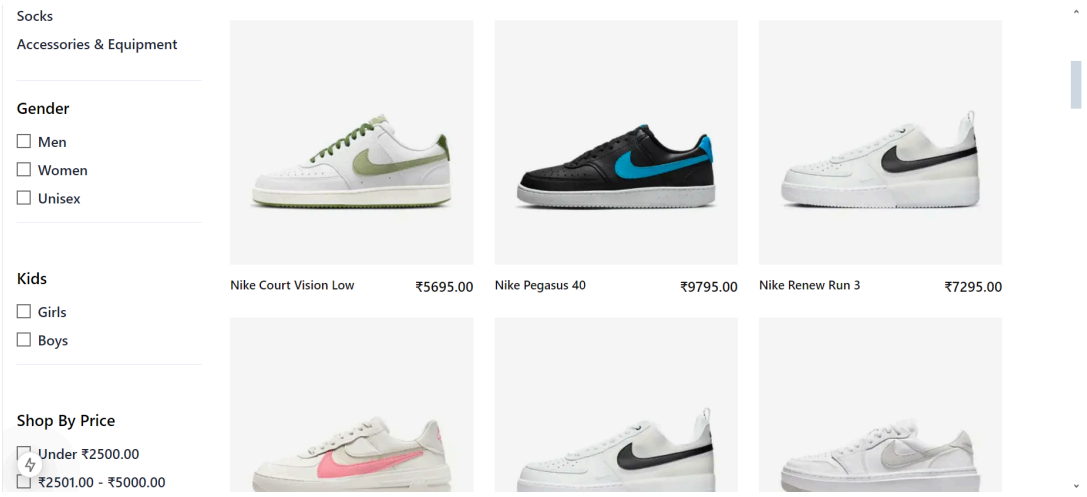
Day-04: Building and Integrating Components for an E-Commerce Website (part 1)

This report explains the steps I took to build and integrate the main components of my e-commerce website. I'll also describe the challenges I faced and how I solved them, especially the issue of missing IDs in the API.

Let's start by Product Listing:



That's how I displayed data from Sanity into Front-end.

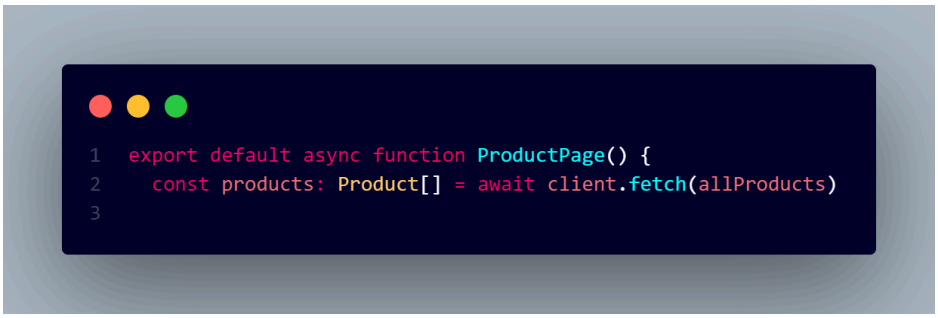
Code for Product listing:

```
{/* Products Section */}
      <main className="w-full md:w-3/4 p-4 grid grid-cols-2 sm:grid-cols-3 lg:grid-cols-3 gap-6">
        {products.map((product: Product) => (
          <ProductCardSidebar key={product.id} product={product} />
        ))}
      </main>
```

GROQ Query for all products.

```
export const allProducts = groq`*[_type == "product"]{
        id,
        productName,
        description,
        price,
        stockStatus,
        "image": image.asset->url
    }
    `;

```

Fetching products through query in Product Page.



```
export default async function ProductPage() {
  const products: Product[] = await client.fetch(allProducts)

