

# Day 3 – API Integration and Data Migration

## General Ecommerce

On Day 3, the main goal was to integrate the provided API, migrate data into Sanity CMS, and set up a functional frontend to display the car rental data. The day involved working through a few challenges, especially with fetching and rendering the data correctly, but by the end, everything came together smoothly. Below is a summary of the journey through the process.

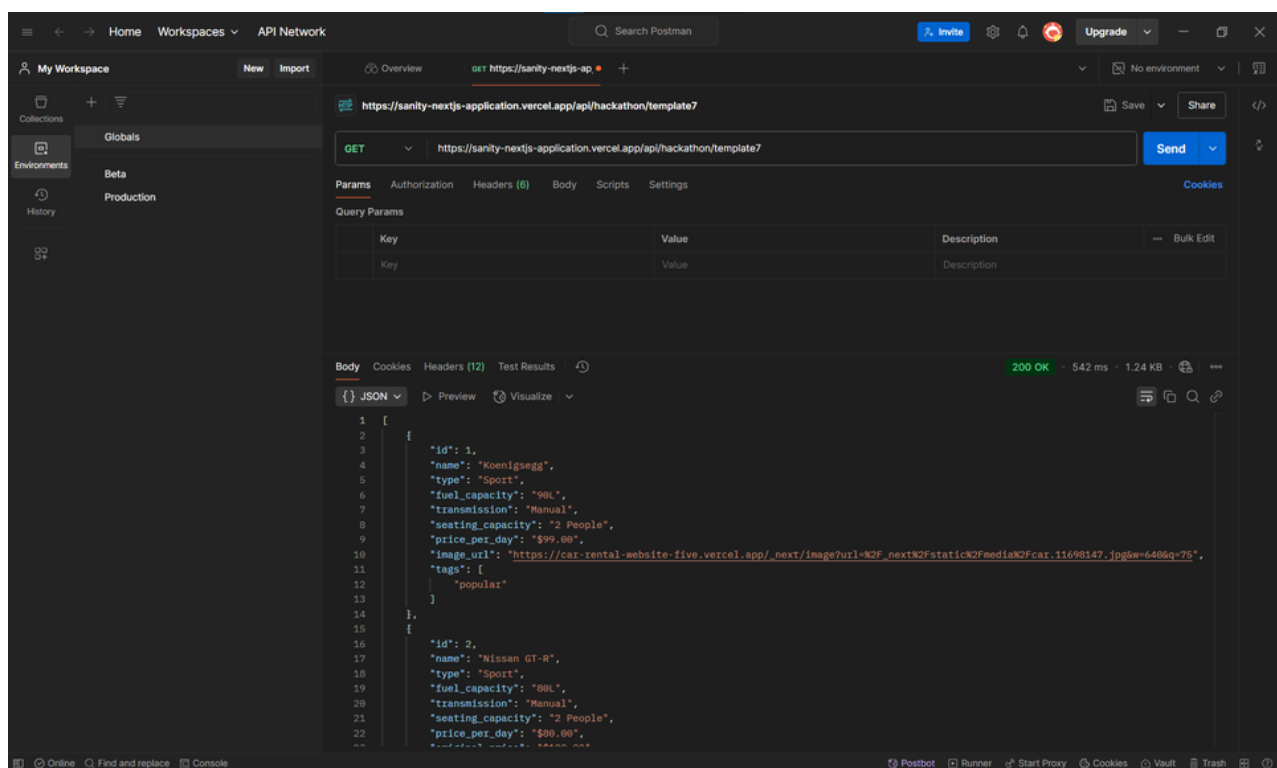
### 1. API Integration Process

For Day 3, I was assigned Template 7, which contained data for rental cars. I used the following API:

API URL: [Template 7 API](https://sanity-nextjs-application.vercel.app/api/hackathon/template7)

### 2. API Testing with Postman

I first tested the API using Postman to ensure that it was returning the correct data. The response was successful, confirming that the API was functioning properly.

