Document Title: "Day 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE - [Stichhub]

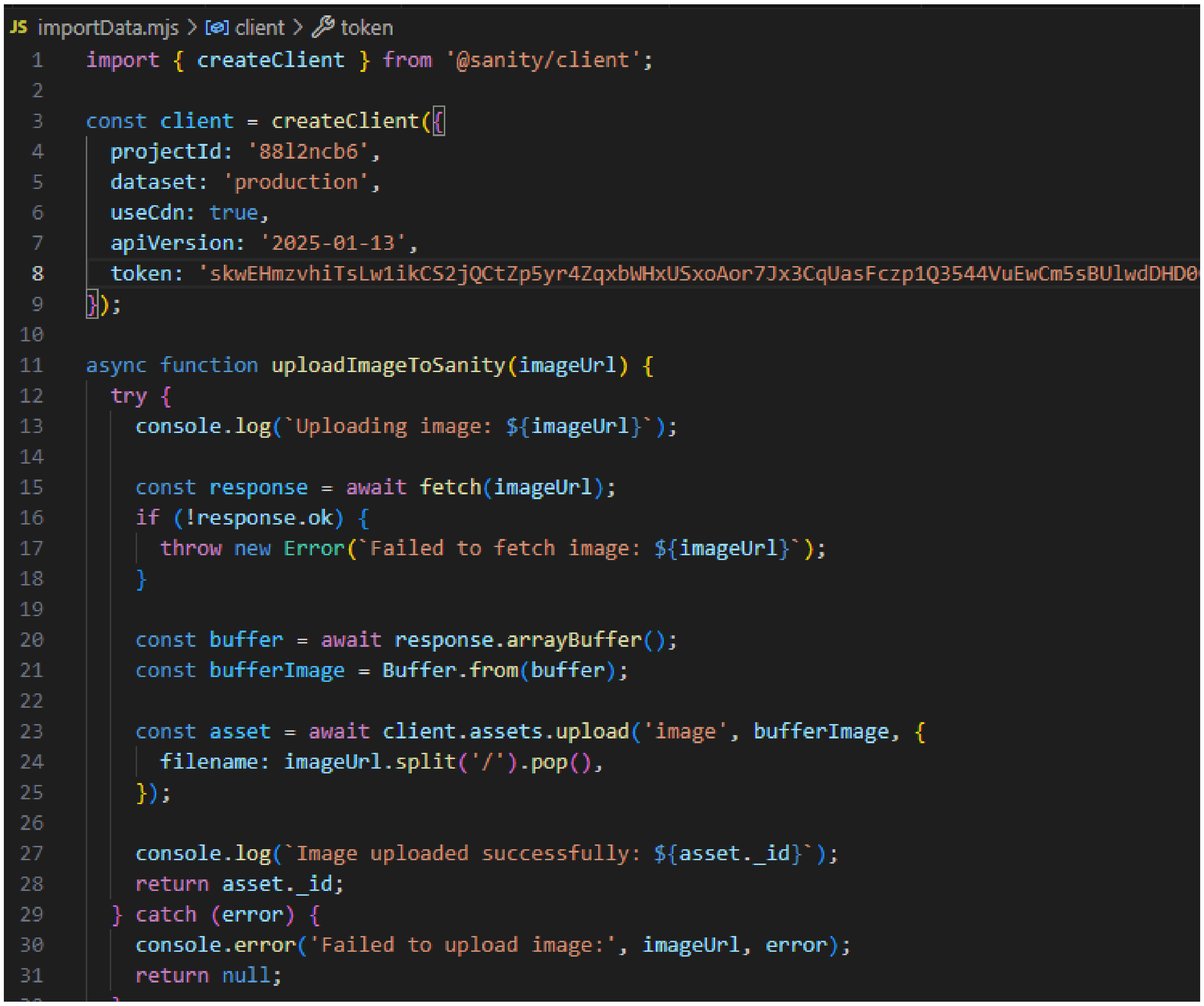
Overview

StichHub is an e-commerce website that showcases a wide range of products, allowing users to browse and make purchases with ease. The website features dynamic pages, product categories, and a seamless shopping experience, including an add-to-cart functionality. The application is built using Next.js and integrates with Sanity for content management, ensuring flexibility and scalability.

