Day 3 - API Integration and Data Migration General Ecommerece

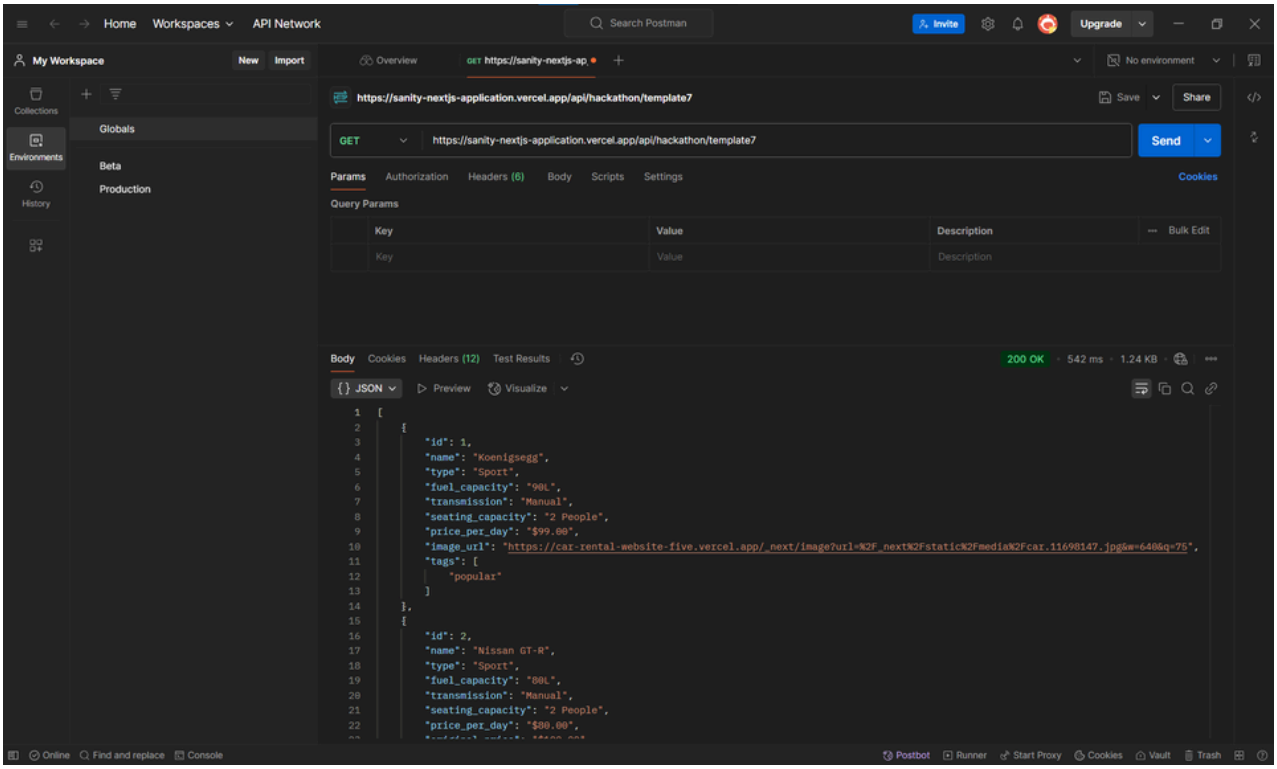
On Day 3, the main goal was to integrate the provided API, migrate data into Sanity CMS, and set up a functional frontend to display the car rental data. The day involved working through a few challenges, especially with fetching and rendering the data correctly, but by the end, everything came together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

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.

