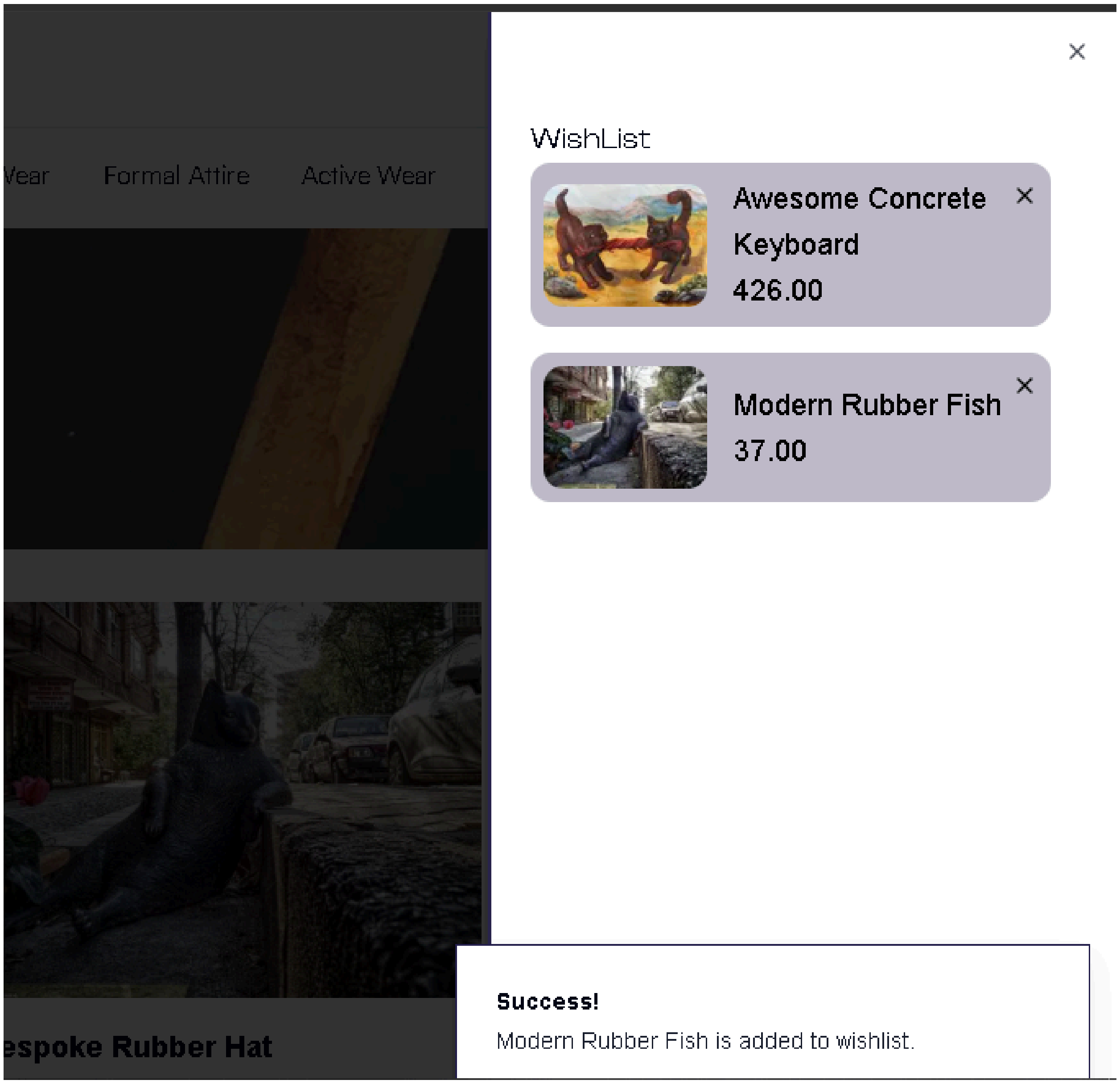
Day 5 - Testing and Backend Refinement - Avion

Step 1: Functional Testing:

1. Wishlist Functionality:



Awesome Concrete Keyboard
426.00

Modern Rubber Fish
37.00

Success!
Modern Rubber Fish is added to wishlist.

1. **Toast for Wishlist Toggle:**

- I added a toast notification for both adding and removing items to improve user feedback and clarity, letting users know their actions were successful.

2. **Simplified `toggleWishlist` Function:**

   - I switched to check by `id` only (ignoring color and size) with wishlist functionality since these details are typically cart-specific. This makes the function more efficient and straightforward.

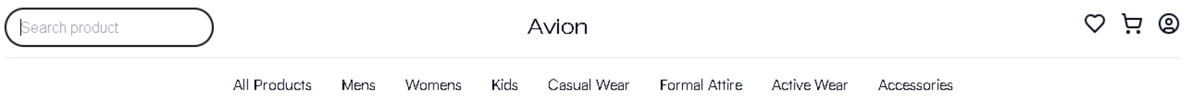
3. **Redirect Issue Fixed:**

   - Fixed Redirect Issue: I Used `event.preventDefault()` and `event.stopPropagation()` in the toggle function to prevent navigation when adding/removing items.

