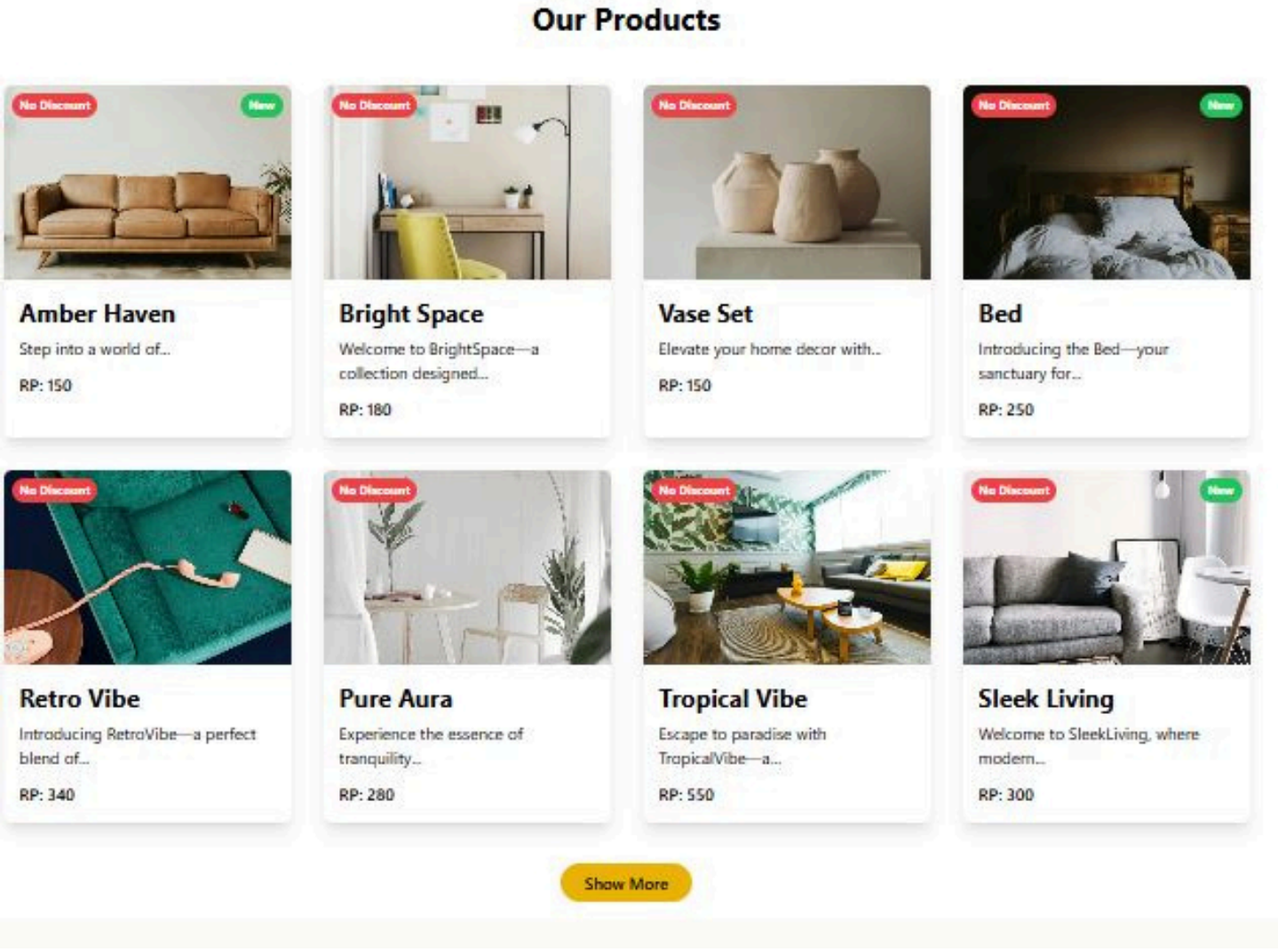
# Day 4 - Dynamic Frontend Components - [Furniro]

