

Real-Time Weather Monitoring System

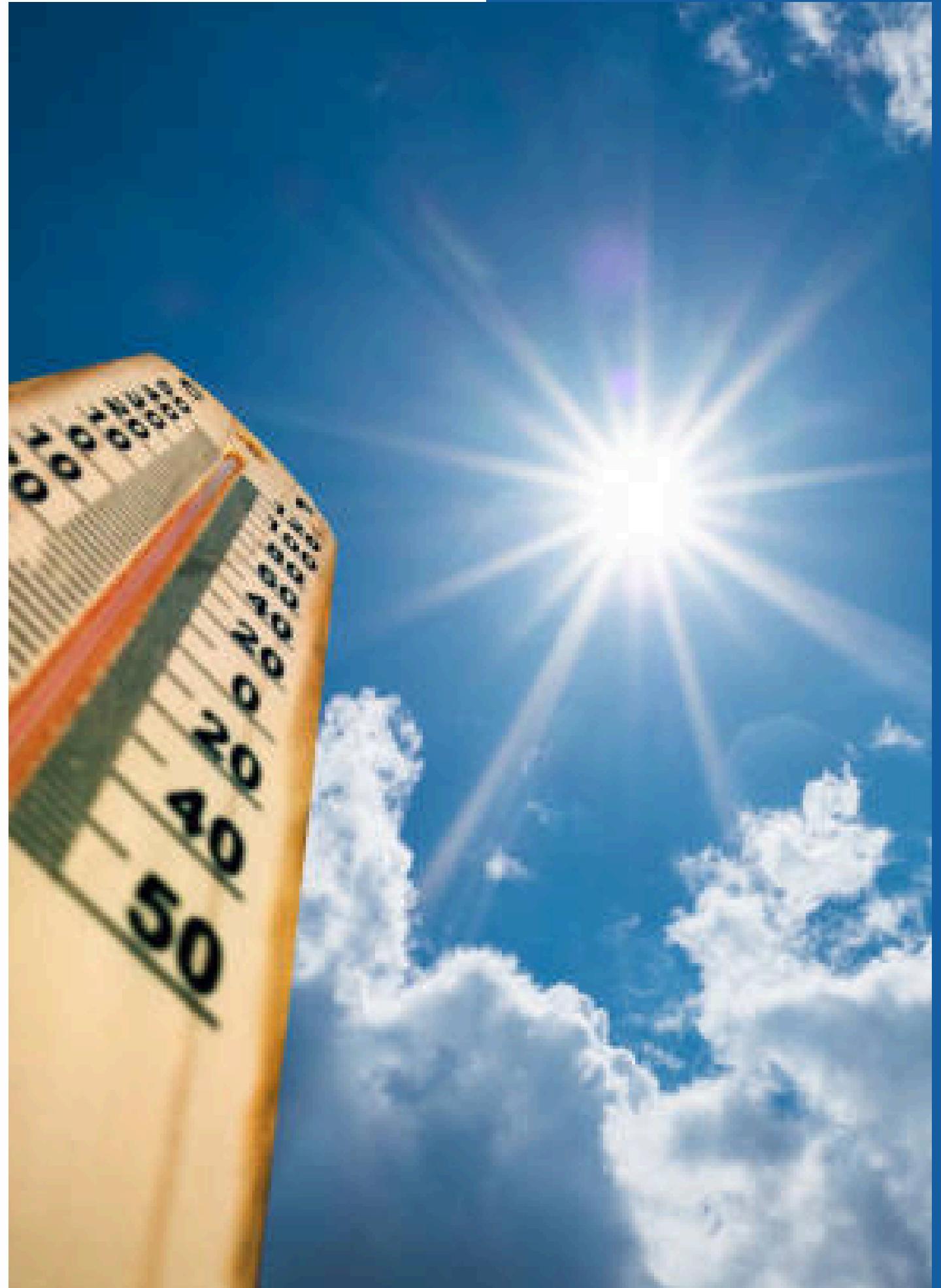
Gyeongwon Song

Student ID: 22297



Overview

▶ Introduction	03
▶ Application Domain	04
▶ System Architecture	05
▶ Data Source	06
▶ Data Processing Workflow	07
▶ User Interface & Demo	10
▶ Challenges and Solutions	12
▶ Conclusion	14
▶ Q&A	15



Introduction

This project builds a **real-time weather monitoring system** using **Kafka, Spark Streaming, MySQL, and Streamlit**.

