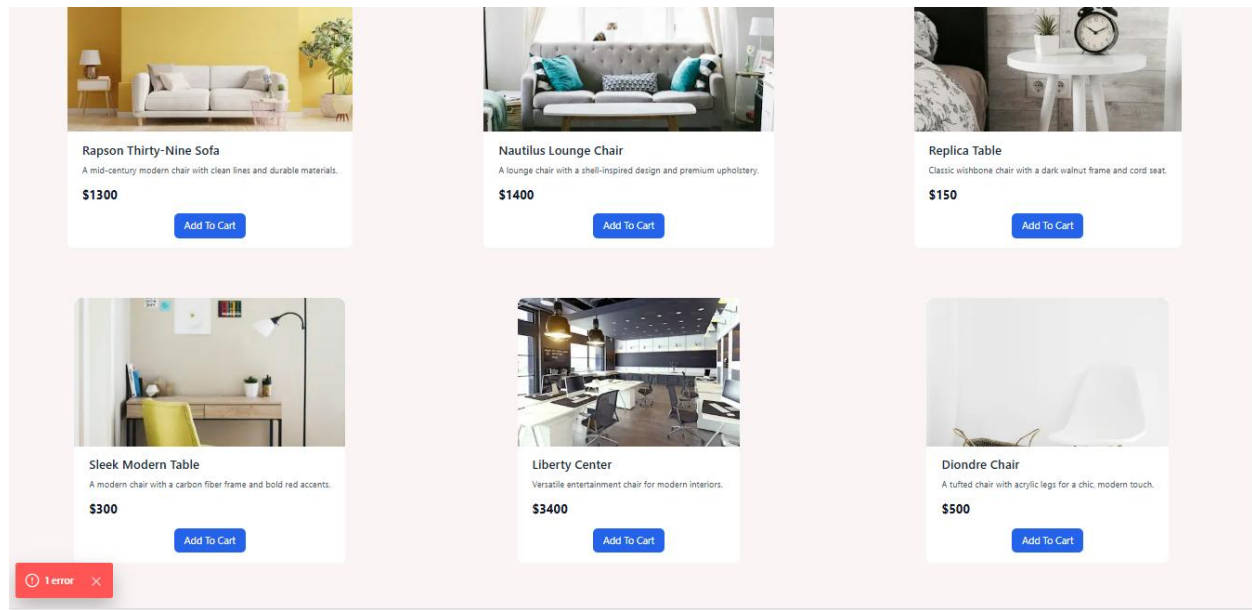


Day 5 - Testing, Error Handling, and Backend Integration Refinement document.

Here is my product list which is perfectly working and showing all the image description and all no problem in it with add to cart button



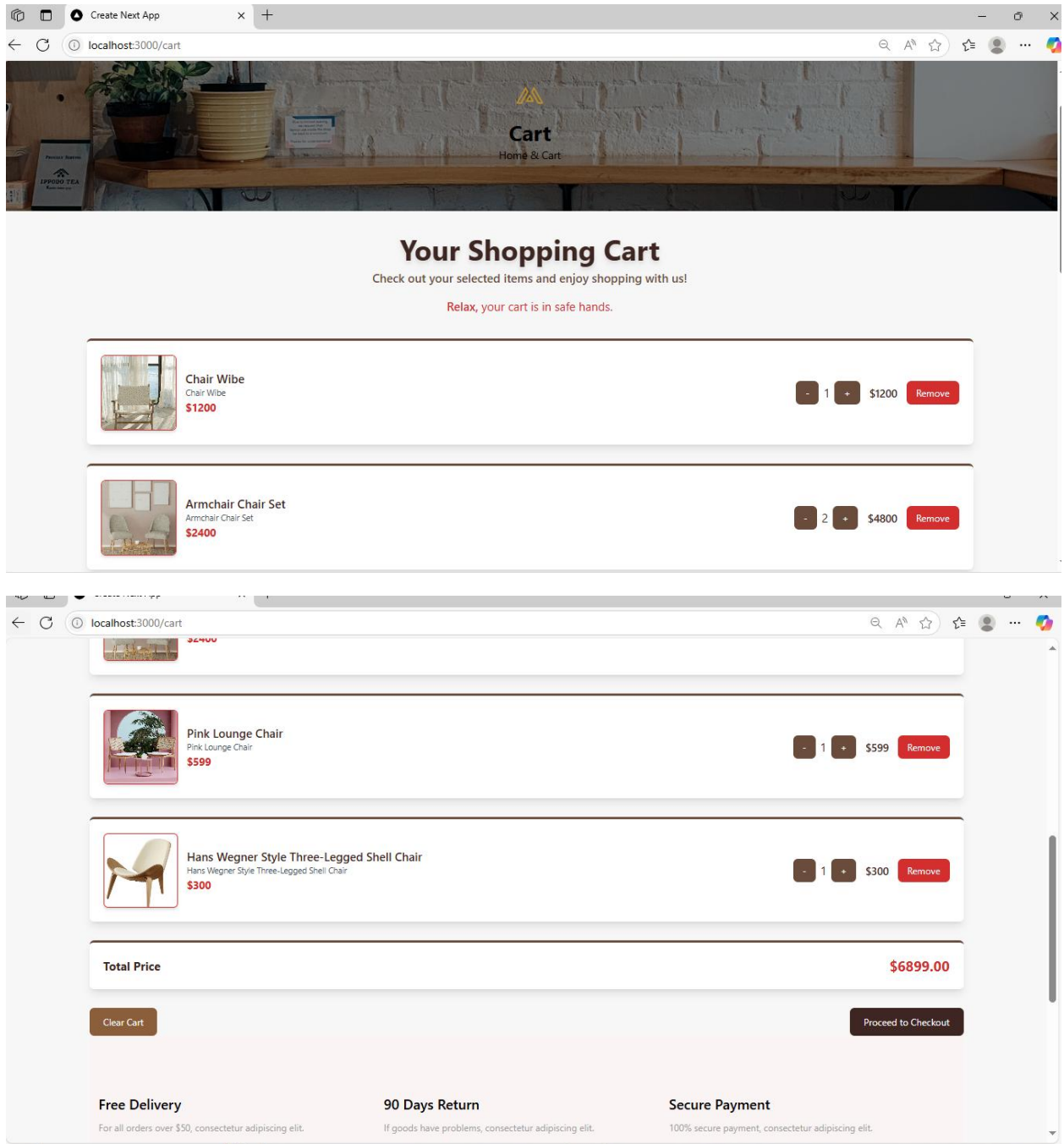
"Our cart page offers a seamless shopping experience with options to easily remove items, adjust product quantities using the increase and decrease buttons, and proceed smoothly to checkout with

