## **Day 3 - API Integration Report – MORENT**

## **API Migration Process**

#### **Step 1: API Endpoint Setup**

- API Endpoint: <a href="https://sanity-nextjs-application.vercel.app/api/hackathon/template7">https://sanity-nextjs-application.vercel.app/api/hackathon/template7</a>
- **Data Structure**: Provides hackathon template data including fields like name, type, seatingCapacity, fuelCapacity, pricePerDay, originalPrice, and image.

#### **Step 2: Data Fetching**

const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7');
const cars = response.data;

## **Adjustments in Schema**

Following changes are made in the schema according to the API provided:

- 1. Added a new id field of type number.
- 2. Introduced a new brand field for specifying the car brand.
- 3. Replaced capacity with seatingCapacity to describe the number of seats.
- 4. Changed fuelCapacity and price fields from number to string with descriptions.
- 5. Renamed price to pricePerDay for clarity.
- 6. Added a tags field for categorization using an array of strings.
- 7. Enabled hotspot option in the image field for better image handling.

```
export default {
  name: 'car',
type: 'document',
title: 'Car',
  fields: [
       name: 'id',
       type: 'number',
       title: 'Id',
     },
       name: 'name',
type: 'string',
       title: 'Car Name',
       name: 'brand',
type: 'string',
       title: 'Brand',
       description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
     },
       name: 'type',
type: 'string',
title: 'Car Type',
       description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
     },
       name: 'fuelCapacity',
       type: 'string',
       title: 'Fuel Capacity',
       description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
       name: 'transmission',
       type: 'string',
       title: 'Transmission',
       description: 'Type of transmission (e.g., Manual, Automatic)',
       name: 'seatingCapacity',
type: 'string',
title: 'Seating Capacity',
       description: 'Number of seats (e.g., 2 People, 4 seats)',
       name: 'pricePerDay',
       type: 'string',
title: 'Price Per Day',
       description: 'Rental price per day',
     },
       name: 'originalPrice',
       type: 'string',
       title: 'Original Price',
       description: 'Original price before discount (if applicable)',
       name: 'tags',
type: 'array'
title: 'Tags'
       title: 'Tags',
of: [{ type: 'string' }],
       options: {
         layout: 'tags',
       },
       description: 'Tags for categorization (e.g., popular, recommended)',
     },
       name: 'image',
type: 'image',
title: 'Car Image',
       options: {
         hotspot: true
```

## **Migration Steps and Tools Used**

### Migration Steps:

#### 1. Environment Setup:

- Added required packages (@sanity/client, axios, dotenv).
- Set up a .env.local file to handle configuration variables.

#### 2. Fetching Data:

- Collected data from the API endpoint using Axios or Fetch.
- Parsed and logged the retrieved data to confirm its structure.

#### 3. Creating Schema:

 Generated Sanity documents by integrating API data with uploaded image references.

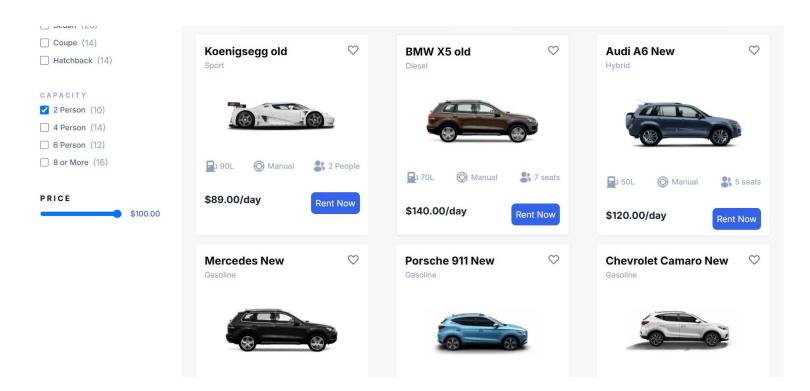
#### **Tools Used:**

- 1. @sanity/client: To interact with Sanity's API.
- 2. **Axios or Fetch API**: For making HTTP requests to fetch data from endpoints.
- 3. **dotenv**: To manage environment variables securely.
- 4. Sanity Asset Manager: For uploading and managing images.

