US Weather and Energy Analysis Project Plan

Project Goal
Develop an automated pipeline that analyzes how weather impacts electricity usage across five major US cities, providing valuable insights for energy companies.

Business Impact
Problem: Inaccurate demand forecasting causes millions in losses for energy companies.
Solution: Integrate weather and energy data to drive more accurate predictions.
Value: Better forecasting, reduced waste, optimized power generation, cost savings.

System Overview
Daily Trigger via Cron or Scheduler
→ Part 1: Data Collection
→ Part 2: Data Cleaning and Quality Assurance
→ Part 3: Statistical Analysis
→ Part 4: Streamlit Visualization Dashboard

Part 1: Data Collection

Objective
Automate the collection of weather and energy usage data daily for 5 cities using public APIs.

You Will Build

src/data_fetcher.py: Python module to collect weather and energy data

src/pipeline.py: Entry script to run the entire ETL process

config/config.yaml: Configuration file storing API keys, station IDs, and region codes

backfill_historical.py: Script to collect 90 days of historical data

Required APIs

NOAA Climate Data API: https://www.ncei.noaa.gov/cdo-web/api/v2

EIA Electricity Data API: https://api.eia.gov/v2/electricity/

NOAA Token: https://www.ncdc.noaa.gov/cdo-web/token

EIA Token: https://www.eia.gov/opendata/register.php

City Reference Table

City	State	NOAA Station ID	EIA Region Code
New York	New York	GHCND:USW00094728	NYIS
Chicago	Illinois	GHCND:USW00094846	PJM
Houston	Texas	GHCND:USW00012960	ERCO
Phoenix	Arizona	GHCND:USW00023183	AZPS
Seattle	Washington	GHCND:USW00024233	SCL

Key Requirements

Handle API failures with logging and continue execution

Implement retry logic with exponential backoff

Store weather and energy data separately in data/raw/

Include UTC timestamps in saved records

Store each day’s file with naming format: weather_YYYYMMDD.json and energy_YYYYMMDD.json

Sample Output Structure (JSON)

weather_20240115.json
[
{ "date": "2024-01-15", "city": "New York", "tmax_f": 45.2, "tmin_f": 32.1, "timestamp_utc": "2024-01-15T12:00:00Z" }]

energy_20240115.json
[
{ "date": "2024-01-15", "region": "NYIS", "demand_mwh": 25847.5, "timestamp_utc": "2024-01-15T12:00:00Z" }]

Part 2: Data Quality Assurance

Objective
Ensure data is clean, consistent, and usable for statistical analysis and visualization.

You Will Build

src/data_processor.py: Cleans and merges data

src/quality_checks.py: Contains quality check logic

Daily quality report written to data/processed/

Checks Performed

Missing data detection

Outlier detection (temps < -50°F or > 130°F, negative demand)

Staleness check: Flag records older than 48 hours

Synchronization: Ensure each city has data for the same date

Processing Steps

Convert temperature from tenths of Celsius to Fahrenheit

Merge weather and energy datasets using city/region-date as key

Generate and store data quality metrics and flags

Output Format

data/processed/merged_20240115.parquet
Columns: date, city, tmax_f, tmin_f, demand_mwh, is_outlier, data_quality_score

---

Part 3: Statistical Analysis

Objective
Uncover trends, patterns, and correlations between temperature and electricity usage.

You Will Build

src/analysis.py: Central module for correlation and pattern discovery

Correlation analysis: Compute Pearson correlation and R-squared

Temporal trend analysis: Weekly and seasonal trends

Heatmap dataset preparation: Temperature vs weekday usage mapping

Analytical Goals

Determine correlation strength between temp and usage

Measure city-by-city seasonal differences

Detect weekend-vs-weekday usage trends

Prepare datasets for heatmap visualizations

Output Files

data/analytics/
correlations.json — Correlation coefficients and R-squared per city
timeseries.parquet — Joined daily time series for plotting
heatmap.parquet — Average usage grouped by temp range and day
summary_stats.json — Calculated insights and descriptive statistics

---

Part 4: Streamlit Visualization Dashboard

Objective
Create a fully interactive dashboard that lets users explore temperature vs electricity demand in real-time.

