

Day 04: Building Dynamic Frontend Components for Marketplace

On Day 4, the development focus was on creating dynamic and reusable frontend components tailored for a robust e-commerce marketplace. Below is an organized breakdown of the key components built:

Key Components to Build

1. Product Listing Component

- **Description:** Displays all available products with relevant details such as image, name, price, and discounts.
-
- **Features:**
 - Pagination for efficient browsing.
 - Lazy loading to optimize performance.
 - Integration with backend data through Sanity API.
- **Outcome:** A responsive grid layout ensuring seamless product display.
-
- **Code snippet:**

```
• import React from 'react'
• import { useState, useEffect } from "react";
• import { client } from "@sanity/lib/client";
• import { BsFillFilterCircleFill } from "react-icons/bs";
• import Link from "next/link";
•
• type Product = {
•   _id: string;
•   productName: string;
•   imageUrl: string;
•   colors?: string[];
•   price: number;
```

```

•   };
•
•   export default function Products() {
•       const [isSidebarOpen, setIsSidebarOpen] = useState(false);
•       const [products, setProducts] = useState<Product[]>([]);
•       const [error, setError] = useState<string | null>(null);
•
•       useEffect(() => {
•           const fetchProducts = async () => {
•               const query = `
•                   *[_type == "product"]{
•                       _id,
•                       productName,
•                       category,
•                       price,
•                       inventory,
•                       colors,
•                       status,
•                       "imageUrl": image.asset->url,
•                       description
•                   }
•               `;
•               const fetchedProducts = await client.fetch(query);
•               setProducts(fetchedProducts);
•           };
•
•           fetchProducts();
•       }, []);
•
•       useEffect(() => {
•           const fetchProducts = async () => {
•               try {
•                   const query = ` *[_type == "product"]{
•                       _id,
•                       productName,
•                       category,
•                       price,
•                       inventory,
•                       colors,
•                       status,
•                       "imageUrl": image.asset->url,
•                       description
•                   }`;
•                   const fetchedProducts = await client.fetch(query);

```

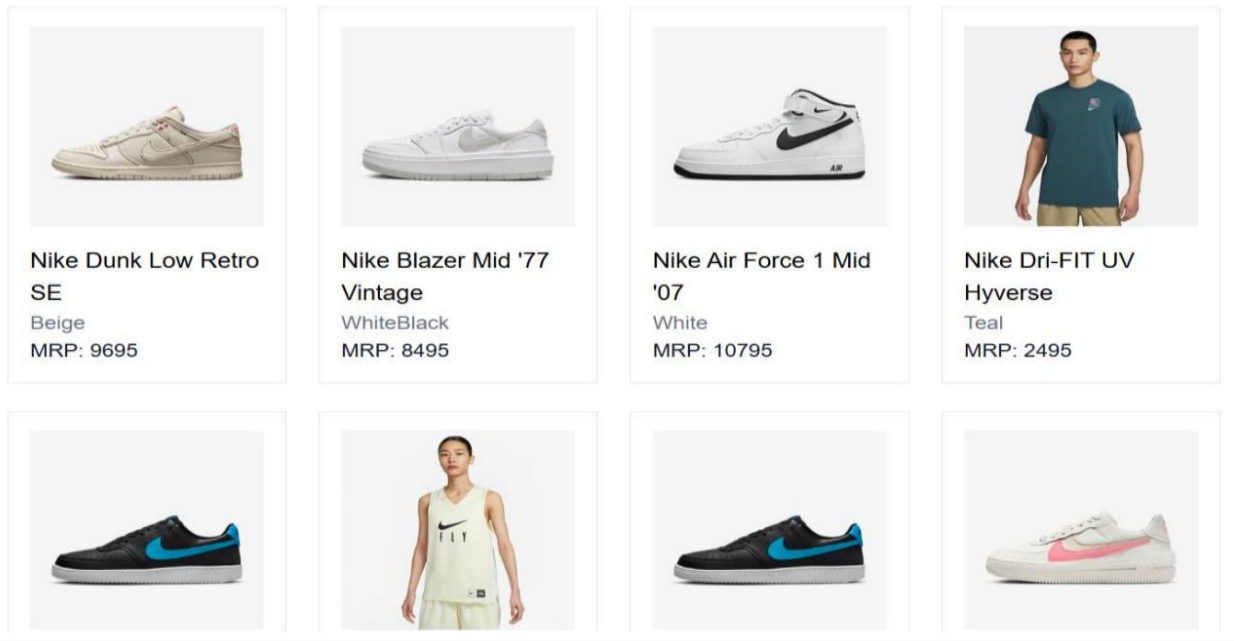
```

    •         setProducts(fetchedProducts); // Update state with fetched
products
    •     } catch (error) {
    •         console.error("Error fetching products:", error); // Log error
for debugging
    •         setError("Failed to fetch products. Please try again later.");
// Update error state
    •     }
    •     };
    •
    •         fetchProducts();
    •     }, []);
    •     const toggleSidebar = () => {
    •         setIsSidebarOpen(!isSidebarOpen);
    •     };
    •     </button>
    • })
    •     <h2 className="text-xl font-semibold">Sort By</h2>
    •     <button className="text-gray-600">Hide Filters</button>
    • </div>
    •
    •     <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3
lg:grid-cols-4 gap-6">
    •         {/* Map through the products array */}
    •         {products.map((product: Product) => (
    •             <Link href={` /Products/${product._id}`} key={product._id}>
    •                 <div className="border p-4">
    •                     <img
    •                         src={product.imageUrl}
    •                         alt={product.productName}
    •                         className="w-full mb-4"
    •                     />
    •                     <h3 className="text-lg font-medium">{product.productName}</h3>
    •                     <p className="text-gray-500">{product.colors}</p>
    •                     <p className="text-gray-900">MRP: {product.price}</p>
    •                 </div>
    •             </Link>
    •         )]}
    •     </div>
    • </main>
    • }
    • </>
    • )
    • }

```

Output snippet:

-



-

2. Product Detail Component

- **Description:** Provides a detailed view of an individual product.
- **Features:**
 - Dynamic data fetching for product-specific information.
 - Includes images, descriptions, reviews, and stock status.
 - "Add to Cart" functionality integrated.
- **Outcome:** Smooth navigation to product detail pages using dynamic routing.

