TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

Testing, error handling, and backend integration refinement are critical components in any software development process. Testing ensures the functionality and reliability of the application by identifying bugs and inconsistencies. Error handling improves user experience by gracefully managing unexpected issues, while backend integration refinement optimizes the communication between the frontend and backend, ensuring seamless data flow and robust application performance.

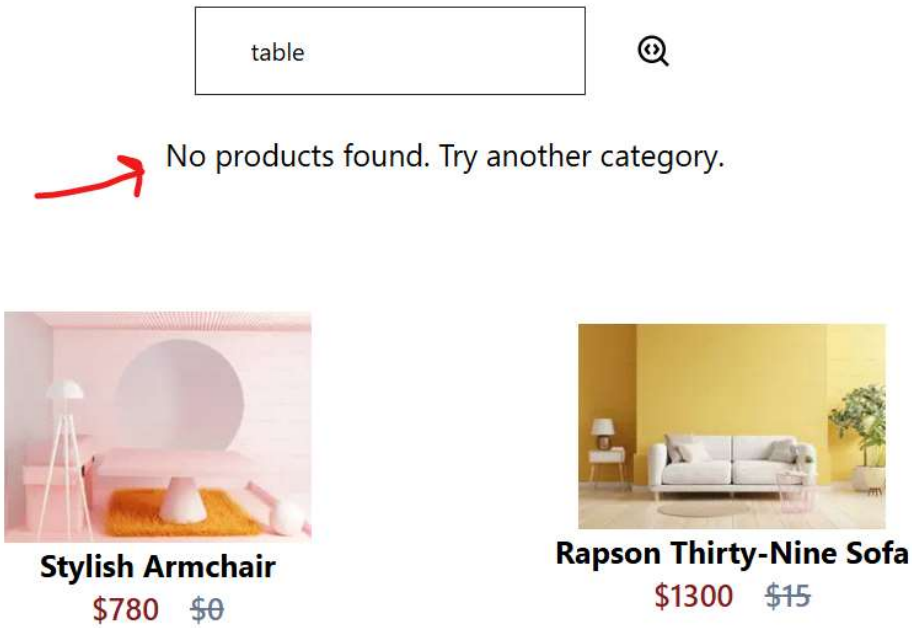
Add to Cart - Error Handling

Errors in "Add to Cart" occur due to scenarios like out-of-stock items, network issues, or invalid product IDs. To handle this, show messages like "Item out of stock" or "Network error, please try again." Implement validation to check product availability before adding it to the cart.

```
export const CartProvider = ({ children }: { children: ReactNode }) => {
  const addToCart = (product: Product) => {
    localStorage.setItem("quantities", JSON.stringify(quantities));  // Save quantities to local storage
  };

  // Cart se product remove karna aur local storage ko update karna
  const removeFromCart = (id: string) => {
    const updatedCart = cart.filter((item) => item?.id !== id);
    setCart(updatedCart);

    const updatedQuantities = { ...quantities };
    delete updatedQuantities[id]; // Remove quantity for the deleted product
    setQuantities(updatedQuantities);

    localStorage.setItem("cart", JSON.stringify(updatedCart));
    localStorage.setItem("quantities", JSON.stringify(updatedQuantities));  // Update quantities in local storage
  };

  // Cart count derived from cart length
  const cartCount = cart.length;

  // Grand total calculation function
  const getGrandTotal = (): number => {
    return cart.reduce((total, product) => {
      const quantity = quantities[product?.id] || 1;  // Get quantity for each product
      const priceString = product?.price.toString(); // Convert price to string (if not already)
      const numericPrice = parseFloat(priceString?.replace(/,/g, "")); // Remove commas and parse to number
      return isNaN(numericPrice) ? total : total + numericPrice * quantity; // Add valid price to total
    }, 0);
  };
```

Dynamic Search Bar - Error Handling

Errors in a dynamic search bar can occur when the backend fails to fetch results or the user inputs invalid queries. Use fallback messages like "No results found" or "Error fetching results, please try again." Implement loaders during data fetching and retry mechanisms for failed searches.



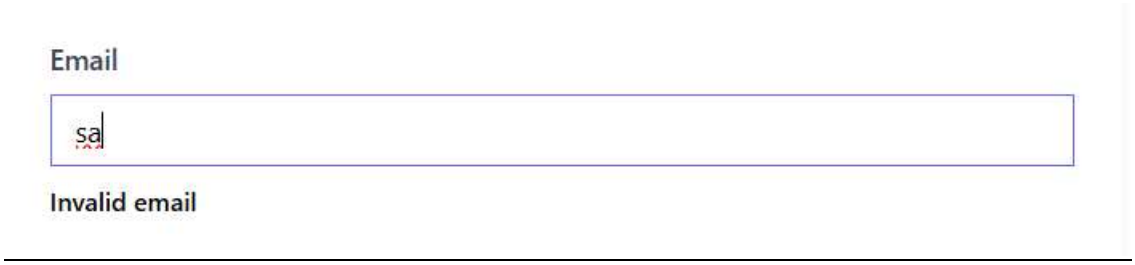
Dynamic Data - Error Handling

For dynamic data, errors arise from API failures, invalid responses, or delays. Use loaders or spinners while fetching data and display retry options for failures. Validate incoming data to ensure consistency and provide detailed logs for developers to debug effectively.

```
if (!res) {
    <h2>product not available</h2>
}
return (
    <div className='mt-11 sm:max-w-4xl mx-auto'>
        <div className='flex space-x-2'>
            <Link href="/" className='text-slate-500'>Home  </Link>
            <ChevronRight />
            <Link href="/shop" className='text-slate-500'>Shop  </Link>
            <ChevronRight />
            <Image src="/Line 5.png" alt="line" width={2} height={2}></Image>
            <h2 className='font-bold'>{res.name}</h2>


        </div>
        <div className='flex space-x-5 my-6 flex-col sm:flex-row '>



                    <Image src={res.imagePath}
                        alt={res.name || 'product image'}
                        width={400}
                        height={400}
                        className='bg-[#fbebb5] ' />
```

