ChatGPT

Weather Forecast App (Python + Streamlit)

This guide shows how to build a **browser-based weather website** using only Python and Streamlit (no HTML/CSS). We'll fetch data from OpenWeatherMap (free up to 1,000 calls/day [1]) to display the current temperature, wind speed, weather condition, and a multi-day forecast for selected cities. The app uses only Streamlit's Python API (widgets like `st.columns`, `st.metric`, `st.text_input`, `st.selectbox`, charts, emojis, status messages, etc.) to create a clean UI/UX. Key Streamlit features we'll use include side-by-side layouts with `st.columns` [2], metric displays with `st.metric` [3], and charts (`st.line_chart`/`st.bar_chart`) for forecasts [4]. We'll also handle errors (e.g. city not found) gracefully and secure our API key via Streamlit secrets. Finally, we'll deploy the app for free on Streamlit Community Cloud.

Setup & Installation

1. **Python & Libraries** – Install Python 3.8+ (Streamlit runs on supported Python versions). In your project folder, run:

```
pip install streamlit requests pandas
```

   This installs Streamlit (for the web UI), `requests` (to call the weather API), and `pandas` (optional, for data handling or charts).
   (You can also create a virtual environment before installing.)

2. **OpenWeatherMap API Key** – Sign up for a free account on OpenWeatherMap (e.g. openweathermap.org) and generate an API key under your Dashboard > API keys [5]. Copy the key; you'll need it to fetch weather data.

3. **Secure the API Key** – Create a directory named `.streamlit` in your project. Inside it, create a file `secrets.toml` and store the key like this:

```
OPENWEATHER_API_KEY = "YOUR_API_KEY_HERE"
```

   (Replace with your actual key.) Streamlit will automatically load this file, and you can access the key in code with `st.secrets["OPENWEATHER_API_KEY"]` [6] [7]. This keeps the key out of your main code. *Do not commit* `secrets.toml` *to public repos!*

Writing the App Code

Create a Python file, e.g. `weather_app.py`, and follow these steps:

1. **Imports and Title:**

```python
import streamlit as st
import requests

st.title("🌥 Weather Forecast App")
```

   We import Streamlit and Requests, then set a title. We added an emoji (🌥) in the title for flair. Streamlit allows text/Markdown with emojis/icons in titles and labels.

2. **City Selection UI:** Use a dropdown of predefined cities and a text input for custom city. For example:

```python
cities = ["Delhi", "Jaipur (Rajasthan)", "Ahmedabad (Gujarat)"]
default_city = st.selectbox("Select a city:", cities)
custom_city = st.text_input("Or enter a city name:")
if custom_city:
    city = custom_city
else:
    city = default_city
```

   This shows a `st.selectbox` for quick choices and a `st.text_input` for any city name. The code uses the custom input if provided, otherwise the selected default. Streamlit updates this in real time.

3. **Fetch Current Weather:** Call OpenWeatherMap's current weather endpoint using the stored API key:

```python
API_KEY = st.secrets["OPENWEATHER_API_KEY"]
url_current = f"http://api.openweathermap.org/data/2.5/weather?q={city}
&units=metric&appid={API_KEY}"
resp = requests.get(url_current)
if resp.status_code == 200:
    data = resp.json()
    temp = data["main"]["temp"]
    wind = data["wind"]["speed"]
    desc = data["weather"][0]["description"].title()
    icon = data["weather"][0]["icon"]
    icon_url = f"http://openweathermap.org/img/wn/{icon}@2x.png"
    # Display metrics and icon in two columns
    col1, col2 = st.columns(2)
    with col1:
        st.metric("🌡 Temperature (°C)", f"{temp:.1f}", help="Current
temperature")
        st.metric("  Wind Speed (m/s)", f"{wind:.1f}", help="Current wind
speed")
    with col2:
        st.image(icon_url, width=100)
        st.write(f"**Condition:** {desc}")
else:
    st.error("  City not found. Please enter a valid city name.")  # Error
handling
```

