

Car Rental System API using C#

Cars

POST /api/Cars/rent

GET /api/Cars/BookingHistory

GET /api/Cars/available

GET /api/Cars/{id}

DELETE /api/Cars/{id}

POST /api/Cars

PUT /api/Cars

User

POST /api/User/register

POST /api/User/login

Car Model

The Car class represents a car in the car rental system and includes the following properties:

- **Id:** A unique identifier for the car.
- **Make:** The make of the car (e.g., Toyota, Ford). It is a required field, and the length must be between 1 and 50 characters.
- **Model:** The model of the car. It is also a required field, with the length constrained between 1 and 50 characters.

- **Year:** The manufacturing year of the car. The custom validation ValidYear ensures that this field contains a valid year value starting from 1990 to current year.
- **PricePerDay:** The rental price per day for the car. It must be a positive value greater than 0.
- **isAvailable:** A boolean flag indicating if the car is currently available for rent. It defaults to true (i.e., available).

User Model

The User class represents a user in the system (either user or an admin) and includes the following properties:

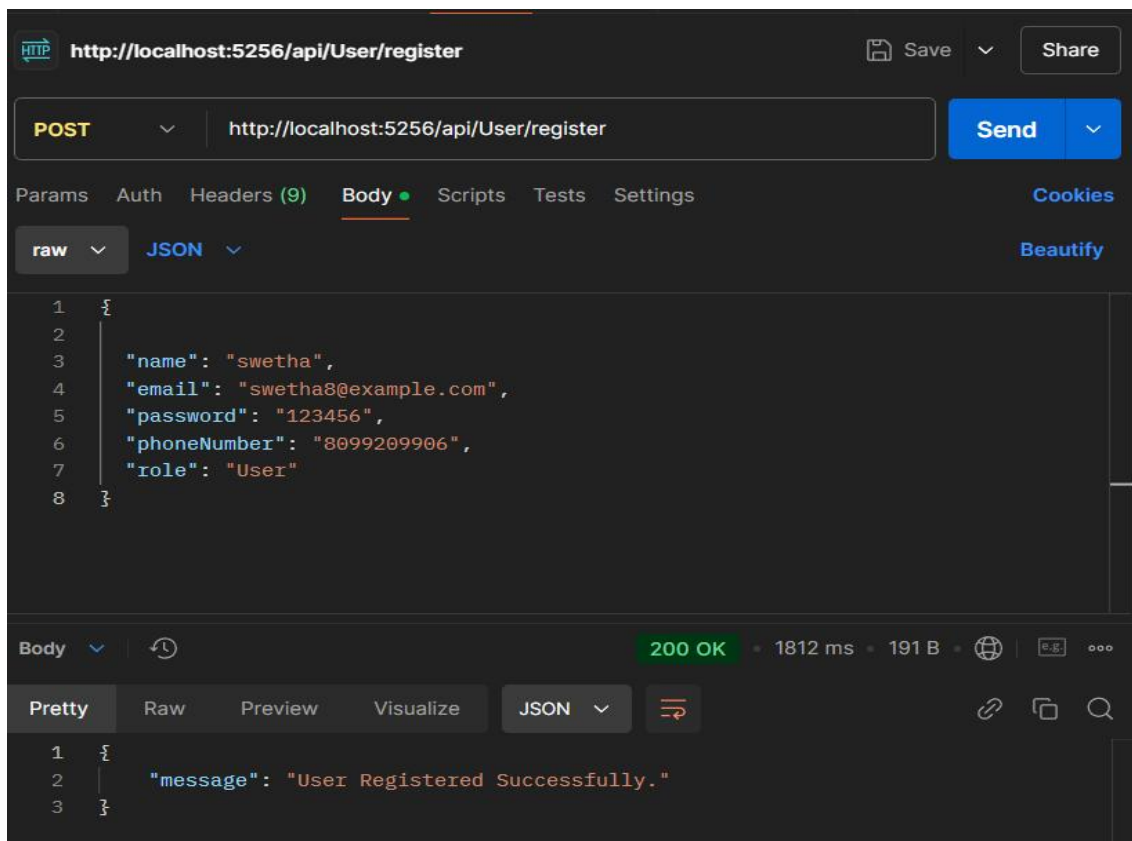
- **Id:** A unique identifier for the user.
- **Name:** The name of the user. It is required and must have a length between 1 and 100 characters.
- **Email:** The user's email address. It is required and must be a valid email format.
- **Password:** The password for user authentication. It is required, and the length must be at least 6 characters.
- **PhoneNumber:** The user's phone number. It is required and must be exactly 10 digits.
- **Role:** The role of the user (either "Admin" or "User"). It is required and validated to ensure that only these two values are allowed.

