Marketplace Hackathon

Day: 4

Completed on
19 January

Prepared by:
Minal Saleem

Building Dynamic Frontend Components for Your Marketplace

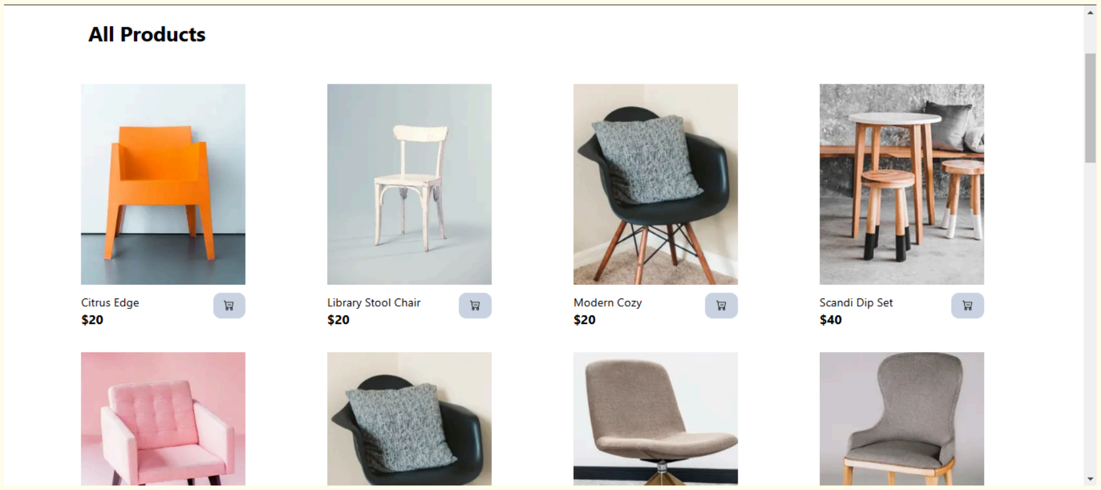
Design and create dynamic frontend components that display marketplace data retrieved from Sanity CMS or APIs, with an emphasis on modular, reusable component design and scalable, responsive web apps.

1: Product Listing Component

Dynamically render products in a **grid layout** with the following fields:

- Product Name

- Price
- Image


All Products

Citrus Edge
$20

Library Stool Chair
$20

Modern Cozy
$20

Scandi Dip Set
$40

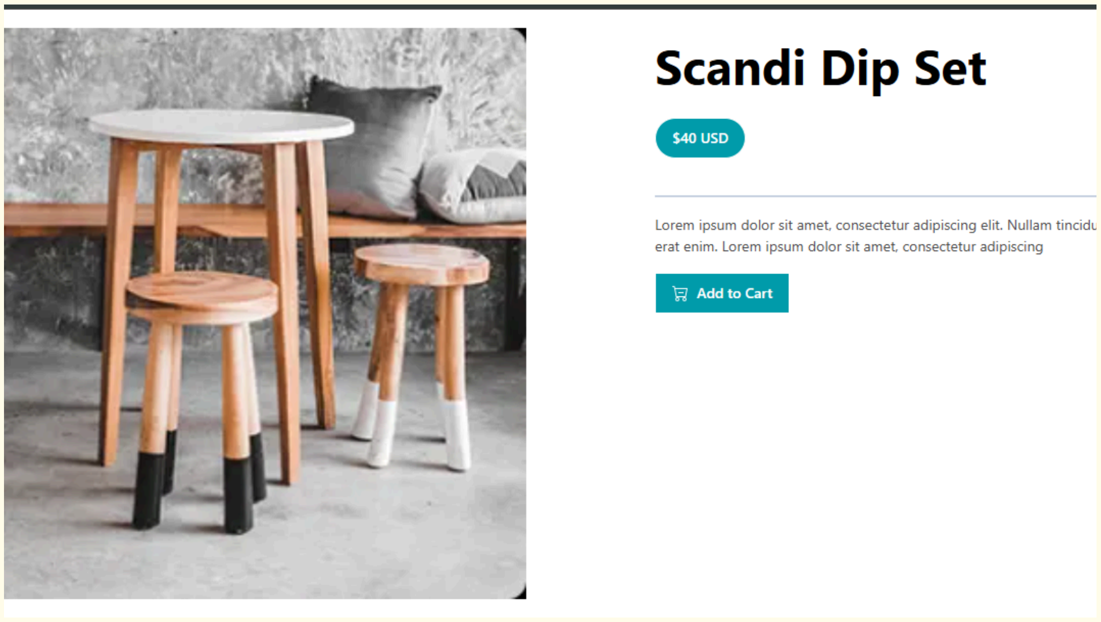
Code snippets of products in a **grid layout:**

```
import React from "react";
import Image from "next/image";
import Link from "next/link";
import { Button } from "@/components/ui/button";
import { BsCartDash } from "react-icons/bs";
import ProductComp from "@/components/(pages)/ProductComp";
import { client } from "@/sanity/lib/client";

export interface Product {
   _id:string
  title:string;
    price:number;
    imageUrl:string;}

export default async function Products() {

  const products:Product[] = await client.fetch(`*[_type ==
"products"]{
    _id,
    title,
    price,
    "imageUrl":image.asset->url,}`
)

  console.log(products);
  return (
    <div>
      <div className="flex flex-row ml-8 lg:ml-[135px] lg:py-14">
        <p className="font-bold text-[32px]">All Products</p>
      </div>

      <div className="mb-32 grid grid-cols-1 sm:grid-cols-1
md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-9 mx-2 xl:mx-24
xl:gap-10">
        {products.map((product, index) => (
          <div
            key={index}
            className="bg-white w-[312px] h-[377px] overflow-hidden
xl:px-7"
          >
            <Image
              src={product.imageUrl}
              alt={product.title}
              width={300}
              height={300}
              className="w-full h-[312px] object-cover"
            />
            <div className="p-0 flex flex-row justify-between mt-3">
              <span>
                <h3 className="text-lg hover:text-[#029FAE] text-black
font-normal">
                  {product.title}
                </h3>
                <p className="text-black text-xl
font-bold">${product.price}</p>
              </span>
              <span className="text-gray-500">
                <Link href={`/products/${product._id}`}>
                <Button variant="outline" className="hover:bg-[#029FAE]
text-black hover:text-white border border-slate-300 bg-slate-300
hover:border rounded-xl"><BsCartDash size={22} /></Button>

                </Link>
              </span>
            </div>
          </div>
        ))}
      </div>

      <ProductComp />
    </div>
  );
}
```

2: Product Detail Component:

Build individual product detail page using **Next.js dynamic routing**. Include fields such as:

- Name
- Price
- Product Description



Scandi Dip Set

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidu
erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

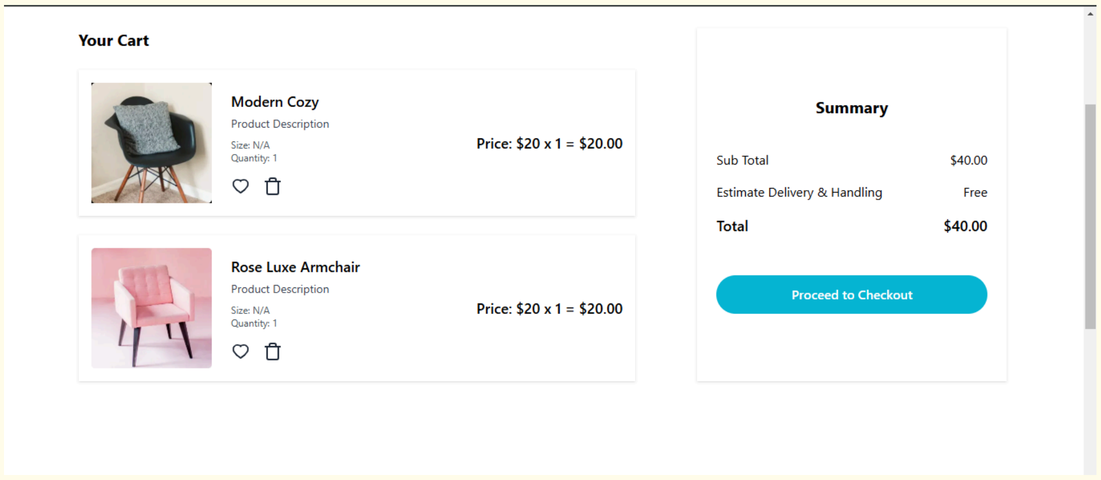
Code snippets of dynamic routing by id:

```
import React from "react";
import DetailComp from "@/components/(pages)/DetailComp";
import { client } from "@/sanity/lib/client";
import { Product } from "../page";
import ProductDetail from "@/components/(pages)/ProductDetail";
```

```
interface Params {
  params: {
    _id: string;
  };
}

const SingleProductDetail = async ({ params }: Params) => {
  const { _id } = await params;

  console.log("Fetching product with id:", _id);

  // Fetch product data from Sanity
  const product: Product | null = await client.fetch(
    `*[_type == "products" && _id == $id][0]{
        _id,
        title,
        price,
        priceWithoutDiscount,
        "imageUrl":image.asset->url,
        description,}
      }`,
    { id:_id }
  );

  if (!product) {
    return <p>Product not found!</p>;
  }

  console.log(product);

  return (
    <div>
      <ProductDetail product={product}/>

      <DetailComp />
    </div>
  );
};

export default SingleProductDetail;
```

