Assignment:03

Python Programming for GUI Development

Name:R.Sandhya Rani
Register number:192372378
Department:CSE-AI
Date of Submission:26/08/2024

Problem 3: Real-Time Traffic Monitoring System

Scenario:

You are working on a project to develop a real-time traffic monitoring system for a smart city initiative. The system should provide real-time traffic updates and suggest alternative routes.

Tasks:

1. Model the data flow for fetching real-time traffic information from an external API and displaying it to the user.
2. Implement a Python application that integrates with a traffic monitoring API (e.g., Google Maps Traffic API) to fetch real-time traffic data.
3. Display current traffic conditions, estimated travel time, and any incidents or delays.
4. Allow users to input a starting point and destination to receive traffic updates and alternative routes,

Deliverables:

- Data flow diagram illustrating the interaction between the application and the API. Pseudocode and implementation of the traffic monitoring system.
- Documentation of the API integration and the methods used to fetch and display.
- traffic data. Explanation of any assumptions made and potential improvements.

Solution:

Real-Time Traffic Monitoring System

1.Data Flow Diagram:

To model the data flow for fetching real-time traffic information, we'll use a high-level data flow diagram (DFD) that includes the main components and their interactions. The primary components are the user, the application, and the external traffic monitoring API.

1. *User*: Interacts with the application to input starting point, destination, and receive traffic updates.

2. *Application*:

   - *User Input Module*: Captures user inputs (starting point and destination).

   - *API Integration Module*: Fetches real-time traffic data from the external API.

   - *Data Processing Module*: Processes and interprets traffic data.

   - *Display Module*: Shows current traffic conditions, estimated travel time, and incidents to the user.

3. *External Traffic Monitoring API*:

   - Provides real-time traffic data including traffic conditions, estimated travel time, incidents, and alternative routes.

```
USER ←——→APPLICATION MODULE ←——-|
                    |                          |
                    |                          \/
                    \/            |EXTERNAL TRAFFIC API|

              |DISPLAY MODULE|
```

2.Implementation

```
import requests


API_KEY = "your_api_key_here"

API_ENDPOINT = "https://maps.googleapis.com/maps/api/directions/json"


def fetch_traffic_data(origin, destination):

    url =
f"{API_ENDPOINT}?origin={origin}&destination={destination}&key={API_KEY}"

    response = requests.get(url)


    if response.status_code == 200:

        traffic_data = response.json()

        return traffic_data

    else:

        return None


def display_traffic_info(traffic_data):

    if traffic_data is not None:

        routes = traffic_data.get("routes", [])


        if routes:

            legs = routes[0].get("legs", [])


            if legs:
```

```
            duration_in_traffic = legs[0].get("duration_in_traffic", {}).get("text", "Not
available")

            print(f"Estimated duration in traffic: {duration_in_traffic}")


            current_speed = legs[0].get("traffic_speed_entry", [{}])[0].get("speed", "Not
available")

            print(f"Current speed: {current_speed} km/h")


            incidents = legs[0].get("traffic", {}).get("incidents", [])

            if incidents:

                print("Incidents:")

                for incident in incidents:

                    incident_type = incident.get("type", "Unknown")

                    incident_description = incident.get("description", "No description")

                    print(f"- {incident_type}: {incident_description}")

            else:

                print("No incidents reported.")

        else:

            print("No legs found in the route.")

    else:

        print("No routes found.")

else:

    print("Failed to fetch traffic data.")


def suggest_alternative_routes(traffic_data):

    if traffic_data is not None:
```

```
        routes = traffic_data.get("routes", [])

        if len(routes) > 1:
            print("Alternative routes:")
            for i in range(1, len(routes)):
                route_summary = routes[i].get("summary", "Route without summary")
                route_duration = routes[i].get("legs", [{}])[0].get("duration", {}).get("text", "Not
available")
                print(f"- Route {i}: {route_summary}, Estimated duration: {route_duration}")
        else:
            print("No alternative routes available.")
    else:
        print("Failed to fetch alternative routes.")

if __name__ == "__main__":
    origin = input("Enter starting point: ")
    destination = input("Enter destination: ")

    traffic_data = fetch_traffic_data(origin, destination)

    if traffic_data is not None:
        display_traffic_info(traffic_data)
        suggest_alternative_routes(traffic_data)
    else:
        print("Failed to fetch traffic information. Please try again.")
```

OUTPUT:

```
+ Code  + Text  Copy to Drive                                  RAM  Disk  ✦ Gemini
                route_summary = routes[i].get("summary", "Route without summary")
                route_duration = routes[i].get("legs", [{}])[0].get("duration", {}).get("text", "Not a
                print(f"- Route {i}: {route_summary}, Estimated duration: {route_duration}")
        else:
            print("No alternative routes available.")
    else:
        print("Failed to fetch alternative routes.")

if __name__ == "__main__":
    origin = input("Enter starting point: ")
    destination = input("Enter destination: ")

    traffic_data = fetch_traffic_data(origin, destination)

    if traffic_data is not None:
        display_traffic_info(traffic_data)
        suggest_alternative_routes(traffic_data)
    else:
        print("Failed to fetch traffic information. Please try again.")

Enter starting point: chennai
Enter destination: chittor
No routes found.
No alternative routes available.
```

DOCUMENTATION:

API Integration:

1. *API Key*: Replace 'YOUR_GOOGLE_MAPS_API_KEY' with your actual API key from Google Cloud.

2. *Fetching Data*: The fetch_traffic_data function makes a request to the Google Maps Directions API with parameters including the origin, destination, and departure time.

3. *Processing Data*: The process_traffic_data function extracts relevant information from the API response, such as estimated duration and distance.

4. *Displaying Data*: The display_traffic_info function outputs the information to the user.

Assumptions:

- The API key is valid and has appropriate permissions.

- The API returns data in a format that includes 'routes', 'legs', and 'duration_in_traffic'.

- Incidents data is not explicitly provided by the API but can be included if the API or additional services are used.

Potential Improvements:

- *Error Handling*: Add error handling for network issues or invalid responses.

- *User Interface*: Develop a graphical user interface (GUI) for better user interaction.

- *Additional Features*: Integrate additional APIs or services for real-time incident reporting and alternative routes.

- *Performance*: Optimize the application to handle large volumes of requests or provide faster responses.

This implementation provides a basic framework for real-time traffic monitoring and can be expanded with more sophisticated features and error handling based on project requirements.