

Hackathon 3: Day\_3

# API Integration and Data Migration

---

## Table of Contents

1. Introduction
2. API Integration Process
3. Schema Adjustments
4. Migration Steps and Tools Used
5. Screenshots
6. Code Snippets
7. Conclusion

# 1. Introduction

This document provides a detailed overview of the API integration and data migration process completed on Day 3 of the Hackathon. The task involved integrating an external API providing product and category data, modifying the existing Sanity schemas, migrating data into Sanity CMS, and ensuring proper data retrieval and display on the front-end.

---

## 2. API Integration Process

### API Base URL:



```
BASE_URL = "https://giaic-hackathon-template-08.vercel.app"
```

### Endpoints:

- Products: `/api/products`

#### FULL URL:



```
https://giaic-hackathon-template-08.vercel.app/api/products
```

- Categories: `/api/categories`

#### FULL URL:

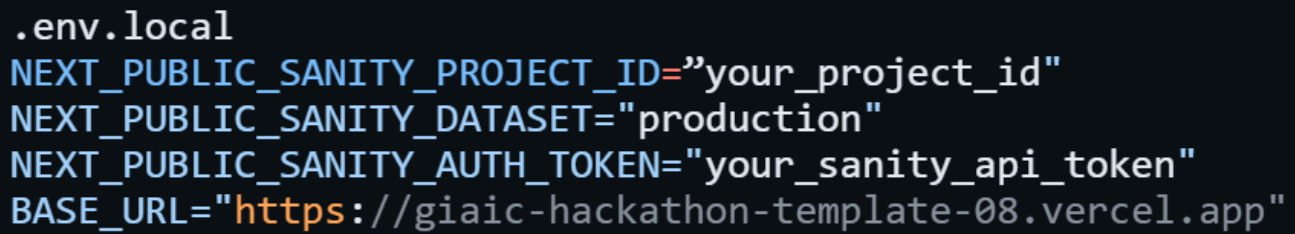


```
https://giaic-hackathon-template-08.vercel.app/api/categories
```



## Steps Taken:

1. **Tested API Endpoints** using Postman to ensure data retrieval.
2. **Set up environment variables** in .env.local:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays the content of a file named .env.local.

```
.env.local
NEXT_PUBLIC_SANITY_PROJECT_ID="your_project_id"
NEXT_PUBLIC_SANITY_DATASET="production"
NEXT_PUBLIC_SANITY_AUTH_TOKEN="your_sanity_api_token"
BASE_URL="https://giaic-hackathon-template-08.vercel.app"
```

### 3. Install Sanity in the Next.js project:

```
npx sanity@latest init
```

- Selected project
- Defined dataset ("production")
- Chose embedded /studio route

## Terminal:

```
E:\hackathon\comforty-marketplace>npx sanity@latest init
Need to install the following packages:
sanity@3.72.1
Ok to proceed? (y) y

✓ You are logged in as yousofhere.dev@gmail.com using Google
✓ Fetching existing projects

? Create a new project or select an existing one Create new project
? Your project name: comforty-marketplace
Your content will be stored in a dataset that can be public or private, depending on
whether you want to query your content with or without authentication.
The default dataset configuration has a public dataset named "production".
? Use the default dataset configuration? Yes
✓ Creating dataset
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes

⚠ It looks like you are using Next.js 15 and React 19
⚠ Please read our compatibility guide.
⚠ https://www.sanity.io/help/react-19

? 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'

added 918 packages, and audited 1293 packages in 9m

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

found 0 vulnerabilities

added 8 packages, and audited 1301 packages in 10s

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

found 0 vulnerabilities

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

E:\hackathon\comforty-marketplace>
```

### 3. Schema Adjustments

Initial Schema (Template 2):

Products Schema:

```
import { defineType, defineField } from "sanity";

export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    defineField({
      name: "category",
      title: "Category",
      type: "reference",
      to: [{ type: "category" }]
    }),
    defineField({ name: "name", title: "Title", type: "string" }),
    defineField({ name: "price", title: "Price", type: "number" }),
  ]
});
```

Categories Schema:

```
import { defineType, defineField } from "sanity";

export const Category = defineType({
  name: "category",
  title: "Category",
  type: "document",
  fields: [
    defineField({ name: "name", title: "Name", type: "string" })
  ]
});
```

Template2 changed to Template8, so the schema was also updated as follows:

Adjusted Schema (Template 8):

