Day 5 - Testing, Error Handling, and Backend Integration Refinement (Comforty)

Testing Report:

Test Case ID	Test Case Description	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Validate product listing page	Products should load dynamically without errors.	Products displayed correctly.	Passed	Low	Fully functional.
TC002	Validate product detail page	Product detail page should load with accurate details.	Product detail page loaded with accurate details.	Passed	Low	Works seamlessly.
TC003	Check category functionality	Relevant products should appear in the selected category.	Relevant products appeared as expected.	Passed	Low	Functional with no issues.
TC004	Test search bar functionality	Dropdown with results or "No products found" message.	Dropdown with results or "No products found" message.	Passed	Low	No issues observed.
TC005	Validate cart functionality	Updates cart details correctly and prevents invalid actions.	Updates cart details correctly and prevents invalid actions.	Passed	Medium	Fully functional.
TC006	Validate reviews component	Reviews should be saved and visible for respective products.	Reviews saved and visible for respective products.	Passed	Medium	Fully functional.
TC007	Test footer and header links	Navigation should be smooth and consistent across pages.	Navigation links function correctly.	Passed	Low	Works seamlessly.
TC008	Validate notifications	Notifications should appear correctly for each action.	Notifications display as expected.	Passed	Medium	No issues.
TC009	Validate contact form	Valid forms should submit, and invalid ones should alert.	Works as expected with accurate validation.	Passed	Low	Validation is robust.
TC010	Validate pagination	Products should load smoothly when	Pagination works as intended.	Passed	Low	Fully functional.

Test Case ID	Test Case Description	Expected Result	Actual Result	Status	Severity Level	Remarks
		switching pages.				

Error Handling:

1. Add Error Messages:

Use `try-catch` blocks to handle API errors. Example:

```
try {
  const data = await fetchProducts();
  setProducts(data);
} catch (error) {
  console.error("Failed to fetch products:", error);
  setError("Unable to load products. Please try again later.");
}
```

2. Fallback UI:

When data is unavailable, provide a fallback UI. Example:

```
if (error) {
  return <p>{error}</p>;
} else if (!products.length) {
  return <p>No products available at the moment.</p>;
}
```

Performance Testing (via Lighthouse):

1. Mobile:

Ensure your website is mobile-friendly and loads efficiently on mobile devices.

- **Mobile Performance Score**:
- **Accessibility Score**:

2. Desktop:

Ensure desktop performance is optimized for a smooth user experience.

- **Desktop Performance Score**:
- **Accessibility Score**:

Submission Checklist:

1. **Functional Testing**: ✔
2. **Error Handling**: ✔
3. **Performance Optimization**: ✔
4. **Cross-Browser and Device Testing**: ✔
5. **Security Testing**: ✔
6. **Documentation**: ✔