STOCK PRICE ANALYSIS WEB APPLICATION

A PYTHON-FLASK, HTML & JAVASCRIPT WEB APPLICATION FOR STOCK DATA ANALYSIS AND PREDICTION

NAME: SAYAN BANERJEE

DATE: 05-03-2025

INTRODUCTION:

In today's fast-paced stock markets, timely and accurate data is essential for making informed investment decisions. this **Stock Price Analysis Web Application** is designed to offer real-time stock price data, historical analysis, and predictive insights to investors, analysts, and traders. By combining powerful backend technologies with an interactive frontend, this web app provides both technical analysis tools and machine learning predictions for stock price trends.

- **Technologies Used:**
  - **Backend**: **Flask** (Python web framework), **Alpha Vantage** API, **Scikit-learn** (for Linear Regression model), **Pandas** (for data manipulation).
  - **Frontend**: **HTML/CSS** for layout and styling, **JavaScript** for API calls and data handling, **Chart.js** for interactive data visualization.

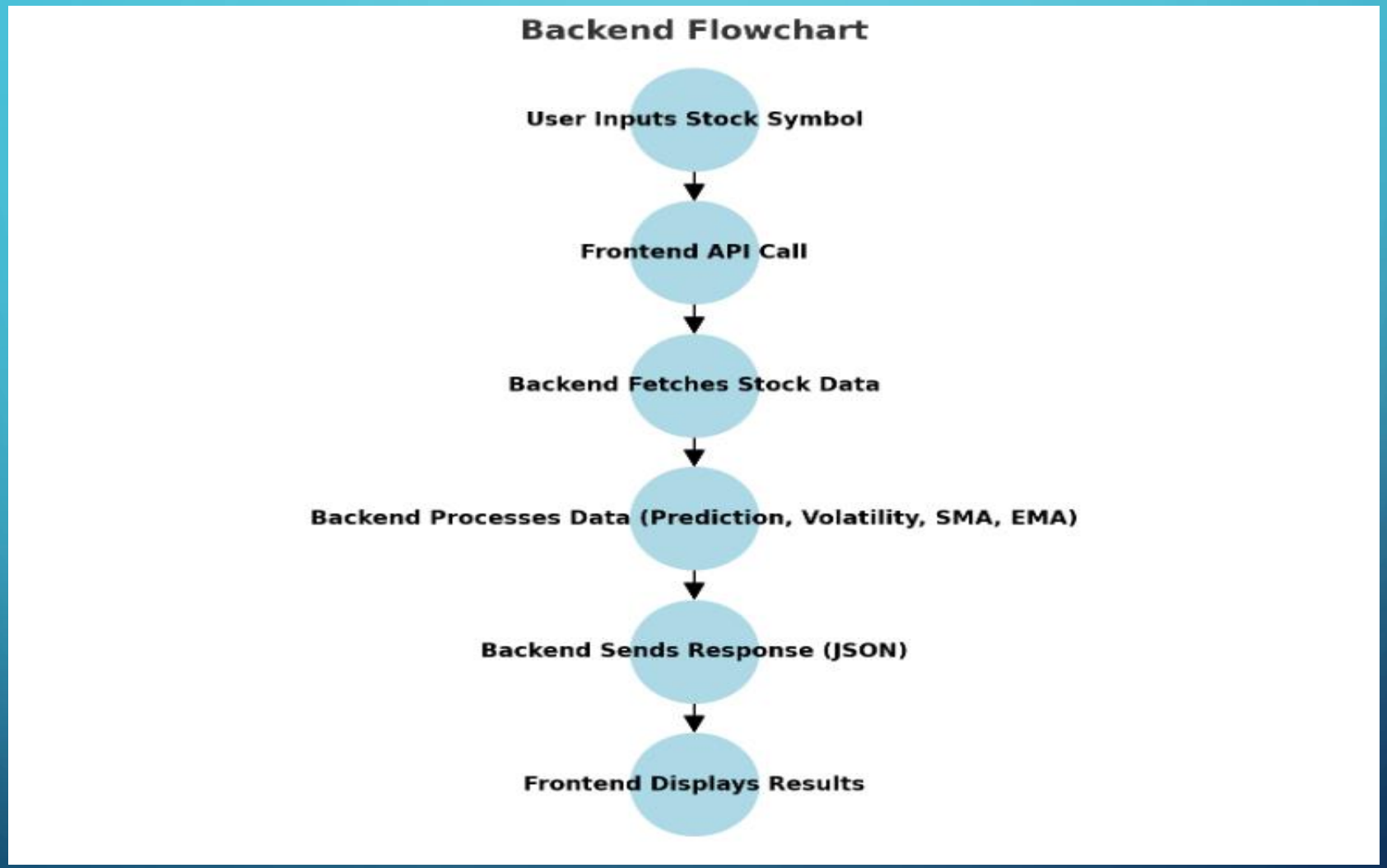
KEY FEATURES:

