Document Title : Day 3 – API Integration Report

What to submit :

- A report documenting :
- API integration process
- Adjustment made to schemas
- Migration steps and tools used

Screenshots of :

- API calls
- Data successfully displayed in the frontend
- Populated sanity CMS fields
- Code snippets For API integration and migration scripts

1. Overview of API Integration:

API Selection:

To integrate data, the given API was utilized for retrieving product details:

Using the provided API ensured accurate data retrieval and compatibility with the project requirements.

Fetching Data:

The Fetch was utilized to send GET requests to the given API. Once the data was fetched, it was processed and structured for seamless integration with Sanity CMS.

Sanity Integration Process

The integration with Sanity CMS followed these main steps:

1. Sanity Client Configuration:

A custom client was set up with the necessary credentials (project ID, dataset, and token).

Image Uploads:

Images were retrieved from URLs, converted into suitable formats, and uploaded to Sanity as assets using a custom function.

Product Data Uploads:

Product details were mapped to a schema and uploaded to Sanity. References for assets and inventory were created to maintain proper linkage.

Key Milestones

- Successfully retrieved product data from the given API.
- Uploaded images and assets into the CMS.
- Established relationships between products and their inventory.
- Integrated structured product data into Sanity,

2. Schema Enhancements

Initial Schema Design

Initially, schemas for products and inventory were set up with attributes like:

- Name
- Price

- Description
- Rating
- Category

Adjustments and Improvements

During development, adjustments were made to:

Resolve Reference Issues:

Ensured all product-inventory relationships were correctly linked.

Add New Fields:

Included additional attributes like category, availability, and discount rate to enhance product details.

Validation Rules:

Implemented checks for empty product names and invalid prices to maintain data quality.

These modifications improved schema efficiency and aligned it with the project's goals.

Migration Steps and Tools Used

Migration Overview:

This migration process focused on transferring data from a given API to the Sanity CMS. The goal was to ensure the data was organized correctly, with all references properly linked for smooth functionality.

Steps Involved:

Setting Up the Sanity Project:

A new project was created in Sanity. Credentials like the project ID, dataset name, and token were obtained and configured to set up the system.

Fetching Data with Fetch:

The Fetch library was used to retrieve data from the given API. Custom error-handling functions were added to manage data fetching efficiently.

Creating and Adjusting Schemas:

Schemas for products, inventory, orders, and shipments were created in Sanity. Relationships between products, images, and inventory were defined to ensure data integrity.

Importing Data:

Data from the given API was processed, and for each product:

Images were uploaded to Sanity using a custom image upload feature.

Inventory references were either found or newly created.

Product details were linked to the schema and saved in Sanity.

Verification:

After completing the migration, all data in Sanity was reviewed to confirm accuracy and proper linking of references.

Tools and Technologies Used:

Fetch: To fetch data from the given API.

Sanity Client: To connect with Sanity CMS and manage data uploads.

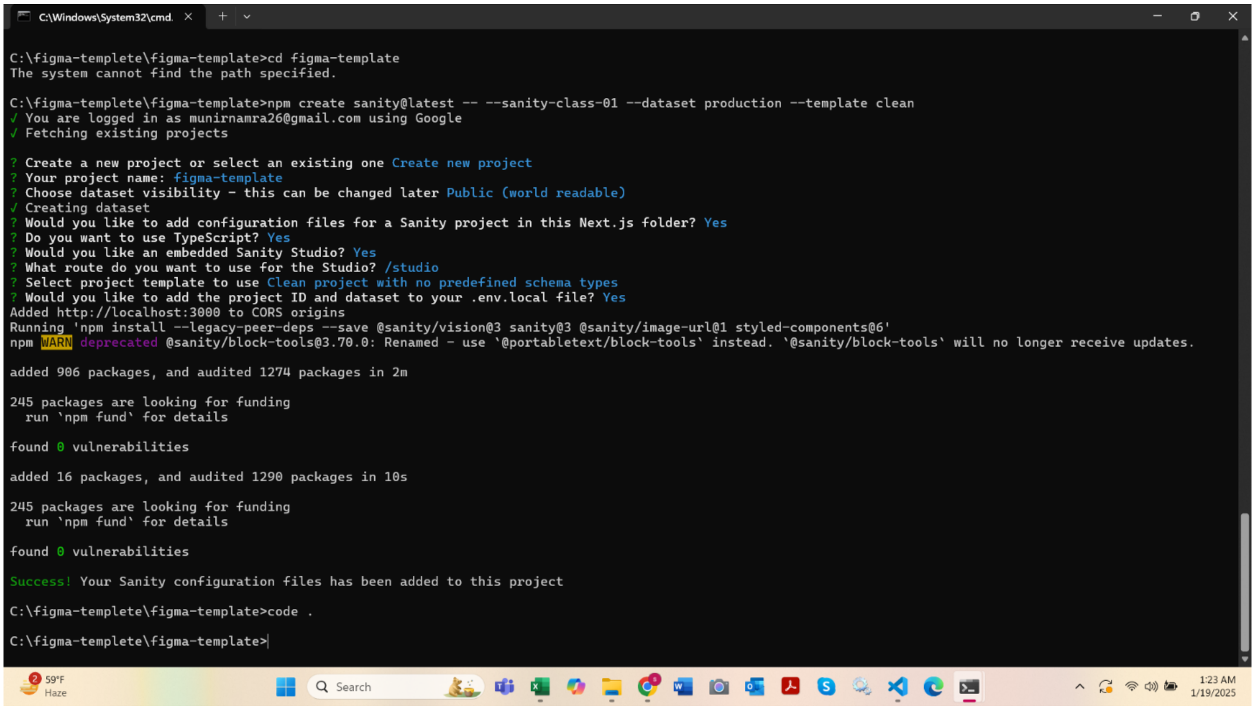
Custom Migration Script: To upload product details, images, and inventory while ensuring all references were properly created or updated.

Node.js: The migration script was written and executed using Node.js.

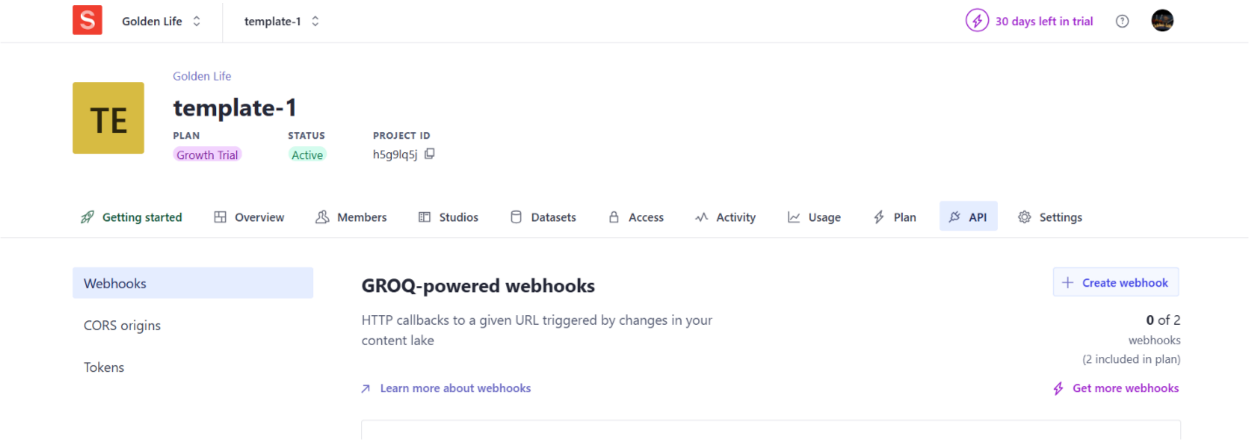
Conclusion

The migration for ShopCo was completed successfully. Data from the given API was transferred to Sanity CMS, with custom functions ensuring smooth image uploads and accurate inventory referencing.

