Homework Assignment: Backend Software Engineer

Estimated time: 2-3 hours

Task Overview

Your task is to design and implement a high-performance RESTful service capable of handling the rigorous demands of high-frequency trading systems. This service will act as a component in the company ABC trading infrastructure, managing and analysing financial data in near real-time.

Functional Requirements

Your service must support three HTTP-based API endpoints communicating via JSON:

1. **POST /add/**
    - **Purpose**: To capture and store trading data points identified by unique symbols.
    - **Input**:
        - `symbol`: A string identifier for the financial instrument (e.g., stock ticker).
        - `value`: A floating-point number representing the latest trading price.
    - **Response**: Confirmation that the data has been recorded, including any relevant status messages.
2. **POST /add_batch/**
    - **Purpose**: Allows the bulk addition of consecutive trading data points for a specific symbol.
    - **Input**:

    - `symbol`: String identifier for the financial instrument.
    - `values`: Array of floating-point numbers representing sequential trading prices.
  - **Response**: Confirmation of the batch data addition.
3. **GET /stats/**
  - **Purpose**: To provide rapid statistical analyses of recent trading data for specified symbols.
  - **Input**:
    - `symbol`: The financial instrument's identifier.
    - `k`: An integer from 1 to 7, specifying the number of **last 1e{k}** data points to analyze
  - **Response**:
    - `min`: Minimum price in the last 1e{k} points.
    - `max`: Maximum price in the last 1e{k} points.
    - `last`: Most recent trading price.
    - `avg`: Average price over the last 1e{k} points.
    - `var`: Variance of prices over the last 1e{k} points.

Technical Requirements

- We are looking for **single-node** implementation;
- **Limits:** There will be no more than 100 unique symbols;
- **Language & Framework**: You may use any backend programming language and framework you find suitable for near-real-time data processing and RESTful API implementation.
- **Concurrency & Performance**: The solution must efficiently handle a high volume of concurrent data entries and statistical requests;
  - 💡**No two concurrent add or/and get requests will occur simultaneously within a given symbol.**

- **Data Handling**: Implement an efficient data structure for real-time data insertion and retrieval of specified requests.

Submission Instructions

1. **Code**: Please submit all the source files.
2. **README.md**: Include a description of how to build and launch your service.

Evaluation Criteria

- It is ok to use code generation tools like Copilot or ChatGPT, etc.
- Code Quality
- Performance
- Documentation