**Hackathon Day 3 Task Submission**

**Team Name: Maryam Saleem**

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

# Day 3 - API Integration and Data Migration

**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.**

## 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.

```
1   // import { defineField } from "sanity";
2   // const productSchema = defineField(
3
4   // {
5   //   name: "product",
6   //   type: "document",
7   //   title: "products",
8   //   fields: [
9   //     { name: "name", type: "string", title: "name" },
10
11  //     { name: "title", type: "string", title: "Title" },
12
13  //       { name: "image", type: "image", title: "Image", options: { hotspot: true } },
14
15  //     // { name: "image", type: "url", title: "Image" },
16  //     { name: "price", type: "number", title: "Price" },
17
18  //     { name: "days", type: "string", title: "Days" },
19  //     { name: "buttonurl", type: "url", title: "Button url" },
20
21  //     {
22  //       name: "slug",
23  //       type: "slug",
24  //       title: "Slug",
25  //       options: {
26  //         source: "title",
27  //         maxLength: 200,
28  //       },
29  //     },
30  //   ],
31  // });
32
33  // export default productSchema;
34  export default {
35    name: 'car',
36    type: 'document',
37    title: 'Car',
38    fields: [
39      {
40        name: 'name',
41        type: 'string',
42        title: 'Car Name',
43      },
44      {
45        name: 'brand',
46        type: 'string',
47        title: 'Brand',
48        description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
49      },
50      {
51        name: 'type',
52        type: 'string',
53        title: 'Car Type',
54        description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
