W.here

Weather here

W.here

INDEX

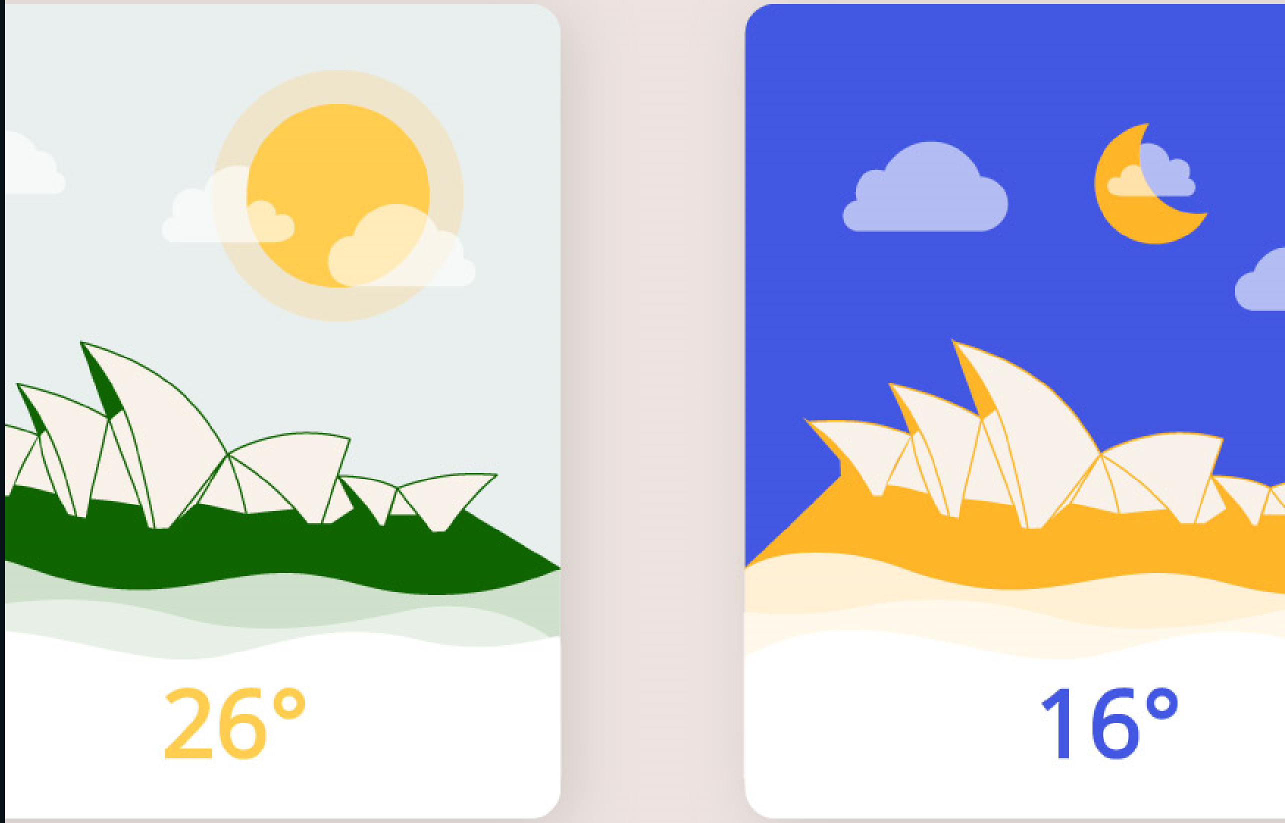
Weather Here

W.here

ABOUT US

SOFTWARE SOLUTIONS COMPANY

- We provide high-end solutions to our renowned clients worldwide.
- Customize the solutions such that it is useful to the end user and innovative enough as well
- Connect and expand with clients across multiple domains.



Weather Here

W.here

OUR TEAM

Bhavi Sureshkumar Dudhat

Web Developer & Business Analyst

Bachelor in Computer Engineering

Neev

Project Manager

Priya Chuahan

Data Analyst

PROJECT SUMMARY

Real –Time Weather Application

The web application - W.here tracks user's public IP address to provide the real-time weather dashboard.

The web application first extracts the user's public IP address, then get the geolocation from IP address, as per the longitude and latitude details of geolocation the Weather API sends local weather details as a response.

This data is then sent to MongoDB(Cloud storage) and also displayed on the Weather dashboard webpage.

Also, configured stand-alone APIs that serves data from the live cloud database to Get all items, Get a range of items and Get item by ID.

W.here

PROJECT SCOPE

1. Extract Geolocation based On Public IP Tracking

Passing the public IP address extracted from request headers to the IP function of geocoder python library gives the local longitude and latitude information

2. Get the local weather data from API

Passing the geolocation parameters from step 1 to the fcc weather API, gives all weather related information as a response in a JSON format.

3. Stored the data in MongoDB

The API data is stored in MongoDB cloud.

4. Develop the Flask application and visualizations

Using python loading the data from API to the flask app and also used the data to design a visualization.

5. Deploy the Web App

Deployed the responsive web-app on Heroku so that anyone can access it using any device.

6. Live API

Created an API using MongoDB Realm to access data from cloud.

TOOLS & TECHNOLOGIES

HTML

Markup language which is used to display the front-end alongwith CSS styling.

