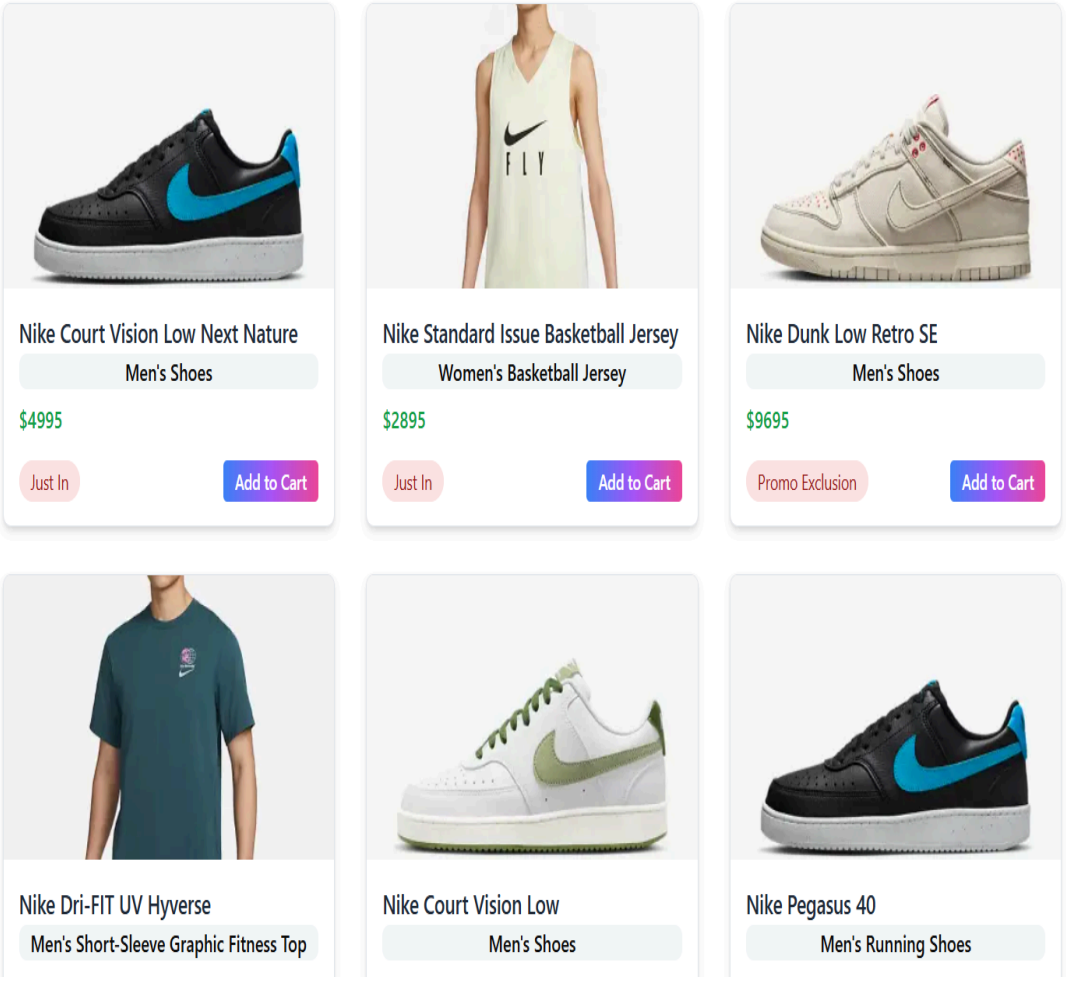
Hackathon 3 day 4

The BestOfAll component is a dynamic product showcase page built with Next.js, Sanity CMS, and Tailwind CSS. It fetches product data from the backend and displays it in a responsive grid. Each product card includes an image, name, category, price, and availability status. Users can click on a product to view details or add it to their cart with a single button. A smooth notification pops up confirming the action using SweetAlert2. The page is interactive, visually appealing, and designed for a seamless shopping experience.

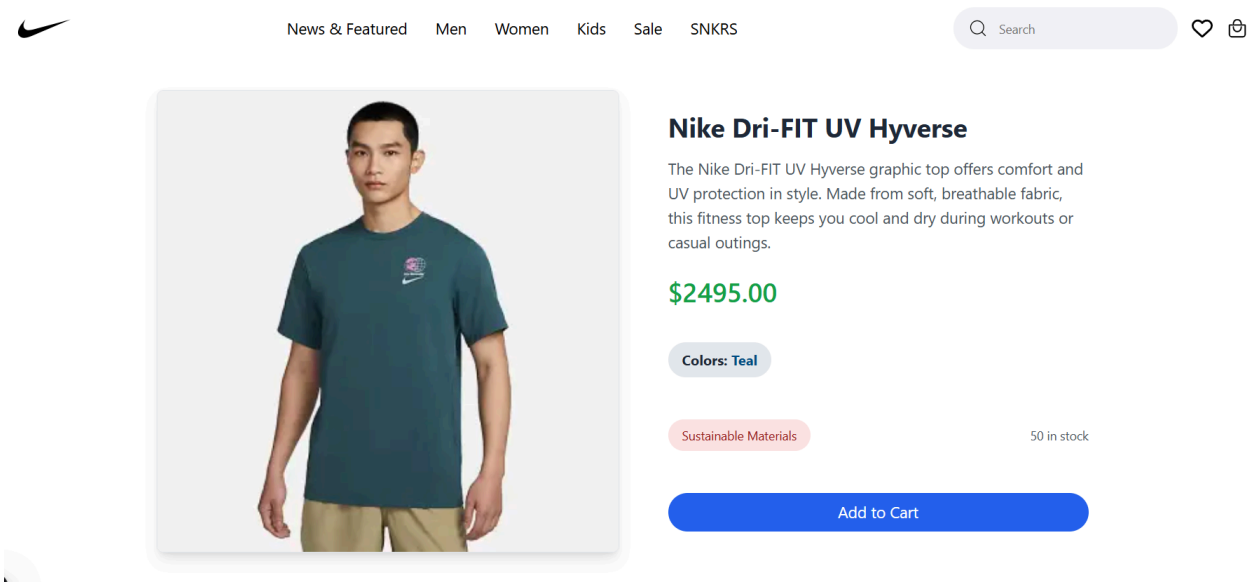
Best Products



Product Page in Next.js with Sanity CMS

DYNAMIC DISPLAY PRODUCT

This Product Page is built using Next.js, Sanity CMS, and TypeScript to dynamically fetch and display product details. The page retrieves product data from Sanity based on a slug parameter and displays relevant product information, including the product name, image, price, description, available colors, stock status, and category.



Key Features:

1. Data Fetching from Sanity CMS

The `getProduct(slug: string)` function queries the Sanity CMS to fetch product details using GROQ query.

