A logo for a university

Description automatically generated

Higher diploma in science in computing

Lecturer: Patrizio Simeoni

Subject: Scripting

Assignment 2

Course: CW\_KRSIT\_H

Date: 11/3/2025

Student: John Paul Larkin

Student number: C00001754

## Introduction

This project retrieves and analyses stock price data from two different sources, with a focus on learning to integrate JSON APIs and web scraping in Python. The assignment objectives were to practice acquiring financial data from a reliable API, parse additional contextual news information via a web scraper, store the combined results in a database, and visualise the data in table and graph form. JSON was chosen for real-time stock information from Alpha Vantage API, because it offers convenient, structured data for the key daily metrics. Web scraping was used for Yahoo Finance headlines, and BeautifulSoup provided a straightforward approach to parse the HTML.

## Data sources

The data source for the stock information was the Alpha Vantage API, specifically the “GLOBAL\_QUOTE” endpoint, which returns a dictionary of information such as symbol, current price, trading volume, opening price, and other metrics. That data, in JSON format, was fetched with an HTTP GET request containing the function parameter set to GLOBAL\_QUOTE and the stock’s symbol.

For the headlines, a Yahoo search URL was invoked with requests, and the response was parsed using BeautifulSoup. The parsing targeted HTML elements for each article’s headline and URL, returning a tuple of (headline, url). Originally, three headlines for each company were collected daily, although they were ultimately not integrated into the final visualisation.

## Data storage

A simple SQLite database with two tables was used to store this information. One table, called “stocks,” contains rows for each company’s daily stock data, including columns for symbol, open, high, low, price, volume, latest trading day, previous close, change, and change\_percent. The schema imposes a uniqueness constraint on the combination of symbol and latest\_trading\_day, preventing duplicates for the same company on the same day.

A second table, called “headlines,” was designed to store the news headlines for each company. It holds symbol, headline, url, and latest\_trading\_day, with a uniqueness constraint on the combination of headline and latest\_trading\_day. A foreign key references the symbol in the “stocks” table so that the related records stay consistent.

A diagram of a stock market

AI-generated content may be incorrect.

## Automation process

To automate the scraping process, I created a simple bash script to run the python script within the virtual environment.

/Users/jpl/code/setu/scripting/assignment\_two/ass\_2\_venv/bin/python /Users/jpl/code/setu/scripting/assignment\_two/stock.py

I made the script executable using:

chmod +x run\_stock.sh

I then edited my crontab with:

0 23 \* \* \* /Users/jpl/code/setu/scripting/assignment\_two/run\_stock.sh >> /Users/jpl/code/setu/scripting/assignment\_two/stock.log 2>&1

This automated the collection of data at 11PM GMT, which is three hours after the close of the Nasdaq stock exchange.

The script loops through each company, calling the Alpha Vantage endpoint for basic financial data and then using the Yahoo scraping routine for headlines. It assembles results into a pair of lists—one for the stock data, one for the headlines—and then writes these to the database by calling an update function that inserts each day’s records into the two tables. This results in daily historical data for each symbol. The script includes methods to verify the data integrity by printing table contents to a text file or the console, although I have commented these out in the submitted verion.

## Data visualisation

The user is prompted to display data for individual companies, or all companies together. In the below example the table shows all data for the four hardcoded companies. Nvidia, Microsoft, Tesla and Google.

Green cells – Indicate the value has risen from the previous day

Red cells – Indicated the value has fallen.

White cells – Indicate the first day of data, or no change.

A screen shot of a black board with numbers and numbers

AI-generated content may be incorrect.

The graph below is an example for a single company (Tesla) represented on a candlestick graph. On each date there are two candlesticks. The left candlestick(in yellow) is the high and low for the stock value during that days trading.

The right candlestick (green or red) represents the opening and closing price for the stock. Red indicates that the stock price fell during that days trading.

The blue line graphs the closing price of the stock on that date.

Note the 8th and 9th are weekend days, so there was no trading. I also had issues with the 4th and 5th, so did not record the data on those days. Ideally, with a longer timeline, I would overcome these issue and present a better view of the data.

A graph with a line pointing at the end

AI-generated content may be incorrect.

## Lessons learned

At first, I stored all the headlines for a given company(and day), in JSON format as a single column alongside the stock data for that day, but this proved unwieldy. I decided to split the data into two separate tables, which made for clearer relationships and easier handling. This approach not only kept the data organised but also simplified maintenance and troubleshooting.

The original URL I used to search for a particular company symbol sometimes did not work correctly because the search autocorrect was attempting to fix the search term's spelling and returning a different result. After experimenting, I discovered that following the “search for instead” (is spell correct) link resulted in a stable URL that worked for all searches.

Another significant observation was that the user experience in a scraping workflow can depend heavily on the structure of the target site. Small changes to the HTML can break the scraping process, highlighting the need for flexible parsing methods.

I also experimented with Python typing through mypy. Using typed dictionaries and lists made the code more explicit and helped catch potential errors early. There was a pretty steep learning curve in this, but ultimately it helped to improve my understanding of py thon types and in turn, improved code quality.

Finally, I originally, I used a complicated process with BeautifulSoup to parse the HTML, target the headline, and then navigate through its parent element and siblings to find the URL. I later realised that by targeting the article elements, iterating through them, and searching for the unique span.s-title, I could greatly simplify the scraping process.

## Potential improvements

Potential improvements would include reintroducing the headlines into the analysis by either summarising them with a language model or calculating sentiment and comparing it with the daily changes in price.

Another enhancement could be to allow the user to add companies themselves by adding another table to the database. This would enable the user to add a company which is then persisted in the database and included in future scraping results.

These additions would create richer insight into the relationships between market news and performance trends. Overall, the project offered practical experience in data integration, provided a starting point for more extensive financial analyses, and helped solidify knowledge about structured data storage and Python typing.

## Conclusion

This project solidified my understanding of structured data storage, JSON-based API consumption, and web scraping. I learned to manage data integrity through relational schemas, adapt to HTML changes, and automate daily updates. Python typing provided clarity, catching errors early and enhancing code reliability.