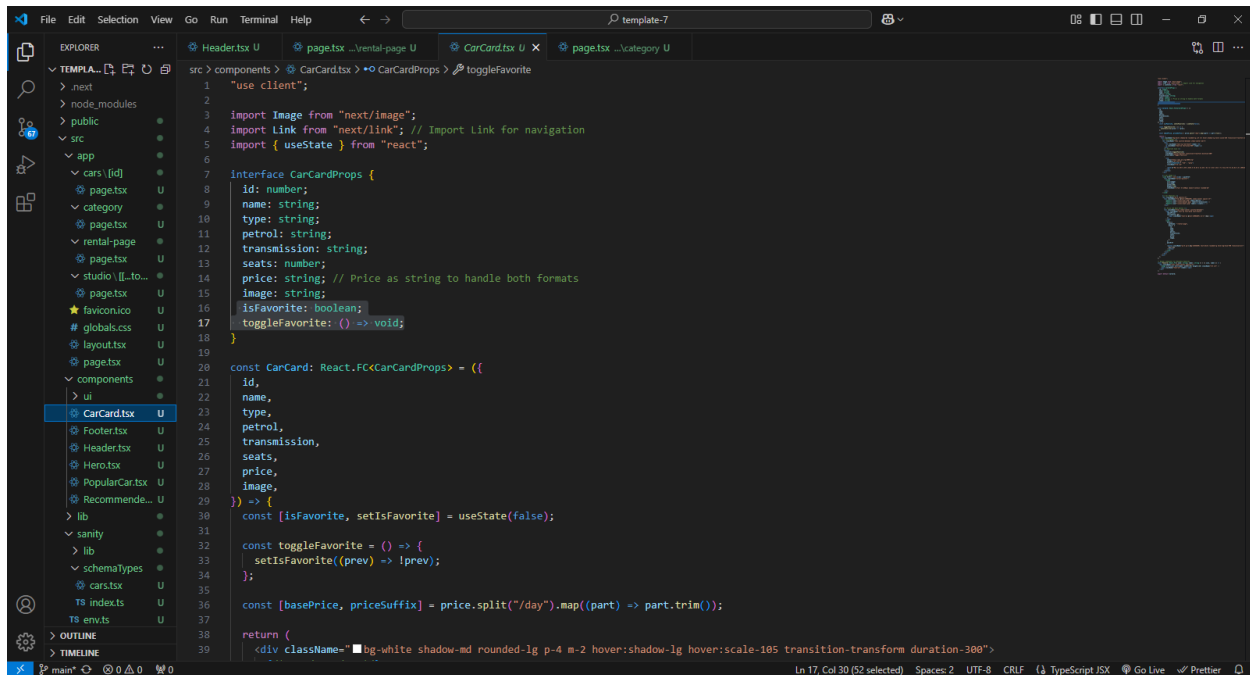


Day-4: Building Dynamic Frontend Components for My Marketplace

1. Reusable Car Card Component

- Created a reusable car card component to display car listings uniformly across the application.
- The component dynamically adjusts to show various car details, including name, type, price, image, and additional specifications.

A screenshot of a code editor (VS Code) showing the implementation of a reusable CarCard component. The Explorer panel on the left shows the project structure with folders like 'components' and 'ui'. The 'CarCard.tsx' file is selected. The main editor displays the code for the component, which includes an interface for props, state management for favoriting, and a render function for the card's HTML structure. The code is written in TypeScript and uses React hooks like useState and useReducer.

