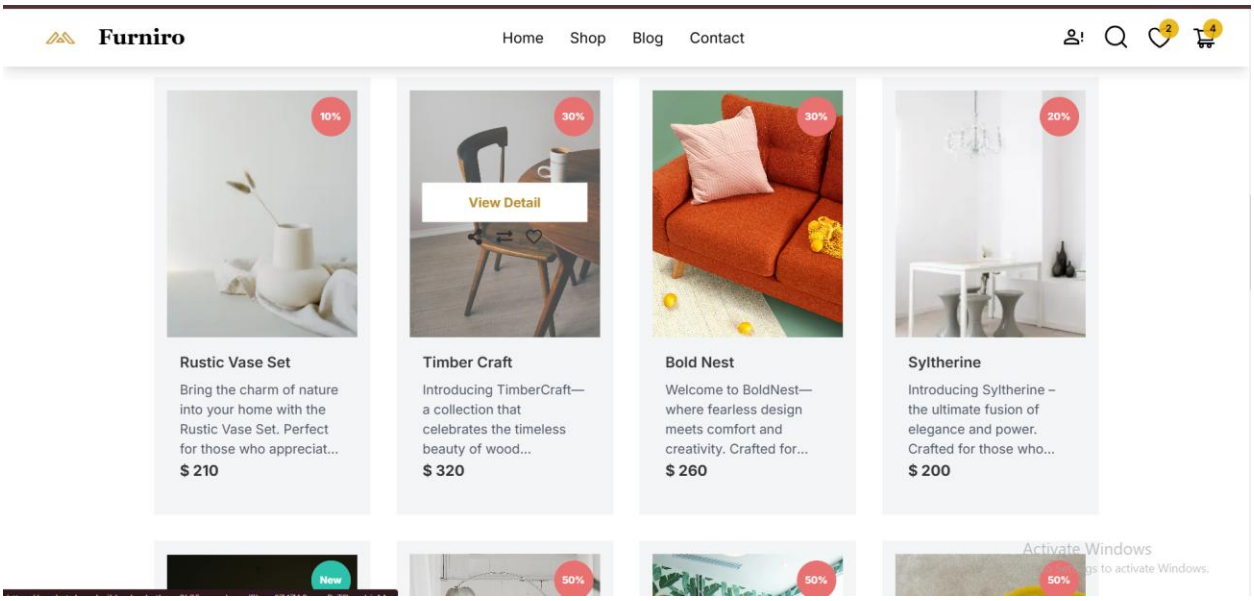
Day 4 - Dynamic Frontend Components Furniro

On Day 4, I focused on Dynamic Frontend Components using Sanity CMS and APIs for a real-world client project, Furniro. I implemented multiple dynamic functionalities on my responsive UI, including search, filters, product listing, cart management, add to cart, wishlist, product comparison, Notification component and dynamic updates.Emphasis was on modular design, state management, and creating a seamless UI/UX with a professional workflow, enhancing both usability and content management.

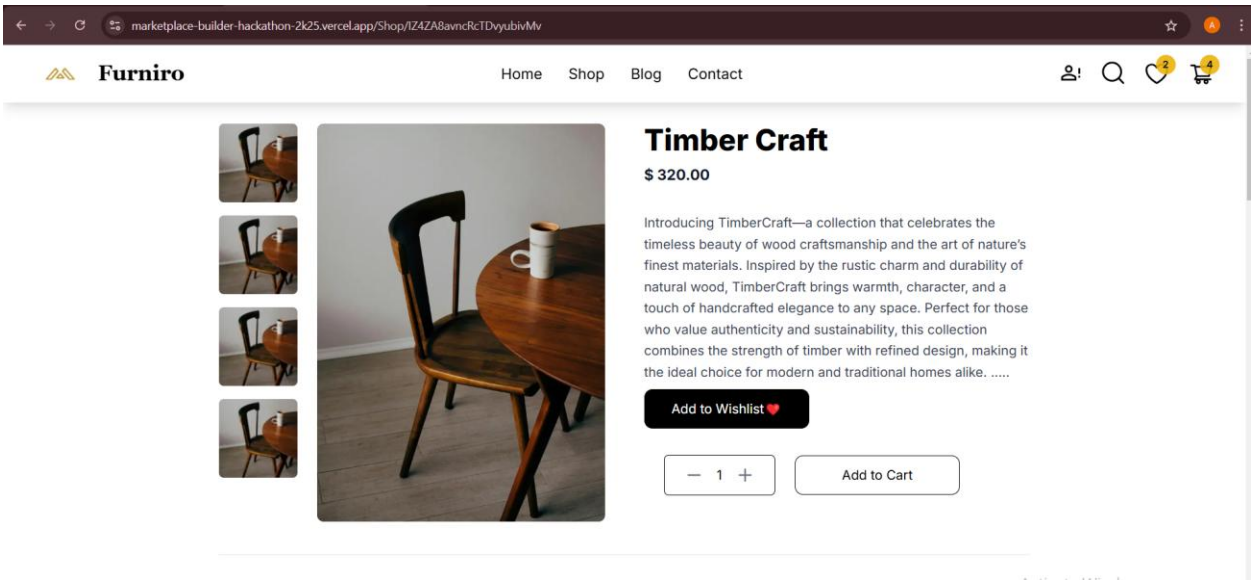
Product Listing Page – Furniro:

This page displays all products, where customer can view a wide range of Furniture. All dynamic data is fetched from Sanity CMS and presented in a well-structured and visually appealing UI, ensuring a seamless and engaging shopping experience.



Product Detail Page – Furniro:

This page is designed for customers seeking detailed information about a specific furniture item. Here, customers can explore comprehensive product details, including add to cart**,** wishlist**,** and other key specifications. Additionally, related products are displayed to offer a more personalized shopping experience. Each product is uniquely identified by a unique ID, enabling dynamic routing and seamless data rendering, delivering a refined and user-friendly experience for the customer.

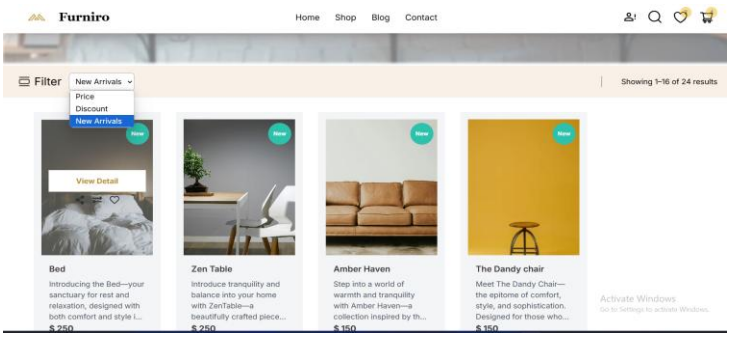


Product Filtering on Furniro:

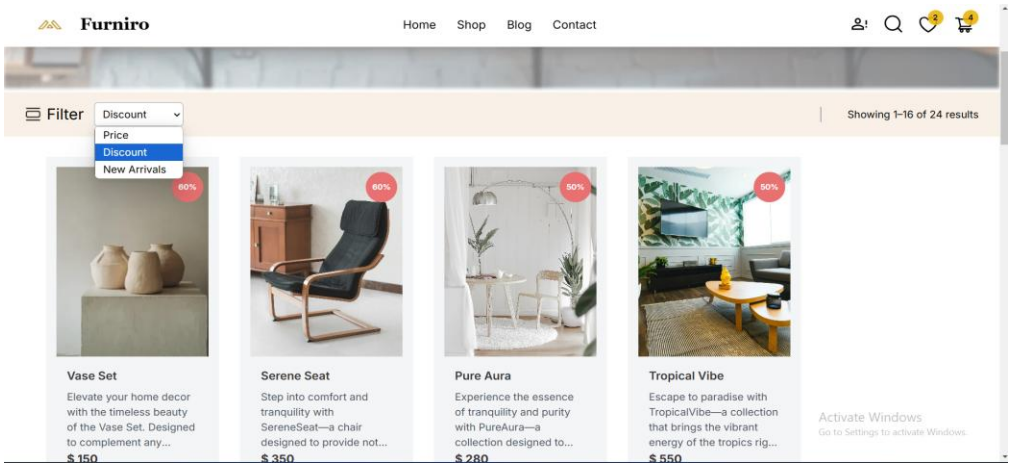
On Furniro, customers can filter products in three ways:

1. **By New Arrivals** – View the latest products.
2. **By Discount** – Filter products with available discounts.
3. **By Price** – Filter products based on price range.

