Hackathon Day 3: Api Integration And Data Migration

Step 01: Understand the Provided APIs

Below are the key points detailing the car data provided by the API, which include essential information about each vehicle available for rental on the platform. This data will be used to display key car features, manage pricing, and improve the user experience by providing relevant details for making informed rental decisions.

Car Data Structure:

- **Car Name**: The name of the car (e.g., "Tesla Model S").
- **Brand**: the Company that makes the car (e.g., Nissan, Tesla, etc.).
- **Car Type**: The category of the car, such as Sedan, SUV, or Sports car.
- **Fuel Capacity**: How much fuel or battery the car can hold (e.g., 90L for fuel or 100kWh for electric vehicles).
- **Transmission**: The type of transmission (e.g., Manual or Automatic).
- **Seating Capacity**: How many people the car can seat (e.g., 2 seats, 4 seats).
- **Price Per Day**: The cost to rent the car for one day.
- **Original Price**: The price of the car before any discounts.
- **Tags**: Labels that categorize the car (e.g., popular, recommended).
- **Car Image**: A picture of the car that shows how it looks.

Step 02: Validate and Adjust Your Schema

Previous Schema Structure (Day-02):

```
// schemas/car.tsx

export default {
  name: 'car',
  title: 'Car',
  type: 'document',
  fields: [
   {
     name: 'id',
     title: 'ID',
     type: 'number',
   },
  {
     name: 'name',
     title: 'Car Name',
     type: 'string',
   },
   {
     name: 'price',
     title: 'Price',
     type: 'number',
   },
   {
     name: 'model',
     title: 'Model',
     type: 'string',
   },
```

```
 {
    name: 'category',
    title: 'Category',
    type: 'string',
  },
  {
    name: 'stock',
    title: 'Stock',
    type: 'number',
  },
  {
    name: 'tags',
    title: 'Tags',
    type: 'array',
    of: [{ type: 'string' }],
  },
  {
    name: 'image',
    title: 'Car Image',
    type: 'image',
  },
 ],
};
```

Modified Schema Structure:

```
// sanity/schemaTypes/cars.ts

export default {
  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
    {
      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',
     of: [{ type: 'string' }],
     options: {
       layout: 'tags',
     },
     description: 'Tags for categorization
     (e.g., popular,
     recommended)',
    },
    {
     name: 'image',
     type: 'image',
     title: 'Car Image',
     options: {
     hotspot: true
     }
    }
  ],
};
```

Step 03: Data Migration Options

The **migration process** is all about moving data from one place to another while making sure it fits perfectly into the new system. For **Sanity**, this means taking data (like car names, brands, prices, etc.) and adding it into Sanity's database, following the structure you've already set up in your schemas.

If you're using APIs, the process becomes automated. You write scripts to fetch data from the API, adjust it to match your schema (if needed), and then insert it into Sanity.

Key Steps in Simple Terms:

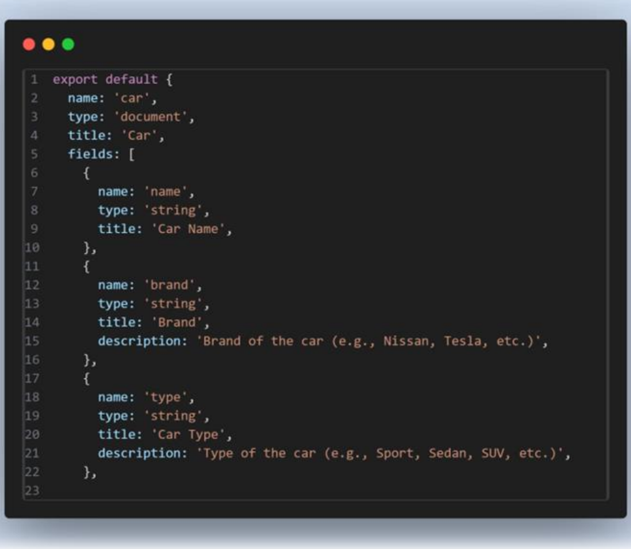
1. **Data Preparation**: Ensure the data fetched from APIs (e.g., car name, brand, type, price, etc.) aligns with Sanity schema fields.

2. **API Integration**: Write scripts to retrieve data from APIs and insert it into Sanity using Sanity's client or dataset import tools.