By linking product details, images, and inventory data effectively, the system is now ready for future use and scaling. This process ensures ShopCo's data is well-organized and easy to manage.



```
C:\figma-templete\figma-template>cd figma-template
The system cannot find the path specified.

C:\figma-templete\figma-template>npm create sanity@latest -- --sanity-class-01 --dataset production --template clean
✓ You are logged in as munirnamra26@gmail.com using Google
✓ Fetching existing projects

? Create a new project or select an existing one Create new project
? Your project name: figma-template
? Choose dataset visibility - this can be changed later Public (world readable)
✓ Creating dataset
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'
npm WARN deprecated @sanity/block-tools@3.70.0: Renamed - use `@portabletext/block-tools` instead. `@sanity/block-tools` will no longer receive updates.

added 906 packages, and audited 1274 packages in 2m

245 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

added 16 packages, and audited 1290 packages in 10s

245 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Success! Your Sanity configuration files has been added to this project

C:\figma-templete\figma-template>code .

C:\figma-templete\figma-template>
```



Golden Life

template-1

PLAN: Growth Trial

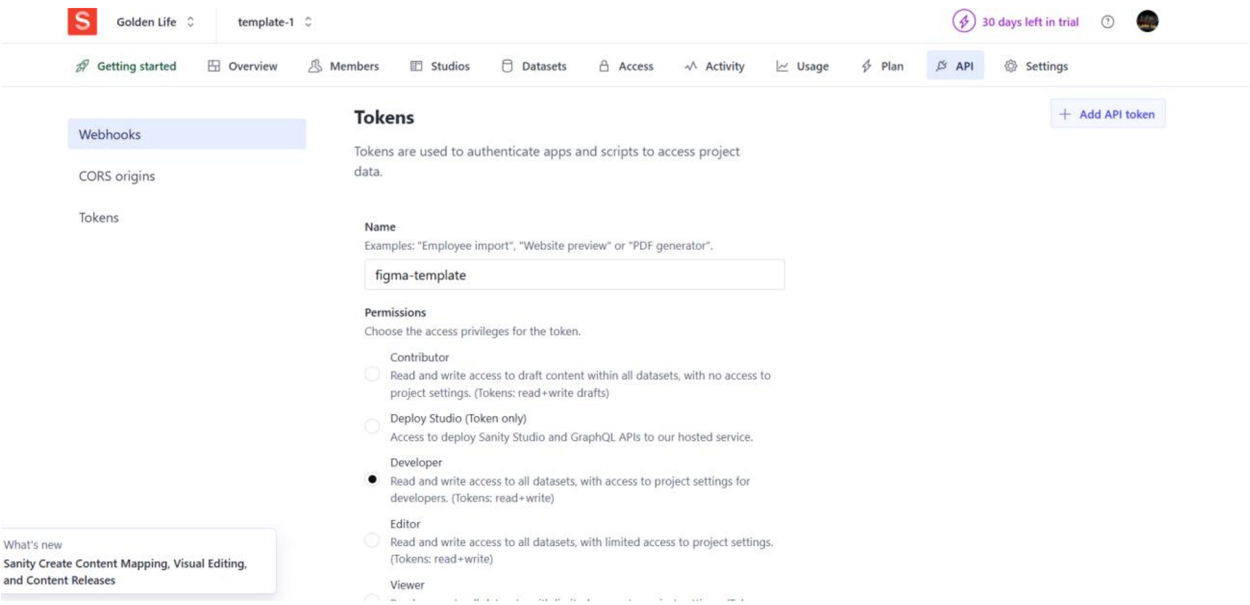
STATUS: Active

PROJECT ID: h5g9lq5j

GROQ-powered webhooks

HTTP callbacks to a given URL triggered by changes in your content lake

0 of 2 webhooks (2 included in plan)



Tokens

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

Name

Examples: "Employee import", "Website preview" or "PDF generator".

figma-template

Permissions

Choose the access privileges for the token.

○ Contributor
Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)

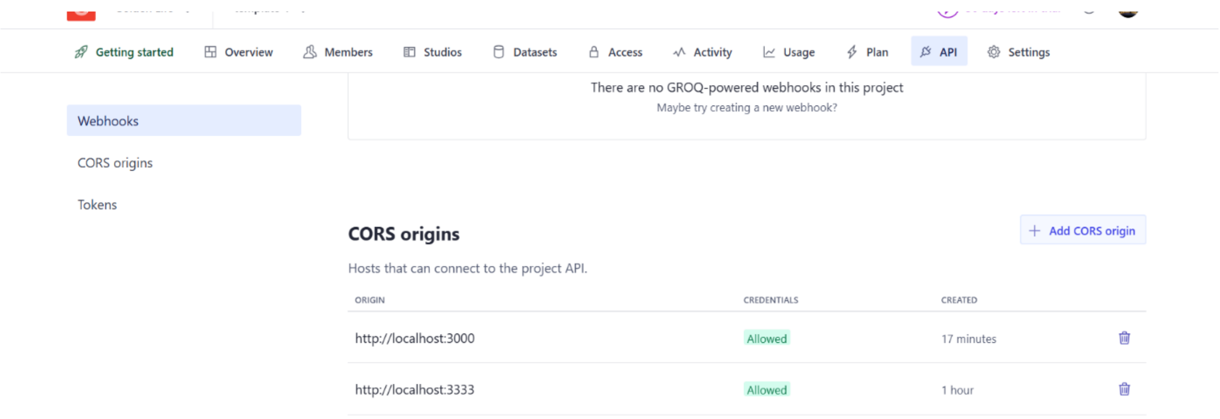
○ Deploy Studio (Token only)
Access to deploy Sanity Studio and GraphQL APIs to our hosted service.

● Developer
Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)

○ Editor
Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)

○ Viewer

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases



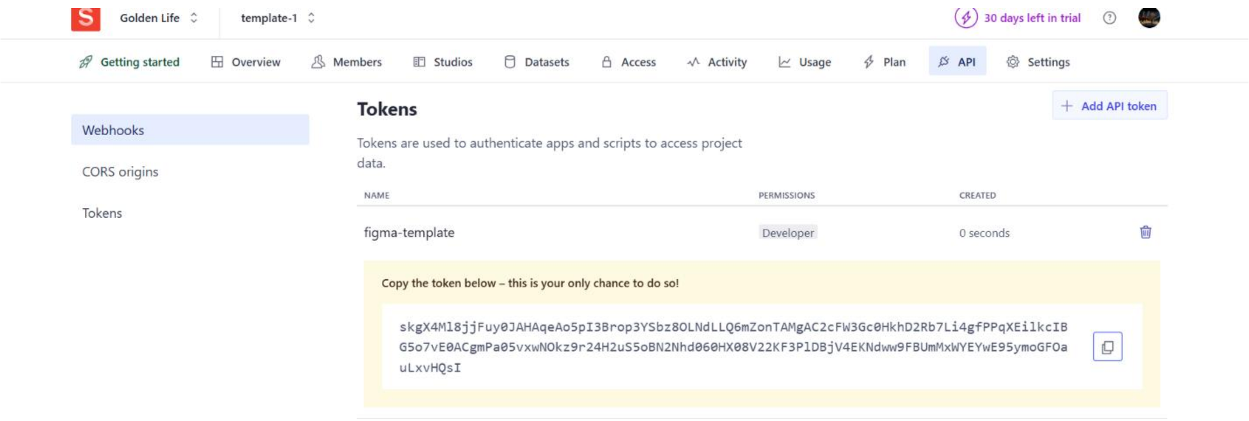
There are no GROQ-powered webhooks in this project

Maybe try creating a new webhook?

CORS origins

Hosts that can connect to the project API.

