Assignment: Python Programming for GUI Development

Name: G.Veerendra

Register Number:192365075

Department:computer science(cyber security)

Date of Submission: 26-08-2024

Problem 3: Real Time Covid 19 Statistical Tracker

Scenario:

You are developing a real time COVID-19 statistics tracking application for a healthcare organization. The application should provide up-to-date information on COVID-19 cases,recoveries,and deaths for a specified region

Tasks:

1. **Model of the data flow for fetching COVID-19 statistics from an external API and displaying it to the user.**
2. **Implement a Python application that integrates with a COVI-19 statistics API (e.g.,disease.sh) to fetch real-time data.**
3. **Display the current number of cases, recoveries and deaths for a specified region.**
4. **Allow users input a region (country,state or a city) and display the corresponding COVID-19 statistics.**

Deliverables:

- ➢ Data flow diagram illustrating the interaction between the application and the API.
- ➢ Pseudocode and implementation of the COVID-19 statistics tracking application.
- ➢ Documentation of the API integration and the methods used to fetch and display COVID-19 data..
- ➢ Explanation of any assumptions made and potential implementation.

Real Time COVID-19 Statistical Tracker

1.Data flow diagram:

```
+---------------+
| API Request |
+---------------+
      |
      |
      v
+---------------+
| API Response |
| (JSON Payload) |
+---------------+
      |
      |
      v
+---------------+
| Data Processing |
| (Python Script) |
+---------------+
      |
      |
      V
```

```
+---------------+
| Data Storage  |
| (Database or  |
|  Data Storage |
|  Service)     |
+---------------+
      |
      |
      v
+---------------+
| Data Visualization|
| (Python Script) |
+---------------+
      |
      |
      |
      |
      v
+---------------+
| User Interaction|
| (Web Interface) |
+---------------+
```

2.Implementation:

```
import requests

url = "https://covid-19-statistics.p.rapidapi.com/regions"
headers = {
        "x-rapidapi-key":
"9ee00c50d3msh948a3782f96e5cfp1e6f3bjsn43c748581cef",
        "x-rapidapi-host": "covid-19-statistics.p.rapidapi.com"
}
response = requests.get(url, headers=headers)
print(response.json())
```

3. Display the codes of the regions:

The above code displays the codes of the every region which we have to enter as the input in the main code after giving the requests

4.User Input:

5.Implementation:

```
import requests

# Set API URL and API key
url = "https://covid-193.p.rapidapi.com/statistics"
headers = {
        "x-rapidapi-key":
"9ee00c50d3msh948a3782f96e5cfp1e6f3bjsn43c748581cef",
        "x-rapidapi-host": "covid-193.p.rapidapi.com"
}
# Get user input for country name
country_name = input("Enter a country name (e.g. United States, Italy,
India, etc.): ")
# Send GET request to API URL
response=requests.get(url,headers=headers,params={"country":country
name})
# Check if API call was successful
if response.status_code == 200:
    # Parse JSON response
    data = response.json()
    # Extract and print COVID-19 statistics
    if len(data["response"]) > 0:
        cases = data["response"][0]["cases"]["total"]
```

```
        deaths = data["response"][0]["deaths"]["total"]
        recovered = data["response"][0]["cases"]["recovered"]
        today_cases = data["response"][0]["cases"]["new"]
        today_deaths = data["response"][0]["deaths"]["new"]
        print(f"Live COVID-19 Statistics for {country_name}:")
        print(f"Cases: {cases}")
        print(f"Deaths: {deaths}")
        print(f"Recovered: {recovered}")
        print(f"Today's Cases: {today_cases}")
        print(f"Today's Deaths: {today_deaths}")
    else:
        print(f"No data found for {country_name}.")
else:
    print("Error:", response.status_code)
```

6. Display the COVID-19 Global Statistics:

Live COVID-19 Statistics for India:

Cases: 45035393

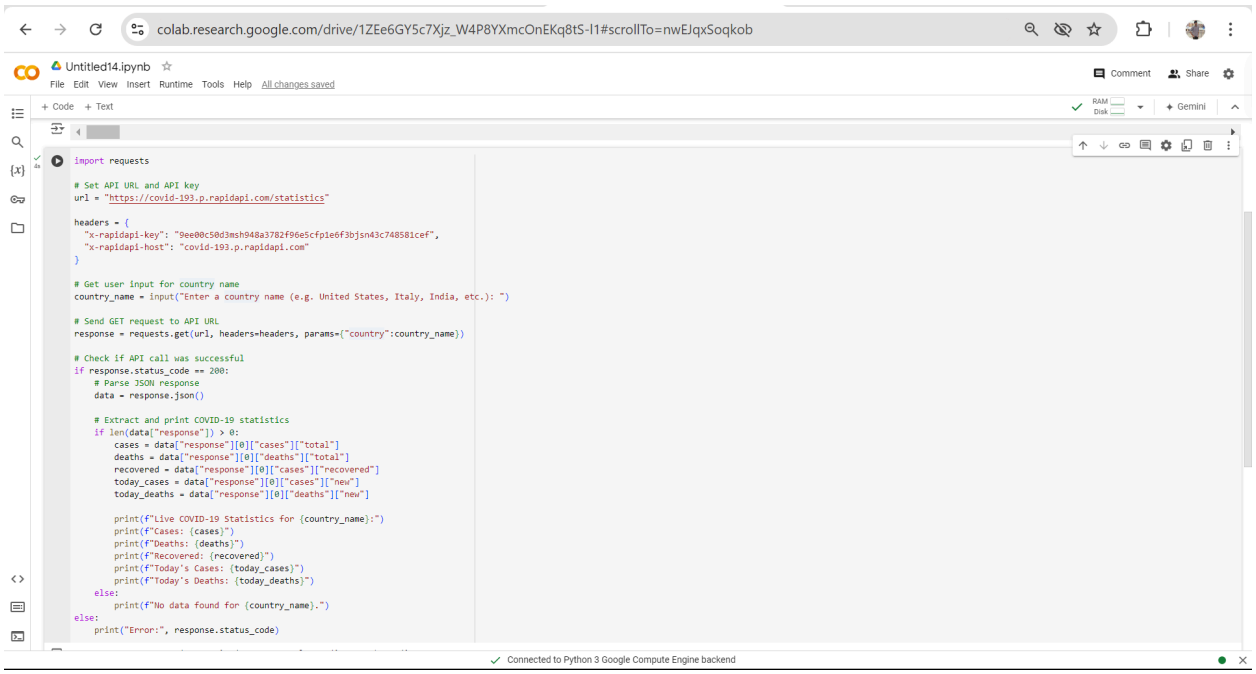
Deaths: 533570

Recovered: None

Today's Cases: None

Today's Deaths: None

7.User Input:



8.Documentation:

Giving and taking requests through importing requests

import requests import json

Set API endpoint

api_endpoint = "https://disease.sh/v3/covid-19/countries"

Get user input for country

country = input("Enter a country: ")

Make API call with query parameter

params = { "query": country } response = requests.get(api_endpoint, params=params)

Check if API call was successful

if response.status_code == 200: # Parse JSON response data = json.loads(response.text)

```
# Extract and print COVID-19 statistics
if len(data) > 0:
    cases = data[0]["cases"]
    deaths = data[0]["deaths"]
    recovered = data[0]["recovered"]
    today_cases = data[0]["todayCases"]
    today_deaths = data[0]["todayDeaths"]
    print(f"Live COVID-19 Statistics for {country}:")
    print(f"Cases: {cases}")
    print(f"Deaths: {deaths}")
    print(f"Recovered: {recovered}")
    print(f"Today's Cases: {today_cases}")
    print(f"Today's Deaths: {today_deaths}")
else:
    print(f"No data found for {country}.")
else: print("Error:", response.status_code)
```

Detailed explanation of the actual code:

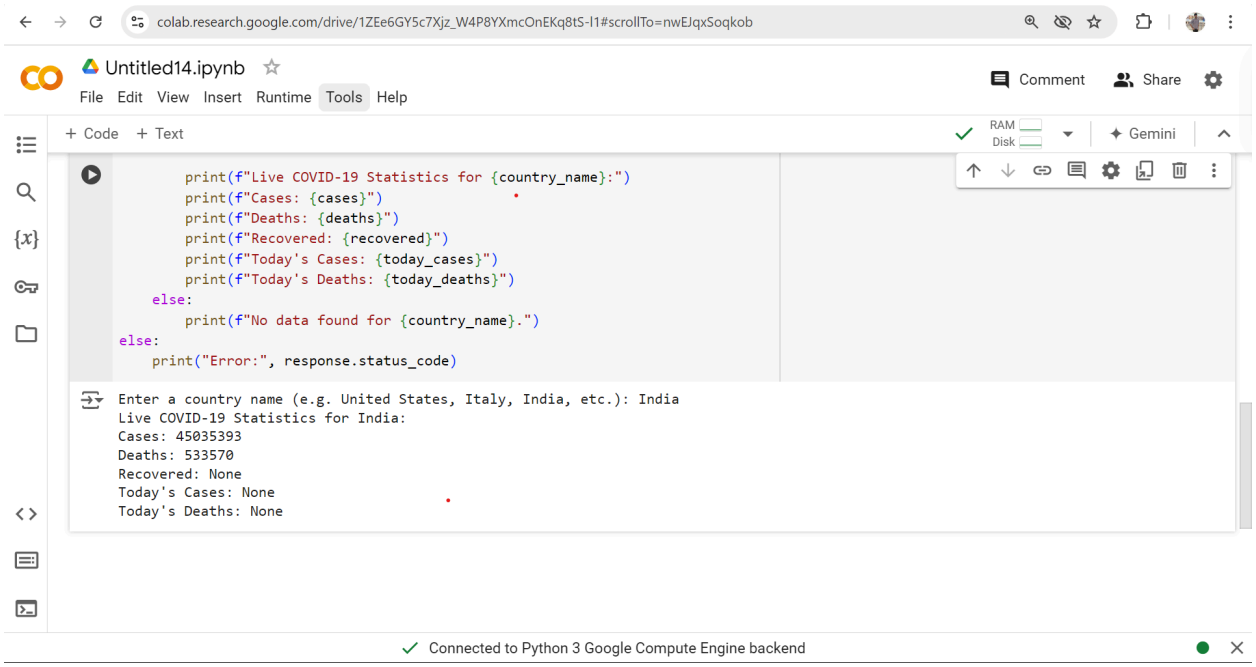
- ➢ This application uses the requests library to make HTTP requests to the COVID-19 API provided by disease.sh. The get_ covid_ stats function takes a region (country, state, or city) as input and returns the current number of cases, recoveries, and deaths for that region.
- ➢ This display_ covid_ stats function is responsible for formatting and printing the COVID-19 statistics in a user-friendly way. It takes the cases, recoveries, and deaths data as input and displays them with appropriate formatting (e.g., adding commas to large numbers).
- ➢ The main function is the entry point of the application. It prompts the user to enter a region, calls the get_ covid_ stats function to fetch the data, and then passes the results to the display_ covid_ stats function to display the information.

Assumptions made (if any):

- The application assumes that the disease.sh API is available and providing accurate real-time COVID-19 data.
- The application assumes that the user will input a valid region (country, state, or city) that the API can recognize.
- Potential Improvements:
- Add error handling to the application to gracefully handle API errors or invalid user input.
- Provide additional features, such as the ability to display historical COVID-19 data, trends, or visualizations.
- Integrate the application with a user interface (e.g., a web application or a mobile app) to improve the user experience.

- Allow users to select multiple regions and compare the COVID-19 statistics side-by-side.
- Provide the ability to set alerts or notifications for significant changes in COVID-19 statistics.

9.Sample Output:



Limitations:

1. The API may have rate limits that restrict the number of requests.
2. The data may not always be up-to-date due to delays in reporting.
3. The application currently only handles countries; state and city-level queries may require additional endpoints.