

# WEATHRU: Navigating through weather



**Developed by:-**

**Riyansha singh  
(18229CMP005)**

**Ruby Singh  
(18229CMP006)**

**Supervisors:-**

**Dr Rakhi Garg**

**Dr Sarvesh Pandey**

## **AIM**

To make a software that can be used to retrieve the real time as well as future weather updates along with air pollution levels of the desired location using efficient Weather APIs.

## **OBJECTIVE**

- Study about weather forecasting system and application
- Using Public APIs to get weather updates.
- Build a Data Engineering cycle(ETL pipeline) for the purpose of retrieving weather information and displaying result to the end users.

# Comparison With Similar Works

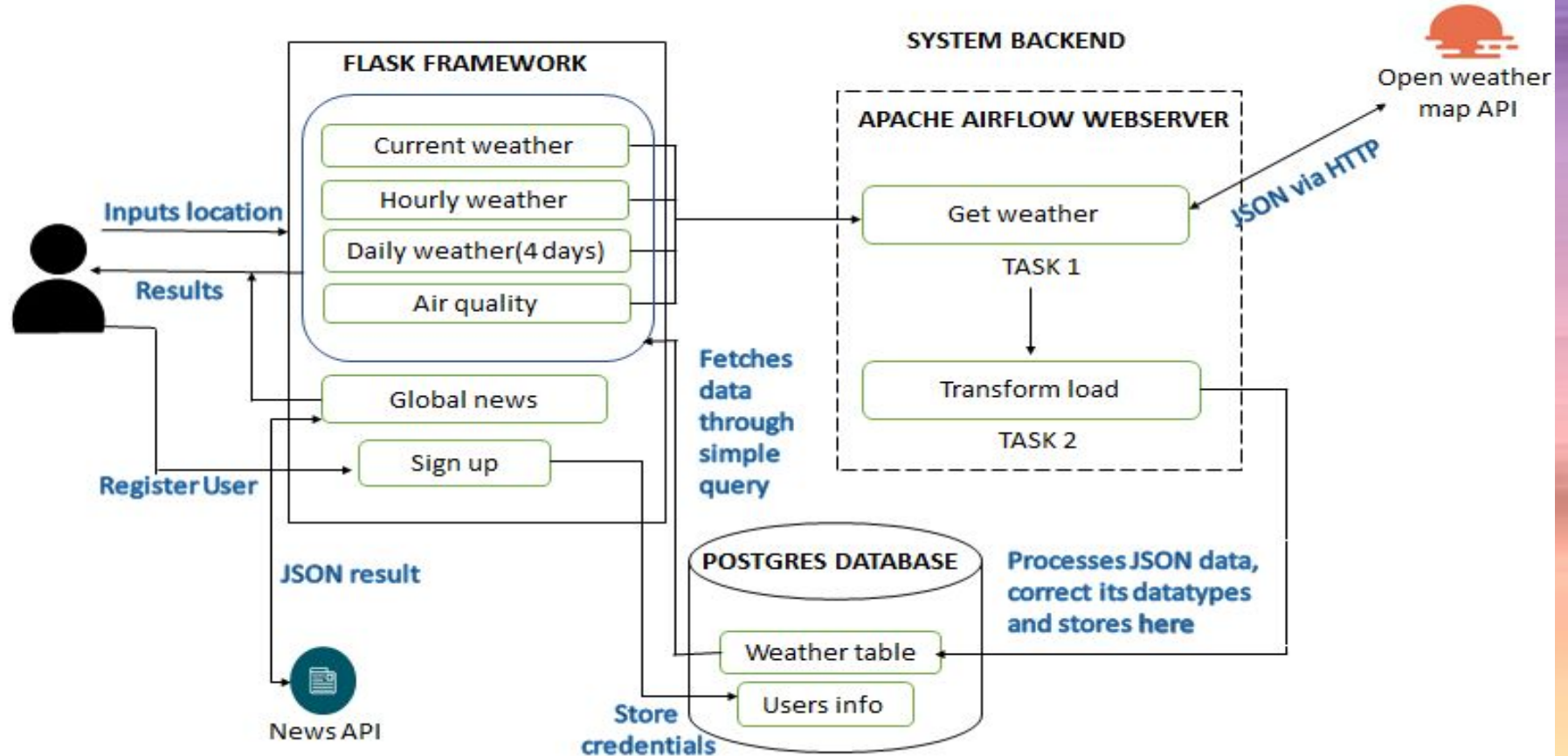
## **1. ACCUWEATHER**

- This free app is designed for “iPhone users”.
- It has a feature “iPhone Weather Station”, that allows to receive alerts for severe weather.
- Since India has mostly android users, so using ACCUWEATHER is meaningless and here is the solution WEATHRU
- Opposite to ACCUWEATHER’s crowded UI, WEATHRU is providing minimal & pleasant design.

## **2. DARK SKY**

- Accurate way of getting hyper local weather information.
- Hourly and daily data are updated every hour.
- Services are restricted over 13 years of age, WEATHRU doesn’t have any such restrictions.

# SYSTEM DESIGN



## Features of “WEATHRU”

WEATHRU provides all necessary updates of weather efficiently through an attractive GUI

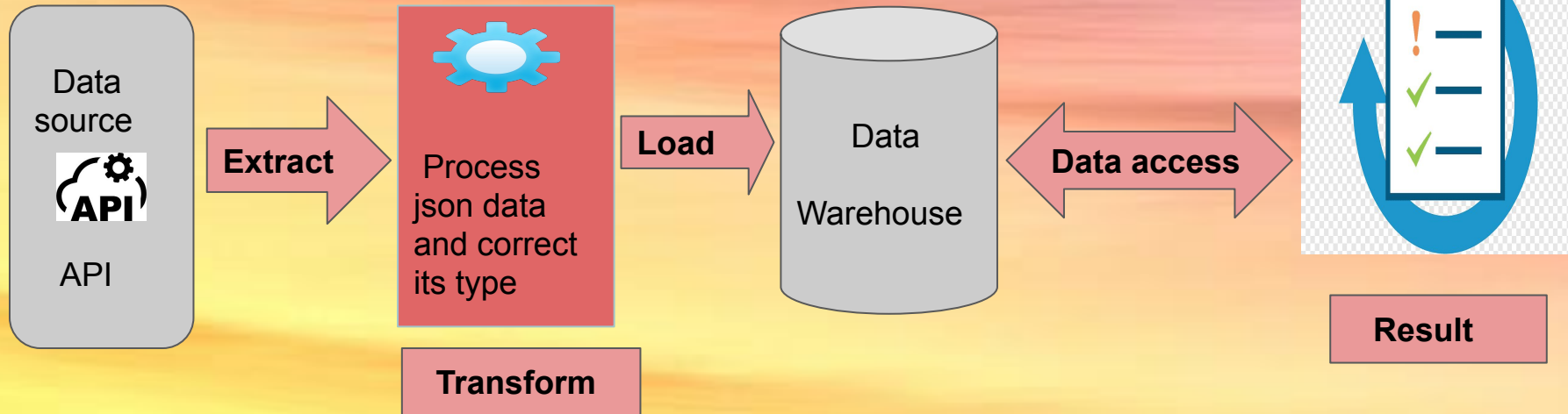
- Current weather (temperature, pressure, wind speed ,humidity)
- Hourly weather (at an interval of 4 hrs)
- Daily forecast (4 days)
- AQI Levels (pie chart illustration)
- News alerts
- User Signup
- Feedback page

# PRINCIPLE

## DATA ENGINEERING

It is a set of operations aimed at creating interfaces & mechanism for the flow and access of information.

The major part of our software designing is implementation of one system architecture to integrate all our work, using concepts of Data Engineering.





# Working with API

- API is a kind of messenger that delivers our request to the provider that one is requesting it from and then delivers the response back to us.
- They work by “exposing” some limited internal functions and features so that applications can share data without developers having direct access to behind-the-scenes code and their large databases.
- For getting our recent weather updates we used **Openweathermap API** as our provider as it is capable of making **100 free API calls per day that too with high accuracy.**

# Home page

← → ↻ localhost:5000

Apps Classes Gmail YouTube Maps Classes New Tab Fingertips Study Ap... DPS VARANASI

Reading

## Weathru

About News Contacts Login

TODAY HOURLY DAILY AIR QUALITY

Australia Search

Current weather


7.0°C

Feels like clear sky

↓ 4.0°C ↑ 8.0°C

Humidity 81.0%

Pressure 1023.0 mb





# Backend development

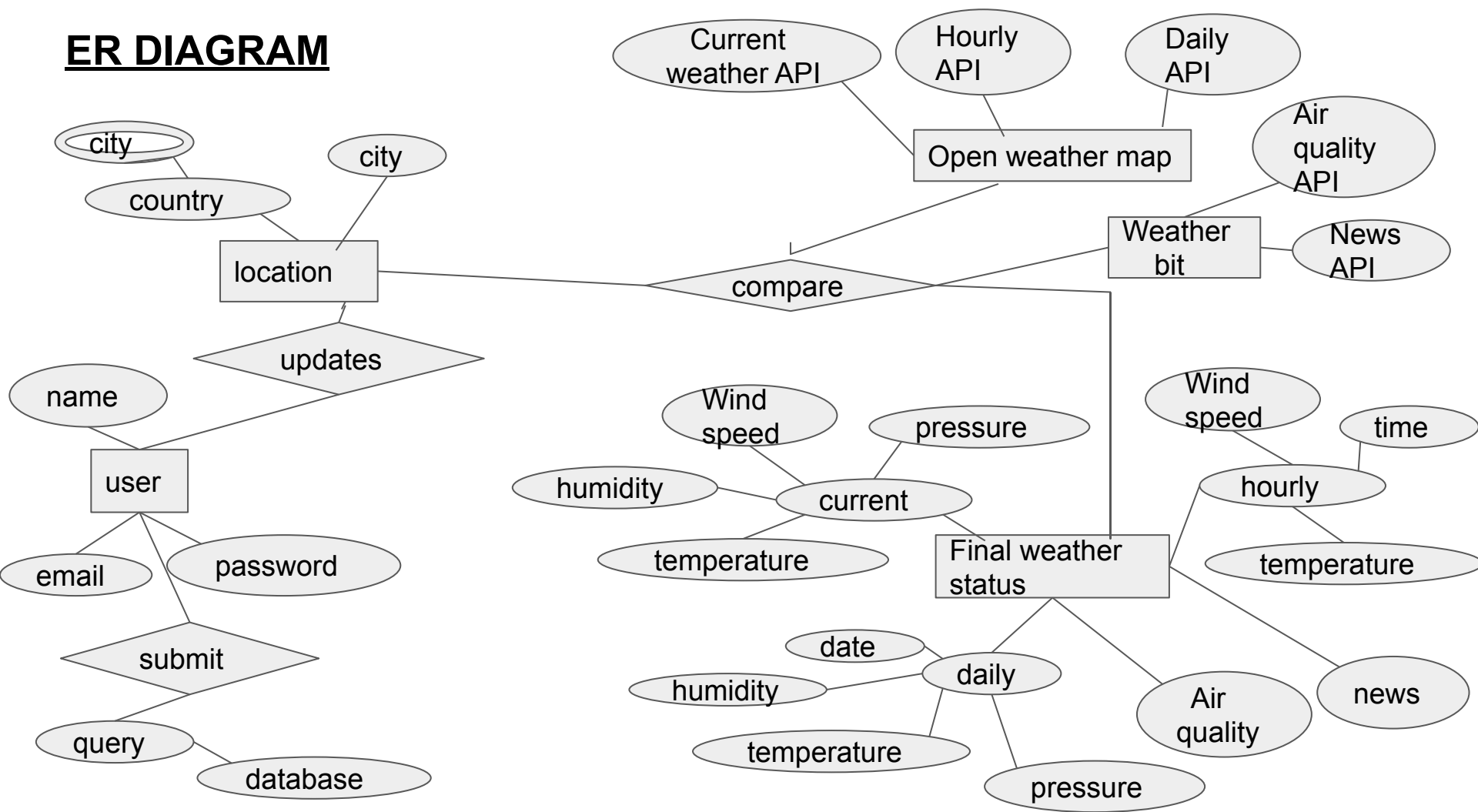
We used Apache Airflow for the purpose of handling our Form submits. It could not be used as frontend tool because it does not support client side scripting.

Therefore we have to integrate it with Flask for user support using Python module AIRFLOW CLIENT.

*Local access link: localhost:8080*

- We have established a Postgres database connection with Airflow to store our API call results in a postgres table of WeatherDB.

**ER DIAGRAM**



# Frontend development

We host our website on a Flask server which is a microframework of Python.

*Access link: localhost:5000*

- Flask is basically a set of tools and libraries that make it easier to build web applications in python.
- It includes support for Jinja templating.
- It provides effective **routing** of web pages through HTML and CSS using `render_template()` and `url_for()`

# Backend Tool: AIRFLOW

Airflow is a open source platform for programmatically authoring, scheduling and monitoring workflows.



## AUTHORING

Workflows in airflow are defined as DAGs which are simple Python scripts.



## SCHEDULING

Users can specify when a workflow should start,end & after what interval It should run again etc.



## MONITORING

Airflow provides an interactive interface to monitor workflows.

# DAGS Creation and Metrics

## ❖ Tree View

A tree representation of DAG that spans across time. If a pipeline is late, one can quickly see where the different steps are and identify blocking ones.

## ❖ Graph View

Visualize your DAGs dependencies and their current status for specific run.

## ❖ Gantt Chart

It lets you analyse task duration and overlaps. One can quickly identify bottlenecks and where the bulk of time is spent for specific DAG runs.

## ❖ Task Duration

The duration of your different tasks over past N runs.

