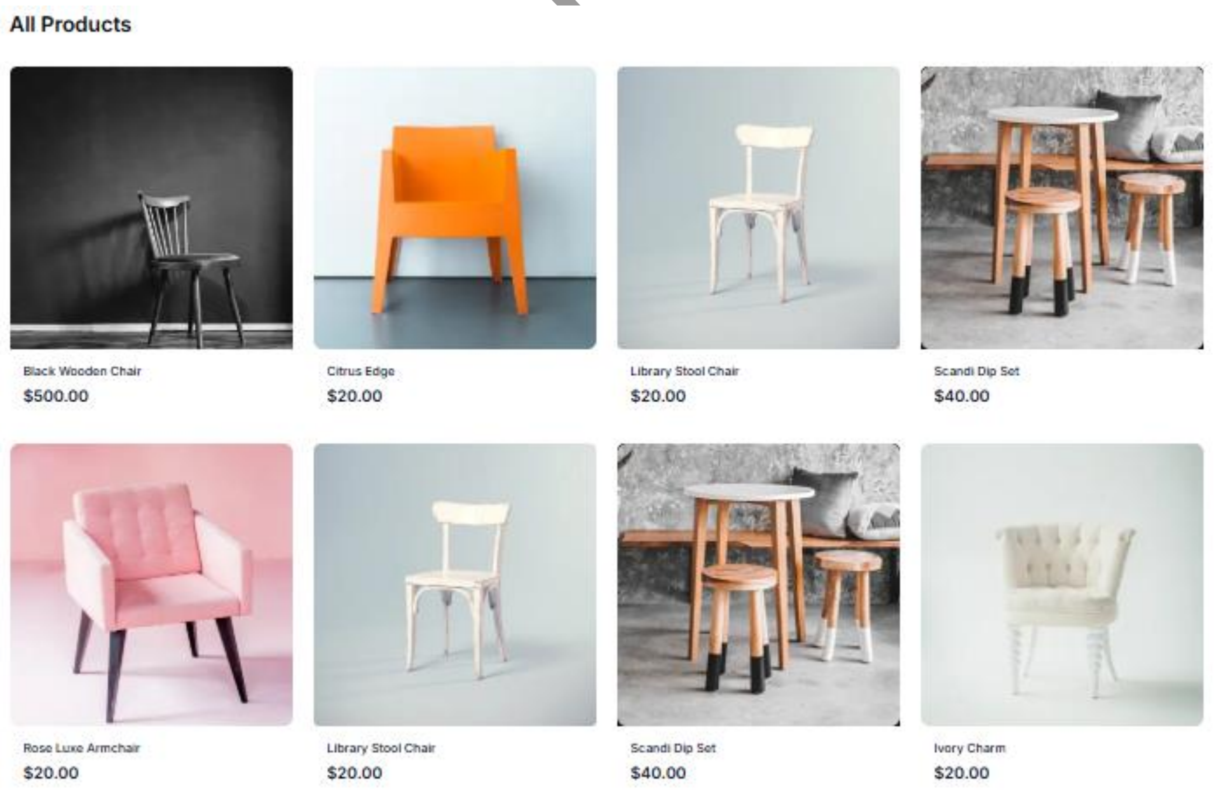
Day 4 - Dynamic Frontend Components – Comforty

Overview:

Comforty is a fully functional e-commerce platform focused on selling chairs, developed using Next.js, Sanity, and Tailwind CSS. It offers a seamless shopping experience with features like a product catalog, detailed product pages, shopping cart, and secure checkout. The platform is responsive, ensuring smooth performance across devices. With an intuitive interface and fast load times, Comforty delivers a modern, user-friendly solution for purchasing chairs online.

1. Functional Deliverables:

- Product Listing with Dynamic Data:



