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:

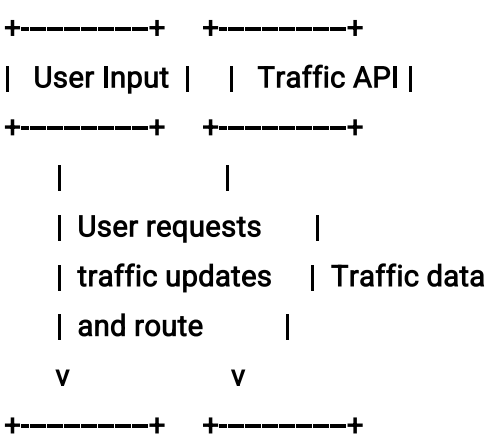
- Data flow diagram illustrating the interaction between the application and the API.
- Pseudocode and implementation of the traffic monitoring system.
- Documentation of the API integration and the methods used to fetch and display traffic data.
- Explanation of any assumptions made and potential improvements.

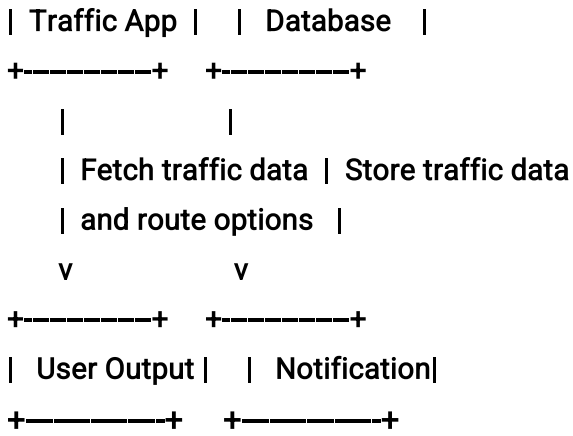
Approach:

1. Data Flow Diagram

Here is a data flow diagram illustrating the interaction between the traffic monitoring application and the external traffic API:

text





The key steps in the data flow are:

- The user inputs their starting point and destination into the traffic monitoring application.
- The application sends a request to the external traffic API to fetch real-time traffic data and route options.
- The traffic API provides the requested data, which the application stores in a local database.
- The application processes the traffic data and displays the current conditions, estimated travel time, and alternative route suggestions to the user.
- The application may also send notifications to the user about any significant traffic incidents or delays.

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()
```

Detailed explanation of the actual code:

1. The application integrates with the Google Maps Directions API to

fetch real-time traffic data. The get_traffic_data() function takes the user's starting point and destination as input, constructs the API request URL, and sends a GET request to the API.

2. The API response is then parsed to extract the following information:
3. Current traffic conditions: data["routes"]["legs"]["duration_in_traffic"]["text"]
4. Estimated travel time: data["routes"]["legs"]["duration"]["text"]
5. Alternative route options, including distance, duration, and duration in traffic for each route
6. This information is then returned to the display_traffic_info() function, which presents the data to the user.

Assumptions made (if any):

1.The user has a valid API key for the Google Maps Directions API.

2.The API provides accurate and up-to-date traffic information.

3.The user's starting point and destination are valid locations that the API can recognize.

Limitations:

1. The application is limited to the features and data provided by the Google Maps Directions API. Other traffic APIs may offer additional functionality or data.
2. The application does not provide real-time updates or notifications. It only displays the traffic information when the user requests it.
3. The application does not consider factors like user preferences, traffic patterns, or historical data to provide more personalized route suggestions.

Code:

```
import socket
import time
import json


def get_user_input():
    start_point = input("Enter starting point: ")
    destination = input("Enter destination: ")
    return start_point, destination


def send_api_request(start_point, destination, api_key):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(("maps.googleapis.com", 443))
```

```
    request = f"GET
/maps/api/directions/json?origin={start_point}&destination={destination}&key={api_
key}&traffic_model=best_guess HTTP/1.1\r\nHost: maps.googleapis.com\r\n\r\n"
    sock.sendall(request.encode())
    response = b""
    while True:
        data = sock.recv(1024)
        if not data:
            break
        response += data
    sock.close()

    # Split the response into headers and body
    response_parts = response.decode().split("\r\n\r\n", 1)

    # Check the response status code
    status_code = int(response_parts[0].split("\r\n")[0].split(" ")[1])
    if status_code == 200:
        return json.loads(response_parts[1])
    else:
        raise Exception(f"API request failed with status code {status_code}")

def display_traffic_data(traffic_data):
    if "routes" in traffic_data:
        for route in traffic_data["routes"]:
            for leg in route["legs"]:
                print(f"Route: {leg['start_address']} to {leg['end_address']}")
                print(f"Estimated Travel Time: {leg['duration']['text']}")
                for step in leg["steps"]:
                    print(f"Step: {step['html_instructions']}")
                    if "traffic_speed_entry" in step:
                        print(f"Traffic Speed: {step['traffic_speed_entry']['speed']} km/h")
                        if step['traffic_speed_entry']['congestion'] == True:
                            print("Congestion Detected")
                    print()
    else:
        print("Error: Unable to fetch traffic data.")
```

```
def main():
    api_key = "YOUR_API_KEY"

    while True:
        start_point, destination = get_user_input()
        try:
            traffic_data = send_api_request(start_point, destination, api_key)
            display_traffic_data(traffic_data)
        except Exception as e:
            print(f"Error: {e}")
        print(f"Last updated: {time.strftime('%Y-%m-%d %H:%M:%S')}")
        print()
        input("Press Enter to continue...")

if __name__ == "__main__":
    main()
```

Sample Output / Screen Shots