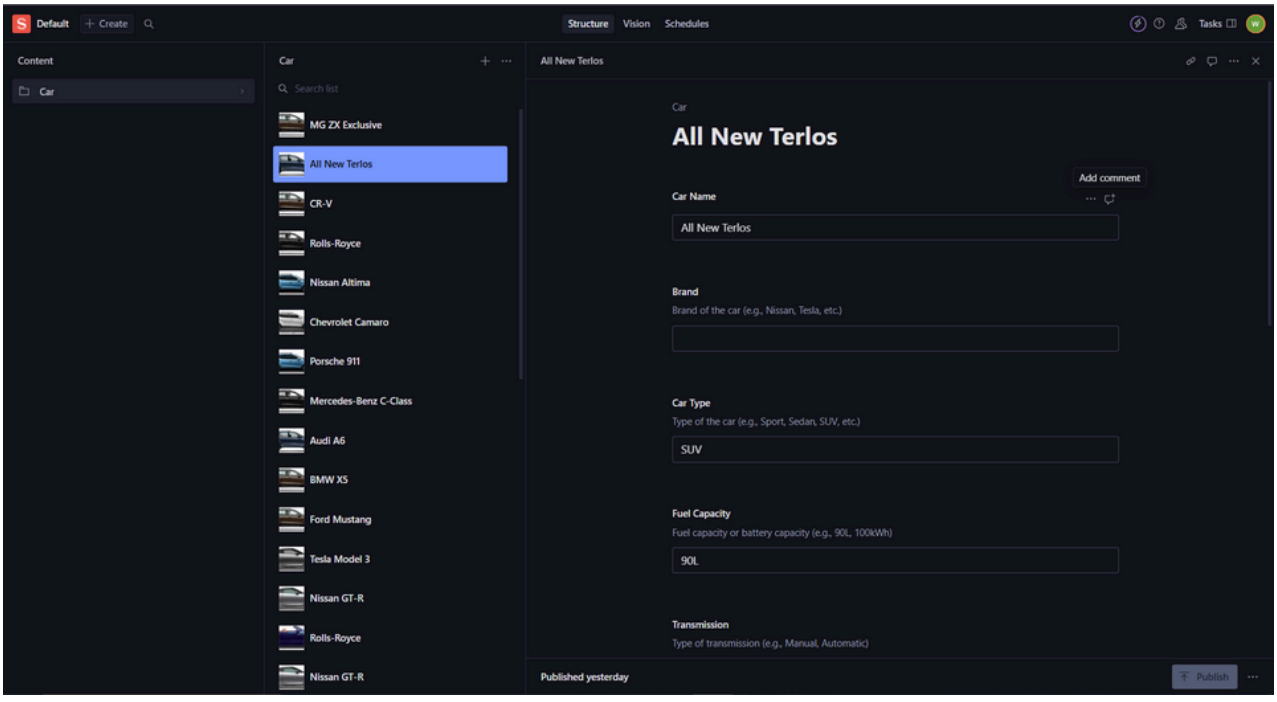


3. Data Migration to Sanity CMS

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.

```
async function uploadImageToSanit (imageUrl) {
  try {        y
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseTyp : 'arraybuffer'
  });                                              e
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop()
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload imag    , imageUrl, error);
    return null;  e:'
  }
}
```

