HACKATHON – 03

DAY 4

SUBMITTED BY

BEENISH RAJPUT

ROLLNO : 490738

UNDER THE SUPERVISION

OF

SIR ANUS SETH

SIR HAMZA

SIRAMEEN ALAM

SIR ZIA KHAN

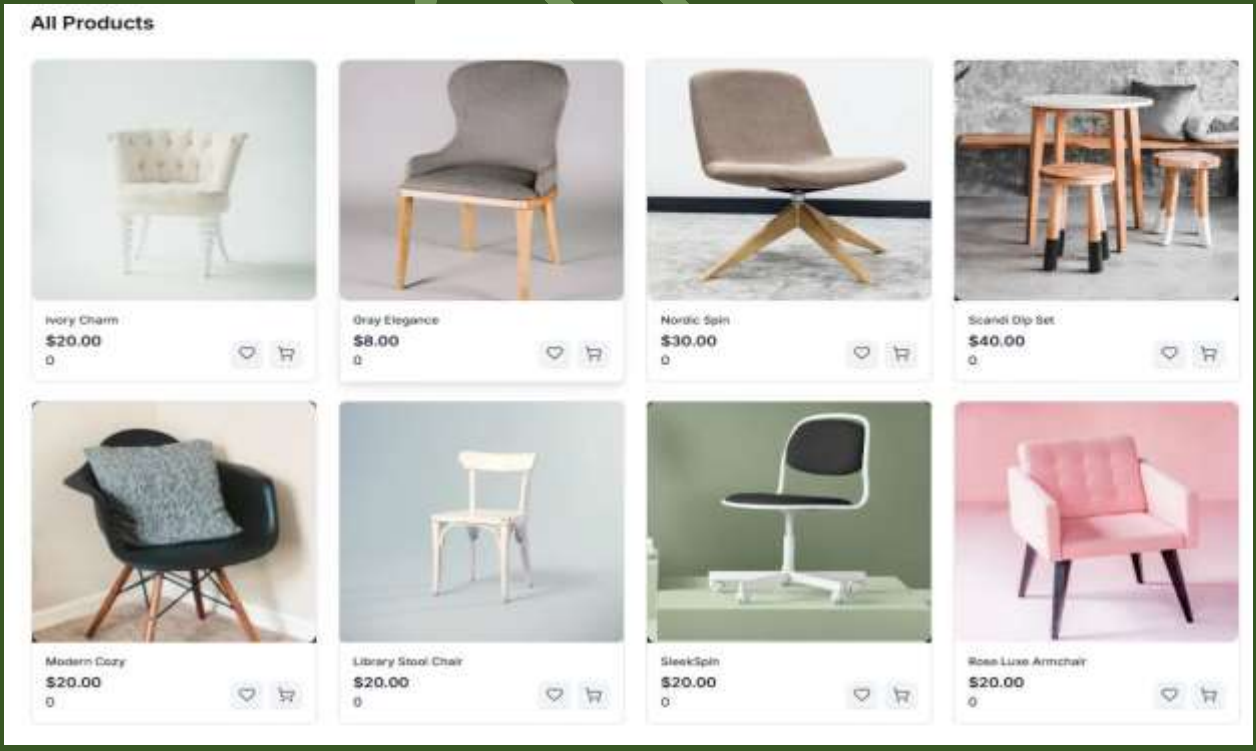
DAY 04: DYNAMIC FRONTEND COMPONENTS – PROJECT:-TEMPLATE 08 {{{COMFORTY}}}

Overview:

In this section, I focused on developing dynamic frontend components for the Comforty-e-commerce website, ensuring the website's user interface is both engaging and highly functional. I used reusable React components to create a flexible, maintainable structure for pages like product listings, product detail views, and the shopping cart. The goal was to provide a seamless and responsive experience for users, allowing them to browse, add products to their cart, and proceed to checkout with ease.

Product Listing with D-Data:-

For the product listing page, I focused on implementing a dynamic approach to fetch and display product data. Instead of hardcoding product details, I integrated the frontend with the backend using API routes to retrieve product information dynamically. This approach allows the product list to automatically update when changes are made in the backend
my all products are:-



