DAY 4 - BUILDING DYNAMIC FRONTEND FOR SHOPVERSE

PRODUCT LISTING PAGE

Gender

- [ ] Men
- [ ] Women

Kids

- [ ] Boys
- [ ] Girls

Shop By Price

- [ ] Under ₹ 2 500.00
- [ ] ₹ 2 501.00 - ₹ 7 500.00

Promo Exclusion

Air Jordan 1 Elevate Low

Women's Shoes

1 Colour

Wishlisted ♥

MRP : ₹ 11895

Best Seller

Nike Pegasus 40

Men's Running Shoes

1 Colour

Wishlisted ♥

MRP : ₹ 9795

Best Seller

Nike Metcon 8

Men's Training Shoes

1 Colour

Wishlisted ♥

MRP : ₹ 10295

Just In

Nike Air Force 1 Mid '07

Men's Shoes

1 Colour

Add whislist ♡

MRP : ₹ 10795

Just In

Nike Air Force 1 PLT.AF.ORM

Women's Shoes

1 Colour

Wishlisted ♥

MRP : ₹ 8695

Just In

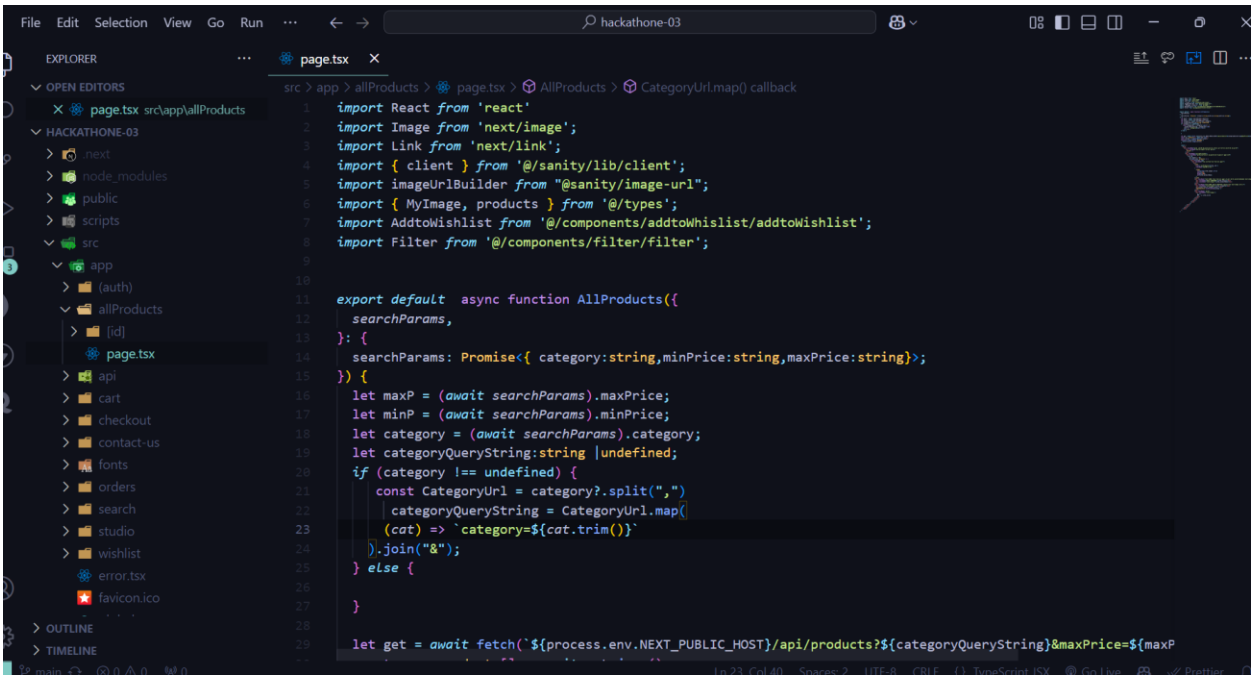
Nike Standard Issue Basketball Jersey

Women's Basketball Jersey

1 Colour

Wishlisted ♥

MRP : ₹ 2895



For Day 4, I implemented a dynamic product listing page with filtering options for categories and price range. The page fetches product data from an API based on user-selected filters and displays it in a responsive grid layout. Product images are fetched and optimized using Sanity's image builder, ensuring high-quality visuals. Additionally, I integrated a wishlist feature that allows users to add products they like. TailwindCSS was used to make the layout responsive and ensure a smooth user experience across different devices.

PRODUCT DETAIL PAGE

localhost:3000/allProducts/2CvABo0fgTAGzXYTL9VQAH

Air Jordan 1 Elevate Low

The Air Jordan 1 Elevate Low features a clean, minimal design with premium materials. Its platform sole adds a touch of height and modern edge, while ensuring comfort and durability for all-day wear.

