Full Stack Developer Coding Challenge – Livello India Pvt Ltd

Objective

Develop a **scalable, real-time IoT dashboard** that integrates backend services with a dynamic frontend. This test assesses **backend architecture, frontend development, real-time data handling, and problem-solving skills**.

Challenge Overview

Build a **real-time IoT monitoring dashboard** that:

- **Processes and stores sensor data** from an API (simulating IoT devices).
- **Displays real-time updates** on a web-based UI.
- **Provides historical data visualization**.
- **Includes user authentication & role-based access**.

Use **Node.js (Nest.js preferred) for the backend**, **React.js (or Next.js) for the frontend**, and **MongoDB** for data storage.

Challenge Requirements

1. Backend (Node.js, Nest.js preferred)

- Develop an **API to ingest and store sensor data**. Use a mock script to generate real-time IoT data (temperature, humidity, power usage).
- Implement a **WebSocket-based real-time update mechanism**.
- Provide **RESTful & GraphQL APIs** for frontend consumption.
- Implement **role-based authentication (JWT/OAuth)**:
  - Admin: Full access
  - User: Read-only access to data
- Write a **Dockerfile** so the backend can be easily containerized and run locally.
- Implement **unit tests** with Jest.

2. Frontend (React.js / Next.js preferred)

- Fetch & display real-time sensor data using WebSockets/API polling.
- Implement **interactive charts** for historical data visualization using Recharts/D3.js.
- Build a **responsive dashboard UI** with a dark/light mode toggle.
- Secure pages based on user roles.
- Optimize **performance & bundle size**.

3. Deployment (Dockerized Local Setup)

- Write **Dockerfiles** for both the backend and frontend so the entire system can be run locally with a single command.
- Ensure **documentation** for setup and execution.

Bonus Challenges (Extra Points)

1. IoT Device Simulation

- Use **MQTT** to simulate real-time IoT sensor data ingestion.

2. Cloud & DevOps

- Deploy the application using **AWS/GCP/Azure (Docker/Kubernetes preferred)**.
- Implement **logging & monitoring** (e.g., Prometheus, ELK, or CloudWatch).
- Use **CI/CD pipelines (GitLab/GitHub Actions)** for deployment automation.

3. Infrastructure as Code

- Deploy using **Terraform** or CloudFormation.

Evaluation Criteria

Category	Weight(%)
Backend Architecture & API Design	30%
Frontend UI/UX & Performance	20%
Containerization & Local Setup	25%
Security, Testing & Best Practices	15%
Bonus Challenges & Creativity	10%

Submission Guidelines

- Submit a **GitHub repository** with setup instructions.
- Provide a **Dockerized solution that runs locally**.
- Submit a **2-minute video** explaining your design decisions.

Deadline: 5 days from receiving the challenge.