3: Cart Component

- Display added items, their quantities, and the total price.
- Use state management to track cart items.



Your Cart

Modern Cozy
Product Description
Size: N/A
Quantity: 1
Price: $20 x 1 = $20.00

Rose Luxe Armchair
Product Description
Size: N/A
Quantity: 1
Price: $20 x 1 = $20.00

Summary

Sub Total	$40.00
Estimate Delivery & Handling	Free
Total	**$40.00**

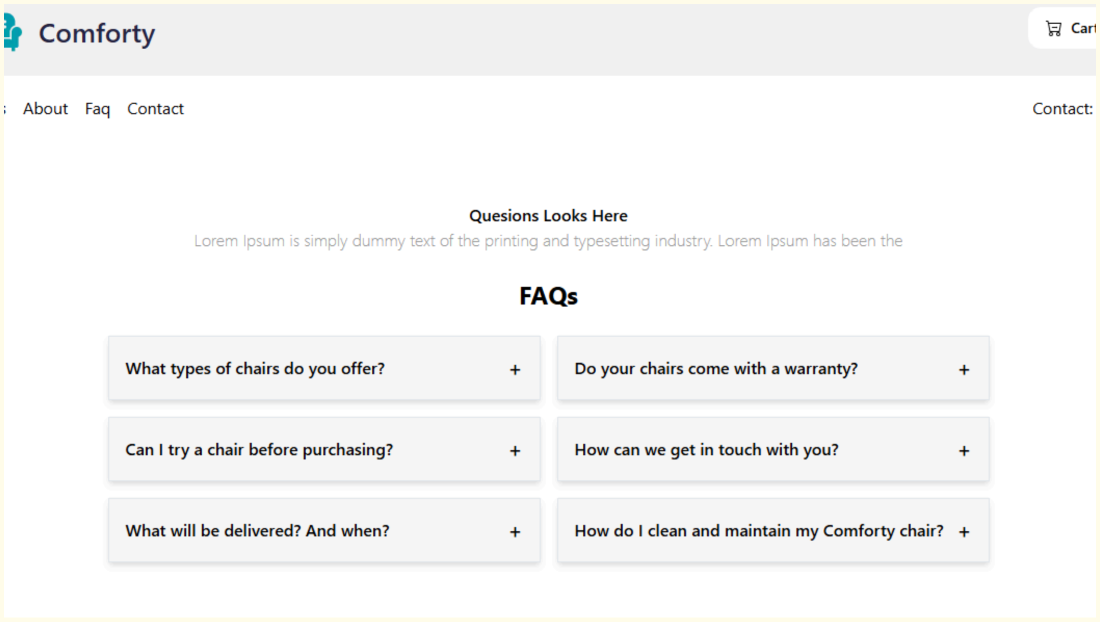
Code snippets of products in a **grid layout**:

```
"use client";

import Image from "next/image";
import { AiOutlineHeart } from "react-icons/ai";
import { FiTrash } from "react-icons/fi";
import { useCart } from "@/components/(pages)/CardContext";

// Define CartItem type
interface CartItem {
  _id: string;
  title: string;
  imageUrl: string;
  price: number;
  quantity: number;
  description?: string;
  size?: string;
}

export default function CartPage() {
  const { state, dispatch } = useCart();

  if (!state || !dispatch) {
    throw new Error("Cart context is not properly initialized.");
  }
  return (
    <div className="container mx-auto p-4 md:p-8 lg:px-24 lg:py-12
max-w-screen-2xl m-auto">
      <div className="flex flex-col md:flex-row justify-between gap-8
md:gap-10">
        <div className="w-full md:w-3/5">
          <h1 className="text-2xl md:text-xl font-bold mb-6">Your
Cart</h1>
          {state.items.length === 0 ? (
            <p className="text-lg">Your cart is empty.</p>
          ) : (
            <div className="flex flex-col gap-6">
              {state.items.map((item:CartItem, i:number) => (
                <div
                  key={i}
                  className="flex flex-col md:flex-row items-center
md:justify-between p-4 bg-white shadow rounded-lg"
                >
                  <div className="flex items-center gap-6">
                    <Image
                      src={item.imageUrl}
                      alt={item.title}
                      className="rounded-lg"
                      width={150}
                      height={150}
                      sizes="(max-width: 768px) 100px, 150px"
                    />
                    <div>
                      <h1 className="font-semibold text-xl md:text-lg">
                        {item.title}
                      </h1>
                      <h2 className="text-base md:text-sm text-gray-700
mt-1">
                        {item.description || "Product Description"}
                      </h2>
                      <div className="text-sm md:text-xs text-gray-600
mt-2">
                        <p>Size: {item.size || "N/A"}</p>
                        <p>Quantity: {item.quantity}</p>
                      </div>
                      <div className="flex gap-4 mt-4 text-gray-800">
```

```
                        <AiOutlineHeart className="text-2xl
cursor-pointer hover:text-red-500" />
                        <FiTrash
                          className="text-2xl cursor-pointer
hover:text-gray-500"
                          onClick={() =>
                            dispatch({
                              type: "REMOVE_FROM_CART",
                              payload: { _id: item._id },
                            })