Here is the code which is work properly

```
1 // context/CartContext.tsx
2
3 "use client";
4
5 import { createContext, useContext, useState, ReactNode } from "react";
6
7 // Define the product type
8 type Product = {
9   _id: string;
10  name: string;
11  price: number;
12  image: string;
13  quantity: number;
14 };
15
16 interface CartContextType {
17   cart: Product[];
18   addToCart: (product: Product) => void;
19   removeFromCart: (productId: string) => void;
20   clearCart: () => void;
21   checkout: () => void; // Checkout function added here
22 }
23
24 const CartContext = createContext<CartContextType | undefined>(undefined);
25
26 export const CartProvider = ({ children }: { children: ReactNode }) => {
27   const [cart, setCart] = useState<Product[]>([]);
28
29   const addToCart = (product: Product) => {
30     setCart((prevCart) => {
31       const existingProduct = prevCart.find((item) => item._id === product._id);
32       if (existingProduct) {
33         return prevCart.map((item) =>
34           item._id === product._id ? { ...item, quantity: item.quantity + 1 } : item
35         );
36       } else {
37         return [...prevCart, { ...product, quantity: 1 }];
38       }
39     });
40   };
41
42   const removeFromCart = (productId: string) => {
43     setCart((prevCart) => prevCart.filter((item) => item._id !== productId));
44   };
45
46   const clearCart = () => setCart([]);
47
48   const checkout = () => {
49     // Checkout logic
50   };
51
52   return (
53     <CartContext.Provider value={{ cart, addToCart, removeFromCart, clearCart, checkout }}>
54       {children}
55     </CartContext.Provider>
56   );
57 };
58
59 export { CartContext, CartProvider };
```

```
1 // Define the CartItem type
2 interface CartItem {
3   _id: string;
4   name: string;
5   price: number;
6   quantity: number;
7   image: string; // You can replace 'any' with a more specific type if you know the structure of the image.
8 }
9
10 const CartPage = () => {
11   const { cart, removeFromCart, clearCart, addToCart } = useCart();
12   const [removeMessage, setRemoveMessage] = useState(false);
13   const [orderProcessing, ] = useState(false);
14   const [orderSuccess, ] = useState(false);
15   const [emptyCartMessage, setEmptyCartMessage] = useState(false);
16
17   const handleIncrease = (item: CartItem) => {
18     addToCart({ ...item, quantity: item.quantity + 1 });
19   };
20
21   const handleDecrease = (item: CartItem) => {
22     if (item.quantity > 1) {
23       addToCart({ ...item, quantity: item.quantity - 1 });
24     }
25   };
26
27   const handleRemove = (itemId: string) => {
28     removeFromCart(itemId);
29     setRemoveMessage(true);
30     setTimeout(() => setRemoveMessage(false), 4000);
31   };
32
33   useEffect(() => {
34     if (cart.length === 0) {
35       setEmptyCartMessage(true);
36     } else {
37       setEmptyCartMessage(false);
38     }
39   }, [cart]);
40
41   return (
42     <div>
43       <CartItem />
44       <div>
45         <button>Add to Cart</button>
46         <button>Remove from Cart</button>
47         <button>Clear Cart</button>
48         <button>Checkout</button>
49       </div>
50     </div>
51   );
52 };
53
54 export { CartPage };
```

"Here's a snapshot of my cart, where you can review your selected items, adjust quantities, and remove products as needed before proceeding to checkout."







