**Assignment: Python Programming For Game Development**

Name: Shreyaa A

Register Number: 192311420

Department: CSE

Date of Submission: 26/08/2024

**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.**

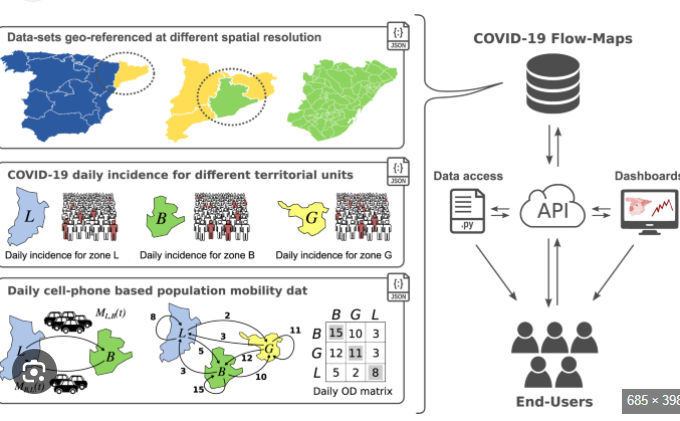
**DELIVERABLES:**

* 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**

**1. DATA FLOW DIAGRAM:**



**2. PSEUDOCODE AND IMPLIMENTATION**

**import requests def fetch\_covid\_stats(region):**

**api\_url = "**[**https://disease.sh/v3/covid-19/all**](https://disease.sh/v3/covid-19/all)**"**

**try: response = requests.get(api\_url)**

**response.raise\_for\_status()**

**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.DOCUMENTATION:**

**1. User Request:**

* **Action:** The user initiates the request to fetch COVID-19 statistics.
* **Example:** The user clicks a button or enters a command on a website/app to get COVID-19 stats.

**2. Frontend/UI:**

* **Action:** The user interface (UI) captures the user’s request.
* **Example:** The frontend (web/mobile) sends the request to the backend.
* **Data:** Request contains relevant parameters such as country name, date range, etc.

**3. Backend/API Call:**

* **Action:** The backend receives the request from the frontend and prepares to make a call to the external COVID-19 statistics API.
* **Example:** The backend creates an HTTP request (e.g., GET) to the external API endpoint.
* **Data:** The request includes headers, query parameters, or body data necessary for the external API.

**4. External COVID-19 API:**

* **Action:** The external API processes the request and returns COVID-19 data (e.g., total cases, recoveries, deaths).
* **Example:** The external API responds with JSON data, including the requested statistics.
* **Data:** JSON response containing data such as **{"confirmed": 100000, "recovered": 98000, "deaths": 2000}**

Modeling a data flow for fetching COVID-19 statistics from an external API and displaying it to the user involves the following steps:

**Data Flow Diagram**

plaintext

Copy code

1. User Request -> 2. Frontend/UI -> 3. Backend/API Call -> 4. External COVID-19 API -> 5. Backend Response Handling -> 6. Frontend Display -> 7. User

**Data Flow Breakdown**

**1. User Request:**

* **Action:** The user initiates the request to fetch COVID-19 statistics.
* **Example:** The user clicks a button or enters a command on a website/app to get COVID-19 stats.

**2. Frontend/UI:**

* **Action:** The user interface (UI) captures the user’s request.
* **Example:** The frontend (web/mobile) sends the request to the backend.
* **Data:** Request contains relevant parameters such as country name, date range, etc.

**3. Backend/API Call:**

* **Action:** The backend receives the request from the frontend and prepares to make a call to the external COVID-19 statistics API.
* **Example:** The backend creates an HTTP request (e.g., GET) to the external API endpoint.
* **Data:** The request includes headers, query parameters, or body data necessary for the external API.

**4. External COVID-19 API:**

* **Action:** The external API processes the request and returns COVID-19 data (e.g., total cases, recoveries, deaths).
* **Example:** The external API responds with JSON data, including the requested statistics.
* **Data:** JSON response containing data such as **{"confirmed": 100000, "recovered": 98000, "deaths": 2000}**.

**5. Backend Response Handling:**

* **Action:** The backend processes the data received from the external API.
* **Example:** The backend may format, sanitize, or filter the response (e.g., handling edge cases, converting units).
* **Data:** Cleaned and formatted data is prepared for frontend use.

**6. Frontend Display:**

* **Action:** The processed data is sent back to the frontend.
* **Example:** The frontend receives the response from the backend and displays it to the user in a readable format (e.g., charts, graphs, or statistics).
* **Data:** Display data, such as **Confirmed: 100,000 | Recovered: 98,000 | Deaths: 2,000**.

**7. User:**

* **Action:** The user views the COVID-19 statistics.
* **Example:** The data is presented in a user-friendly manner (e.g., on a dashboard or mobile screen).
* **Outcome:** The user gains insights into the current COVID-19 situation based on the data.

