Big Data and Intelligent Analytics

# Implementing Nowcasting Intelligence System using FastAPI

| Name | NU ID |
| --- | --- |
| Sai Saketh Vootla | 001568984 |
| Shika Shyam | 002194543 |
| Sai Raghavendra Viravalli | 001568587 |

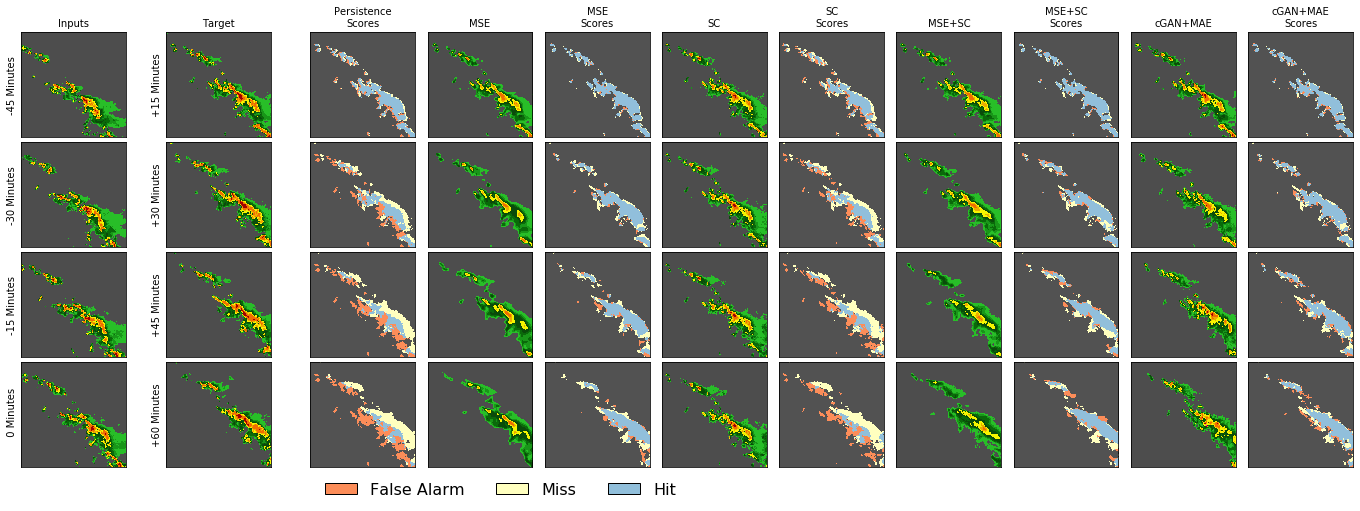
[GITHUB](https://github.com/shikashyam/BigDataSystemsCoursework/tree/main/Assignment3)

# Introduction

This aim of this document is to elaborate in detail about using the Nowcasting model provided by SEVIR and package it within an API using FastAPI. We will be creating the API end points and making it accessible to our end users which is the NOAA. Within the underlying code in the API, we will use the inputs from the User to find the respective event within the SEVIR database, and then use the nowcasting model to take the 12 images as input and return 12 images as output, which will be stored and the path made available to the user.

# 

## Nowcast (Radar Nowcasting)



Nowcasts are high resolution, short-term (e.g. up to 2 hours) weather forecasts of radar echos, precipitation, cloud coverage or other meteorological quantities widely used in public safety, air traffic control, and many other areas that require high fidelity and rapidly updating forecasts. Previous work on deep learning for Nowcasting includes convolutional Long Short Term Memory (ConvLSTM) models, recurrent architectures and fully convolutional networks for precipitation nowcasting.

# 

# Gretel

One of the biggest issues that persist today in the way to create scalable machine learning solutions is the issues of privacy and skewness in Data. Valuable Use Cases in the medical field are limited by the privacy of the data that surrounds healthcare data. Additionally, there are many instances of companies making what they believe are data-driven decisions but making unfair and uninformed decisions because the data is skewed and hence the model is a biased model. It is at a juncture like this that an AI-powered tool like Gretel brings in much-needed expertise in the field of synthetic data generation.

Synthetic data can be thought of as artificial information generated by computer algorithms or simulations that can be used as an alternative to real-world data. While artificial, high-quality synthetic data is capable of capturing the mathematical insights, statistics, and dynamism of real-world data; research shows that it can be [as good or even better than real-world data](https://towardsdatascience.com/reducing-ai-bias-with-synthetic-data-7bddc39f290d) for datasets for analysis and training AI models as it can be engineered to [reduce biases](https://towardsdatascience.com/reducing-ai-bias-with-synthetic-data-7bddc39f290d) and [increase privacy](https://gretel.ai/blog/practical-privacy-with-synthetic-data) vs. real-world datasets.

Gretel makes available three resources for the generation of synthetic data.

* **Gretel Cloud:** The control plane for scheduling work such as creating models and generating, classifying, or transforming data. This includes the Gretel REST API, Console, and CLI tool. The REST API is hosted as a service and is used to manage accounts, projects, and metadata for models.
* **Gretel Configurations:** Declarative objects that are used to create models. Gretel offers several [configuration templates](https://github.com/gretelai/gretel-blueprints/tree/main/config_templates/gretel) to help you get started with popular use cases such as creating synthetic datasets or anonymizing PII. These configurations are sent to the Gretel REST API to create models. These models can then be used to generate, transform, and classify data.
* **Gretel Workers:** Containers that consume Gretel Configurations and handle requests to process records. When a worker consumes a Gretel Configuration, it creates a re-usable model. Additionally, workers can utilize existing models to generate, transform, and classify record

These components work together to enable developers to build robust and flexible privacy engineering workflows.

