Day 04 of Hakathon 03

Car Rental Website Documentation

Project Overview

This project is a dynamic car rental e-commerce website developed as part of **Hackathon 3 - Day 4**. The platform enables users to browse available cars and rent them directly. The website is built using Next.js ,tailwind css and Sanity CMS for managing dynamic content. Key features include:

- Dynamic product listing using Sanity CMS.
- Individual car detail pages with dynamic routing.
- Responsive and user-friendly design.

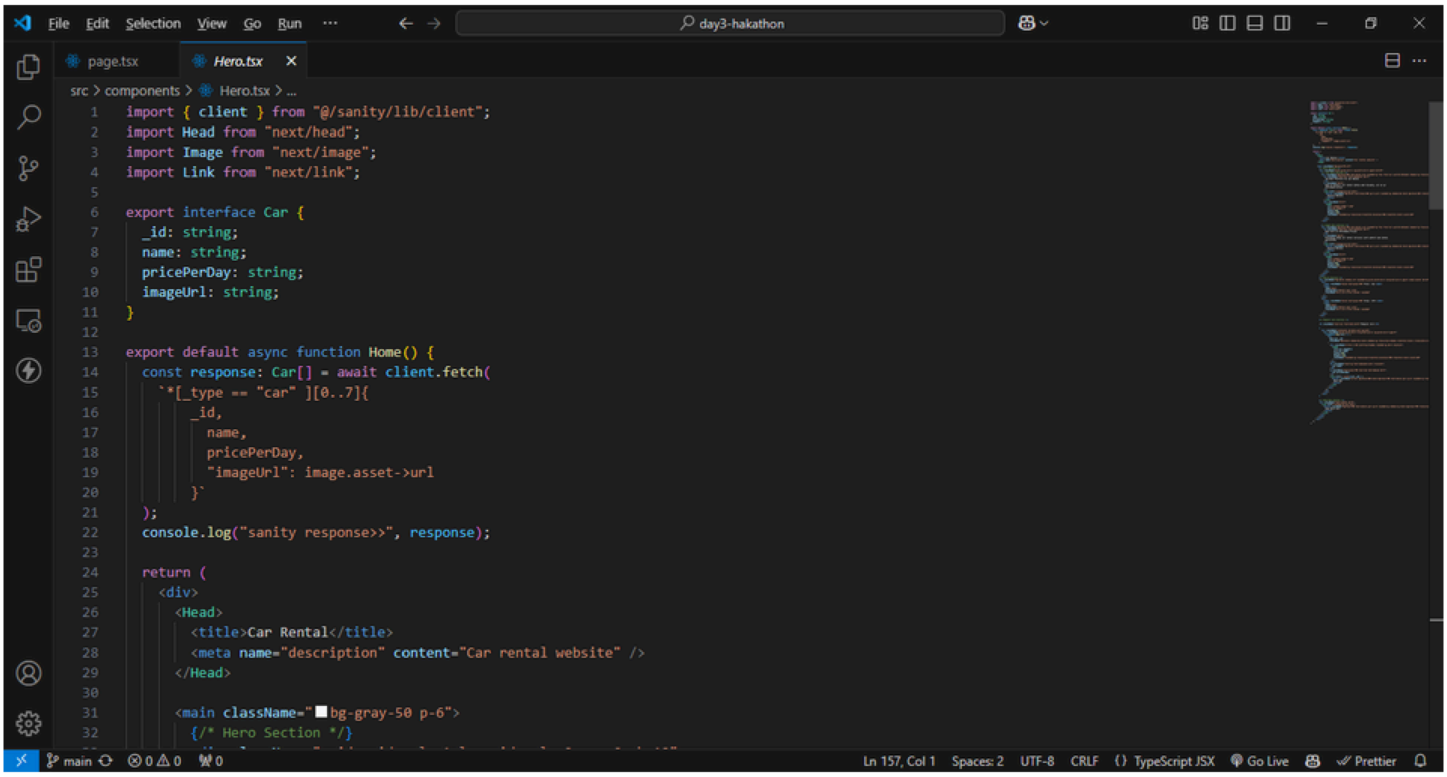
Dynamic Features Implemented

1. Product Listing Page

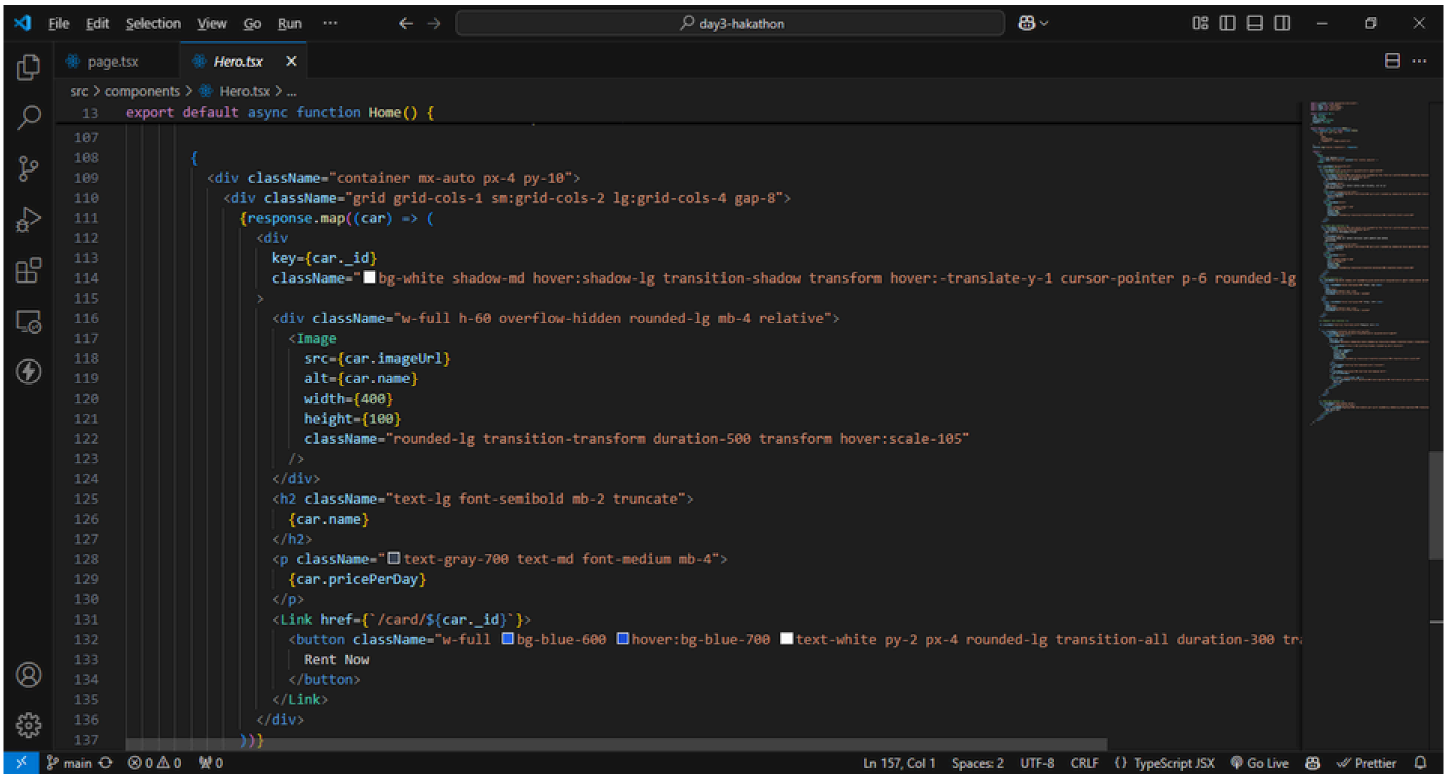
Description:

- Displays a list of cars dynamically fetched from Sanity CMS.
- Each car is displayed as a card with the following details:
  - Car title
  - Image
  - Price per day
- Includes a "Rent Now" button on each card.

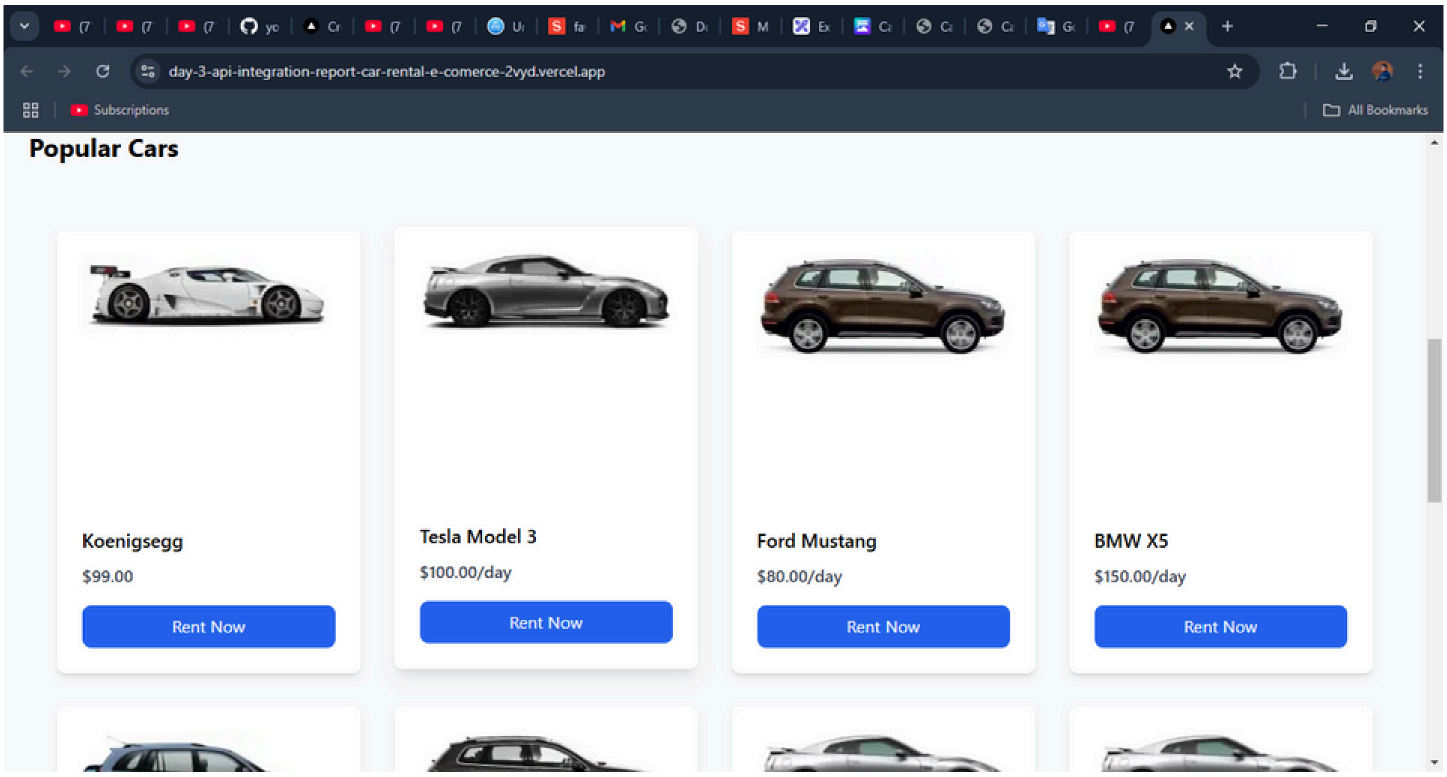
Code Snippet:


File Edit Selection View Go Run ···
day3-hakathon
page.tsx
Hero.tsx
src > components > Hero.tsx > ...
main 0 0 0
Ln 157, Col 1 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Prettier

```tsx
import { client } from "@/sanity/lib/client";
import Head from "next/head";
import Image from "next/image";
import Link from "next/link";

export interface Car {
  _id: string;
  name: string;
  pricePerDay: string;
  imageUrl: string;
}

export default async function Home() {
  const response: Car[] = await client.fetch(
    `*[_type == "car" ][0..7]{
      _id,
        name,
        pricePerDay,
        "imageUrl": image.asset->url
      }`
  );
  console.log("sanity response>>", response);

  return (
    <div>
      <Head>
        <title>Car Rental</title>
        <meta name="description" content="Car rental website" />
      </Head>

      <main className="bg-gray-50 p-6">
        {/* Hero Section */}
```


File Edit Selection View Go Run ···
day3-hakathon
page.tsx
Hero.tsx
src > components > Hero.tsx > ...
main 0 0 0
Ln 157, Col 1 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Prettier

```tsx
export default async function Home() {

        {
          <div className="container mx-auto px-4 py-10">
            <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-8">
              {response.map((car) => (
                <div
                  key={car._id}
                  className="bg-white shadow-md hover:shadow-lg transition-shadow transform hover:-translate-y-1 cursor-pointer p-6 rounded-lg
                >
                  <div className="w-full h-60 overflow-hidden rounded-lg mb-4 relative">
                    <Image
                      src={car.imageUrl}
                      alt={car.name}
                      width={400}
                      height={100}
                      className="rounded-lg transition-transform duration-500 transform hover:scale-105"
                    />
                  </div>
                  <h2 className="text-lg font-semibold mb-2 truncate">
                    {car.name}
                  </h2>
                  <p className="text-gray-700 text-md font-medium mb-4">
                    {car.pricePerDay}
                  </p>
                  <Link href={`/card/${car._id}`}>
                    <button className="w-full bg-blue-600 hover:bg-blue-700 text-white py-2 px-4 rounded-lg transition-all duration-300 tr
                      Rent Now
                    </button>
                  </Link>
                </div>
              ))}
```


day-3-api-integration-report-car-rental-e-comerce-2vyd.vercel.app
Subscriptions
All Bookmarks