4. **Scrollable Wishlist in Sheet:**

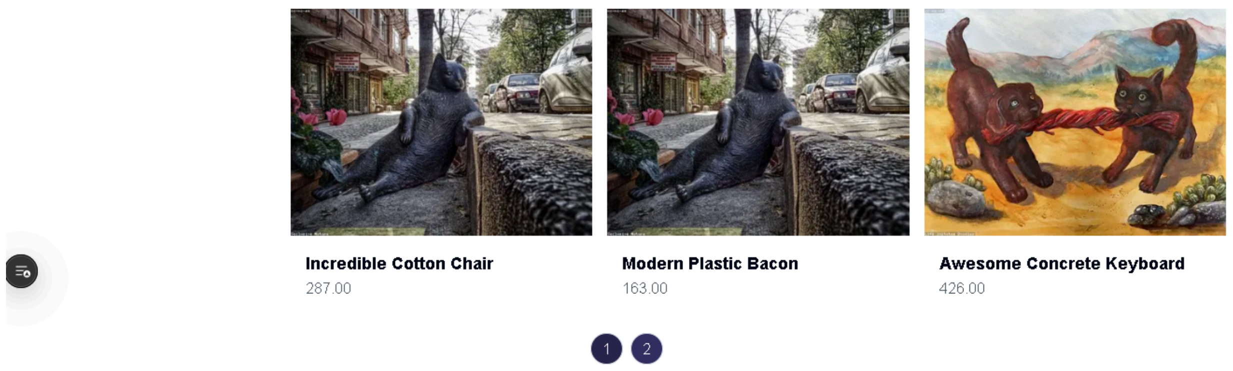
   - Implemented a scroll area within the wishlist sheet to enhance usability, especially for users with longer wishlists, allowing them to browse items seamlessly.

2. Search Products Functionality:



- **Responsive Search Product Function:** The input now appears only after clicking the search icon, saving space and enhancing the UI.
- **Adjusted Input Responsiveness:** Manually targeted smaller and medium screens to adjust the input width and maintain its horizontal position.
- **Header Icons Order:** Set the order of header icons to adjust automatically on smaller screens for better layout.

3. Pagination functionality:



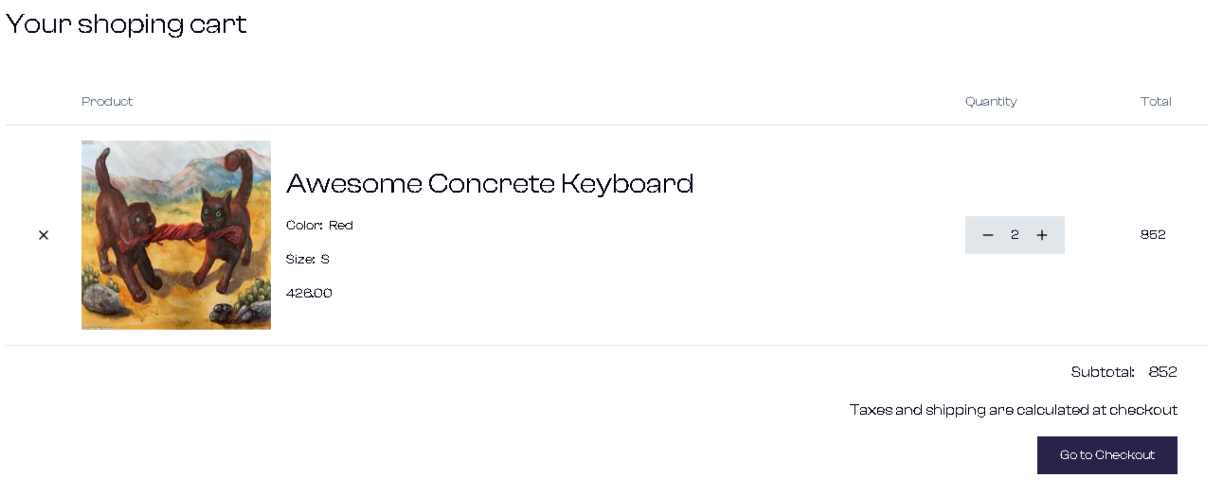
Incredible Cotton Chair
287.00

Modern Plastic Bacon
163.00

Awesome Concrete Keyboard
426.00

Pagination Testing: Tested pagination by navigating to the next and previous pages, ensuring smooth functionality with no errors.

4. Cart Functionality:



Your shoping cart

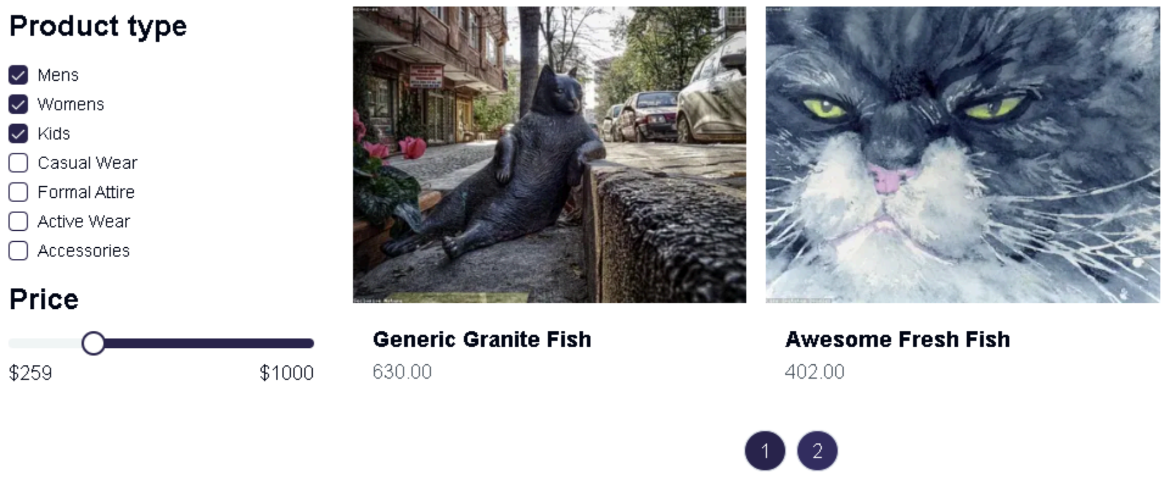
	Product		Quantity	Total
×		Awesome Concrete Keyboard Color: Red Size: S 426.00	− 2 +	852