The notebook can be found here: <https://github.com/shikashyam/BigDataSystemsCoursework/blob/main/Assignment3/notebooks/Using_GRETEL_to_create_synthetic_data.ipynb>

# Designing the API

Logo, company name

Description automatically generated

When designing an [API](http://www.mulesoft.com/platform/api), you don’t just want to build an [API](http://www.mulesoft.com/platform/api). You want to design and build an [API](http://www.mulesoft.com/platform/api) with a long life that your users will love. An [API](http://www.mulesoft.com/platform/api) that is carefully created to be extendable and flexible, and will save your users time, money, and energy in the long run.

**Step - 1: Designing the API interface**

The first step in creating an API is designing the API. You want to know what problems your API needs to solve, then determine what endpoints and data are needed. The decisions you make during the design phase must be documented somewhere. Describing REST APIs is most commonly done with the OpenAPI definition, so designing your API means creating an OpenAPI document for it.

**Step - 2: Mocking the API Server**

A mock API server is an imitation of how your final API server will look. The server itself is real, but temporary, and is meant to respond with fake data. It can be utilized during the design process by providing responses to requests. You can even have dynamic mock data, with the mock server returning data aligned with the expected type of the response field

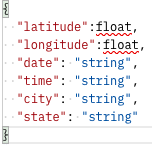
**Step - 3: Building the API**

Now, it’s time to build the real API. This is where many would start, but you’re much more confident about how the API will be used than those approaching code-first. You can use your preferred language to code the API

In our case, we are building an API to use the nowcast model for generating the data of the next hour by passing an input of 13 image files from the SEVIR data. The API is built in such a way that the user is prompted to give an input which will be taken in a JSON format and passed into the API which will trigger the Nowcast model to generate data. There is also a default JSON object stored in the config file which has the null values of the prompted user inputs. The user can change these default values and pass in lat, long OR date, time, city and state.

In the scenario where the user fails to give any inputs, the API will take in the default values of the JSON file, and will give an error stating that there was no information found. When the user fails to give appropriate inputs, the API will prompt about the specific error as well.

The default values in a JSON file looks like this:



# Applications and Assumptions in NOAA DomainLogo Description automatically generated

The National Oceanic and Atmospheric Administration is an American scientific and regulatory agency within the [United States Department of Commerce](https://en.wikipedia.org/wiki/United_States_Department_of_Commerce) that forecasts weather, monitors oceanic and atmospheric conditions, charts the seas, conducts deep sea exploration, and manages fishing and protection of marine mammals and endangered species in the U.S. [exclusive economic zone](https://en.wikipedia.org/wiki/Exclusive_economic_zone).

The NOAA provides a range of services and here we will talk about how our Nowcasting API can be beneficial to NOAA in providing those services effectively:

**National Weather Service:**

Logo

Description automatically generated

The [National Weather Service](https://en.wikipedia.org/wiki/National_Weather_Service) (NWS) is tasked with providing weather, hydrologic and climate forecasts and warnings for the [United States](https://en.wikipedia.org/wiki/United_States), its territories, adjacent waters and ocean areas, for the protection of life and property and the enhancement of the national economy. This is done through a collection of national and regional centers, 13 river forecast centers (RFCs), and more than 120 local weather forecast offices (WFOs). They are charged with issuing weather and river [forecasts](https://en.wikipedia.org/wiki/Weather_forecasting), [advisories, watches, and warnings](https://en.wikipedia.org/wiki/Severe_weather_terminology_(United_States)) on a daily basis. They issue more than 734,000 weather and 850,000 river forecasts, and more than 45,000 [severe weather warnings](https://en.wikipedia.org/wiki/Severe_weather_terminology_(United_States)) annually. NOAA data is also relevant to the issues of [climate change](https://en.wikipedia.org/wiki/Climate_change) and [ozone depletion](https://en.wikipedia.org/wiki/Ozone_depletion).

They currently own the radars and sensors that create and log SEVIR data. The nowcasting model can be useful to them in order to make accurate weather predictions in order for broadcasting them to the entire continental US.

**National Ocean Service**

The [National Ocean Service](https://en.wikipedia.org/wiki/National_Ocean_Service) (NOS) focuses on ensuring that ocean and coastal areas are safe, healthy, and productive. NOS scientists, natural resource managers, and specialists serve America by ensuring safe and efficient marine transportation, promoting innovative solutions to protect coastal communities, and conserving marine and coastal places.

Our API can be used to provide effective weather forecasts for coastal areas and oceans and accordingly ensure that NOS’s requirements of safe marine transportation is met. It can also aid in recharting marine routes based on storm warnings.

#### **Office of Oceanic and Atmospheric Research**

NOAA's research, conducted through the Office of Oceanic and Atmospheric Research (OAR), is the driving force behind NOAA environmental products and services that protect life and property and promote economic growth. Research, conducted in OAR laboratories and by extramural programs, focuses on enhancing our understanding of environmental phenomena such as tornadoes, hurricanes, climate variability, solar flares, changes in the ozone etc. NOAA research also develops innovative technologies and observing systems.

Our model with some additional capabilities can help NOAA convert this research information to understand changing climate patterns in coastal areas and in oceans.

#### **NOAA ships and aircraft Office of Marine and Aviation Operations**

[The Office of Marine and Aviation Operations](https://en.wikipedia.org/wiki/NOAA_ships_and_aircraft) is responsible for the fleet of NOAA ships, aircraft, and diving operations. It has the largest research fleet of the Federal government.

Our API can facilitate safe movement of these fleet with accurate nowcasting of the charted paths for these ships.

