VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018, KARNATAKA

PARALLEL PROGRAMMING PROJECT REPORT ON

"Weather data analysis using hybrid programming"

Submitted By

ZEENATH BANU S 4PM22CD063

Submitted in the partial fulfillment of the requirements for the PP Project (BAD701)

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

UNDER THE GUIDANCE OF	COURSE INSTRUCTOR
Mrs Chethana T	Mrs Nithya H L
Assistant professor	Assistant professor
CSE(DS)	CSD

HEAD OF THE DEPARTMENT

Dr Sunitha B S

Head of the department CSE(DS)

Prerana Educational and Social Trust ®

PES INSTITUTE OF TECHNOLOGY & MANAGEMENT

NH-206, SAGAR ROAD, SHIVAMOGGA-577204

(Approved by AICTE, New Delhi | Affiliated to VTU, Belagavi | Recognized by Govt. of Karnataka)

NBA Accredited Programs: CSE, CV, ECE, ISE, EEE & ME

Telephone:- +91 9380741865, 8147053063 | Website:- pestrust.edu.in/pesitm | Email:- principal_pesitm@pes.edu

Contents

List of Figures

Abstract

Weather data analysis has become increasingly important for various sectors such as agriculture, transportation, disaster management, and daily life planning. Accurate analysis of weather information helps predict trends, plan resources, and make timely decisions. This project, "Weather Data Analysis Using Hybrid Programming," focuses on analyzing datasets that include city, weather type, temperature, humidity, wind speed, and other meteorological parameters. By using a hybrid programming approach, the project combines C with MPI (Message Passing Interface) for high-performance data processing and Python with Streamlit for creating interactive dashboards. The C module efficiently handles large-scale computations, while the Python module allows real-time visualization and user-friendly exploration of weather patterns across different cities.

The hybrid system enables users to compare weather trends, identify anomalies, and generate insights for multiple locations simultaneously. Using MPI improves computation speed and scalability, especially when processing large datasets, whereas Streamlit provides interactive graphs, charts, and downloadable reports for easy interpretation. Sample datasets and screenshots demonstrate how this approach can effectively summarize and analyze weather information, providing both speed and usability. This project highlights the benefits of combining high-performance computing with modern visualization tools to deliver an effective weather data analysis system that can support both research and practical decision-making.

Chapter 1:

Introduction

Weather significantly impacts human life, agriculture, transportation, and overall planning. Accurate weather data analysis is essential for predicting conditions, managing resources, and mitigating the effects of extreme weather events. With the increasing availability of data from multiple cities and sources, it becomes crucial to process and analyze this information efficiently. Weather datasets typically contain parameters such as **city, weather type, temperature, humidity, wind speed, and precipitation**, which, when properly analyzed, can reveal patterns, trends, and anomalies across different regions. Such insights are useful not only for daily planning but also for long-term studies in climate change, disaster management, and urban development.

Traditional methods of processing weather data are often slow and unable to handle large datasets effectively. To overcome these challenges, this project employs a **hybrid programming approach** combining **C with MPI (Message Passing Interface)** for high-performance parallel data processing and **Python with Streamlit** for interactive visualization. The C module ensures faster computation of large datasets, while Python provides a user-friendly platform to explore, visualize, and interpret the results. This hybrid approach allows users to **compare weather conditions across cities, analyze different weather types, and generate meaningful reports** in real time, making it a robust solution for both research and practical applications.

Chapter 2:

2.1 Problem Statement :

With the growing availability of weather data from multiple cities, including city names, weather types, temperature, humidity, wind speed, and precipitation, traditional methods of analysis have become slow, inefficient, and difficult to scale. Users often face challenges in quickly extracting meaningful insights or visualizing trends due to the large volume and complexity of the data. This project addresses the problem of efficiently processing, analyzing, and visualizing large-scale weather datasets by using a hybrid programming approach that combines C with MPI for high-performance computation and Python with Streamlit for interactive dashboards. The system enables fast processing, clear visualization, and easy comparison of weather patterns across different cities and weather types, supporting better decision-making and analysis.

Chapter 2:

2.2 Objectives :

1. To process large weather datasets including city, weather type, temperature, and humidity.
2. To use C with MPI for fast computation and Python with Streamlit for visualization.
3. To analyze weather trends across multiple cities.
4. To provide interactive dashboards and reports.
5. To improve speed and scalability of data processing.
6. To help users make informed decisions from weather analysis.

