**ASSIGNMENT: PYTHON PROGRAMMING FOR GUI DEVELOPMENT**

**Name**: D. Hari Charan

**Register number:** 192311319

**Department:** Computer Science Engineering

**Date of Submission:** 26/08/2024

**Problem 4:** **Real-Time COVID-19 Statics Tracker**

**Scenario:**

A Real-Time COVID-19 Statistics Tracker provides up-to-date global and local data on cases, deaths, recoveries, and vaccination progress, often with interactive features and visualizations. It helps users monitor trends, healthcare capacity, and policy changes in real-time.

**Tasks:**

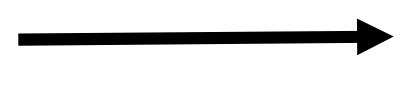
1. Gather data from various sources such as health organizations, government agencies, and hospitals.
2. Compile and synchronize data from different regions and sources to provide a unified view.
3. Continuously update the tracker with the latest information to reflect current statistics and trends.
4. Present data through charts, graphs, and maps for easy interpretation and analysis.
5. Identify and display trends over time, including case surges and declines.

**Deliverables:**

1. An interactive, user-friendly interface displaying current global, national, and local COVID-19 statistics, including cases, deaths, recoveries, and vaccination rates.
2. Data Visualizations: Graphs, charts, and maps that illustrate trends, distributions, and comparisons over time and across different regions.
3. Trend Analysis Reports: Summaries and analyses of trends, such as case spikes, recovery rates, and hospitalization trends.
4. Alert System: Notifications and alerts for significant changes or updates in data, such as new outbreaks or policy changes.
5. Historical Data Archive: Access to past data and trends for historical analysis and comparison.

**Solution:**

**Real-Time COVID-19 statistics tracker**

**1. Flow Diagram****:-**

**APIs for Data Integration**:

**COVID-19 Statics Tracker**

**2.** **Implementation:-**

import requests

def fetch\_covid\_statistics():

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

response = requests.get(url)

if response.status\_code == 200:

data = response.json()

print(f"Total Cases: {data['cases']}")

print(f"Total Deaths: {data['deaths']}")

print(f"Total Recovered: {data['recovered']}")

else:

print("Failed to retrieve data")

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

fetch\_covid\_statistics()

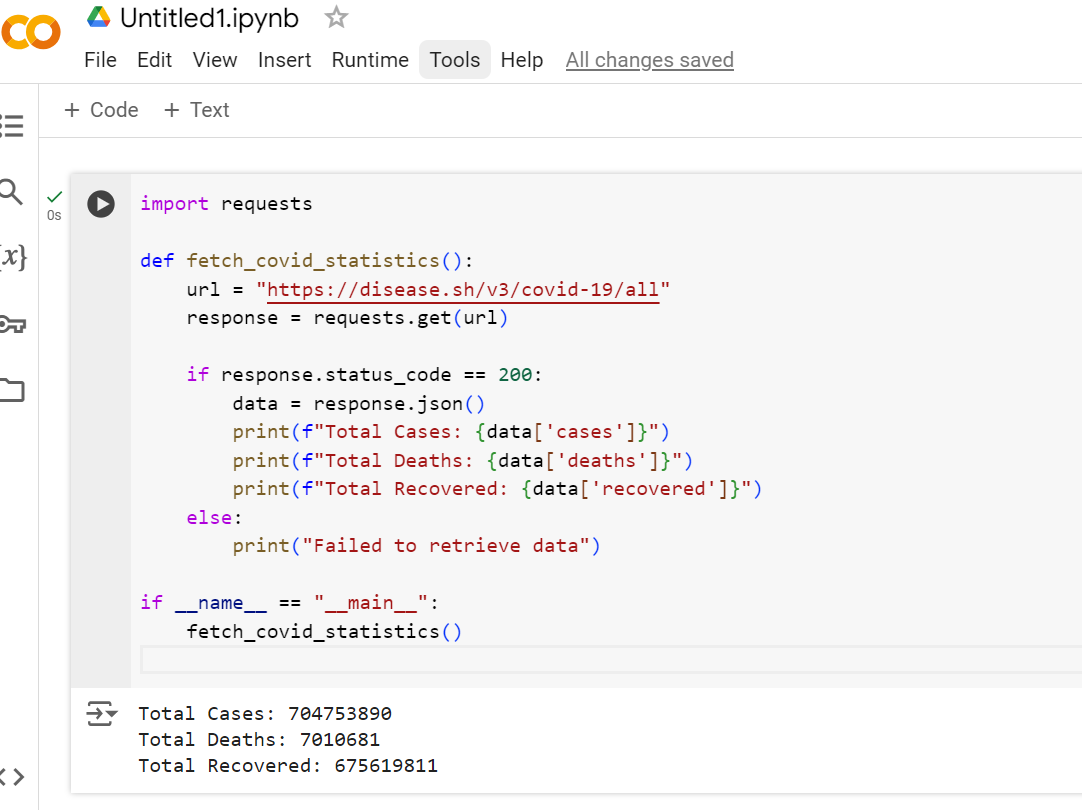
**3.Display covid 19 statistics information:**

Enter city: Kurnool

Total cases: 200

Total deaths:50

Total recover:150

**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.