**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.

**Data Flow Diagram:**

+-----------------+ +-----------------------+ +------------------------------+

| User Interface | <---> | Python Application | <---> | Traffic API (e.g., Google |

| (User Input) | | (Backend) | | Maps Traffic API) |

+-----------------+ +-------------------------+ +-------------------------------+

|

V

+-----------------------------+

| Traffic Data Processing |

| and Display |

+----------------------------+

**Pseudocode:**

text

import requests

def get\_traffic\_data(start, end):

# Call the traffic API to fetch real-time data

api\_key = "your\_api\_key" url=f"https://maps.googleapis.com/maps/api/directions/json?origin={start}&destination={end}&key={api\_key}"

response = requests.get(url)

data = response.json()

# Extract relevant traffic information

current\_traffic = data["routes"][0]["legs"][0]["duration\_in\_traffic"]["text"]

estimated\_travel\_time = data["routes"][0]["legs"][0]["duration"]["text"]

alternative\_routes = []

for route in data["routes"]:

alt\_route = {

"distance": route["legs"][0]["distance"]["text"],

"duration": route["legs"][0]["duration"]["text"],

"duration\_in\_traffic": route["legs"][0]["duration\_in\_traffic"]["text"] }

alternative\_routes.append(alt\_route)

return current\_traffic, estimated\_travel\_time, alternative\_routes

def display\_traffic\_info(current\_traffic, estimated\_travel\_time, alternative\_routes):

print(f"Current traffic conditions: {current\_traffic}")

print(f"Estimated travel time: {estimated\_travel\_time}")

print("Alternative routes:")

for route in alternative\_routes:

print(f"- Distance: {route['distance']}, Duration: {route['duration']}, Duration in traffic: {route['duration\_in\_traffic']}")

def main():

start = input("Enter your starting point: ")

end = input("Enter your destination: ")

current\_traffic, estimated\_travel\_time, alternative\_routes = get\_traffic\_data(start, end)

display\_traffic\_info(current\_traffic, estimated\_travel\_time, alternative\_routes)

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

**main()**

**Reports and Alerts**

• **Traffic Updates:** Real-time updates on traffic conditions including traffic speed,

delays, and incidents.

• **Alternative Routes:** Suggestions for alternative routes based on current traffic

conditions to optimize travel time.

• **User Alerts:** Alerts users about significant traffic congestion or incidents affecting

their chosen route.

**Assumptions and Limitations**

**Assumptions:**

• Assume consistent and reliable data availability from TomTom's Traffic API.

• Users have access to a stable internet connection for real-time data retrieval.

**Limitations:**

• Dependency on third-party API (TomTom's Traffic API) for data accuracy and

availability.

• Potential discrepancies in real-time traffic updates due to varying data refresh rates

and traffic conditions.

**Code:**

import requests

import urllib.parse

def fetch\_traffic\_data(api\_key, origin\_lat, origin\_lon, dest\_lat, dest\_lon):

base\_url = "https://api.tomtom.com/traffic/services/4/route/traffic/json"

params = {

"key": api\_key,

"departureTime": "now",

"traffic": "true",

"travelMode": "car",

"avoid": "unpavedRoads",

"language": "en",

"computeBestOrder": "true",

"sectionType": "traffic",

"routeType": "fastest",

"origin": f"{origin\_lat},{origin\_lon}",

"destination": f"{dest\_lat},{dest\_lon}"

}

try:

encoded\_params = urllib.parse.urlencode(params, quote\_via=urllib.parse.quote)

url = f"{base\_url}?{encoded\_params}"

response = requests.get(url)

response.raise\_for\_status()

data = response.json()

if data:

return data

else:

print(f"Empty response or invalid JSON: {response.text}")

return None

except requests.exceptions.RequestException as e:

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

return None

def display\_traffic\_data(traffic\_data):

if traffic\_data:

route = traffic\_data.get('routes', [])[0]

if route:

print(f"Route from {route['sections'][0]['start']['name']} to {route['sections'][-1]['end']['name']}:")

print(f"Estimated Total Travel Time: {route['summary']['travelTimeInSeconds'] // 60} minutes")

for section in route['sections']:

print("\nSection Details:")

print(f"Road Name: {section['roadName']}")

if 'currentTraffic' in section:

print(f"Current Traffic Speed: {section['currentTraffic']['speedKMH']} km/h")

print(f"Free Flow Speed: {section['currentTraffic']['freeFlowSpeedKMH']} km/h")

print(f"Traffic Delay: {section['currentTraffic']['delayInSeconds']} seconds")

if section['currentTraffic']['delayInSeconds'] > 0:

print("Traffic congestion detected.")

else:

print("No traffic congestion detected.")

else:

print("No traffic data available for this section.")

print()

else:

print("No route data available.")

else:

print("No data to display.")

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

API\_KEY = 'uUYBGWyrApuCmV0fKZzGoc46MtOcON2J'

origin\_lat = input("Enter origin latitude (e.g., 40.7577): ")

origin\_lon = input("Enter origin longitude (e.g., -73.9857): ")

dest\_lat = input("Enter destination latitude (e.g., 40.7850): ")

dest\_lon = input("Enter destination longitude (e.g., -73.9682): ")

traffic\_data = fetch\_traffic\_data(API\_KEY, origin\_lat, origin\_lon, dest\_lat, dest\_lon)

display\_traffic\_data(traffic\_data)