All Products

Ivory Charm
$20.00
0

Gray Elegance
$8.00
0

Nordic Spin
$30.00
0

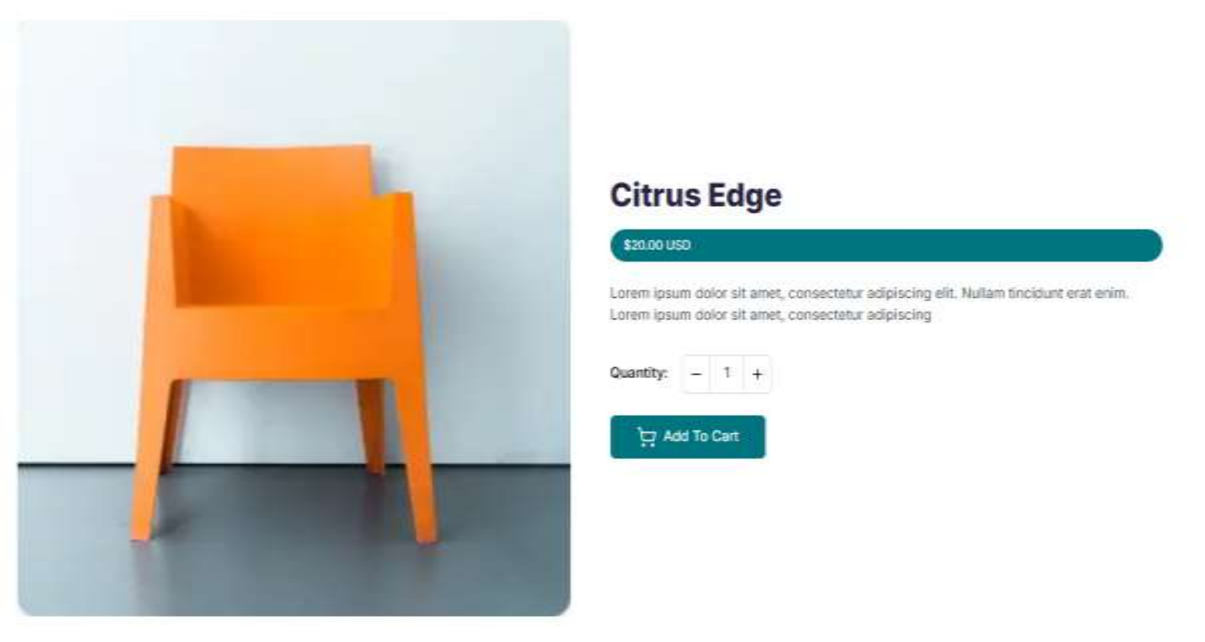
Scandi Dip Set
$40.00
0

Modern Cozy
$20.00
0

Library Stool Chair
$20.00
0

SleekSpin
$20.00
0

Rose Luxe Armchair
$20.00
0

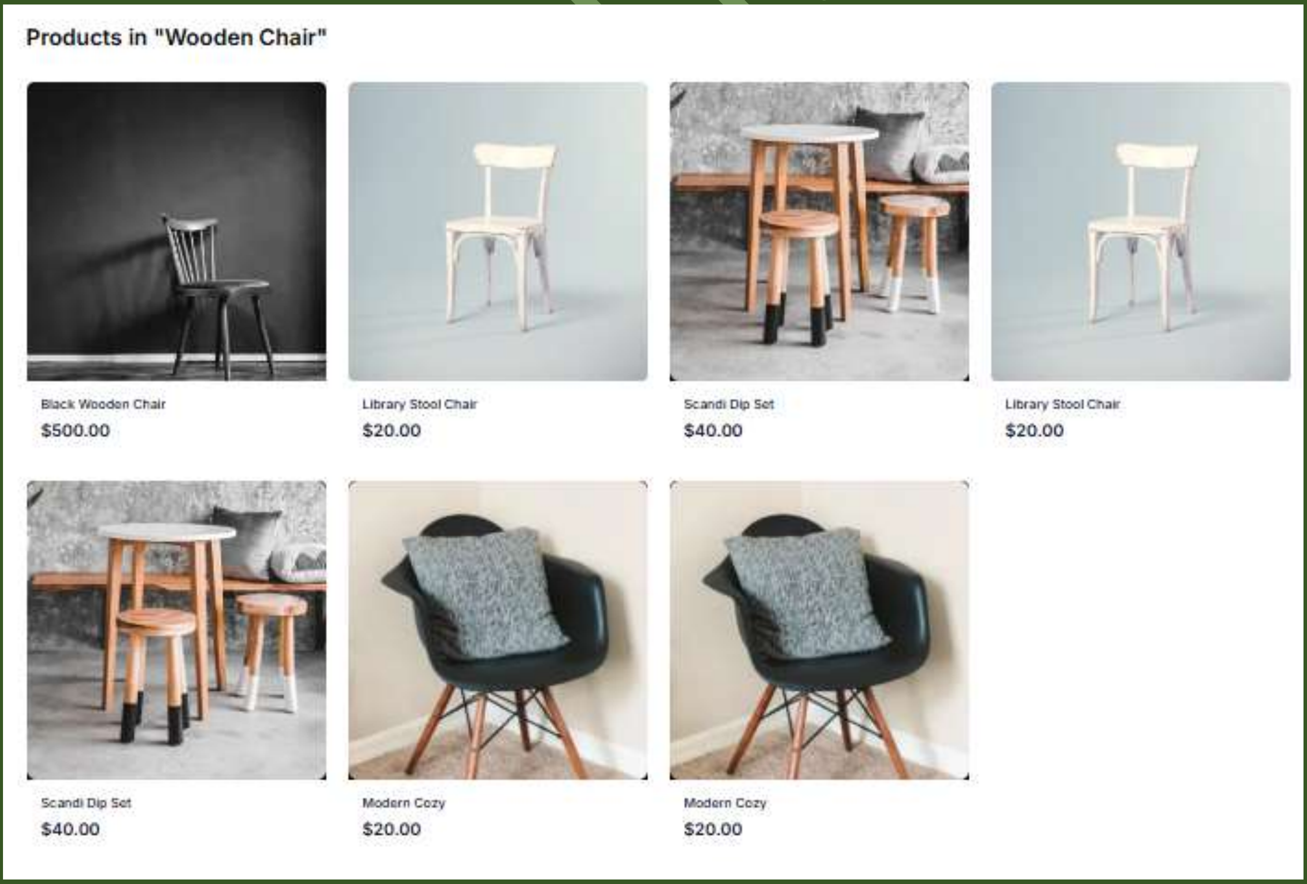


Citrus Edge

$20.00 USD

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Category Filter:



