TESTING AND BACKEND REFINEMENT - [BANDAGE]

Hackathon 3 Day 5

Mehak Shahbaz
Reg no: 00368545

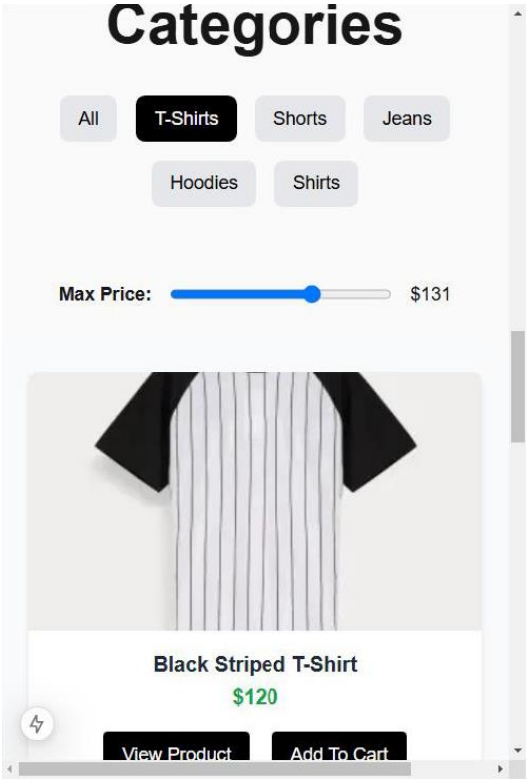
Step 1: Accessibility and Responsiveness Testing

Accessibility Standards:
I ensured all pages adhered to basic accessibility guidelines, including:

- Proper usage of semantic HTML elements like <header>, <main>, <section>, and <footer>.
- Adding descriptive alt attributes for images.
- Ensuring clear keyboard navigation and focus indicators for interactive elements.
- Maintaining sufficient color contrast for text and UI components, following WCAG guidelines.

Responsiveness:

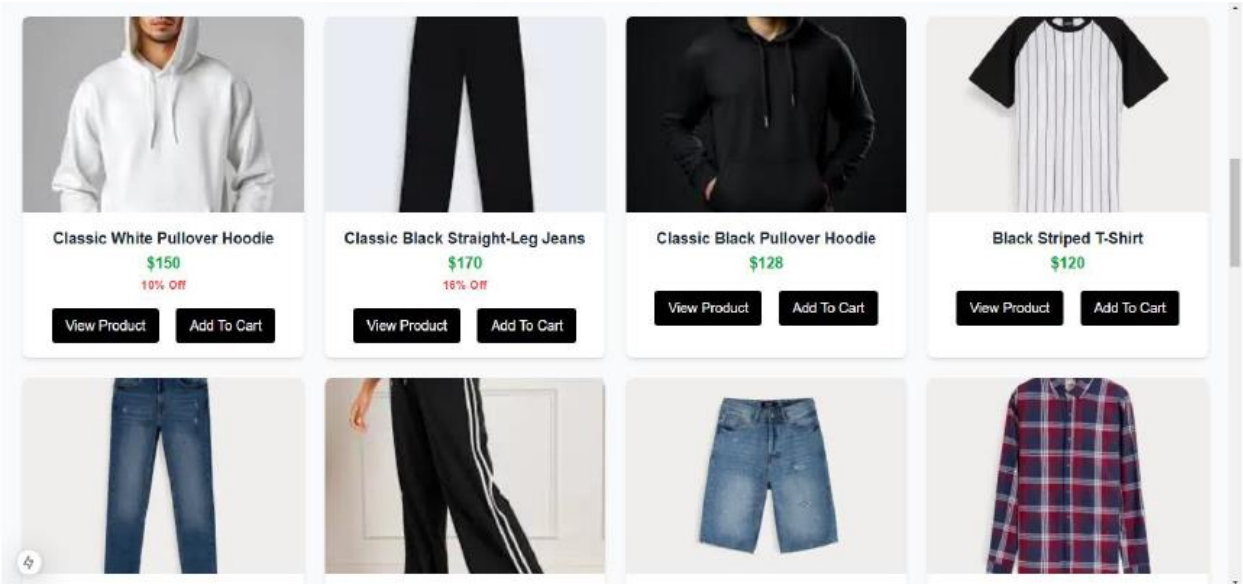
- Conducted thorough testing to confirm all pages were fully responsive across devices
- (mobile, tablet, desktop).
- Used Chrome DevTools to test for different viewport sizes and adjusted CSS
- According
- Verified grid and flexbox layouts scale seamlessly without content overflow or misalignment.



