

TESTING, ERROR HANDLING, AND BACKEND INTEGRATION

REFINEMENT-Day 5

Prepared by: Fatima Faisal

Objective: Ensuring all components are thoroughly tested, optimized for performance, and ready to handle customer-facing traffic. The emphasis is on testing backend integrations, implementing error handling, and refining the user experience.

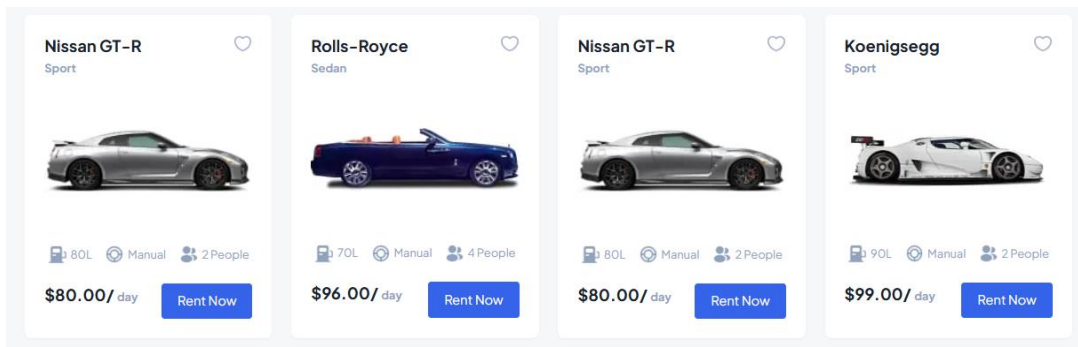
Steps for Implementation:

Step 1: Functional Testing

1. Test Core Features:

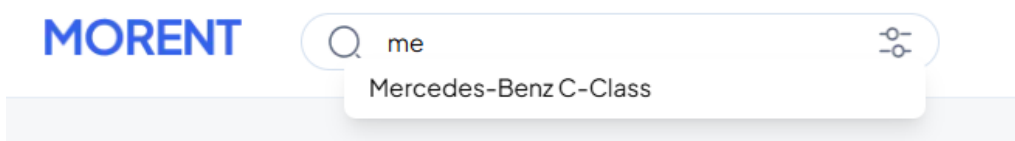
- **Product listing:**

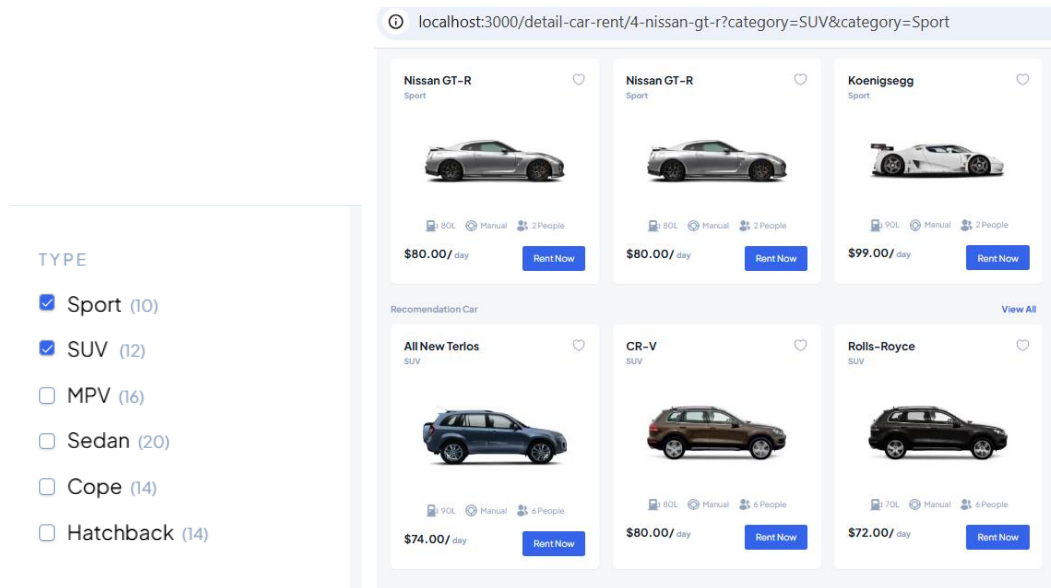
All the products are displayed accurately and correctly.



- **Filters and Search:**

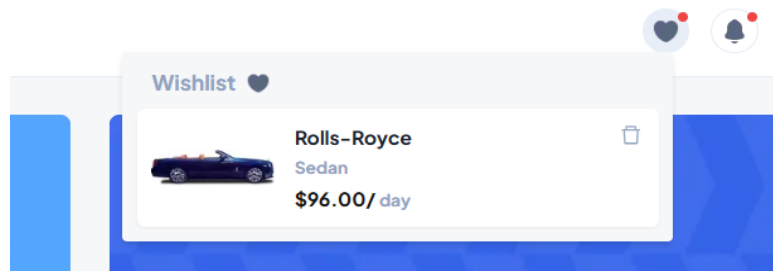
Search results and filters are producing accurate results based on user inputs and selections.