Color

₹ 11895

Add To Cart

localhost:3000/allProducts/8rHgms8cRnJSVtFj2mfif4

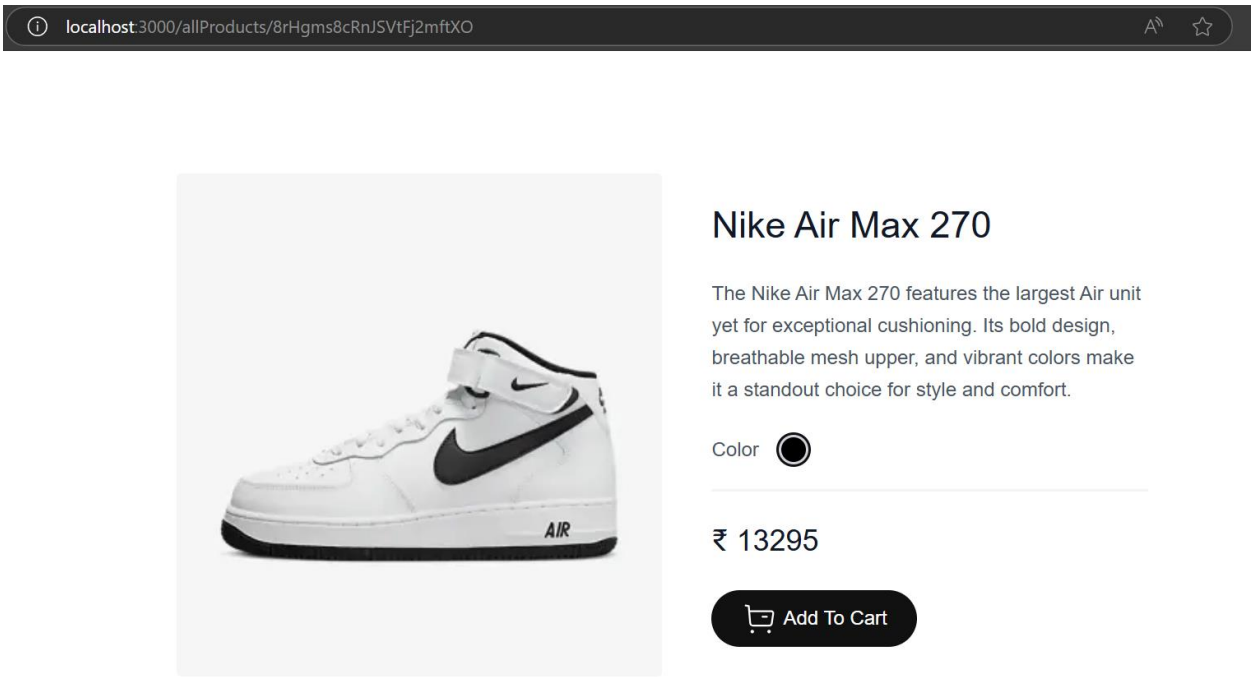
Nike Air Force 1 PLT.AF.ORM

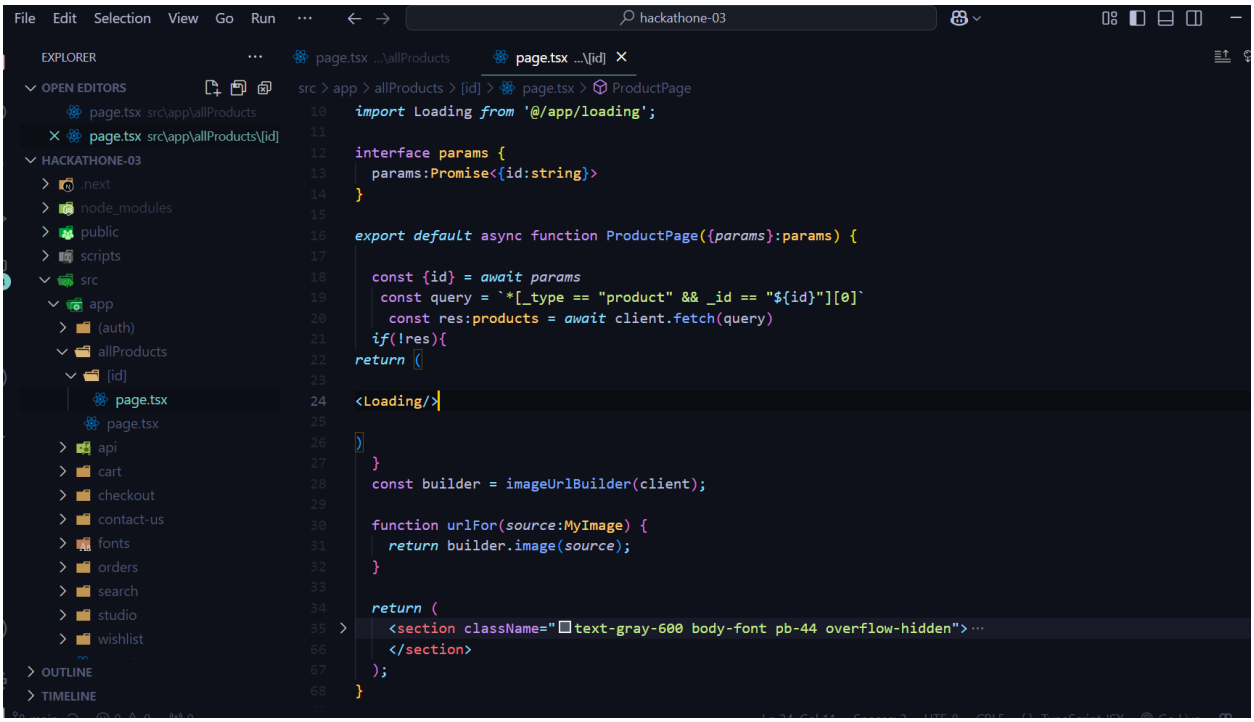
The Nike Air Force 1 PLT.AF.ORM elevates the iconic AF1 silhouette with bold design updates. Featuring a platform sole and premium materials, this women's shoe redefines style and comfort in a fresh, modern way.

Color

₹ 8695

Add To Cart

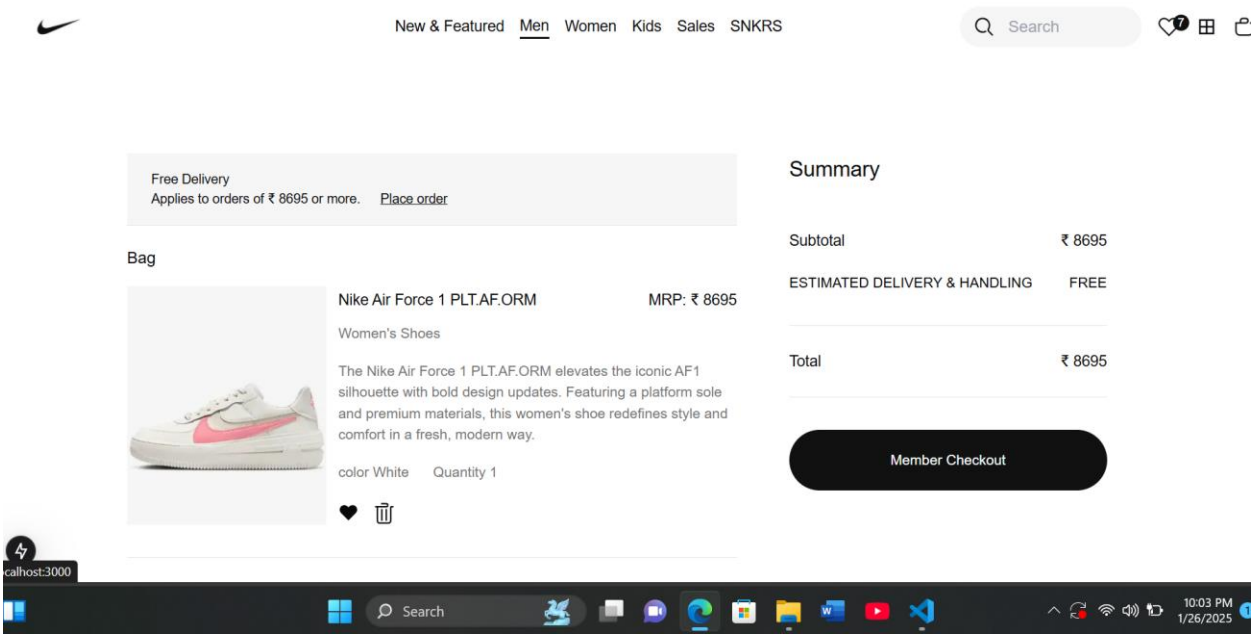


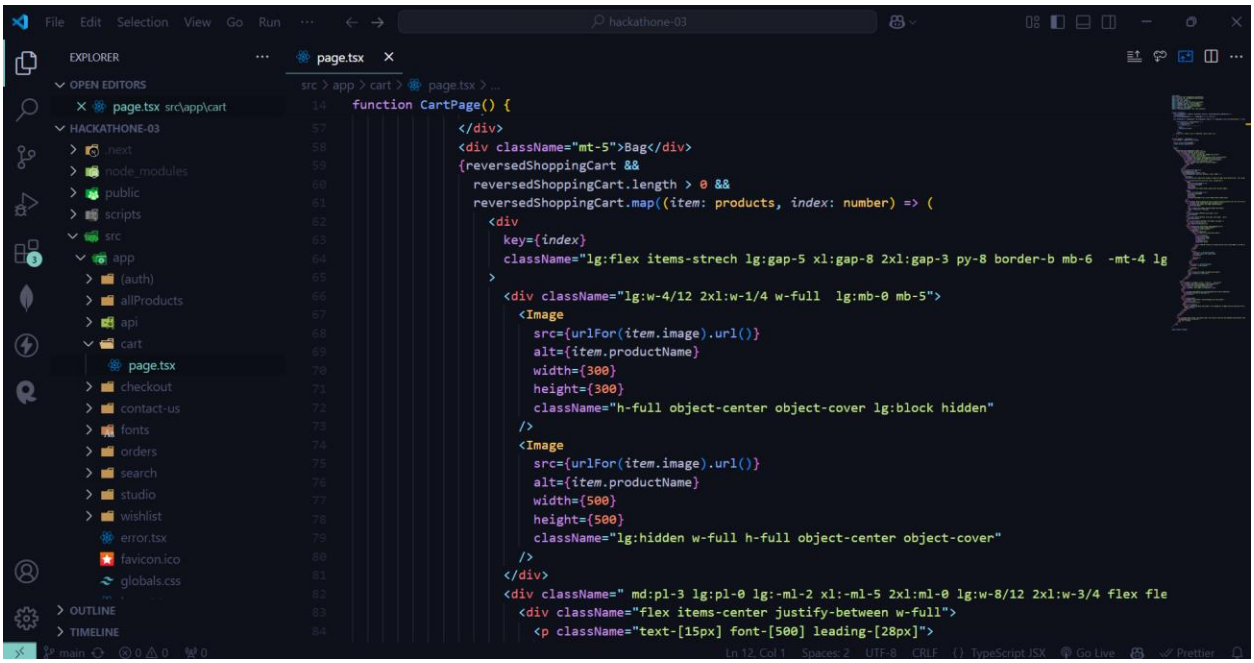


The Product Detail Page fetches product data using Next.js's dynamic routing to display detailed information about a specific product. The product is queried from Sanity CMS using its unique `id` via the `client.fetch()` method. The page shows the product's image, description, colors, and price, with image optimization handled by Sanity's

`imageUrlBuilder`. If the product data or image is loading, a `Skeleton` loader is displayed to improve the user experience. The `Color` component allows users to view available product colors, while the `AddToCart` component lets users add items to their cart. A responsive layout is used, with TailwindCSS ensuring the page looks good on all screen sizes. In case of missing product data, a `Loading` component is rendered to indicate that the page is still fetching the content.

