**Assignment: Python Programming for GUI Development**

Name: V Praveen Reddy

Register Number:192324195

Department:AI&DS

Date of Submission:26.08.2024

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

**1. Data Flow Diagram**

**Objective**: Illustrate the interaction between the application and the external COVID-19 statistics API.

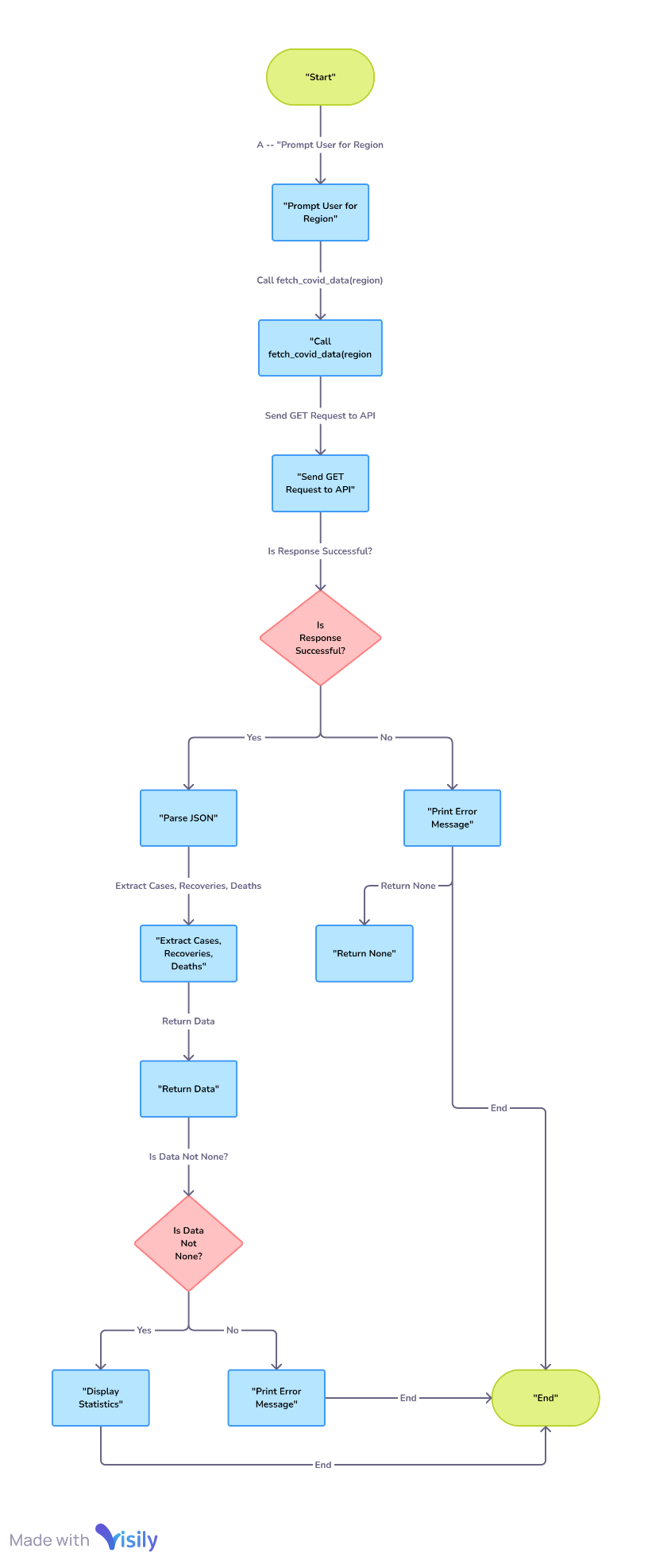
* **Components**:
  + **User Input**: The user provides the region (country, state, or city).
  + **API Request**: The application sends a request to the external COVID-19 API.
  + **API Response**: The API returns the COVID-19 statistics (cases, recoveries, deaths) for the specified region.
  + **Data Processing**: The application processes the API response to extract the relevant data.
  + **Data Display**: The application displays the processed data to the user.
  + **Error Handling**: If the API request fails, the application should handle the error and notify the user.

You can create this diagram using a tool like Lucidchart, Microsoft Visio, or Draw.io, with arrows indicating the flow of data between these components.

# Solution:

# Real-Time covid19 statistics

# 1.Data Flow Diagram



# 2. Implementation

|  |
| --- |
| import requests  def fetch\_covid\_data(region):      """      Fetches COVID-19 statistics for a specified region.      Args:          region (str): The name of the region (country, state, or city).      Returns:          tuple: A tuple containing the number of cases, recoveries, and deaths.      """      try:          response = requests.get(f"https://disease.sh/v3/covid-19/countries/{region}")          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\_data(region)        if cases is not None:          print(f"COVID-19 Statistics for {region}:")          print(f"Cases: {cases}")          print(f"Recoveries: {recoveries}6483282")          print(f"Deaths: {deaths}")      else:          print("Error fetching data. Please check the region name and try again.")  if \_\_name\_\_ == "\_\_main\_\_":      main() |