- **Real-Time Data:** The application fetches up-to-date stock prices and historical data using the **Alpha Vantage API**.
- **Stock Price Prediction:** It leverages **Linear Regression** to predict future stock prices based on historical trends.
- **Volatility Measurement:** It calculates stock volatility, helping users understand the potential risk associated with different stocks.
- **Moving Averages:** Displays **Simple Moving Average (SMA)** and **Exponential Moving Average (EMA)**, which are key indicators for technical analysis.
- **Interactive Charts:** Visualize stock prices, moving averages, and volatility trends in an engaging, user-friendly format using **Chart.js**.

BACKEND OVERVIEW:

Backend Architecture:

- **Flask Framework:**
  - Handles API requests and responses.
  - Provides routes like /stock-history for fetching stock data.
- **API Integration (Alpha Vantage):**
  - Fetches real-time and historical stock data (e.g., stock prices).
  - Uses TIME_SERIES_DAILY to get daily stock price data.
- **Data Processing:**
  - **Linear Regression** for stock price prediction.
  - **Volatility Calculation**: Standard deviation of daily returns.
  - **Moving Averages:**
    - **Simple Moving Average (SMA)**: 10-day average.
    - **Exponential Moving Average (EMA)**: 10-day weighted average.
- **Data Response:**
  - Returns stock prices, predicted prices, volatility, SMA, and EMA as a **JSON response**.
- **Error Handling:**
  - Returns error messages if data fetching or analysis fails (e.g., invalid symbol, API issues).


Backend Flowchart
User Inputs Stock Symbol
Frontend API Call
Backend Fetches Stock Data
Backend Processes Data (Prediction, Volatility, SMA, EMA)
Backend Sends Response (JSON)
Frontend Displays Results

BACKEND CODE:

- **Flask Setup:**
  - **Flask:** A lightweight framework to handle HTTP requests.
  - **CORS:** Allows the frontend to make requests to the backend hosted on a different domain.

```python
from flask import Flask, jsonify, request
import requests
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
```

```python
API_KEY = "YOUR_API_KEY"

@app.route('/stock-history', methods=['GET'])
def get_stock_history():
    stock_symbol = request.args.get('symbol')
    if not stock_symbol:
        return jsonify({"error": "Stock symbol is required"}), 400
```

- **Fetching Stock Data:**
  - **API Key:** Used to authenticate requests to Alpha Vantage.
  - **Route '/stock-history':** Fetches historical stock data for the requested symbol.
  - **Error Handling:** Returns an error if no stock symbol is provided.

BACKEND CODE:

- **Data Processing & Prediction:**
  - **Data Processing:** Extracts dates and prices from the API response.
  - **Reversing Data:** Ensures the data is in chronological order for analysis.

```
historical_data = data["Time Series (Daily)"]
dates = []
prices = []
for date, values in historical_data.items():
    dates.append(date)
    prices.append(float(values["4. close"]))
dates.reverse()
prices.reverse()
```

```
date_nums = np.arange(len(dates)).reshape(-1, 1)
prices_arr = np.array(prices).reshape(-1, 1)

model = LinearRegression()
model.fit(date_nums, prices_arr)

next_day = np.array([[len(dates)]])
predicted_price = model.predict(next_day)[0][0]
```

- **Linear Regression for Prediction:**
  - **Linear Regression:** Predicts the next day's stock price based on historical data.
  - **Prediction:** Uses the number of days as the independent variable and stock price as the dependent variable.

