Document Title: "Day 3 - API Integration Report - [Stichhub]

Introduction

The reports outlines the process of integration of APIs ,migration of data to the sanity CMS for sttichhub,make necessary adjustment in the schema and display the data on my product page and create dynamic route

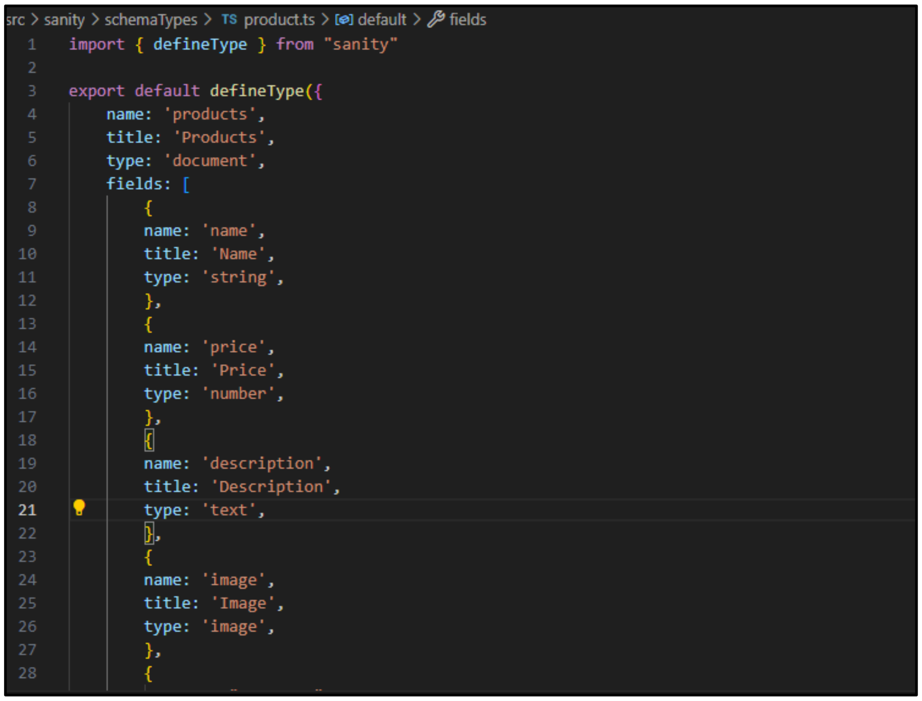
1. API Integration Process

During this phase, I integrated the necessary APIs for the Stichhub project to fetch product listings, categories I carefully reviewed the API documentation for the assigned template to understand the endpoints available and their structures.

