

SHOP.CO – Hassan RJ

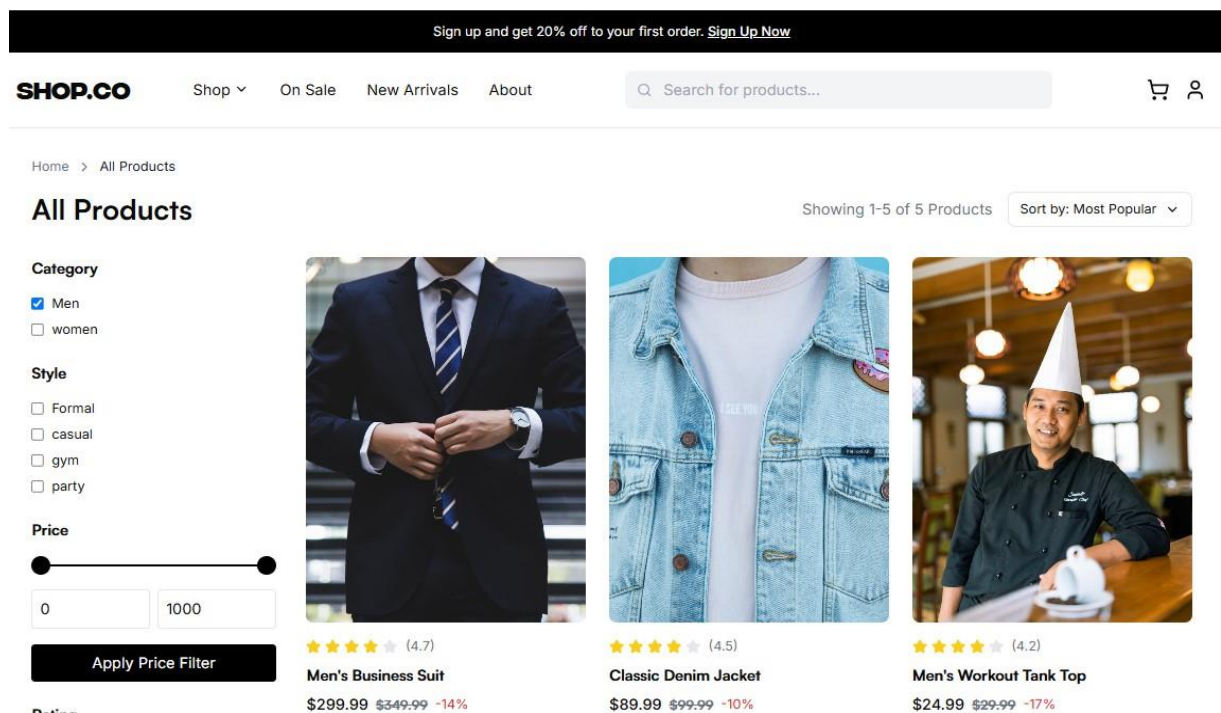
Marketplace Self Validation Checklist by HassanRJ

I. Functional Deliverables

1. Product Listing Page:

A grid layout showcasing dynamic product data, including:

