

Day-4 Dynamic Frontend Components - Furnitures-Market:

Overview:

In Day 4, I build dynamic components to display marketplace data fetched from sanity CMS. These components are, modular reuseable and highly responsive. Enhances UI, make the website more user friendly and scalable.

What I have done on Day - 4:

- ❖ Fetching the data from sanity and building dynamic components to display correctly in the product listing page.
- ❖ Implemented sign in functionality and created a middleware for redirecting.
- ❖ Created a skeleton loader for displaying while data is being fetching.
- ❖ Add to cart functionality, users can add the items into cart also can delete items.
- ❖ Created a Checkout page. By clicking checkout button user can fill the form and place the order.
- ❖ Category check: the user can select a category, and items will be obtained based on that category.
- ❖ Created a custom 404 – not found page.
- ❖ Ensuring all the pages are responsive.
- ❖ Successfully completed the payment functionality, using stripe to receive customer's payment.

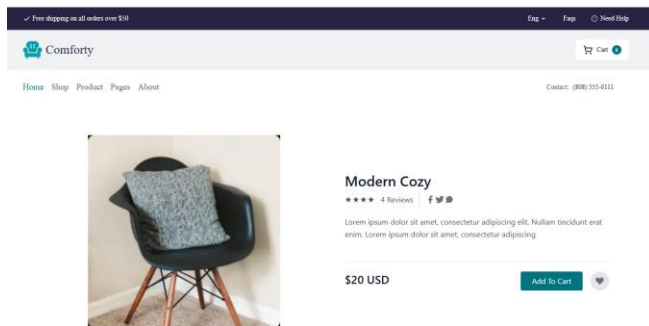
Key components that I have constructed:

1. Product Listing Page:

Created a component displayed the data fetching from sanity (Headless CMS). Render dynamically in a grid layout.

2. Product Detailed Component:

Created a dynamic product detailed component, fetching the data from sanity based on slug. Here user can see the detail of the product and can add the items into cart.



```
1
2 import { client } from "@sanity/lib/client";
3 import ProductList from "@components/productList";
4
5 export interface IProduct {
6   _id:string;
7   slug:string;
8   description: string;
9   imageUrl: string;
10  title: string;
11  price: number;
12 }
13
14
15
16 const Productpage = async ({params}:{params:{slug:string}}) => {
17   const { slug } = params;
18
19   const data: IProduct[] | null = await client.fetch(
20     `*[ _type == "products" && slug.current == $slug ]{
21       _id,
22       title,
23       "imageUrl": image.asset->url,
24       "slug": slug.current,
25       description,
26       price
27     }`,
28     { slug }
29   );
30   console.log(data)
31
32   if (!data) {
33     return <div>Product not found</div>;
34   }
35
36   return (
37     <div>
38       <ProductList products={data}/>
39     </div>
40   )
41 }
42
43 export default Productpage
```