All Products

Black Wooden Chair
$500.00

Citrus Edge
$20.00

Library Stool Chair
$20.00

Scandi Dip Set
$40.00

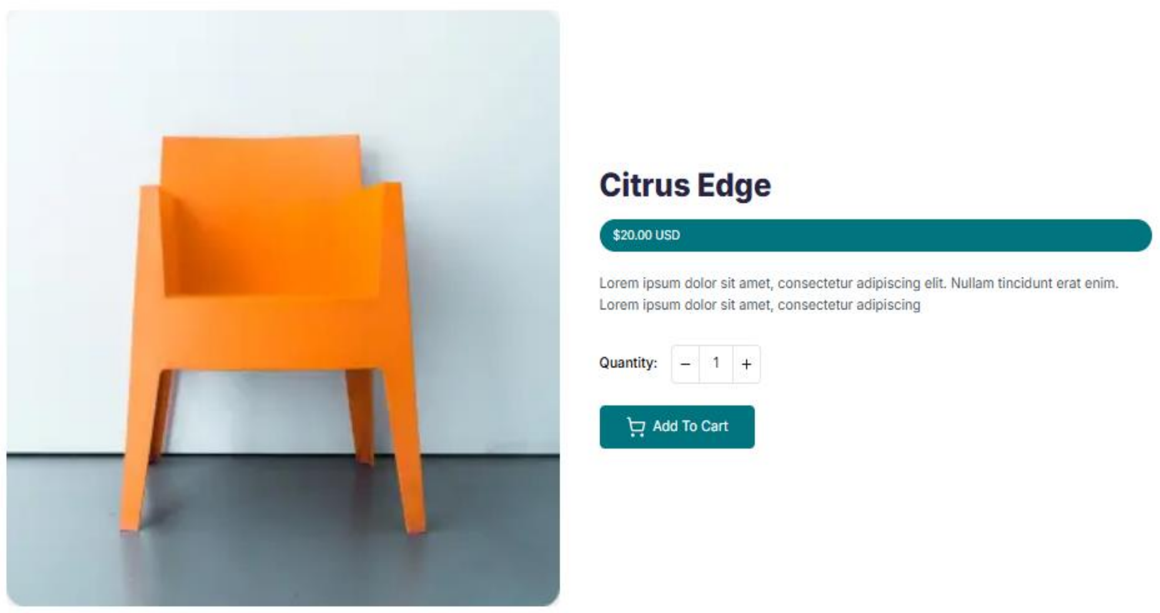
Rose Luxe Armchair
$20.00

Library Stool Chair
$20.00

Scandi Dip Set
$40.00

Ivory Charm
$20.00

- Product Detail Page:



