

# Vehicle finder assessment:

Author: Godwill Makhubela

Cell: 081 592 0542

Email: [godwill.makhubela@gmail.com](mailto:godwill.makhubela@gmail.com) / [godwillm@cybermite.co.za](mailto:godwillm@cybermite.co.za)

## Forword:

I would like to take this time to thank Mix Telematics for inviting me to this interview process. Instead of developing just a console application for this solution I have taken a rather different approach to showcase my dotnet skills where I have used dotnet core 6 and some design patterns to solve the problem as per the requirement document.

To elaborate, I have developed a vehicle finder solution (**VehicleFindetSLN**) in Visual Studio 2022 which consist of a seamless WEBAPI project named **VehicleFinderBackend**. The reason for this is to allow a loosely coupled integration should it be used as a microservice running separately and making provision for a front-end integration via an endpoint. In this approach I have exposed two methods on the API's endpoint namely: **GetVehiclePositionByCoordinates** – which takes two floating-point values of latitude and longitude in a JSON string and returns a payload of a vehicle data parameters in JSON, and **GetVehiclePositionsByLsofCoordinates** - which takes a list of coordinates in JSON and returns a JSON payload consisting of a list of searched vehicle data parameters.

For simplicity I have attached Swagger suite for easy API testing and documentation. To improve the issue of latency during the vehicle search on the vehicle positions datafile which is provided, I have applied dependency injection whereby the service which reads the datafile is injected as an object hence cutting down the time taken in instantiating objects each time the datafile reading services are called. Furthermore, I have applied interface segregation and single responsibility principles.

## Testing the solution:

### Requirements:

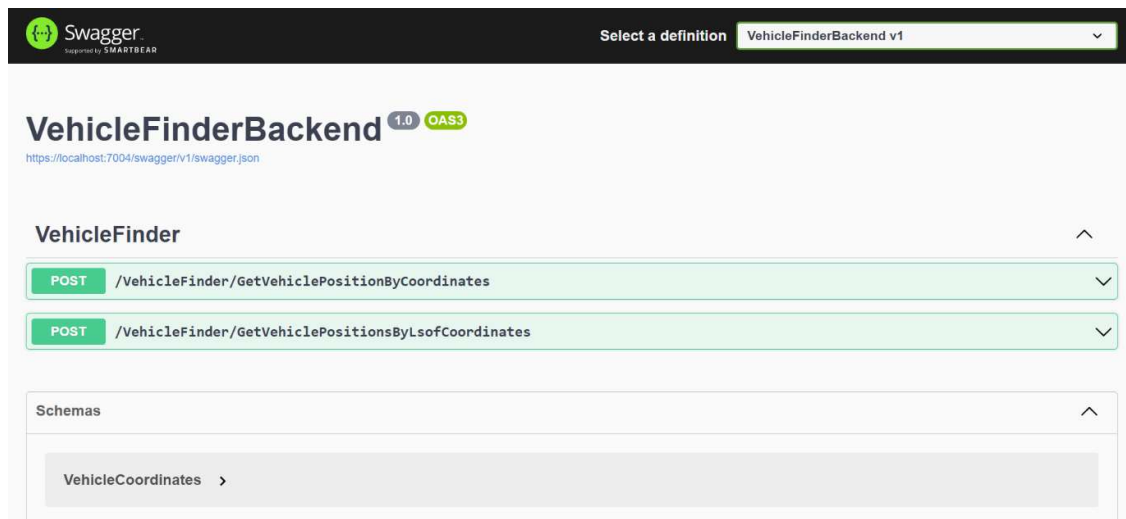
*Before running the solution, please ensure that DON NET CORE 6 is installed and Visual studio 2022 which currently support dotnet core 6, Alternatively you can use Visual studio code.*

*This project has some values displaying in console, so if you are running in Visual Studio, make sure to run the project in Kestrel mode.*

**Lets GO!**

Clone the project then build and run the project in kestrel mode to download required packages and observe console output.