Products in "Wooden Chair"

Black Wooden Chair
$500.00

Library Stool Chair
$20.00

Scandi Dip Set
$40.00

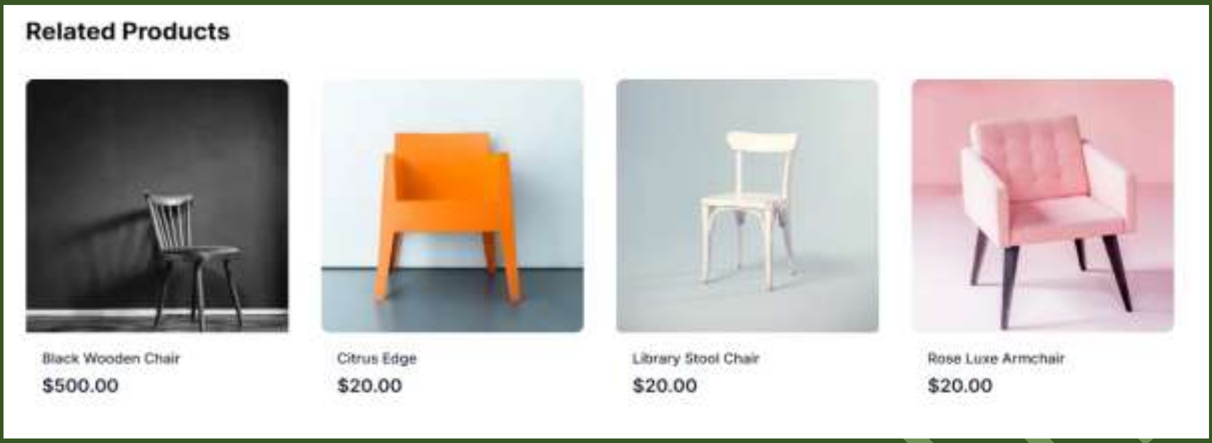
Library Stool Chair
$20.00

Scandi Dip Set
$40.00

Modern Cozy
$20.00

Modern Cozy
$20.00

Related Products:



Related Products

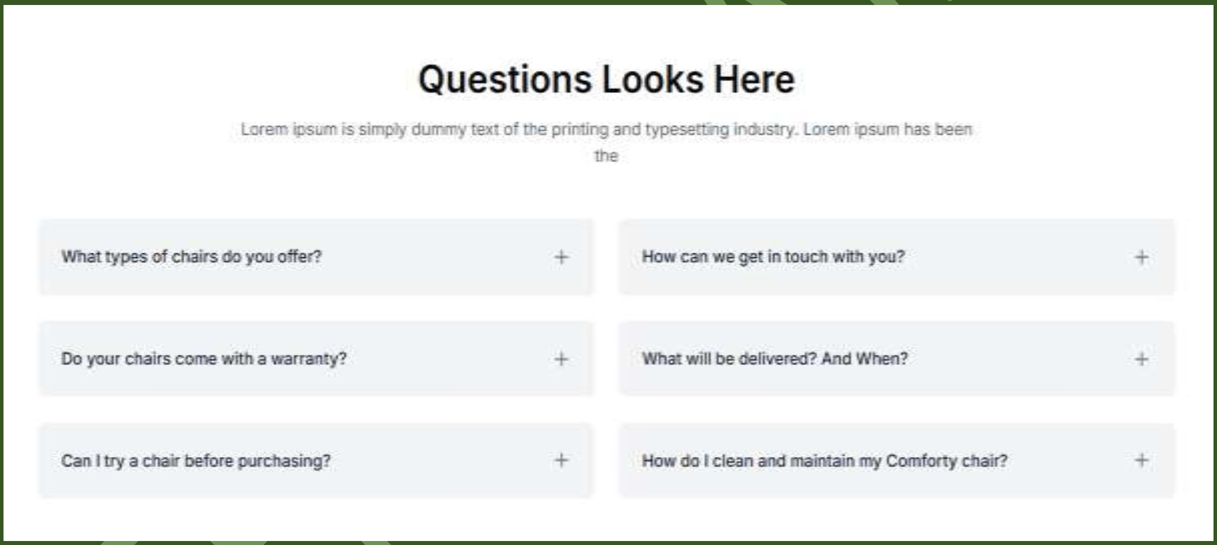
Black Wooden Chair
$500.00

Citrus Edge
$20.00

Library Stool Chair
$20.00

Rose Luxe Armchair
$20.00

FAQS:



Questions Looks Here

Lorem ipsum is simply dummy text of the printing and typesetting industry. Lorem ipsum has been the