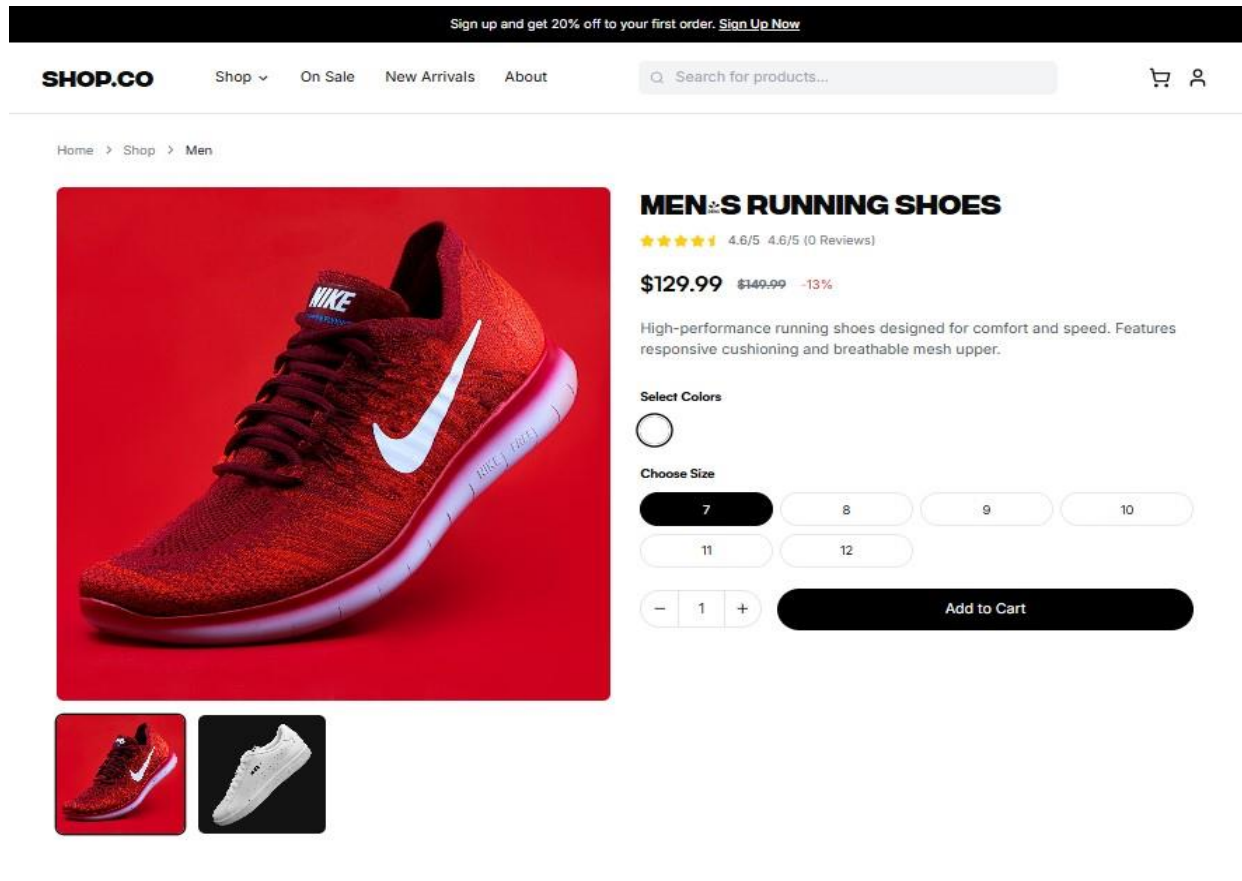
- Product Name
- Price
- Image
- Reviews
- Ratings



2. Product Detail Page

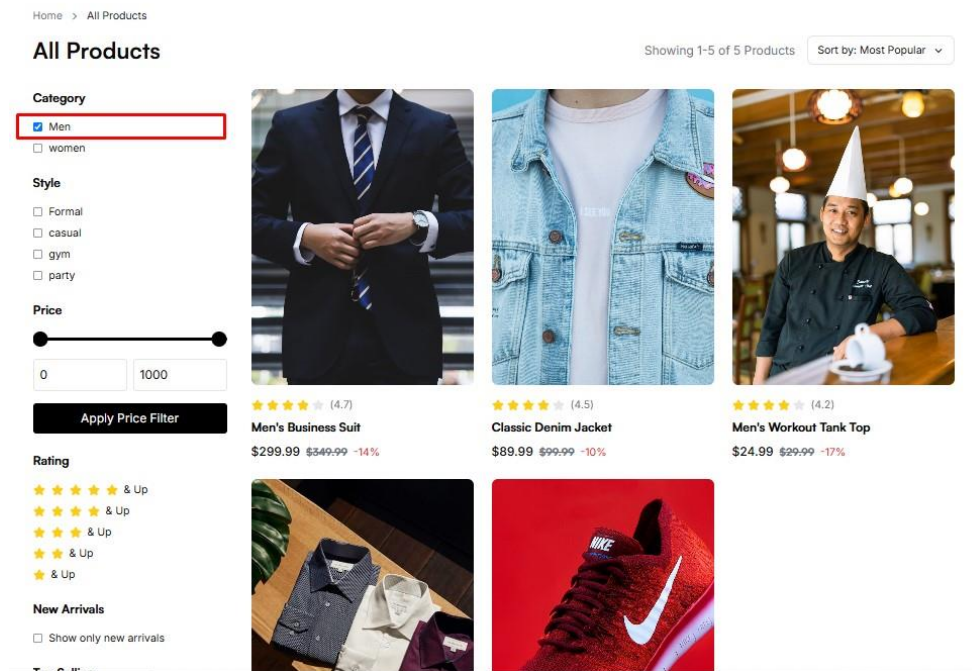
Dynamic product pages implemented with routing. Displayed details include:

- a. Product Name
- b. Description
- c. Price
- d. Sizes
- e. Colors
- f. Images
- g. Inventory
- h. Category
- i. Style
- j. Tags



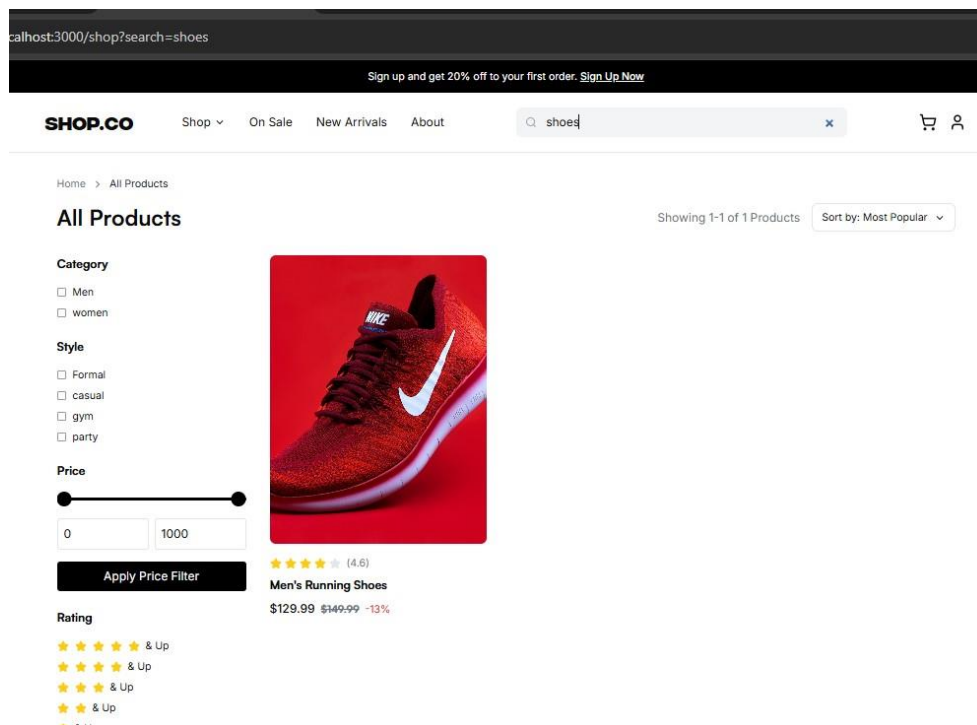
3. Category Filters

Fully functional category filters allowing users to filter products dynamically



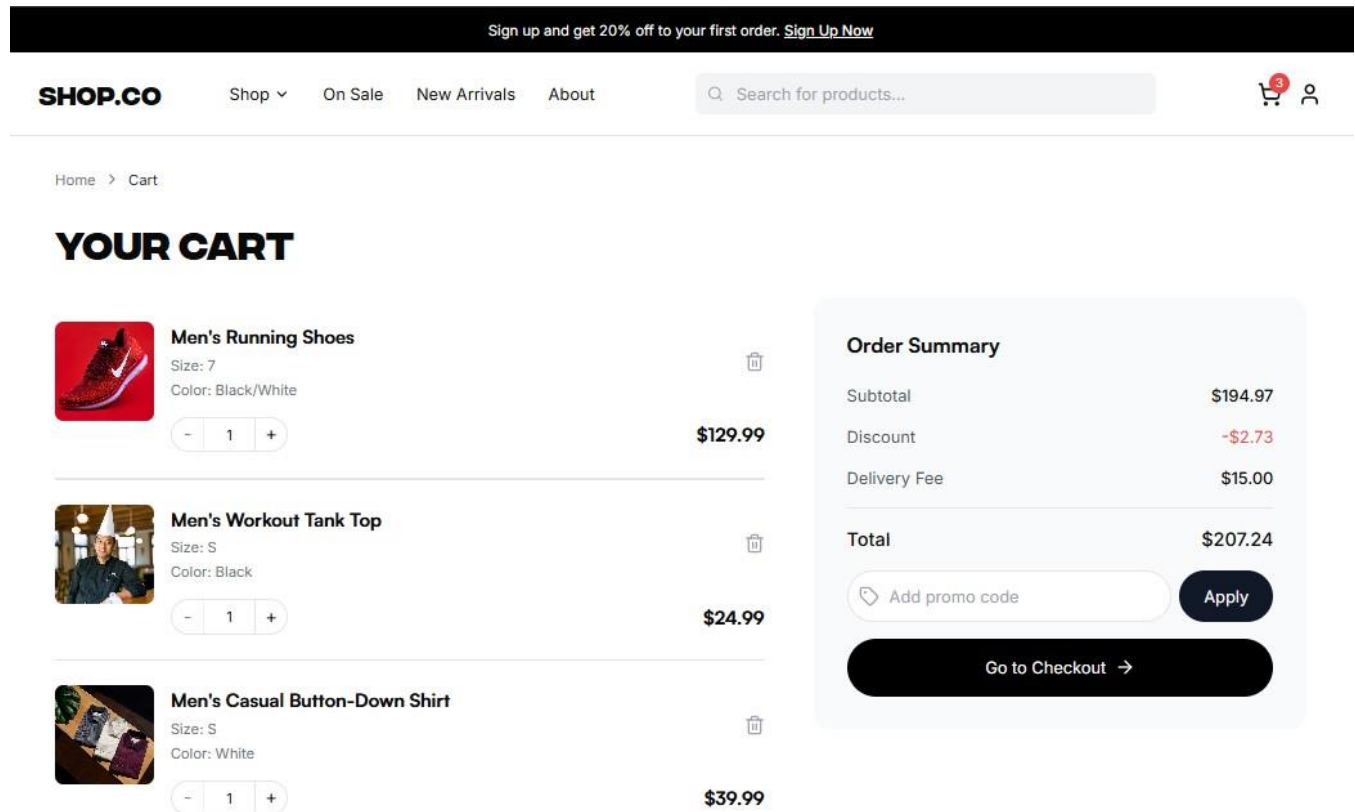
4. Search Bar

Implemented search functionality to filter products by name or tags



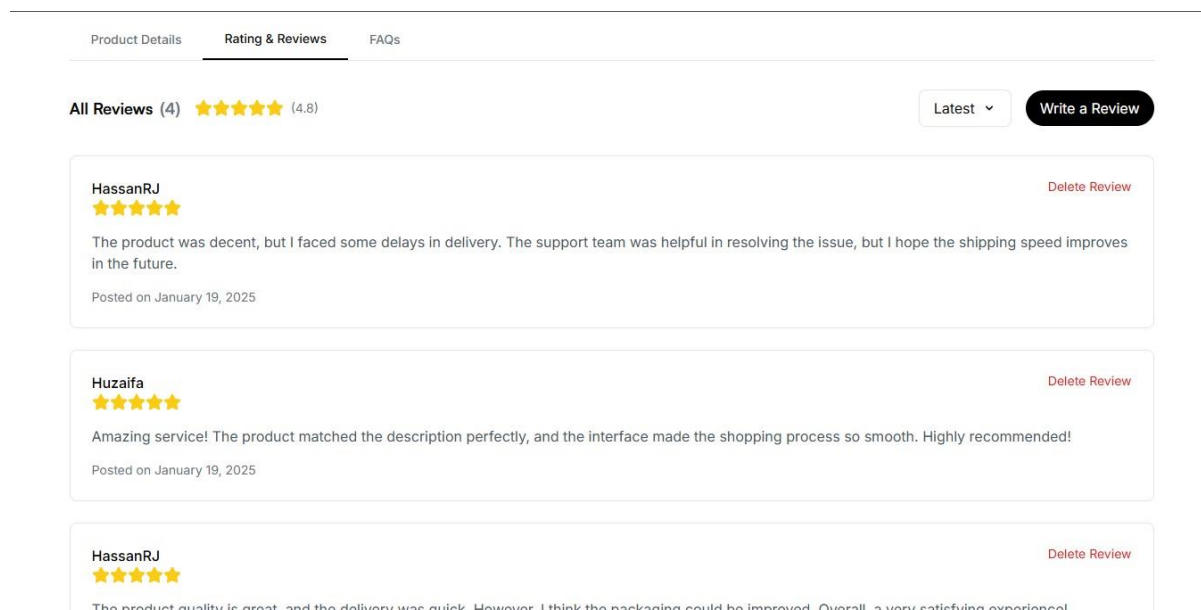
5. Add To Cart

Users can add products to the cart, view them, and remove items dynamically.



6. Reviews and Ratings

Users can add, view, and delete reviews dynamically.



II. Code Snippets

1. ProductCard Component:

Used to render individual product cards in the listing

```

1 import Image from 'next/image'
2 import { Star } from 'lucide-react'
3 import { cn } from '@lib/utils'
4 import { satoshi } from '@app/ui/fonts'
5 import { Product } from '@types/product'
6
7 type ProductCardProps = Pick<Product, '_id' | 'title' | 'price' | 'originalPrice' | 'rating' | 'slug' | 'category' | 'style' | 'colors' | 'sizes'> & {