It demonstrates expertise in data pipelines, real-time processing, and UI development. The goal is to provide a **scalable** solution for collecting and visualizing live weather data from multiple cities in an **interactive format**.

Application Domain

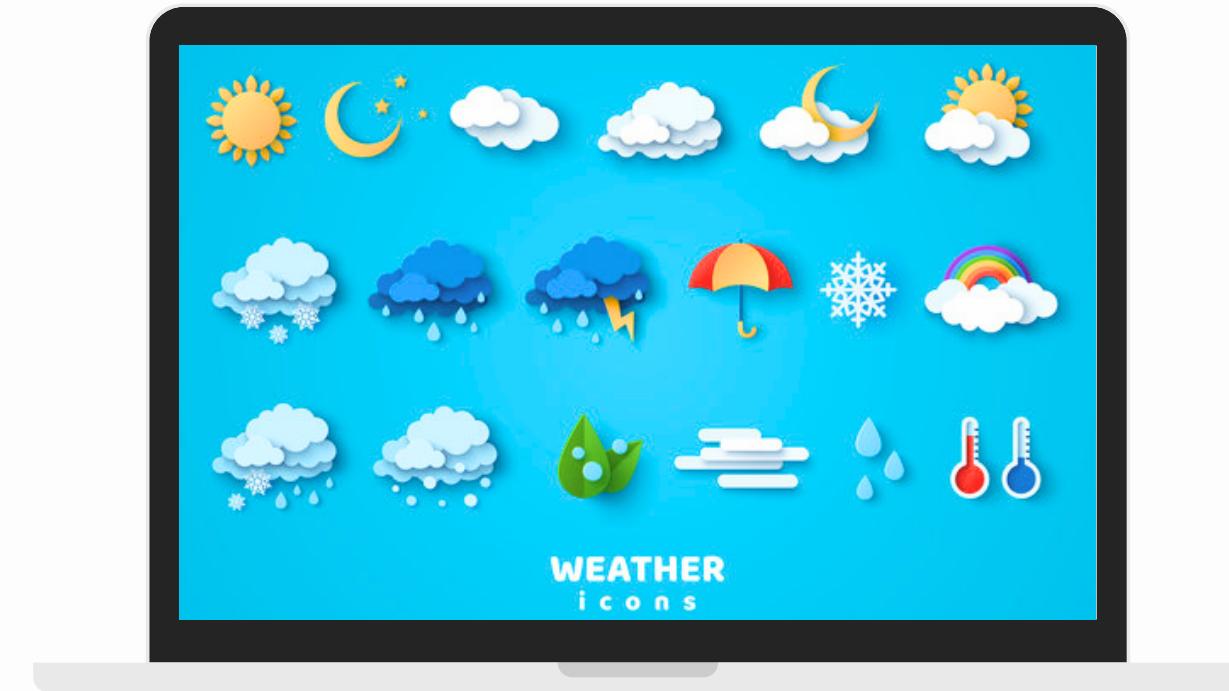
- Real-time weather monitoring and analytics system
- Provides live weather data from 20 major cities worldwide

