Project Documentation: Real-Time Data Processing System for Weather Monitoring with Rollups and Aggregates

# 1. Project Overview

The Real-Time Data Processing System is designed to monitor weather conditions using real-time data from the OpenWeatherMap API. It processes the retrieved data to provide summarized insights through rollups and aggregates, including daily weather summaries, temperature conversions, alerting thresholds, and visualizations.

The backend is built using Spring Boot for processing the weather data, PostgreSQL for storing daily weather summaries, and the frontend uses React with Vite to display visualizations.

# 2. System Components

• Backend: Spring Boot for RESTful APIs.

• Database: PostgreSQL for storing daily summaries and weather data.

• Frontend: React + Vite for rendering visualizations and user interactions.

• Testing: Postman for API testing.

# 3. Setup and Installation Guide

## Prerequisites

1. Java 17 and above.

2. PostgreSQL (any version) installed and running.

3. Node.js (for frontend - version 16 and above).

4. Postman for API testing (optional but recommended).

5. An OpenWeatherMap API key (Sign up at https://openweathermap.org/).

## Backend Setup (Spring Boot)

1. Clone the repository:

git clone <repository-url>

cd <project-directory>/backend

2. Update application.properties with the PostgreSQL database configuration and OpenWeatherMap API key:

spring.datasource.url=jdbc:postgresql://localhost:5432/weather\_db  
spring.datasource.username=your\_db\_username  
spring.datasource.password=your\_db\_password  
  
openweathermap.api.key=your\_api\_key

3. Build and run the Spring Boot application:

./mvnw clean install  
./mvnw spring-boot:run

## Frontend Setup (React + Vite)

1. Go to the frontend directory:

cd <project-directory>/frontend

2. Install the dependencies:

npm install

3. Run the development server:

npm run dev

# Testing the Application

Use Postman to test the API endpoints. Ensure the backend is running and connected to PostgreSQL.

You can test the API by sending requests to the weather monitoring endpoints.

# Java Spring Boot Functions

1. WeatherDataController: Handles API requests and retrieves weather data from the OpenWeatherMap API.  
2. WeatherDataService: Contains the logic for processing, converting temperatures, and handling rollups/aggregates.  
3. AlertService: Monitors thresholds and triggers alerts if temperature or other weather conditions breach the defined limits.

# Conclusion

This project demonstrates the implementation of a real-time weather monitoring system using Spring Boot, PostgreSQL, and React. By utilizing OpenWeatherMap API data, the system efficiently processes and stores weather data, generates rollups and aggregates, and triggers alerts based on configurable thresholds. The frontend offers a visualization of weather summaries and trends.