ORIGIN	CREDENTIALS	CREATED
http://localhost:3000	Allowed	17 minutes
http://localhost:3333	Allowed	1 hour



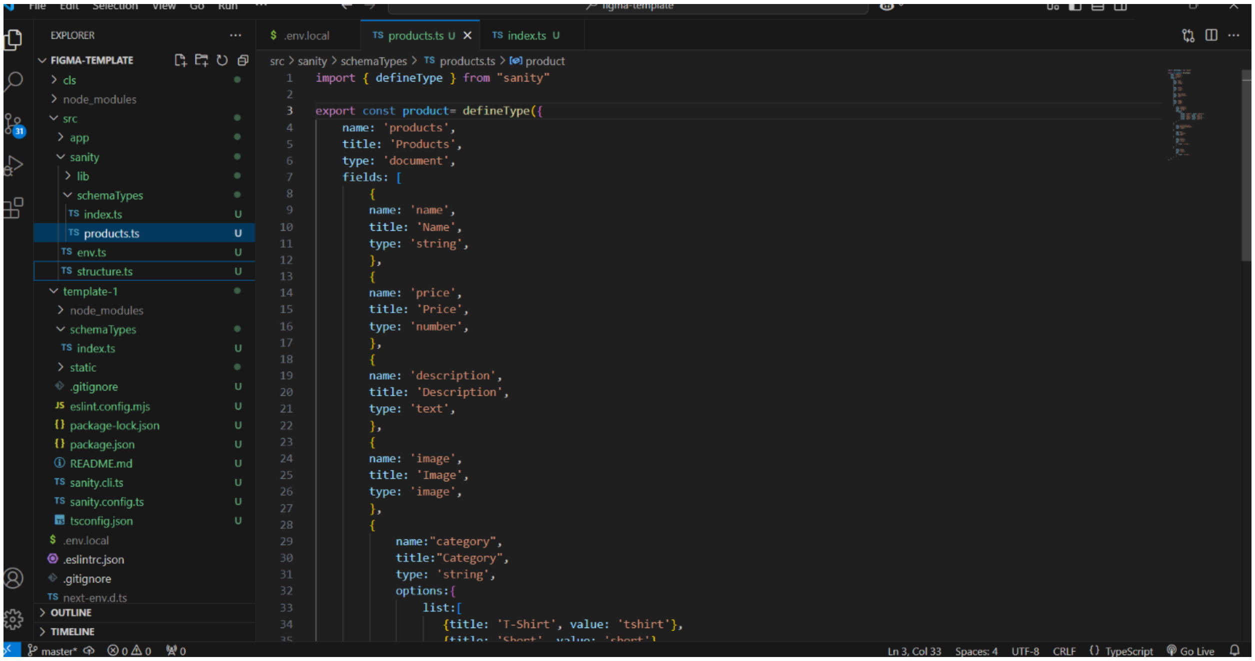
Tokens

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

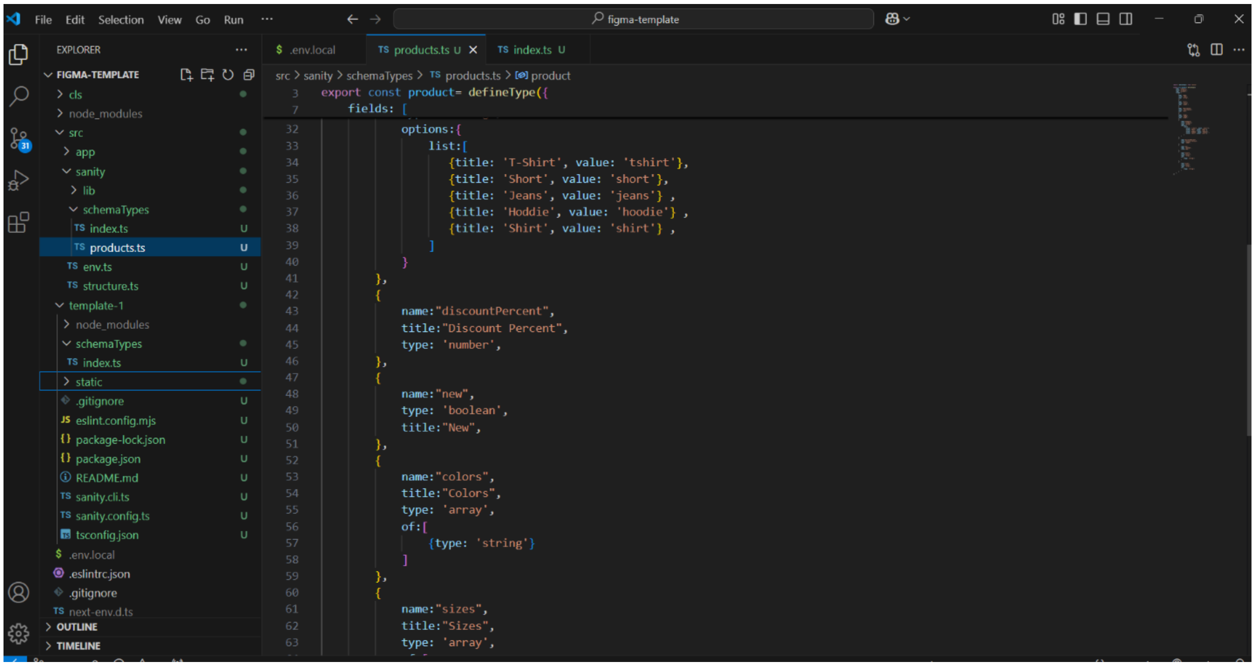
NAME	PERMISSIONS	CREATED
figma-template	Developer	0 seconds

Copy the token below – this is your only chance to do so!

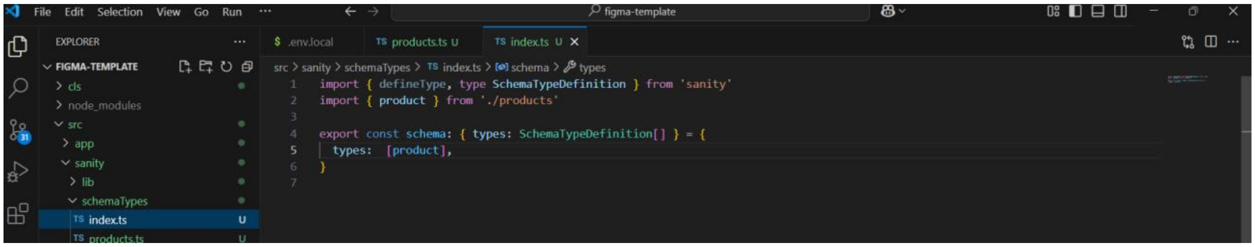
```
skgX4Ml8jjFuy0JAHAqeAo5pI3Brop3YSbz8OLNdLLQ6mZonTAMgAC2cFW3Gc0HkhD2Rb7Li4gfPPqXEilkcIB
G5o7vE0ACgmPa05vxwNOkz9r24H2uS5oBN2Nhd060HX08V22KF3PlDBjV4EKNdww9FBUmMxWYEYwE95ymoGFOa
uLxvHQsI
```



