

## 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](mailto: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.

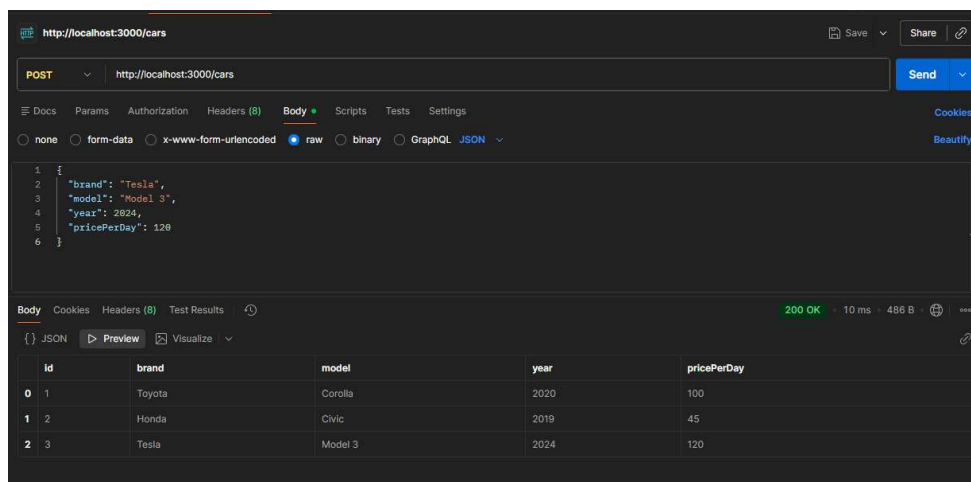
Repository Link: <https://github.com/Vijay-Anand-VJ/NM-IBM>

## 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 REST client interface with the following details:

- URL:** `http://localhost:3000/cars`
- Method:** POST
- Body (raw JSON):**

```
1 {
2   "brand": "Tesla",
3   "model": "Model 3",
4   "year": 2024,
5   "pricePerDay": 120
6 }
```
- Status:** 200 OK, 10 ms, 486 B
- Response Body (Table):**

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

## 2. Reading Data (GET Request)

Postman interface showing a GET request to `http://localhost:3000/cars`. The response is a 200 OK status with a 7 ms response time and 413 B of data. The body is displayed as a JSON array of two car objects.

```
{
  "cars": [
    {
      "id": 1,
      "brand": "Toyota",
      "model": "Corolla",
      "year": 2020,
      "pricePerDay": 100
    },
    {
      "id": 2,
      "brand": "Honda",
      "model": "Civic",
      "year": 2019,
      "pricePerDay": 45
    }
  ]
}
```

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

## 3. Updating a Car (PUT/PATCH Request)

Postman interface showing a PUT request to `http://localhost:3000/cars/1`. The request body is a JSON object with car details. The response is a 200 OK status with an 8 ms response time and 485 B of data. The body is displayed as a JSON array of three car objects.

```
{
  "brand": "Hyundai",
  "model": "Exter",
  "year": 2025,
  "pricePerDay": 180
}
```

	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)