2. Creating Sanity Project and Schema

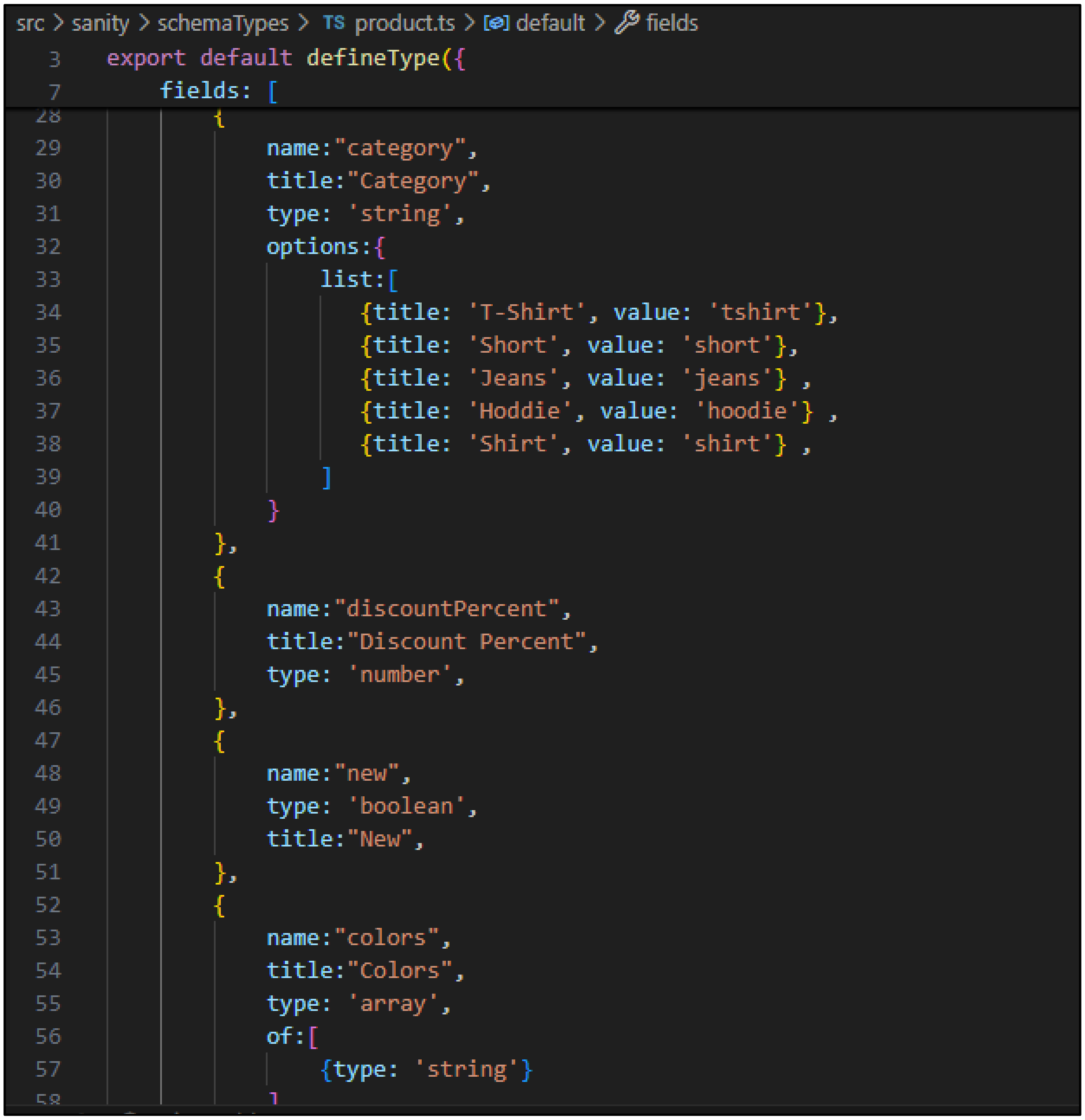
The first step in the process was setting up the Sanity project. I created a new Sanity project, ensuring that it was configured properly to handle the product data and other necessary information.

Next, I created the **Sanity schema**. The schema defines the structure of the content in the Sanity CMS, such as product details, categories.



src > sanity > schemaTypes > product.ts > default > fields

```
import { defineType } from "sanity"

export default defineType({
    name: 'products',
    title: 'Products',
    type: 'document',
    fields: [
        {
        name: 'name',
        title: 'Name',
        type: 'string',
        },
        {
        name: 'price',
        title: 'Price',
        type: 'number',
        },
        {
        name: 'description',
        title: 'Description',
        type: 'text',
        },
        {
        name: 'image',
        title: 'Image',
        type: 'image',
        },
        {
```



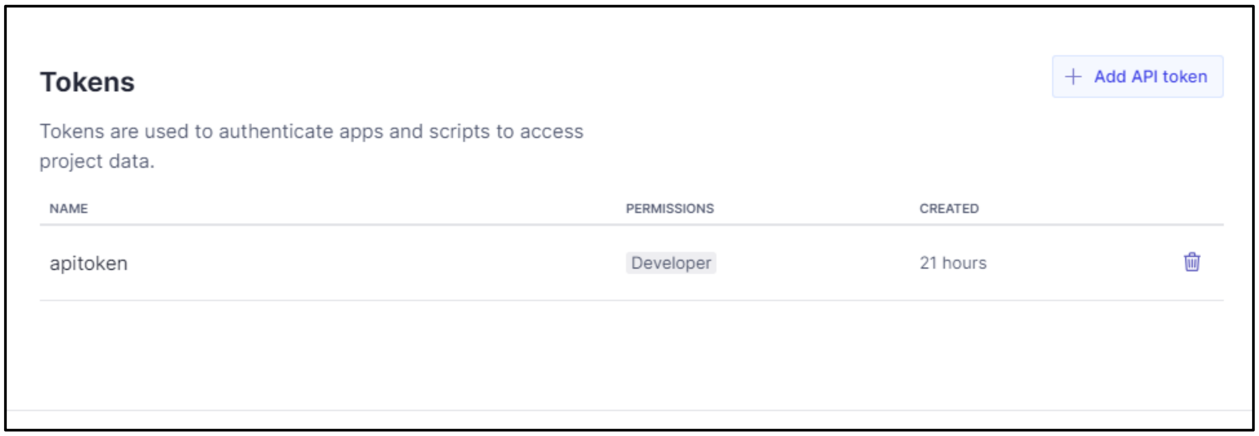
src > sanity > schemaTypes > TS product.ts > default > fields

```
export default defineType({
    fields: [
        {
            name:"category",
            title:"Category",
            type: 'string',
            options:{
                list:[
                    {title: 'T-Shirt', value: 'tshirt'},
                    {title: 'Short', value: 'short'},
                    {title: 'Jeans', value: 'jeans'} ,
                    {title: 'Hoddie', value: 'hoodie'} ,
                    {title: 'Shirt', value: 'shirt'} ,
                ]
            }
        },
        {
            name:"discountPercent",
            title:"Discount Percent",
            type: 'number',
        },
        {
            name:"new",
            type: 'boolean',
            title:"New",
        },
        {
            name:"colors",
            title:"Colors",
            type: 'array',
            of:[
                {type: 'string'}
            ]
```