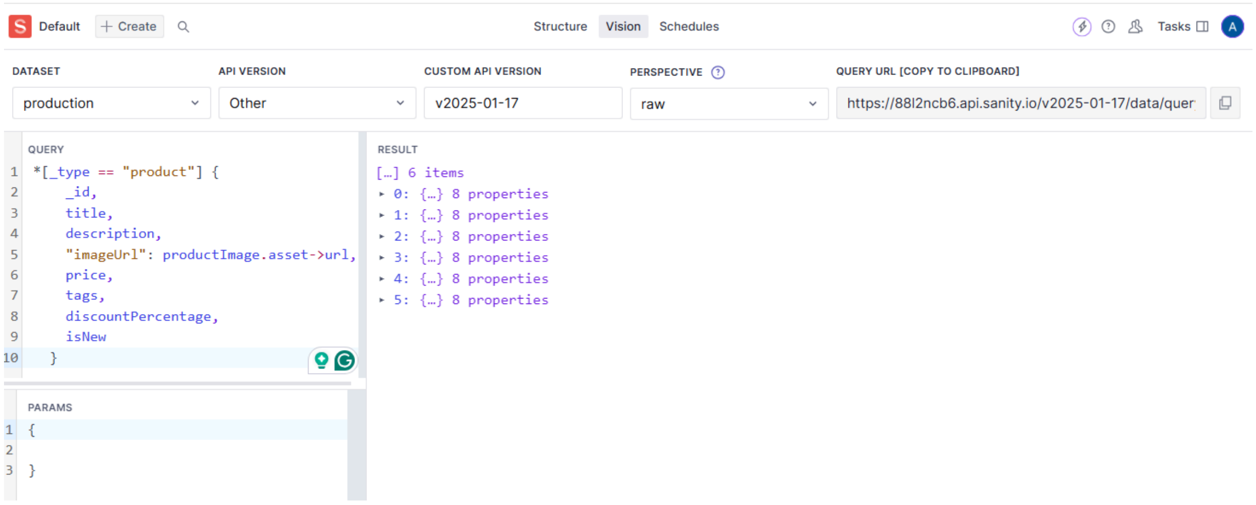
Features

1. API Integration & Data Migration

To bring the product data into the application, an API was integrated to fetch data from an external source. This data is then migrated to Sanity, ensuring that it is stored and managed centrally. By using Sanity's flexible schema, the product data is organized and made accessible for use in the website.


JS importData.mjs > client > token

```js
import { createClient } from '@sanity/client';

const client = createClient({
  projectId: '88l2ncb6',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token: 'skwEHmzvhiTsLw1ikCS2jQCtZp5yr4ZqxbWHxUSxoAor7Jx3CqUasFczp1Q3544VuEwCm5sBUlwdDHD0
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
```



DATASET: production
API VERSION: Other
CUSTOM API VERSION: v2025-01-17
PERSPECTIVE: raw
QUERY URL [COPY TO CLIPBOARD]: https://88l2ncb6.api.sanity.io/v2025-01-17/data/quer

QUERY

```
*[_type == "product"] {
    _id,
    title,
    description,
    "imageUrl": productImage.asset->url,
    price,
    tags,
    discountPercentage,
    isNew
  }
```

PARAMS

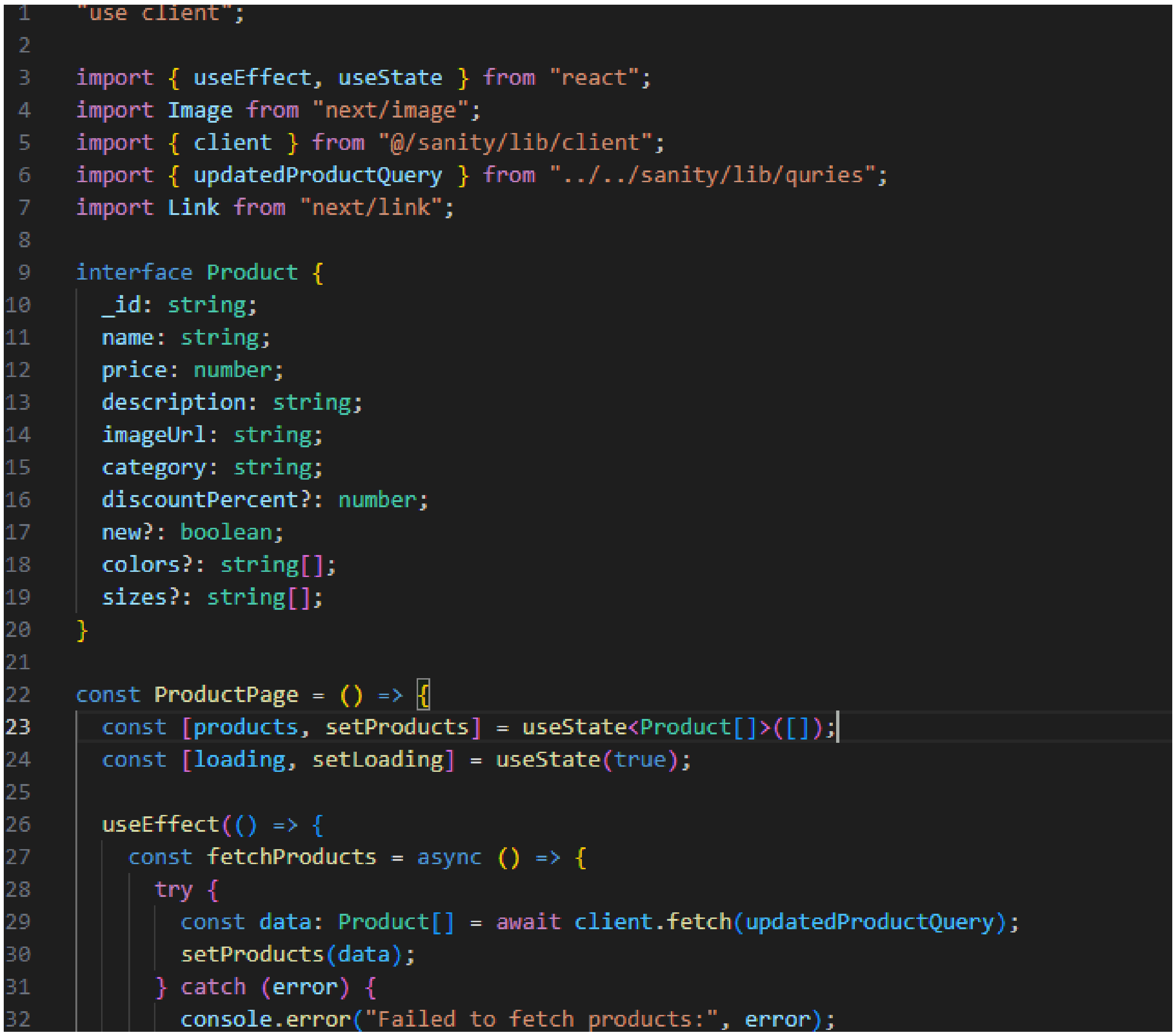
```
{

}
```

RESULT

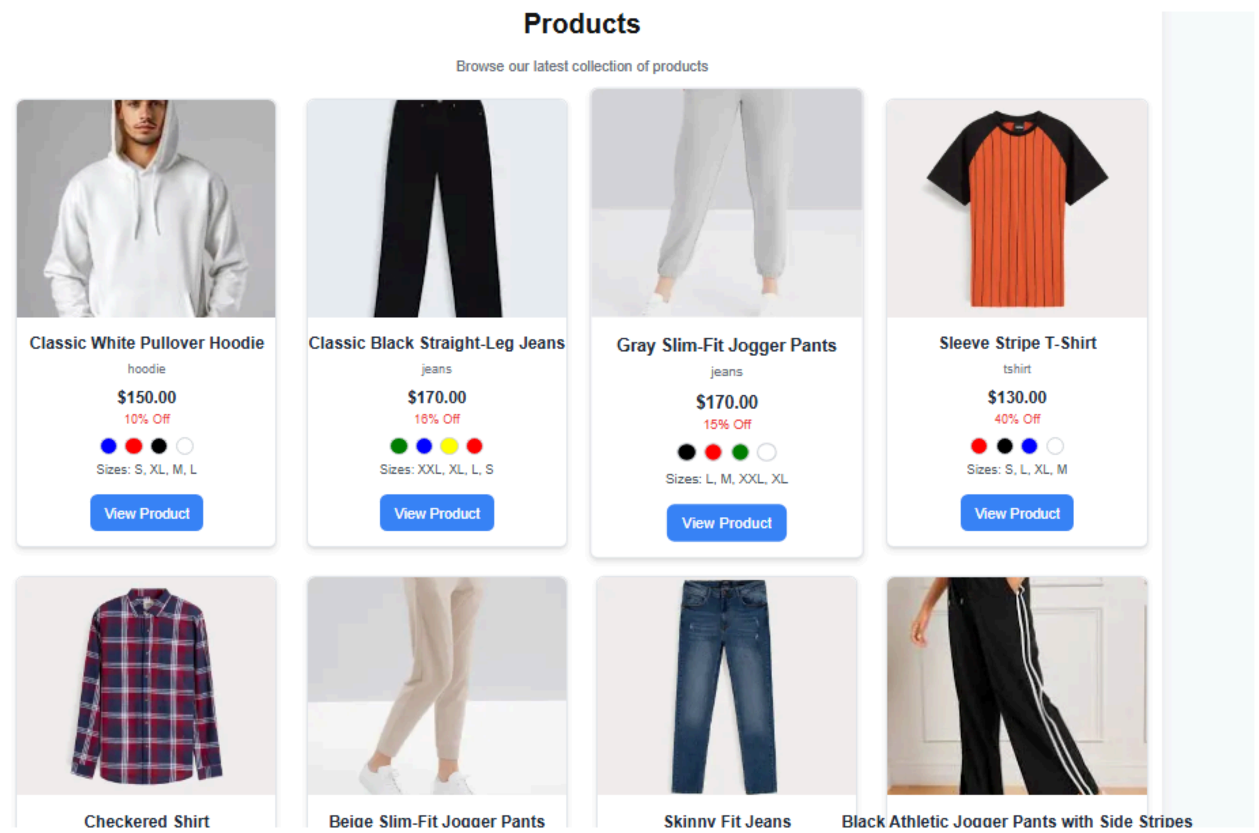
```
[…] 6 items
▸ 0: {…} 8 properties
▸ 1: {…} 8 properties
▸ 2: {…} 8 properties
▸ 3: {…} 8 properties
▸ 4: {…} 8 properties
▸ 5: {…} 8 properties
```

2. Product Data Fetching

To display the product data on the website, the `fetch` method is utilized in Next.js to call and retrieve the data from Sanity. This method ensures that the product listings are updated in real-time, reflecting any changes made in the Sanity CMS.



```
"use client";

import { useEffect, useState } from "react";
import Image from "next/image";
import { client } from "@/sanity/lib/client";
import { updatedProductQuery } from "../../sanity/lib/quries";
import Link from "next/link";

interface Product {
  _id: string;
  name: string;
  price: number;
  description: string;
  imageUrl: string;
  category: string;
  discountPercent?: number;
  new?: boolean;
  colors?: string[];
  sizes?: string[];
}

const ProductPage = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const data: Product[] = await client.fetch(updatedProductQuery);
        setProducts(data);
      } catch (error) {
        console.error("Failed to fetch products:", error);
```



Products

Browse our latest collection of products

Classic White Pullover Hoodie
hoodie
$150.00
10% Off
● ● ● ○
Sizes: S, XL, M, L
View Product

Classic Black Straight-Leg Jeans
jeans
$170.00
16% Off
● ● ● ●
Sizes: XXL, XL, L, S
View Product

Gray Slim-Fit Jogger Pants
jeans
$170.00
15% Off
● ● ● ○
Sizes: L, M, XXL, XL
View Product