1.User

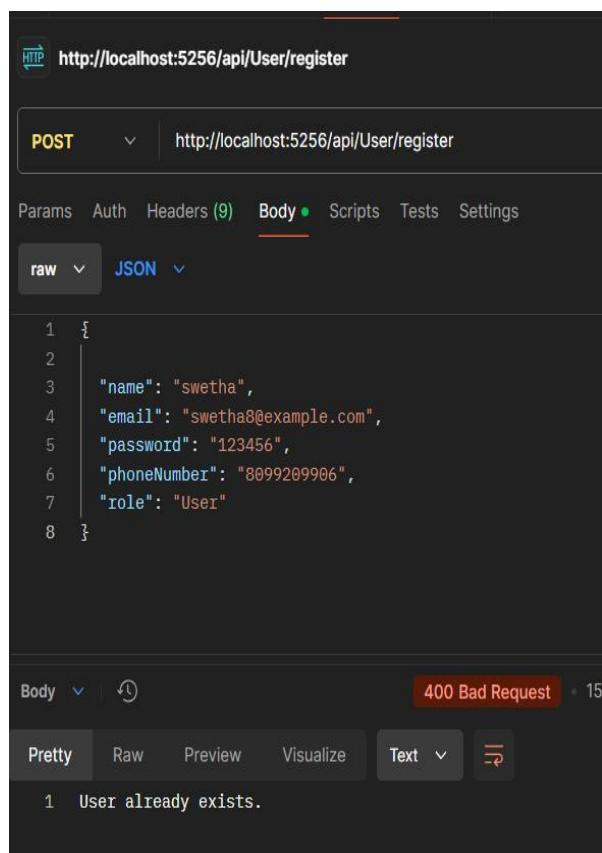
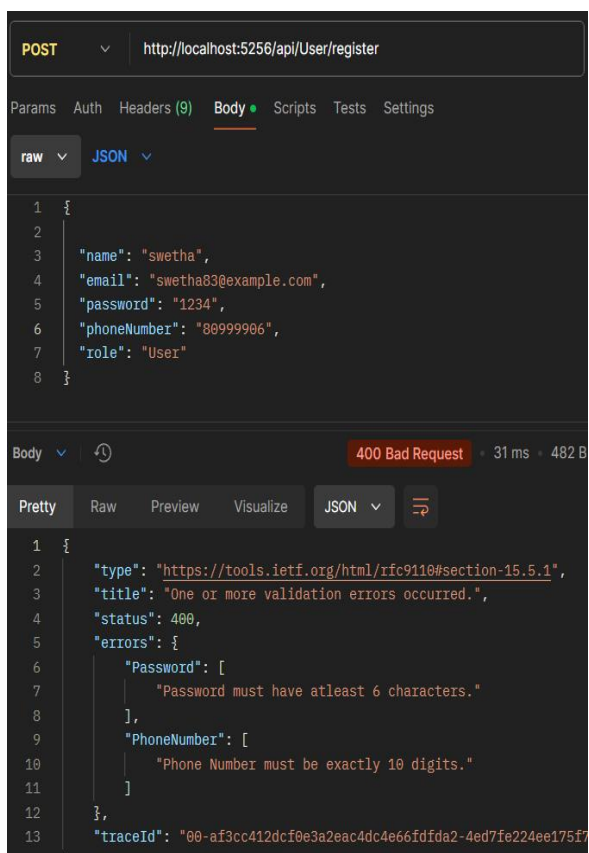
1. POST /api/User/register:

- Registers a new user in the system. It takes user details as input and creates user by storing in database. Outputs user created message and successful status code.
- Password is hashed by using bcrypt

Successful user Creation:



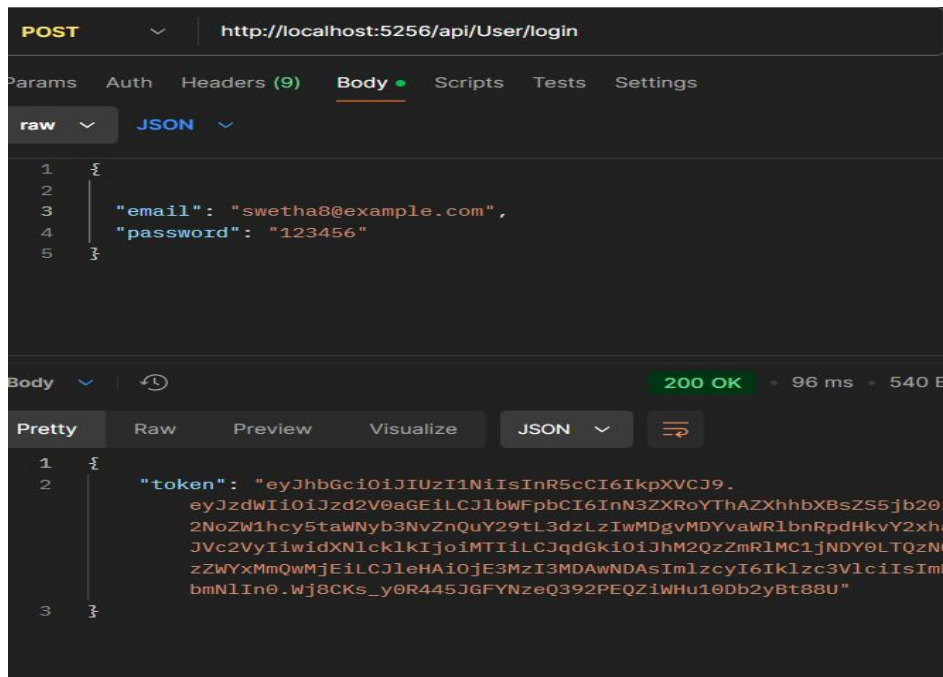
Invalid user



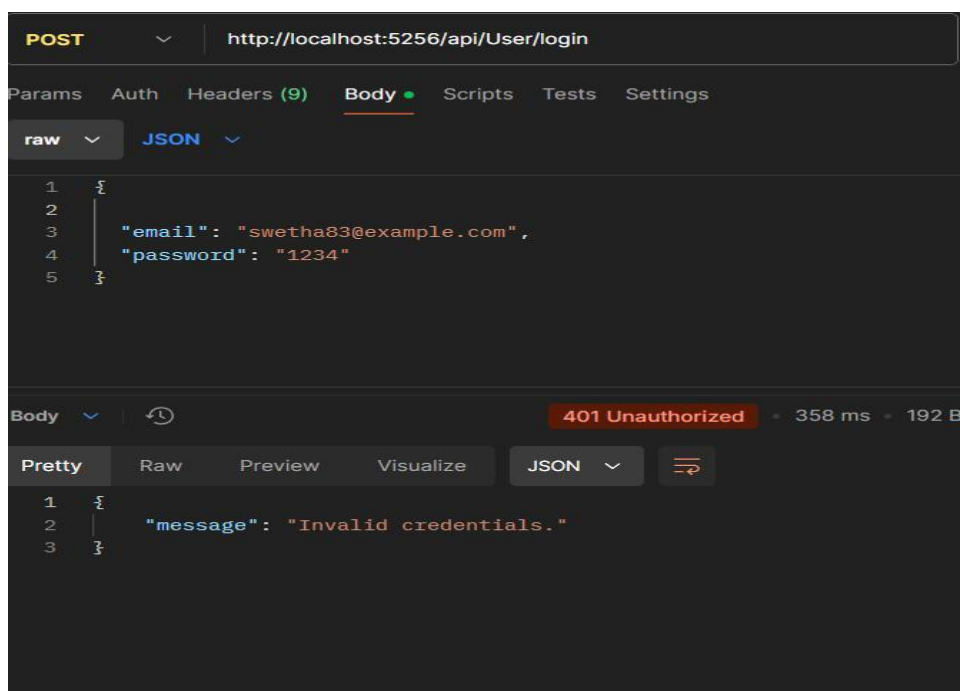
2.POST /api/User/login

- Authenticates a user and generates a JWT token.Takes Valid email and password as input.
- Provides Authentication success with a JWT token.

Valid Login:



Invalid Login:

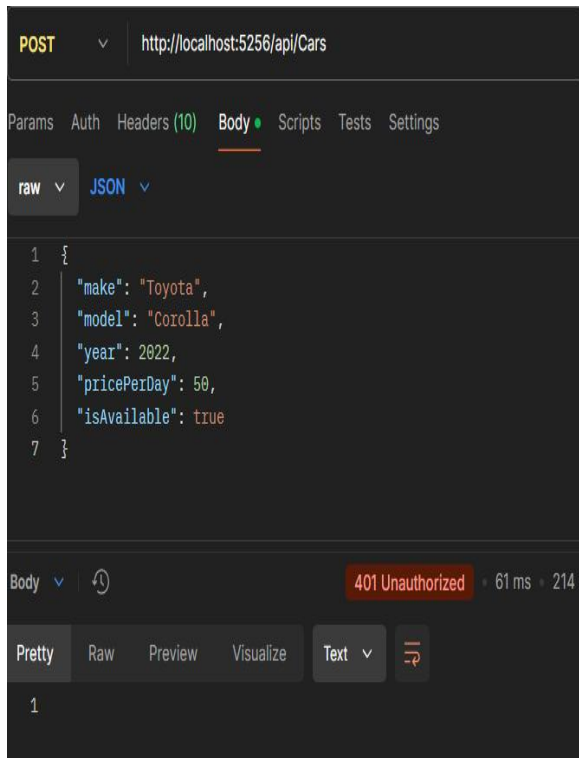


Car Management (Admin)