3.Generating API Token

After creating the Sanity project and schema, I generated an **API token**. This token is essential for authenticating and securely connecting the migration process with Sanity, allowing the imported data to be stored in the Sanity CMS.

I inserted this API token into the **migration file**, ensuring that the connection between the migration script and Sanity CMS was secure and functional.


Tokens

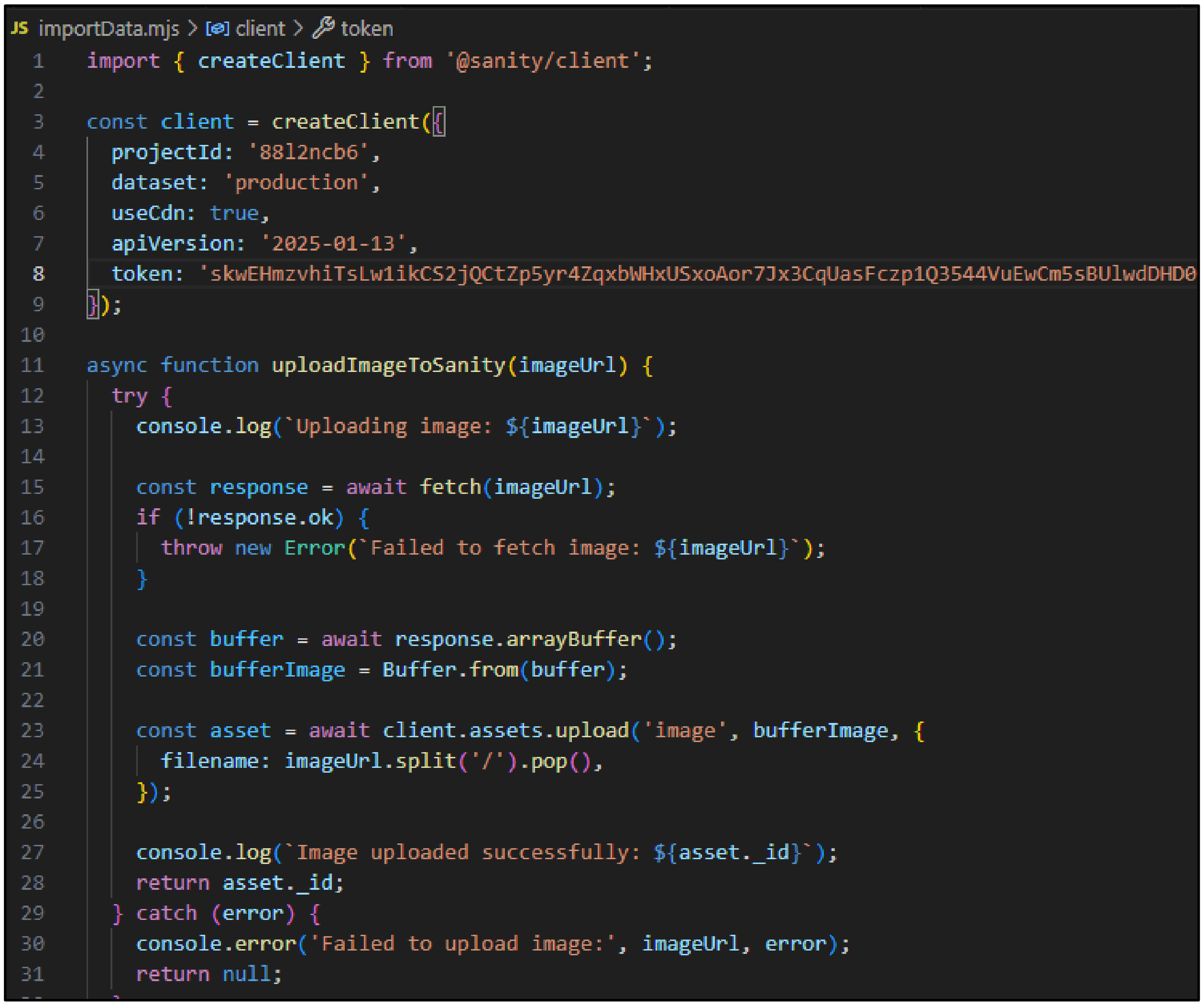
Tokens are used to authenticate apps and scripts to access project data.