JavaScript

Javascript is a lightweight programming language with first-class functions used to display data using google chart

GitHub

Stored code on Github for hosting on Heroku

Flask

Flask is a python framework used in core web development of the project

MongoDB & MongoRealm

Cloud storage and Live API building

Heroku

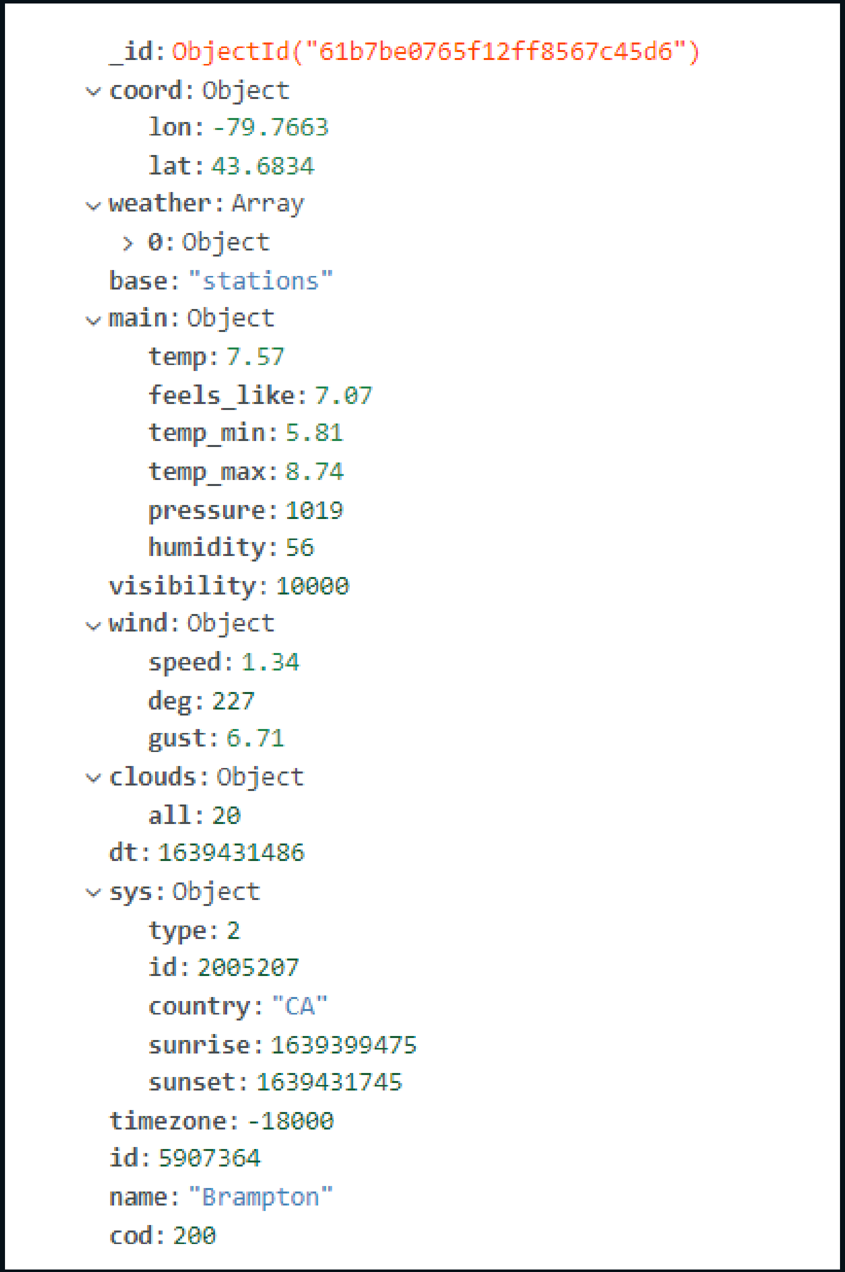
Cloud platform for web deployment

W.here

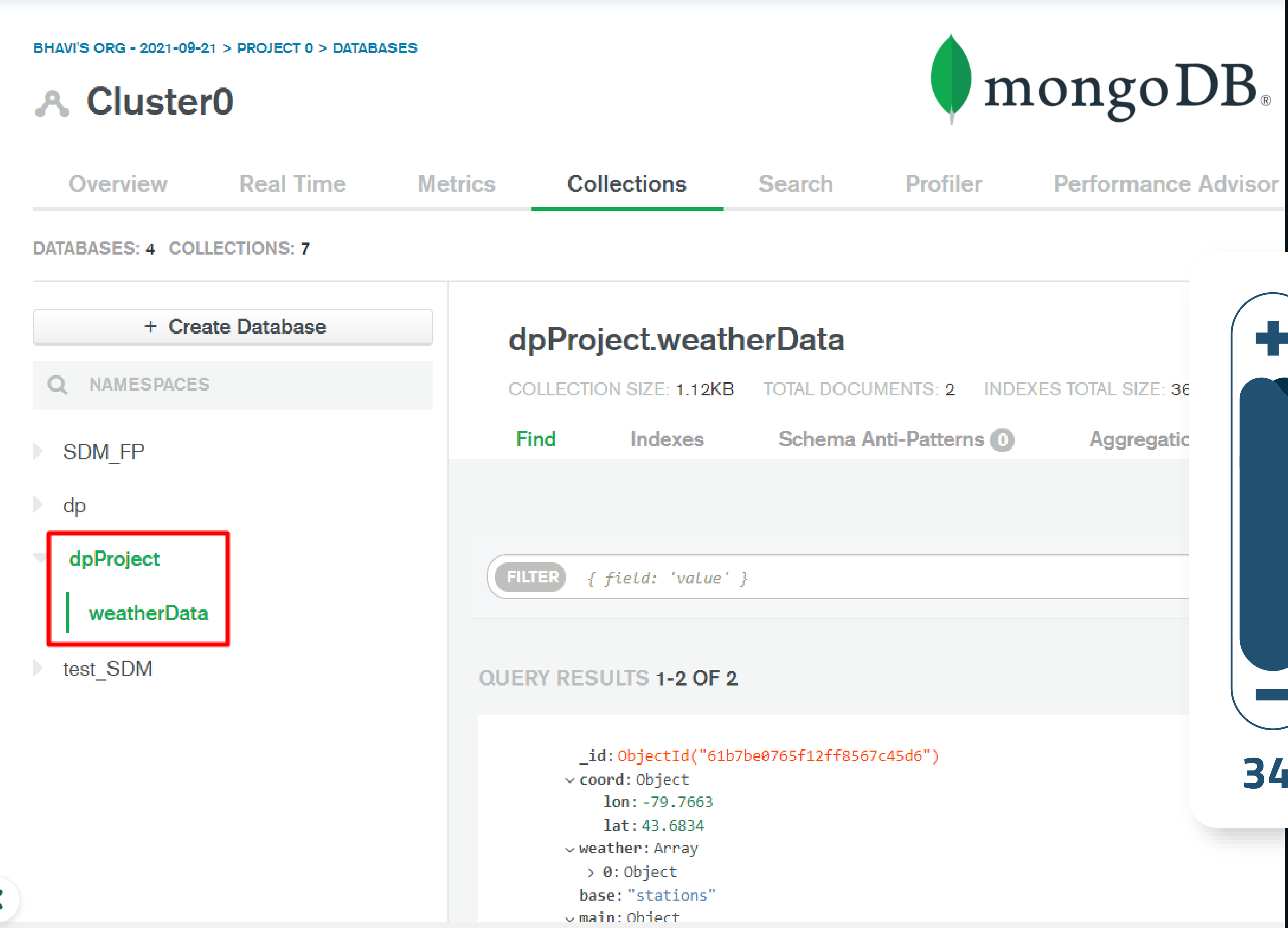
DATA MODEL

Weather Data

The API response sent to MongoDB storage is as shown in the image, it contains geolocation and weather related Data.



```
  _id: ObjectId("61b7be0765f12ff8567c45d6")
⌄ coord: Object
    lon: -79.7663
    lat: 43.6834
⌄ weather: Array
  › 0: Object
  base: "stations"
⌄ main: Object
    temp: 7.57
    feels_like: 7.07
    temp_min: 5.81
    temp_max: 8.74
    pressure: 1019
    humidity: 56
  visibility: 10000
⌄ wind: Object
    speed: 1.34
    deg: 227
    gust: 6.71
⌄ clouds: Object
    all: 20
  dt: 1639431486