Citrus Edge

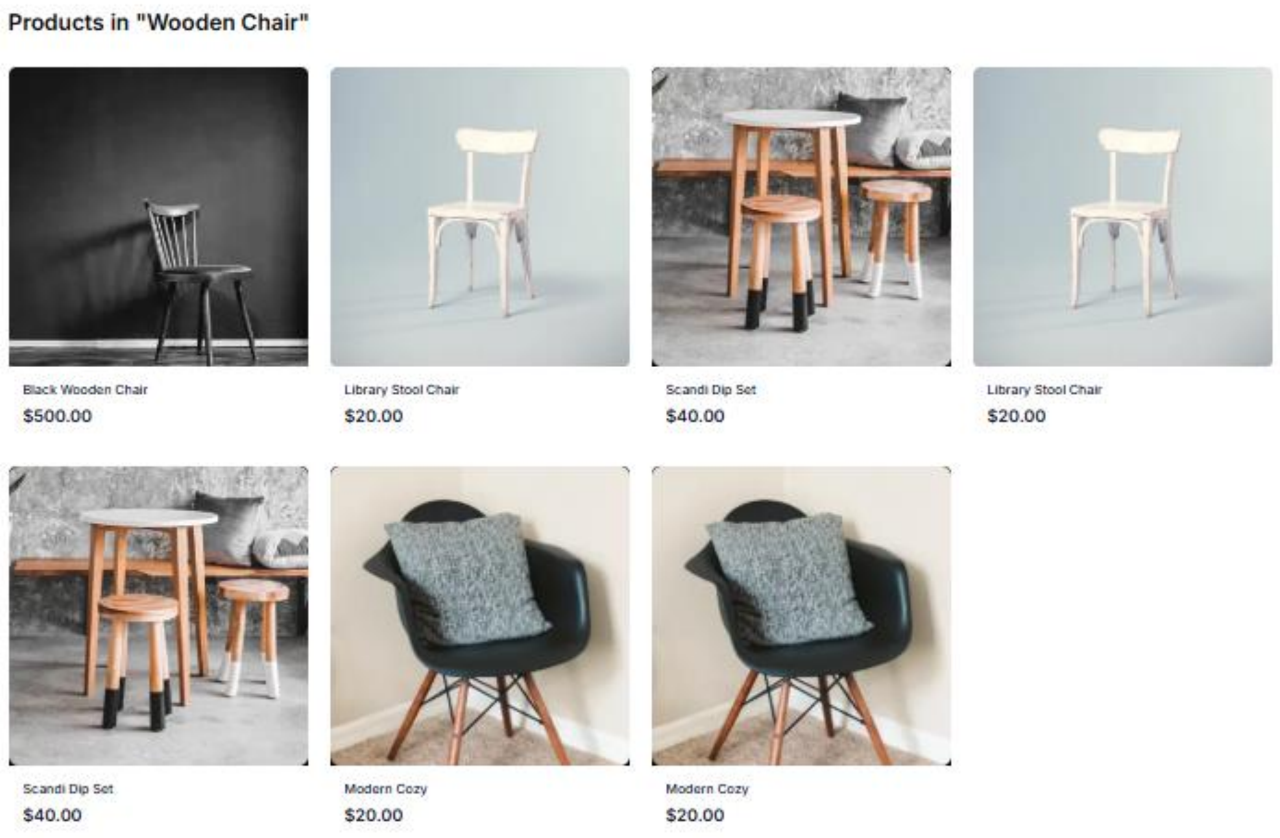
$20.00 USD

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

Quantity: – 1 +

Add To Cart

- Category Filter:



Products in "Wooden Chair"

Black Wooden Chair
$500.00

Library Stool Chair
$20.00

Scandi Dip Set
$40.00

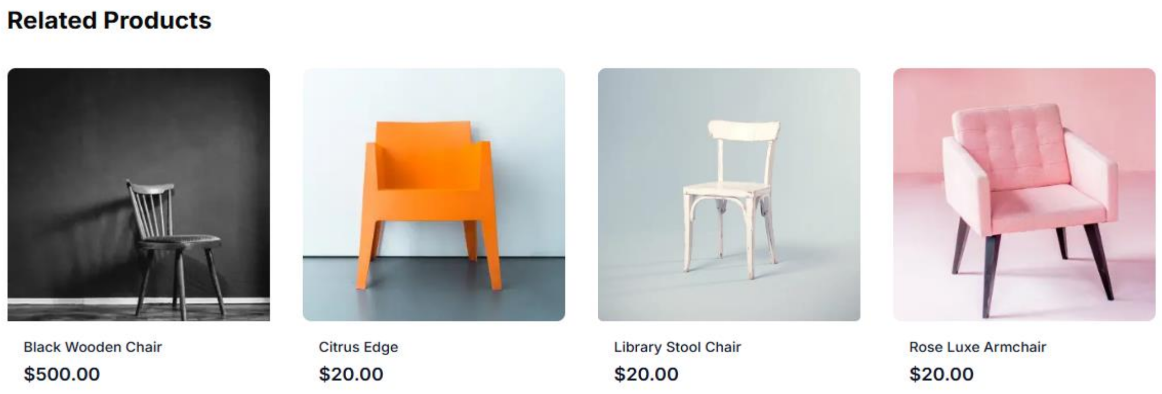
Library Stool Chair
$20.00

Scandi Dip Set
$40.00

Modern Cozy
$20.00

Modern Cozy
$20.00

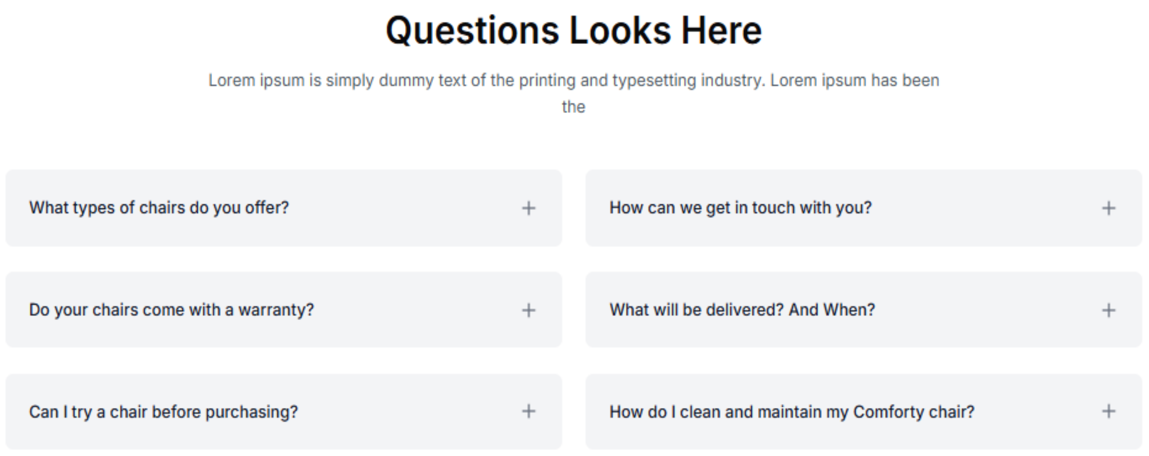
Related Products:



Related Products

Black Wooden Chair	Citrus Edge	Library Stool Chair	Rose Luxe Armchair
$500.00	$20.00	$20.00	$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? +