The query retrieves specific fields such as product name, price, category, inventory, colors, status, description, and slug.

```
  return client.fetch(
    `*[_type == "product" && slug.current == $slug][0]{
          _id,
          productName,
          _type,
          image,
          price,
          category,
          inventory,
          colors,
          status,
          description,
          slug
        }`,
    { slug }
  );
}

Tabnine | Edit | Test | Explain | Document
export default async function ProductPage({ params }: ProductPageProps) {
  const { slug } = await params;
  const product = await getProduct(slug);
```

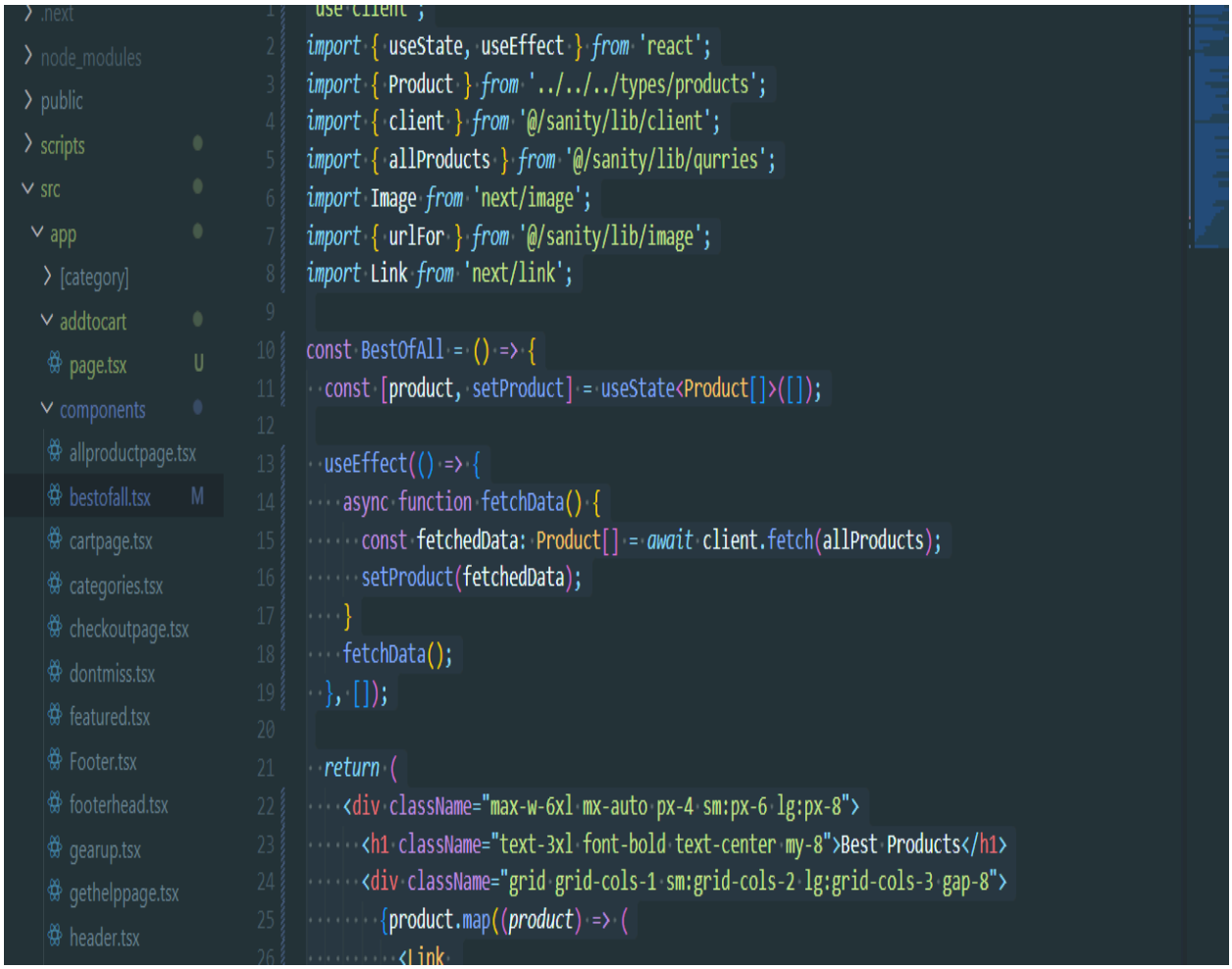
2. Dynamic Routing in Next.js

- The component receives the `slug` from the URL parameters to dynamically fetch and display product details.
- The page is rendered on the server, making the product details available at build time or request time.

3. Product Display Section

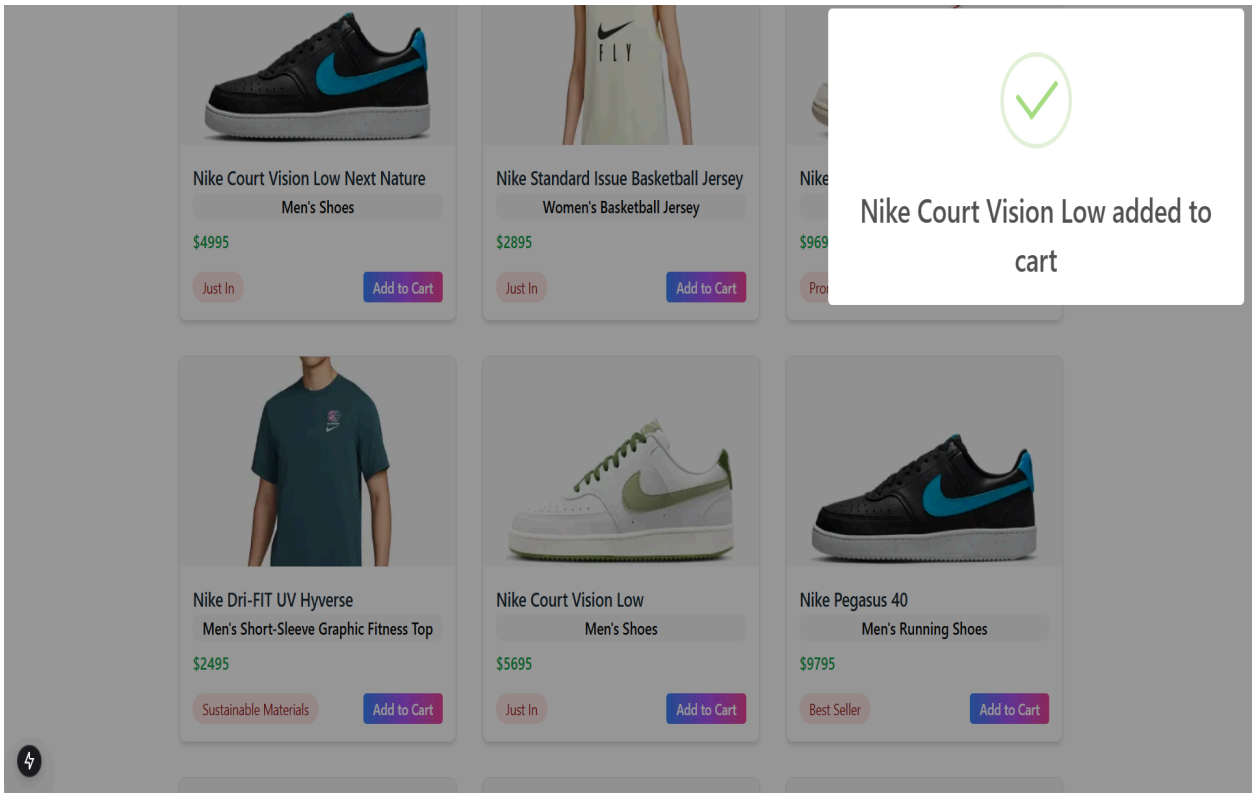
- The product image is displayed using the Next.js `Image` component, ensuring optimized and responsive image loading.
- The product details section includes:
    - Product name (displayed as a large heading)
    - Product description
    - Price (formatted to two decimal places)
    - Available colors (if specified)
    - Stock status (shown as an available or unavailable label)
    - Inventory count