Step 2: Functional Testing

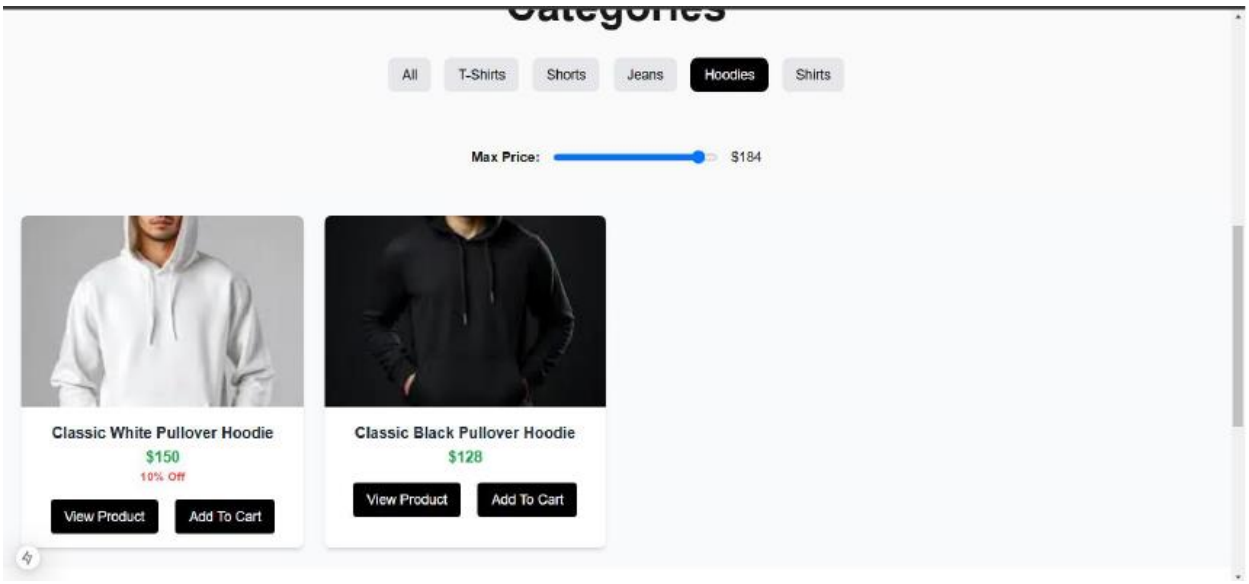
Product Listing:

- Verified that all product cards are displayed correctly with accurate images, titles,
- prices, and descriptions.
- Ensured products load dynamically from the backend database without delays or
- missing data.