Subtotal: 852

Taxes and shipping are calculated at checkout

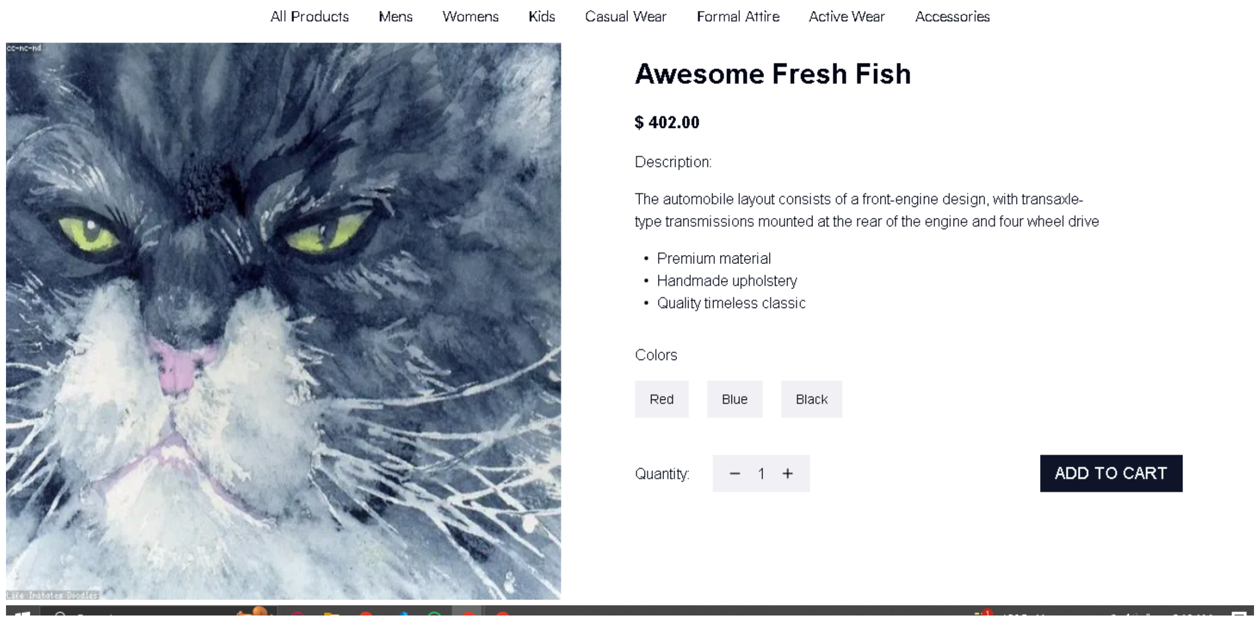
Cart Functionality Testing: Tested adding, removing, and updating item quantities in the cart, with no issues found.

5. Filters:



Product Filtering Testing: Tested filtering products by categories and prices, with everything working smoothly and no issues found.

6. Dynamic routing:



Awesome Fresh Fish

$ 402.00

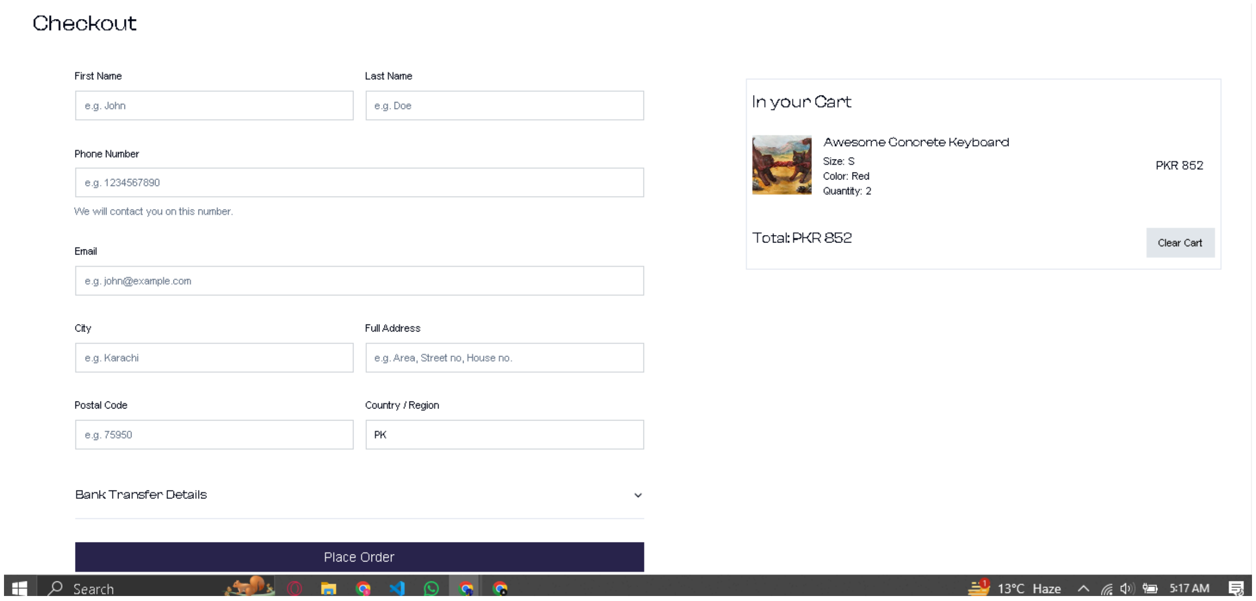
Description:

The automobile layout consists of a front-engine design, with transaxle-type transmissions mounted at the rear of the engine and four wheel drive

- Premium material
- Handmade upholstery
- Quality timeless classic

Dynamic Routing Testing: Tested navigation from the product listing to the product details page and verified the details, with no issues found.

