

Moiz Ansari
Roll No: 00202478
Timing: 7 to 10 PM (Friday)

Day 3 - API Integration and Data Migration

Objective

Day 3 focuses on integrating APIs and migrating data into **Sanity CMS** to develop a functional marketplace backend. The goal is to equip students with the skills to:

- Utilize APIs effectively.
- Migrate data into **Sanity CMS**.
- Ensure compatibility with existing templates.

This hands-on approach mirrors real-world practices and enables students to manage diverse client needs, including:

- **Headless API integration.**
 - **Migrating data from popular eCommerce platforms.**
-

Document Title:

Day 3 - API Integration Report - [Comforty Ecommerce]

Submission Requirements

What to Submit?

Students are required to submit a well-structured report covering the following aspects:

1. API Integration Process

- Explanation of the API integration approach.
- Tools and technologies used.
- Challenges encountered and solutions implemented.

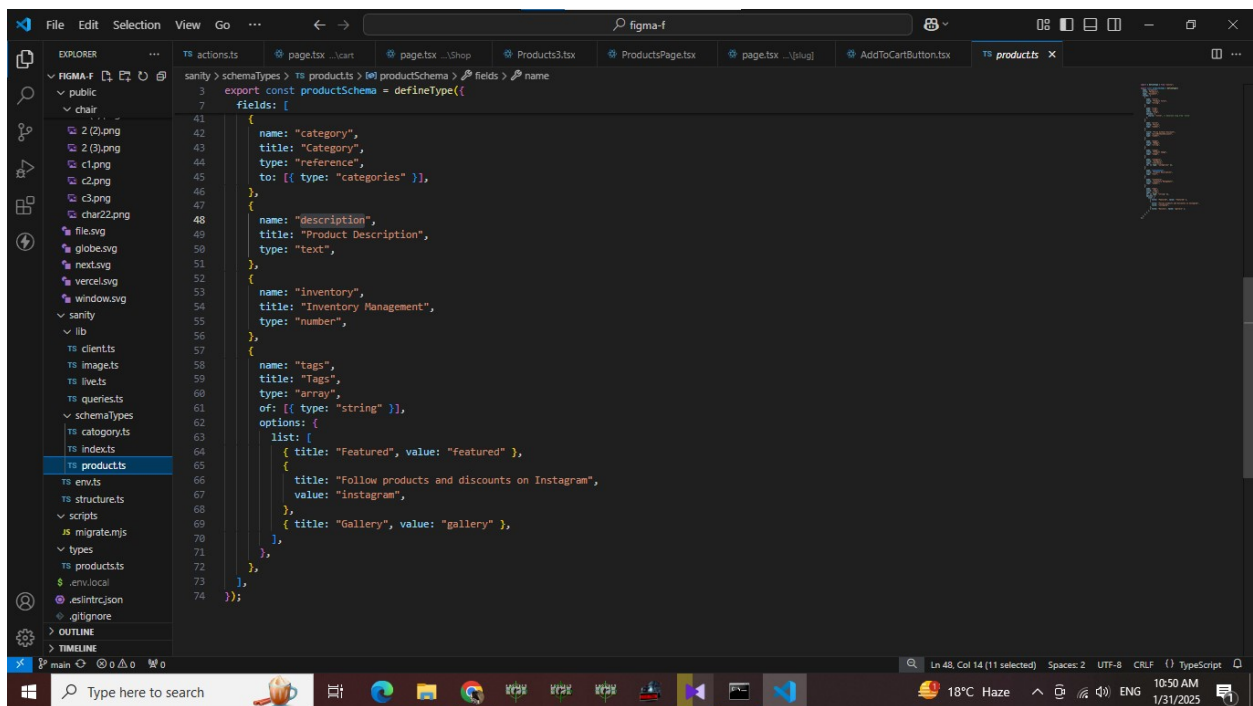
2. Schema Adjustments

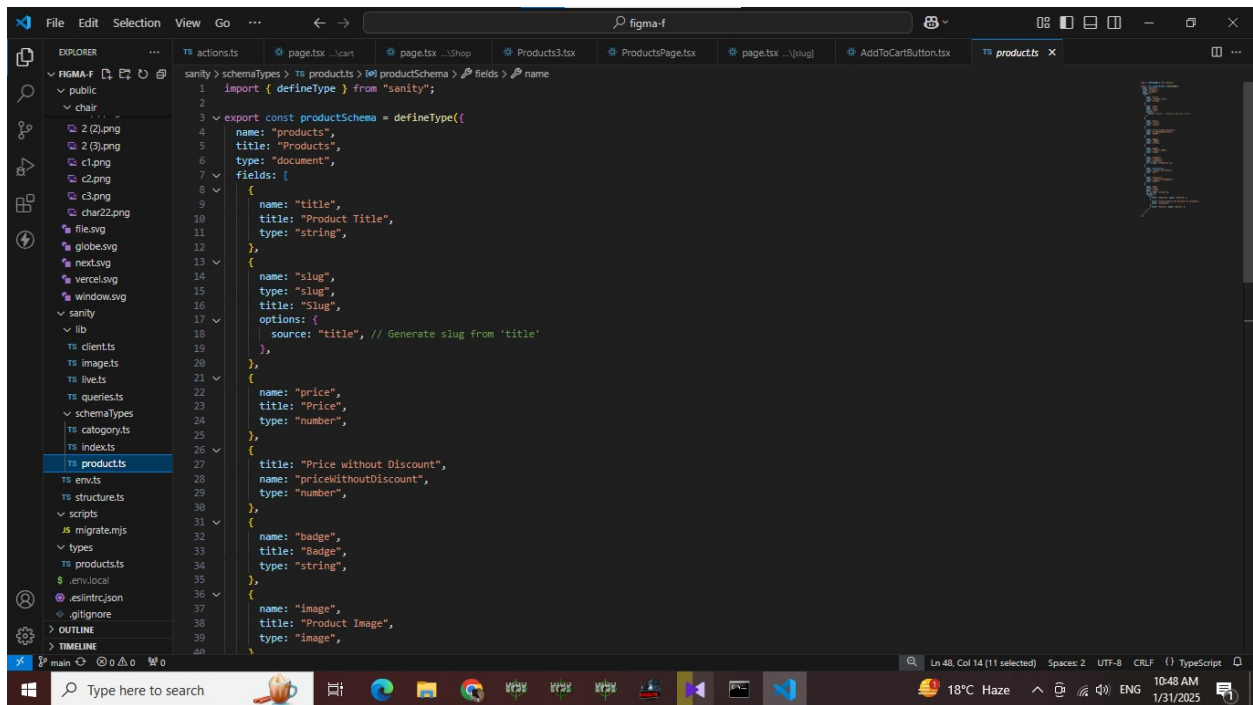
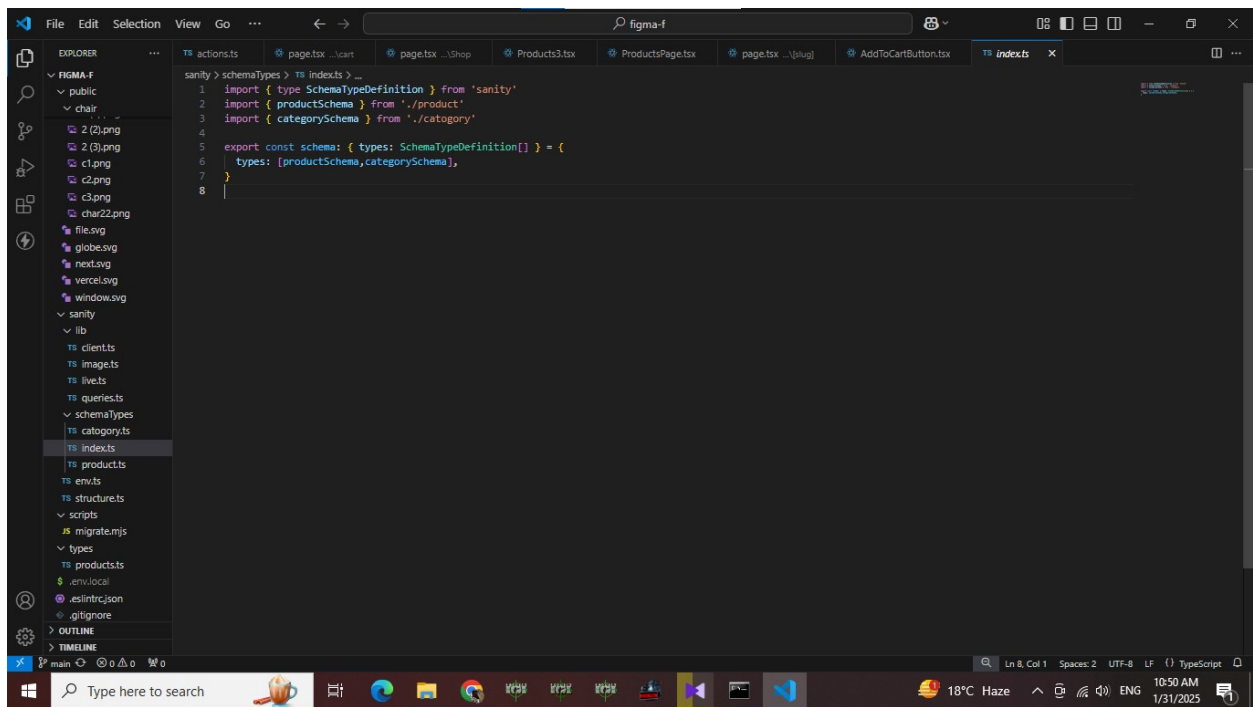
- Changes made to accommodate API data.
- Custom schema modifications.
- Justifications for the adjustments.

