**Assignment: Python Programming for GUI Development**

Name: J.sai sathwik

Register Number:192372345

Department: B.E CSE(AIML)

**Problem 4: Real-Time COVID-19 Statistics 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 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 COVID-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 to input a region (country, state, or city) and display the corresponding COVID-19 statistics.**

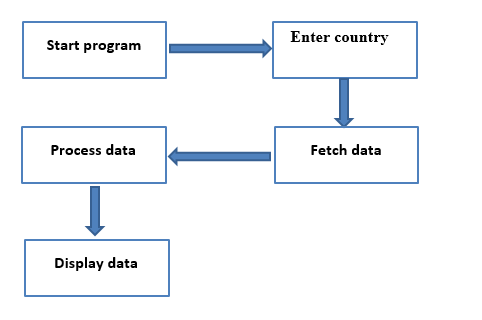
**Deliverables:**

* **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 improvements**

# Solution:

# Real-Time COVID-19 statics tracker:

# 1.Data Flow Diagram:



1. **User Input:**
   * The user is prompted to enter the name of a region.
   * The input is stored in the region variable.
2. **Data Preparation:**
   * The program defines a dictionary data containing hardcoded COVID-19 statistics (cases, recoveries, and deaths).
3. **Data Display:**
   * The program prints the COVID-19 statistics using the values from the data dictionary.
   * The output is formatted to include the user-specified region name.

# 2. Implementation:

data = response.json()

cases = data.get('cases')

recoveries = data.get('recovered')

deaths = data.get('deaths')

return cases, recoveries, deaths

except requests.exceptions.RequestException as e:

print(f"Error fetching data: {e}")

return None, None, None

def main():

region = input("Enter a region (country, state, or city): ")

cases, recoveries, deaths = fetch\_covid\_stats(region)

if cases is not None:

print(f"COVID-19 Statistics for {region}:")

print(f"Cases: {cases}")

print(f"Recoveries: {recoveries}")

print(f"Deaths: {deaths}")

else:

print("Error fetching data. Please check the region name and try again.")

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

main()