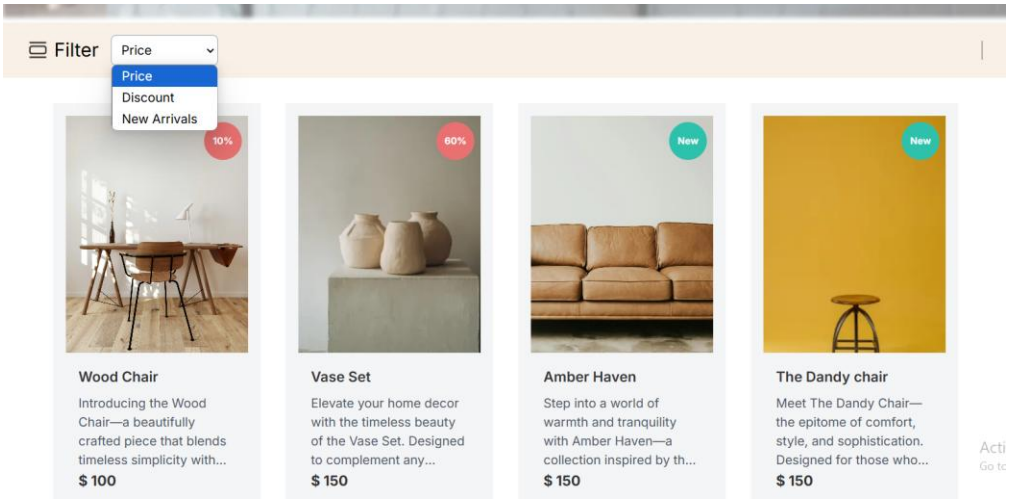
By New Arrivals:



By Discount:

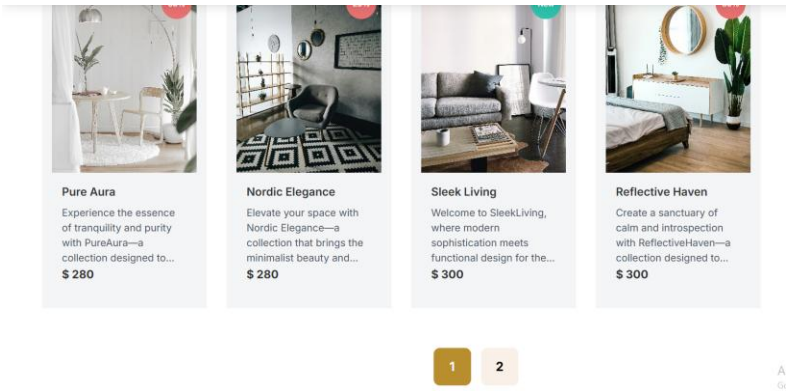


By Price:



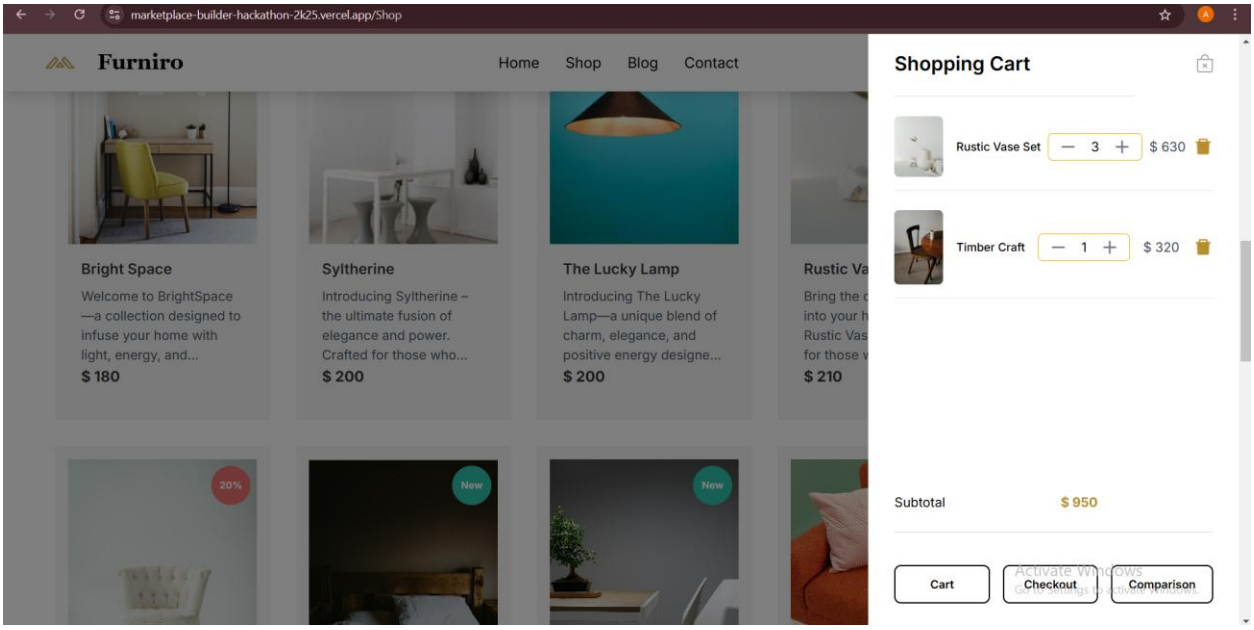
Pagination:

I have created two pages for pagination, allowing customers to easily navigate through multiple pages of products for a seamless shopping experience.



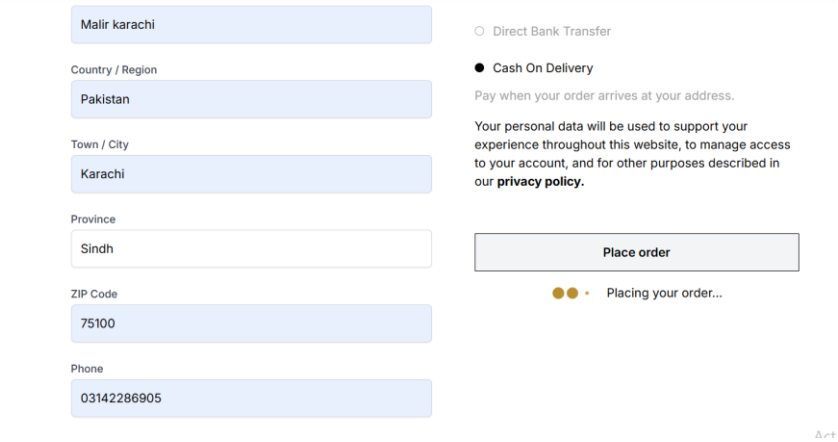
Shopping Cart:

In this cart, users can view and manage their selected items. It offers a smooth experience to review, adjust quantities, and proceed to checkout effortlessly.



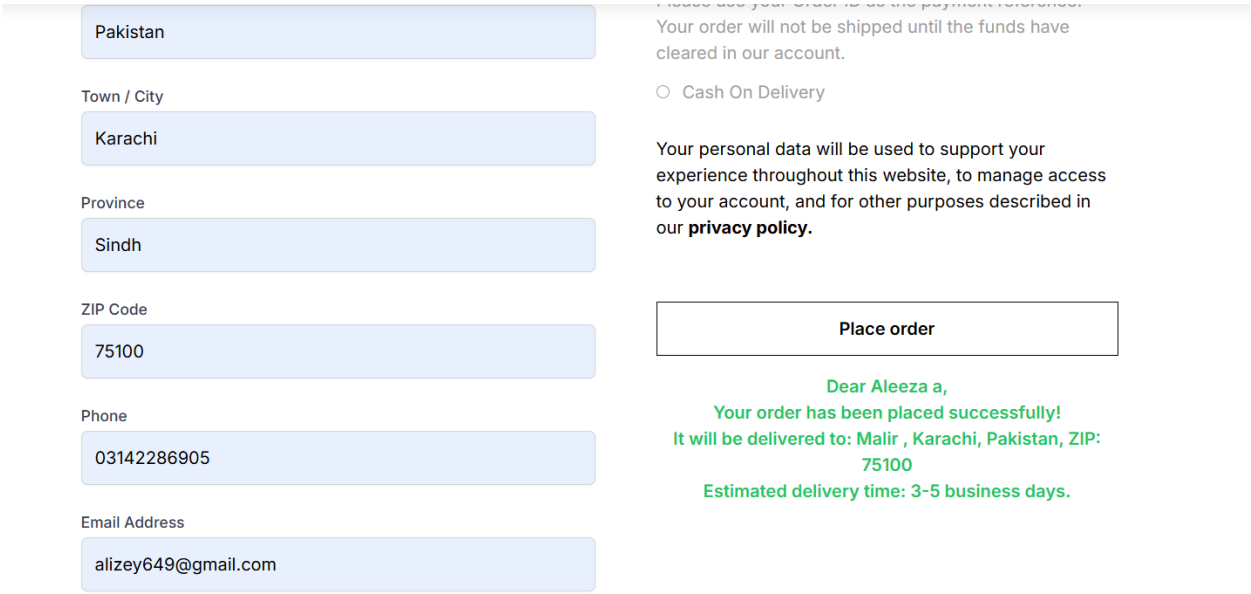
Checkout Page:

On the checkout page, all cart items are carried over for the final review. Customer can select their preferred payment method and securely place their order.



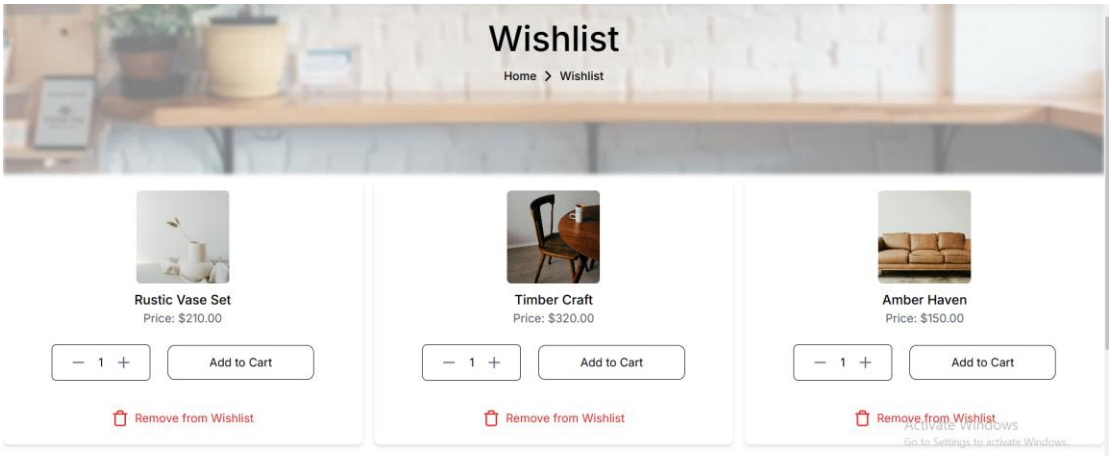
Order Successful:

When the order is successfully placed, the customer will see a confirmation message along with their order details.



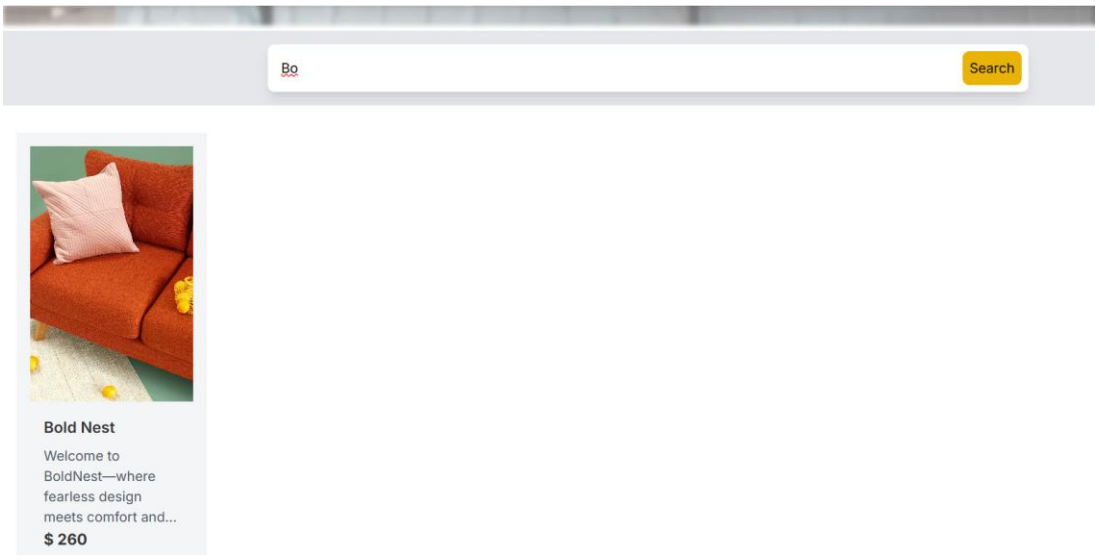
Wishlist Component:

The wishlist component allows customers to view and manage their selected items. Each item includes options to remove from Wishlist or Add to Cart**,** providing a smooth and intuitive shopping experience.



