Project Report: Formula 4

I. Project Title:

Real-Time Weather Data Retrieval with OpenWeatherMap API in Python

II. Team Members:

Name	SRN
Hari Yuvan N	PES2UG25AM103
Harshith U Shetty	PES2UG25AM107
Jeevan N	PES2UG25CS232
Jayanth K P	PES2UG25CS231

III. Problem Statement:

The problem is to design a desktop application that provides real-time weather data and forecast for any city, in a clean dashboard-style interface, combining current conditions, short-term forecast, and AI-based practical advice in a single screen.

IV. Approach, Methodology and Data Structure:

Methodology

The development follows a modular process divided into four distinct phases:

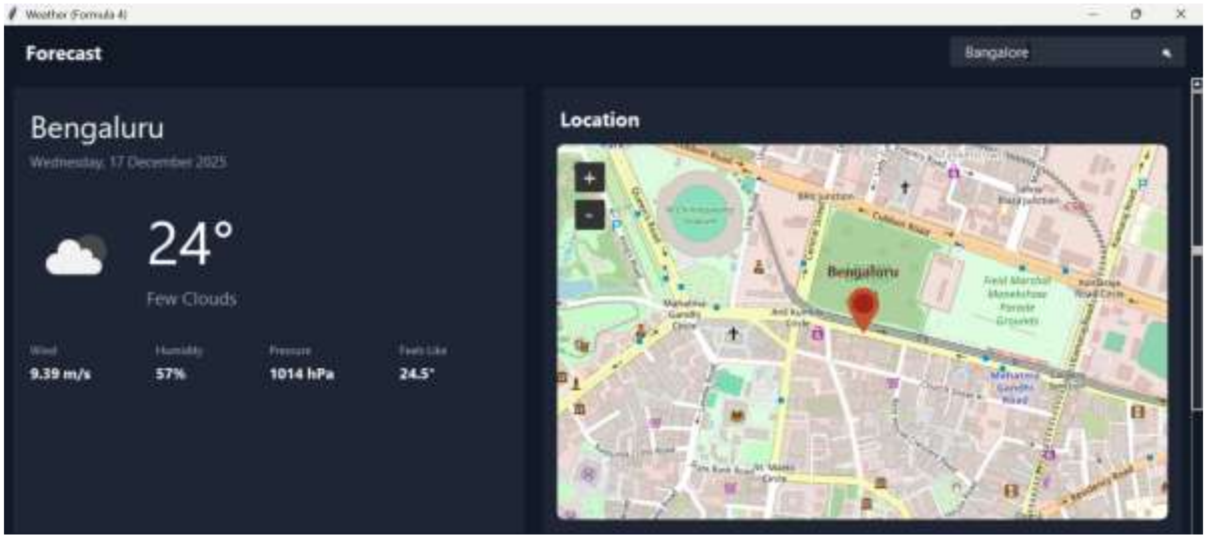
- **API Integration-**
  Requests fetches current weather and 5-day forecast from OpenWeatherMap APIs using city name and API key. Error handling shows messagebox for invalid cities/network issues. Gemini API analyzes data with model fallbacks.
- **Data Processing-**
  JSON responses parse into dictionaries/lists. Extracts main.temp, weather.description, groups forecast by date, processes timestamps with datetime. Downloads/resizes icons via PIL Image.open(BytesIO()). Builds structured AI prompts.
- **GUI Construction-**
  Tkinter Canvas + scrollable_frame creates scrollable dashboard with mousewheel binding. 3x2 grid layout uses create_card() factory for dark theme consistency. Widgets: Entry (Enter-bound), Button, Treeview, Text, TkinterMapView. ttk.Style() customizes dark Treeview.

- **Data Visualization-**
  Hero card shows temp/icon/stats. Dynamic 5-day cards group by date. Treeview populates hourly data. Map updates via set_position(lat, lon). AI text displays in scrollable panel. Real-time widget updates without refresh.

Data Structure

- The application mainly uses Python dictionaries and lists to store and process the JSON data returned by the OpenWeatherMap API (current weather and 5-day forecast).
- Dictionaries hold key sections like main, weather, wind, and coord, while lists store the forecast entries and are also used to group data by date and prepare rows for the hourly table.
- Strings and tuples are used for formatted dates, descriptions, fonts, and coordinate pairs, and widget references are stored in variables so the GUI can be updated dynamically when new data is fetched.

