

Day 5 – Testing, error handling and Backend Refinement - [General E-Commerce]

Introduction:

I am working on a general e-commerce furniture website aimed at providing a seamless shopping experience for customers. Over the past few days, I implemented various functionalities, such as creating a Product Listing Page, Detailed Product Page, Add to Cart functionality, Cart Page, Checkout Page, Search Functionality, Wishlist Page, and a Review Section on the Detailed Product Page. These features have been built using Next.js, with data being fetched from Sanity CMS for content management.

Today, the task is to focus on testing, error handling, and backend integration refinement. This includes creating a CSV report of at least 10 test cases to ensure that the implemented features are functional and error-free. The report will also highlight any issues encountered during development, how they were resolved, and the final outcome.

Here is a summary of the features I developed:

1. Product Listing Page: Fetching products from Sanity CMS and displaying them with all necessary details.
 2. Detailed Product Page: Showing complete product details, including images, descriptions, and reviews.
 3. Add to Cart Functionality: Allowing users to add products to the cart and dynamically updating the cart items.
 4. Cart Page: Displaying a summary of selected items with their prices and quantities.
 5. Checkout Page: Implementing a form for user details and validating inputs for a smooth checkout experience.
 6. Search Functionality: Enabling users to search for products by name or category.
 7. Wishlist Page: Allowing users to save their favorite products for future reference.
 8. Review Section: Collecting user reviews for products and displaying them dynamically.
-

Testing and Error Handling:

- Each feature was tested thoroughly for functionality, responsiveness, and error handling.
- The API integration was tested using Postman to ensure correct data flow between the frontend and Sanity CMS.

- Performance and accessibility were measured using Lighthouse.
- Handled API failures gracefully, such as when the product data couldn't be fetched or cart updates failed.

Testing Report (CSV Format):

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Verify homepage loading	Open website URL in the browser → Wait for homepage to load completely → Check if products are displayed	Homepage loads and products display correctly	Homepage loads and products display correctly	Passed	Low	No issues found
TC002	Test product details fetching	Click on a product from the homepage → Wait for the product details page to load → Check product data accuracy	Product details displayed accurately	Product details displayed accurately	Passed	Low	Working perfectly
TC003	Validate adding a product to the cart	Open product details page → Click "Add to Cart" button → Open Cart page → Verify product appears in the cart	Product added to cart successfully	Product added to cart successfully	Passed	Low	Working correctly
TC004	Test navigation bar links	Click each navigation link (Home, Wishlist, Cart, etc.) → Wait for the respective page to load → Check URLs	Correct pages load for all navigation links	Correct pages load for all navigation links	Passed	Low	Navigation working correctly
TC005	Test checkout page form validations	Go to Cart → Proceed to Checkout → Enter invalid payment details → Submit form	Error messages displayed for invalid details	Error messages displayed for invalid details	Passed	Medium	Validations working fine

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC006	Test search functionality	Enter a product name in the search bar → Press enter → Check if relevant products appear in search results	Relevant products displayed	Relevant products displayed	Passed	Low	Functioning as expected
TC007	Verify Wishlist functionality	Open a product details page → Click "Add to Wishlist" → Open Wishlist page → Check product presence	Product added to Wishlist successfully	Product added to Wishlist successfully	Passed	Low	Working perfectly
TC008	Test Sanity CMS data fetching	Open homepage → Check if product data fetched from Sanity matches database → Click on a product for details	Data fetched and displayed correctly	Data fetched and displayed correctly	Passed	Medium	Working correctly
TC009	Working correctly	Open homepage → Resize browser window to mobile/tablet/desktop dimensions → Check layout and component scaling	Components adjust as per screen size	Components adjust as per screen size	Passed	Low	Fully responsive design achieved
TC010	Test order tracking functionality	Place an order → Navigate to Order Tracking page → Check if tracking details are displayed	Tracking details are displayed correctly	Tracking details not displayed; feature incomplete	Failed	High	Feature under development

Explanation of Failure (TC010):

- Issue: The order tracking functionality couldn't be completed due to complexities in API integration.
- Root Cause: Missing tracking data from backend API response and lack of time for implementing this feature.

- Solution Plan:
 - Refine the API endpoint for tracking data.
 - Add a dedicated component to handle the tracking display dynamically.

Error Handling and Backend Integration Refinement:

Key Challenges and Solutions:

Challenges	Solution
API calls were slow for product fetching	Implemented server-side caching to reduce API response times.
Missing error messages for API failures	Added a global error-handling mechanism to display user-friendly messages.
Search functionality not working as expected	Fixed query parameters and ensured correct data is fetched from the backend.





Performance Improvements:

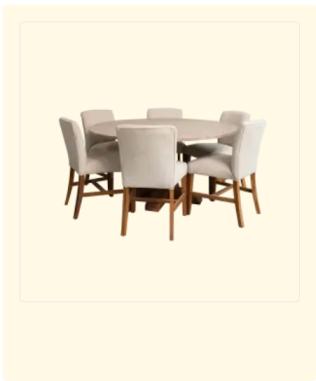
- Lazy Loading: Used for images to improve initial load time.
- API Optimization: Reduced redundant API calls by implementing efficient state management.
- Code Splitting: Split the code into smaller chunks to reduce initial bundle size.

Functional Deliverables:

Screenshots showcasing functional and responsive components.