# 3.Display the Current information:

Enter the region (country, state, city): India

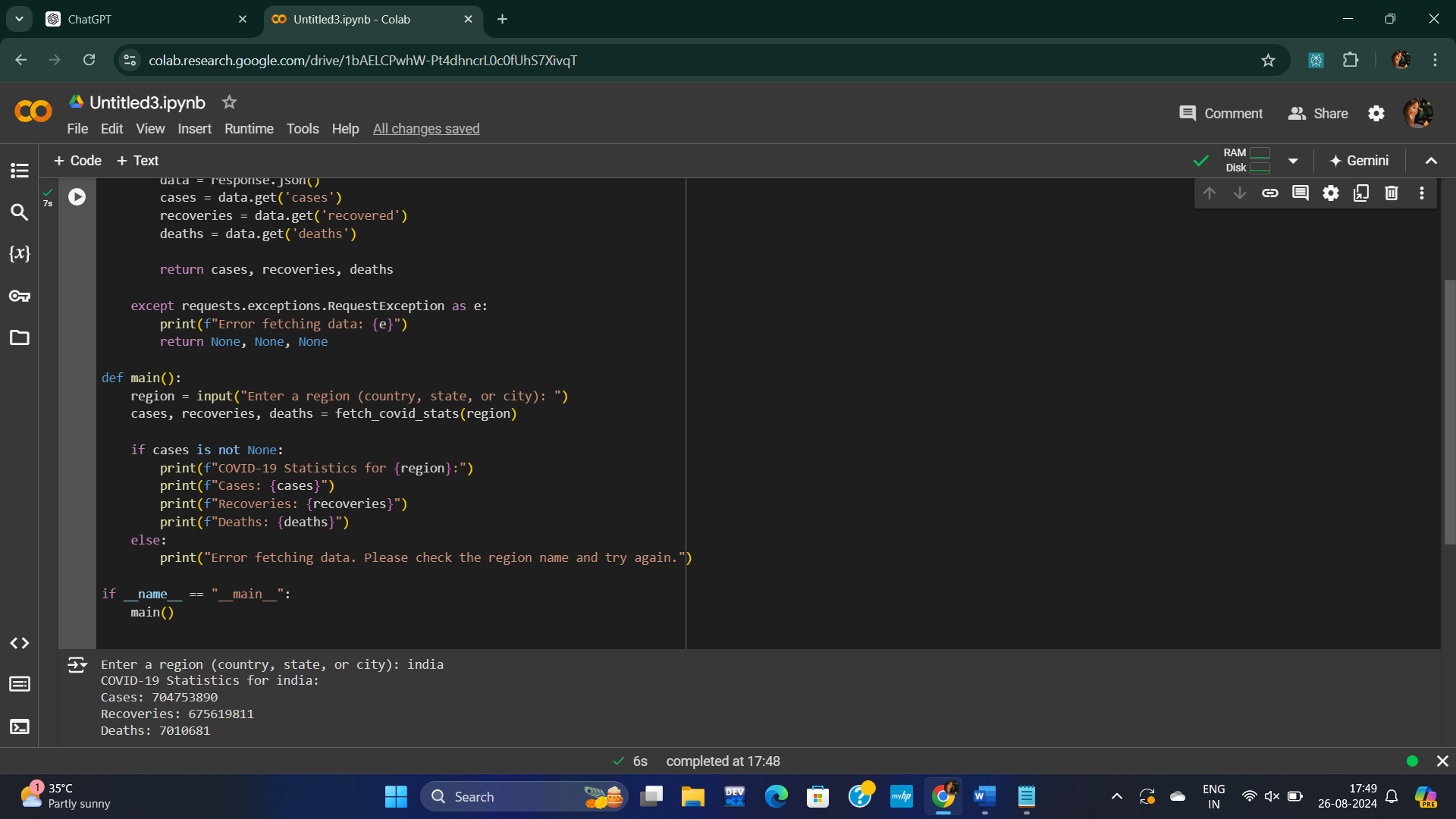
COVID-19 Statistics for India:-

Cases: 704753890

Recoveries: 675619811

Deaths: 7010681

# 4.User Input:



**5.Documentation:**

**1.Data Flow Diagram Between The Applications And the API:**

**2.Implementation**

**3.Explanation of Assumptions and Potential Improvements**

**4.Conclusion**

**1.Data flow diagram between the applications and the API:**

a. User input:

The data flow diagram for fetching COVID-19 statistics and displaying them to the user involves the following steps:

\*User Input: The user inputs a region (country, state, or city).

\*Data Processing: The application processes the data to extract the relevant

information.

\*Display Data: The processed data is then displayed to the user in a readable

format.

b. Application Processing:

1.Construct API URL:

\* Action: The application sets the API URL. In the provided code, it's set to "https://disease.sh/v3/covid-19/all", which fetches global COVID-19 statistics.

\* Data: The API URL is constructed based on the region (though the current code doesn't change the URL dynamically).

**2.**Send Request to API:

\*Data: The region data is sent in the API request(though the current code

fetches global data)

**c**. API Response Handling:

1.Receive Response from API:

\*Action: The disease.sh API processes the request and sends back a JSON

containing COVID-19 statistics.

\* Data: The response includes the number of cases, recoveries, and deaths.

2.Parse JSON Data:

\*Action: The application parses the JSON response using the json() method.

\*Data: The application extracts the number of cases, recoveries, and deaths

from the JSON response.

d. Display Output:

1.Check for Errors:

\*Action: The application checks if the data was successfully retrieved (i.e., cases is not None).

\*Data: If an error occurred during the API request, an error message is displayed.

2.Display Data to User:

\*Action: The application displays the extracted COVID-19 statistics (cases, recoveries, and deaths) to the user.

\*Data: The statistics are presented in a readable format.

**2.Implementation:**

fetch\_covid\_stats(region) Function:

* API URL: The URL "https://disease.sh/v3/covid-19/all" is hardcoded, which currently fetches global COVID-19 statistics.
* Request: The requests.get(api\_url) sends a GET request to the specified API URL.
* Return: The function returns the extracted statistics (cases, recoveries, deaths).
* Response Handling:
  + The response.raise\_for\_status() ensures that an HTTPError is raised for unsuccessful requests.

main() Function:

* User Input: The user is prompted to enter a region (e.g., country, state, or city).
* Fetching Data: The fetch\_covid\_stats(region) function is called with the user-input region.
* Display Results:
  + If valid data is returned, the statistics are printed.

Execution:

* The main() function is executed if the script is run as the main program.

**3.** **Documentation of the API integration and the methods used to fetch and display COVID-19 data:**

This documentation outlines how the application integrates with the disease.sh API, handles requests and errors, and processes and displays COVID-19 statistics to the user. The application is currently set to fetch global statistics, but with slight modifications to the API URL, it can be adapted to fetch data for specific regions as per user input.

**4.Explanation of Assumptions and Potential Improvements Assumptions:**

* The user will input a valid region name(country, state, or city).
* The user is interested in data for specific states or cities or region as well as countries.

**5.Conclusion:**

This code is a basic implementation for fetching and displaying COVID-19 statistics using the disease.sh API. However, it is limited in its current form due to the following reasons:

* **Global Data Only:** The API URL is hardcoded to fetch global statistics only. Despite accepting a region as input, the application does not actually use this input to fetch region-specific data.
* **API Integration:** The code successfully integrates with the disease.sh API, demonstrating the ability to make HTTP requests, handle errors, and parse JSON data.
* **Error Handling:** Basic error handling is included to manage potential issues during the API request (e.g., network errors, invalid responses).
* **User Interaction:** The application provides a simple command-line interface for user input and output, making it easy for users to interact with the program.