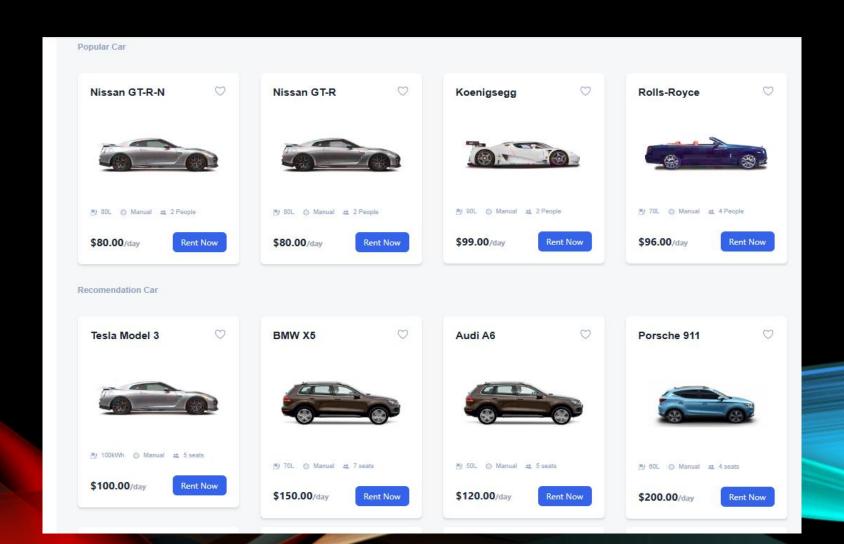
DAY 4 HAKATHON MORENT RENTAL CAR

PRODUCT DISPLAY PAGE WITH DYNAMIC DATA

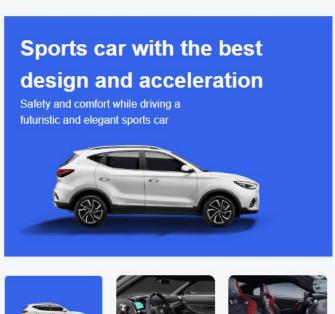


INDIVIDUAL PRODUCT DETAIL PAGES

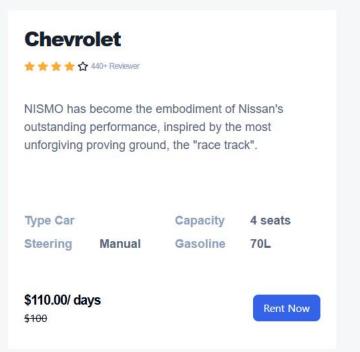
Select Car Type Choose the car type you are interested in. Diesel Gasoline Sedan Sports

