PHASE 3 : PUBLIC TRANSPORT OPTIMIZATION

PLANNING AND DESIGN :

- Define the project scope: Determine the objectives and requirements of the IoT-enabled public transportation optimization system.

- Identify the IoT sensors required: Decide on the specific sensors (e.g., GPS, passenger counters) to be deployed in public transportation vehicles.

- Design the data model: Plan the structure of the data to be collected, including vehicle ID, route ID, GPS coordinates, passenger count, and timestamp.

- Choose the transit information platform: Select or design a platform to receive and process the real-time data from IoT sensors.

PROTOTYPING :

- Develop a Python script: Create a prototype script to simulate the behavior of an IoT sensor on a public transportation vehicle.

- Simulate GPS and passenger count data: Generate random GPS coordinates and passenger counts to represent real-world data.

- Set up the API endpoint: Define the URL of the transit information platform's API endpoint where the data will be sent.

- Implement error handling: Consider error scenarios and exception handling for network communication.

DEVELOPMENT :

- Integration with real IoT hardware: Replace the simulated data generation with actual data from IoT sensors installed in public transportation vehicles.
- Secure data transmission: Implement secure communication protocols (e.g., HTTPS) to ensure data is sent safely to the platform.
- Optimize the script: Enhance the code for efficiency, scalability, and reliability in a production environment.
- Test and validate: Perform rigorous testing to ensure the script works as expected with real data.

DEPLOYMENT :

- Deploy IoT sensors: Install the IoT sensors (e.g., GPS and passenger counters) in public transportation vehicles across the fleet.
- Deploy the Python script: Distribute and run the Python script on IoT devices within the vehicles.
- Monitor and maintain: Set up monitoring and maintenance procedures to ensure the continued operation of IoT sensors and the data transmission script.
- Scale and expand: Plan for scalability as the transportation system grows, and consider adding more features to the

transit information platform, such as data analysis and visualization.

SOURCE CODE :

```
import random

import time

import requests


# Simulated vehicle information

vehicle_id = "Bus123"

route_id = "RouteA"


# Transit information platform endpoint

platform_url = "https://your-transit-platform-url.com/api/data"


while True:

    # Simulate GPS coordinates (latitude and longitude)

    latitude = random.uniform(37.0, 38.0)

    longitude = random.uniform(-122.0, -123.0)


    # Simulate passenger count

    passenger_count = random.randint(0, 50)


    # Create a JSON payload with vehicle data

    data = {
        "vehicle_id": vehicle_id,
        "route_id": route_id,
        "latitude": latitude,
        "longitude": longitude,
        "passenger_count": passenger_count,
        "timestamp": int(time.time())
    }

    try:
        # Send the data to the transit information platform
        response = requests.post(platform_url, json=data)

        if response.status_code == 200:
            print(f"Data sent successfully: {data}")
        else:
            print(f"Failed to send data. Status Code: {response.status_code}")
    except requests.exceptions.RequestException as e:
        print(f"Error sending data: {e}")

    # Sleep for a specified interval (e.g., 30 seconds)
    time.sleep(30)
```

This script serves as a basic starting point for sending data to a transit information platform.