Weather Application Project

1. Project Overview

Github Project Link - https://github.com/rahul420206/Weather-Forecast-App

My Portfolio - https://rahul420206.github.io/Portfolio/

The Weather Application is designed to provide real-time weather data for multiple cities using the OpenWeatherMap API. It allows users to view current weather conditions, set temperature thresholds for alerts, and visualize historical weather data over specified time periods.

2. Technology Stack

- Backend Framework: Flask
- Database: SQLite
- Data Visualization: Matplotlib
- API Integration: OpenWeatherMap API
- Threading: Python threading for periodic tasks

3. Application Architecture

The application follows a client-server architecture where:

- Client: HTML templates rendered by Flask to present data.
- Server: Flask application handling requests, fetching weather data, and managing the database.

4. Project Structure

The project structure consists of the following key files:

- `main.py`: The main Flask application containing routes, logic for fetching, storing weather data.
- `weather_database.py`: Module for handling database interactions.
- `weather_app.html`: Frontend HTML template for rendering weather data and visualizations.

Weather Application Project Documentation

5. Code Explanation

5.1. Imports and Initialization

```python
from flask import Flask, render_template, request, jsonify
import time
import requests
import threading
from datetime import datetime
from weather_database import initialize_database, store_weather_data, fetch_weather_data, ...
import matplotlib.pyplot as plt

app = Flask(__name__)
initialize_database()
```

This section imports necessary modules and initializes the Flask app and the database.

5.2. API Configuration

```python
API_KEY = 'YOUR_API_KEY'
BASE_URL = 'https://api.openweathermap.org/data/2.5/weather'
CITIES = ['Delhi', 'Mumbai', 'Chennai', 'Bangalore', 'Kolkata', 'Hyderabad']
user_threshold_temp = 25 # User-defined threshold for alerts
```

API configuration including the API key and the list of cities for which weather data is fetched.

5.3. Fetching Current Weather

```python
def fetch_current_weather(city):
    ...
    if response.status_code == 200:
        data = response.json()
        ...
        return {'temperature': current_temp}
```

This function fetches current weather data for a given city and returns the temperature.

5.4. Weather Fetching Logic

```python
def fetch_weather(city):
    ...
    if response.status_code == 200:
        ...
        if temp_celsius >= user_threshold_temp:
            alert_message = f"ALERT: {city} temperature is {temp_celsius:.2f}°C, which exceeds the threshold of {user_threshold_temp}°C."
    return weather_data.get(city, {}), alert_message
```

The `fetch_weather` function fetches weather data and checks if the current temperature exceeds the user-defined threshold, returning an alert message if it does.

5.5. Routes

```python
@app.route("/", methods=["GET"])
def home():
    ...
    return render_template('weather_app.html', ...)
```

The home route handles the main page of the application, fetching data based on user input and rendering the HTML template.

5.6. Alert Checking

```python
def check_alerts(city):
    ...
    while True:
        ...
        if current_temp > user_threshold:
            print(f"Alert: The current temperature in {city} is {current_temp}°C, exceeding your set threshold.")
        time.sleep(300) # Check every 5 minutes
```

A separate thread checks the current temperature at regular intervals and prints an alert if the threshold is exceeded.

6. Database Design

Weather Application Project Documentation

The SQLite database includes the following tables:

- Weather Data Table: Stores current weather data for each city.