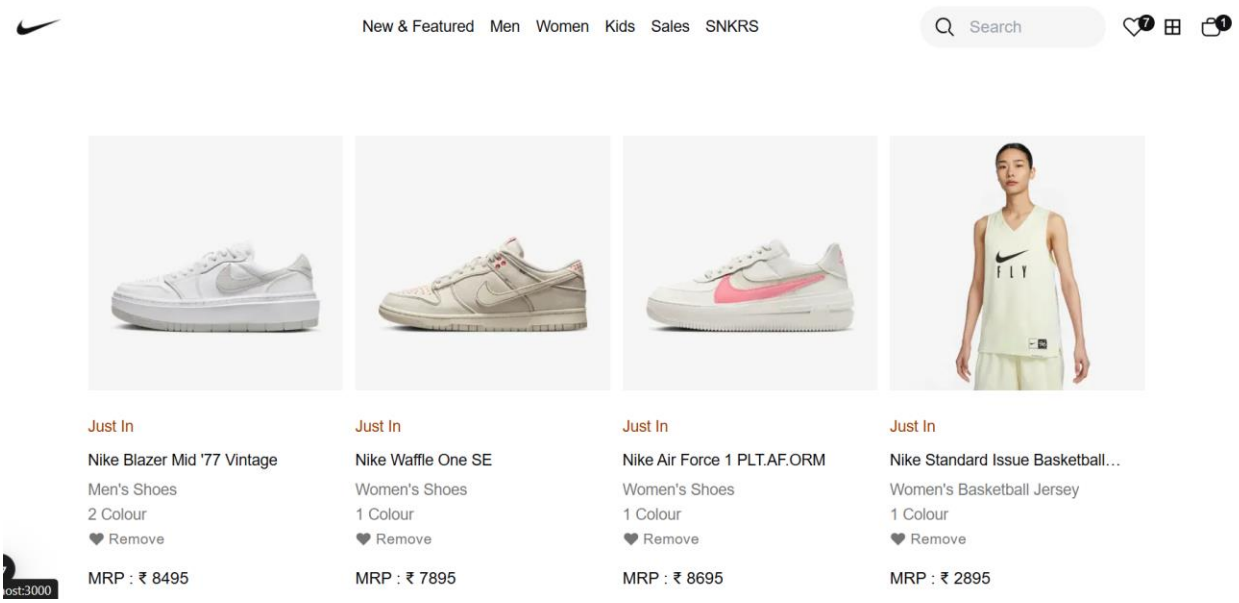
CART PAGE

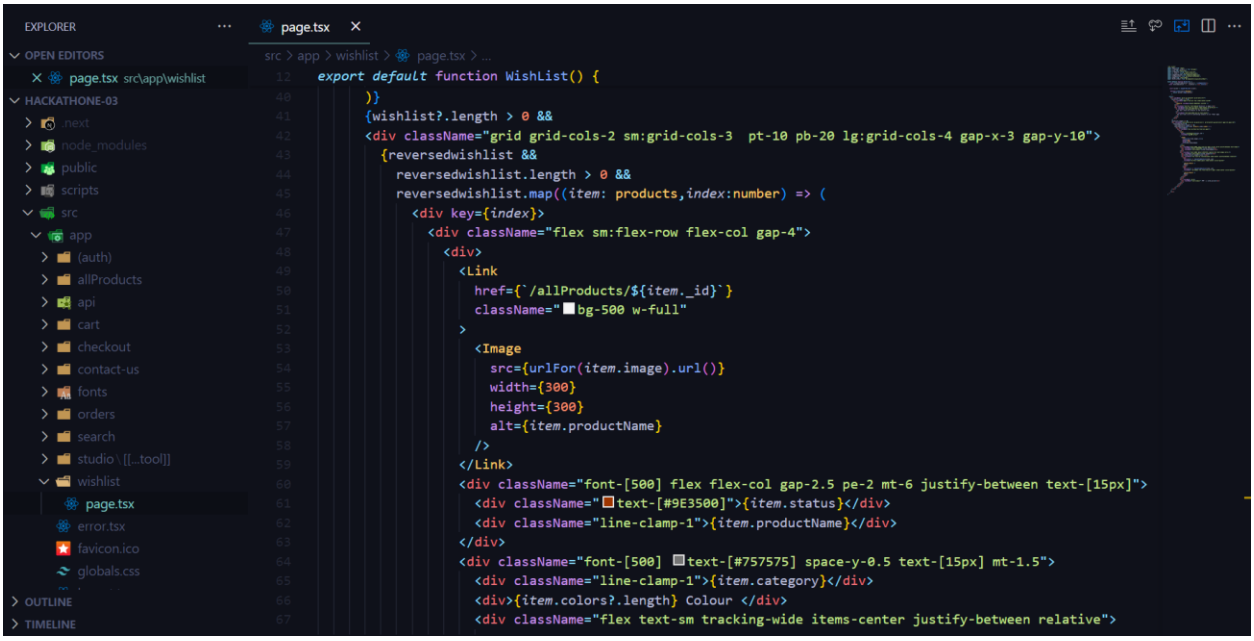




The **Cart Page** displays the user's shopping cart with a detailed view of each item, including the product image, name, price, description, selected color, and quantity. Users can add products to their wishlist or remove items from the cart using the heart and trash icons, respectively. The subtotal for the cart is dynamically calculated and displayed, along with a "Free Delivery" message for eligible orders. If the cart is empty, a message with a shopping cart icon prompts the user. The page features a responsive layout using TailwindCSS, ensuring compatibility across various devices. For checkout, users can proceed with a "Member Checkout" button that redirects to the checkout page. If items are present, their details are rendered with images optimized via Sanity's `imageUrlBuilder`. The page also allows for quick adjustments to cart items and provides a smooth, interactive shopping experience.

