Car Data API – Spring boot

In this challenge, you are part of a team that is building a Car Data API. One requirement is for a REST API service to manage car data details using Spring boot. You will need to add functionality to add and show car data as well as list all the car data in the system. The team has come up with a set of requirements, including the API format, response codes, and the structure for the queries you must implement.

The definitions and detailed requirements list follow. You will be graded on whether your application performs data retrieval and manipulation based on given use cases exactly as described in the requirements.

Each product has the following structure:

- id: the unique ID for the car data [int]
- model: the name of the model [String]
- manufacturedYear: The manufactured year of the model [int]
- company: company name [String]
- bodyType: The type of the model [String]
- price: The price of the model [float]
- fuelType : The fuel type of the body type [String]
- seatingCapacity: The seating capacity of the body type [int]

Sample Product JSON

```
{

“model”: “City”,

“manufacturedYear”: 2024,

“company”: “Honda”,

“bodyType”: ” sedan”,

“price”: 2000000.0“,

“fuelType”: “petrol”,

“seatingCapacity”: 5

}
```

API’s:

“/carData/add”:

Post Method:

Get the details (model, manufacturedYear, company, bodyType, price, fuelType, seatingCapacity) from the user and store it in the database, then return it as a response with status code 201.

```
{
“model”: “City”,
“manufacturedYear”: 2024,
“company”: “Honda”,
“bodyType”:” sedan”,
“price”: 2000000.0“,
“fuelType”: “petrol”,
“seatingCapacity”: 5
}
```

Request

```
{
    “id”: 1,
    “model”: “City”,
     “manufacturedYear”: 2024,
     “company”: “Honda”,
     “bodyType”:” sedan”,
     “price”: 2000000.0“,
      “fuelType”: “petrol”,
     “seatingCapacity”: 5
  }
```

Response

If is there any error while posting the detail, then return the error with status code 400 as a Response.

“/carData/list”:

Get Method:

Get all the objects from the carData model then return it as a response with status code 200.

Request: localhost:8080/carData/list

[{“id”:1, “model”: “City”, “manufacturedYear”: 2024, “company”: “Honda”,

“bodyType”: ” sedan”, “price”: 2000000.0“, “fuelType”: “petrol”, “seatingCapacity”: 5},

{“id”:2, “model”: “Nexon”, “manufacturedYear”: 2023, “company”: “Tata”,

“bodyType”: ” SUV”, “price”:1650000.0“, “fuelType”: “EV”, “seatingCapacity”: 7}]

Response

If no data is available in the carData model, then return “No Data Available” with a status code 400.

“/carData/update/<int:id>”:

Patch Method:

Get the detail [price] to be patched from the user and update it in carData object with given id, then return it as a response with status code 200.

{

“price”: 800000.0

}

Request

{

“id”:1,

“model”: “City”,

“manufacturedYear”: 2024,

“company”: “Honda”,

“bodyType”: ” sedan”,

“price”: 800000.0 “,

“fuelType”: “petrol”,

“seatingCapacity”: 5

}

Response

If no data provided in the request body to update, then return “No Data Provided” as a response with status code 400.

If the given id not found, then return message “Not Found” with status code 404.

“/carData/delete/<int:id>”:

Delete Method:

Get the object with given id from carData model and delete it then return 200 as status

code.

If the given id not found, then return message “Not found” with status code 404.