

Homework 6: Server-side Scripting using Python Flask, JSON and Stocks 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 Tiingo Stocks API and newsapi.org's News API
- Getting hands-on experience in GCP, AWS or Azure.

1.1 Cloud Exercise

- The backend of this homework must be implemented in the cloud on GCP, AWS or Azure using Python.
- See homework 5 for installation of either one of these platforms. You only have to select one platform to implement your backend.
- 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.
- You must refer to the grading guidelines, the video, the specs and Piazza. Styling will be graded and the point's breakup is mentioned in the grading guidelines.
- The link for the video is - <https://youtu.be/hqK1ZE5bwFM>

2. Description

In this exercise, you are asked to create a webpage that allows you to search for stock information using the [Tiingo 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 News API.

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.

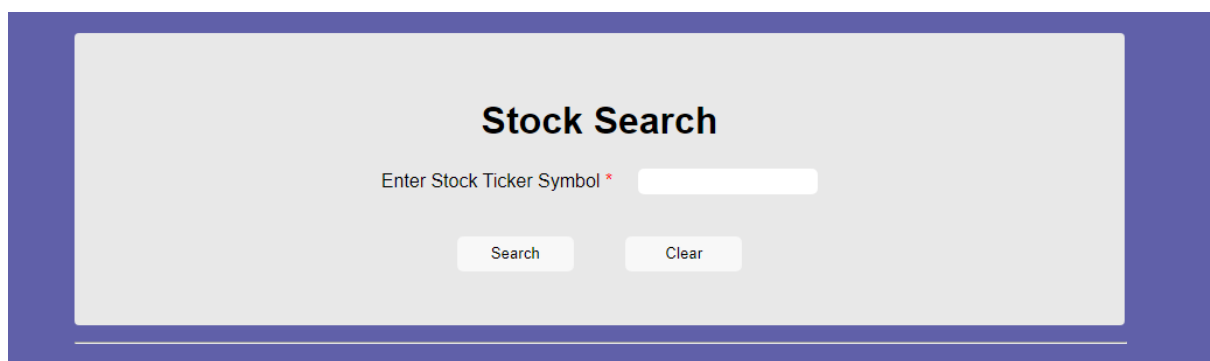
The image shows a web form titled "Stock Search" centered on a light gray background. Below the title is a text input field with the placeholder text "Enter Stock Ticker Symbol *". Underneath the input field are two buttons: "Search" on the left and "Clear" on the right. The entire form is enclosed in a thin gray border.

Figure 1: Initial search page

The search form has two buttons:

1. Search button:

Once the user has provided valid data (a stock ticker is required), and clicked on the **Search** button, 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 Tiingo Stocks 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 Tiingo Stocks 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**).

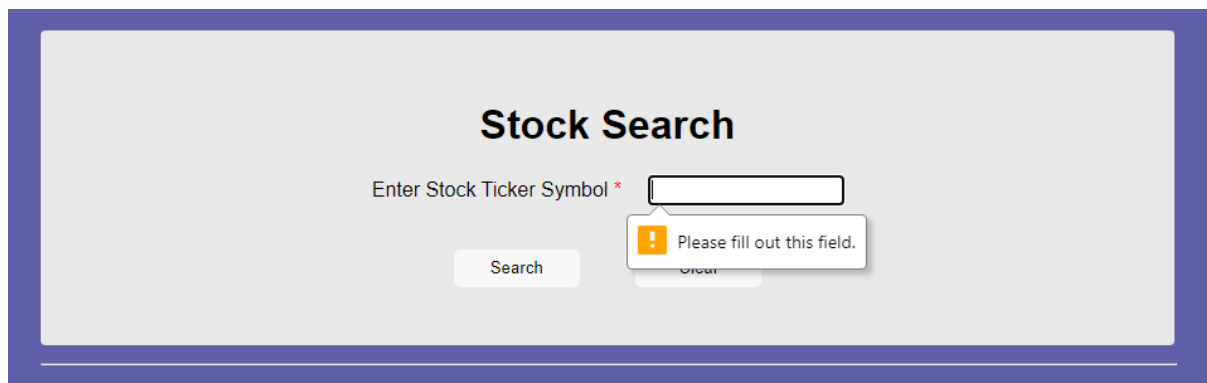


Figure 2: An Example of Empty Input alert

2. Clear button:

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 *Tiingo APIs*, *News API* and display the results in the web page. There are basically **four** components to display in a “tabs” from on the web page: **Company Outlook**, **Stock Summary**, **Charts** and **Latest News**.

