MARKETPLACE BUSSINESS PLAN DAY 4

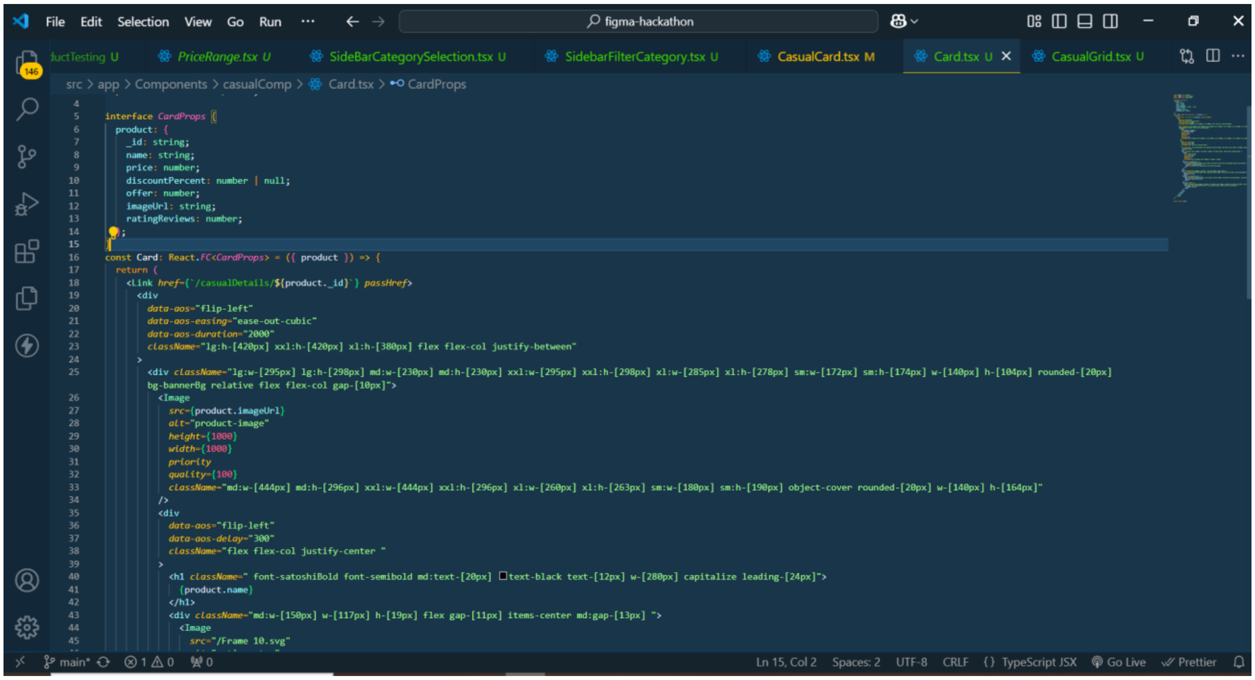
Building Dynamic Frontend Components for my Marketplace – SHOP.CO

In last 3 days, I've focused on how we can plan our marketplace and how we can start our e-commerce business after that all concepts I'm going to create my hackathon in dynamic way how we can do dynamic routing , product listing and many other components. For this I used dynamic and modular components which is reusable. So here is my essential components for my hackathon . I'll explain how I created components and I'll also share my code snippets. Here is breakdown of my components;

Building key Components

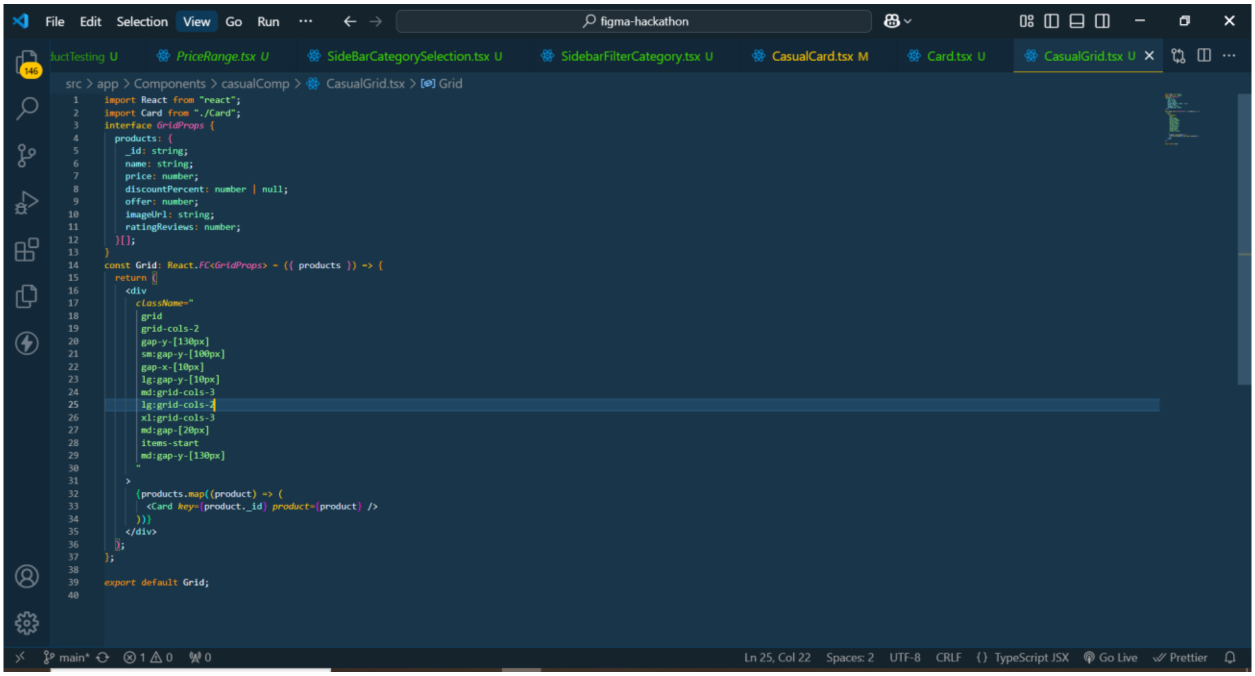
1. Product Listing Page:

I've build product listing page dynamically where I've used grid layout for the professional work flow;