Chapter 3 :

3.1 Methodology :

1. Data Collection: Weather data (past, current, and forecast) for multiple cities is fetched using the WeatherAPI through a C program.

2. Parallel Processing : The C code uses MPI (Message Passing Interface) to process multiple cities' data simultaneously, improving speed and efficiency.

3. Data Storage : All collected weather parameters such as city, temperature, humidity, wind speed, and weather type are saved into a CSV file.

4. Data Processing : The Python module (Streamlit app) reads and cleans the CSV data for visualization.

5. Visualization : The processed data is displayed using interactive charts, graphs, and dashboards in Streamlit to show weather trends clearly.

6. User Interaction : Users can analyze, compare, and interpret weather data for different cities in real time through the dashboard.

3.2 Flowchart:

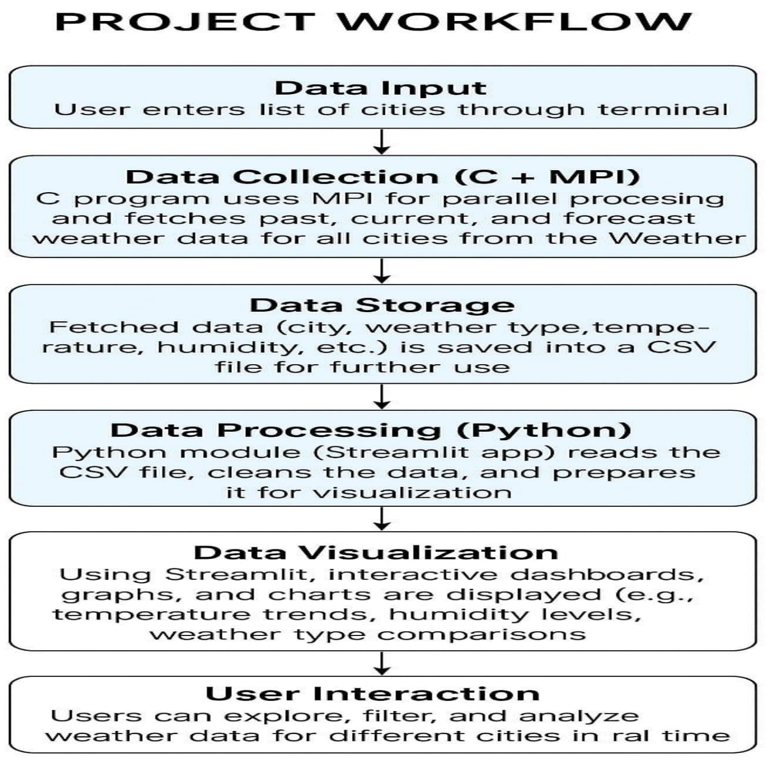
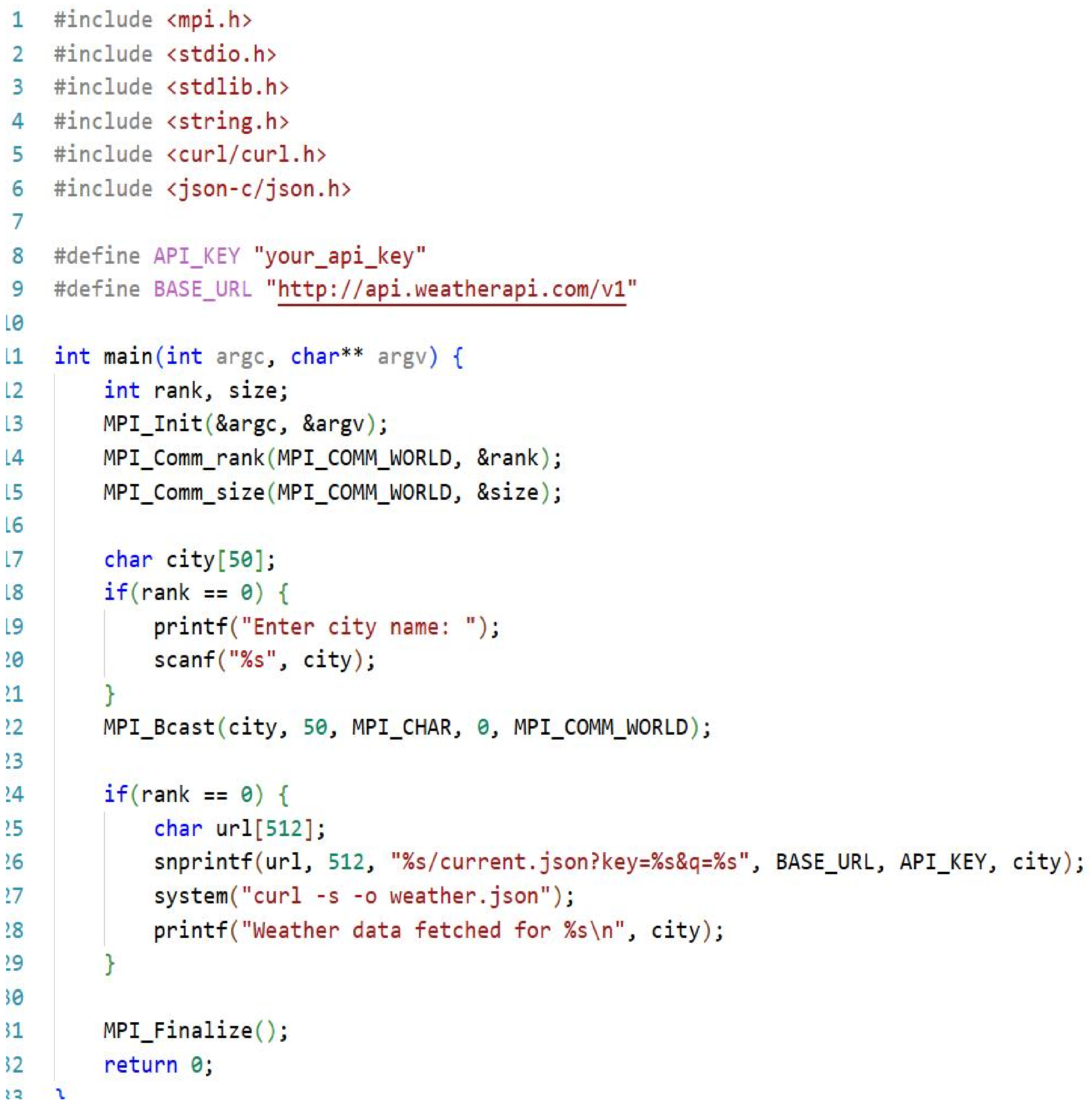