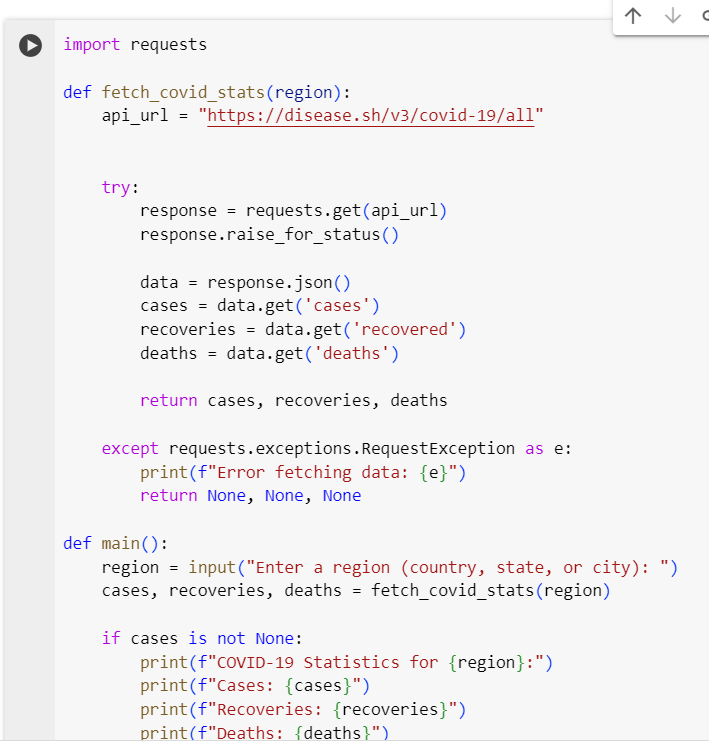
**4. USER INTERFACE:**

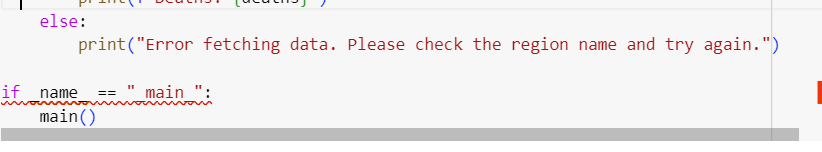
**User interactive program:**

This program allows the user to inter the input values and allows the user to interact freely with the program even though the user would have to give a lot of inputs for the working of the program

**Sample input and output:**

**(from google colab)**

****

****

**5. ASSUMPTIONS AND IMPROVEMENTS:**

ASSUMPTIONS:

When designing and building a Real-Time COVID-19 Statistics Tracker, several assumptions are often made:

1. **Data Accuracy and Reliability**:
   * **Assumption**: The data provided by sources such as government health organizations, hospitals, and international health agencies (e.g., WHO, CDC) is accurate, up-to-date, and consistent across all reporting jurisdictions.
   * **Implication**: Data discrepancies between different regions (e.g., delays in reporting, different data collection methodologies) may occur. The tracker assumes that these data variances are minor and do not significantly affect the overall trends.
2. **Data Availability**:
   * **Assumption**: Data sources will consistently provide real-time data feeds, or at least frequently updated datasets. APIs or other forms of access to data will remain operational and accessible.
   * **Implication**: Outages, inconsistencies in updates, or changes in data access policies could lead to gaps in the tracker’s performance.
3. **Standardization Across Geographies**:
   * **Assumption**: The tracker assumes that all regions define metrics (e.g., "cases," "recoveries," "deaths") in the same way. This includes uniformity in testing protocols and criteria for declaring a recovery or death caused by COVID-19.
   * **Implication**: Variability in definitions could result in skewed or misinterpreted data, especially in global comparisons.
4. **Model Accuracy for Projections (if applicable)**:
   * **Assumption**: If the tracker provides projections or future trends (e.g., predicting case increases), it assumes that the underlying model reflects current conditions such as transmission rates, vaccination rates, and governmental policies (e.g., lockdowns, mask mandates).
   * **Implication**: Changes in public behavior, policy interventions, or emerging variants could disrupt the accuracy of these projections.
5. **User Behavior**:
   * **Assumption**: The tracker assumes that users will interpret the data responsibly and will not make decisions solely based on displayed statistics, but will consult with health authorities or experts.
   * **Implication**: Misinterpretation of data or reliance on incomplete information could lead to public misjudgment regarding the severity of the pandemic.
6. **Privacy and Data Ethics**:
   * **Assumption**: Any personal data (e.g., case location data, patient demographics) used by the tracker will be anonymized and in compliance with data protection regulations such as GDPR or HIPAA.
   * **Implication**: Data breaches or failure to anonymize information properly could lead to privacy violations or misuse of sensitive data.

**IMPROVEMENTS**

1. **Improved Data Standardization**:
   * Implement stricter protocols for standardizing data from different regions. Introduce a uniform reporting framework that all regions must adhere to when contributing to the tracker.
   * **Improvement**: This would reduce variability in definitions and make global comparisons more accurate.
2. **Enhanced Data Sources**:
   * Incorporate more diverse data sources such as contact tracing apps, social media reports (where valid), wastewater analysis, and mobility data to get a more holistic view of the virus’s spread.
   * **Improvement**: This would increase data granularity, particularly in regions with less robust public health reporting.
3. **Predictive Modeling Enhancements**:
   * Use machine learning to enhance predictive models, making them more responsive to changing variables like new variants, vaccination rates, and behavioral changes. Add layers of scenario-based projections to account for different possible outcomes.
   * **Improvement**: Projections would be more adaptive to real-world developments, increasing their reliability.
4. **Better Visualization Tools**:
   * Offer improved data visualizations, such as heat maps, trend graphs over time, and user-customizable dashboards to explore specific data points (e.g., age demographics, regional trends).
   * **Improvement**: Improved UI/UX can enhance user understanding and accessibility of complex data, making it easier for users to identify trends and risks.
5. **Increased Data Transparency**:
   * Publish more detailed information on how data is collected and processed. For example, clarify any time lags, data quality issues, and methods used for projections.
   * **Improvement**: This would build public trust in the data and clarify how reliable the tracker is at any given time.