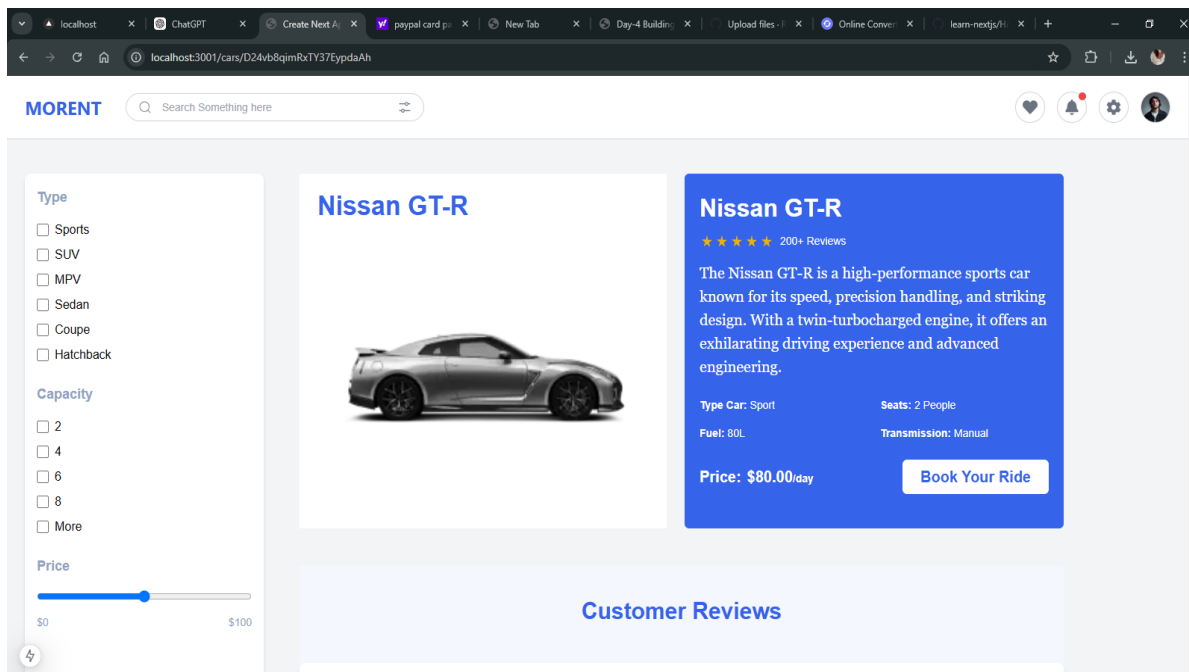
```
1  "use client";
2
3  import Image from "next/image";
4  import Link from "next/link"; // Import Link for navigation
5  import { useState } from "react";
6
7  interface CarCardProps {
8    id: number;
9    name: string;
10   type: string;
11   petrol: string;
12   transmission: string;
13   seats: number;
14   price: string; // Price as string to handle both formats
15   image: string;
16   isFavorite: boolean;
17   toggleFavorite: () => void;
18 }
19
20 const CarCard: React.FC<CarCardProps> = ({
21   id,
22   name,
23   type,
24   petrol,
25   transmission,
26   seats,
27   price,
28   image,
29 }) => {
30   const [isFavorite, setIsFavorite] = useState(false);
31
32   const toggleFavorite = () => {
33     setIsFavorite((prev) => !prev);
34   };
35
36   const [basePrice, priceSuffix] = price.split("/day").map((part) => part.trim());
37
38   return (
39     <div className="bg-white shadow-md rounded-lg p-4 m-2 hover:shadow-lg hover:scale-105 transition-transform duration-300">
```

2. Dynamic Car Detail Page

- Developed a fully dynamic car detail page that displays specific details of the selected car.
- Ensured that clicking any car redirects users to a detail page showing information such as car name, type, price, transmission, fuel type, seats, and image.