"Below is the code for the checkout page, where users can review and input all necessary details, including shipping address, contact information, and payment options. This ensures a secure and seamless finalization of the order process."

```
my-app > src > app > Checkout > page.tsx > Checkout
10 const Checkout = () => {
32   const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
35     ...formData,
36     [name]: value,
37   });
38 };
39
40 const handleConfirmOrder = () => {
41   // Simulate order confirmation logic here
42   setOrderSuccess(true);
43
44   // Reset form and cart after a brief delay
45   setTimeout(() => {
46     setOrderSuccess(false);
47     // Clear cart if necessary
48     clearCart();
49   }, 5000);
50 };
51
52 return (
53   <div className="relative w-full">
54     {/* Header Section */}
55     <div className="relative w-full h-[316px]">
56       
61
62       <div className="absolute inset-0 flex flex-col items-center justify-center bg-black bg-opacity-50 space-y-1">
63         
64
65         {/* Shop link */}
66         <h1 className="text-balck text-4xl font-bold">
67           <Link href="/Checkout">
68             Checkout
69           </Link>
70         </h1>
71       </div>
54   </div>
55 );
```

"Here are the output images showcasing the checkout page, where customers can easily input their details, review their order summary, and proceed with payment. This visual demonstrates how the user interface looks when all customer information is filled out and ready for submission."

Everything is working properly

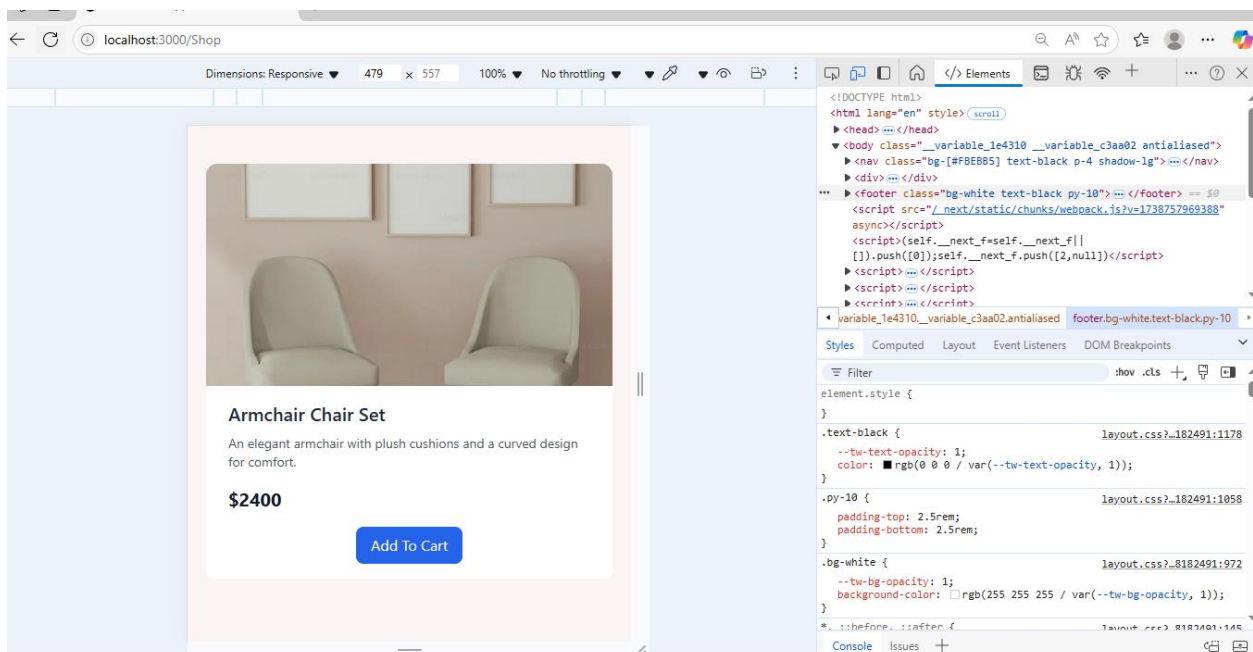
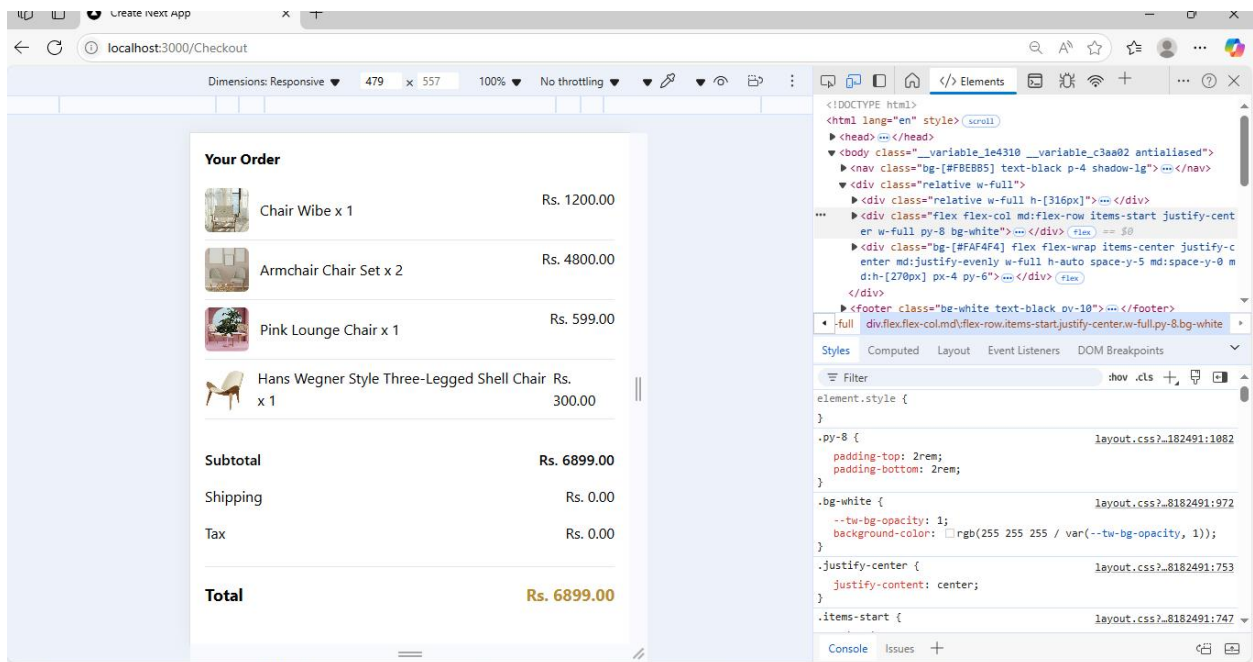
Billing Details	
First Name	Last Name
<input type="text" value="Enter your first name"/>	<input type="text" value="Enter your last name"/>
Email	
<input type="text" value="Enter your email"/>	
Phone Number	
<input type="text" value="Enter your phone number"/>	
Address	
<input type="text" value="Enter your address"/>	
City	
<input type="text" value="Enter your city"/>	
Zip Code	
<input type="text" value="Enter your zip code"/>	
Country	
<input type="text" value="Enter your country"/>	
Province	
<input type="text" value=""/>	