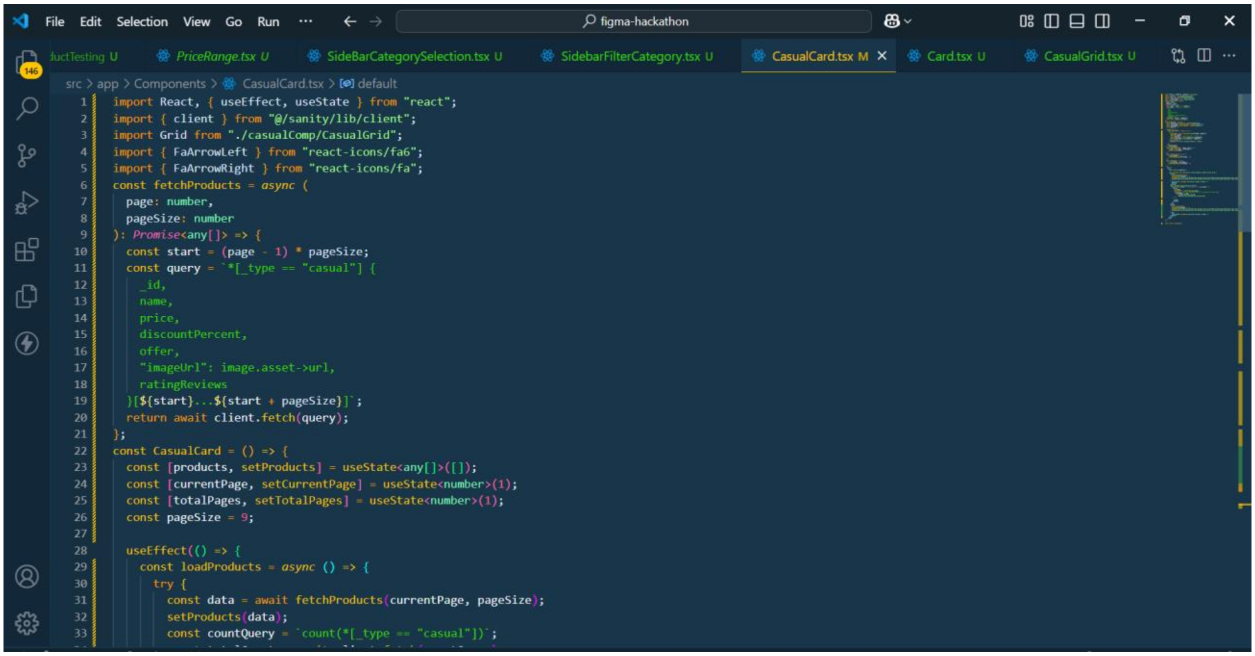
```tsx

interface CardProps {
  product: {
    _id: string;
    name: string;
    price: number;
    discountPercent: number | null;
    offer: number;
    imageUrl: string;
    ratingReviews: number;
  };
}
const Card: React.FC<CardProps> = ({ product }) => {
  return (
    <Link href={`/casualDetails/${product._id}`} passHref>
      <div
        data-aos="flip-left"
        data-aos-easing="ease-out-cubic"
        data-aos-duration="2000"
        className="lg:h-[420px] xxl:h-[420px] xl:h-[380px] flex flex-col justify-between"
      >
        <div className="lg:w-[295px] lg:h-[298px] md:w-[230px] md:h-[230px] xxl:w-[295px] xxl:h-[298px] xl:w-[285px] xl:h-[278px] sm:w-[172px] sm:h-[174px] w-[140px] h-[104px] rounded-[20px]
        bg-bannerBg relative flex flex-col gap-[10px]">
          <Image
            src={product.imageUrl}
            alt="product-image"
            height={1000}
            width={1000}
            priority
            quality={100}
            className="md:w-[444px] md:h-[296px] xxl:w-[444px] xxl:h-[296px] xl:w-[260px] xl:h-[263px] sm:w-[180px] sm:h-[190px] object-cover rounded-[20px] w-[140px] h-[164px]"
          />
          <div
            data-aos="flip-left"
            data-aos-delay="300"
            className="flex flex-col justify-center "
          >
            <h1 className=" font-satoshiBold font-semibold md:text-[20px] text-black text-[12px] w-[280px] capitalize leading-[24px]">
              {product.name}
            </h1>
            <div className="md:w-[150px] w-[117px] h-[19px] flex gap-[11px] items-center md:gap-[13px] ">
              <Image
                src="/Frame 10.svg"
```

In **card** component I've design my card style where I've style my image, name, price etc...


File Edit Selection View Go Run
figma-hackathon
ductTesting U
PriceRange.tsx U
SideBarCategorySelection.tsx U
SidebarFilterCategory.tsx U
CasualCard.tsx M
Card.tsx U
CasualGrid.tsx U
src > app > Components > casualComp > CasualGrid.tsx > Grid
main* 1 0 0
Ln 25, Col 22 Spaces: 2 UTF-8 CRLF TypeScript JSX Go Live Prettier

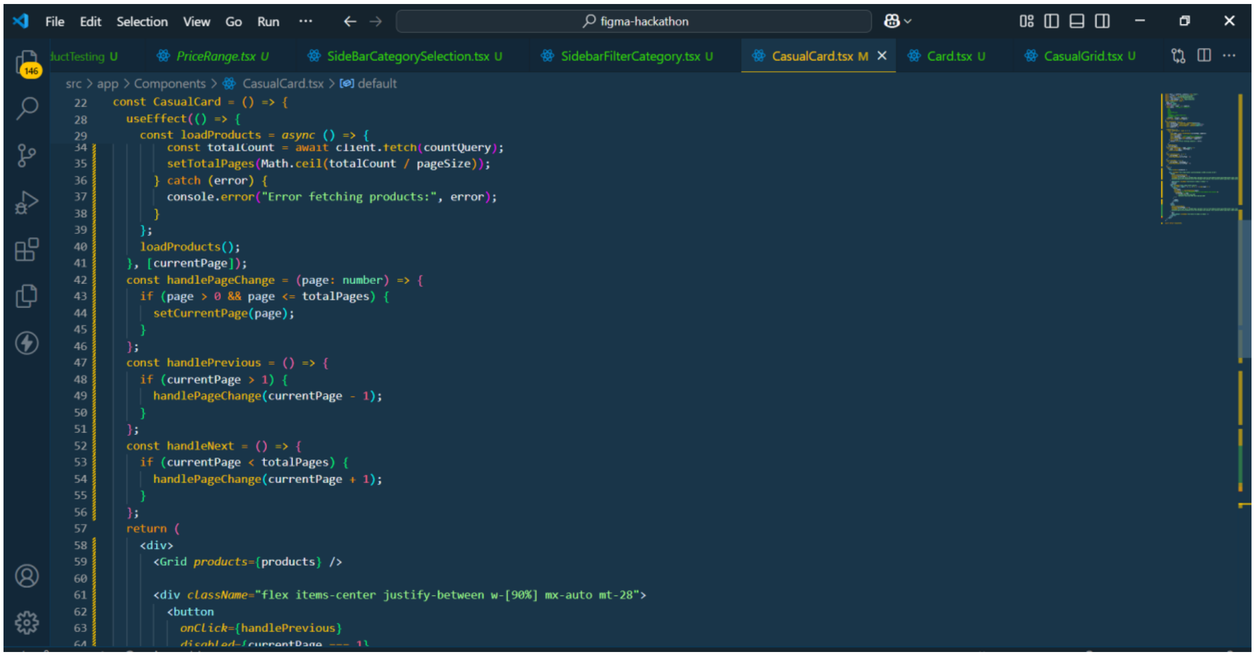
```tsx
import React from "react";
import Card from "./Card";
interface GridProps {
  products: {
    _id: string;
    name: string;
    price: number;
    discountPercent: number | null;
    offer: number;
    imageUrl: string;
    ratingReviews: number;
  }[];
}
const Grid: React.FC<GridProps> = ({ products }) => {
  return (
    <div
      className="
       grid
       grid-cols-2
       gap-y-[130px]
       sm:gap-y-[100px]
       gap-x-[10px]
       lg:gap-y-[10px]
       md:grid-cols-3
       lg:grid-cols-2
       xl:grid-cols-3
       md:gap-[20px]
       items-start
       md:gap-y-[130px]
      "
    >
      {products.map((product) => (
        <Card key={product._id} product={product} />
      ))}
    </div>
  );
};

export default Grid;

```

In this Grid component I've used grid layout for responsiveness and also I've used props for product displaying and used map function for fetch all products data.


File Edit Selection View Go Run
figma-hackathon
ductTesting U
PriceRange.tsx U
SideBarCategorySelection.tsx U
SidebarFilterCategory.tsx U
CasualCard.tsx M
Card.tsx U
CasualGrid.tsx U
src > app > Components > CasualCard.tsx > default