⌄ sys: Object
    type: 2
    id: 2005207
    country: "CA"
    sunrise: 1639399475
    sunset: 1639431745
  timezone: -18000
  id: 5907364
  name: "Brampton"
  cod: 200
```


BHAVI'S ORG - 2021-09-21 > PROJECT 0 > DATABASES

Cluster0

mongoDB®

Overview | Real Time | Metrics | Collections | Search | Profiler | Performance Advisor

DATABASES: 4 COLLECTIONS: 7

+ Create Database

NAMESPACES

- SDM_FP
- dp
- dpProject
  - weatherData
- test_SDM

dpProject.weatherData

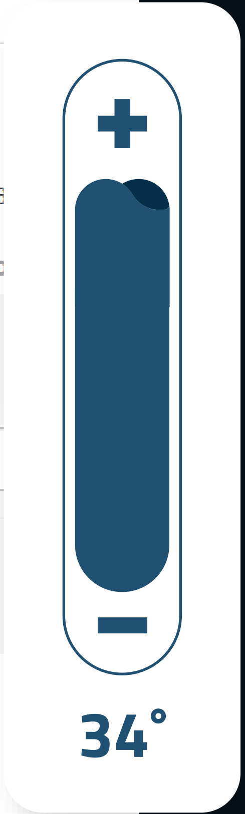
COLLECTION SIZE: 1.12KB TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36

Find | Indexes | Schema Anti-Patterns 0 | Aggregatio

FILTER { field: 'value' }

QUERY RESULTS 1-2 OF 2

```
  _id: ObjectId("61b7be0765f12ff8567c45d6")
v coord: Object
    lon: -79.7663
    lat: 43.6834
v weather: Array
   > 0: Object
  base: "stations"
v main: Object
```



DATA STORAGE

MongoDB Cloud

Stored the data in weatherData

collection in the dpProject database

On MongoDB Atlas cloud.