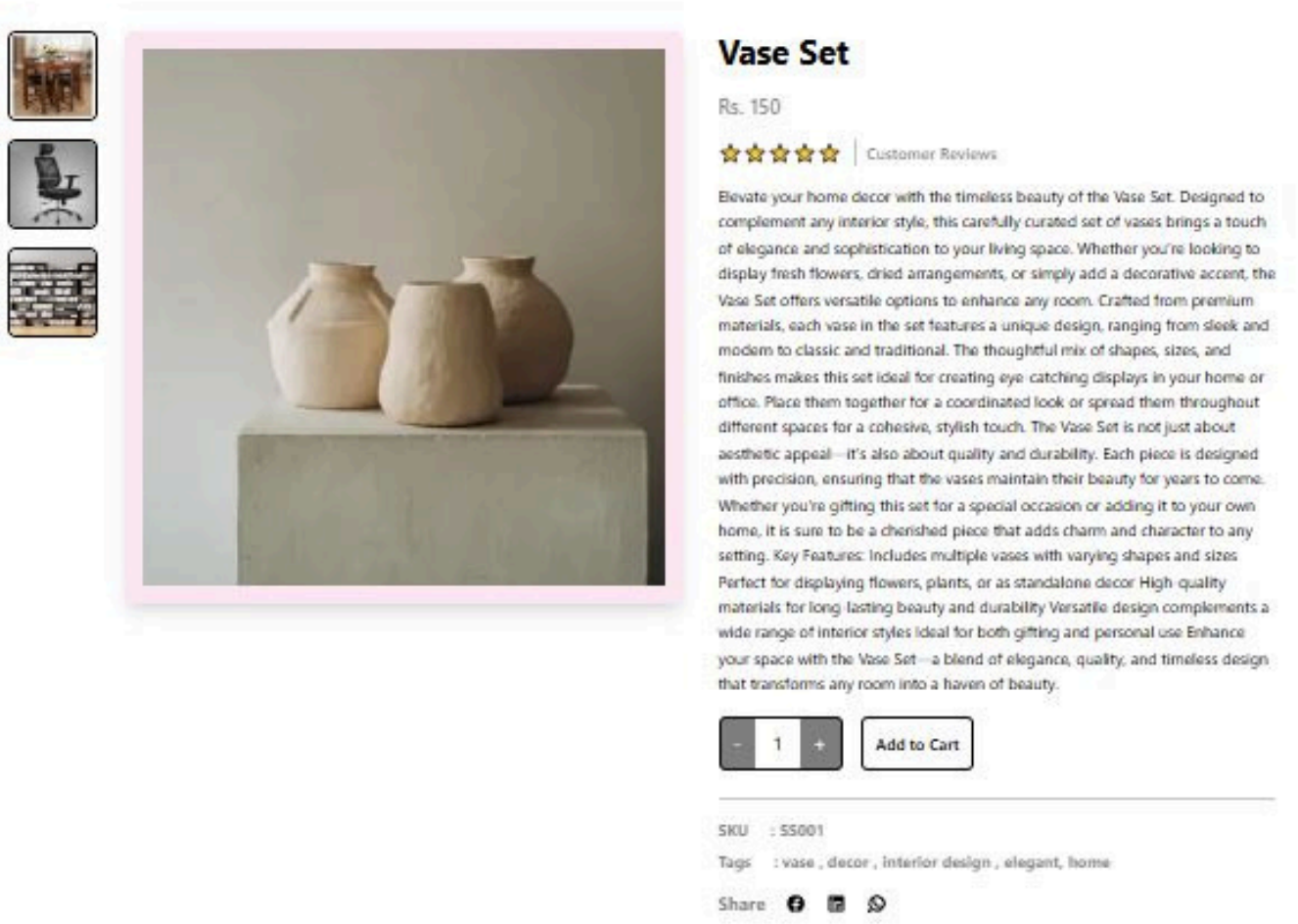
**PREPARED BY : ZOHAIB MEMON**

## 1. FUNCTIONAL DELIVERABLES

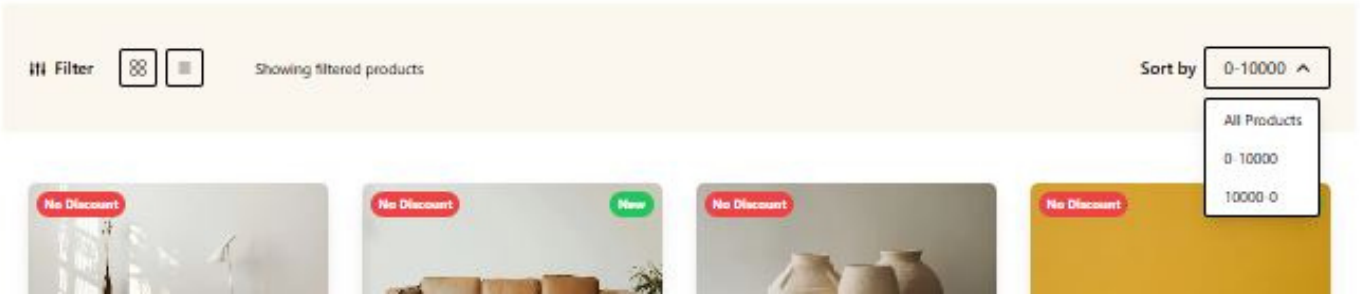
**Product Listing Page:**



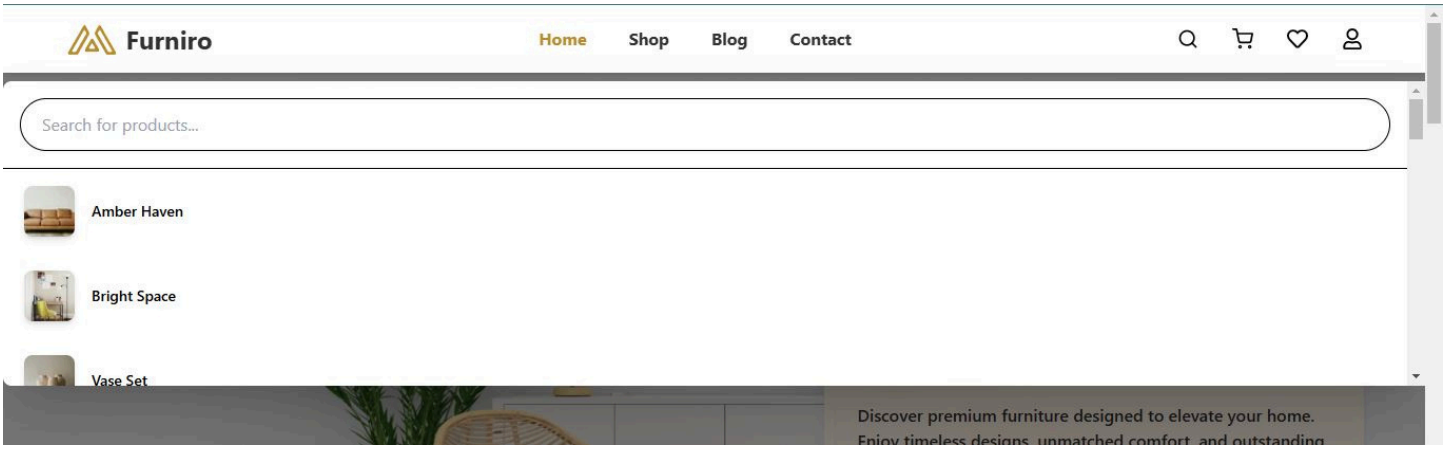

**Product Detail Pages:**



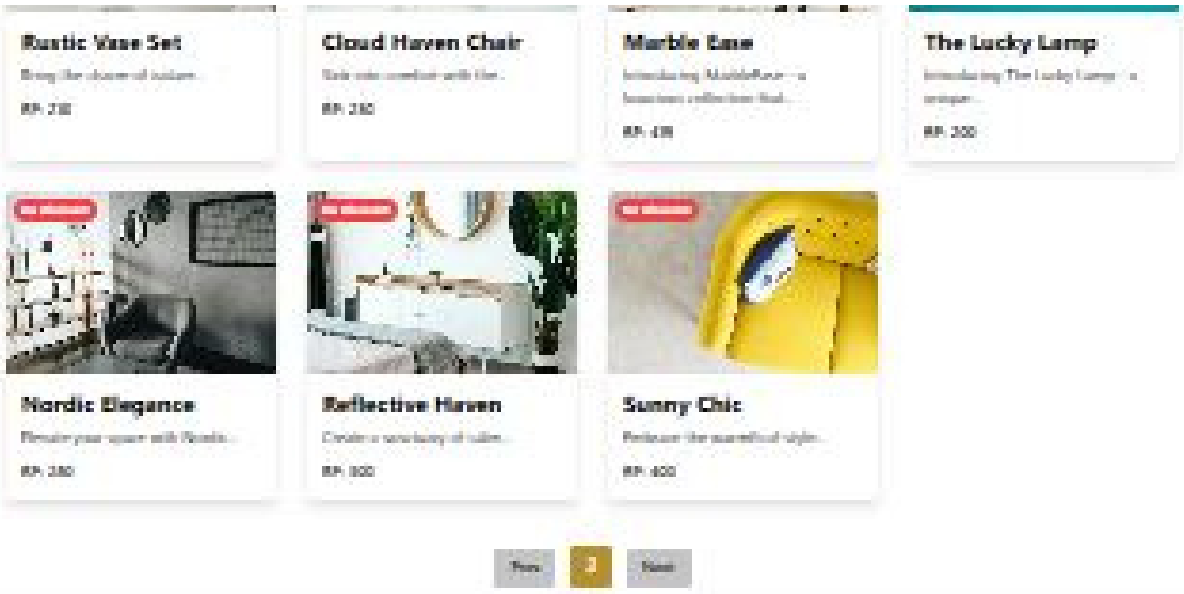

**Filters Bar:** Filtering products by tags, price low to high and price high to low.



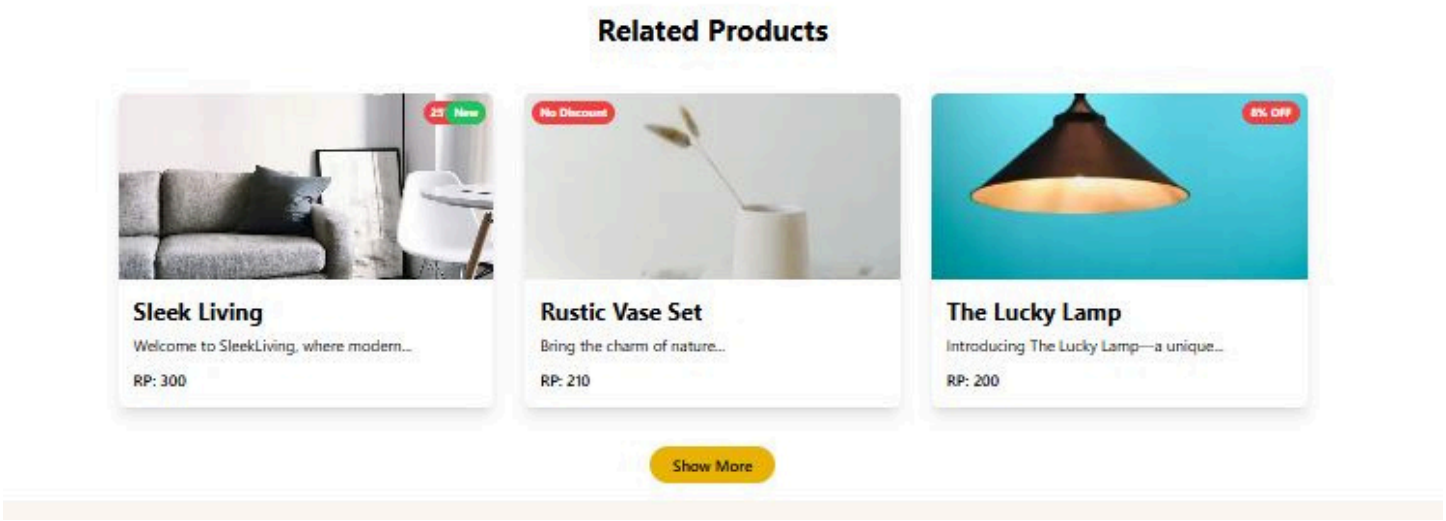

**Search Bar:** Searching products dynamically based on title.



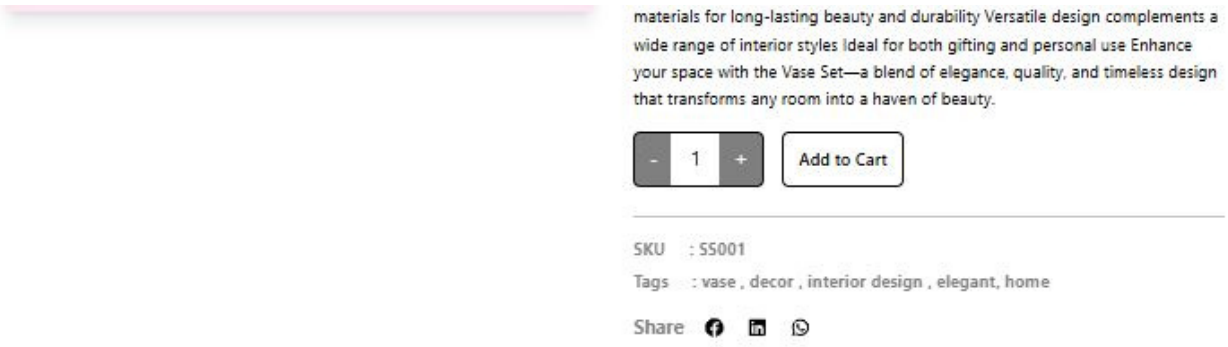

**Pagination:** Navigating between multiple product pages.




**Related Products:** Suggest similar or complementary products on the product detail page Fetch data based on tags.

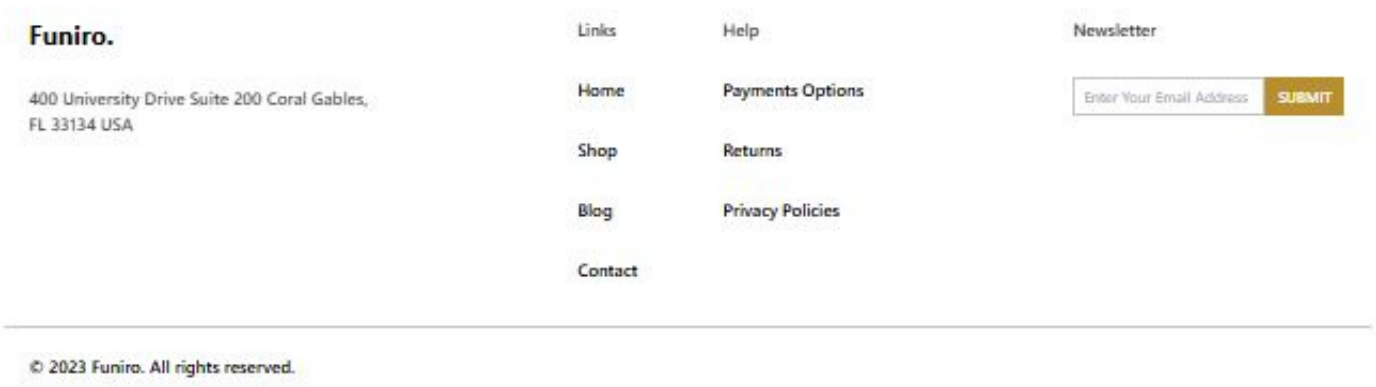



**Social Media Sharing:** Enable users to share products or product link directly to social media platforms.

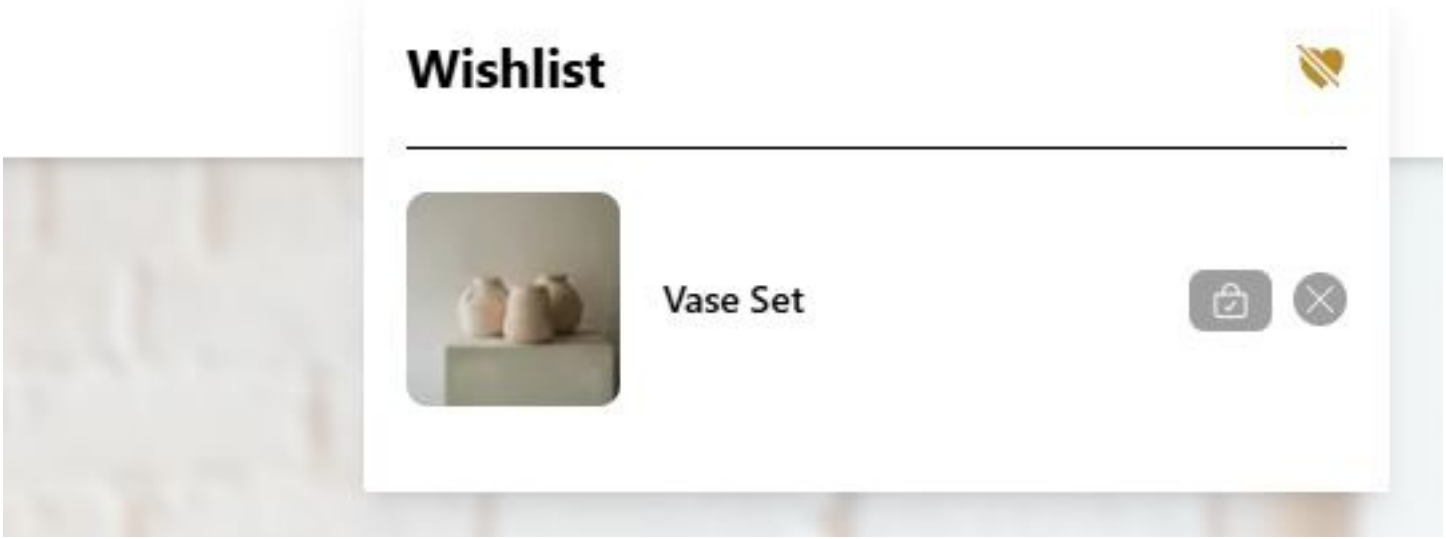



**Navbar:** include links and icons to key pages (e.g., Home, Shop, Blog, Contact).

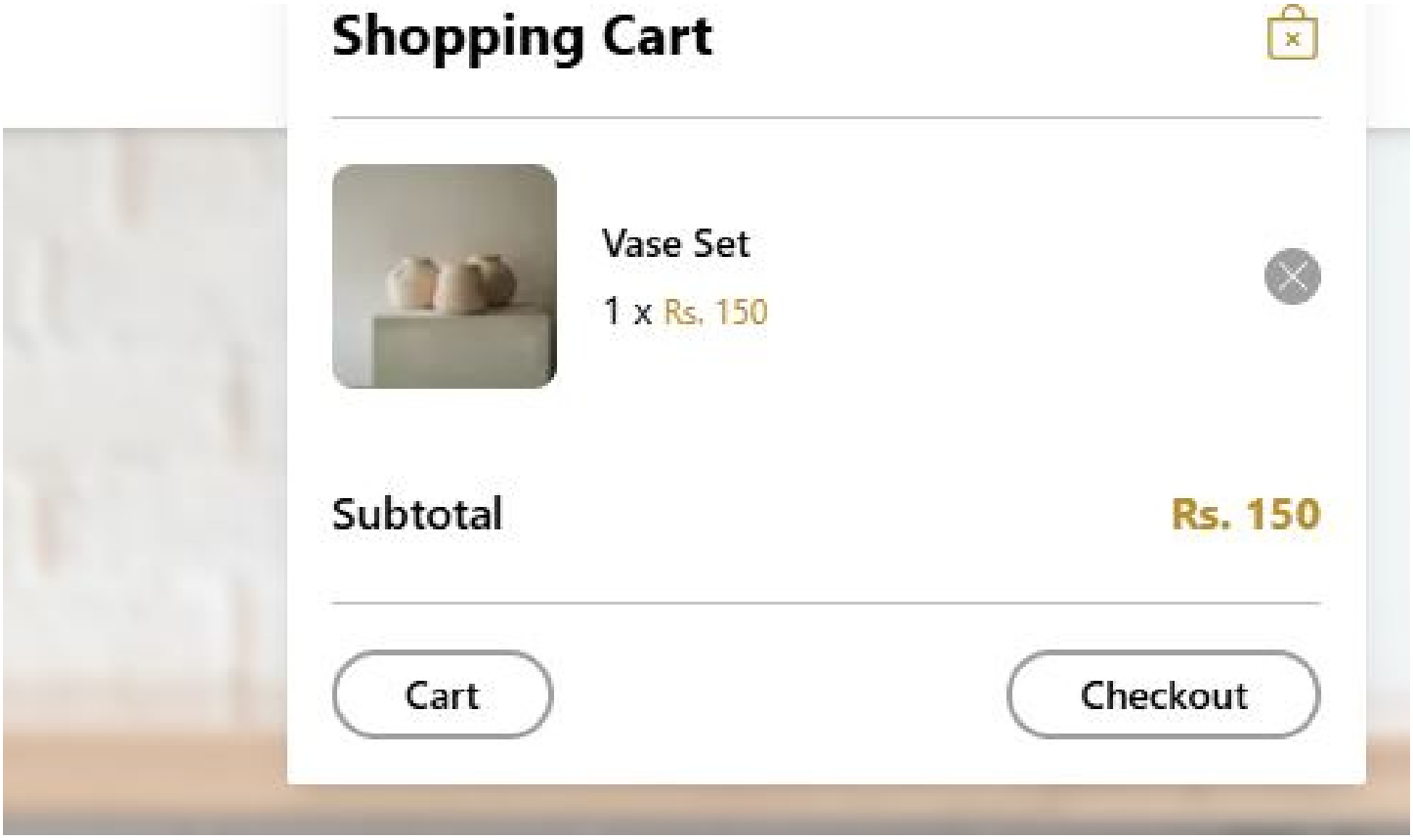