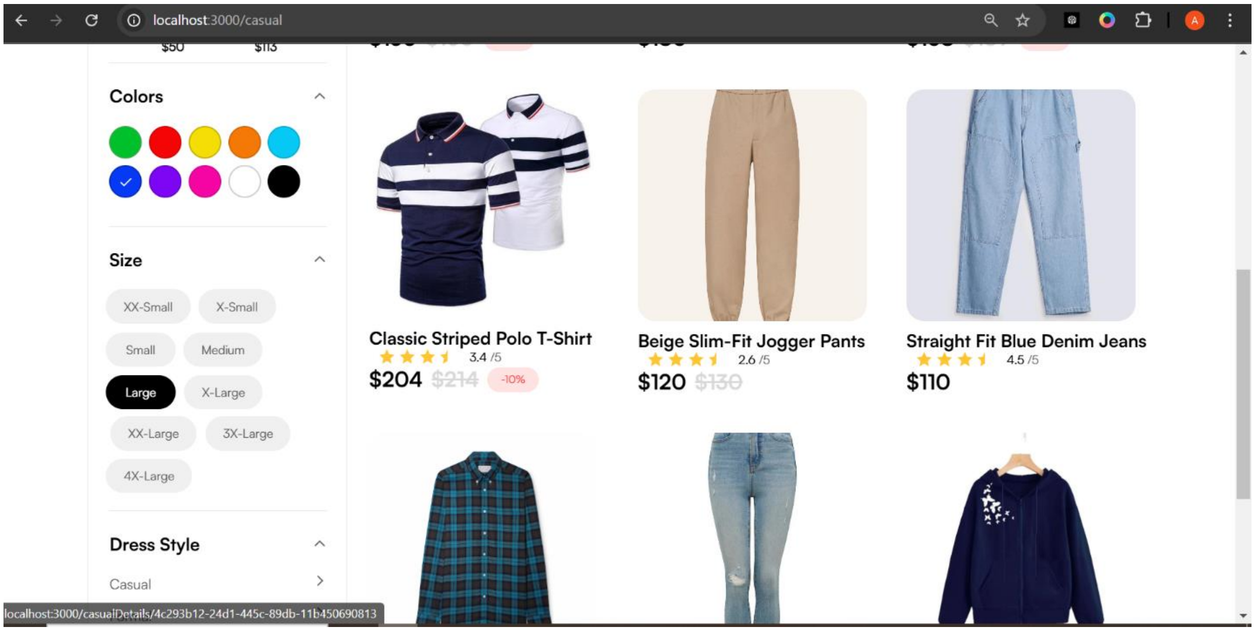
```tsx
import React, { useEffect, useState } from "react";
import { client } from "@/sanity/lib/client";
import Grid from "./casualComp/CasualGrid";
import { FaArrowLeft } from "react-icons/fa6";
import { FaArrowRight } from "react-icons/fa";
const fetchProducts = async (
  page: number,
  pageSize: number
): Promise<any[]> => {
  const start = (page - 1) * pageSize;
  const query = `*[_type == "casual"] {
    _id,
    name,
    price,
    discountPercent,
    offer,
    "imageUrl": image.asset->url,
    ratingReviews
  }[${start}...${start + pageSize}]`;
  return await client.fetch(query);
};
const CasualCard = () => {
  const [products, setProducts] = useState<any[]>([]);
  const [currentPage, setCurrentPage] = useState<number>(1);
  const [totalPages, setTotalPages] = useState<number>(1);
  const pageSize = 9;

  useEffect(() => {
    const loadProducts = async () => {
      try {
        const data = await fetchProducts(currentPage, pageSize);
        setProducts(data);
        const countQuery = `count(*[_type == "casual"])`;
```

In CasualCard component I've used Groq query for fetching data from sanity CMS and it is displaying dynamically.


File Edit Selection View Go Run ···
figma-hackathon
ductTesting U
PriceRange.tsx U
SideBarCategorySelection.tsx U
SidebarFilterCategory.tsx U
CasualCard.tsx M
Card.tsx U
CasualGrid.tsx U
146
src > app > Components > CasualCard.tsx > default

```tsx
const CasualCard = () => {
  useEffect(() => {
    const loadProducts = async () => {
        const totalCount = await client.fetch(countQuery);
        setTotalPages(Math.ceil(totalCount / pageSize));
      } catch (error) {
        console.error("Error fetching products:", error);
      }
    };
    loadProducts();
  }, [currentPage]);
  const handlePageChange = (page: number) => {
    if (page > 0 && page <= totalPages) {
      setCurrentPage(page);
    }
  };
  const handlePrevious = () => {
    if (currentPage > 1) {
      handlePageChange(currentPage - 1);
    }
  };
  const handleNext = () => {
    if (currentPage < totalPages) {
      handlePageChange(currentPage + 1);
    }
  };
  return (
    <div>
      <Grid products={products} />

      <div className="flex items-center justify-between w-[90%] mx-auto mt-28">
        <button
          onClick={handlePrevious}
          disabled={currentPage === 1}
```

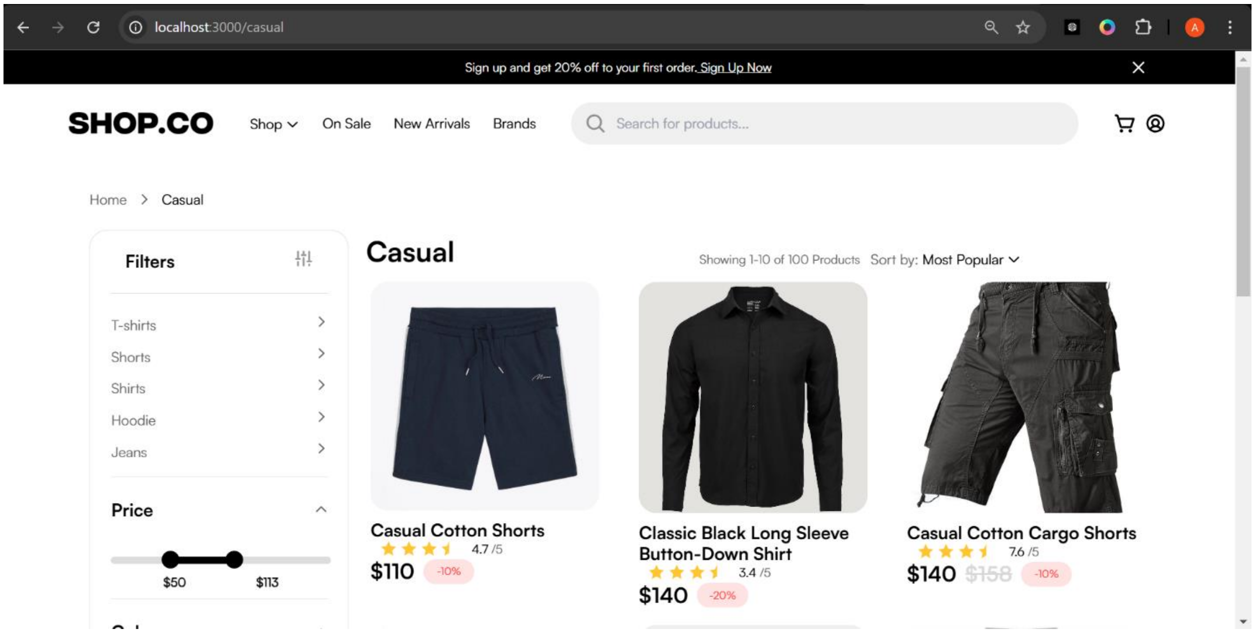
So in short in this step I’ve fetch data dynamically form sanity CMS and using groq query and grid layout for responsiveness.

