Day 4 - Dynamic Frontend Components - OutfitPlus

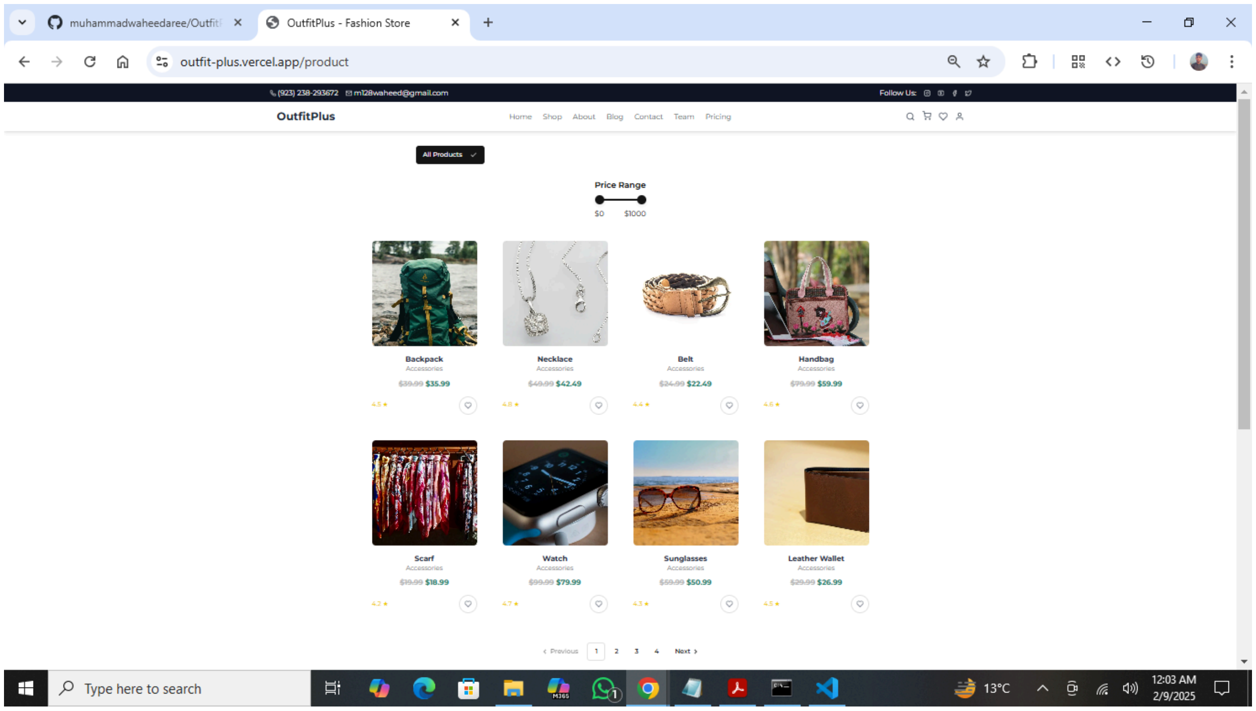
Functional Deliverables

Product Listing Page with Dynamic Data

The product listing page dynamically fetches and displays products from the backend. Key features include:

- **Dynamic product loading**: Products are fetched and displayed in a grid format.
- **Real-time updates**: Changes in product data reflect instantly.

Screenshot:


OutfitPlus - Fashion Store
outfit-plus.vercel.app/product
OutfitPlus
Home Shop About Blog Contact Team Pricing
All Products
Price Range
$0 $1000
Backpack Accessories $39.99 $35.99
Necklace Accessories $49.99 $42.49
Belt Accessories $24.99 $22.49
Handbag Accessories $79.99 $59.99
Scarf Accessories $19.99 $18.99
Watch Accessories $99.99 $79.99
Sunglasses Accessories $59.99 $50.99
Leather Wallet Accessories $29.99 $26.99
Previous 1 2 3 4 Next
Type here to search

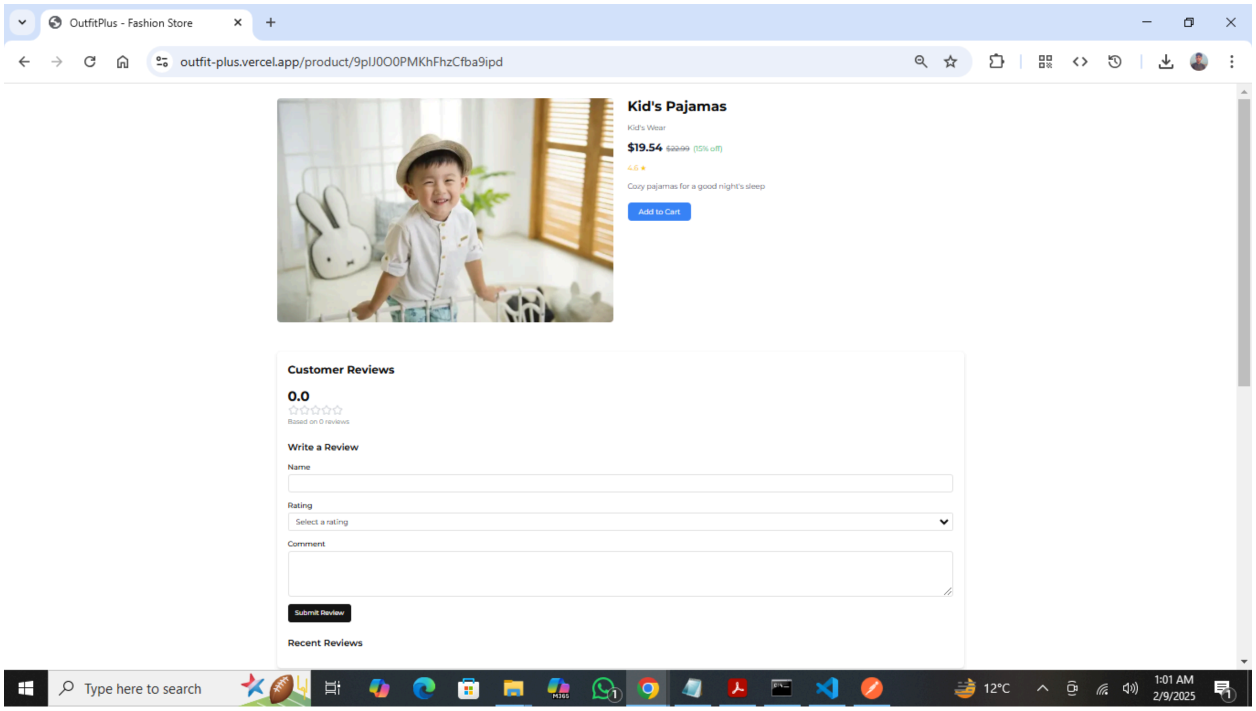
Individual Product Detail Pages with Accurate Routing & Data Rendering

Each product has its own dedicated page with detailed information, dynamically fetched from the database. Features include:

- **Accurate URL routing**: Product details appear based on the correct product ID.

- **Detailed product information**: Name, category, price, discount, and reviews are displayed.
- **Wishlist functionality**: Users can add products to their wishlist.

Screenshot:



OutfitPlus - Fashion Store
outfit-plus.vercel.app/product/9plJ0O0PMKhFhzCfba9ipd

Kid's Pajamas

Kid's Wear

$19.54 ~~$22.99~~ (15% off)

4.6 ★

Cozy pajamas for a good night's sleep

Customer Reviews

0.0

☆☆☆☆☆

Based on 0 reviews

Write a Review

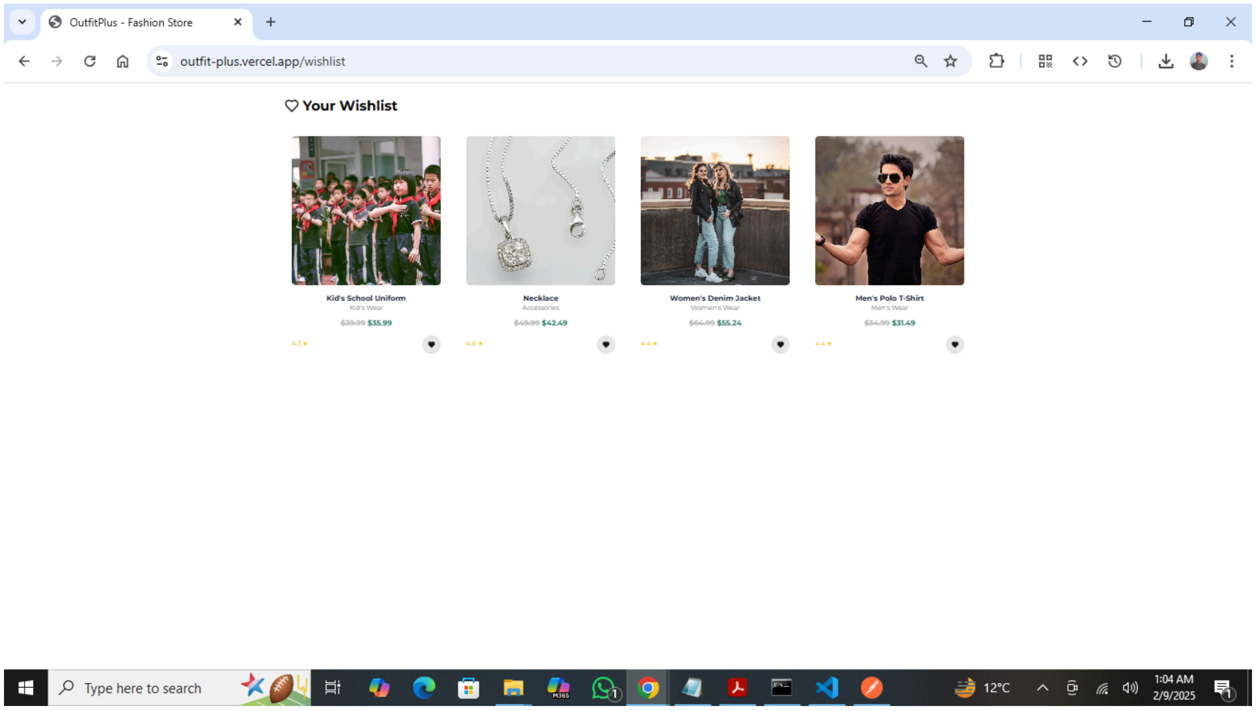
Name:

Rating: Select a rating

Comment:

Recent Reviews

Type here to search
12°C
1:01 AM
2/9/2025



♡ Your Wishlist

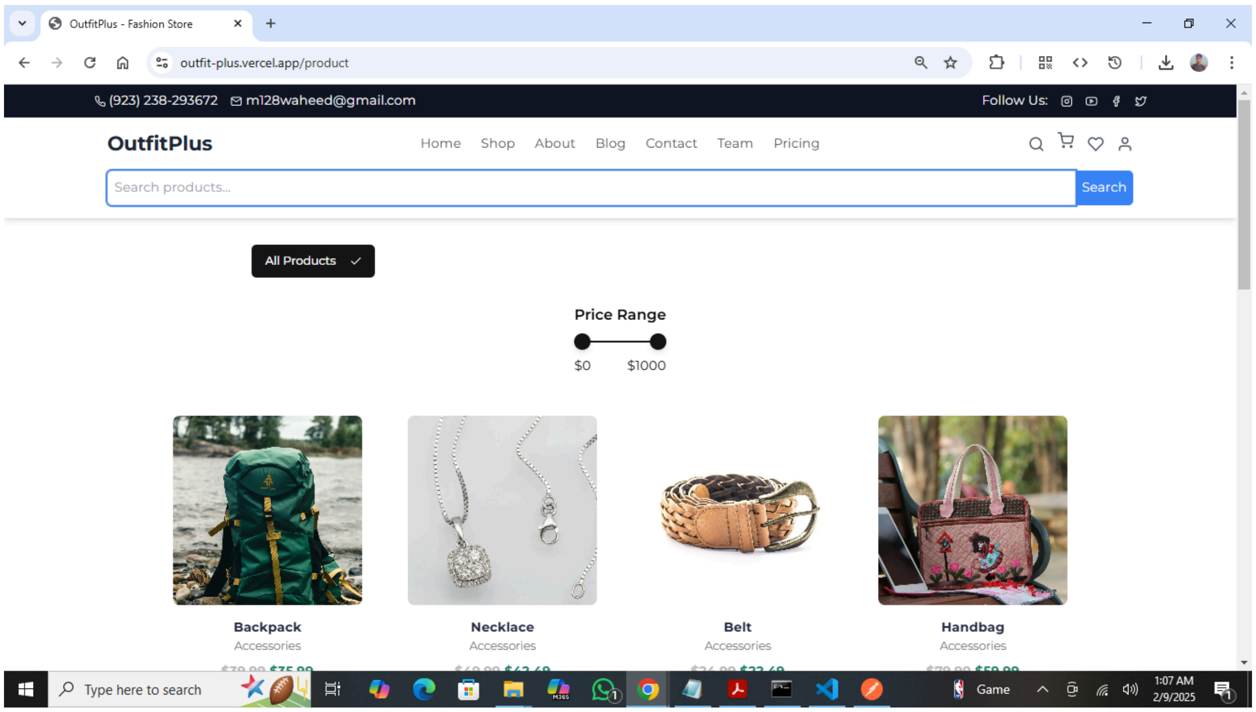
Kid's School Uniform	Necklace	Women's Denim Jacket	Men's Polo T-Shirt
Kid's Wear	Accessories	Women's Wear	Men's Wear
~~$39.99~~ $35.99	~~$49.99~~ $42.49	~~$64.99~~ $55.24	~~$34.99~~ $31.49
4.3 ★	4.8 ★	4.4 ★	4.4 ★