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

Figure 3: A Sample Search Result for Apple Inc (AAPL)

2.2.1 Displaying Company Outlook Tab

We use the *Tiingo “Meta Endpoint” API*. Please refer to

<https://api.tiingo.com/documentation/end-of-day>

for more details on the API.

API Sample: https://api.tiingo.com/tiingo/daily/keyword?token=API_KEY

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

1. The first value, the **keyword**, 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.

For example, to get information about AAPL, the stock ticker or Apple, you would call the API as:

<https://api.tiingo.com/tiingo/daily/aapl?token=0108ecd7f84bf97f8f11998a4e4561bx51da92c8>

The **response** of this Python request is a **JSON-formatted object**. **Figure 4** 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 1**.

```
{
  "description": "Apple Inc. (Apple) designs, manufactures and markets mobile communication and media dev",
  "endDate": "2020-09-04",
  "exchangeCode": "NASDAQ",
  "startDate": "1980-12-12",
  "name": "Apple Inc",
  "ticker": "AAPL"
}
```

Figure 4: Sample JSON response from Tiingo API’s metadata endpoint

Tabular Data	Data from Tiingo Metadata API response JSON
Company Name	The value of key “name”
Stock Ticker Symbol	The value of key “ticker”

Stock Exchange Code	The value of key “ <i>exchangeCode</i> ”
Company Start Date	The value of key “ <i>startDate</i> ”
Description	The value of key “ <i>description</i> ”. The description needs to be truncated with an ellipsis (“triple dots”) so as to only span the first FIVE lines.

Table 1: Mapping JSON data and company outlook display

After extracting the data, the data should be displayed in a tabular format. A sample output is shown in **Figure 5**.

Company Outlook	Stock Summary	Charts	Latest News
Company Name	Apple Inc		
Stock Ticker Symbol	AAPL		
Stock Exchange Code	NASDAQ		
Company Start Date	1980-12-12		
Description	Apple Inc. (Apple) designs, manufactures and markets mobile communication and media devices, personal computers, and portable digital music players, and a variety of related software, services, peripherals, networking solutions, and third-party digital content and applications. The Company's products and services include iPhone, iPad, Mac, iPod, Apple TV, a portfolio of consume...		

Figure 5: Example of Company Outlook Tab

2.2.2 Displaying Stock Summary Tab

We use the **Tiingo “Current Top-of-Book & Last Price” API** to fetch the stock information for the ticker. Please refer to <https://api.tiingo.com/documentation/iex> for more details on the API.

API Sample: https://api.tiingo.com/iex/keyword?token=API_KEY

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

1. The first value, the **keyword**, 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.

For example, to get information about AAPL, the stock ticker or Apple, you would call the API as:

<https://api.tiingo.com/iex/aapl?token=0108ecd7f84bf97f8f11998a4e4545ea51da92c8>

The **response** of this Python request is a **JSON-formatted object**. **Figure 6** 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**.

```
[
  {
    "prevClose": 120.88,
    "mid": null,
    "lastSaleTimestamp": "2020-09-04T20:00:00+00:00",
    "open": 120.07,
    "askPrice": null,
    "low": 110.89,
    "ticker": "AAPL",
    "timestamp": "2020-09-04T20:00:00+00:00",
    "lastSize": null,
    "tngoLast": 120.96,
    "last": 120.96,
    "high": 123.7,
    "askSize": null,
    "quoteTimestamp": "2020-09-04T20:00:00+00:00",
    "bidPrice": null,
    "bidSize": null,
    "volume": 332607163
  }
]
```