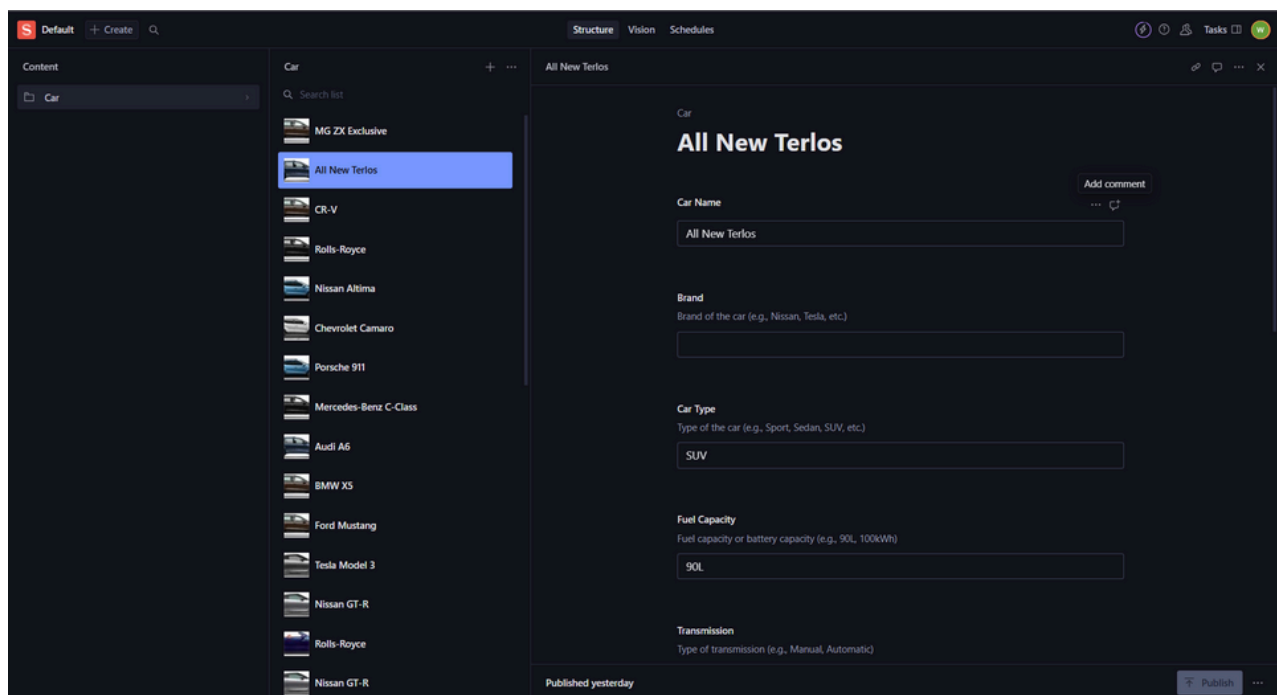


### 3. Data Migration to Sanity CMS

I proceeded with migrating the data from the provided API into Sanity CMS. To do this, I utilized the migration script provided in the template.

```
1 async function uploadImageToSanit (imageUrl) {
2   try {
3     console.log(`Uploading image: ${imageUrl}`);
4     const response = await axios.get(imageUrl, { responseType: 'arraybuffer'
5   });
6     const buffer = Buffer.from(response.data);
7     const asset = await client.assets.upload('image', buffer, {
8       filename: imageUrl.split('/').pop()
9     });
10    console.log(`Image uploaded successfully: ${asset._id}`);
11    return asset._id;
12  } catch (error) {
13    console.error('Failed to upload image', imageUrl, error);
14    return null;
15  }
```

The data was successfully migrated to Sanity CMS, including product listings, categories, and car rental details.