STOCK PRICE PREDICTION LOGIC:

Linear Regression Algorithm:

1. **Input:**
   - A set of data points (e.g., date index and stock prices).
2. **Goal:**
   - Find the line that best fits the data: $y = mx + c$, where:
     - m is the slope (rate of change).
     - c is the y-intercept (predicted value when $x = 0$).
3. **Steps:**
   1. **Find the average of X (dates) and Y (stock prices):**
      - Calculate the mean of all x values (dates) and y values (stock prices).
   2. **Calculate the Slope (m):**
      - $m = \frac{\sum(x_i - \text{mean}_x) \cdot (y_i - \text{mean}_y)}{\sum(x_i - \text{mean}_x)^2}$
      - This shows how much the stock price changes for each day.
   3. **Calculate the Intercept (c):**
      - $c = \text{mean}_y - m \cdot \text{mean}_x$
      - The intercept is where the line crosses the y-axis.
   4. **Predict Future Values:**
      - Use the formula $y = mx + c$ to predict future stock prices.
4. **Output:**
   - A model that can predict future stock prices based on the calculated line.

BACKEND CODE:

```
returns = np.diff(prices) / prices[:-1]
volatility = np.std(returns)

df = pd.DataFrame({"Date": dates, "Price": prices})
df["SMA_10"] = df["Price"].rolling(window=10).mean()
df["EMA_10"] = df["Price"].ewm(span=10, adjust=False).mean()
```

- **Volatility and Moving Averages:**
  - Volatility: Measures the standard deviation of daily returns to assess risk.
  - SMA and EMA: Moving averages help smooth out short-term fluctuations to identify trends.

FRONTEND OVERVIEW:

- **Frontend Technologies:**
  - **HTML:**
    - Provides the basic structure and layout of the page.
    - Contains input fields, buttons, and placeholders for displaying results (e.g., stock price, volatility).
  - **JavaScript:**
    - Responsible for making API calls to the backend.
    - Handles the dynamic display of stock data and chart rendering.
  - **Chart.js:**
    - A JavaScript library used for rendering interactive charts.
    - Used to display stock price trends, moving averages (SMA and EMA), and the predicted price.
- **Key Features:**
  - **User Input:**

    Input field for stock symbol and a button to trigger data analysis.
  - **API Interaction:**

    JavaScript fetches stock data from the backend.
  - **Display Data:**

    Shows current stock price, volatility, predicted price, and moving averages.
  - **Charts (Chart.js):**

    Line chart displays stock prices, SMA (10-day), and EMA (10-day).
  - **Error Handling:**

    Displays error messages for invalid input or connection issues.

FRONTEND CODE:

- **Fetching Data from Backend:**
  - **User Input:** Retrieves the stock symbol entered by the user.
  - **API Request:** Sends the stock symbol to the backend for analysis.
  - **Error Handling:** Displays error messages if the input is invalid or no data is returned.

```
function fetchStockHistory() {
    let stockSymbol = document.getElementById('stock-symbol').value.trim().toUpperCase();

    if (!stockSymbol) {
        document.getElementById('volatility').innerHTML = "<span class='error'>Please enter a stock symbol.</span>";
        return;
    }

    fetch(`https://aea48765-a8d0-4d9f-83b8-4d953d585e21-00-69c2x65msclv.pike.replit.dev/stock-history?symbol=${stockSymbol}`)
        .then(response => response.json())
        .then(data => { /* Handle response */ });
}
```