The code snippet demonstrates a React component that fetches a collection of products from a Content Management System (CMS) using the Sanity client and displays them on a web page.



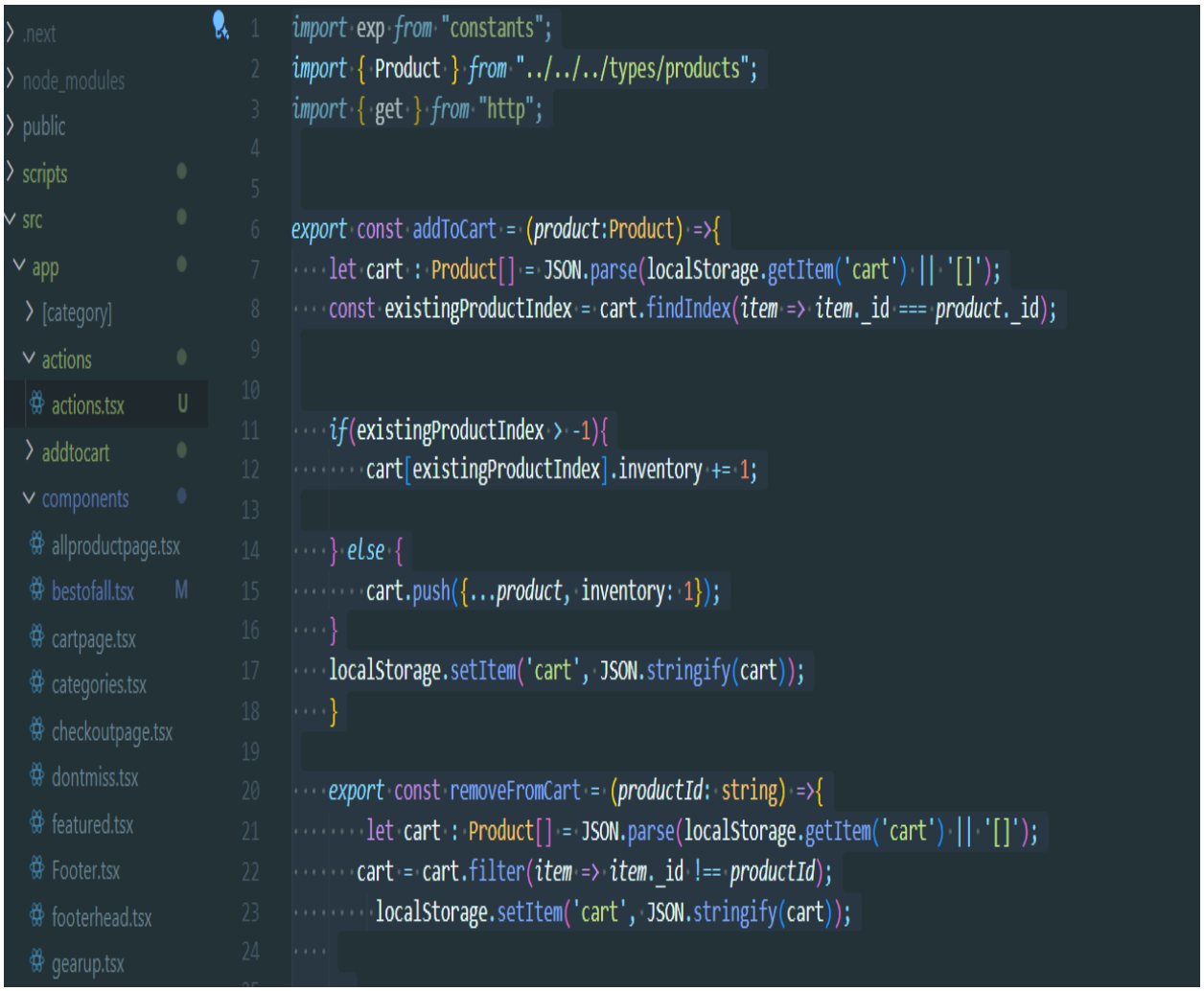
Add To Cart Functionality

An overlay notification appears after clicking "Add to Cart," confirming that the item has been successfully added. This alert features the product name (e.g., "Nike Court Vision Low added to

