Akıllı Sensör Takip Sistemi

📌 Genel Bakış

Bu proje, bir fabrikadaki IoT sensörlerinden MQTT protokolü ile verileri toplayan, bu verileri gerçek zamanlı yayınlayan, hata yönetimi ve logging mekanizması sunan bir backend servisini kapsar. Ayrıca, kullanıcıların logları ne sıklıkla gördüklerini takip eden bir sistem de dahil edilmelidir.

🏗️ Teknoloji Stack

- **Backend:** Node.js (NestJS)
- **Veri Tabanı:** PostgreSQL, InfluxDB
- **Gerçek Zamanlı Veri:** MQTT (paho-mqtt) veya WebSockets
- **Kimlik Doğrulama:** JWT veya API Key
- **Logging:** Structured JSON Logging (basit dosya logging veya alternatif log yaklaşımı)
- **MQTT Broker:** Eclipse Mosquitto veya alternatifi
- **Frontend (Opsiyonel):** Jinja2, React veya Vue.js
- **Containerization & Deployment:** Docker

🏗️ Mimari

- **Kullanıcı Yönetimi:** sistemde 3 tip kullanıcı bulunur.
    - **System Admin:** Genel entegrasyonu yönetecek olan kullanıcı rolüdür **(God Mode)**. Şirket ve müşteri kaydı oluşturabilir, kullanıcılar ekleyebilir ve tüm rolleri güncelleyebilir. IoT entegrasyonlarını yapabilir ve tüm loglara erişim yetkisine sahiptir. System Admin kullanıcısı, diğer kullanıcılar tarafından görünemez.
    - **Company Admin:** Şirketine kullanıcılar ekleyebilir, entegre edilen IoT cihazlardan gelen verileri görüntüleyebilir ve kullanıcı davranışlarını inceleyebilir. Ayrıca, kullanıcı bazlı cihaz görüntüleme yetkileri atayabilir.
    - **User:** Yetkisi olduğu IoT verilerini görüntüleyebilir.
- **IoT Sensörleri** MQTT ile veri gönderir.
- **Backend API, MQTT broker'dan gelen verileri işler ve saklar.**
- **Gerçek zamanlı veri yayını** için WebSockets veya MQTT kullanılır.
- **Kullanıcı davranışları** loglanır (kim, ne zaman log sayfasına baktı?)

- **Loglar JSON formatında kaydedilir ve analiz edilir.**

MQTT Veri Akışı

1. Sensörler belirli aralıklarla MQTT broker'a aşağıdaki formatta veri yollar:

```
{
    "sensor_id": "temp_sensor_01",
    "timestamp": 1710772800,
    "temperature": 25.4,
    "humidity": 55.2
}
```

2. Backend API, MQTT broker'a subscribe olur ve gelen veriyi işler.
3. Hatalı veri tespit edilirse loglanır.
4. Veri uygun formatta veritabanına kaydedilir.
5. WebSocket veya MQTT üzerinden istemcilere yayınlanır.

Güvenlik Gereksinimleri

- API'ler JWT veya API Key ile korunmalı.
- MQTT broker TLS/SSL ile korunmalı.
- Rate limiting uygulanmalı (DDoS koruması)
- Loglar yetkisiz erişime karşı korunmalı.

Kullanıcı Log Takibi

Kullanıcıların log sayfasını ne sıklıkla ziyaret ettiğini takip etmek için:

- Kullanıcı her log sayfasını ziyaret ettiğinde bir kayıt oluştur.
- Kayıt formatı:

```
{
    "user_id": "user_123",
    "timestamp": 1710772800,
    "action": "viewed_logs"
}
```

- Kullanıcı davranışları analiz edilerek hangi zaman dilimlerinde yoğunluk olduğu tespit edilir.
- Kullanıcı davranışları analiz edilerek hangi zaman dilimlerinde yoğunluk olduğu tespit edilir.

🎯 Bonus

- Python Kullanımı: Projeyi Python ile tekrar yazın.
- **Grafiksel Dashboard**: IoT verilerini görselleştirmek için bir frontend (React/Vue) ekleyin.
- **Log Analitikleri**: Kullanıcı davranışlarını analiz edip, öngörücü raporlar oluşturun. (Basit bir logic olabilir veya ML yaklaşımları)
- **Backend API Testleri**: Jest (Node.js), Pytest (Python) kullanarak API endpointlerini test edin.
- **MQTT Bağlantı Testleri**: MQTT.fx veya Mosquitto CLI ile mesaj akışını doğrulayın.
- **CI/CD**: GitHub Actions ile otomatik deploy mekanizması oluşturun.
- **Gerçek Zamanlı Veri Testleri**: WebSocket üzerinden gelen verileri Postman veya WebSocket UI ile kontrol edin.
- **Containerization**: Backend servisini Docker ile paketleyin.
- **Tüm süreçlerin e2e olarak bir sunucu da deploy edilmesi.**

☑ Beklenen Teslimatlar

1. **Public bir Github repo da kaynak kodun paylaşılması**
2. **MQTT veri alım ve yayın servisi**
3. **Gerçek zamanlı veri yayın mekanizması**
4. **Logging ve hata yönetimi mekanizması**
5. **Kullanıcı yönetim mekanizması**
6. **Kullanıcı log takip mekanizması**
7. **Doküman: API endpointleri, mimari tasarım ve deployment rehberi**