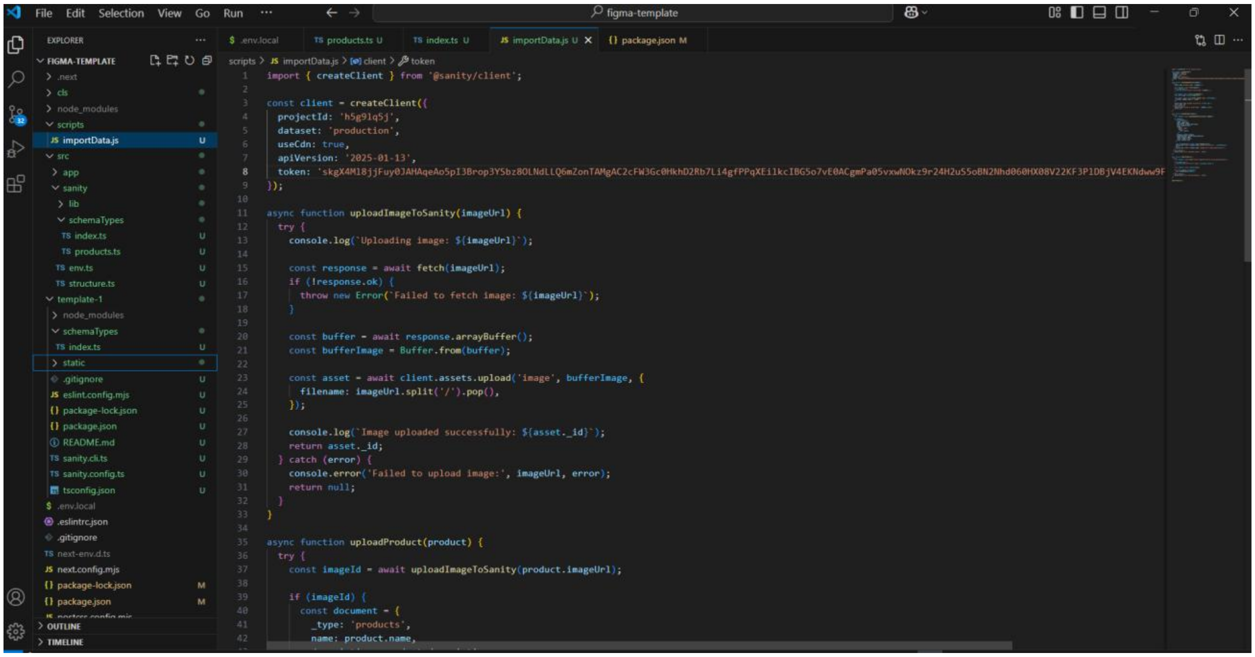
```
import { defineType } from "sanity"

export const product= 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',
        },
        {
            name:"category",
            title:"Category",
            type: 'string',
            options:{
                list:[
                    {title: 'T-Shirt', value: 'tshirt'},
                    {title: 'Short', value: 'short'}
```



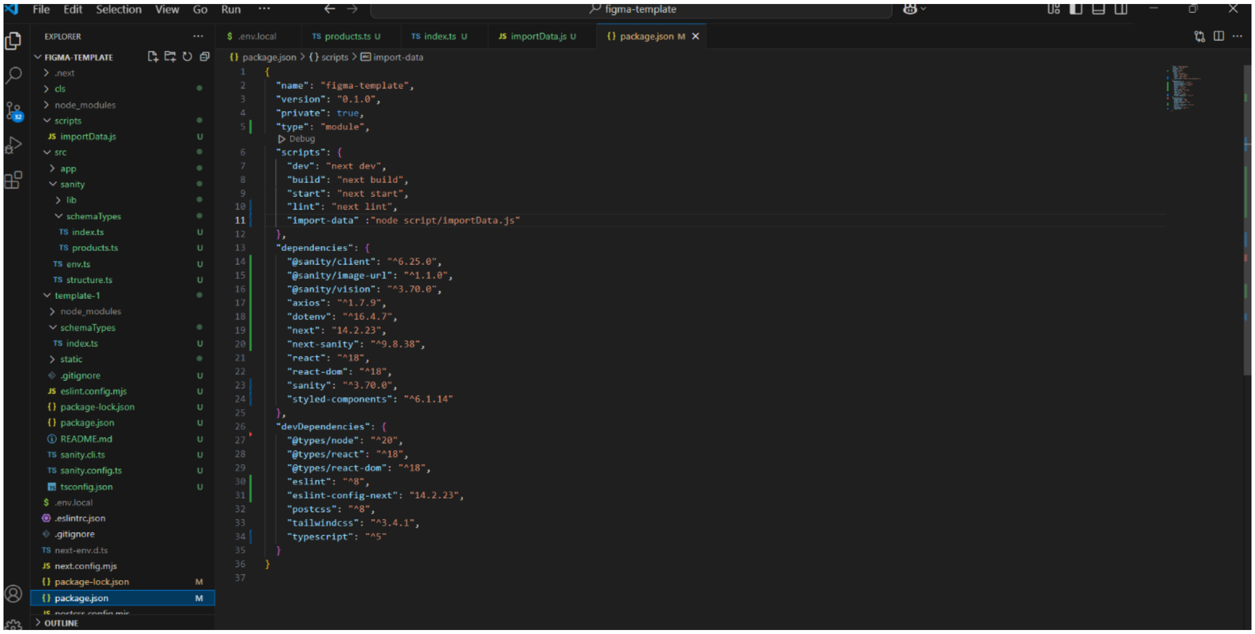
```
export const product= defineType({
    fields: [
            options:{
                list:[
                    {title: 'T-Shirt', value: 'tshirt'},
                    {title: 'Short', value: 'short'},
                    {title: 'Jeans', value: 'jeans'} ,
                    {title: 'Hoddie', 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'}
            ]
        },
        {
            name:"sizes",
            title:"Sizes",
            type: 'array',
```



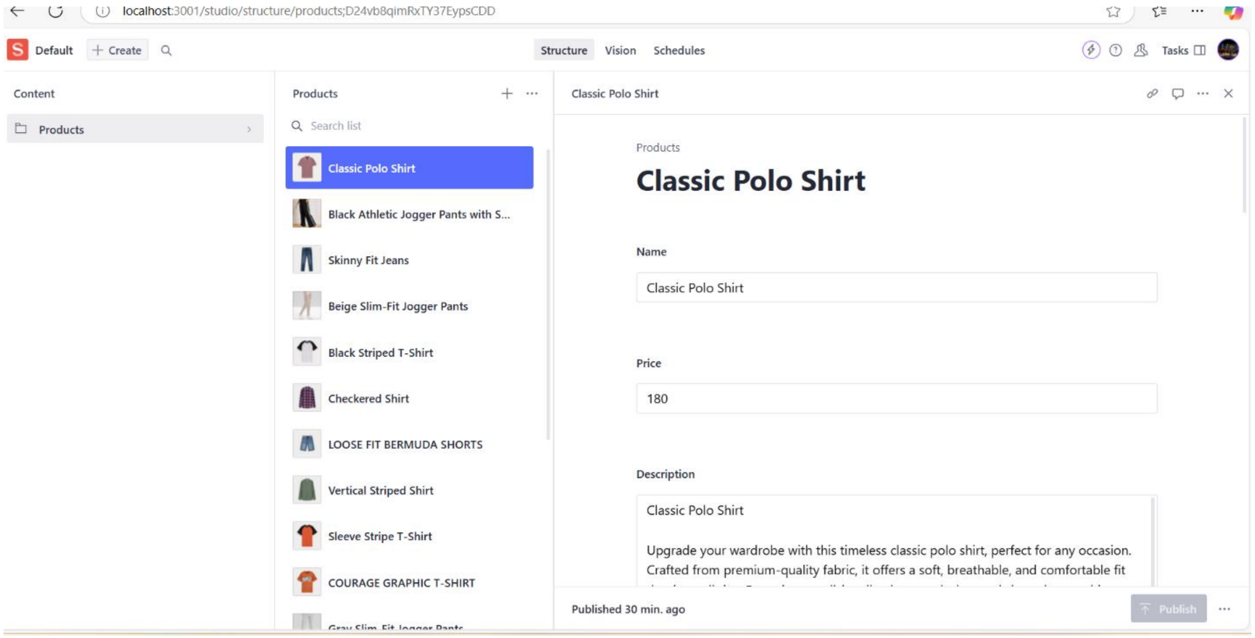
```
import { defineType, type SchemaTypeDefinition } from 'sanity'
import { product } from './products'

export const schema: { types: SchemaTypeDefinition[] } = {
  types:  [product],
}

