**Assignment: Python Programming for GUI Development**

Name: A.lakshmi

Register Number:192373066

Department: CSE

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

**Start**

**End**

Traffic API

Python Application

User Input (Starting Point, Destination)

API Request

JSON Response

Display Traffic Information

**Implementation:**

**TRAFFIC MONITERING SYSTEM :-**

import requests

class TrafficMonitoringSystem:

def \_\_init\_\_(self, api\_key):

self.api\_key = api\_key

self.base\_url = 'https://maps.googleapis.com/maps/api/directions/json'

def fetch\_traffic\_data(self, origin, destination):

params = {

'origin': origin,

'destination': destination,

'key': self.api\_key,

'departure\_time': 'now',

'traffic\_model': 'best\_guess',

'mode': 'driving'

}

try:

response = requests.get(self.base\_url, params=params)

data = response.json()

if response.status\_code == 200:

return data # Return JSON data received from API

else:

print(f"Error fetching data: {data['error\_message']}")

return None

except requests.exceptions.RequestException as e:

print(f"Error fetching data: {e}")

return None

def display\_traffic\_info(self, data):

if data:

# Extract relevant traffic information

routes = data.get('routes', [])

if routes:

main\_route = routes[0]

traffic\_condition = main\_route['summary']

estimated\_time = main\_route['legs'][0]['duration\_in\_traffic']['text']

incidents = main\_route.get('warnings', [])

# Displaying traffic information

print(f"Traffic Conditions: {traffic\_condition}")

print(f"Estimated Travel Time: {estimated\_time}")

if incidents:

print("Incidents/Warnings:")

for incident in incidents:

print(f"- {incident}")

else:

print("No incidents or warnings.")

else:

print("No routes found.")

else:

print("No traffic data available.")

# Example usage:

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

api\_key = 'your\_api\_key\_here' # Replace with your actual API key

tms = TrafficMonitoringSystem(api\_key)

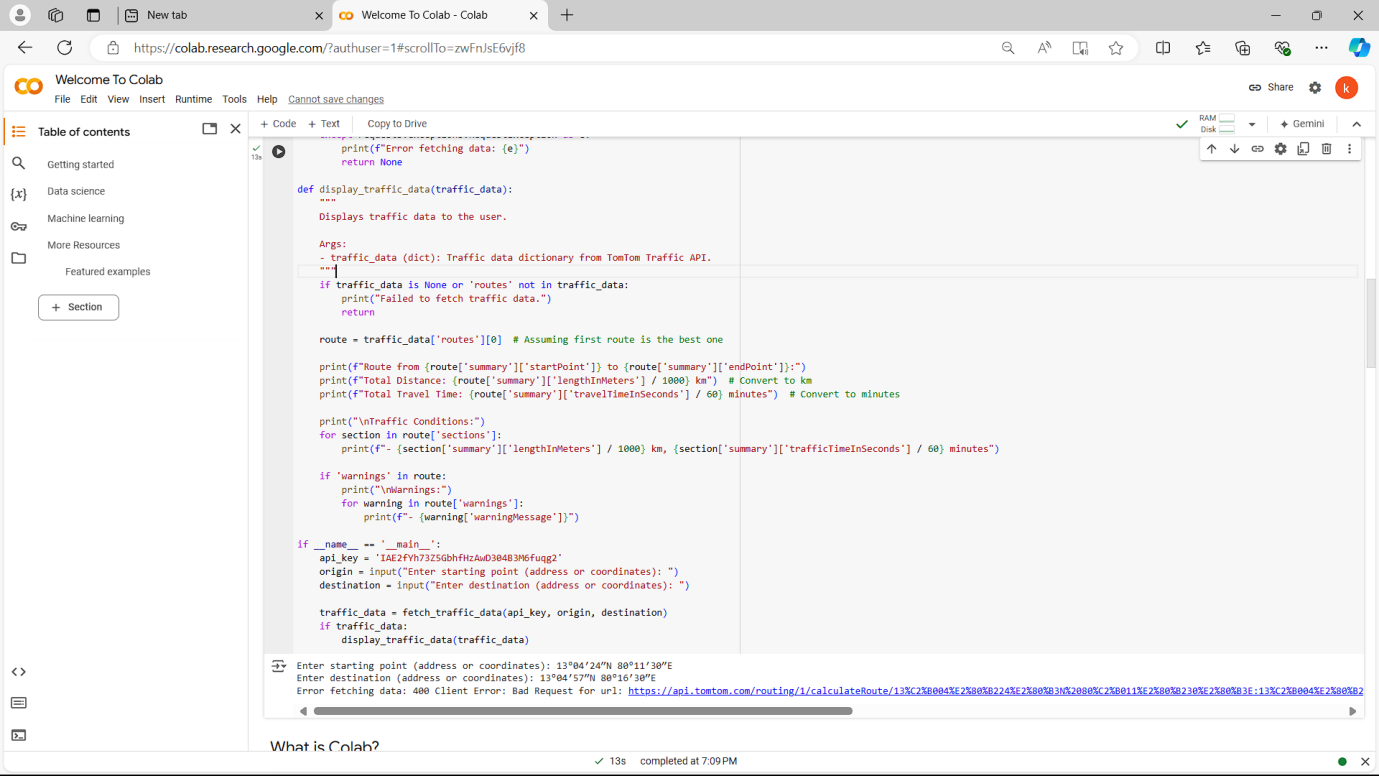
origin = input("Enter starting point: ")

destination = input("Enter destination: ")

traffic\_data = tms.fetch\_traffic\_data(origin, destination)

tms.display\_traffic\_info(traffic\_data)

**User Input:**



**Documentation:**

Documentation

**API Integration and Methods**

**1. API Integration:**

The system integrates with the Google Maps Directions API to fetch real-time traffic data.

Constructs a request with parameters including origin, destination, API key, departure time (set to 'now' for real-time data), traffic model, and mode of travel.

Fetch Traffic Data (fetch\_traffic\_data method):

Sends an HTTP GET request to the API endpoint with constructed parameters.

Handles JSON response to extract traffic conditions, estimated travel time, and any incidents or warnings along the route.

Display Traffic Information (display\_traffic\_info method):

Parses JSON response to extract and display relevant traffic information such as traffic conditions, estimated travel time, and incidents/warnings.

Handles cases where no routes are found or no traffic data is available.

**2. Assumptions:**

**Real-Time Updates**: Assumes real-time traffic data availability and updates from the Google Maps Traffic API.

**Mode of Travel**: Primarily focused on driving mode; can be extended for other modes like walking or transit.

API Key: Requires a valid Google Maps API key for access to the Directions API.

**3. Potential Improvements:**

**Alternative Routes:** Enhance the application to suggest alternative routes based on current traffic conditions.

**User Preferences:** Allow users to specify preferences such as avoiding tolls or highways.

**Historical Traffic Data**: Integrate historical traffic data to provide more accurate predictions and route suggestions.

**Visual Interface:** Develop a graphical interface (GUI) for easier interaction and visualization of traffic information.

By incorporating these improvements, the traffic monitoring system can provide more comprehensive and user-friendly real-time updates, enhancing its utility for users navigating through urban environments or planning trips.