Take-Home Assignment: Mini Risk Monitoring System

1. Assignment Overview

You will build a **mini risk monitoring application** that demonstrates your ability to create a **full-stack solution**—covering:

1. **Database Construction**
2. **Backend API** (including business logic for margin calculation)
3. **Frontend Dashboard** displaying client portfolio risk

You have **5 to 7 days** to complete this assignment. We will evaluate **not just whether it works, but how** you structure your code, maintain readability, and present your solution.

2. Core Requirements

2.1 Data Ingestion & Storage

1. **Fetch Real-Time Market Data**
    - Use a publicly available stock market API that provides real-time or near real-time price quotes (e.g., Twelve Data, IEX Cloud, Alpha Vantage with a real-time plan, etc.).
2. **Database Schema**
    - **Positions Table**: symbol, quantity, cost basis, client_id, etc.
    - **Market Data Table**: symbol, current_price, timestamp.
    - **Margin Table** : loan, margin requirement for each client.
3. **Persistence**
    - Store the incoming market data and the client's positions in your database (MySQL, PostgreSQL, MongoDB, etc.).
    - Ensure the schema can accommodate **multiple** positions, multiple clients, and reference **loan amounts** for each client.

2.2 Margin Calculation

You only need to implement **one** simplified margin requirement logic, as explained below:

1. **Key Terms**
    - $\boldsymbol{Qi}$: Quantity of stock i

- $\boldsymbol{Pi}$: Current market price of stock i
- **Loan Amount**: Total loan for the client's portfolio
- **Maintenance Margin Rate** (MMR): Use **25%** for all stocks
- n: Number of different stocks in the portfolio

2. **Portfolio Market Value**

$$Portfolio\ Market\ Value = \sum_{i=1}^{n} (Q_i \times P_i)$$

3. **Net Equity**

$$NetEquity = Portfolio\ Market\ Value - Loan\ Amount$$

4. **Total Margin Requirement**

$$Total\ Margin\ Requirement = Maintenance\ Margin\ Rate * Portfolio\ Market\ Value$$

With MMR = 25%,

$$Total\ Margin\ Requirement = 0.25 \times \sum_{i=1}^{n} (Q_i \times P_i)$$

5. **Margin Shortfall**

$$MarginShortfall = TotalMarginRequirement - NetEquity$$

- If Margin Shortfall≤0, **no margin call** is needed.
- If Margin Shortfall>0, a **margin call** should be triggered.

2.3 Backend API

Create an API (RESTful or GraphQL) that includes, at minimum:

1. **Retrieve Current Market Data**
   - E.g., GET /api/market-data
2. **Retrieve Client Positions**
   - E.g., GET /api/positions/:clientId
3. **Calculate Margin Status**

- E.g., GET /api/margin-status/:clientId
- Returns a response with portfolio value, loan amount, net equity, margin requirement, margin shortfall, and a boolean indicating if a margin call is triggered.

You can also add **update endpoints** (e.g., POST /api/positions) if you wish to demonstrate data creation or editing.

2.4 Frontend Dashboard

1. **Framework**
   - React, Vue, Angular, or any web stack of your choice.
2. **Display**
   - List of client positions (symbol, quantity, current price).
   - Calculated **Margin Status** (including shortfall or margin call).
3. **Interactivity**
   - At minimum, show real-time or near real-time updates of current prices and reflect margin status changes.
   - A simple table or basic UI is fine; advanced design is optional.
4. **Optional Extra**
   - Basic authentication, data visualizations, or alert notifications if margin call is triggered.

3. Documentation & Instructions

Your submission must include a **README** that contains:

1. **High-Level Architecture**
   - A brief description or diagram of how the database, backend, and frontend interact.
2. **Setup/Installation**
   - Clear steps on how to run the application locally (e.g., Docker commands, npm/yarn, database migrations, etc.).
3. **Tech Stack Explanation**

- Why you chose your database, backend framework, and frontend framework.

4. **Usage**
    - Instructions for accessing the dashboard and making API requests (via Postman/cURL, if relevant).
5. **Testing**
    - Show or explain how you tested your margin logic and APIs (e.g., sample screenshots, test files, unit tests).
6. **Known Limitations**
    - For example, if you mocked data or faced any constraints with third-party APIs.

4. Submission Instructions

1. **Project Repository**
    - Provide a link to your repo (GitHub, GitLab, Bitbucket).
    - If private, please share the necessary access with us.
2. **Demo (Optional)**
    - If you can, record a short video or provide screenshots that walk us through the functionality.
    - This is optional but can help illustrate how the system works end-to-end.

5. How You Will Be Assessed

1. **Completeness**
    - Database, API, and frontend all operational and meeting the core requirements.
2. **Code Quality & Organization**
    - Readable, maintainable code; sensible folder structure; use of version control.
3. **Database & API Design**
    - Logical schemas, consistent naming, clear endpoints.
4. **Margin Logic**

- Correct and consistent application of the margin formula.

5. **User Experience**
    - Frontend is easy to use; margin status is clearly displayed.
6. **Documentation**
    - Thorough README with instructions and explanations.

6. Example Use Case

- **Example**

Suppose a user has 2 stocks in the portfolio:

100 shares of Stock A @ $50 each

200 shares of Stock B @ $20 each

The user has taken a $3,000 loan.

Maintenance Margin Rate: 25%.

$$\boldsymbol{Portfolio\ Market\ Value} = (100 \times 50) + (200 \times 20) = 5000 + 4000 = \$9{,}000$$

$$\boldsymbol{Net\ Equity} = 9000 - 3000 = \$6{,}000$$

$$\boldsymbol{Total\ Margin\ Requirement} = 0.25 \times 9000 = \$2{,}250$$

$$\boldsymbol{Margin\ Shortfall} = 2250 - 6000 = -\$3{,}750$$

Since this is negative, there is **no margin shortfall** and no margin call.

- The user visits a simple dashboard to see their margin status. If a shortfall is detected, the UI displays a **margin call** alert or highlight in red.

Good Luck & Happy Coding!

We look forward to seeing your approach and the quality of your solution. If you have any clarifications, feel free to reach out!