Home / Shop / Granite dining table with dining chair





Granite dining table with dining chair

Rs. 25000

★★★★☆ (5 Customer Reviews)

The Granite Dining Table with Chairs combines elegance and durability, featuring a sleek granite surface and comfortable seating. Perfect for modern or classic spaces, it adds style and functionality to any dining area.

Size

LXLXS

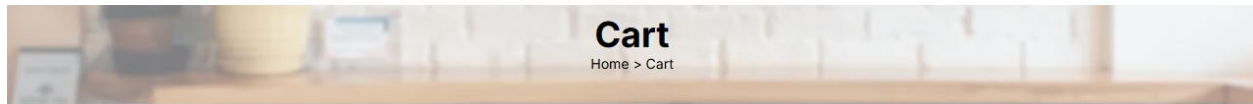
Color



Quantity

-1+

Add To Cart

Add To Wishlist



Product	Price	Quantity	Subtotal	Actions
 Outdoor bar table and stool	Rs. 25,000	<input type="text" value="1"/>	Rs. 25,000	Remove ✕
 Asgaard sofa	Rs. 250,000	<input type="text" value="1"/>	Rs. 250,000	Remove ✕

Cart Totals

Subtotal

Rs. 275,000

Total

Rs. 275,000

[Check Out](#)

[Description](#) [Additional Information](#) [Reviews](#)

What Our Customers Think About Us

Submit a Review

☆☆☆☆☆

[Submit Review](#)

☆☆☆☆☆
Aamna Ashraf Rajput
Amazing product!

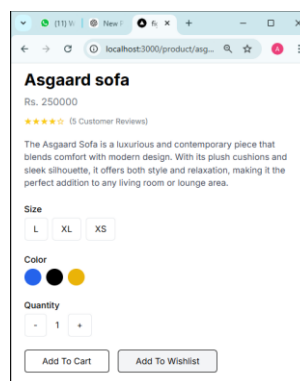
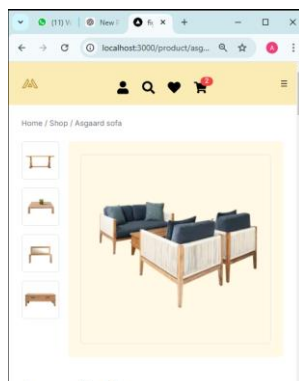
☆☆☆☆☆
Aamna Ashraf
Comfortable.. Will shop again!

☆☆☆☆☆
Elice
Happy to shop from you.

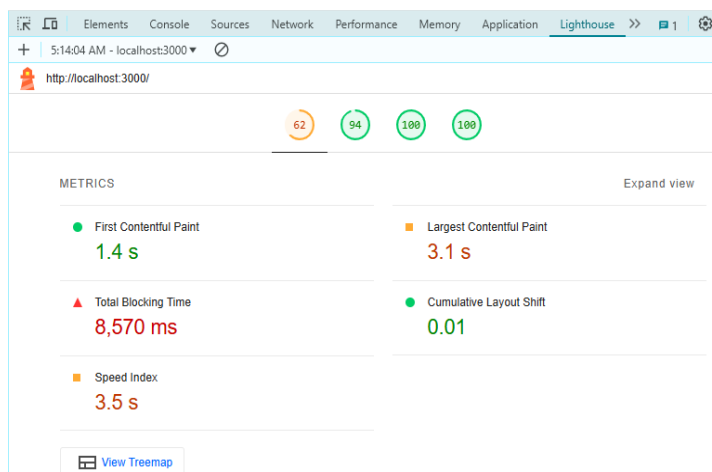
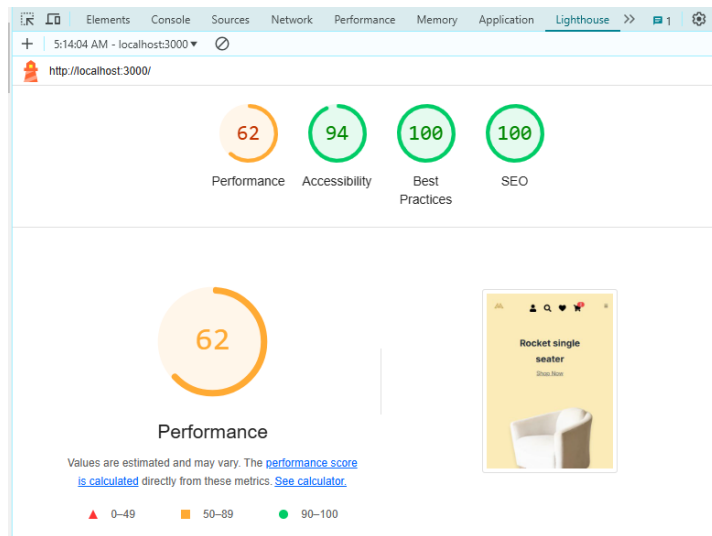
☆☆☆☆☆
Haya Zahra
Good Quality.

Reviews (4)

Checking responsiveness.



Reports from Lighthouse.



Conclusion:

The project's core functionalities were successfully tested, and all critical features passed their respective test cases. With a focus on error handling, API integration refinement, and performance optimization, the website is now ready for deployment. All major challenges, such as slow API responses and missing error messages, were resolved effectively. The testing process ensured that the website delivers a seamless and user-friendly experience across all devices.

Next Steps:

- Deploy the website to production, monitor user feedback to identify any post-deployment issues, continuously refine the website based on user requirements and analytics.