WISHLIST PAGE

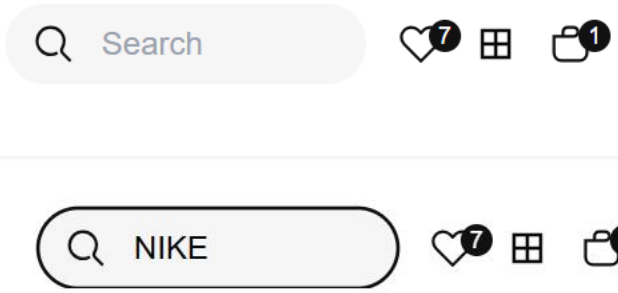


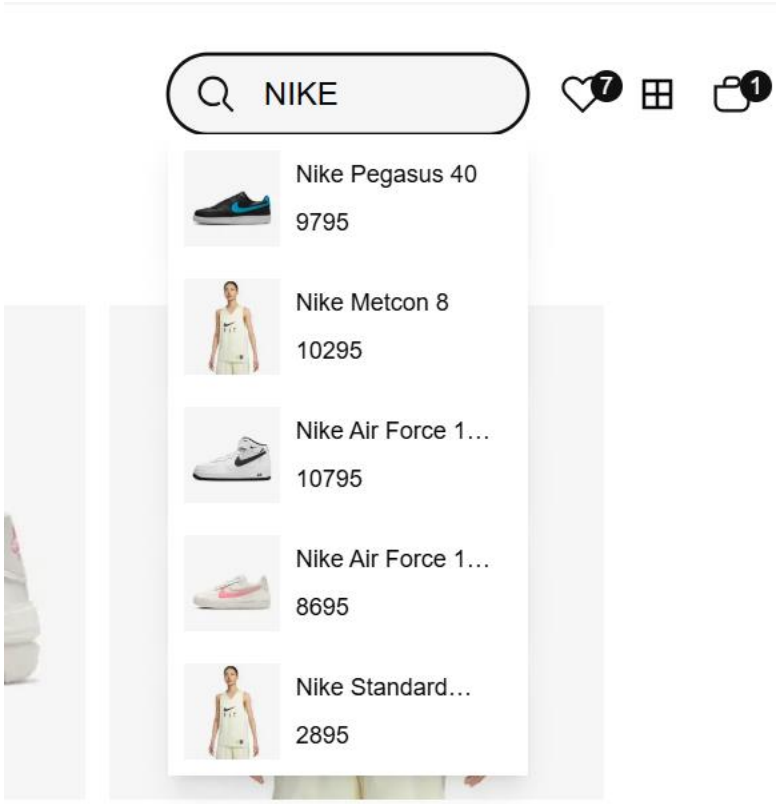


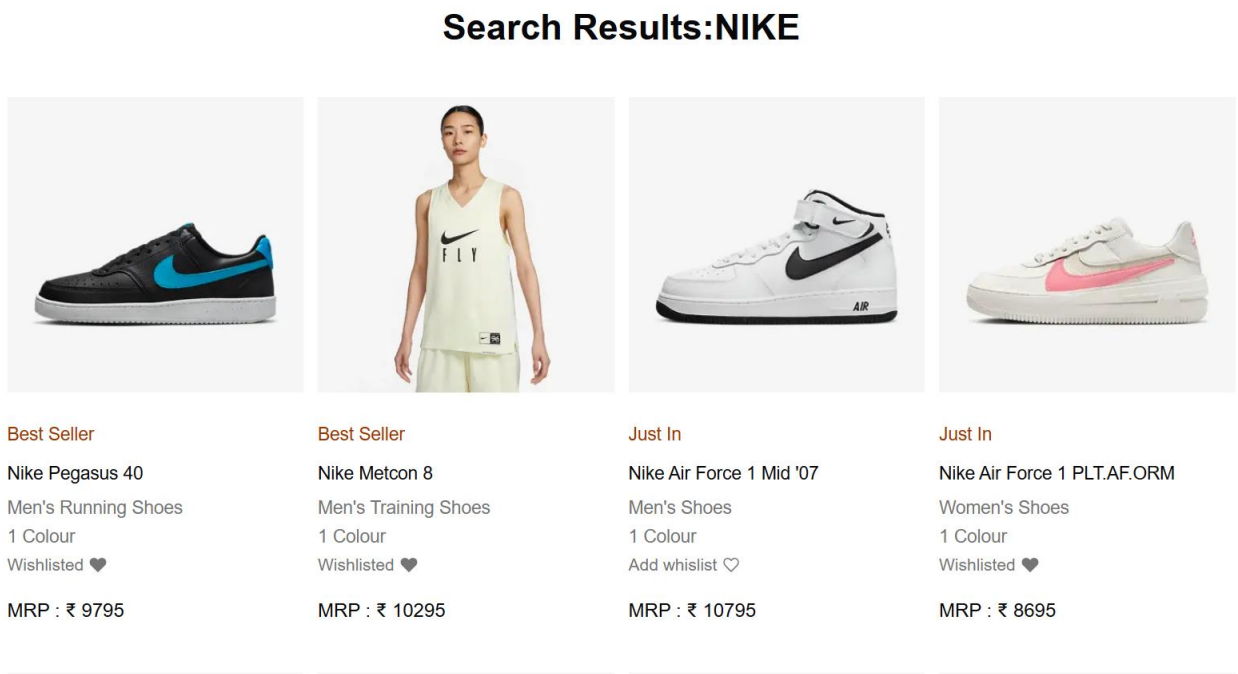
The **Wishlist Page** allows users to view and manage products they've saved for later. If the wishlist is empty, a message with an icon is displayed, encouraging users to explore the shop. The page dynamically displays each saved product with an image, name, status, category, available colors, and price. The images are optimized using Sanity's `imageUrlBuilder` to ensure fast loading times. Users can easily remove items from their wishlist by clicking the "Remove" button next to each item, which triggers the `removeFromWishlist` function from the app context. The page is responsive, showing a grid of products that adjusts to various screen sizes using TailwindCSS. Each product links

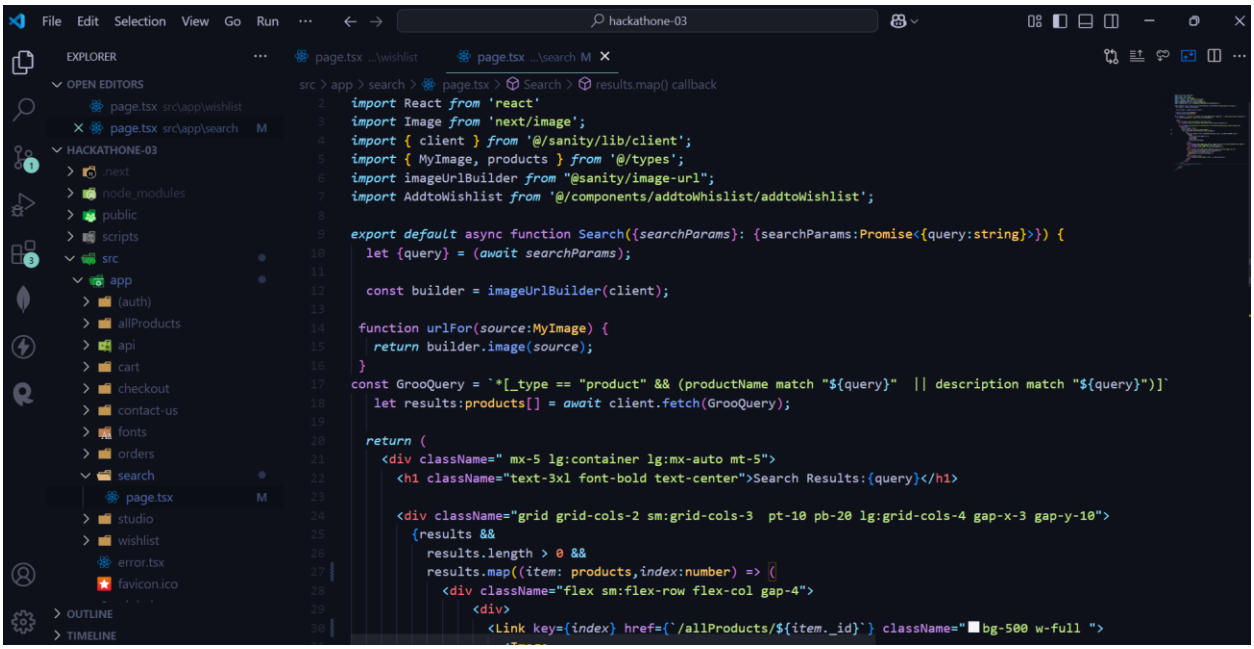
to its detail page for more information. Overall, the wishlist provides a seamless experience for managing saved products within the shopping process.

SEARCH BAR





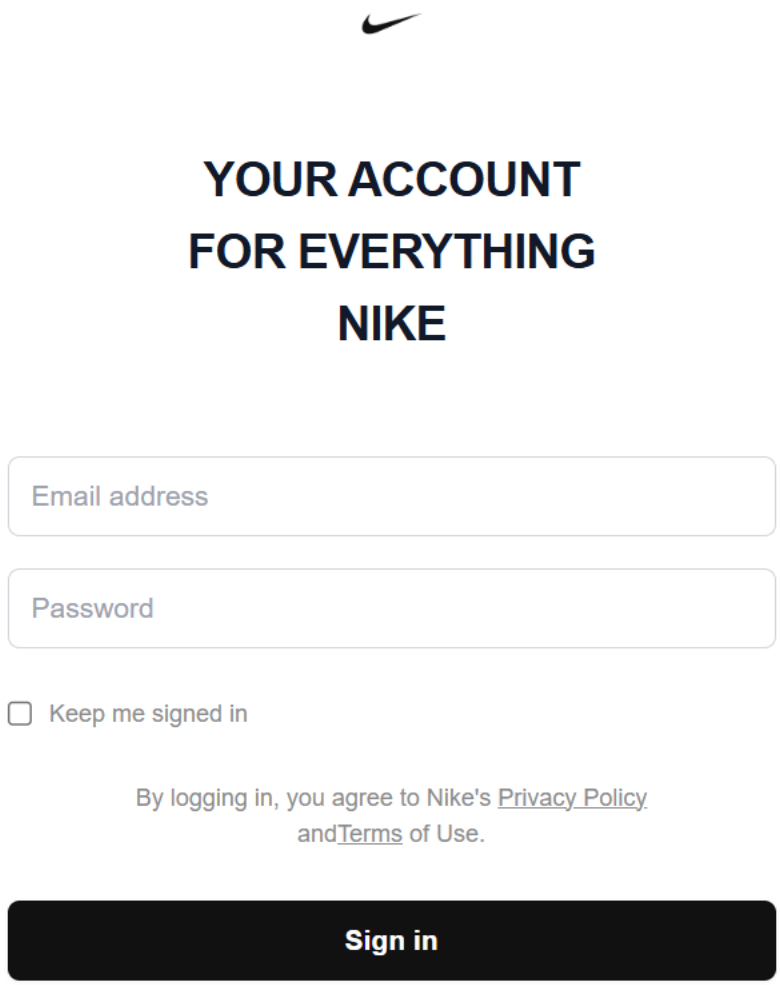




The **Search Page** displays results based on the user's search query. It dynamically fetches products from Sanity CMS using the `client.fetch()` method with a query that matches either the product name or description. If results are found, they are displayed in a responsive grid layout, showing the product image, name, status, category, available colors, and price. Each product is clickable, linking to its detailed page. The images are optimized using Sanity's `imageUrlBuilder` for better performance. Additionally, the page

includes an "Add to Wishlist" feature for users to save their favorite items. If no results are found, the user is still presented with an organized and user-friendly layout. TailwindCSS is used to ensure the layout adapts well to various screen sizes.

