Autonomous electric vehicles

PROBLEM STATEMENT

Design and implement a microservice for an autonomous electric vehicle system with the following features:

- Check if the vehicle can reach the destination without charging
- Find charging stations - If the destination can not be reached with current charging level. Appropriate handling to be done if the destination cannot be reached even with charging

The features are described in detail below:

Check if vehicle can reach destination without charging

Create a microservice which exposes one REST endpoint and can receive the request in the given request format

```
{ "vin": "vehicle identification number eg: W1K2062161F0014", "source": "source name", "destination": "destinat
ion name" }
```

and the response format will be-

```
{ "transactionId": "043020211 //A unique numerical value", "vin": "W1K2062161F0014 //vehicle identification nu
mber", "source": "source name", "destination": "destination name", "distance": "100 //distance between the sour
ce and destination in miles", "currentChargeLevel": "1 //current charge level in percentage , 0<=charge<=100",
"isChargingRequired": "true/false //whether the vehicle has to stop for charging?.If true populate charging statio
ns", "chargingStations": [ "s1", "s2" ], "errors": [ { "Id": 8888, "description": "Unable to reach the destination with t
he current charge level" }, { "id": 9999, "description": "Technical Exception" } ] }
```

Find out whether the vehicle will be able to reach the destination with the current charge level without charging at any charging stations in between the source and the destination. Assume 1% of charge is required for travelling 1 mile.

Example: Assume the distance between the source and the destination is 50 miles and the available charge level is 60%. The vehicle will reach the destination without stopping anywhere as the current charge level is sufficient to travel 50 miles.

You can use the following APIs:

Current Charge Level

Call the following API for getting the current charge level of the vehicle-

URL- https://restmock.techgig.com/merc/charge_level

Method- POST

Request:

```
{ "vin": "vehicle identification number"}
```

Response:

```
{ "vin": "vehicle identification number", "currentChargeLevel": "current battery charge level in percentage, 0<=ch
arge<=100 eg: 1", "error": "It will be null if No Error" }
```

Find Travel Distance

Use the following API for getting the distance between the source and the

destination.

URL- https://restmock.techgig.com/merc/distance

Method- POST

Request:

```
{ "source": "source name", "destination": "destination name" }
```

Response:

```
{ "source": "source name", "destination": "destination name""distance": "100 //distance between the source and
destination in miles", "error": "It will be null if No Error" }
```

Find charging station If the vehicle is not able to reach the destination with the current charge level, find out the charging stations where the vehicle has to stop for recharging the battery, from the list of available charging stations between the source and destination. If there are multiple answers possible print the lexicographically smallest possible.

Lexicographic order: It means the dictionary order. The one which is small will be present earlier.

Example: The vehicle requires only 1 station to recharge in between. Station 1 and Station 2 are both good options for him. The answer would be Station 1 as it is lexicographically smallest.

Use the following API to get available charging stations between the source and destination-

URL- https://restmock.techgig.com/merc/charging_stations

Method- POST

Request:

```
{ "source": "source name", "destination": "destination name" }
```

Response:

```
{ "source": "source name", "destination": "destination name", "chargingStations": [ { "name": "s1 //Name of the c
harging station", "distance": "10 //Distance from the source in miles", "limit": "60 //Available charge level. If the c
urrent battery level is 1%, vehicle can charge upto 61% from this station " }, { "name": "s2", "distance": "20", "lim
it": "30" }, { "name": "s3", "distance": "30", "limit": "30" }, { "name": "s4", "distance": "60", "limit": "40" } ], "error": "I
t will be null if No Error" }
```

Note: Vehicle should stop as minimum as possible, i.e., the vehicle should use minimum number of charging stations.

If the vehicle cannot reach the destination even after recharging at the charging stations, return the following error code in response.

```
"error": [ { "id": 8888, "description": "Unable to reach the destination with the current charge level " } ]
```

If the provided input is invalid for any of the given parameters, return the following error code in the response.

```
"errors": [ { "id": 9999, "description": "Technical Exception" } ]
```

Rules to consider

If a vehicle stops at a station with a charging limit of X, it can recharge the battery up to - currentChargeLevel + X.

For example: Limit at a station is 60% and the current charge level is 1%. The vehicle can be charged up to 1+60= 61%

- If the vehicle reaches the destination with zero charge, that means it has arrived.
- If the vehicle reaches the charging station with zero charge, this means that it can charge.
- Assume 1% of charge is required for travelling 1 mile
- Charging level for the vehicle, 0<= charge <=100
- 0<=Number of stations<= distance between the source and destination