5. Checkout user flow:



Checkout

First Name	Last Name
e.g. John	e.g. Doe

Phone Number

e.g. 1234567890

We will contact you on this number.

Email

e.g. john@example.com

City	Full Address
e.g. Karachi	e.g. Area, Street no, House no.
Postal Code	Country / Region
e.g. 75950	PK

Bank Transfer Details

Place Order

In your Cart

Awesome Concrete Keyboard
Size: S
Color: Red
Quantity: 2

PKR 852

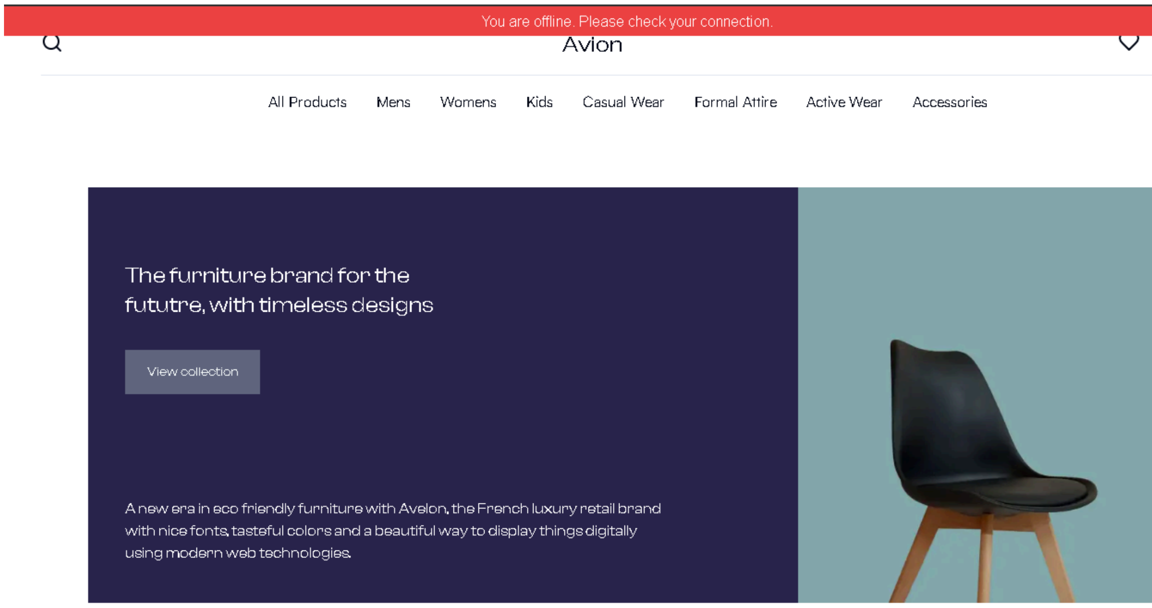
Total: PKR 852

Clear Cart

Checkout Flow Testing: Tested the entire checkout user flow and form validation, ensuring everything works correctly.

Step 2: Error Handling:

Network Failure:



You are offline. Please check your connection.

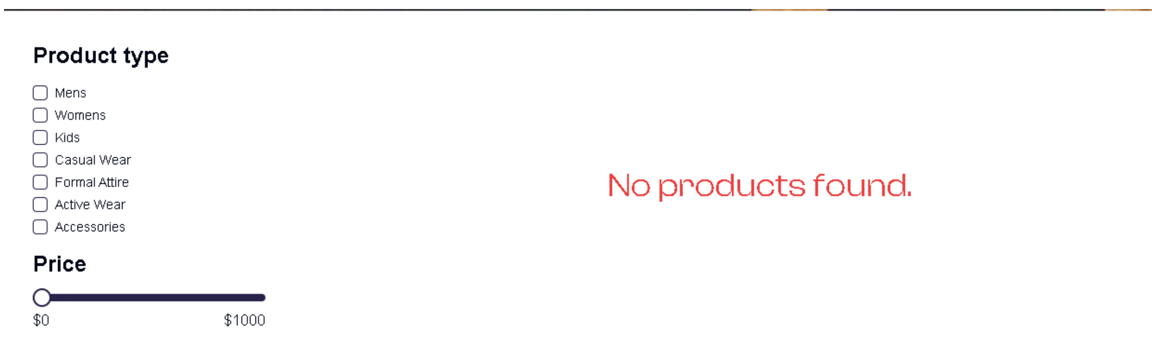
The furniture brand for the
fututre, with timeless designs

A new era in eco friendly furniture with Avelon, the French luxury retail brand with nice fonts, tasteful colors and a beautiful way to display things digitally using modern web technologies.

Added a user-friendly offline error message that appears at the top of the UI when the network connection is lost.

- Used the `navigator.onLine` API and `online`/`offline` events to detect network changes in real-time.

