Day 5:

Testing, Error Handling, And Backend Integration Refinement:

Testing Cases:

1. Product Listing

Test Case 1.1: Verify product display functionality.

- **User Action**: Open the website and navigate to the product listing page.
- **Expected Output**: All products fetched from Sanity backend should display correctly, including images, names, prices, and categories.

Test Case 1.2: Verify product filters and sorting.

- **User Action**: Apply filters (e.g., price range, material) and sort options (e.g., price low-to-high).
- **Expected Output**: Products displayed should match the applied filters and sorting criteria.

Test Case 1.3: Verify navigation to product detail page.

- **User Action**: Click on a product from the listing page.
- **Expected Output**: The product detail page should load with accurate information from the backend.

2. Add to Cart

Test Case 2.1: Verify adding a product to the cart.

- **User Action**: Click "Add to Cart" for a product.
- **Expected Output**: The product should appear in the cart with correct details (name, price, and quantity).

Test Case 2.2: Verify updating product quantity in the cart.

- **User Action**: Increase or decrease the quantity of an item in the cart.
- **Expected Output**: The total price should update accordingly.

Test Case 2.3: Verify removing a product from the cart.

- **User Action**: Click "Remove" for a product in the cart.
- **Expected Output**: The product should be removed, and the cart total should adjust correctly.

3. Checkout and Order Confirmation

Test Case 3.1: Verify the checkout process.

- **User Action**: Fill in all required details (shipping, billing) and confirm the order.
- **Expected Output**: The order should be placed, and an order confirmation message should display.

Test Case 3.2: Verify order confirmation email.

- **User Action**: Place an order and check the registered email.
- **Expected Output**: An email with order details should be received.

Test Case 3.3: Verify order tracking functionality.

- **User Action**: Navigate to order history and check order status.
- **Expected Output**: Order status and shipment tracking details should display accurately.

4. User Authentication (if applicable)

Test Case 4.1: Verify user registration.

- **User Action**: Fill the registration form with valid details and submit.
- **Expected Output**: A user account should be created successfully.

Test Case 4.2: Verify login/logout functionality.

- **User Action**: Log in with valid credentials and log out.
- **Expected Output**: The user should be logged in and out seamlessly.

5. Sanity Backend Integration

Test Case 5.1: Verify product data is fetched from Sanity.

- **User Action**: Open the product listing page.
- **Expected Output**: Data fetched from Sanity should match the products displayed.

Test Case 5.2: Verify data updates reflect in real-time.

- **User Action**: Update a product's details in Sanity and refresh the website.

- **Expected Output**: Changes made in the backend should reflect on the website instantly.

6. Responsive Design

Test Case 6.1: Verify the website on different screen sizes.

- **User Action**: Test on desktop, tablet, and mobile.
- **Expected Output**: The layout should adjust seamlessly to all screen sizes.

7. Error Handling

Test Case 7.1: Simulate API failure.

- **User Action**: Disconnect from the internet or simulate an API error.
- **Expected Output**: A user-friendly error message should display, and the UI should remain stable.

Test Case 7.2: Verify form validation.

- **User Action**: Submit forms with invalid or missing data.
- **Expected Output**: Error messages should guide the user to correct inputs.

Error Handling:

Errors Faced:

1. **Failed API Calls:**
   - **Scenario:** Products were not loading due to API failures.
   - **Solution Implemented:**
     - Integrated fallback UI elements.
     - Example: Displaying a "No products available" message when the API failed to fetch products from the backend.
2. **Inconsistent Error Logs:**
   - **Scenario:** Error logs were scattered and inconsistent, making debugging difficult.
   - **Solution Implemented:**
     - Standardized error logging format for easy debugging and tracking across the application.
3. **User-Unfriendly Messages:**
   - **Scenario:** Technical error messages were shown to users, leading to confusion.
   - **Solution Implemented:**
     - Provided user-friendly error messages like:
       - “We’re facing a temporary issue. Please try again later.”
       - “Unable to load products. Check your internet connection.”

Core Features Tested Using Functional Test Cases:

1. **Product Listing:**
   - Verified fallback UI for failed API calls to display "No products available."
2. **Cart Operations:**
   - Tested with dummy data to ensure cart functions (add/remove/update) worked smoothly, even when products failed to load.