Your Order	
 Chair Wibe x 1	Rs. 1200.00
 Armchair Chair Set x 2	Rs. 4800.00
 Pink Lounge Chair x 1	Rs. 599.00
 Hans Wegner Style Three-Legged Shell Chair x 1	Rs. 300.00
Subtotal	Rs. 6899.00
Shipping	Rs. 0.00
Tax	Rs. 0.00
Total	Rs. 6899.00
<button>Confirm Order</button>	

About Responsive

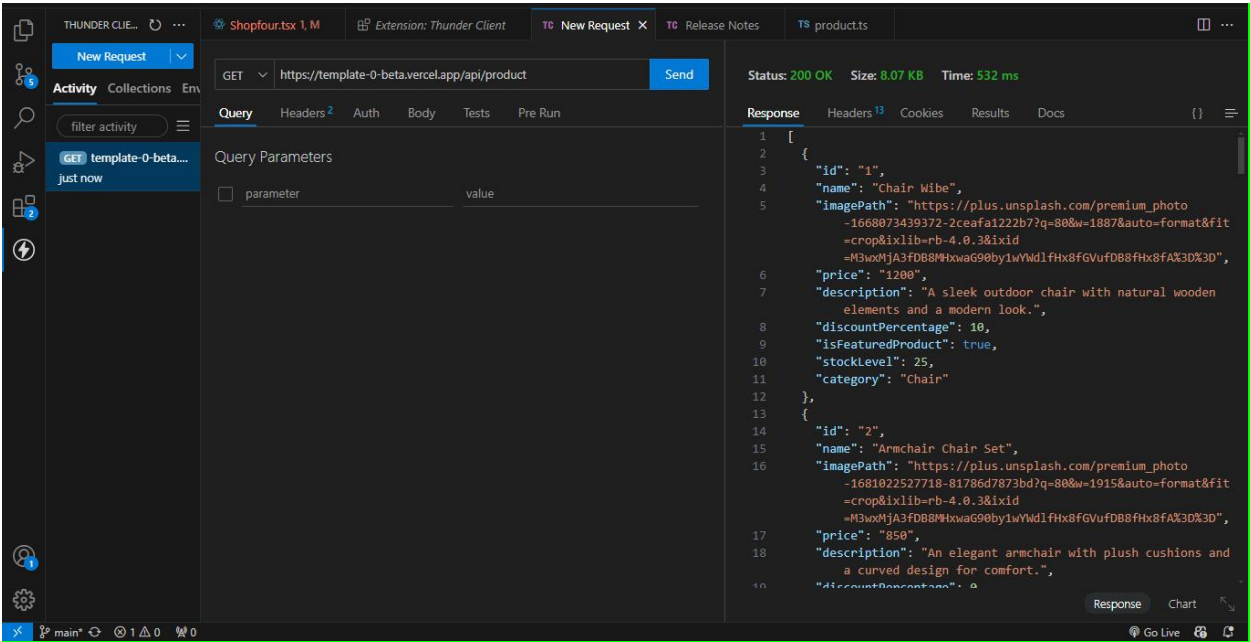
"These output images highlight the responsive design of the checkout page, ensuring that users can seamlessly input their details and complete their purchase, whether on desktop, tablet, or mobile. The layout automatically adjusts for an optimal experience on any device."