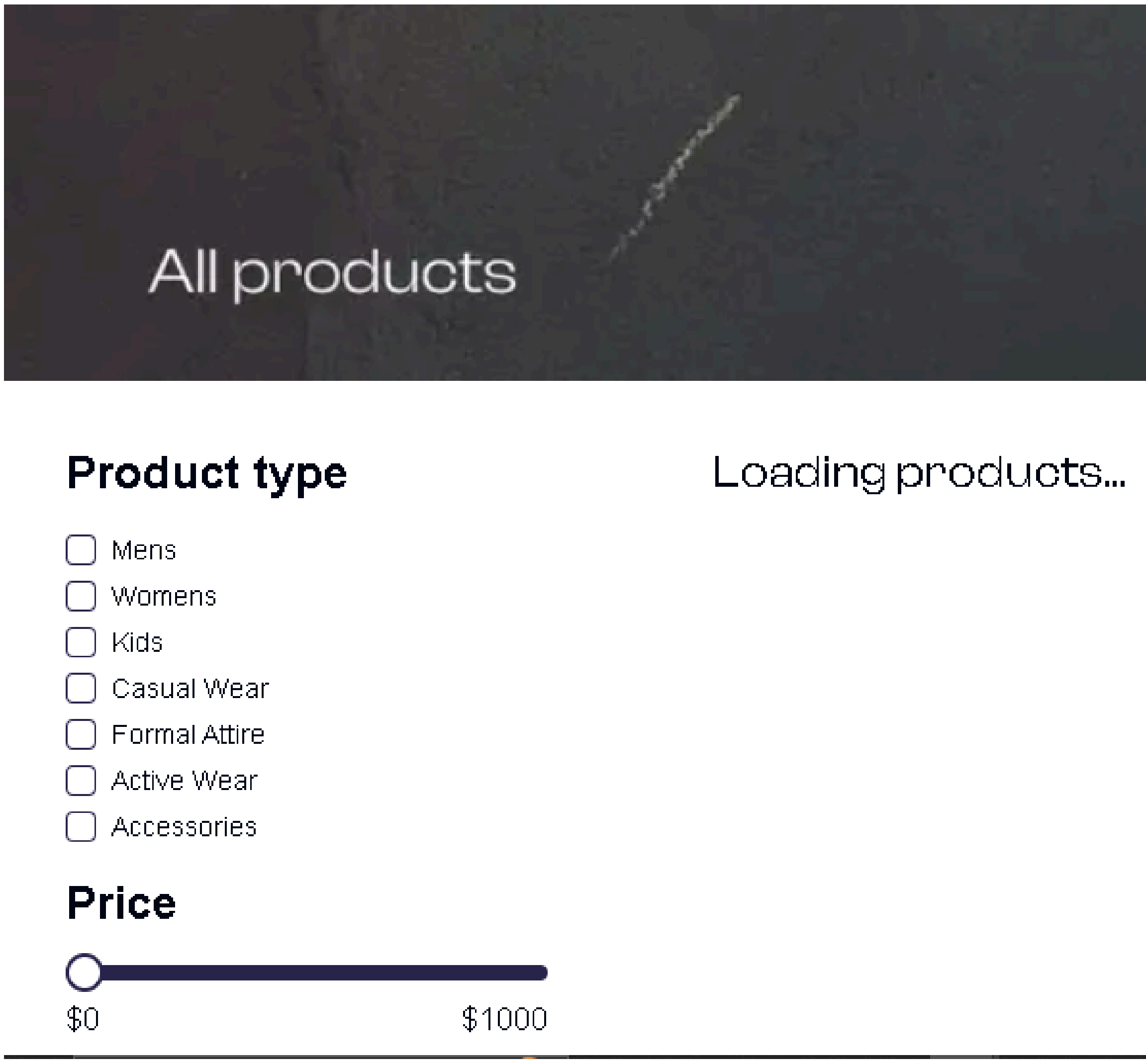
Fallback UI:



No products found.

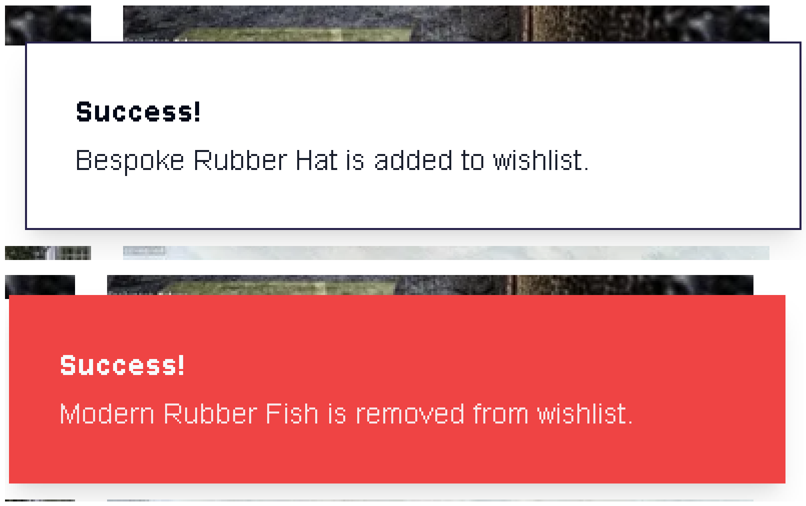
I used try/catch blocks when fetching products or performing any task to catch any errors to display user-friendly messages to users when products are not found or when something went wrong.

Loading UI:



I used loading states to show users when a page is loading to make the page interactive even when it’s loading.

Toast Notifications:



Success!