NAME	PERMISSIONS	CREATED
apitoken	Developer	21 hours

4. **Creating Migration File for Data Import**

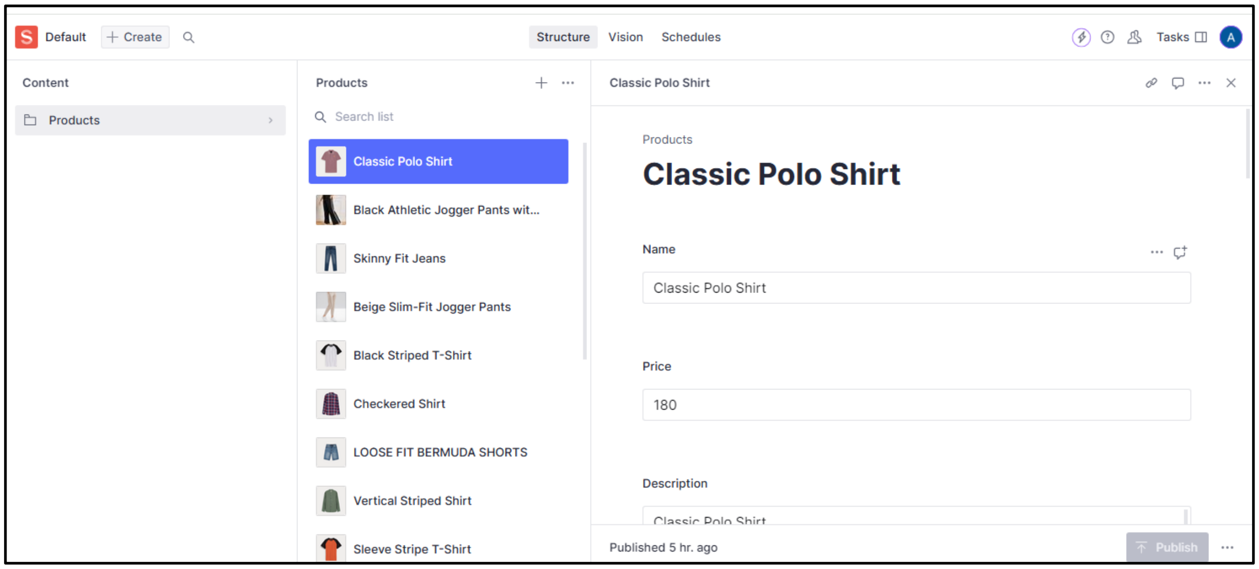
To streamline the migration of data from the external API into Sanity, I created a **migration file** named importdata.mjs. This file contains the necessary logic to fetch and transform the data from the API and then insert it into the Sanity CMS.

Once the migration file was set up, I ran the command npm run migrate to initiate the migration process. This successfully transferred the data from the API into the Sanity Studio.