PROJECT WORKFLOW
Data Input
User enters list of cities through terminal
Data Collection (C + MPI)
C program uses MPI for parallel procesing and fetches past, current, and forecast weather data for all cities from the Weather
Data Storage
Fetched data (city, weather type,tempe-rature, humidity, etc.) is saved into a CSV file for further use
Data Processing (Python)
Python module (Streamlit app) reads the CSV file, cleans the data, and prepares it for visualization
Data Visualization
Using Streamlit, interactive dashboards, graphs, and charts are displayed (e.g., temperature trends, humidity levels, weather type comparisons
User Interaction
Users can explore, filter, and analyze weather data for different cities in ral time

Fig 3.2: Work Flow of weather data analysis

Chapter 4:
4.1 Implementation and Coding :

4.1.1 Backend Processing (C - MPI Module):

This C program uses MPI for parallel processing to fetch and analyze weather data (past, current, and forecast) for multiple cities using the WeatherAPI.

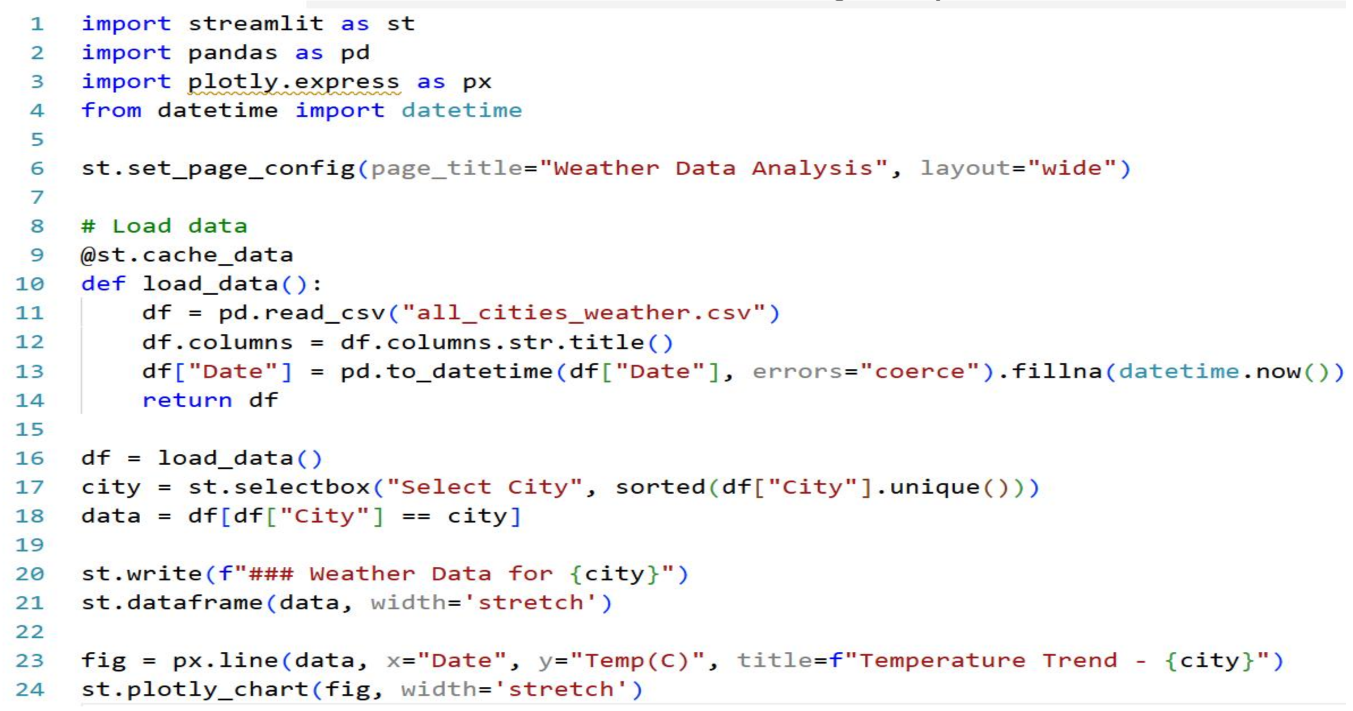


```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>
#include <json-c/json.h>

#define API_KEY "your_api_key"
#define BASE_URL "http://api.weatherapi.com/v1"

int main(int argc, char** argv) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    char city[50];
    if(rank == 0) {
        printf("Enter city name: ");
        scanf("%s", city);
    }
    MPI_Bcast(city, 50, MPI_CHAR, 0, MPI_COMM_WORLD);

    if(rank == 0) {
        char url[512];
        snprintf(url, 512, "%s/current.json?key=%s&q=%s", BASE_URL, API_KEY, city);
        system("curl -s -o weather.json");
        printf("Weather data fetched for %s\n", city);
    }

    MPI_Finalize();
    return 0;
}
```

It processes data efficiently and saves all results, including temperature, humidity, and conditions, into a CSV file for further visualization.