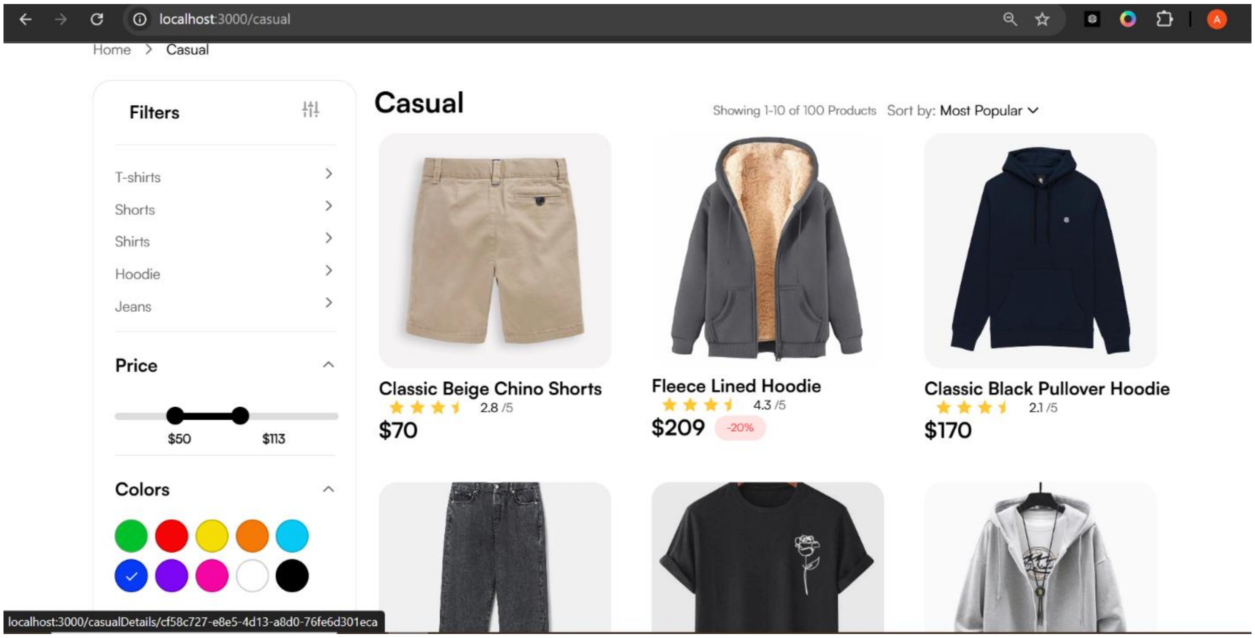


Classic Striped Polo T-Shirt
3.4 /5
$204 ~~$214~~ -10%

Beige Slim-Fit Jogger Pants
2.6 /5
$120 ~~$130~~

Straight Fit Blue Denim Jeans
4.5 /5
$110

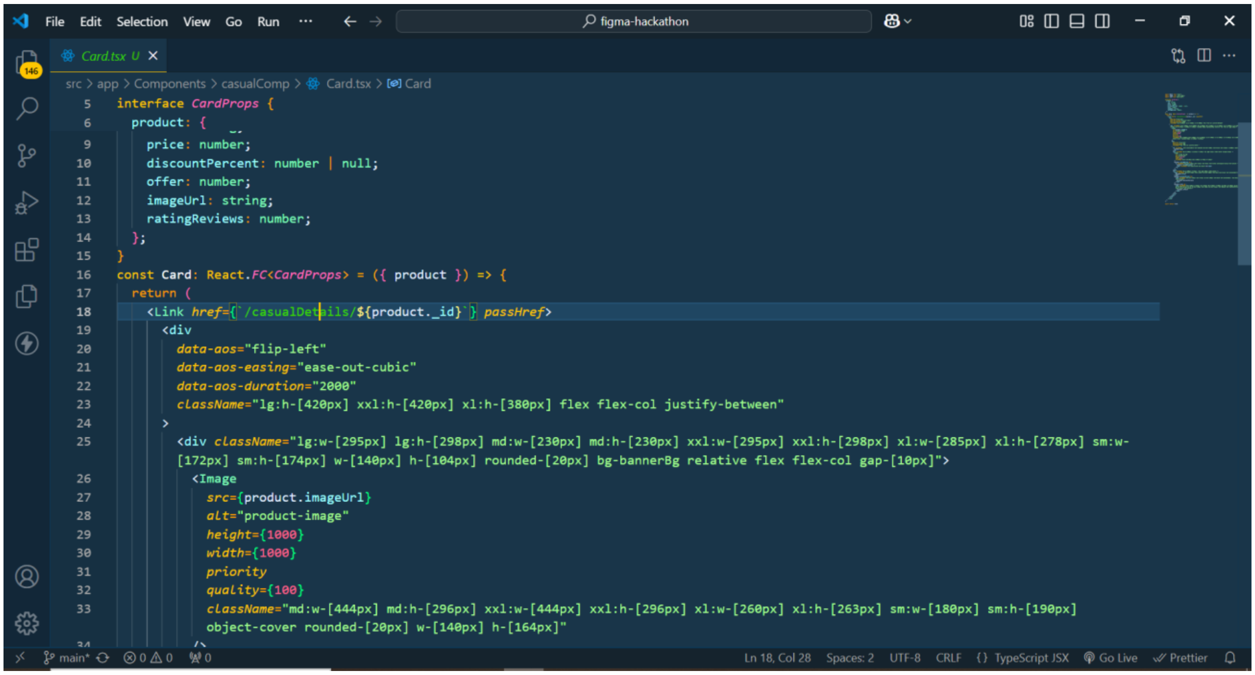




This is my frontend output these all products I've fetched from sanity and displayed on the frontend.

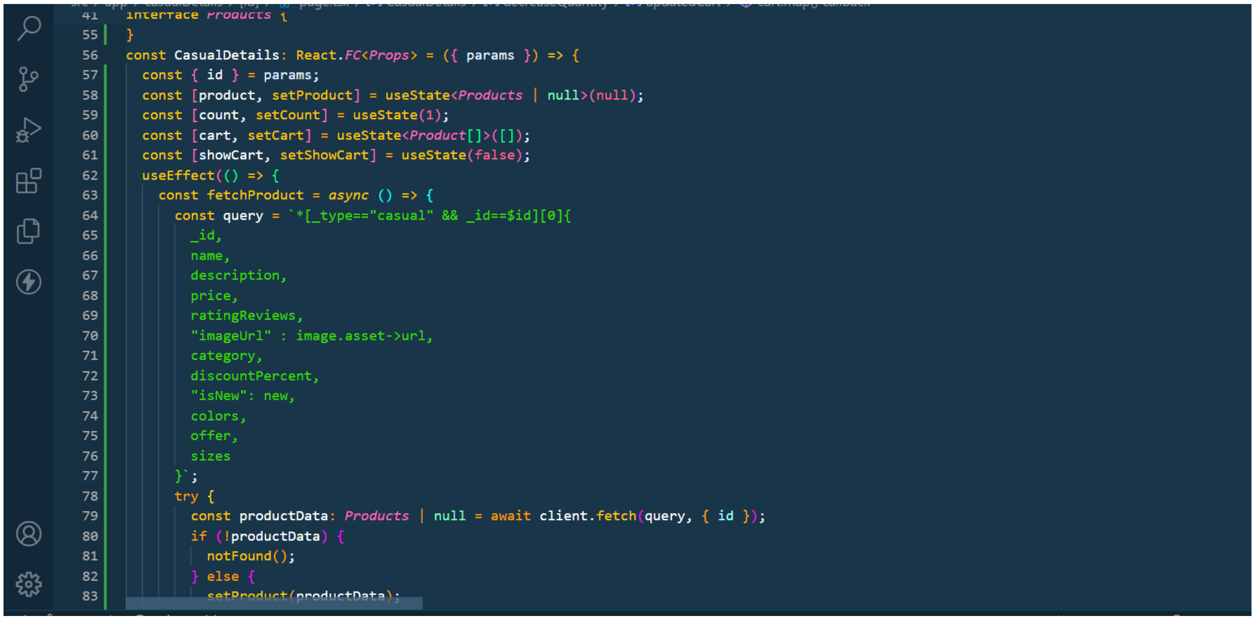
2. Product Detailed Component:

After fetching multiple products from sanity CMS, now I'm going to create single product page by using dynamic routing where user can visit further detail of product like product description, size, color, reviews etc...


File Edit Selection View Go Run ···
figma-hackathon
Card.tsx U
src > app > Components > casualComp > Card.tsx > Card
main* ⊗ 0 ⚠ 0 0
Ln 18, Col 28 Spaces: 2 UTF-8 CRLF {} TypeScript JSX Go Live Prettier

