Day 3 - API Integration Report - [Ann Fashion Store]

Hackathon Task #3

Component Overview

The CataloguePage component fetches product data from **Sanity CMS** and renders it in a user-friendly interface. It is divided into two main parts:

1. **Data Fetching:** Retrieves product data from Sanity CMS using GROQ queries.
2. **CatalogueUI:** Displays the fetched data in a responsive grid layout with product details and images.

- **Purpose:** Fetches product data from Sanity CMS.
- **Error Handling:** Displays an error message if the fetch fails.
- **Loading State:** Shows a loading message while data is being fetched.

Key Features

1. **Responsive Design:**
   - Adapts to screen sizes using Tailwind CSS grid utilities.
2. **Error and Loading States:**
   - Displays appropriate messages during data fetching or errors.
3. **Dynamic Data Rendering:**
   - Uses data from Sanity CMS to populate product details dynamically.

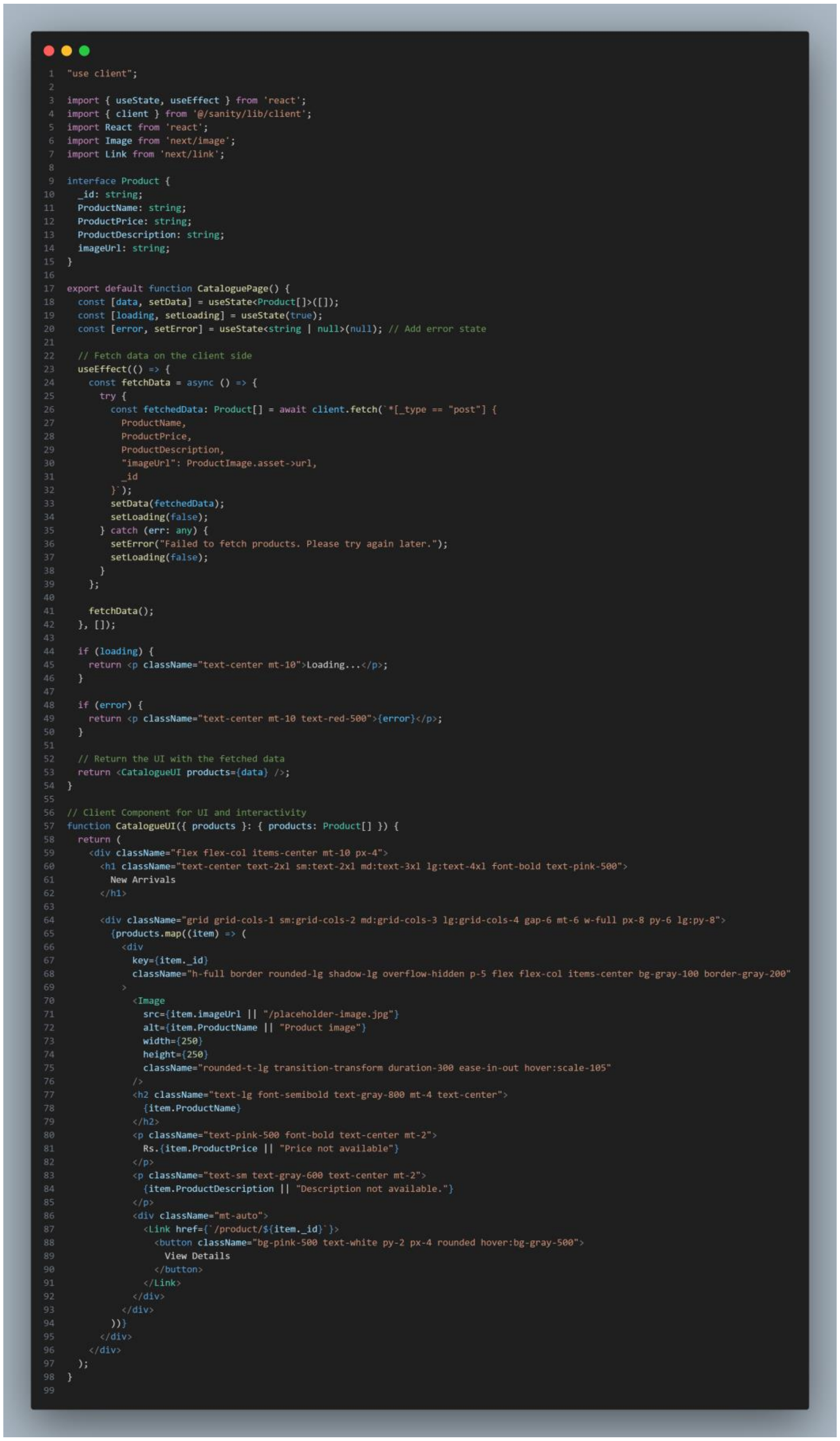
Usage

- **Fetch and Display Products:** Automatically fetches data from Sanity CMS when the component loads.
- **Responsive Layout:** Optimized for all screen sizes, ensuring a seamless user experience.
- **Navigation:** Links to individual product pages for more details.

Benefits

- Simplifies managing and displaying product listings with minimal manual effort.
- Provides a scalable solution for integrating product catalogs into e-commerce projects.
- Enhances the user experience with responsive design and real-time data fetching.

This component is ideal for building visually appealing and functional product listing pages in a marketplace project.



```tsx
"use client";

import { useState, useEffect } from 'react';
import { client } from '@/sanity/lib/client';
import React from 'react';
import Image from 'next/image';
import Link from 'next/link';

interface Product {
  _id: string;
  ProductName: string;
  ProductPrice: string;
  ProductDescription: string;
  imageUrl: string;
}

export default function CataloguePage() {
  const [data, setData] = useState<Product[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null); // Add error state

  // Fetch data on the client side
  useEffect(() => {
    const fetchData = async () => {
      try {
        const fetchedData: Product[] = await client.fetch(`*[_type == "post"] {
