IoT Sensor Data Processing & Management System - Project Guide

This project is an IoT-based system that collects sensor data (Temperature, Humidity, Pressure), processes it, and stores it in a database using a microservices architecture.

1. Project Architecture

The project consists of two main microservices running in Docker containers: • Microservice A: Simulates or collects sensor data, converts it into JSON format, and sends it to Microservice B via HTTP POST. • Microservice B: Receives JSON data, processes it, stores it in a MySQL database, and sends back a success or failure response.

2. Workflow

1. Microservice A listens to simulated temperature, humidity, and pressure sensor readings. 2. Each reading is converted into a JSON object with fields like SensorType, SensorValue, Timestamp, etc. 3. Microservice A sends the JSON data to Microservice B via HTTP POST. 4. Microservice B receives the request, parses the JSON, validates it, and stores it in the MySQL database. 5. Microservice B responds with 'Good' if the data is valid or 'Bad' if not. 6. All components are containerized with Docker for isolated execution.

3. Database

• Database: MySQL • Table stores: SensorType, SensorValue, ID1, ID2, Timestamp • Microservice B is responsible for DB connection and insertion of sensor data.

4. Code Overview

• Microservice A: - Generates random sensor data (for simulation). - Uses HTTP POST to send data. • Microservice B: - Connects to MySQL using Go's database/sql package. - Receives and parses JSON requests. - Inserts valid data into the database. - Sends back HTTP status code and message.

5. Deployment

• Docker is used to run each microservice in isolated containers. • docker-compose.yml is used to start Microservice A, Microservice B, and MySQL together. • This ensures easy setup and networking between services.

6. Conclusion

This project demonstrates the use of microservices in IoT data processing. It covers generating or collecting sensor data, sending it via HTTP, processing it in another service, and storing it in a database — all running in Docker containers.