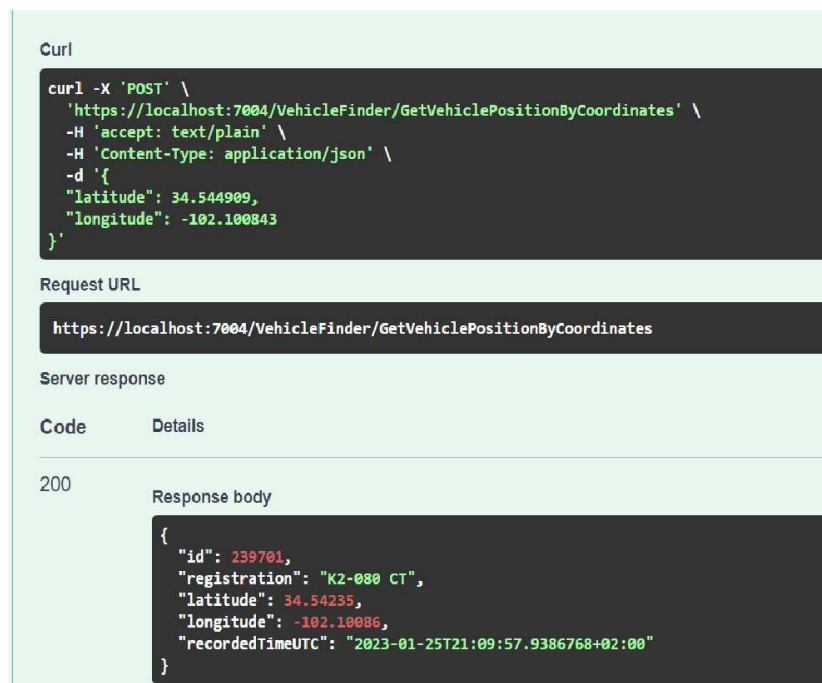
1. A swagger front-end will pop up in a browser with the following:



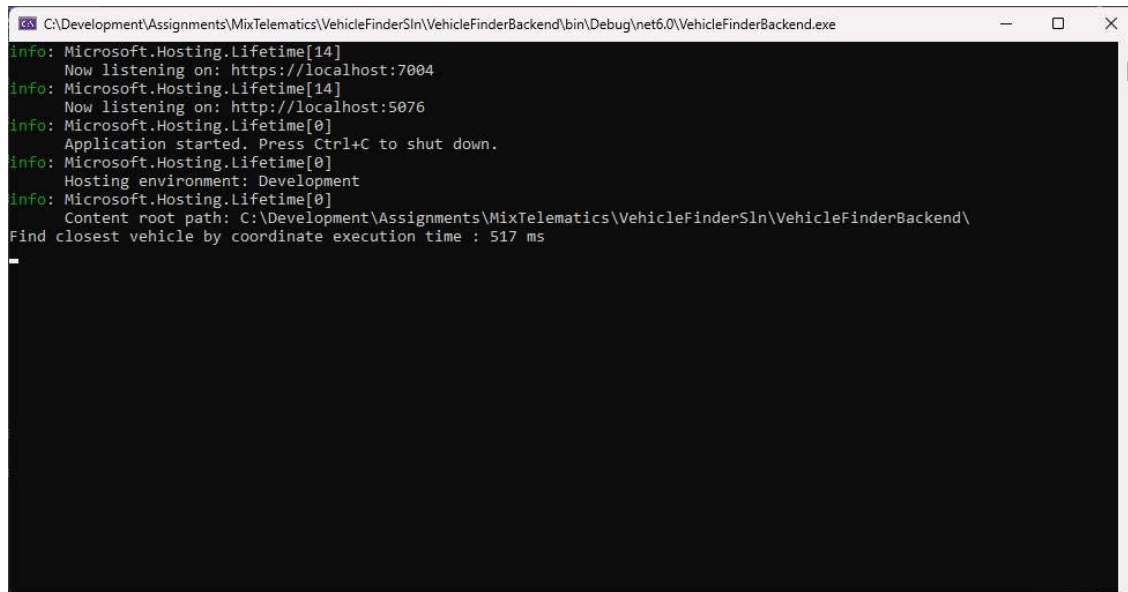
2. Click on either of the exposed methods and use the following JSON input payloads to test:
  - a. For the **GetVehiclePositionByCoordinates** click on **Try it out** and paste this on the request body, replacing the current JSON, then click on execute.

```
{
  "latitude": 34.544909,
  "longitude": -102.100843
}
```