Popular Cars

Koenigsegg
$99.00
Rent Now

Tesla Model 3
$100.00/day
Rent Now

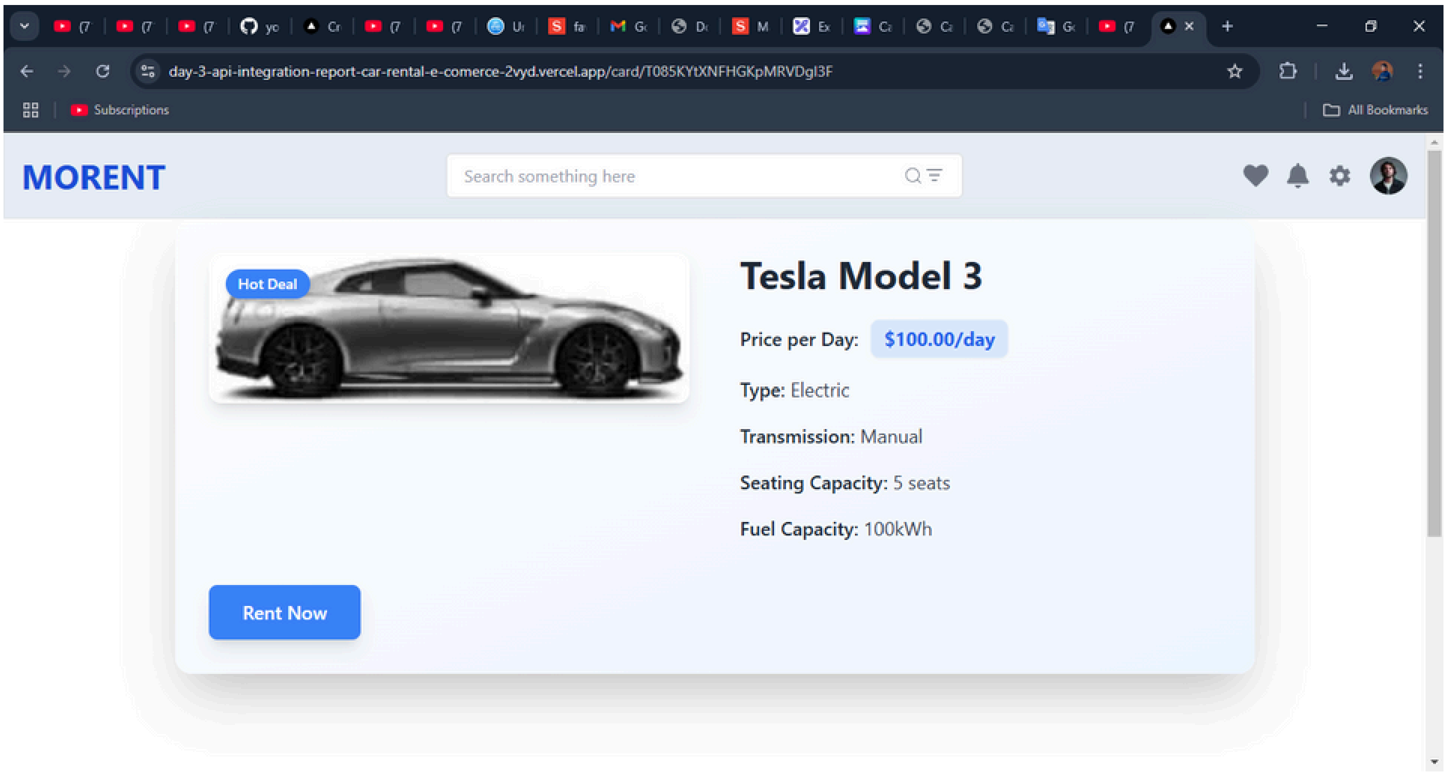
Ford Mustang
$80.00/day
Rent Now

BMW X5
$150.00/day
Rent Now

2. Dynamic Routing for Product Details

- **Description:**
  - Clicking "Rent Now" on a car card navigates to a dynamic detail page.
  - The detail page shows the car's:
    - Title
    - Price per day
    - Description
    - Image
- **Dynamic Route:** card/[_id]

Code Snippet:


day-3-api-integration-report-car-rental-e-comerce-2vyd.vercel.app/card/T085KYtXNFHGKpMRVDgl3F
Subscriptions
All Bookmarks
MORENT
Search something here
Hot Deal

