Hackathon Day 3 Task Submission

Team Name: Maryam Saleem

Project Name: Sanity: A Step-by-Step Guide

Day 3 - API Integration and Data Migration

Submission Content for Your Task

Here's a step-by-step guide to write the submission content:

1. Project Title:
"Dynamic Product Display Using Sanity and Next.js"

2. Objective:
The goal of this project was to fetch product data from Sanity CMS and display it dynamically on a Next.js frontend with proper styling and responsiveness.

3. Key Features:

- **Sanity Integration: Successfully connected Sanity CMS to Next.js using GROQ queries.**
- **Dynamic Data Fetching: Used Sanity's APIs to retrieve product details, including name, price, description, and image.**
- **Responsive Frontend Design: Built a responsive layout using Tailwind CSS, ensuring compatibility across devices.**
- **Clean Code Structure: Used modular functions for fetching data and organized React components efficiently.**

4. Technologies Used:

- **Frontend: Next.js (React Framework)**
- **Backend: Sanity CMS**
- **Styling: Tailwind CSS**
- **Programming Language: TypeScript/JavaScript**

5. Step-by-Step Implementation:

1. **Set Up Sanity CMS:**
   - **Created a new dataset in Sanity.**
   - **Added a schema for products with fields like name, price, description, image, and category.**
2. **Configured Sanity Client:**
   - **Installed the Sanity client in the Next.js project.**
   - **Set up a reusable client instance to connect to Sanity.**
3. **Created GROQ Query:**
   - **Wrote a GROQ query to fetch the required product fields.**
4. **Fetched Data in Next.js:**
   - **Used a custom fetchProducts function to call Sanity's APIs.**
   - **Managed the fetched data using React's useState and useEffect hooks.**
5. **Frontend Rendering:**
   - **Dynamically rendered product details, including name, price, description, and images.**
   - **Used Image from Next.js for optimized image loading.**
6. **Styling with Tailwind CSS:**
   - **Designed a responsive grid layout.**
   - **Styled individual product cards with hover effects for better user interaction.**

6. Challenges and Solutions:

- **Challenge: Handling dynamic images from Sanity in Next.js.**
  Solution: Used next/image and created a function to generate image URLs from Sanity assets.
- **Challenge: Managing dynamic and real-time updates from Sanity.**
  Solution: Ensured data fetching and rendering are handled efficiently with optimized GROQ queries.

7. Output:
A fully functional webpage that dynamically displays product data from Sanity CMS in a clean and responsive design.

9. Learning Outcome:
Through this project, I gained hands-on experience with:

- **Connecting a CMS to a modern frontend framework.**
- **Writing GROQ queries for fetching data efficiently.**
- **Implementing responsive designs with Tailwind CSS.**