**Footer:** include links to key pages (e.g., Home, Shop, Blog, Contact).




**Wishlist Component:** Allow users to save products for future reference.




**Cart Component:** Display added items, quantity, remove button and total price .

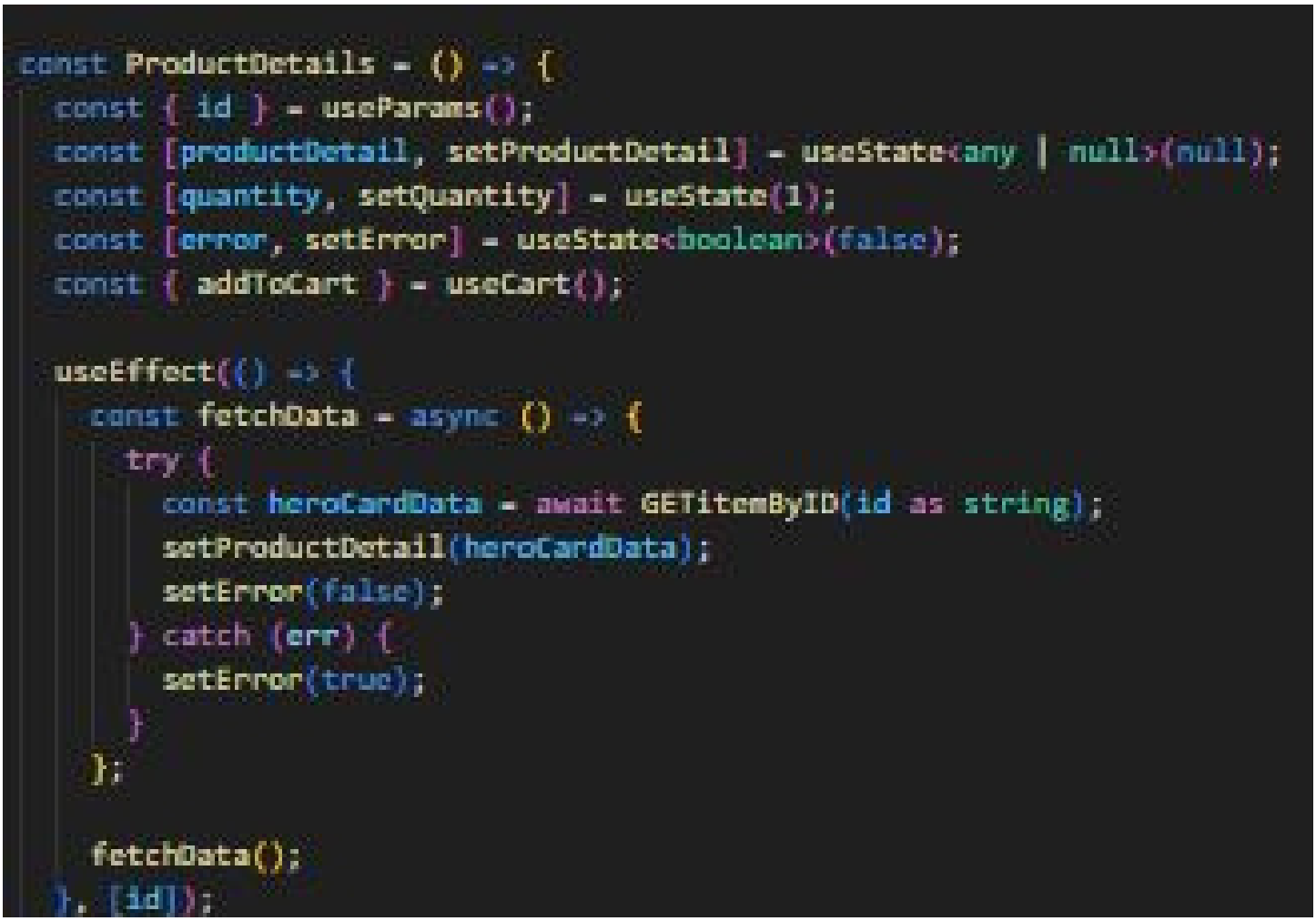



## 2. CODE DELIVERABLES

**Product Listing Page:**

```
  price: string;
  oldPrice?: string;
  discountPercentage: string;
  imageUrl: string;
  isNew: boolean;
  tags: string[];
}

const itemsPerPage = 12;

const ProductList: React.FC = () => {
  const [products, setProducts] = useState<SanityProduct[]>([]);
  const [filteredProducts, setFilteredProducts] = useState<SanityProduct[]>([]);
  const [currentPage, setCurrentPage] = useState(1);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<boolean>(false);

  const totalPages = Math.ceil(filteredProducts.length / itemsPerPage);
  const currentProducts = filteredProducts.slice(
    (currentPage - 1) * itemsPerPage,
    currentPage * itemsPerPage
  );

  const fetchProducts = async () => {
    setLoading(true);
    try {
      const response = await fetch("http://localhost:3000/api/products");
      const data = await response.json();
      setProducts(data);
      setFilteredProducts(data);