W.here

APP CODE

Python Web Application

Flask backend code for accessing geolocation

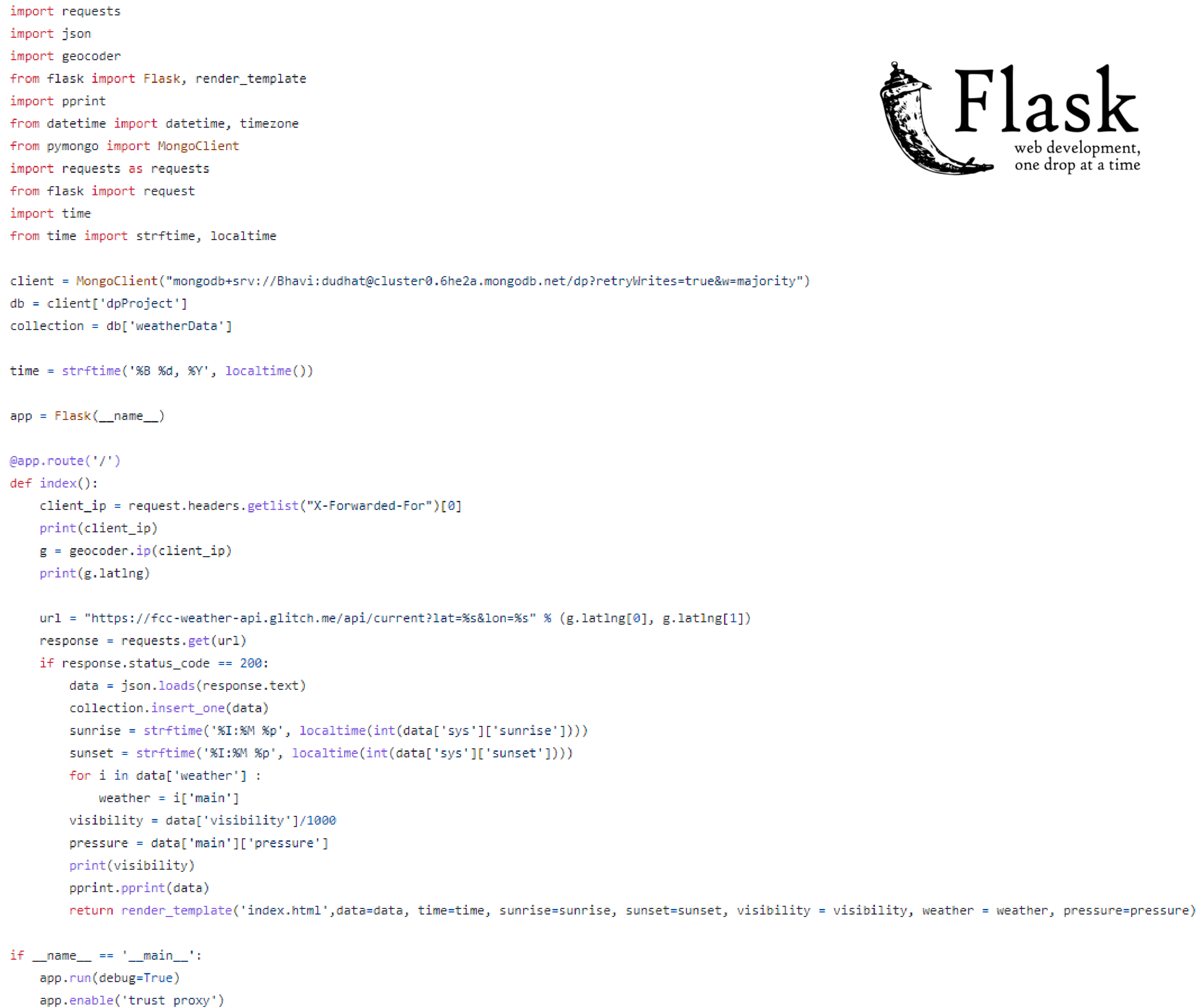
Data from request headers, passing to API and

Gathering respective Weather data as a

Response, displaying it on webpage and

sending to the mongoDB cloud.

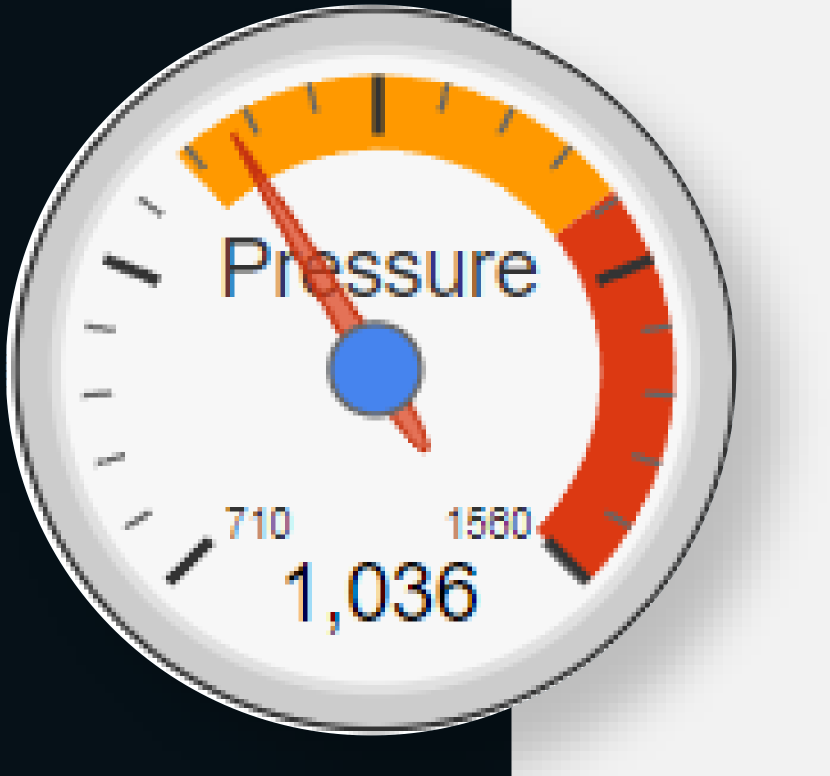
Weather Here



```python
import requests
import json
import geocoder
from flask import Flask, render_template
import pprint
from datetime import datetime, timezone
from pymongo import MongoClient
import requests as requests
from flask import request
import time
from time import strftime, localtime

client = MongoClient("mongodb+srv://Bhavi:dudhat@cluster0.6he2a.mongodb.net/dp?retryWrites=true&w=majority")
db = client['dpProject']
collection = db['weatherData']

time = strftime('%B %d, %Y', localtime())

app = Flask(__name__)

@app.route('/')
def index():
    client_ip = request.headers.getlist("X-Forwarded-For")[0]
    print(client_ip)
    g = geocoder.ip(client_ip)
    print(g.latlng)

    url = "https://fcc-weather-api.glitch.me/api/current?lat=%s&lon=%s" % (g.latlng[0], g.latlng[1])
    response = requests.get(url)
    if response.status_code == 200:
        data = json.loads(response.text)
        collection.insert_one(data)
        sunrise = strftime('%I:%M %p', localtime(int(data['sys']['sunrise'])))
        sunset = strftime('%I:%M %p', localtime(int(data['sys']['sunset'])))
        for i in data['weather'] :
            weather = i['main']
        visibility = data['visibility']/1000
        pressure = data['main']['pressure']
        print(visibility)
        pprint.pprint(data)
        return render_template('index.html',data=data, time=time, sunrise=sunrise, sunset=sunset, visibility = visibility, weather = weather, pressure=pressure)

if __name__ == '__main__':
    app.run(debug=True)
    app.enable('trust proxy')
```