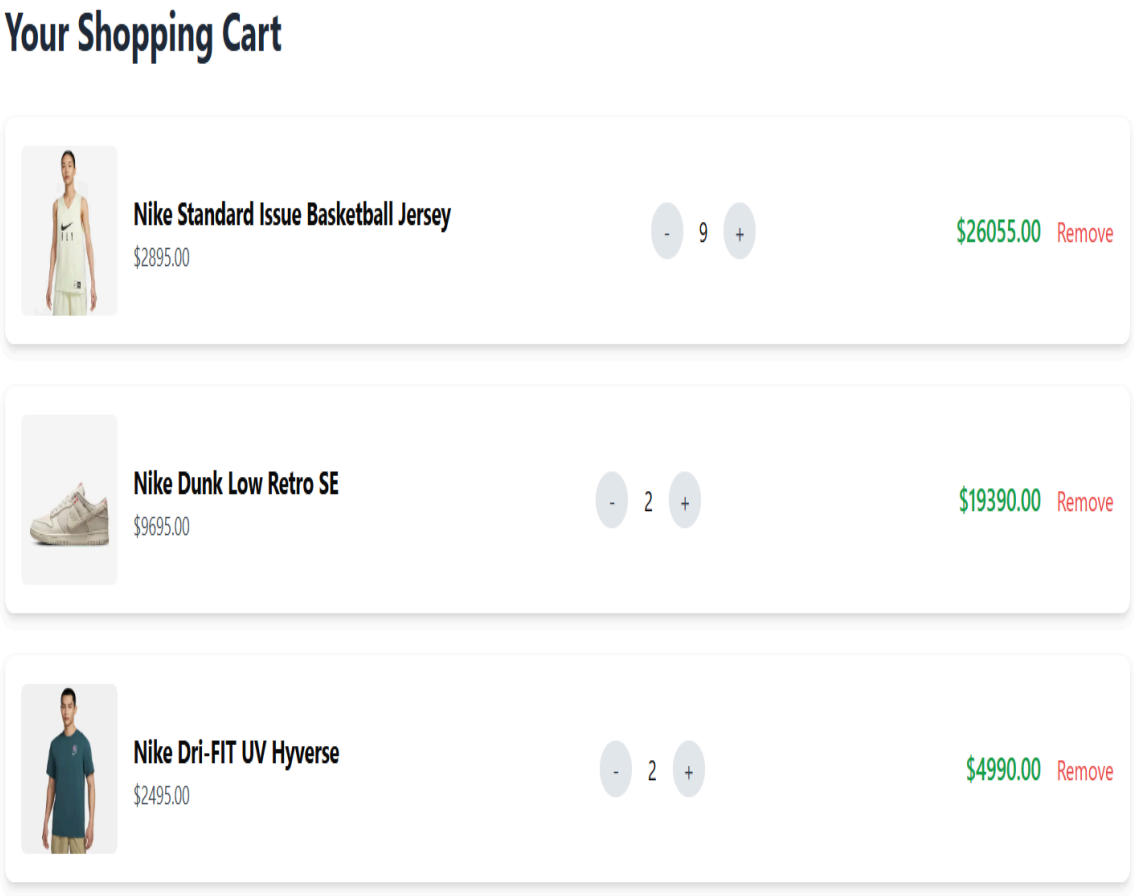


cart") and a green checkmark, indicating success.

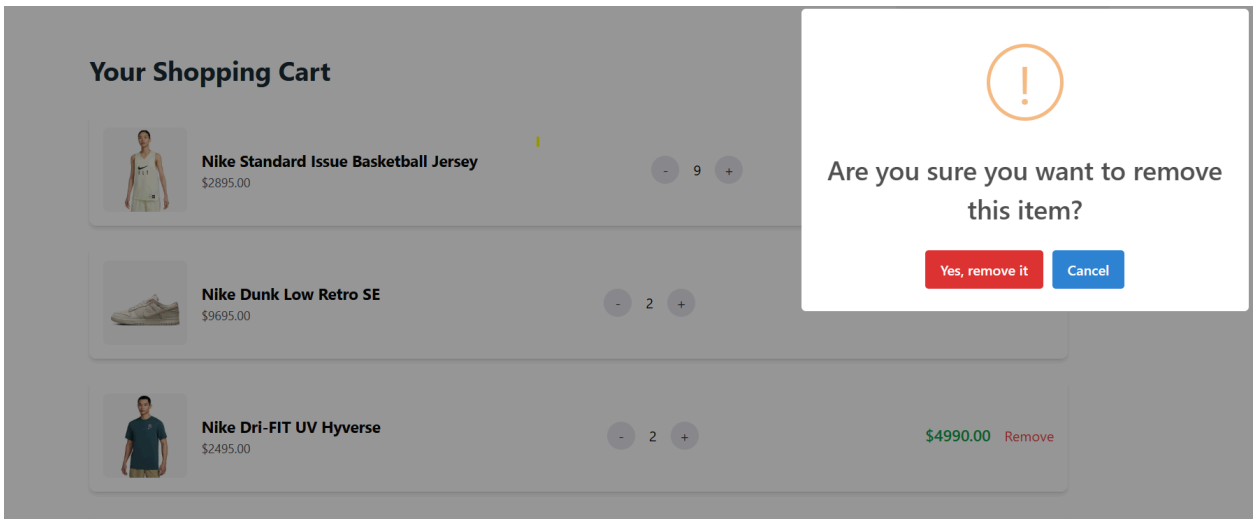
1. Image: A clear product image for easy identification.
2. Name: The product title (e.g., "Nike Court Vision Low Next Nature").
3. Category: A brief description of the product type (e.g., "Men's Shoes").
4. Price: The product price in a clear, prominent format (e.g., "$4995").
5. Tags: Visual labels such as "Just In" or "Sustainable Materials" for product categorization.
6. Add to Cart: A button to add the product to the cart.

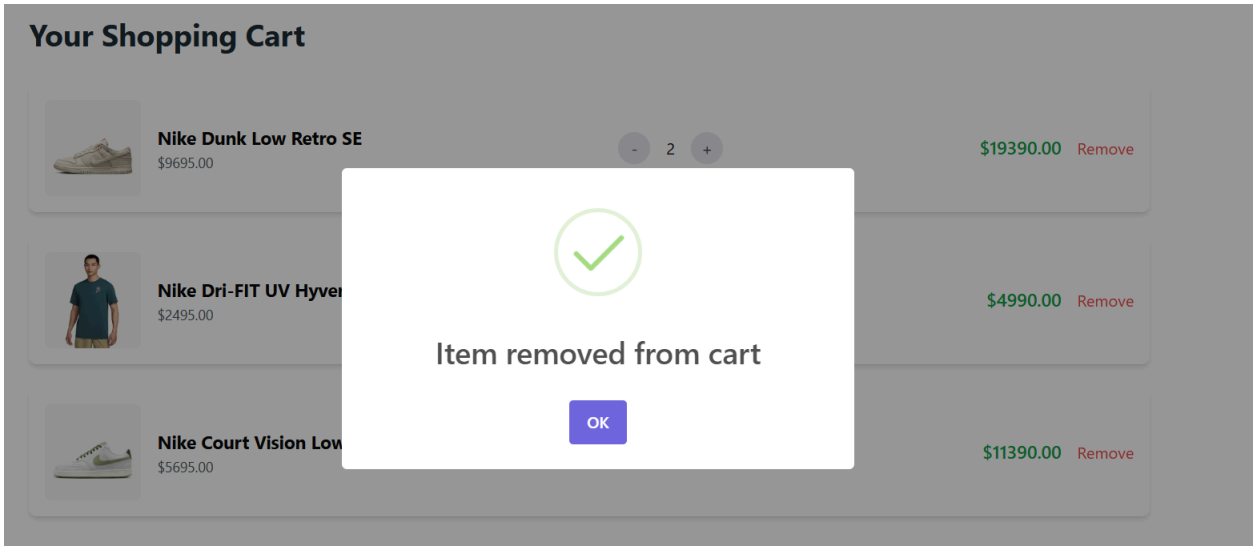

.next
node_modules
public
scripts
src
app
[category]
actions
actions.tsx U
addtocart
components
allproductpage.tsx
bestofall.tsx M
cartpage.tsx
categories.tsx
checkoutpage.tsx
dontmiss.tsx
featured.tsx
Footer.tsx
footerhead.tsx
gearup.tsx
import exp from "constants";
import { Product } from "../../../types/products";
import { get } from "http";
export const addToCart = (product:Product) =>{
let cart : Product[] = JSON.parse(localStorage.getItem('cart') || '[]');
const existingProductIndex = cart.findIndex(item => item._id === product._id);
if(existingProductIndex > -1){
cart[existingProductIndex].inventory += 1;
} else {
cart.push({...product, inventory: 1});
}
localStorage.setItem('cart', JSON.stringify(cart));
}
export const removeFromCart = (productId: string) =>{
let cart : Product[] = JSON.parse(localStorage.getItem('cart') || '[]');
cart = cart.filter(item => item._id !== productId);
localStorage.setItem('cart', JSON.stringify(cart));

