Task: Build a Smart Weather Forecasting AI Agent

📝 Objective

Design and implement an **AI-powered weather agent** capable of understanding user queries, generating weather forecasts, and presenting the output in both **tabular** and **chart** formats based on user instructions.

🎯 Key Goals

Your AI Weather Agent should:

1. **Ingest and process historical weather data** (CSV or API-based).
2. **Forecast weather** for upcoming days (using ML/statistical methods).
3. **Respond to natural language questions**, e.g.:
    - "What's the weather like for the next 3 days in Mumbai or any city or country?"
    - "Show me the temperature forecast for this week as a chart."
    - "Summarize last week's weather in table format."
4. **Render results** in:
    - Natural language (text)
    - **Tables** (e.g., temperature, humidity by day)
    - **Charts** (line/bar chart of forecast data)

🛠️ Functional Requirements

1. **Data Ingestion**
    - Use publicly available historical weather data (e.g., NOAA, Kaggle, or CSV file provided).
    - Optional: Connect to a live weather API like OpenWeatherMap.
2. **Forecasting**
    - Forecast weather conditions (temperature, humidity, etc.) for the next few days.
    - You may use simple statistical models (e.g., moving average) or ML models (e.g., linear regression, Prophet, etc.).
3. **Query Handling**

    - Agent should understand user queries and determine:
        - Location (if applicable)
        - Duration (e.g., next 3 days, last 7 days)
        - Output format (chart, table, text)
4. **Output Presentation**
    - Text: Descriptive summary.
    - Table: Use pandas/HTML tables to show data.
    - Chart: Use matplotlib or plotly to generate forecast charts dynamically.

📦 Tech Stack (Recommended)

You are free to choose the tools, but here's a suggested stack:

- **Backend**: Python with FastAPI or Flask.
- **Data**: CSV, Pandas
- **Forecasting**: scikit-learn, Prophet, statsmodels, or any ML/time-series package
- **Visualization**: matplotlib, seaborn, or plotly
- **Optional LLM-based** Query Understanding: transformers, langchain, etc.

✅ Deliverables

1. ✅ Codebase with instructions to run locally.
2. ✅ Sample dataset or API integration.
3. ✅ API or CLI to test the agent's responses.
4. ✅ Sample queries and outputs:
    - Sample natural language questions.
    - Corresponding outputs: chart images, tables, or text.

💡 Bonus (Optional)

- 🌍 Multi-city support.
- 📈 Streamed/chart-first responses.
- 🔑 Add simple authentication (API key or JWT).
- 🧠 Use LLM for parsing query intent (e.g., HuggingFace models).

- 🌐 Basic web interface to test the agent.

📅 Timeline

- Expected Time: 2–3 days.
- Please ensure that the submission is clean, documented, and easy to run.