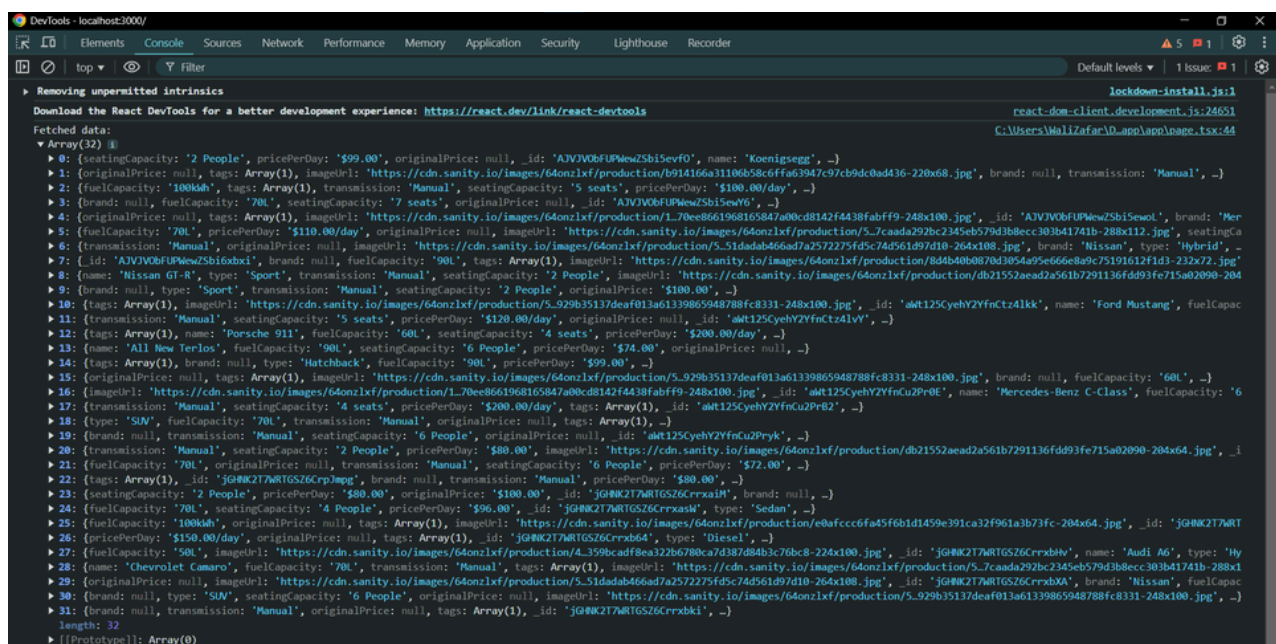


## 4. Fetching Data from Sanity CMS

Once the data was migrated, the next challenge was fetching it from Sanity CMS to display it in the Next.js frontend. I wrote queries to fetch the data, but initially encountered issues with missing fields and image rendering.

**Query Issues & Fixes:** I faced errors when attempting to fetch the data, especially with the images not loading. After investigating the issue, I discovered that the query was incorrect, which caused the problem. I fixed the query and got the images to display correctly.

```
1 import { groq } from "next-sanity";
2
3 export const allProducts = groq`
4   *[_type == "car"]
5   {
6     _id,
7     name,
8     brand,
9     type,
10    fuelCapacity,
11    transmission,
12    seatingCapacity,
13    pricePerDay,
14    originalPrice,
15    tags,
16    "imageUrl": image.asset->url
17 `;
```