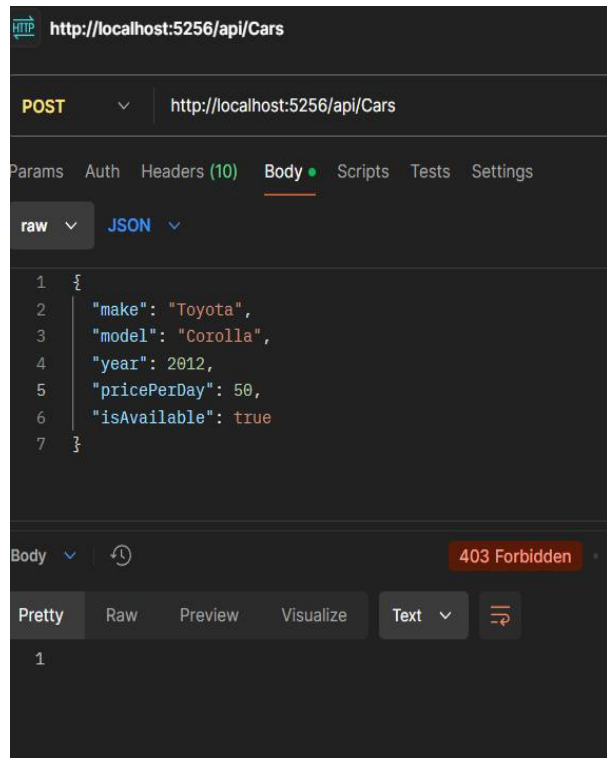
1. POST /api/Cars

- Adds a new car to the system (admin-only operation). Car details such as Make, Model, Year, and PricePerDay as input
- Gives Confirmation message with the added car details as output.

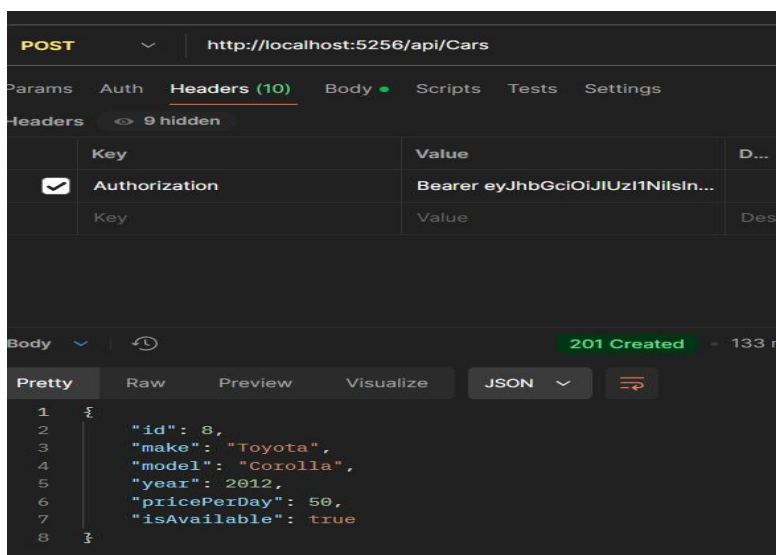
Unauthorized User:



Authorized but Role: User



Valid Creation:



2. PUT /api/Cars

- **Purpose:** Updates details of an existing car .Car id and fields to update.
- Updated car details gives as output.

Updated car details price from 50 to 80 and year to 2014.

A screenshot of a REST client interface. The top bar shows a PUT request to `http://localhost:5256/api/Cars`. Below the bar, tabs for Params, Auth, Headers (10), Body, Scripts, Tests, and Settings are visible. The 'Body' tab is selected, showing a JSON payload in raw format. The payload is a JSON object with the following fields: `"Id": 8`, `"make": "Toyota"`, `"model": "Corolla"`, `"year": 2014`, `"pricePerDay": 80`, and `"isAvailable": true`. Below the raw JSON, there are tabs for Pretty, Raw, Preview, and Visualize. The 'Pretty' tab is selected, showing the same JSON object formatted. At the bottom right, a green status bar indicates a `200 OK` response with a response time of 36 ms.

```
1 {
2   "Id": 8,
3   "make": "Toyota",
4   "model": "Corolla",
5   "year": 2014,
6   "pricePerDay": 80,
7   "isAvailable": true
8 }
```

200 OK • 36 ms

```
1 {
2   "id": 8,
3   "make": "Toyota",
4   "model": "Corolla",
5   "year": 2014,
6   "pricePerDay": 80,
7   "isAvailable": true
8 }
```

Car Id not found:

A screenshot of a REST client interface. The top bar shows a PUT request to `http://localhost:5256/api/Cars`. Below the bar, tabs for Params, Auth, Headers (10), Body, Scripts, Tests, and Settings are visible. The 'Body' tab is selected, showing a JSON payload in raw format. The payload is a JSON object with the following fields: `"Id": 10`, `"make": "Toyota"`, `"model": "Corolla"`, `"year": 2014`, `"pricePerDay": 80`, and `"isAvailable": true`. Below the raw JSON, there are tabs for Pretty, Raw, Preview, and Visualize. The 'Pretty' tab is selected, showing the same JSON object formatted. At the bottom right, a red status bar indicates a `404 Not Found` response with a response time of 63 ms. Below the status bar, the response body is shown in text format: `1 Car with Id 10 not found`.

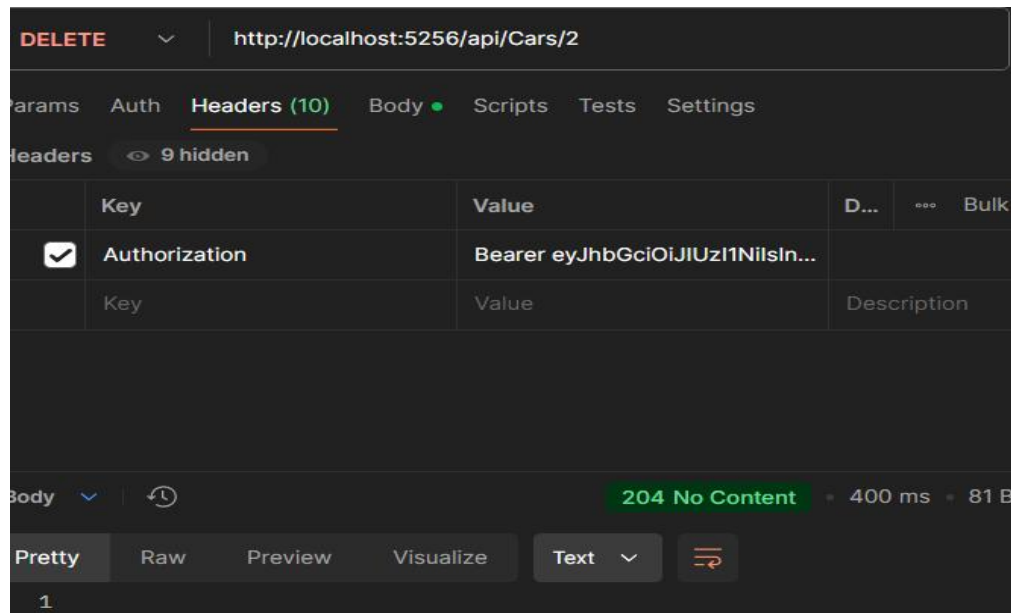
```
1 {
2   "Id": 10,
3   "make": "Toyota",
4   "model": "Corolla",
5   "year": 2014,
6   "pricePerDay": 80,
7   "isAvailable": true
8 }
```

404 Not Found • 63 ms

```
1 Car with Id 10 not found
```

3. DELETE /api/Cars/{id}

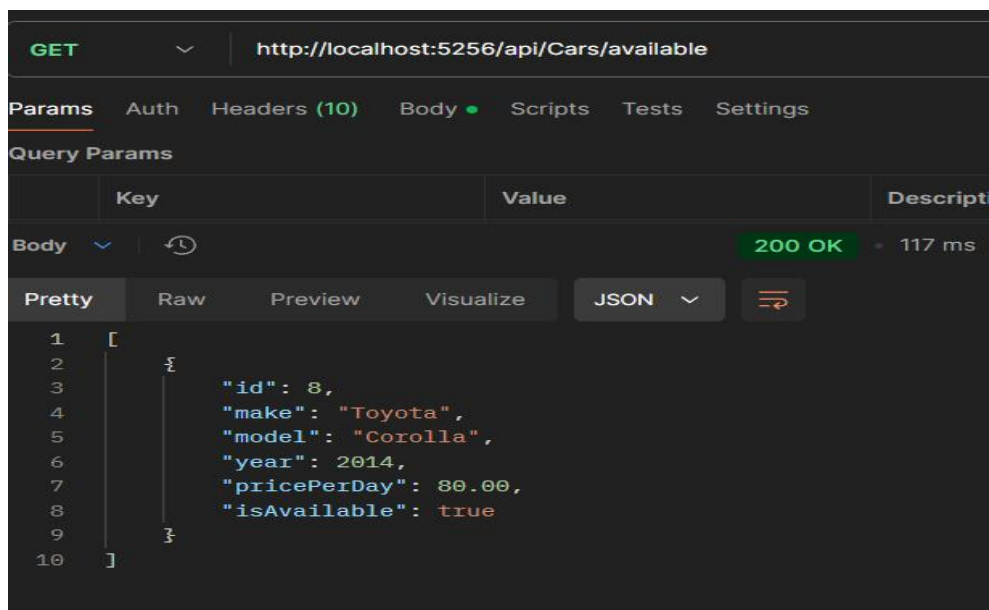
- Deletes a car record from the system (admin-only operation) by taking id as input.
- Expected Output: No Content upon successful deletion.