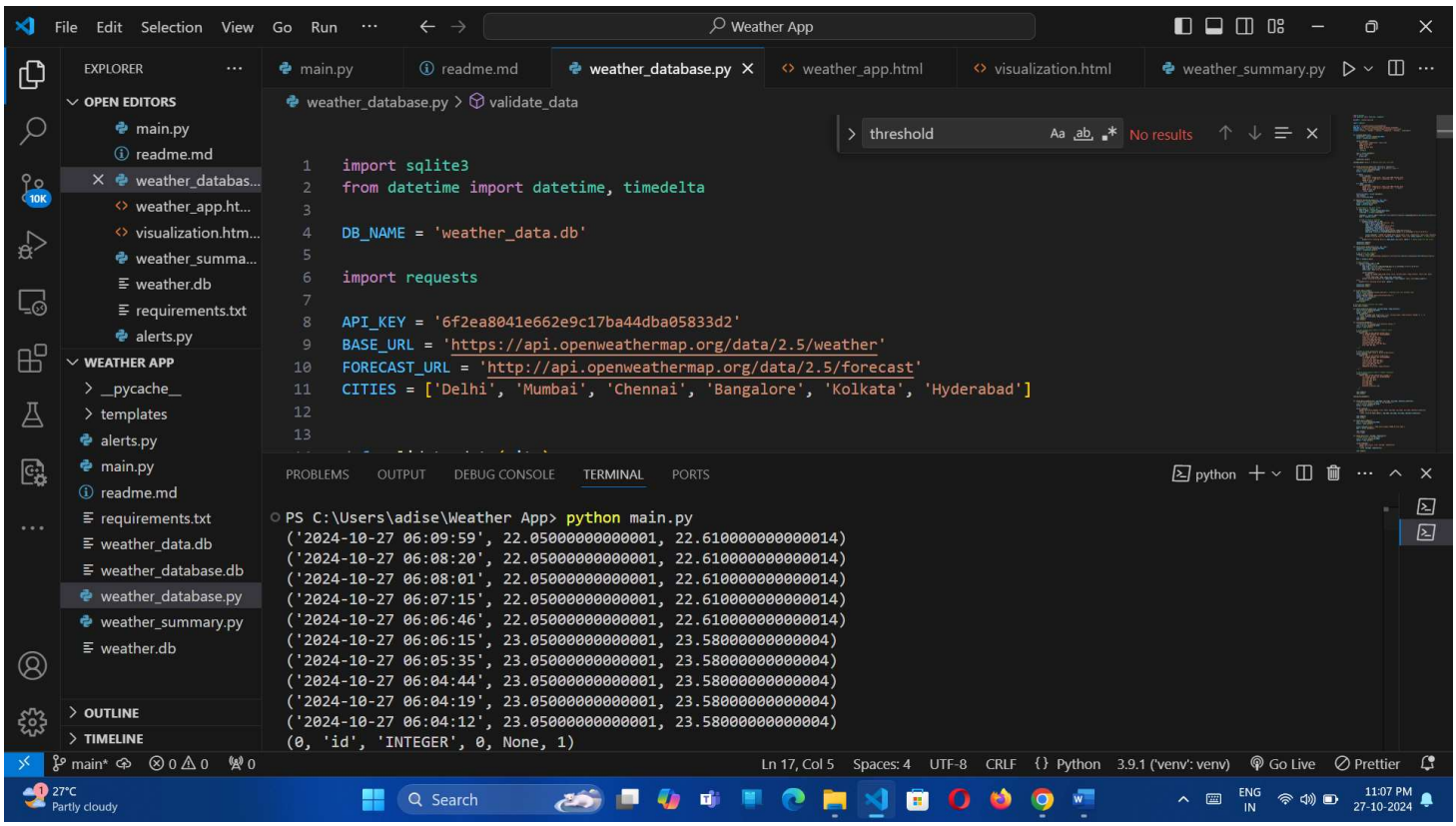
- Forecast Data Table: Stores forecasted weather data.

- Historical Data Table: Stores historical weather data for visualizations.