Code snippet:

```
• "use client";
• import { ToastContainer, toast } from "react-toastify";
• import "react-toastify/dist/ReactToastify.css";
• import { useCart } from "@components/CartContext";
•
• interface Product {
•   _id: string;
•   productName: string;
```

```

•   imageUrl: string;
•   colors: string[];
•   price: number;
•   description: string;
• }
•
•
• export default function ProductDetailClient({ product }: { product:
Product }) {
•   const { addToCart } = useCart();
•
•   const handleAddToCart = () => {
•     addToCart({
•       id: product._id,
•       name: product.productName,
•       price: product.price,
•       imageUrl: product.imageUrl,
•       quantity: 1,
•     });
•     toast.success(`${product.productName} added to your bag!`, {
•       position: "top-right",
•       autoClose: 3000,
•       hideProgressBar: false,
•       closeOnClick: true,
•       pauseOnHover: true,
•       draggable: true,
•       progress: undefined,
•     });
•   };
•
•   return (
•     <div className="p-6">
•       <div className="flex flex-col lg:flex-row gap-8">
•         <div className="w-full lg:w-1/2">
•           <img
•             src={product.imageUrl}
•             alt={product.productName}
•             className="w-full h-96 rounded-lg shadow-md"
•           />
•         </div>
•         <div className="w-full lg:w-1/2">
•           <h1 className="text-2xl font-bold mb-
4">{product.productName}</h1>
•           <p className="text-gray-700 mb-4">{product.description}</p>
•           <p className="text-gray-900 text-lg font-semibold mb-4">
Price: ₹ {product.price}

```

```

•       </p>
•       {product.colors && (
•         <div className="mb-4">
•           <h3 className="font-medium mb-2">Available Colors:</h3>
•           <div className="flex gap-2">
•             {product.colors.map((color, index) => (
•               <span
•                 key={index}
•                 className="block w-6 h-6 rounded-full border border-
gray-300"
•                 style={{ backgroundColor: color }}
•               ></span>
•             ))}
•         </div>
•         >
•         Add to Cart
•       </button>
•     </div>
•   </div>
•   { /* Toastify container */}
•   <ToastContainer />
• </div>
• );
• }

```

Output snippet:



Nike Standard Issue Basketball Jersey

The Nike Standard Issue Basketball Jersey delivers a lightweight, breathable fit for top performance. Designed for athletes, it combines style and functionality to keep you cool and comfortable on and off the court.

Price: ₹ 2895

Available Colors:



Add to Cart

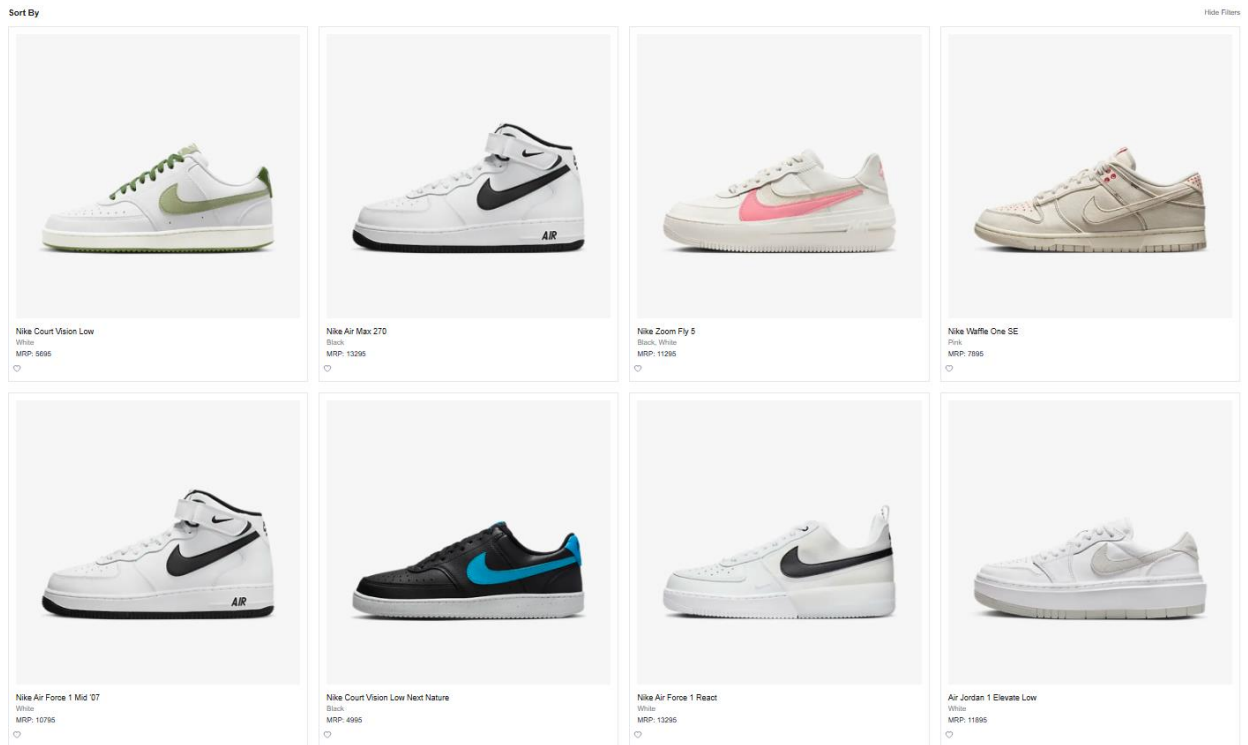
By: Asma Yaseen

3. Category Component

- **Description:** Enables browsing products by specific categories.
- **Features:**
 - Dynamic fetching of categories from the backend.
 - Display of associated products under each category.
 - Filter options within categories.
- **Outcome:** Organized category navigation improving the user experience.
- **Code snippet:**

```
• "use client";
• import { usePathname } from 'next/navigation';
• import Filter from "@components/Filter";
• import Mens from "@components/Categories/Mens";
• import Womens from "@components/Categories/Womens";
• import Products from '@components/Products';
• import SNKRS from '@components/Categories/SNKRS';
• import Sales from '@components/Categories/Sales';
• import Kids from '@components/Categories/Kid';
•
• export default function Categories() {
•   const pathname = usePathname(); // Get the current URL path
•
•   const renderContent = () => {
•     switch (pathname) {
•       case "/mens":
•         return <Mens />;
•       case "/womens":
•         return <Womens />;
•       case "/kids":
•         return <Kids />;
•       case "/sale":
•         return <Sales />;
•       case "/SNKRS":
•         return <SNKRS />;
•       default:
•         { /* Sidebar - Filters */}
•         <Filter />
•
•         { /* Main Content */}
•         {renderContent()}
•
•       </div>
•
•     }
•   }
• }
```

- **Output snippet:**



4. Search Bar

- **Description:** Helps users find specific products efficiently.
- **Features:**
 - Auto-complete dropdown for matching results.
 - Error handling for invalid or irrelevant search terms.
 - Optimized to handle large datasets quickly.
- **Outcome:** Real-time search capability enhancing usability.
- **Code snippet:**

```
// components/SearchBar.tsx
import { useState } from 'react';
import { client } from '../lib/sanityImage.mjs'; // Sanity client import

interface SearchBarProps {
  setQuery: React.Dispatch<React.SetStateAction<string>>;
}
//
```



```

const SearchBar: React.FC<SearchBarProps> = ({ setQuery }) => {
  const [searchTerm, setSearchTerm] = useState('');
  const [results, setResults] = useState<any[]>([]); // To store search results

  // Function to fetch search results from Sanity
  const handleSearch = async () => {
    const query = `*[_type == "product" && name match $searchTerm]{
      name,
      price,
      description,
      image
    }`;

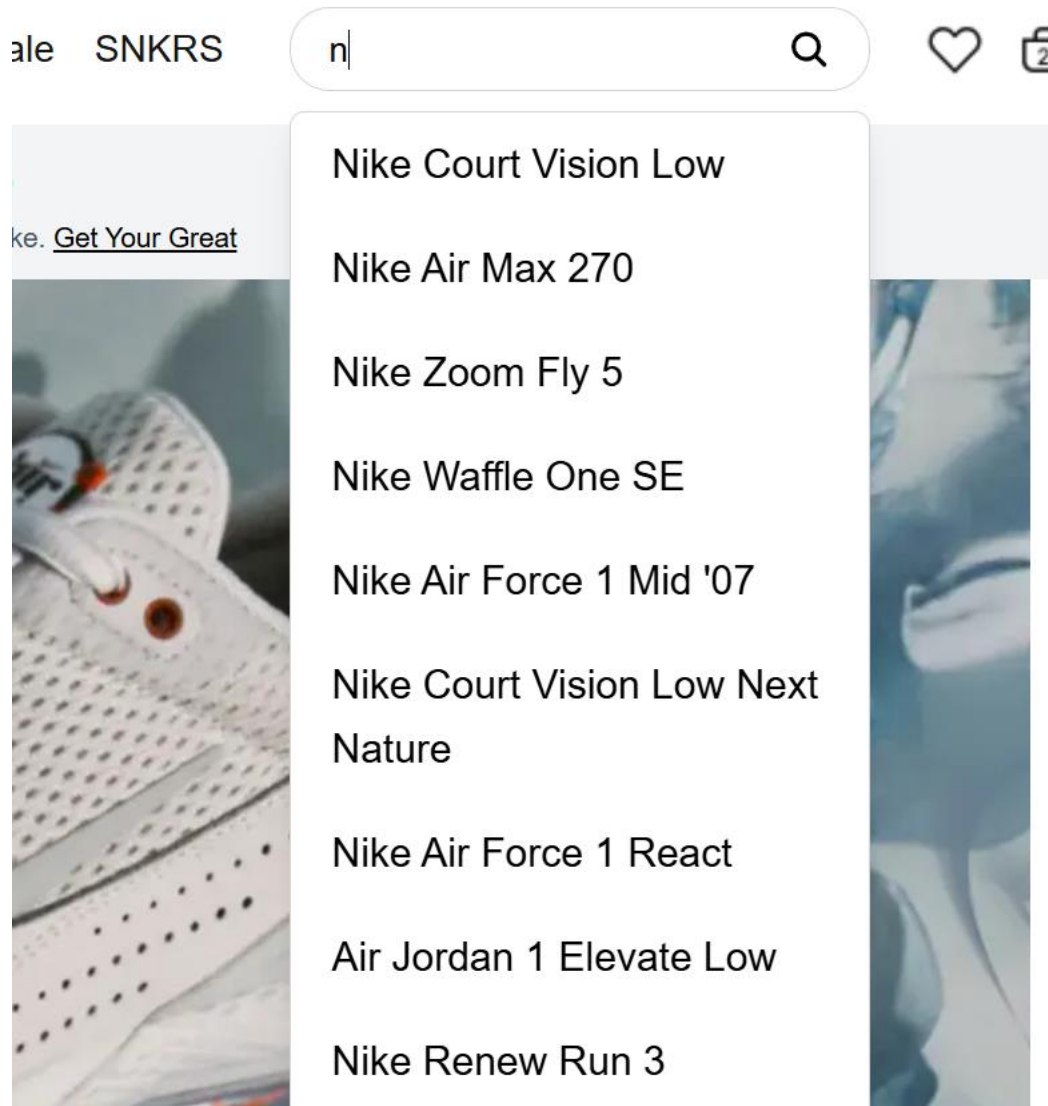
    try {
      const res = await client.fetch(query, { searchTerm: `*${searchTerm}*` });
      setResults(res);
      console.error('Error fetching products:', error);
    }
  };

  return (
    <div className="flex items-center justify-center my-6">
      <input
        type="text"
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
      />
      <div>
        results.map((product) => (
          <div key={product._id} className="border-b py-2">
            <h3>{product.name}</h3>
            <p>{product.price}</p>
            <p>{product.description}</p>
          </div>
        ))
      </div>
      <div>
        <p>No results found.</p>
      </div>
    </div>
  );
};

export default SearchBar;

