**Assignment: Python Programming for GUI Development**

**Name:** SMD. Khaja Niyaz

**Register Number:**192311371

**Department:** Computer Science Engineering

**Date of Submission:**

**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.**

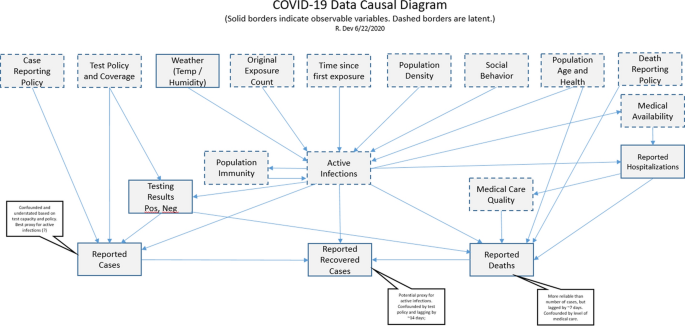
**Deliverables:**

* 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 Statistics Tracker**

**1.Data Flow Diagram:**



**2. Implementation:**

import requests

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

response = requests.get(url)

data = response.json()

# Accessing the data

cases = data["cases"]

deaths = data["deaths"]

recovered = data["recovered"]

# Printing the data

print("COVID-19 Statistics for India:")

print("-------------------------------")

print(f"Cases today: {data['todayCases']}")

print(f"Deaths today: {data['todayDeaths']}")

print(f"Recovered today: {data['todayRecovered']}")

print(f"Active cases: {data['active']}")

print(f"Critical cases: {data['critical']}")

print(f"Cases per million: {data['casesPerOneMillion']}")

print(f"Deaths per million: {data['deathsPerOneMillion']}")

print(f"Tests done: {data['tests']}")

print(f"Tests per million: {data['testsPerOneMillion']}")

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

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

print(f"Total recovered: {recovered}"

**3.Display of COVID-12 Statistics information**

COVID-19 Statistics for India:

-------------------------------

Cases today: 0

Deaths today: 0

Recovered today: 0

Active cases: 44501823

Critical cases: 0

Cases per million: 32016

Deaths per million: 379

Tests done: 935879495

Tests per million: 665334

Total cases: 45035393

Total deaths: 533570

Total recovered: 0

**4.User Input**



**5.Documentation:**

**1. Introduction:-**

**Overview:** Briefly describe what the COVID-19 statistics tracker does, its purpose, and its importance.

**Features**: List the main features of the tracker, such as data visualization, daily updates, geographical filters, etc.

**Audience:** Specify who the intended audience is (e.g., public users, healthcare professionals, data analysts, developers).

**2. Getting Started:-**

**Installation**: Provide step-by-step instructions on how to install the tracker, including system requirements and dependencies**.**

**Configuration**: Describe any configuration settings that need to be adjusted for the tracker to work properly, such as API keys, database connections, etc.

**Running the Tracker:** Explain how to start and stop the tracker, including any commands or scripts.

**3. User Guide:-**

**Navigating the Interface:** Explain the layout and functionality of the user interface, including menus, buttons, and options.

**Viewing Statistics**: Guide users on how to view different statistics, such as total cases, recoveries, deaths, etc.

**Filters and Search:** Describe how users can filter data by date, location, or other criteria.

**Data Export:** Explain how users can export data for external use, such as CSV or Excel formats.

**4. Developer Guide:-**

**Architecture Overview**: Provide a high-level overview of the system architecture, including the main components and their interactions.

**Codebase Structure**: Explain the organization of the codebase, including directories and key files.

**APIs and Endpoints**: Document the APIs provided by the tracker, including available endpoints, request formats, and response formats.

**Data Sources:** Detail the external data sources used by the tracker, such as public health APIs or datasets, and how they are integrated.

**Customization and Extension:** Guide developers on how to customize or extend the functionality of the tracker.

**5. Administrator Guide**

**User Management:** Describe how to manage user accounts, roles,and permissions.

**Data Updates:** Explain how the system updates its data and how administrators can manually update or backfill data if necessary.

**Monitoring and Logging**: Provide information on how to monitor system performance and access logs for troubleshooting.

**Backup and Recovery:** Outline procedures for backing up data and recovering the system in case of failure.

**6. Troubleshooting**

**Common Issues:** List common problems users or administrators might encounter and provide solutions or workarounds.

**Contact Support:** Provide contact information or resources for additional support if needed.

**7. Security and Privacy**

**Data Security:** Describe the measures taken to secure the datawithin the tracker, including encryption, access controls, etc.

**Privacy Policy:** Outline the privacy policy, especially if the tracker collects any personal data.

**8. Appendix:-**

**Glossary**: Define any technical terms or abbreviations used throughout the documentation.

**References:** Provide references to any external documents, websites, or tools mentioned.

**9. Changelog:-**

**Version History**: Keep a log of changes, updates, and bug fixes to track the development history of the tracker.