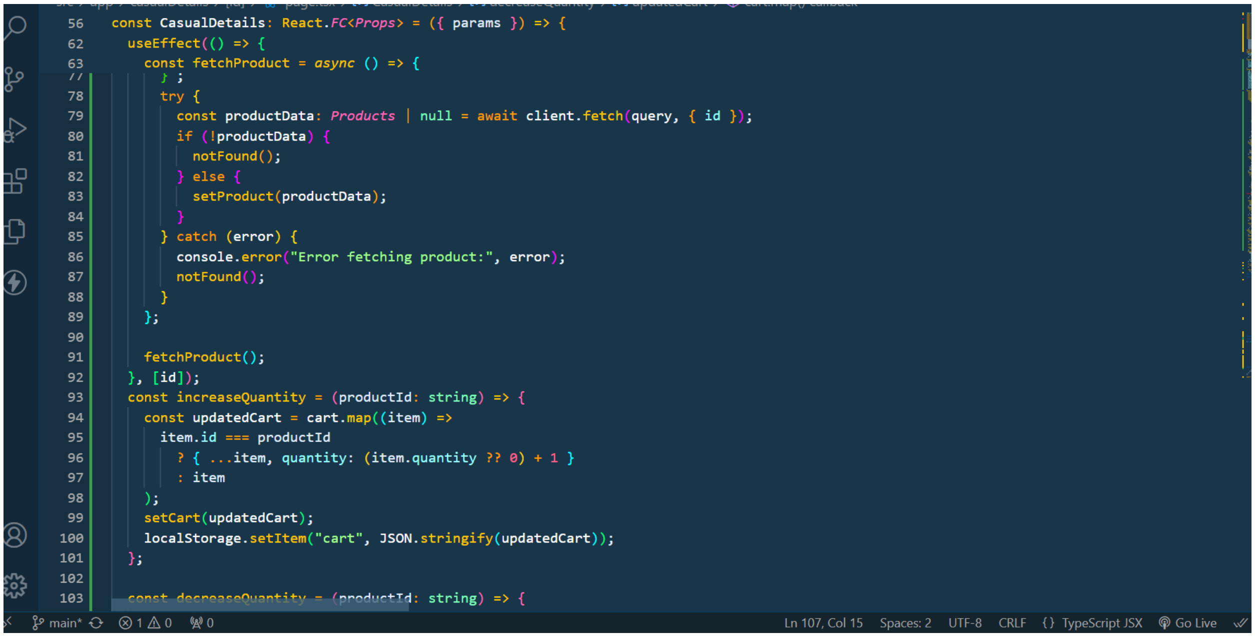
```tsx
interface CardProps {
  product: {
    price: number;
    discountPercent: number | null;
    offer: number;
    imageUrl: string;
    ratingReviews: number;
  };
}
const Card: React.FC<CardProps> = ({ product }) => {
  return (
    <Link href={`/casualDetails/${product._id}`} passHref>
      <div
        data-aos="flip-left"
        data-aos-easing="ease-out-cubic"
        data-aos-duration="2000"
        className="lg:h-[420px] xxl:h-[420px] xl:h-[380px] flex flex-col justify-between"
      >
        <div className="lg:w-[295px] lg:h-[298px] md:w-[230px] md:h-[230px] xxl:w-[295px] xxl:h-[298px] xl:w-[285px] xl:h-[278px] sm:w-[172px] sm:h-[174px] w-[140px] h-[104px] rounded-[20px] bg-bannerBg relative flex flex-col gap-[10px]">
          <Image
            src={product.imageUrl}
            alt="product-image"
            height={1000}
            width={1000}
            priority
            quality={100}
            className="md:w-[444px] md:h-[296px] xxl:w-[444px] xxl:h-[296px] xl:w-[260px] xl:h-[263px] sm:w-[180px] sm:h-[190px] object-cover rounded-[20px] w-[140px] h-[164px]"
```

So firstly in my card component I used link component for navigation I've passed **casualDetails** on the basis of **id.**



```tsx
interface Products {
}
const CasualDetails: React.FC<Props> = ({ params }) => {
  const { id } = params;
  const [product, setProduct] = useState<Products | null>(null);
  const [count, setCount] = useState(1);
  const [cart, setCart] = useState<Product[]>([]);
  const [showCart, setShowCart] = useState(false);
  useEffect(() => {
    const fetchProduct = async () => {
      const query = `*[_type=="casual" && _id==$id][0]{
        _id,
        name,
        description,
        price,
        ratingReviews,
        "imageUrl" : image.asset->url,
        category,
        discountPercent,
        "isNew": new,
        colors,
        offer,
        sizes
      }`;
      try {
        const productData: Products | null = await client.fetch(query, { id });
        if (!productData) {
          notFound();
        } else {
          setProduct(productData);
```

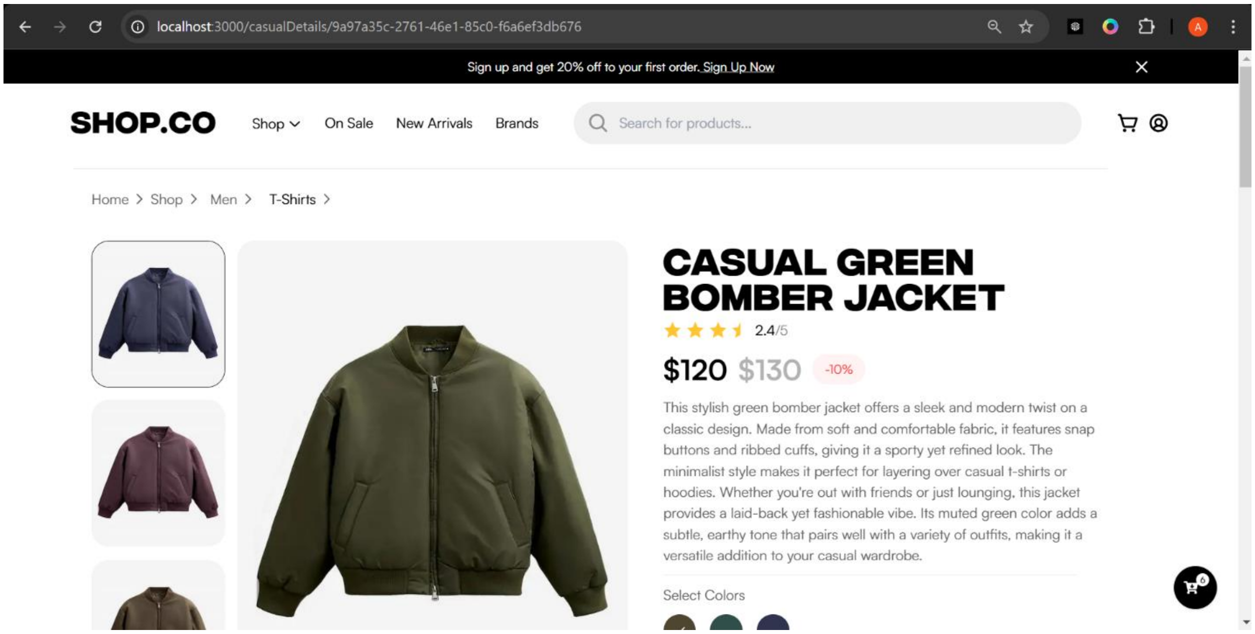
Here is my dynamic routing route where user can visit detailed for product I've used id for the dynamic routing on the basis of id I've used dynamic routing.



```tsx
const CasualDetails: React.FC<Props> = ({ params }) => {
  useEffect(() => {
    const fetchProduct = async () => {
      } ;
      try {
        const productData: Products | null = await client.fetch(query, { id });
        if (!productData) {
          notFound();
        } else {
          setProduct(productData);
        }
      } catch (error) {
        console.error("Error fetching product:", error);
        notFound();
      }
    };

    fetchProduct();
  }, [id]);
  const increaseQuantity = (productId: string) => {
    const updatedCart = cart.map((item) =>
      item.id === productId
        ? { ...item, quantity: (item.quantity ?? 0) + 1 }
        : item
    );
    setCart(updatedCart);
    localStorage.setItem("cart", JSON.stringify(updatedCart));
  };

  const decreaseQuantity = (productId: string) => {
```

main* ⓧ 1 ⚠ 0 0
Ln 107, Col 15 Spaces: 2 UTF-8 CRLF {} TypeScript JSX Go Live

And here I've used try and catch for error handling if user buy product which is available in my stock then my try method will implemented otherwise the error with through because of unavailability of product.