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



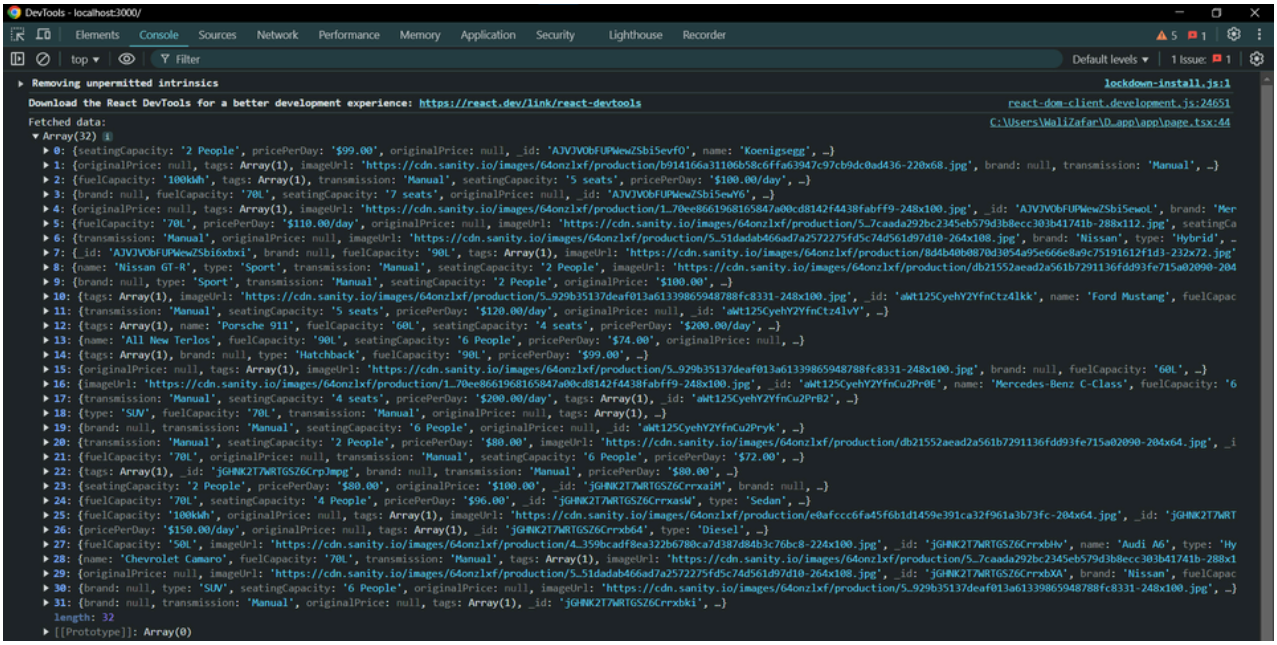
4. Fetching Data from Sanity CMS

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.

```
import { groq } from "next-sanity";

export const allProducts = groq`
   *[_type == "car"]
{    _id,
     name,
     brand,
     type,
     fuelCapacity,
     transmission,
     seatingCapacity,
     pricePerDay,
     originalPrice,
     tags,
     "imageUrl": image.asset->url
  }
`;
```