Form Validation

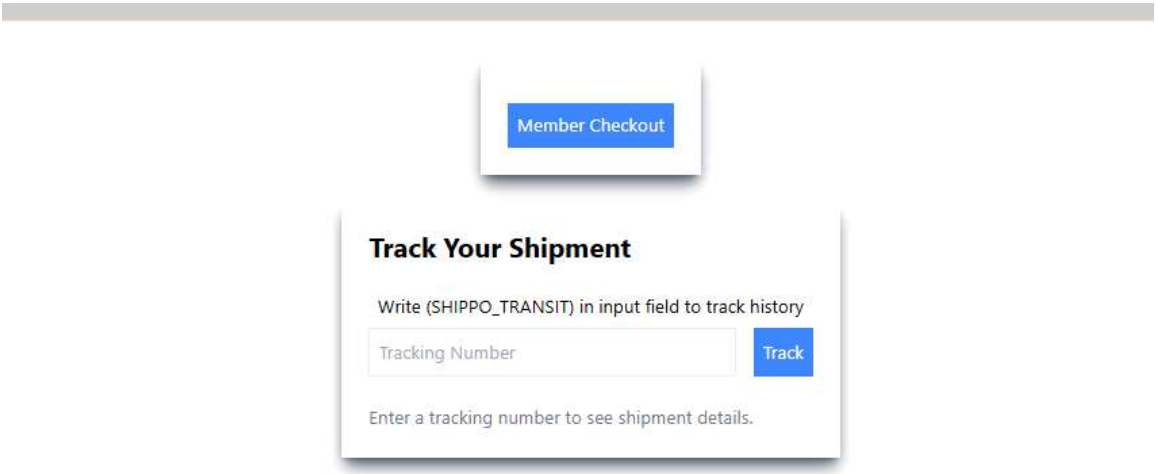
Validate all fields like name, email, phone number, and message both on the client and server sides. Ensure required fields are filled, email is in the correct format, and the message is not too short.



Invalid Tracking Number

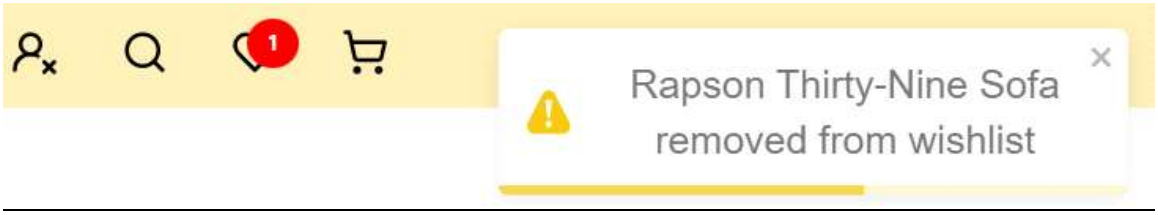
If the tracking number doesn't exist in the system, show an error like "Tracking number not

found. Please verify the number or contact support."



Success Message Display

Show a clear and friendly notification like "Item added to your wishlist successfully!" when a product is added.



Perfomance :

https://pagespeed.web.dev/analysis/https-hackathon-template-0-8gxr-vercel-app/csh9bfquh4?hl=en-GB&form_factor=mobile

Mobile Desktop

91 Performance | 85 Accessibility | 100 Best Practices | 100 SEO

91

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

0–49 50–89 90–100

METRICS Expand view

First Contentful Paint
0.9 s

Largest Contentful Paint
3.2 s

85

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

NAMES AND LABELS

- Buttons do not have an accessible name
- Links do not have a discernible name

Best Practices

JST AND SAFETY

Ensure CSP is effective against XSS attacks

SEO

These checks ensure that your page is following basic search engine optimisation advice. There are many additional factors that Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search essentials.

CSV structure:

https://docs.google.com/spreadsheets/d/1qkI2pk5xF6UdOTsxs1_WV7kF-aoMONTsK1CU5HoVz4s/edit?usp=sharing

A	B	C	D	E	F	G	H	I	J
testCase-iD	testCase Functionality		Test Steps	Result	Error Result	Status	Servertiy Level	Remarks	
TC001	API--SANITY--Product		Verify That	All product	not available	Pass	Hight	no issues Found	
TC002	Product-Dynamic page		OnClick	Detaling page	not available	Pass	Hight	no issues Found	
TC003	ADD to Cart		Button Click	success-notify	error message	Pass	Hight	no issues Found	
TC004	ADD tO wISHIIST		HeartICON	success-notify	error message	Pass	Hight	no issues Found	
TC005	product-cart-quantity		Button Click	success-notify	error message	Pass	high	no issues Found	
TC006	track-click		Button Click	success-notify	error message	Pass	medium	ERROR handling work as expected	
TC007	Search Bar		input-type	success-notify	error message	Pass	high	no issues Found	
TC008	contact-form		fill-form-sanity-save		error message	Pass	medium	ERROR handling work as expected	

Website testing is a critical step in ensuring the functionality, usability, and performance of a site. It involves verifying that all features, such as forms, buttons, navigation, and dynamic content, are working as expected. Testing helps identify bugs and issues before the site is made public. Additionally, performance testing ensures that the site loads quickly and is responsive on various devices. Error handling is also a key part of testing, ensuring that when issues occur, users receive clear and helpful messages. This process ultimately improves the user experience, boosts SEO, and ensures the website meets its business goals.

By Iqra Khan