3. Migration Steps and Tools Used

- Step-by-step breakdown of the migration process.
- Software and tools utilized for migration.
- Ensuring compatibility and data integrity.

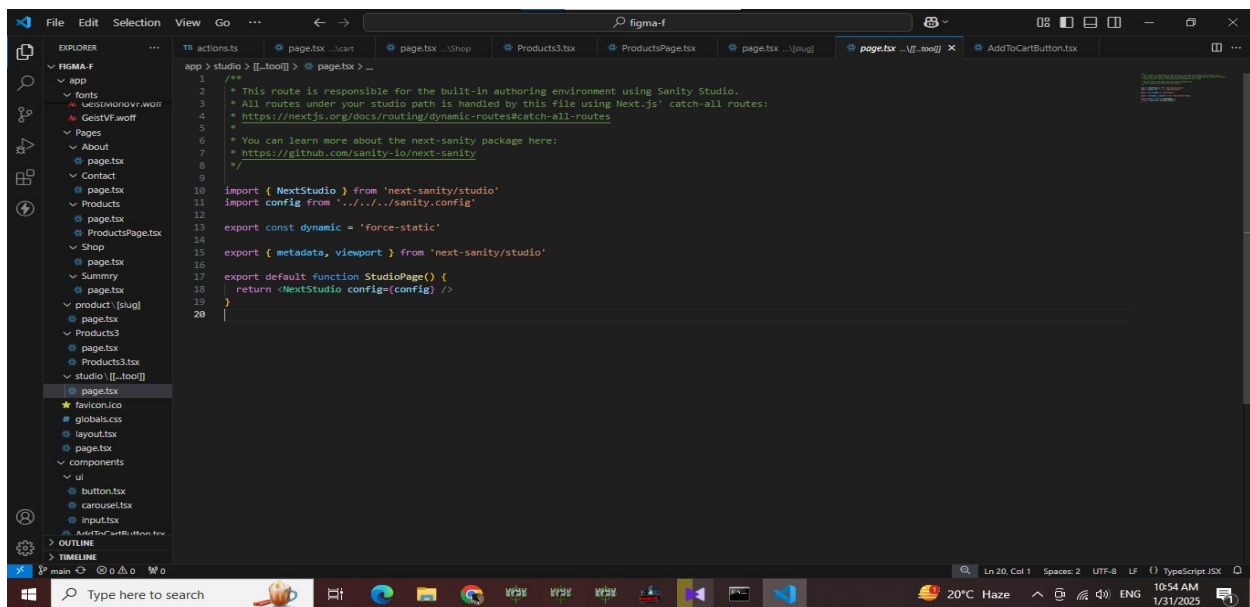
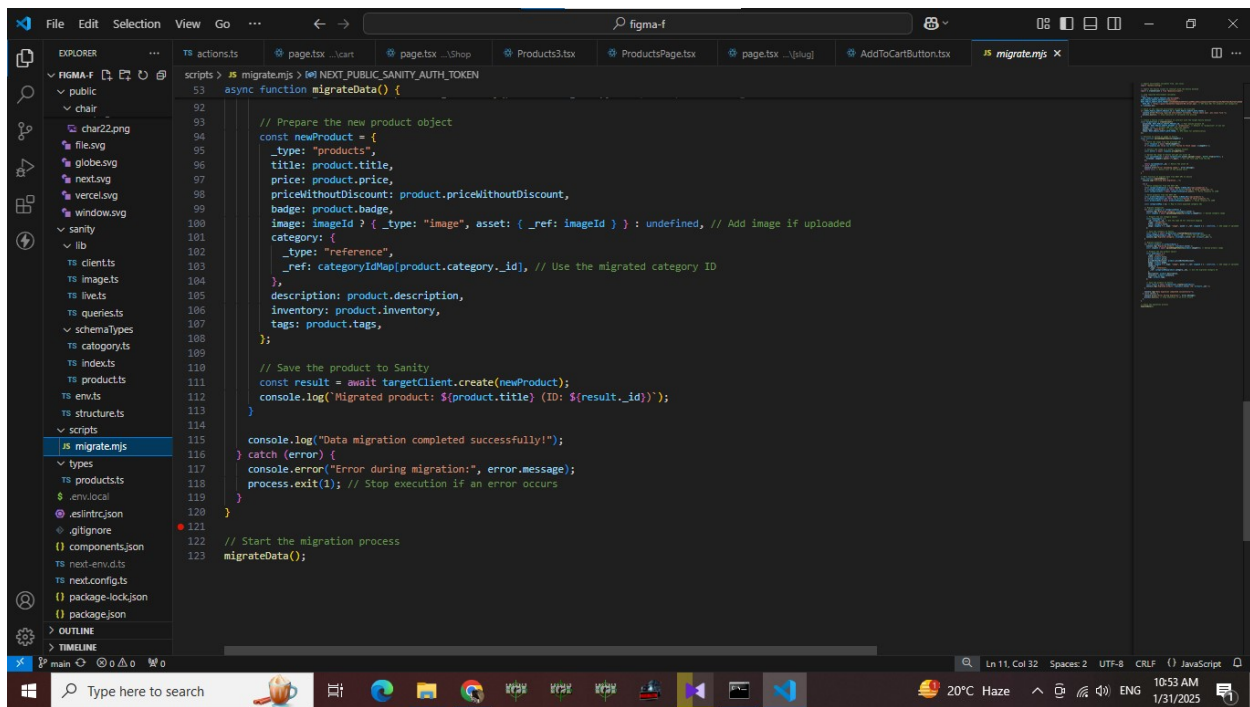
This report will serve as documentation of the API integration and data migration process, demonstrating an understanding of real-world development scenarios. Ensure the report is well-organized, concise, and detailed for proper evaluation.