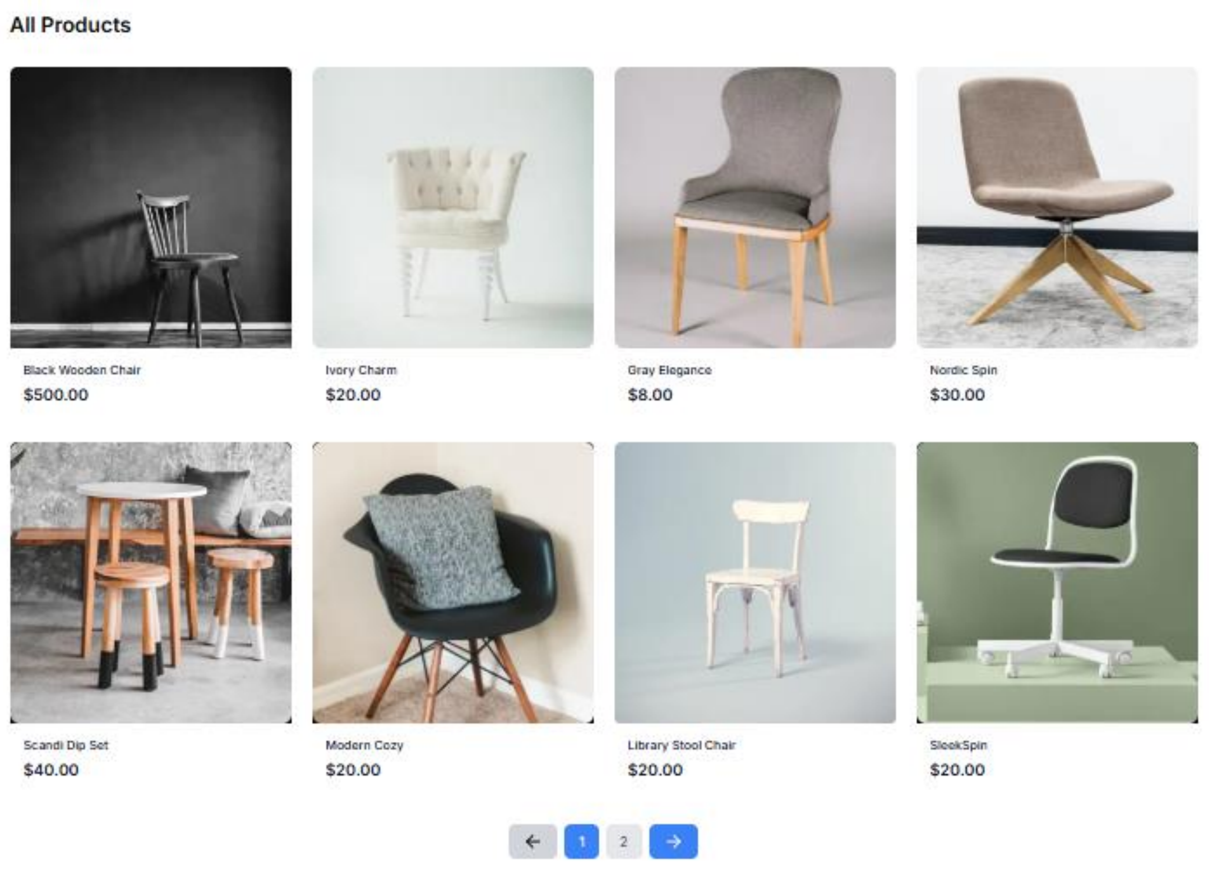
Do your chairs come with a warranty? +

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