4.1.2 Frontend Visualization (Python - Streamlit Module) :

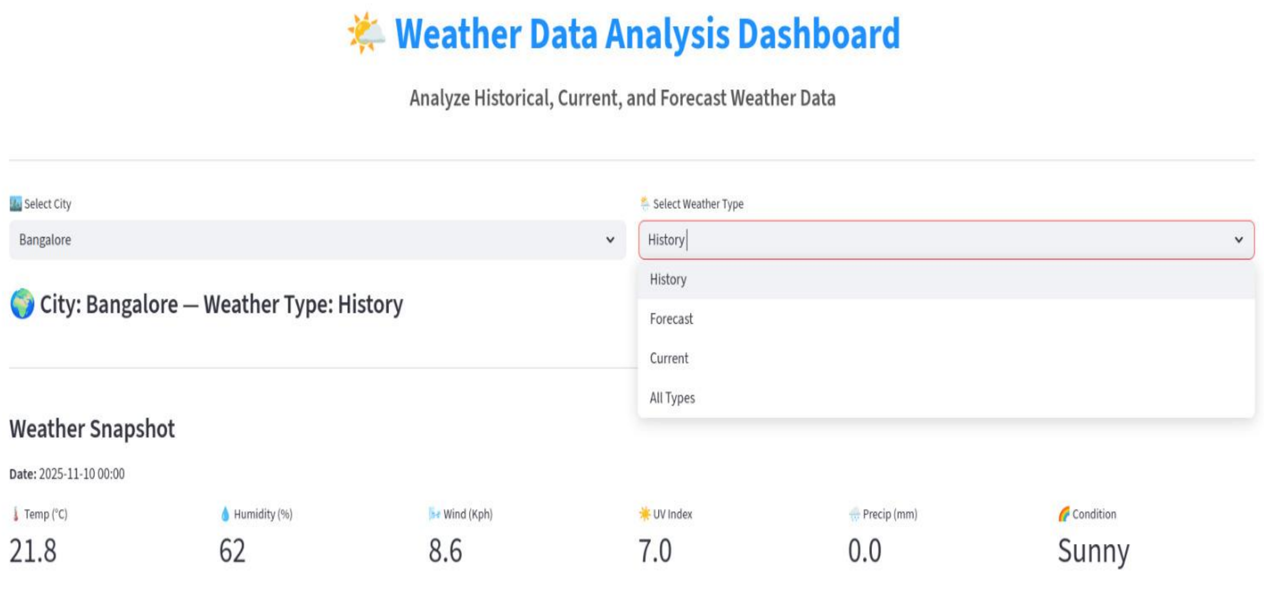
This Streamlit application provides an **interactive weather data analysis dashboard** that visualizes historical, current, and forecast data using **Plotly charts**.



```python
import streamlit as st
import pandas as pd
import plotly.express as px
from datetime import datetime

st.set_page_config(page_title="Weather Data Analysis", layout="wide")

# Load data
@st.cache_data
def load_data():
    df = pd.read_csv("all_cities_weather.csv")
    df.columns = df.columns.str.title()
    df["Date"] = pd.to_datetime(df["Date"], errors="coerce").fillna(datetime.now())
    return df

df = load_data()
city = st.selectbox("Select City", sorted(df["City"].unique()))
data = df[df["City"] == city]

st.write(f"### Weather Data for {city}")
st.dataframe(data, width='stretch')

fig = px.line(data, x="Date", y="Temp(C)", title=f"Temperature Trend - {city}")
st.plotly_chart(fig, width='stretch')
```

Chapter 5: 5.1 Results (screenshots of output) :

5.1.1 Weather Dashboard : For displaying the interactive Streamlit dashboard with current, past, and forecast weather information.