Sleeve Stripe T-Shirt
tshirt
$130.00
40% Off
● ● ● ○
Sizes: S, L, XL, M
View Product

Checkered Shirt

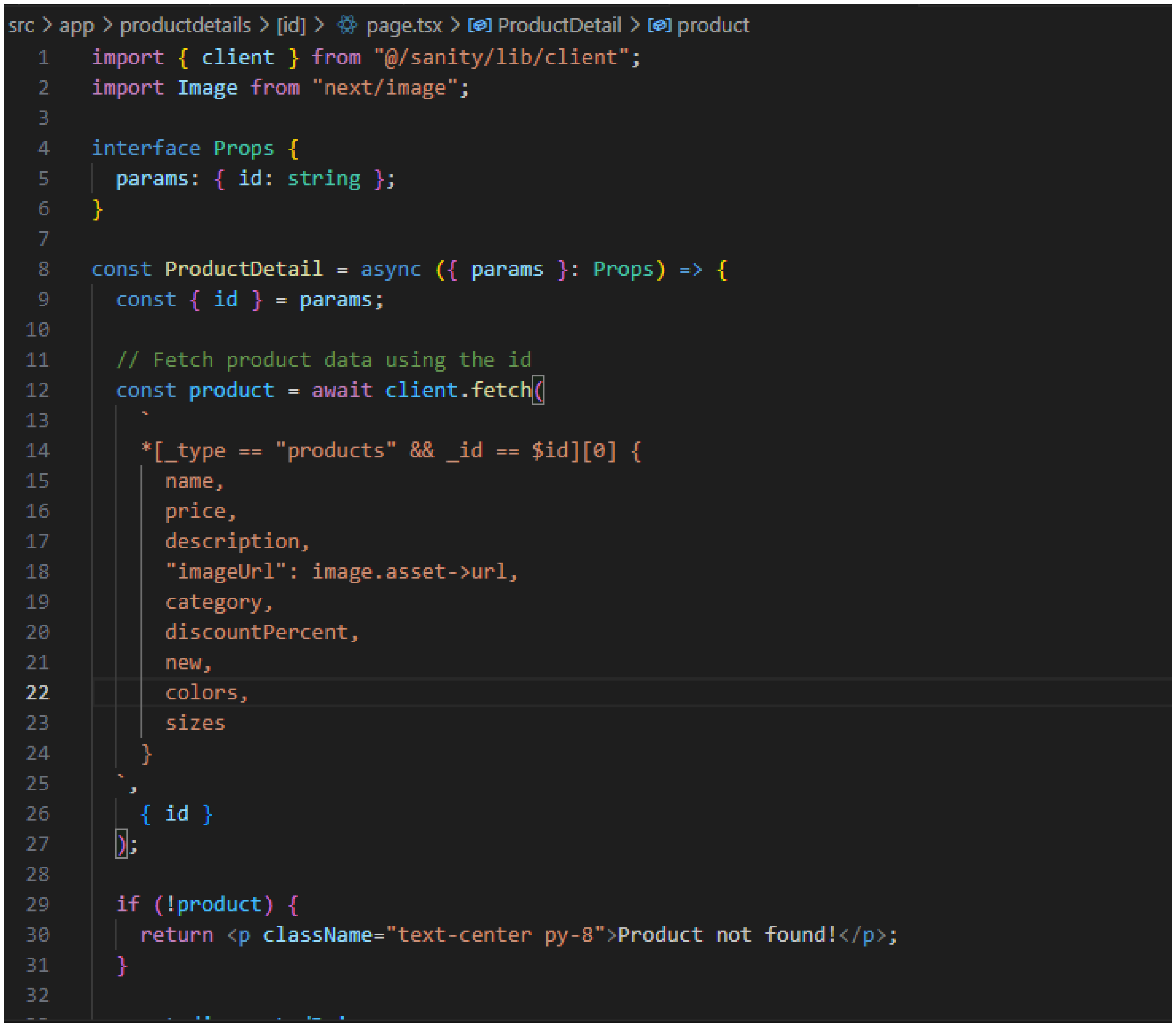
Beige Slim-Fit Jogger Pants

Skinny Fit Jeans

Black Athletic Jogger Pants with Side Stripes

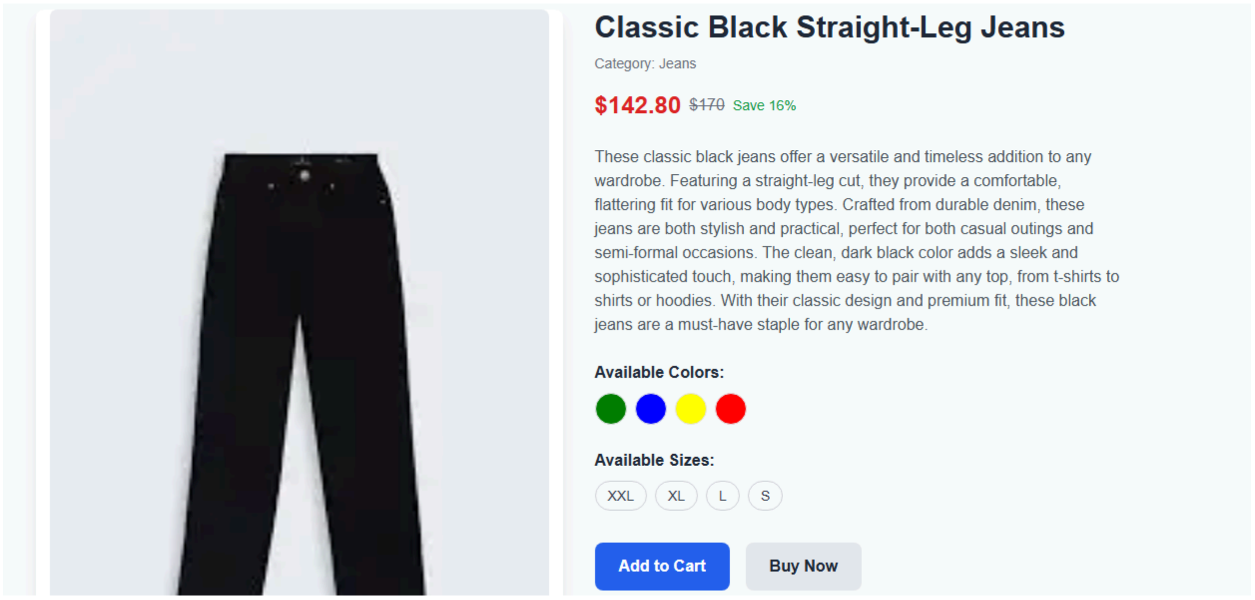
3. Dynamic Product Pages

A dynamic product page is created to allow users to view individual products. The page is generated based on the product's unique slug, which is part of the product data stored in Sanity. This allows users to access detailed information about each product, such as its description, price, images, sizes, and colors.



src > app > productdetails > [id] > page.tsx > ProductDetail > product

```tsx
import { client } from "@/sanity/lib/client";
import Image from "next/image";

interface Props {
  params: { id: string };
}

const ProductDetail = async ({ params }: Props) => {
  const { id } = params;

  // Fetch product data using the id
  const product = await client.fetch(
    `