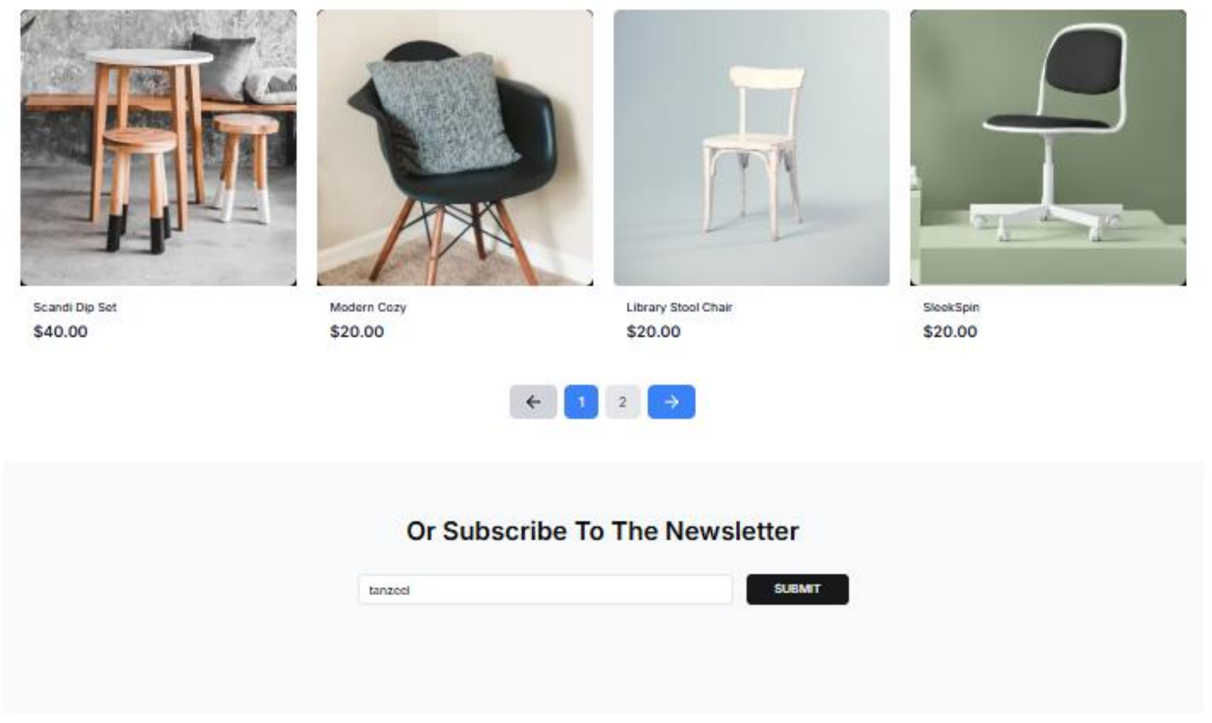
Scandi Dip Set
$40.00

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

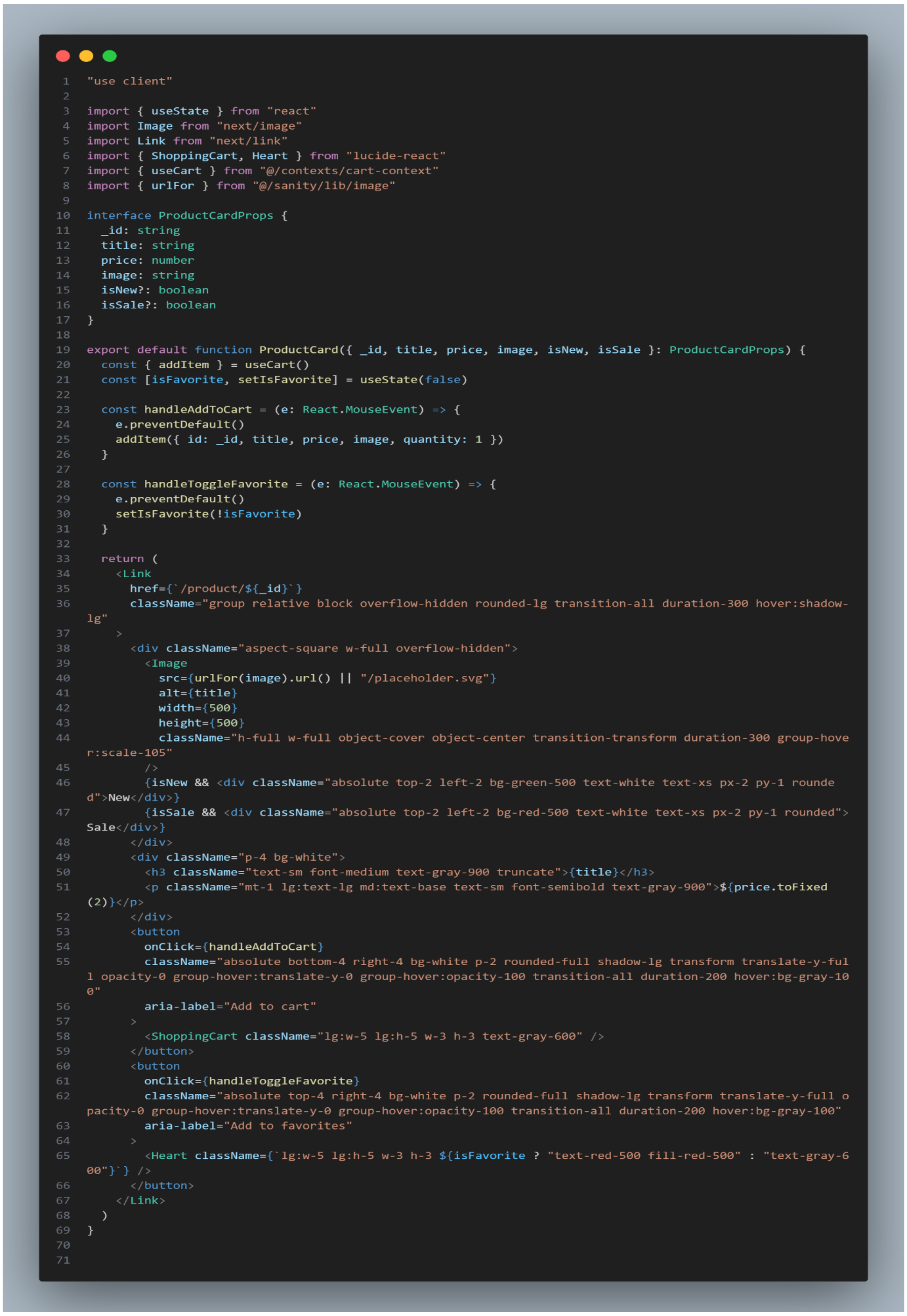
