Research Report: Historical Price API + Graph Library

Hamid Mian

Objective:

- Find free public APIs that provide historical price data
    - What are the rate limits, how to handle throttling? How far back does the data span?
- Learn how to display this historical price data as a graph to display on the stocks page
    - What libraries can I use to graph the data in React?
    - How can I make the graph interactable/responsive?

Sources:

- Historical Price API: https://polygon.io/docs
- Graph Library: https://recharts.org

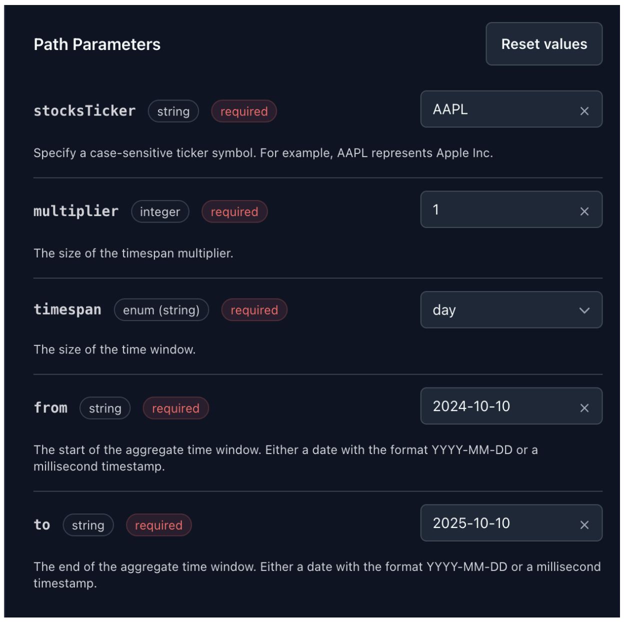
Time Spent:

- 30 minutes – locating a free api that provides historical price data and getting familiar with its various endpoints
- 30 minutes - locating the best library to graph our historical data in our React Frontend
- 90 minutes - writing report including summaries & notes on how to apply this knowledge

Summary - Historical Price API:

- **Polygon.io** provides an extensive amount of US stock data through its various free and paid api packages. You can create a free account to generate your own API key.
- The free plan limits users to **5 API calls per minute**, which is pretty restrictive. This could however be a potential option if we only used this API for pulling historical data, as many other companies have this data behind a paywall.
- For the free option, Polygon restricts us to historical data **up to 2 years back**.
- The endpoint that we want to use is: **Custom Bars (OHLC)**
    - */v2/aggs/ticker/{stocksTicker}/range/{multiplier}/{timespan}/{from}/{to}*

- Even though there are some heavy restrictions, this api is still a very good option to collect detailed historical data, particularly because you can specify exact date ranges and also specify the timespan of each window. This means you can get hour by hour or even minute by minute data for a stock, spanning from two years ago all the way until today – returned in a single response.
- **Note:** In the request the dates are in YYYY-MM-DD format, however in the responses they are in a Unix timestamp format – which we will likely need to decode in order to build the graph.

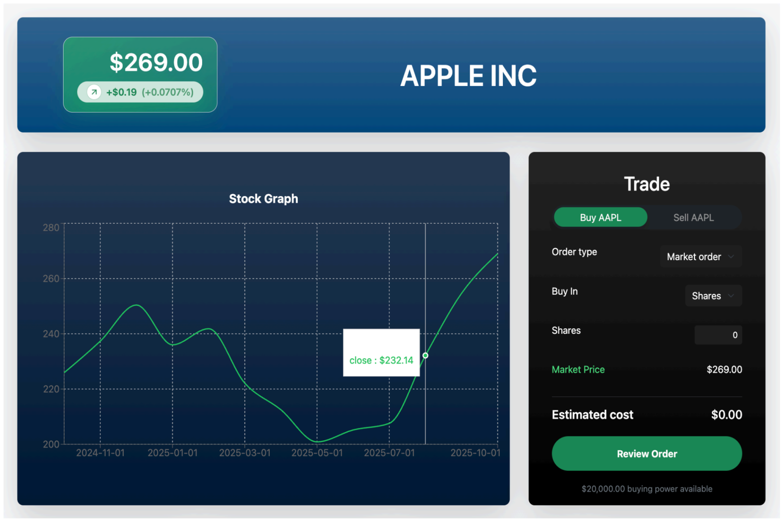


stocksTicker	string	required	AAPL
Specify a case-sensitive ticker symbol. For example, AAPL represents Apple Inc.			
multiplier	integer	required	1
The size of the timespan multiplier.			
timespan	enum (string)	required	day
The size of the time window.			
from	string	required	2024-10-10
The start of the aggregate time window. Either a date with the format YYYY-MM-DD or a millisecond timestamp.			
to	string	required	2025-10-10
The end of the aggregate time window. Either a date with the format YYYY-MM-DD or a millisecond timestamp.			

Summary - Graph Library:

- One of the main features for our app is the ability to research stocks and their historical price changes. The best way to showcase historical price data in a readable format is to provide the user of a graph on page load, with options to change the date range.

- **Recharts** is a charting library that allows developers to easily create **customizable** and **interactive** charts/graphs using provided data.
- Since Recharts is built exclusively for React, it is very easy to integrate it within your own app and get started by importing required components
- Install Dependencies: ***npm install recharts***
- Component Hierarchy:
    - **div**: outer div must specify height and width, or else error
        - **ResponsiveContainer**: makes chart resize to fit the div
            - **LineChart**: takes the data list as argument
                - **CartesianGrid**: builds the dotted grid shown in example
                - **XAxis**: required dataKey argument for the x-values
                - **YAxis**: specifies the axis domain (optional)
                - **Tooltip**: shows extra data details on hover
                - **Line**: draws the actual line based, uses XAxis key and requires another dataKey argument for the y-values



Order type	Market order
Buy In	Shares
Shares	0
Market Price	$269.00
Estimated cost	$0.00

$20,000.00 buying power available

Notes:

- We might want to add some functionality to change the date range that the graph shows:
    - 2 year - 1 year - year to date - 3 months - 1 months - 1 week - 1 day
    - When the user clicks a new option, the graph will make a new API request and get the appropriate data from the historical price API, and then the graph will rerender. In order to not make excessive requests to the API, we might want to be smart about when to call the API for new data and when to just slice data off of the original 2Y request. We will likely only need to call the API again if we want 1 month, 1 week, or today's data - because we would likely want to make the time windows smaller in order to increase the resolution of the graph.
- Later if we choose that we want our graphs to be dynamically updated to showcase real time incoming price changes of a stock during market hours, we will likely need to do additional research on how to rerender the graph based on new data coming through a websocket.