

Day 3 KT Task Submission

- College Name: Dhanalakshmi srinivasan college of engineering and technology
- College Code :3105
- Name : Vijay anand.S
- Department :B.E/CSE(Computer science and engineering)
- Naan Mudhalvan (NM)
ID:8A295265B95B015B6974A2C381057C7B
- Git Repository : <https://github.com/Vijay-Anand-VJ/NM-IBM>
- Phone Number:6379313305
- Email ID: s.vijay.102006@gmail.com

Rental Car Management API

Project Description

I have developed a Backend API for a Rental Car Management System. This project allows users to manage a car inventory using standard API requests.

Source Code:

The complete source code, installation steps, and API documentation are available in the GitHub Repository linked above.

Repository Link: <https://github.com/Vijay-Anand-VJ/NM-IBM>

Key Features Completed:

- Built using Node.js and Express.js.
- Implemented CRUD Operations (Create, Read, Update, Delete).
- Used a JSON file (cars.json) as a database to store car details.
- Documented all API endpoints in the repository README.

Project Output (Screenshots)

1. Creating a Car (POST Request)

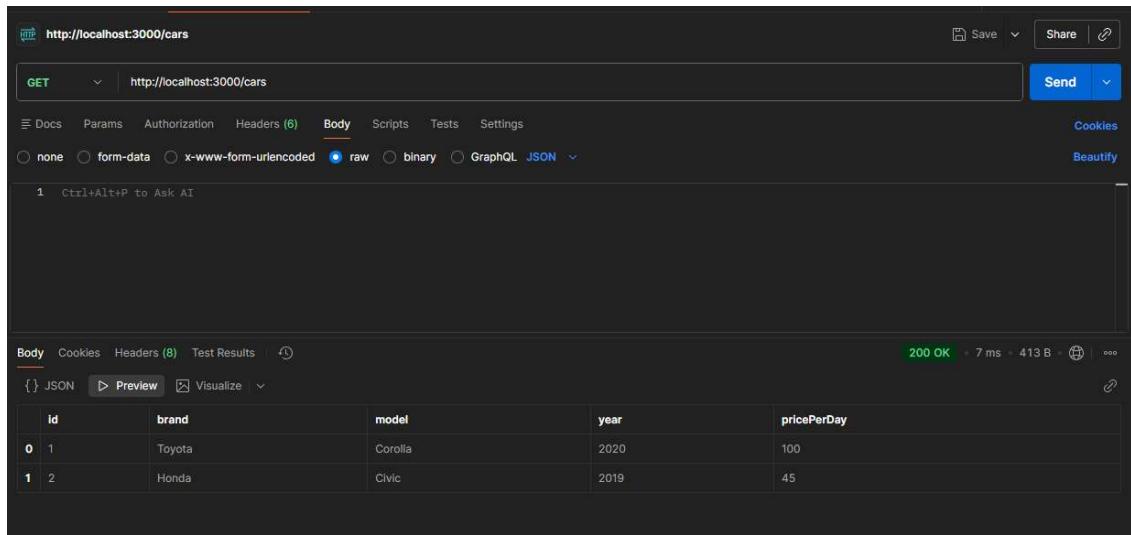
The screenshot shows a Postman interface. The URL is `http://localhost:3000/cars`. The method is set to `POST`. In the `Body` tab, the `raw` option is selected, and the JSON payload is:

```
1 {
2   "brand": "Tesla",
3   "model": "Model 3",
4   "year": 2024,
5   "pricePerDay": 120
6 }
```

The response status is `200 OK` with a response time of `10 ms` and a size of `486 B`. The response data is displayed in a table:

	<code>id</code>	<code>brand</code>	<code>model</code>	<code>year</code>	<code>pricePerDay</code>
0	1	Toyota	Corolla	2020	100
1	2	Honda	Civic	2019	45
2	3	Tesla	Model 3	2024	120

2. Reading Data (GET Request)



HTTP <http://localhost:3000/cars>

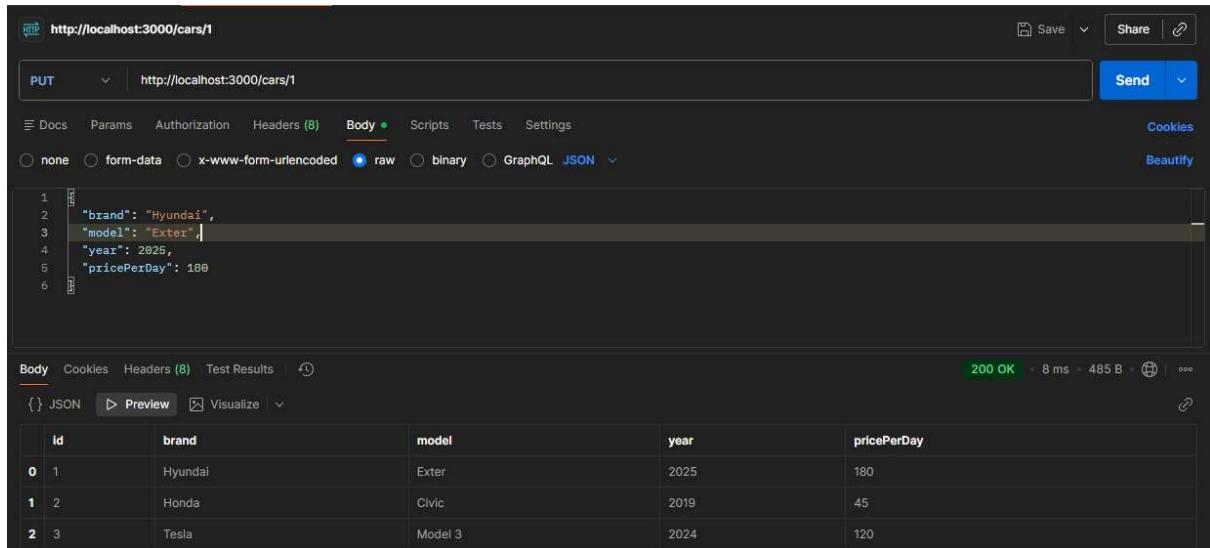
GET <http://localhost:3000/cars>

Body raw binary GraphQL JSON

200 OK · 7 ms · 413 B

	id	brand	model	year	pricePerDay
0	1	Toyota	Corolla	2020	100
1	2	Honda	Civic	2019	45

3. Updating a Car (PUT/PATCH Request)



HTTP <http://localhost:3000/cars/1>

PUT <http://localhost:3000/cars/1>

Body raw binary GraphQL JSON

200 OK · 8 ms · 485 B

	id	brand	model	year	pricePerDay
0	1	Hyundai	Exter	2025	180
1	2	Honda	Civic	2019	45
2	3	Tesla	Model 3	2024	120

4. Deleting a Car (DELETE Request)

The screenshot shows the Postman application interface. At the top, the URL is set to `http://localhost:3000/cars/2`. The method dropdown shows `DELETE`. Below the URL, there are tabs for `Docs`, `Params`, `Authorization`, `Headers (6)`, `Body`, `Scripts`, `Tests`, and `Settings`. The `Body` tab is selected. Under the `Body` tab, there are options for `none`, `form-data`, `x-www-form-urlencoded`, `raw` (which is selected), `binary`, `GraphQL`, and `JSON`. A note at the bottom says `1 Ctrl+Alt+P to Ask AI`. In the main body area, there is a JSON object with one field: `message: "Car deleted successfully"`. Below the body, the response status is `200 OK`, with a green success icon, and it indicates the response took `9 ms` and was `305 B` in size. There is also a `Preview` button.