Responsive work properly



Thender client report

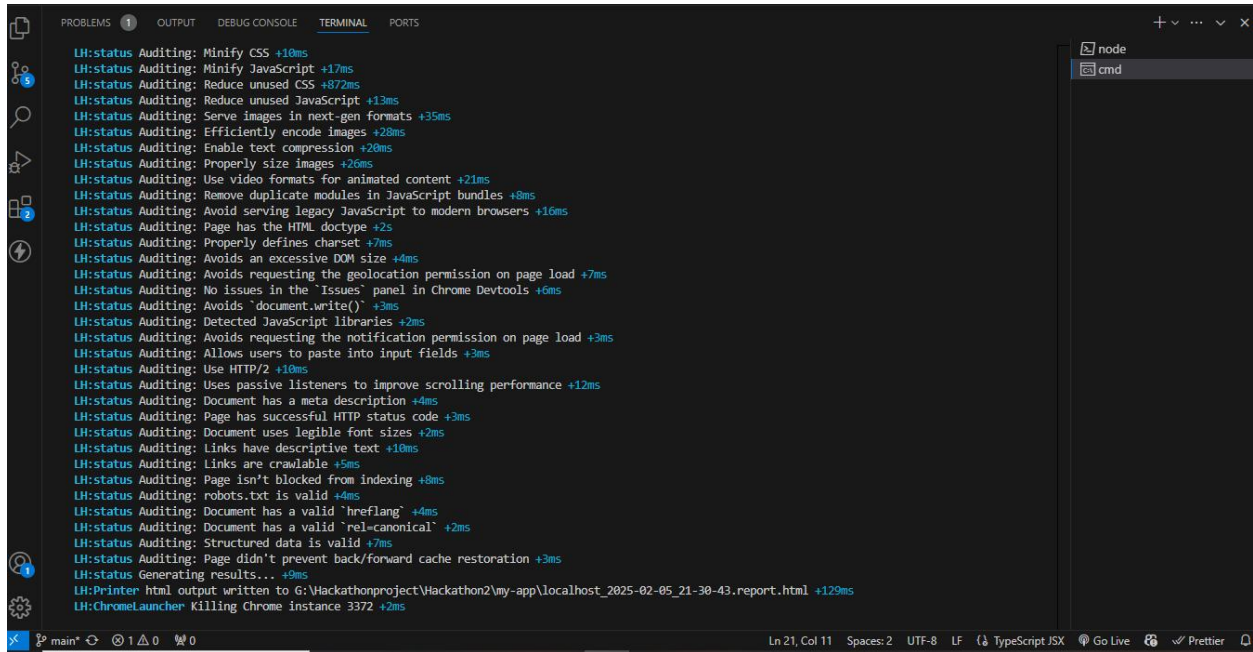
Get report



Testing report

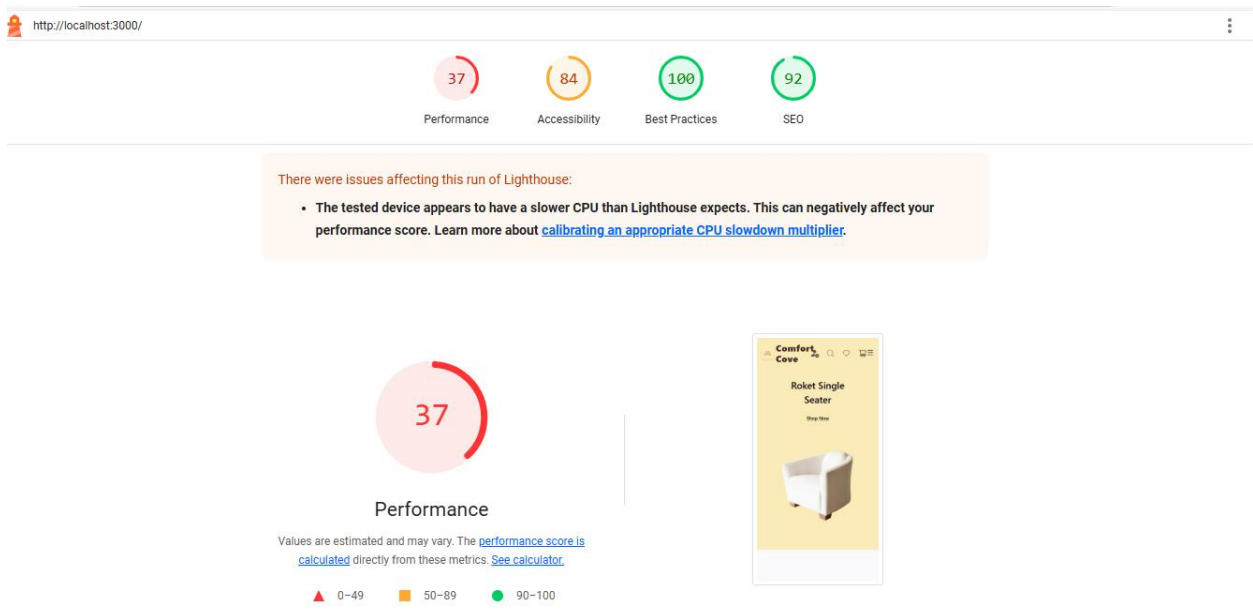
Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
2	TC001	Validate product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	high	No issues found
3	TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	Handled gracefully
4	TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	High	Works as expected
5	TC004	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	high	Test successful