Flask
web development,
one drop at a time

W.here



VISUALIZATION

Google Charts

Using the live Atmospheric pressure data received a s a Response from the API in googlecharts.js Gauge chart to visualize it as a **Barometer**

W.here

WEBSITE FRONTEND DESIGN

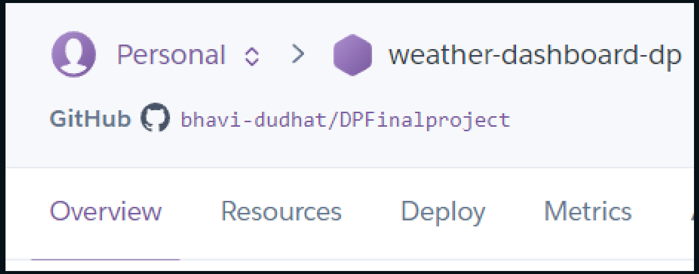
- Used different cards for various parameters
- Responsive layout using Bootstrap and CSS
- Rich UI design to fulfill aesthetic demand

W.here

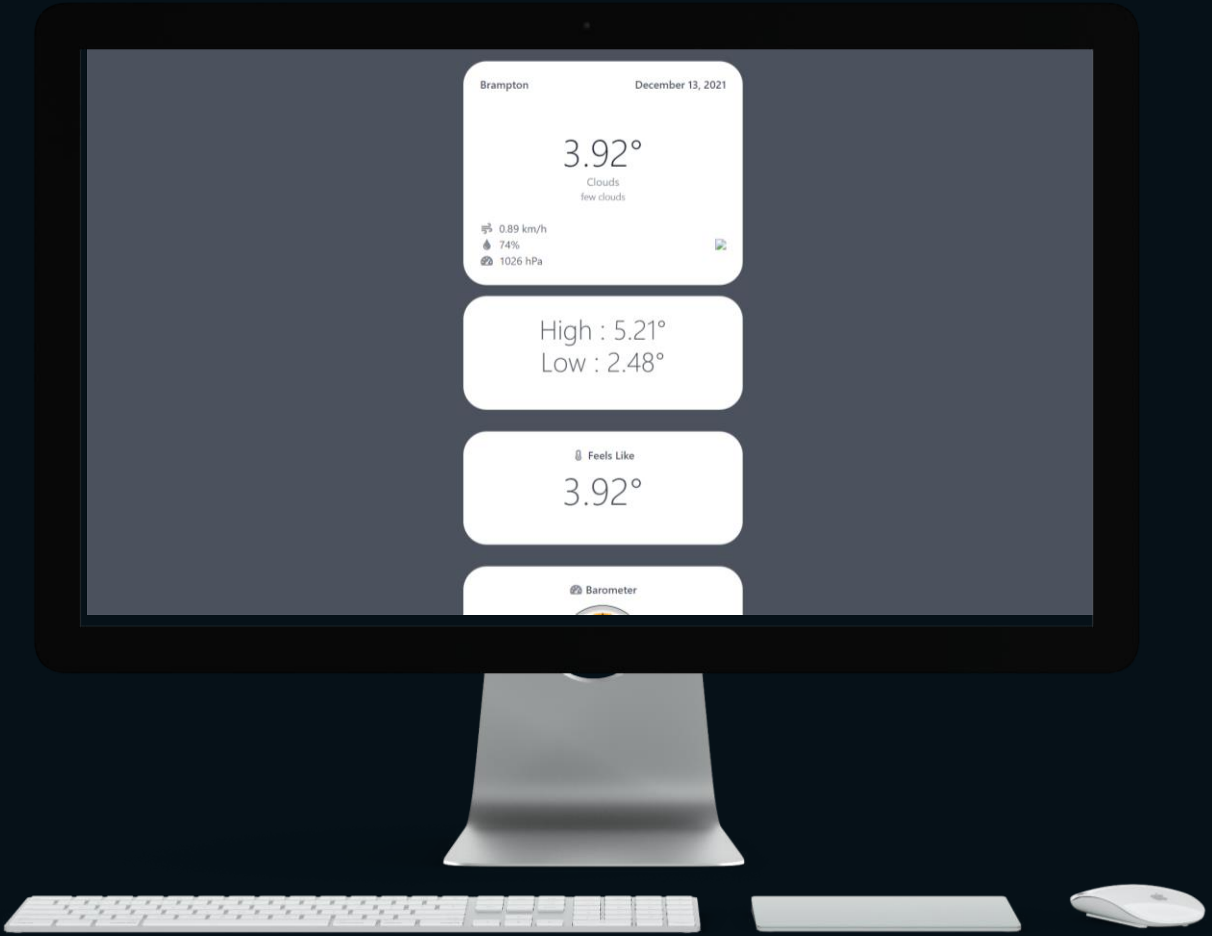
WEBSITE DEPLOYMENT