```



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

const client = createClient({
  projectId: 'h5g91q5j',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token: 'skgX4M18jjFuy0JAHAqeAo5pI3Brop3Y5bz8OLNdLLQ6mZonTAMgAC2cFW3Gc0HkhD2Rb7Li4gfPPqXEilkcIBG5o7vE0ACgmPa05vxwNOkz9r24H2uS5oBN2Nhd060HX08V22KF3P1DBjV4EKNdww9F
});

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;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'products',
        name: product.name,
```



```json
{
  "name": "figma-template",
  "version": "0.1.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data" :"node script/importData.js"
  },
  "dependencies": {
    "@sanity/client": "^6.25.0",
    "@sanity/image-url": "^1.1.0",
    "@sanity/vision": "^3.70.0",
    "axios": "^1.7.9",
    "dotenv": "^16.4.7",
    "next": "14.2.23",
    "next-sanity": "^9.8.38",
    "react": "^18",
    "react-dom": "^18",
    "sanity": "^3.70.0",
    "styled-components": "^6.1.14"
  },
  "devDependencies": {
    "@types/node": "^20",
    "@types/react": "^18",
    "@types/react-dom": "^18",
    "eslint": "^8",
    "eslint-config-next": "14.2.23",
    "postcss": "^8",
    "tailwindcss": "^3.4.1",
    "typescript": "^5"
  }
}
```



Classic Polo Shirt

Name: Classic Polo Shirt

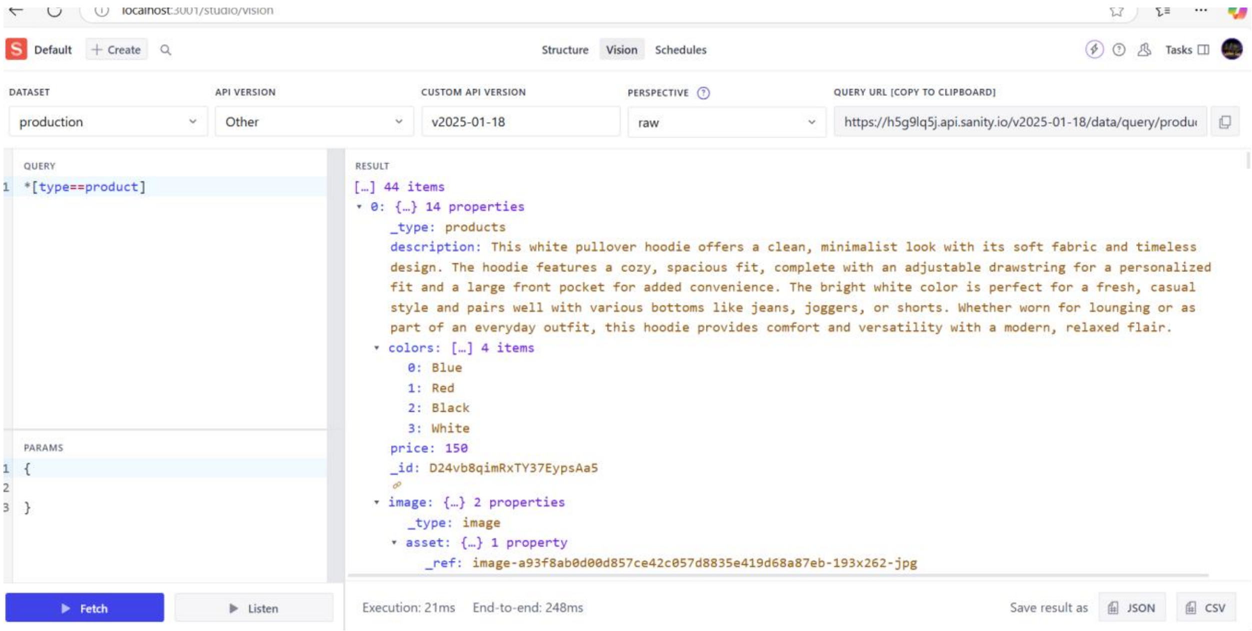
Price: 180

Description:

Classic Polo Shirt

Upgrade your wardrobe with this timeless classic polo shirt, perfect for any occasion. Crafted from premium-quality fabric, it offers a soft, breathable, and comfortable fit

Published 30 min. ago



DATASET: production

API VERSION: Other

CUSTOM API VERSION: v2025-01-18

PERSPECTIVE: raw

QUERY URL [COPY TO CLIPBOARD]: https://h5g9lq5j.api.sanity.io/v2025-01-18/data/query/produ

QUERY