      setError(false);
    } catch {
      setError(true);
```

**Product Detail Pages:**



```
const ProductDetails = () => {
  const { id } = useParams();
  const [productDetail, setProductDetail] = useState<any | null>(null);
  const [quantity, setQuantity] = useState(1);
  const [error, setError] = useState<boolean>(false);
  const { addToCart } = useCart();

  useEffect(() => {
    const fetchData = async () => {
      try {
        const heroCardData = await GETitemByID(id as string);
        setProductDetail(heroCardData);
        setError(false);
      } catch (err) {
        setError(true);
      }
    };

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

**Filters Bar:** Filtering products by tags, price low to high and price high to low.

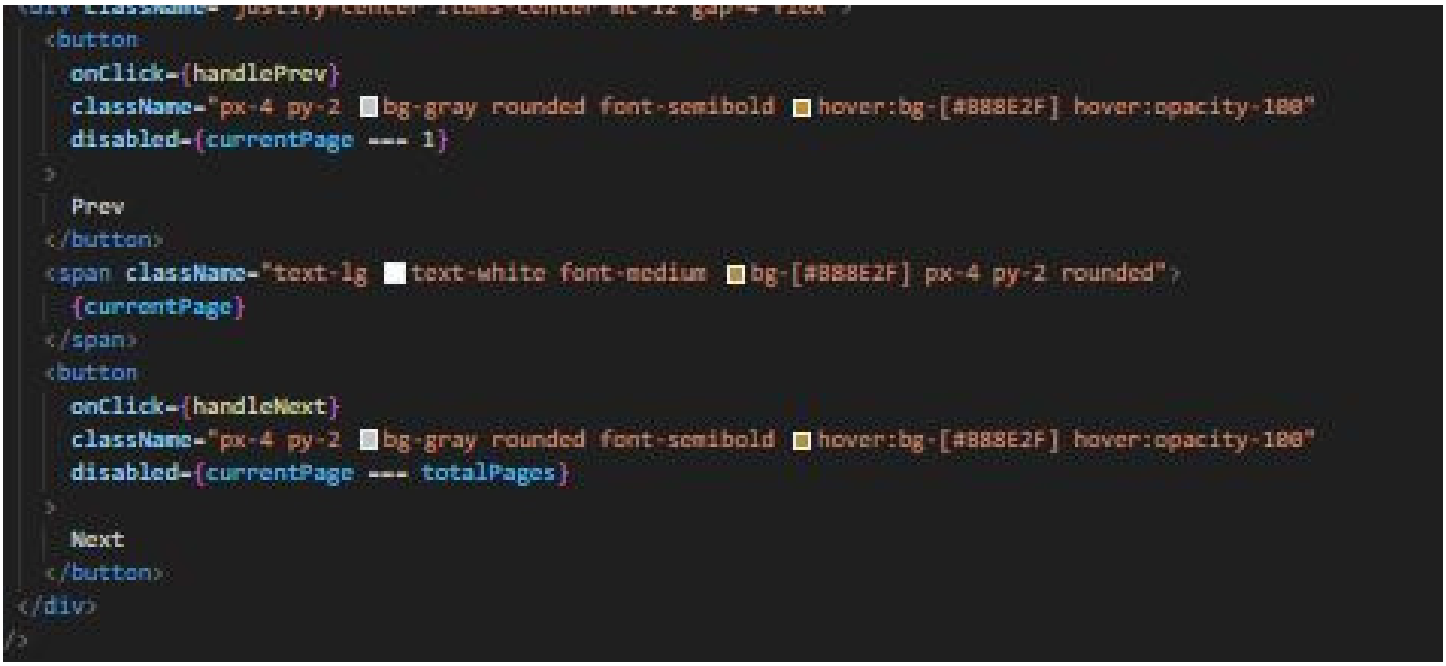
```tsx
interface FilterBarProps {
  applyFilter: (option: string) => void;
}

const FilterBar: React.FC<FilterBarProps> = ({ applyFilter }) => {
  const [sortOption, setSortOption] = useState("All Products");
  const [isDropdownOpen, setIsDropdownOpen] = useState(false);

  const sortingOptions = ["All Products", "0-10000", "10000-0"];

  const handleSortChange = (option: string) => {
    setSortOption(option);
    setIsDropdownOpen(false);
    applyFilter(option);
  };

  return (
    <div className="bg-[#FBF6EE] py-10">
      <div className="flex flex-wrap items-center justify-between gap-3 sm:gap-4 max-w-7xl mx-auto px-4 sm:px-6">
        <div className="flex flex-col items-center text-center sm:flex-row sm:text-left sm:items-center gap-3 sm:gap-6">
          <button className="flex items-center gap-2 text-black font-medium text-sm sm:text-base hover:text-gray-700">
            <HiAdjustments className="text-base sm:text-lg" />
            <span>Filter</span>
          </button>

          <div className="flex items-center gap-2">
            <button className="p-2 border border-gray-300 rounded hover:border-black text-sm sm:text-base">
              <BsGrid />
            </button>
            <button className="p-2 border border-gray-300 rounded hover:border-black text-sm sm:text-base">
              <BsList />
            </button>
          </div>
```



**Search Bar:** Searching products dynamically based on title.