```

Output snippet:



5. Cart Component

- **Description:** Manages selected items and provides an overview of the cart.
- **Features:**
 - Dynamic item addition/removal.
 - Real-time price calculation, including discounts and taxes.
 -

By: Asma Yaseen

- Persistent cart data for user convenience.
- **Outcome:** A user-friendly cart system ensuring smooth checkout preparation.
- **Code snippet:**

```

• "use client"
• import Recommendation from "@components/Recommendation";
• import Link from "next/link";
• import { useCart } from "@components/CartContext";
• import Image from "next/image";
•
• export default function Bag() {
•   const { cart, removeFromCart, updateCartItemQuantity } = useCart();
•   const subtotal = cart.reduce((total, item) => total + item.price *
    item.quantity, 0);
•
•   return (
•     <div className="min-h-screen bg-[#FFFFFF]">
•       {/* Container */}
•       <div className="max-w-7xl mx-auto px-4 py-8 grid grid-cols-1
lg:grid-cols-3 gap-8">
•         {/* Bag Section */}
•         <div className="lg:col-span-2 bg-[#FFFFFF] p-6">
•           {/* Free Delivery Banner */}
•           <div className="bg-gray-100 p-4 mb-6 rounded-md text-sm">
•             <span>Free Delivery</span> applies to orders of ₹10,000 or
more.{ " "}
•             <a href="#" className="text-[#111111] underline">
•               View details
•             </a>
•           </div>
•
•           {/* Bag Items */}
•           <h2 className="text-lg font-bold mb-2">Your Cart</h2>
•           <div className="space-y-8">
•             {cart.length === 0 ? (
•               <div className="flex w-full items-center justify-center">
•                 <Image
•                   src={"/assets/addtocart.jpg"}
•                   width={300}
•                   height={300}
•                   alt={"Your Cart Is Empty"}
•                   className="mb-4"
•                 />
•               </div>
•             ) : (

```

```

•       <div className="space-y-8">
•         {cart.map((item) => (
•           <div key={item.id} className="flex gap-4">
•             <img
•               src={item.imageUrl}
•               alt={item.name}
•               className="w-24 h-24 object-cover rounded-md border"
•             />
•             <div className="flex-1">
•               <h3 className="font-semibold">{item.name}</h3>
•               <p>Price: ₹{item.price}</p>
•               <div className="flex items-center space-x-4">
•                 <button
•                   className="px-2 py-1 bg-gray-200 rounded-full
text-black font-semibold"
•                   onClick={() => updateCartItemQuantity(item.id,
item.quantity - 1)}
•                   disabled={item.quantity <= 1}
•                 >
•                   -
•                 </button>
•                 <p>Quantity: {item.quantity}</p>
•                 <button
•                   className="px-2 py-1 bg-gray-200 rounded-full
text-black font-semibold"
•                   onClick={() => updateCartItemQuantity(item.id,
item.quantity + 1)}
•                 >
•                   +
•                 </button>
•               </div>
•               <button
•                 className="text-red-500 mt-2"
•                 onClick={() => removeFromCart(item.id)}
•               >
•                 Remove
•               </button>
•             </div>
•           </div>
•         )})}
•       </div>
•     </div>
•

```

```


•      { /* Favourites Section */ }
•      <h3 className="text-lg font-bold mt-12">Favourites</h3>
•      <p className="text-gray-500">There are no items saved to your
favourites.</p>
•      </div>
•
•
•      </button>
•      </Link>
•      </div>
•      </div>
•
•      { /* Recommendations */ }
•      <Recommendation/>
•    </div>
•  );
• }

```

• Output snippet:

Free Delivery applies to orders of ₹10,000 or more. [View details](#)

Your Cart




Nike Court Vision Low

Price: ₹5695

-
Quantity: 1
+

[Remove](#)



Nike Zoom Fly 5

Price: ₹11295

-
Quantity: 1
+

[Remove](#)

Summary

Subtotal:	Rs 16,990
Estimated Delivery & Handling:	Free
Total:	Rs 16,990

[Member Checkout](#)

6. Checkout Flow Component

- **Description:** Handles the entire checkout process, ensuring user satisfaction.
- **Features:**
 - Multi-step form for shipping, payment, and confirmation.
 - Validation checks for form inputs.
 -

By: Asma Yaseen

- Integration with APIs for order placement.
- **Outcome:** A streamlined checkout process improving conversion rates.