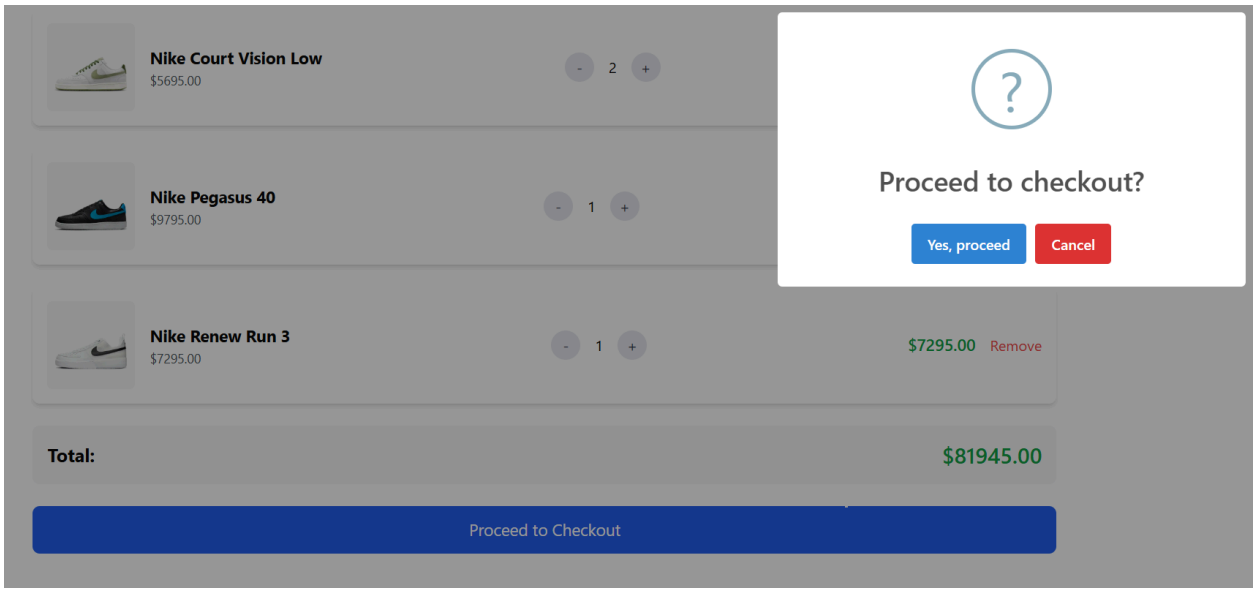
This is a shopping cart UI displaying selected products with their names, prices, quantities, total prices, and options to adjust quantities or remove items

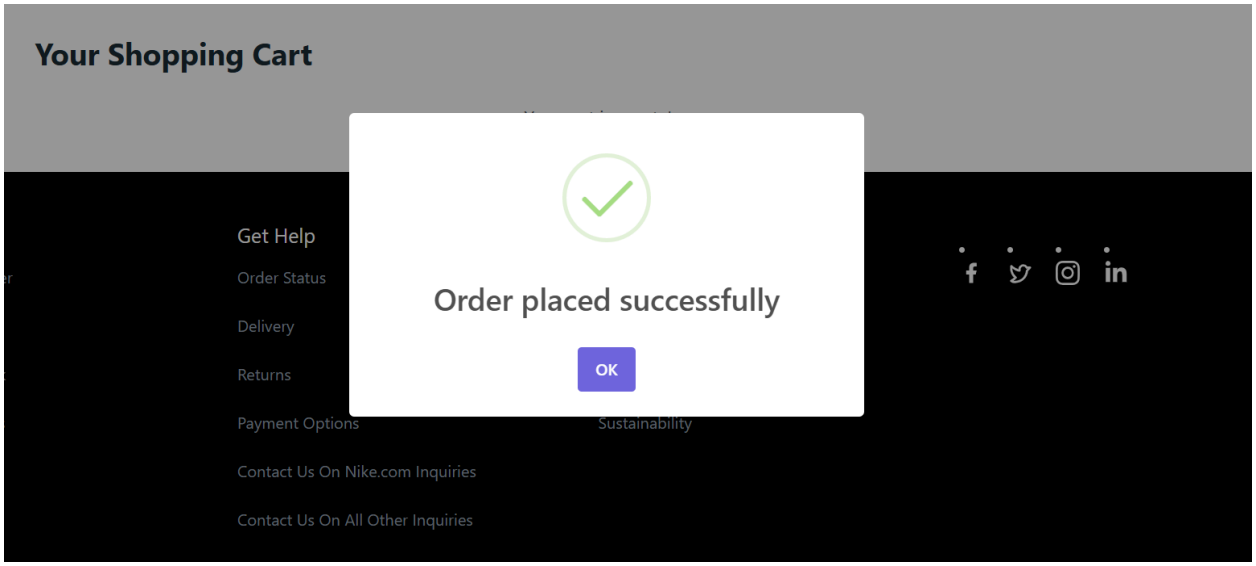


This `CartPage` component displays a shopping cart interface where users can view items, adjust quantities, remove products, and proceed to checkout. It uses React hooks, SweetAlert for confirmations, and Tailwind CSS for styling.

