# **Day 4 - Dynamic Frontend Components - Nike Shoe Marketplace**

## **Key Components Implemented**

### **1. Product Listing Component**

* Functionality:
  + Dynamically fetches product data from Sanity CMS.
  + Displays products in a responsive grid layout with placeholders for images.
  + Shows product name, price, category, and stock status.

Code Snippet:  
// components/ProductCard.js

import Link from 'next/link';

export default function ProductCard({ product }) {

return (

<div className="flex-1">

<div className="flex justify-between items-center mb-6">

<div className="flex items-center gap-4">

<Button

variant="ghost"

size="icon"

onClick={() => setViewType("grid")}

>

<Grid className={viewType === "grid" ? "text-black" : "text-gray-400"} />

</Button>

<Button

variant="ghost"

size="icon"

onClick={() => setViewType("list")}

>

<List className={viewType === "list" ? "text-black" : "text-gray-400"} />

</Button>

</div>

<Button variant="ghost" className="text-sm">

Sort By <ChevronDown className="ml-2 h-4 w-4" />

</Button>

</div>

{/\* Loading State \*/}

{loading ? (

<p>Loading products...</p>

) : (

<div

className={`grid gap-6 ${viewType === "grid" ? "grid-cols-1 sm:grid-cols-2 lg:grid-cols-4" : "grid-cols-1"}`}

>

{products.map((product) => (

<Link href={`/product/${product.id}`} key={product.id} className="group">

<div className="aspect-square bg-gray-100 rounded-lg overflow-hidden mb-4">

<Image

src={product.image}

alt={product.name}

width={300}

height={300}

className="w-full h-full object-cover group-hover:scale-105 transition-transform duration-300"

/>

</div>

<h3 className="font-medium">{product.name}</h3>

<p className="text-gray-600">{product.description}</p>

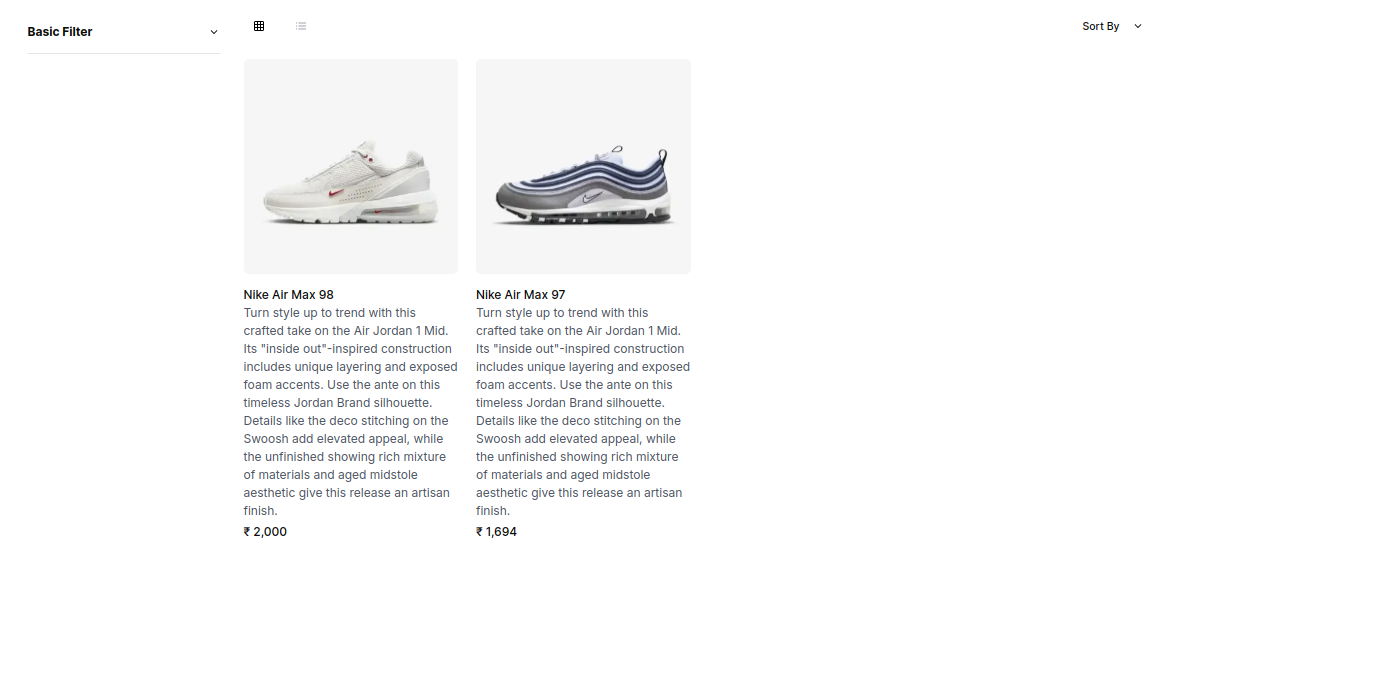
<p className="font-medium mt-1">₹ {product.price.toLocaleString()}</p>

</Link>

))}

</div>

)}</div>

**Visulization:  
  
**

### **2. Product Detail Component**

* Functionality:
  + Uses dynamic routing (/product/[id]) to display detailed product pages.
  + Shows size/color options, descriptions, and an "Add to Cart" button.