W.here

https://weather-dashboard-dp.herokuapp.com/


Personal > weather-dashboard-dp

GitHub bhavi-dudhat/DPFinalproject


Brampton December 13, 2021

3.92°

Clouds

few clouds

0.89 km/h

74%

1026 hPa

High : 5.21°

Low : 2.48°

Feels Like

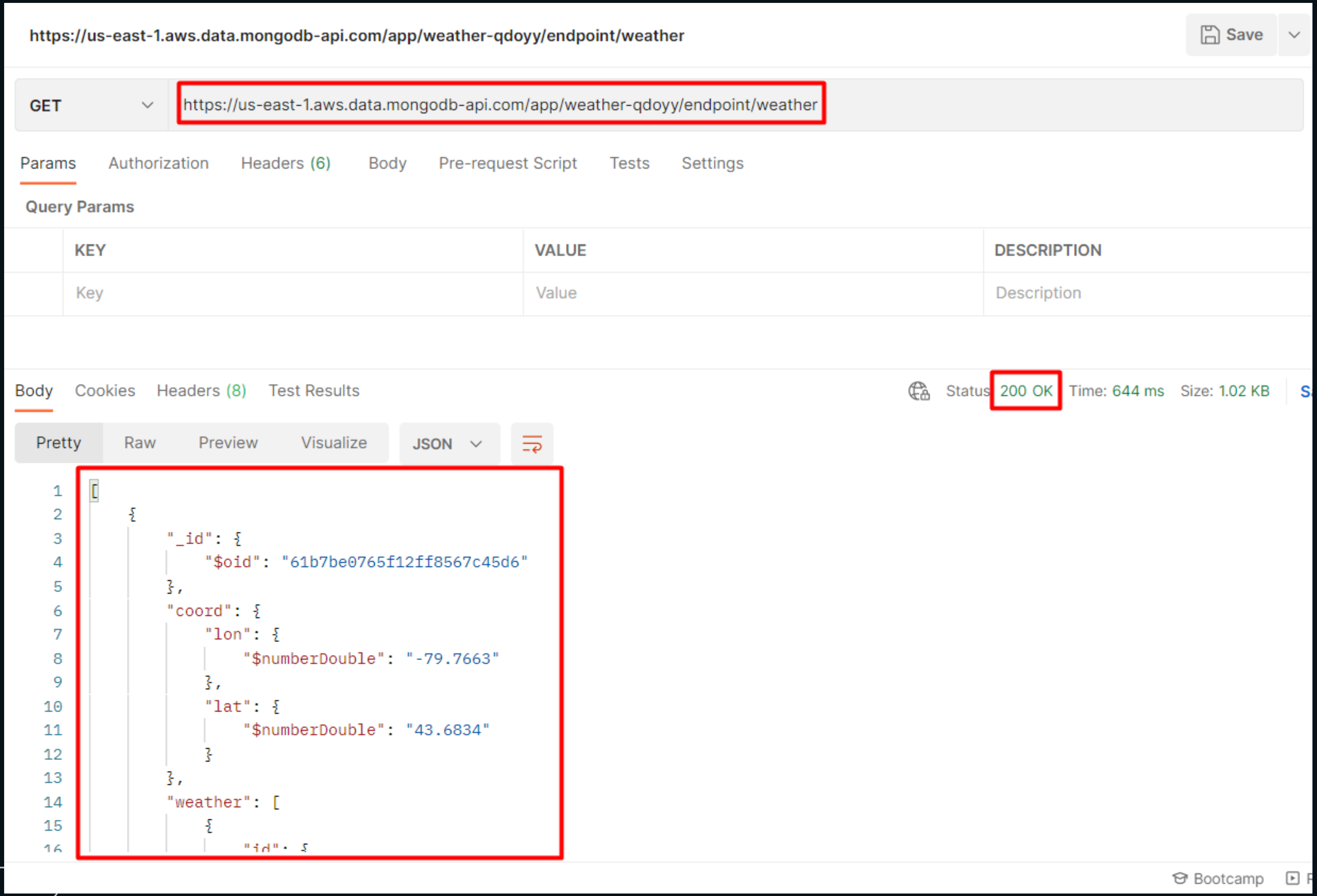
3.92°

Barometer

W.here

LIVE API Get all items

https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weather


https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weather