```

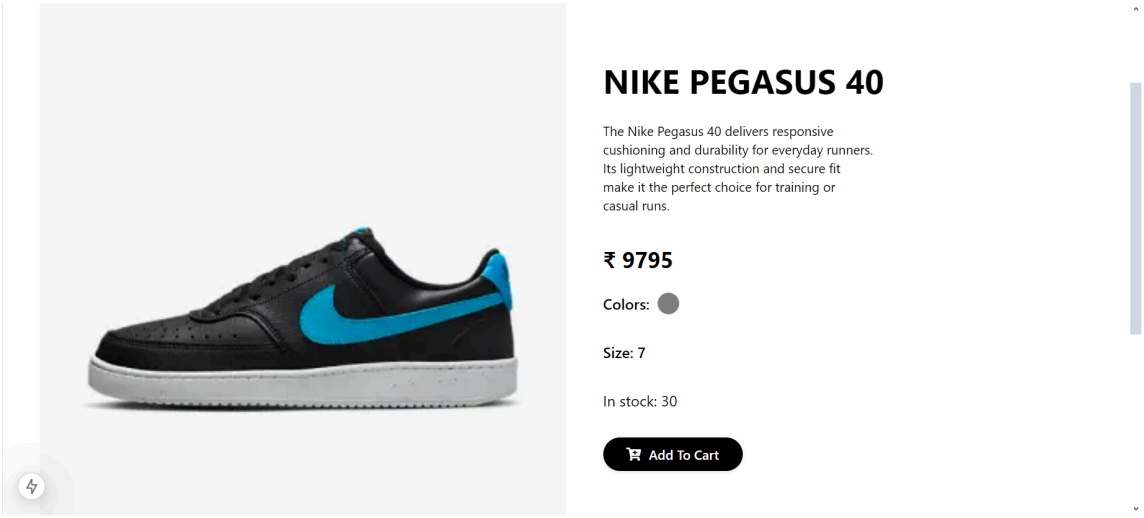
Code for Front-end styling of product card.

```tsx
import React from "react"
import Image from "next/image"
import Link from "next/link"
import type { Product } from "../../types/product"

interface ProductCardSidebarProps {
  product: Product
}

export function ProductCardSidebar({ product }: ProductCardSidebarProps) {
  return (
    <Link href={`/products/${encodeURIComponent(product.id)}`}>
      <div className="flex flex-col text-sm sm:text-base leading-relaxed max-w-[150px] sm:max-w-[250px] lg:max-w-[300px]">
        <Image
          src={product.image || "/placeholder.svg"}
          alt={product.productName}
          width={300}
          height={300}
          className="w-full h-auto object-cover"
        />
        <div className="flex flex-wrap gap-4 sm:gap-10 items-start justify-between mt-3 bg-white">
          <div className="flex flex-col text-xs sm:text-sm pr-1">
            <div className="font-medium text-neutral-900">{product.productName}</div>
            <div className="text-neutral-500">{product.category}</div>
          </div>
          <div className="self-start font-medium text-right text-neutral-900">₹{product.price.toFixed(2)}</div>
        </div>
      </div>
    </Link>
  )
}

```

Product Details Page:

The product details page shows detailed information about a selected product.

such as Product name, Product Description, Product Price, Product Size, Product Color, and more..



I fetched the product data from Sanity CMS based on the ID passed in the URL.

Styling for Product detail page.

```jsx
return (
    <div className="flex flex-col md:flex-row justify-between p-5 sm:p-10">
      <div className="md:w-1/2">
        <Image
          src={product.image || "/placeholder.svg"}
          alt={product.productName}
          width={653}
          height={653}
          className="h-auto max-w-full max-h-[650px] object-cover"
        />
      </div>

      <div className="md:w-1/2 flex justify-center items-left mt-5 md:mt-0">
        <div className="flex justify-center items-left flex-col gap-3 p-10">
          <h2 className="text-4xl font-bold uppercase text-left">{product.productName}</h2>
          <p className="text-sm leading-5 w-full md:w-[60%] pt-3 pb-2 text-left">{product.description}</p>
          <p className="text-2xl font-bold text-left pt-3 pb-2">₹ {product.price}</p>

          <div className="flex flex-wrap gap-2 mb-4">
            <p className="font-semibold">Colors:</p>
            {product.colors &&
              product.colors.map((color, index) => (
                <div
                  key={index}
                  className="w-6 h-6 rounded-full border border-gray-300"
                  style={{ backgroundColor: color }}
                ></div>
              ))}
          </div>

          <div className="mb-4">
            <p className="font-semibold">Size: {product.size}</p>
          </div>

          <div className="mb-4">
            <p>In stock: {product.inventory}</p>
          </div>

          <Button className="flex items-center gap-2 w-[150px]" onClick={handleAddToCart} disabled={addedToCart}>
            <FaCartPlus />
            {addedToCart ? "Added to Cart" : "Add To Cart"}
          </Button>
          {addedToCart && <p className="text-green-600 mt-2">Product added to cart successfully!</p>}
        </div>
      </div>
    </div>
  )
```