2. Car Management (allow Anonymous users)

1. GET /api/Cars/available

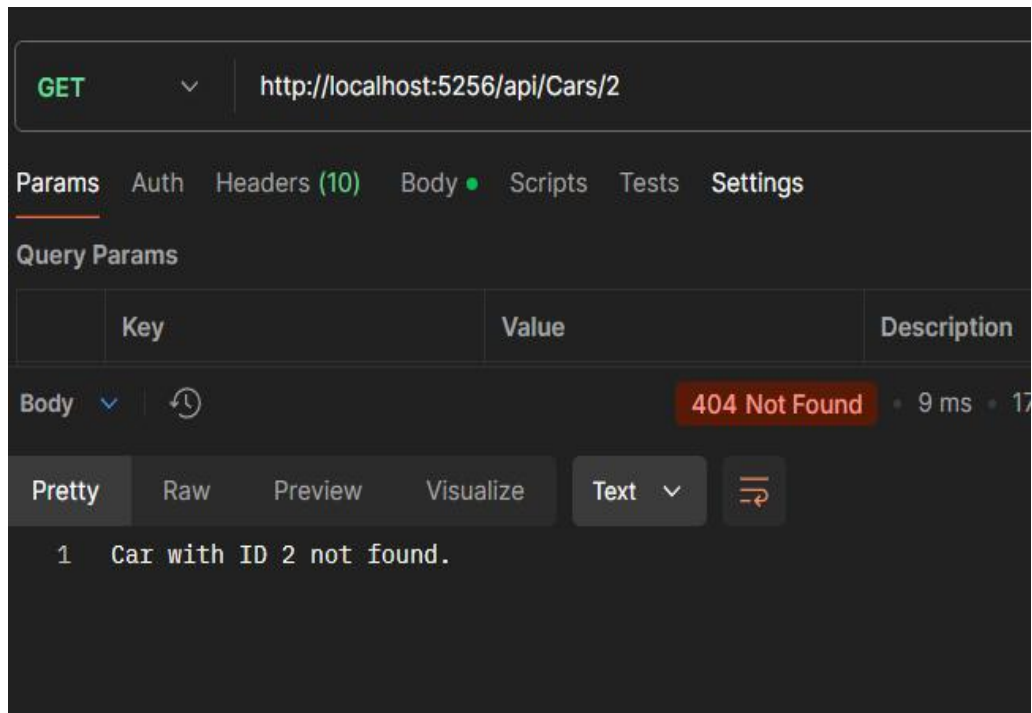
- Fetches a list of cars currently available for rent.
- Expected Output: Array of car objects marked as IsAvailable: true.



