Envint Global LLP - Hiring

Technical Round 2 Task(s) for Backend Skill Evaluation

Objective:

Evaluate the candidate's ability to design scalable, efficient, and logically complex backend systems. The assignment will assess their skills in **data processing, concurrency handling, API design, error handling, and performance optimization**.

Problem Statement:

Scenario:

You are developing a **Financial Risk Assessment API** that processes and analyzes **corporate financial statements** from various companies. The system must **ingest, filter, and structure** the data efficiently while handling **large-scale concurrent requests** and ensuring robustness.

Before candidates start implementing the solution, they should familiarize themselves with financial risk assessment concepts. The dataset below represents simplified corporate financial records used for risk analysis. Candidates should analyze key financial indicators and derive meaningful insights.

Financial data includes:

- **Company Identifiers** (company_id, name, reporting_period, industry_sector)
- **Financial Metrics** (total_assets, total_liabilities, revenue, net_profit, cash_flow)
- **Risk Indicators** (debt_to_equity_ratio, operating_margin, return_on_equity, interest_coverage_ratio, Altman Z-score, risk_score)


```
[
 {
  "company_id": "C12345",
  "company_name": "TechCorp Ltd.",
  "reporting_period": "2023-Q4",
  "industry_sector": "Technology",
  "total_assets": 5000000,
  "total_liabilities": 2000000,
  "revenue": 1500000,
  "net_profit": 300000,
```

```
    "debt_to_equity_ratio": 1.2,
    "cash_flow": 500000,
    "operating_margin": 15.0,
    "return_on_equity": 10.5,
    "interest_coverage_ratio": 3.5,
    "z_score": 2.8,
    "risk_score": 75
  },
  {
    "company_id": "C67890",
    "company_name": "RetailCo Inc.",
    "reporting_period": "2023-Q4",
    "industry_sector": "Retail",
    "total_assets": 8000000,
    "total_liabilities": 6000000,
    "revenue": 2500000,
    "net_profit": 500000,
    "debt_to_equity_ratio": 2.5,
    "cash_flow": 700000,
    "operating_margin": 10.2,
    "return_on_equity": 7.8,
    "interest_coverage_ratio": 1.8,
    "z_score": 1.5,
    "risk_score": 50
  }
]
```

Task Requirements:

1. Database & Data Ingestion:

- Design a **DynamoDB (or No SQL-based)** schema for storing financial data.
- Implement an API (`/uploadFinancialData`) to **ingest JSON data** from different corporate filings. The API should:
    - Accept batch uploads of up to **500 records per request**.
    - Prevent **duplicate entries** by checking a unique `company_id` and `reporting_period`.
    - Process uploads asynchronously using a **queue (e.g., AWS SQS, RabbitMQ, or in-memory queue)**.
    - Return a response with **successful and failed records**.

2. Data Processing & Risk Analysis:

- Implement an API (`/getRiskAssessment`) to **retrieve financial risk scores** based on the following filters:
    - Filter by `company_id`, `reporting_period`, and `industry_sector`.
    - Compute a **risk score** based on predefined formulas considering **debt-to-equity ratio, revenue growth, and cash flow stability**.
    - Paginate results for efficient querying (e.g., **limit = 50 per request**).

Debt-to-Equity Ratio: $\frac{\text{Total Liabilities}}{\text{Total Assets - Total Liabilities}}$

Operating Margin: $\frac{\text{Net Profit}}{\text{Revenue}} \times 100$

Return on Equity: $\frac{\text{Net Profit}}{\text{Total Assets - Total Liabilities}} \times 100$

Interest Coverage Ratio: (Use provided value; requires interest expense data)

Altman Z-Score: (Simplified) $3.3\frac{\text{Net Profit}}{\text{Total Assets}} + 0.6\frac{\text{Total Assets - Total Liabilities}}{\text{Total Liabilities}} + 1.0\frac{\text{Revenue}}{\text{Total Assets}}$

3. Concurrency Handling & Retry Mechanism:

- Ensure multiple users can fetch records without **race conditions**.
- Implement a **retry mechanism** for failed API calls due to temporary issues.
- Use **locking or transactions** to prevent duplicate processing of the same data in concurrent requests.

4. Performance Optimization:

- Optimize database queries to **reduce response time under 200ms**.
- Use **caching mechanisms** (e.g., Redis) to store frequent queries.
- Implement **background processing** for complex risk calculations.

5. Security & API Best Practices:

- Implement **JWT-based authentication**.
- Secure endpoints against **SQL injection, XSS, and request overload**.
- Implement **rate-limiting** (e.g., max **100 requests per minute per user**).

Deliverables:

1. **Backend Code** (Node.js with database setup)
2. **Postman Collection** (for testing APIs)
3. **README File** (setup instructions + design decisions)
4. **Short Report (Max 1 page)** – Describe:
   Architecture & Flow, Optimization Strategies, Challenges Faced & Solutions

Evaluation Criteria:

✅ **Code Quality & Readability** – Is it modular, well-structured, and maintainable?
✅ **Database Design & Efficiency** – How well is data structured for scalability?
✅ **Concurrency & Error Handling** – Is the system robust under high load?
✅ **API Performance & Security** – Are best practices followed?
✅ **Problem-Solving Approach** – Does the candidate think critically about optimizations?

Bonus (Not Mandatory but Preferred):

- **Unit Tests** to validate API functionality.

Deliverables:

1. A fully functional Backend application.
2. GitHub Repository: Share the GitHub link of the project after completion.

This task requires a strong understanding of backend technologies, with a focus on delivering a secure and efficient application.

The deadline for this assignment is 11:59 PM, Thursday 13th March 2025.