Day 4 Documentation

Project Title: Building Dynamic Frontend Components for Marketplace

Objective

To dynamically render product listings and product detail pages by integrating data from Sanity CMS and implementing dynamic routing in a Next.js application.

Tasks Completed

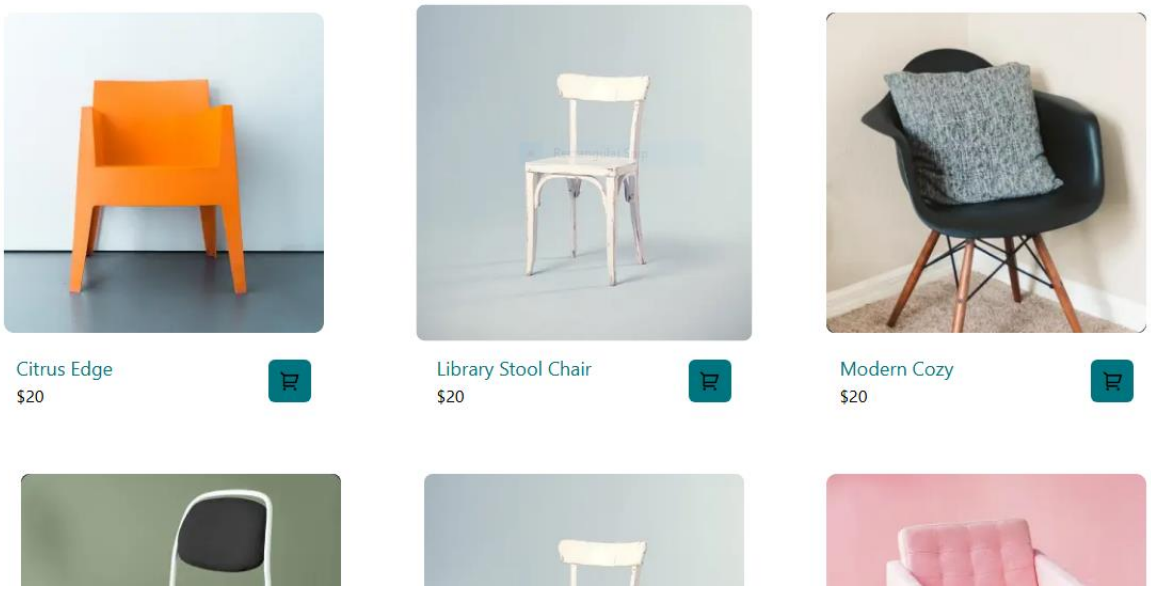
1. Fetching Data from Sanity CMS for Product Listing

```
export default async function ProductPage() {
  const resp = await client.fetch(`*[_type=="products"]{
    _id, title, price, description, "image":image.asset->url, badge, inventory, _createdAt, tags
  }`);

  return (
```

- **What I Did:** Replaced static product data with dynamic data fetched from Sanity CMS.
- **How I Did It:**
  - Used the client.fetch method to retrieve products from Sanity.
  - Implemented a query to fetch product details including title, price, description, and image.
- **Query Example:**

```
export default async function ProductPage() {
  const resp = await client.fetch(`*[_type=="products"]{
    _id, title, price, description, "image":image.asset->url, badge, inventory, _createdAt, tags
  }`);

  return (
```

- **Result:** All products are dynamically rendered on the product listing page.



2. Created a Reusable ProductCard Component

- **What I Did:** Built a ProductCard component to display individual product data.
- **How I Did It:**
  - Used props to pass product details.
  - Rendered Image, title, and price inside the card.
- **Example Code:**

```tsx
import Image from "next/image";
import Link from "next/link";
export default function ProductCard({ product }) {
  return (
    <Link href={`/Products/${product._id}`}>
      <div key={product._id} className="h-[400px] w-[270px]">
      <Image
        src={product.image}
        alt={product.title}
        width={250}
        height={250}
        className="rounded-lg hover:scale-105 cursor-pointer"
      />
      <div className="w-full h-[90px] flex items-center justify-between px-3">
        <div>
          <h2 className="text-[18px] font-normal text-[#007580]">{product.title}</h2>
          <p>${product.price}</p>
        </div>
        <div className="h-[35px] w-[35px] bg-[#007580] flex items-center justify-center rounded-md cursor-pointer
          <i className="ri-shopping-cart-2-line text-xl"></i>
        </div>
      </div>
    </div>
    </Link>

  );
}
```

${product.price}

- **Result: The component is reused in the product listing page, dynamically rendering products.**

```tsx
import ProductCard from "../components/productCard";

export default async function ProductPage() {
  const resp = await client.fetch(`*[_type=="products"]{
    _id, title, price, description, "image":image.asset->url, badge, inventory, _createdAt, tags
  }`);

  return (
    <div className="px-7 py-10 flex flex-col gap-5">

      <div className="w-full px-[7vw] grid grid-cols-1 md:space-x-4 sm:grid-cols-2  md:grid-cols-2 lg:grid-cols-4 gap-
        {resp.map((product) => {
          return(<ProductCard key={product._id} product={product} />)

  })}
      </div>
```

3. Implemented Dynamic Routing for Product Detail Pages

- **What I Did:** Created a dynamic route to handle individual product detail pages.
- **How I Did It:**
    - Created a folder structure: app/Products/[id]/page.tsx.

- Used params.id to fetch data for a specific product.

- **Example Code:**

```
import { client } from "@/sanity/lib/client";
import Image from "next/image";
import { urlFor } from "@/sanity/lib/image";
import Link from "next/link";
import ProductCard from "@/app/components/productCard";

export default async function ProductDetail({params}: {params: { id: string }}) {
  let product = await client.fetch(`*[_type == "products" && _id == $id][0]`, {
    id: params.id,
  });

  if (!product) {
    return <div>PRODUCT NOT FOUND</div>;
  }
  console.log(product.image);
```

- **Result:** Clicking on a product card navigates to the detail page with complete product information.

4. Solved Image Rendering Issues

- **What I Did:** Fixed errors related to empty src attributes in the Image component.
- **How I Did It:**
  - Checked if product.image was valid before rendering.

- Used Sanity's image URL builder to convert the image object to a URL.

- **Example Code with Sanity Image Builder:**

```
<div className=" w-full md:w-[50%] h-full ">
  <Image
    src={urlFor(product.image).url()}
    alt="Pink Chair"
    width={500}
    height={500}
    className="object-cover object-center rounded-xl"
  />
</div>
```

- **Result: The error was resolved, and images are now correctly displayed.**

5. Created a Cart Component

- **What I Did**: Built a `Cart` component to display products added to the cart.
- **How I Did It**:
  - Implemented state management to track items.
  - Displayed product details and total price in the cart.
- **Result**: Users can view and manage items in their cart.

6. Created an FAQ Component

- **What I Did**: Developed an FAQ section to display common questions and answers.
- **How I Did It**:
  - Created a collapsible UI for questions and answers.
  - Styled using Tailwind CSS for better UX.
- **Result**: A responsive FAQ section that improves user interaction.

Challenges Faced

1. **404 Page Not Found Error**: Resolved by ensuring the correct folder structure and URL matching.

2. **Empty Image Source Error:** Fixed by implementing conditional rendering and using the image builder.

Technologies Used

- Next.js for frontend framework and dynamic routing.
- Sanity CMS for content management and data fetching.
- Tailwind CSS for responsive design and styling.

Conclusion

This task helped me understand how to fetch dynamic data from a CMS, create reusable components, and implement dynamic routing effectively. It also improved my problem-solving skills by debugging image rendering issues and routing errors.