AUTHENTICATION PAGE



```usefulness_ref_1
"use client"
import Link from 'next/link';
import React, { useEffect, useState } from 'react'
import { useRouter } from "next/navigation";
import toast from "react-hot-toast";

function LoginPage() {
 const [email,setEmail] = useState<string>('')
 const [password,setPassword] = useState<string>('')
 const [disabledbtn, setDisabledbtn] = useState<boolean>(false)
  const router = useRouter();

const Login = async (evt: React.FormEvent<HTMLFormElement>) => {
   evt.preventDefault();
   setDisabledbtn(true);
   let data = { email,password};
   let res = await fetch("/api/login", {
     method: "POST",
     headers: {
       "Content-Type": "application/json",
     },
     body: JSON.stringify(data)
   });
   setDisabledbtn(false);
   let json = await res.json()
  if(json.success){
     localStorage.setItem("token",json.token)
     toast.success("You have successfully logged in")
     router.push('/')
```

You have successfully logged in

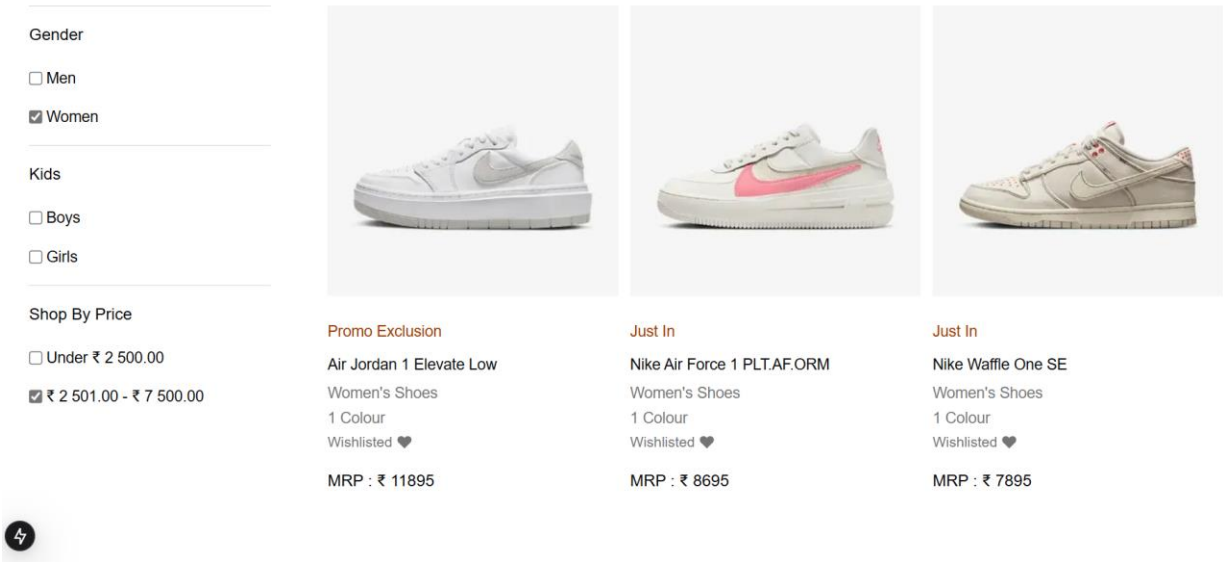
w & Featured Men Women Kids Sales SNKRS

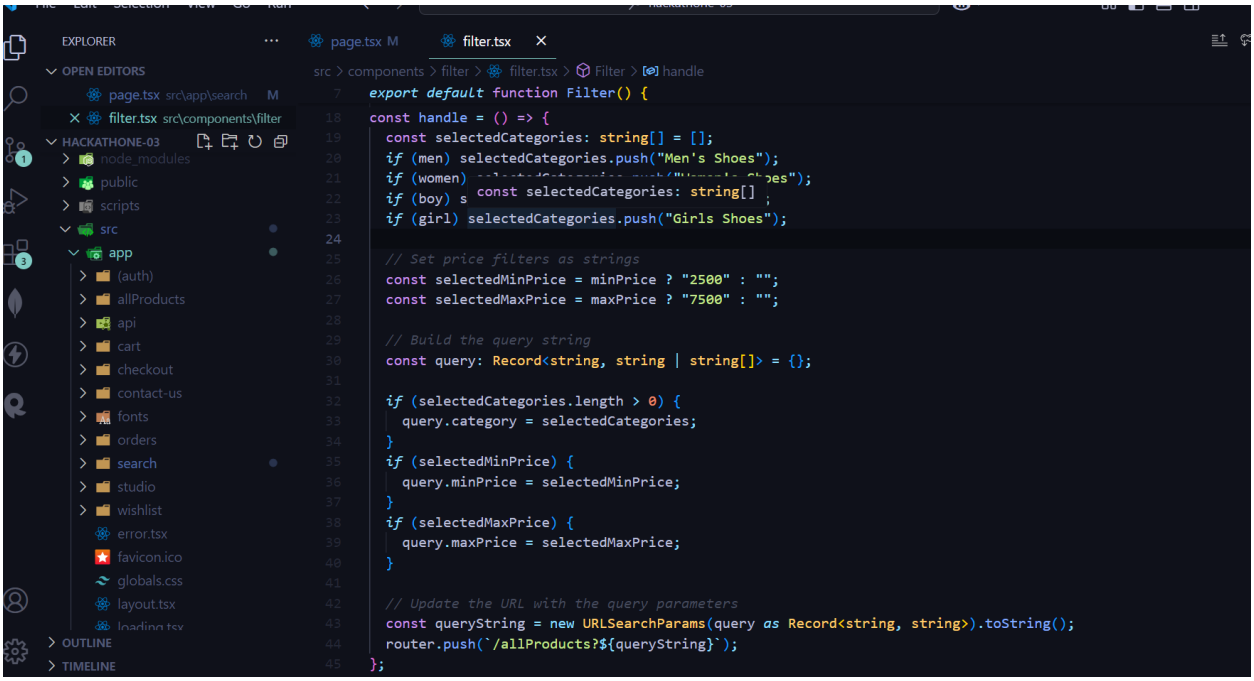
```usefulness_ref_2
   });
   setDisabledbtn(false);
   let json = await res.json()
  if(json.success){
     localStorage.setItem("token",json.token)
     toast.success("You have successfully logged in")
     router.push('/')
  }else{
  toast.error("INVALID CREDENTIALS")

  }
  setEmail('')
  setPassword('')
 }
```

The **Login Page** allows users to authenticate by entering their email and password. The form captures the user's input, and upon submission, it sends a POST request to the

`/api/login` route. If the credentials are valid, the user is logged in, a token is saved to `localStorage`, and a success message is shown using `toast`. The user is then redirected to the homepage. If invalid credentials are entered, an error message is displayed. If the user is already logged in (i.e., a token exists in `localStorage`), they are redirected to the homepage. The page also includes a link to the **Join Us** page for new users and a "Keep me signed in" checkbox (hidden for small screens).