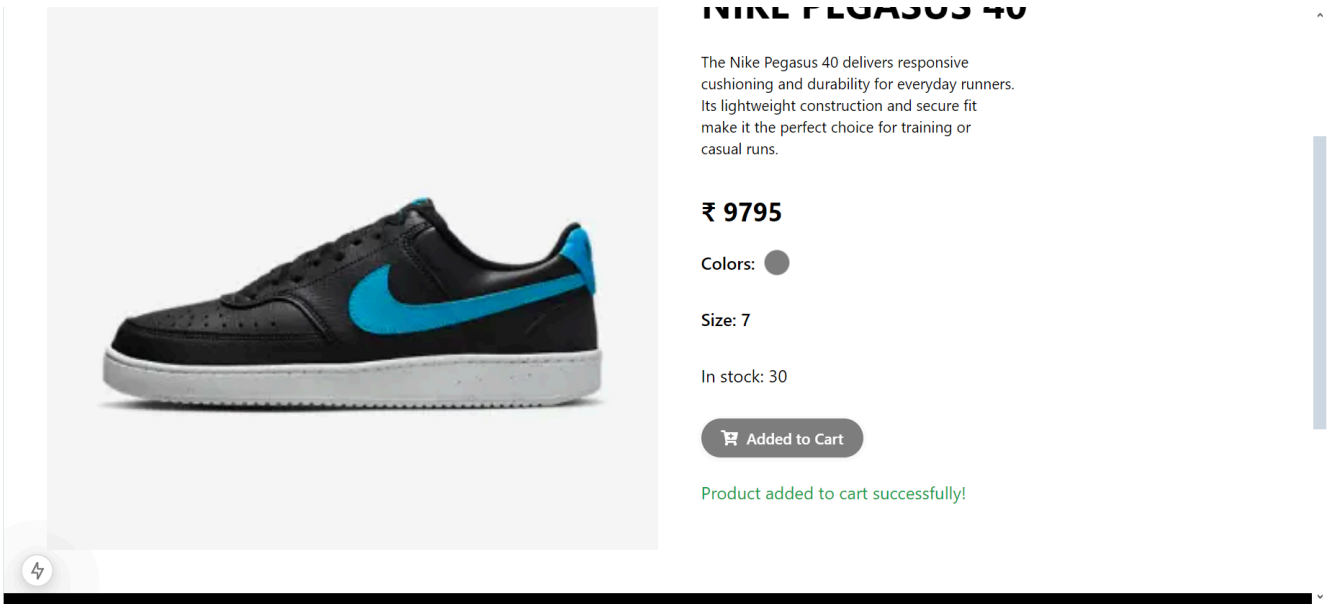
GROQ Query for specific product detail through ID.

```groq
export const prodetail = (id: string) => groq`*[_type == "product" && id == "${id}"][0]{
        id,
        productName,
        category,
        price,
        inventory,
        colors,
        stockStatus,
        "image": image.asset->url,
        description,
        size
      }`;
```

Fetching product detail of a specific id through query in product detail page.

```tsx
export default async function ProductDetailPage({ params }: { params: { id: string } }) {
  const { id } = params
  const product: ProductDetailPage = await client.fetch(prodetail(id), { id })

  if (!product) {
    return <div>Loading...</div>
  }

```

Notification.

