Perfect project idea ? Let’s build your **“Stock Analysis Dashboard”** step by step using **yfinance**, **plotly**, and **Streamlit**.

**Project Structure**

stock-analysis-dashboard/

app.py # Streamlit dashboard

requirements.txt # Dependencies

README.md # Usage guide

reports/ # Sample analysis reports (CSV exports)

screenshots/ # Dashboard screenshots

**requirements.txt**

yfinance>=0.2

pandas>=2.0

numpy>=1.25

plotly>=5.20

streamlit>=1.37

**app.py**

import streamlit as st

import yfinance as yf

import pandas as pd

import numpy as np

import plotly.graph\_objs as go

from datetime import datetime, timedelta

# ------------------------------

# Helper Functions

# ------------------------------

def fetch\_data(ticker, period="6mo"):

data = yf.download(ticker, period=period)

data.reset\_index(inplace=True)

return data

def calculate\_indicators(df):

df["SMA20"] = df["Close"].rolling(window=20).mean()

df["EMA20"] = df["Close"].ewm(span=20, adjust=False).mean()

# RSI

delta = df["Close"].diff()

gain = np.where(delta > 0, delta, 0)

loss = np.where(delta < 0, -delta, 0)

avg\_gain = pd.Series(gain).rolling(window=14).mean()

avg\_loss = pd.Series(loss).rolling(window=14).mean()

rs = avg\_gain / avg\_loss

df["RSI"] = 100 - (100 / (1 + rs))

return df

def plot\_candlestick(df, ticker):

fig = go.Figure(data=[go.Candlestick(

x=df["Date"],

open=df["Open"],

high=df["High"],

low=df["Low"],

close=df["Close"],

name="Candlestick"

)])

fig.update\_layout(title=f"{ticker} Candlestick Chart", xaxis\_rangeslider\_visible=False)

return fig

def plot\_moving\_averages(df, ticker):

fig = go.Figure()

fig.add\_trace(go.Scatter(x=df["Date"], y=df["Close"], mode="lines", name="Close"))

fig.add\_trace(go.Scatter(x=df["Date"], y=df["SMA20"], mode="lines", name="SMA20"))

fig.add\_trace(go.Scatter(x=df["Date"], y=df["EMA20"], mode="lines", name="EMA20"))

fig.update\_layout(title=f"{ticker} Moving Averages", xaxis\_title="Date", yaxis\_title="Price")

return fig

def plot\_rsi(df, ticker):

fig = go.Figure()

fig.add\_trace(go.Scatter(x=df["Date"], y=df["RSI"], mode="lines", name="RSI"))

fig.add\_hline(y=70, line\_dash="dash", line\_color="red")

fig.add\_hline(y=30, line\_dash="dash", line\_color="green")

fig.update\_layout(title=f"{ticker} RSI Indicator", xaxis\_title="Date", yaxis\_title="RSI")

return fig

# ------------------------------

# Streamlit UI

# ------------------------------

st.set\_page\_config("📈 Stock Analysis Dashboard", layout="wide")

st.title("📊 Stock Analysis Dashboard")

# Sidebar

st.sidebar.header("Stock Settings")

ticker = st.sidebar.text\_input("Enter Stock Ticker (e.g., AAPL, TSLA, MSFT):", "AAPL")

period = st.sidebar.selectbox("Select Period", ["1mo", "3mo", "6mo", "1y", "2y", "5y", "max"])

if st.sidebar.button("Analyze"):

with st.spinner("Fetching data..."):

df = fetch\_data(ticker, period)

df = calculate\_indicators(df)

# Show summary

st.subheader(f"Performance Summary: {ticker}")

st.write(df.describe()[["Open", "High", "Low", "Close", "Volume"]])

# Charts

col1, col2 = st.columns(2)

with col1:

st.plotly\_chart(plot\_candlestick(df, ticker), use\_container\_width=True)

with col2:

st.plotly\_chart(plot\_moving\_averages(df, ticker), use\_container\_width=True)

st.plotly\_chart(plot\_rsi(df, ticker), use\_container\_width=True)

# CSV Export

csv = df.to\_csv(index=False).encode("utf-8")

st.download\_button("📥 Download Data as CSV", csv, f"{ticker}\_analysis.csv", "text/csv")

**README.md**

# Stock Analysis Dashboard

## Objective

Analyze and visualize stock trends with indicators.

## Tools

- `yfinance` – fetch historical stock data

- `plotly` – interactive charts

- `streamlit` – dashboard UI

## Features

1. Input stock ticker.

2. Fetch historical price data from Yahoo Finance.

3. Calculate \*\*SMA, EMA, RSI\*\* indicators.

4. Display \*\*candlestick\*\* and \*\*moving average\*\* charts.

5. Show RSI chart with overbought/oversold zones.

6. Display performance summary table.

7. Export data as CSV.

## Run

```bash

pip install -r requirements.txt

streamlit run app.py

**Deliverables**

* Dashboard (app.py)
* Analysis reports (.csv exports)
* Screenshots (/screenshots)

---

\*\*Deliverables included\*\*:

- Interactive \*\*Streamlit dashboard\*\* (`app.py`)

- Indicators (SMA, EMA, RSI)

- Candlestick + line charts (\*\*Plotly\*\*)

- Performance summary table

- CSV export option