**DAY 5 - TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT**

**Step 1: Functional Testing**

**1. Test Core Features**

* **Product Listing**: Ensure products are displayed correctly on the homepage and product listing pages.
* **Filters and Search**: Test product filters (e.g., category, price range, etc.) and search functionality to ensure accurate results.
* **Cart Operations**: Test adding, updating, and removing items from the shopping cart.
* **Dynamic Routing**: Check that each product's individual detail page loads correctly when accessed directly.

**2. Testing Tools**

* **Stripe**Test API responses to ensure they are returning expected data for endpoints like /products, /checkout, etc.
* **React Testing Library**: Write unit tests for React components such as ProductCard,

**Step 2: Error Handling**

**1. Add Error Messages**

Use try-catch blocks to handle errors from API calls or any asynchronous operations. Display useful error messages for the user.

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**

Display alternative content or loading indicators when data is unavailable. For example, show a "No items found" message for an empty product list or display a loading spinner while waiting for product data.

Example Fallback UI:

js

if (loading) return <div>Loading...</div>;

if (error) return <div>{error}</div>;

if (products.length === 0) return <div>No items found</div>;

**Step 3: Performance Optimization**

**1. Optimize Assets**

* Compress images using tools like **TinyPNG** or **ImageOptim** to reduce the file sizes.
* Use **lazy loading** for large images or assets (e.g., images on product detail pages).

Example of Lazy Loading:  
  
<Imageg src="product-image.jpg" loading="lazy" alt="Product" width=  
“96” height“96” />

Following these steps will help to ensure that **Quollex** is ready for production, with robust functionality, optimized performance, and secure operations.