3. Category check:

Also implement a category page and where user can select the items according to their category, and can see the all items in the category.

```
import { client } from "g/sanity/lib/client";
import Image from "next/image";
import Link from "next/link";

interface IProduct {
  _id: string;
  title: string;
  imageurl: string;
  price: number;
  priceWithoutDiscount: number;
  category: {
    _id: string;
    title: string;
    slug: string;
  };
  slug: string;
}

interface ICategoryPageProps {
  params: {
    slug: string;
  };
}

export default async function CategoryPage({ params }: ICategoryPageProps) {
  const { slug } = params;

  // Fetch products for a specific category
  const products: IProduct[] = await client.fetch(
    `*[_type == "products" && category.slug.current == "${categorySlug}"] {
    _id,
    title,
    price,
    priceWithoutDiscount,
    imageurl,
    category->{
      _id,
      title,
      slug: slug.current
    },
    "slug": slug.current
  }`,
    { categorySlug: slug }
  );

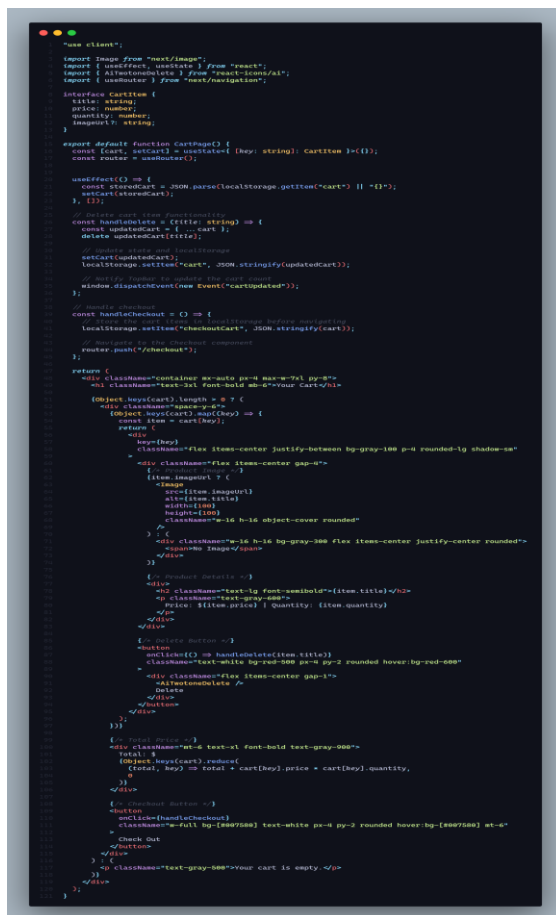
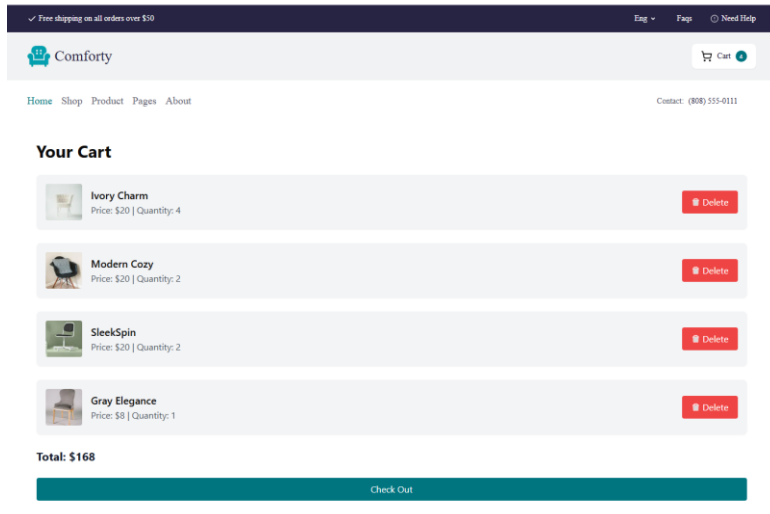
  return (
    <div className="mx-auto max-w-7xl p-5">
      <h1 className="text-3xl font-bold mb-8 italic text-center">
        Products in Category: {products[0].category.title}
      </h1>
      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
        {products.map((product) => (
          <div key={product._id} className="border rounded-lg overflow-hidden shadow-lg">
            <Image
              src={product.imageurl}
              alt={product.title}
              width={100}
              height={100}
              className="w-full h-48 object-cover"
            />
            <div className="p-4">
              <h2 className="text-xl font-sonibold">{product.title}</h2>
              <p className="text-gray-600">{product.category.title}</p>
              <div className="flex justify-between items-center">
                <Link
                  href={`/shop/${product.slug}`}
                  className="mt-4 inline-block px-4 py-2 bg-blue-500 text-white rounded-md hover:bg-blue-600 transition-colors">
                    View Product
                </Link>
                <div className="flex gap-1">
                  <p>{product.price}</p>
                  <p className="text-gray-400 line-through">{product.priceWithoutDiscount}</p>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
}
```

Top categories



4. Add To Cart:

Implemented cart page, where user can add items into cart and see the can see the quantity and individual and total prices of cart items.



5. Checkout Component:

In this page user can provide the details and place their order successfully and when their order is placed successfully, user receive a pop up of thank you for shopping and the cart items will be empty.