FILTER APPLY FUNCTNALITY



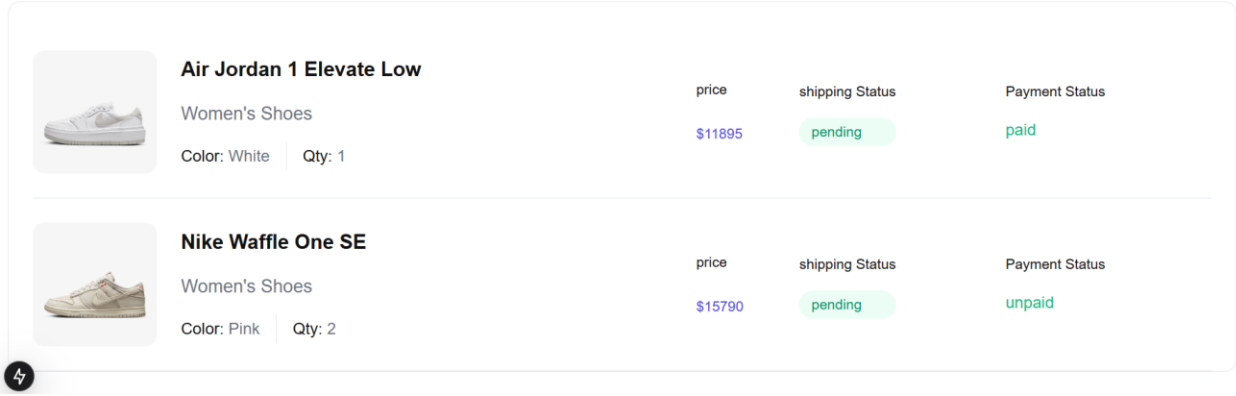


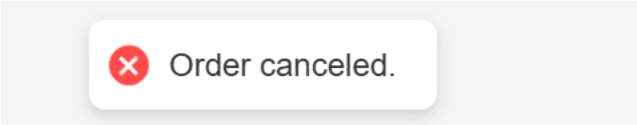
The **Filter Component** allows users to filter products by gender (Men, Women, Boys, Girls) and price range (Under ₹2,500 or ₹2,501 to ₹7,500). The state of these filters is managed with React's `useState` and updates the URL query parameters when a filter is applied. The component reacts to changes in the filter options and updates the product list accordingly. Additionally, a mobile-friendly version is displayed when the `filter` state is `true`, featuring collapsible filters and a close button. The filter state is controlled through the `setFilter` function from the context.

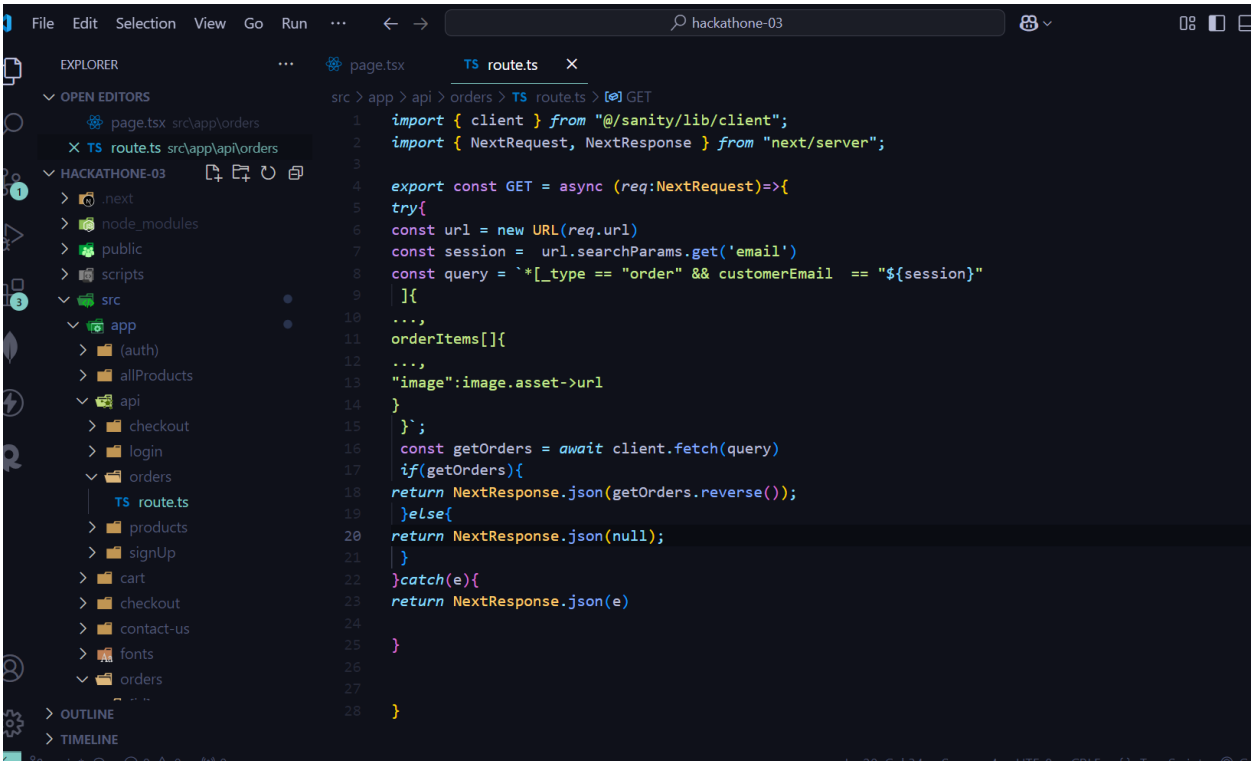
ORDER LISTING PAGE

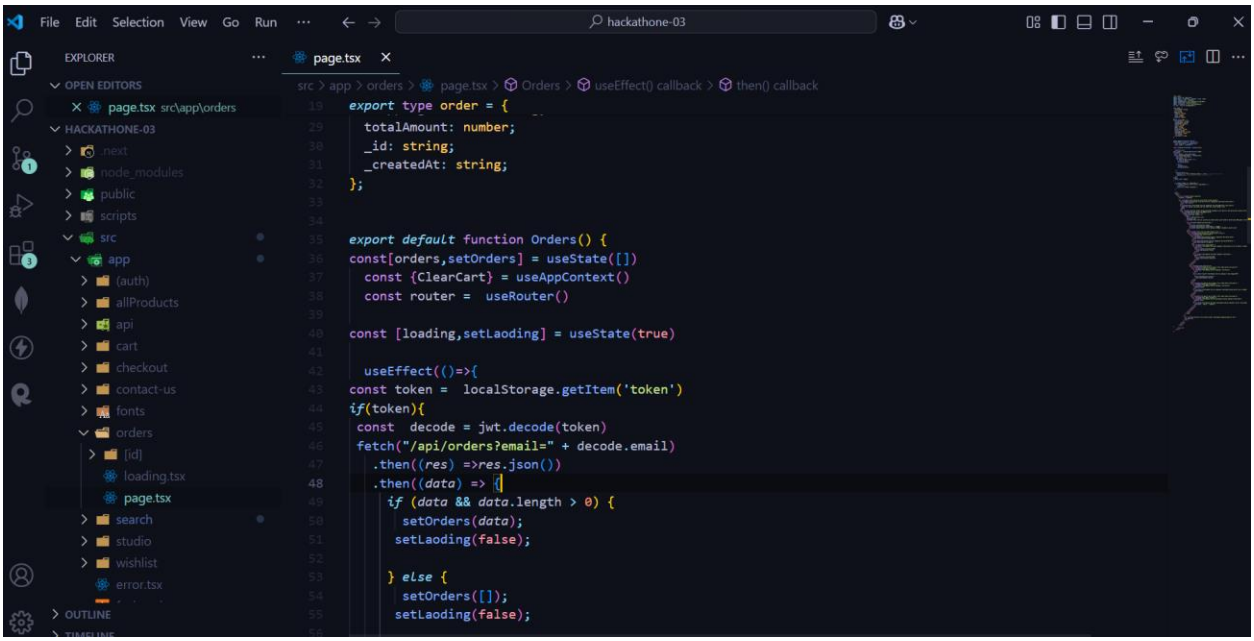
Your order history

Thanks for making a purchase you can check your order summary from below









The **Order Listing Page** displays a user's past orders. It fetches orders associated with the logged-in user's email using a JWT token stored in localStorage. Each order includes details like product name, quantity, price, shipping status, and payment status. Users can click on individual orders to view more detailed information. The page shows a loading state while fetching data and handles error scenarios gracefully. If no orders exist, a "No order Yet" message is shown. The page also supports automatic cart clearing and toast notifications for order cancellations.

ORDER DETAIL PAGE

localhost:3000/orders/N3HGTmq4DJkdZuEv8bg0Nf

Your Order Confirmed

Hello, zeeshan 1231321313

Your order has been completed and be delivery in only two days.