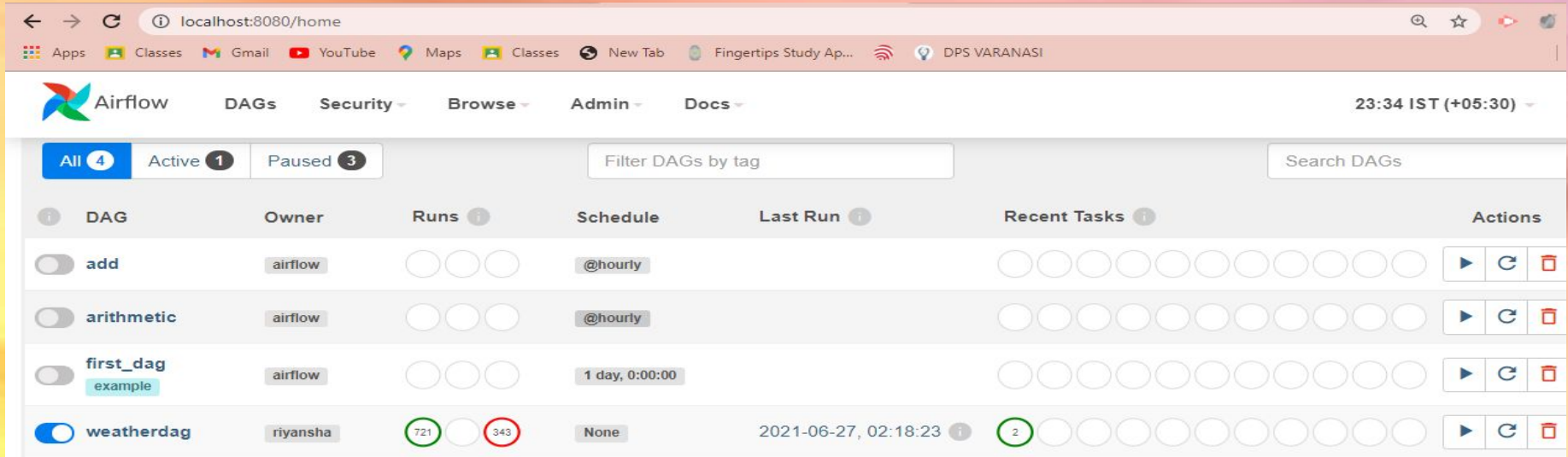
## ❖ Code View

Transparency is everything. It is a quick way to get the code that generates DAG.

# Building the pipeline model

We have created a DAG in Airflow by the name **weatherdag** which comprises of two tasks or basically two works

- **get\_weather**
- **transform\_load**



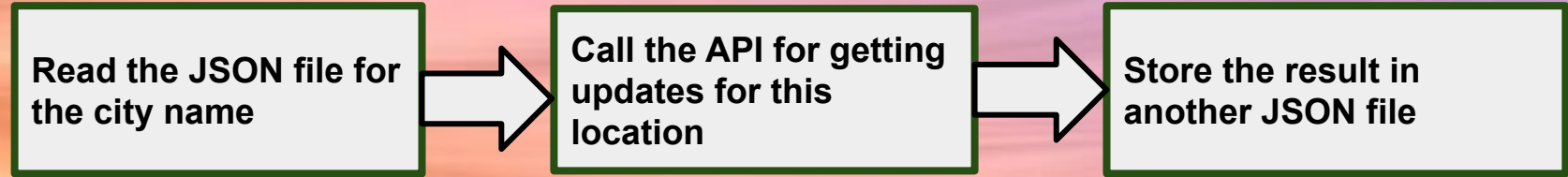
The screenshot displays the Apache Airflow web interface at localhost:8080/home. The interface shows a list of DAGs (Directed Acyclic Graphs) with columns for DAG name, Owner, Runs, Schedule, Last Run, Recent Tasks, and Actions. The 'weatherdag' DAG is highlighted with a blue toggle switch, indicating it is active. It has 2 runs (green circle) and 343 tasks (red circle). The other DAGs shown are 'add', 'arithmetic', and 'first\_dag example', all with grey toggle switches and 0 runs.

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions
<input type="checkbox"/> add	airflow	0	@hourly		0	<a href="#">▶</a> <a href="#">↺</a> <a href="#">✖</a>
<input type="checkbox"/> arithmetic	airflow	0	@hourly		0	<a href="#">▶</a> <a href="#">↺</a> <a href="#">✖</a>
<input type="checkbox"/> first_dag example	airflow	0	1 day, 0:00:00		0	<a href="#">▶</a> <a href="#">↺</a> <a href="#">✖</a>
<input checked="" type="checkbox"/> weatherdag	riyansha	2 (343)	None	2021-06-27, 02:18:23	2	<a href="#">▶</a> <a href="#">↺</a> <a href="#">✖</a>



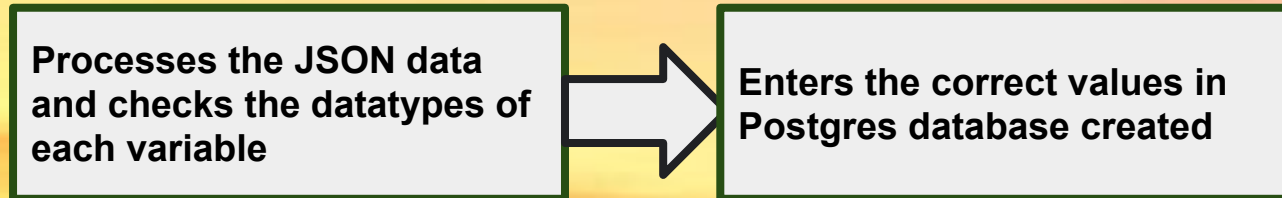
## Get\_weather

This is a simple task executed using **Bash operator** used to get **getweather.py** script run. The script functions in following way-



