Assignment: Universal Weather Tracker with Alerts

Objective

Build a Weather Tracking Application with the following features:

Real-time weather fetching.

Historical data tracking.

Email alerts for extreme weather conditions.

(Optional) Frontend UI for user interaction.

Core Features (For Backend Developers)

1. RESTful API

GET /weather/{city}

Fetch real-time weather for a given city.

Response (JSON format):

City: String

Temperature: Double (°C)

Weather Condition: String (e.g., Clear, Rainy)

Humidity: Integer (%)

Wind Speed: Double (km/h)

Timestamp: DateTime (when the data was fetched).

GET /weather/history/{city}

Fetch the last 5 requests for the given city from the database.

Response (JSON format):

List of weather records containing:

City.

Temperature.

Weather condition.

Timestamp.

2. Database

Store weather data for every /weather/{city} request.

Schema:

City: String

Temperature: Double

Weather Condition: String

Humidity: Integer

Wind Speed: Double

Timestamp: DateTime

3. Alerts

If the temperature exceeds 40°C or drops below 0°C, send an email alert to a predefined email address.

Alert Email:

Subject: “Extreme Weather Alert for [City]!”

Body: Include the temperature, weather condition, and city name.

4. External API Integration

Fetch weather data from a public weather API like OpenWeatherMap or WeatherAPI.

Use server-side HTTP requests (Axios for JavaScript, requests for Python, WebClient for Java, etc.).

5. Error Handling

404: If the city is invalid or not found.

503: If the weather API is down or inaccessible.

6. Logging

Log the following:

Incoming API requests.

External API responses.

Errors.

Email notifications.

Optional Full-Stack Features (For MERN/React/Full-Stack Developers)

Frontend

Build a simple UI for users to:

Enter a city and view current weather.

View historical weather data for the city.

Display an alert banner if the city has extreme weather conditions.

UI Design

Use React or any frontend framework to create the following components:

Search Bar: Input city name.

Weather Display: Show current weather data.

History Table: Show last 5 historical weather requests.

Alert Notification: Banner for extreme weather alerts.

Stack-Specific Guidance

MERN Stack (MongoDB, Express, React, Node.js)

Backend:

Use Node.js and Express to create the API.

Use MongoDB for storing historical weather data.

Integrate Nodemailer for sending email alerts.

Frontend:

Use React with Axios for consuming the API.

Display weather data in a user-friendly interface.

Python (Flask/Django)

Backend:

Use Flask or Django for the API.

Store weather data in SQLite or PostgreSQL.

Use the smtplib or Django Email for sending email alerts.

Frontend (Optional):

Use Flask templates or integrate with a React frontend.

Java (Spring Boot)

Backend:

Use Spring Boot to develop the API.

Use H2 or MySQL for storing historical data.

Use Spring Mail for sending email alerts.

Frontend (Optional):

Use Thymeleaf templates or a React frontend for the UI.

JavaScript (React + Node.js)

Backend:

Create a Node.js + Express API.

Use a lightweight database like SQLite or PostgreSQL.

Send email alerts using Nodemailer.

Frontend:

Build a React app to consume the API.

Bonus Features (Optional for All Stacks)

JWT Authentication:

Protect the API with token-based authentication.

Only authenticated users can fetch data or view history.

Rate Limiting:

Prevent abuse by limiting API calls (e.g., max 5 requests per minute per IP).

Swagger/OpenAPI: