**Assignment: Python Programming for Innovation**

Name: A.Dhatrika

Register Number:192311429

Department: CSE

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:

**COVID-19 STATISTICS TRACKING APPLICATION:-**

1.Data Flow Diagram

**Start**

**End**

COVID-19 Statistics API

Python Application

User Input (Region)

API Request

JSON Response

Display COVID-19 Statistics

**Implementation:**

**COVID-19 STATISTICS TRACKING APPLICATION:-**

import requests

class Covid19StatisticsTracker:

def \_\_init\_\_(self):

self.base\_url = 'https://disease.sh/v3/covid-19'

def fetch\_covid\_stats(self, region):

endpoint = f'/countries/{region}' if region else '/all'

url = self.base\_url + endpoint

try:

response = requests.get(url)

data = response.json()

if response.status\_code == 200:

return data # Return JSON data received from API

else:

print(f"Error fetching data: {data['message']}")

return None

except requests.exceptions.RequestException as e:

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

return None

def display\_covid\_stats(self, data):

if data:

# Extract relevant COVID-19 statistics

cases = data.get('cases', 'N/A')

deaths = data.get('deaths', 'N/A')

recovered = data.get('recovered', 'N/A')

active = data.get('active', 'N/A')

# Displaying COVID-19 statistics

print(f"COVID-19 Statistics:")

print(f"Total Cases: {cases}")

print(f"Total Deaths: {deaths}")

print(f"Total Recovered: {recovered}")

print(f"Active Cases: {active}")

else:

print("No COVID-19 data available for the specified region.")

# Example usage:

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

tracker = Covid19StatisticsTracker()

region = input("Enter country name (leave blank for global stats): ")

covid\_data = tracker.fetch\_covid\_stats(region)

tracker.display\_covid\_stats(covid\_data)

**Output:**

Enter country name (leave blank for global stats): india

COVID-19 Statistics:

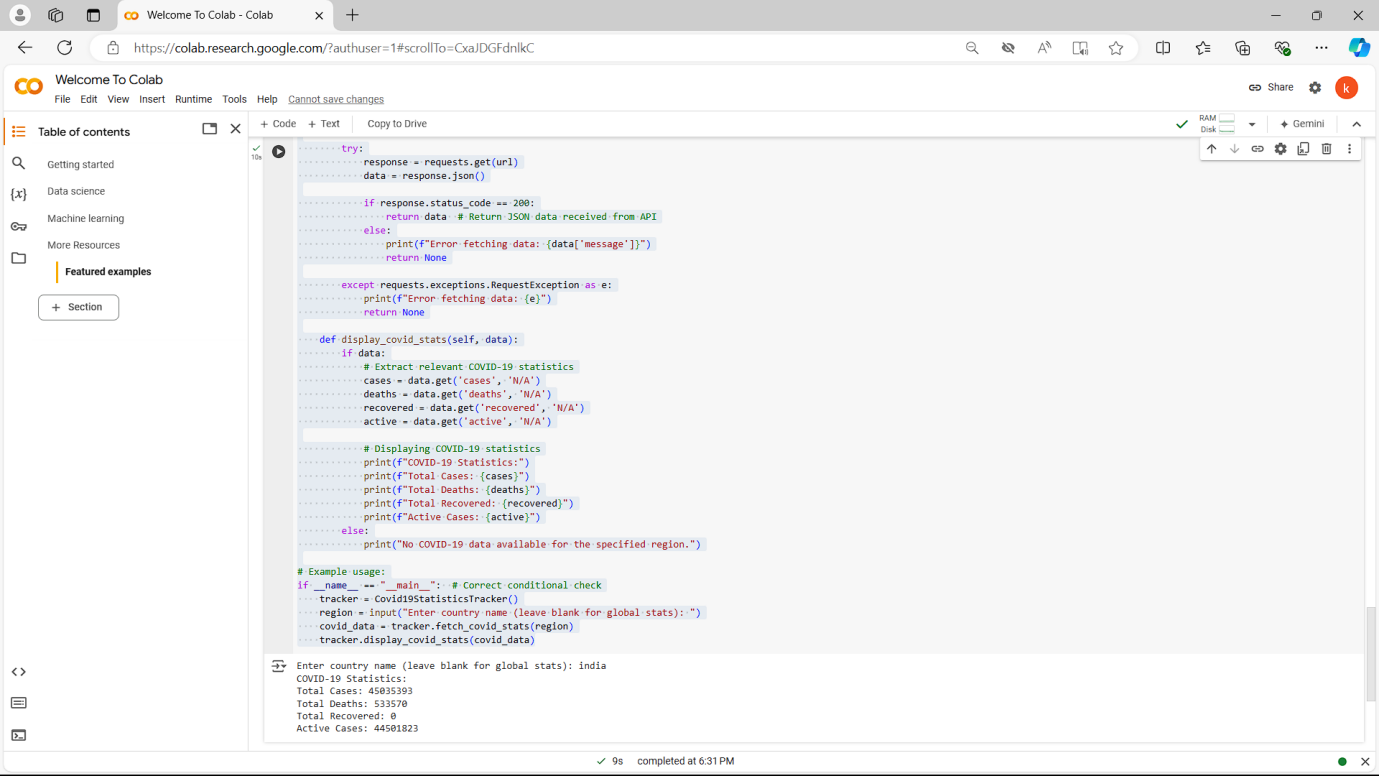
Total Cases: 45035393

Total Deaths: 533570

Total Recovered: 0

Active Cases: 44501823

**User Input:**



**Documentation**

**API Integration and Methods:**

**API Integration:**

The application integrates with the disease.sh COVID-19 API to fetch real-time statistics.

Constructs the API request dynamically based on user input (country name) to fetch statistics for a specific region or global statistics.

Fetch COVID-19 Statistics (fetch\_covid\_stats method):

Sends an HTTP GET request to the appropriate endpoint (/all for global stats or /countries/{country} for country-specific stats).

Handles JSON response to extract COVID-19 statistics including total cases, deaths, recoveries, and active cases.

**Display COVID-19 Statistics (display\_covid\_stats method):**

Parses JSON response to extract and display relevant COVID-19 statistics.

Handles cases where no data is available for the specified region.

**Assumptions:**

**Data Availability:** Assumes real-time availability and updates from the disease.sh API.

**Region Input:** Users input country names primarily; can be extended to handle state or city names based on API capabilities.

**API Reliability:** Assumes the disease.sh API is reliable and returns accurate data.

**Potential Improvements:**

**Historical Data:** Incorporate historical COVID-19 data for trend analysis and visualization.

**Visualization:** Develop graphical charts and maps to visualize COVID-19 statistics for easier interpretation.

**Notifications:** Implement notifications or alerts for significant changes in COVID-19 statistics.

**Localized Data:** Enhance support for retrieving and displaying regional data (e.g., state-level data within a country).

By implementing these improvements, the COVID-19 statistics tracking application can provide more comprehensive insights and support for decision-making within healthcare organizations and public health agencies.