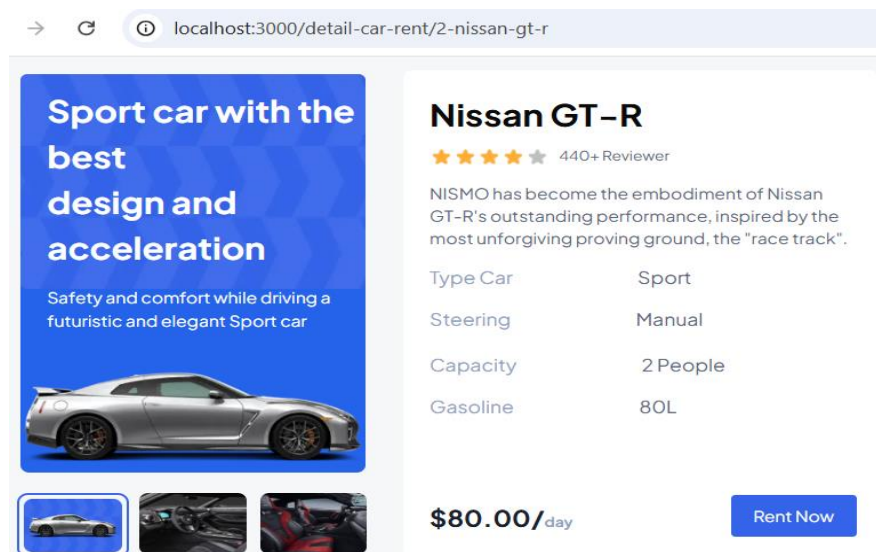
- **Wishlist Operations:**

Adding, updating and deleting items from the wish list.



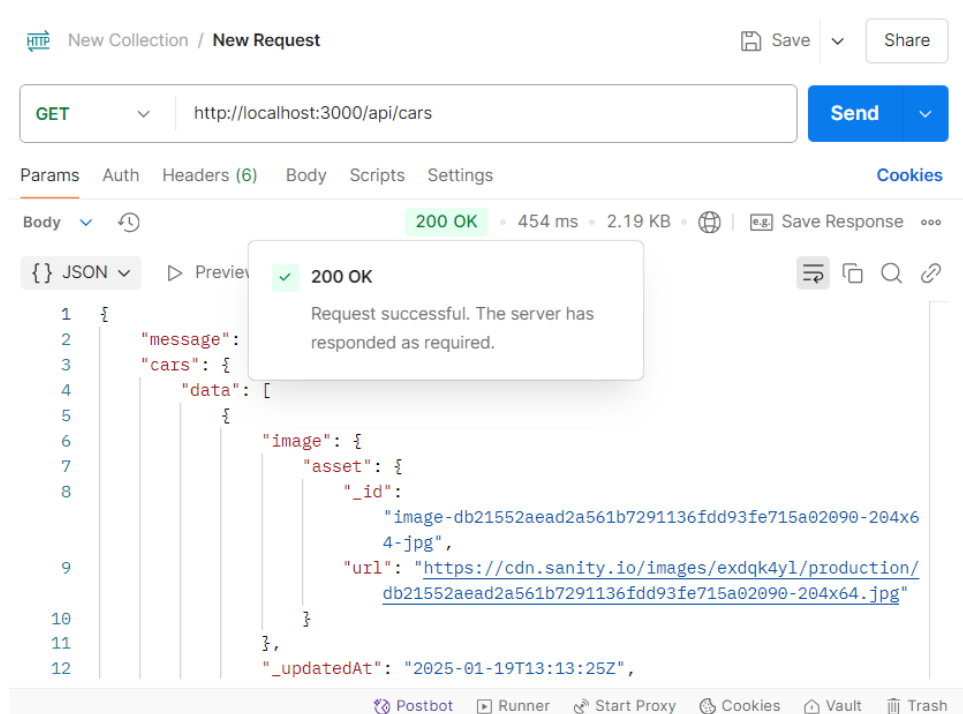
- **Dynamic Routing:**

Implementation of dynamic routing through slug, displaying individual product details correctly.