    *[_type == "products" && _id == $id][0] {
      name,
      price,
      description,
      "imageUrl": image.asset->url,
      category,
      discountPercent,
      new,
      colors,
      sizes
    }
  `,
    { id }
  );

  if (!product) {
    return <p className="text-center py-8">Product not found!</p>;
  }

```


Classic Black Straight-Leg Jeans

Category: Jeans

$142.80 ~~$170~~ Save 16%

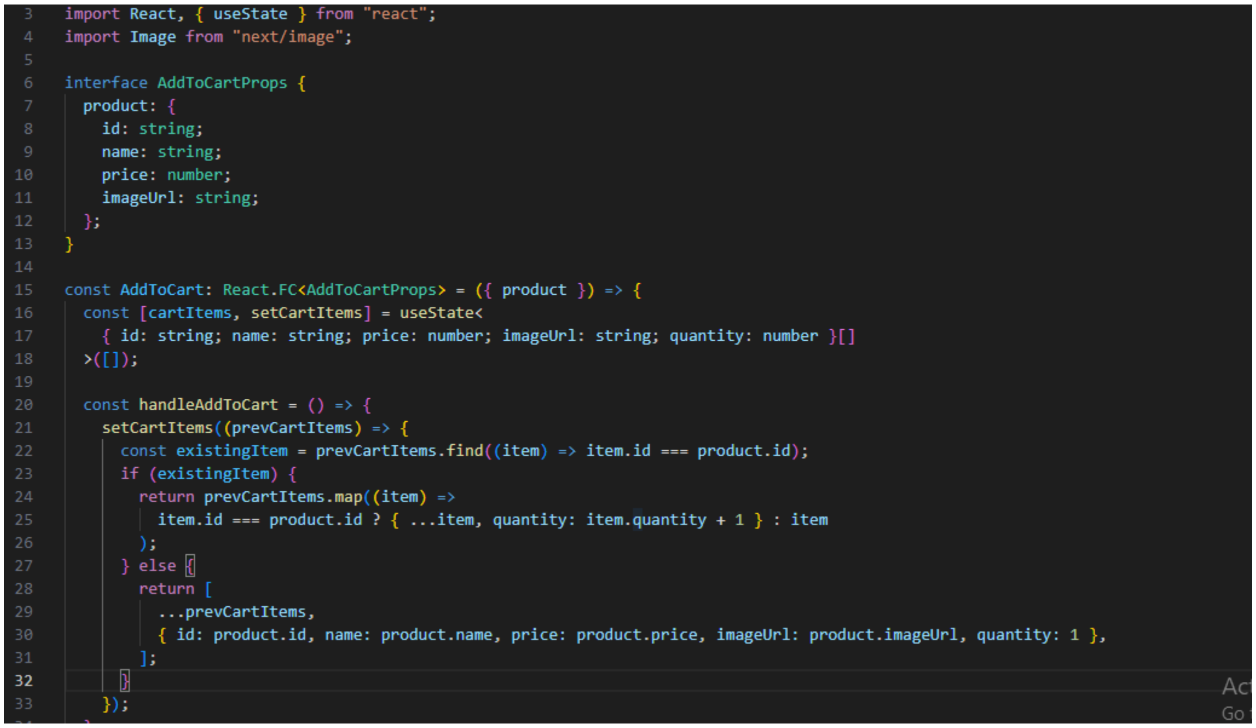
These classic black jeans offer a versatile and timeless addition to any wardrobe. Featuring a straight-leg cut, they provide a comfortable, flattering fit for various body types. Crafted from durable denim, these jeans are both stylish and practical, perfect for both casual outings and semi-formal occasions. The clean, dark black color adds a sleek and sophisticated touch, making them easy to pair with any top, from t-shirts to shirts or hoodies. With their classic design and premium fit, these black jeans are a must-have staple for any wardrobe.