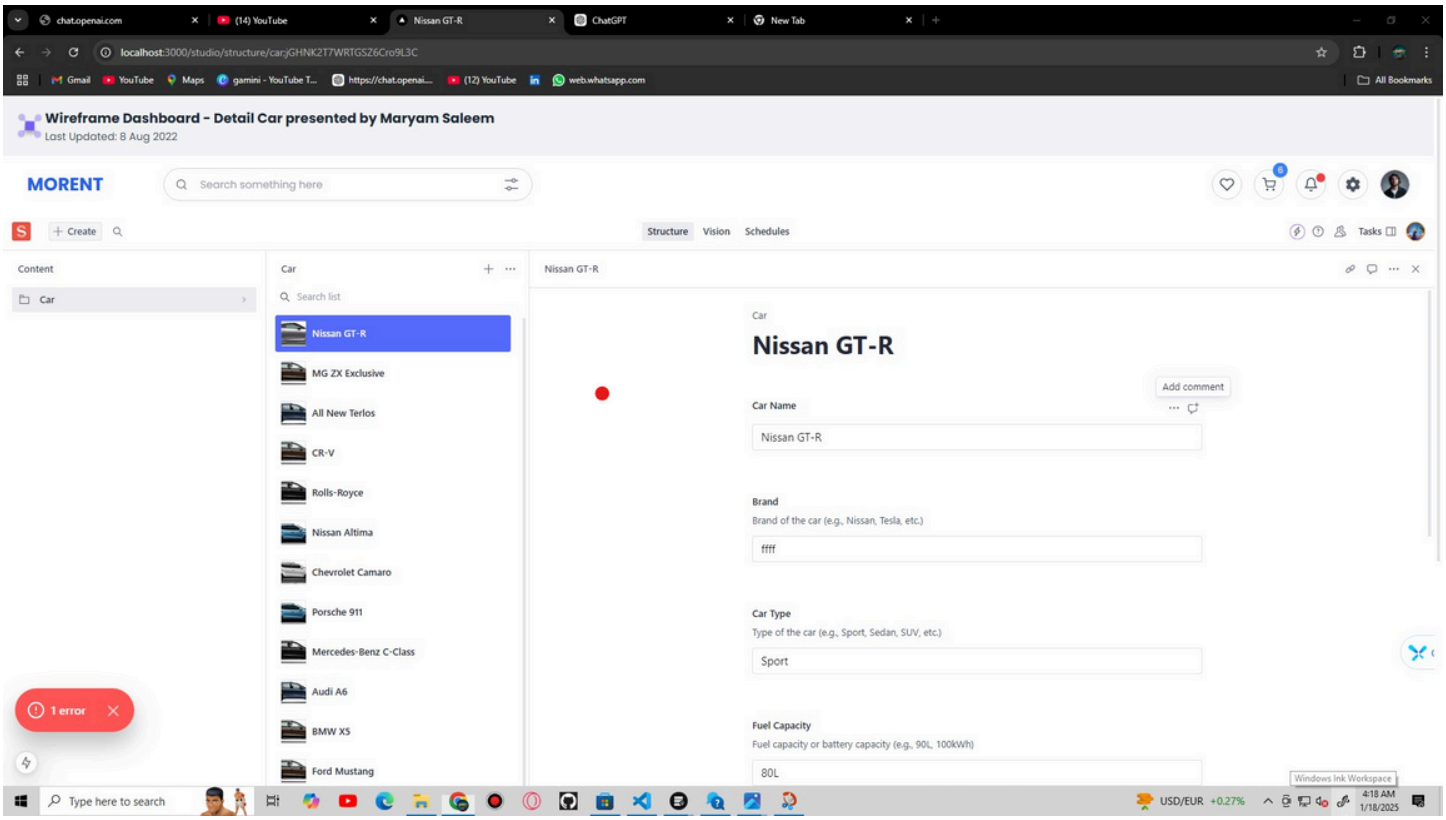
```
// import { defineField } from "sanity";
// const productSchema = defineField(

//  {
//   name: "product",
//   type: "document",
//   title: "products",
//   fields: [
//     { name: "name", type: "string", title: "name" },

//     { name: "title", type: "string", title: "Title" },

//       { name: "image", type: "image", title: "Image", options: { hotspot: true } },

//     // { name: "image", type: "url", title: "Image" },
//     { name: "price", type: "number", title: "Price" },

//     { name: "days", type: "string", title: "Days" },
//     { name: "buttonurl", type: "url", title: "Button url" },

//     {
//       name: "slug",
//       type: "slug",
//       title: "Slug",
//       options: {
//         source: "title",
//         maxLength: 200,
//       },
//     },
//   ],
// });

// export default productSchema;
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
      }
    }
  ],
};
```