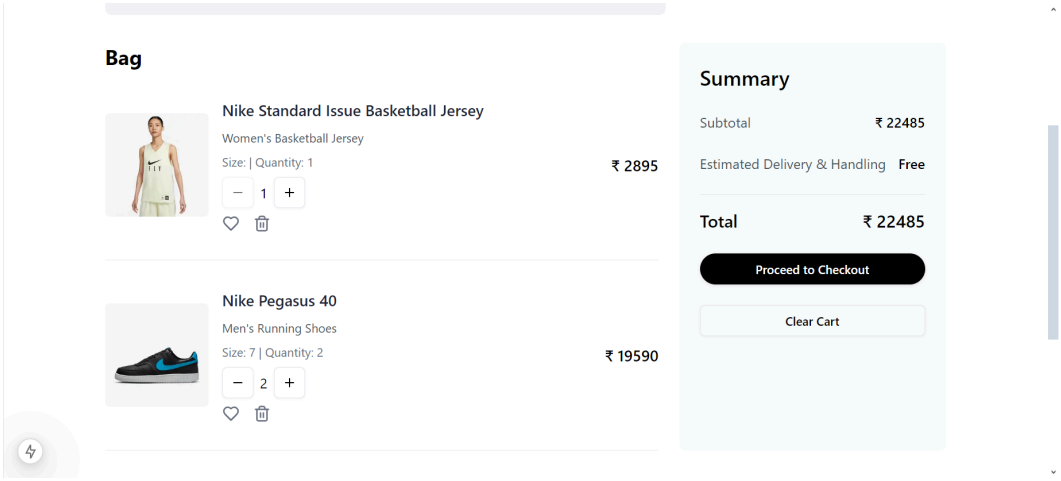


I used a simple toast-style notification that disappears after a few seconds in cart button, that triggered on specific action "Item added to cart" .. shows message to users about action “Product added to cart successfully”.

I used React state to manage the visibility of notifications.

Components Built and Integrated

1. Cart Component



The cart component allows users to add products they want to buy.

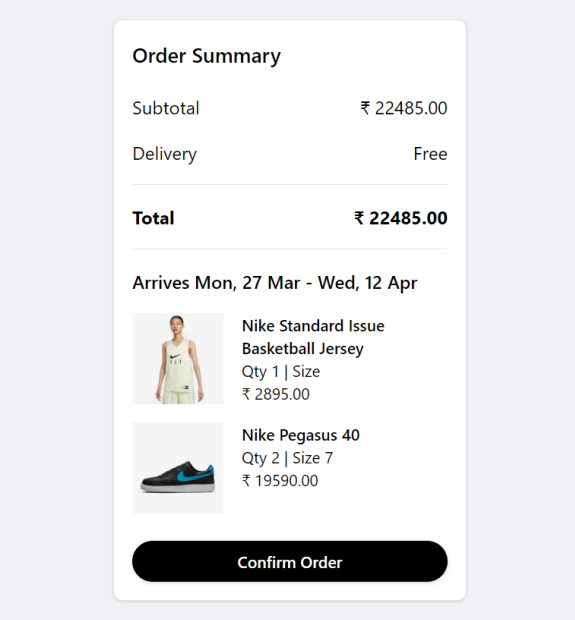
I fetched product data from the Sanity CMS using GROQ queries.

I added a button on the product details page to send the product data (ID, name, price, etc.) to the cart.

React state to store the cart items. Whenever an item is added or removed, the state updates the cart display.

I created a simple UI to show all products in the cart, including their quantity and total price.

2. Checkout Component



The checkout component helps users complete their purchase.

I passed the cart data to the checkout component using props.

Before placing the order, the checkout page shows a summary of all cart items and the total amount.

Challenges Face by me.

1. Missing IDs in API

The product data from the API did not include IDs. IDs are important for routing, managing state, and identifying products.

Solution:

I manually added unique IDs to each product in Sanity CMS. This allowed me to use these IDs for routing and other features in the project.

2. GROQ Query Errors

I faced many errors in fetching data from Sanity CMS due to incorrect GROQ queries.

Solution:

I carefully reviewed the GROQ query syntax and cross-checked it with the Sanity documentation. Once I corrected the query, the data was fetched successfully.

There are many more components I have created, I will add them in my next part of Technical Documentation.

Created by *ARISHBA KHAN -SENIOR STUDENT*

SUNDAY 2 TO 5PM.