Stock information web interface

Group Project: Team 6

Kinjal Shukla, Gloria Rumao, Ruchi Pandey, Moksh Ajmera
Presentation Date: December 12, 2021
Financial Engineering 520: Python for Financial Application
Stevens Institute of Technology

Table of Contents

1. Introduction

This project is comprised of three parts: Use Flask to host a web server for stock information; store and fetch data from SQL database; and displaying the stock price and volume information on web pages.
This report covers the libraries and functions used in the code.

2. Project Flow chart

For this project, stock market data is first downloaded from the web browser and stored in local SQL database. There is a provision of dynamically storing the data by as per the date range of stock price. Using Flask, the data is then uploaded in a web browser in a tabular format.

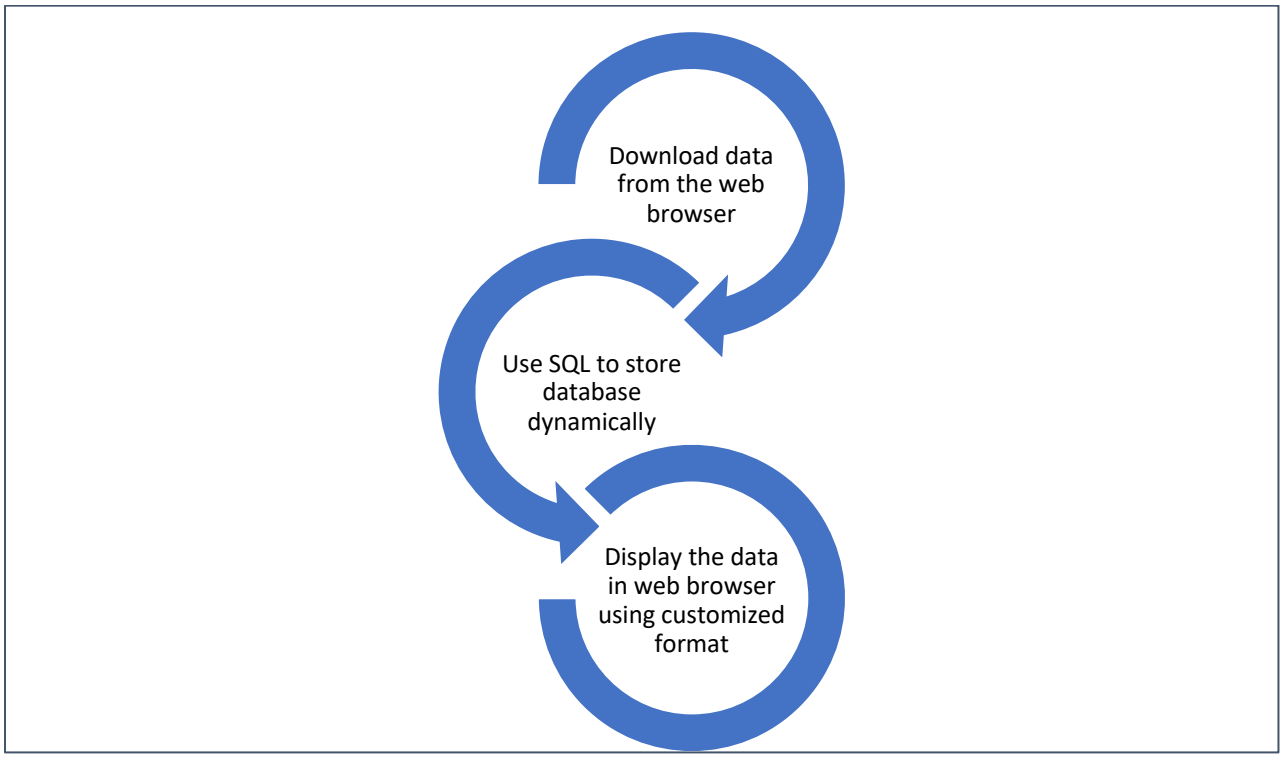

Figure 1: Description of the project flow

3. Libraries Used

Python has a provision of strong in-built libraries which help to use standard functionalities used in day-to-day automation. The libraries covered in this project are: flask, json, psycopg2, pandas, yfinance, and ipython display

3.a. Flask

Flask is a lightweight Web Server Gateway Interface web application framework. It is essentially a wrapper function used to display information on webpage.

Render_template is another function used. Templates are files that contain static data as well as placeholders for dynamic data. A template is rendered with specific data to produce a final document. Flask uses the Jinja template library to render templates. This project will use templates to render HTML which will display in the user's browser. In Flask, Jinja is configured to *auto escape* any data that is rendered in HTML templates. This means that it's safe to render user input; any characters they've entered that could mess with the HTML, such as < and > will be *escaped* with *safe* values that look the same in the browser but don't cause unwanted effects.