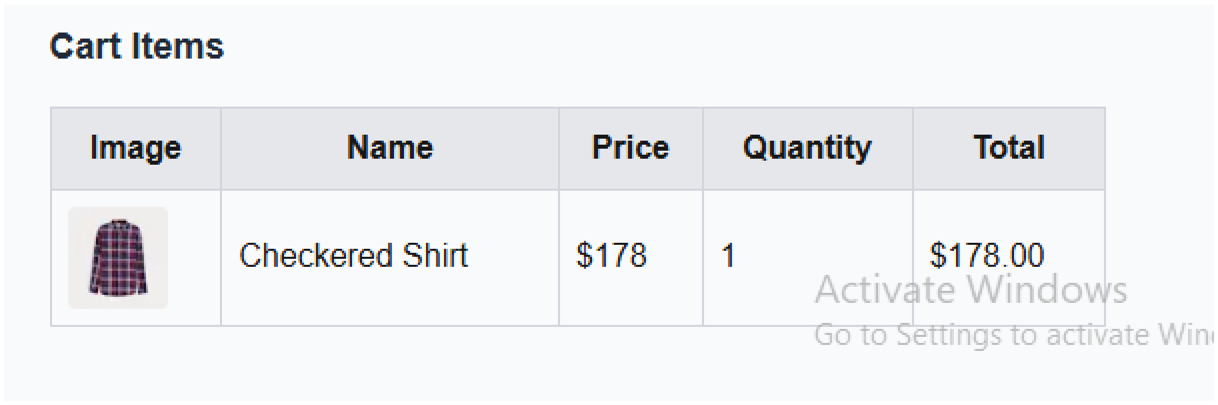
Available Colors:

Available Sizes:

4. Add-to-Cart Functionality

An essential feature of the website is the add-to-cart functionality. Users can easily add products to their cart, view the cart contents, and proceed to checkout. The shopping cart dynamically updates as products are added or removed, providing a seamless experience.


```
import React, { useState } from "react";
import Image from "next/image";

interface AddToCartProps {
  product: {
    id: string;
    name: string;
    price: number;
    imageUrl: string;
  };
}

const AddToCart: React.FC<AddToCartProps> = ({ product }) => {
  const [cartItems, setCartItems] = useState<
    { id: string; name: string; price: number; imageUrl: string; quantity: number }[]
  >([]);

  const handleAddToCart = () => {
    setCartItems((prevCartItems) => {
      const existingItem = prevCartItems.find((item) => item.id === product.id);
      if (existingItem) {
        return prevCartItems.map((item) =>
          item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item
        );
      } else {
        return [
          ...prevCartItems,
          { id: product.id, name: product.name, price: product.price, imageUrl: product.imageUrl, quantity: 1 },
        ];
      }
    });
```