**ASSUMPTIONS about NOAA users:**

1. We assume the NOAA users have access to information required as an input to the API – i.e. Either Latitude and Longitude or Date, Time, City and State
2. Since the SEVIR Catalog consists of historical data and this API is a POC, we assume that NOAA is looking for nowcasting of historical data
3. Since the SEVIR Catalog is limited and is not a continually updated list of events, we assume that the nowcasting information NOAA user is looking for exists in the CATALOG.
4. Since SEVIR is ultimately a dataset that consists of only environmental events like storms and random events, we assume that NOAA is looking for nowcasting based on storm or random event data not normal weather data.
5. Since our user is NOAA, and they have access to satellites and radar information, we assume that they have access to latitude and longitude for specific locations and want to see targeted nowcasting for just that location.
6. Since the CATALOG is a limited dataset, we have only one event per lat,long, because SEVIR events are rare. Because of this we do not need any additional datetime based information should the user choose to give latitude and longitude as input.

# Underlying Python Logic

With the help of the Boto3 command, we can extract the desired files from the SEVIR s3 Bucket

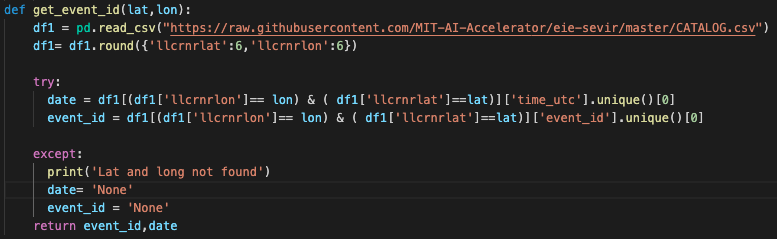
Boto3 is the name of the Python SDK for AWS. It allows you to directly create, update, and delete AWS resources from your Python scripts.

Based on the input given by the user, we will either search the catalog by latitude and longitude or date, time, city and state like shown below:

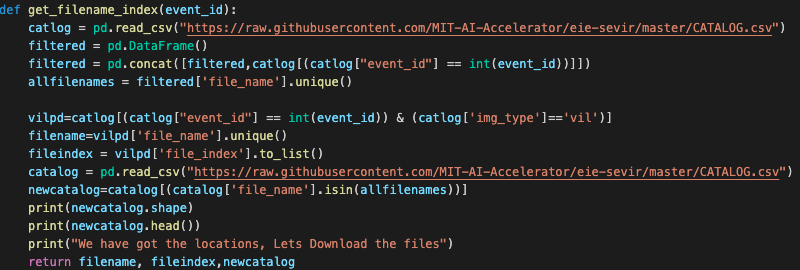
Text

Description automatically generated

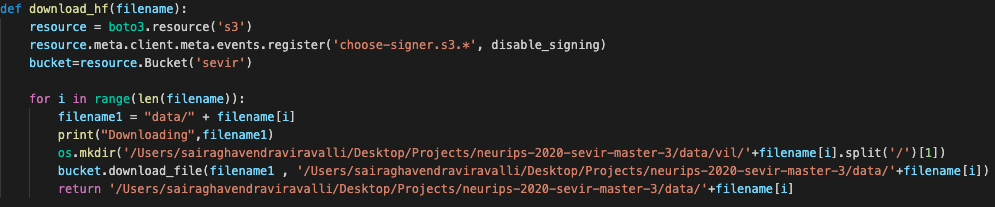
Below is the implicit function used to search catalog based on latitude and longitude:



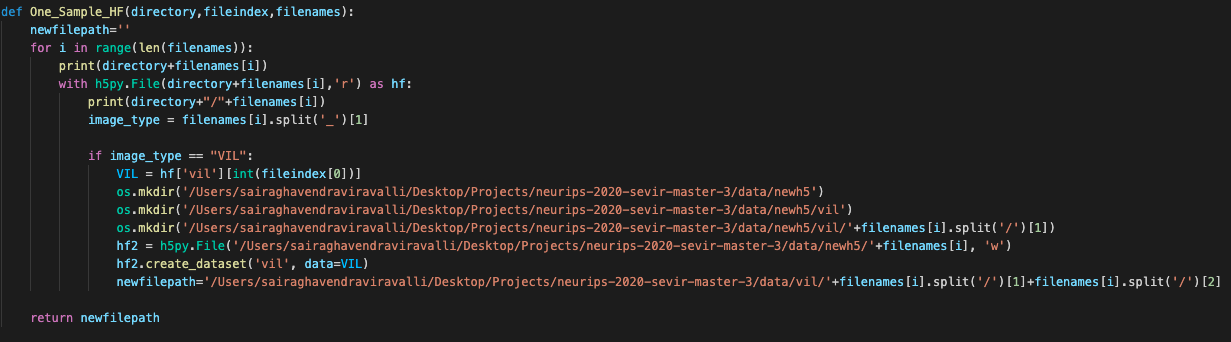
Now that we have the event\_id, we need to pass into another function that can extract the filename and file index of the desired event\_id to locate the h5 files inside the s3 bucket using the boto3 command



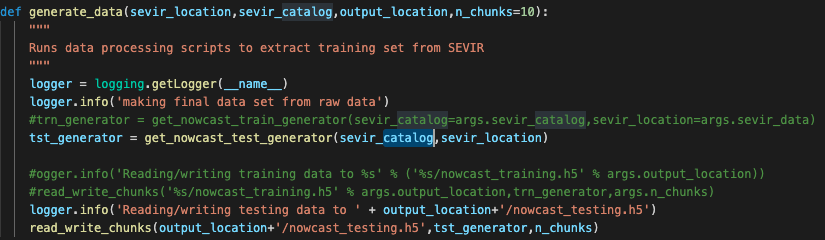
Given the filename and location, we have now downloaded the h5 files from the SEVIR bucket of the respectively given event\_id



Once we have the downloaded files ready we are creating individual h5 files with respect to the image type which we gathered using the above function



As everything is ready now with the files and the filtered catalog.csv file, it now time to hit the nowcast model to generate the data



Once the nowcast model has generated the data we can visualize it and save the plot into an image

After this, we move it to the bucket and share the bucket URL location with the user

# Postman Testing



Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

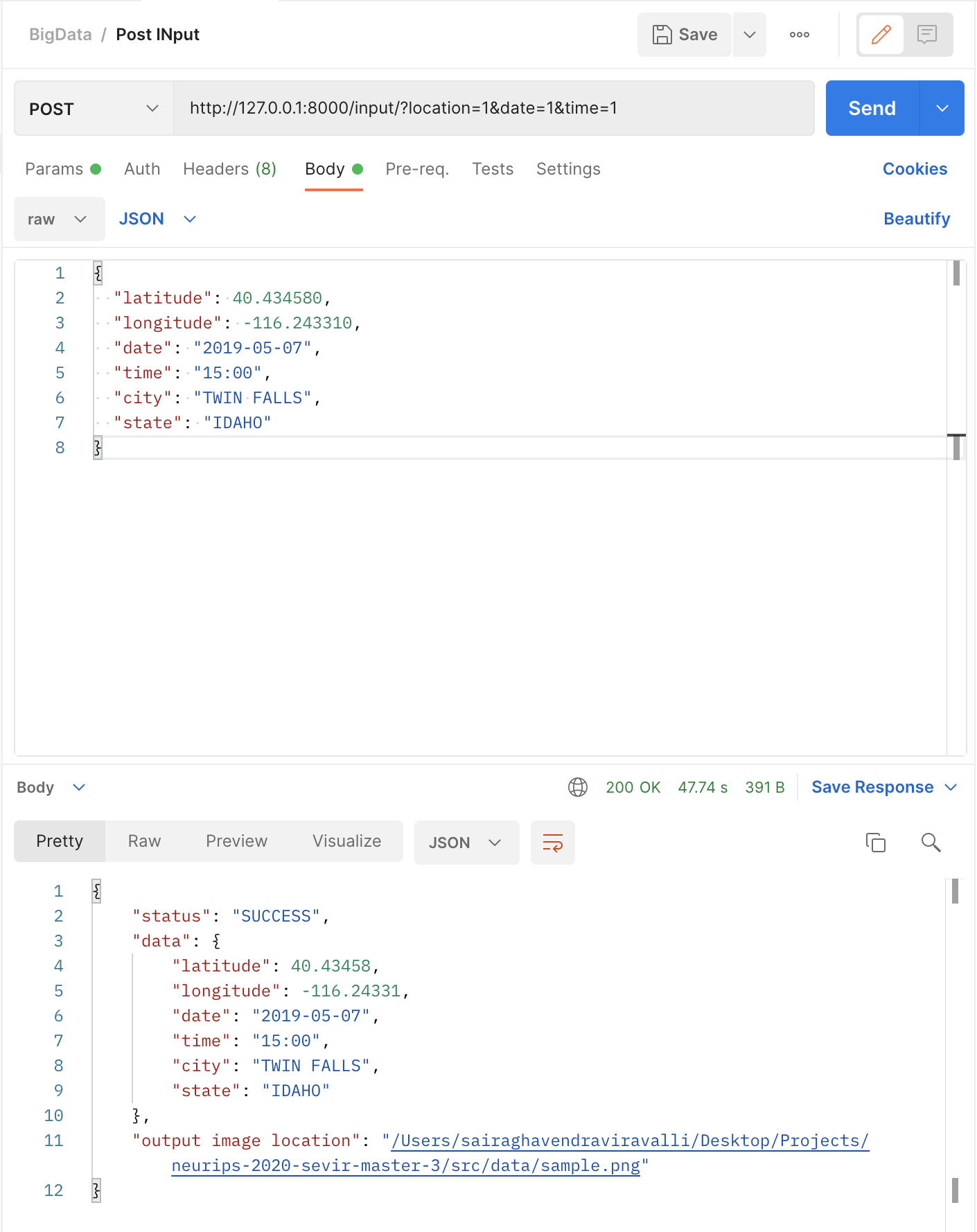
Postman is a great tool when trying to dissect RESTful APIs made by others or test ones you have made yourself. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality.

Postman can run PUT, PATCH, DELETE, and various other request methods as well, and also has utilities to help with developing APIs.