Request object: The request context keeps track of the request-level data during a request. When the Flask application handles a request, it creates a Request object based on the environment it received from the WSGI server. Because a *worker* (thread, process, or coroutine depending on the server) handles only one request at a time, the request data can be considered global to that worker during that request. Flask uses the term *context local* for this.
Flask automatically *pushes* a request context when handling a request.

Redirect function: Flask class has a redirect() function. When called, it returns a response object and redirects the user to another target location with specified status code.

url_for in Flask is used for creating a URL to prevent the overhead of having to change URLs throughout an application (including in templates). Without url_for, if there is a change in the root URL of your app then you have to change it in every page where the link is present.

flash method is used to generate informative messages in the flask. It creates a message in one view and renders it to a template view function called next. In other words, the flash() method of the flask module passes the message to the next request which is an HTML template.

3.b. Json

JSON JavaScript Object Notation is a format for structuring data. It is mainly used for storing and transferring data between the browser and the server. Python supports use of JSON by providing library for the same.

3.c. Psycopg2

Psycopg is the PostgreSQL database adapter for the Python programming language. It features client-side and server-side cursors, asynchronous communication, and notifications, "COPY TO/COPY FROM" support. Many Python types are supported out-of-the-box and adapted to matching PostgreSQL data types; adaptation can be extended and customized

3.d. NumPy

NumPy is a Python library used for working with arrays. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. It aims to provide an array object that is up to 50x faster than traditional Python lists. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

3.e. Pandas

Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data, including:

- Data cleansing
- Data fill
- Data normalization
- Merges and joins
- Data visualization
- Statistical analysis
- Data inspection
- Loading and saving data

3.f. yfinance

Yfinance is a python package that enables us to fetch historical market data from Yahoo Finance API in a Pythonic way. It solves the problem by allowing users to download data using python and it has some features also which makes it favorable to use for stock data analysis.

3.g. Ipython Display -> HTML

Used to display the HTML representation of an object

4. Code References:

Below are the code experts for different sections.

4.a. Libraries:

```
from flask import
Flask,render_template,request,redirect,url_for,flash
import json
import psycopg2
import numpy as np
import pandas as pd
import yfinance as yf
from IPython.display import HTML
```

4.b. Flask:

```
app = Flask(__name__)
app.secret_key = '1234'
@app.route('/')

def home():

    return render_template('home.html')

@app.route('/data', methods = ['GET','POST'])
```

4.c. Database creation:

```
conn = psycopg2.connect(database="DB_Appl1", user='postgres',
password='root', host='127.0.0.1', port= '5432')
        conn.autocommit = True
        cur = conn.cursor()
        #print("iNSIDE dATABASE")
        cur.execute('''CREATE TABLE Appl_records
                    (
                        Date DATE NOT NULL,
                        Open FLOAT NOT NULL,
                        High FLOAT NOT NULL,
                        Low FLOAT NOT NULL,
                        Close FLOAT NOT NULL,
                        Adj_Close FLOAT NOT NULL,
                        Volume BIGINT NOT NULL,
                        Ticker VARCHAR(255) NOT NULL
                        );''')
```