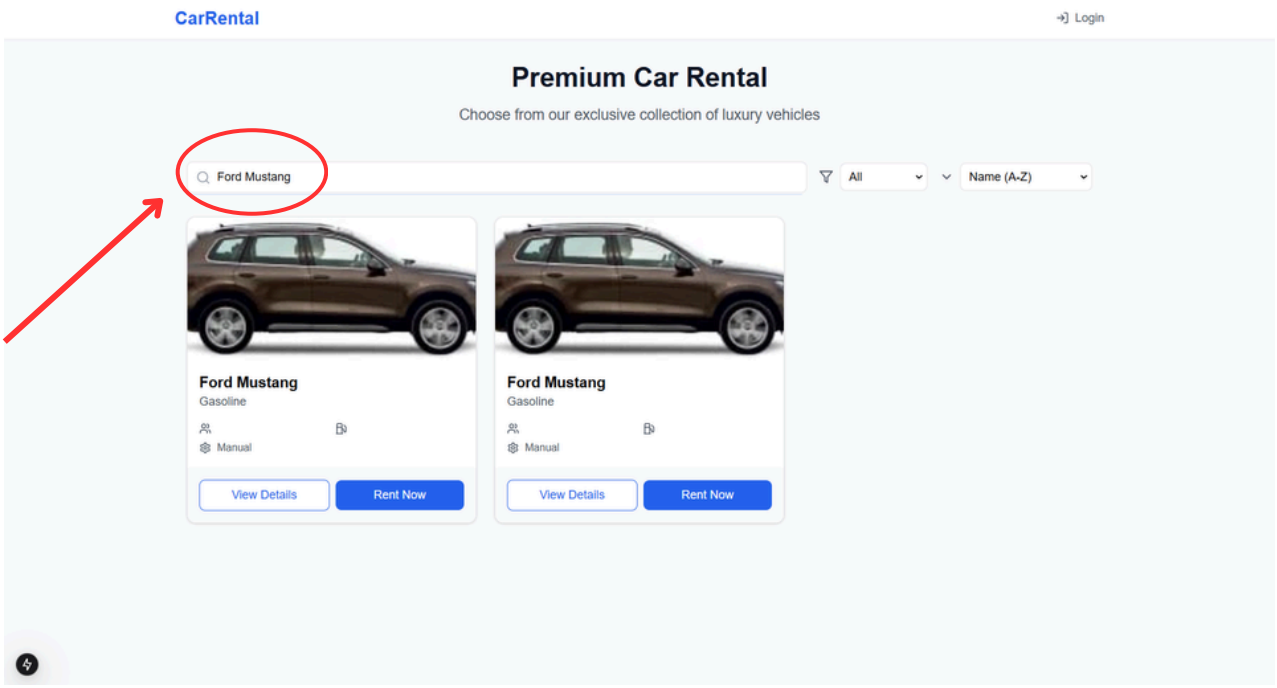


5. Custom UI and Features

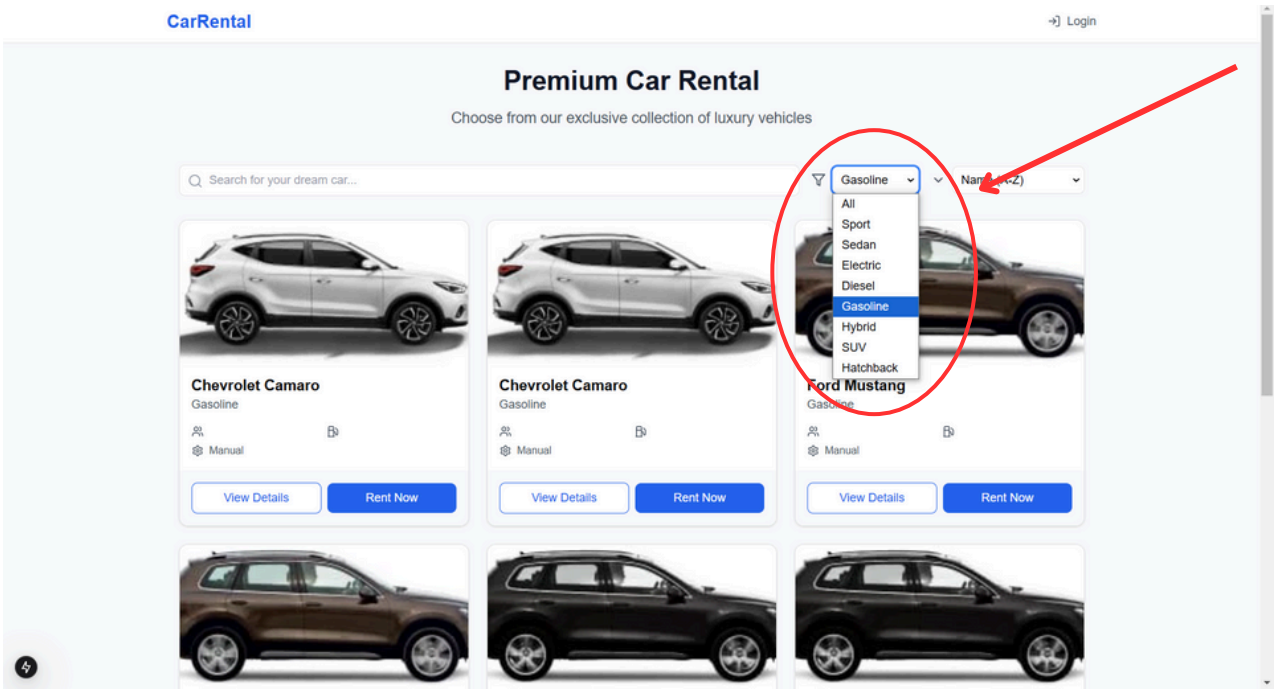
After successfully fetching the data, I moved on to building a custom UI to display the fetched car rental data.

UI Features Implemented

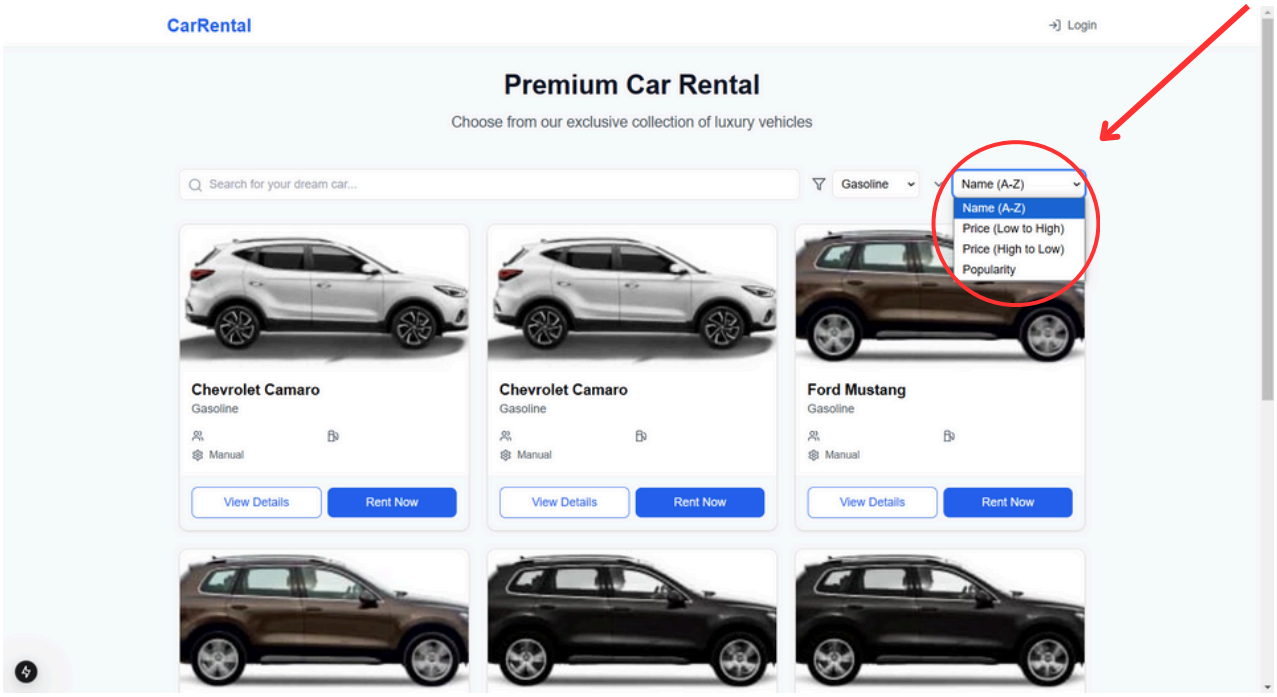
- **Search Functionality:** I implemented a search option to allow users to find cars by name.



