IoT Sensor Backend Service (Node.js + MongoDB + MQTT)

📌 Project Overview

This project is a Node.js backend service built as part of a pre-assessment assignment.
It ingests IoT sensor temperature readings, stores them in MongoDB Atlas, and exposes REST APIs to retrieve the latest reading for a given device.

Additionally, the project implements an **MQTT subscriber** to consume real-time sensor data published on MQTT topics and persist it automatically to the database.

Project Demo

Watch the demo video on Drive

🚀 Features

- REST API to ingest sensor temperature data
- REST API to fetch the latest reading per device
- MongoDB Atlas integration using Mongoose
- Input validation for required fields
- Automatic timestamp handling
- MQTT subscriber for real-time IoT ingestion (Bonus Task)
- Clean and modular project structure

🛠 Tech Stack

- **Node.js** (v18+)
- **Express.js**
- **MongoDB Atlas**
- **Mongoose**
- **MQTT.js**
- **MQTTX (for testing MQTT)**
- **Postman (for API testing)**

📂 Project Structure

```
iot-backend/
│
├── src/
│   ├── models/
│   │   └── SensorReading.js
│  ├── routes/
│  │  └── sensorRoutes.js
│  ├── mqttSubscriber.js
│  └── app.js
│
├── server.js
├── .env
├── package.json
├── README.md
└── node_modules/
```

IoT Backend System

⚙️ Setup Instructions

1️⃣ Clone the Repository

```
git clone <your-github-repo-url>
cd iot-backend
```

2️⃣ Install Dependencies

```
npm install
```

3️⃣ Create MongoDB Atlas Cluster

1. Go to https://www.mongodb.com/atlas
2. Create a Free Shared Cluster
3. Create a database user
4. Allow network access (0.0.0.0/0)
5. Copy the MongoDB connection string

4️⃣ Configure Environment Variables

Create a .env file in the root directory:

```
PORT=5000
MONGO_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/iotDB
```

5️⃣ Start the Server

```
npm run dev
```

Expected output:

```
Server running on port 5000
MongoDB Connected
MQTT
```

🔌 REST API Endpoints

Ingest Sensor Data

POST `/api/sensor/ingest`

Request Body:

```
{
  "deviceId": "sensor-01",
  "temperature": 29.3,
  "timestamp": 1705312440000
}
```

deviceId and temperature are required

timestamp is optional (defaults to current time)

Success Response:

```
{
  "message": "Sensor data saved successfully",
  "data": {
    "deviceId": "sensor-01",
    "temperature": 29.3,
    "timestamp": 1705312440000
  }
}
```

Get Latest Reading for a Device

GET `/api/sensor/:deviceId/latest`

Example:

```
GET /api/sensor/sensor-01/latest
```

Response:

```
{
  "deviceId": "sensor-01",
  "temperature": 29.3,
  "timestamp": 1705312440000,
  "createdAt": "2026-01-28T10:30:00.000Z"
}
```

📡 MQTT Integration (Bonus Task)

🔹 Broker Configuration

Host: broker.hivemq.com

Port: 1883

Protocol: MQTT (TCP)

🔹 Subscribed Topic Pattern

```
iot/sensor/+/temperature
```

Example topic:

```
iot/sensor/sensor-01/temperature
```

🔹 MQTT Payload Format

Example payload:

```
38.4
```

🔹 MQTT Flow

1. Sensor publishes temperature to MQTT topic
2. Node.js subscribes to the topic
3. Device ID is extracted from topic

4. Temperature is saved to MongoDB automatically

🔹 MQTT Testing (Using MQTTX)

1. Download MQTTX: https://mqttx.app/

2. Create a new connection:

   Host: `broker.hivemq.com`

   Port: `1883`

   SSL: `OFF`

3. Publish:

   Topic: `iot/sensor/sensor-01/temperature`

   Payload: `38.4`

Terminal output:

```
MQTT Message received
Data saved to MongoDB
```

🧪 Testing Tools

Postman – REST API testing

MQTTX – MQTT publish testing

MongoDB Atlas UI – Data verification

🧠 Key Design Decisions

Used Express.js for simplicity and scalability

Used Mongoose for schema enforcement

MQTT subscriber runs as a background service

Topic wildcard (+) allows handling multiple devices

Timestamp defaults handled at server level

📈 Evaluation Coverage

✔ API correctness

✔ MongoDB Atlas integration

✔ Input validation

✔ Proper REST design

✔ MQTT bonus implementation

✔ Clean code structure

✔ Documentation