## **WeatherWise Application – Code Explanation**

### **1. Importing Required Libraries**

python

CopyEdit

import os

import requests

import matplotlib.pyplot as plt

import ipywidgets as widgets

from IPython.display import display, clear\_output

**Explanation:**

* os: Helps access environment variables (used in API keys if needed).
* requests: Sends HTTP requests to fetch weather data from APIs.
* matplotlib.pyplot: Used to create temperature and precipitation graphs.
* ipywidgets: Creates interactive UI components in Jupyter/Colab.
* IPython.display: Handles display and clearing of dynamic outputs.

### **2. get\_weather\_data() – Fetch Weather from wttr.in**

python

CopyEdit

def get\_weather\_data(location, forecast\_days=5):

url = f"https://wttr.in/{location}?format=j1"

...

**Explanation:**

* Constructs a URL to get weather info in JSON format from [wttr.in](https://wttr.in).
* Tries to send a GET request and return the parsed JSON.
* Handles any errors by returning a readable message.

### **3. parse\_weather\_question() – Understand the User’s Question**

python

CopyEdit

def parse\_weather\_question(question):

...

**Explanation:**

* Analyzes the user's natural language question.
* Extracts the **weather attribute** (e.g., temperature or precipitation).
* Detects the **location** mentioned after the word "in".

### **4. generate\_weather\_response() – AI-style Answer**

python

CopyEdit

def generate\_weather\_response(parsed\_question, weather\_data):

...

**Explanation:**

* Uses the parsed question and weather data to give a friendly AI-style response.
* Responds differently based on whether the user asked about temperature, rain, or general weather.

### **5. create\_temperature\_visualisation() – Graph of Temperatures**

python

CopyEdit

def create\_temperature\_visualisation(weather\_data):

...

**Explanation:**

* Extracts average daily temperatures from the forecast.
* Creates a line chart using matplotlib to show temperature trends across days.

### **6. create\_precipitation\_visualisation() – Graph of Rainfall**

python

CopyEdit

def create\_precipitation\_visualisation(weather\_data):

...

**Explanation:**

* Extracts precipitation values from hourly forecasts.
* Builds a bar chart showing rainfall amounts over the forecast period.

### **7. launch\_weatherwise\_interface() – Build the Interactive UI**

python

CopyEdit

def launch\_weatherwise\_interface():

...

**Explanation:** This function sets up and launches the **main interactive interface** using ipywidgets.

#### **UI Components:**

* location\_input: Text input for the city.
* forecast\_input: Numeric input for selecting forecast days (1 to 5).
* question\_input: Text box for user to ask natural language weather questions.
* fetch\_button: Button to fetch weather data.
* temp\_button: Button to display the temperature graph.
* rain\_button: Button to display the rainfall graph.
* ai\_button: Button to analyze the question and generate a human-like response.

#### **Internal Functions:**

* on\_fetch\_clicked: Fetches weather data when the "Fetch" button is clicked.
* on\_temp\_clicked: Shows the temperature chart.
* on\_rain\_clicked: Shows the precipitation chart.
* on\_ai\_clicked: Responds to the weather question in natural language.

#### **Display Area:**

* output: An area used to display dynamic text, responses, and graphs.

### **8. Run the App**

python

CopyEdit

launch\_weatherwise\_interface()

**Explanation:**

* This line launches the whole WeatherWise system in your notebook.
* The user can now interact through buttons, input text, and get AI-style weather answers.

## **Summary**

This notebook demonstrates:

* **API integration** with real-time data (from wttr.in).
* **Natural language understanding** using simple parsing logic.
* **AI-enhanced interactivity** with response generation.
* **Data visualization** with clear, labelled graphs.
* **Interactive design** using ipywidgets to create an intuitive dashboard.