localhost:3000/casualDetails/9a97a35c-2761-46e1-85c0-f6a6ef3db676

Sign up and get 20% off to your first order. Sign Up Now

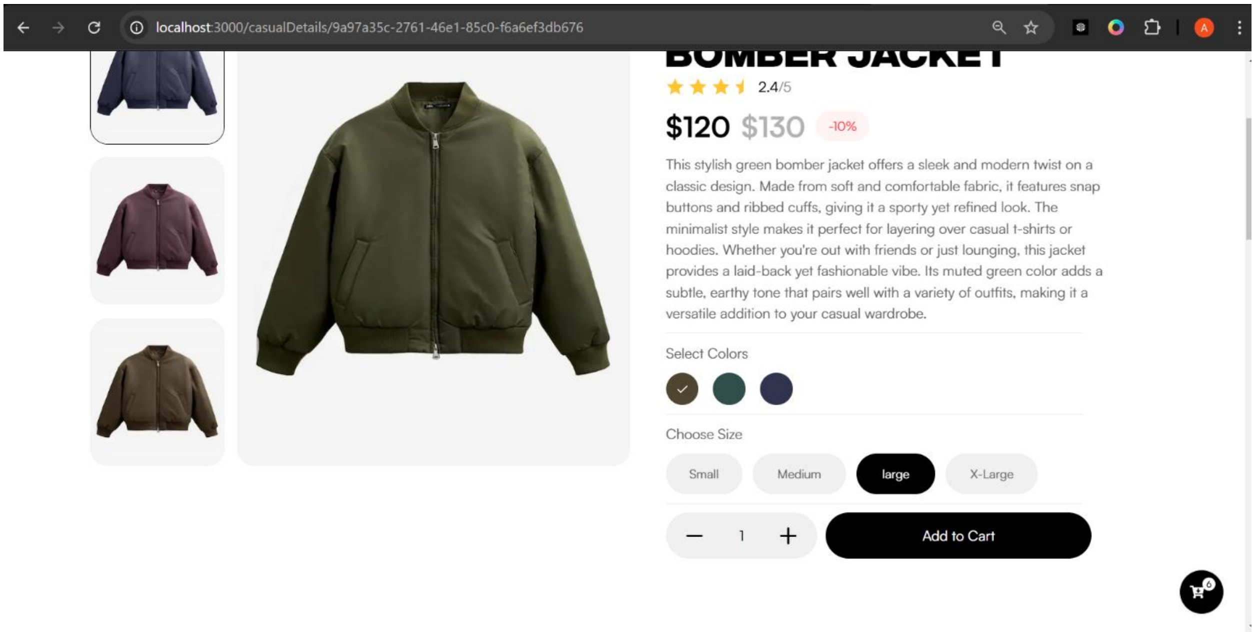
CASUAL GREEN BOMBER JACKET

2.4/5

$120 $130 -10%

This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a sporty yet refined look. The minimalist style makes it perfect for layering over casual t-shirts or hoodies. Whether you're out with friends or just lounging, this jacket provides a laid-back yet fashionable vibe. Its muted green color adds a subtle, earthy tone that pairs well with a variety of outfits, making it a versatile addition to your casual wardrobe.

Select Colors



localhost:3000/casualDetails/9a97a35c-2761-46e1-85c0-f6a6ef3db676

BOMBER JACKET

2.4/5

$120 $130 -10%

This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a sporty yet refined look. The minimalist style makes it perfect for layering over casual t-shirts or hoodies. Whether you're out with friends or just lounging, this jacket provides a laid-back yet fashionable vibe. Its muted green color adds a subtle, earthy tone that pairs well with a variety of outfits, making it a versatile addition to your casual wardrobe.

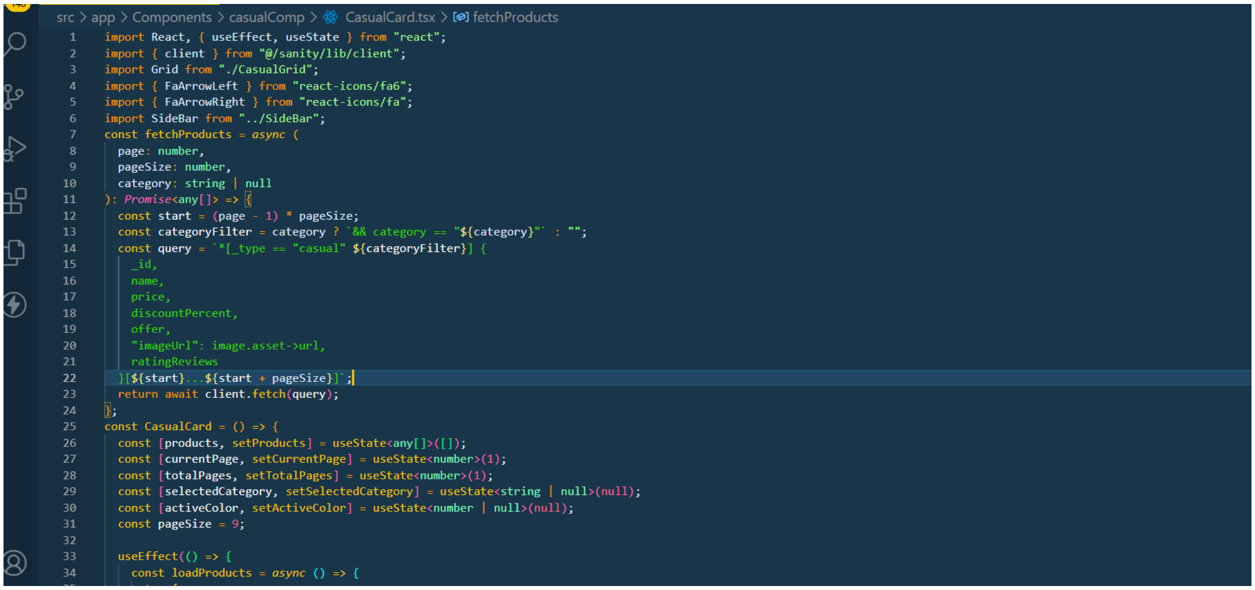
Select Colors

Choose Size

So when user click on the card, user will navigate the single product details page on the basis of unique id. So now in step 2 I've implemented my dynamic routing successfully.

3. Category Component:

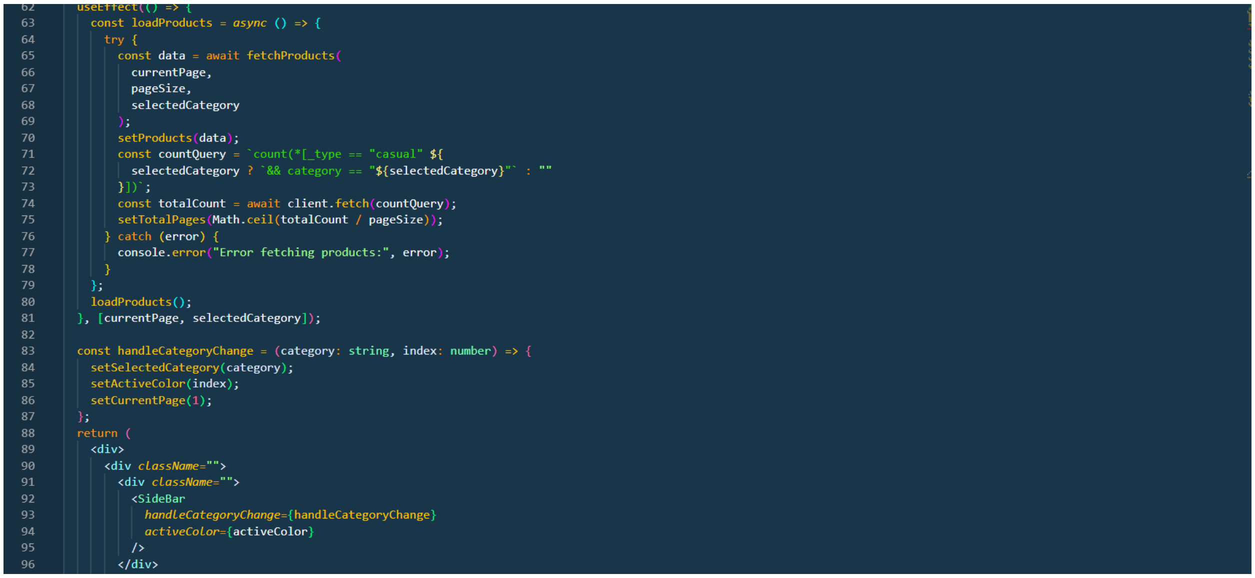
After dynamic routing I've to implement **Category** component. Firstly I've to used sanity groq query because the data is stored in sanity then I've to fetch category by using groq query. For this I've use State for selected category and then I've used groq query