Lighthouse report

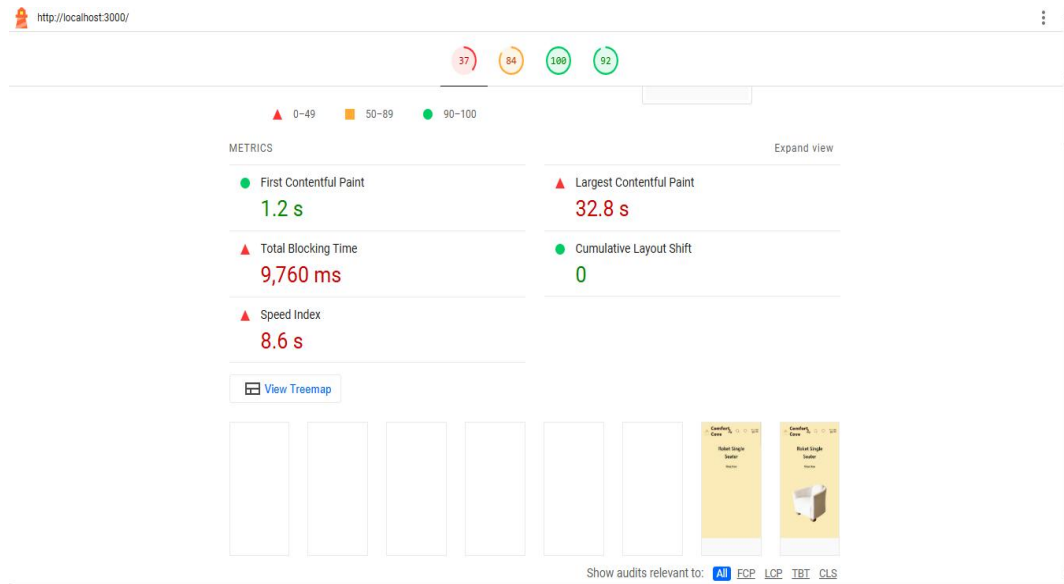


```
LH:status Auditing: Minify CSS +10ms
LH:status Auditing: Minify JavaScript +17ms
LH:status Auditing: Reduce unused CSS +872ms
LH:status Auditing: Reduce unused JavaScript +13ms
LH:status Auditing: Serve images in next-gen formats +35ms
LH:status Auditing: Efficiently encode images +28ms
LH:status Auditing: Enable text compression +20ms
LH:status Auditing: Properly size images +26ms
LH:status Auditing: Use video formats for animated content +21ms
LH:status Auditing: Remove duplicate modules in JavaScript bundles +8ms
LH:status Auditing: Avoid serving legacy JavaScript to modern browsers +16ms
LH:status Auditing: Page has the HTML doctype +2s
LH:status Auditing: Properly defines charset +7ms
LH:status Auditing: Avoids an excessive DOM size +4ms
LH:status Auditing: Avoids requesting the geolocation permission on page load +7ms
LH:status Auditing: No issues in the 'Issues' panel in Chrome Devtools +6ms
LH:status Auditing: Avoids 'document.write()' +3ms
LH:status Auditing: Detected JavaScript libraries +2ms
LH:status Auditing: Avoids requesting the notification permission on page load +3ms
LH:status Auditing: Allows users to paste into input fields +3ms
LH:status Auditing: Use HTTP/2 +10ms
LH:status Auditing: Uses passive listeners to improve scrolling performance +12ms
LH:status Auditing: Document has a meta description +4ms
LH:status Auditing: Page has successful HTTP status code +3ms
LH:status Auditing: Document uses legible font sizes +2ms
LH:status Auditing: Links have descriptive text +10ms
LH:status Auditing: Links are crawlable +5ms
LH:status Auditing: Page isn't blocked from indexing +8ms
LH:status Auditing: robots.txt is valid +4ms
LH:status Auditing: Document has a valid 'hreflang' +4ms
LH:status Auditing: Document has a valid 'rel=canonical' +2ms
LH:status Auditing: Structured data is valid +7ms
LH:status Auditing: Page didn't prevent back/forward cache restoration +3ms
LH:status Generating results... +9ms
LH:Printer html output written to G:\Hackathonproject\Hackathon2\my-app\localhost_2025-02-05_21-30-43.report.html +129ms
LH:Chromelauncher Killing Chrome instance 3372 +2ms
```

Lighthouse command work properly and give result



"The Lighthouse audit for this page shows excellent performance, with high scores across all categories, including performance, accessibility, SEO, and best practices. The page loads quickly, follows modern web standards, and provides a great user experience."