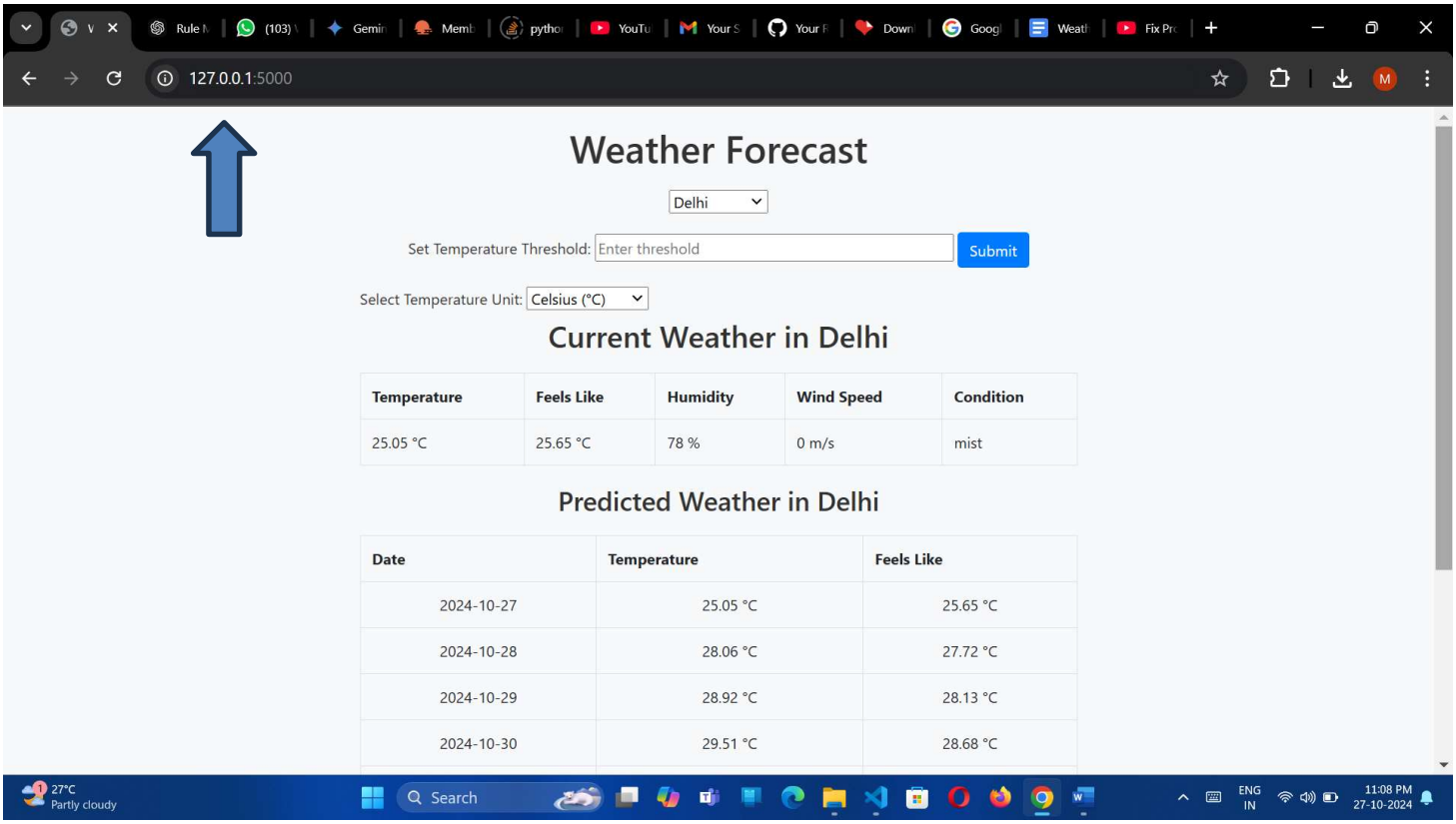
- Alerts Table: Logs alerts based on user-defined thresholds.