GET https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weather

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

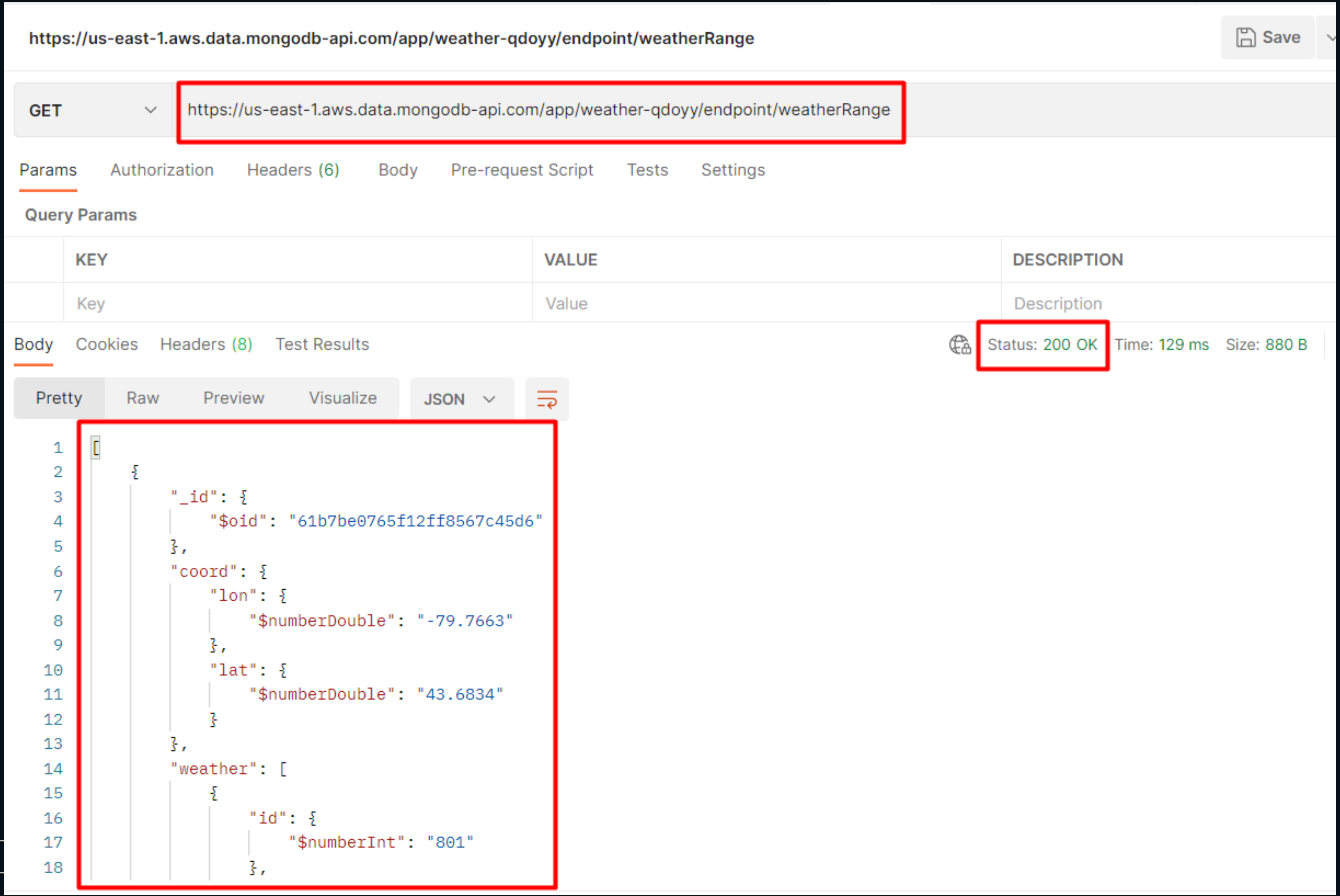
Status: 200 OK Time: 644 ms Size: 1.02 KB

```
[
    {
        "_id": {
            "$oid": "61b7be0765f12ff8567c45d6"
        },
        "coord": {
            "lon": {
                "$numberDouble": "-79.7663"
            },
            "lat": {
                "$numberDouble": "43.6834"
            }
        },
        "weather": [
            {
```