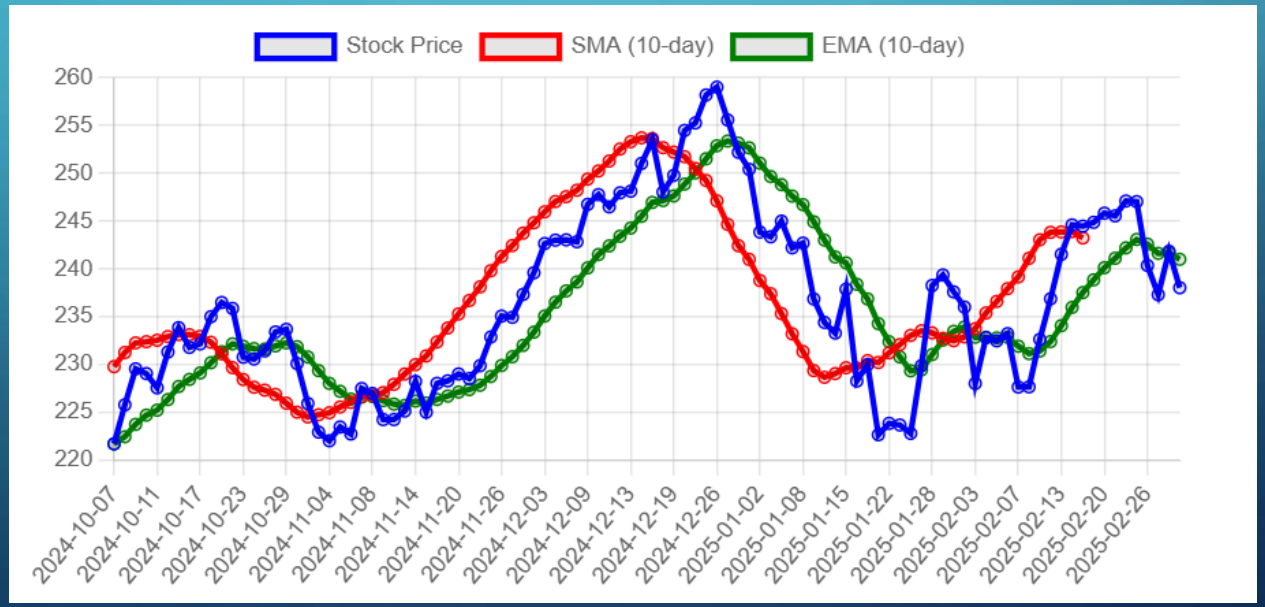
```
.then(data => {
    if (data.prices) {
        document.getElementById('volatility').innerHTML = `<b>Volatility:</b> ${data.volatility}`;
        document.getElementById('current-price').innerHTML = `<b>Current Price:</b> $${data.prices[data.prices.length - 1]}`;
        document.getElementById('predicted-price').innerHTML = `<b>Predicted Price:</b> $${data.predicted_price}`;

        let ctx = document.getElementById('stockChart').getContext('2d');
        new Chart(ctx, { /* Chart configuration */ });
    }
})
```

- **Displaying Data & Chart:**
  - **UI Updates:** Displays volatility, current price, and predicted price.
  - **Chart.js:** Plots the stock prices, SMA, and EMA for visualization.

INTERACTIVE CHART DISPLAY:

- **Stock Price Visualization:** Line chart displaying stock prices with dynamic updates.
- **SMA & EMA Indicators:**
  - **SMA (10-day):** Average stock price over the last 10 days.
  - **EMA (10-day):** Gives more weight to recent prices for trend detection.
- **Interactive:**
  - Hover to view specific values (price, SMA, EMA).
  - Zoom and pan for detailed analysis.
- **Customizable:**
  - Compare multiple datasets (e.g., stock price vs. indicators).
  - Toggle visibility of SMA/EMA from the chart legend.



STOCK VOLATILITY, SMA, AND EMA EXPLANATION:

- **Stock Volatility:**
  - Measures how much stock prices fluctuate over time.
  - Calculated as the **standard deviation of daily returns**.
  - Higher volatility = higher risk, lower volatility = more stable stock.
- **2. Simple Moving Average (SMA - 10 day):**
  - Average of the last **10 days' closing prices**.
  - Smooths out price fluctuations to identify trends.
- **3. Exponential Moving Average (EMA - 10 day):**
  - Similar to SMA but gives **more weight to recent prices**.
  - Reacts faster to new price changes, making it better for trend detection.

USE CASES & APPLICATIONS:

- **Stock Market Analysis:** Helps investors track stock trends using real-time and historical data.
- **Risk Assessment:** Volatility analysis aids in evaluating investment risks.
- **Trading Strategy Development:** SMA & EMA indicators help traders make informed decisions.
- **Machine Learning in Finance:** Predicts future stock prices using historical data and linear regression.
- **Educational Tool:** Useful for students and professionals learning about stock market trends and analytics.

FUTURE DEVELOPMENTS:

- More advanced predictive models (e.g., Neural Networks, ARIMA).
- Additional technical indicators (e.g., RSI, Bollinger Bands).
- User authentication & personalized tracking.
- Integration with multiple stock APIs for broader market analysis.
- Mobile-friendly design for better accessibility.

CONCLUSION:

This stock analysis application combines real-time data tracking, historical analysis, and predictive modeling to help users make informed investment decisions. By integrating **Flask (Python) for backend processing, machine learning (Linear Regression) for stock prediction**, and **Chart.js for interactive visualizations**, the platform provides a comprehensive tool for analyzing market trends.

With features like **volatility measurement, SMA & EMA indicators, and stock price forecasting**, the application serves traders, investors, and learners alike. Its **user-friendly interface** and **data-driven insights** make it a valuable resource for stock market analysis and financial decision-making.

THANK YOU