JS importData.mjs > client > token

```javascript
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;
```

Successfully migrated the data



Products

Search list

- Classic Polo Shirt
- Black Athletic Jogger Pants wit...
- Skinny Fit Jeans
- Beige Slim-Fit Jogger Pants
- Black Striped T-Shirt
- Checkered Shirt
- LOOSE FIT BERMUDA SHORTS
- Vertical Striped Shirt
- Sleeve Stripe T-Shirt

Classic Polo Shirt

Products

Classic Polo Shirt

Name: Classic Polo Shirt

Price: 180

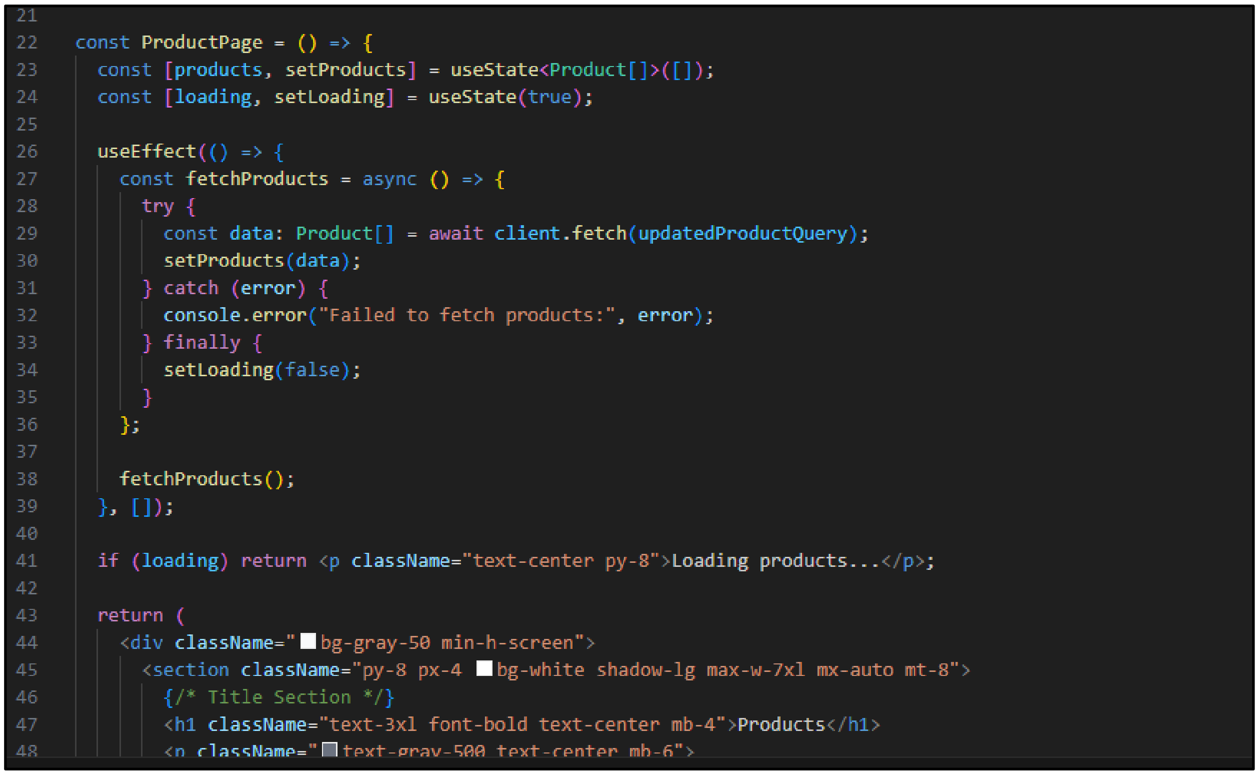
Description: Classic Polo Shirt

Published 5 hr. ago

5.Querying and Fetching Data in Next.js Project

After successfully migrating the data to Sanity, the next step was to fetch and display this data in the frontend of the Next.js project. I created a **query** to retrieve the necessary product information from Sanity, ensuring that all relevant data such as product names, descriptions, prices, and images were fetched correctly.

