

Day 3 - API Integration Report – MORENT

API Migration Process

Step 1: API Endpoint Setup

- **API Endpoint:** <https://sanity-nextjs-application.vercel.app/api/hackathon/template7>
- **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.

```
1  export default {
2    name: 'car',
3    type: 'document',
4    title: 'Car',
5    fields: [
6      {
7        name: 'id',
8        type: 'number',
9        title: 'Id',
10     },
11     {
12       name: 'name',
13       type: 'string',
14       title: 'Car Name',
15     },
16     {
17       name: 'brand',
18       type: 'string',
19       title: 'Brand',
20       description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
21     },
22     {
23       name: 'type',
24       type: 'string',
25       title: 'Car Type',
26       description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
27     },
28     {
29       name: 'fuelCapacity',
30       type: 'string',
31       title: 'Fuel Capacity',
32       description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
33     },
34     {
35       name: 'transmission',
36       type: 'string',
37       title: 'Transmission',
38       description: 'Type of transmission (e.g., Manual, Automatic)',
39     },
40     {
41       name: 'seatingCapacity',
42       type: 'string',
43       title: 'Seating Capacity',
44       description: 'Number of seats (e.g., 2 People, 4 seats)',
45     },
46     {
47       name: 'pricePerDay',
48       type: 'string',
49       title: 'Price Per Day',
50       description: 'Rental price per day',
51     },
52     {
53       name: 'originalPrice',
54       type: 'string',
55       title: 'Original Price',
56       description: 'Original price before discount (if applicable)',
57     },
58     {
59       name: 'tags',
60       type: 'array',
61       title: 'Tags',
62       of: [{ type: 'string' }],
63       options: {
64         layout: 'tags',
65       },
66       description: 'Tags for categorization (e.g., popular, recommended)',
67     },
68     {
69       name: 'image',
70       type: 'image',
71       title: 'Car Image',
72       options: {
73         hotspot: true
74       }
75     }
76   ]
77 }
```

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:

```
1
2 // Create a builder instance
3 const builder = imageUrlBuilder(client);
4
5 // Function to generate image URL from Sanity
6 const urlFor = (source: any) => builder.image(source).url();
7
8 const CategoriesMain = () => {
9   const router = useRouter(); // Initialize useRouter
10  const [cars, setCars] = useState<any[]>([]); // State to store car data
11
12  // Fetch car data from Sanity
13  useEffect(() => {
14    const fetchCars = async () => {
15      try {
16        // Sanity GROQ query to get car data
17        const query = `*[_type == "car"]{
18          name,
19          type,
20          "image": image.asset->url,
21          fuelCapacity,
22          transmission,
23          seatingCapacity,
24          pricePerDay,
25          originalPrice,
26          tags
27        }`;
28
29        const result = await client.fetch(query); // Fetch the data from Sanity
30        const carsWithImageUrls = result.map((car: any) => ({
31          ...car,
32          imageUrl: urlFor(car.image), // Directly assign the URL
33        }));
34        setCars(carsWithImageUrls); // Update the state with the fetched data and image URLs
35      } catch (error) {
36        console.error("Error fetching car data:", error);
37      }
38    };
39
40    fetchCars();
41  }, []);
```

Data Successfully Displayed in Frontend:

- ☐ Sedan (20)
- ☐ Coupe (14)
- ☐ Hatchback (14)

CAPACITY

- ☒ 2 Person (10)
- ☐ 4 Person (14)
- ☐ 6 Person (12)
- ☐ 8 or More (16)

PRICE

\$100.00

Koenigsegg old



Sport



90L



Manual



2 People

\$89.00/day

Rent Now

BMW X5 old



Diesel



70L



Manual



7 seats

\$140.00/day

Rent Now

Audi A6 New



Hybrid



50L



Manual



5 seats

\$120.00/day

Rent Now

Mercedes New



Gasoline



Porsche 911 New



Gasoline



Chevrolet Camaro New



Gasoline



Populated Sanity Schema Fields:

MORENT

Search something here

S

Default

+ Create

Q

StructureVisionSchedules

⚡

⌚

👤

Tasks

🔍

Sign In

Content

Homepage

Cars Document

Recommended Cars Document

Car

Car

Search list

Nissan Altima old

Chevrolet Camaro old

Porsche 911 Old

Mercedes old

Audi A6 old

BMW X5 New

Ford Mustang New

Tesla Model 3 New

Porsche 911 Old

Id

27

Car Name

Porsche 911 Old

Brand

Brand of the car (e.g., Nissan, Tesla, etc.)

localhost:3000/studio/structure/car;YlhK9NX1vGHZafm0luIfdA

Code Snippet For API Migration

```
1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 });
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log(`Uploading image: ${imageUrl}`);
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop()
28     });
29     console.log(`Image uploaded successfully: ${asset._id}`);
30     return asset._id;
31   } catch (error) {
32     console.error('Failed to upload image:', imageUrl, error);
33     return null;
34   }
35 }
36
37 async function importData() {
38   try {
39     console.log('Fetching car data from API...');
40
41     // API endpoint containing car data
42     const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7');
43     const cars = response.data;
44
45     console.log(`Fetched ${cars.length} cars`);
46
47     for (const car of cars) {
48       console.log(`Processing car: ${car.name}`);
49
50       let imageRef = null;
51       if (car.image_url) {
52         imageRef = await uploadImageToSanity(car.image_url);
53       }
54
55       const sanityCar = {
56         _type: 'car',
57         name: car.name,
58         brand: car.brand || null,
59         type: car.type,
60         fuelCapacity: car.fuel_capacity,
61         transmission: car.transmission,
62         seatingCapacity: car.seating_capacity,
63         pricePerDay: car.price_per_day,
64         originalPrice: car.original_price || null,
65         tags: car.tags || [],
66         image: imageRef ? {
67           _type: 'image',
68           asset: {
69             _type: 'reference',
70             _ref: imageRef,
71           },
72         } : undefined,
73       };
74
75       console.log(`Uploading car to Sanity:`, sanityCar.name);
76       const result = await client.create(sanityCar);
77       console.log(`Car uploaded successfully: ${result._id}`);
78     }
79
80     console.log('Data import completed successfully!');
81   } catch (error) {
82     console.error('Error importing data:', error);
83   }
84 }
85
86 importData();
```