SUV
Hybrid
Hatchback

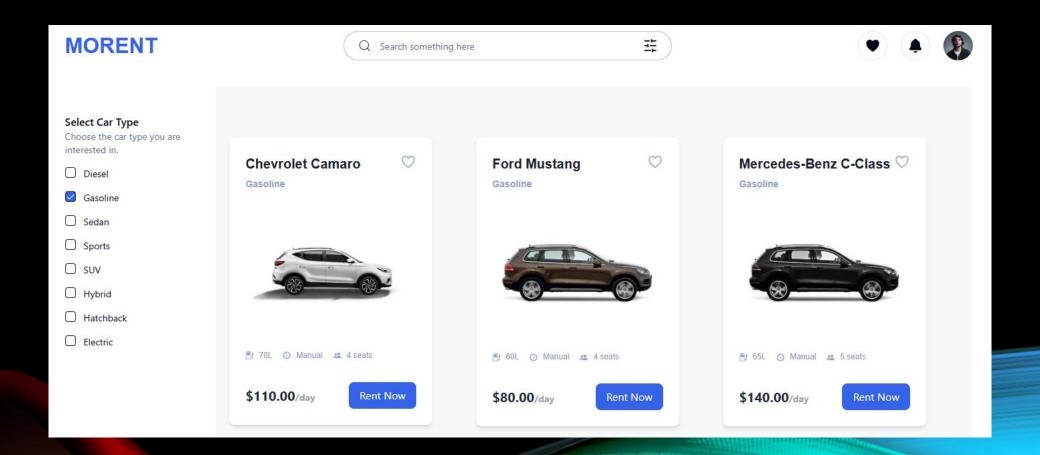
☐ Electric



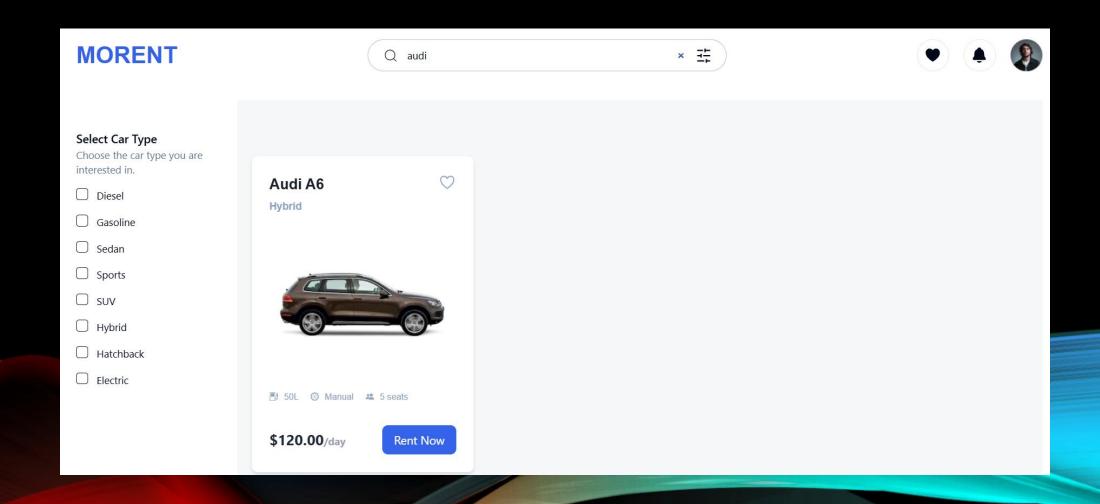
Q Search something here



WORKING CATEGORY FILTERS



WORKING CATEGORY SEARCHBAR



DOCUMENTATION

STEPS TAKEN TO BUILD AND INTEGRATE COMPONENTS

Setup and Configuration:

- •Initialized a Next.js project and configured Tailwind CSS for styling.
- •Integrated Sanity CMS to manage car data (e.g., car name, type, price, images).

2. Dynamic Routing:

- •Implemented dynamic routes for car details using **generateStaticParams** to pre-render pages for each car based on its slug.
- •Created a [Car] dynamic route to display detailed information about a specific car.

3.Data Fetching:

- •Used **getCars** to fetch car data from Sanity CMS.
- •Implemented USEEffect to fetch data on the client side for filtering and rendering.

4. Filtering Functionality:

- Added a sidebar with checkboxes for filtering cars by type (e.g., SUV, Sedan) and seating capacity.
- •Used USEState to manage selected filters and dynamically update the list of displayed cars.

DOCUMENTATION

CHALLENGES FACED AND SOLUTIONS IMPLEMENTED

1.Dynamic Routing Errors:

- Challenge: Errors occurred when accessing car.slug.current due to missing or undefined Slug fields.
- •Solution: Added conditional checks to ensure car.slug and car.slug.current exist before rendering links.

2. Client-Side vs Server-Side Rendering:

- **Challenge:** The filtering functionality required client-side interactivity, but the page was treated as a Server Component by default.
- •Solution: Added the "USE Client" directive to mark the component as a Client Component, enabling the use of React hooks like useState and useEffect.

3.Data Fetching:

- **Challenge:** Initially used async/await directly in the component, which caused issues with server-side rendering.
- •Solution: Moved data fetching logic to use Effect to handle it on the client side.

DOCUMENTATION

BEST PRACTICES FOLLOWED DURING DEVELOPMENT

1. Modular Component Design:

•Created reusable components (e.g., CatalogCard, Sidebar) to promote code reusability and maintainability.

2. Dynamic Routing:

 Used Next.js's generateStaticParams to pre-render dynamic routes, improving SEO and performance.

3.State Management:

Used useState to manage filtering states and useEffect for side effects like data fetching.

4. Error Handling:

- Added conditional checks and fallback mechanisms to handle missing or undefined data gracefully
- Use try cath method or error handling dunring data fetching.

DOCUMENTATION WEBSITE FUNCTIONALITIES

1.Car Catalog:

- •Displays a list of cars with details like name, type, price, seating capacity, and transmission.
- •Users can click on a car to view its detailed information.

2. Dynamic Car Details:

• Each car has a dedicated page with detailed information, including images, specifications, and pricing.

3. Filtering:

Users can filter cars by type (e.g., SUV, Sedan) and seating capacity using the sidebar.

4. Responsive Design:

•The website is fully responsive, ensuring a seamless experience on desktops, tablets, and mobile devices.

5.Image Optimization:

•Car images are optimized for fast loading using Next.js's Image component.

DOCUMENTATION WEBSITE FUNCTIONALITIES

Conclusion

The car rental website successfully integrates dynamic routing, client-side filtering, and responsive design to provide a seamless user experience. By following best practices and addressing challenges effectively, the website is both performant and user-friendly. Future enhancements could include user authentication, booking functionality, and integration with a payment gateway.