Weather Information Services

Accurate and up-to-date weather updates for individuals and businesses

Smart Cities

Adaptive traffic control and public safety alerts based on weather conditions



Travel and Tourism

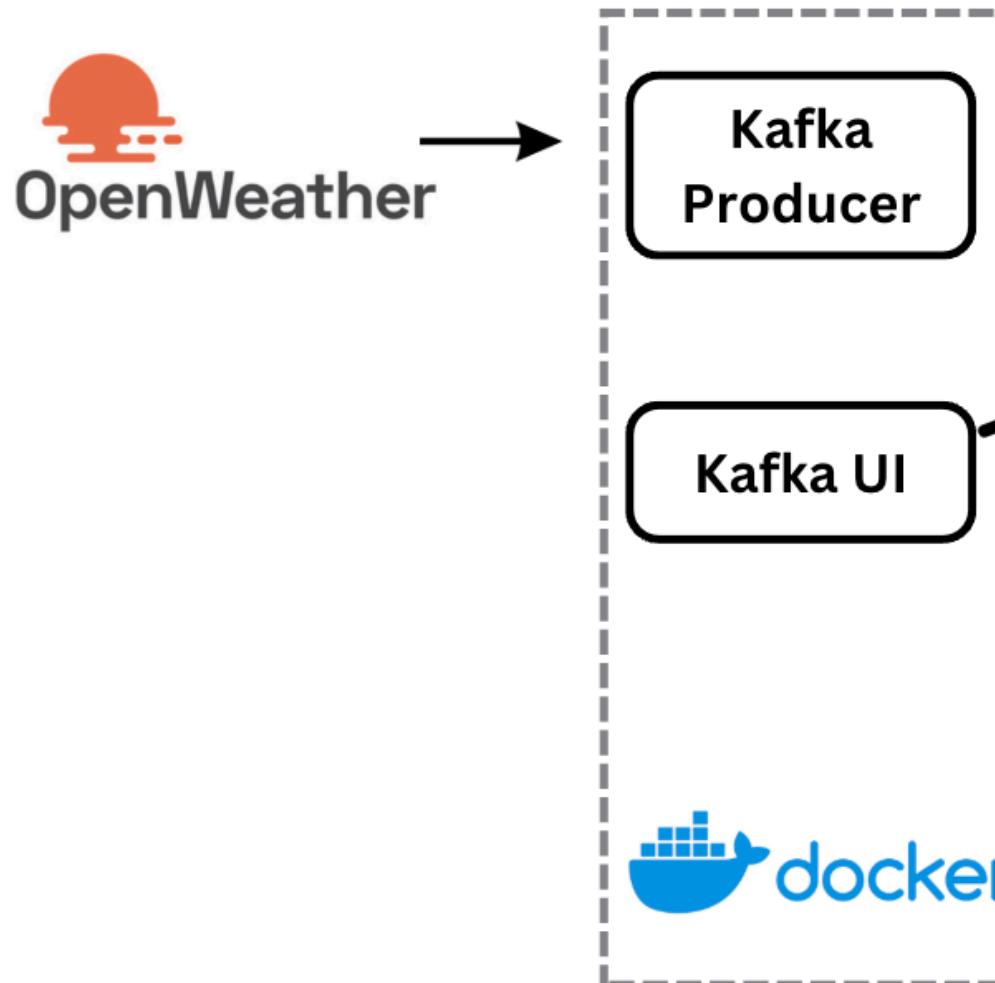
Provides real-time weather insights for better travel planning