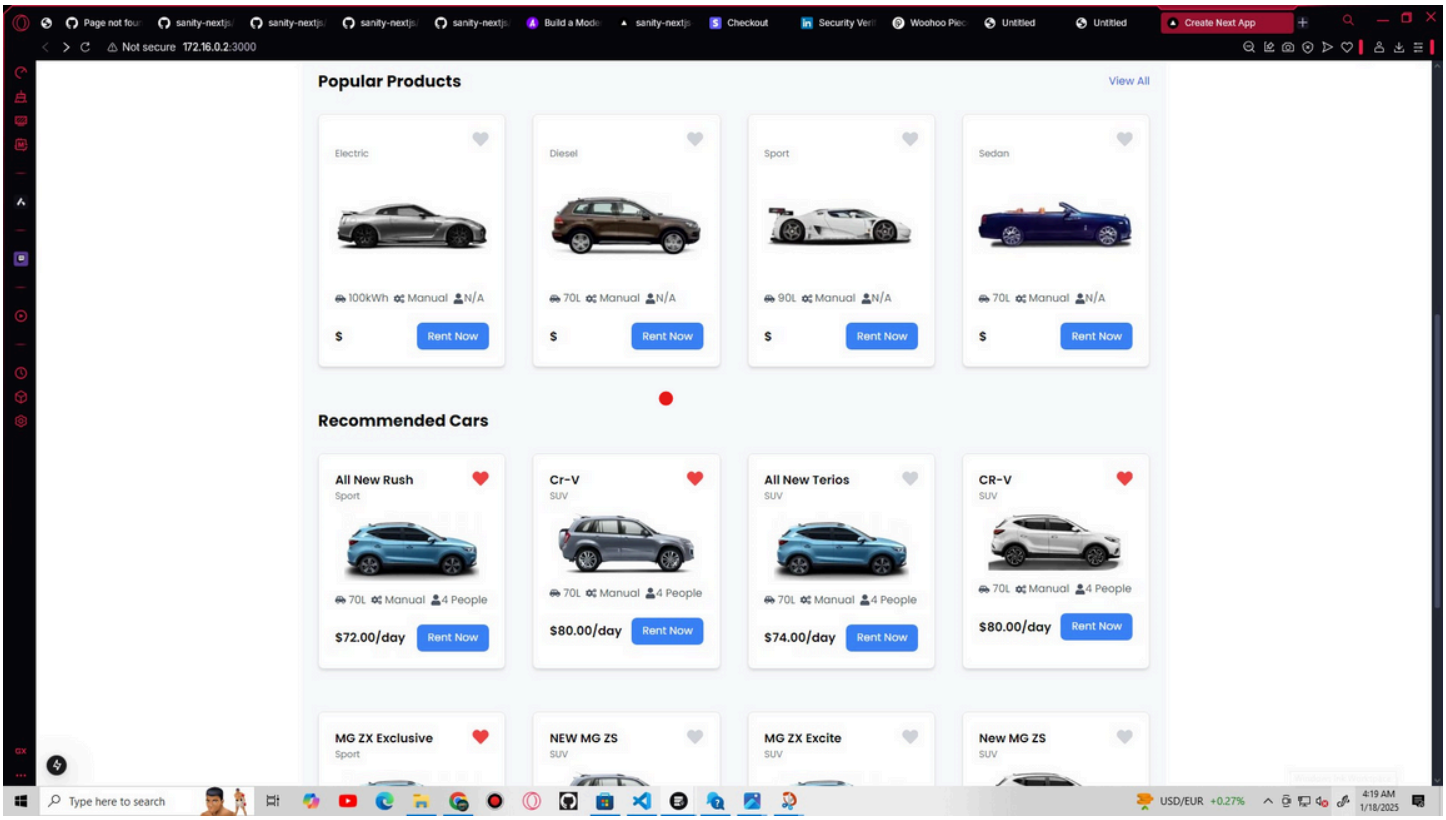
```tsx
import type { Metadata } from "next";
import "./globals.css";
import Header from "../components/homepage/header";
import Footer from "../components/homepage/footer";
import Navbar from "../components/homepage/navbar";
import { CartProvider } from "@/components/cart-component/CartContext";
import { Poppins } from "next/font/google";
import Hero from "@/components/homepage/cards1";

const poppins = Poppins({
  subsets: ["latin"],
  weight: ["200", "300", "400", "500", "600", "700"],
});

export const metadata: Metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <CartProvider>
      <html lang="en" className={poppins.className}>
        <body>
          <Header />
          <Navbar />
          {children}
          <Footer />
          {/* <Hero/> */}
        </body>
      </html>
    </CartProvider>
  );
}