Figure 6: Sample JSON response from Tiingo API's Current Top-of-Book & Last Price endpoint

Tabular Data	Data from Tiingo Current Top-of-Book & Last Price API response JSON
Stock Ticker Symbol	The value of key <i>"ticker"</i>
Trading Day	The value of the key <i>"timestamp"</i> . <i>Truncate the value to just display the date.</i>
Previous Closing Price	The value of key <i>"prevClose"</i>
Opening Price	The value of key <i>"open"</i>
High Price	The value of key <i>"high"</i>
Low Price	The value of key <i>"low"</i>
Change	Difference between the value of key <i>"last"</i> and <i>"open"</i> . Also display the upward arrow or downward arrow depending on whether the difference value is positive or negative. Reference to the image links in Section 3.
Change Percent	Percentage difference between the value of key <i>"last"</i> and <i>"open"</i> . Also display the upward arrow or downward arrow depending on whether the difference value is positive or negative. Reference to the image in Section 3.
Number of Shares Traded	The value of key <i>"volume"</i>

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 7**.

Company Outlook		Stock Summary	Charts	Latest News
Stock Ticker Symbol		AAPL		
Trading Day		2020-09-04		
Previous Closing Price		120.88		
Opening Price		120.07		
High Price		123.7		
Low Price		110.89		
Change		0.89 ▲		
Change Percent		0.01% ▲		
Number of Shares Traded		332607163		

Figure 7: Example of Stock Summary Information Tab

2.2.3 Displaying Stock Price/Volume Chart 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/volume. The chart is provided by **HighCharts**. Find more information about HighCharts at:

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

The historical data for the ticker can be obtained from **Tiingo's "Historical Intraday Prices Endpoint" API**.

Please refer to <https://api.tiingo.com/documentation/iex> for more details on the API.

API Sample:

https://api.tiingo.com/iex/keyword/prices?startDate=prior_date&resampleFreq=12hour&columns=open,high,low,close,volume&token=API_KEY

When constructing the python request required obtaining the Company Details, you need to provide 5 values:

1. The first value, the **keyword**, 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.
3. The third value is the **startDate**, a prior date which should be 6 months 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.
4. The fourth value, **columns**, indicates what you want returned: open, high, low and close prices and volume.
5. The fifth value, **resampleFreq**, indicates in hours the resampling frequency. See the Tiingo documentation for allowed values.

For example, to get information about AAPL Historical Intraday Prices, you would call the API as:

<https://api.tiingo.com/iex/aapl/prices?startDate=2020-03-04&resampleFreq=12hour&columns=open,high,low,close,volume&token=0108ecd7f84bf97f8f11998a4e4561ea51da92c8>

The **response** of this Python request is a **JSON-formatted object**. **Figure 8** shows a sample response from the request. You need to parse this JSON object and extract some fields as required.

```
[
  {
    "date": "2020-03-04T17:00:00.000Z",
    "open": 295.6,
    "high": 303.02,
    "low": 295.425,
    "close": 302.84,
    "volume": 499116.0
  },
  {
    "date": "2020-03-05T17:00:00.000Z",
    "open": 297.45,
    "high": 297.74,
    "low": 291.525,
    "close": 292.65,
    "volume": 251683.0
  },
]
```

Figure 8: Sample JSON response from Tiingo API's Historical Intraday Prices Endpoint

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

Chart Data	Data from Tiingo Historical Intraday Prices API response JSON
Date	The value of key "date"
Stock Price	The value of key "close"
Volume	The value of key "volume"

Table 3: Mapping JSON data and HighCharts data

For mapping the Stock price data to data for HighCharts, create an array of data points (x1, y1) where x1 will be the date(in UTC format) 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 (also in UTC format) and y2 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 9** shows an example of the Stock Price/Volume chart.