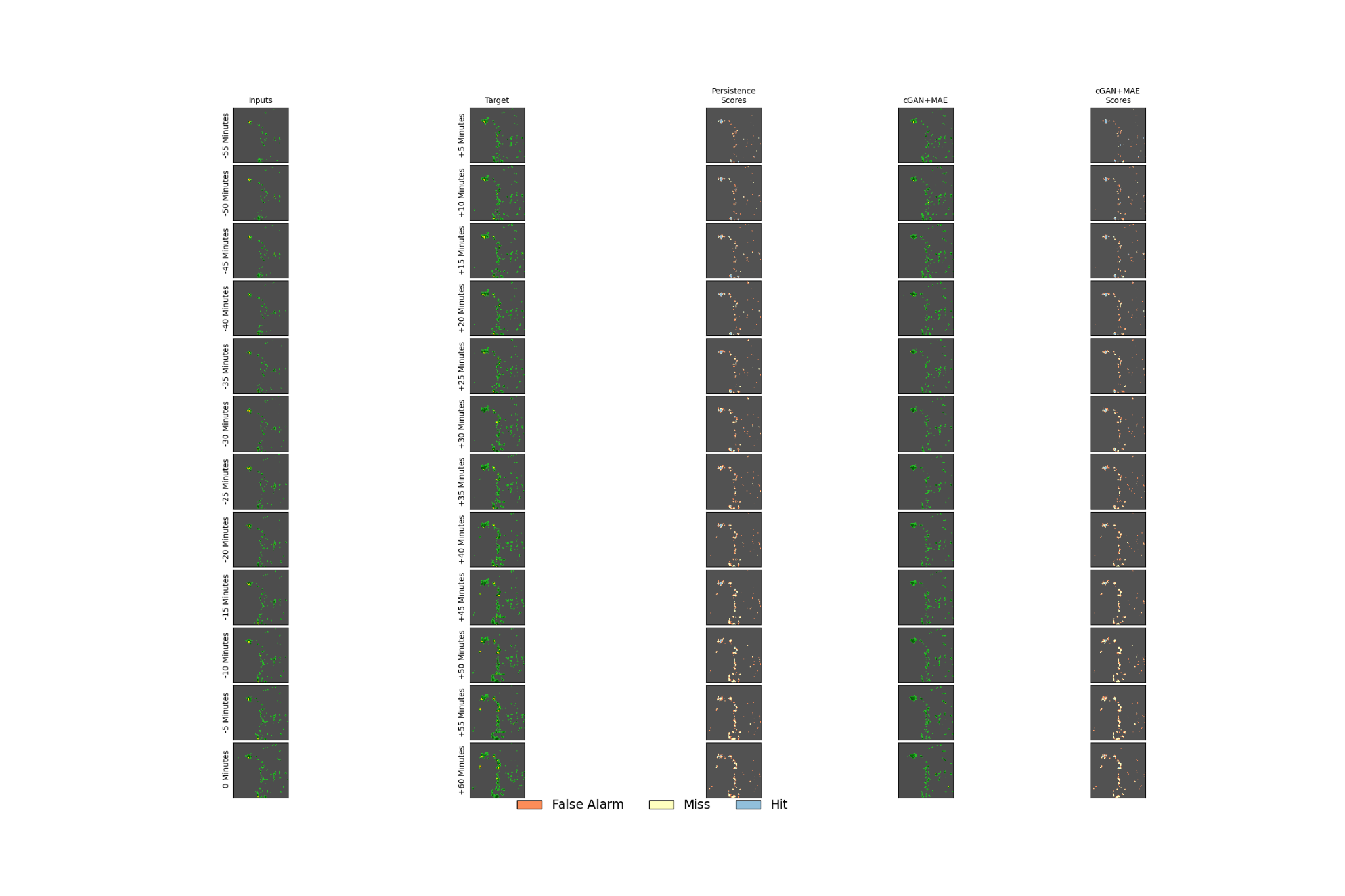
Below is an example of an input given through Postman and the returned value:

