DAY 3 API INTEGRATION AND DATA MIGRATION

API Integration Process

I have integrated the API provided by the faculty while working on Template 7, according to my roll number. Below are the steps I followed to successfully integrate the API:

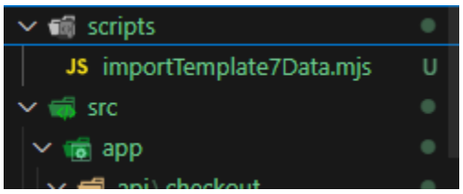
1. Understanding the API Documentation

Before starting the integration, I thoroughly reviewed the API documentation provided by the faculty. This helped me understand the structure, endpoints, and response formats required for the implementation.

2. Data Migration

I utilized the provided data migration file to transfer the API data to Sanity CMS. The images above demonstrate:

- The directory structure of my data migration file.



- Migration File insuring Data is process through secure processes.

```
// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
});
```

- The script added to my JSON file to ensure the migration was executed correctly.

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "Cars": "node scripts/importTemplate7Data.mjs"
},
```

3. Showcasing Data in Sanity Backend

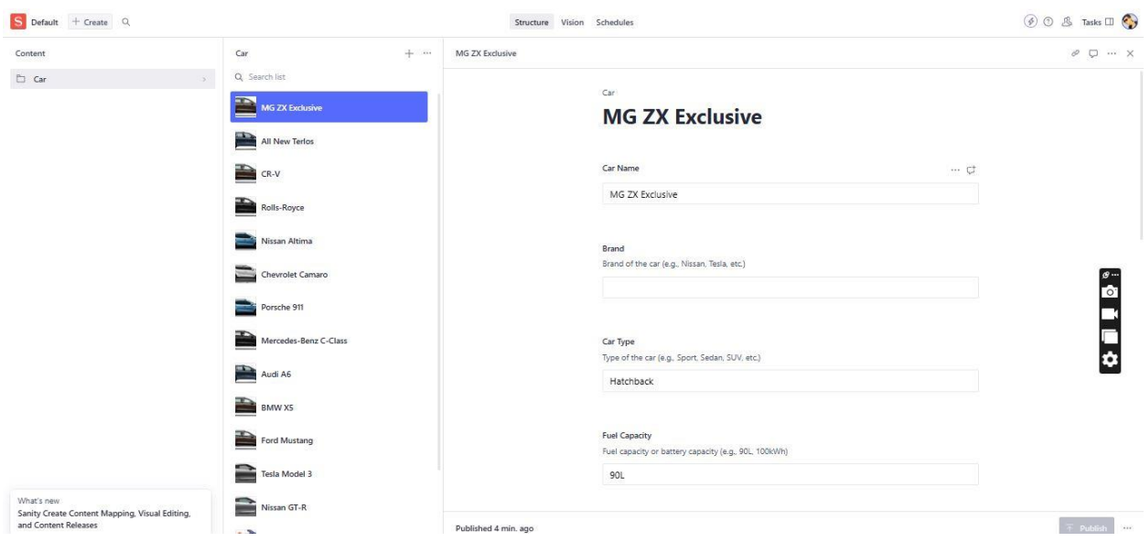
The images of my console confirm that the data migration was successful. Additionally, the Sanity Studio screenshots show that all data, including images and specifications, have been migrated properly.

Console Image