Delivery Date	Order	Payment Method	Address
1/27/2025	**#N3HGTmq4DJkdZuEv**	AMEX	**korangi no 5/1/2...**

Air Jordan 1 Elevate Low

Quantity : **1**

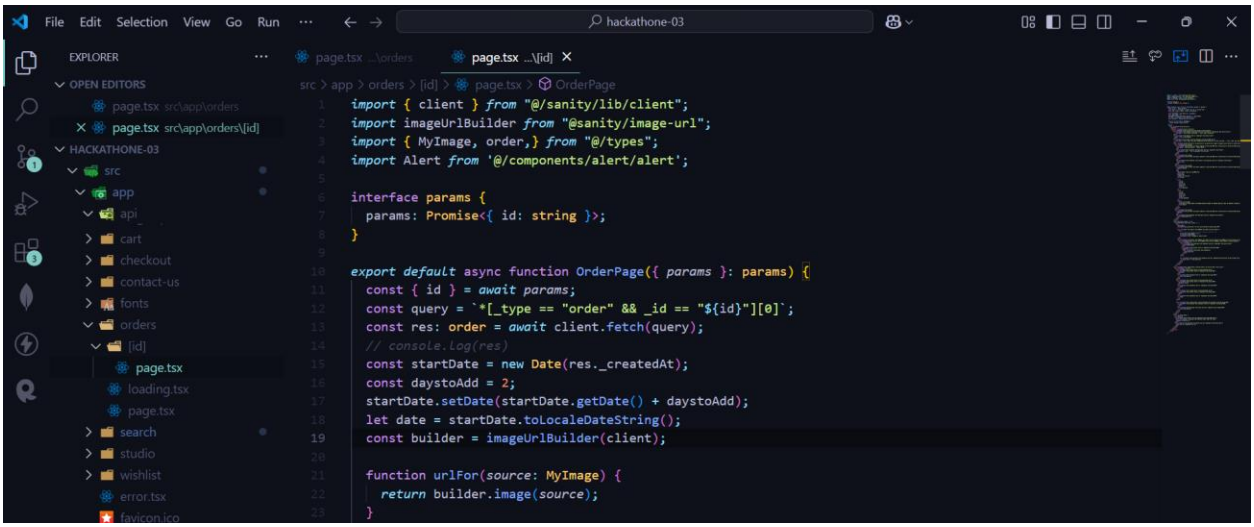
₹ 11895

Subtotal	**₹ 11895**
Shipping Charge	**₹ 0.00**
Taxes	**₹ 0.00**
Total	**₹ 11895**

We'll be sending a shipping confirmation email when the items shipped successfully.

Thank you for shopping with us!

Thank you for your order.



This **Order Details Page** fetches the order details from Sanity based on a specific `id` parameter in the URL. It displays information such as order status, customer name, shipping address, payment method, and delivery date. If the order has been confirmed, it shows the expected delivery date; if canceled, it communicates the cancellation status. The page also renders a detailed list of the ordered items, including the product name, quantity, price, and total amount. The layout adapts for both mobile and desktop views. It provides clear sections for order information, itemized prices, and total cost, ensuring a smooth user experience.

CHECK OUT PAGE

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

Order Summary

Subtotal	₹ 9795
ESTIMATED DELIVERY & HANDLING	FREE
Total	**₹ 9795**

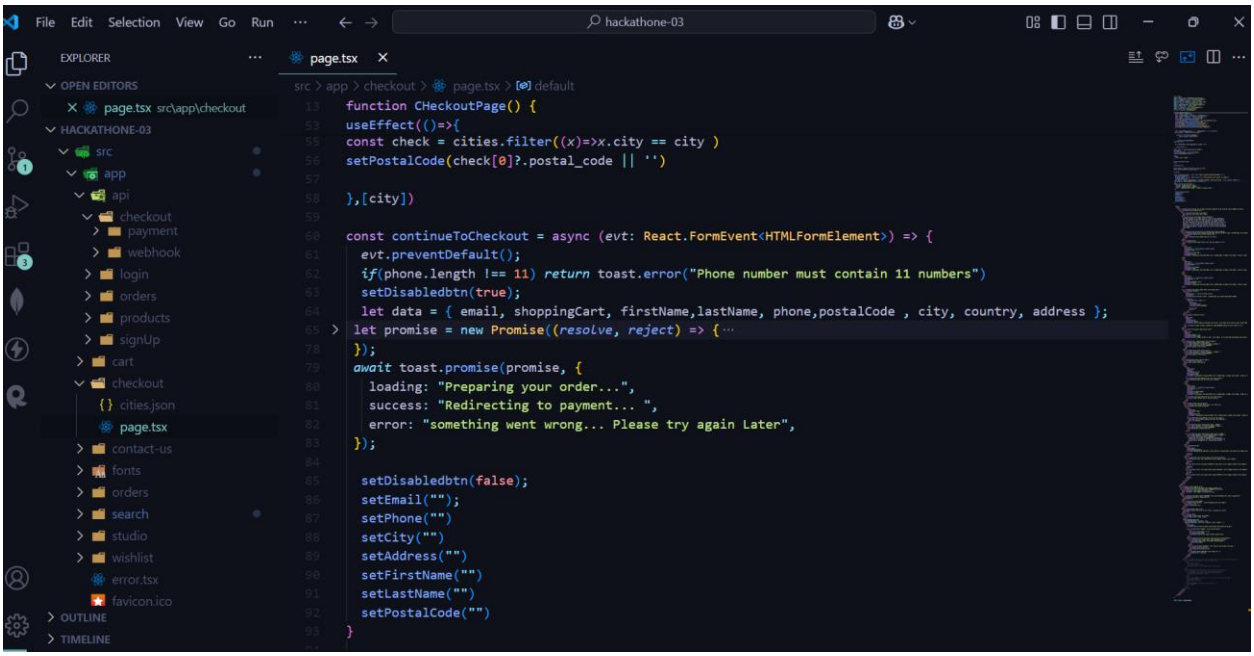
(The total reflects the price of your order, including all duties and taxes)

Arrives Mon, 27 Mar - Wed, 12 Apr

Nike Pegasus 40

color Gray
Quantity 1
₹ 9795

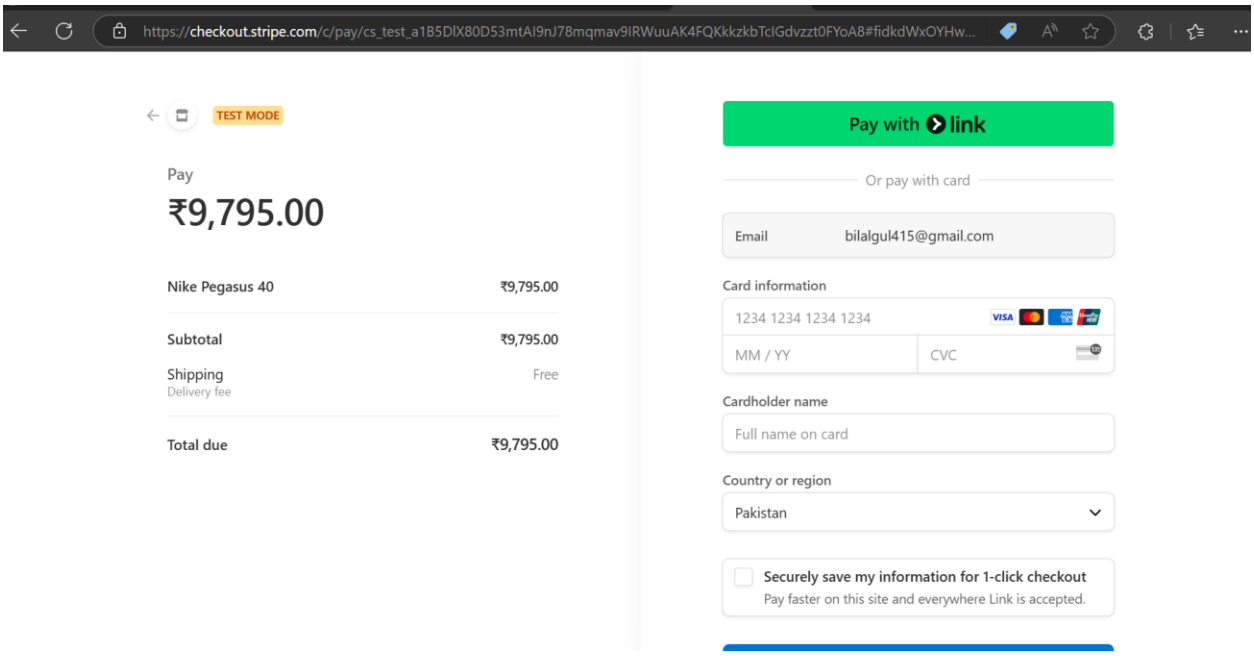
Preparing your order...