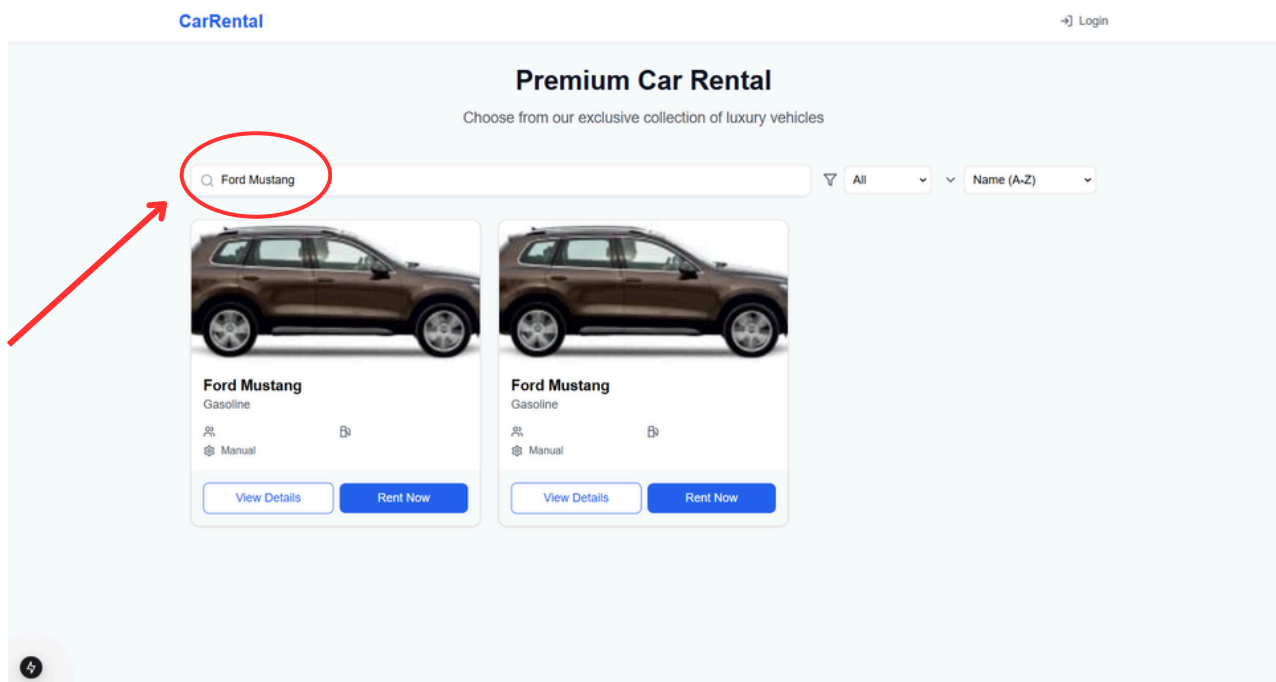


## 5. Custom UI and Features

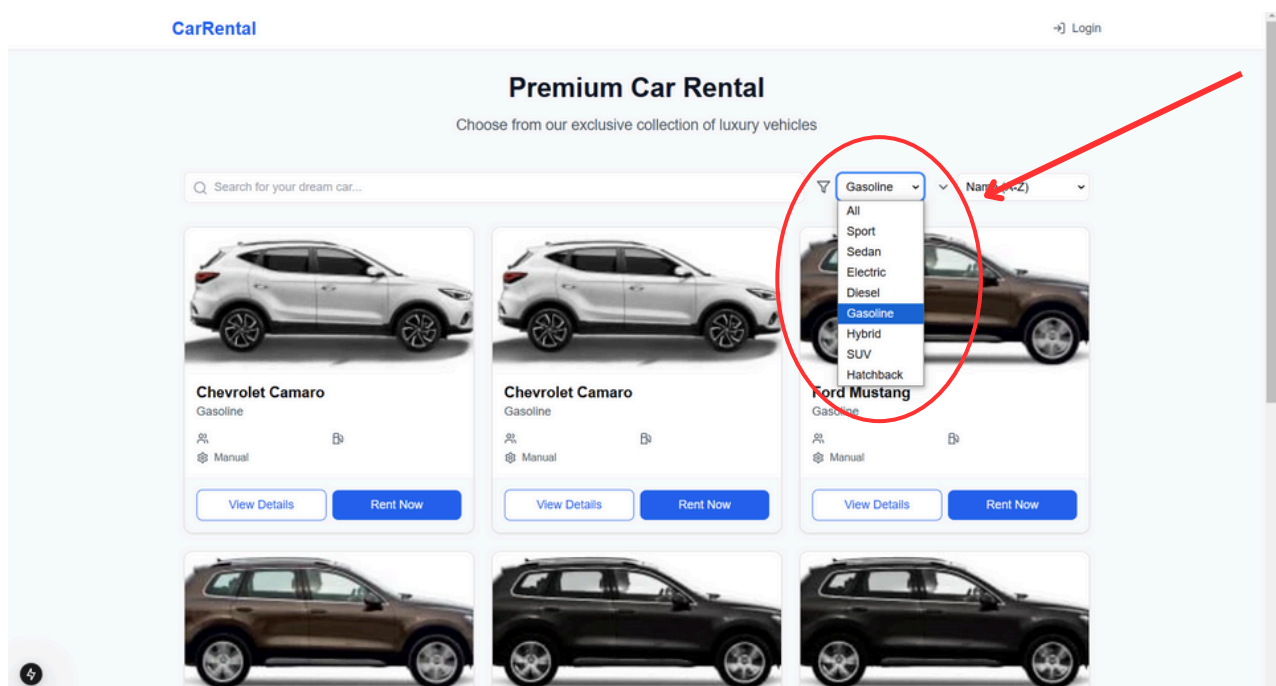
After successfully fetching the data, I moved on to building a custom UI to display the fetched car rental data.