55      },
56      {
57        name: 'fuelCapacity',
58        type: 'string',
59        title: 'Fuel Capacity',
60        description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
61      },
62      {
63        name: 'transmission',
64        type: 'string',
65        title: 'Transmission',
66        description: 'Type of transmission (e.g., Manual, Automatic)',
67      },
68      {
69        name: 'seatingCapacity',
70        type: 'string',
71        title: 'Seating Capacity',
72        description: 'Number of seats (e.g., 2 People, 4 seats)',
73      },
74      {
75        name: 'pricePerDay',
76        type: 'string',
77        title: 'Price Per Day',
78        description: 'Rental price per day',
79      },
80      {
81        name: 'originalPrice',
82        type: 'string',
83        title: 'Original Price',
84        description: 'Original price before discount (if applicable)',
85      },
86      {
87        name: 'tags',
88        type: 'array',
89        title: 'Tags',
90        of: [{ type: 'string' }],
91        options: {
92          layout: 'tags',
93        },
94        description: 'Tags for categorization (e.g., popular, recommended)',
95      },
96      {
97        name: 'image',
98        type: 'image',
99        title: 'Car Image',
100       options: {
101         hotspot: true
102       }
103     }
104   ],
105 };
```

```
1  import type { Metadata } from "next";
2  import "./globals.css";
3  import Header from "../components/homepage/header";
4  import Footer from "../components/homepage/footer";
5  import Navbar from "../components/homepage/navbar";
6  import { CartProvider } from "@/components/cart-component/CartContext";
7  import { Poppins } from "next/font/google";