SleekSpin
$20.00

Or Subscribe To The Newsletter

tanzeel

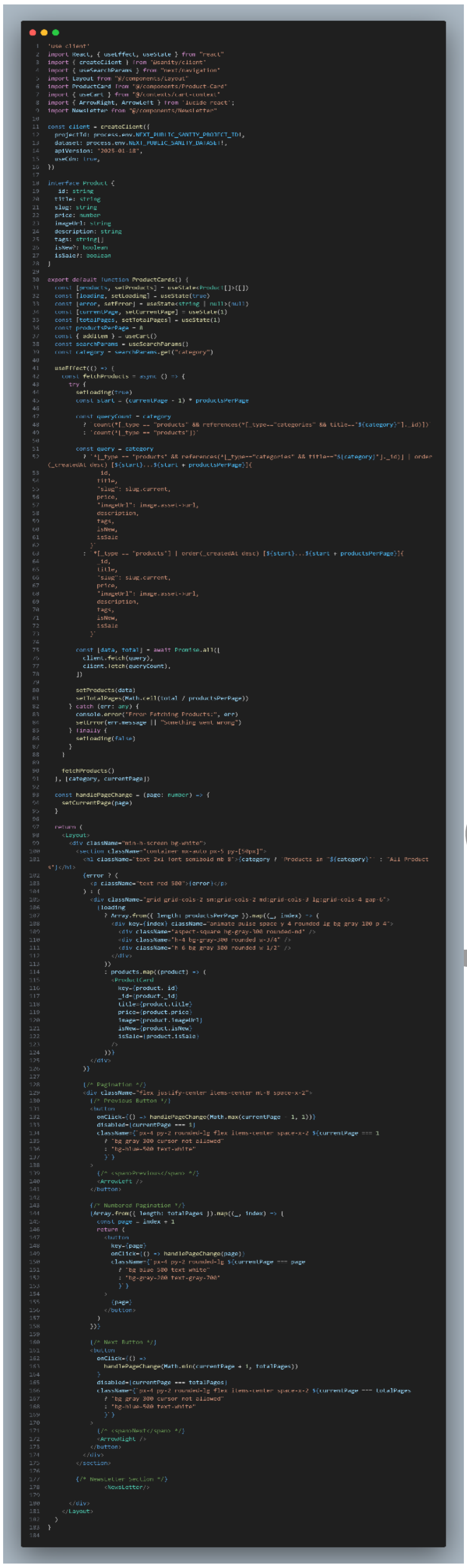
2. Code Deliverable:

- Product Card:



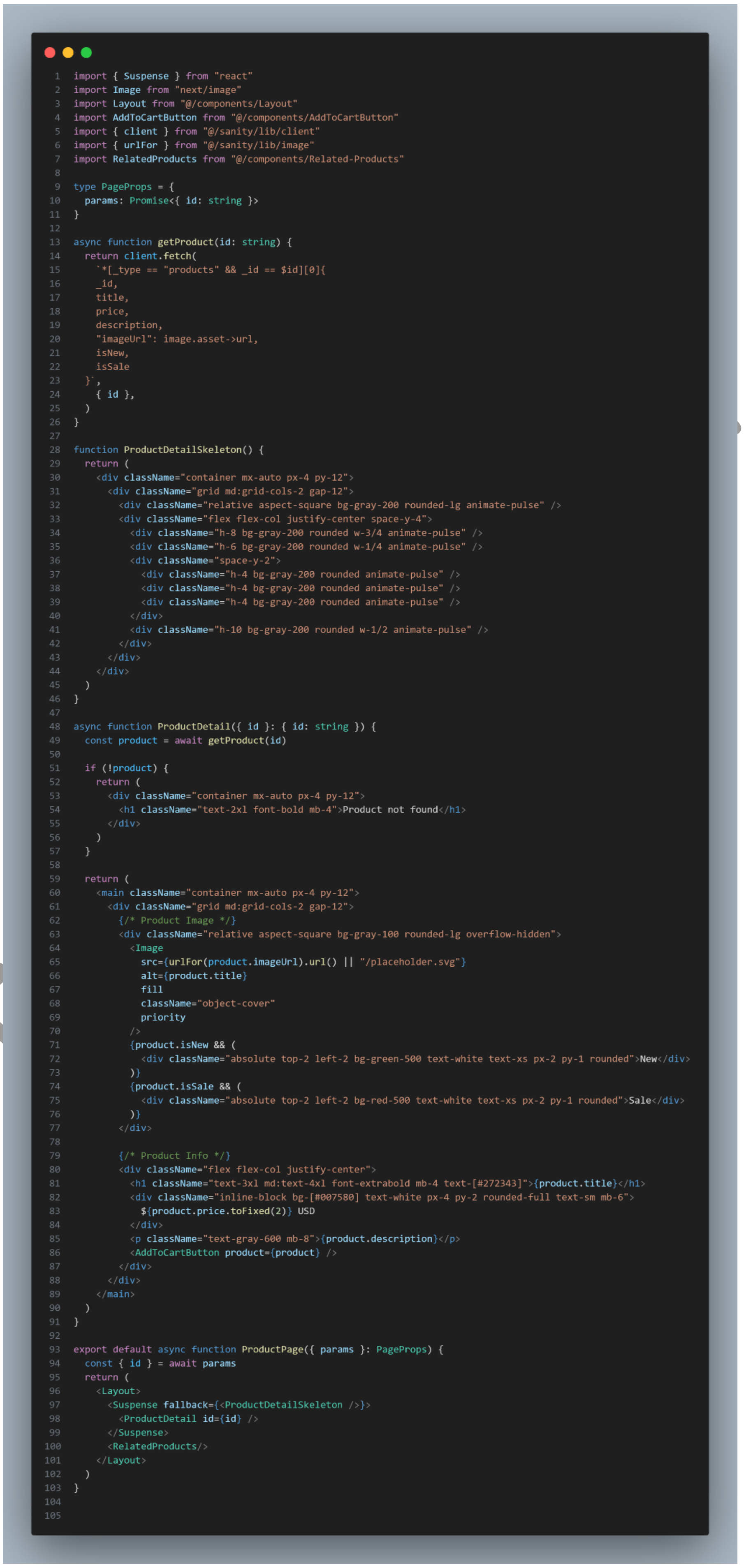
```tsx
"use client"

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>
  )
}
```