Environmental Monitoring

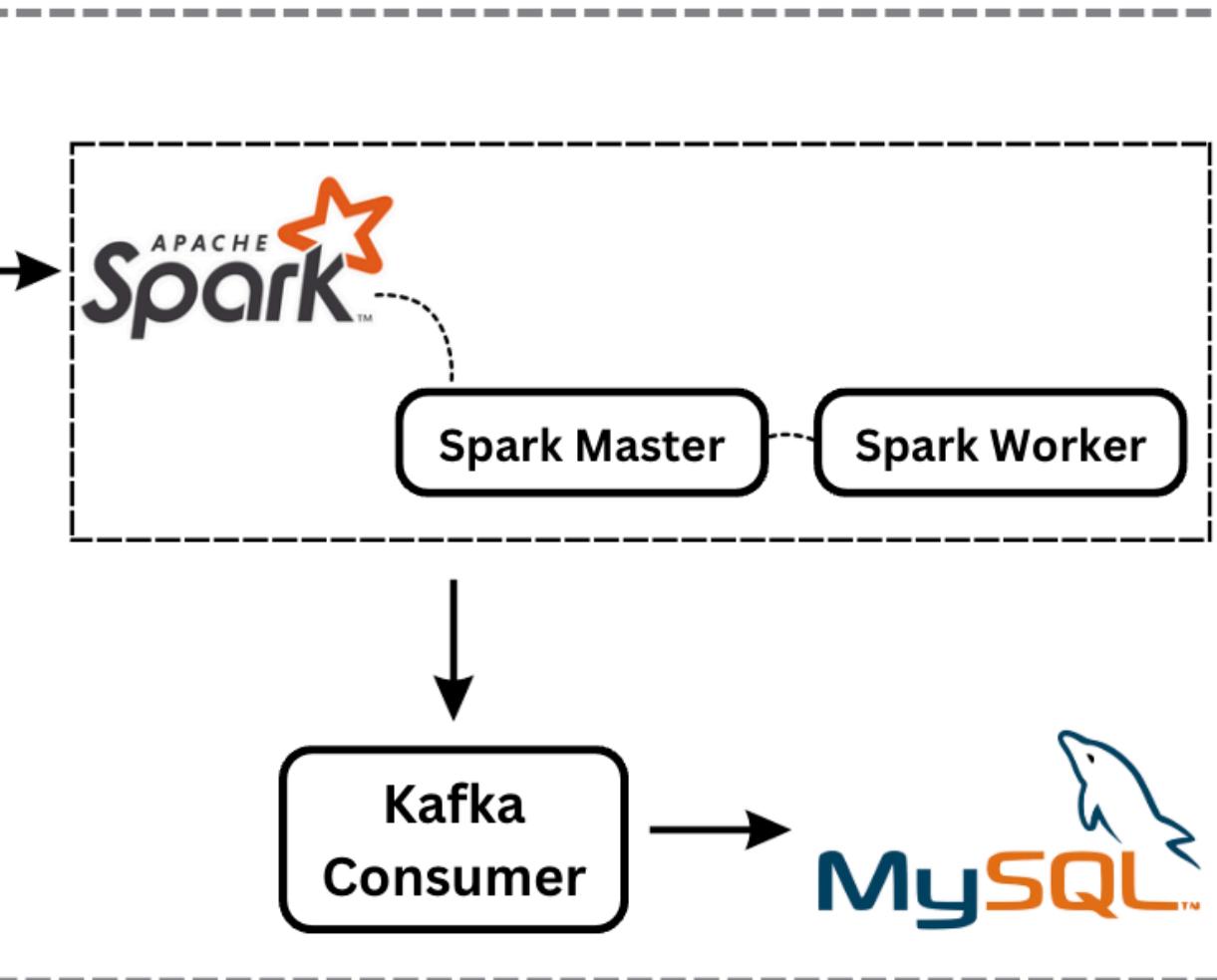
Supports climate change studies and environmental analysis

System Architecture

Data Collection



Real-Time Data Processing



User Interface



Containerization

Data Storage

Data Source



OpenWeather API

The weather data is accessed through the OpenWeather API via REST API calls, which require authentication using an API key. This API offers a range of features, including the "One Call API," which delivers structured JSON responses.

- **City Name**
- **Country Code**
- **Temperature**
- **Humidity**
- **Timestamp**
- **Feels Like**
- **Min/Max Temperature**
- **Pressure**
- **Visibility**
- **Wind Speed**
- **Wind Direction (Wind Degree)**
- **Rain (Rain 1h)**
- **Cloud Coverage**
- **Description**
- **Icon**

Data Processing Workflow

Data Ingestion

- **Frequency:** Weather data is fetched from the OpenWeather API every 5 minutes.
- **Method:** The data is sent to the Kafka topic named weatherDetails.

Stream Processing

- Subscribes to the weatherDetails Kafka topic to consume incoming data.
- Processes the data in real-time for immediate analysis.

Metric Calculations

Computes important weather metrics:

- **Heat Index:** Reflects how hot it feels considering humidity.
- **Dew Point:** Indicates moisture content in the air.
- **Wind Chill:** Represents how cold it feels due to wind.

Data Processing Workflow

Metric Calculations

Computes important weather metrics:

- **Heat Index:** Reflects how hot it feels considering humidity.
- **Dew Point:** Indicates moisture content in the air.
- **Wind Chill:** Represents how cold it feels due to wind.

Heat Index (HI):

$$HI = 0.5 \times (T + 61.0 + ((T - 68.0) \times 1.2) + (H \times 0.094))$$

- where T is the **temperature** and H is the **humidity (%)**

Dew Point:

$$\alpha = \frac{a \times T}{b + T} + \ln \left(\frac{H}{100} \right)$$

- where T is the **temperature**, H is the **humidity**, a = 17.27, and b = 243.12.

Wind Chill:

$$WindChill = 13.12 + 0.6215 \times T - 11.37 \times (V^{0.16}) + 0.3965 \times T \times (V^{0.16})$$

- where T is the **temperature** and V is the **wind speed (km/h)**.