src > app > Components > casualComp > CasualCard.tsx > fetchProducts

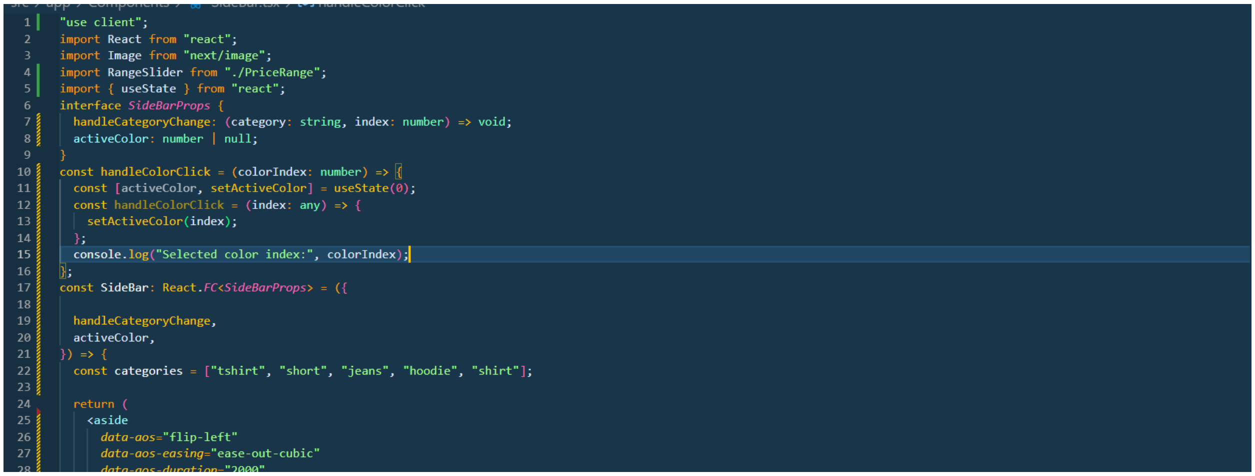
```
import React, { useEffect, useState } from "react";
import { client } from "@/sanity/lib/client";
import Grid from "./CasualGrid";
import { FaArrowLeft } from "react-icons/fa6";
import { FaArrowRight } from "react-icons/fa";
import SideBar from "../SideBar";
const fetchProducts = async (
  page: number,
  pageSize: number,
  category: string | null
): Promise<any[]> => {
  const start = (page - 1) * pageSize;
  const categoryFilter = category ? `&& category == "${category}"` : "";
  const query = `*[_type == "casual" ${categoryFilter}] {
    _id,
    name,
    price,
    discountPercent,
    offer,
    "imageUrl": image.asset->url,
    ratingReviews
  }[${start}...${start + pageSize}]`;
  return await client.fetch(query);
};
const CasualCard = () => {
  const [products, setProducts] = useState<any[]>([]);
  const [currentPage, setCurrentPage] = useState<number>(1);
  const [totalPages, setTotalPages] = useState<number>(1);
  const [selectedCategory, setSelectedCategory] = useState<string | null>(null);
  const [activeColor, setActiveColor] = useState<number | null>(null);
  const pageSize = 9;

  useEffect(() => {
    const loadProducts = async () => {
```

Here is my code snippet you can see how functionality work.



```
  useEffect(() => {
    const loadProducts = async () => {
      try {
        const data = await fetchProducts(
          currentPage,
          pageSize,
          selectedCategory
        );
        setProducts(data);
        const countQuery = `count(*[_type == "casual" ${
          selectedCategory ? `&& category == "${selectedCategory}"` : ""
        }])`;
        const totalCount = await client.fetch(countQuery);
        setTotalPages(Math.ceil(totalCount / pageSize));
      } catch (error) {
        console.error("Error fetching products:", error);
      }
    };
    loadProducts();
  }, [currentPage, selectedCategory]);

  const handleCategoryChange = (category: string, index: number) => {
    setSelectedCategory(category);
    setActiveColor(index);
    setCurrentPage(1);
  };
  return (
    <div>
      <div className="">
        <div className="">
          <SideBar
            handleCategoryChange={handleCategoryChange}
            activeColor={activeColor}
          />
        </div>
```

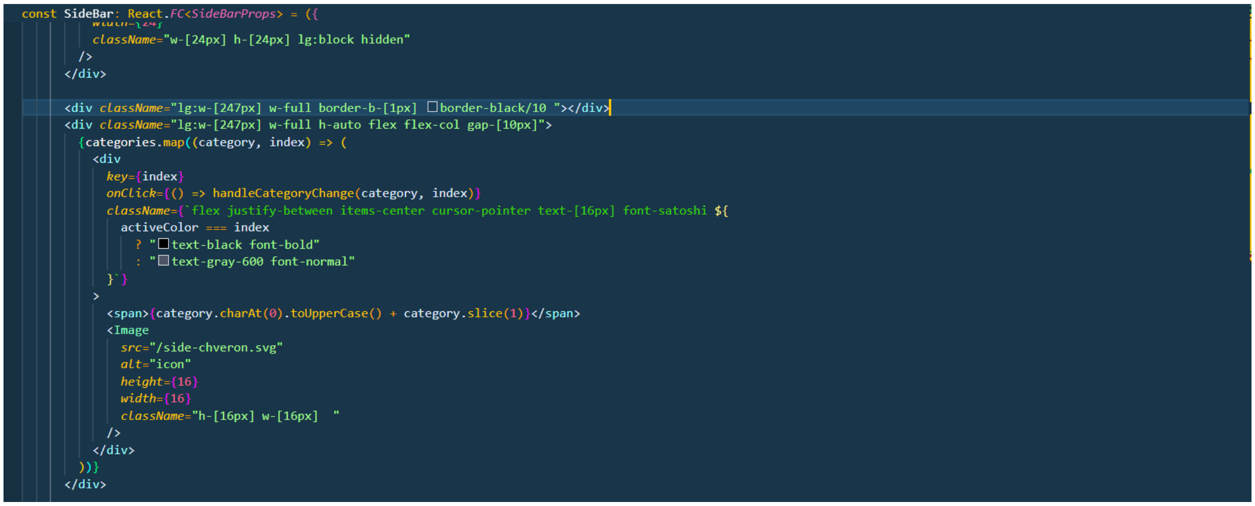
For reusability I created **SideBar** component and then I’ve passed props.