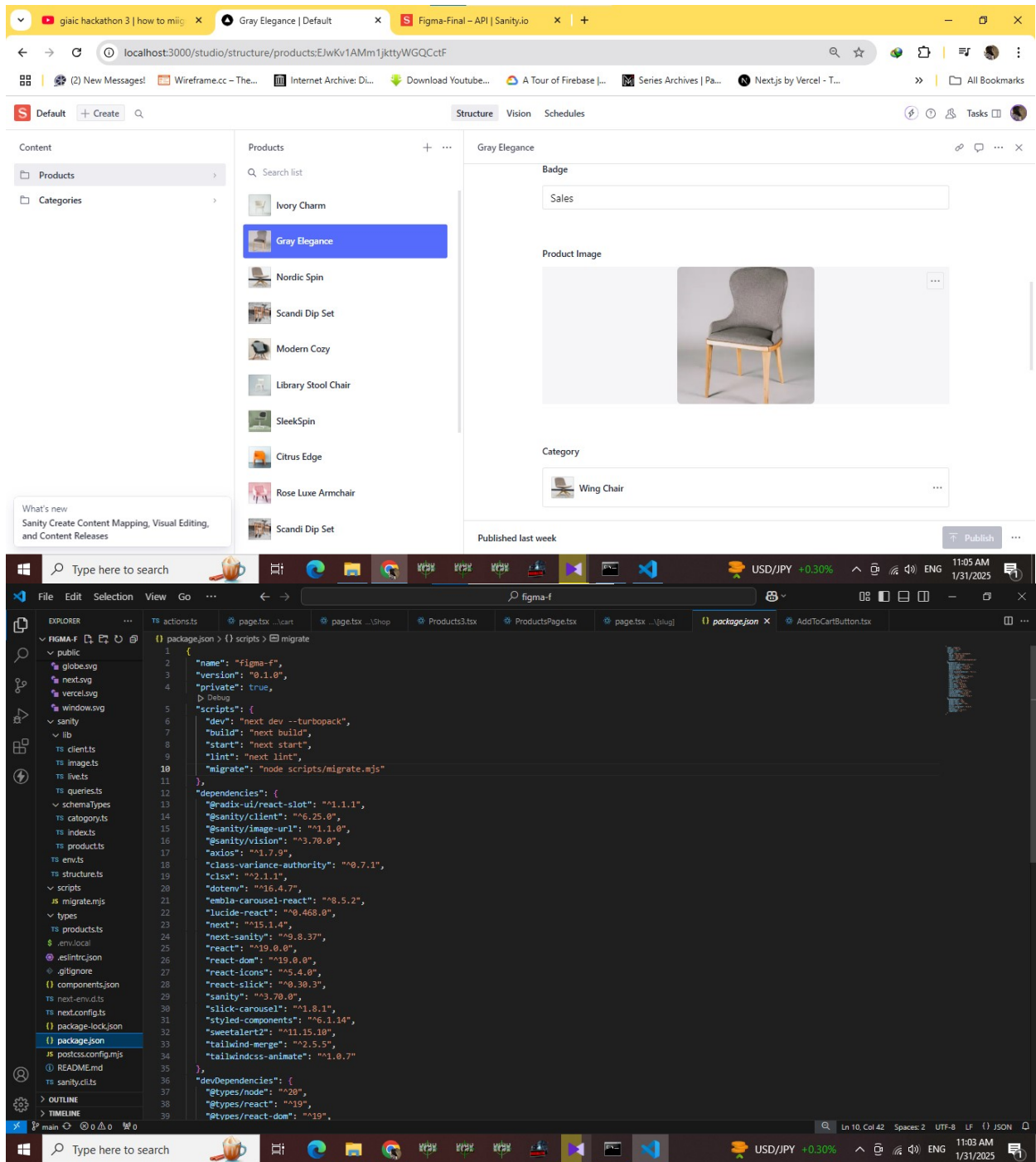




```
11 scripts > migrate.mjs > NEXT_PUBLIC_SANITY_AUTH_TOKEN
12 async function uploadImageToSanity(imageUrl) {
13   // Upload the image to Sanity and get its asset ID
14   const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
15     filename: imageUrl.split("/").pop(), // Use the file name from the URL
16   });
17   return uploadedAsset.id; // Return the asset ID
18 } catch (error) {
19   console.error("Error uploading image:", error.message);
20   return null; // Return null if the upload fails
21 }
22 // Main function to migrate data from REST API to Sanity
23 async function migrateData() {
24   console.log("Starting data migration...");
25   try {
26     // Fetch categories from the REST API
27     const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
28     if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
29     const categoriesData = await categoriesResponse.json(); // Parse response to JSON
30     // Fetch products from the REST API
31     const productsResponse = await fetch(`${BASE_URL}/api/products`);
32     if (!productsResponse.ok) throw new Error("Failed to fetch products.");
33     const productsData = await productsResponse.json(); // Parse response to JSON
34     const categoryIdMap = {}; // Map to store migrated category IDs
35     // Migrate categories
36     for (const category of categoriesData) {
37       console.log("Migrating category: ${category.title}");
38       const imageId = await uploadImageToSanity(category.imageUrl); // Upload category image
39       // Prepare the new category object
40       const newCategory = {
41         _id: category._id, // Use the same ID for reference mapping
42         type: "categories",
43         title: category.title,
44         image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
45       };
46       // Save the category to Sanity
47       const result = await targetClient.createOrReplace(newCategory);
48       categoryIdMap[category._id] = result._id; // Store the new category ID
49       console.log("Migrated category: ${category.title} (ID: ${result._id})");
50     }
51     // Migrate products
52     for (const product of productsData) {
53       console.log("Migrating product: ${product.title}");
54       const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image
55       // Prepare the new product object
56       const newProduct = {
57         _type: "products",
58         title: product.title,
59         price: product.price,
60         priceWithoutDiscount: product.priceWithoutDiscount,
61         badge: product.badge,
62         image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
63         category: {
64           _type: "reference",
65           _ref: categoryIdMap[product.category_id], // Use the migrated category ID
66         },
67         description: product.description,
68         inventory: product.inventory,
69         tags: product.tags,
70       };
71       // Save the product to Sanity
72     }
73   } catch (error) {
74     console.error("Error migrating data:", error.message);
75   }
76 }
77