W.here

LIVE API Get a range of items

https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weatherRange



https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weatherRange

GET https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weatherRange

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

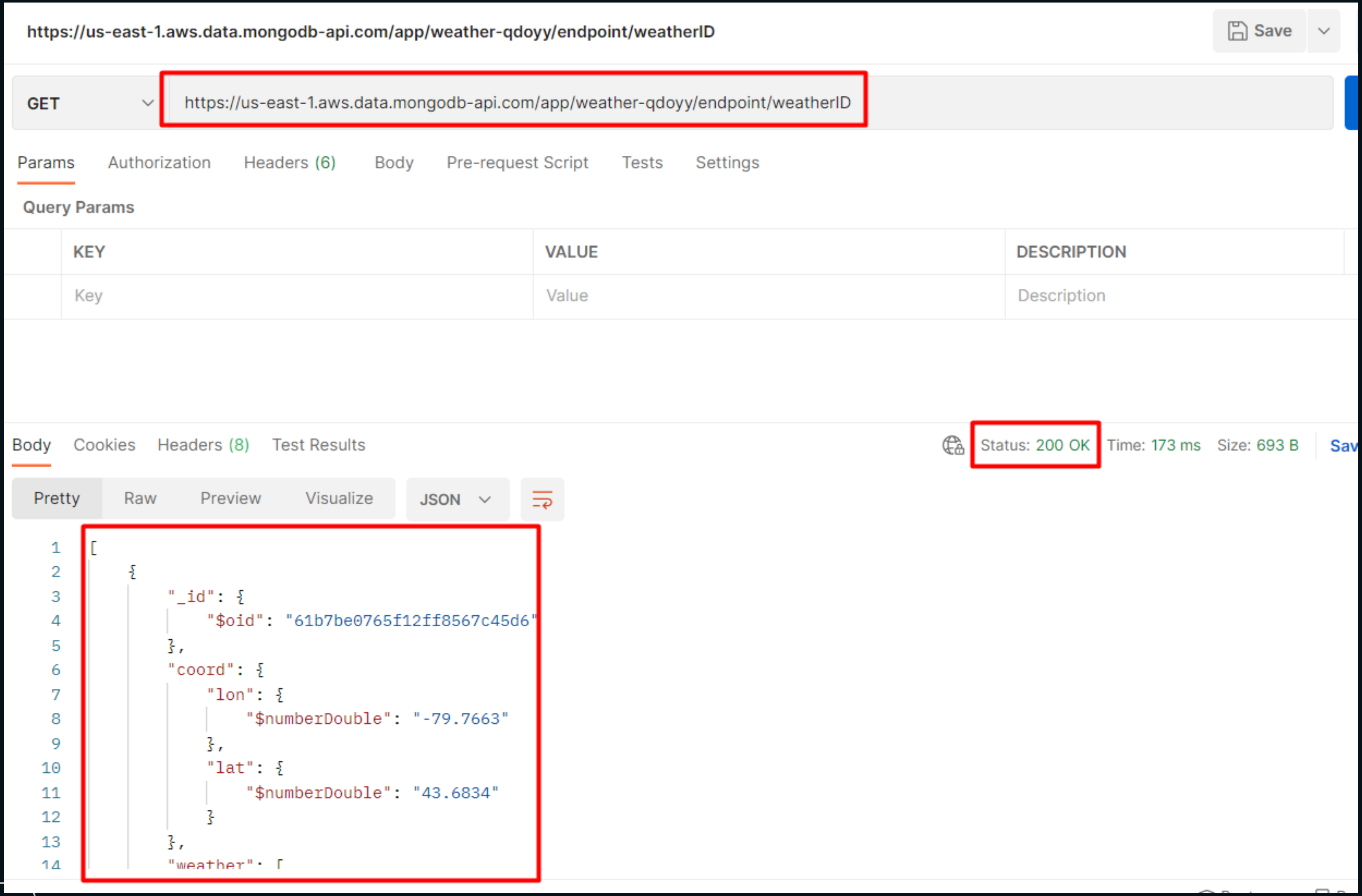
Status: 200 OK Time: 129 ms Size: 880 B

```
[
    {
        "_id": {
            "$oid": "61b7be0765f12ff8567c45d6"
        },
        "coord": {
            "lon": {
                "$numberDouble": "-79.7663"
            },
            "lat": {
                "$numberDouble": "43.6834"
            }
        },
        "weather": [
            {
                "id": {
                    "$numberInt": "801"
                },
```

W.here

LIVE API Get item by ID

https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weatherID


https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weatherID

GET https://us-east-1.aws.data.mongodb-api.com/app/weather-qdoyy/endpoint/weatherID

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

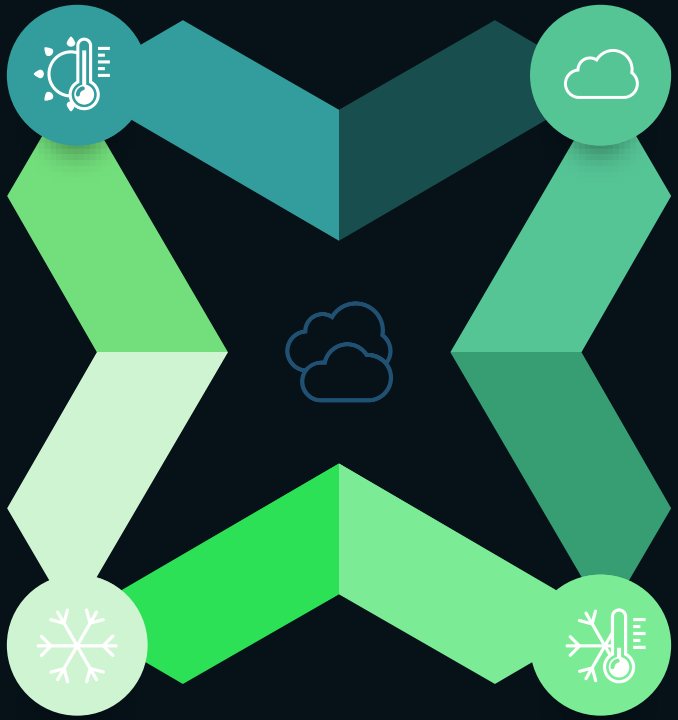
Body Cookies Headers (8) Test Results — Status: 200 OK Time: 173 ms Size: 693 B

Pretty Raw Preview Visualize JSON

```
[
    {
        "_id": {
            "$oid": "61b7be0765f12ff8567c45d6"
        },
        "coord": {
            "lon": {
                "$numberDouble": "-79.7663"
            },
            "lat": {
                "$numberDouble": "43.6834"
            }
        },
        "weather": [
```

W.here

FIT GAP ANALYSIS



One card for any 4 cities weather as per user preferences

Weather forecasting feature can be added

Default API response and other API limitations can be worked around

More Visualizations could be performed

LIVE DEMONSTRATION

YouTube Link :

https://www.youtube.com/watch?v=zbw9nL6vRSQ

GitHub Link :

https://github.com/bhavi-dudhat/DPFinalproject

Weather Dashboard

https://weather-dashboard-dp.herokuapp.com/

THANK YOU