Tesla Model 3

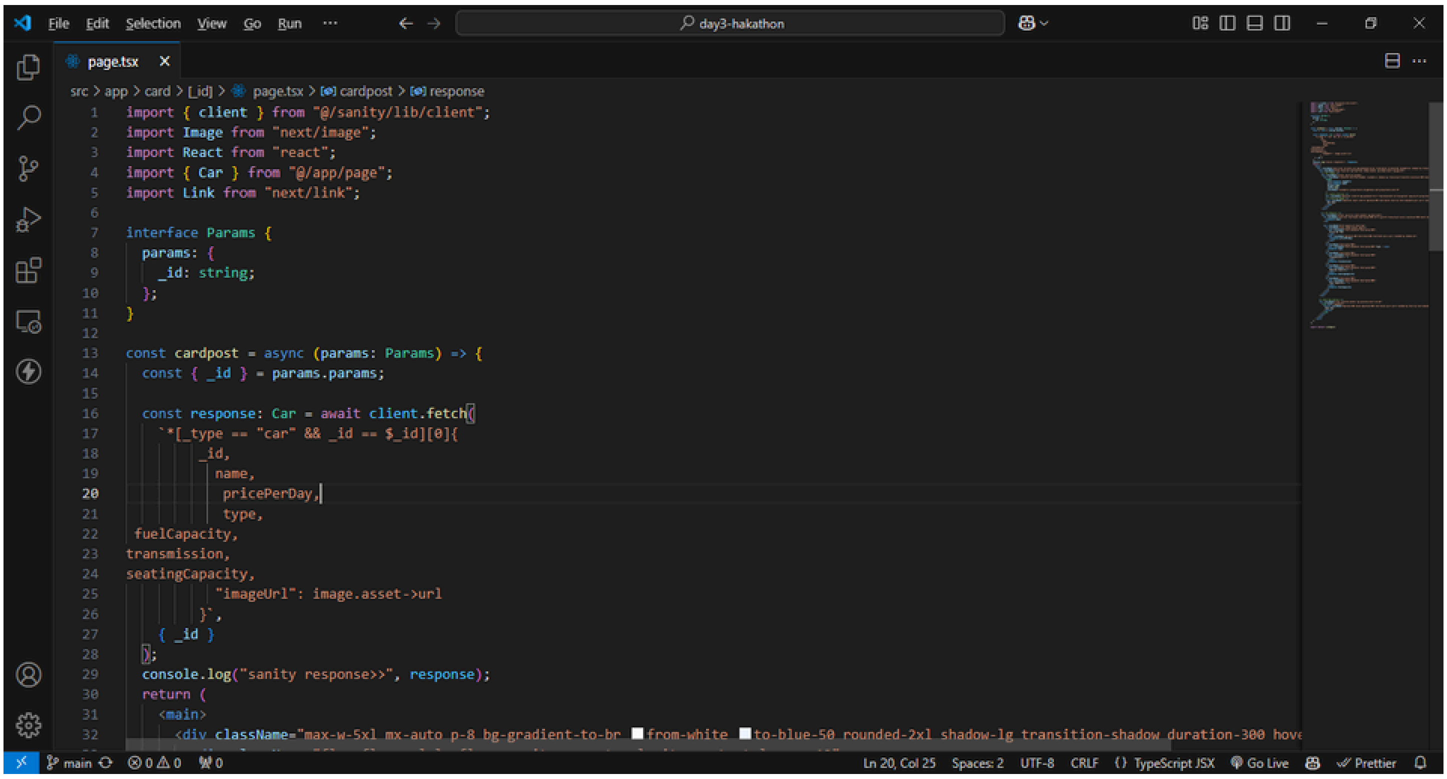
Price per Day: $100.00/day

Type: Electric

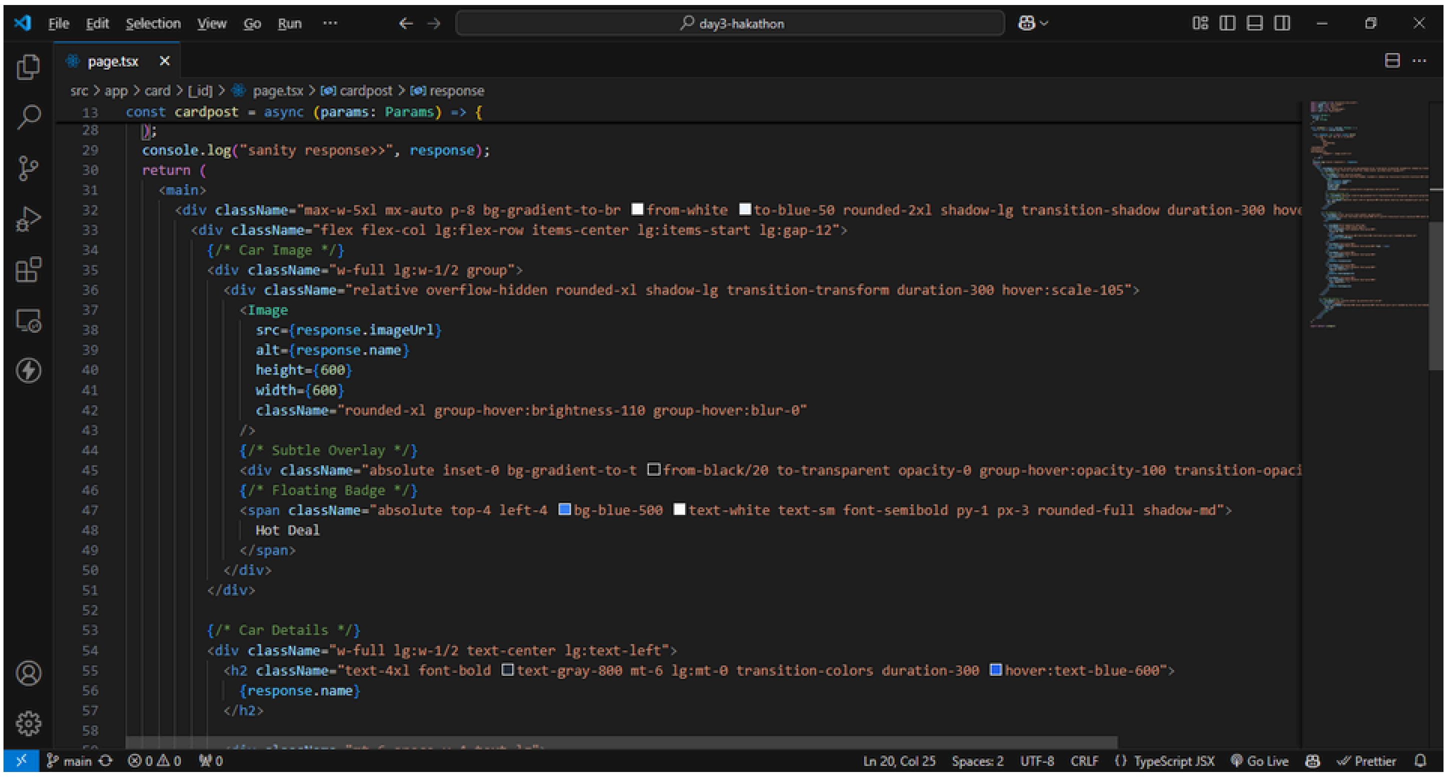
Transmission: Manual

Seating Capacity: 5 seats

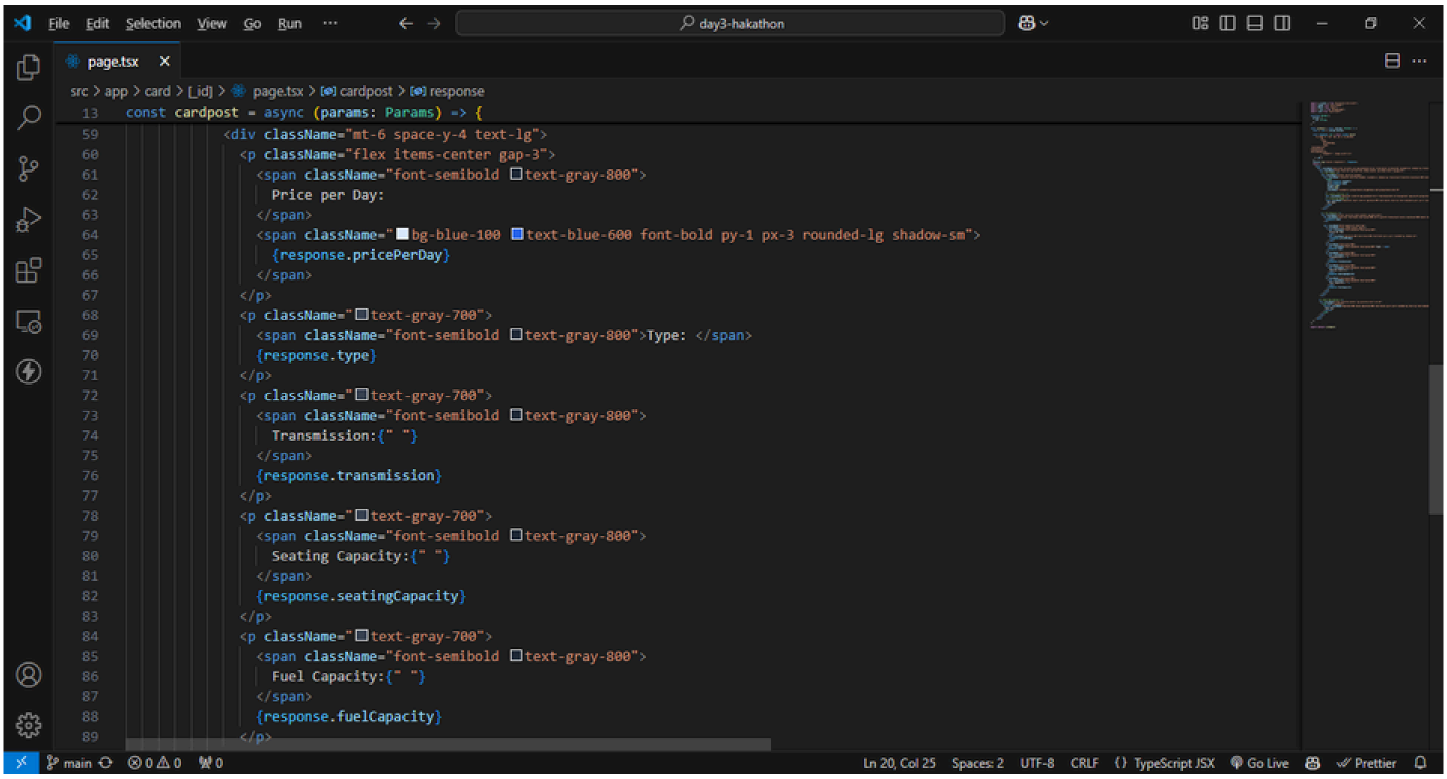
Fuel Capacity: 100kWh


File Edit Selection View Go Run ···
day3-hakathon
page.tsx
src > app > card > [_id] > page.tsx > cardpost > response
main ⊗ 0 ⚠ 0 ⑀ 0
Ln 20, Col 25 Spaces: 2 UTF-8 CRLF {} TypeScript JSX Go Live Prettier

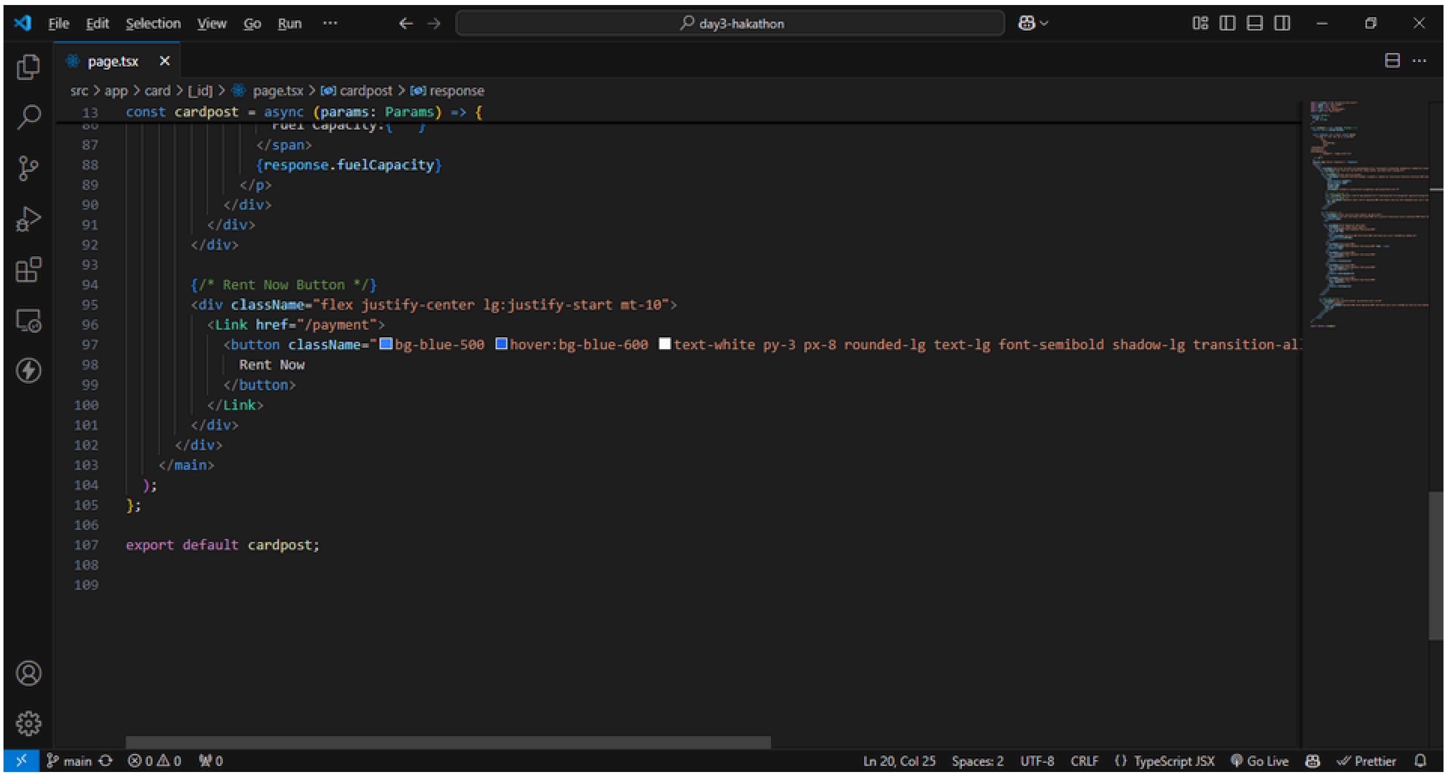
```
import { client } from "@/sanity/lib/client";
import Image from "next/image";
import React from "react";
import { Car } from "@/app/page";
import Link from "next/link";

interface Params {
  params: {
    _id: string;
  };
}

const cardpost = async (params: Params) => {
  const { _id } = params.params;

  const response: Car = await client.fetch(
    `*[_type == "car" && _id == $_id][0]{
          _id,
           name,
            pricePerDay,
            type,
 fuelCapacity,
transmission,
seatingCapacity,
           "imageUrl": image.asset->url
         }`,
    { _id }
  );
  console.log("sanity response>>", response);
  return (
    <main>
      <div className="max-w-5xl mx-auto p-8 bg-gradient-to-br from-white to-blue-50 rounded-2xl shadow-lg transition-shadow duration-300 hove
```