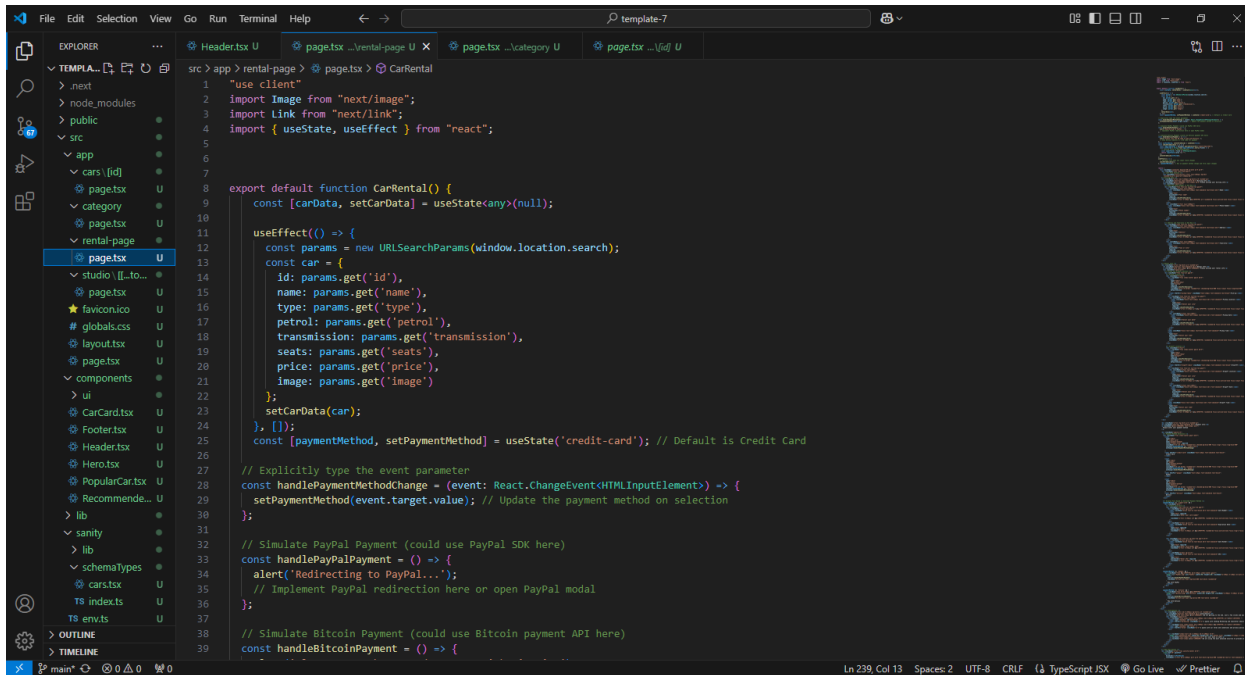
The screenshot displays the Visual Studio Code (VS Code) editor interface. The top bar shows the standard menu (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar. The Explorer sidebar on the left shows the project structure, with the file explorer expanded to show the 'src' directory. The file 'page.tsx' is selected, and its content is displayed in the main editor area. The code is a React component named 'CarDetailPage' that uses the 'useState' and 'useEffect' hooks from 'react'. It also uses 'useParams' from 'next/navigation' and 'Sanity client' from 'sanity.cli'. The component defines a 'CarDetailProps' interface and a 'CarDetailPage' function that uses the 'useState' hook to manage the 'isFavorite' state. The component also uses the 'toggleFavorite' function to toggle the 'isFavorite' state. The component is rendered with the 'CarDetail' component from 'components/ui/sheet'.

```
src > app > cars > [id] > page.tsx > CarDetailPage
1  "use client";
2  import { useState, useEffect } from "react";
3  import Image from "next/image";
4  import { useParams } from "next/navigation"; // Import useParams
5  import { client } from "../../sanity.cli"; // Sanity client
6  import Link from "next/link";
7  import ImageUrlBuilder from "@sanity/image-url"; // Import the image URL builder
8  import { Sheet, SheetContent, SheetTrigger, SheetTitle } from "@components/ui/sheet";
9  import { Menu } from "lucide-react";
10
11 interface CarDetailProps {
12   car: {
13     id: string;
14     name: string;
15     description: string;
16     type: string;
17     fuelCapacity: string;
18     transmission: string;
19     seatingCapacity: number;
20     pricePerDay: string;
21     image: any; // Update type to reflect image object from Sanity
22     features: string[];
23   };
24 }
25
26 // Create the image URL builder instance
27 const builder = imageUrlBuilder(client);
28
29 // Helper function to generate image URL
30 const urlFor = (source: any) => builder.image(source).url();
31
32 const CarDetailPage = () => {
33   const [isFavorite, setIsFavorite] = useState(false);
34   const [car, setCar] = useState<CarDetailProps["car"] | null>(null); // Initial state is null to handle undefined car
35   const { id } = useParams(); // Get the id from params using useParams hook
36
37   const toggleFavorite = () => {
38     setIsFavorite((prev) => !prev);
39   };
40 }
```