Bespoke Rubber Hat is added to wishlist.

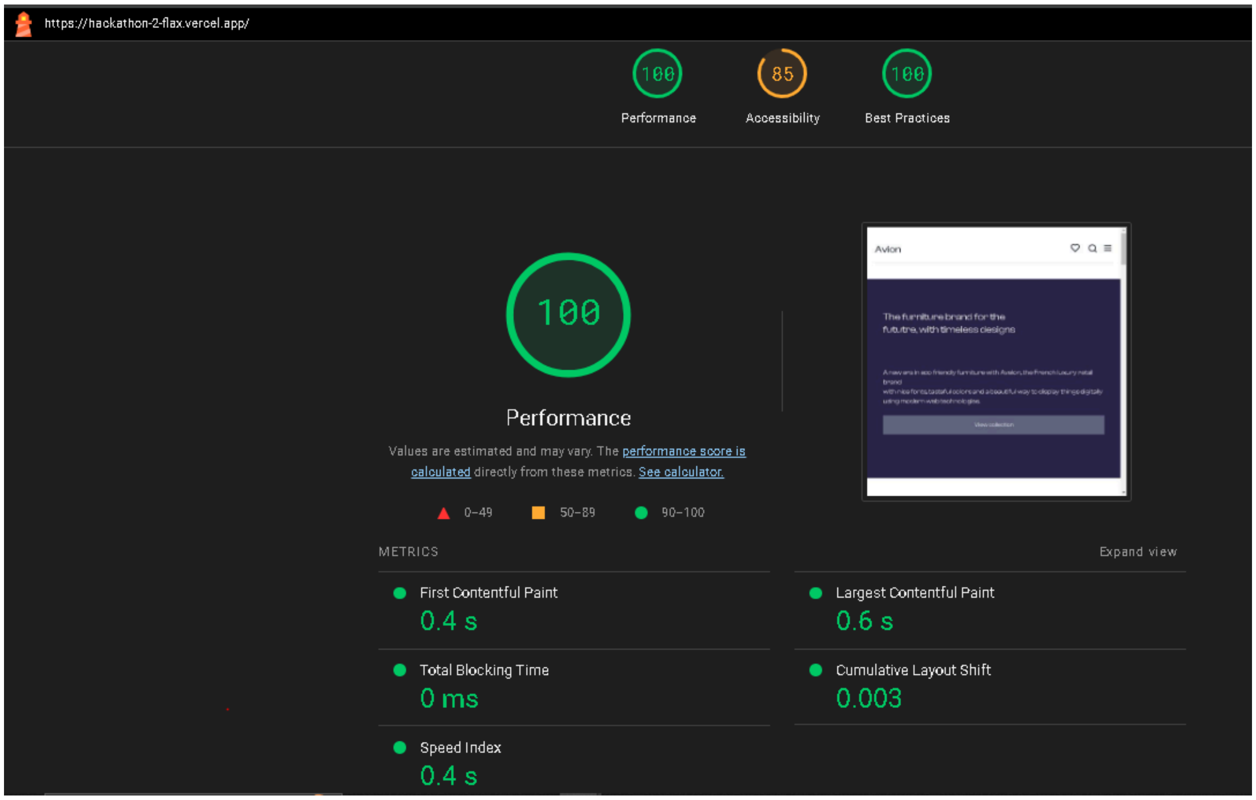
Success!

Modern Rubber Fish is removed from wishlist.

Used the ShadcnUI Toast component for dynamic messages on actions like adding/removing items from the cart or wishlist, and handling missing color/size selection.

Step 3: Performance Optimization:

Performance Analysis :