File Edit Selection View Go Run ···
day3-hakathon
page.tsx
src > app > card > [id] > page.tsx > cardpost > response
main 0 0 0
Ln 20, Col 25 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Prettier

```tsx
const cardpost = async (params: Params) => {
  });
  console.log("sanity response>>", response);
  return (
    <main>
      <div className="max-w-5xl mx-auto p-8 bg-gradient-to-br from-white to-blue-50 rounded-2xl shadow-lg transition-shadow duration-300 hove
        <div className="flex flex-col lg:flex-row items-center lg:items-start lg:gap-12">
          {/* Car Image */}
          <div className="w-full lg:w-1/2 group">
            <div className="relative overflow-hidden rounded-xl shadow-lg transition-transform duration-300 hover:scale-105">
              <Image
                src={response.imageUrl}
                alt={response.name}
                height={600}
                width={600}
                className="rounded-xl group-hover:brightness-110 group-hover:blur-0"
              />
              {/* Subtle Overlay */}
              <div className="absolute inset-0 bg-gradient-to-t from-black/20 to-transparent opacity-0 group-hover:opacity-100 transition-opaci
              {/* Floating Badge */}
              <span className="absolute top-4 left-4 bg-blue-500 text-white text-sm font-semibold py-1 px-3 rounded-full shadow-md">
                Hot Deal
              </span>
            </div>
          </div>

          {/* Car Details */}
          <div className="w-full lg:w-1/2 text-center lg:text-left">
            <h2 className="text-4xl font-bold text-gray-800 mt-6 lg:mt-0 transition-colors duration-300 hover:text-blue-600">
              {response.name}
            </h2>
```


File Edit Selection View Go Run ···
day3-hakathon
page.tsx
src > app > card > [id] > page.tsx > cardpost > response
main 0 0 0
Ln 20, Col 25 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Prettier

```tsx
const cardpost = async (params: Params) => {
            <div className="mt-6 space-y-4 text-lg">
              <p className="flex items-center gap-3">
                <span className="font-semibold text-gray-800">
                  Price per Day:
                </span>
                <span className="bg-blue-100 text-blue-600 font-bold py-1 px-3 rounded-lg shadow-sm">
                  {response.pricePerDay}
                </span>
              </p>
              <p className="text-gray-700">
                <span className="font-semibold text-gray-800">Type: </span>
                {response.type}
              </p>
              <p className="text-gray-700">
                <span className="font-semibold text-gray-800">
                  Transmission:{" "}
                </span>
                {response.transmission}
              </p>
              <p className="text-gray-700">
                <span className="font-semibold text-gray-800">
                  Seating Capacity:{" "}
                </span>
                {response.seatingCapacity}
              </p>
              <p className="text-gray-700">
                <span className="font-semibold text-gray-800">
                  Fuel Capacity:{" "}
                </span>
                {response.fuelCapacity}
              </p>
```



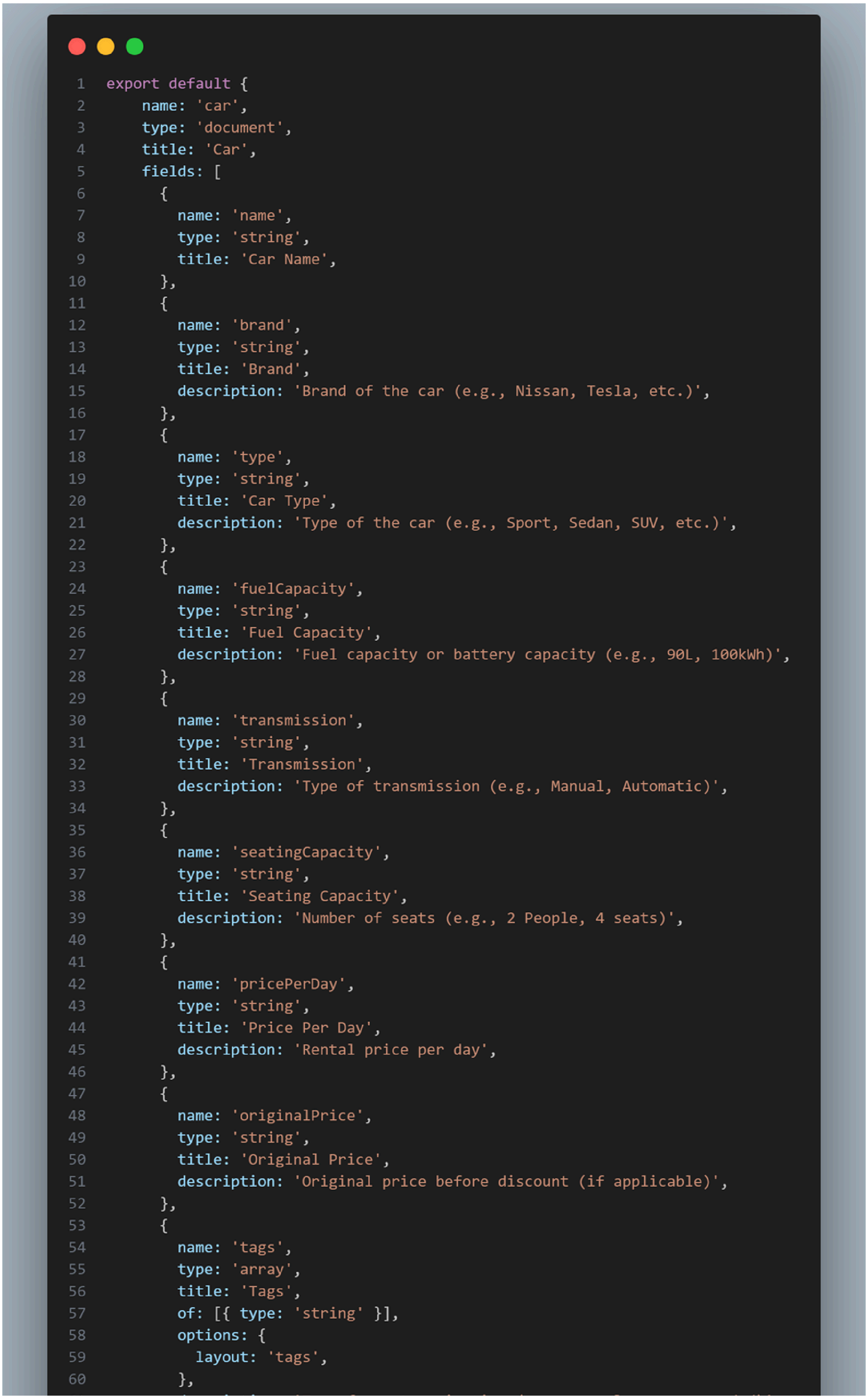
File Edit Selection View Go Run ···
day3-hakathon
page.tsx
src > app > card > [id] > page.tsx > cardpost > response
main 0 0 0
Ln 20, Col 25 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Prettier

```
const cardpost = async (params: Params) => {
                Fuel Capacity:{" "}
              </span>
              {response.fuelCapacity}
            </p>
          </div>
        </div>
      </div>

      {/* Rent Now Button */}
      <div className="flex justify-center lg:justify-start mt-10">
        <Link href="/payment">
          <button className="bg-blue-500 hover:bg-blue-600 text-white py-3 px-8 rounded-lg text-lg font-semibold shadow-lg transition-al
            Rent Now
          </button>
        </Link>
      </div>
    </div>
  </main>
  );
};

export default cardpost;
```