                          }
                        />
                      </div>
                    </div>
                  </div>
                  <div className="text-xl md:text-lg font-semibold mt-4
md:mt-0">
                    Price: ${item.price} x {item.quantity} = $
                    {(item.price * item.quantity).toFixed(2)}
                  </div>
                </div>
              ))}
            </div>
          )}
        </div>

        <div className="w-full md:w-1/3 flex flex-col justify-center
items-center gap-6 bg-white shadow rounded-lg p-6">
          <h1 className="text-2xl md:text-xl font-bold
mb-4">Summary</h1>
          <div className="w-full space-y-4">
            <div className="flex justify-between text-lg md:text-base">
              <span>Sub Total</span>
              <span>
                $
                {state.items
                  .reduce((total, item) => total + item.price *
item.quantity, 0)
                  .toFixed(2)}
              </span>
            </div>
            <div className="flex justify-between text-lg md:text-base">
              <span>Estimate Delivery & Handling</span>
              <span>Free</span>
            </div>
            <div className="flex justify-between text-xl md:text-lg
font-semibold">
              <span>Total</span>
              <span>
                $
                {state.items
                  .reduce((total, item) => total + item.price *
item.quantity, 0)
                  .toFixed(2)}
              </span>
            </div>
          </div>
          <button className="w-full text-lg md:text-base font-semibold
text-white bg-cyan-500 rounded-full py-3 mt-6 hover:bg-cyan-600
transition">
            Proceed to Checkout
          </button>
        </div>
      </div>
    </div>
  );
}
```

4: FAQ Component:

Include a searchable FAQ section.



Quesions Looks Here

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the

FAQs

What types of chairs do you offer? +	Do your chairs come with a warranty? +
Can I try a chair before purchasing? +	How can we get in touch with you? +
What will be delivered? And when? +	How do I clean and maintain my Comforty chair? +

Code snippets of dynamic routing by id:

```
import React from "react";
import DetailComp from "@/components/(pages)/DetailComp";
import { client } from "@/sanity/lib/client";
import { Product } from "../page";
import ProductDetail from "@/components/(pages)/ProductDetail";

interface Params {
  params: {
    _id: string;
  };
}

const SingleProductDetail = async ({ params }: Params) => {
  const { _id } = await params;

  console.log("Fetching product with id:", _id);

  // Fetch product data from Sanity
  const product: Product | null = await client.fetch(
    `*[_type == "products" && _id == $id][0]{
        _id,
        title,
        price,
        priceWithoutDiscount,
        "imageUrl":image.asset->url,
        description,}
      }`,
    { id:_id }
  );

  if (!product) {
    return <p>Product not found!</p>;
  }

  console.log(product);

  return (
    <div>
      <ProductDetail product={product}/>

      <DetailComp />
    </div>
  );
};

export default SingleProductDetail;
```