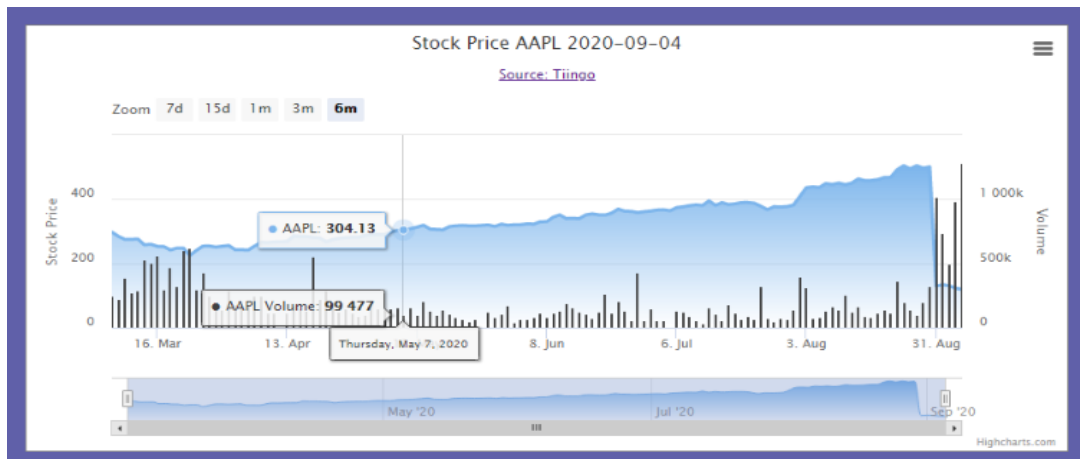


Figure 9: An Example of Chart showing Stock Price/Volume for 6 months

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: Tiingo” and should hyperlink to the Tiingo website: <https://api.tiingo.com/>. 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 10** shows an example of the Stock Price/Volume chart for 15 days zoom level.

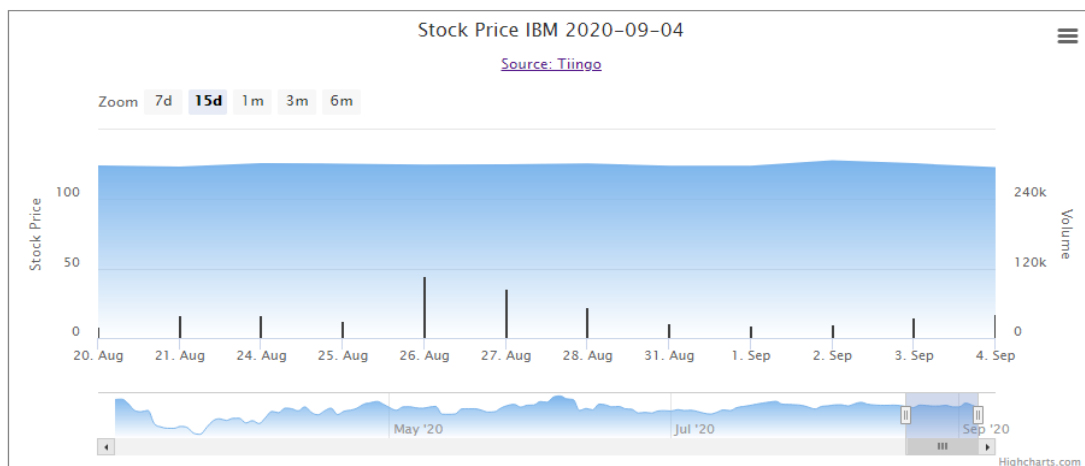


Figure 10: An Example of Chart showing Stock Price/Volume for 15 days

2.2.4 Displaying Stock News Tab

To display news for the company, we should use the stock ticker symbol from the form data to construct a Python request to the **News API** and display the results in your page. Please refer to <https://newsapi.org/docs> for more details.