4. We use `st.secrets["OPENWEATHER_API_KEY"]` to get the key [7].
5. The current weather JSON contains temperature, wind, and a description. We display temperature and wind using `st.metric` in one column and the weather icon + description in another. The `st.columns(2)` call creates two side-by-side containers [2]. Inside each `with` block, we add content to that column. For example, `st.metric("🌡 Temperature", f"{temp}°C")` shows a large number with a label [3] [8]. We added emojis (🌡,) to labels for a friendlier UI.
6. If the API returns an error (e.g. invalid city), we show an error message with `st.error()` [9]. In our code, a non-200 status triggers `st.error("City not found…")`.
7. **Multi-Day Forecast Chart:** After fetching current data, we can also fetch a 5-day forecast (3-hour interval) and plot daily trends:

```
url_forecast = f"http://api.openweathermap.org/data/2.5/forecast?q={city}
&units=metric&appid={API_KEY}"
resp2 = requests.get(url_forecast)
if resp2.status_code == 200:
    forecast_json = resp2.json()
    dates, temps = [], []
    for entry in forecast_json["list"]:
        dt = entry["dt_txt"]  # e.g. '2025-08-01 12:00:00'
        if "12:00:00" in dt:  # take midday data for each day
            dates.append(dt.split(" ")[0])
            temps.append(entry["main"]["temp"])
    st.subheader("Forecast (Daily)")
    st.line_chart(data={"Temp": temps}, x=dates)  # simple line chart
    # Alternatively, use st.bar_chart(temps) or allow user to choose chart
type
```

Here we collect one temperature per day (at noon) and use `st.line_chart` to plot it. Streamlit can take a dict or Pandas data. This will render an interactive chart. (You could also use `st.bar_chart` to create a bar chart of the temperatures [4].) This example gives a simple multi-day trend.

8. **UI/UX Enhancements:** Streamlit widgets handle layout automatically, so we don't write any HTML/CSS. We use headings (`st.subheader`), separators (`st.write("---")`), emojis (in text), and status messages (`st.info`, `st.warning`, etc.) to make the interface friendly. For example, you might add `st.info("Data provided by OpenWeatherMap")` or use `st.success` when data loads. The key is to leverage Streamlit's built-in styling (e.g. markdown, icons) to improve UX.

Example Code Snippet

Below is a condensed example combining the above components (comments added for clarity):

```
import streamlit as st
import requests

st.title("⛅ Weather Forecast App")

# Predefined cities and custom input
cities = ["Delhi", "Jaipur (Rajasthan)", "Ahmedabad (Gujarat)"]
default_city = st.selectbox("Select a city:", cities)
custom_city = st.text_input("Or enter a city name:")
city = custom_city or default_city

# Fetch current weather
API_KEY = st.secrets["OPENWEATHER_API_KEY"]
url = f"http://api.openweathermap.org/data/2.5/weather?q={city}
&units=metric&appid={API_KEY}"
res = requests.get(url)
if res.status_code == 200:
    data = res.json()
    temp = data["main"]["temp"]
    wind = data["wind"]["speed"]
    desc = data["weather"][0]["description"].title()
    icon = data["weather"][0]["icon"]
    icon_url = f"http://openweathermap.org/img/wn/{icon}@2x.png"

    # Display side-by-side columns with metrics and icon
    col1, col2 = st.columns(2)
    with col1:
        st.metric("🌡 Temperature (°C)", f"{temp:.1f}", help="Current
temperature")
        st.metric("  Wind Speed (m/s)", f"{wind:.1f}", help="Current wind
speed")
    with col2:
        st.image(icon_url, width=100)
        st.write(f"**Condition:** {desc}")

    # Multi-day forecast (5-day)
    url2 = f"http://api.openweathermap.org/data/2.5/forecast?q={city}
&units=metric&appid={API_KEY}"
    res2 = requests.get(url2)
    if res2.status_code == 200:
        forecast = res2.json()
        dates, temps = [], []
        for entry in forecast["list"]:
            if "12:00:00" in entry["dt_txt"]:
                dates.append(entry["dt_txt"].split(" ")[0])
                temps.append(entry["main"]["temp"])
        st.subheader("3-Day Forecast")
        st.line_chart(data={"Temp": temps}, x=dates)
    else:
        st.info("Forecast data not available.")
else:
    st.error("  City not found. Please try another name.")
```