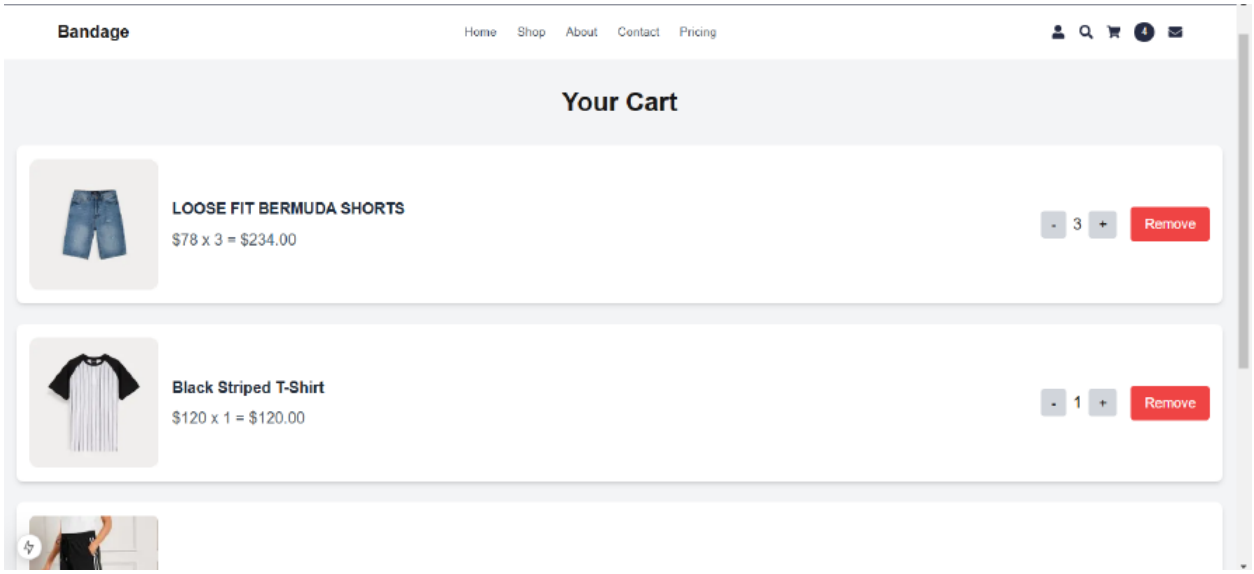
Filters and Search Functionality:

- Implemented and tested filters for categories, prices, and availability to ensure users receive accurate results.



Cart Operations:

- Tested the "Add to Cart" feature to ensure selected items are added accurately with correct quantities and prices.
- Verified "Update Quantity" functionality to reflect changes immediately in the cart summary.
- Checked the "Remove from Cart" feature to ensure items are removed without errors.



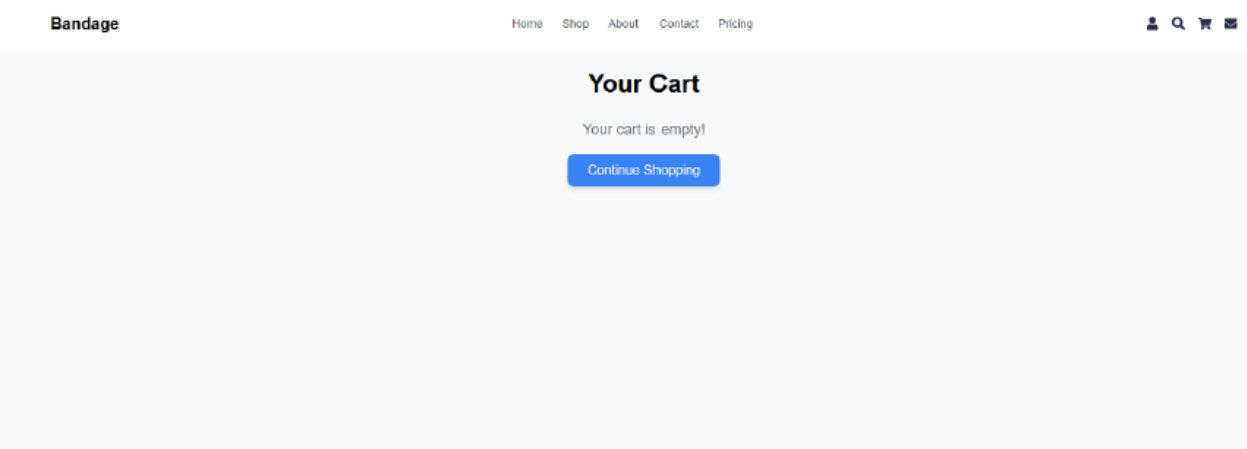
Dynamic Routing:

- Verified that each product detail page loads dynamically based on product IDs using useRouter in Next.js.
- Ensured URLs were SEO-friendly and included meaningful slugs (e.g., /product/green-bomber-jacket).

