

**AI Weather Forecaster – Project Report**

Academic Task-3: Project(INT-428)

Submitted By:

Name: Ashna Jain

Registration Number: 12304693

Roll No: 29

Section: K23GN

Submitted To:

Dr. Ishan Kumar

**Table Of Content:**

1. Introduction
2. Problem Statement
3. Objectives
4. Real-Life Use Case
5. Technologies and Libraries Used
6. System Architecture
7. Project Structure
8. Data Flow & Processing
9. Working Mechanism
10. Output and Visualization
11. Security Considerations
12. Conclusion
13. Future Enhancements

**INTRODUCTION:**

Weather forecasting is essential in our daily lives, impacting everything from travel plans to agricultural activities and event scheduling. Having instant access to accurate weather updates helps people make informed decisions quickly. To address this need, we developed the Weather Voice Assistant—a smart, user-friendly application built using Python.

This assistant allows users to get real-time weather updates simply by speaking. By fetching the latest data from **OpenWeatherMap**, it delivers weather information in both text and audio formats, making it accessible and convenient. Additionally, it generates 5-day weather forecast graphs, providing not just immediate updates but also a visual overview of upcoming weather trends.

Whether you're planning a trip, managing crops, or organizing an outdoor event, this assistant offers a seamless way to stay informed about the weather—just ask, and it responds!

**PROBLEM STATEMENT:**

While numerous mobile weather apps are available, many fall short in key areas:

* Limited or no support for hands-free interaction
* Absence of engaging, two-way communication
* Lack of visual tools to present trends like precipitation probability or humidity levels
* Inaccessibility for users with visual or physical impairments due to missing voice interaction features

**Proposed Solution: The Weather Voice Assistant**

This system aims to overcome the above limitations by:

* Incorporating speech recognition to understand and respond to spoken queries
* Retrieving and interpreting real-time weather data
* Providing spoken feedback using text-to-speech technology
* Presenting weather trends, such as rain chances and humidity, through interactive visualizations

**OBJECTIVES:**

* Develop a voice-enabled interface to allow users to access weather information through speech
* Retrieve both current conditions and future forecasts using the OpenWeatherMap API
* Generate spoken reports summarizing key weather metrics
* Visualize weather trends such as temperature changes, humidity levels, and rainfall probability
* Design the system to function reliably in various usage scenarios and environments

**REAL-LIFE USE CASE EXAMPLE:**

**Scenario 1: Hands-Free Weather Check While Cooking**

Situation: A user is preparing a meal in the kitchen and wants to know the outside weather without touching their phone.

Interaction:

User: "Current weather."

Assistant: "Which city?"

User: "Delhi."

Assistant: "The current weather in Delhi is clear with a temperature of 32°C, feels like 34°C. Humidity is 45%, and wind speed is 5.2 m/s."

Benefit:

No need to pause cooking or handle devices—just ask and get an instant spoken response.

**Scenario 2: Checking Weather Before a Trip**

Situation: A user is planning a trip and wants to check upcoming weather conditions at their destination.

Interaction:

User: "Forecast for Mumbai."

Assistant: "Fetching the 5-day forecast for Mumbai..."

(Displays a graph showing temperature trends and rain probability)

Helps in packing appropriately and planning activities based on weather trends.

**TECHNOLOGIES USED:**

**Technology Purpose**

**Python** Core logic and scripting

**speech\_recognition** Captures and converts voice to text

**pyttsx3** Converts text responses into speech

**OpenWeatherMap API** Weather Data Provider

**requests** Handles API HTTP requests

**matplotlib** Visualizes weather data

**pandas** Data handling and summarization

**datetime** Manages date-time formatting

**SYSTEM ARCHITECTURE:**

**[User Voice]**

**↓**

**[speech\_recognition]**

**↓**

**[Chatbot Logic]**

**↓**

**[WeatherForecaster Class]**

**↓**

**[API Request → OpenWeatherMap]**

**↓**

**[Response → Data Processing]**

**↓**

**→ [Text Summary]**

**→ [Audio Feedback (pyttsx3)]**

**→ [Forecast Graph (matplotlib)]**

**PROJECT STRUCTURE:**

**WeatherForecaster class: Core class that handles weather data fetching and processing.**

get\_current\_weather(city\_name): Retrieves live weather.

get\_forecast(city\_name): Retrieves 5-day forecast.