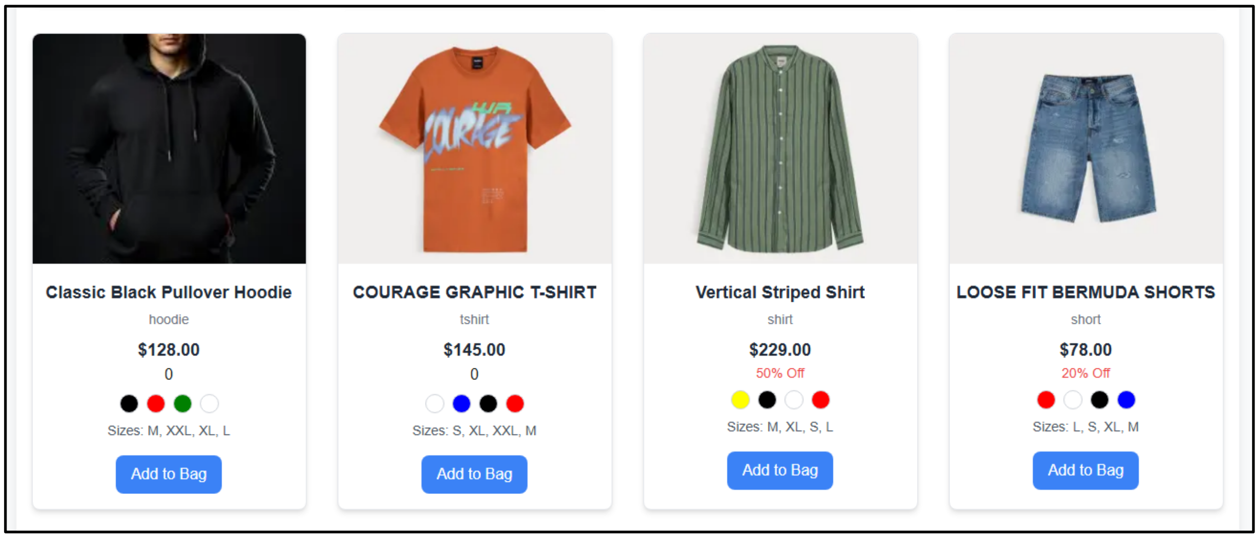
I then used this query in my Next.js project to populate the product page with dynamic content.



```
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);
      } finally {
        setLoading(false);
      }
    };

    fetchProducts();
  }, []);

  if (loading) return <p className="text-center py-8">Loading products...</p>;

  return (
    <div className="bg-gray-50 min-h-screen">
      <section className="py-8 px-4 bg-white shadow-lg max-w-7xl mx-auto mt-8">
        {/* Title Section */}
        <h1 className="text-3xl font-bold text-center mb-4">Products</h1>
        <p className="text-gray-500 text-center mb-6">
```

6.Styling Product Page with Tailwind CSS

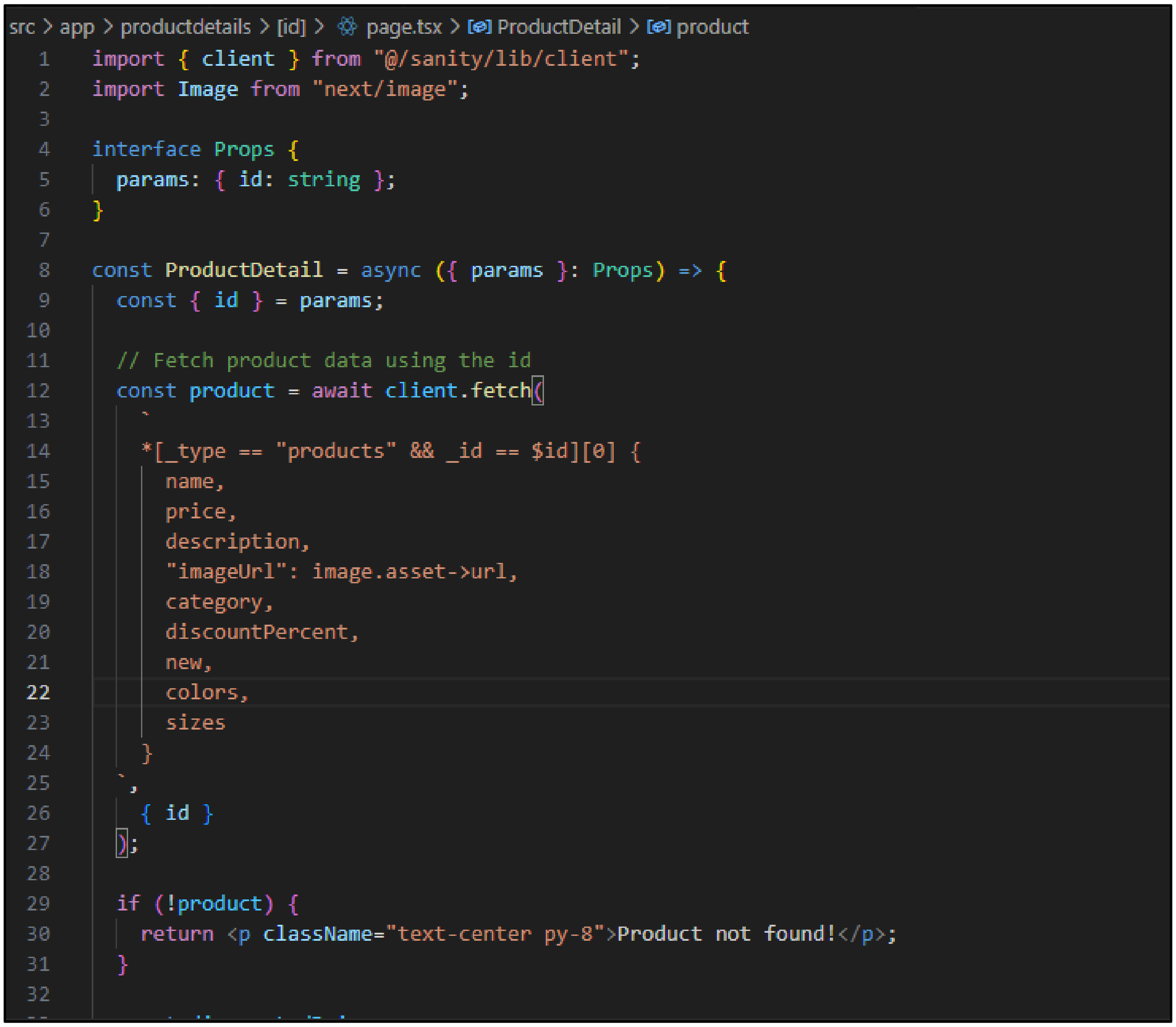
To improve the user interface and ensure a visually appealing product page, I applied **Tailwind CSS**. This allowed me to create a responsive, clean, and modern design for the product listings page.