⛅ Weather Data Analysis Dashboard

Analyze Historical, Current, and Forecast Weather Data

Select City: Bangalore

Select Weather Type: History

🌍 City: Bangalore — Weather Type: History

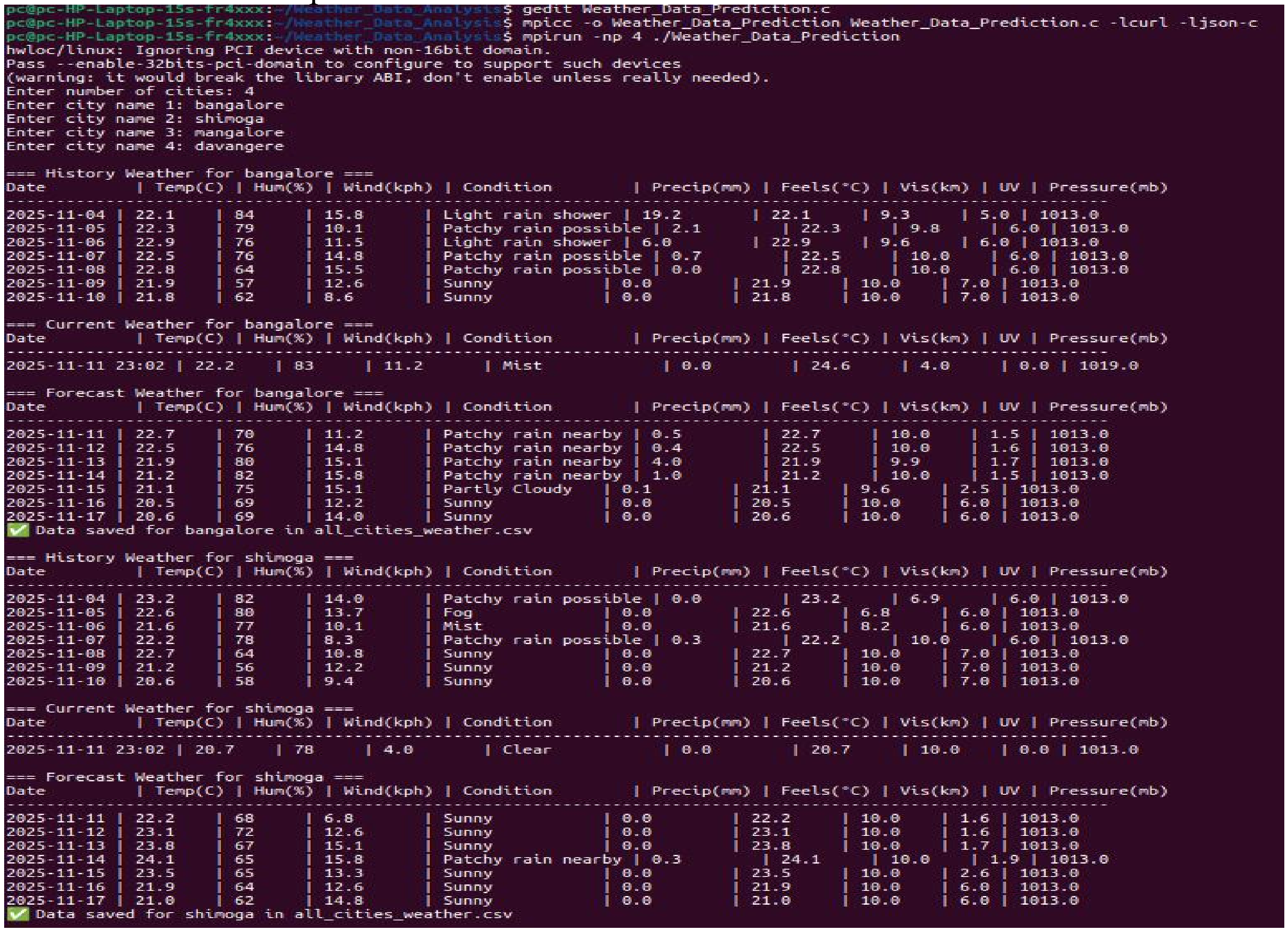
Weather Snapshot

Date: 2025-11-10 00:00

Temp (°C)	Humidity (%)	Wind (Kph)	UV Index	Precip (mm)	Condition
21.8	62	8.6	7.0	0.0	Sunny