```
*[type==product]
```

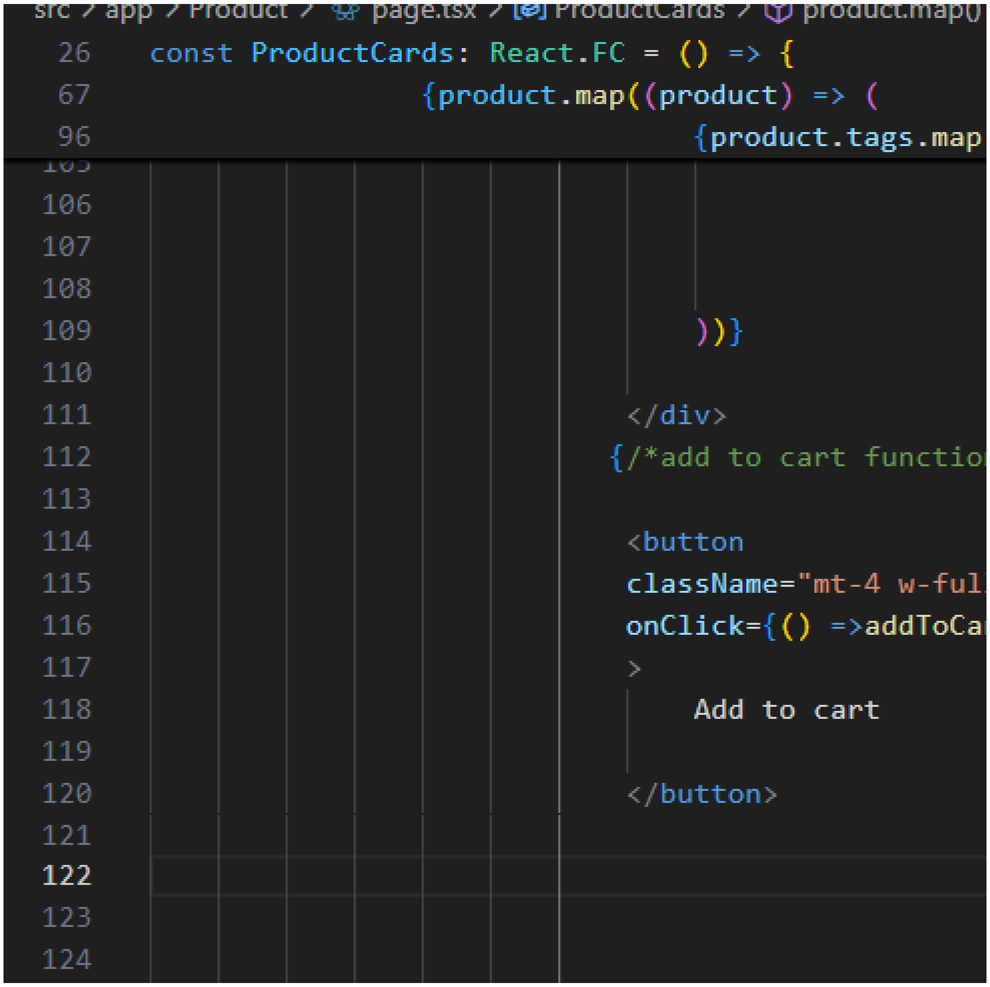
PARAMS

```
{

}
```

RESULT

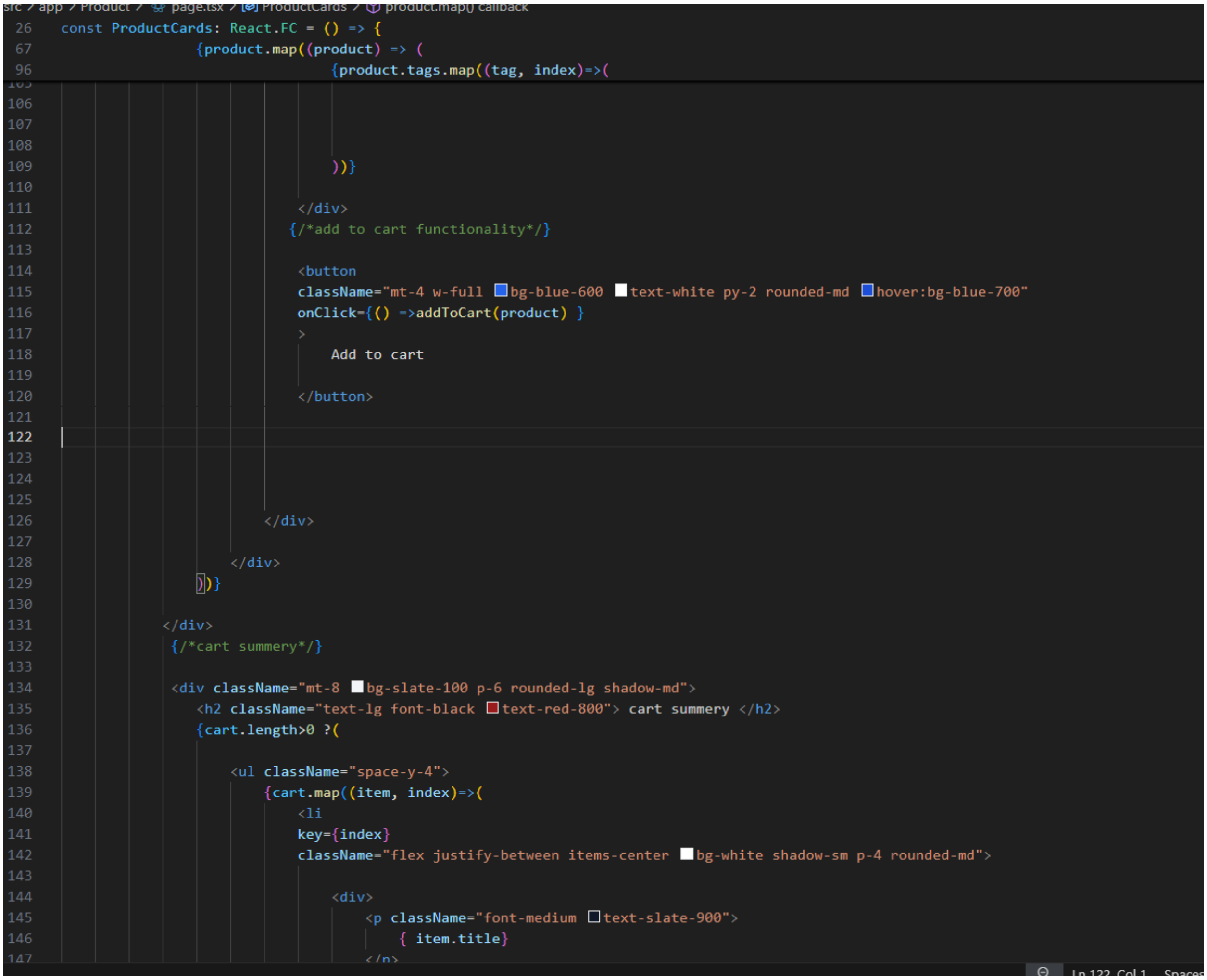
```
[…] 44 items
▾ 0: {…} 14 properties
    _type: products
    description: This white pullover hoodie offers a clean, minimalist look with its soft fabric and timeless
    design. The hoodie features a cozy, spacious fit, complete with an adjustable drawstring for a personalized
    fit and a large front pocket for added convenience. The bright white color is perfect for a fresh, casual
    style and pairs well with various bottoms like jeans, joggers, or shorts. Whether worn for lounging or as
    part of an everyday outfit, this hoodie provides comfort and versatility with a modern, relaxed flair.
  ▾ colors: […] 4 items
      0: Blue
      1: Red
      2: Black
      3: White
    price: 150
    _id: D24vb8qimRxTY37EypsAa5
  ▾ image: {…} 2 properties
      _type: image
    ▾ asset: {…} 1 property
        _ref: image-a93f8ab0d00d857ce42c057d8835e419d68a87eb-193x262-jpg
```

Execution: 21ms End-to-end: 248ms

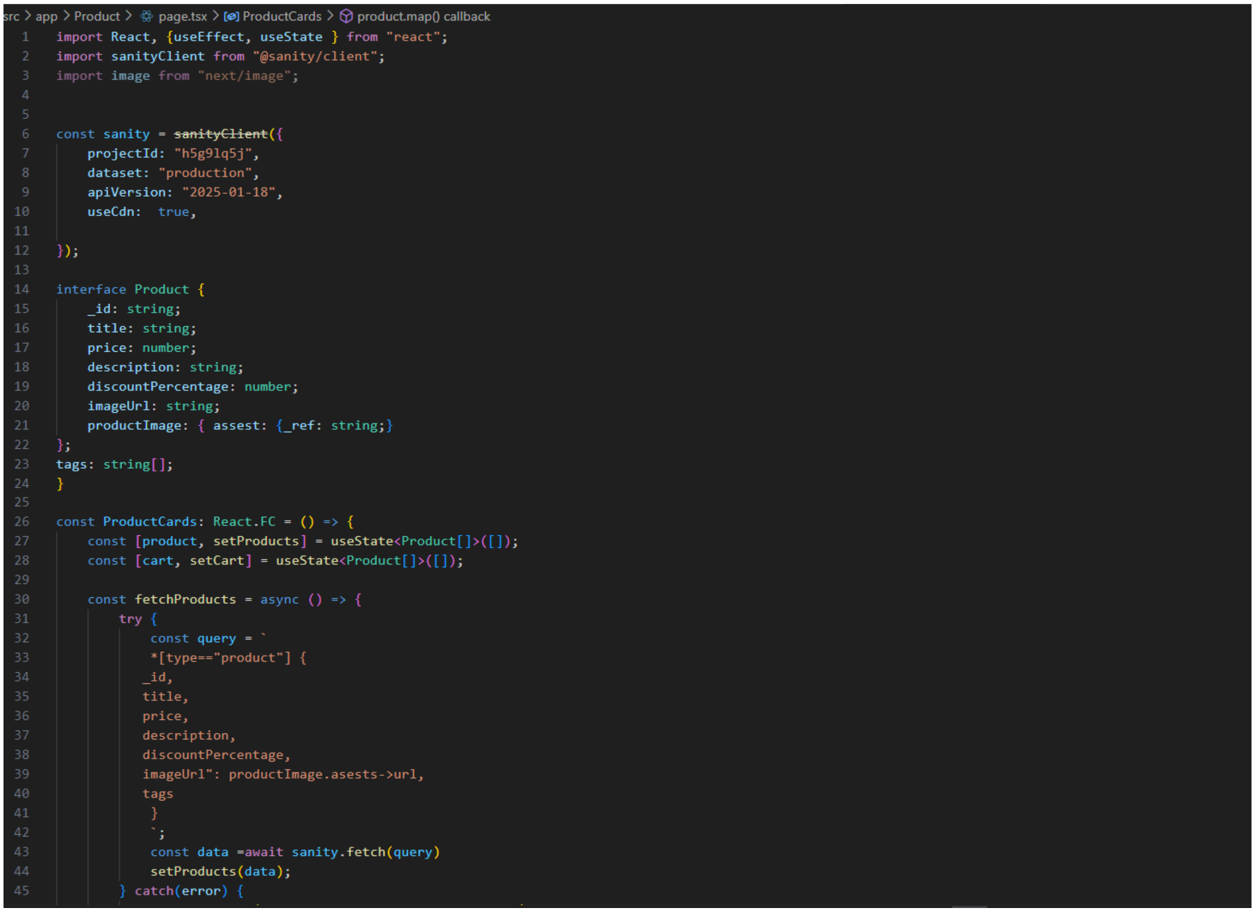


src > app > Product > page.tsx > ProductCards > product.map()

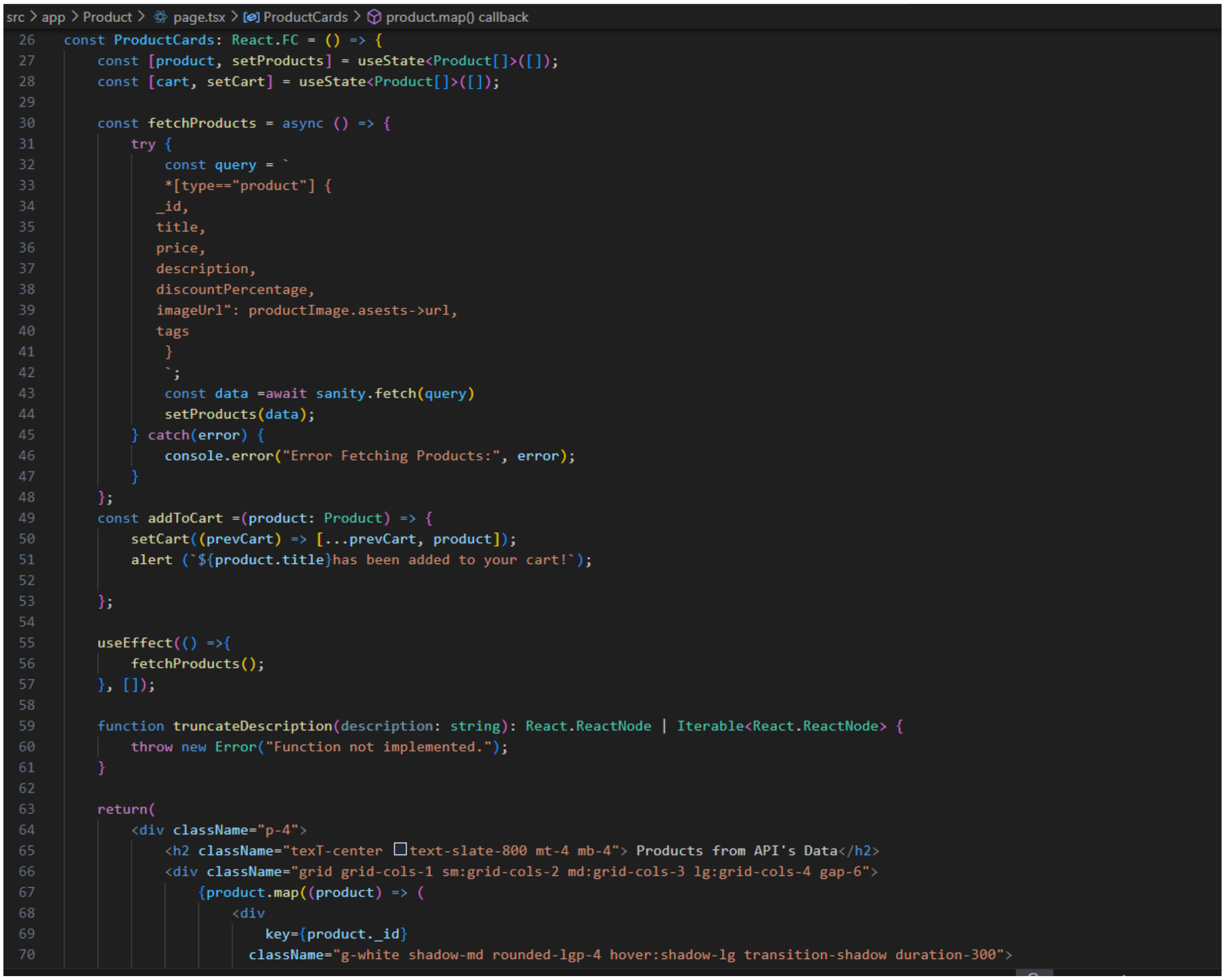
```
const ProductCards: React.FC = () => {
                {product.map((product) => (
                                {product.tags.map