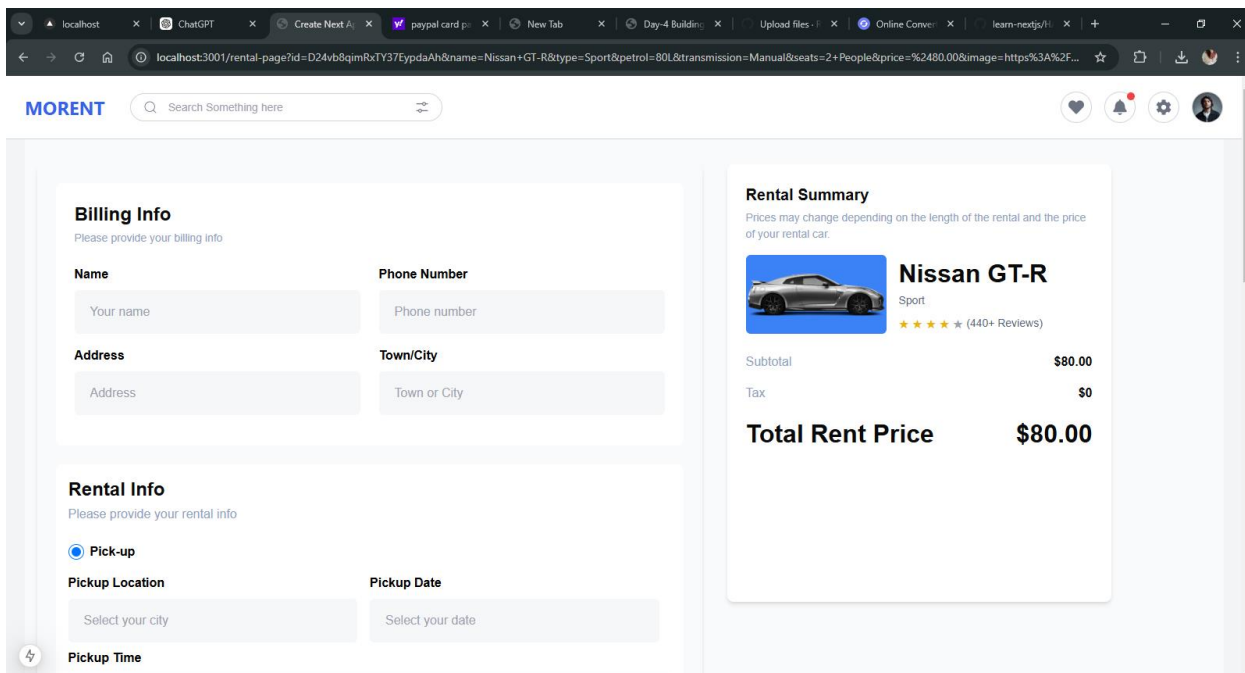


3. Dynamic "Rent Now" Button

- Integrated a dynamic "Rent Now" button on the car detail page.
- Clicking the button displays the car's price and a checkout form for booking.



```
1 "use client"
2 import Image from "next/image";
3 import Link from "next/link";
4 import { useState, useEffect } from "react";
5
6
7
8 export default function CarRental() {
9   const [carData, setCarData] = useState<any>(null);
10
11   useEffect(() => {
12     const params = new URLSearchParams(window.location.search);
13     const car = {
14       id: params.get('id'),
15       name: params.get('name'),
16       type: params.get('type'),
17       petrol: params.get('petrol'),
18       transmission: params.get('transmission'),
19       seats: params.get('seats'),
20       price: params.get('price'),
21       image: params.get('image')
22     };
23     setCarData(car);
24   }, []);
25   const [paymentMethod, setPaymentMethod] = useState('credit-card'); // Default is Credit Card
26
27   // Explicitly type the event parameter
28   const handlePaymentMethodChange = (event: React.ChangeEvent<HTMLInputElement>) => {
29     setPaymentMethod(event.target.value); // Update the payment method on selection
30   };
31
32   // Simulate PayPal Payment (could use PayPal SDK here)
33   const handlePayPalPayment = () => {
34     alert('Redirecting to PayPal...');
35     // Implement PayPal redirection here or open PayPal modal
36   };
37
38   // Simulate Bitcoin Payment (could use Bitcoin payment API here)
39   const handleBitcoinPayment = () => {
```



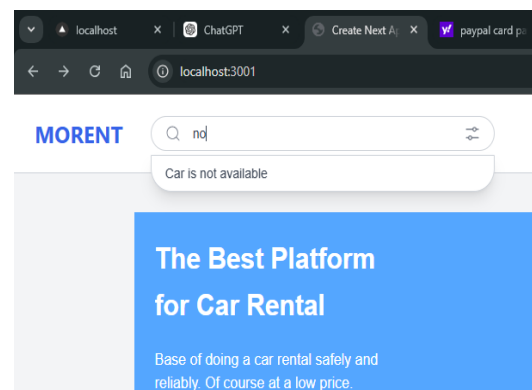
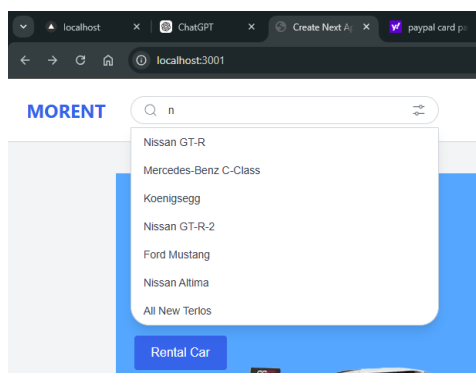
4. Checkout Form with Validation

- Designed a user-friendly checkout form to collect customer details during the booking process.
- Added validation to ensure that all required fields are filled before proceeding, enhancing data accuracy and user experience.

