**Assignment: Python Programming for GUI**

**Development**

Name: Devika.G

Register Number: 192371086

Department: CSE(Bioscience)

Date of Submission: 26.08.2024

**Problem: Air Pollution API**

1. Current, forecast and historical air pollution data
2. Forecast for 4 days ahead with 1-hour step
3. Air Pollution API includes both Air Quality Index and indices for CO, NO, NO2, O3, SO2, NH3,PM2.5, PM10 .
4. Included in both free and paid subscriptions

**Deliverables:**

1. Data flow diagram illustrating the interaction between the application and the API .
2. Pseudo code and implementation
3. Documentation of the API integration and the methods used to fetch and display the data.
4. Explanation of any assumptions made and potential improvements

**Solution:**

## Air Pollution API

**1. Data Flow Diagram**

Start

**Define API Key and City**

**Construct API Request URL**

**Make API Request**

**Check Response Status**

**Failure**

**success**

**Handle HTTP Error**

**Validate Response Data**

**Data Valid?**

**End**

**Parse Response Data**

**Handle Data Validation Error**

**|**

**Check Data Consistency Data**

**Plot Data**

**Response Status**

**Save Plot**

**Print Statistics**

**Response Status**

**end**

**2. Implementation**

import requests

def get\_air\_pollution\_data(api\_key,lat,lon):

url =

f"http://api.openweathermap.org/data/2.5/air\_pollution?lat={lat}&lon={lon}&appid

{api \_ key}

response=requests.get(url)

if response.status\_code==200:

data=response.json()

aqi=data['list'][0]['main']['aqi']

components=data['list'][0]['components']

print("Air Quality Index (AQI):", aqi)

print("Concentration of pollutants in µg/m³:")

print("CO:", components['co'])

print("NO:", components['no'])

print("NO2:",components['no2'])

print("O3:",components['o3'])

print("SO2:",components['so2'])

print("PM2.5:",components['pm2\_5'])

print("PM10:",components['pm10'])

print("NH3:",components['nh3'])

else:

print("Failed to retrieve data. HTTP Status code:", response .status \_code)

if \_name\_ == "\_main\_":

api \_ key = "7388dca5f7653c38ee1f85baa3f7a900"

lat = "37.7749"

lon = "-122.4194"

get \_ air \_ pollution \_data( api \_key, lat, lon)

**3. Display the air pollution information**

Air Quality Index (AQI): 1

Concentration of pollutants in µg/m³:

CO: 220.3

NO: 0.5

NO2: 3.26

O3: 49.35

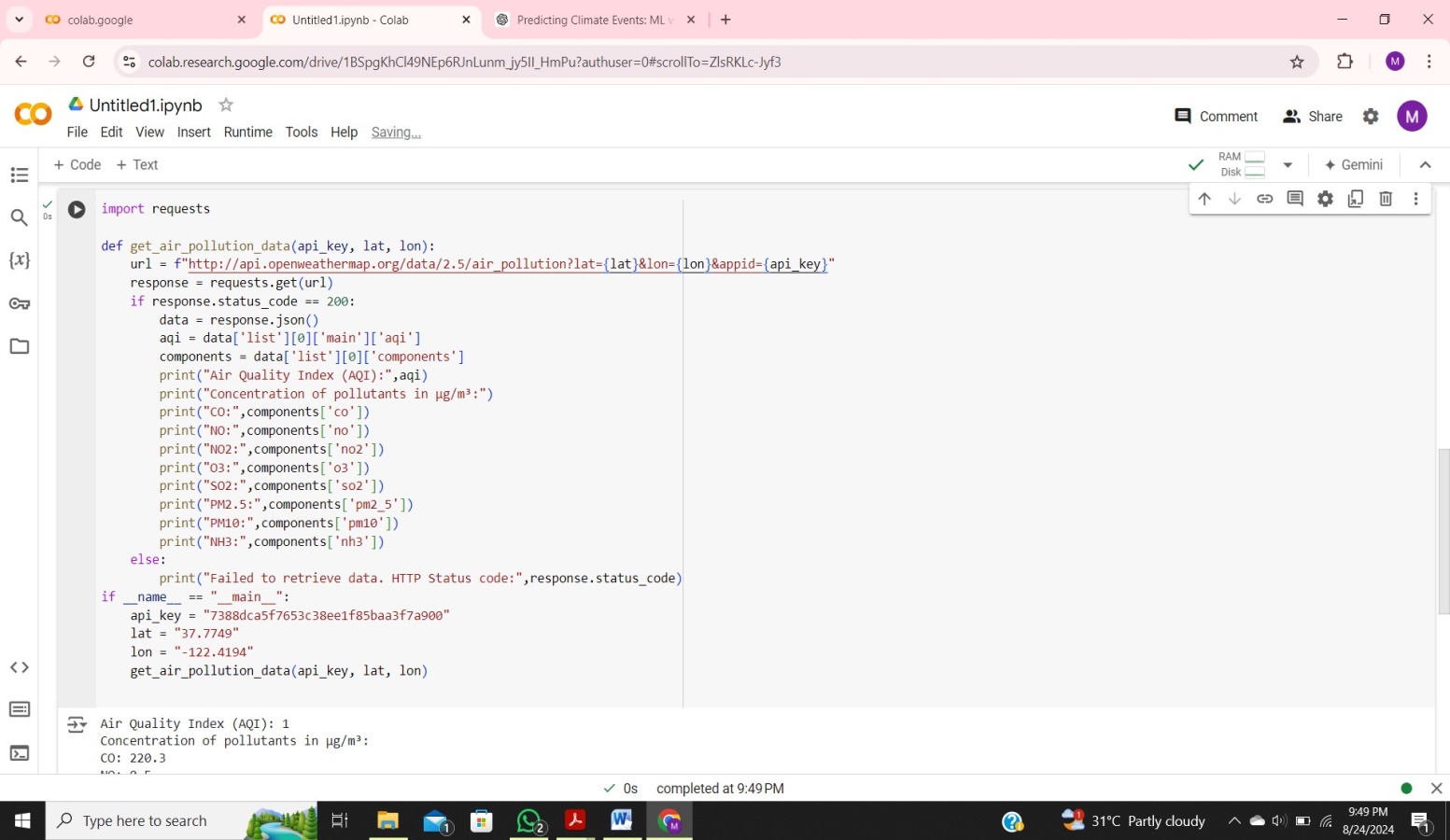
SO2: 0.54

PM2.5: 1.79

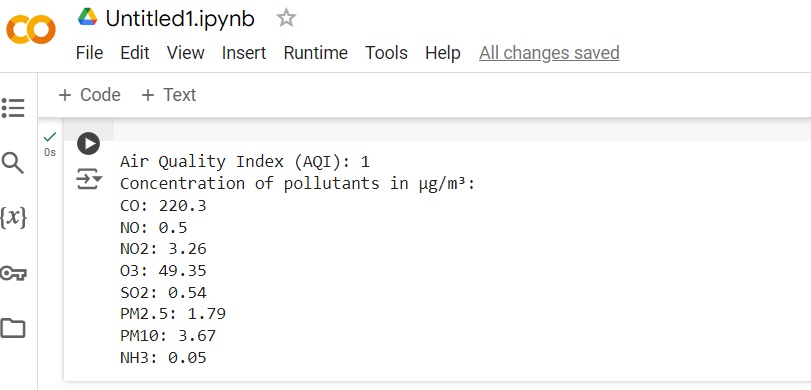
PM10: 3.67

NH3: 0.05

**4. User Input**



**5. user output:**

****

**6. Documentation**

**Explanation of the code:**

**Imports**: Uses the `requests` library to make HTTP requests.

**Function Definition**: `get\_air\_pollution\_data (api \_key,lat ,lon)` fetches air pollution data from OpenWeatherMap based on latitude and longitude.

**API Request**: Constructs a URL with the API key and coordinates, then sends a GET request to the API.

**Response Handling**: Checks if the request was successful (status code 200). If so, parses the JSON response.

**Data Extraction**: Retrieves the Air Quality Index (AQI) and pollutant concentrations from the response.

**Output**: Prints the AQI and concentrations of pollutants in µg/m³.

**Error Handling**: If the request fails, prints an error message.

**Usage Example**: Replaces `"YOUR\_API\_KEY"` with your actual API key and specifies the coordinates to fetch and display the data.

**Assumptions made:**

**API Key Validity**: Assumes that the provided API key is valid, has not expired, and has sufficient permissions to access the air pollution data.

**Endpoint Availability**: Assumes that the API endpoint (`http://api.openweathermap.org/data/2.5/air\_ pollution`) is available and functioning correctly.

**Response Format**: Assumes that the API response will follow the documented format. For OpenWeatherMap, this includes having a `'list'` key with pollution data, which contains `'main'` for AQI and `'components'` for pollutant concentrations.

**Geographical Coordinates**: Assumes that the latitude and longitude coordinates provided are valid and correctly formatted. The API should be able to return data based on these coordinates.

**Data Completeness**: Assumes that the API response will include all expected data fields, such as AQI and pollutant concentrations. In case of missing data, additional handling might be needed.

**Rate Limits**: Assumes that the API usage adheres to rate limits and usage policies specified by the API provider. Exceeding these limits might result in throttling or denial of service.

**Localization:** Assumes that the data returned by the API is in a standard format, usually in a universal unit like µg/m³ (micrograms per cubic meter) for pollutant concentrations.

**Error Handling**: Assumes that the code has appropriate error handling for unexpected responses, such as API outages or errors in the request.

**Limitations:**

**Data Accuracy**: Variable accuracy and coverage, especially in less monitored areas.

**API Limits**: Rate and quota limits restrict the number of requests and data volume.

**Granularity**: Limited parameters and data resolution may affect detail and precision.

**Cost**: Higher-quality or more frequent data often requires a paid subscription.

**Reliability:** Dependence on third-party service availability and potential API errors.