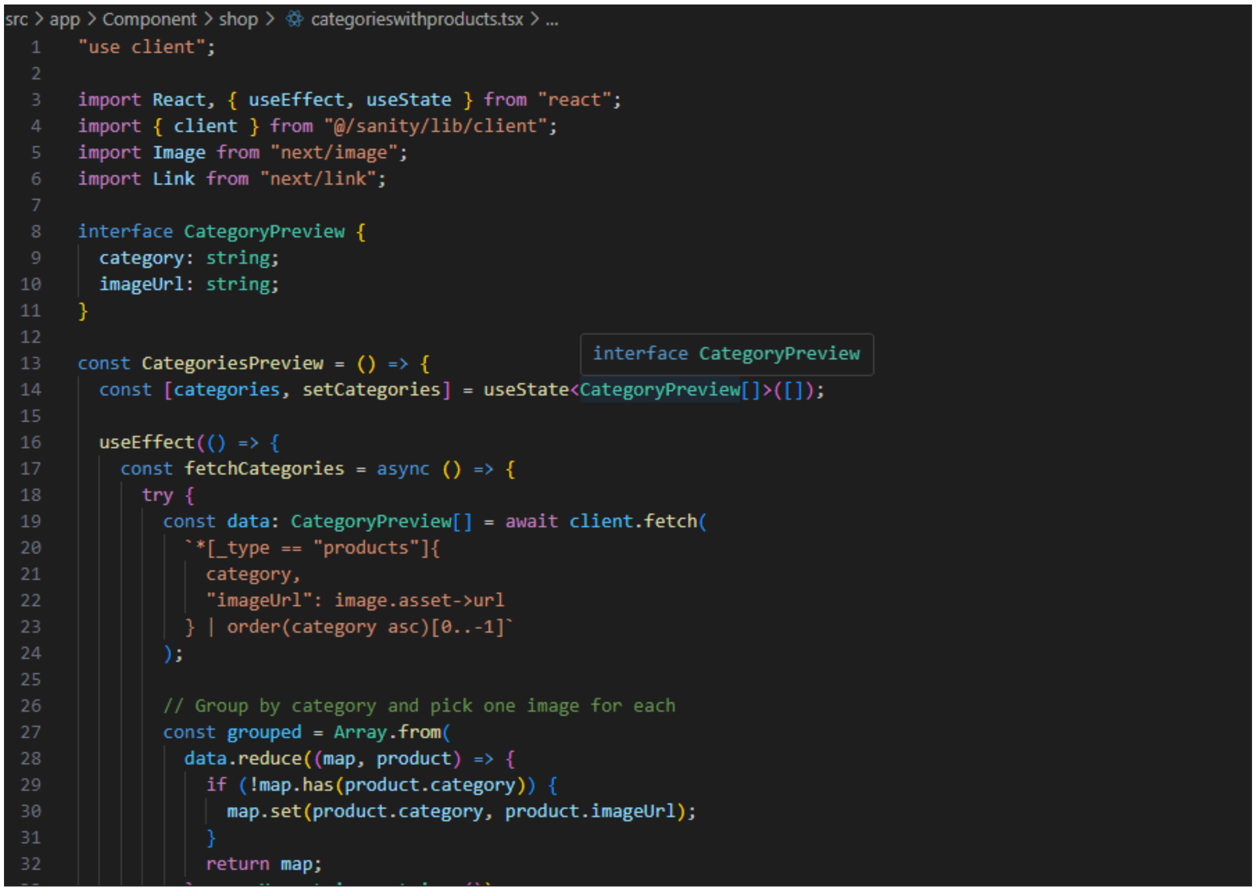


Cart Items

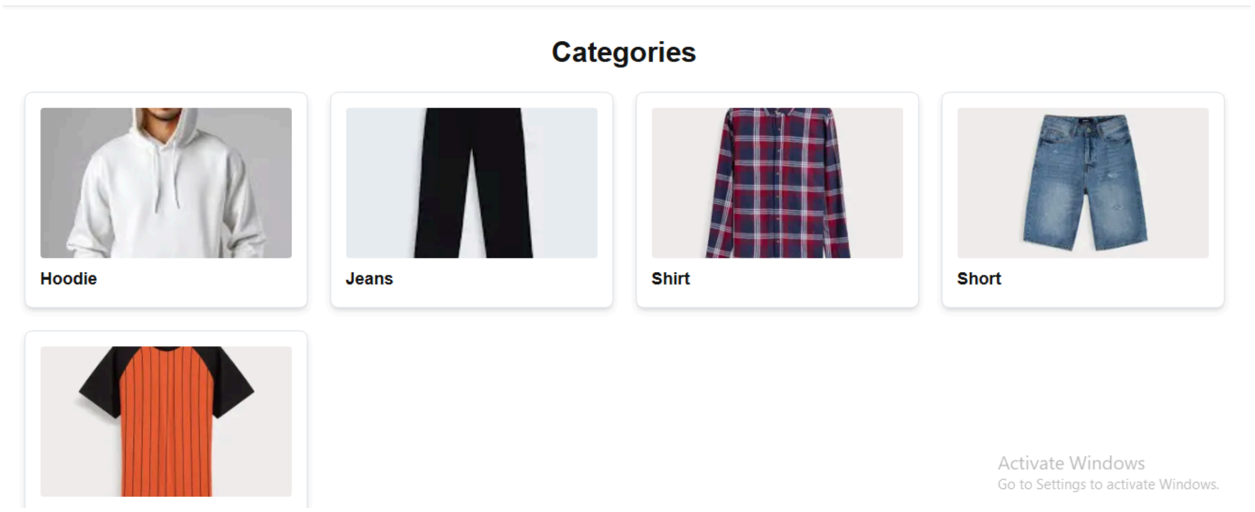
Image	Name	Price	Quantity	Total
	Checkered Shirt	$178	1	$178.00

5. Product Categories

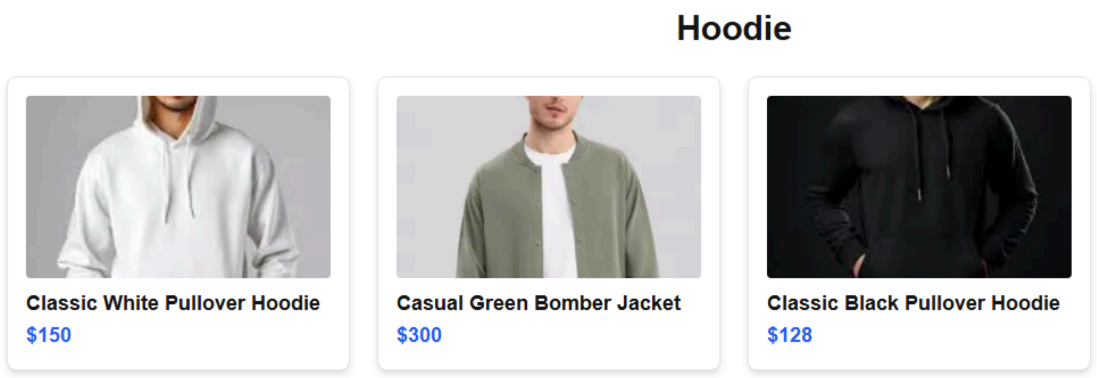
To enhance product discoverability, categories are implemented. Products are grouped by their respective categories, and users can filter the products based on these categories. This allows customers to easily browse and find products that meet their needs.

