In the communication.py, when a new vehicle or sensor is added, we’ll be able to add them to the VehicleDatabase.

In the communication.py, you can adapt it to interact with this database:

# If contains sensor and vehicle information

def handle\_vehicle\_message(message, vehicle\_db):

vehicle\_id = message['vehicle\_id']

sensor\_id = message['sensor\_id']

sensor\_type = message['sensor\_type']

sensor\_data = message.get('sensor\_data', {})

# Check if vehicle exists in the database, if not, create it

vehicle = vehicle\_db.get\_vehicle(vehicle\_id)

if not vehicle:

vehicle = Vehicle(vehicle\_id, message['vehicle\_model'])

vehicle\_db.add\_vehicle(vehicle)

# Create or update the sensor data

sensor = vehicle.get\_sensor(sensor\_id)

if not sensor:

sensor = Sensor(sensor\_id, sensor\_type)

vehicle\_db.add\_sensor\_to\_vehicle(vehicle\_id, sensor)

# Update sensor data

sensor.update\_data(sensor\_data)

Use the VehicleDatabase to store and track vehicles and sensors.

# Create a database instance

vehicle\_db = VehicleDatabase()

# Example incoming message for a new vehicle with sensor data

message = {

'vehicle\_id': 'vehicle\_1',

'vehicle\_model': 'Tesla Model S',

'sensor\_id': 'sensor\_1',

'sensor\_type': 'LIDAR',

'sensor\_data': {'distance': 100, 'angle': 30}

}

# Handle the vehicle message

handle\_vehicle\_message(message, vehicle\_db)

# Get the sensor data of a specific vehicle

vehicle\_data = vehicle\_db.get\_vehicle\_sensors('vehicle\_1')

print(vehicle\_data) # Output: {'sensor\_1': {'distance': 100, 'angle': 30}}