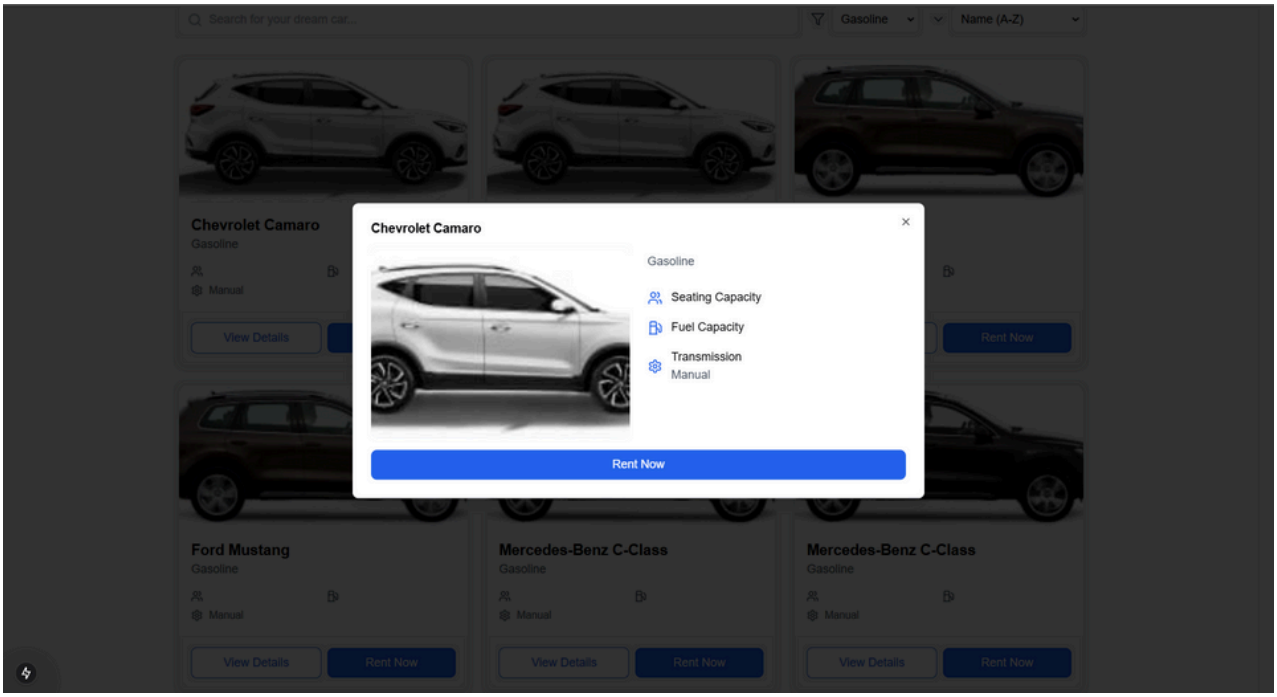
- **Filter Functionality:** I added a filter option to sort cars by type (e.g., hybrid, diesel, petrol).



- **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 cliked 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.