\_process\_current\_weather(data): Extracts and formats current weather information.

\_process\_forecast(data): Organizes and prepares forecast data for display and visualization.

display\_current\_weather(data): Displays structured current weather details.

plot\_forecast(data): Visualizes trends using bar and line plots.

**Voice Functions**

speak(text): Speaks out assistant responses.

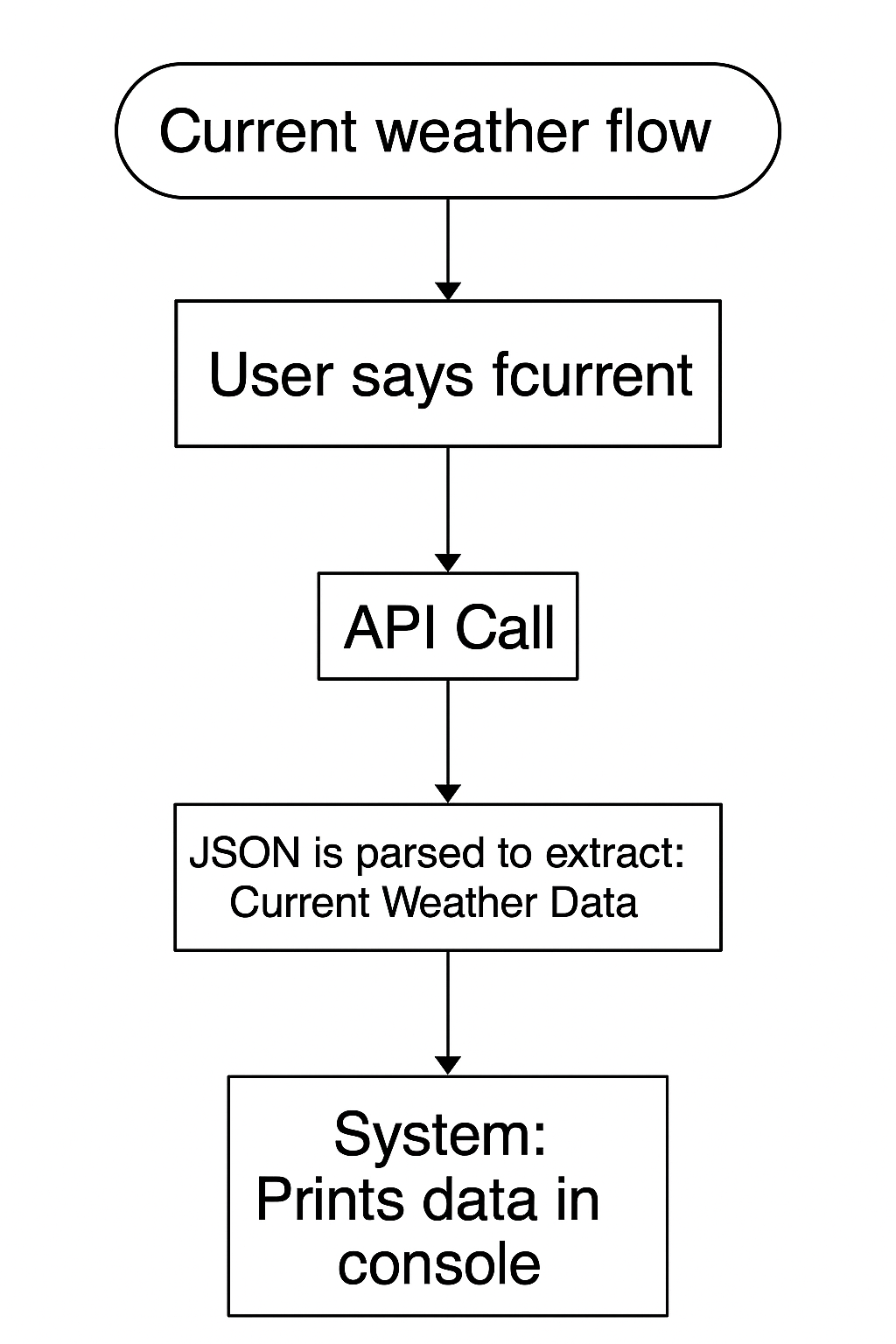
listen(): Records user voice and converts it to text.