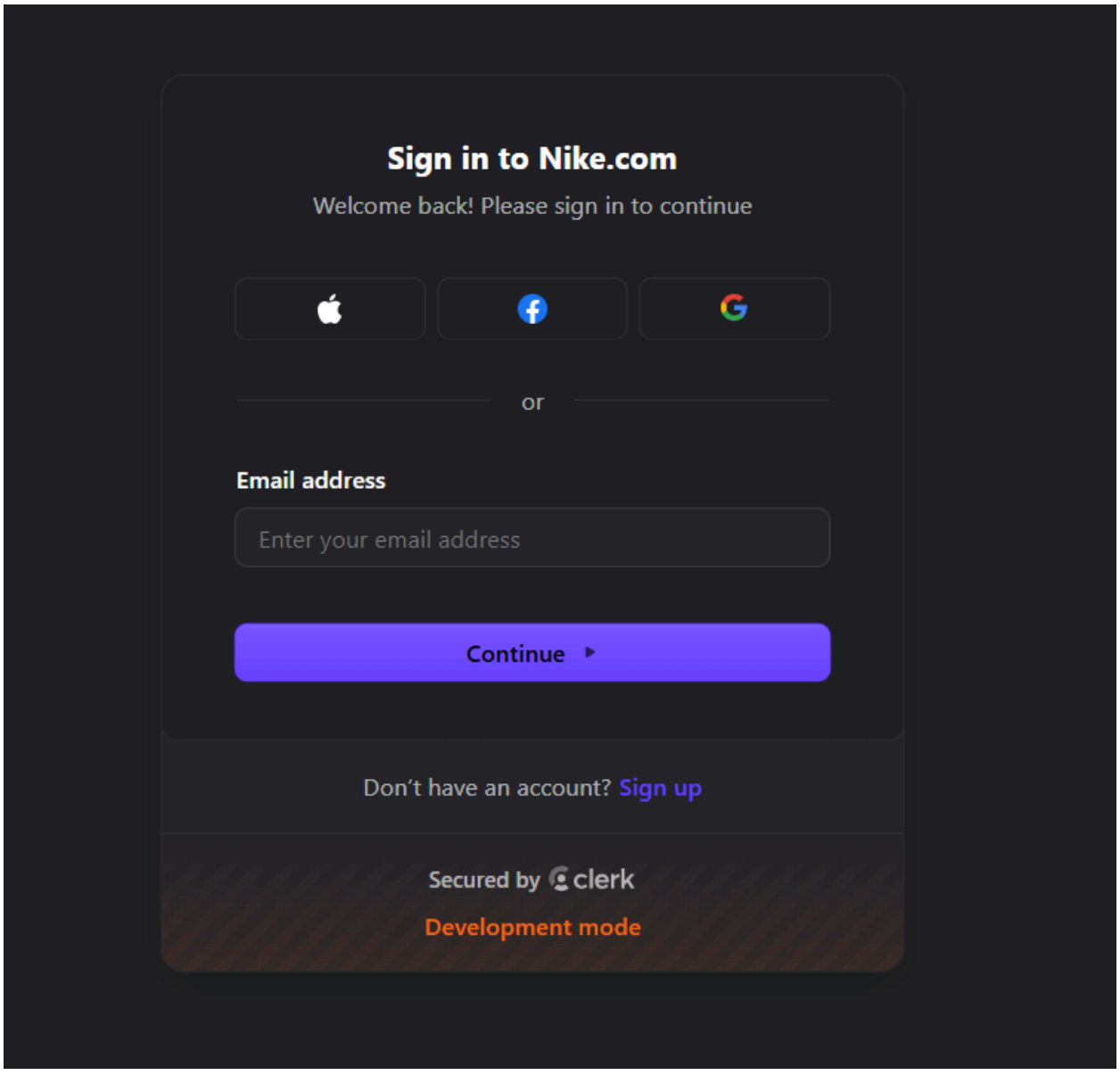



Your Shopping Cart
Nike Dunk Low Retro SE
$9695.00
- 2 +
$19390.00 Remove
Nike Dri-FIT UV Hyver
$2495.00
$4990.00 Remove
Nike Court Vision Low
$5695.00
$11390.00 Remove
Item removed from cart
OK


Nike Court Vision Low
$5695.00
- 2 +
Nike Pegasus 40
$9795.00
- 1 +
Nike Renew Run 3
$7295.00
- 1 +
$7295.00 Remove
Total:
$81945.00
Proceed to Checkout
Proceed to checkout?
Yes, proceed
Cancel


Your Shopping Cart
Order placed successfully
OK
Get Help
Order Status
Delivery
Returns
Payment Options
Contact Us On Nike.com Inquiries
Contact Us On All Other Inquiries
Sustainability

AuthGuard Component for Next.js Authentication



Key Features:

1. Authentication Handling with Clerk

- **Uses `useUser()` from @clerk/nextjs to check if the user is signed in.**
- **If `isSignedIn` is false, the user is redirected to the sign-in page (`/DynamicRoutes/signin`).**
- **If `isSignedIn` is true, the page content is rendered.**

2. Automatic Redirection for Unauthorized Users

- **Uses `useRouter` from Next.js to navigate programmatically.**
- **Calls `router.replace("/DynamicRoutes/signin")` inside `useEffect` to redirect unauthenticated users without adding extra history entries.**