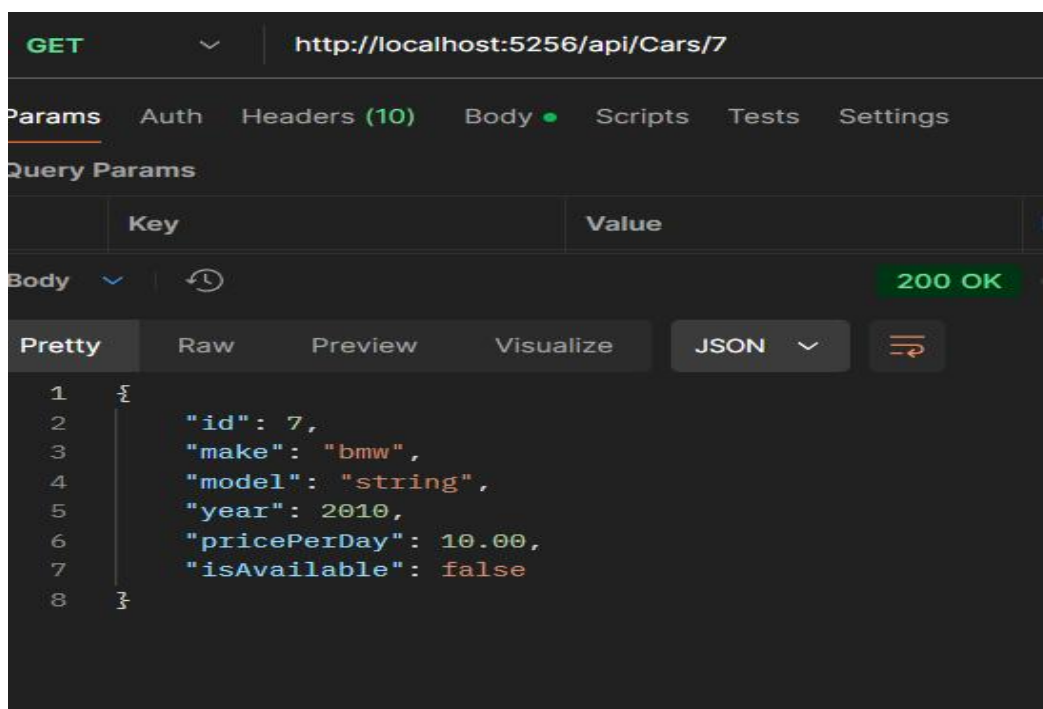
2. GET /api/Cars/{id}

- Retrieves detailed information for a specific car based on the car's id.
- Expected Output: Car details including make, model, year, price, and availability.

When carId is not present:



Valid id:

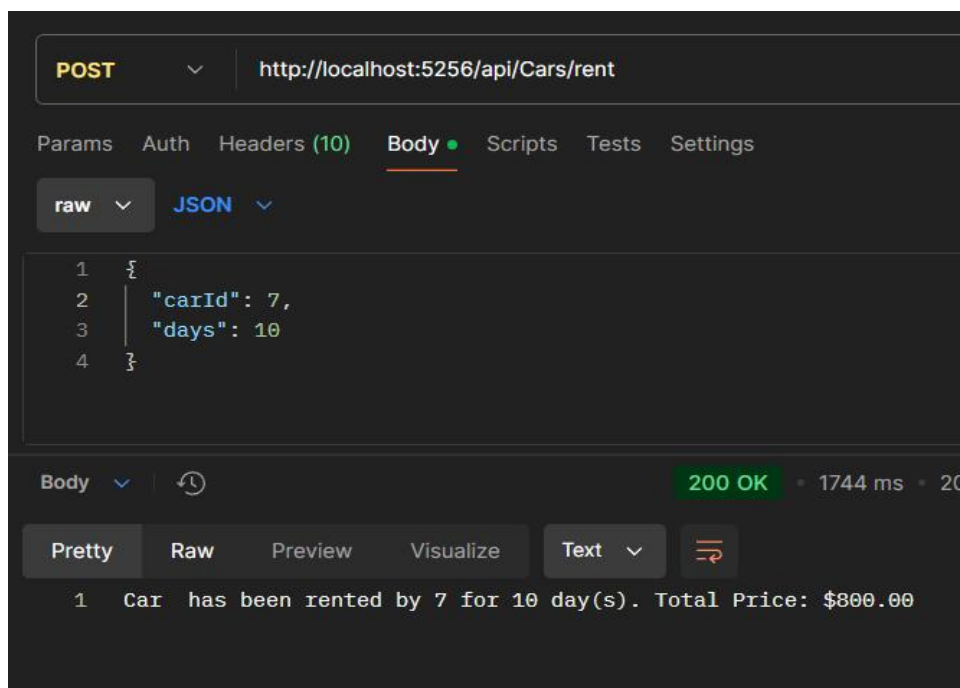


3. Car Rental (Only Authenticated Users)

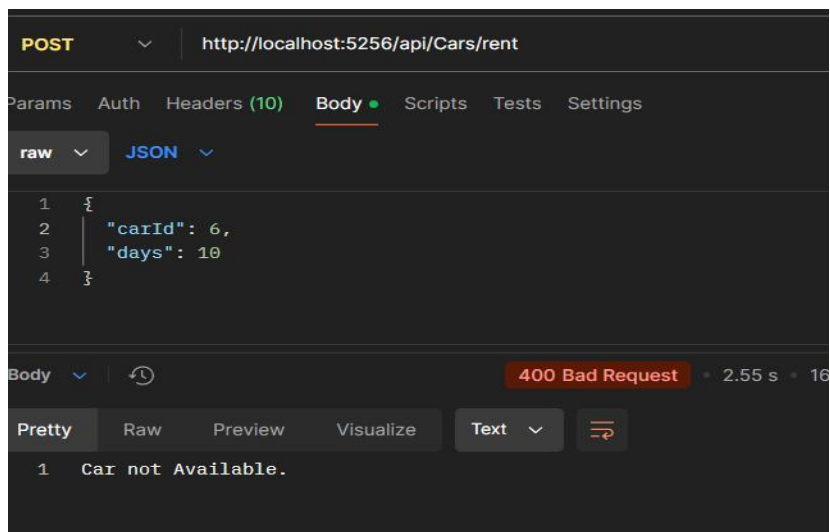
1. POST /api/Cars/rent

- Allows users to rent a car by providing required details like CarId, rental dates. User info is retrieved from the token.
- Expected Output: Rental confirmation message with calculated total cost.
- Sends SMS to user phoneNumber on successful booking and stores the info in transaction table.