Products Page:



```tsx
'use client'
import React, { useEffect, useState } from "react"
import { createClient } from "@sanity/client"
import { useSearchParams } from "next/navigation"
import Layout from "@/components/Layout"
import ProductCard from "@/components/Product-Card"
import { useCart } from "@/contexts/cart-context"
import { ArrowRight, ArrowLeft } from 'lucide-react';
import Newsletter from "@/components/Newsletter"

const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID!,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET!,
  apiVersion: "2025-01-18",
  useCdn: true,
})

interface Product {
   _id: string
  title: string
  slug: string
  price: number
  imageUrl: string
  description: string
  tags: string[]
  isNew?: boolean
  isSale?: boolean
}

export default function ProductCards() {
  const [products, setProducts] = useState<Product[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [currentPage, setCurrentPage] = useState(1)
  const [totalPages, setTotalPages] = useState(1)
  const productsPerPage = 8
  const { addItem } = useCart()
  const searchParams = useSearchParams()
  const category = searchParams.get("category")

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        setLoading(true)
        const start = (currentPage - 1) * productsPerPage

        const queryCount = category
          ? `count(*[_type == "products" && references(*[_type=="categories" && title=="${category}"]._id)])`
          : `count(*[_type == "products"])`

        const query = category
          ? `*[_type == "products" && references(*[_type=="categories" && title=="${category}"]._id)] | order
(_createdAt desc) [${start}...${start + productsPerPage}]{
               _id,
              title,
              "slug": slug.current,
              price,
              "imageUrl": image.asset->url,
              description,
              tags,
              isNew,
              isSale
            }`
          : `*[_type == "products"] | order(_createdAt desc) [${start}...${start + productsPerPage}]{
              _id,
              title,
              "slug": slug.current,
              price,
              "imageUrl": image.asset->url,
              description,
              tags,
              isNew,
              isSale
            }`

        const [data, total] = await Promise.all([
          client.fetch(query),
          client.fetch(queryCount),
        ])

        setProducts(data)
        setTotalPages(Math.ceil(total / productsPerPage))
      } catch (err: any) {
        console.error("Error Fetching Products:", err)
        setError(err.message || "Something went wrong")
      } finally {
        setLoading(false)
      }
    }

    fetchProducts()
  }, [category, currentPage])

  const handlePageChange = (page: number) => {
    setCurrentPage(page)
  }

  return (
    <Layout>
      <div className="min-h-screen bg-white">
        <section className="container mx-auto px-5 py-[50px]">
          <h1 className="text-2xl font-semibold mb-8">{category ? `Products in "${category}"` : "All Product
s"}</h1>
          {error ? (
            <p className="text-red-500">{error}</p>
          ) : (
            <div className="grid grid-cols-2 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
              {loading
                ? Array.from({ length: productsPerPage }).map((_, index) => (
                  <div key={index} className="animate-pulse space-y-4 rounded-lg bg-gray-100 p-4">
                    <div className="aspect-square bg-gray-300 rounded-md" />
                    <div className="h-4 bg-gray-300 rounded w-3/4" />
                    <div className="h-6 bg-gray-300 rounded w-1/2" />
                  </div>
                ))
                : products.map((product) => (
                  <ProductCard
                    key={product._id}
                    _id={product._id}
                    title={product.title}
                    price={product.price}
                    image={product.imageUrl}
                    isNew={product.isNew}
                    isSale={product.isSale}
                  />
                ))}
            </div>
          )}

          {/* Pagination */}
          <div className="flex justify-center items-center mt-8 space-x-2">
            {/* Previous Button */}
            <button
              onClick={() => handlePageChange(Math.max(currentPage - 1, 1))}
              disabled={currentPage === 1}
              className={`px-4 py-2 rounded-lg flex items-center space-x-2 ${currentPage === 1
                ? "bg-gray-300 cursor-not-allowed"
                : "bg-blue-500 text-white"
                }`}
            >
              {/* <span>Previous</span> */}
              <ArrowLeft />
            </button>

            {/* Numbered Pagination */}
            {Array.from({ length: totalPages }).map((_, index) => {
              const page = index + 1
              return (
                <button
                  key={page}
                  onClick={() => handlePageChange(page)}
                  className={`px-4 py-2 rounded-lg ${currentPage === page
                    ? "bg-blue-500 text-white"
                    : "bg-gray-200 text-gray-700"
                    }`}
                >
                  {page}
                </button>
              )
            })}

            {/* Next Button */}
            <button
              onClick={() =>
                handlePageChange(Math.min(currentPage + 1, totalPages))
              }
              disabled={currentPage === totalPages}
              className={`px-4 py-2 rounded-lg flex items-center space-x-2 ${currentPage === totalPages
                ? "bg-gray-300 cursor-not-allowed"
                : "bg-blue-500 text-white"
                }`}
            >
              {/* <span>Next</span> */}
              <ArrowRight />
            </button>
          </div>
        </section>

        {/* NewsLetter Section */}
              <NewsLetter/>

      </div>
    </Layout>
  )
}
```

Products Detail Page:



```tsx
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-8">{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>
  )
}
```

3. Documentation:

✓ *Steps taken to build and integrate components:*

- **Tech Stack**: Chose React.js, Node.js/Express, and MongoDB.
- **Design**: Created wireframes in Figma.
- **Frontend**: Built React components and used Material-UI.
- **Backend**: Developed APIs with Node.js, Passport.js for auth, JWT for secure login.
- **API Integration**: Integrated third-party APIs and tested them.
- **Database**: Used MongoDB for user data storage and validation.
- **Testing**: Conducted unit tests and manual testing.
- **Deployment**: Deployed on Heroku/AWS (backend) and Netlify (frontend).
- **Optimization**: Improved performance and UI.

✓ *Challenges faced and solutions implemented:*

✓ During the Comforty hackathon, I faced challenges with UI design, API integration, and performance. I used Figma, AI, and friend input to refine the UI, debugged APIs with Postman, and solved issues through research. I optimized the database and performance, with AI suggestions, and resolved deployment issues on Vercel. Most challenges were tackled independently, with minimal help.

✓ *Best practices followed during development:*

I followed best practices such as writing modular code, using Git for version control, documenting key sections, ensuring mobile-friendly design, and implementing robust error handling, optimized performance, and set up Vercel for smooth deployment, all while using an iterative development approach.