API Sample: https://newsapi.org/v2/everything?apiKey=API_KEY&q=keyword

When constructing the python request required obtaining the News articles, you need to provide two values:

1. The first value is your **apiKey**, which you created as described in section 3.

2. The second value, **q**, is the stock ticker the user entered in the form.

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. **Only the articles that have *title*, *url*, *urlToImage* and *publishedAt* 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.

```
{
  "status": "ok",
  "totalResults": 1895,
  "articles": [
    {
      "source": {
        "id": "cnn",
        "name": "CNN"
      },
      "author": "Paul R. La Monica, CNN Business",
      "title": "Apple is now worth $2 trillion",
      "description": "Among its many accomplishments -- the iPhone",
      "url": "https://www.cnn.com/2020/08/19/tech/apple-stock-two-t",
      "urlToImage": "https://cdn.cnn.com/cnnnext/dam/assets/2008131",
      "publishedAt": "2020-08-19T14:53:04Z",
      "content": "New York (CNN Business)Among its many accomplishm",
    },
    {
      "source": {
        "id": null,
        "name": "MacRumors"
      },
      "author": "MacRumors Staff",
      "title": "Top Stories: Epic vs. Apple Escalates. AAPL Worth $"
```

Figure 11: Sample JSON response from News API's Endpoint

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

Card Data	Data from News API JSON response
Image	The value of key " <i>urlToImage</i> "
Title	The value of key " <i>title</i> "
Date	The value of key " <i>publishedAt</i> "
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 12**. The **"See Original Post"** hyperlink opens the original post in a new tab in the browser.

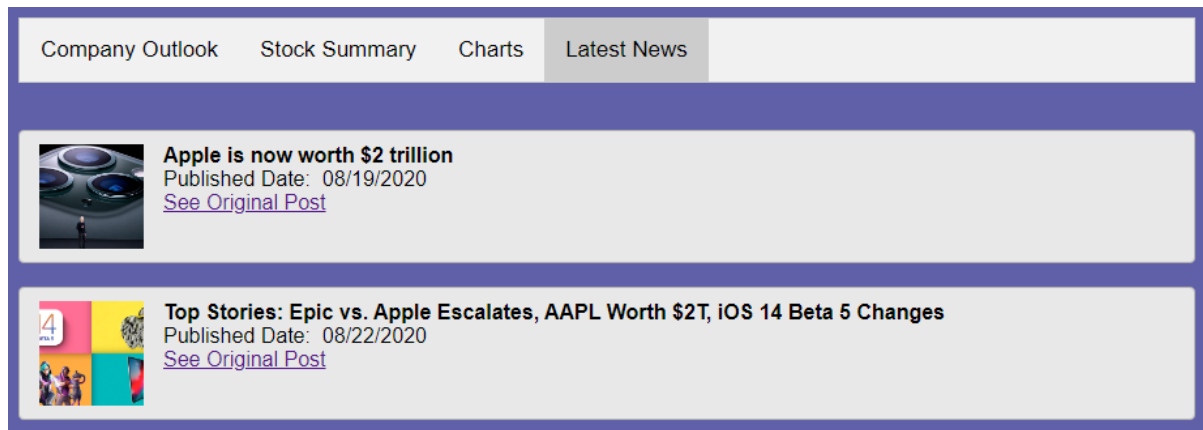


Figure 12: Example of Latest News

2.2.5 Displaying Error Message

If the Tiingo API service returns an Invalid Call error message, the page should display *“Error : No record has been found, please enter a valid symbol.”* **Figure 13** shows an example when searching for non-existent stock symbol “xyz”.