8  import Hero from "@/components/homepage/cards1";
9
10 const poppins = Poppins({
11   subsets: ["latin"],
12   weight: ["200", "300", "400", "500", "600", "700"],
13 });
14
15 export const metadata: Metadata = {
16   title: "Create Next App",
17   description: "Generated by create next app",
18 };
19
20 export default function RootLayout({
21   children,
22 }: Readonly<{
23   children: React.ReactNode;
24 }>) {
25   return (
26     <CartProvider>
27       <html lang="en" className={poppins.className}>
28         <body>
29           <Header />
30           <Navbar />
31           {children}
32           <Footer />
33           {/* <Hero/> */}
34         </body>
35       </html>
36     </CartProvider>
37   );
38 }
39
```

```
1  import { type SchemaTypeDefinition } from 'sanity'
2  import productSchema from './cars'
3
4  export const schema: { types: SchemaTypeDefinition[] } = {
5    types: [productSchema],
6  }
7
```
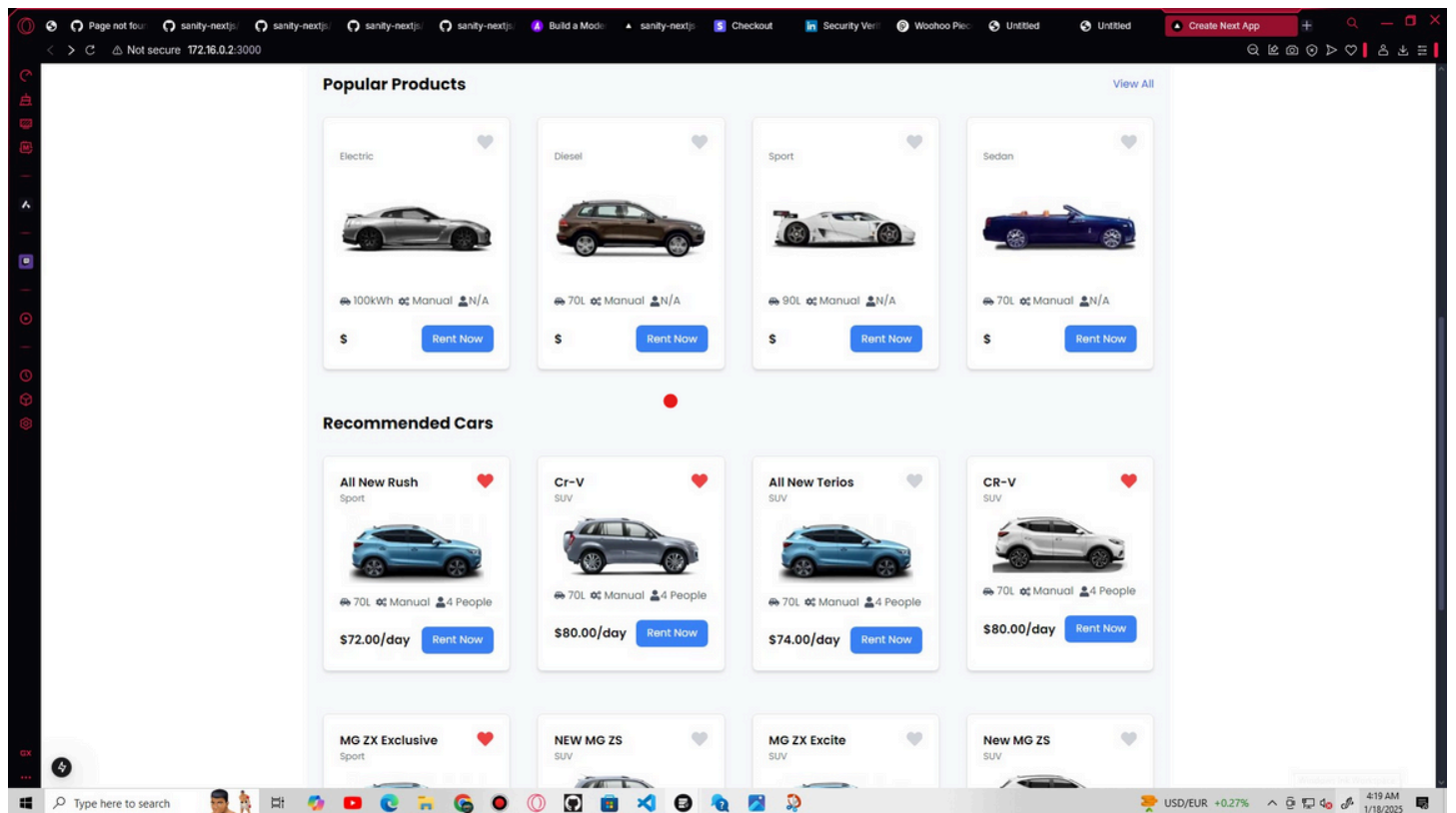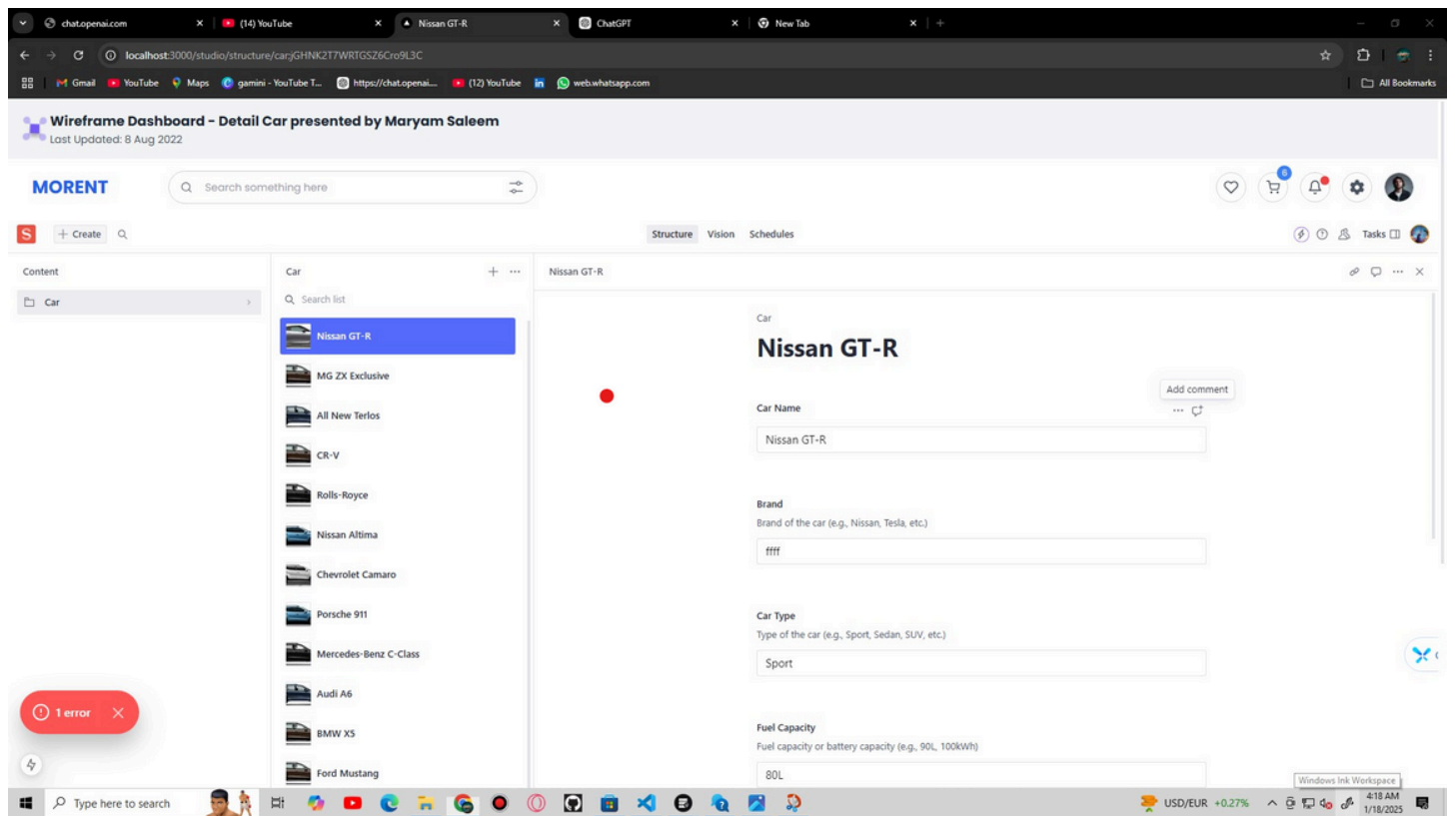
```tsx
import Link from 'next/link';
import Image from 'next/image';
import { client } from '@/sanity/lib/client';
import imageUrlBuilder from '@sanity/image-url';
import { FaHeart, FaUser, FaCarSide, FaCogs } from 'react-icons/fa'; // Import icons

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>
  );
}
```
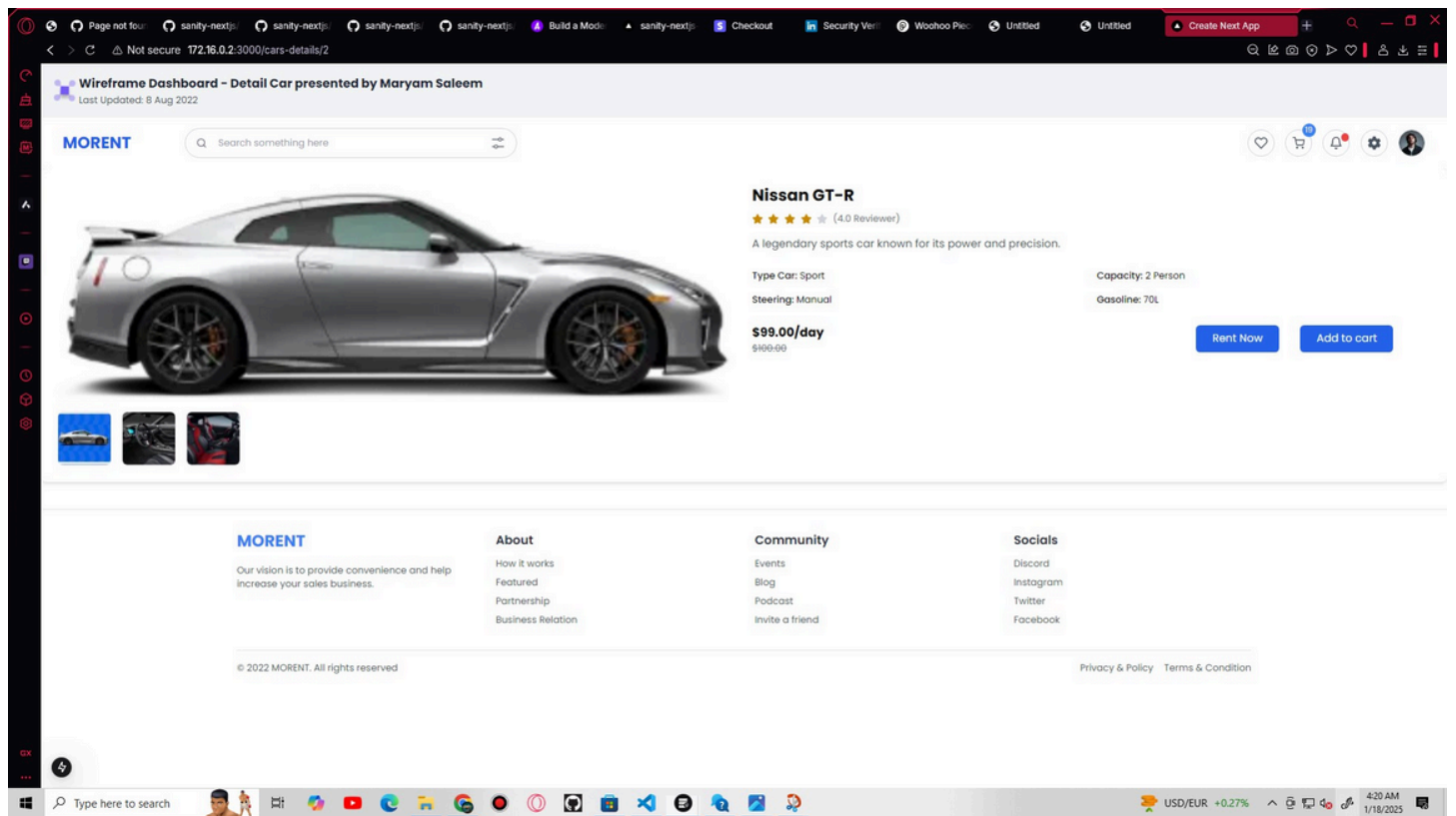
```javascript
1   import { createClient } from '@sanity/client';
2   import axios from 'axios';
3   import dotenv from 'dotenv';
4   import { fileURLToPath } from 'url';
5   import path from 'path';
6
7   // Load environment variables from .env.local
8   const __filename = fileURLToPath(import.meta.url);
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();
```