V. Sample Input/Output:



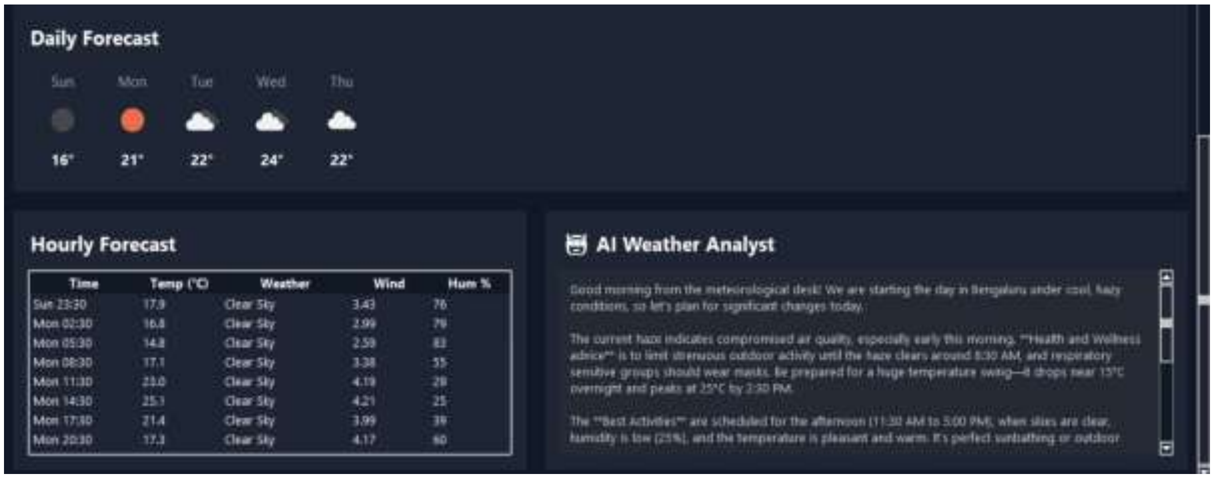
Bengaluru

Wednesday, 17 December 2025

24°

Few Clouds

Wind	Humidity	Pressure	Feels Like
9.39 m/s	57%	1014 hPa	24.5°



Sun	Mon	Tue	Wed	Thu
16°	21°	22°	24°	22°

Time	Temp (°C)	Weather	Wind	Hum %
Sun 23:30	17.9	Clear Sky	3.43	76
Mon 02:30	16.8	Clear Sky	2.99	79
Mon 05:30	14.8	Clear Sky	2.59	83
Mon 08:30	17.1	Clear Sky	3.38	55
Mon 11:30	23.0	Clear Sky	4.19	29
Mon 14:30	25.1	Clear Sky	4.21	25
Mon 17:30	21.4	Clear Sky	3.99	39
Mon 20:30	17.3	Clear Sky	4.17	60

Good morning from the meteorological desk! We are starting the day in Bengaluru under cool, hazy conditions, so let's plan for significant changes today.

The current haze indicates compromised air quality, especially early this morning. **Health and Wellness advice** is to limit strenuous outdoor activity until the haze clears around 8:30 AM, and respiratory sensitive groups should wear masks. Be prepared for a huge temperature swing—it drops near 15°C overnight and peaks at 25°C by 2:30 PM.

The **Best Activities** are scheduled for the afternoon (11:30 AM to 5:00 PM), when skies are clear, humidity is low (25%), and the temperature is pleasant and warm. It's perfect sunbathing or outdoor

Challenges faced:

- Handling API errors (invalid cities, network failures, rate limits) required robust messagebox error handling to prevent crashes.
- Creating a responsive scrollable dashboard layout mixing grid/pack and Canvas for complex sections like map, table, and AI panel.
- Integrating external libraries (TkinterMapView, PIL, Gemini) involved API key setup, image reference management, and model fallback logic.
- Crafting effective AI prompts while staying within token limits and handling quota errors gracefully.

Scope for improvement:

- Add support for more OpenWeatherMap endpoints (air quality, alerts) and user preferences (saved cities, units, themes).
- Implement offline caching of recent weather data for connectivity issues.
- Include interactive charts for temperature/humidity trends and responsive design for different screen sizes.
- Enhance AI with personalized recommendations (sports, health alerts) and multi-language support.