ห้ามใช้ AIในการทำแบบทดสอบ

Problem: Evacuation Planning and Monitoring API

Scenario:

You are tasked with building an **Evacuation Planning and Monitoring API** for a Disaster Response Team. This API will help coordinate evacuation operations in disaster-affected areas by optimizing the use of available vehicles and tracking the movement of evacuees to safe locations.

Requirements:

1. **API Endpoints**:
    - **POST /api/evacuation-zones**: Adds information about evacuation zones, including location, number of people needing evacuation, and urgency level.
    - **POST /api/vehicles**: Adds information about available vehicles, such as capacity, type, and location.
    - **POST /api/evacuations/plan**: Generates a plan that assigns vehicles to evacuation zones, prioritizing areas based on urgency and vehicle capacity.
    - **GET /api/evacuations/status**: Returns the current status of all evacuation zones, including the number of people evacuated and remaining.
    - **PUT /api/evacuations/update**: Updates the evacuation status by specifying the number of evacuees moved from an area and the vehicle used for the operation.
    - **DELETE /api/evacuations/clear**: Clears all current evacuation plans and resets the data (useful for restarting operations after a completed evacuation).
2. **Input Data**:
    - **Evacuation Zones**:
        - **Zone ID**: Unique identifier for the evacuation zone.
        - **Location Coordinates**: Latitude and longitude of the zone.
        - **Number of People**: Total number of people needing evacuation.
        - **Urgency Level**: Integer from 1 to 5 (1 = low urgency, 5 = high urgency).

- **Vehicles**:
    - **Vehicle ID**: Unique identifier for each vehicle.
    - **Capacity**: Number of people the vehicle can transport in one trip.
    - **Type**: Type of vehicle (e.g., bus, van, boat).
    - **Location Coordinates**: Latitude and longitude of the vehicle's current location.
    - **Speed**: Average speed of the vehicle in km/h.

3. **Logic Requirements**:
    - **Distance Calculation**: Calculate the distance between each vehicle and the evacuation zones to prioritize the closest vehicles.
    - **Urgency Priority**: Assign vehicles to zones with higher urgency levels first.
    - **Capacity Optimization**: Optimize assignments so that vehicles with the appropriate capacity are allocated to zones. For instance, if a zone has 50 people, a bus (capacity 40) should be preferred over multiple smaller vehicles, if available.
    - **Travel Time Estimation**: Estimate the time it will take each vehicle to reach the evacuation zone based on its speed and distance.
    - **Status Monitoring**: Track the number of evacuees transported from each zone and update the status with each trip.
4. **Output**:
    - **Evacuation Plan**: List of assignments where each assignment contains:
        - **Zone ID**
        - **Vehicle ID**
        - **Estimated Time of Arrival (ETA)**
        - **Number of People to be Evacuated**
    - **Evacuation Status**: For each zone, include:

        - **Zone ID**
        - **Total Evacuated**
        - **Remaining People**
        - **Last Vehicle Used** (optional)

5. **Special Features**:
    - **Distance Calculation using Haversine Formula**: To accurately calculate the distance between coordinates (lat/long), use the Haversine formula. This should be implemented as a helper method in your API.
    - **Status Storage in Redis**: Store the evacuation status of each zone in Redis to provide quick updates and allow for persistent monitoring.
    - **Azure Deployment**: Deploy the API to Azure for live testing and to demonstrate real-time evacuation planning and status updates.
    - **Logging**: Add logging to record each evacuation operation, including vehicle assignment, ETA, and completion status.

Example:

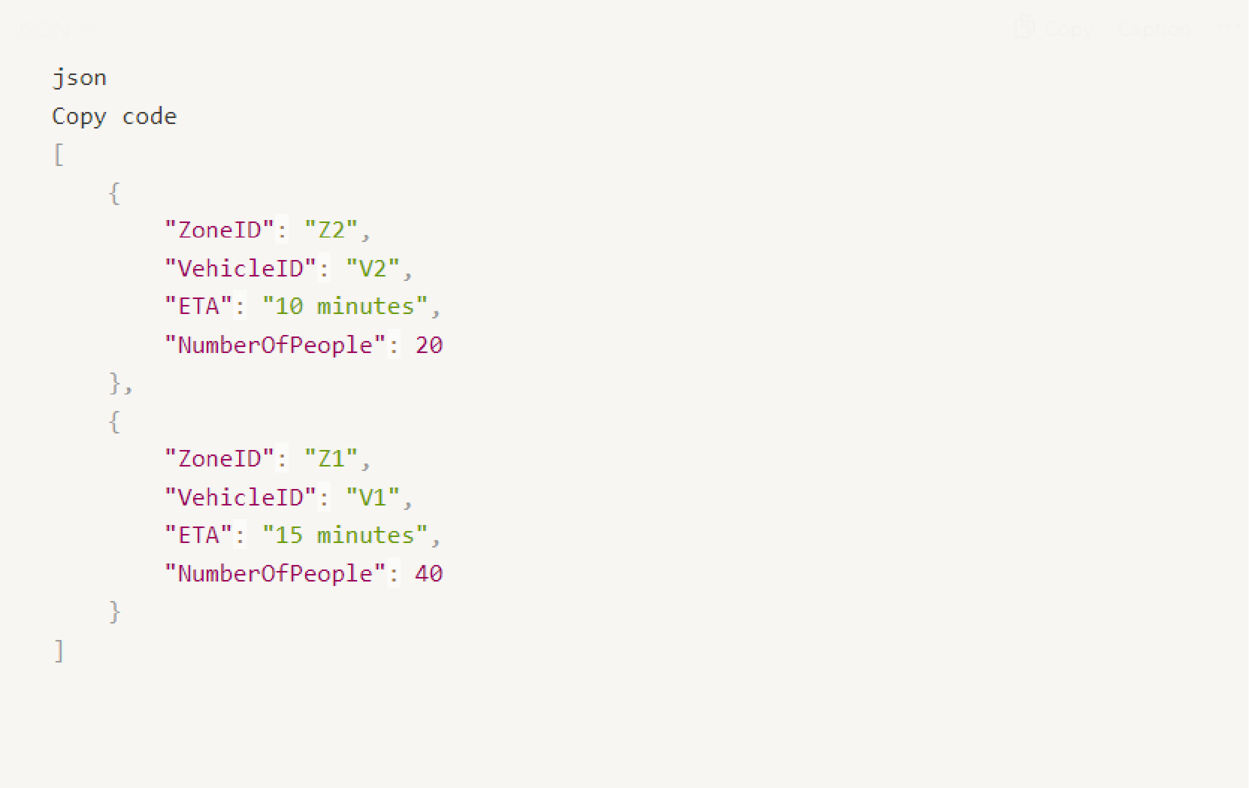
- Suppose you have the following data:
- **Evacuation Zones**:

```
json
Copy code
[
    {
        "ZoneID": "Z1",
        "LocationCoordinates": {"latitude": 13.7563, "longitude": 100.5018},
        "NumberOfPeople": 100,
        "UrgencyLevel": 4
    },
    {
        "ZoneID": "Z2",
        "LocationCoordinates": {"latitude": 13.7367, "longitude": 100.5231},
        "NumberOfPeople": 50,
        "UrgencyLevel": 5
    }
]
```

Vehicles:

```
json
Copy code
[
    {
        "VehicleID": "V1",
        "Capacity": 40,
        "Type": "bus",
        "LocationCoordinates": {"latitude": 13.7650, "longitude": 100.5381},
        "Speed": 60
    },
    {
        "VehicleID": "V2",
        "Capacity": 20,
        "Type": "van",
        "LocationCoordinates": {"latitude": 13.7320, "longitude": 100.5200},
        "Speed": 50
    }
]
```

Expected Evacuation Plan:



```
json
Copy code
[
    {
        "ZoneID": "Z2",
        "VehicleID": "V2",
        "ETA": "10 minutes",
        "NumberOfPeople": 20
    },
    {
        "ZoneID": "Z1",
        "VehicleID": "V1",
        "ETA": "15 minutes",
        "NumberOfPeople": 40
    }
]
```

Challenge:

1. **Implement the API Endpoints**: Develop each endpoint based on the logic and requirements specified.
2. **Redis Integration for Monitoring**: Use Redis to store and retrieve the evacuation status, enabling quick access to real-time data.
3. **Deploy to Azure**: Deploy the API on Azure and share the URL to demonstrate its live functionality.
4. **Distance Calculation**: Use the Haversine formula to calculate distances and ensure your API accurately considers travel time.
5. **Error Handling**: Account for situations like:
   - No available vehicles within a reasonable distance.
   - Vehicles with insufficient capacity for larger zones.
   - Simultaneous requests for the same vehicle (use locking mechanisms if needed).

This problem tests the developer’s ability to implement complex logic, integrate third-party tools (Redis), and deploy applications on the cloud for real-time operational needs. Good luck!

การส่งแบบทดสอบ

1.ส่งวิดีโอพร้อมอธิบายการทำงานของโจทย์ที่ได้รับ

2.ส่ง Source code แบบทดสอบ