### UI Features Implemented

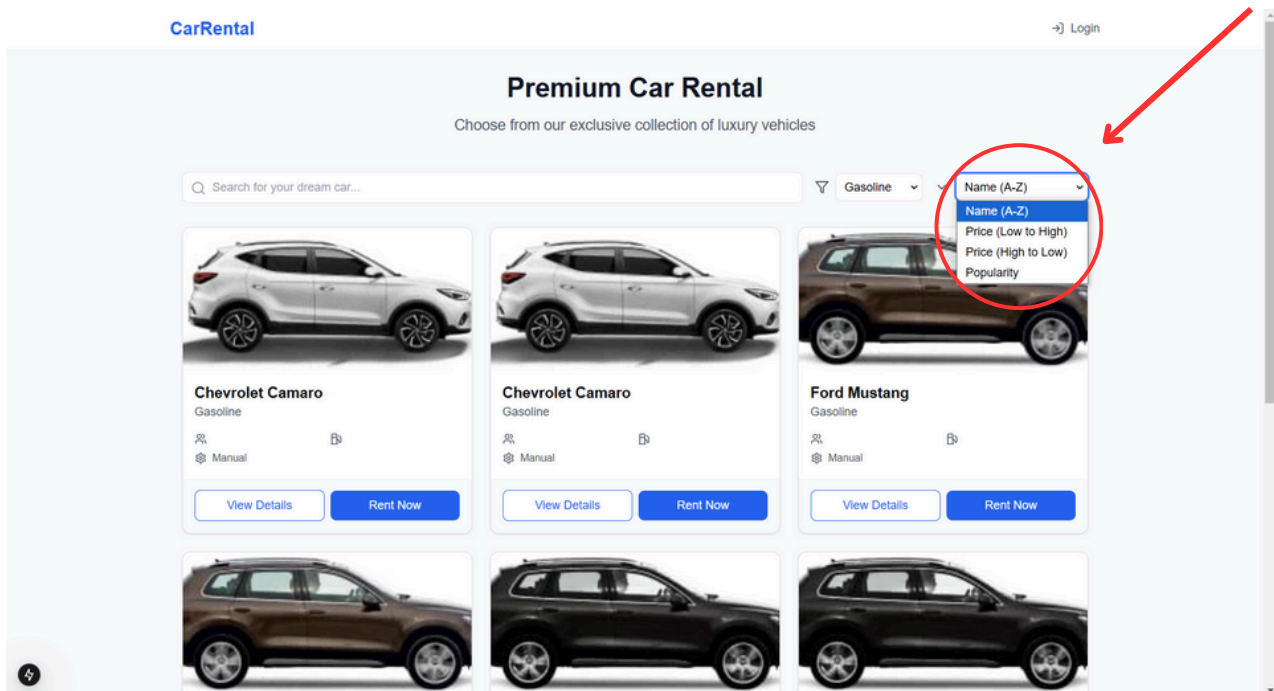
- **Search Functionality:** I implemented a search option to allow users to find cars by name.