Sample Data

For unit testing, consider the following set of locations as destination

- National Library
- Central Park

- Movie Theatre
- Zoo
- Lake
- Power plant
- Home - Lake
    - Distance= 20 mile
    - Charging stations= [S1,10,100] //Format[Name, Distance from source, Limit]
- Home - Power Plant
    - Distance= 30 mile
    - Charging stations= [ ] // no charging stations
- Home - National Library
    - Distance= 80 mile
    - Charging stations= [S1,10,30], [S2, 30, 50], [S3, 70, 20]
- Home - Zoo
    - Distance= 140 mile
    - Charging stations= [S1,30,100] , [S2, 70, 10], [S3, 90, 90], [S4, 130, 30]
- Home - Central Park
    - Distance= 100 mile
    - Charging stations= [[S1,10,60],[S2,20,30],[S3,30,30],[S4,60,40]]
- Home - Movie Theatre
    - Distance= 50 mile
    - Charging stations= [[S1,10,20],[S2,25,15],[S3,33,10],[S4,40,10]]

and some valid vehicle identification numbers will be:

- W1K2062161F0014
- W1K2062161F0015
- W1K2062161F0016
- W1K2062161F0017
- W1K2062161F0113

Sample Test Cases

Test case 1

Input:

```
{ "vin": "W1K2062161F0033", "source": "Home", "destination": "Lake" }
```

Output:

```
{ "transactionId": 2, "vin": "W1K2062161F0033", "source": "Home", "destination": "Lake", "distance": 20, "current
ChargeLevel": 93, "isChargingRequired": false }
```

Explanation:

Results from external call:

- Distance: 20
- Charge level: 93
- Charging stations: [S1,10,100]

The current charging level for the given vehicle is 93.

The vehicle has to move from Home(source) to Lake(destination). The distance between the source and the destination is 20 miles.

With the current charge level(93), the vehicle does not need to stop to recharge at the charging station S1 and has sufficient charging to reach the destination from the source.

Test case 2

Input:

```
{ "vin": "W1K2062161F0080", "source": "Home", "destination": "Airport" }
```

Output:

```
{ "transactionId": 3, "vin": "W1K2062161F0080", "source": "Home", "destination": "Airport", "distance": 60, "curre
ntChargeLevel": 2, "isChargingRequired": true, "errors": [ { "id": 8888, "description": "Unable to reach the destin
ation with the current fuel level" } ] }
```

Explanation:

Results from external call:

Distance: 60
Charge level: 2
Charging stations: [S1,10,10],[S2,25,5]

The distance between Home(Source) and Airport(Destination) is 60 miles. The current charging level of the vehicle is 2. It is not possible with the current charge level to reach the destination. Also, it is not possible for the vehicle to reach any of the charging stations.

<**Note:** Even if the vehicle was able to reach the charging stations, they would not have provided enough recharge level to the vehicle to reach the destination.

Test case 3

Input:

```
{ "vin": "W1K2062161F0080", "source": "@$%%%", "destination": "Airport" }
```

Output:

```
{"transactionId": 1,"errors": [{"id": 9999,"description": "Technical Exception"}]}
```

Explanation:

For the given input, the source value is 0 which is an invalid value for the source name. The response would be an error depicted above.

Test Case 4:

Input:

```
{ "vin": "W1K2062161F0046", "source": "Home", "destination": "Movie Theatre" }
```

Output:

```
{ "transactionId": 5, "vin": "W1K2062161F0046", "source": "Home", "destination": "Movie Theatre", "distance": 5
0, "currentChargeLevel": 17, "isChargingRequired": true, "chargingStations": [ { "name": "S1", "distance": 10, "li
mit": 20 }, { "name": "S2", "distance": 25, "limit": 15 } ] }
```

Explanation:
Results from external call:
Distance: 50
Charge level: 17
Charging stations: [S1,10,20],[S2,25,15],[S3,33,10],[S4,40,10]

The distance between Home(source) and Movie Theatre(destination) is 50 miles. The current charge level of the vehicle is 17. The vehicle needs to stop at the charging stations to reach the destination. The aim is to minimize these stops.

Step 1: The vehicle will have to stop at charging station S1. The charge level now will be 7 + 20 = 27. (The vehicle has already spent 10% charge level to reach the charging station, S1). With the current charge level (27), it is still not possible to reach the destination.

Step 2: The vehicle will have to stop at charging station S2. The charge level now will be 12 + 15 = 27. (The vehicle has spent 15% charge level to reach the charging station, S2 from S1). Now, it is possible to reach the destination with the current charging level. The distance left is 25 and the current charging level is 27. Vehicle will not stop any further.

Note: After stop 1,the charging level was 27. The vehicle could have reached charging station 2 or 3 but we chose charging station 2 because the vehicle after recharging at station 3 would not have been able to reach the destination and would require the vehicle to stop at charging station 4 making the number of stops to be 3.

So, the minimum number of stops required is 2, i.e., charging stations S1 and S2.

IMPORTANT INSTRUCTION

Microservice code should contain Dockerfile and README.md which should have the necessary steps to build the image and start service at port 8080.

EVALUATION CRITERIA

- The panel of evaluators will evaluate the solutions based on the following criteria:
- Working prototype
- Test case performance
- Documentations

DELIVERABLES

- **Source repo:** This should contain all the necessary source files of a complete application containing Dockerfile and README file. The README file should contain clear step-by-step instructions on how to build, deploy, and use the services.
- **Presentation:** This should contain an architecture diagram of the application, challenges, solution demonstration, and why your solution should be considered for the next round.