```

```
53 scripts > migrate.mjs > NEXT_PUBLIC_SANITY_AUTH_TOKEN
54 async function migrateData() {
55   // Fetch categories from the REST API
56   const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
57   if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
58   const categoriesData = await categoriesResponse.json(); // Parse response to JSON
59   // Fetch products from the REST API
60   const productsResponse = await fetch(`${BASE_URL}/api/products`);
61   if (!productsResponse.ok) throw new Error("Failed to fetch products.");
62   const productsData = await productsResponse.json(); // Parse response to JSON
63   const categoryIdMap = {}; // Map to store migrated category IDs
64   // Migrate categories
65   for (const category of categoriesData) {
66     console.log("Migrating category: ${category.title}");
67     const imageId = await uploadImageToSanity(category.imageUrl); // Upload category image
68     // Prepare the new category object
69     const newCategory = {
70       _id: category._id, // Use the same ID for reference mapping
71       type: "categories",
72       title: category.title,
73       image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
74     };
75     // Save the category to Sanity
76     const result = await targetClient.createOrReplace(newCategory);
77     categoryIdMap[category._id] = result._id; // Store the new category ID
78     console.log("Migrated category: ${category.title} (ID: ${result._id})");
79   }
80   // Migrate products
81   for (const product of productsData) {
82     console.log("Migrating product: ${product.title}");
83     const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image
84     // Prepare the new product object
85     const newProduct = {
86       _type: "products",
87       title: product.title,
88       price: product.price,
89       priceWithoutDiscount: product.priceWithoutDiscount,
90       badge: product.badge,
91       image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
92       category: {
93         _type: "reference",
94         _ref: categoryIdMap[product.category_id], // Use the migrated category ID
95       },
96       description: product.description,
97       inventory: product.inventory,
98       tags: product.tags,
99     };
100     // Save the product to Sanity
101   }
102 }
103
```





S

Abdul Moiz

Figma-Final

16 days left in trial

Getting started

Overview

Members

Studios

Datasets

Access

Activity

Usage

Plan

API

Settings

Webhooks

CORS origins

Tokens

CORS origins

+ Add CORS origin

Hosts that can connect to the project API.

ORIGIN	CREDENTIALS	CREATED	
http://localhost:3000	Allowed	2 weeks	
http://localhost:3333	Allowed	2 weeks	

Tokens

+ Add API token

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

S

Default

+ Create

S

Content


Products


Categories


Products


+ ...


Search list


 Ivory Charm


 Gray Elegance


 Nordic Spin


 Scandi Dip Set


 Modern Cozy

 Library Stool Chair

 SleekSpin

 Citrus Edge

 Rose Luxe Armchair

 Scandi Dip Set

What's new

Sanity Create Content Mapping, Visual Editing, and Content Releases