```

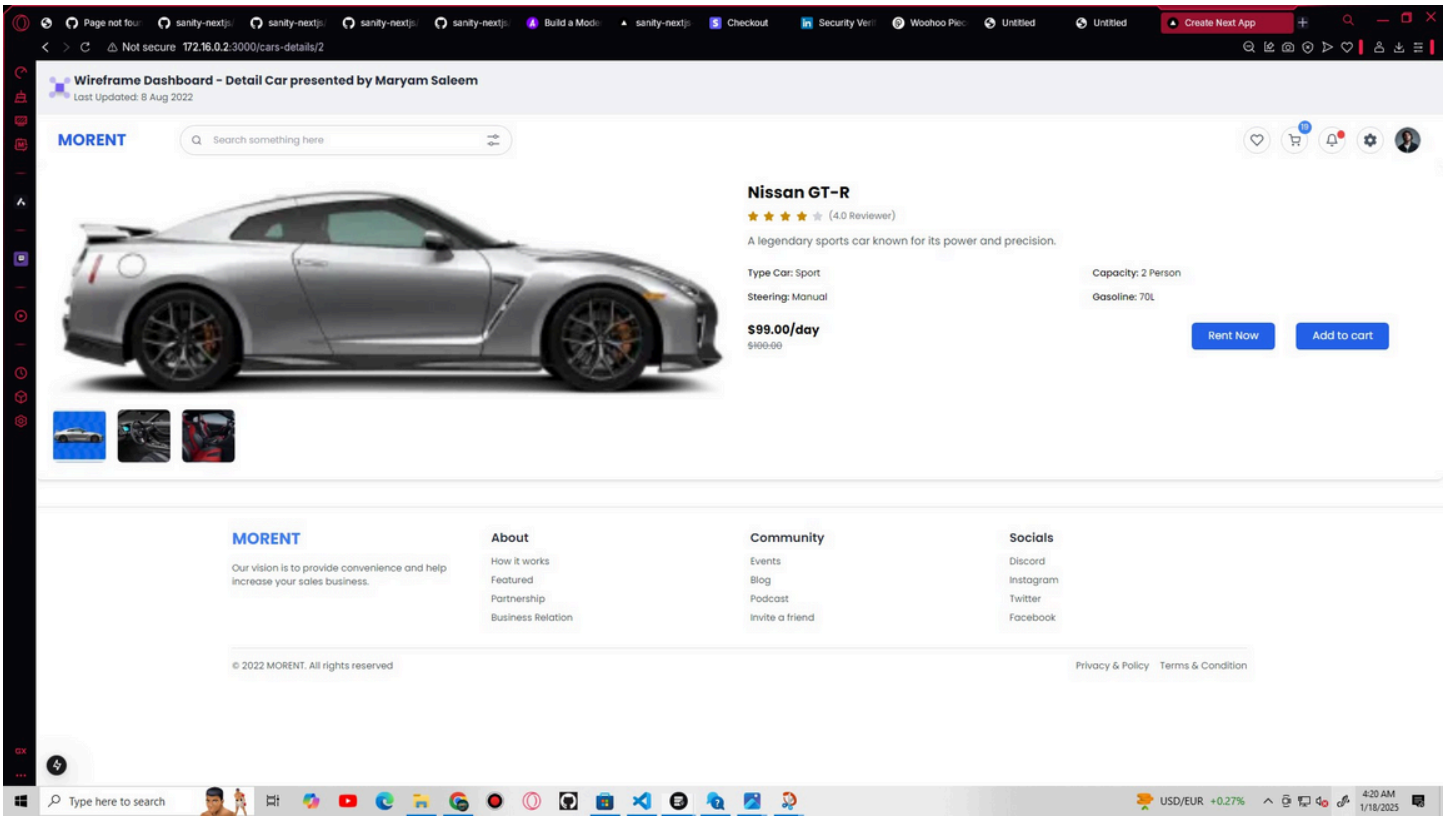
```tsx
import { type SchemaTypeDefinition } from 'sanity'
import productSchema from './cars'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [productSchema],
}