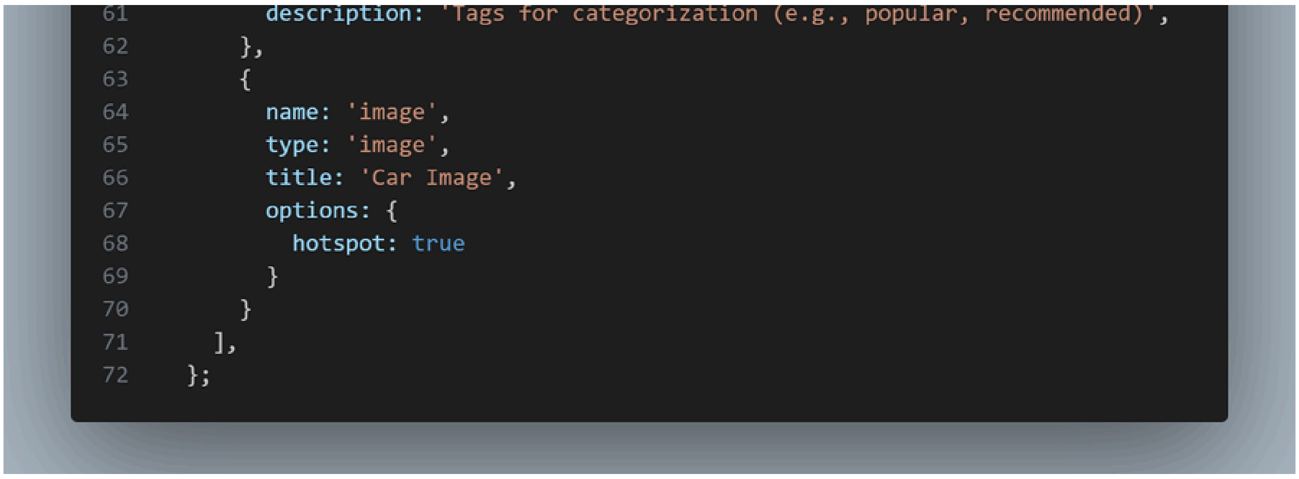
3. Sanity CMS Integration

- **Description:**
  - The data for cars (title, price and image) is managed dynamically through Sanity CMS.
- **Steps to Configure:**
  1. Define a schema in Sanity CMS for car with the following fields:
     - title (String)
     - price (Number)
     - image (Image)
     - id (String)
  2. Use the Sanity client in Next.js to fetch this data.

Code 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.)',
      },
      {
        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
      }
    }
  ],
};
```

Best Practices Followed

1. **Modular Component Design:**
   - Each feature is broken into reusable components for scalability.
2. **Responsive Design:**
   - The website is mobile-friendly, ensuring a seamless experience across devices.
3. **Dynamic Content Management:**
   - Sanity CMS ensures easy updates to content without changing the code.

Best Practices Followed

1. **Modular Component Design:**
   - Each feature is broken into reusable components for scalability.
2. **Responsive Design:**
   - The website is fully responsive, ensuring a seamless experience across all devices, including desktops, tablets, and mobile phones.
3. **Dynamic Content Management:**
   - Sanity CMS ensures easy updates to content without changing the code.

Future Enhancements

1. Add a booking confirmation page with payment integration.
2. Implement advanced search and filters (e.g., by price range or car type).
3. Add user authentication for personalized experiences.
4. Include reviews and ratings for cars.