This snippet demonstrates key parts: widgets for input, API calls with `requests`, data parsing, and Streamlit output components (`st.columns`, `st.metric`, `st.image`, `st.line_chart`, `st.error`) [8] [4]. Notice how no HTML/CSS is written – all layout and styling come from Streamlit's Python API.

requirements.txt Example

Create a `requirements.txt` file listing the dependencies, for example:

```
streamlit
requests
pandas
```

(You can generate this with `pip freeze > requirements.txt` or write it manually. The Streamlit cloud will use this to install packages [10].)

Testing Locally

To test the app on your computer, run:

```
streamlit run weather_app.py
```

This will launch the app in your browser [11]. Enter different city names or select from the dropdown to see the weather data. Ensure your `.streamlit/secrets.toml` file is present locally with your API key.

Deployment on Streamlit Community Cloud

1. **Push to GitHub:** Commit your `weather_app.py`, `requirements.txt`, and optionally a placeholder `secrets.toml` (but *do not* commit the real API key).
2. **Create a Streamlit Cloud app:** Go to share.streamlit.io, log in, and click **"Create app"**. Choose the GitHub repo and branch, and specify the file path (e.g. `weather_app.py`) [10] .
3. **Configure Secrets:** In the "Advanced settings" of the app creation dialog, paste the contents of your local `secrets.toml` (or just the line with your key) into the "Secrets" box [12] . This securely provides the API key to the cloud instance.
4. **Deploy:** Click **Deploy**. Streamlit will install the dependencies and launch your app. Once done, you'll get a unique `*.streamlit.app` URL. Your app is now live and accessible for free!

After deployment, any new commits to the repo (with the same file paths) will update the app automatically. You can manage the app (set a custom subdomain, etc.) in the Streamlit Cloud dashboard.

Summary

In this project we built a fully Python/Streamlit weather app. We used `st.selectbox` and `st.text_input` for user input, `st.columns` and `st.metric` for a responsive layout of key stats [2] [3] , and `st.line_chart` for the forecast [4] . The OpenWeatherMap API key is kept secret via Streamlit's `secrets.toml` mechanism [6] , and we handled errors (invalid city) by showing an `st.error` message [9] . Finally, we deployed the app on Streamlit Community Cloud for free [10] [12] . No HTML or CSS was written—Streamlit's native widgets and theming created the UI. The result is a simple, user-friendly weather site that runs in the browser and is easy to extend or customize.

Sources: Streamlit documentation and tutorials for `st.columns` and `st.metric` [2] [3] ; OpenWeatherMap API docs [1] ; Streamlit Community Cloud deployment guide [10] [12] ; community examples using Streamlit for weather apps [4] [5] .

---

[1] Weather API - OpenWeatherMap
https://openweathermap.org/api

[2] [8] [9] [11] Lets create a simple weather App 🌧- Part 1 - DEV Community
https://dev.to/engineeredsoul/lets-create-a-simple-weather-app-part-1-5147

[3] st.metric - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.metric

[4] How I Built a Weather App with Python Streamlit – Be on the Right Side of Change
https://blog.finxter.com/how-i-built-a-weather-app-with-python-streamlit/

[5] Weather Forecasting using Streamlit, OpenAI and OpenWeatherMap to explore LangChain | by Blessin Varkey | Medium
https://medium.com/@contact.blessin/weather-forecasting-using-streamlit-openai-and-openweathermap-to-explore-langchain-ca526f1c0ca1

[6] [7] secrets.toml - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/connections/secrets.toml

10 12 Deploy your app on Community Cloud - Streamlit Docs
https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app/deploy