Code Snippet:  
// pages/product/[id].jsx

return {

id: product.\_id,

name: product.name,

price: product.price,

description: product.description,

image: product.image?.asset?.url || "",

};

} catch (error) {

console.error("Error fetching product:", error);

return null;

* }

};

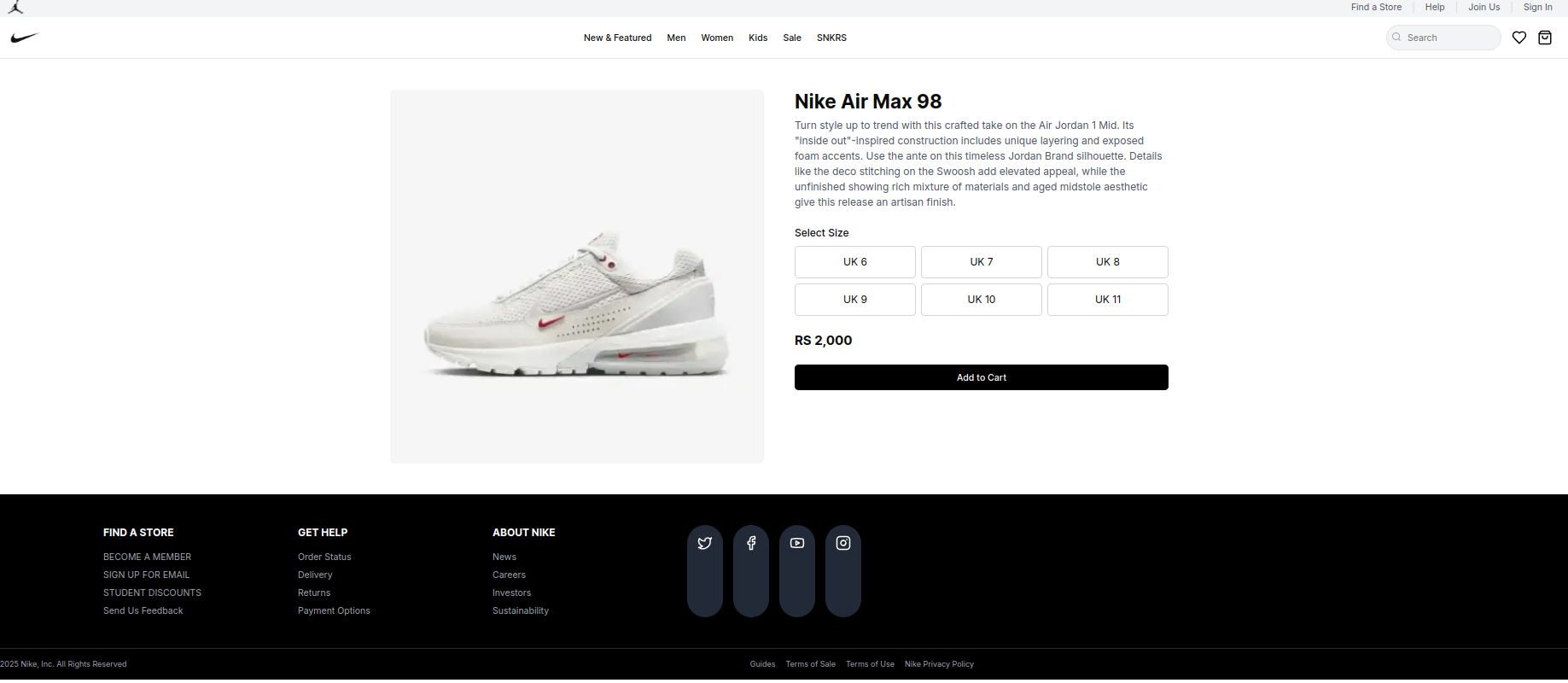
export async function generateMetadata({ params, }: { params: { id: string }; }): Promise<Metadata> { const product = await fetchProduct(params.id); return { title: product ? product.name : "Product Not Found", }; }

const ProductPage = async ({ params }: { params: { id: string } }) => { const product = await fetchProduct(params.id);

if (!product) { return <div>Product not found</div>; }

return <ProductDetail product={product} />; };

export default ProductPage;

Visulaisation:  
  


### 3. \*\*Cart Component\*\*

- \*\*Functionality:\*\*

- Manages cart items using React Context API.

- Persists cart data in local storage.

- \*\*Code Snippet:\*\*

```javascript

// context/CartContext.js

import { createContext, useState, useEffect } from 'react';

export const CartContext = createContext();

export function CartProvider({ children }) {

const [cartItems, setCartItems] = useState([]);

useEffect(() => {

const savedCart = localStorage.getItem('cart');

if (savedCart) setCartItems(JSON.parse(savedCart));

}, []);

const addToCart = (product) => {

const updatedCart = [...cartItems, product];

setCartItems(updatedCart);

localStorage.setItem('cart', JSON.stringify(updatedCart));

};

return (

<CartContext.Provider value={{ cartItems, addToCart }}>

{children}

</CartContext.Provider>

);

}

## **Best Practices Followed**

1. Modular Components:
   1. Reusable ProductCard, Dynamic Product page, and Cart components.
2. Responsive Design:
   1. Used Tailwind CSS for mobile-first layouts.
3. State Management:
   1. Global cart state with React Context API.
4. Error Handling:
   1. Fallback UI for empty product lists or API errors.