```tsx
<div
  className={`absolute top-14 left-0 right-0 mt-4 bg-white shadow-2xl rounded-lg max-h-72 overflow-y-auto z-50 transition-all ease-out duration-300 ${
    isSearchOpen ? "opacity-100 scale-100" : "opacity-0 scale-95"
  }`}
>
  <div className="p-4 border-b border-gray-200">
    <input
      type="text"
      placeholder="Search for products..."
      value={searchQuery}
      onChange={(e) => setSearchQuery(e.target.value)}
      className="w-full px-5 py-3 text-gray-800 bg-gray-100 rounded-full border border-gray-300 focus:ring-2 focus:ring-[#B88E2F] focus:border-[#B88E2F] transition duration
    />
  </div>
  <div className="max-h-100 overflow-y-auto transition-all duration-300 ease-in-out">
    {searchQuery && filteredProducts.length === 0 ? (
      <div className="flex items-center justify-center h-32 text-gray-500">
        <span className="animate-pulse">Searching...</span>
      </div>
    ) : filteredProducts.length > 0 ? (
      filteredProducts.map((product) => {
        const escapedSearchQuery = searchQuery.replace(
          /[.*+?^=!:${}()|\[\]\/\\]/g,
          "\\$&"
        );

        const highlightedTitle = searchQuery
          ? product.title
              .split(new RegExp(`(${escapedSearchQuery})`, "gi"))
              .map((part: string, index: number) =>
                part.toLowerCase() === searchQuery.toLowerCase() ? (
                  <span
                    key={index}
                    className="bg-[#B88E2F] text-white px-1 py-0.5 rounded"
                  >
                    {part}
                  </span>
                ) : (
                  part
                )
              )
          : product.title;
```

**Pagination:** Navigating between multiple product pages.



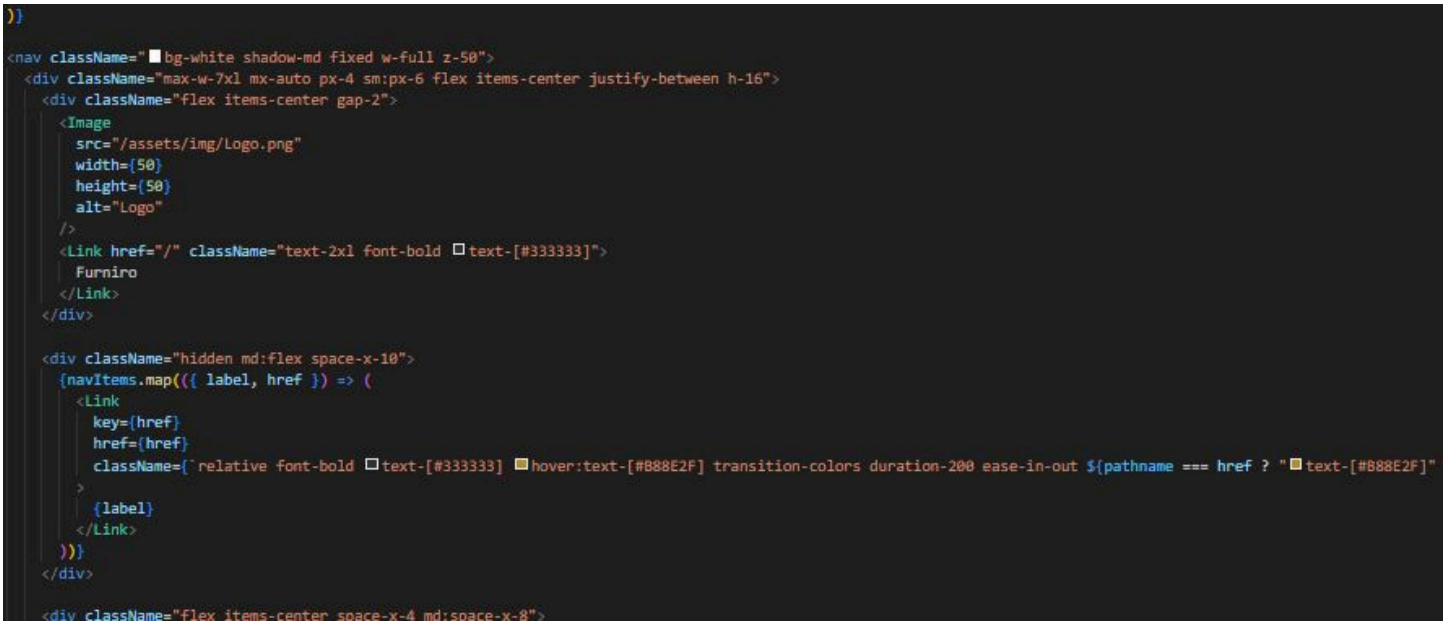

**Related Products:** Suggest similar or complementary products on the product detail page Fetch data based on tags.

```tsx
interface RelatedProductProps {
  productId: string;
  tags: string[];
}

const RelatedProduct: React.FC<RelatedProductProps> = ({ productId, tags }) => {
  const [relatedProducts, setRelatedProducts] = useState<SanityProduct[]>([]);
  const [error, setError] = useState<boolean>(false);

  useEffect(() => {
    const fetchRelatedProducts = async () => {
      try {
        console.log("Fetching related products with tags:", tags, "and productId:", productId);

        const data = await GETRelatedProductsByTags(tags, productId);
        if (data.length > 0) {
          console.log("Related products found:", data);
          setRelatedProducts(data);
        } else {
          console.log("No related products found. Fetching all products as fallback.");
          const fallbackResponse = await GET();
          const fallbackData = await fallbackResponse.json();
          setRelatedProducts(fallbackData);
        }
      } catch (err) {
        console.error("Error fetching related products:", err);
        setError(true);
      }
    };

    fetchRelatedProducts();
  }, [productId, tags]);

