Real-Time Wind Turbine Health Monitoring System

1. Introduction

This document provides a complete technical, architectural, and operational explanation of the Real-Time Wind Turbine Health Monitoring System. It is intended for developers, engineers, architects, and DevOps teams as a reference guide for understanding, maintaining, and extending the application.

2. Business Purpose

Wind turbines generate telemetry every 10 seconds. Monitoring over 2,200 turbines manually is impossible. The system automates health monitoring, anomaly detection, and predictive maintenance using real-time and aggregated telemetry data.

3. Key Features

- Real-time turbine health dashboard
- High-frequency telemetry ingestion
- Hourly aggregation using parallel processing
- Rule■based anomaly detection
- Region/farm■based performance analytics
- Predictive maintenance indicators
- Containerized deployment (Docker)
- REST API architecture

4. User Roles & User Stories

- Operations Engineer: Live turbine status
- Plant Supervisor: Region/farm filtering
- Analyst: Historical performance trends
- Maintenance Planner: Daily generation metrics
- Backend Engineer: Hourly data aggregation
- Platform Engineer: Parallel processing
- DevOps Engineer: Deploy using Docker

5. System Architecture

The system consists of:
1. Frontend Application (Angular/React)
2. REST API Layer (Spring Boot)
3. Service Layer (Business Logic)
4. Data Processing Layer (Schedulers & Aggregation)
5. Persistence Layer (MySQL)

6. Deployment Layer (Docker + CI/CD)

6. Database Design

Tables:
- turbines: Metadata of each turbine
- telemetry_raw: 10■second telemetry data
- telemetry_hourly: Hourly aggregated data

7. Backend Components

Controllers:
• /api/turbines – list turbines
• /api/telemetry/latest/{id} – latest turbine status
• /api/process/aggregate/{id} – manual hourly aggregation

Services:
• TurbineService – handles turbine metadata
• TelemetryService – real■time telemetry
• AggregationService – hourly aggregation logic

8. Telemetry Data Flow

1. Telemetry arrives into telemetry_raw table every 10 seconds
2. Latest value served to UI immediately
3. Scheduler aggregates hourly data
4. Anomaly detection applied
5. Aggregated data stored into telemetry_hourly

9. Anomaly Detection Logic

This system currently uses rule■based anomaly detection:
• If avgPower < 30 → anomaly = 1
• Else → anomaly = 0
This can be extended to ML-based prediction in future versions.

10. Scheduler & Parallel Processing

The hourly aggregation uses Spring Scheduler:
CRON: 0 0 * * * * (runs every hour)
Parallel stream processes all turbines simultaneously for scalability.

11. Technology Stack

Backend:
- Java 21
- Spring Boot 3+
- JPA/Hibernate
Frontend:
- Angular / React
Database:
- MySQL
DevOps:
- Docker, Docker Compose

12. Deployment Guide

1. Build backend using mvn clean install
2. Package Docker image
3. Bring up containers using docker-compose
4. Access backend via http://localhost:8000
5. Access Swagger UI at /swagger-ui.html

13. Conclusion

The Real-Time Wind Turbine Health Monitoring System is a scalable IoT solution designed for large-scale wind energy operations. It offers fault detection, predictive maintenance insights, real-time dashboards, and industrial-grade performance.