**Main Flow**

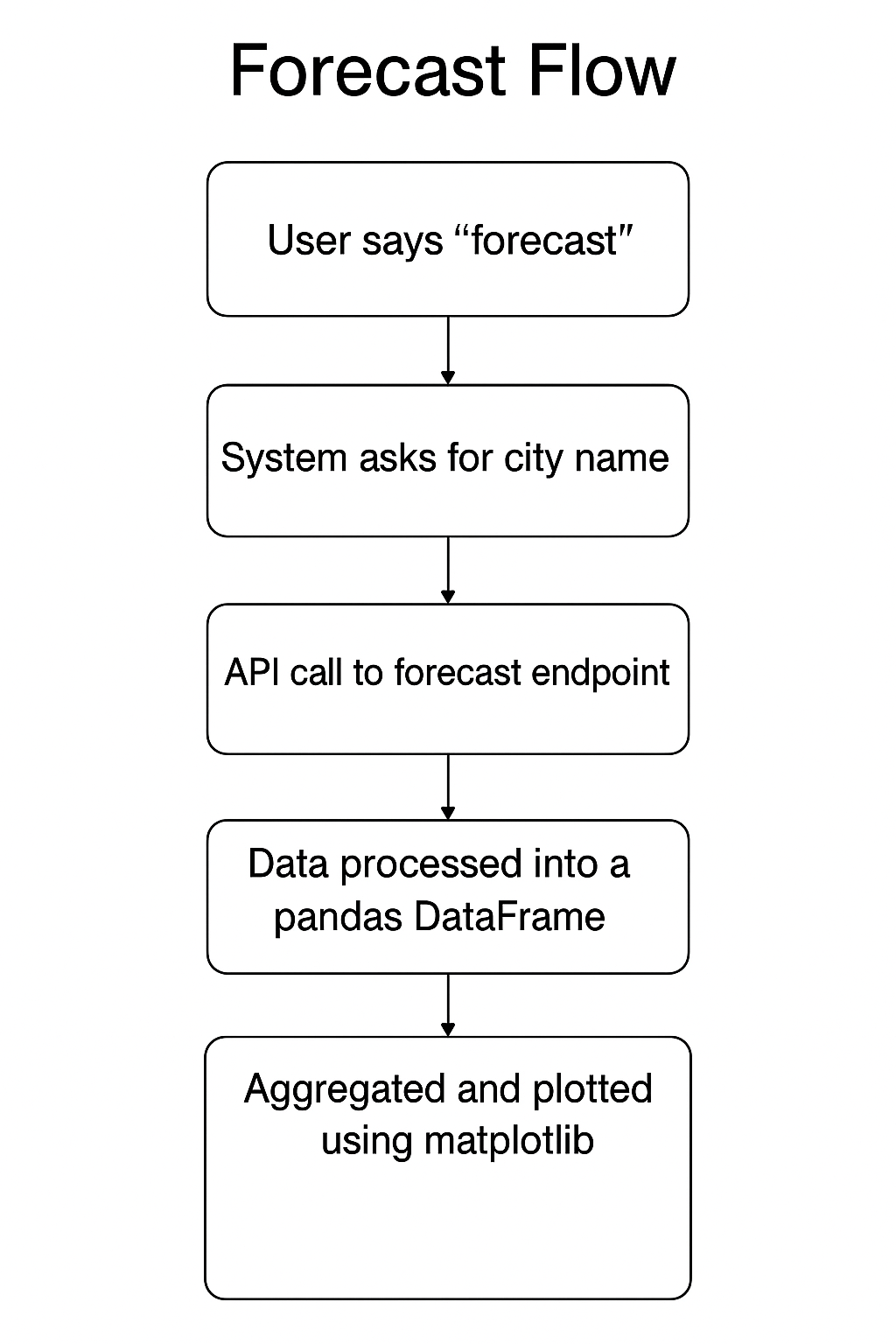
chatbot(): Controls assistant's decision-making based on user commands.