You Will Build
dashboards/app.py — Main Streamlit application
Includes 4 scrollable sections:

1. Geographic Overview
Interactive US map
Shows city name, temperature, energy usage, and daily change
Color-coded: Red = high usage, Green = low usage
Shows last data update timestamp

2. Time Series Analysis
Dual-axis line chart of temp and usage
Dropdown to select one city or all cities
Weekend shading
Legend, labeled axes, and smoothing trendlines

3. Correlation Analysis
Scatterplot of temp vs usage

One color per city
Regression line with R-squared and correlation shown
Hover tooltips for city, date, and values

4. Heatmap
Temp ranges on y-axis: <50°F, 50-60°F, 60-70°F, 70-80°F, 80-90°F, >90°F
Weekdays on x-axis
Colors from blue (low) to red (high usage)
Numeric values shown in cells
Filter by city

Dashboard Features
Sidebar filters for date range and city multiselect
Auto-refresh every hour
Mobile-compatible layout
Chart export as PNG

```
project1-energy-analysis/
│
├── README.md              ← Summary, setup steps, and business context
├── AI_USAGE.md            ← Documentation of AI tools, prompts, errors, and fixes
├── pyproject.toml         ← Dependency declarations for reproducibility
├── video_link.md          ← Paste your Loom or YouTube video URL here
│
├── config/                ← Configuration directory
│   └── config.yaml        ← Stores API keys, region codes, and city metadata
│
├── src/                   ← Core pipeline source code
│   ├── data_fetcher.py    ← Fetches weather and energy data from NOAA and EIA APIs
│   ├── data_processor.py  ← Cleans, transforms, and merges raw data
│   ├── quality_checks.py  ← Implements missing data, outlier, and freshness checks
│   ├── analysis.py        ← Performs correlation, trend, and heatmap analysis
│   └── pipeline.py        ← Main orchestration script that runs the whole workflow
│
├── dashboards/            ← Streamlit-based interactive visualizations
│   └── app.py             ← Streamlit dashboard application (with 4 views)
│
├── logs/                  ← Logs directory for execution and debugging
│   └── pipeline.log       ← Logs pipeline run results, errors, timestamps
│
├── data/                  ← Stores all intermediate and final data files
│   ├── raw/               ← Unprocessed API responses (JSON format)
```

```
│   │   ├── weather_YYYYMMDD.json
│   │   └── energy_YYYYMMDD.json
│   ├── processed/          ← Cleaned, merged datasets (Parquet format)
│   │   └── merged_YYYYMMDD.parquet
│   └── analytics/          ← Output of analysis and modeling
│       ├── correlations.json        ← Correlation coefficients and R-squared
│       ├── timeseries.parquet       ← Timeseries data for dashboard
│       ├── heatmap.parquet          ← Data matrix for heatmap view
│       └── summary_stats.json       ← Descriptive insights and statistical summaries
│
├── notebooks/              ← Optional exploratory analysis in notebook format
│   └── exploration.ipynb      ← Jupyter notebook for early data exploration
│
└── tests/                  ← Unit and integration tests
    └── test_pipeline.py       ← Tests key functions like fetching, processing, and analysis
```

Development Timeline (Weekly Breakdown)

Week 1

Connect to NOAA and EIA APIs

Fetch and store one city's data

Handle errors and logging

Run the backfill script

Week 2

Build data processor and cleaner

Implement all quality checks

Generate sample reports and parquet files

Week 3

Complete correlation and seasonal analysis

Prepare all aggregated datasets

Validate your stats with basic visual plots

Week 4

Build Streamlit app

Add all visualizations

Add sidebar filtering, export, and caching

Final code cleanup, testing, and documentation

Expected Insights

Correlation ($r > 0.7$) between extreme temperature and energy demand

High demand in Phoenix during summer

High demand in Seattle during winter

Weekday energy usage > weekend usage in all cities

Different cities peak at different temperature ranges

Video Presentation Script Template (3 minutes)

0:00 to 0:30 — Introduce the problem (energy forecasting losses)
0:30 to 2:00 — Walk through the pipeline, show live code
- API call
- Merged data
- Quality flags
- Streamlit dashboard
2:00 to 2:30 — Insights and surprises in data
2:30 to 3:00 — How AI helped you, bugs you fixed, what you'd improve

Requirements

Screen recording (Loom or YouTube)

Voice narration

Show actual code execution and dashboard in use

AI Usage Documentation (AI_USAGE.md)

What to include:

Which tools you used: ChatGPT, Gemini, Copilot, Claude, etc.

Best prompts and why they worked

Any bugs from AI-generated code and how you fixed them

What you learned from debugging AI output

Estimate of time saved due to AI involvement

Example Entry
Tool: ChatGPT
Prompt: "Write a function to fetch TMAX and TMIN from NOAA API with exponential backoff"
Bug: It missed the Celsius-to-Fahrenheit conversion
Fix: Manually added conversion and unit test
Time saved: About 2 hours of manual lookup and debugging

Getting Started Checklist

[ ] Clone or create GitHub repository

[ ] Get API keys for NOAA and EIA

[ ] Create config/config.yaml with keys and cities

[ ] Test API responses

[ ] Install dependencies

[ ] Fetch one day of data for one city

[ ] Run the full pipeline once successfully

Success Metrics

Technical

Pipeline runs on schedule daily

Logs errors, retries properly

Dashboard loads in under 5 seconds

Visualizations update with real data

Business

Clear insights generated

Actionable correlation patterns found

Project is well-documented and reusable

Video and repository are professional-grade