Code snippet:

```
"use client";
import React, { useState } from "react";
import { useCart } from "@components/CartContext";
import { useRouter } from "next/navigation";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import emailjs from "emailjs-com";

export default function CheckOut() {
  const { cart } = useCart();
  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    phone: "",
    pan: "",
    address: "",
  });

  const [orderId, setOrderId] = useState<string | null>(null);
  const [showDialog, setShowDialog] = useState(false);
  const router = useRouter();

  const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

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

    // Check if all form fields are filled
    if (
      !formData.firstName ||
      !formData.lastName ||
      !formData.email ||
      !formData.phone ||
      !formData.pan ||

```

```

    !formData.address
  ) {
    toast.error("Please fill in all the required fields.");
    return;
  }

  // Check if cart is empty
  if (cart.length === 0) {
    toast.error("Your cart is empty. Please add items to your cart.");
    return;
  }

  // Generate a random order ID
  const generatedOrderId = `ORD-${Math.floor(Math.random() * 1000000)}`;

  // Log emailContent for debugging
  console.log("Email Content:", emailContent);

  // Send email using EmailJS
  emailjs
    .send(
      process.env.NEXT_PUBLIC_YOUR_SERVICE_ID!,
      process.env.NEXT_PUBLIC_YOUR_TEMPLATE_ID!,
      emailContent,
      process.env.NEXT_PUBLIC_YOUR_USER_ID!
    )
    .then((response) => {
      console.log("Email sent successfully:", response);
      toast.success("Order confirmation sent to your email!");
    })
    .catch((error) => {
      console.error("Email send error:", error);
      toast.error("Failed to send confirmation email.");
    });
};

const handleTrackOrder = () => {
  // Redirect to order tracking page with the generated order ID
  router.push(`/Order-Tracking/${orderId}`);
};

return (
  <div className="min-h-screen bg-[#FFFFFF] p-6">
    <

```

```

        className="p-4 border rounded-md w-full"
      />
    </div>
    <div className="grid grid-cols-1 gap-4">
      <input
        name="address"
        value={formData.address}
        onChange={handleInputChange}
        className="p-4 border rounded-md w-full"
      />

      {/* PAN */}
      <div>
        <h2 className="text-lg font-medium mb-4">
          What's your PAN?
        </h2>
        <input
          type="text"
          placeholder="PAN"
          name="pan"
          value={formData.pan}
          onChange={handleInputChange}
          className="p-4 border rounded-md w-full"
        />
      </div>

      {/* Continue Button */}
      <button className="w-full p-4 bg-[#111111] text-white rounded-3xl">
        Continue
      </button>
    </form>
  </div>

  {/* Right Section: Order Summary */}
  <div className="bg-white w-full lg:w-[320px] h-auto lg:h-[721px] p-6">
    <h2 className="text-xl font-bold mb-6">Order Summary</h2>
    <ul className="space-y-6">
      {cart.map((item) => (
        <li key={item.id} className="flex items-center">
          <img
            src={item.imageUrl}
            alt={item.name}
            className="w-[128px] h-[128px] rounded-md border mr-4"

```



```

        />
        <div>
            <p className="font-normal text-[13px]">{item.name}</p>
            <p className="text-gray-500">Qty: {item.quantity}</p>

        {/* Dialog Box */}
        {showDialog && orderId && (
            <div className="fixed inset-0 flex items-center justify-center bg-black
bg-opacity-50">
                <div className="bg-white p-6 rounded-md w-[400px]">
                    <h2 className="text-lg font-medium mb-4">
                        Your Order ID: {orderId}
                    </h2>
                    <p className="mb-4">
                        We have received your order! A confirmation email has been sent
to{" "}
                        {formData.email}.
                    </p>
                    <button
                        onClick={handleTrackOrder}
                        className="w-full p-4 bg-[#111111] text-white rounded-3xl"
                    >
                        Track Order
                    </button>
                </div>
            </div>
        )}

        {/* Toast Container */}
        <ToastContainer />
    </div>
    );
}

```

Output snippet:

By: Asma Yaseen

How would you like to get your order?

Customs regulation for India require a copy of the recipient's KYC. The address on the KYC needs to match the shipping address. Our courier will contact you via SMS/email to obtain a copy of your KYC. The KYC will be stored securely and used solely for the purpose of clearing customs (including sharing it with customs officials) for all orders and returns. If your KYC does not match your shipping address, please click the link for more information. [Learn More](#)

 Deliver it

Enter your name and address:

First Name

Last Name

Address Line 1

Address Line 2

What's your contact information?

Email

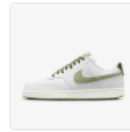
Phone Number

What's your PAN?

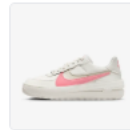
PAN

Continue

Order Summary



Nike Court Vision Low
Qty: 1



Nike Zoom Fly 5
Qty: 1

Subtotal: ₹16990.00
Delivery: Free
Total: ₹16990.00

7. Filter Panel Component

- **Description:** Enables users to refine product searches.
- **Features:**
 - Filters for price range, ratings, and availability.
 - Dynamic updates without page reloads.
 - Integration with search and category components.
- **Outcome:** Precise and quick product filtering.

By: Asma Yaseen

• Code snippet:

```
• "use client";
• import { useState } from "react";
•
• export default function Filter(){
•   const [isSidebarOpen, setIsSidebarOpen] = useState(false);
•   const toggleSidebar = () => {
•     setIsSidebarOpen(!isSidebarOpen);
•   };
•   return(
•     <aside
•       className={`fixed top-0 left-0 h-full bg-[#FFFFFF] p-4 pl-12 z-40
transition-transform duration-300 ease-in-out transform ${
•         isSidebarOpen ? "translate-x-0" : "-translate-x-full"
•       } lg:relative lg:translate-x-0 lg:w-1/4`}
•     >
•       <button
•         className="lg:hidden absolute top-4 right-4 text-gray-600 text-
xl"
•         onClick={toggleSidebar}
•       >
•         ✕
•       </button>
•       <h2 className="text-lg font-semibold mb-4">New (500)</h2>
•       <div className="mb-6">
•         <h3 className="font-semibold mb-2">Shop</h3>
•         <ul className="space-y-2">
•           <li><a href="#" className="text-gray-700">Shoes</a></li>
•           <li><a href="#" className="text-gray-700">Sports
Bras</a></li>
•           <li><a href="#" className="text-gray-700">Tops & T-
Shirts</a></li>
•           { /* Add more categories here */ }
•         </ul>
•       </div>
•       <div className="mb-6">
•         <h3 className="font-semibold mb-2">Gender</h3>
•         <ul className="space-y-2">
•           <li><label><input type="checkbox" className="mr-2 accent-
black" />Men</label></li>
•           <li><label><input type="checkbox" className="mr-2 accent-
black" />Women</label></li>
•           <li><label><input type="checkbox" className="mr-2 accent-
black" />Unisex</label></li>
```

```

•         </ul>
•     </div>
•     <div className="mb-6">
•         <h3 className="font-semibold mb-2">Kids</h3>
•         <ul className="space-y-2">
•             <li><label><input type="checkbox" className="mr-2 accent-
black" />Boys</label></li>
•             <li><label><input type="checkbox" className="mr-2 accent-
black" />Girls</label></li>
•         </ul>
•     </div>
•     <div>
•         <h3 className="font-semibold mb-2">Shop By Price</h3>
•         <ul className="space-y-2">
•             <li><label><input type="radio" name="price" className="mr-2
accent-black" />Under ₹ 7,500</label></li>
•             <li><label><input type="radio" name="price" className="mr-2
accent-black" />₹ 7,500 - ₹ 15,000</label></li>
•         </ul>
•     </div>
• </aside>
• )
• }

```

• **Output snippet:**

• By: Asma Yaseen

New (500)

Shop

Shoes

Sports Bras

Tops & T-Shirts

Gender

☐ Men

☐ Women

☐ Unisex

Kids

☐ Boys

☐ Girls

Shop By Price

☐ Under ₹ 7,500

☐ ₹ 7,500 - ₹ 15,000

•

8. Header and Footer Components

- **Description:** Consistent elements providing site navigation and essential links.
- **Features:**
 - Sticky header with navigation options and a mini-cart.
 - Footer with links to FAQs, contact, social media, and legal information.
- **Outcome:** Enhanced navigation and accessibility.
- **Output snippet:**



•

•

By: Asma Yaseen

FIND A STORE	GET HELP	ABOUT NIKE
Become a Member	Order Status	News
Sign Up for Email	Delivery	Careers
Send Us Feedback	Returns	Investors
Student Discounts	Payment Options	Sustainability

9. FAQ and Help Center Component

- **Description:** Centralized location for user queries and support.
- **Features:**
 - Accordion layout for question categorization.
 - Searchable FAQs.
 - Integration with contact support for unresolved issues.
- **Outcome:** Reduced customer inquiries via direct answers.

10. Social Media Sharing Component

- **Description:** Allows users to share products or pages on social platforms.
- **Features:**
 - Integration with platforms like Facebook, Twitter, and Instagram.
 - Custom share messages and preview images.
- **Outcome:** Increased visibility and user engagement.
- **Code snippet:**

```

import React from "react";
import { FaFacebook, FaInstagram, FaTwitter } from "react-icons/fa";
export default function Social_Media() {
  return (
    <div className="w-full mt-6">
      <div className="border-t border-gray-700 pt-6 flex justify-between items-center text-sm text-gray-400">
        <p>© 2023 Nike, Inc. All Rights Reserved</p>

```

```

•      <a href="#" className="hover:text-white">
•      <FaTwitter className="h-6 w-6" />
•      </a>
•      <a href="#" className="hover:text-white">
•      <FaFacebook className="h-6 w-6" />
•      </a>
•      <a href="#" className="hover:text-white">
•      <FaInstagram className="h-6 w-6" />
•      </a>
•    </div>
•  </div>
• </div>
• );
• }
•

```

Output snippet:

© 2023 Nike, Inc. All Rights Reserved



API Integration

• Sanity API:

- Setup for efficient data fetching from the Sanity backend.
- Client configuration placed in the `lib` folder for global usage.

```

NEXT_PUBLIC_SANITY_PROJECT_ID="..."
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="..."
NEXT_PUBLIC_SANITY_API_VERSION=2025-01-18

```

Dynamic Routing

• Implementation:

- Routes like `/Products/:id` dynamically handle individual product pages.
- Simplifies navigation while supporting SEO best practices.