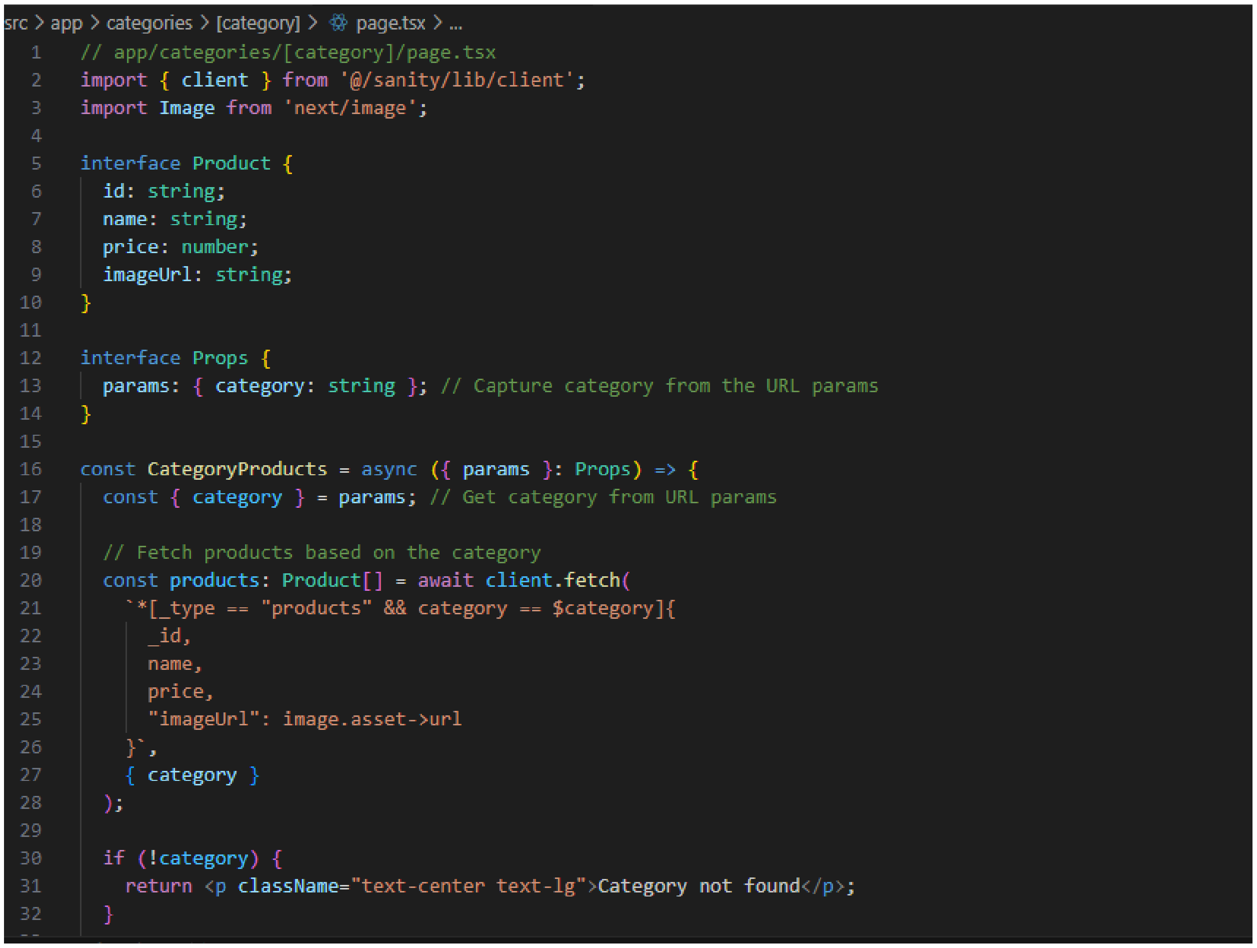

src > app > Component > shop > categorieswithproducts.tsx > ...
interface CategoryPreview

```tsx
"use client";

import React, { useEffect, useState } from "react";
import { client } from "@/sanity/lib/client";
import Image from "next/image";
import Link from "next/link";

interface CategoryPreview {
  category: string;
  imageUrl: string;
}

const CategoriesPreview = () => {
  const [categories, setCategories] = useState<CategoryPreview[]>([]);

  useEffect(() => {
    const fetchCategories = async () => {
      try {
        const data: CategoryPreview[] = await client.fetch(
          `*[_type == "products"]{
            category,
            "imageUrl": image.asset->url
          } | order(category asc)[0..-1]`
        );

        // Group by category and pick one image for each
        const grouped = Array.from(
          data.reduce((map, product) => {
            if (!map.has(product.category)) {
              map.set(product.category, product.imageUrl);
            }
            return map;
```



Dynamic page for showing products based on their categories


Hoodie
Classic White Pullover Hoodie
$150
Casual Green Bomber Jacket
$300
Classic Black Pullover Hoodie
$128



```tsx
// app/categories/[category]/page.tsx
import { client } from '@/sanity/lib/client';
import Image from 'next/image';

interface Product {
  id: string;
  name: string;
  price: number;
  imageUrl: string;
}

interface Props {
  params: { category: string }; // Capture category from the URL params
}

const CategoryProducts = async ({ params }: Props) => {
  const { category } = params; // Get category from URL params

  // Fetch products based on the category
  const products: Product[] = await client.fetch(
    `*[_type == "products" && category == $category]{
      _id,
      name,
      price,
      "imageUrl": image.asset->url
    }`,
    { category }
  );

  if (!category) {
    return <p className="text-center text-lg">Category not found</p>;
  }
```

Technical Details

- Frontend Framework: Next.js
- CMS: Sanity
- Data Fetching: `fetch` method
- Cart Management: Custom solution for managing cart state

Conclusion

StichHub is a fully functional e-commerce platform with dynamic product pages, real-time data fetching from Sanity, and an intuitive shopping experience. The integration of categories and the add-to-cart functionality enhances user experience, making it easy for customers to navigate the site and make purchases.