# Test Cases

## Test Case 1 - Passed

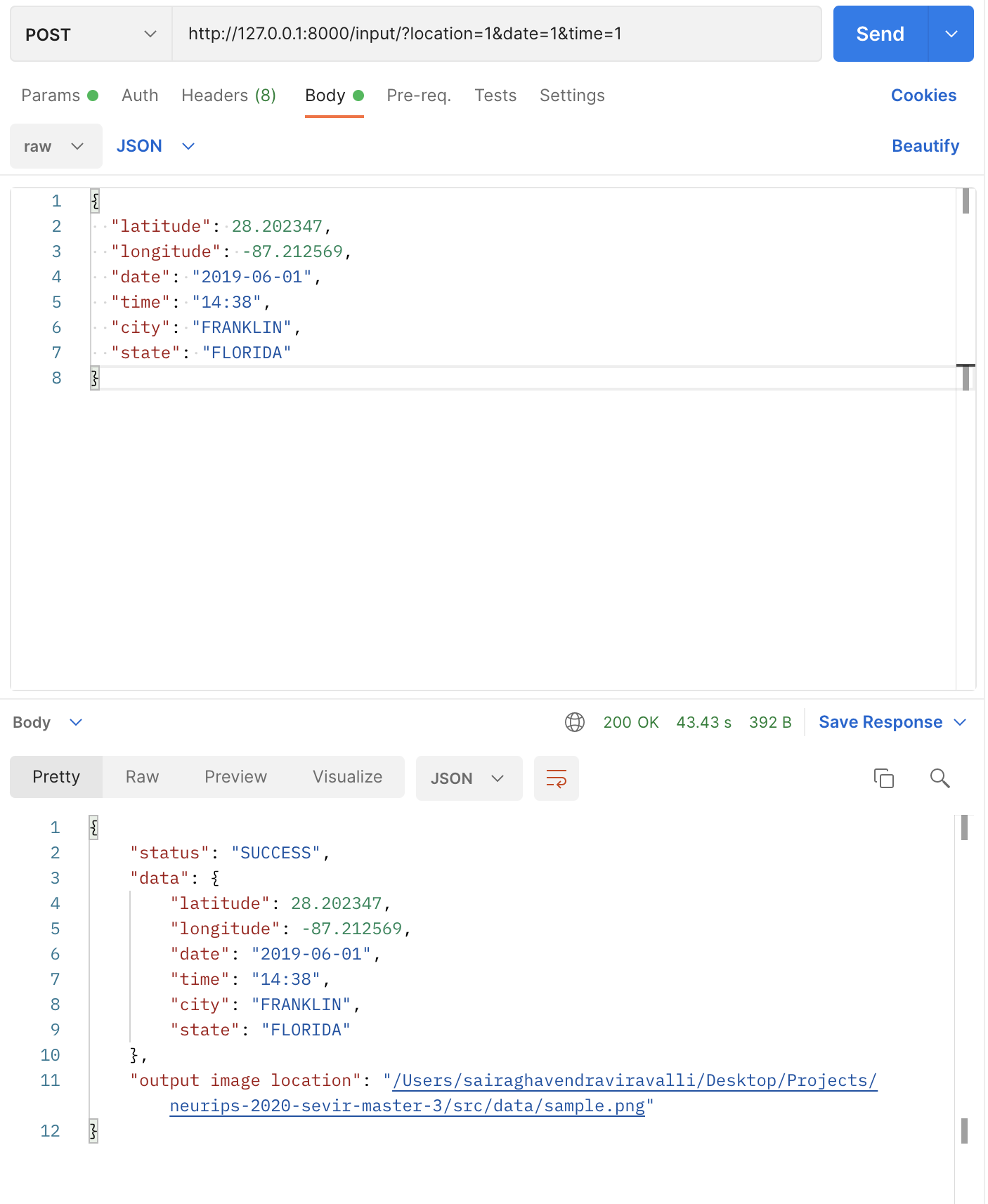


Sample Image:-

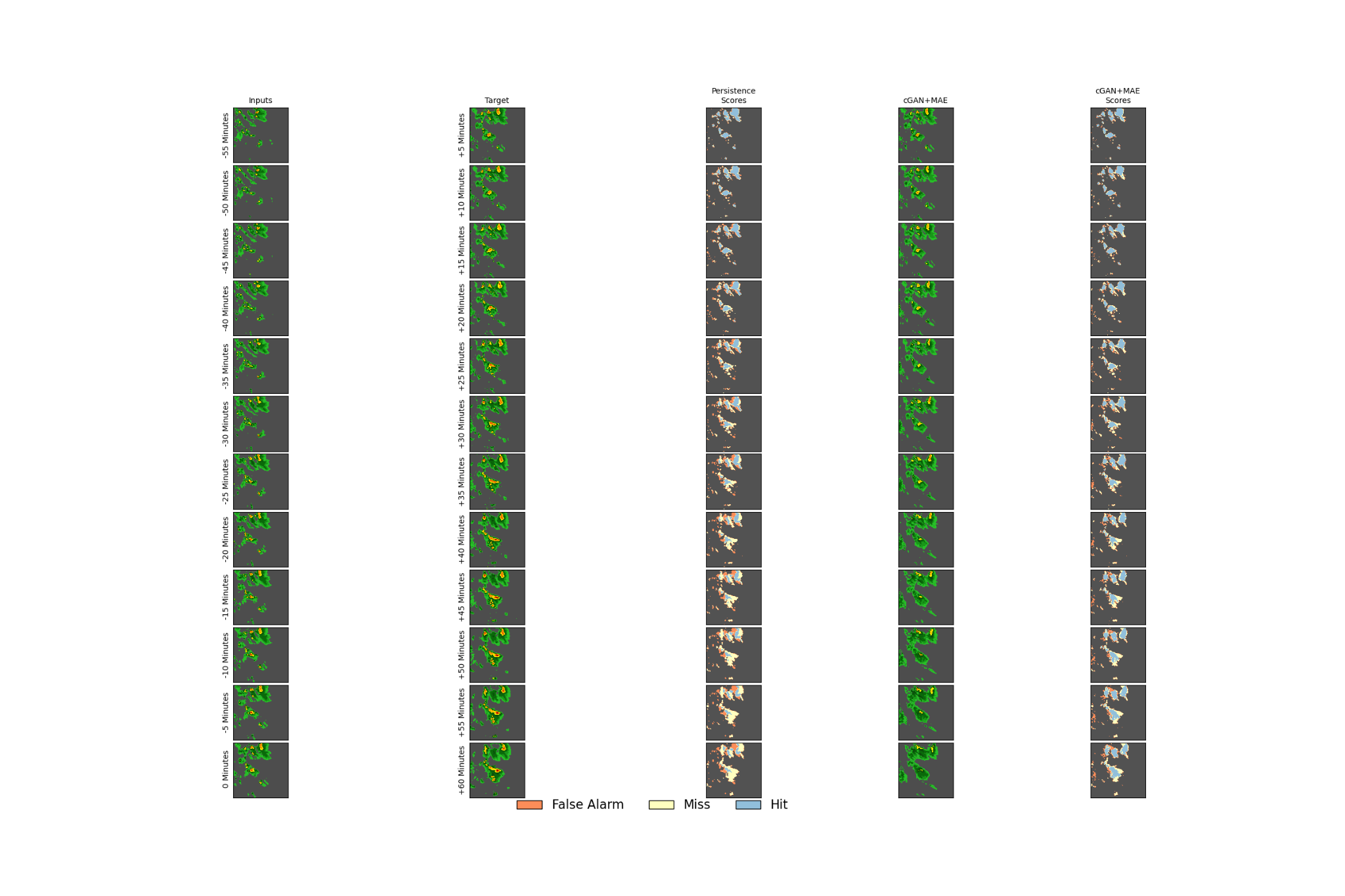


# 

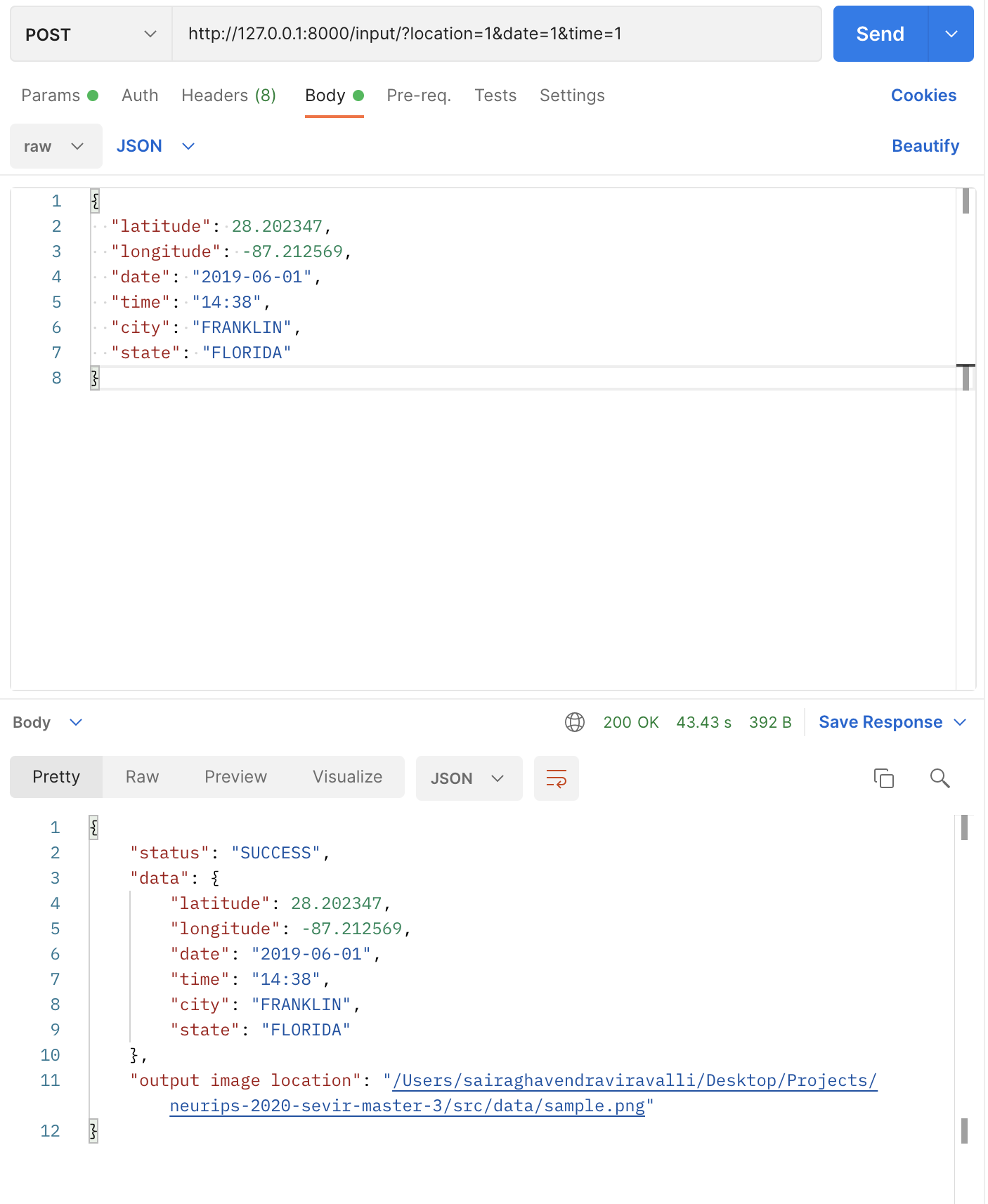
## Test Case 2 - Passed



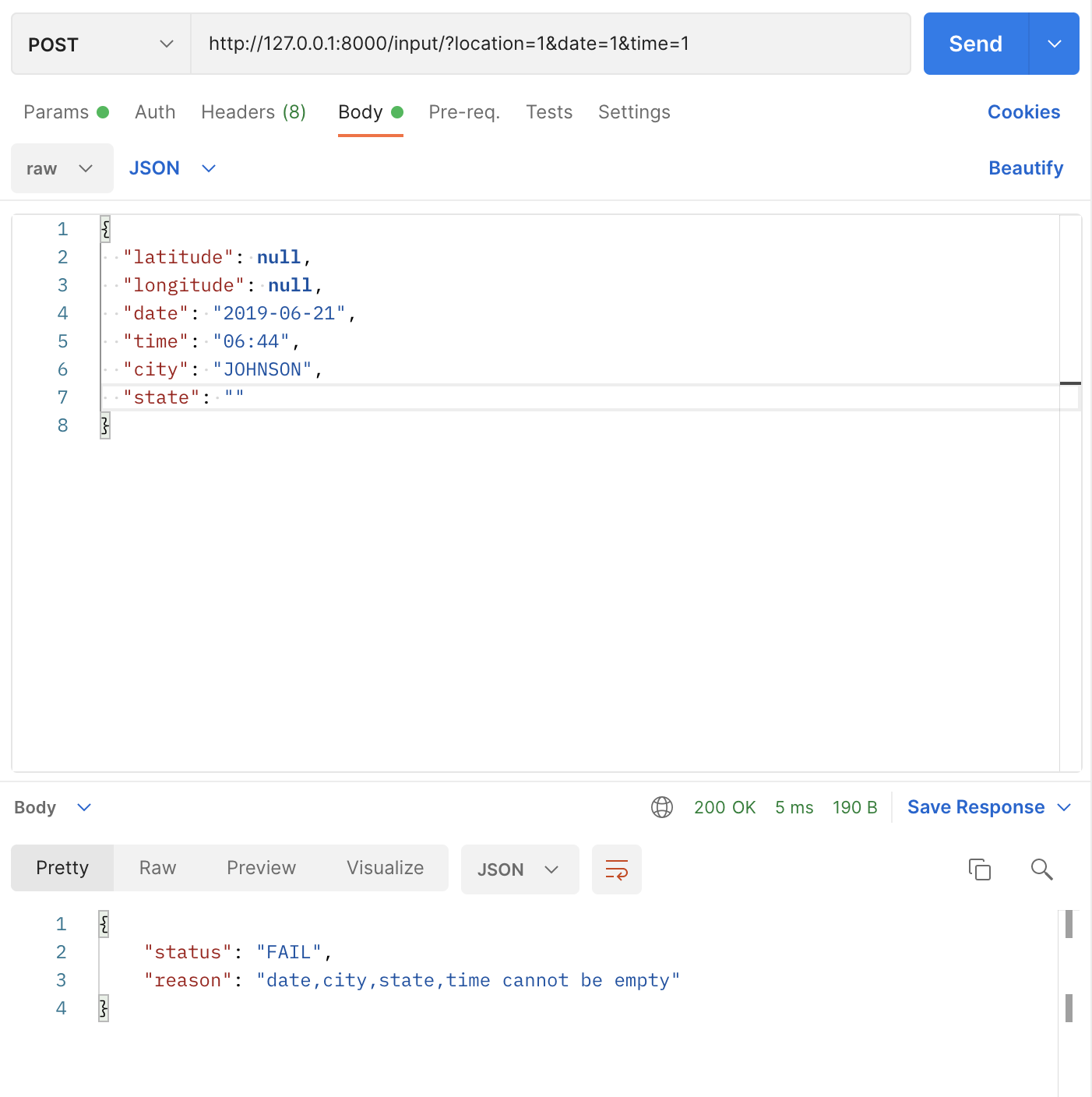
## Output Image



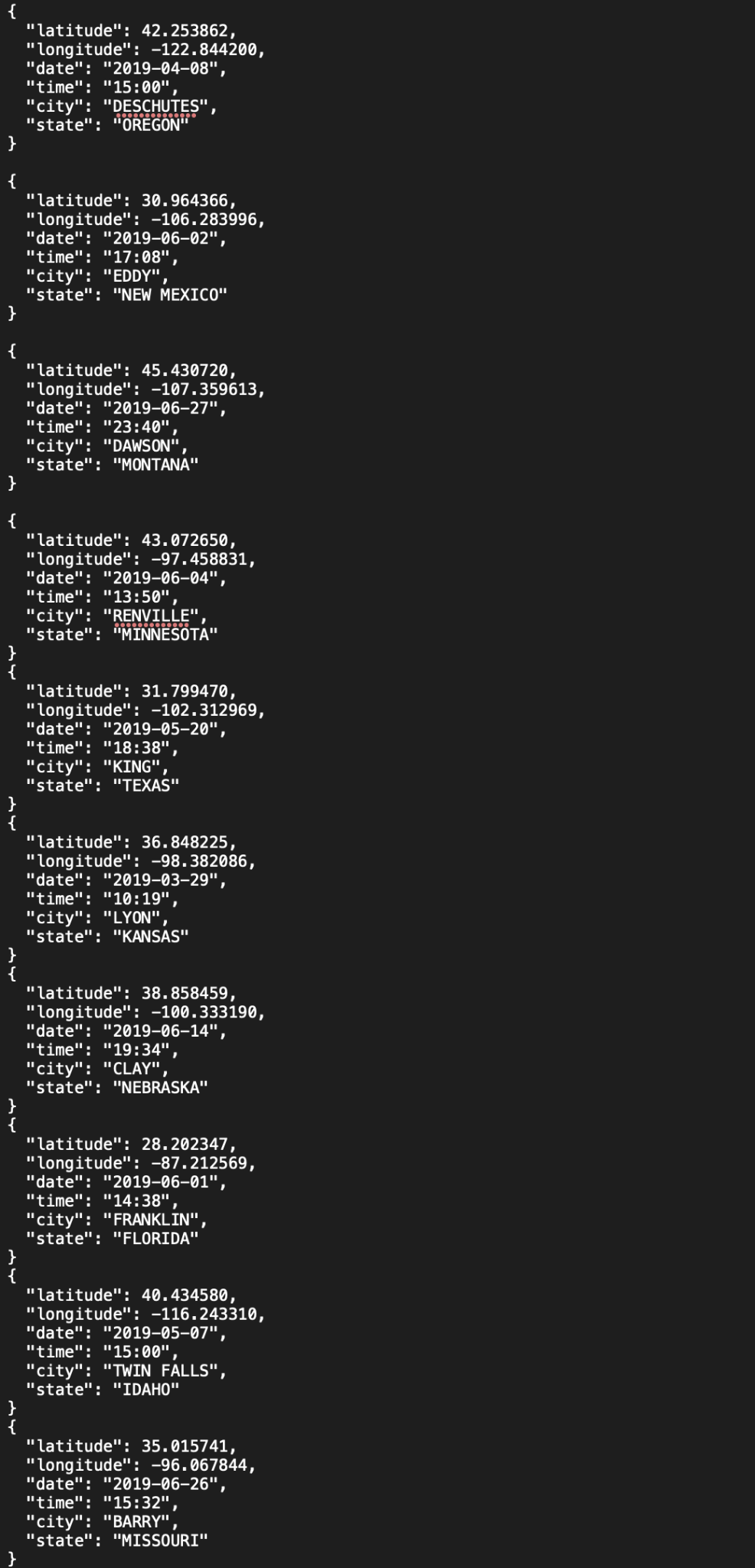
## Test Case 3- Failed



## Test Case 4 -Failed



## Test Cases.py used for testing



# Jupyter Notebook demonstrating the API



We have also written a Jupyter notebook, which illustrates the usage of the API we just designed, and uses the structure.json default json input file and change the values in the json to give the desired inputs to the API. The notebook shows the results received for two sample API requests, one is a SUCCESS scenario, the other a fail.

The notebook can be found at :

<https://github.com/shikashyam/BigDataSystemsCoursework/blob/main/Assignment3/notebooks/Illustrating_use_of_API_for_Nowcasting.ipynb>

# References

* <https://github.com/MIT-AI-Accelerator/eie-sevir>
* <https://github.com/MIT-AI-Accelerator/neurips-2020-sevir>
* <https://github.com/MIT-AI-Accelerator/sevir_challenges>
* <https://sevir.mit.edu>
* <https://www.noaa.gov/>
* <https://en.wikipedia.org/wiki/National_Oceanic_and_Atmospheric_Administration>
* <https://fastapi.tiangolo.com/>
* <https://github.com/tiangolo/fastapi>
* <https://realpython.com/fastapi-python-web-apis/>
* <https://www.toptal.com/python/build-high-performing-apps-with-the-python-fastapi-framework>