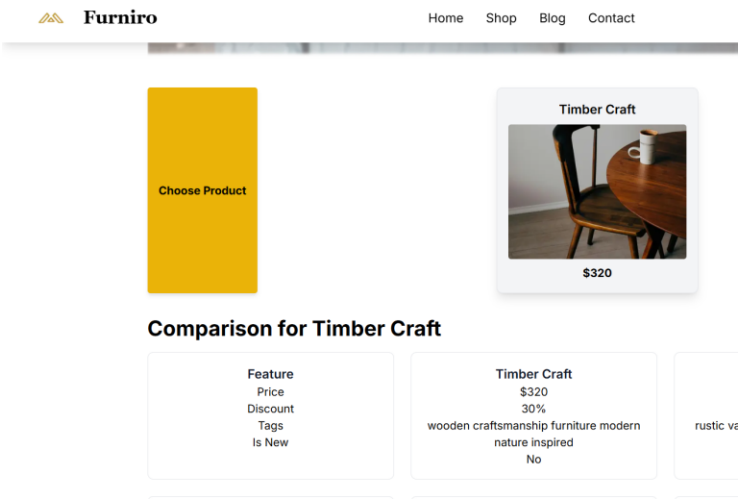
Search Component:

The search component allows customers to easily find any furniture item. If a product is not available, a '**Not Found**' message is displayed, ensuring a smooth search experience.



Comparison Page:

This comparison page allows customers to select multiple furniture items and view a detailed comparison side by side. Customers can compare features, specifications, and prices to make informed decisions.



Dynamic Product Page Code Snippet:

```tsx
const ProductPage = () => {
  const { id } = useParams();
  const [product, setProduct] = useState<Product | null>(null);
  const [image, setImage] = useState<string>("");
  const [loading, setLoading] = useState<boolean>(true);
  const [products, setProducts] = useState<Product[]>([]);
  const [reviews, setReviews] = useState([]);
  const handleShare = () => {
    navigator.clipboard.writeText(window.location.href);
    alert("Product link copied to clipboard!");
  };
  // State to manage visibility of sections
  const [activeSection, setActiveSection] = useState<'description' | 'additional' | 'reviews'>('description');

  useEffect(() => {
    const fetchProduct = async () => {
      try {
        const fetchedProducts = await client.fetch<Product[]>(`
          *[_type == "product" && _id == $id]{
            _id,
            title,
            price,
            discountPercentage,
            tags,
            isNew,
            description,
            "productImage": productImage.asset->url
          }
        `, { id });

        if (fetchedProducts.length > 0) {
          setProduct(fetchedProducts[0]);
          setImage(fetchedProducts[0].productImage);
        }
      } catch (error) {
        console.error("Failed to fetch product", error);
      } finally {
        setLoading(false);
      }
    };

    fetchProduct();
  }, [id]);
```

Search Bar Code Snippet:

```tsx
const Search = () => {
  const [query, setQuery] = useState("");
  const [products, setProducts] = useState<Product[]>([]);
  const [filteredProducts, setFilteredProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState<boolean>(true);


  useEffect(() => {
    const fetchProducts = async () => {
      const query = await client.fetch(
        `*[_type == "product"]{
          _id,
          title,
          description,
          price,
          discountPercentage,
          tags,
          isNew,
          "productImage": productImage.asset->url
        }`
      );
      setProducts(query);
      setFilteredProducts(query);
      setLoading(false);
    };

    fetchProducts();
  }, []);

  useEffect(() => {
    if (query.trim() === "") {
      setFilteredProducts(products);
    } else {
      const filtered = products.filter((product) =>
        product.title.toLowerCase().includes(query.toLowerCase())
      );
      setFilteredProducts(filtered);
    }
  }, [query, products]);

  const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setQuery(e.target.value);
  };

  const handleSearchSubmit = (e: React.FormEvent) => {
    e.preventDefault();
  };

```

Self-Validation Checklist:

Frontend Component Development	Styling and Responsiveness	Code Quality	Documentation and Submission	Final Review
✓	✓	✓	✓	✓