3. **Importing Data**: Use API scripts or tools to seamlessly push the fetched data into Sanity's dataset.

4. **Validation and Testing**: Verify the imported data for accuracy, schema alignment, and functionality using queries or tools like GROQ and Sanity Studio.

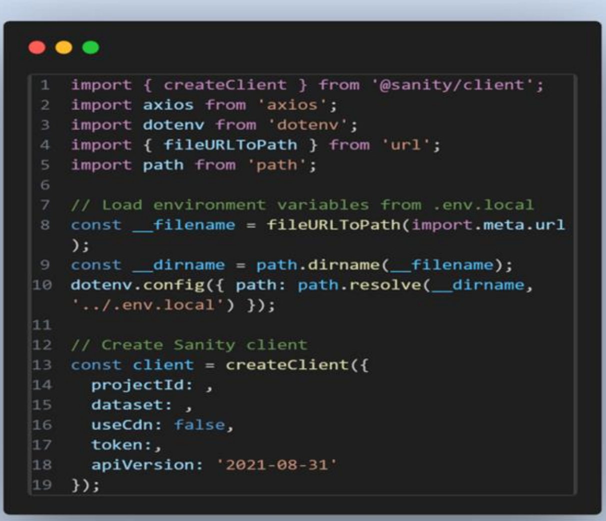
// Sanity Schema.



```
export default {
  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
    {
      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.)',
    },

```

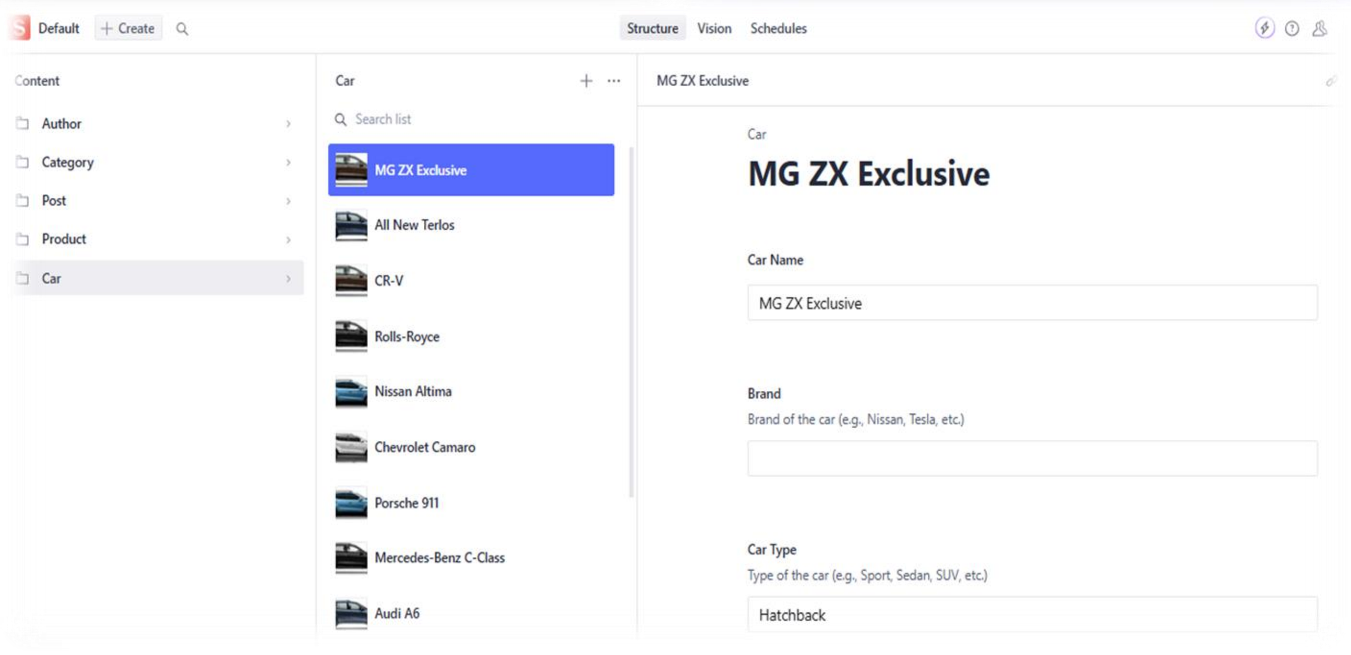
// Script.mjs



```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } 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: ,
  dataset: ,
  useCdn: false,
  token:,
  apiVersion: '2021-08-31'
});
```



MG ZX Exclusive

Car Name

MG ZX Exclusive

Brand

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

Car Type

Type of the car (e.g., Sport, Sedan, SUV, etc.)

