# Homework 4: 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
- Getting hands-on experience with AWS

## 1.1 Cloud Exercise

- The backend of this homework must be implemented in the cloud on AWS using Python.
- See homework 3 for installation of AWS platforms.
- 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.

## 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 a tabular format.

## 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.

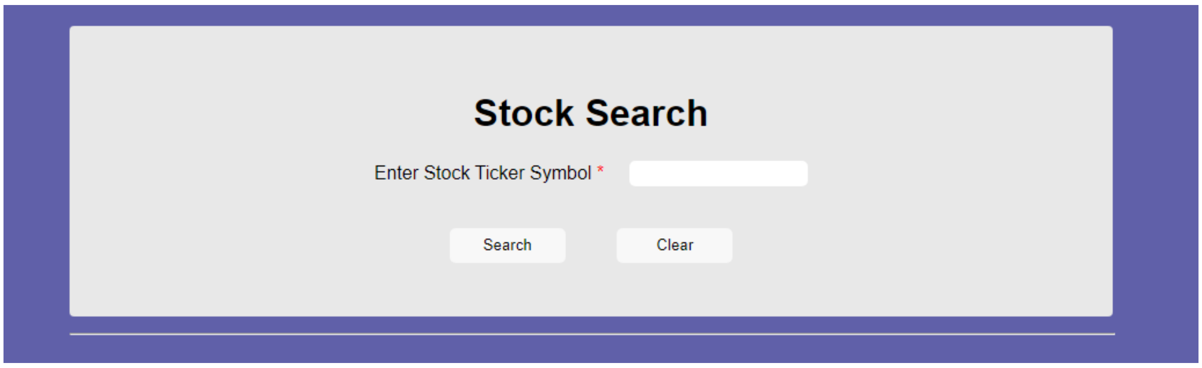


**Figure 1: Initial search page**

The search form has two buttons:

1. **Search button:**
   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**).

