Project Structure

```
WeatherTrendPredictor/
│
├── data/
│   ├── raw_weather.csv
│   └── processed_weather.csv
│
├── src/
│   ├── fetch_data.py       # Get weather data
│   ├── process_data.py     # Clean & enrich
│   └── predict_weather.py  # Model & forecast
│
├── main.py              # Orchestrates the pipeline
├── requirements.txt       # Dependencies
└── README.md            # Instructions & API setup
```

Requirements.txt

```
requests

pandas

numpy

scikit-learn

statsmodels

python-dotenv
```

fetch_data.py

```
# src/fetch_data.py

import os, requests, time, datetime

import pandas as pd

from dotenv import load_dotenv
```

```
load_dotenv()
API_KEY = os.getenv("OPENWEATHER_API_KEY")
LOCATION = os.getenv("LOCATION", "Chattogram,Bangladesh")

def fetch_last_90_days():
    end = datetime.datetime.utcnow()
    start = end - datetime.timedelta(days=90)
    records = []

    for day in pd.date_range(start=start, end=end, freq='D'):
        ts = int(day.replace(hour=12, minute=0).timestamp())
        resp = requests.get(
            "https://api.openweathermap.org/data/2.5/onecall/timemachine",
            params=/span>
                "lat": os.getenv("LAT"),
                "lon": os.getenv("LON"),
                "dt": ts,
                "appid": API_KEY,
                "units": "metric"
            }
        )
        data = resp.json()
        if 'current' in data:
            rec = {
                "date": day.date(),
                "temp": data['current']['temp'],
```

```
            "humidity": data['current']['humidity'],
            "pressure": data['current']['pressure']
        }
        records.append(rec)
    time.sleep(1)  # respect rate limit

df = pd.DataFrame(records)
df.to_csv('data/raw_weather.csv', index=False)
return df
```