And this should return this response after clicking execute:



And the CMD.exe output from should look like the picture below, showing the execution time of the request.



```
C:\Development\Assignments\MixTelematics\VehicleFinderSln\VehicleFinderBackend\bin\Debug\net6.0\VehicleFinderBackend.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7004
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5076
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Development\Assignments\MixTelematics\VehicleFinderSln\VehicleFinderBackend\
Find closest vehicle by coordinate execution time : 517 ms
```

- b. For the **GetVehiclePositionsByLsofCoordinates** click on **Try it out** and paste this on the request body, replacing the current JSON, then click on execute.

```
{
  "latitude": 34.544909,
  "longitude": -102.100843
},
{
  "latitude": 32.345544,
  "longitude": -99.123124
},
{
  "latitude": 33.234235,
  "longitude": -100.214124
},
{
  "latitude": 35.195739,
  "longitude": -95.348899
},
{
  "latitude": 31.895839,
  "longitude": -97.789573
},
{
  "latitude": 32.895839,
  "longitude": -101.789573
},
{
  "latitude": 34.115839,
  "longitude": -100.225732
},
{
  "latitude": 32.345544,
  "longitude": -99.123124
},
{
  "latitude": 32.335839,
  "longitude": -99.992232
},
{
  "latitude": 33.535339,
  "longitude": -94.792232
},
{
  "latitude": 32.234235,
  "longitude": -100.222222
}
]
```

And this should return this response after clicking execute:

Request URL

```
https://localhost:7004/VehicleFinder/GetVehiclePositionsByLsofCoordinates
```

Server response

Code	Details
200	<p>Response body</p> <pre>[   {     "id": 239701,     "registration": "K2-080 CT",     "latitude": 34.54235,     "longitude": -102.10086,     "recordedTimeUTC": "2023-01-25T21:18:22.2893365+02:00"   },   {     "id": 864907,     "registration": "3I-388 XE",     "latitude": 32.344444,     "longitude": -99.12403,     "recordedTimeUTC": "2023-01-25T21:18:24.4979854+02:00"   },   {     "id": 835420,     "registration": "77-0126 BV",     "latitude": 33.23224,     "longitude": -100.21374,     "recordedTimeUTC": "2023-01-25T21:18:22.9576861+02:00"   },   {     "id": 1590684,     "registration": "65-31272 KE",     "latitude": 35.195534,     "longitude": -95.34728,     "recordedTimeUTC": "2023-01-25T21:18:23.2648137+02:00"   } ]</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 25 Jan 2023 19:18:24 GMT server: Kestrel</pre>

Responses

Code	Description
200	Success

And the CMD.exe output from should look like the picture below, showing the execution time of the request.

```
C:\Development\Assignments\MixTelematics\VehicleFinderSln\VehicleFinderBackend\bin\Debug\net6.0\VehicleFinderBackend.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7004
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5076
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Development\Assignments\MixTelematics\VehicleFinderSln\VehicleFinderBackend\
Find closest vehicle by coordinate execution time : 517 ms
Find list of vehicles per coordinates execution time : 3558 ms
```

*Thank you, please feel free to contact me should you require a live demo into this and or go through the code.*