By: Asma Yaseen

```
src > app > Products > [id] > page.tsx > ProductDetail > [e] query
1 import { client } from '@sanity/lib/client';
2
3 type Product = {
4   _id: string;
5   productName: string;
6   imageUrl: string;
7   colors?: string[];
8   price: number;
9   description: string;
10 };
11
12 export default async function ProductDetail({ params }: { id:
13   string }) {
14   const query = `
15     *[_type == "product" && _id == "${params.id}"][0] {
16       _id,
17       productName,
18       "imageUrl": image.asset->url,
19       colors,
20       price,
21       description
22     }
23   `;
24   const product: Product = await client.fetch(query);
25
26   if (!product) {
27     return <div>Product not found</div>;
28   }
29
30   return (
31     <div className="p-6">
32       <div className="flex flex-col lg:flex-row gap-8">
33         <div className="w-full lg:w-1/2">
34           <img src={product.imageUrl} alt={product.productName}
```

```
src > app > Products > [id] > page.tsx > ...
12 export default async function ProductDetail({ params }: { id:
13   string }) {
14   const query = `
15     *[_type == "product" && _id == "${params.id}"][0] {
16       _id,
17       productName,
18       "imageUrl": image.asset->url,
19       colors,
20       price,
21       description
22     }
23   `;
24   const product: Product = await client.fetch(query);
25
26   if (!product) {
27     return <div>Product not found</div>;
28   }
29
30   return (
31     <div className="p-6">
32       <div className="flex flex-col lg:flex-row gap-8">
33         <div className="w-full lg:w-1/2">
34           <img src={product.imageUrl} alt={product.productName}
35             className="w-full h-96 rounded-lg shadow-md" />
36         </div>
37         <div className="w-full lg:w-1/2">
38           <h1 className="text-2xl font-bold mb-4">{product.productName}</h1>
39           <p className="text-gray-700 mb-4">{product.description}</p>
40           <p className="text-gray-900 text-lg font-semibold mb-4">Price: ₹
41             {product.price}</p>
42           <div className="mb-4">
43             <h3 className="font-medium mb-2">Available Colors:</h3>
44             <div className="flex gap-2">
45               {product.colors.map((color, index) => (
46                 <span
47                   key={index}
48                   className="block w-6 h-6 rounded-full border
49                     border-gray-300
50                     style={{ backgroundColor: color }}
51                 ></span>
52               ))}
53             </div>
54           </div>
55           <button className="bg-black text-white py-2 px-4 rounded mt-6
56             hover:bg-gray-800">
57             Add to Cart
58           </button>
59         </div>
60       </div>
61     </div>
62   );
63 }
```

-
-
-
-

Get Product Data Dynamically:



-

Nike Dri-FIT UV Hyverse

The Nike Dri-FIT UV Hyverse graphic top offers comfort and UV protection in style. Made from soft, breathable fabric, this fitness top keeps you cool and dry during workouts or casual outings.

Price: ₹ 2495

Available Colors:



Add to Cart

Wishlist Component

- **Description:** Allows users to save and manage their favorite products for later viewing or purchasing, enhancing user engagement and retention.
- **Features:**
 - Add/remove items to/from the wishlist with a single click.

By: Asma Yaseen

- o Display of product details like image, name, price, discount, and stock status.

- o Persistent wishlist using local storage or user account integration.

- o Option to move items from the wishlist to the cart.

- o Notification if a wishlist item goes on sale or is back in stock.

- **Outcome:** Increased user interaction and potential conversions by allowing users to revisit their preferred products easily.

Code snippet:

```
"use client"
import React, { createContext, useContext, useState, ReactNode } from "react";

type WishListItem = {
  id: string;
  name: string;
  price: number;
  imageUrl: string;
};

type WishListContextType = {
  wishList: WishListItem[];
  addToWishList: (item: WishListItem) => void;
  removeFromWishList: (id: string) => void;
  clearWishList: () => void;
};

const WishListContext = createContext<WishListContextType |
undefined>(undefined);

export const WishListProvider = ({ children }: { children: ReactNode }) => {
  const [wishList, setWishList] = useState<WishListItem[]>([]);

  const addToWishList = (item: WishListItem) => {
    setWishList((prevList) => {
      const existingItem = prevList.find((listItem) => listItem.id === item.id);
      if (existingItem) return prevList;
      return [...prevList, item];
    });
  };
};
```

```

const removeFromWishList = (id: string) => {
  setWishList((prevList) => prevList.filter((item) => item.id !== id));
};

const clearWishList = () => setWishList([]);

return (
  <WishListContext.Provider value={{ wishList, addToWishList,
removeFromWishList, clearWishList }}>
    {children}
  </WishListContext.Provider>
);
};

export const useWishList = () => {
  const context = useContext(WishListContext);
  if (!context) {
    throw new Error("useWishList must be used within a WishListProvider");
  }
  return context;
};

```

Output snippet:

My Wish List

Clear Wish List



Nike Court Vision Low
MRP: 5695

Remove



Nike Blazer Mid '77 Vintage
MRP: 8495

Remove



Air Jordan 1 Elevate Low
MRP: 11895

Remove

Order Tracking Component

- **Description:** Enables users to track the status of their orders post-purchase, fostering trust and transparency.

- **Features:**

- o Order status updates (e.g., Confirmed, Packed, Shipped, Out for Delivery, Delivered).
- o Display of estimated delivery date and real-time shipment tracking via API integration.
- o Order history with details like order ID, product list, total price, and delivery address.
- o Option to download invoices and report issues with an order.
- o Push/email notifications for significant status changes.

- **Outcome:** Enhanced post-purchase experience, improving customer satisfaction and loyalty.