  if (error) {
```

**Social Media Sharing:** Enable users to share products or product link directly to social media platforms.

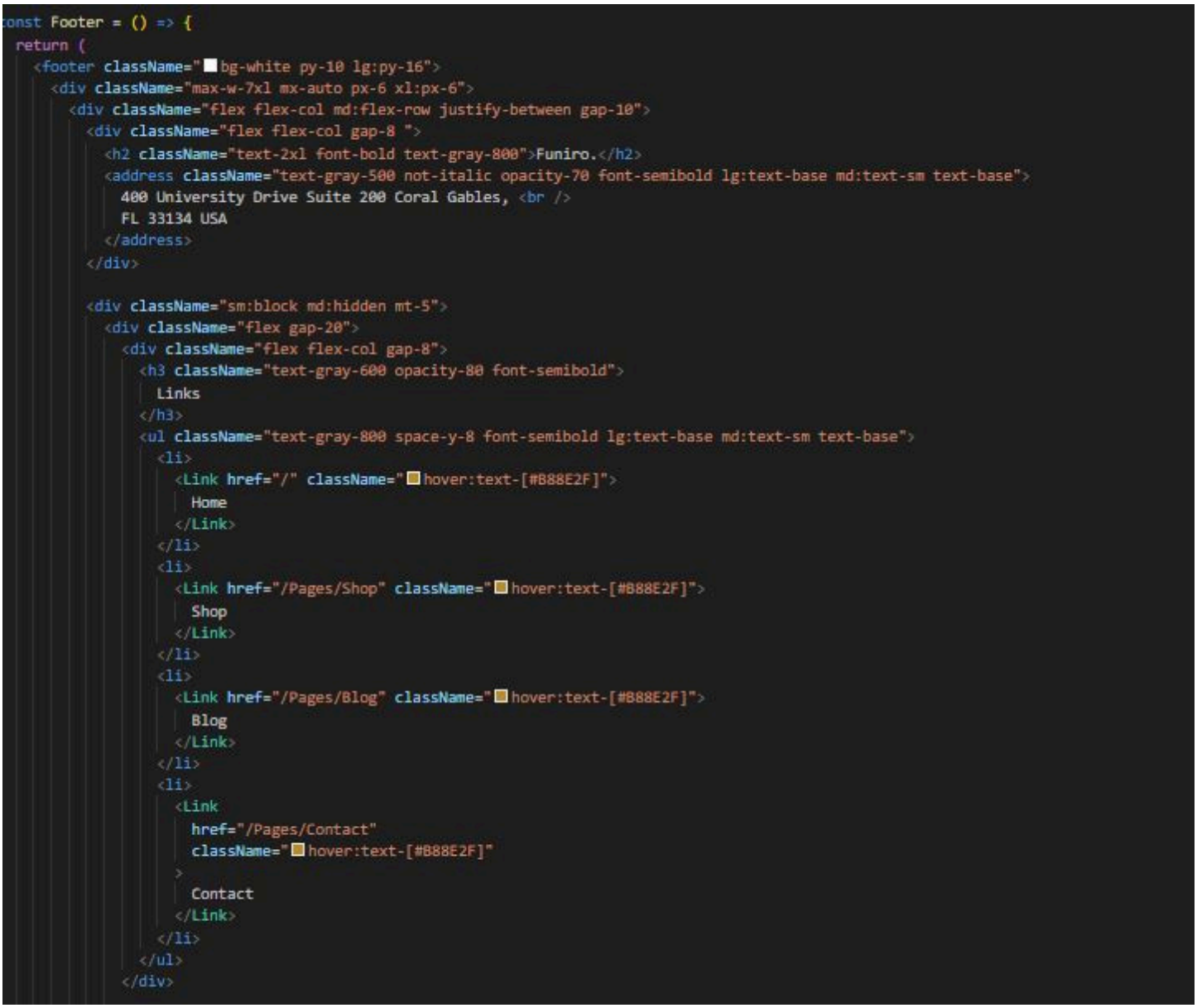
```tsx
        <div className="flex items-center space-x-4 mt-4">
<span className="text-sm sm:text-base font-bold text-black opacity-50">
 Share
</span>

<Link
 href={`https://www.facebook.com/sharer/sharer.php?u=${encodeURIComponent(window.location.href)}&quote=Check%20out%20this%20product%20on%20our%20website!`}
 target="_blank"
 className="w-5 h-5 flex items-center justify-center bg-gray-200 rounded-full hover:text-[#888E2F] transition"

 <FaFacebook size={16} />
</Link>

<Link
 href={`https://www.linkedin.com/shareArticle?mini=true&url=${encodeURIComponent(window.location.href)}&title=Amazing%20Product%20on%20Our%20Website&summary=Check%20out%20this%20product%20on%20our%20website%20now!`}
 target="_blank"
 className="w-5 h-5 flex items-center justify-center bg-gray-200 rounded-full hover:text-[#888E2F] transition"

 <FaLinkedin size={16} />
</Link>

<Link
 href={`https://wa.me/?text=Check%20out%20this%20product%20on%20our%20website!%20${encodeURIComponent(window.location.href)}`}
 target="_blank"
 className="w-5 h-5 flex items-center justify-center bg-gray-200 rounded-full hover:text-[#888E2F] transition"

 <FaWhatsapp/>
</Link>
</div>
```

**Navbar:** include links and icons to key pages (e.g., Home, Shop, Blog, Contact).



```
)}

<nav className="bg-white shadow-md fixed w-full z-50">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 flex items-center justify-between h-16">
    <div className="flex items-center gap-2">
      <Image
        src="/assets/img/Logo.png"
        width={50}
        height={50}
        alt="Logo"
      />
      <Link href="/" className="text-2xl font-bold text-[#333333]">
        Furniro
      </Link>
    </div>

    <div className="hidden md:flex space-x-10">
      {navItems.map(({ label, href }) => (
        <Link
          key={href}
          href={href}
          className={`relative font-bold text-[#333333] hover:text-[#B88E2F] transition-colors duration-200 ease-in-out ${pathname === href ? "text-[#B88E2F]"
        >
          {label}
        </Link>
      ))}
    </div>

    <div className="flex items-center space-x-4 md:space-x-8">
```

**Footer:** include links to key pages (e.g., Home, Shop, Blog, Contact).

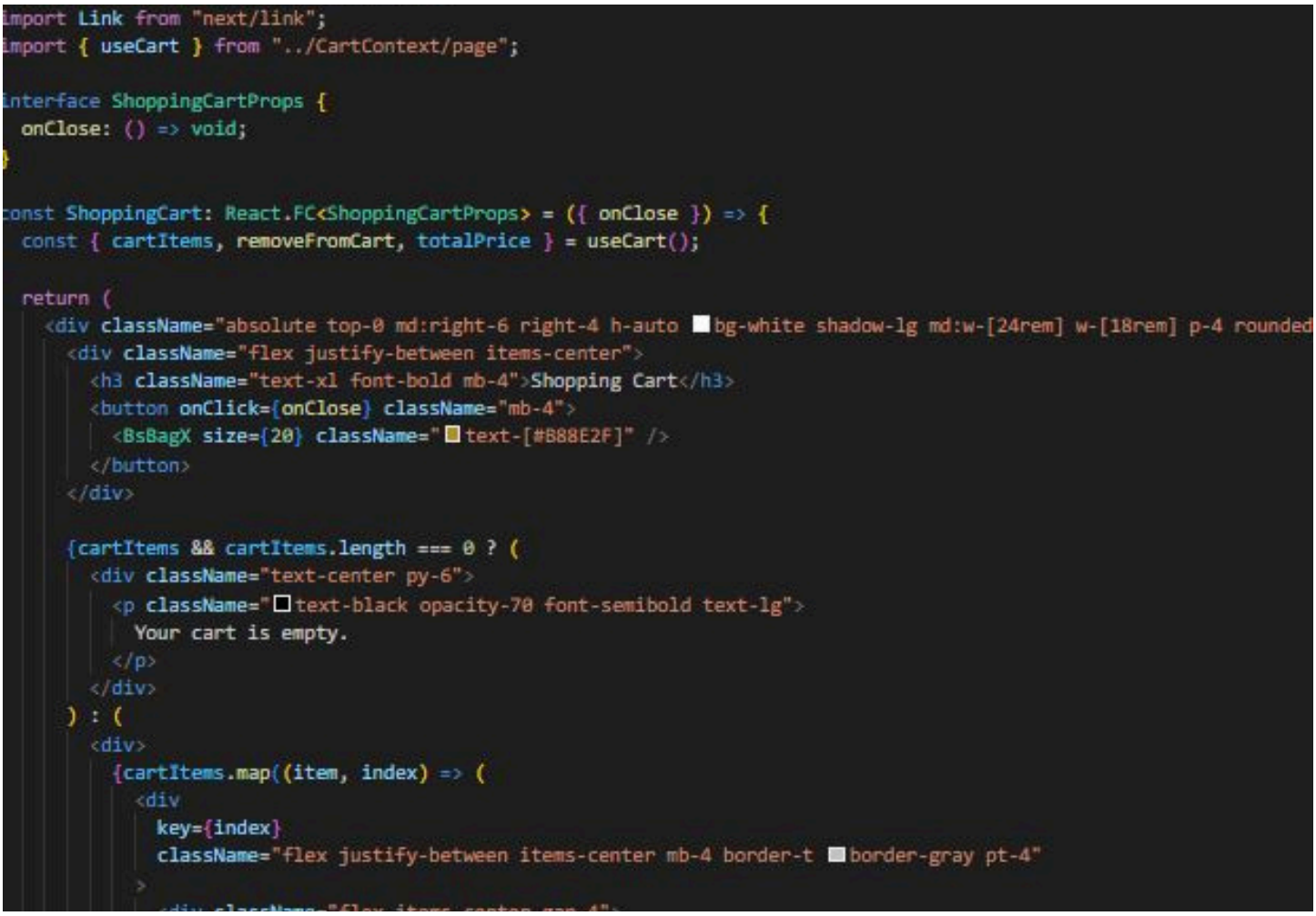