### Products Schema:

```
import { defineType } from "sanity";

export const products = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    { name: "title", title: "Product Title", type: "string" },
    { name: "price", title: "Price", type: "number" },
    { name: "badge", title: "Badge", type: "string" },
    { name: "image", title: "Product Image", type: "image" },
    { name: "category", title: "Category", type: "reference", to: [{ type: "categories" }] },
    { name: "tags", title: "Tags", type: "reference", to: [{ type: "tags" }] },
    { name: "description", title: "Product Description", type: "text" }
  ]
});
```

### Categories Schema:

```
import { defineType } from "sanity";

export const categories = defineType({
  name: 'categories',
  title: 'Categories',
  type: 'document',
  fields: [
    { name: 'title', title: 'Category Title', type: 'string' },
    { name: 'image', title: 'Category Image', type: 'image' }
  ]
});
```

## 4. Migration Steps and Tools Used

Tools Used:

- **Sanity CMS** for schema management
- **Next.js** for frontend display
- **Postman** for API testing
- **Node.js** for writing migration script

### Migration Script (scripts/migration.mjs):

Imports:

```
import * as dotenv from "dotenv";
dotenv.config({ path: ".env.local" });

import { createClient } from "@sanity/client";
```

Load required environment variables:

```
const {
  NEXT_PUBLIC_SANITY_PROJECT_ID, // Sanity project ID
  NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
  NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
  BASE_URL, // API base URL for products and categories
} = process.env;

// Check if the required environment variables are provided
if (!NEXT_PUBLIC_SANITY_PROJECT_ID || !NEXT_PUBLIC_SANITY_AUTH_TOKEN) {
  console.error("Missing required environment variables. Please check your .env.local file.");
  process.exit(1); // Stop execution if variables are missing
}
```

Create a Sanity client:



```
// Create a Sanity client instance to interact with the target Sanity dataset
const targetClient = createClient({
  projectId: NEXT_PUBLIC_SANITY_PROJECT_ID, // Your Sanity project ID
  dataset: NEXT_PUBLIC_SANITY_DATASET || "production", // Default to "production" if not set
  useCdn: false, // Disable CDN for real-time updates
  apiVersion: "2023-01-01", // Sanity API version
  token: NEXT_PUBLIC_SANITY_AUTH_TOKEN, // API token for authentication
});
```

## Function to upload an image to Sanity:

```
async function uploadImageToSanity(imageUrl) {
  try {
    // Fetch the image from the provided URL

    const response = await fetch(imageUrl);

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

    // Convert the image to a buffer (binary format)

    const buffer = await response.arrayBuffer();

    // Upload the image to Sanity and get its asset ID

    const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
      filename: imageUrl.split("/").pop(), // Use the file name from the URL
    });

    return uploadedAsset._id; // Return the asset ID
  }
  catch (error)
  {
    console.error("Error uploading image:", error.message);
    return null; // Return null if the upload fails
  }
}
```

## Main Function to fetch the Products and Categories from BASE\_URL and insert migrate to the Sanity CMS:

```

async function migrateData() {
  console.log("Starting data migration...");

  try {
    // Fetch categories from the REST API
    const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
    if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
    const categoriesData = await categoriesResponse.json(); // Parse response to JSON

    // Fetch products from the REST API
    const productsResponse = await fetch(`${BASE_URL}/api/products`);
    if (!productsResponse.ok) throw new Error("Failed to fetch products.");
    const productsData = await productsResponse.json(); // Parse response to JSON

    const categoryIdMap = {}; // Map to store migrated category IDs

    // Migrate categories
    for (const category of categoriesData) {
      console.log(`Migrating category: ${category.title}`);
      const imageId = await uploadImageToSanity(category.imageUrl); // Upload category image

      // Prepare the new category object
      const newCategory = {
        _id: category._id, // Use the same ID for reference mapping
        _type: "categories",
        title: category.title,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
      };

      // Save the category to Sanity
      const result = await targetClient.createOrReplace(newCategory);
      categoryIdMap[category._id] = result._id; // Store the new category ID
      console.log(`Migrated category: ${category.title} (ID: ${result._id})`);
    }

    // Migrate products
    for (const product of productsData) {
      console.log(`Migrating product: ${product.title}`);
      const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image

      // Prepare the new product object
      const newProduct = {
        _type: "products",
        title: product.title,
        price: product.price,
        priceWithoutDiscount: product.priceWithoutDiscount,
        badge: product.badge,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
        category: {
          _type: "reference",
          _ref: categoryIdMap[product.category._id], // Use the migrated category ID
        },
        description: product.description,
        inventory: product.inventory,
        tags: product.tags,
      };

      // Save the product to Sanity
      const result = await targetClient.create(newProduct);
      console.log(`Migrated product: ${product.title} (ID: ${result._id})`);
    }

    console.log("Data migration completed successfully!");
  } catch (error) {
    console.error("Error during migration:", error.message);
    process.exit(1); // Stop execution if an error occurs
  }
}

// Start the migration process

