**ASSIGNMENT: PYTHON PROGRAMMING FOR GUI DEVELOPMENT**

**Name**: T.Rupendra

**Register number:** 192311325

**Department:** Computer Science Engineering

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

**Project Overview: Real-Time COVID-19 statistics tracker**

**Objective:** Develop a Real-Time COVID-19 statistics tracker to provide current and accurate information on COVID-19 cases, recoveries, and fatalities on a global and local scale. The tool should aggregate data from credible sources, offer an intuitive interface, and include features such as historical data analysis, trend visualization, and alert notifications.

**Tasks:**

1. **Data Collection:** Aggregate data from various reliable sources, including health organizations, government websites, and hospitals.
2. **Data Integration:** Harmonize and synchronize data from multiple regions and sources to present a consolidated view.
3. **Continuous Updates:** Ensure the data monitor is regularly updated with the latest information to reflect current statistics and trends.
4. **Data Presentation:** Use charts, graphs, and maps to display data in a clear and accessible manner.
5. **Trend Identification:** Analyze and present trends over time, including spikes and declines in case numbers.

**Deliverables:**

1. **Interactive Interface:** A user-friendly interface showcasing real-time global, national, and local COVID-19 statistics, including cases, deaths, recoveries, and vaccination rates.
2. **Visual Data Representations:** Charts, graphs, and maps to illustrate trends, distributions, and comparisons over time and across different regions.
3. **Trend Analysis Reports:** Detailed summaries and analyses of trends, such as case surges, recovery rates, and hospitalizations.
4. **Alert System:** Notifications and alerts for significant changes or updates in the data, such as new outbreaks or policy adjustments.
5. **Historical Data Archive:** Access to historical data for comparative and trend analysis.

**Flow chart:**

Implementation code:

import requests

def get\_covid\_data():

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("Error retrieving data")

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

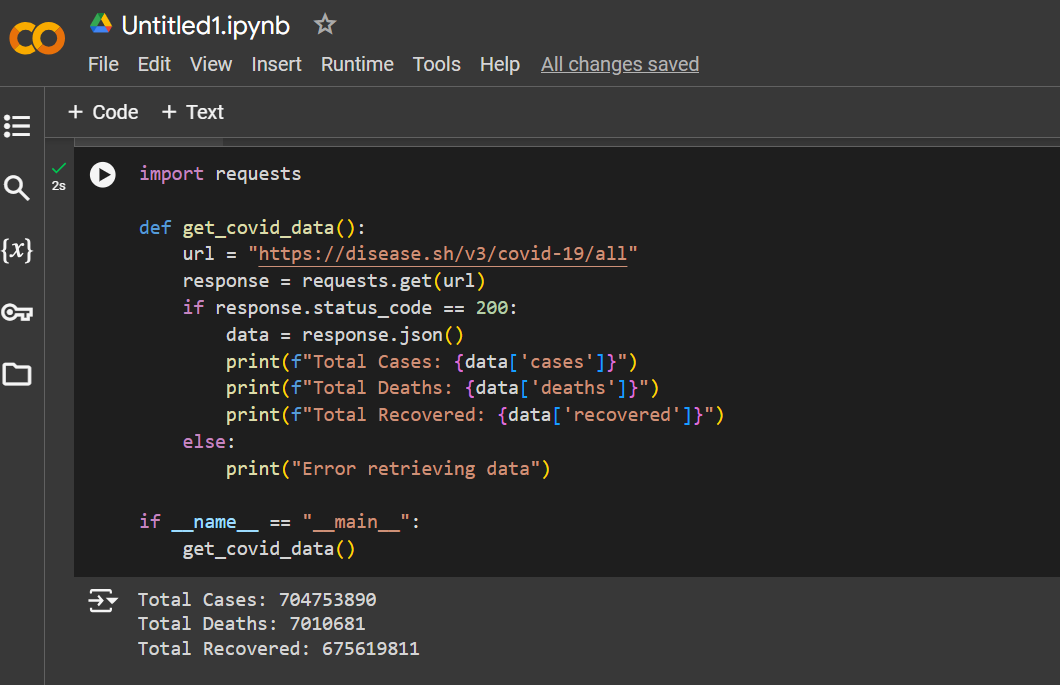
get\_covid\_data()

display output:

total cases:7047533890

total deths:7010681

total recovery:675619811



**User Interaction:**

Interface for users to input location data and view specific statistics.

**Documentation:**

**1. Introduction**

**Overview:** The Real-Time COVID-19 Data Monitor is a web application providing live updates on COVID-19 statistics. It aggregates information from reliable sources to deliver accurate data on confirmed cases, recoveries, and fatalities. The platform features an intuitive interface for effective data visualization and analysis.

**Key Features:**

Real-time data updates.

Global and local statistics.

Data visualization with charts, graphs, and maps.

Historical data and trend analysis.

Alerts for significant data changes.

**Use Cases:**

* + - Public awareness.
    - Health officials' decision-making.
    - Research and academic analysis.

**2. Getting Started**

* + **Prerequisites:** Python 3.7+, Pip, Flask, Requests.
  + **Installation:**

Clone Repository:

bash

Copy code

git clone https://github.com/your-repository/covid-monitor.git

cd covid-monitor

Set Up Virtual Environment (Optional):

bash

Copy code

python -m venv venv

source venv/bin/activate # Windows: `venv\Scripts\activate`

Install Dependencies:

bash

Copy code

pip install -r requirements.txt

* + - Run Application:

bash

Copy code

python covid\_monitor.py

Access at: http://127.0.0.1:5000/.

**3. User Guide**

**Navigating the Interface:**

**Home Page:** Displays global statistics and navigation options.

**Statistics Page:** Provides detailed regional data and trends.

**Data Visualization:** Interactive charts and maps.

**Features:**

Search and filters for specific data points.

Alerts for significant updates.

Historical data access for trend analysis.

**4. Troubleshooting:**

**Common Issues:**

Data not updating: Check internet and API status.

Errors: Refer to error messages and Developer Guide.

**Contact Support:** support@example.com

**5. Developer Guide**

**Project Structure:**

covid\_monitor.py: Main script for Flask application.

requirements.txt: Required packages.

templates/: HTML templates for web pages.

static/: Static files like CSS and JavaScript.

**Code Overview:**

**Data Fetching:** get\_covid\_data() function to retrieve data.

**Backend:** Flask routes for main page and API endpoint.

**Frontend:** HTML templates for dynamic content.

**Extending the Application:**

Add new features and handle additional data sources.

**Testing and Debugging:**

Use unittest or pytest for testing.

Flask debugging tools for troubleshooting.

**Deployment:**

Prepare for production with Gunicorn or uWSGI.

Deploy on cloud platforms or use Docker for consistent environments.