Classic Black Pullover Hoodie	COURAGE GRAPHIC T-SHIRT	Vertical Striped Shirt	LOOSE FIT BERMUDA SHORTS
hoodie	tshirt	shirt	short
$128.00	$145.00	$229.00	$78.00
0	0	50% Off	20% Off
● ● ● ●	● ● ● ●	● ● ● ●	● ● ● ●
Sizes: M, XXL, XL, L	Sizes: S, XL, XXL, M	Sizes: M, XL, S, L	Sizes: L, S, XL, M
Add to Bag	Add to Bag	Add to Bag	Add to Bag

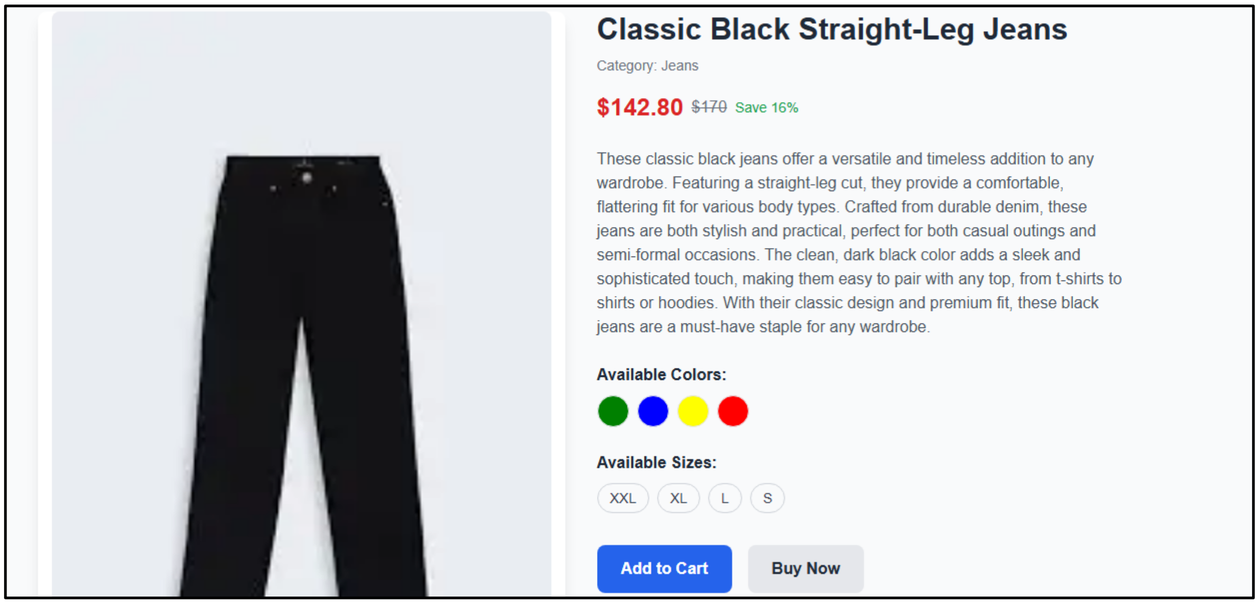
7. Creating Dynamic Product Page for Single Product View