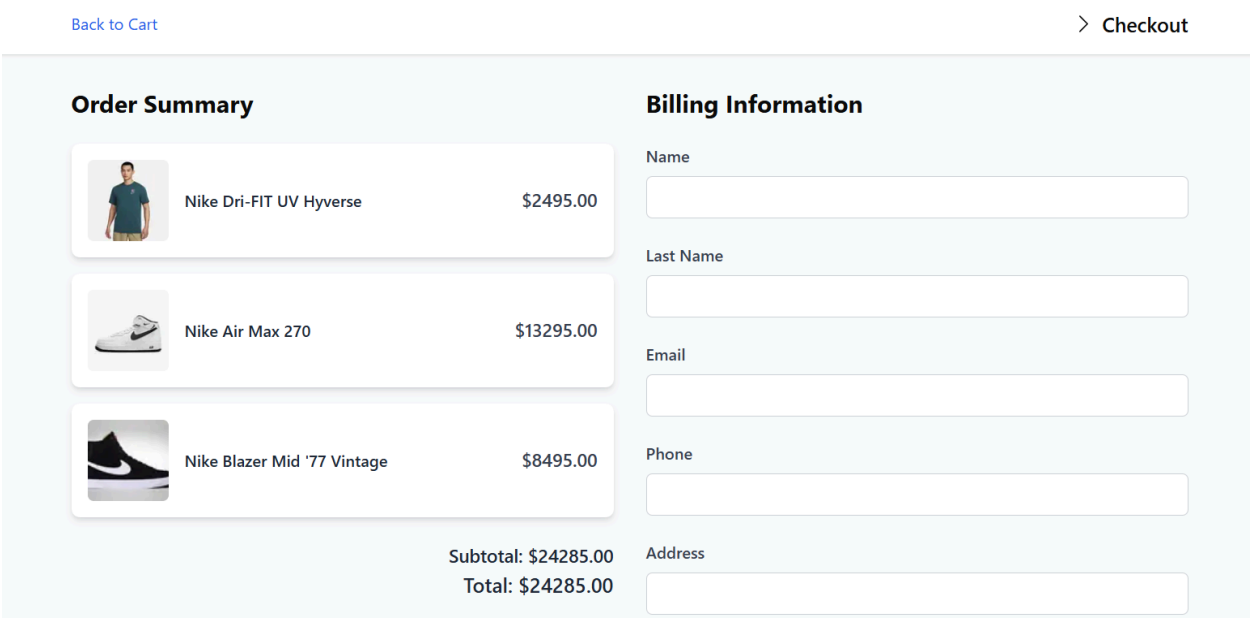
3. Loading State for Smooth UX

- **Uses `useState` (`isLoading`) to prevent content flickering while checking authentication.**
- **Displays a "Loading..." message until authentication status is determined.**

```
const CartPage = () => {
    <AuthGuard >
    <div className="max-w-6xl mx-auto px-4 py-10">
      <h1 className="text-2xl sm:text-3xl font-bold text-gray-800 mb-8 text-center sm:text-left">
        Your Shopping Cart
      </h1>

      {cart.length === 0 ? (
        <p className="text-lg text-gray-600 text-center">Your cart is empty!</p>
      ) : (
        <div className="space-y-6">
          {cart.map((product) => (
            <div
              key={product._id}
              className="flex flex-col sm:flex-row items-center sm:justify-between bg-white shadow-md rounded-lg p-4 gap-4"
            >
              <div className="flex flex-col sm:flex-row items-center gap-4 w-full sm:w-auto">
                {product.image && (
                  <Image
                    src={urlFor(product.image).url()}
                    alt={product.productName}
                    width={500}
                    height={500}
                    className="w-24 h-24 object-cover rounded-md"
                  />
                )}
```

Checkout Page in Next.js with Sanity CMS Integration



Key Features:

1. Cart Management

- **Retrieves cart items from local storage using `getCartItems()`.**
- **Displays product details such as name, image, and price.**

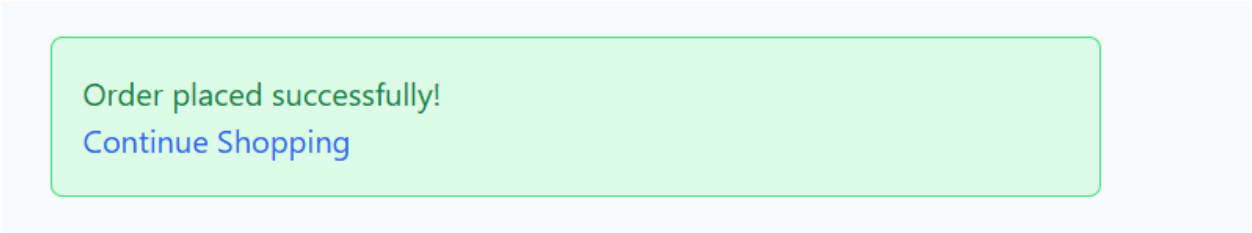
- **Calculates the subtotal, applies any stored discount, and computes the final total price.**

2. Billing Form with Validation

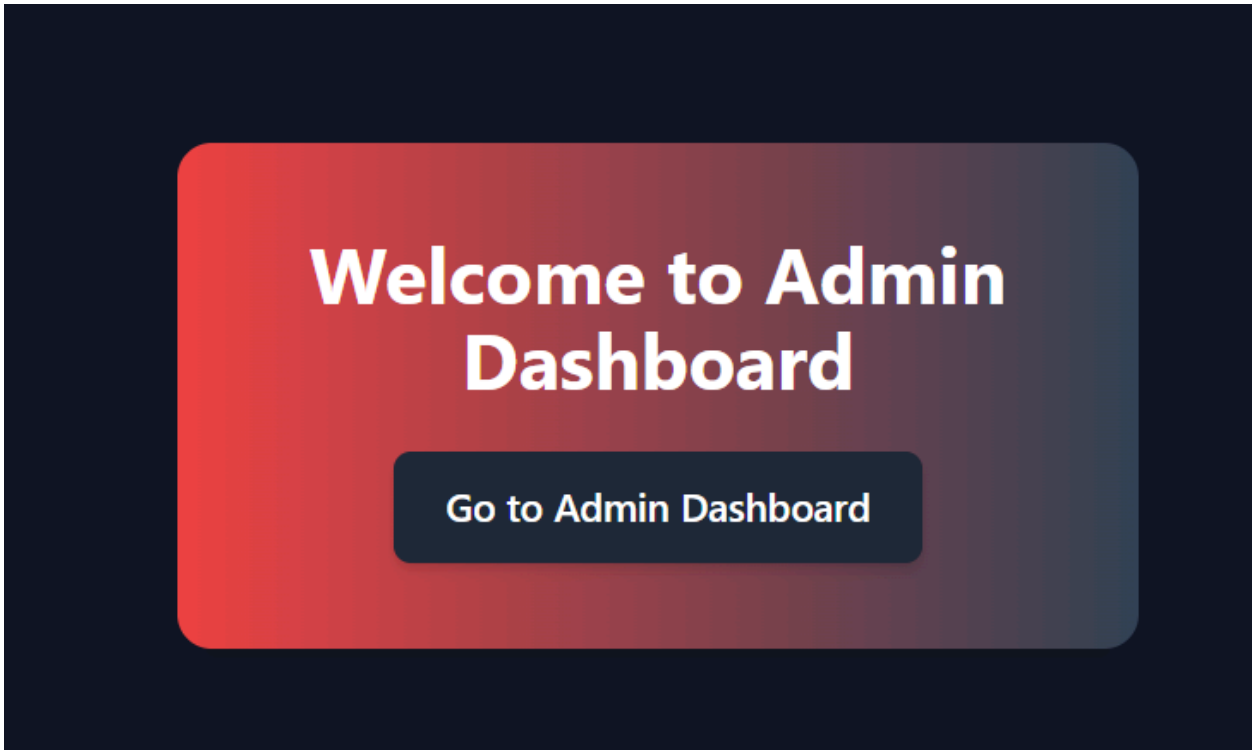
- **The checkout form includes fields for name, email, phone number, address, and zip code.**
- **Uses form state management (`useState`) to store and validate user input.**
- **Implements real-time validation, ensuring inputs are properly formatted (e.g., valid email, zip code, and phone number).**