```
Fetching car data from API...
Fetched 16 cars
Processing car: Koenigsegg
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcar.11698147.jpg&w=640&q=
75
Image uploaded successfully: image-8d4b40b0870d3054a95e666e8a9c75191612f1d3-232x72-jpg
Uploading car to Sanity: Koenigsegg
Car uploaded successfully: IKYNzEDaTej6htqeupYtw3
Processing car: Nissan GT-R
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcar(1).cab606a9.jpg&w=640
&q=75
Image uploaded successfully: image-db21552aead2a561b7291136fdd93fe715a02090-204x64-jpg
Uploading car to Sanity: Nissan GT-R
Car uploaded successfully: kFvA8Slm5ZuJlJOXbVxZ2W
Processing car: Rolls-Royce
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(2).bd07489a.jpg&w=120
0&q=75
Image uploaded successfully: image-b914166a31106b58c6ffa63947c97cb9dc0ad436-220x68-jpg
Uploading car to Sanity: Rolls-Royce
Car uploaded successfully: kFvA8Slm5ZuJlJOXbVxZ9h
Processing car: Nissan GT-R
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcar(1).cab606a9.jpg&w=120
0&q=75
Image uploaded successfully: image-db21552aead2a561b7291136fdd93fe715a02090-204x64-jpg
Uploading car to Sanity: Nissan GT-R
Car uploaded successfully: IKYNzEDaTej6htqeupYuDl
Processing car: Tesla Model 3
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(13).37182fc4.jpg&w=12
00&q=75
```

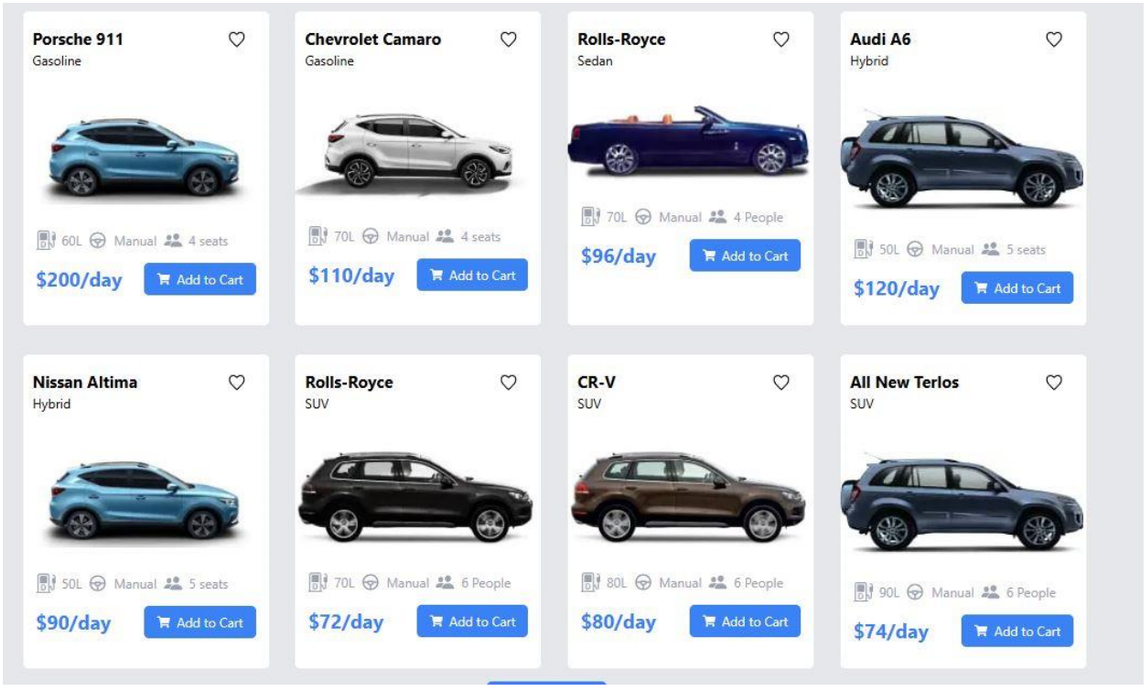
```
Car uploaded successfully: kFvA8Slm5ZuJlJOXbVxZzN
Processing car: Rolls-Royce
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(17).574834dc.jpg&w=12
00&q=75
Image uploaded successfully: image-11870ee8661968165847a00cd8142f4438fabff9-248x100-jpg
Uploading car to Sanity: Rolls-Royce
Car uploaded successfully: kFvA8Slm5ZuJlJOXbVxaUx
Processing car: CR-V
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(15).5f4e5799.jpg&w=12
00&q=75
Image uploaded successfully: image-50e929b35137deaf013a61339865948788fc8331-248x100-jpg
Uploading car to Sanity: CR-V
Car uploaded successfully: kFvA8Slm5ZuJlJOXbVxac8
Processing car: All New Terlos
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(16).fc285c8d.jpg&w=12
00&q=75
Image uploaded successfully: image-46d359bcadf8ea322b6780ca7d387d84b3c76bc8-224x100-jpg
Uploading car to Sanity: All New Terlos
Car uploaded successfully: kFvA8Slm5ZuJlJOXbVxaz6
Processing car: MG ZX Exclusive
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(15).5f4e5799.jpg&w=12
00&q=75
Image uploaded successfully: image-50e929b35137deaf013a61339865948788fc8331-248x100-jpg
Uploading car to Sanity: MG ZX Exclusive
Car uploaded successfully: B3Cpo73hdROFms1bDtaP5B
Data import completed successfully!
```

Sanity Studio



4. Product Listing on My Frontend

Another image showcases the successful fetching of products from the API and their correct display on my frontend application.



5. GROQ Query Implementation

To retrieve and manage data efficiently, I implemented GROQ queries:

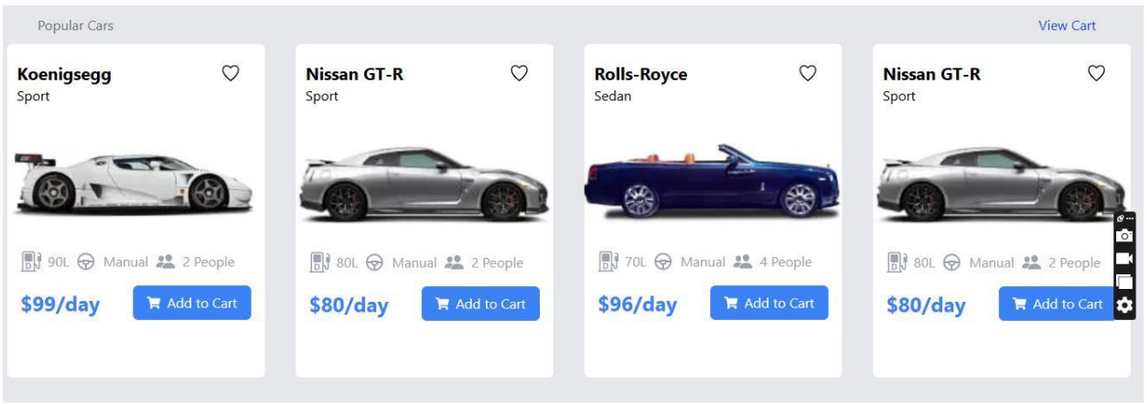
- A general GROQ query fetches data correctly, as shown in the image.

- A specific query retrieves products by index, allowing me to display only four products in the 'Popular' section while ensuring the integrity of the original 16-product dataset.

