**Weather Monitoring Project Report**

**1. Project Overview**

* **Title**: Weather Monitoring Application
* **Objective**: To develop a weather monitoring application that retrieves real-time weather data from the OpenWeatherMap API, processes it, stores it in a MySQL database, and provides alerts for specific weather conditions.

**2. Project Description**

* The application fetches weather data for multiple cities, computes daily summaries (average, max, and min temperatures, and humidity), and generates alerts based on predefined criteria (e.g., severe weather conditions).
* It utilizes a structured database schema to store weather data and alerts efficiently.

**3. Technologies Used**

* **Programming Language**: Python
* **Database**: MySQL
* **Libraries**:
  + mysql-connector-python for database connectivity.
  + datetime for date handling.

**4. Database Design**

* **Tables Created**:
  + weather\_data: Stores real-time weather data for various cities.
  + daily\_summary: Contains daily summaries of weather data (average temperature, max/min temperature, dominant weather condition).
  + alerts: Records alerts generated based on specific weather conditions.
* **Schema Example**:

sql

Copy code

CREATE TABLE IF NOT EXISTS weather\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

city VARCHAR(100),

timestamp DATETIME,

temp FLOAT,

humidity FLOAT,

weather\_condition VARCHAR(100)

);

CREATE TABLE IF NOT EXISTS daily\_summary (

id INT AUTO\_INCREMENT PRIMARY KEY,

city VARCHAR(100),

date DATE,

avg\_temp FLOAT,

max\_temp FLOAT,

min\_temp FLOAT,

dominant\_condition VARCHAR(100),

UNIQUE KEY (city, date)

);

CREATE TABLE IF NOT EXISTS alerts (

alert\_id INT AUTO\_INCREMENT PRIMARY KEY,

alert\_type VARCHAR(100),

city VARCHAR(100),

message TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**5. Key Features**

* **Data Retrieval**: Uses the OpenWeatherMap API to fetch real-time weather data.
* **Data Processing**: Calculates daily summaries and manages weather alerts.
* **Alert System**: Provides notifications for severe weather conditions based on the data received.

**6. Implementation**

* **Code Snippets**:
  + **Daily Summary Calculation**:

python

Copy code

def calculate\_daily\_summary(city\_name):

# Code to fetch weather data, calculate averages, and insert into daily\_summary

* + **Alert Insertion**:

python

Copy code

def insert\_alert(alert\_type, city, message):

# Code to insert alerts into the alerts table

**7. Testing**

* Describe any testing methodology used (e.g., unit testing for functions, integration testing for the entire application).

**8. Challenges Faced**

* Issues with data retrieval from the API due to rate limits.
* Handling null values in database fields during calculations.

**9. Conclusion**

* The weather monitoring application successfully meets the project requirements by providing accurate weather data processing and alerting mechanisms. Future improvements could include a user interface and additional features such as historical data analysis.

**10. References**

* OpenWeatherMap API documentation.
* MySQL documentation for database operations.