3. Order Submission to Sanity CMS

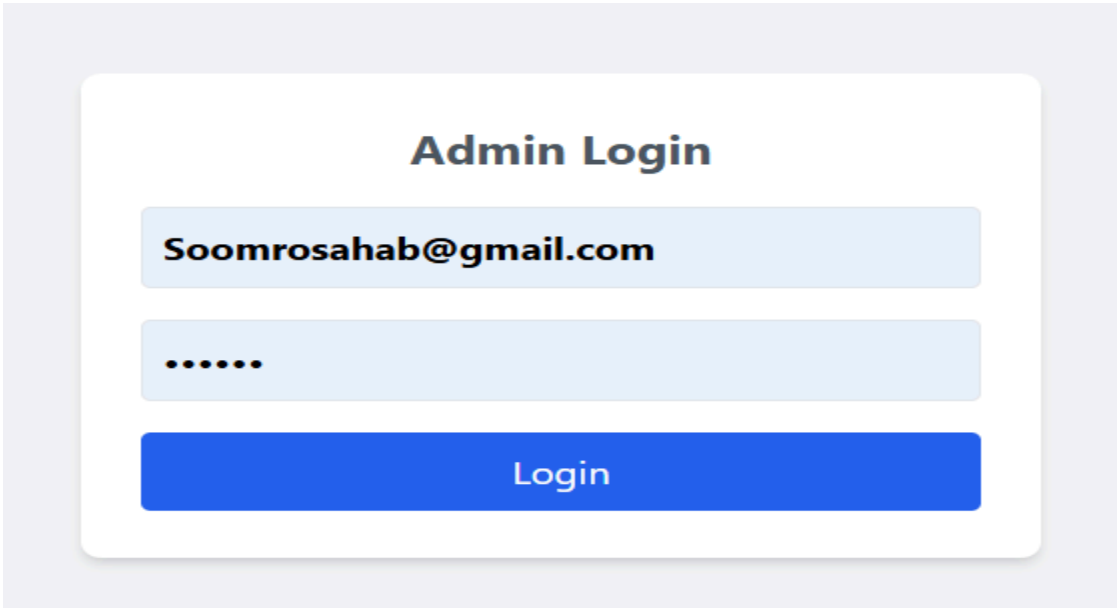
- **When the user clicks "Place Order", the system:**
-
    - **Validates the form.**
    - **Ensures the cart is not empty.**
    - **Sends an order request to Sanity CMS via `client.create(orderData)`.**
- **If successful:**
    - **The order is stored in Sanity CMS as a new entry.**
    - **The form and cart are reset.**
    - **A success message is displayed with a link to continue shopping.**



Admin Dashboard - Order Management System



This Admin Dashboard is a Next.js-based protected route designed to manage customer orders efficiently. It interacts with Sanity.io as the backend database and provides essential functionalities such as order filtering, updating order statuses, and deleting orders.



1️⃣ Login Validation

- **Uses `useState` to manage email and password input fields.**
- **On form submission (`handleLogin` function):**
  - **Checks if the input matches the predefined admin credentials.**
  - **If correct, stores login state in `localStorage` and redirects to the dashboard.**
  - **If incorrect, displays an error message using SweetAlert2.**

2️⃣ Storing Login Status

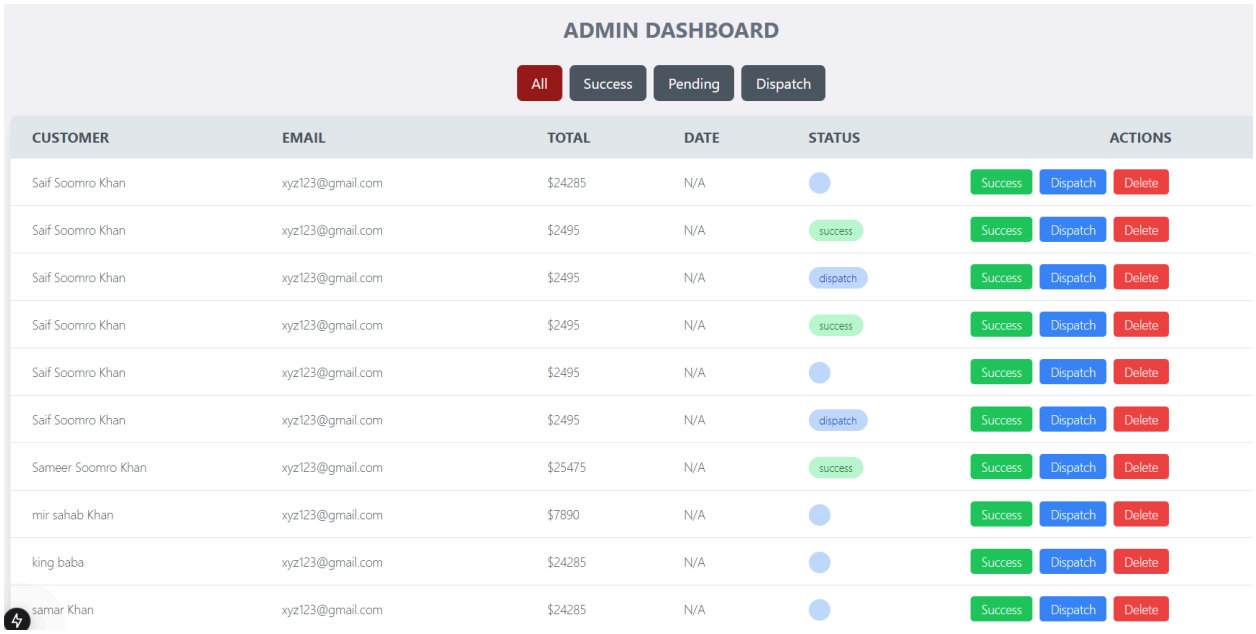
- **Uses `localStorage.setItem('islogedIn', 'true')` to persist admin login.**
- **This can be later used to restrict unauthorized access to admin pages.**

3️⃣ Redirecting to Admin Dashboard

- **Uses Next.js router (`useRouter`) to navigate to `/admin/dashboard` after a successful login.**

4️⃣ User-Friendly UI

- **Form Fields: Email & Password fields with validation.**
- **Button: A blue login button that changes color on hover.**
- **Centered Layout: Uses Flexbox for a visually appealing, centered login**



-

This dashboard allows administrators to view a list of customer orders

View Orders: Display a list of customer orders with details such as customer name, email, total price, and the order date.

Filter Orders: Filter orders by their status (e.g., "All", "Success", "Pending", "Dispatch") for easy management.

Update Order Status: Change the order status to "Success", "Pending", or "Dispatch" directly from the table with a single click. This helps in managing the lifecycle of the order.

Delete Orders: Delete orders with a confirmation prompt to ensure no accidental deletions.

Real-time Alerts: Notifications are provided after updating or deleting an order, using SweetAlert2 for displaying success or error messages.

Front-End Nike E commerce website

News & Featured Men Women Kids Sale SNKRS Search

Hello Nike App

Download the app to access everything Nike.Get Your Great

First Look

Nike Air Max Pulse

Extreme comfort. Hyper durable. Max volume. Introducing the Air Max Pulse—designed to push you past your limits and help you go to the max

Notify Me Shop Air Max

Best Products

Featured

Gear Up

Shop Men's

Nike Dri-FIT ADV TechKnit Ultra $3895

Men's Short-Sleeve Running Top

Add to Cart Buy Now

Nike Dri-FIT Challenger $2495

Men's Short-Sleeve Running Top

Add to Cart Buy Now

Shop Women's

Nike Dri-FIT ADV Run Division $5295

Women's Long-Sleeve Running Top

Add to Cart Buy Now

Nike Fast $3795

Mid-Rise 7/8 Running Leggings

Add to Cart Buy Now

FLIGHT ESSENTIALS

Your built-to-last, all-week wears—but with style only Jordan Brand can deliver.

Shop

The Essentials

• CANDIDATE • ID • DAY • SLOT • TEACHER	SAFDAR ALI 00410893 SUNDAY 2PM TO 5PM SIR - JAWWAD