**Wireframe Dashboard - Detail Car presented by Maryam Saleem**
Last Updated: 8 Aug 2022

**MORENT**

Search something here

Structure Vision Schedules

Tasks

Content

Car

Car +

Nissan GT-R

Search list

Nissan GT-R

Car

# Nissan GT-R

MG ZX Exclusive

All New Terlos

Add comment

CR-V

**Car Name**

Rolls-Royce

Nissan GT-R

Nissan Altima

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

Chevrolet Camaro

ffff

Porsche 911

Mercedes-Benz C-Class

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

Audi A6

Sport

BMW X5

**Fuel Capacity**
Fuel capacity or battery capacity (e.g., 90L, 100kWh)

Ford Mustang

80L

1 error ✕

Windows Ink Workspace

Type here to search    USD/EUR +0.27%    4:18 AM 1/18/2025

---

Page not four | sanity-nextj | sanity-nextj | sanity-nextj | sanity-nextj | Build a Mode | sanity-nextjs | Checkout | Security Ver | Woohoo Piec | Untitled | Untitled | Create Next App | +

Not secure 172.16.0.2:3000

## 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

$         Rent Now

$         Rent Now

$         Rent Now

## Recommended Cars

**All New Rush**
Sport

**Cr-V**
SUV

**All New Terios**
SUV

**CR-V**
SUV

🚗 70L ⚙ Manual 👤 4 People

🚗 70L ⚙ Manual 👤 4 People

🚗 70L ⚙ Manual 👤 4 People

🚗 70L ⚙ Manual 👤 4 People

$72.00/day   Rent Now

$80.00/day   Rent Now

$74.00/day   Rent Now

$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

# THANK YOU