8   imageUrl: string;
9 }
10
11 export function ProductCard({
12   title,
13   price,
14   originalPrice,
15   rating,
16   imageUrl,
17 }: ProductCardProps) {
18   return (
19     <div className="group cursor-pointer">
20       <div className="relative aspect-[3/4] bg-gray-100 rounded-lg overflow-hidden">
21         {imageUrl ? (
22           <Image
23             src={imageUrl}
24             alt={title}
25             fill
26             sizes="(max-width: 768px) 100vw, (max-width: 1200px) 50vw, 33vw"
27             className="object-cover transition-transform group-hover:scale-105"
28           />
29         ) : (
30           <div className="w-full h-full flex items-center justify-center text-gray-400">
31             No image available
32           </div>
33         )}
34       </div>
35       <div className="mt-4 space-y-2">
36         <div className="flex items-center gap-1">
37           {[...Array(5)].map((_, i) => (
38             <Star
39               key={i}
40               className={cn(
41                 "h-4 w-4",
42                 i < Math.floor(rating) ? "fill-yellow-400 text-yellow-400" : "fill-gray-200 text-gray-200"
43               )}
44             />
45           ))}
46           <span className="text-sm text-gray-500 ml-1">({rating})</span>
47         </div>
48         <h3 className={cn("font-medium line-clamp-1", satoshi.className)}>{title}</h3>
49         <div className="flex items-center gap-2">
50           <span className="font-medium">${price}</span>
51           {originalPrice && originalPrice > price && (
52             <div>
53               <span className={cn(satoshi.className, "text-sm text-gray-500 line-through")}>${originalPrice}</span>
54               <span className="text-sm text-red-600">
55                 -(Math.round(((originalPrice - price) / originalPrice) * 100))%
56               </span>
57             </div>
58           )}
59         </div>
60       </div>
61     </div>
62   )
63 }
64
65

