Functional Testing

S.No	Test Case Description	Steps Taken	Outcome	Status
1.	Dynamic routing	Applied dynamic routing to fetch data	Routes are dynamic as expected	Passed
2	Fetch through API	Use API to fetch data	Fetched data properly through API	Passed
3	Add to Cart Functionality	Add, remove , delete cart and cart details	Performs add to cart acurately	Passed
4	Products Details Page	Each product has detailed page	Show product detail page with description	Passed
5	Responsiveness on Mobile	Ensure responsiveness	Responsiveness added as expected	Passed
6	Apply filter	Product filteration through their price	Filteration through min and max price	Passed
7	Image Optimization	Products image loads correctly	Images are displayed without any distortion	Passed

```
}

const [product, setProduct] = useState<Product | null>(null);
const [loading, setLoading] = useState(true);
const [error, setError] = useState<string | null>(null);
const { addToCart } = useCart();
const router = useRouter();

useEffect(() => {
  Codeium: Refactor | Explain | Generate JSDoc | X
  const fetchProductData = async () => {
    if (slug) {
      try {
        const fetchedProduct = await getData(slug);
        if (!fetchedProduct) {
          setError('Product not found. Please check the URL or try again later.');
        } else {
          setProduct(fetchedProduct);
        }
      } catch (err) {
        setError('There was an error while fetching the product. Please try again later.');
        console.error('Error fetching product data:', err);
      }
      setLoading(false);
    }
  };
  fetchProductData();
}, [slug]);

Codeium: Refactor | Explain | Generate JSDoc | X
const handleAddToCart = () => {
  if (product) {
    const cartItem = {
      id: product._id,
      name: product.title,
      price: product.price,
      image: product.imageUrl || '',
      quantity: 1,
    };
```

```
Codeium: Refactor | Explain
interface Product {
    _id: string;
    title: string;
    slug: string;
    price: number;
    badge?: string;
    badgeColor?: string;
    priceWithoutDiscount?: number;
    imageUrl?: string;
  }


Codeium: Refactor | Explain
interface FilterProps {
  products: Product[];
  onFilter: (filteredProducts: Product[]) => void;
}

Codeium: Refactor | Explain | Generate JSDoc | ×
const Filter: React.FC<FilterProps> = ({ products, onFilter }) => {
  const [minPrice, setMinPrice] = useState<number>(0); // Set initial min price
  const [maxPrice, setMaxPrice] = useState<number>(1000); // Set initial max price

  // Price range filter function
  Codeium: Refactor | Explain | ×
  const handleFilter = () => {
    const filtered = products.filter(
      (product) => product.price >= minPrice && product.price <= maxPrice
    );
    onFilter(filtered); // Update the filtered product list
  };

  // Reset filter function
  Codeium: Refactor | Explain | ×
  const handleResetFilter = () => {
    setMinPrice(0);
    setMaxPrice(100);
    onFilter(products); // Reset to show all products
  };

```

```
    }`
  );
  return res;
};

Codeium: Refactor | Explain | Generate JSDoc | ×
const Product = () => {
  const [products, setProducts] = useState<Product[]>([]); // State to hold products
  const [loading, setLoading] = useState<boolean>(true); // To manage loading state

  useEffect(() => {
    // Fetch the data when the component mounts
    Codeium: Refactor | Explain | ×
    const fetchProducts = async () => {
      const data = await getData();
      setProducts(data);
      setLoading(false); // Set loading to false after data is fetched
    };

    fetchProducts(); // Call the fetch function
  }, []);

  if (loading) {
    return <div>Loading...</div>; // You can replace this with a loading spinner if needed
  }
```

```
          <Link href="#" className="text-sm md:text-base text-[#007580] hover:text-[#FFA500] ml-
            View all
          </Link>
        </p>
      </div>

      {/* Product Grid */}
      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-8 px-6 lg:px-
        {isLoading ? (
          <div className="text-center col-span-full text-gray-500">Loading products...</div>
        ) : sanityProducts.length > 0 ? (
          sanityProducts.map((product) => (
            <div key={product._id} className="group relative">
              {/* Image with Hover Effect */}
              <div className="w-full h-72 sm:h-80 md:h-96 relative">
                <Image
                  src={product.imageUrl || '/placeholder-image.png'}
                  alt={product.title}
                  layout="fill"
                  objectFit="cover"
                  className="rounded-lg transition-transform duration-300 group-hover:scale-105"
                />
              </div>

              {/* Product Details */}
              <div className="flex justify-between mt-4">
                <div className="text-[#272343] font-normal text-base sm:text-lg">
                  {product.title}
                </div>
                <div className="font-bold text-base sm:text-lg">
                  {`$${product.price.toFixed(2)}`}
                </div>
              </div>
            </div>
          ))
        ) : (
          <p className="text-center col-span-full text-gray-500">No featured products found.</p>
        )}
      </div>
```

```
};

const CartContext = createContext<CartContextType | undefined>(undefined);

Codeium: Refactor | Explain | Generate JSDoc | X
export const CartProvider = ({ children }: { children: ReactNode }) => {
  const [cart, setCart] = useState<CartItem[]>([]);

  // Load cart from localStorage on first render
  useEffect(() => {
    const savedCart = JSON.parse(localStorage.getItem('cart') || '[]');
    setCart(savedCart);
  }, []);

  // Save cart to localStorage whenever it changes
  useEffect(() => {
    if (cart.length > 0) {
      localStorage.setItem('cart', JSON.stringify(cart));
    }
  }, [cart]);

  Codeium: Refactor | Explain | Generate JSDoc | X
  const addToCart = (item: CartItem) => {
    setCart((prevCart) => {
      const existingItem = prevCart.find((i) => i.id === item.id);
      if (existingItem) {
        return prevCart.map((i) =>
          i.id === item.id ? { ...i, quantity: i.quantity + 1 } : i
        );
      }
      return [...prevCart, item];
    });
  };

  Codeium: Refactor | Explain | Generate JSDoc | X
  const removeFromCart = (id: string) => {
    setCart((prevCart) => prevCart.filter((item) => item.id !== id));
  };

  Codeium: Refactor | Explain | Generate JSDoc | X
  const updateQuantity = (id: string, quantity: number) => {
    setCart((prevCart) =>
      prevCart.map((item) =>
```

11:23:08 PM - localhost:3000

http://localhost:3000/products

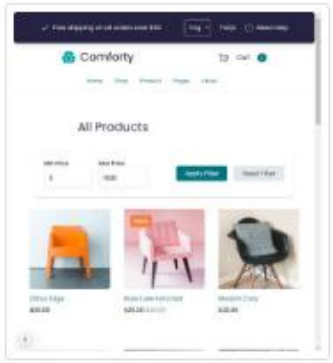
Performance	Accessibility	Best Practices	SEO
79	96	78	92

79

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 0.3 s	■ Largest Contentful Paint 1.8 s
● Total Blocking Time 30 ms	▲ Cumulative Layout Shift 0.255
● Speed Index 0.7 s	

PASSED AUDITS (19) Show

96

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.

CONTRAST

▲ Background and foreground colors do not have a sufficient contrast ratio.

These are opportunities to improve the legibility of your content.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (24) Show

NOT APPLICABLE (32) Show