                                ))}

                            </div>
                           {/*add to cart functio

                            <button
                            className="mt-4 w-ful
                            onClick={() =>addToCa
                            >
                                Add to cart

                            </button>




```


src > app > Product > page.tsx > ProductCards > product.map() callback
Ln 122, Col 1
Spaces

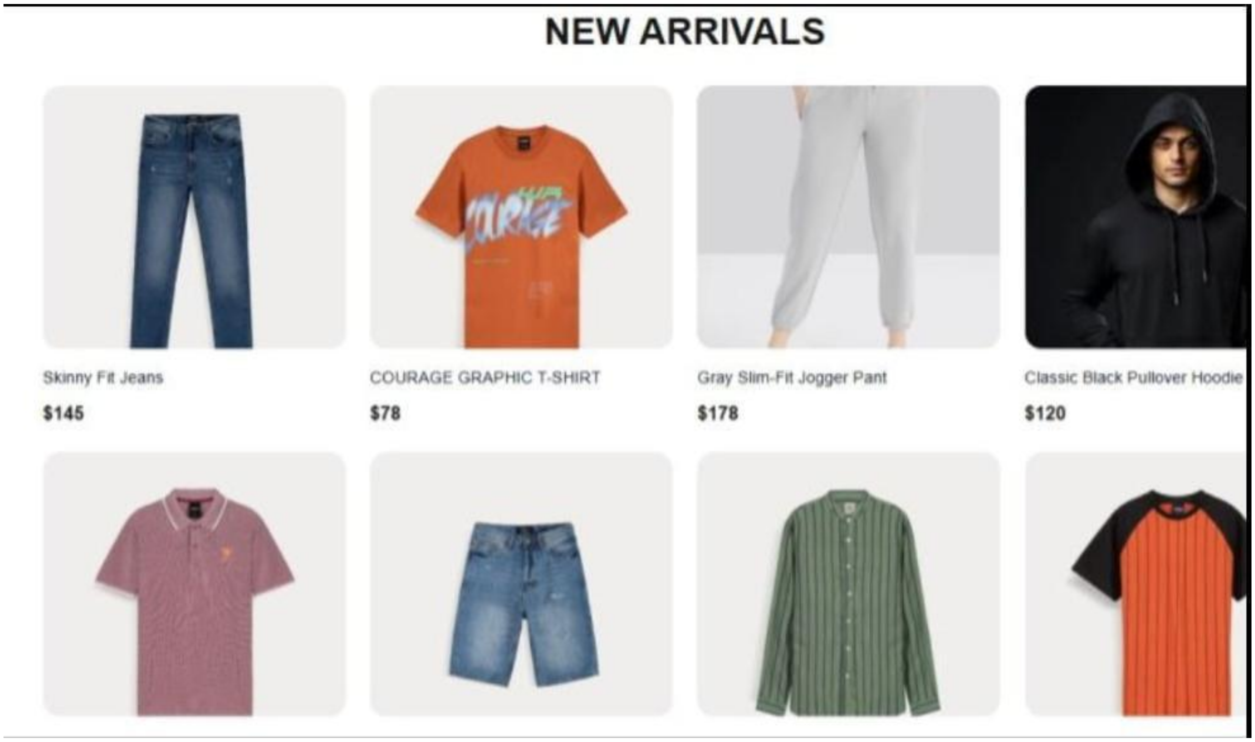
```
const ProductCards: React.FC = () => {
                {product.map((product) => (
                                {product.tags.map((tag, index)=>(