To allow users to view individual product details, I created a **dynamic page** in Next.js. This page is designed to display a single product's details when the user clicks on a product from the listings. The dynamic page is routed based on the product's unique identifier.



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.

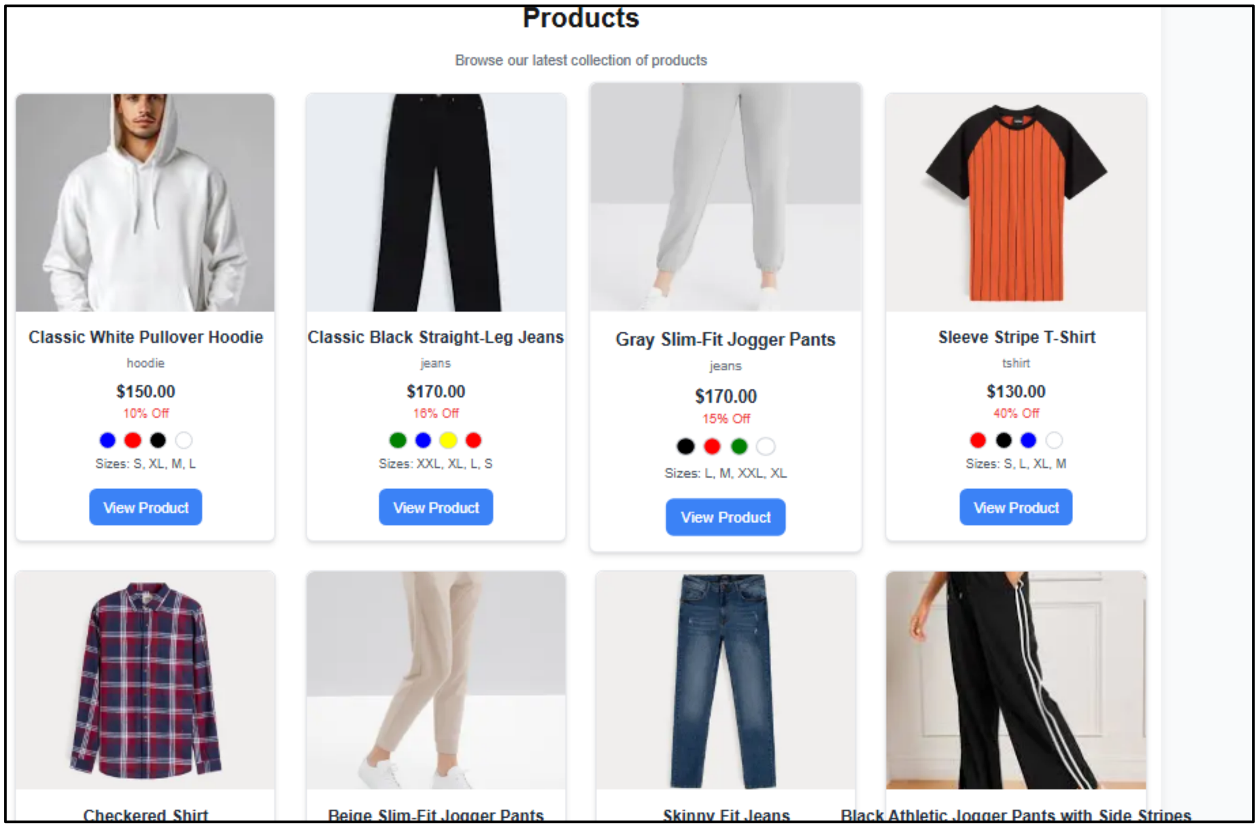
Available Colors:

Available Sizes:

XXL XL L S

Add to Cart Buy Now

Here is the website final view



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

Conclusion:

This concludes the report on the API integration and data migration process for the Stichhub project. The steps outlined here demonstrate a successful migration and integration of API data into the Sanity CMS and its display on the Next.js frontend.