```

```tsx
import Link from 'next/link';
import Image from 'next/image';
import { client } from '@/sanity/lib/client';
import imageUrlBuilder from '@sanity/image-url';
import { FaHeart, FaUser, FaCarSide, FaCogs } from 'react-icons/fa'; // Import icons

interface IProduct {
  name: string;
  title: string;
  image: any;
  _id: string;
  stock: string;
  slug: { current: string };
  price: number;
  type?: string; // Add type if available
  fuelCapacity?: string; // Add fuelCapacity if available
  transmission?: string; // Add transmission if available
  capacity?: string; // Add capacity if available
  favorite?: boolean; // Add favorite if available
}

const builder = imageUrlBuilder(client);

function urlFor(source: any) {
  return builder.image(source).url();
}

export default async function Hero() {
  const res: IProduct[] = await client.fetch(`
    *[_type == "product"][0..3]{
      price,
      name,
      _type,
      _id,
      image,
      title,
      slug{current},
      type,
      fuelCapacity,
      transmission,
      capacity,
      favorite
    }
  `);

  return (
    <div className="max-w-screen-xl mx-auto px-6 py-8 bg-gray-50">
      <div className="flex justify-between items-center mb-8">
        <h2 className="text-2xl font-bold">Popular Products</h2>
        <Link href="/categories" className="text-blue-600 hover:underline">
          View All
        </Link>
      </div>
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-10">
        {res.map((product) => (
          <div
            key={product._id}
            className="border rounded-lg p-6 shadow-md bg-white hover:shadow-lg transition-shadow"
          >
            {/* Product Header */}
            {/* Product Header */}
<div className="flex justify-between items-start">
  <h3 className="text-lg font-semibold">{product.title}</h3>
  <button className="hover:text-red-500 transition-colors duration-200">
    <FaHeart
      className={`text-2xl ${
        product.favorite
          ? 'text-red-500' // Keep it red if favorite
          : 'text-gray-300 hover:text-red-500' // Gray by default, red on hover
      }`}
    />
  </button>
</div>
<p className="text-sm text-gray-500 mb-4">{product.type}</p>
            {/* Product Image */}
            {product.image && product.image.asset ? (
              <div className="relative w-full h-40">
                <Image
                  src={urlFor(product.image).toString()}
                  alt={product.title || 'Product Image'}
                  fill
                  className="object-contain"
                />
              </div>
            ) : (
              <div className="h-40 bg-gray-200 flex items-center justify-center rounded-lg">
                <p className="text-center text-gray-600">No Image Available</p>
              </div>
            )}

            {/* Product Details */}
            <div className="flex justify-between items-center mt-4">
              <div className="flex space-x-2 text-gray-600">
                <div className="flex items-center space-x-1">
                  <FaCarSide />
                  <span>{product.fuelCapacity || 'N/A'}</span>
                </div>
                <div className="flex items-center space-x-1">
                  <FaCogs />
                  <span>{product.transmission || 'N/A'}</span>
                </div>
                <div className="flex items-center">
                  <FaUser />
                  <span>{product.capacity || 'N/A'}</span>
                </div>
              </div>
            </div>

            {/* Product Price and Rent Button */}
            <div className="mt-6 flex justify-between items-center">
              <p className="text-lg font-semibold">${product.price}</p>
              <Link href={`/product/${product.slug.current}`}>
                <button className="bg-blue-500 text-white px-4 py-2 rounded-lg hover:bg-blue-600">
                  Rent Now
                </button>
              </Link>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
}
```

```javascript
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: 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) {
  try {
    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);
    return null;
  }
}

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 imageRef = null;
      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,
        pricePerDay: car.price_per_day,
        originalPrice: car.original_price || null,
        tags: car.tags || [],
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      console.log('Uploading car to Sanity:', sanityCar.name);
      const result = await client.create(sanityCar);
      console.log(`Car uploaded successfully: ${result._id}`);
    }

    console.log('Data import completed successfully!');
  } catch (error) {
    console.error('Error importing data:', error);
  }
}

importData();
```


Wireframe Dashboard - Detail Car presented by Maryam Saleem
Last Updated: 8 Aug 2022
MORENT
Search something here
Create
Structure
Vision
Schedules
Tasks
Content
Car
Search list
Nissan GT-R
MG ZX Exclusive
All New Terlos
CR-V
Rolls-Royce
Nissan Altima
Chevrolet Camaro
Porsche 911
Mercedes-Benz C-Class
Audi A6
BMW X5
Ford Mustang
Car
Nissan GT-R
Add comment
Car Name
Nissan GT-R
Brand
Brand of the car (e.g., Nissan, Tesla, etc.)
ffff
Car Type
Type of the car (e.g., Sport, Sedan, SUV, etc.)
Sport
Fuel Capacity
Fuel capacity or battery capacity (e.g., 90L, 100kWh)
80L
1 error
Type here to search
USD/EUR +0.27%
4:18 AM
1/18/2025


Not secure 172.16.0.2:3000
Create Next App
Popular Products
View All
Electric
Diesel
Sport
Sedan
100kWh Manual N/A
70L Manual N/A
90L Manual N/A
70L Manual N/A
$
Rent Now
Recommended Cars
All New Rush
Sport
70L Manual 4 People
$72.00/day
Rent Now
Cr-V
SUV
70L Manual 4 People
$80.00/day
Rent Now
All New Terios
SUV
70L Manual 4 People
$74.00/day
Rent Now
CR-V
SUV
70L Manual 4 People
$80.00/day
Rent Now
MG ZX Exclusive
Sport
NEW MG ZS
SUV
MG ZX Excite
SUV
New MG ZS
SUV
Type here to search
USD/EUR +0.27%
4:19 AM
1/18/2025



10. Conclusion:

This project demonstrates the ability to integrate a CMS backend (Sanity) with a modern frontend framework (Next.js) for dynamic content rendering, providing a robust and scalable solution for real-world applications.

Thanks for Attention