**Creating a Stock Dashboard with Streamlit**

[[Sugath Mudali](https://medium.com/@sugath.mudali?source=post_page-----441039bdf7e0--------------------------------)](https://medium.com/@sugath.mudali?source=post_page-----441039bdf7e0--------------------------------)

[Sugath Mudali](https://medium.com/@sugath.mudali?source=post_page-----441039bdf7e0--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Ff54d3df0bb58&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40sugath.mudali%2Fcreating-a-stock-dashboard-with-streamlit-441039bdf7e0&user=Sugath+Mudali&userId=f54d3df0bb58&source=post_page-f54d3df0bb58----441039bdf7e0---------------------post_header-----------)

4 min read

·

Sep 8

34

In this article, we will explore how to build a Stock dashboard using [Streamlit](https://streamlit.io/).

*Disclaimer: The information provided here is provided solely for educational purposes and is not intended to be personal financial, investment, or other advice.*

[Streamlit](https://streamlit.io/) is an open-source app framework that allows you to quickly create visually appealing web applications. The framework library’s various components are described in the Streamlit [documentation](https://docs.streamlit.io/).

The dashboard takes advantage of the [multi-page app](https://blog.streamlit.io/introducing-multipage-apps/) design strategy made popular in Streamlit’s 1.10 update. The current scope only consists of two pages, Home and Earnings, but it is easily expandable by adding more pages. In addition to the multi-page approach, the dashboard periodically refreshes pages using the [*streamlit-autorefresh*](https://pypi.org/project/streamlit-autorefresh/) module. For tables, I opted for the [AgGrid](https://blog.streamlit.io/building-a-pivottable-report-with-streamlit-and-ag-grid/) library instead of Streamlit’s tables, as it offers greater features. See [7 Reasons Why You Should Use the Streamlit AgGrid Component](https://towardsdatascience.com/7-reasons-why-you-should-use-the-streamlit-aggrid-component-2d9a2b6e32f0) for an overview of Streamlit and AgGrid.

We will use [*yfinance*](https://pypi.org/project/yfinance/) for historic and other stock related data and the [Alpha Vantage API](https://www.alphavantage.co/documentation/) for the Earnings page. Check out my article, [Company Earnings and Income Dashboards with MongoDB and Alpha Vantage API](https://medium.com/insiderfinance/company-earnings-and-income-dashboards-with-mongodb-and-alpha-vantage-api-8c03341f8d3c) for accessing stock data via the Alpha Advantage API.

The code is available on [GitHub](https://github.com/smudali/stocks-analysis/tree/main/dasboard).

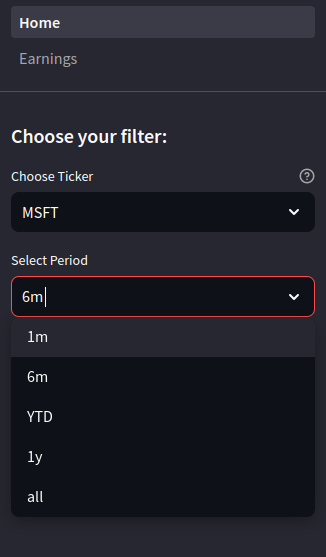
**Python Libraries**

The required Python libraries are:

* [yfinance](https://pypi.org/project/yfinance/): to access financial market data
* [streamlit](https://docs.streamlit.io/), st\_aggrid: UI components
* [plotly](https://plotly.com/python/): plotting graphs
* [pandas](https://pandas.pydata.org/docs/getting_started/install.html): DataFrame
* [sqlalchemy](https://www.sqlalchemy.org/): access SQLite database
* [TA-Lib](https://github.com/TA-Lib/ta-lib-python): a Python wrapper for [TA-LIB](http://ta-lib.org/); used for various technical indicators such as MA, MACD, RSI, etc.

**Home Page**

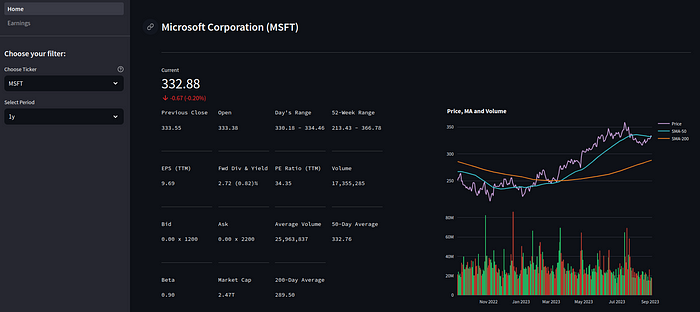
The dashboard consists of a side panel with two pages, as shown below:



Side Panel

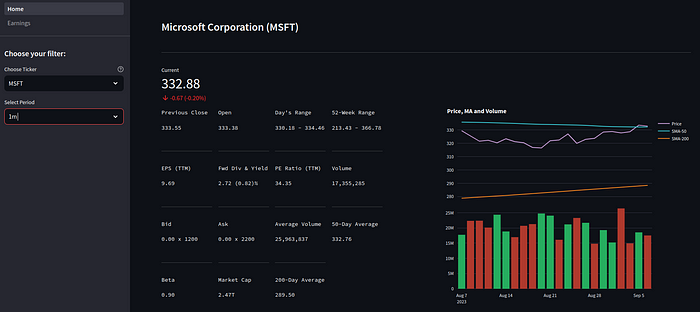
A list of ticker symbols is defined in *01\_Home.py*. The selection period offers five periods for charts. A SQLite database is used to store historical data. If no historic data is found, data for a two-year period is downloaded and stored in the SQLite database using the *yfinance* API. The start date is set to the most recent date to download the delta if historical data is already available for a ticker. This prevents the download of past historical data that hasn’t changed. The *load\_history\_data* method in [*01\_Home.py*](https://github.com/smudali/stocks-analysis/blob/main/dasboard/01_Home.py) has this logic.