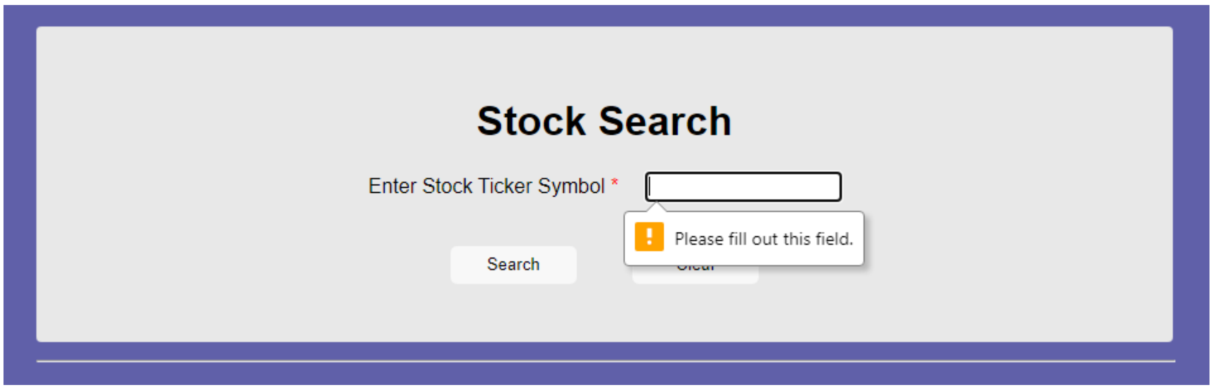


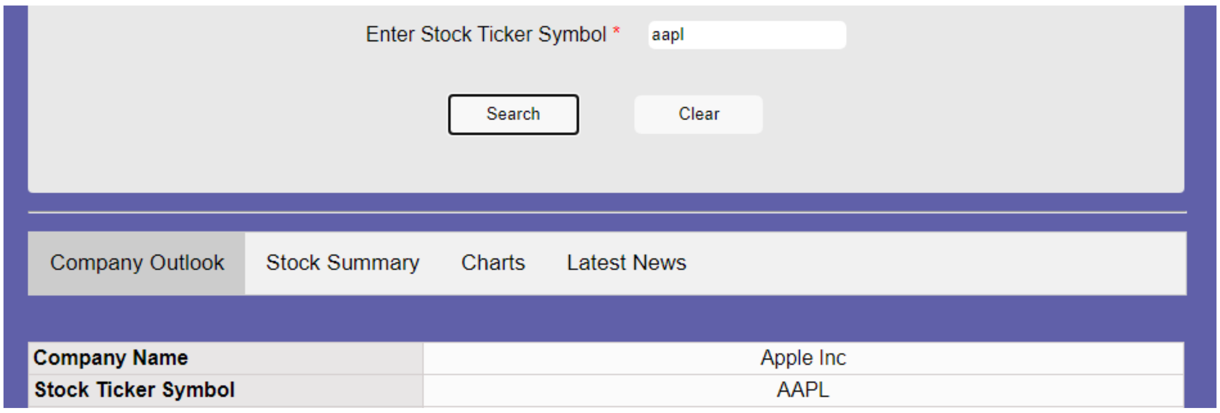
**Figure 2: An Example of Empty Input alert**

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**.



| | |
|---|---|
| **Company Name** | Apple Inc |
| **Stock Ticker Symbol** | AAPL |

**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*

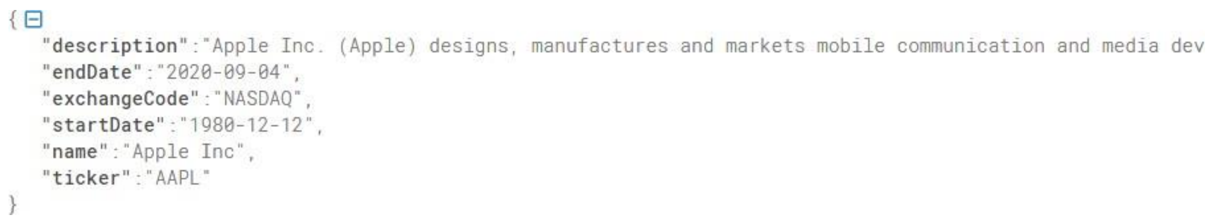
The process to create your API key is explained in section 3. When constructing the python request required obtaining the Company Details, you need to provide two values in the URL:

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"
}
```

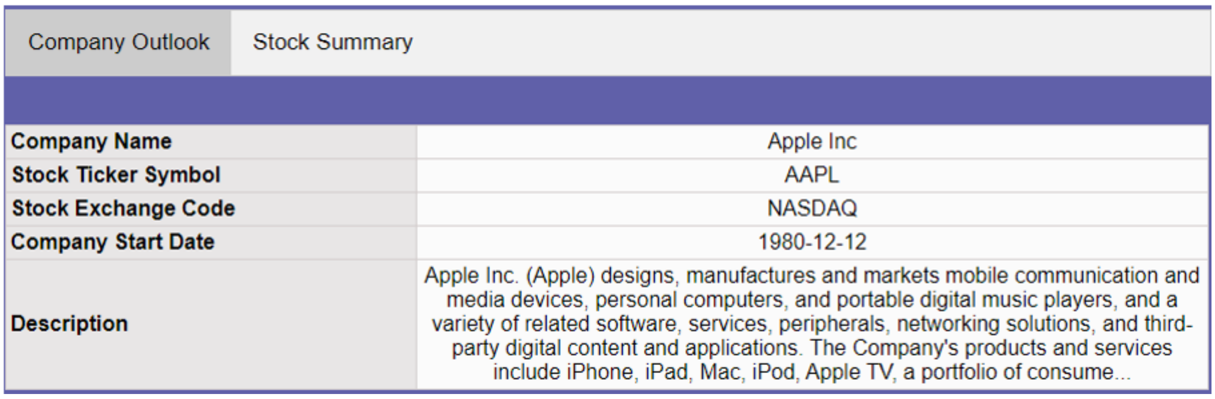
**Figure 4: Sample JSON response from Tiingo API's metadata endpoint**

| **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 *"exchangeCode"* |
| Company Start Date | The value of key *"startDate"* |
| Description | The value of key *"description"*. 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 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*

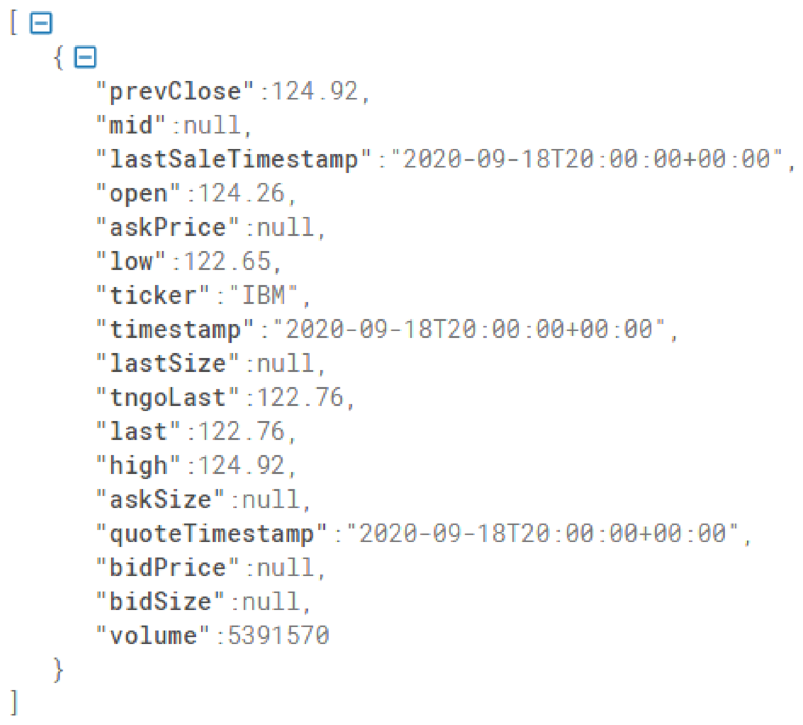
When constructing the python request required obtaining the Company Details, you need to provide two values in the URL:

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**.



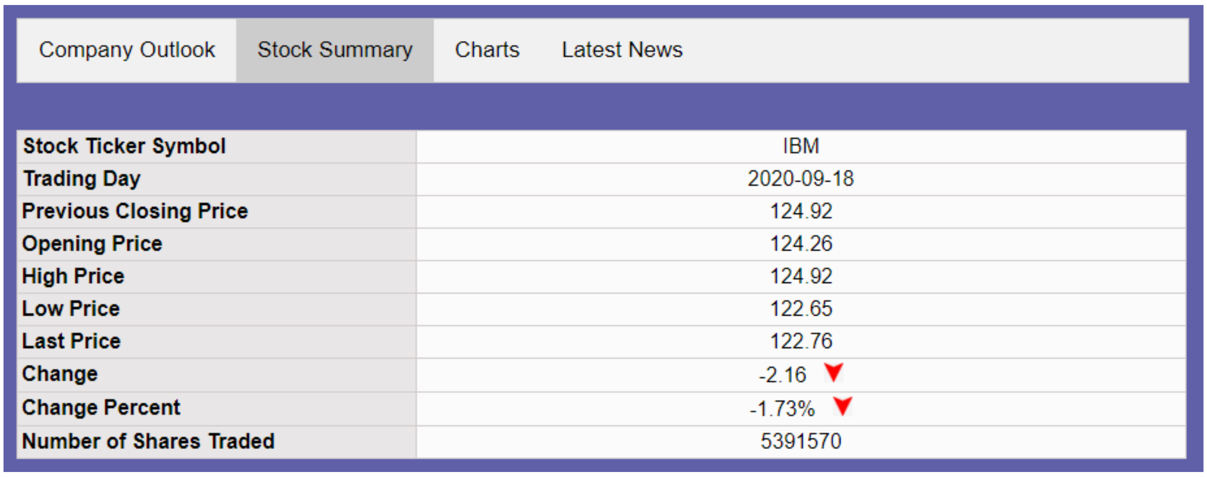
```
[
  {
    "prevClose":124.92,
    "mid":null,
    "lastSaleTimestamp":"2020-09-18T20:00:00+00:00",
    "open":124.26,
    "askPrice":null,
    "low":122.65,
    "ticker":"IBM",
    "timestamp":"2020-09-18T20:00:00+00:00",
    "lastSize":null,
    "tngoLast":122.76,
    "last":122.76,
    "high":124.92,
    "askSize":null,
    "quoteTimestamp":"2020-09-18T20:00:00+00:00",
    "bidPrice":null,
    "bidSize":null,
    "volume":5391570
  }
]
```

**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 *"ticker"* |
| Trading Day | The value of the key *"timestamp". Truncate the value to just display the date.* |
| Previous Closing Price | The value of key *"prevClose"* |
| Opening Price | The value of key *"open"* |
| High Price | The value of key *"high"* |
| Low Price | The value of key *"low"* |
| Last Price | The value of key *"last"* |
| Change | Difference between the value of key *"last"* and *"prevClose".* 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 = (last - prevClose)** |