5. Output

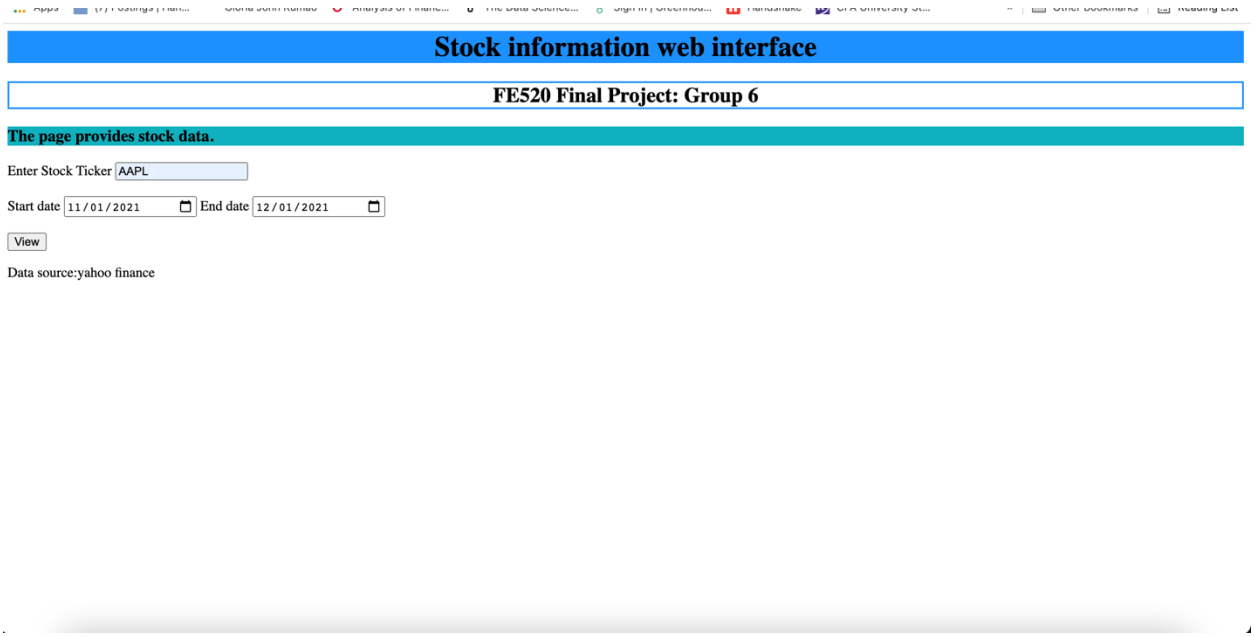

Figure 2: Use of Flask to get data

DB_Appll1/postgres@PostgreSQL 14

Query Editor

```
1  Select * from appl_records;
```

Notifications | Data Output | Messages | Explain | Query History

	date date	open double precision	high double precision	low double precision	close double precision	adj_close double precision	volume bigint	ticker character varying (255)
1	2021-11-01	148.99000549316406	149.6999969482422	147.8000030517578	148.9600067138672	148.742919921875	74588300	AAPL
2	2021-11-02	148.66000366210938	151.57000732421875	148.64999389648438	150.02000427246094	149.80137634277344	69122000	AAPL
3	2021-11-03	150.38999938964844	151.97000122070312	149.82000732421875	151.49000549316406	151.2692413330078	54511500	AAPL
4	2021-11-04	151.5800018310547	152.42999267578125	150.63999938964844	150.9600067138672	150.74000549316406	60394600	AAPL
5	2021-11-05	151.88999938964844	152.1999969482422	150.05999755859375	151.27999877929688	151.27999877929688	65414600	AAPL
6	2021-11-08	151.41000366210938	151.57000732421875	150.16000366210938	150.44000244140625	150.44000244140625	55020900	AAPL
7	2021-11-09	150.1999969482422	151.42999267578125	150.05999755859375	150.80999755859375	150.80999755859375	56787900	AAPL
8	2021-11-10	150.02000427246094	150.1300048828125	147.85000610351562	147.9199981689453	147.9199981689453	65187100	AAPL
9	2021-11-11	148.9600067138672	149.42999267578125	147.67999267578125	147.8699951171875	147.8699951171875	41000000	AAPL
10	2021-11-12	148.42999267578125	150.39999389648438	147.47999572753906	149.99000549316406	149.99000549316406	63632600	AAPL
11	2021-11-15	150.3699951171875	151.8800048828125	149.42999267578125	150	150	59222800	AAPL
12	2021-11-16	149.94000244140625	151.49000549316406	149.33999633789062	151	151	59256200	AAPL
13	2021-11-17	151	155	150.99000549316406	153.49000549316406	153.49000549316406	88807000	AAPL
14	2021-11-18	153.7100067138672	158.6699981689453	153.0500030517578	157.8699951171875	157.8699951171875	137827700	AAPL
15	2021-11-19	157.6499938964844	161.02000427246094	156.52999877929688	160.55000305175781	160.55000305175781	117305600	AAPL

Figure 3: Date gets stored in a db file

results.html

	Open	High	Low	Close	Adj Close	Volume	Ticker
2021-11-01	148.990005	149.699997	147.800003	148.960007	148.742920	74588300	AAPL
2021-11-02	148.660004	151.570007	148.649994	150.020004	149.801376	69122000	AAPL
2021-11-03	150.389999	151.970001	149.820007	151.490005	151.269241	54511500	AAPL
2021-11-04	151.580002	152.429993	150.639999	150.960007	150.740005	60394600	AAPL
2021-11-05	151.889999	152.199997	150.059998	151.279999	151.279999	65414600	AAPL
2021-11-08	151.410004	151.570007	150.160004	150.440002	150.440002	55020900	AAPL
2021-11-09	150.199997	151.429993	150.059998	150.809998	150.809998	56787900	AAPL
2021-11-10	150.020004	150.130005	147.850006	147.919998	147.919998	65187100	AAPL
2021-11-11	148.960007	149.429993	147.679993	147.869995	147.869995	41000000	AAPL
2021-11-12	148.429993	150.399994	147.479996	149.990005	149.990005	63632600	AAPL

Figure 3: Output of stock data in tabular format from SQL database

6. Conclusion

The code was clearly able to demonstrate the use of Flask and SQL to display and store data. To further improve the project, one can work on the following points:

1. Use local data instead of web data to store in SQL
2. Use of technical indicators to display buy-sell logic on web page
3. Use plot functions to display the chart of stock price.

7. References

- docs.python.org
- https://pypi.org/project/yfinance/
- https://flask.palletsprojects.com/en/2.0.x/
- https://pypi.org/project/psycopg2/