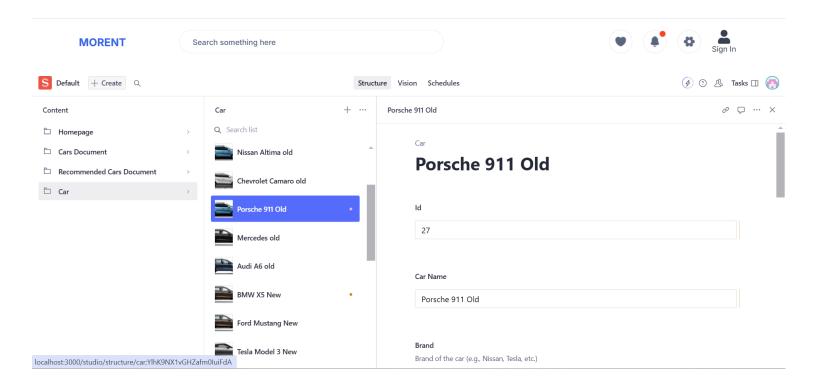
## **API Call:**

```
// Create a builder instance
   const builder = imageUrlBuilder(client);
   // Function to generate image URL from Sanity
   const urlFor = (source: any) => builder.image(source).url();
   const CategoriesMain = () => {
     const router = useRouter(); // Initialize useRouter
      const [cars, setCars] = useState<any[]>([]); // State to store car data
      useEffect(() => {
      const fetchCars = async () => {
            const query = `*[_type == "car"]{
              name,
              type,
              "image": image.asset->url,
              fuelCapacity,
              transmission,
              seatingCapacity,
              pricePerDay,
             originalPrice,
              tags
            const result = await client.fetch(query); // Fetch the data from Sanity
            const carsWithImageUrls = result.map((car: any) => ({
              imageUrl: urlFor(car.image), // Directly assign the URL
            }));
            setCars(carsWithImageUrls); // Update the state with the fetched data and image URLs
          } catch (error) {
            console.error("Error fetching car data:", error);
        fetchCars();
      }, []);
```

# Data Successfully Displayed in Frontend:



## **Populated Sanity Schema Fields:**



## **Code Snippet For API Migration**

```
import axios from 'axios';
import dotenv from 'dotenv';
import { fileUntToPath } from 'url';
import path from 'path';
// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../
                                                                                                                '../.env.local') });
// 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'
 async function uploadImageToSanity(imageUrl) {
        cry {
   console.log(`Uploading image: ${imageUrl}`);
   const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
   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 image:', imageUrl, error);
async function importData() {
     try {
   console.log('Fetching car data from API...');
         // API endpoint containing car data
const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7');
const cars = response.data;
          console.log(`Fetched ${cars.length} cars`);
          for (const car of cars) {
  console.log(`Processing car: ${car.name}`);
              let imagekef = nuil;
if (car.image_url) {
  imageRef = await uploadImageToSanity(car.image_url);
              const sanityCar = {
    _type: 'car',
    name: car.name,
    brand: car.brand || null,
    type: car.type,
    fuelCapacity: car.fuel_capacity,
    transmission: car.transmission,
    seatingCapacity: car.seating_capacity,
    price paper day.
                  seatingCapacity: car.seating_capacity,
pricePerDay: car.price_per_day,
originalPrice: car.original_price || null,
tags: car.tags || [],
image: imageRef ? {
    _type: 'image',
    asset: {
    _type: 'reference',
    _ref: imageRef,
              console.log('Uploading car to Sanity:', sanityCar.name);
const result = await client.create(sanityCar);
console.log('Car uploaded successfully: ${result._id}');
          catch (error) {
console.error('Error importing data:', error);
 importData();
```