Certainly, let’s begin building the IoT-enabled public transportation optimization system. We’ll start with the initial steps:

1. **Define Project Scope and Objectives:** Clearly define what you aim to achieve with the system. Identify the key optimization goals for public transportation.

2. **Select IoT Sensors:** Choose appropriate IoT sensors (e.g., GPS, passenger counters) that will collect essential data for your optimization goals.

3. **Hardware Setup:** Install the selected sensors in a representative public transportation vehicle. Ensure they are securely mounted and properly powered.

4. **Data Communication:** Develop or configure the communication methods for IoT sensors to send data. You might choose MQTT, HTTP, or other protocols for data transmission.

5. **Transit Information Platform:** Set up the platform (server) that will receive and process data from the sensors. Consider using cloud services like AWS, Azure, or Google Cloud.

6. **Python Script Development:** Create Python scripts to run on the IoT sensors. These scripts should collect data (e.g., GPS coordinates and passenger counts) and send it to the transit information platform in real-time.

Please specify which specific part you’d like to dive into, or if you have any questions related to these initial steps.

Deploying IoT sensors like GPS and passenger counters in public transportation vehicles can provide valuable data for improving efficiency and service quality. These sensors can help track the vehicle’s location, monitor passenger occupancy, and collect various other data points. It’s crucial to ensure proper installation and data management to make the most of this technology.

To develop a Python script for IoT sensors to send real-time location and ridership data to a transit information platform, you'll need to use appropriate libraries and protocols for communication. Here's a high-level outline of what the script might look like:

~~~
```python
Import time
Import requests
Import json

# Define the IoT sensor's data
Vehicle_id = "Vehicle123"
Latitude = 40.7128  # Example latitude
Longitude = -74.0060  # Example longitude
Passenger_count = 30  # Example passenger count

# Define the transit information platform's API endpoint
Api_url = https://your-transit-platform-api.com/data

While True:
    # Collect sensor data
    Data = {
        "vehicle_id": vehicle_id,
        "latitude": latitude,
        "longitude": longitude,
        "passenger_count": passenger_count
    }

    # Send data to the transit information platform
    Headers = {"Content-Type": "application/json"}
~~~

```
    Response = requests.post(api_url, data=json.dumps(data), headers=headers)


    If response.status_code == 200:
        Print("Data sent successfully")
    Else:
        Print("Failed to send data. Status code:", response.status_code)


    # Adjust the time interval as needed for real-time updates
    Time.sleep(30)  # Send data every 30 seconds
```

In this script:

1. You define the IoT sensor's data, such as vehicle ID, latitude, longitude, and passenger count. These values can be obtained from the actual sensors on the public transportation vehicle.

2. You specify the API endpoint of the transit information platform where you want to send the data.

3. Inside the loop, the script collects sensor data, sends it to the platform using an HTTP POST request with JSON data, and checks the response status.

4. You can adjust the time interval (in seconds) based on your real-time requirements. The example sends data every 30 seconds, but you can modify it according to your needs.

Make sure to install the necessary libraries like `requests` for making HTTP requests and configure the script to match your specific IoT sensor and transit platform requirements.