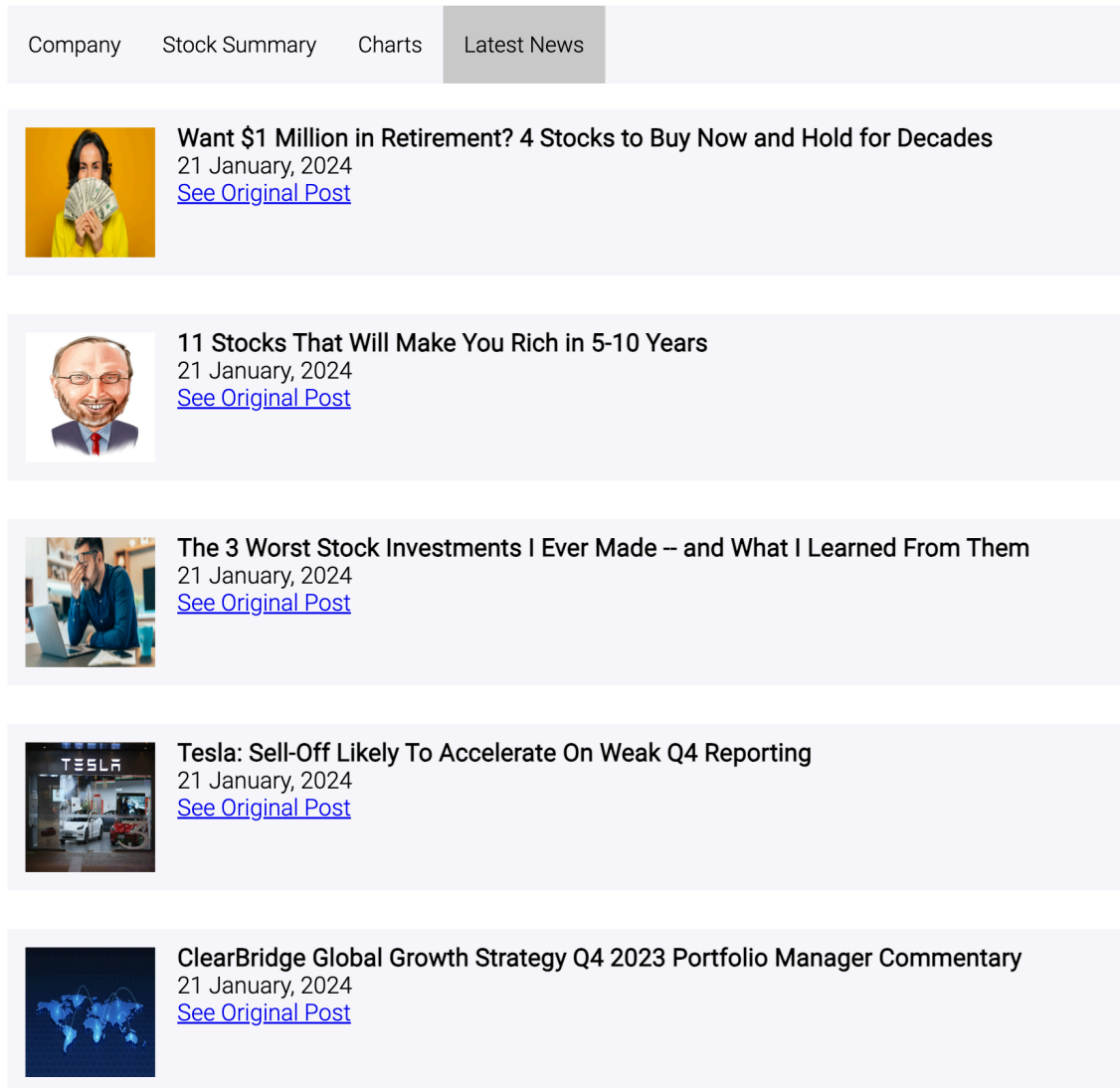


Figure 15: Example of Latest News

## 2.3 Saving Previous Inputs

In addition to displaying the results, the page should maintain the entered value while displaying the current result. For example, if a user searches for “AAPL”, the user should see what was provided in the search form when displaying the results. Specifically, when clicking on the **Search** button, the page should display the result retrieved from the *Finnhub API* service and keep the value provided in the search form.

### 3. Hints

#### 3.1 Finnhub API Documentation

To apply for an API key on Finnhub and read the API documentation, please go to:

<https://finnhub.io/>

Click on the “**Get free API key**” button. Enter your **Name**, your @usc.edu **E-mail** address and **Password**, and click on **Sign-up** and go to get your **API\_KEY**.

<https://finnhub.io/dashboard>

#### 3.2 Polygon.io API Documentation

To apply for an API key on Polygon.io and read the API documentation, please go to:

<https://polygon.io/>

Click on the “**Sign Up**” button. Enter your @usc.edu **Email** address and a **password** and click **Sign Up**, or use a **GitHub** account. In the polygon.io Dashboard, scroll down, and you will see your free **API Key**.

#### 3.3 Deploy Python file to the cloud (GCP/AWS/Azure)

You should use the domain name of the GCP/AWS/Azure service you created in HW #5 to make the request. For example, if your GCP/AWS/Azure server domain is called **example.appspot.com** or **example.elasticbeanstalk.com** or **example.azurewebsites.net** the following links will be generated:

GAE - <http://example.appspot.com/index.html>

AWS - <http://example.elasticbeanstalk.com/index.html>

Azure - <http://example.azurewebsites.net/index.html>

The *example* subdomain in the above URLs will be replaced by your choice of subdomain from the cloud service. You may also use a different page than index.html.

#### 3.4 References for HighCharts

1. <https://api.highcharts.com/highstock/>
2. <https://www.highcharts.com/demo/stock/intraday-area>
3. <https://www.highcharts.com/demo/stock/compare>

### 4. Files to Submit

In your course homework page, you should update the **Assignment 2 link** to refer to your new initial web search page for this exercise (for example, **index.html**). Your files must be hosted on GCP, AWS or Azure cloud services. Graders will verify that this link is indeed pointing to one of the cloud services.



Also, submit your source code file to D2L Brightspace. Submit a ZIP file of both front-end and back-end code, plus any additional files needed to build your app (e.g., yaml file). **The timestamp of the ZIP file will be used to verify if you used any “grace days.”**

**\*\*IMPORTANT\*\*:**

- All discussions and explanations in Piazza related to this homework are part of the homework description and grading guidelines. So please review all Piazza threads, before finishing the assignment. If there is a conflict between Piazza and this description and/or the grading guidelines, **Piazza always rules.**
- You can use jQuery for Assignment 2 but it is not required.
- You should **not** use **Bootstrap** for Assignment 2.
- You **should not call any of the APIs directly from JavaScript**, bypassing the Python proxy. Implementing any one of them in JavaScript instead of Python will result in a **4-point penalty**. This includes calls to both Finnhub and Polygon.io APIs.
- **APPEARANCE OF ALL VIEWS, TABLES AND CHARTS should be similar to the reference video as much as possible.**