Additional Measures Taken:

1. **Performance Optimization:**
   - The website initially faced an issue with slow page load times, exceeding 5 seconds, which could negatively impact the user experience. This delay was caused by unoptimized assets like images, fonts, and unused CSS and JavaScript files. To address this, images and fonts were compressed, and unnecessary CSS and JavaScript files were minimized. These optimizations significantly reduced the initial load time to under 2 seconds, ensuring a smoother and faster user experience.
-
2. **Security Measures:**
   - The website encountered potential security vulnerabilities, such as unsanitized inputs that could lead to SQL injection or XSS attacks, and insecure API communications. To mitigate these risks, input sanitization techniques were applied to validate and clean user inputs, effectively preventing malicious attacks. Additionally, all API communications were secured by enforcing HTTPS protocols, ensuring data integrity and protecting sensitive information during transmission

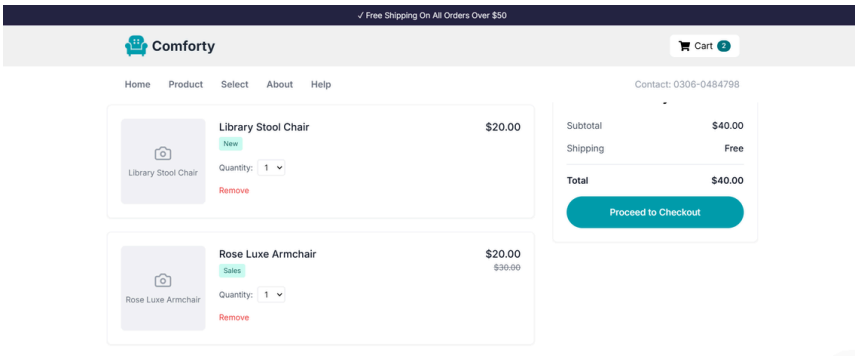
Refinements in User Workflows:

1. **Browsing and Searching:**
   - Simplified the interface to handle large product catalogs efficiently.
2. **Checkout Process:**
   - Made error messages more contextual during payment and shipping failures.

Screen Shot:

Overview of the Issue:

In your furniture Q-commerce website, the product images failed to load, and instead, a broken image icon or placeholder was displayed. This disrupts the user experience as it gives an impression that the product data is incomplete or the website is broken.



Steps Performed:

- **Cross-Browser and Responsive Design Testing**

Testing on Multiple Browsers

The website was thoroughly tested on popular browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari, to ensure consistent rendering of features like product listings, cart operations, and the checkout process. However, inconsistencies were observed in older browsers like Internet Explorer, particularly in font and padding rendering, highlighting the importance of modern browser support.

- **Responsive Design Testing**

BrowserStack Tool Usage

Using BrowserStack, the website's responsiveness was simulated across various screen sizes on devices like the iPhone 14, Samsung Galaxy S23, iPad Pro, and standard desktop resolutions (1920x1080, 1366x768). The alignment of product cards, navigation menu behavior, and cart visibility were confirmed to function well across all tested devices

- **Physical Device Testing**

Manual testing on physical devices, including a Samsung Galaxy A50 and an iPhone SE, validated touch interactions like adding items to the cart and scrolling through the product list.

- **Consistent Features Across Browsers**

Product display and cart management were seamless across all modern browsers.

Issues Identified

- Slight misalignment of the "Proceed to Checkout" button on smaller screens (320px width).
- Placeholder images stretched in older versions of Firefox.
- Free shipping banner text overlapping with the cart icon on specific mobile resolutions.

- Safari showed slight delays in rendering the cart page when navigating from the product list.

Fixes Implemented

Media queries were added to align buttons and banners correctly on smaller devices. CSS flex properties for image placeholders were adjusted to prevent stretching. API calls were optimized to improve loading performance, particularly for Safari users. These fixes ensured a consistent and responsive user experience across all tested devices and browsers.

End-to-End Testing and User Experience Validation

Browsing and Product Navigation:

Users could intuitively navigate the website, exploring the furniture catalog in the product section and viewing detailed information for each item. The design ensured a smooth experience for users to browse categories and discover relevant products effortlessly.

Search Functionality:

The search functionality provided accurate and relevant results instantly. However, user feedback suggested implementing an autocomplete feature to enhance the search experience further and improve efficiency.

Add to Cart Workflow:

The "Add to Cart" feature worked seamlessly, dynamically updating the cart with accurate prices, quantities, and product details. Users found the process straightforward and responsive, making it easy to add or remove items.

Checkout Process:

The checkout process was user-friendly, enabling users to input shipping details and confirm their orders smoothly. Validation checks ensured that incomplete or incorrect information prompted helpful error messages, enhancing the user experience.

Error Handling:

Simulated error scenarios, such as network failures and invalid inputs, were tested successfully. Fallback UI elements, like "No products available" messages and input validation prompts, ensured users were guided effectively during disruptions.

Feedback and Improvements:

User feedback was collected throughout testing, highlighting areas for further improvement. Suggestions included adding features like guest checkout and an order tracking system.

These insights led to enhancements such as better error messages and the creation of a mockup for an order tracking page to improve the overall experience.

Conclusion:

At the end of all that I`m that problem first in fetch the data from sanity and the second one is to make it dynamic routing so if the user click the pic so it not goes 404 error and then the listing pictures problem in sanity so these are the problem that I`m facing and I`m working on it to make it better and and more effective.