## Transform\_load

We observe that the JSON data we load from the API is a dictionary with String key-value pairs. **Transform\_load** is a **Python operator** task which-



# Step 1- Passing form input to a JSON file

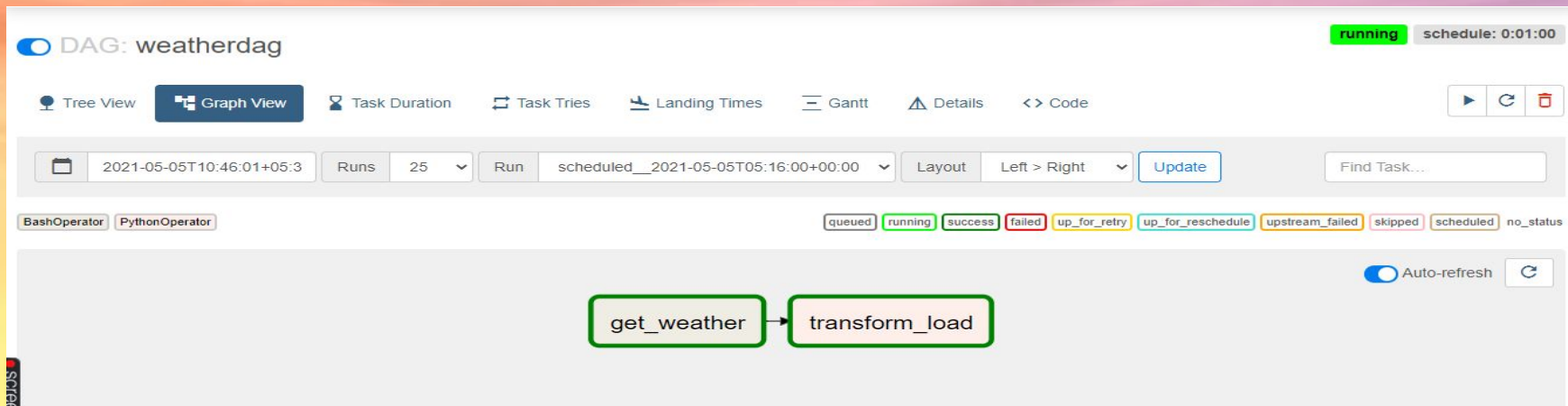
When the user selects location in the form box, it is passed to a python variable via requests library.

It is then loaded into an existing JSON file from there.

```
{
  "city": "Lucknow"
},
{
  "city": "delhi"
},
{
  "city": "Varanasi"
},
{
  "city": "Delhi"
},
{
```

# Step 2-Triggering Airflow DAG externally

Our code is built in such a way that the Airflow **weatherdag** gets triggered as soon as the JSON file is updated.



The two tasks in the DAG **get\_weather** and **transform\_load** execute sequentially one after the other.

## Step 3-Display the result to user from Postgres Database

We present the data to user from Postgres database using **Psycopg2** database adapter which suffice our requirement to work with PostgreSQL database from our main python program.

```
try:
    conn = psycopg2.connect(database="WeatherDB", user="postgres", password="riyansha", host="localhost", port="5432")
    print("connected")
```

We query the database to get recent results(updation takes time).

To fetch the recent result we arrange the data according to date and time first and then select most recent out of it.

## **Future Works**

1. Give weather updates using location pin codes.
2. We can make a well built contact system to give our registered users direct weather updates of their area through email.
3. We can also give historical data conditions.
4. Build a Machine learning model to predict amount of rainfall useful in agricultural sectors.



**THANK YOU**