1. Node.js - Car details

Car Details - Node.js

In this challenge, you need to create a back-end application to manage the details of cars using **Node.js** and **MongoDB.**

Database -> **cars**

Collections -> **cars**

Collections:

There is a single file for collections namely **cars.js** that resides inside **src/mongoose/models**. The schema for the collections is given below,

cars		
sl.no	**field_name**	**types**
1	_id	ObjectID
2	name	String
3	type	String
4	price	Number
5	manufacturer	String
6	capacity	Number
7	__v	Number

Routers:

There is a single file namely **cars.js** that contain all the endpoints of the app and resides inside **src/routers**. The endpoints and their functionalities are given below.

1) */cars* -> POST Method -> This route should create a new document inside the **cars** collection with the data that comes with the request body.

Sample data sent with the request:

```
{
  "name": "New car",
  "price": 2040000,
  "capacity": 5,
  "type": "Sedan",
  "manufacturer": "Hundai"
}
```

- If the route was executed successfully and the data was created in the database then, you should send a response code of **201**
- If something went wrong and the creation was unsuccessful then, you should send a response code of **400**.

2) ***/cars*** **-> GET Method ->** This method should fetch the data from the **cars** collection. This method also has four optional query parameters namely **price**, **manufacturer**, **capacity** and **search.**

- If no query parameters are passed with the request, then all the data from the cars collection should be fetched as the response.
- If the **search** query parameter is passed with the request, then the data that has the value of search query parameter either in name or manufacturer field should be fetched. **Case sensitivity should be ignored**.
- If the **capacity** query parameter is passed with the request, then the data that has the value of capacity field equal to the value of capacity query parameter should be fetched.
- If the **manufacturer** query parameter is passed with the request, then the data that has the value of manufacturer field equal to the value of manufacturer query parameter should be fetched.
- If **price** parameter is sent with the request, then based on the value of the price parameter, the data should be fetched as follows:
  - If the value of price is **asc,** then the data should be ordered in ascending order of the price parameter.
  - If the value of price is **desc,** then the data should be ordered in descending order of the price parameter.
- Note that **more than one query parameter** can be passed with the request. In such cases all the filters should be applied and the data should be fetched.
- If the data was fetched successfully, then you should send a response code of **200**.
- If something went wrong in the execution of the request and the data fetching was unsuccessful, then you should send a response code of **400**.

3) ***/cars/:id*** **-> PATCH Method ->** This route should update the data in the **cars** collection that has the _id equal to the id that comes with the request URL. The data to be updated will be sent as the request body.

Sample request: /cars/63be57ab7f0eca3e9db6a3b6

```
{
  "price": 1960000
}
```

The price of the data that has the _id equal to 63be57ab7f0eca3e9db6a3b6 should be updated as 1960000.

- If the route was executed successfully and the data was updated in the database then, you should send a response code of **200.**
- If something went wrong and the updation was unsuccessful then, you should send a response code of **400**.

4) ***/cars/:id*** **-> DELETE Method ->** This route should delete the data in the **cars** collection that has the _id equal to the id that comes with the request URL.

Sample request: /cars/63be57ab7f0eca3e9db6a3b6

The data that has the _id equal to 63be57ab7f0eca3e9db6a3b6 should be deleted from the database.

- If the route was executed successfully and the data was deleted in the database then, you should send a response code of **200.**
- If something went wrong and the deletion was unsuccessful then, you should send a response code of **400**.

MongoDB commands:

- You can open the mongo shell by running ***mongo*** from the terminal.
- You can view all the data from the database in MongoDB by running ***show dbs*** from the mongo shell.
- You can select the database by running **use cars**.
- You can view the names of collections by running ***show collections***.
- You can view the data inside a collection by running ***db.cars.find().***
- Enter ***ctrl+c*** to exit.