```
const fetchCars = async () => {
  try {
    const query = `*[_type == "car"][0...15]{
      name,
      pricePerDay,
      type,
      fuelCapacity,
      transmission,
      seatingCapacity,
      "id": _id,
      "imageUrl": image.asset->url,
    }`;
```

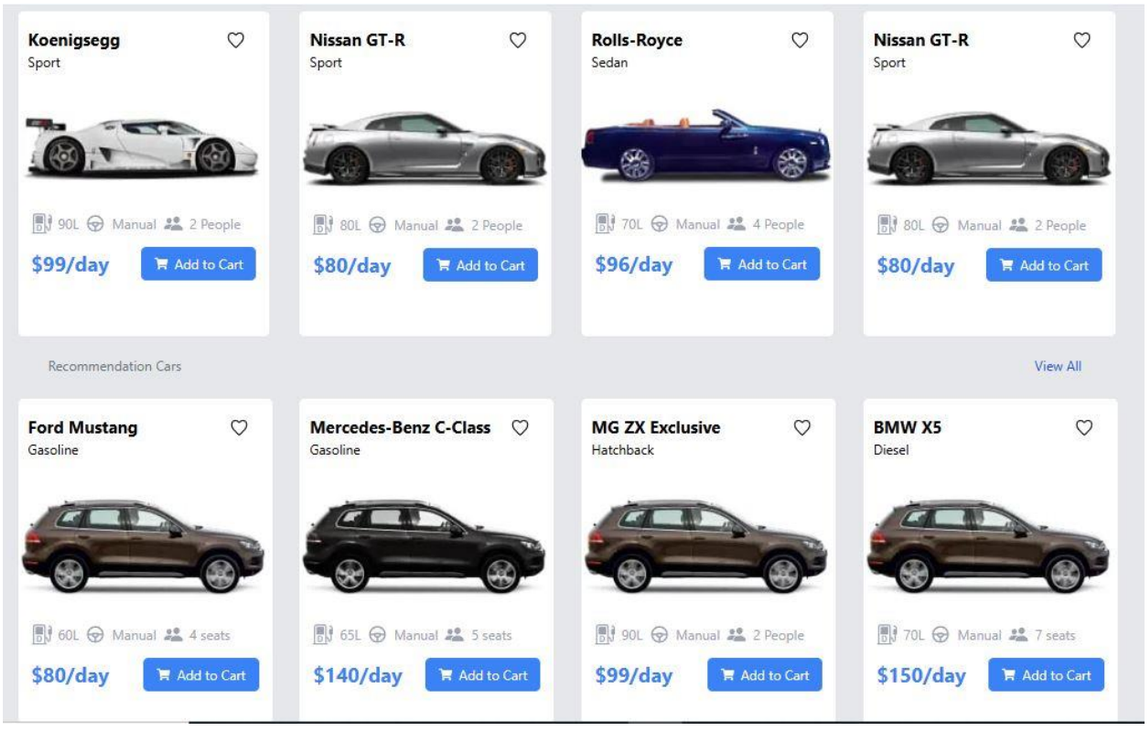
```
const data = await sanity.fetch(query);

const specificIndices = [4, 5, 10, 9];
const filteredProducts = specificIndices
  .map((index) => data[index])
  .filter(Boolean);
```

Result



6. Recommended Section Implementation

Similar to the 'Popular' section, I implemented the 'Recommended' section based on the UI guidelines provided.



Additionally, I used `parseFloat` to convert item prices from strings to numbers because the price was given in string type in Schema, ensuring accurate calculations of the total amount. The related query is demonstrated in the provided image.

```
const processedData = filteredProducts.map((item: any) => ({
  ...item,
  pricePerDay:
    typeof item.pricePerDay === "string"
      ? parseFloat(item.pricePerDay.replace(/[^\d.]/g, ""))
      : item.pricePerDay,
}));
```

Through these steps, I have successfully integrated and displayed the API data on my frontend while maintaining proper data management in Sanity CMS.