Data Processing Workflow

Data Publishing

Processed metrics are published to two separate Kafka topics:

- **metricsWeather**: For real-time calculated metrics.
- **aggregatedWeather**: For hourly aggregated data.

Data Storage

MySQL Database:

- Raw and processed data are stored for historical analysis.

Visualization

Streamlit:

- Fetches data from the MySQL database.
- Dashboard updates automatically every 3 minutes for real-time insights.

User Interface & Demo

The screenshot shows the "Real-Time Weather" section of the application. At the top, there is a "Select a city" label and a dropdown menu with the placeholder "Choose an option". A list of cities is visible in the dropdown, including Bolzano, Seoul, London, New York, Paris, Mumbai, Sydney, and Beijing. Below the dropdown, there is a timestamp "2025-01-22 22:00:00" and a "Max temperature" value of "2.97".

The screenshot shows the "Real-Time Weather" section after selecting "Bolzano". The dropdown now shows "Bolzano". Below it, a message says "Fetching the latest weather for Bolzano...". Under the "Latest Weather in Bolzano" heading, there is an "Aggregated Weather Data" section and a "Key Weather Metrics" section. Both sections display the same timestamp "2025-01-22 22:00:00" and a "Max temperature" value of "2.97".

User Interface & Demo

The screenshot displays two views of the Weather Data Viewer application. The left view shows 'Key Weather Metrics' for a specific timestamp: **2025-01-22 22:55:31**. The right view shows 'Aggregated Weather Data' for a window from **2025-01-22 22:00:00** to **2025-01-22 23:00:00**.

Key Weather Metrics (Left):

- Timestamp: **2025-01-22 22:55:31**
- Dew point: **0.977597**
- Wind chill: **2.97**
- Heat index: **-2.944**

Aggregated Weather Data (Right):

- Window start: **2025-01-22 22:00:00**, Max temperature: **2.97**
- Window end: **2025-01-22 23:00:00**, Avg humidity: **87.0**
- Avg temperature: **2.97**, Min humidity: **87.0**
- Min temperature: **2.97**, Max humidity: **87.0**

The screenshot shows the 'Latest Weather Details' for Italy (IT). The data includes various weather parameters such as temperature, feels like, visibility, wind speed, and cloud coverage.

Latest Weather Details:

- Country: **IT**
- Temperature: **2.17**
- Feels like: **2.17**
- Temp min: **2.17**
- Temp max: **6.65**
- Pressure: **1017**
- Humidity: **84**
- Visibility: **794**
- Wind speed: **0.45**
- Wind deg: **0**
- Rain 1h: **0.0**
- Cloud coverage: **93**
- Description: **overcast clouds**
- Timestamp: **2025-01-23 00:22:33**

Challenges and Solutions



Kafka Configuration Errors

- **Challenge:** Configuration issues delayed setup.
- **Solution:** reviewed Kafka's configuration, optimized partitioning, and tested the setup.

Scalability Issues

- **Challenge:** Handling high data volumes across multiple cities
- **Solution:** Optimized batch intervals, used window functions for aggregation, and scaled resources.

Front-End Development Expertise

- **Challenge:** Delays due to limited expertise in complex front-end technologies.
- **Solution:** Shifted to Streamlit for an easier development process.

Challenges and Solutions

Database Selection

- **Challenge:** Problems with PostgreSQL installation.
- **Solution:** Switched to MySQL for stable data storage.

Time Management

- **Challenge:** Need for time to learn new technologies.
- **Solution:** Allocated specific time for research and skills improvement.



Conclusion

01

Successful Integration of Real-Time Technologies

Successfully integrated technologies like Kafka, Spark, MySQL, and Streamlit to create a scalable weather monitoring system.

02

Impactful Insights Through Advanced Metrics

Provided useful metrics like heat index, dew point, and wind chill for enhanced weather analysis.

03

User-Friendly and Interactive Design

Delivered an intuitive Streamlit interface for real-time and aggregated weather data across 20 cities.

04

Lessons in Adaptability and Scalability

Overcame challenges like front-end limitations and scalability through solutions like MySQL and Docker.

Q&A

Thank you for your attention!