The screenshot shows a web browser window with the URL `localhost:3001/rental-page?id=D24vb8qimRxTY37EypdaAh&name=Nissan+GT-R&type=Sport&petrol=80L&trans`. The page features the MORENT logo and a search bar. Below the search bar, there are two main sections: "Billing Info" and "Rental Info". The "Billing Info" section includes fields for Name, Phone Number, Address, and Town/City. The "Rental Info" section includes a "Pick-up" section with a "Pickup Location" dropdown and a "Pickup Date" date picker, and a "Pickup Time" section.

5. Search Bar Functionality

- Implemented a search bar to allow users to find cars by their name.
- Clicking the search result navigates users to the corresponding car detail page.
- Added an error message ("Car is not available") for cases where the entered car name does not match any record.



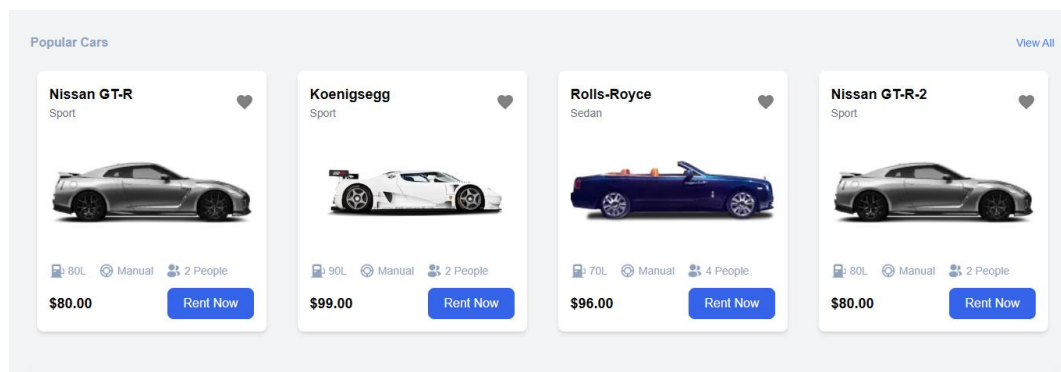
```

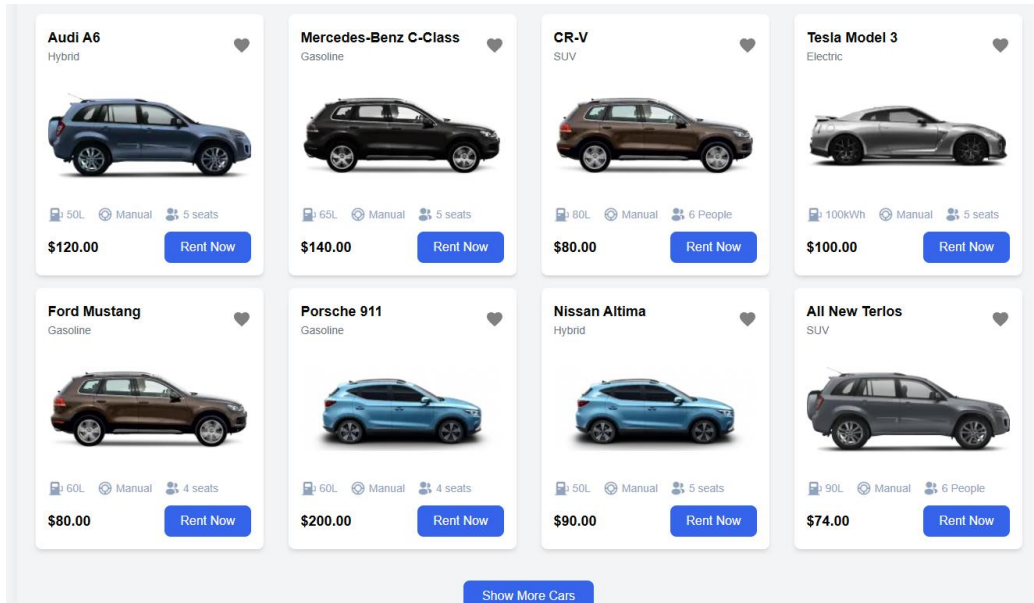
7 import { client } from "../../sanity.cli";
8 import Link from "next/link";
9 import Image from "next/image";
10
11 const Header = () => {
12   const [searchTerm, setSearchTerm] = useState("");
13   const [cars, setCars] = useState<any[]>([]);
14   const [filteredCars, setFilteredCars] = useState<any[]>([]);
15   const [showSuggestions, setShowSuggestions] = useState(false);
16   const router = useRouter();
17
18   useEffect(() => {
19     const fetchCars = async () => {
20       const carData = await client.fetch(
21         `*_type == "car"[]{name, _id}`
22       );
23       setCars(carData);
24       setFilteredCars(carData);
25     };
26     fetchCars();
27   }, []);
28
29   const handleSearch = (event: React.ChangeEvent<HTMLInputElement>) => {
30     const searchTerm = event.target.value.toLowerCase();
31     setSearchTerm(searchTerm);
32     setShowSuggestions(!searchTerm);
33     const filtered = cars.filter((car) =>
34       car.name.toLowerCase().includes(searchTerm)
35     );
36     setFilteredCars(filtered);
37   };
38
39   const handleSearchResultClick = (id: string) => {
40     setSearchTerm(""); // Clear the search input
41     setShowSuggestions(false); // Hide the dropdown
42     router.push(`/cars/${id}`); // Navigate to the car detail page
43   };
44
45   return (