7. Features/Usage Instructions

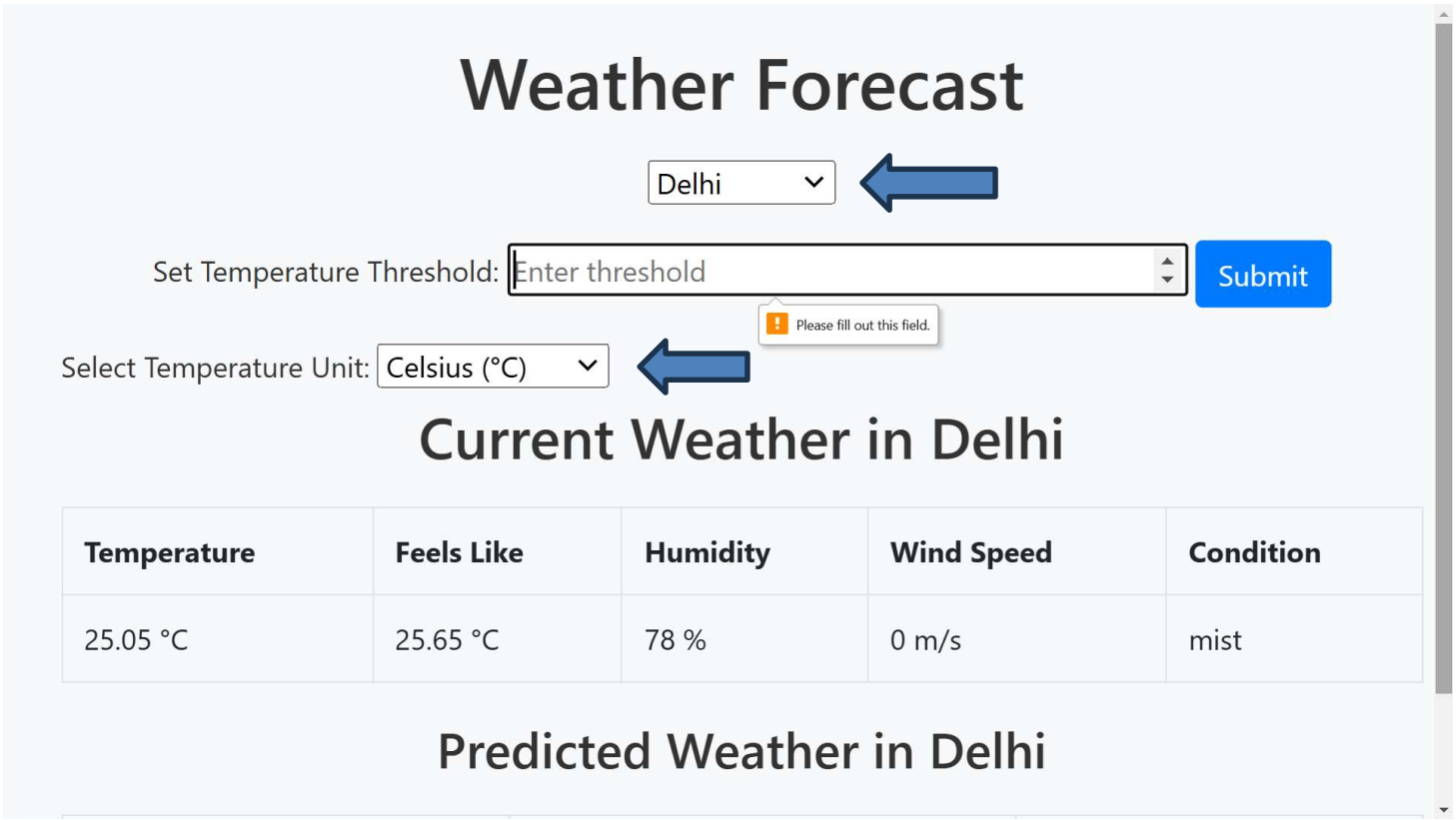
1. Run the Application: Execute `python main.py` to start the Flask server.



