Homework Assignment: Backend Software Engineer

Estimated time: 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.

Evaluation Criteria (ordered by importance)

- Performance & Algorithms
- Code Quality
- Documentation

Submission Instructions

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

Functional Requirements

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

1. **POST /add_batch/**
    - **Purpose**: Allows the bulk addition of consecutive trading data points for a specific symbol. (in-memory storage)
    - **Input**:
        - `symbol`: String identifier for the financial instrument.

        - `values`: Array of up to 10000 floating-point numbers representing sequential trading prices ordered from oldest to newest.
    - **Response**: Confirmation of the batch data addition.
2. **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 8, 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

- **Data Handling**: Implement an efficient data structure for real-time data insertion and retrieval of specified requests.
- We are looking for a **single-node, in-memory (no persistent storage)** implementation, assuming the server has enough RAM (but not infinite);
- **Limits:** There will be no more than 10 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.**
    - The time complexity for calculating stats should be better than O(n). O(n) complexity is insufficient for this task.
- It is ok to use code generation tools like Copilot or ChatGPT, etc.