What types of chairs do you offer? +

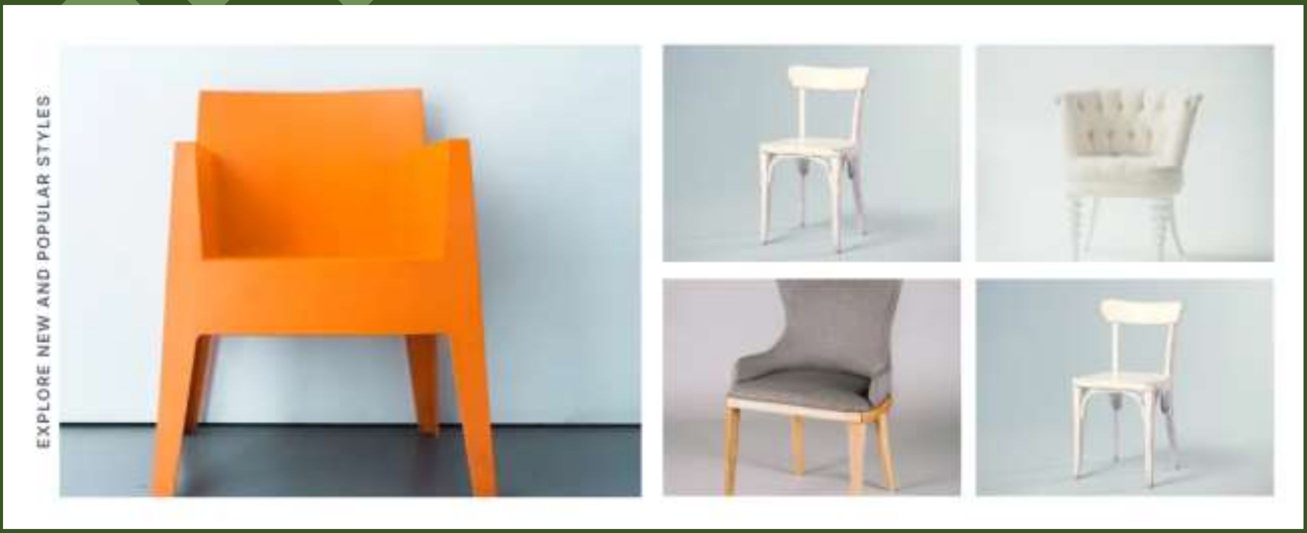
How can we get in touch with you? +

Do your chairs come with a warranty? +

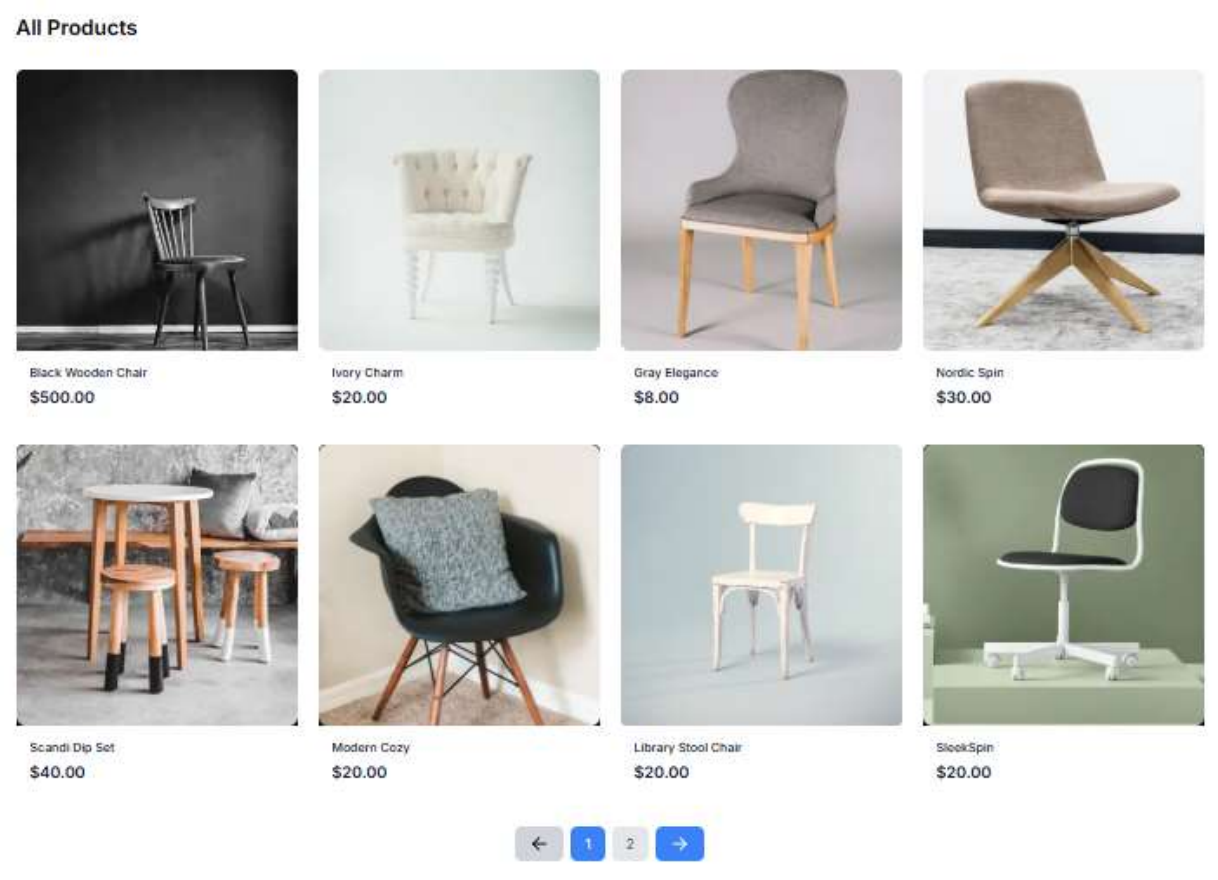
What will be delivered? And When? +

Can I try a chair before purchasing? +

How do I clean and maintain my Comforty chair? +



Pagination:



All Products

Black Wooden Chair
$500.00

Ivory Charm
$20.00

Gray Elegance
$8.00

Nordic Spin
$30.00

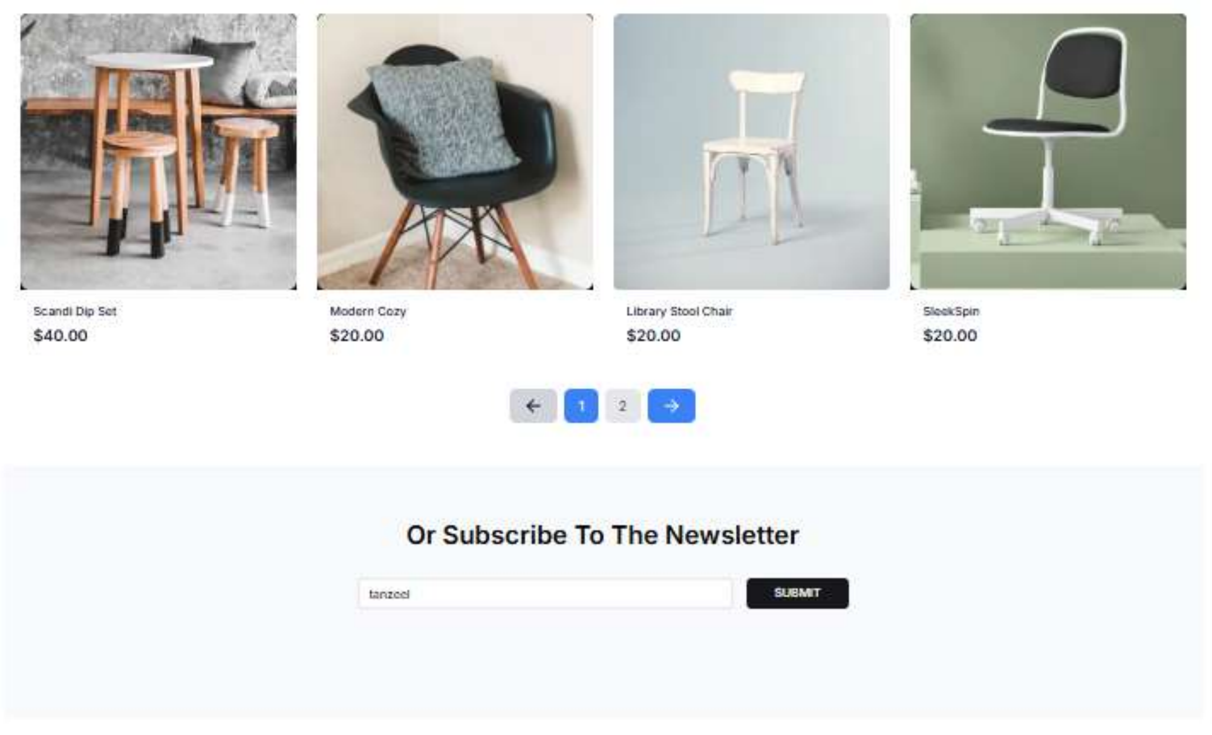
Scandi Dip Set
$40.00

Modern Cozy
$20.00

Library Stool Chair
$20.00

SleekSpin
$20.00

NewsLetter Section:



Scandi Dip Set
$40.00

Modern Cozy
$20.00

Library Stool Chair
$20.00

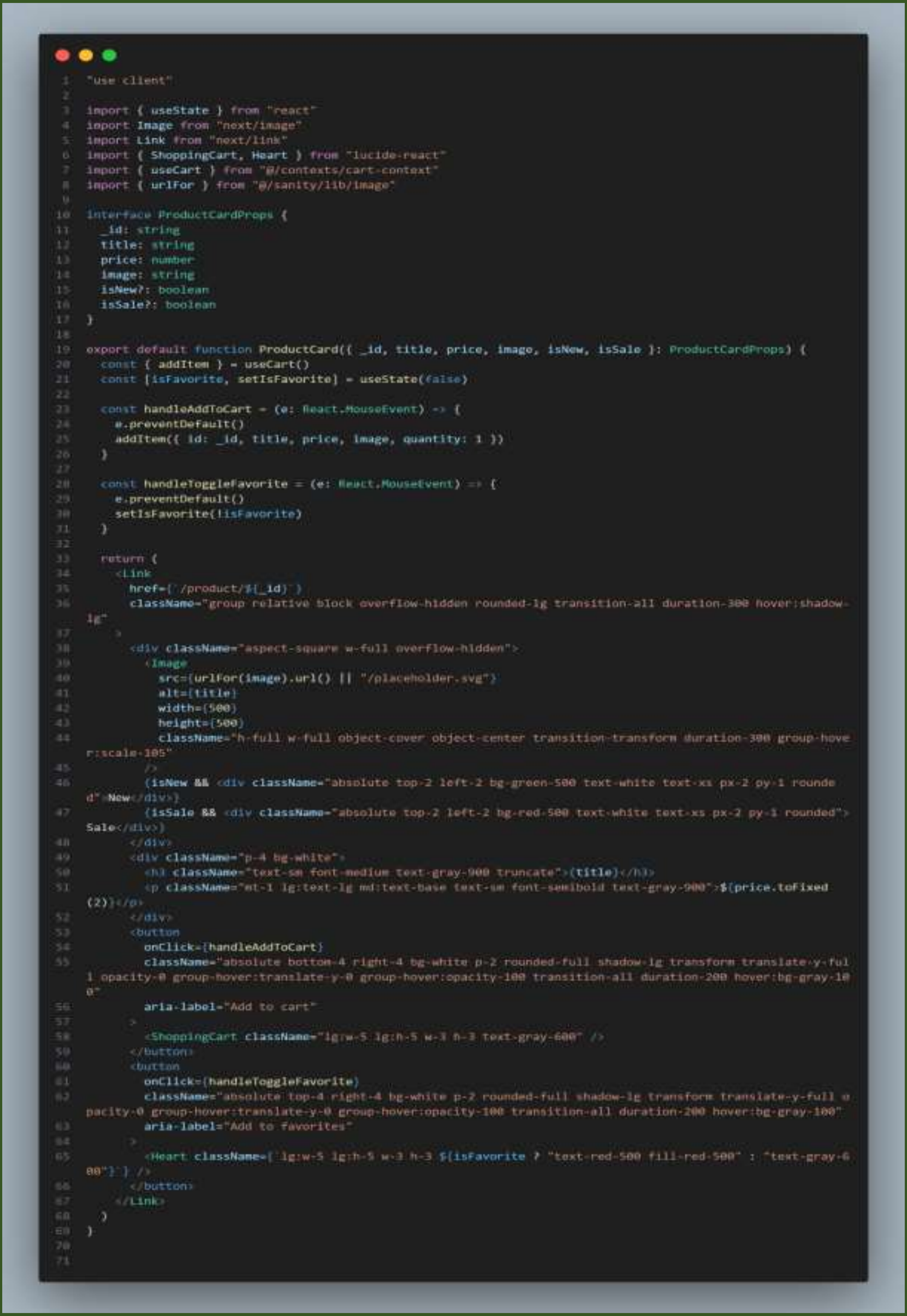
SleekSpin
$20.00

Or Subscribe To The Newsletter

tanzeel

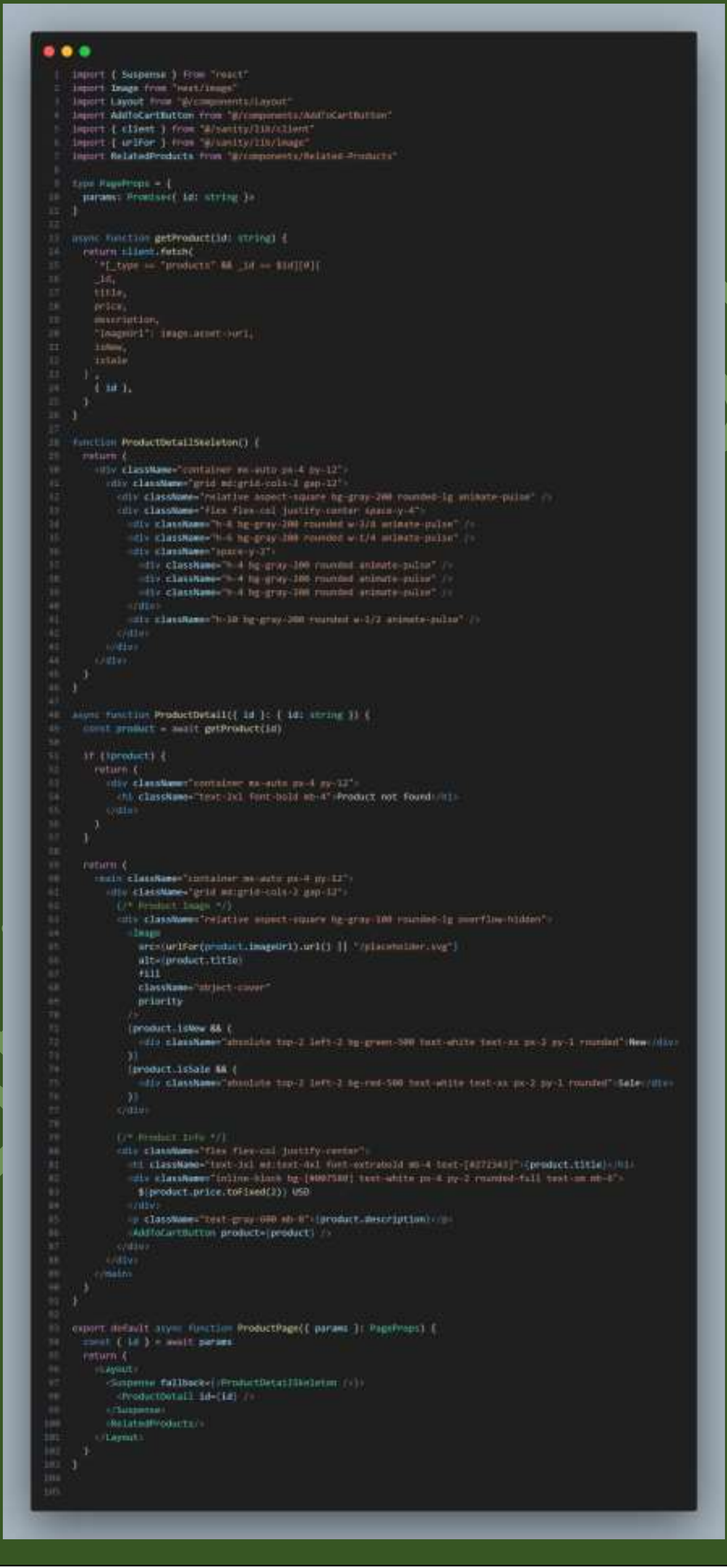
1. MY PRODUCT CARD CODE:

Products:



```tsx
"use client"

import { useState } from "react"
import Image from "next/image"
import Link from "next/link"
import { ShoppingCart, Heart } from "lucide-react"
import { useCart } from "@/contexts/cart-context"
import { urlFor } from "@/sanity/lib/image"

interface ProductCardProps {
  _id: string
  title: string
  price: number
  image: string
  isNew?: boolean
  isSale?: boolean
}

export default function ProductCard({ _id, title, price, image, isNew, isSale }: ProductCardProps) {
  const { addItem } = useCart()
  const [isFavorite, setIsFavorite] = useState(false)

  const handleAddToCart = (e: React.MouseEvent) => {
    e.preventDefault()
    addItem({ id: _id, title, price, image, quantity: 1 })
  }

  const handleToggleFavorite = (e: React.MouseEvent) => {
    e.preventDefault()
    setIsFavorite(!isFavorite)
  }

  return (
    <Link
      href={`/product/${_id}`}
      className="group relative block overflow-hidden rounded-lg transition-all duration-300 hover:shadow-lg"
    >
      <div className="aspect-square w-full overflow-hidden">
        <Image
          src={urlFor(image).url() || "/placeholder.svg"}
          alt={title}
          width={500}
          height={500}
          className="h-full w-full object-cover object-center transition-transform duration-300 group-hover:scale-105"
        />
        {isNew && <div className="absolute top-2 left-2 bg-green-500 text-white text-xs px-2 py-1 rounded">New</div>}
        {isSale && <div className="absolute top-2 left-2 bg-red-500 text-white text-xs px-2 py-1 rounded">Sale</div>}
      </div>
      <div className="p-4 bg-white">
        <h3 className="text-sm font-medium text-gray-900 truncate">{title}</h3>
        <p className="mt-1 lg:text-lg md:text-base text-sm font-semibold text-gray-900">${price.toFixed(2)}</p>
      </div>
      <button
        onClick={handleAddToCart}
        className="absolute bottom-4 right-4 bg-white p-2 rounded-full shadow-lg transform translate-y-full opacity-0 group-hover:translate-y-0 group-hover:opacity-100 transition-all duration-200 hover:bg-gray-100"
        aria-label="Add to cart"
      >
        <ShoppingCart className="lg:w-5 lg:h-5 w-3 h-3 text-gray-600" />
      </button>
      <button
        onClick={handleToggleFavorite}
        className="absolute top-4 right-4 bg-white p-2 rounded-full shadow-lg transform translate-y-full opacity-0 group-hover:translate-y-0 group-hover:opacity-100 transition-all duration-200 hover:bg-gray-100"
        aria-label="Add to favorites"
      >
        <Heart className={`lg:w-5 lg:h-5 w-3 h-3 ${isFavorite ? "text-red-500 fill-red-500" : "text-gray-600"}`} />
      </button>
    </Link>
  )
}
```

PRODUCT DETAILING CODE :



```
import { Suspense } from "react"
import Image from "next/image"
import Layout from "@/components/Layout"
import AddToCartButton from "@/components/AddToCartButton"
import { client } from "@/sanity/lib/client"
import { urlFor } from "@/sanity/lib/image"
import RelatedProducts from "@/components/Related-Products"

type PageProps = {
  params: Promise<{ id: string }>
}

async function getProduct(id: string) {
  return client.fetch(
    `*[_type == "products" && _id == $id][0]{
      _id,
      title,
      price,
      description,
      "imageUrl": image.asset->url,
      isNew,
      isSale
    }`,
      { id },
  )
}

function ProductDetailSkeleton() {
  return (
    <div className="container mx-auto px-4 py-12">
      <div className="grid md:grid-cols-2 gap-12">
        <div className="relative aspect-square bg-gray-200 rounded-lg animate-pulse" />
        <div className="flex flex-col justify-center space-y-4">
          <div className="h-8 bg-gray-200 rounded w-3/4 animate-pulse" />
          <div className="h-6 bg-gray-200 rounded w-1/4 animate-pulse" />
          <div className="space-y-2">
            <div className="h-4 bg-gray-200 rounded animate-pulse" />
            <div className="h-4 bg-gray-200 rounded animate-pulse" />
            <div className="h-4 bg-gray-200 rounded animate-pulse" />
          </div>
          <div className="h-10 bg-gray-200 rounded w-1/2 animate-pulse" />
        </div>
      </div>
    </div>
  )
}

async function ProductDetail({ id }: { id: string }) {
  const product = await getProduct(id)

  if (!product) {
    return (
      <div className="container mx-auto px-4 py-12">
        <h1 className="text-2xl font-bold mb-4">Product not found</h1>
      </div>
    )
  }

  return (
    <main className="container mx-auto px-4 py-12">
      <div className="grid md:grid-cols-2 gap-12">
        {/* Product Image */}
        <div className="relative aspect-square bg-gray-100 rounded-lg overflow-hidden">
          <Image
            src={urlFor(product.imageUrl).url() || "/placeholder.svg"}
            alt={product.title}
            fill
            className="object-cover"
            priority
          />
          {product.isNew && (
            <div className="absolute top-2 left-2 bg-green-500 text-white text-xs px-2 py-1 rounded">New</div>
          )}
          {product.isSale && (
            <div className="absolute top-2 left-2 bg-red-500 text-white text-xs px-2 py-1 rounded">Sale</div>
          )}
        </div>

        {/* Product Info */}
        <div className="flex flex-col justify-center">
          <h1 className="text-3xl md:text-4xl font-extrabold mb-4 text-[#272343]">{product.title}</h1>
          <div className="inline-block bg-[#007580] text-white px-4 py-2 rounded-full text-sm mb-6">
            ${product.price.toFixed(2)} USD
          </div>
          <p className="text-gray-600 mb-6">{product.description}</p>
          <AddToCartButton product={product} />
        </div>
      </div>
    </main>
  )
}

export default async function ProductPage({ params }: PageProps) {
  const { id } = await params
  return (
    <Layout>
      <Suspense fallback={<ProductDetailSkeleton />}>
        <ProductDetail id={id} />
      </Suspense>
      <RelatedProducts/>
    </Layout>
  )
}
```

