Phase 3 project:

Project Title: PUBLIC TRANSPORT OPTIMIZATION

Project ID: proj\_223732\_Team\_2

College: Gnanamani College of Technology

Branch: B.Tech/Information Techology

Year: IIIrd year

College code:6208

User

Deploy IoT sensors (e.g., GPS, passenger counters) in public transportation vehicles to gather data.

Develop a Python script on the IoT sensors to send real-time location and ridership data to the transit information platform.

Hardware Setup:

Choose the IoT sensors (e.g., GPS, passenger counters) that suit your needs.

Install the sensors in the public transportation vehicles.

Connectivity:

Ensure the sensors have access to the internet, either through cellular networks, Wi-Fi, or other means.

Python Script:

Write a Python script that runs on the IoT sensors. You may use a platform like Raspberry Pi for this purpose.

Use appropriate libraries (e.g., gpsd for GPS data) to collect real-time data from the sensors.

Create functions to capture and process data from the passenger counters.

Data Transmission:

Establish a connection to the transit information platform using a secure protocol (e.g., MQTT, HTTP, WebSocket).

Send the collected data in real-time to the platform. You may need to implement error handling and data buffering to ensure reliability.

Data Formatting:

Format the data in a structured way so that the transit information platform can process it efficiently. This might involve using JSON or other data formats.

Security:

Implement security measures, such as encryption and authentication, to protect the data being transmitted.

Testing and Debugging:

Test the Python script thoroughly in real-world conditions to ensure it's collecting and transmitting data correctly.

Implement logging and error handling to facilitate debugging.

Scalability:

Ensure that the system can scale to accommodate a large number of vehicles and sensors.

Integration with Transit Platform:

On the transit information platform, develop a component that can receive, process, and store the incoming data.

Create APIs or endpoints for the IoT sensors to interact with the platform.

Visualization and Analysis:

Implement a system for visualizing the data in real-time and for conducting data analysis. This could be done using web-based dashboards or analytics tools.

Maintenance and Updates:

Regularly update the Python script and the transit information platform to address bugs, security vulnerabilities, and to add new features.

Compliance:

Ensure that your system complies with any relevant data protection regulations and privacy requirements.This is a high-level overview of the process. The specifics of your project will depend on the choice of sensors, connectivity options, and the transit information platform you are working with. Be sure to consult the documentation of the selected hardware and software components for detailed implementation instructions

Certainly, here's a simplified Python script that sends real-time location and ridership data from IoT sensors to a hypothetical transit information platform. This is a basic example and may need to be adapted to your specific hardware and platform. For this example, I'm using MQTT as the communication protocol. Make sure to install the necessary libraries if not already installed (e.g., paho-mqtt for MQTT).

python

Copy code

import paho.mqtt.client as mqtt

import json

import random

import time

# Simulated IoT sensor data

vehicle\_id = "bus123"

passenger\_count = 0

# Connect to MQTT broker

mqtt\_broker = "mqtt.eclipse.org" # Replace with your broker's address

mqtt\_port = 1883

mqtt\_topic = "transit/realtime\_data"

def on\_connect(client, userdata, flags, rc):

print(f"Connected with result code {str(rc)}")

client.subscribe(mqtt\_topic)

def on\_publish(client, userdata, mid):

print("Message Published")

# Initialize MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_publish = on\_publish

client.connect(mqtt\_broker, mqtt\_port, 60)

while True:

# Simulate location data

latitude = round(random.uniform(30.0, 40.0), 6)

longitude = round(random.uniform(-100.0, -90.0), 6)

# Simulate passenger count data

passenger\_count += random.randint(0, 5)

# Create a data dictionary

data = {

"vehicle\_id": vehicle\_id,

"location": {

"latitude": latitude,

"longitude": longitude

},

"passenger\_count": passenger\_count

}

# Publish data to the MQTT topic

client.publish(mqtt\_topic, json.dumps(data))

print("Sent: ", data)

time.sleep(5) # Adjust the interval as needed

In this script:

We simulate data for the vehicle's location and passenger count. In a real-world scenario, you'd replace this with actual sensor data.

We use the MQTT protocol to publish the data to the specified topic on the MQTT broker. You should replace mqtt\_broker with the address of your actual MQTT broker.

The script runs in an infinite loop to continuously send data at a defined interval (5 seconds in this example).

Please adapt and expand this script to match your specific hardware, sensors, and transit platform. Additionally, make sure to handle errors and implement security measures if the data transfer involves sensitive information.