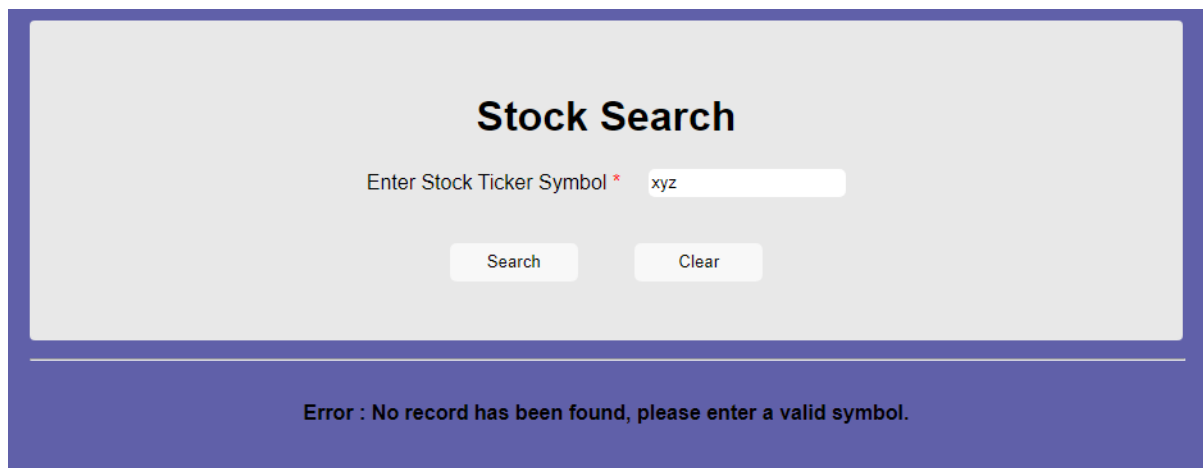


Figure 13: Search Result when there is no matching result

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 *Tiingo API* service and keep the value provided in the search form.

3. Hints

3.1 Tiingo API Documentation

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

<https://api.tiingo.com/>.

Click on the **Sign-up** button. Enter your **Username**, your @usc.edu **E-mail** address, select and confirm a **Password**, and click on **Sign-up**. Verify your account and go to

<https://api.tiingo.com/account/api/token>

to get your API_KEY.

3.2 Image References

Regarding the marker icon rendered beside the values of *Change*, *Change Percent*, a red-down arrow icon is displayed if the value is negative while a green-up arrow icon is displayed if the value is positive. The icons can be found at:

<https://csci571.com/hw/hw6/images/GreenArrowUp.jpg>

<https://csci571.com/hw/hw6/images/RedArrowDown.jpg>

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

You should use the domain name of the GAE/AWS/Azure service you created in HW #5 to make the request. For example, if your GAE/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 How to get News API Key

To apply for an API Key for newsapi.org, go to <https://newsapi.org/docs>.

Click on the “**Get API key**” button. You should fill out the form as shown in **Figure 14**. Click **Submit**. The API Key (News API key) will be displayed. Copy this key as you will use it in all the *News API* calls.

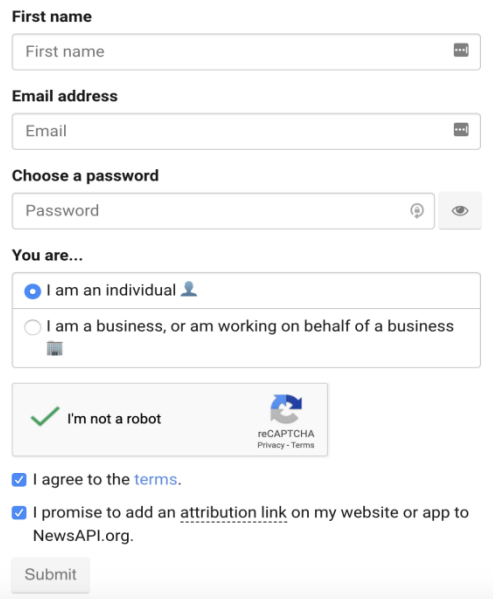
Register for API key


First name


Email address

Choose a password

You are...

☒ I am an individual 

☐ I am a business, or am working on behalf of a business 

☒ I'm not a robot 

☒ I agree to the [terms](#).

☒ I promise to add an [attribution link](#) on my website or app to NewsAPI.org.

Submit

Figure 14: Google News Register for API Key Form

3.5 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 Homework 6 link to refer to your new initial web search page for this exercise** (for example, **index.html**). Your files must be hosted on GAE, AWS or Azure cloud service. Graders will verify that this link is indeed pointing to one of the cloud services.

****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 should **not use JQuery or Bootstrap for Homework 6**.
- 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**.