- ▲ Serve images in next-gen formats — Potential savings of 11,579 KiB
- ▲ Page prevented back/forward cache restoration — 3 failure reasons
- Image elements do not have explicit width and height
- Minify JavaScript — Potential savings of 5 KiB
- Eliminate render-blocking resources — Potential savings of 0 ms
- Defer offscreen images — Potential savings of 11,868 KiB
- Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB
- Reduce unused JavaScript — Potential savings of 456 KiB
- Avoid enormous network payloads — Total size was 19,300 KiB
- Avoid long main-thread tasks — 15 long tasks found
- Avoids an excessive DOM size — 215 elements
- Avoid chaining critical requests — 1 chain found
- Minimize third-party usage — Third-party code blocked the main thread for 0 ms



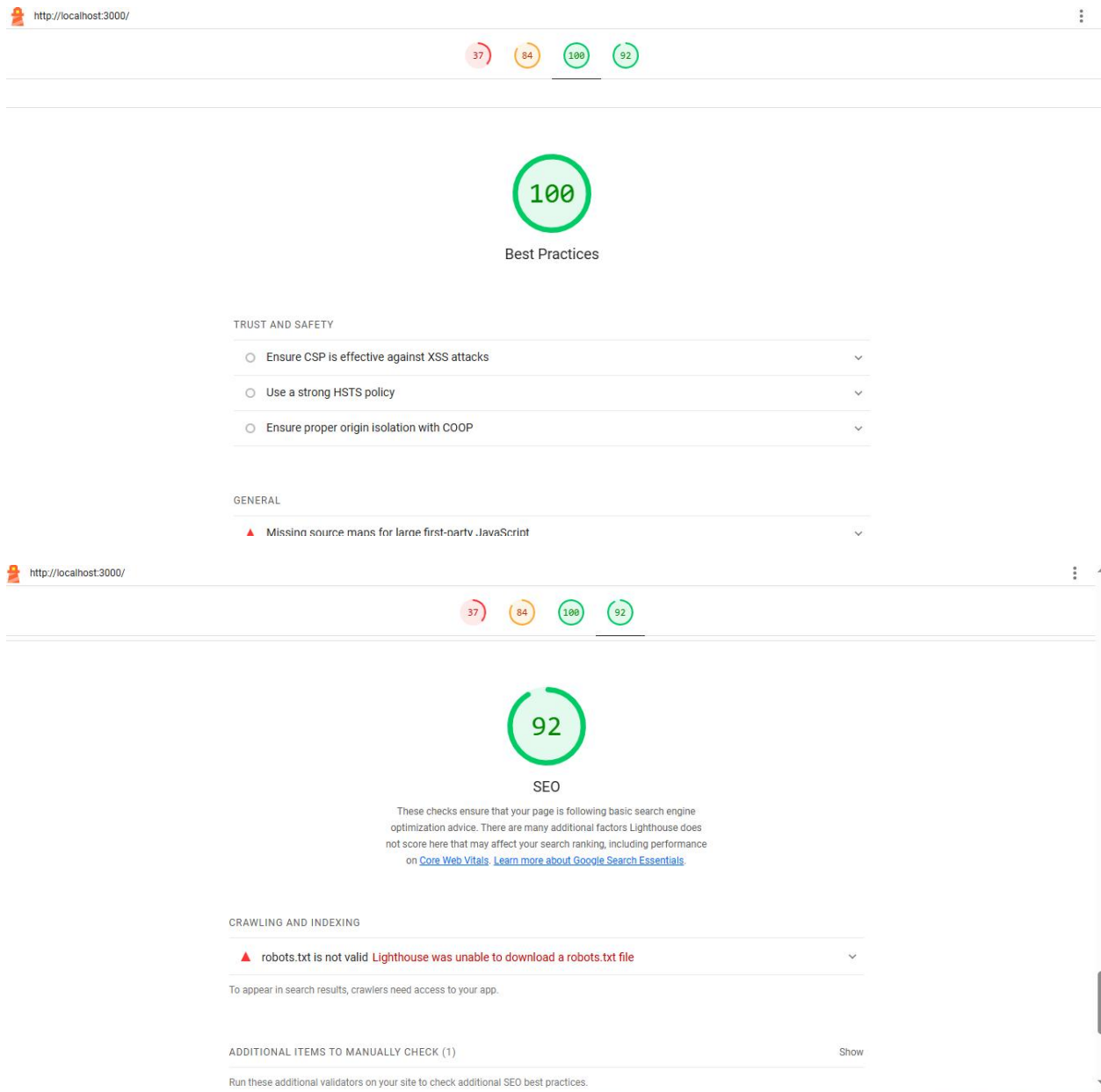
Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