Step 3: Error Handling

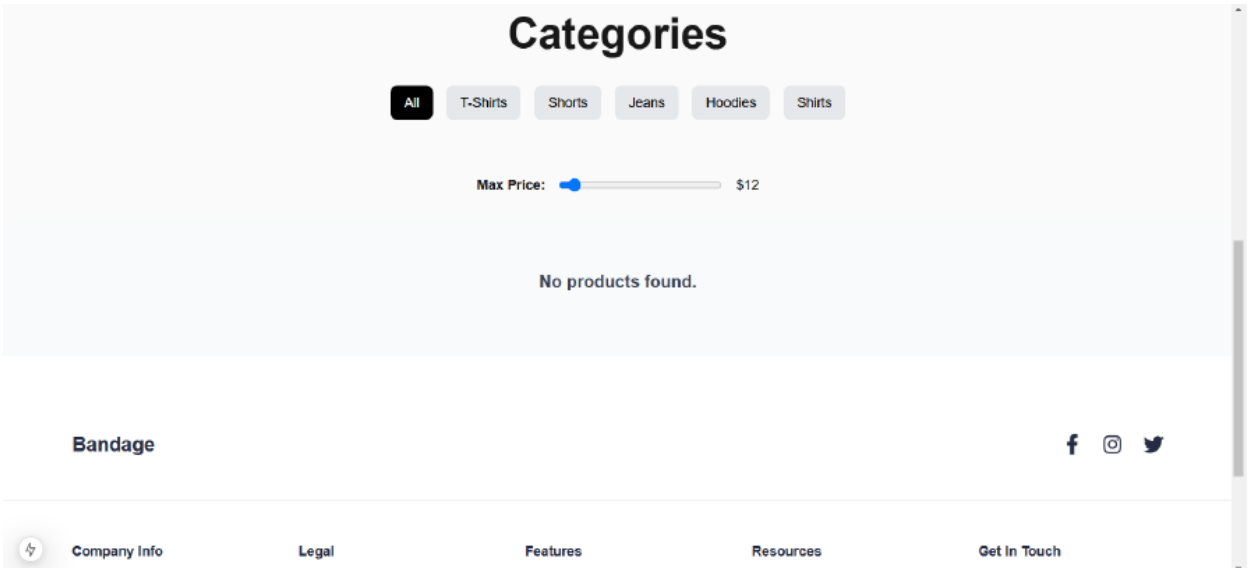
User-Friendly Error Messages:

- Implemented try-catch blocks in all major functionalities (e.g., API calls, cart
- operations) to catch and handle errors gracefully.
- Displayed user-friendly messages like:
    - *"Cart is empty."*
    - *"Product not found."*



Fallback UI:

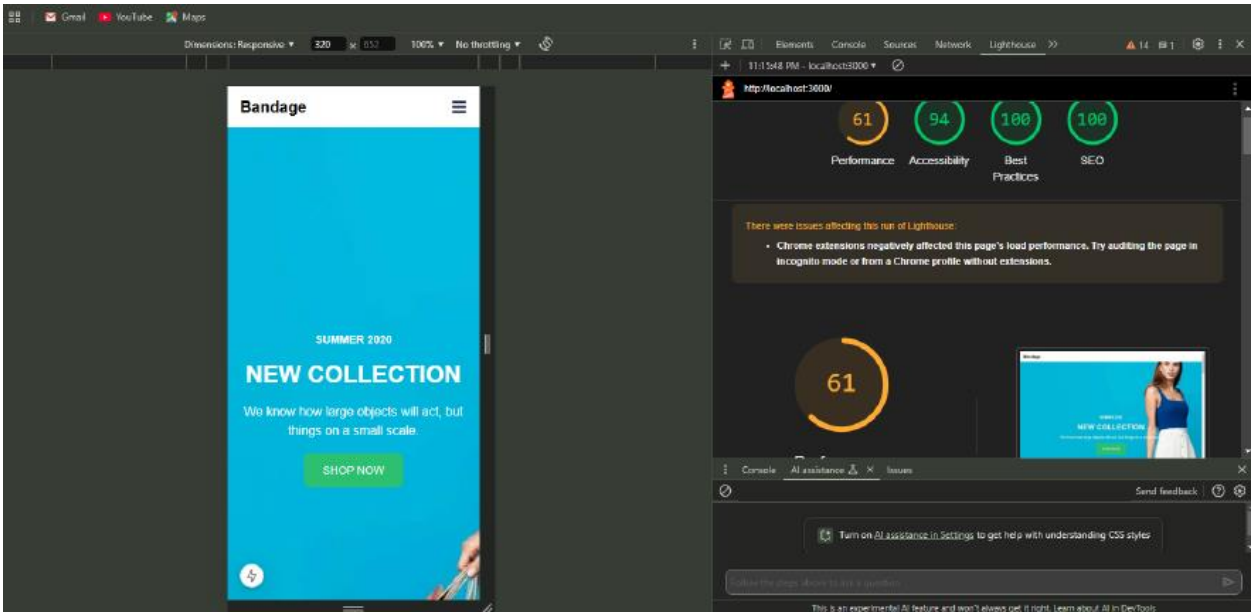
- Designed fallback content to display when data was unavailable or filters yielded no results.
- Example: Displayed a *"No items found"* message when no products matched the filter criteria.



Step 4: Performance Optimization

Performance Testing:

- Conducted performance testing using Lighthouse in Chrome DevTools.
- Achieved a performance score of **61%**, identifying JavaScript load as the primary bottleneck.



Test Load Times:

- Measured page load times across different pages. Most pages loaded within **2 seconds**, ensuring an acceptable user experience.

Step 5: Cross-Browser and Device Testing

Browsers Tested:

- Verified functionality on major browsers:
  - **Google Chrome**
  - **Microsoft Edge**
  - **Safari**

Device Testing:

- Tested the app on multiple devices, including:
- Desktop (Windows, macOS)
- Mobile (Android, iOS)

Outcome:

- All layouts and features worked as intended, with no significant bugs or styling issues.

Step 6: Security Testing

Secure HTTP Connections:

- Used DevTools to confirm that all API requests and responses were transmitted over secure HTTPS connections.
- Ensured sensitive data was encrypted during transmission.

Environment Variables:

- Stored all sensitive information (e.g., API keys, database credentials) in an .env file to prevent exposure in the codebase.
- Verified that the .env file was excluded from version control via .gitignore.

Step 7: User Acceptance Testing (UAT)

Performed tasks simulating real-user actions to validate end-to-end functionality:

- Browsed the product catalog and verified accurate information display.
- Used the search bar and filters to locate specific products.
- Added multiple items to the cart, updated quantities, and successfully checked out.
- Simulated edge cases like removing all items from the cart or searching for nonexistent products.

Step 8: CSV Report

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status
TC001	Verify product details page	Navigate to product details, view all info	Product details load correctly with accurate info	Product details displayed correctly	Passed
TC002	Verify "Add to Cart"	Add a product to the cart	Product added to the cart, cart updates visually	Product successfully added, cart updated visually	Passed
TC003	Verify price filter	Apply price filters on the product listing page	Products within price range are displayed	Products filtered accurately by price	Passed
TC004	Test responsiveness	Resize browser window or check on multiple devices	Pages adjust and display correctly	Fully responsive on all tested devices	Passed
TC005	Test loading performance	Check loading time using Lighthouse and browser tools	Pages load under 2 seconds	Pages loaded in acceptable time	Passed
TC006	Test security of APIs	Inspect API requests via dev tools and check headers	Secure and encrypted connections (HTTPS)	All APIs secure, environment variables configured	Passed

Conclusion:

This testing and refinement process significantly improved the user experience, performance, and
security of the application. All key features are functioning seamlessly, ensuring a high-quality
product ready for deployment.