```


2. Dynamic Product Detail Page Routing:

Logic for dynamic product pages:

```

1  import { Metadata } from 'next';
2  import { ProductDetail } from '@components/product/product-details'
3  import { notFound } from 'next/navigation';
4  import { productQuery } from '@sanity/lib/queries';
5  import { client } from '@sanity/lib/client';
6  import { Product } from '@types/product';
7  import { ProductReviews } from '@components/product/product-reviews';
8  import RelatedProducts from '@components/product/related-products';
9  import { Suspense } from 'react';
10 import Loader from '@components/Loader';
11 type ProductPageProps = {
12   params: { slug: string };
13 };
14 export async function generateMetadata({ params }: ProductPageProps): Promise<Metadata> {
15   try {
16     const product: Product | null = await client.fetch(productQuery, { slug: params.slug });
17
18     if (!product) {
19       return {
20         title: 'Product Not Found',
21         description: 'The product you are looking for does not exist.',
22       };
23     }
24     return {
25       title: `${product.title} - Shop.co`,
26       description: product.description,
27       openGraph: {
28         title: `${product.title} - Shop.co`,
29         description: product.description,
30         images: [
31           {
32             url: product.images[0],
33             width: 800,
34             height: 600,
35             alt: product.title,
36           },
37         ],
38       },
39     };
40   } catch (error) {
41     console.error('Error generating metadata:', error);
42     return {
43       title: 'Error',
44       description: 'An error occurred while fetching product information.',
45     };
46   }
47 }
48 export default async function ProductPage({ params }: ProductPageProps) {
49   try {
50     const product: Product | null = await client.fetch(productQuery, { slug: params.slug });
51
52     if (!product) {
53       notFound();
54     }
55     return (
56       <Suspense fallback=<div><Loader /></div>>
57       <ProductDetail product={product} />
58       <ProductReviews product={product} />
59       <RelatedProducts />
60     </Suspense>
61   );
62   } catch (error) {
63     console.error('Error fetching product:', error);
64     throw error;
65   }
66 }
67
68

```

3. Add to Cart Functionality:

Logic for adding and removing items from the cart:

actions.ts

```
1 import { CartItem, CartAction } from '@types/cart'
2
3 export const addToCart = (dispatch: React.Dispatch<CartAction>, item: CartItem) => {
4   dispatch({ type: 'ADD_TO_CART', payload: item })
5 }
6
7 export const removeFromCart = (dispatch: React.Dispatch<CartAction>, item: Omit<CartItem, 'quantity'>) => {
8   dispatch({ type: 'REMOVE_FROM_CART', payload: item })
9 }
10
11 export const updateQuantity = (dispatch: React.Dispatch<CartAction>, item: CartItem) => {
12   dispatch({ type: 'UPDATE_QUANTITY', payload: item })
13 }
14
15 export const clearCart = (dispatch: React.Dispatch<CartAction>) => {
16   dispatch({ type: 'CLEAR_CART' })
17 }
18
```

Function from Frontend HandleAddToCart

```
1 const handleAddToCart = () => {
2   const item = {
3     id: product._id,
4     name: product.title,
5     price: product.price,
6     image: product.images[0],
7     color: selectedColor.toString(),
8     size: selectedSize,
9     quantity: quantity
10  }
11 }
12
13 const existingItem = state.items.find(
14   i => i.id === item.id && i.color === item.color && i.size === item.size
15 )
16
17 if (existingItem) {
18   const newQuantity = existingItem.quantity + quantity
19   if (newQuantity > product.inventory) {
20     toast.error(`Sorry, we only have ${product.inventory} items in stock.`)
21     return
22   }
23 }
24
25 addToCart(dispatch, item)
26 toast.success('Product added to cart!')
27 }
```

Dynamic Public Marketplace – SHOP.CO By Hassan RJ