2. Access the Application: Open a web browser and navigate to `http://127.0.0.1:5000`.

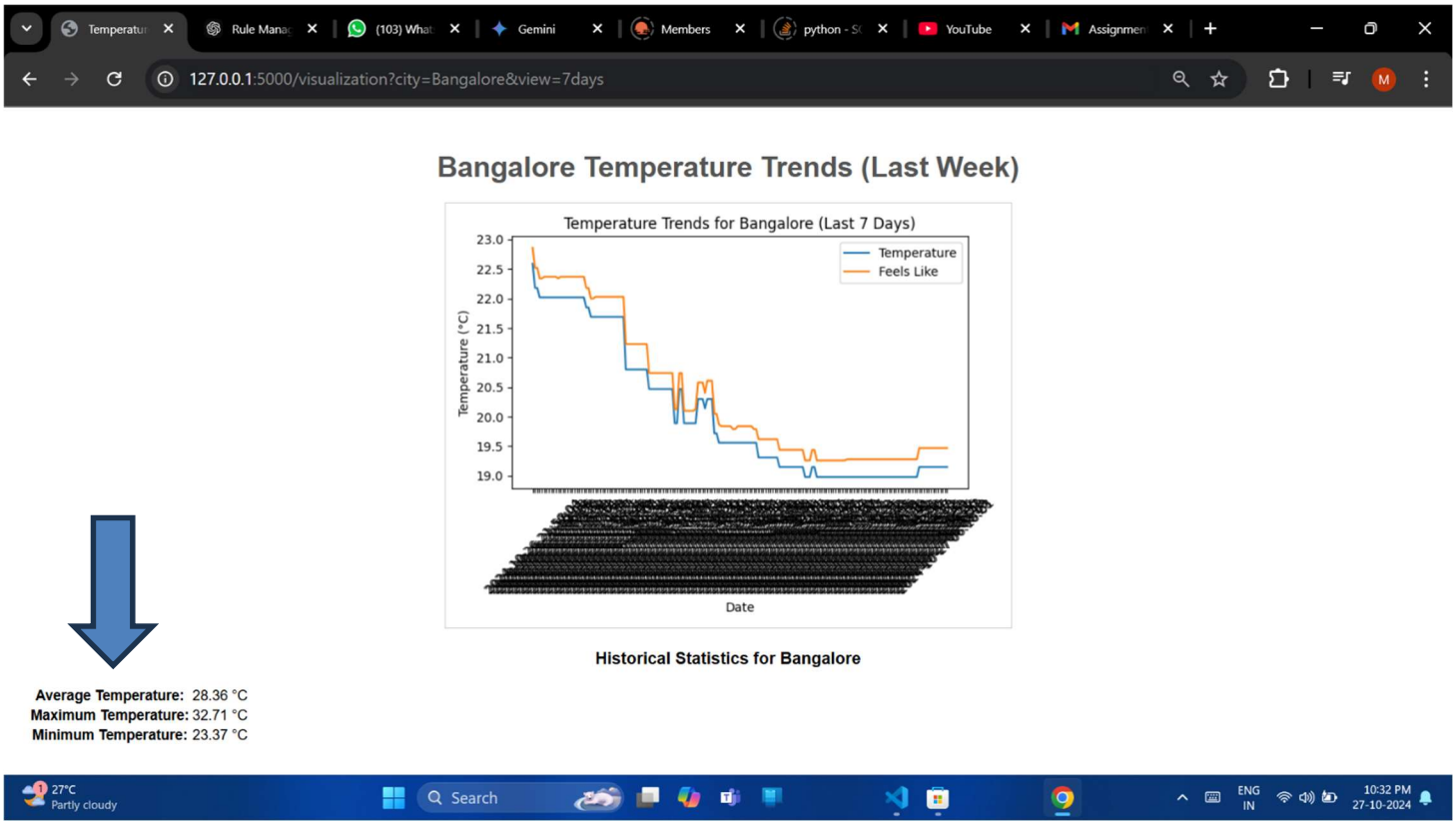


3. Set Threshold/Set Temperature/Select CIty: Use the UI to set a temperature threshold for alerts.



Weather Application Project Documentation

4. Visualize Data: Navigate to the visualization route to view historical weather data. And also metrics from the observed weather trends



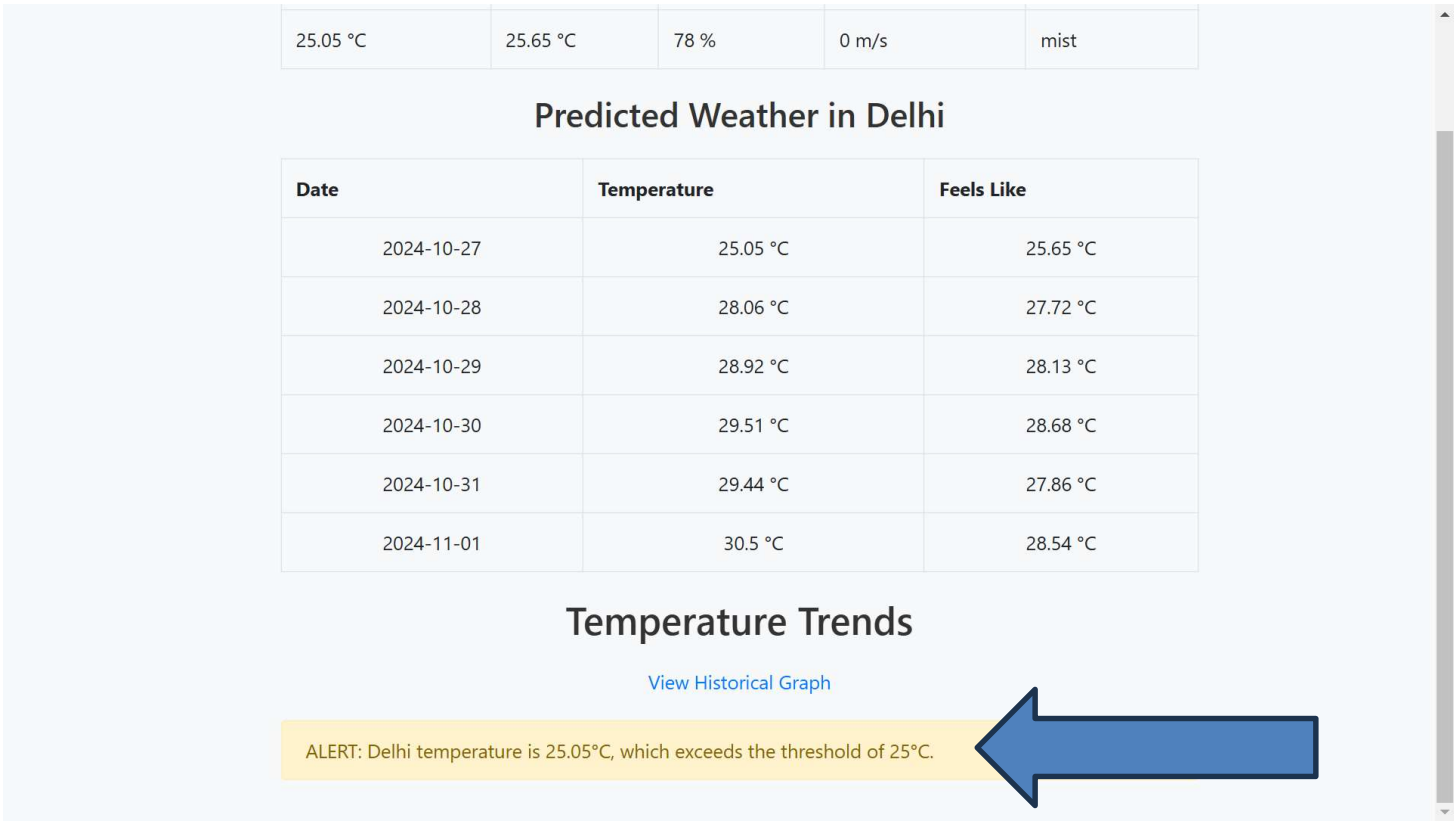
(The black thing below the graph is the dates but because of the weather update for every 5 minutes it got so packed up and looked like this)