Place Order

Name

Phone

Address Line 1

Address Line 2

City

State


Postal Code

Country Code

Residential Address


Yes

Cart Items




Ivory Charm

Price: \$20 | Quantity: 4




Modern Cozy

Price: \$20 | Quantity: 2



SleekSpin

Price: \$20 | Quantity: 2



Gray Elegance

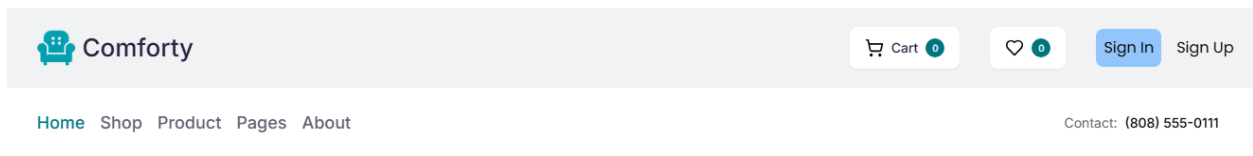
Price: \$8 | Quantity: 1

Place Order

Go Back to Shop →

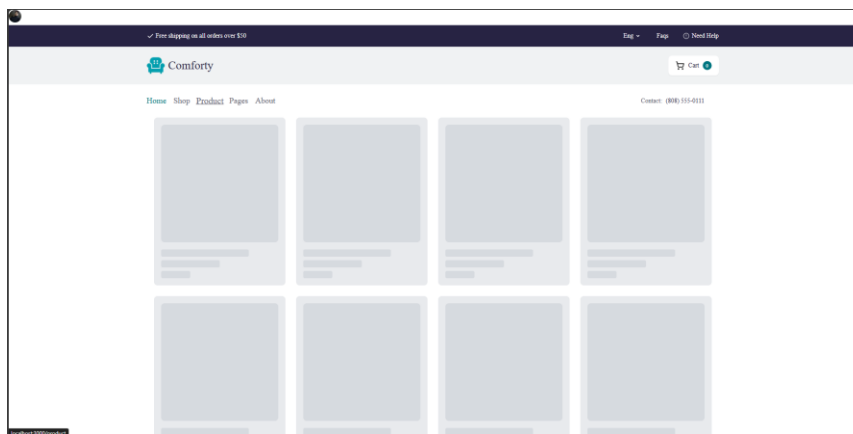
6. Implemented sign in functionality:

User can sign in into the website from their google or GitHub account.
And after sign in, can see the detail of their profile.



7. Skeleton Loader:

Created a skeleton loader in loading.tsx file. So, while fetching when component mount, at that time skeleton loader will appear on the UI for good user experience.




```

1
2
3 const loading = () => {
4   return (
5     <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6 px-4 max-w-7xl mx-auto">
6       {[...Array(8)].map((_, index) => {
7         <div key={index} className="w-full max-w-[358px] mx-auto">
8           <div className="bg-gray-200 rounded-lg p-4 animate-pulse">
9             <div className="aspect-square bg-gray-300 rounded-lg"></div>
10            <div className="mt-4 h-4 bg-gray-300 rounded w-3/4"></div>
11            <div className="mt-2 h-4 bg-gray-300 rounded w-1/2"></div>
12            <div className="mt-2 h-4 bg-gray-300 rounded w-1/4"></div>
13          </div>
14        </div>
15      )}] </div>
16    )
17  }
18
19  export default loading

```

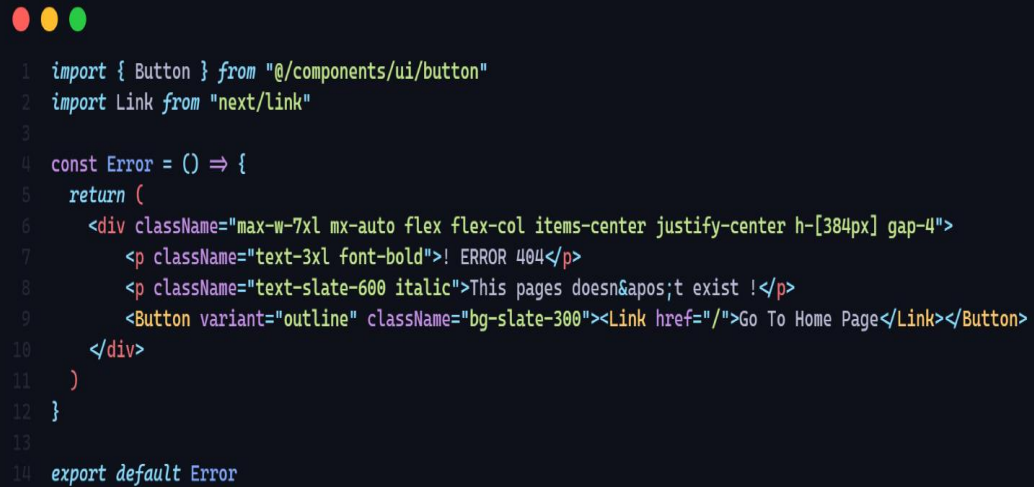
8. Custom 404 – not found page:

Also create a custom 404 page in not-found.tsx for seamless user experience. Providing a button to go to home page if route is incorrect.

! ERROR 404

This pages doesn't exist !

[Go To Home Page](#)



```

1 import { Button } from "@components/ui/button"
2 import Link from "next/link"
3
4 const Error = () => {
5   return (
6     <div className="max-w-7xl mx-auto flex flex-col items-center justify-center h-[384px] gap-4">
7       <p className="text-3xl font-bold">! ERROR 404</p>
8       <p className="text-slate-600 italic">This pages doesn't exist !</p>
9       <Button variant="outline" className="bg-slate-300"><Link href="/">Go To Home Page</Link></Button>
10     </div>
11   )
12 }
13
14 export default Error

```

9. Payment Integration:

Implementing payment functionality has been a wonderful experience playing with real world scenarios really boost the confidence. Now customer can place the order and pay the payment easily.

← Default sandbox **TEST MODE**

Ivory Charm

\$20.00



Or pay with card

Email

Card information

4242 4242 4242 4242 	
MM / YY	CVC 

Cardholder name

Country or region

Pakistan ▼

☐

Securely save my information for 1-click checkout
Pay faster on Default sandbox and everywhere Link is accepted.

Pay

Powered by  **stripe** | [Terms](#) [Privacy](#)

And while doing all the things I also make sure that all the pages should be responsive and correctly structured.

THANK YOU!