**DATA FLOW AND PROCESSING:**

* **CURRENT WEATHER DATA FLOW:**



* **WEATHER FORECASR DATA:**



**Working Mechanism:**

The Weather Voice Assistant follows a continuous loop of listening, interpreting, and responding to user voice commands. Here's how it works:

**Initialization & Greeting**

The assistant starts by greeting the user with spoken instructions using pyttsx3, outlining available commands: “current”, “forecast”, or “exit”.

**Listening for Commands**

Using speech\_recognition, the assistant listens to voice input and interprets it using Google’s speech-to-text engine.

**Processing Input**

If the command is "current", it prompts for a city, fetches live weather data from OpenWeatherMap, displays the results, and speaks a summary.

If "forecast", it prompts for a city, retrieves the 5-day forecast, generates a temperature/humidity plot using matplotlib.

If "exit", the assistant thanks the user and terminates.

**Error Handling**

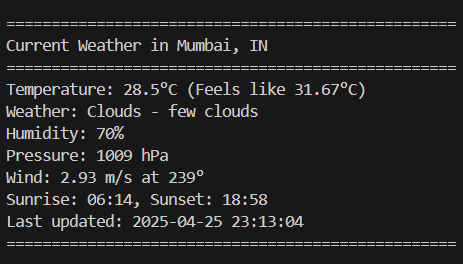
The assistant handles unrecognized speech or failed API responses gracefully by prompting the user to try again.

**Looping**

The assistant continues to listen and respond until the user says “exit”.