Fig 5.1.1: Weather Dashboard

5.1.2 Terminal Output : For showing the program's console output, including fetched weather data for multiple cities.



```
pc@pc-HP-Laptop-15s-fr4xxx:~/Weather_Data_Analysis$ gedit Weather_Data_Prediction.c
pc@pc-HP-Laptop-15s-fr4xxx:~/Weather_Data_Analysis$ mpicc -o Weather_Data_Prediction Weather_Data_Prediction.c -lcurl -ljson-c
pc@pc-HP-Laptop-15s-fr4xxx:~/Weather_Data_Analysis$ mpirun -np 4 ./Weather_Data_Prediction
hwloc/linux: Ignoring PCI device with non-16bit domain.
Pass --enable-32bits-pci-domain to configure to support such devices
(warning: it would break the library ABI, don't enable unless really needed).
Enter number of cities: 4
Enter city name 1: bangalore
Enter city name 2: shimoga
Enter city name 3: mangalore
Enter city name 4: davangere

=== History Weather for bangalore ===
Date          | Temp(C) | Hum(%) | Wind(kph) | Condition           | Precip(mm) | Feels(°C) | Vis(km) | UV | Pressure(mb)
-------------------------------------------------------------------------------------------------------------------
2025-11-04 | 22.1     | 84     | 15.8       | Light rain shower | 19.2        | 22.1      | 9.3       | 5.0 | 1013.0
2025-11-05 | 22.3     | 79     | 10.1       | Patchy rain possible | 2.1        | 22.3      | 9.8       | 6.0 | 1013.0
2025-11-06 | 22.9     | 76     | 11.5       | Light rain shower | 6.0         | 22.9      | 9.6       | 6.0 | 1013.0
2025-11-07 | 22.5     | 76     | 14.8       | Patchy rain possible | 0.7        | 22.5      | 10.0      | 6.0 | 1013.0
2025-11-08 | 22.8     | 64     | 15.5       | Patchy rain possible | 0.0        | 22.8      | 10.0      | 6.0 | 1013.0
2025-11-09 | 21.9     | 57     | 12.6       | Sunny             | 0.0         | 21.9      | 10.0      | 7.0 | 1013.0
2025-11-10 | 21.8     | 62     | 8.6        | Sunny             | 0.0         | 21.8      | 10.0      | 7.0 | 1013.0

=== Current Weather for bangalore ===
Date          | Temp(C) | Hum(%) | Wind(kph) | Condition           | Precip(mm) | Feels(°C) | Vis(km) | UV | Pressure(mb)
-------------------------------------------------------------------------------------------------------------------
2025-11-11 23:02 | 22.2    | 83     | 11.2       | Mist               | 0.0         | 24.6      | 4.0       | 0.0 | 1019.0

=== Forecast Weather for bangalore ===
Date          | Temp(C) | Hum(%) | Wind(kph) | Condition           | Precip(mm) | Feels(°C) | Vis(km) | UV | Pressure(mb)
-------------------------------------------------------------------------------------------------------------------
2025-11-11 | 22.7     | 70     | 11.2       | Patchy rain nearby | 0.5         | 22.7      | 10.0      | 1.5 | 1013.0
2025-11-12 | 22.5     | 76     | 14.8       | Patchy rain nearby | 0.4         | 22.5      | 10.0      | 1.6 | 1013.0
2025-11-13 | 21.9     | 80     | 15.1       | Patchy rain nearby | 4.0         | 21.9      | 9.9       | 1.7 | 1013.0
2025-11-14 | 21.2     | 82     | 15.8       | Patchy rain nearby | 1.0         | 21.2      | 10.0      | 1.5 | 1013.0