          ProductName,
          ProductPrice,
          ProductDescription,
          "imageUrl": ProductImage.asset->url,
          _id
        }`);
        setData(fetchedData);
        setLoading(false);
      } catch (err: any) {
        setError("Failed to fetch products. Please try again later.");
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  if (loading) {
    return <p className="text-center mt-10">Loading...</p>;
  }

  if (error) {
    return <p className="text-center mt-10 text-red-500">{error}</p>;
  }

  // Return the UI with the fetched data
  return <CatalogueUI products={data} />;
}

// Client Component for UI and interactivity
function CatalogueUI({ products }: { products: Product[] }) {
  return (
    <div className="flex flex-col items-center mt-10 px-4">
      <h1 className="text-center text-2xl sm:text-2xl md:text-3xl lg:text-4xl font-bold text-pink-500">
        New Arrivals
      </h1>

      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6 mt-6 w-full px-8 py-6 lg:py-8">
        {products.map((item) => (
          <div
            key={item._id}
            className="h-full border rounded-lg shadow-lg overflow-hidden p-5 flex flex-col items-center bg-gray-100 border-gray-200"
          >
            <Image
              src={item.imageUrl || "/placeholder-image.jpg"}
              alt={item.ProductName || "Product image"}
              width={250}
              height={250}
              className="rounded-t-lg transition-transform duration-300 ease-in-out hover:scale-105"
            />
            <h2 className="text-lg font-semibold text-gray-800 mt-4 text-center">
              {item.ProductName}
            </h2>
            <p className="text-pink-500 font-bold text-center mt-2">
              Rs.{item.ProductPrice || "Price not available"}
            </p>
            <p className="text-sm text-gray-600 text-center mt-2">
              {item.ProductDescription || "Description not available."}
            </p>
            <div className="mt-auto">
              <Link href={`/product/${item._id}`}>
                <button className="bg-pink-500 text-white py-2 px-4 rounded hover:bg-gray-500">
                  View Details
                </button>
              </Link>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
}

```