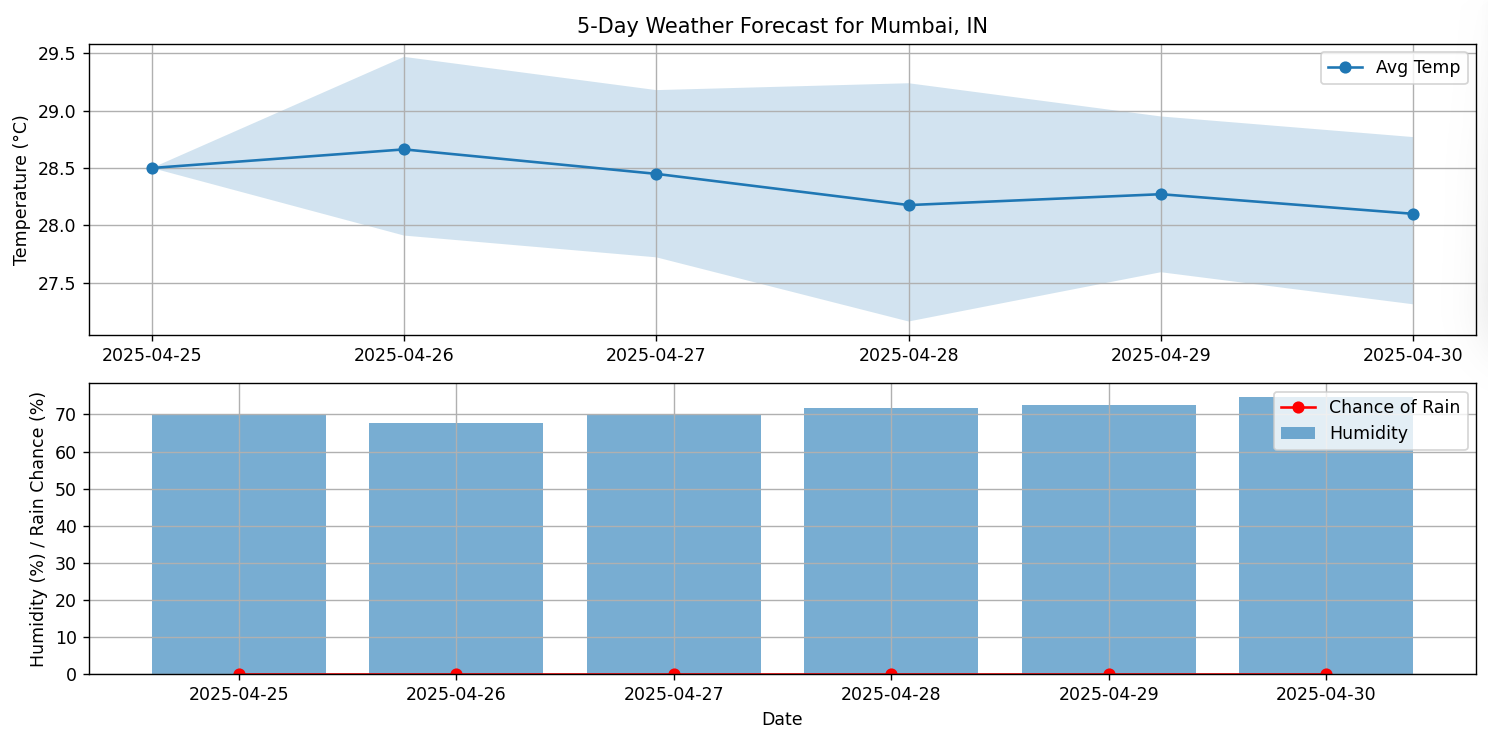
**OUTPUT AND VISUALIZATION:**

**Example: Current Weather of Mumbai**



**Voice Output:**

The current weather in Mumbai, IN is Clouds with few clouds. The temperature is 28.5 degrees Celsius, but feels like 31.67 degrees. Humidity is at 70 percent, wind speed is 2.93 meters per second. Sunrise is at 06:14 and sunset is at 18:58.

**5 Days Weather Forecast of Mumbai:**

**SECURITY CONSIDERATIONS:**

* **Local Processing:** All voice and API requests are processed locally. No sensitive data is transmitted externally beyond the OpenWeatherMap API call.
* **Error Handling:** The assistant handles errors such as network issues, invalid cities, and voice recognition failures gracefully.
* **Voice Input Privacy:** Microphone access is initiated only during listening and not continuously active.
* **API Security:** While the API key is hardcoded here, it can be stored securely using environment variables or config files in production.

**CONCLUSION:**

The Weather Voice Assistant is an innovative and user-friendly application that merges artificial intelligence with data visualization, making it both practical and accessible. It showcases the power of voice-controlled technology in everyday scenarios, allowing users to interact effortlessly with weather data.

This project goes beyond just convenience—it fosters inclusivity by offering a hands-free, easy-to-use interface that caters to individuals with varying needs. As a starting point, it lays the groundwork for future smart assistants, with the potential to expand and include even more advanced features.

**FUTURE ENHANCEMENTS:**

Multilingual Support: Extend voice input and output to other languages.

Weather Alerts: Notify users of storms or severe conditions.

Smart Device Integration: Connect with Alexa or Google Home.

Location-Based Auto Detection: Use geolocation to auto-fetch weather without city name.

User Interface: Add GUI with buttons and real-time animations.

Data Caching: Store previously fetched data to reduce API usage.

Logging System: Maintain a history of voice commands and results.

**Special Thanks**

I would like to sincerely thank the following for their invaluable support throughout this project:

Dr. Ishan Kumar – For his academic guidance and mentorship.

Open-source Developers – For providing essential libraries and compilers that supported the development.

Friends and Testers – For offering diverse test cases and feedback, enhancing the system’s performance.

The Python Community – For the extensive documentation and modules that enabled optimization of the project.