```
const Footer = () => {
  return (
    <footer className="bg-white py-10 lg:py-16">
      <div className="max-w-7xl mx-auto px-6 xl:px-6">
        <div className="flex flex-col md:flex-row justify-between gap-10">
          <div className="flex flex-col gap-8 ">
            <h2 className="text-2xl font-bold text-gray-800">Funiro.</h2>
            <address className="text-gray-500 not-italic opacity-70 font-semibold lg:text-base md:text-sm text-base">
              400 University Drive Suite 200 Coral Gables, <br />
              FL 33134 USA
            </address>
          </div>

          <div className="sm:block md:hidden mt-5">
            <div className="flex gap-20">
              <div className="flex flex-col gap-8">
                <h3 className="text-gray-600 opacity-80 font-semibold">
                  Links
                </h3>
                <ul className="text-gray-800 space-y-8 font-semibold lg:text-base md:text-sm text-base">
                  <li>
                    <Link href="/" className="hover:text-[#B88E2F]">
                      Home
                    </Link>
                  </li>
                  <li>
                    <Link href="/Pages/Shop" className="hover:text-[#B88E2F]">
                      Shop
                    </Link>
                  </li>
                  <li>
                    <Link href="/Pages/Blog" className="hover:text-[#B88E2F]">
                      Blog
                    </Link>
                  </li>
                  <li>
                    <Link
                      href="/Pages/Contact"
                      className="hover:text-[#B88E2F]"
                    >
                      Contact
                    </Link>
                  </li>
                </ul>
              </div>
```

**Wishlist Component:** Allow users to save products for future reference.

```tsx
interface WhislistCardProps {
  onClose: () => void;
}

const WhistlistCard: React.FC<WhislistCardProps> = ({ onClose }) => {
  const { removeFromWishlist, wishlistItems } = useWishlist();
  const { addToCart } = useCart();

  const handleAddToCart = (item: any) => {
    addToCart(item);
    removeFromWishlist(item.id);
  };

  return (
    <div className="absolute top-0 md:right-6 right-4 h-auto bg-white shadow-lg md:w-[24rem] w-[18rem] p-4 rounded-sm
      <div className="flex justify-between items-center">
        <h3 className="text-xl font-bold mb-4">Wishlist</h3>
        <button onClick={onClose} className="mb-4">
          <IoHeartDislikeSharp size={20} className="text-[#888E2F]" />
        </button>
      </div>

      {wishlistItems?.length === 0 ? (
        <div className="text-center py-6">
          <p className="text-black opacity-70 font-semibold text-lg">
            Your Wishlist is empty.
          </p>
        </div>
      ) : (
        <div>
          {wishlistItems.map((item, index) => (
            <div
              key={index}
              className="flex justify-between items-center mb-4 border-t border-gray-200 pt-4"
            >
              <div className="flex items-center gap-4">
                <Image
                  src={item.image}
                  alt={item.title}
                  width={50}
                  height={50}
```

**Cart Component:** Display added items, quantity, remove button and total price .



```tsx
import Link from "next/link";
import { useCart } from "../CartContext/page";

interface ShoppingCartProps {
  onClose: () => void;
}

const ShoppingCart: React.FC<ShoppingCartProps> = ({ onClose }) => {
  const { cartItems, removeFromCart, totalPrice } = useCart();

  return (
    <div className="absolute top-0 md:right-6 right-4 h-auto bg-white shadow-lg md:w-[24rem] w-[18rem] p-4 rounded
      <div className="flex justify-between items-center">
        <h3 className="text-xl font-bold mb-4">Shopping Cart</h3>
        <button onClick={onClose} className="mb-4">
          <BsBagX size={20} className="text-[#B88E2F]" />
        </button>
      </div>

      {cartItems && cartItems.length === 0 ? (
        <div className="text-center py-6">
          <p className="text-black opacity-70 font-semibold text-lg">
            Your cart is empty.
          </p>
        </div>
      ) : (
        <div>
          {cartItems.map((item, index) => (
            <div
              key={index}
              className="flex justify-between items-center mb-4 border-t border-gray pt-4"
            >
```

## 3: TECHNICAL REPORT

**Steps Taken to Build and Integrate Components**:

1. **Component Creation**:
   - Designed reusable components such as, ProductCard, Navbar, Footer, .
   - Ensured modularity and separation of concerns for better maintainability.
2. **API Integration**:
   - Implemented dynamic data fetching from APIs using fetch or axios for product listing and details.
   - Set up server-side or client-side rendering based on requirements.
3. **Dynamic Routing**:
   - Configured dynamic routes to render individual product detail pages using product [id].
4. **Interactive Features**:
   - A filters bar to display products by price low to high and high to low.
   - A search bar for real-time product research using title.
   - A pagination to navigate through multiple pages of products.
5. **Enhancements**:
   - A related products and user profile components for a more comprehensive user experience.

**Challenges Faced and Solutions Implemented**:

1. **Challenge**: Handling Slow API Responses.
   - **Solution**: Added custom loading indicators to enhance user experience.
2. **Challenge**: Pagination Performance.
   - **Solution**: A server-side pagination to handle large datasets into multiple pages.

**Best Practices Followed During Development**:

1. **Reusable Components**:
   - Built reusable and scalable components like ProductCard, Navbar, and Footer for consistency and maintainability.
2. **Error Handling**:
   - Added comprehensive error handling for API calls to manage failures gracefully.
3. **Responsive Design**:
   - All components were mobile-first and fully responsive.
4. **Scalability**:
   - Followed a modular architecture to allow easy extension of features in the future.