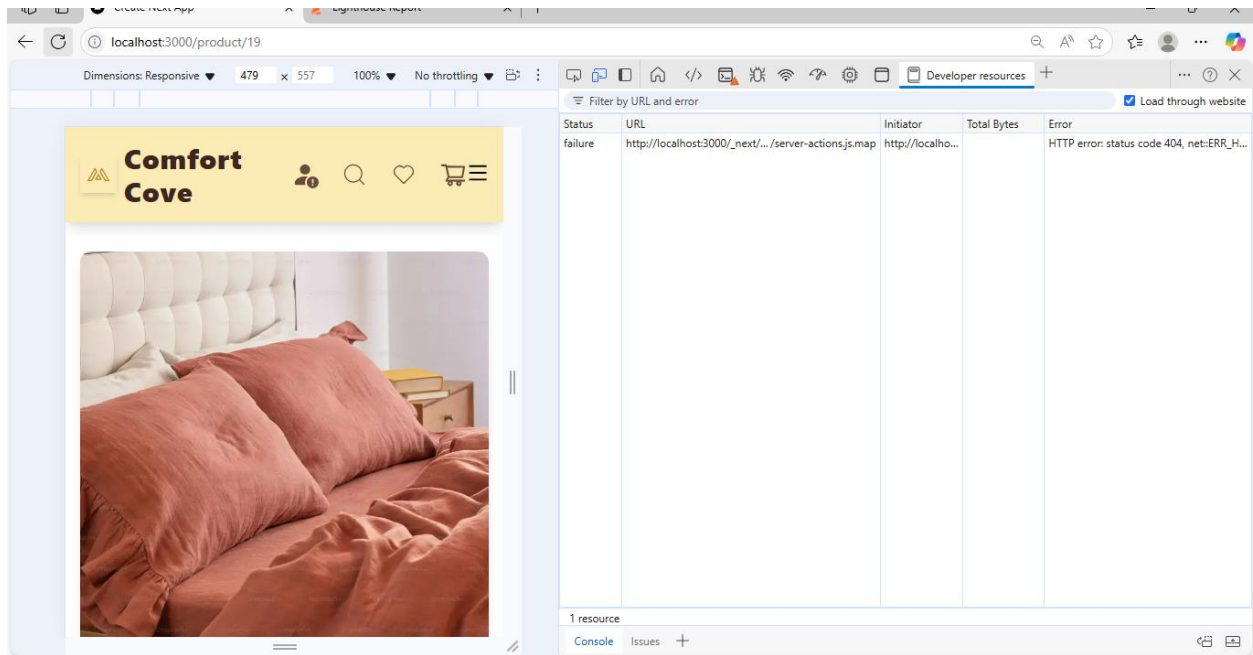
NAMES AND LABELS

- ▲ Buttons do not have an accessible name
- ▲ Links do not have a discernible name

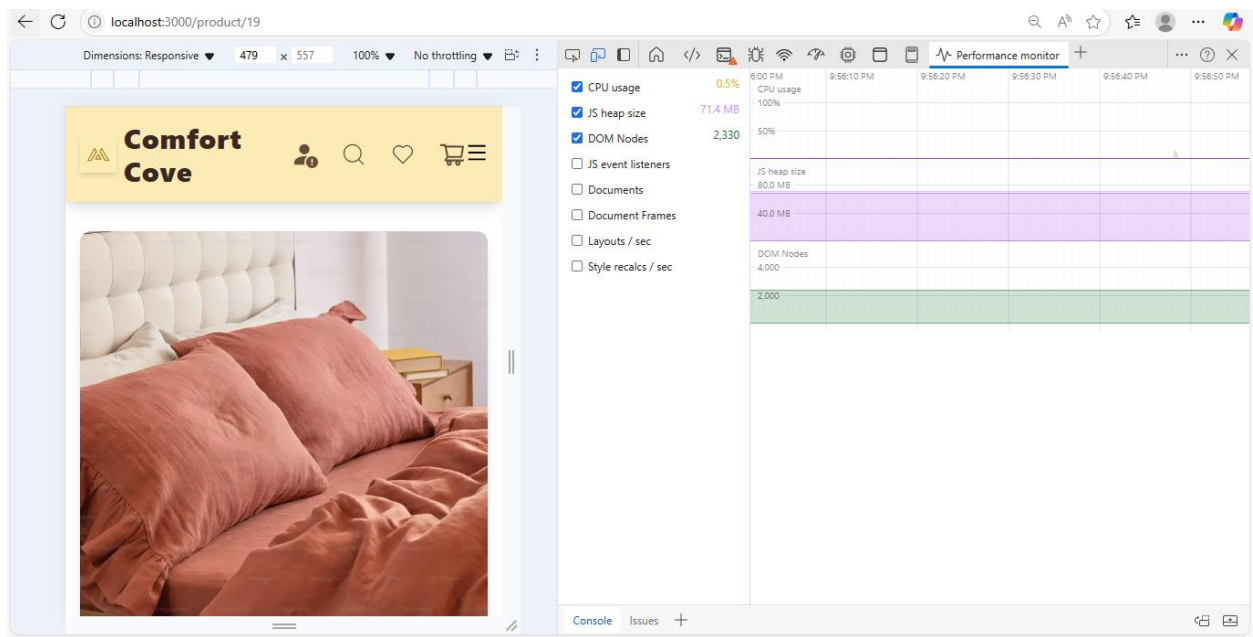
These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.



Developer resources



Performance monitor



Security