2. Testing Tools:

- **API Response Testing with Postman:**



Step 2: Error Handling

1. Error Messages:

Try-Catch blocks are handling errors messages when the response from API fails.

```
try {
  const cars = popularCarList && recommendedCarList;
  return NextResponse.json({ message: "OK", cars }, { status: 200 });
} catch (error) {
  return NextResponse.json(
    { message: "Error, cars not found" },
    { status: 500 }
  );
}
```

2. Fallback UI:

To display alternative content when data is unavailable.

```
if (carsToDisplay.length === 0) {
  return <div>No cars available</div>;
}
```

Step 3: Performance Optimization

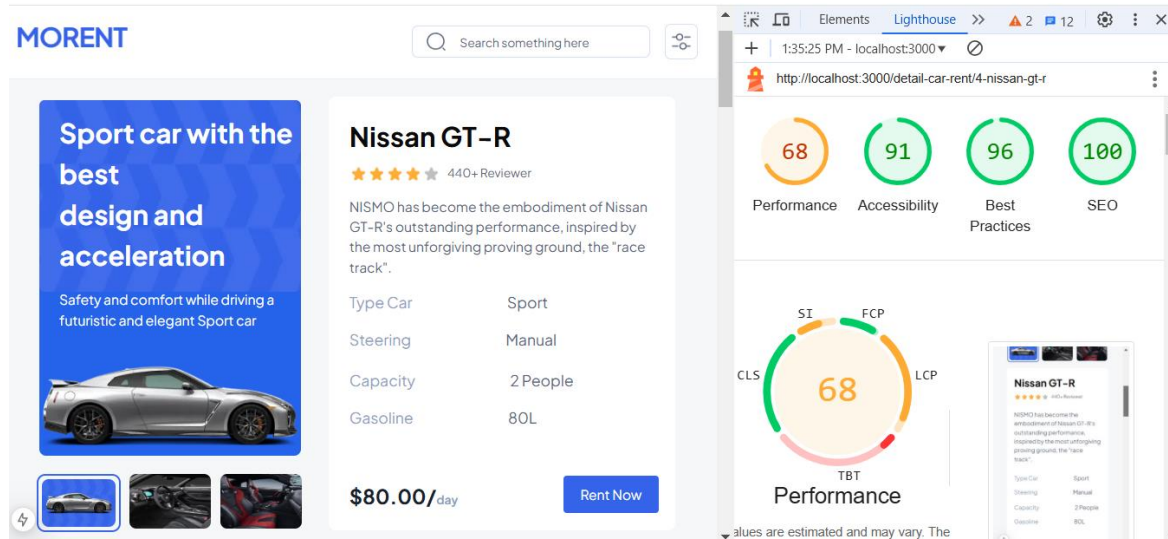
1. Optimized Assets:

Lazy loading is enabled for images to improve page performance.

```
<img alt="car-image" loading="lazy" width="300" |
```



2. Performance Analysis with Lighthouse:



3. Tested Load Time:

1.9 - 2.5 seconds

Step 4: Cross-Browser & Device Testing

1. Browser Testing:

Consistent rendering, functionality and responsiveness on different browsers like Chrome, Safari, and Edge.

2. Device Testing:

Manually tested on a mobile device to ensure responsiveness.

Step 5: Security Testing

Secure API Communication:

Sensitive data like API KEYS stored in secure environment variables.

Step 6: User Acceptance Test (UAT)

Simulated Real-World Usage:

- Performed tasks like browsing cars, adding them to the wish list.
- No usability issues found.

Test Cases:

Test Case ID	Feature/Module	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity	Remarks
TC001	Car Listing	Verify all products display correctly.	Navigated to the pages having car listings.	All products are displayed correctly.	Got the expected results	Passed	Low	Added fallback UI if no products.
TC002	Search Functionality	Verify the search feature returns relevant results.	1. Entered "Audi" in the search bar.	Only Audi displayed in the results.	Got the expected results	Passed	Low	Added "No cars found" fallback.
TC003	Filter Functionality	Verify filters work as intended.	Applied category filters	Products are sorted by category.	Cars appeared with selected categories	Passed	High	Added "No cars found" fallback if doesn't match.
TC004	Wishlist Operations	Verify adding cars to the wish list.	1. Added a car to the wish list. 2. Verified it appears in the wishlist.	The product appears in the wishlist.	Got the expected results	Passed	Medium	Added "Your wishlist is empty!" fallback if no cars in the wishlist.
TC005	Dynamic Routing	Verify product detail pages	1. Clicked on a car.	Product detail page	Got the expected	Passed	Medium	Rendered successfully

		load correctly.	2. Verified the detail page loads with relative information	loads with correct details.	ed results			
TC006	Error Handling	Verify error message on API failure.	1. Simulate an API failure by disconnecting the internet. 2. Check the error message displayed.	Displays “localhost is currently unable to handle this request.” with status 500	Expected : “Error, cars not found ” status: 500	Failed	High	Failed to get the expected message.
TC007	Performance Optimization	Verify page load time is under 2 seconds.	1. Run a Lighthouse audit. 2. Measure load time and performance score.	Load time is was around 1.9s	Got the expected results	Passed	High	Page loaded under required time unexpectedly.
TC008	Responsive Design	Verify the site is responsive on different devices.	1. Test the website for desktop, tablet, and mobile.	The layout adjusts correctly on all devices.	Got the expected results	Passed	Medium	Worked well.
TC009	Cross-Browser Testing	Verify functionality across major browsers.	1. Test on Chrome and Edge.	Site works consistently on all browsers.	Got the expected results	Passed	Medium	Worked well.