2025-11-15 | 21.1     | 75     | 15.1       | Partly Cloudy     | 0.1         | 21.1      | 9.6       | 2.5 | 1013.0
2025-11-16 | 20.5     | 69     | 12.2       | Sunny             | 0.0         | 20.5      | 10.0      | 6.0 | 1013.0
2025-11-17 | 20.6     | 69     | 14.0       | Sunny             | 0.0         | 20.6      | 10.0      | 6.0 | 1013.0
✅ Data saved for bangalore in all_cities_weather.csv

=== History Weather for shimoga ===
Date          | Temp(C) | Hum(%) | Wind(kph) | Condition           | Precip(mm) | Feels(°C) | Vis(km) | UV | Pressure(mb)
-------------------------------------------------------------------------------------------------------------------
2025-11-04 | 23.2     | 82     | 14.0       | Patchy rain possible | 0.0        | 23.2      | 6.9       | 6.0 | 1013.0
2025-11-05 | 22.6     | 80     | 13.7       | Fog               | 0.0         | 22.6      | 6.8       | 6.0 | 1013.0
2025-11-06 | 21.6     | 77     | 10.1       | Mist              | 0.0         | 21.6      | 8.2       | 6.0 | 1013.0
2025-11-07 | 22.2     | 78     | 8.3        | Patchy rain possible | 0.3        | 22.2      | 10.0      | 6.0 | 1013.0
2025-11-08 | 22.7     | 64     | 10.8       | Sunny             | 0.0         | 22.7      | 10.0      | 7.0 | 1013.0
2025-11-09 | 21.2     | 56     | 12.2       | Sunny             | 0.0         | 21.2      | 10.0      | 7.0 | 1013.0
2025-11-10 | 20.6     | 58     | 9.4        | Sunny             | 0.0         | 20.6      | 10.0      | 7.0 | 1013.0

=== Current Weather for shimoga ===
Date          | Temp(C) | Hum(%) | Wind(kph) | Condition           | Precip(mm) | Feels(°C) | Vis(km) | UV | Pressure(mb)
-------------------------------------------------------------------------------------------------------------------
2025-11-11 23:02 | 20.7    | 78     | 4.0        | Clear              | 0.0         | 20.7      | 10.0      | 0.0 | 1013.0

=== Forecast Weather for shimoga ===
Date          | Temp(C) | Hum(%) | Wind(kph) | Condition           | Precip(mm) | Feels(°C) | Vis(km) | UV | Pressure(mb)
-------------------------------------------------------------------------------------------------------------------
2025-11-11 | 22.2     | 68     | 6.8        | Sunny             | 0.0         | 22.2      | 10.0      | 1.6 | 1013.0
2025-11-12 | 23.1     | 72     | 12.6       | Sunny             | 0.0         | 23.1      | 10.0      | 1.6 | 1013.0
2025-11-13 | 23.8     | 67     | 15.1       | Sunny             | 0.0         | 23.8      | 10.0      | 1.7 | 1013.0
2025-11-14 | 24.1     | 65     | 15.8       | Patchy rain nearby | 0.3         | 24.1      | 10.0      | 1.9 | 1013.0
2025-11-15 | 23.5     | 65     | 13.3       | Sunny             | 0.0         | 23.5      | 10.0      | 2.6 | 1013.0
2025-11-16 | 21.9     | 64     | 12.6       | Sunny             | 0.0         | 21.9      | 10.0      | 6.0 | 1013.0
2025-11-17 | 21.0     | 62     | 14.8       | Sunny             | 0.0         | 21.0      | 10.0      | 6.0 | 1013.0
✅ Data saved for shimoga in all_cities_weather.csv
```