MY PRODUCTS

BM

" APPROACH FOR BUILDING AND INTEGRATING COMPONENTS "

STEP 1: Set Up Project Structure

- I set up the Next.js project with an organized folder structure for components, pages, and API routes.

STEP 2: Create and Structure Components

- I focused on creating reusable components such as Product Card, Product List, Header, Footer, etc., to maintain clean and modular code.

STEP 3: Integrate Dynamic Product Data

- I used API routes in Next.js to fetch product data dynamically from the backend and render it on the Product Listing page.

STEP 4: Implement Pagination for Product Listing

- I implemented pagination to handle large numbers of products, enhancing user experience and performance.

STEP 5: Create Product Detail Page

- I built a Product Detail page to display detailed information about each product fetched from the backend.

STEP 6: Integrate Add to Cart Functionality

- I implemented an "Add to Cart" button in the Product Card component, enabling users to add products to their shopping cart.

STEP 7: Set Up State Management for Cart

- I used React's `useState` and `useContext` to manage the shopping cart state globally across pages.

STEP 8: Design Responsive Layout

- I ensured that the layout for the Product Listing, Product Detail, and Cart pages was responsive and mobile-friendly using CSS Modules.

STEP 9: Implement Checkout Page

- I built the Checkout page where users can review their cart, input shipping details, and proceed to payment.

STEP 10: Connect with Payment Gateway

- I integrated Stripe’s payment gateway into the Checkout page to handle user payments securely.

STEP 11: Set Up Authentication

- I implemented user authentication using Clerk, ensuring that only signed-in users can proceed with the checkout process.

STEP 12: Test Components and Pages

- I thoroughly tested each component and page to ensure functionality, responsiveness, and correct integration of dynamic data.

OBSTACLES ENCOUNTERED AND THEIR RESOLUTIONS

1. Handling Dynamic Product Data:
I faced the challenge of fetching and displaying dynamic product data across the website. To solve this, I used API routes in Next.js to retrieve the product details from the backend, ensuring that the data displayed was always up to date.

2. Pagination for Large Product Lists:
Displaying a large number of products on a single page created performance issues. I solved this by implementing pagination, which allowed me to break down the product listing into smaller chunks and improved load times for users.

3. Dynamic Product Detail Pages:
When users click on a product, I needed to ensure that detailed product information was loaded dynamically. To address this, I set up dynamic routes in Next.js to display the relevant product data based on the unique product ID, allowing each product page to display specific details.

4. Cart State Management:
Managing the cart across multiple components was a challenge. I used the React Context API to create a global state for the cart, ensuring that product data could be accessed and updated across the entire website without issues.

5. User Authentication for Secure Checkout:
Ensuring secure authentication for the checkout process was critical. I integrated Clerk, a third-party authentication service, to securely handle user sign-ins and sign-ups, providing a safe and smooth checkout experience.

6. Responsive Design Issues:
To ensure the website was mobile-friendly, I used CSS Modules in combination with media queries. This allowed me to create a responsive design that adjusted to different screen sizes, improving the user experience on mobile devices.

7. Payment Gateway Integration:
Integrating a secure payment gateway was crucial. I followed Stripe's official documentation to securely process payment information, ensuring users could safely complete transactions.

8. Simplifying the Checkout Process:
The checkout process needed to be user-friendly and straightforward. I focused on creating a simple, step-by-step process with validation at each step to prevent errors and improve the user experience.

DEVELOPMENT BEST PRACTICES AND METHODOLOGIES

During this Hackathon I faced so many challenges but I believe I focused on creating reusable components and maintaining a clean, modular code structure. I integrated APIs with proper error handling and validation, used React Context for state management, and ensured responsive, mobile-first design. Security was prioritized through Clerk for user authentication, and performance was optimized using lazy loading and asset compression. I conducted regular code reviews, ensured accessibility, and adopted continuous testing to catch bugs early. This approach ensured a high-quality, maintainable, and user-friendly website.

SPECIAL MESSAGE :

During this hackathon, I faced many difficulties, but I believe that to achieve something, one has to sacrifice something. After many errors and sacrificing several nights of sleep, my hard work finally paid off, Alhamdulillah. I will always be grateful to my teachers, especially :-

{Sir Anus Seth, Sir Hamza ,& Sir Ameen Alam}
Thank you all very much.

2.