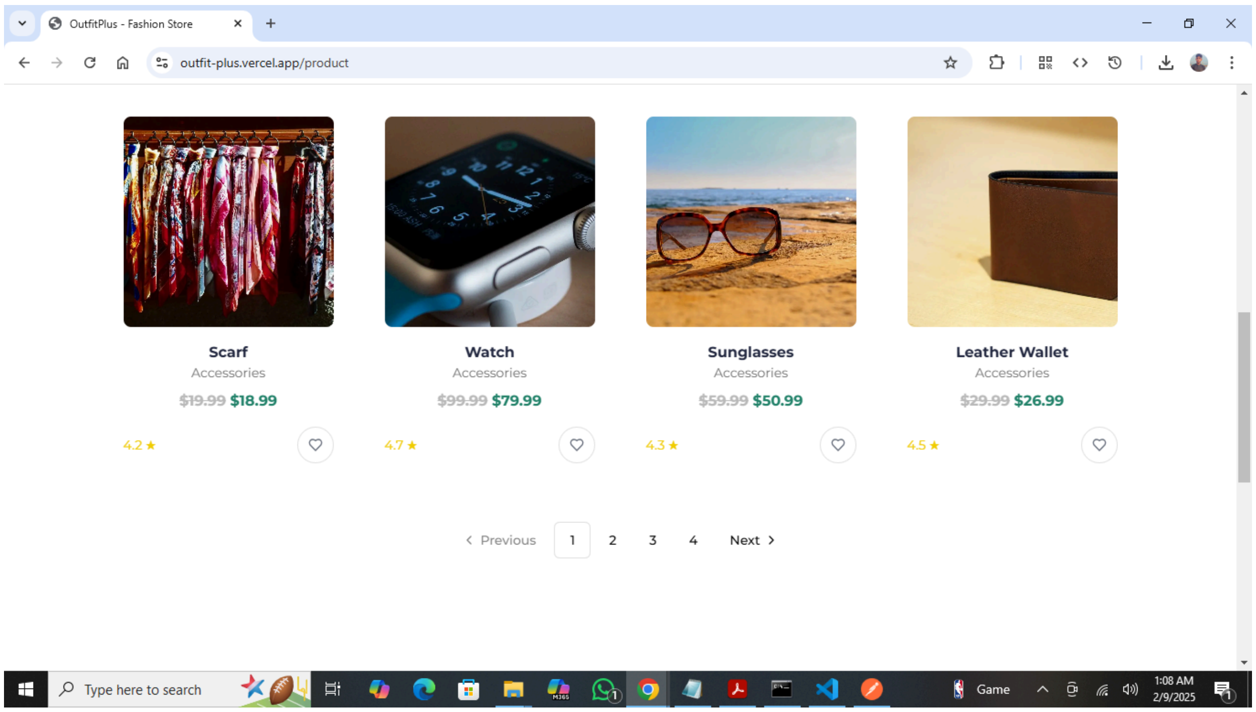
Working Search Bar, and Pagination

The product listing page includes filtering and search capabilities to enhance user experience.

- **Search bar**: A search feature allows users to find products by keywords.
- **Pagination**: Users can navigate between pages for better browsing.

Screenshot:

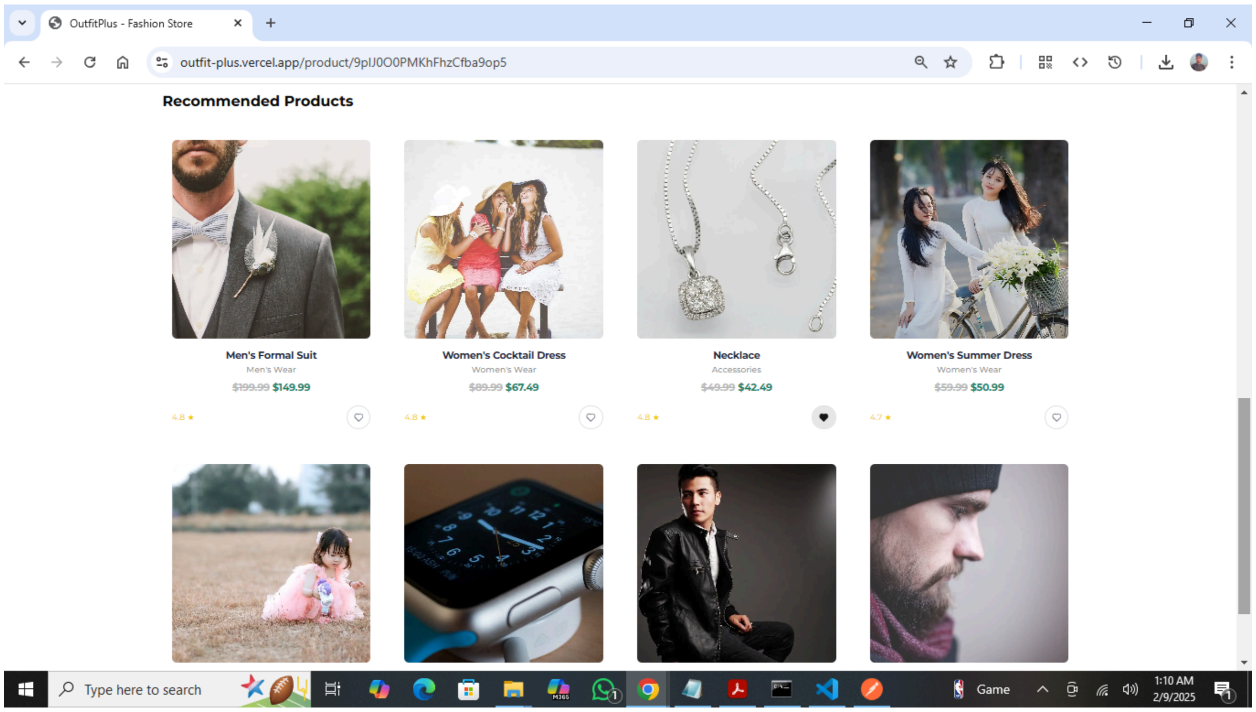




Additional Features Implemented

- **Related Products Section**: Displays similar products based on category.
- **User Profile Component**: Shows user-specific preferences and wishlist items.