Hatchback

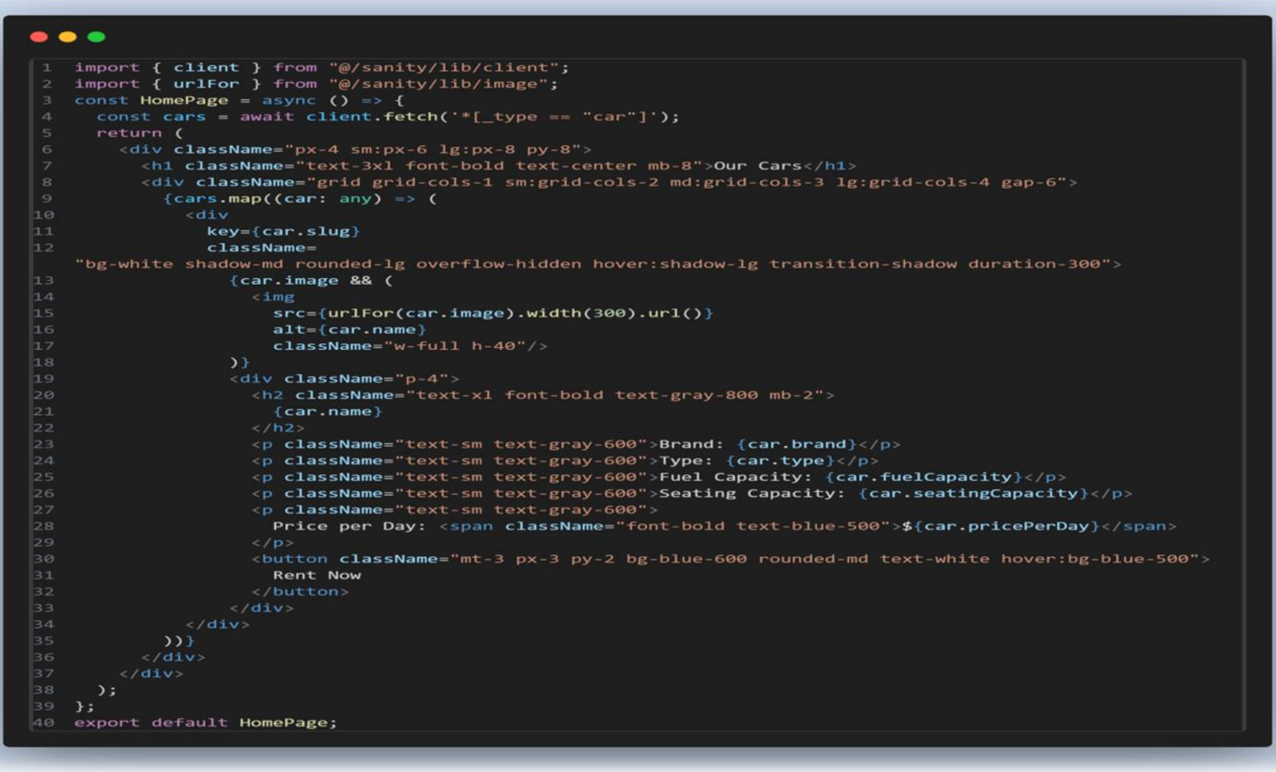
Step 04: API Integration in Next.js

API integration in **Next.js** involves connecting your frontend project to external APIs, including **Sanity's APIs**, to fetch or send data. The integration process includes creating utility functions to handle API requests, rendering the fetched data in components, and ensuring smooth functionality through error handling and testing.