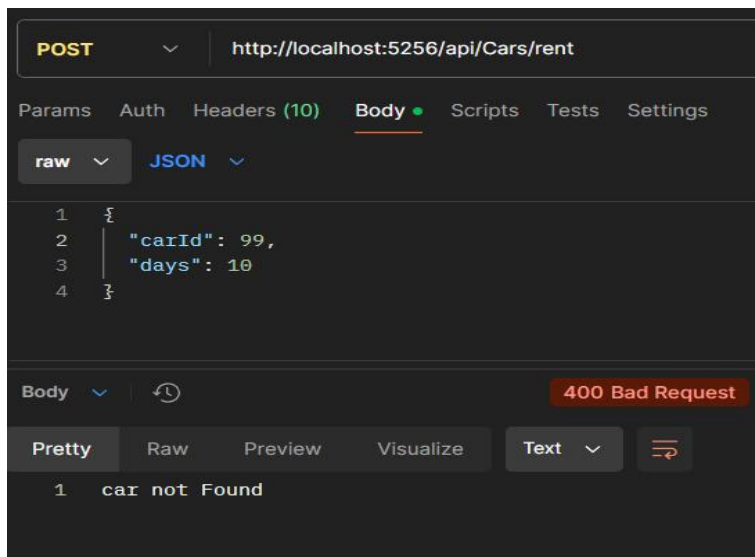
Case1: Success



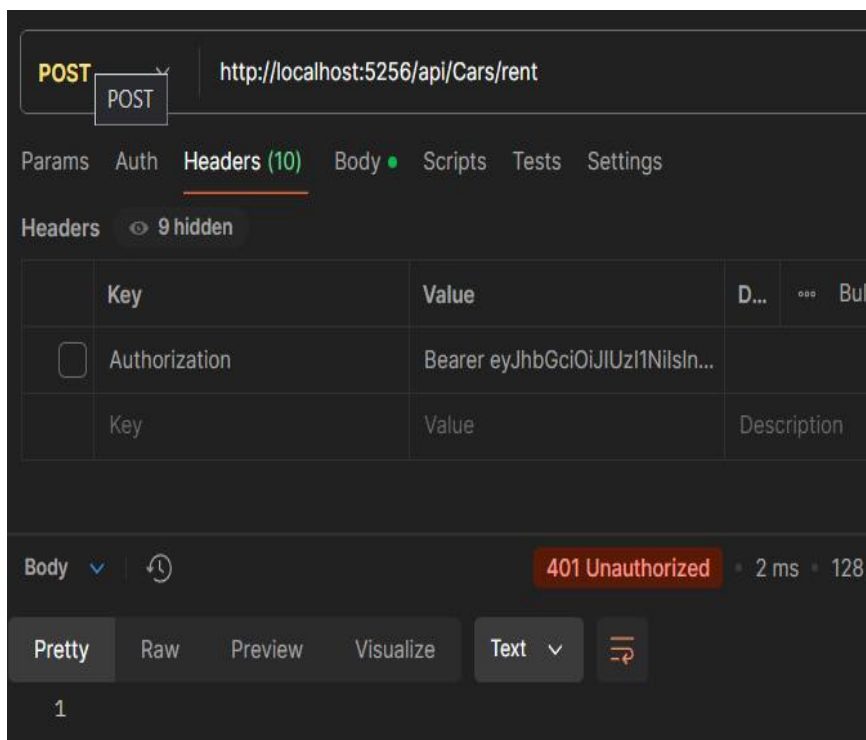
Case2: Requested Car not Available.



Case3: Requested CarId does not exist



4.Case4: Unauthorized Access:



2. GET /api/Cars/BookingHistory

- Retrieves the booking history for the authenticated user.
- Expected Output: List of all past and current bookings for the user.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:5256/api/Cars/BookingHistory
- Headers:** 10 headers are listed, with 9 hidden. The visible header is:

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiIs...	
- Status:** 200 OK, 51 ms, 537 B
- Body:** Pretty view of the JSON response.

```
1  [
2    {
3      "id": 6,
4      "userId": 7,
5      "carId": 7,
6      "rentDays": 10,
7      "totalAmount": 100.00,
8      "rentDate": "2024-11-26T23:23:25.5475334",
9      "user": null,
10     "car": null
11   },
12   {
13     "id": 12,
14     "userId": 7,
15     "carId": 7,
16     "rentDays": 10,
17     "totalAmount": 800.00,
18     "rentDate": "2024-11-27T16:02:37.9120975",
19     "user": null,
```