https://hackathon-2-flax.vercel.app/

100 Performance

85 Accessibility

100 Best Practices

100

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
0.4 s

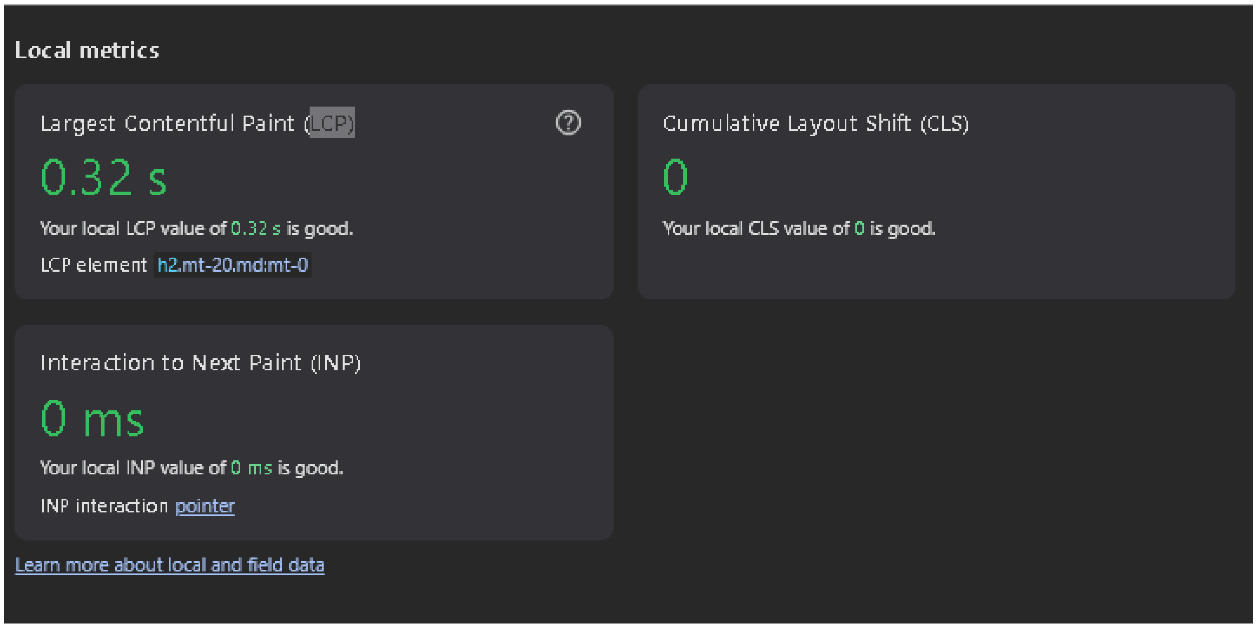
● Largest Contentful Paint
0.6 s

● Total Blocking Time
0 ms

● Cumulative Layout Shift
0.003

● Speed Index
0.4 s

HomePage Load:



Local metrics

Largest Contentful Paint (LCP)

0.32 s

Your local LCP value of 0.32 s is good.

LCP element h2.mt-20.md:mt-0

Cumulative Layout Shift (CLS)

0

Your local CLS value of 0 is good.

Interaction to Next Paint (INP)

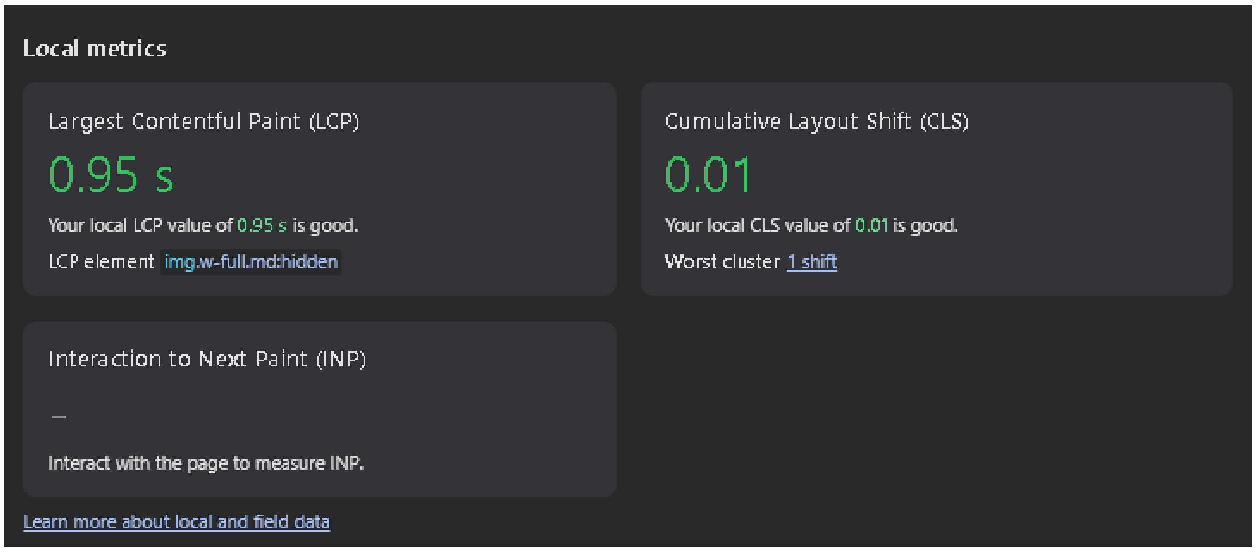
0 ms

Your local INP value of 0 ms is good.

INP interaction pointer

Learn more about local and field data

Product Listing Page Load:



Local metrics

Largest Contentful Paint (LCP)

0.95 s

Your local LCP value of 0.95 s is good.

LCP element img.w-full.md:hidden

Cumulative Layout Shift (CLS)

0.01

Your local CLS value of 0.01 is good.

Worst cluster 1 shift

Interaction to Next Paint (INP)

–

Interact with the page to measure INP.

Learn more about local and field data

Cumulative Layout Shift (CLS) Fix: Addressed poor CLS on the product listing page by setting the loading UI and "No products found" message to use `min-h-screen`, resolving the issue.