Screenshot:


OutfitPlus - Fashion Store
outfit-plus.vercel.app/product/9plJ0O0PMKhFhzCfba9op5
Type here to search
Game
1:10 AM
2/9/2025

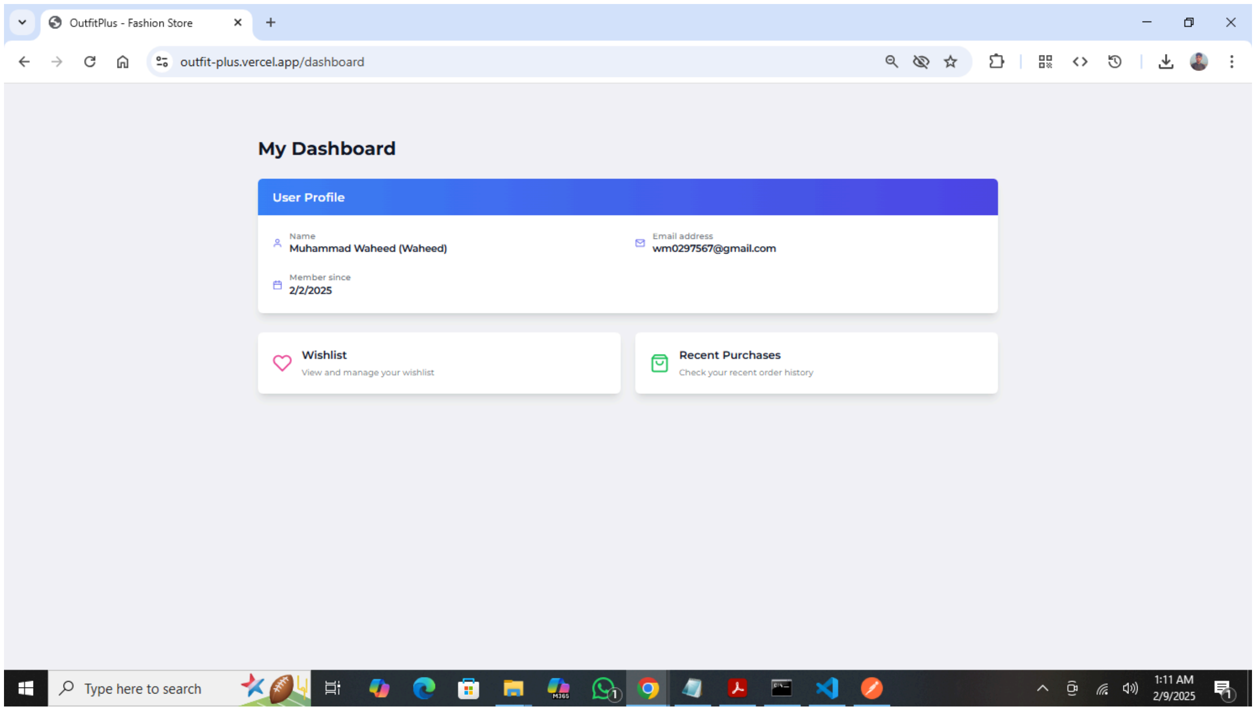
Recommended Products

Men's Formal Suit
Men's Wear
~~$199.99~~ $149.99
4.8 ★

Women's Cocktail Dress
Women's Wear
~~$89.99~~ $67.49
4.8 ★

Necklace
Accessories
~~$49.99~~ $42.49
4.8 ★

Women's Summer Dress
Women's Wear
~~$59.99~~ $50.99
4.7 ★



My Dashboard

User Profile

Name
Muhammad Waheed (Waheed)