5.1.3 Graphs and Visualizations : For showing charts and graphs like temperature trends, humidity, and weather type comparisons.

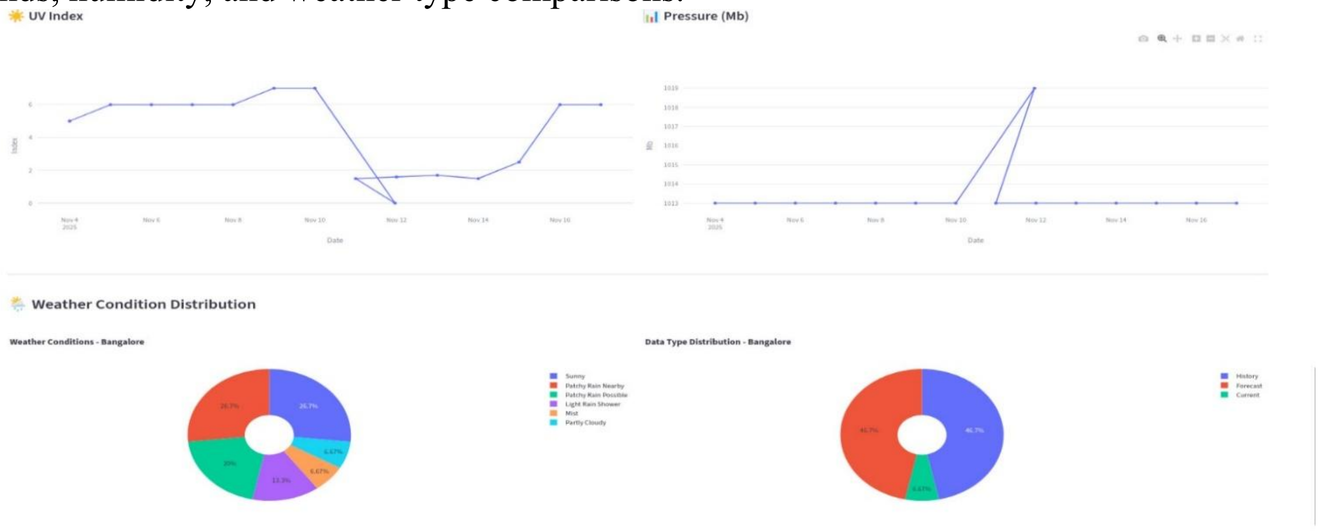


Fig 5.1.3 :Graphs and Visualizations

Chapter 6:

Conclusion

The project “Weather Data Analysis Using Hybrid Programming” demonstrates an efficient way to process and analyze large-scale weather datasets from multiple cities. By combining C with MPI for high-performance computation and Python with Streamlit for interactive visualization, the system provides both speed and user-friendly analysis. Users can explore weather trends, compare different weather types, and generate insightful reports quickly. The hybrid approach ensures scalability, accuracy, and clear presentation of complex data, making it a useful tool for researchers, planners, and general users. Overall, this project highlights the benefits of integrating high-performance computing with modern visualization tools to deliver an effective and practical weather analysis system.