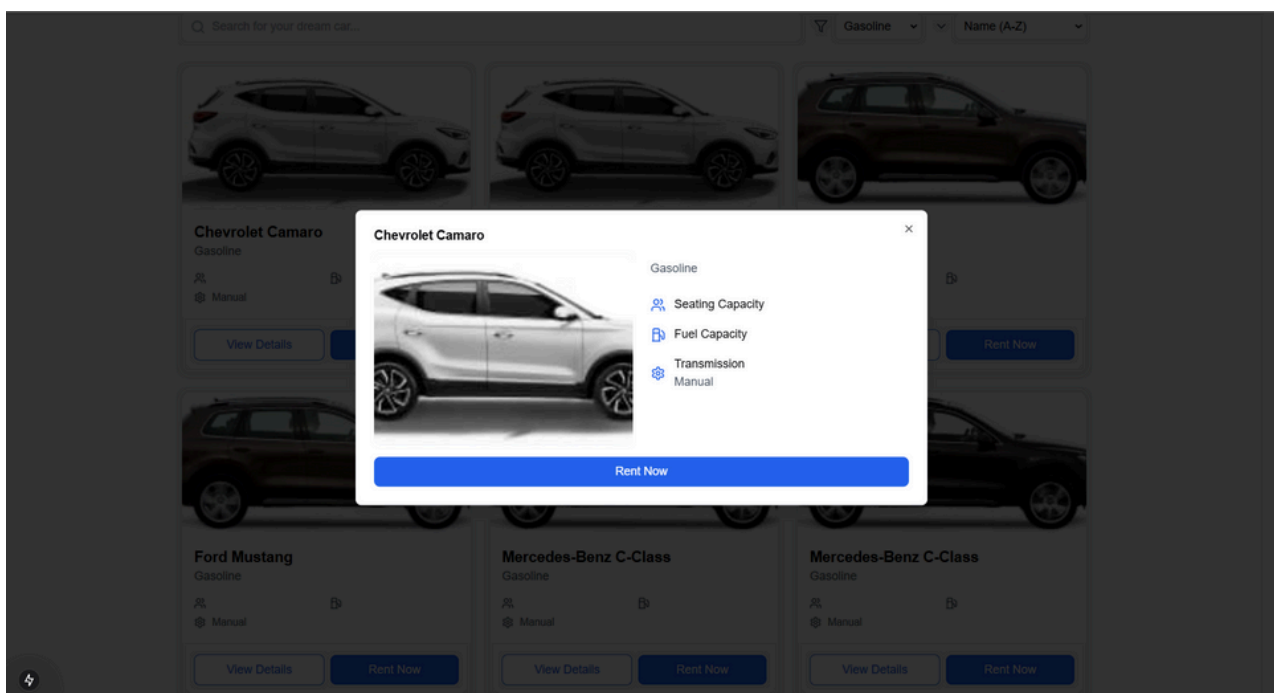
- **Filter Functionality:** I added a filter option to sort cars by type (e.g., hybrid, diesel, petrol).



- **Sort by Price:** I also included a sort option to sort the cars by their price.



- **View Details Modal:** I created a modal to display detailed information about each car when clicked on View Details button.



## 6. Conclusion

Day 3 was a journey of integrating the API, migrating data, and building a frontend to display the data. Despite the challenges with fetching and rendering the data, the process was successful, and the car rental platform now includes a fully functional search, filter, sort, and detail view feature.