Code snippet:

```
"use client";
import { useState, useEffect } from "react";
import { toast, ToastContainer } from "react-toastify"; // Import toastify
import "react-toastify/dist/ReactToastify.css"; // Import the required CSS

const OrderTracking = () => {
  const [orderIdInput, setOrderIdInput] = useState<string>(""); // State to
  store inputted order ID
  const [orderDetails, setOrderDetails] = useState<any | null>(null);
  const [isModalOpen, setIsModalOpen] = useState(false); // State to manage
  modal visibility
  const [isTracking, setIsTracking] = useState(false); // State to manage
  tracking action

  const handleOrderIdChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setOrderIdInput(e.target.value); // Update input value
  };

  const handleTrackOrder = () => {
    setIsTracking(true); // Start tracking order when the button is clicked
  };

  useEffect(() => {
    if (isTracking) {
      const savedOrderDetails = sessionStorage.getItem("orderDetails");
      if (savedOrderDetails) {
        const { formData, cart, orderId } = JSON.parse(savedOrderDetails);
        if (orderId === orderIdInput) {
          setOrderDetails({ formData, cart, orderId });
        }
      }
    }
  }, [isTracking, orderIdInput]);
};
```

```

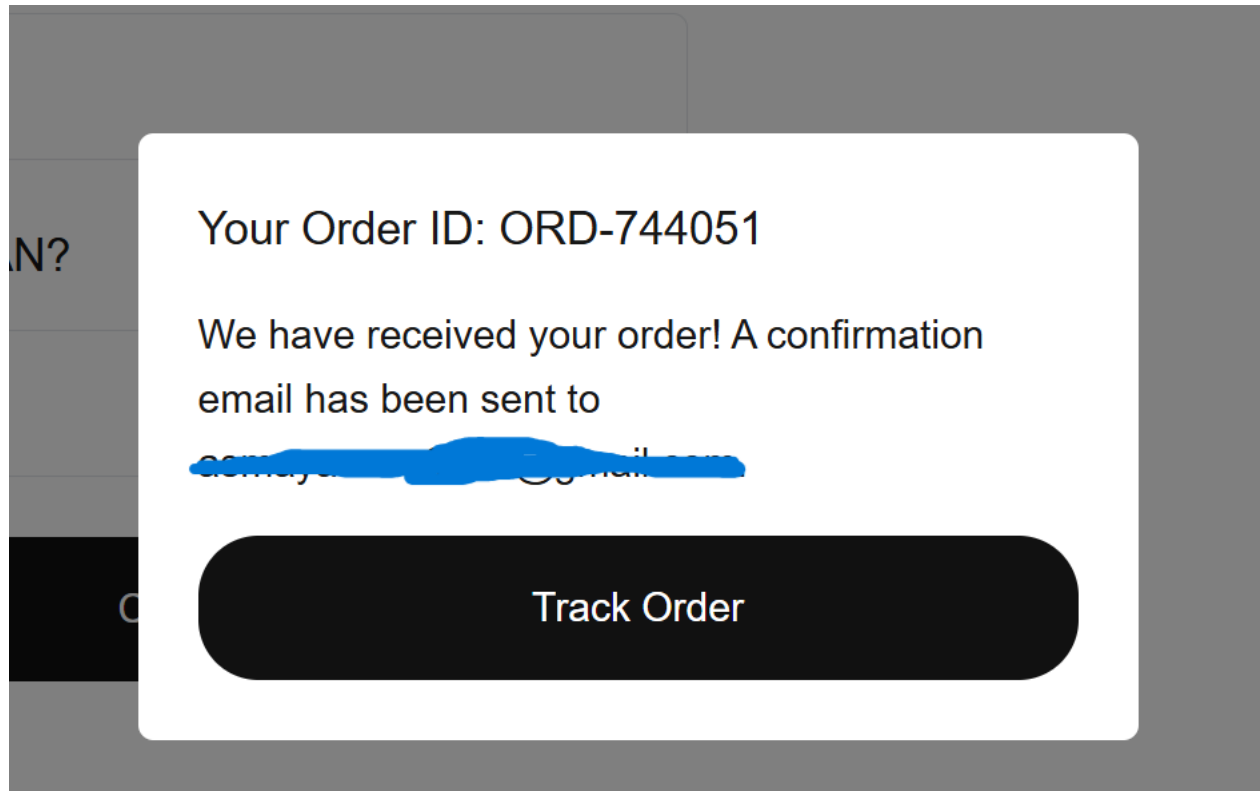
        setIsModalOpen(true); // Open the modal with order details
    } else {
        setOrderDetails(null); // Reset if no matching order
        toast.error("Order ID not found! Please check and try again."); // Sh
        <h3 className="font-bold text-xl mb-4">Order Details</h3>
        <p><strong>Name:</strong> {orderDetails?.formData.firstName}
{orderDetails?.formData.lastName}</p>
        <p><strong>Address:</strong> {orderDetails?.formData.address}</p>
        <ul className="mb-4">
            <strong>Order Items:</strong>
            {orderDetails?.cart.map((item: any, index: number) => (
                <li key={index}>
                    {item.name} (x{item.quantity}) - Rs {item.price *
item.quantity}
                </li>
            ))}
        </ul>
        <p>
            <strong>Total: Rs{orderDetails?.cart.reduce(
                (total: number, item: any) => total + item.price *
item.quantity,
                0
            )}</strong>
        </p>
        <div className="mt-4 text-center">
            <button
                onClick={closeModal}
                className="w-full p-4 bg-red-600 text-white rounded-3xl"
            >
                Close
            </button>
        </div>
    </div>
</div>
    )}
</div>

    { /* Toast Container */ }
    <ToastContainer />
</div>
);
};

export default OrderTracking;

```

Output snippet:



By: Asma Yaseen



Track Your Order

Enter your Order ID

Track Order

Final Output

The components built during this phase are reusable, scalable, and optimized for performance. These elements lay the foundation for an engaging and user-friendly marketplace platform.

Final CheckList:

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

By: Asma Yaseen