                                ))}

                            </div>
                           {/*add to cart functionality*/}

                            <button
                            className="mt-4 w-full bg-blue-600 text-white py-2 rounded-md hover:bg-blue-700"
                            onClick={() =>addToCart(product) }
                            >
                                Add to cart

                            </button>





                        </div>

                    </div>
                ))}

            </div>
             {/*cart summery*/}

             <div className="mt-8 bg-slate-100 p-6 rounded-lg shadow-md">
                <h2 className="text-lg font-black text-red-800"> cart summery </h2>
                {cart.length>0 ?(

                    <ul className="space-y-4">
                        {cart.map((item, index)=>(
                            <li
                            key={index}
                            className="flex justify-between items-center bg-white shadow-sm p-4 rounded-md">

                                <div>
                                    <p className="font-medium text-slate-900">
                                        { item.title}
                                    </p>
```


src > app > Product > page.tsx > ProductCards > product.map() callback

```
import React, {useEffect, useState } from "react";
import sanityClient from "@sanity/client";
import image from "next/image";


const sanity = ~~sanityClient~~({
    projectId: "h5g9lq5j",
    dataset: "production",
    apiVersion: "2025-01-18",
    useCdn:  true,

});

interface Product {
    _id: string;
    title: string;
    price: number;
    description: string;
    discountPercentage: number;
    imageUrl: string;
    productImage: { assest: {_ref: string;}
};
tags: string[];
}

const ProductCards: React.FC = () => {
    const [product, setProducts] = useState<Product[]>([]);
    const [cart, setCart] = useState<Product[]>([]);

    const fetchProducts = async () => {
        try {
            const query = `
            *[type=="product"] {
           _id,
           title,
           price,
           description,
           discountPercentage,
           imageUrl": productImage.asests->url,
           tags
            }
            `;
            const data =await sanity.fetch(query)
            setProducts(data);
        } catch(error) {
```


src > app > Product > page.tsx > ProductCards > product.map() callback

```
const ProductCards: React.FC = () => {
    const [product, setProducts] = useState<Product[]>([]);
    const [cart, setCart] = useState<Product[]>([]);

    const fetchProducts = async () => {
        try {
            const query = `
            *[type=="product"] {
           _id,
           title,
           price,
           description,
           discountPercentage,
           imageUrl": productImage.asests->url,
           tags
            }
            `;
            const data =await sanity.fetch(query)
            setProducts(data);
        } catch(error) {
            console.error("Error Fetching Products:", error);
        }
    };
    const addToCart =(product: Product) => {
        setCart((prevCart) => [...prevCart, product]);
        alert (`${product.title}has been added to your cart!`);

    };

    useEffect(() =>{
        fetchProducts();
    }, []);

    function truncateDescription(description: string): React.ReactNode | Iterable<React.ReactNode> {
        throw new Error("Function not implemented.");
    }

    return(
        <div className="p-4">
            <h2 className="texT-center □text-slate-800 mt-4 mb-4"> Products from API's Data</h2>
            <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
                {product.map((product) => (
                    <div
                        key={product._id}
                      className="g-white shadow-md rounded-lgp-4 hover:shadow-lg transition-shadow duration-300">
```



NEW ARRIVALS

Skinny Fit Jeans

$145

COURAGE GRAPHIC T-SHIRT

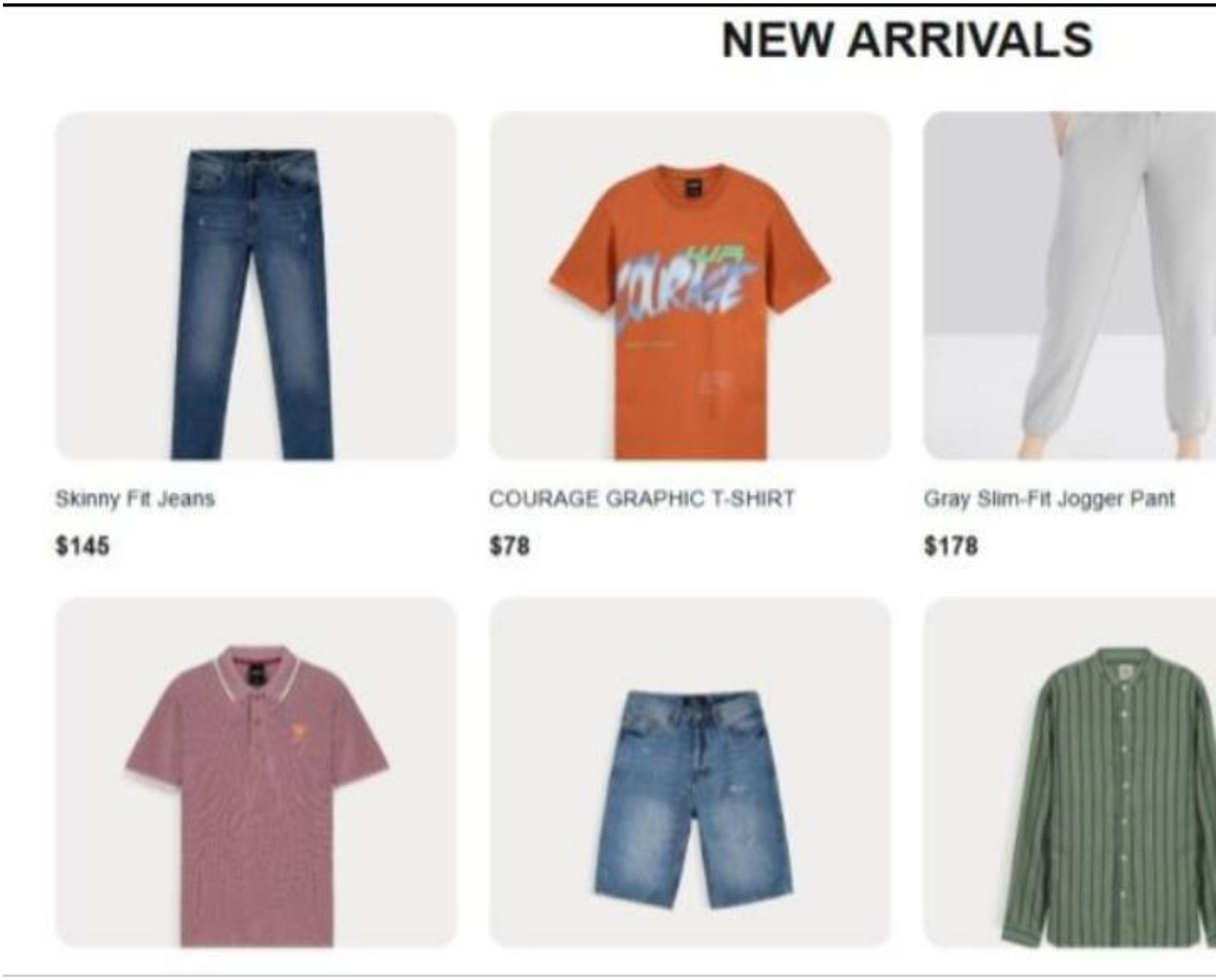
$78

Gray Slim-Fit Jogger Pant

$178

Classic Black Pullover Hoodie

$120



NEW ARRIVALS

Skinny Fit Jeans

$145

COURAGE GRAPHIC T-SHIRT

$78

Gray Slim-Fit Jogger Pant

$178