5. Future Weather predictions

Predicted Weather in Delhi

Date	Temperature	Feels Like
2024-10-27	25.05 °C	25.65 °C
2024-10-28	28.06 °C	27.72 °C
2024-10-29	28.92 °C	28.13 °C
2024-10-30	29.51 °C	28.68 °C
2024-10-31	29.44 °C	27.86 °C
2024-11-01	30.5 °C	28.54 °C

6. Alerts for the temperatures that exceeds the threshold temperature



25.05 °C	25.65 °C	78 %	0 m/s	mist

Predicted Weather in Delhi

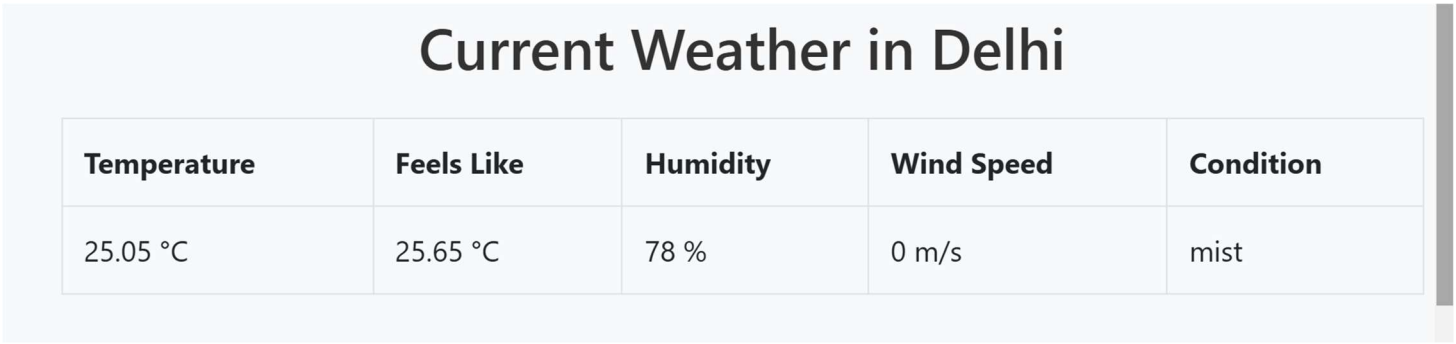
Date	Temperature	Feels Like
2024-10-27	25.05 °C	25.65 °C
2024-10-28	28.06 °C	27.72 °C
2024-10-29	28.92 °C	28.13 °C
2024-10-30	29.51 °C	28.68 °C
2024-10-31	29.44 °C	27.86 °C
2024-11-01	30.5 °C	28.54 °C

Temperature Trends

View Historical Graph

ALERT: Delhi temperature is 25.05°C, which exceeds the threshold of 25°C.

7. Informative current weather details



Current Weather in Delhi

Temperature	Feels Like	Humidity	Wind Speed	Condition
25.05 °C	25.65 °C	78 %	0 m/s	mist

8. Conclusion

The Weather Application provides a comprehensive platform for monitoring and visualizing weather data, utilizing real-time API integration, database storage, and data visualization techniques.