src > app > Components > SideBar.tsx > handleColorClick

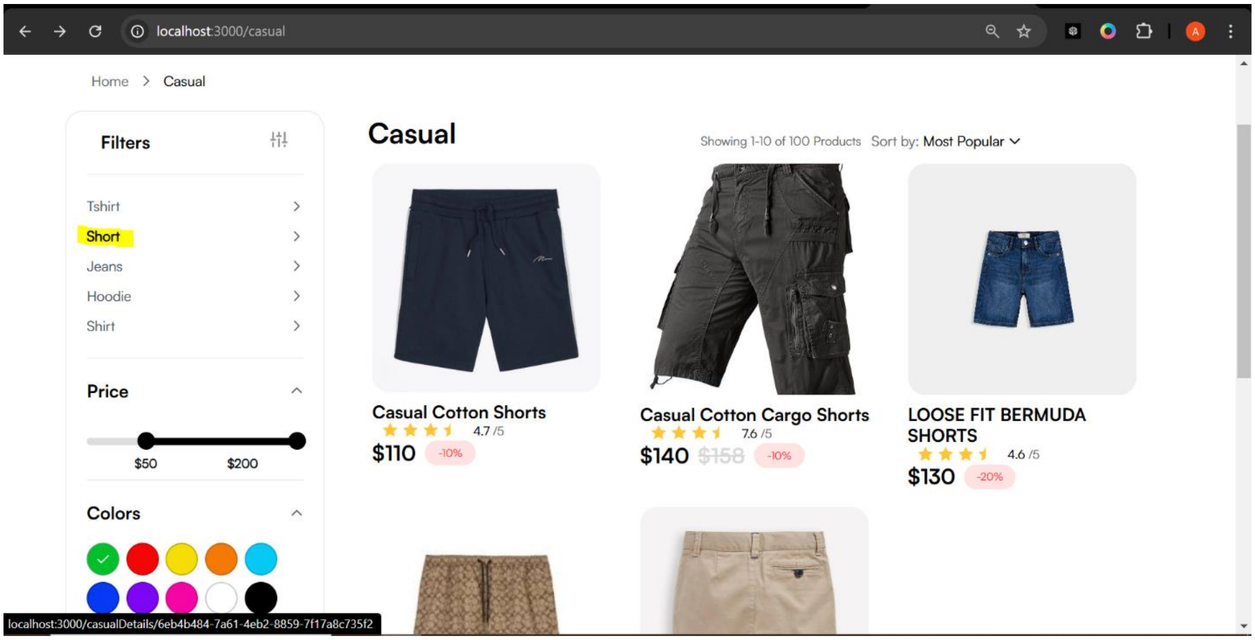
```
"use client";
import React from "react";
import Image from "next/image";
import RangeSlider from "./PriceRange";
import { useState } from "react";
interface SideBarProps {
  handleCategoryChange: (category: string, index: number) => void;
  activeColor: number | null;
}
const handleColorClick = (colorIndex: number) => {
  const [activeColor, setActiveColor] = useState(0);
  const handleColorClick = (index: any) => {
    setActiveColor(index);
  };
  console.log("Selected color index:", colorIndex);
};
const SideBar: React.FC<SideBarProps> = ({

  handleCategoryChange,
  activeColor,
}) => {
  const categories = ["tshirt", "short", "jeans", "hoodie", "shirt"];

  return (
    <aside
      data-aos="flip-left"
      data-aos-easing="ease-out-cubic"
      data-aos-duration="2000"
```

This Is my sidebar component where i used props an array for categories.

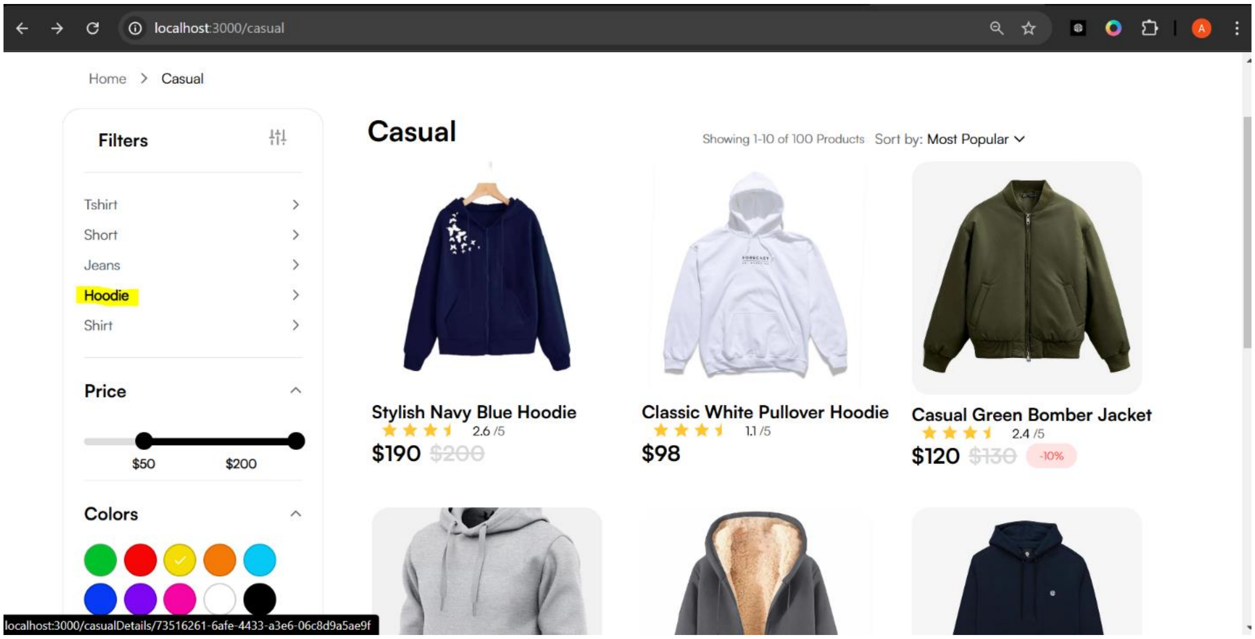


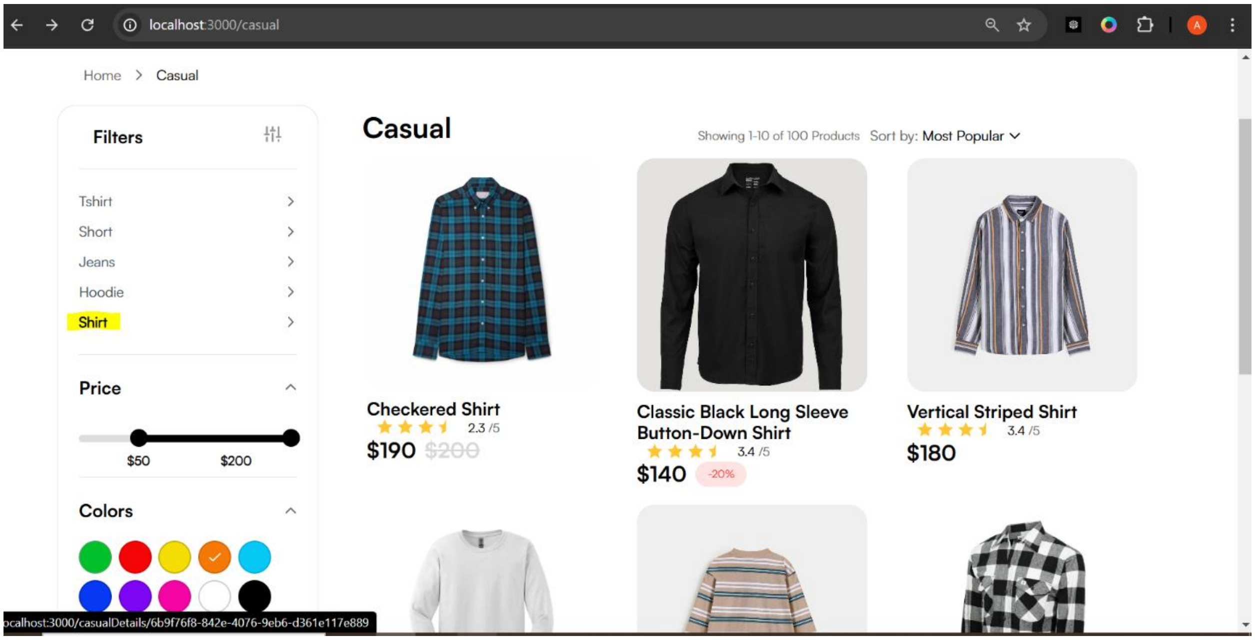
```
const SideBar: React.FC<SideBarProps> = ({
            width={24}
            className="w-[24px] h-[24px] lg:block hidden"
          />
        </div>

        <div className="lg:w-[247px] w-full border-b-[1px] border-black/10 "></div>
        <div className="lg:w-[247px] w-full h-auto flex flex-col gap-[10px]">
          {categories.map((category, index) => (
            <div
              key={index}
              onClick={() => handleCategoryChange(category, index)}
              className={`flex justify-between items-center cursor-pointer text-[16px] font-satoshi ${
                activeColor === index
                  ? "text-black font-bold"
                  : "text-gray-600 font-normal"
              }`}
            >
              <span>{category.charAt(0).toUpperCase() + category.slice(1)}</span>
              <Image
                src="/side-chveron.svg"
                alt="icon"
                height={16}
                width={16}
                className="h-[16px] w-[16px]  "
              />
            </div>
          ))}
        </div>
```

And then using map function I’ve displayed my all categories dynamically.



So here is my frontend where user can easily navigate through any category .





So I’ve completed my step 3 successfully. In this step I’ve created dynamic category component.