Day 4 - Dynamic Frontend Components - [Ann Fashion Store]

Hackathon Task #4

Dynamic Frontend Components Documentation:

Objective

This document outlines the implementation and development process for creating dynamic frontend components to display marketplace data fetched from Sanity CMS or APIs. The focus is on modular and reusable component design to ensure scalability and responsiveness, aligning with real-world practices.

Completed Work

- **Dynamic Design**: Developed a product page that dynamically fetches and displays data from Sanity CMS.
- **Reusable Structure**: Components are designed to be modular, facilitating reuse and easy maintenance.
- **Responsive Layout**: Implemented responsive design principles to ensure the product page displays optimally on all devices.
- **Data Fetching**: Integrated Sanity CMS to fetch real-time product data.

- **Dynamic Routing**: Each product has its unique route, dynamically generated based on the fetched data.
- **Real-Time Updates**: Any changes in the Sanity CMS reflect immediately on the frontend.
- **Basic State Management**: Used hooks like useState and useEffect to manage component state and handle API calls.
- **UI/UX**: Designs with animations, transitions, and intuitive layouts.

```tsx
import { client } from "@/sanity/lib/client";
import ProductDetailUI from "@/app/Components/ProductDetailUI";

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

export default async function ProductDetailPage({ params }: { params: { id: string } }) {

  const { id } = params;

  // Fetch product data from Sanity
  const product: Product | null = await client.fetch(
    `*[_type == "post" && _id == $id][0] {
      ProductName,
      ProductPrice,
      ProductDescription,
      "imageUrl": ProductImage.asset->url,
      _id
    }`,
    { id }
  );

  if (!product) {
    return <p className="text-center mt-10">Product not found.</p>;
  }

  // Pass the fetched product data to the client-side UI
  return <ProductDetailUI product={product} />

}
```