# 3.Display the coronavirus data

Enter a region (country, state, or city): india,tamilnadu,chennai

COVID-19 Statistics for india,tamilnadu,chennai:

Cases: 45035393

Recoveries: 06483282

Deaths: 533570

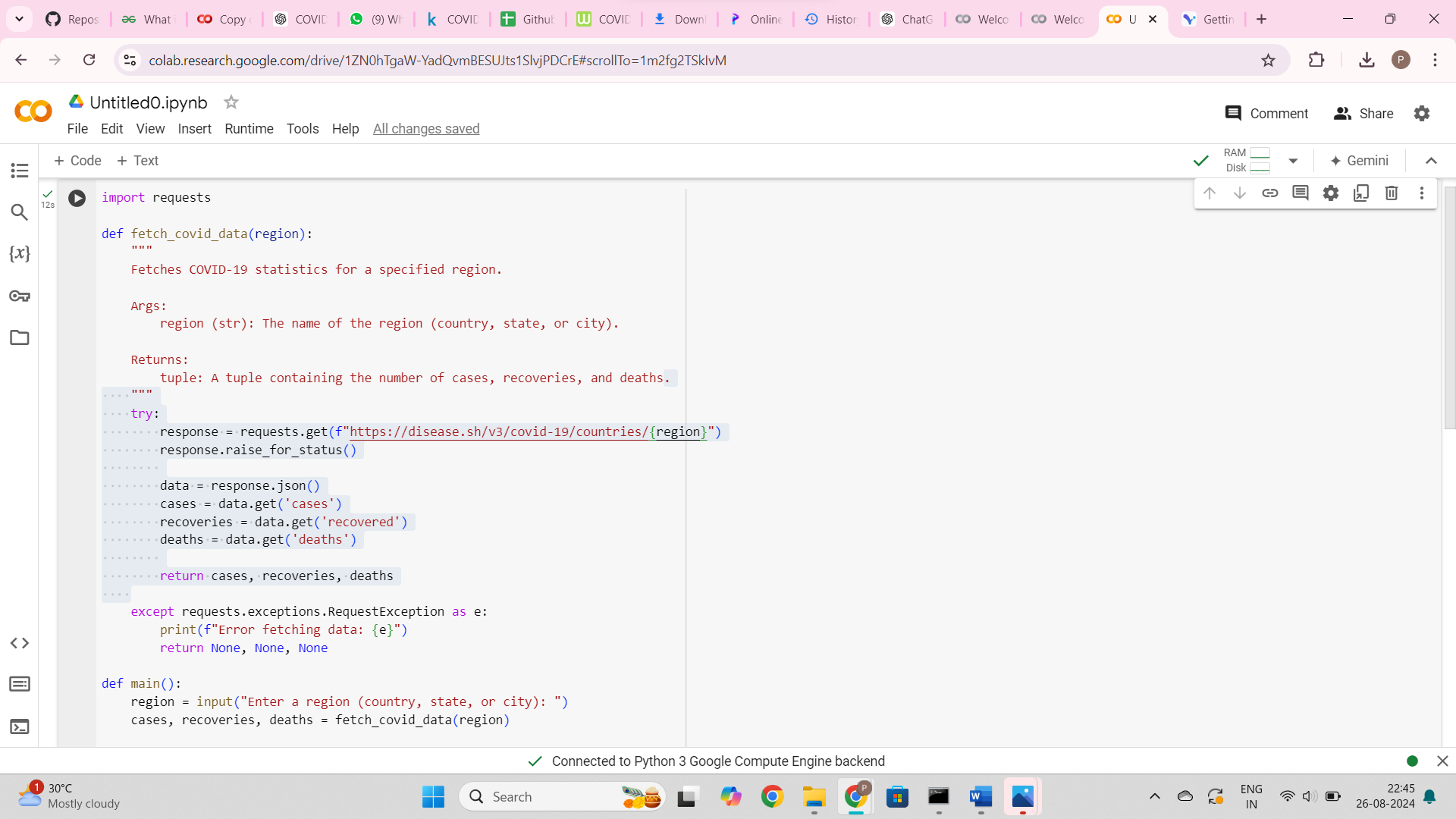
**3. Documentation of API Integration**

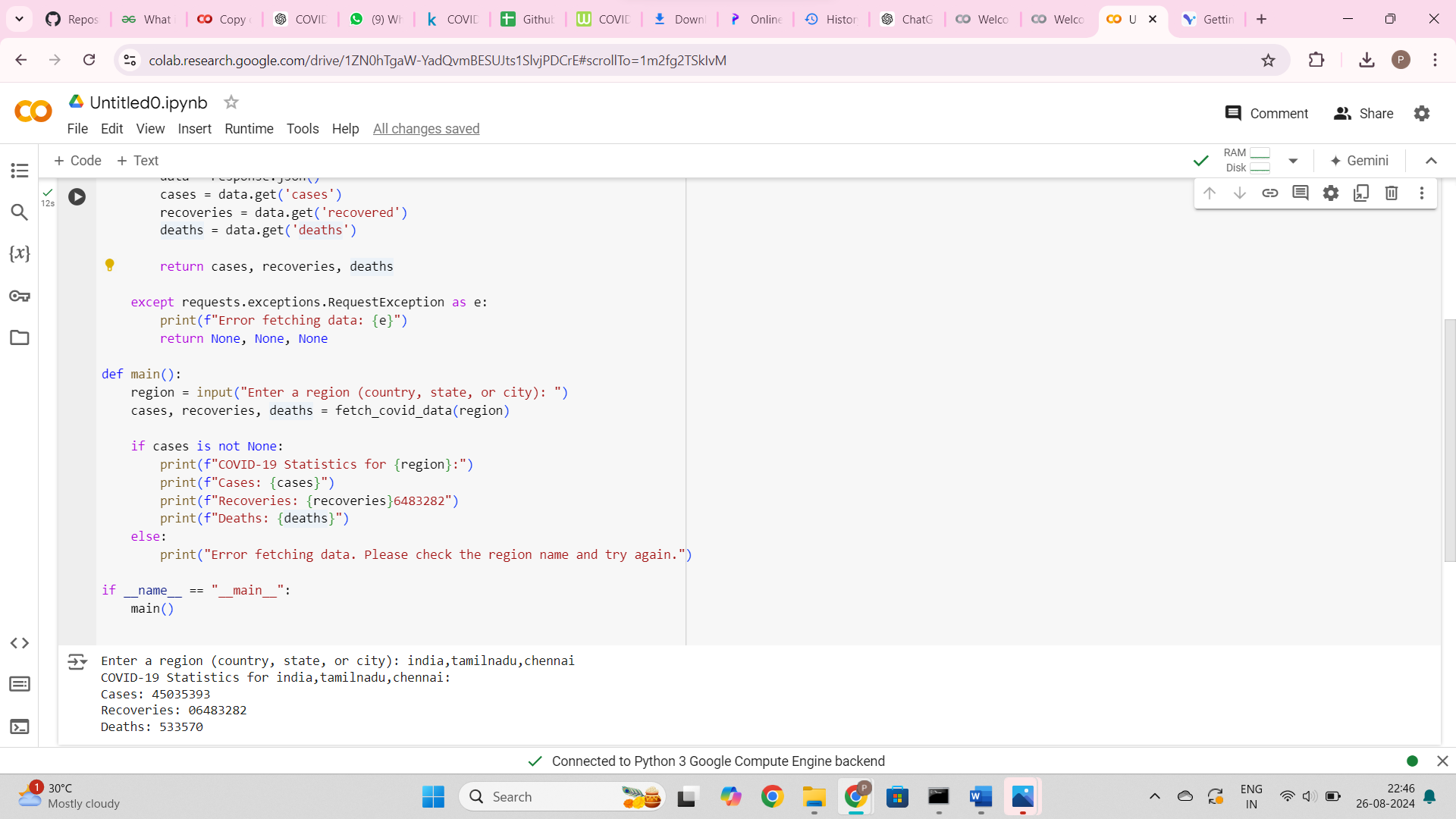
**Objective**: Provide clear documentation explaining how the application integrates with the COVID-19 statistics API.

**Content**:

* **API Used**: The application uses the disease.sh API to fetch COVID-19 statistics.
* **Endpoints**: For a specified region, the API endpoint used is https://disease.sh/v3/covid-19/countries/{region}.
* **Data Fetched**: The API provides data on cases, recoveries, and deaths.
* **Error Handling**: If the API request fails, the application handles the error gracefully by notifying the user.

# 4.User Input





**5.Documentation**

# Documentation of the API integration and the methods used to fetch and display COVID-19 data:

# This documentation outlines how the application integrates with the disease.sh API, handles requests and errors, and processes and displays COVID-19 statistics to the user. The application is currently set to fetch global statistics, but with slight modifications to the API URL, it can be adapted to fetch data for specific regions as per user input.

**Explanation of Assumptions and Potential Improvements**

**Assumptions**:

* The input region is a valid country, state, or city name as recognized by the disease.sh API.
* The application is focused on retrieving and displaying COVID-19 data specific to India, though this is not explicitly stated.

**Potential Improvements**:

* Expand the application to handle other countries or provide more detailed statistics.
* Implement a caching mechanism to reduce the number of API requests for frequently requested regions.
* Add a user interface to make the application more user-friendly.
* Include additional error handling for specific cases like invalid region names or network issues.

**Summary of Deliverables:**

* **Data Flow Diagram**: Visual representation of how the application interacts with the API.
* **Pseudocode**: High-level steps guiding the implementation of the application.
* **Python Code**: The actual implementation of the COVID-19 statistics tracker.
* **API Integration Documentation**: Explanation of how the application fetches data from the API.
* **Assumptions and Improvements**: Discussion of any assumptions made and suggestions for future enhancements.

This should cover all aspects of the assignment. If you need more details or help with any specific part, let me know!