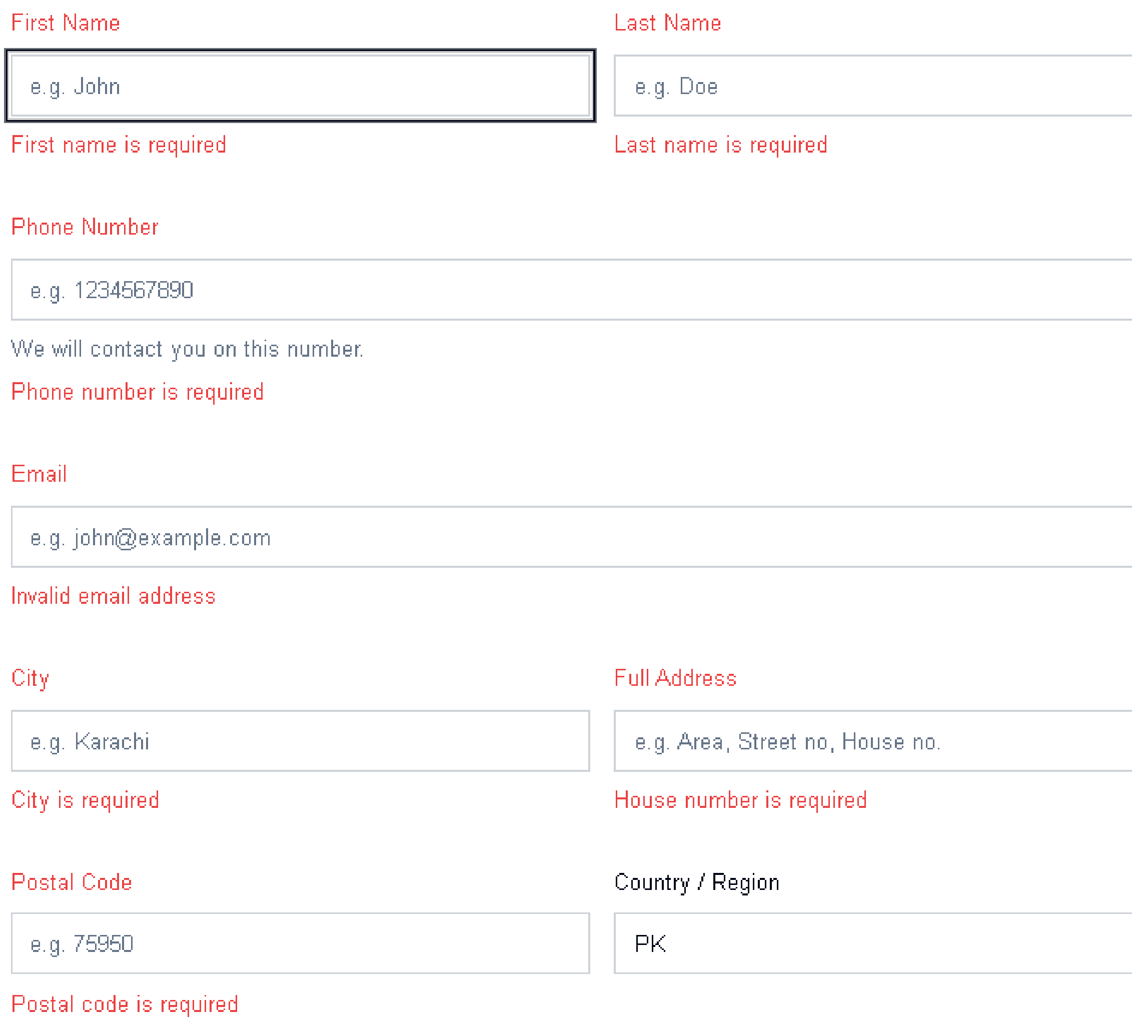
Step 4: Cross-Browser and Device Testing:

Browser Testing: Tested on Chrome, Firefox, Safari, and Edge to ensure consistent rendering and functionality across all browsers.

Manual Mobile Testing: Tested responsiveness manually on my mobile phone to ensure a seamless user experience across devices.

Step 5: Security Testing:

Input Validation:



First Name	Last Name
e.g. John	e.g. Doe
First name is required	Last name is required
Phone Number	
e.g. 1234567890	
We will contact you on this number.	
Phone number is required	
Email	
e.g. john@example.com	
Invalid email address	
City	Full Address
e.g. Karachi	e.g. Area, Street no, House no.
City is required	House number is required
Postal Code	Country / Region
e.g. 75950	PK
Postal code is required	

Used Zod and React Hook Form in the checkout form to sanitize inputs and prevent SQL injection or XSS attacks.

Secure API Calls: All API calls are made over HTTPS, ensuring encrypted communication between the client and server for enhanced security.

Sensitive Data Management: Stored sensitive information, such as API keys and database credentials, securely in a `.env` file to prevent exposure and enhance security.

Step 6. User Acceptance Testing (UAT): I Simulated real-world usage by performing tasks such as browsing products, adding items to the cart, and checking out to ensure functionality meets user expectations.

Testing Report:

Test Case ID	Test Case Description	Test Steps	Expected Result	**Actual Result**	**Status**	**Severity Level**	Remarks
TC001	Validate Product listing page	Open products page > Verified products	Products displayed correctly.	Products displayed correctly.	Passed	High	Passed
TC002	Validate Dynamic routing for product details	Click on a product > Verify its details.	Product details displayed correctly.	Product details displayed correctly.	Passed	High	Passed
TC003	Validate Dynamic routing for product categories	Navigate through different categories > Verified products based on their category	Products display based on their category.	Products display based on their category.	Passed	High	Passed
TC004	Check Cart functionality	Add product to cart > Verify cart contents	Cart items with correct color, size, and quantity.	Cart updates as expected.	Passed	Medium	Works as expected
TC005	Check Checkout functionality	Navigated to checkout from cart page > Verified form submission & cart Items.	Valid form submission & Cart items display with subtotal amount.	Valid form submission & Cart items display with subtotal amount.	Passed	High	Handled gracefully
TC006	Check Search products functionality	Searched for products by their names > Verified search functionality.	Correct product display, based on searched items.	Correct product display, based on searched items.	Passed	Low	Test successful
TC007	Check Products filtering functionality	Open products page > Performed filtering of products by	Correct products display based on categories	Correct products display based on categories selected and	Passed	High	Test successful

		categories and prices.	selected and filtered price.	filtered price.			
TC008	Check Wishlist functionality	Toggled products in wishlist from products listing > Verified wishlist.	Correct wish items displayed in the wishlist.	Correct wish items displayed in the wishlist.	Passed	High	Test successful
TC009	Check Pagination functionality	Open products listing and > Vrfied pagination functionality.	products displayed based on products per page.	products displayed based on products per page.	Passed	Medium	No issues found
TC010	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	Medium	Test successful
TC011	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	Handled gracefully