| Change Percent | The Change Percent between the value of key *"last" and "prevClose".* 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. **Change Percent = (Change / prevClose) * 100** Please refer to below link for formula: https://www.skillsyouneed.com/num/percent-change.html |
| Number of Shares Traded | The value of key *"volume"* |

**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**.

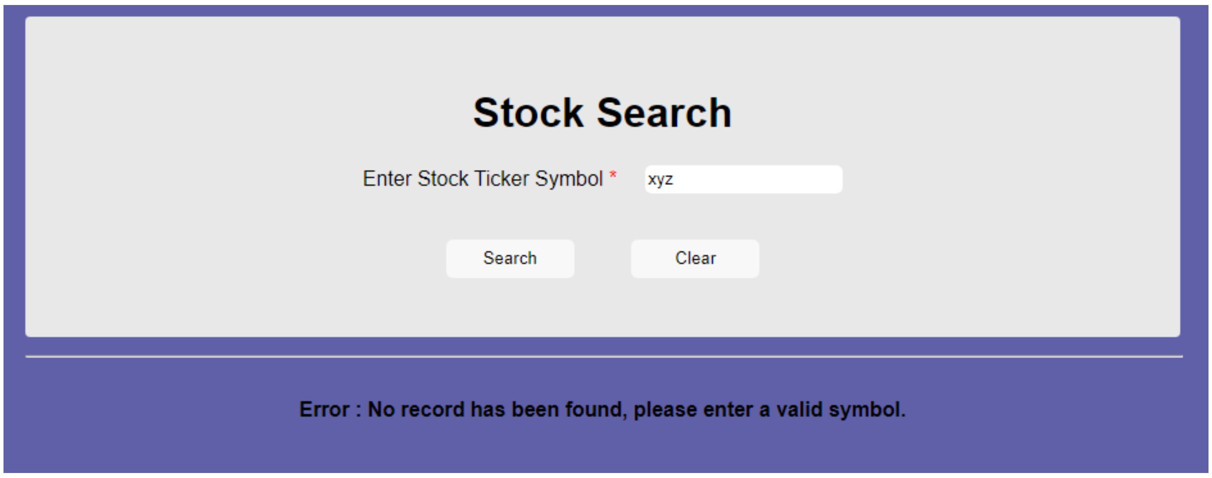


|  |  |
|---|---|
| Stock Ticker Symbol | IBM |
| Trading Day | 2020-09-18 |
| Previous Closing Price | 124.92 |
| Opening Price | 124.26 |
| High Price | 124.92 |
| Low Price | 122.65 |
| Last Price | 122.76 |
| Change | -2.16 ▼ |
| Change Percent | -1.73% ▼ |
| Number of Shares Traded | 5391570 |

**Figure 7: Example of Stock Summary Information Tab**

### 2.2.3 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*".



Error : No record has been found, please enter a valid symbol.

**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 @csueastbay.edu **E-mail** address, select andconfirm 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 on Canvas

### 3.3 Deploy Python file to the cloud (AWS)

You should use the domain name of the AWS service you created in HW #3 to make the request. For example, if your AWS server domain is called **example.elasticbeanstalk.com** :

AWS - http://example.elasticbeanstalk.com/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.

## 1. Files to Submit

Your files must be hosted on AWS cloud service. In your readme file, include the URL for your instance and submit it to Canvas

****IMPORTANT**:**

- 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.**