```

6. Fully Dynamic "Popular Cars" and "Recommended Cars" Components

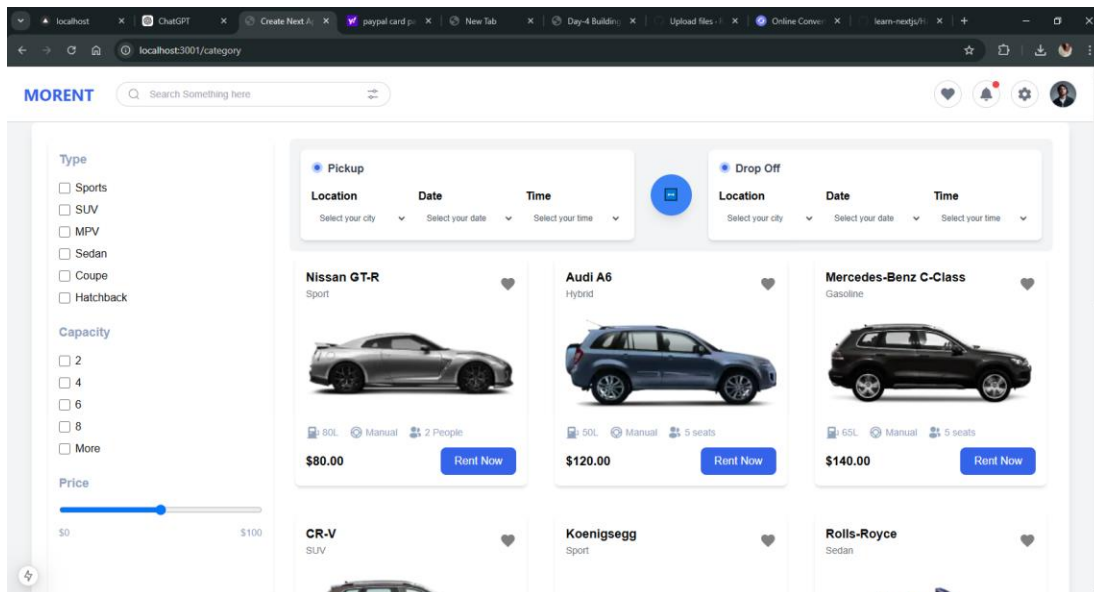
- Developed a "Popular Cars" component to display a list of cars tagged as popular.
- Created a "Recommended Cars" component for displaying a curated selection of recommended vehicles.
- Both components are fully dynamic, ensuring that clicking any car in these sections takes the user to its respective detail page.





7. Dynamic Category Page

- Built a category page that shows all the cars in my sanity with the functionality of show more and show less button.
- Clicking on any car in a category redirects users to the respective car detail page dynamically.



Conclusion

We have successfully developed our marketplace with the following key components:

- A reusable car card component used across multiple sections.
- Fully dynamic "Popular Cars" and "Recommended Cars" components for specialized car listings.
- A dynamic car detail component displaying individual car information.
- A fully dynamic rental page with a validated checkout form.
- A search bar component with error handling for unavailable car searches.
- A category page which show all the available cars.

This comprehensive development ensures an intuitive and seamless user experience across all pages of the marketplace.