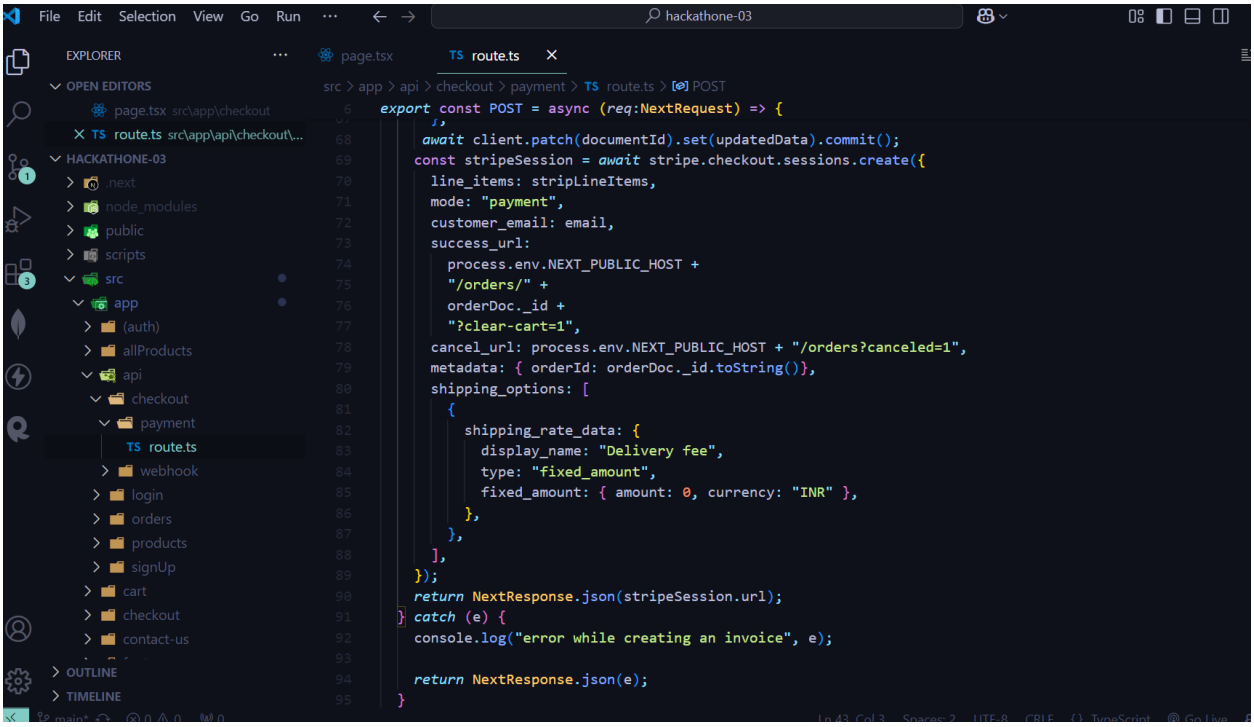


- **Purpose**: The checkout page collects user information, processes the order, and redirects the user to a payment gateway.
- **State Management**: Utilizes `useState` to manage form inputs (email, phone, address, city, etc.) and `useEffect` for initial setup and token validation.
- **Authentication**: Checks for a JWT token in `localStorage` to verify the user is logged in; redirects to login if not authenticated.
- **City Selection & Postal Code**: Users can select a city from a list (`cities.json`), and the postal code is auto-filled based on the selected city.
- **Form Validation**: Ensures the phone number has exactly 11 digits before submission, and shows error messages using `toast`.
- **Order Summary**: Displays a list of products from the shopping cart with details like quantity, color, and price.
- **Checkout API Call**: Submits user data and order details to an API (`/api/checkout/payment`) and handles the response by redirecting to the payment page.
- **UI/UX**: Includes form fields for personal details and payment options, with a disabled submit button during processing to prevent multiple submissions.

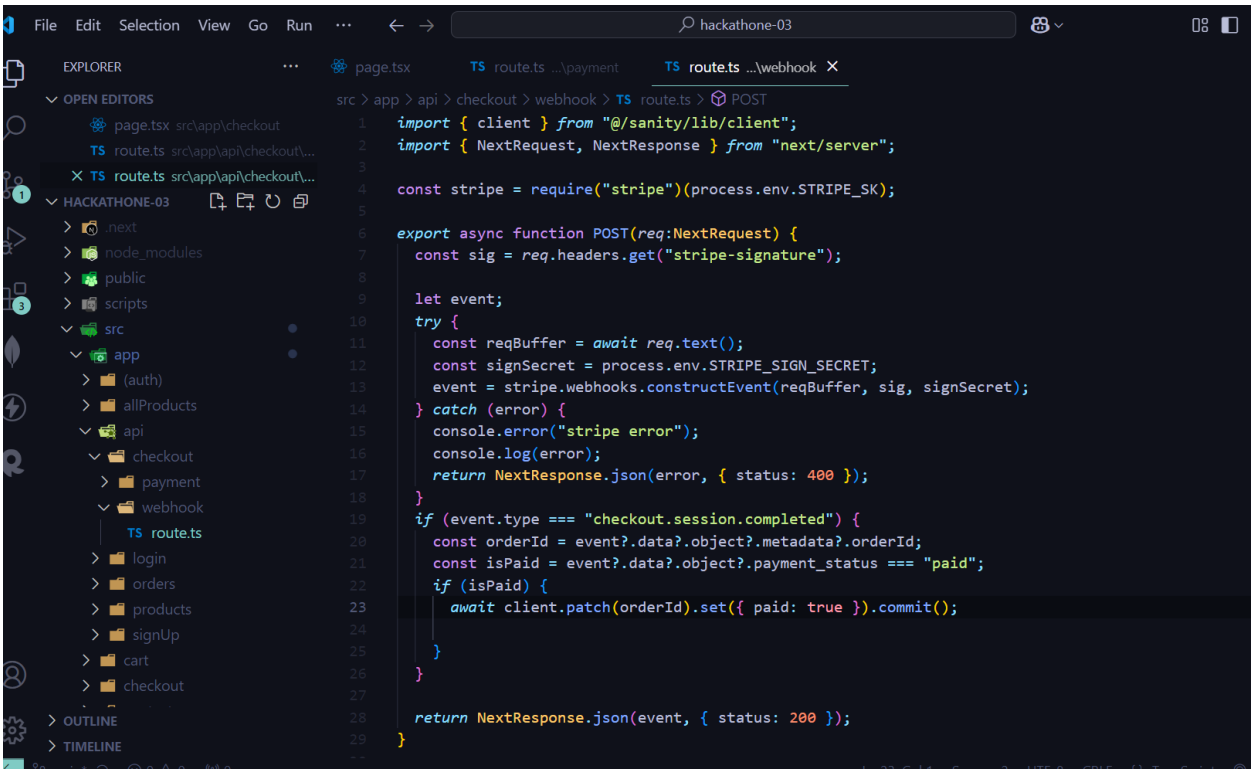
Stripe Payment Gateway Integration (Checkout Session)

This integration handles e-commerce transactions in a Next.js application using **Sanity CMS** for backend management and **Stripe Checkout** for secure payment processing.





Stripe Payment Integration & Webhook



1. Payment Checkout Process:

- **Order Creation**: When a user places an order, an order document is created in **Sanity CMS** with customer details, order items, and shipping information.
- **Stripe Checkout**: The system generates a Stripe Checkout session with the order details and redirects the customer to the Stripe payment page.
- **Success/Failure URLs**: After payment, the customer is redirected to different URLs based on whether the payment was successful or canceled.

2. Stripe Webhook for Payment Confirmation:

- **Webhook Handling**: Stripe sends events (like `checkout.session.completed`) to the configured webhook endpoint to notify the system about the payment status.
- **Order Update**: When the payment is confirmed as successful, the order status in **Sanity CMS** is updated to `paid`.
- **Event Verification**: The webhook verifies the authenticity of events by checking the Stripe signature.

Workflow Overview:

1. **Create an order** in Sanity CMS with customer and product details.

2. **Redirect to Stripe Checkout** for secure payment processing.
3. **Handle Stripe Webhook** to confirm payment and update the order status in Sanity CMS.