Here is a snapshot of the MSFT ticker with a one-year history period.



Snapshot for the ticker MSFT with the history set to one year

Changes to the selection period are reflected in the graphs, as shown below:



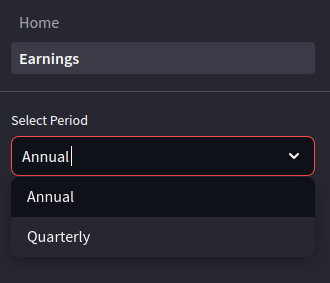
Snapshot for the ticker MSFT with the history set to one month

Other information, such as the current price, bid, etc. are refreshed periodically by the *streamlit-autorefresh* (set to 5 minutes) module.

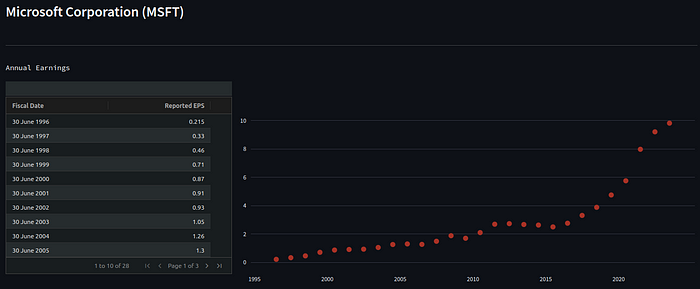
**Earnings Page**

Using a script and the Alpha Vantage API, earnings data is pre-populated. Since Earnings are updated quarterly, it is prudent to pre-populate them rather than accessing them in real time. The script is available on [GitHub](https://github.com/smudali/stocks-analysis/blob/main/dasboard/utils/insert_earnings.py)

The earnings page offer two periods: Annual or Quarterly

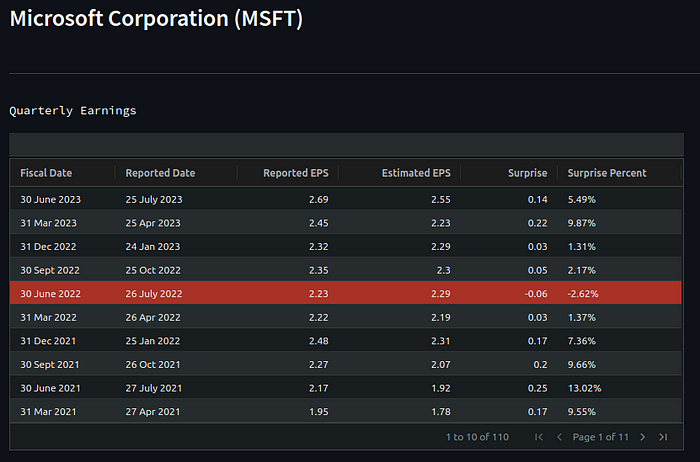


Two selection periods for Earnings



Annual Earnings for the ticker MSFT

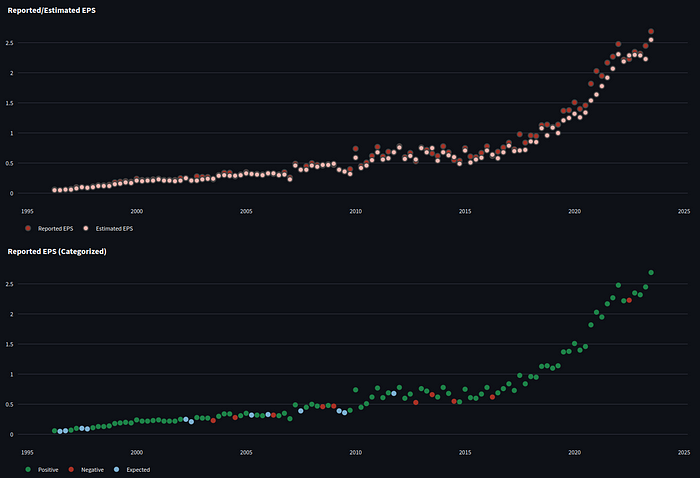
Quarterly earnings:



Quarterly Earnings for the ticker MSFT

Negative earnings are highlighted in red

Two charts plot Reported/Estimated Earnings and Earnings categorized into three groups: Positive (above expected), Expected (as expected) and Negative (below expected)



Quarterly Earnings charts for the ticker MSFT

**Notes**

Streamlit has recently introduced an experimental connection module dubbed [st.experimental\_connection](https://docs.streamlit.io/library/api-reference/connections/st.experimental_connection). The loading of data from a SQLite table into a DataFrame was successful. However, saving from a DataFrame to a table did not function as expected because the underlying SQLAlchmey library threw an exception because it could not access a cursor object from the engine.

**Further Reading**

1. [Plot Candlestick, RSI, Bollinger Bands, and MACD charts using yfinance Python API](https://medium.com/insiderfinance/plot-candlestick-rsi-bollinger-bands-and-macd-charts-using-yfinance-python-api-1c2cb182d147)
2. [Plot Moving Averages using Python APIs](https://medium.com/@sugath.mudali/plot-moving-averages-using-python-apis-9a313b2b75ae)
3. [Historical Stock Price Data using Python APIs](https://medium.com/analytics-vidhya/historical-stock-price-data-using-python-apis-eb80e8c39016)