Key steps:

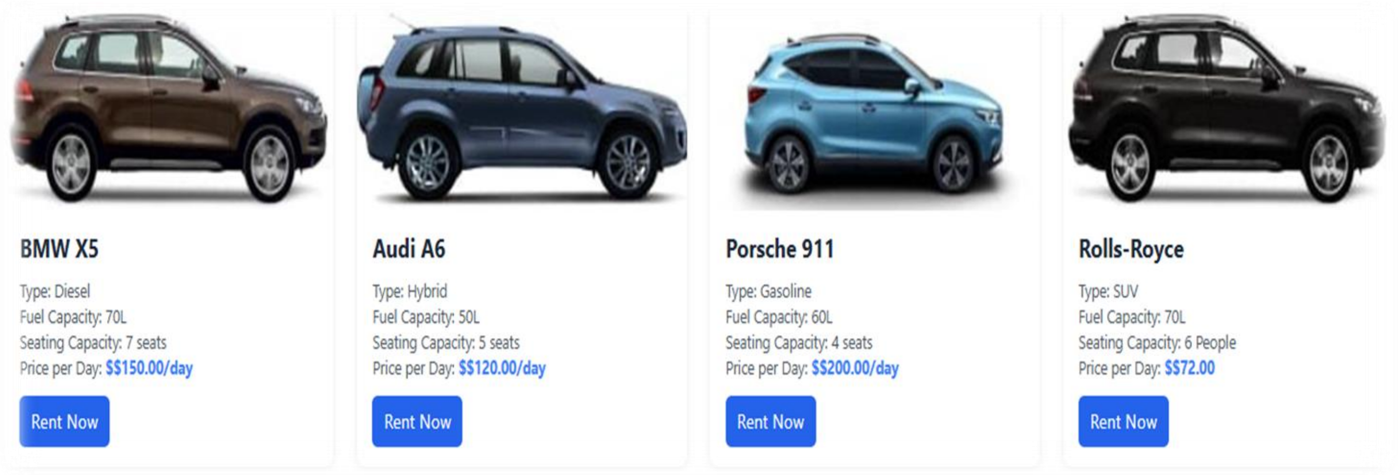
1. **Create Utility Functions**: Functions that fetch data from Sanity or external APIs.
2. **Render Data in Components**: Display the data dynamically in your frontend.
3. **Test API Integration**: Use tools like Postman or browser tools to ensure data consistency.

In this case, data is fetching using Sanity APIs with queries like GROQ:



```jsx
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
const HomePage = async () => {
  const cars = await client.fetch('*[_type == "car"]');
  return (
    <div className="px-4 sm:px-6 lg:px-8 py-8">
      <h1 className="text-3xl font-bold text-center mb-8">Our Cars</h1>
      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
        {cars.map((car: any) => (
          <div
            key={car.slug}
            className=
"bg-white shadow-md rounded-lg overflow-hidden hover:shadow-lg transition-shadow duration-300">
              {car.image && (
                <img
                  src={urlFor(car.image).width(300).url()}
                  alt={car.name}
                  className="w-full h-40"/>
              )}
              <div className="p-4">
                <h2 className="text-xl font-bold text-gray-800 mb-2">
                  {car.name}
                </h2>
                <p className="text-sm text-gray-600">Brand: {car.brand}</p>
                <p className="text-sm text-gray-600">Type: {car.type}</p>
                <p className="text-sm text-gray-600">Fuel Capacity: {car.fuelCapacity}</p>
                <p className="text-sm text-gray-600">Seating Capacity: {car.seatingCapacity}</p>
                <p className="text-sm text-gray-600">
                  Price per Day: <span className="font-bold text-blue-500">${car.pricePerDay}</span>
                </p>
                <button className="mt-3 px-3 py-2 bg-blue-600 rounded-md text-white hover:bg-blue-500">
                  Rent Now
                </button>
              </div>
          </div>
        ))}
      </div>
    </div>
  );
};
export default HomePage;
```

Data is fetched and displayed in Next.js.



BMW X5
Type: Diesel
Fuel Capacity: 70L
Seating Capacity: 7 seats
Price per Day: **$$150.00/day**
Rent Now

Audi A6
Type: Hybrid
Fuel Capacity: 50L
Seating Capacity: 5 seats
Price per Day: **$$120.00/day**
Rent Now

Porsche 911
Type: Gasoline
Fuel Capacity: 60L
Seating Capacity: 4 seats
Price per Day: **$$200.00/day**
Rent Now

Rolls-Royce
Type: SUV
Fuel Capacity: 70L
Seating Capacity: 6 People
Price per Day: **$$72.00**
Rent Now

THANK YOU