Email address
wm0297567@gmail.com

Member since
2/2/2025

Wishlist
View and manage your wishlist

Recent Purchases
Check your recent order history

Code Deliverables

Key Components

ProductCard.tsx

```
import type React from "react"
import Image from "next/image"
import Link from "next/link"
import { Wishlist } from "@/components/wishlist"

export interface Product {
  _id: string
  id: string
  name: string
  imageUrl: string
  category: string
  price: number
  discount: number
  rating: string
}

interface ProductCardProps {
  product: Product
}

const ProductCard: React.FC<ProductCardProps> = ({ product }) => {
  const discountedPrice = product.price * (1 - (product.discount || 0) / 100)

  const formatPrice = (price: number | string) => {
    const numPrice = typeof price === "string" ? Number.parseFloat(price) : price
    return isNaN(numPrice) ? "0.00" : numPrice.toFixed(2)
  }

  return (
    <div className="flex flex-col items-center p-4 rounded-lg transition-all transform
hover:scale-105 hover:shadow-lg">
      <Link href={`/product/${product._id}`} className="relative w-full pb-[100%]">
        <Image
          src={product.imageUrl || "/placeholder.svg"}
          alt={product.name}
          layout="fill"
          objectFit="cover"
          className="rounded-lg"
        />
      </Link>
      <div className="mt-4 text-center w-full">
        <h3 className="text-[#252B42] text-[16px] font-bold">{product.name}</h3>
        <p className="text-[#737373] text-[14px]">{product.category}</p>
        <p className="text-[#BDBDBD] text-[16px] font-bold mt-2">
          <span className="line-through">${formatPrice(product.price)}</span>{" "}
          <span className="text-[#23856D]">${formatPrice(discountedPrice)}</span>
        </p>
        <div className="flex justify-between items-center mt-4">
          <div className="text-[#F3CD03] text-[14px]">{product.rating} ★</div>
          <Wishlist productId={product._id} />
        </div>
      </div>
    </div>
  )
}

export default ProductCard
```

SearchBar.tsx

```
"use client";
import { useState } from "react";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Search } from "lucide-react";

interface SearchBarProps {
  onSearch: (query: string) => void;
}

export function SearchBar({ onSearch }: SearchBarProps) {
  const [query, setQuery] = useState("");

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    onSearch(query);
  };

  return (
    <form
```

```
      onSubmit={handleSubmit}
      className="flex w-full max-w-sm items-center space-x-2 mb-8"
    >
      <Input
        type="text"
        placeholder="Search products..."
        value={query}
        onChange={(e) => setQuery(e.target.value)}
      />
      <Button type="submit">
        <Search className="h-4 w-4 mr-2" />
        Search
      </Button>
    </form>
  );
}
```

API Integration

API for Paginated Products

```
import type { NextApiRequest, NextApiResponse } from "next";
import { getPaginatedProducts } from "@/sanity/lib/client";

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse
) {
  const { page = "1", perPage = "8", categories, search } = req.query;

  try {
    const result = await getPaginatedProducts(
      Number(page),
      Number(perPage),
      categories
        ? Array.isArray(categories)
          ? categories
          : [categories]
        : undefined,
      search ? String(search) : undefined
    );

    res.status(200).json(result);
  } catch (error) {
    console.error("Error fetching products:", error);
    res.status(500).json({ error: "Error fetching products" });
  }
}
```

Technical Report

Steps Taken

1. **Built Dynamic Components**: Created `ProductCard`, `ProductList`, and `SearchBar`.
2. **Implemented API Fetching**: Integrated API for paginated and filtered product data.
3. **Enabled Routing**: Configured dynamic routes for individual product pages.
4. **Added Search and Filters**: Built filtering logic and implemented a responsive search bar.
5. **Optimized Performance**: Used `Suspense` for better loading experience.

Challenges Faced & Solutions

- **Challenge**: Ensuring accurate pagination and filtering.
    - **Solution**: Implemented robust query handling with server-side functions.
- **Challenge**: Managing dynamic image loading.
    - **Solution**: Used `next/image` for optimized image handling.

Best Practices Followed

- **Component-based architecture**
- **Efficient API calls and error handling**
- **User-friendly UI with responsiveness**

Submission Format: This document is provided in PDF/Markdown as per the guidelines.