```

```
migrateData();
```

🌐 [comforty-furnitures/scripts/migrate.mjs at main · MYousuf-Cod...](#)

## Steps to Run Migration:

1. Navigate to the project root:

```
cd my-nextjs-project
```

Run the migration script:

```
node scripts/migration.mjs
```

## Terminal:

Check if the script is successfully fetching the data from an REST API and then is migrating category and product data to Sanity CMS:

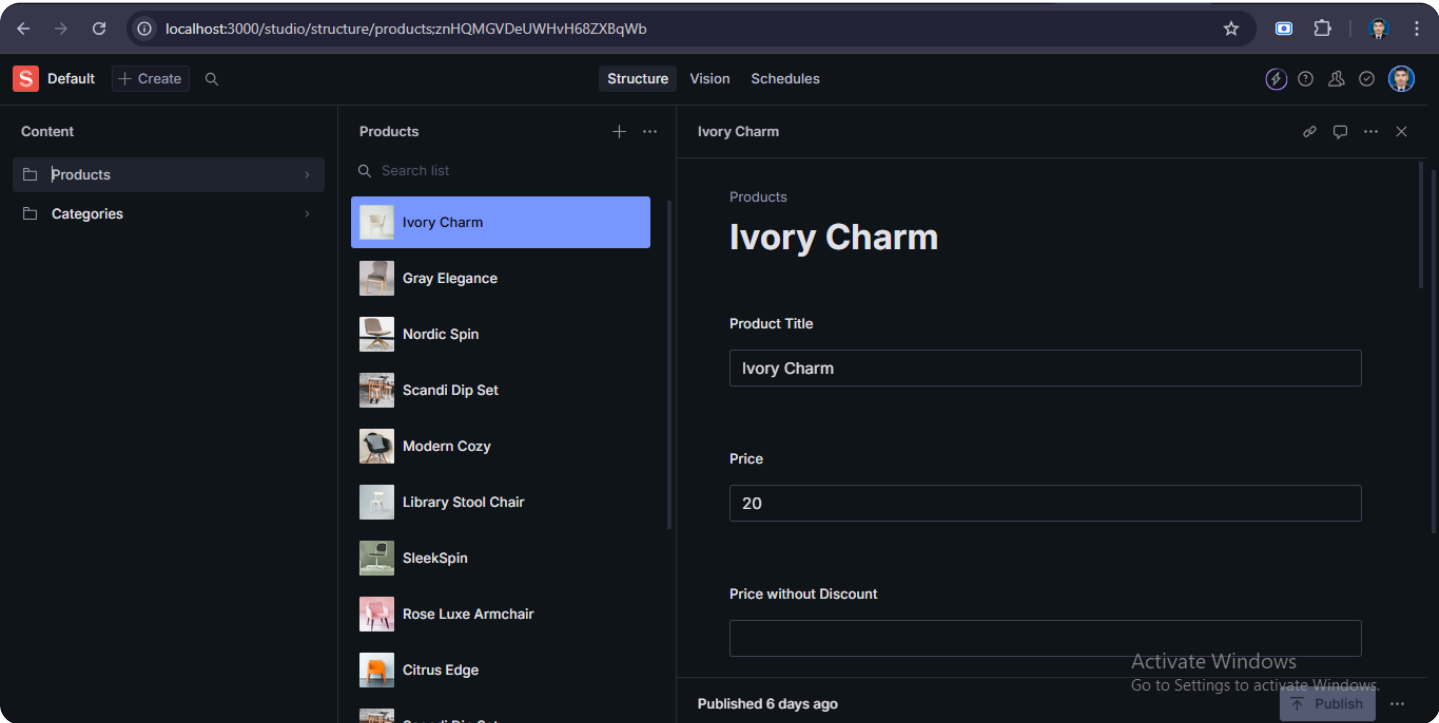
```
C:\Windows\System32\cmd.exe

> comferty-marketplace@0.1.0 migrate
> node scripts/migrate.mjs

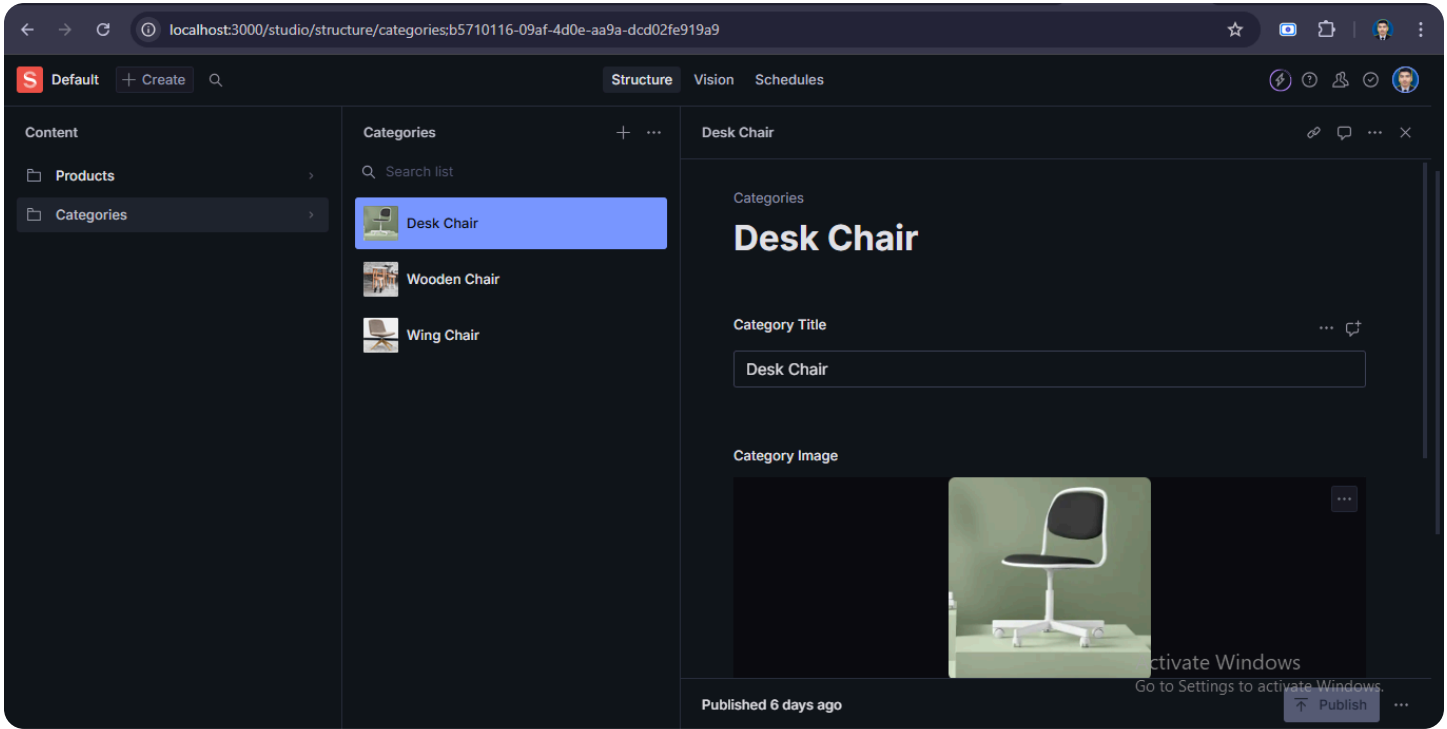
Starting data migration...
Migrating category: Wing Chair
Migrated category: Wing Chair (ID: 26fd7176-3c4d-40fc-a73a-3b85a9b5e15f)
Migrating category: Wooden Chair
Migrated category: Wooden Chair (ID: 407a8583-6203-4f61-becf-8e8b4c5461b6)
Migrating category: Desk Chair
Migrated category: Desk Chair (ID: b5710116-09af-4d0e-aa9a-dcd02fe919a9)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: 0zPjWAGkHgZUBjxT4g300b)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: 0zPjWAGkHgZUBjxT4g3QRD)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: znHQMVGDeUWHvH68ZX85GZ)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: znHQMVGDeUWHvH68ZX87TZ)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: znHQMVGDeUWHvH68ZX87pF)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: znHQMVGDeUWHvH68ZX88TR)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: znHQMVGDeUWHvH68ZX897D)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: znHQMVGDeUWHvH68ZX89kz)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: 0zPjWAGkHgZUBjxT4g3mAN)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: 0zPjWAGkHgZUBjxT4g3mZh)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: znHQMVGDeUWHvH68ZX8d71)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: znHQMVGDeUWHvH68ZX8d1X)
Migrating product: Nordic Spin
Migrated product: Nordic Spin (ID: znHQMVGDeUWHvH68ZX8dyn)
Migrating product: Gray Elegance
Migrated product: Gray Elegance (ID: znHQMVGDeUWHvH68ZX8q1f)
Migrating product: Ivory Charm
Migrated product: Ivory Charm (ID: znHQMVGDeUWHvH68ZX8qwb)
Data migration completed successfully!
```

Categories and Products are successfully fetched and migrated to the Sanity CMS:

## Populated Products in Sanity:



## Populated Categories in Sanity:

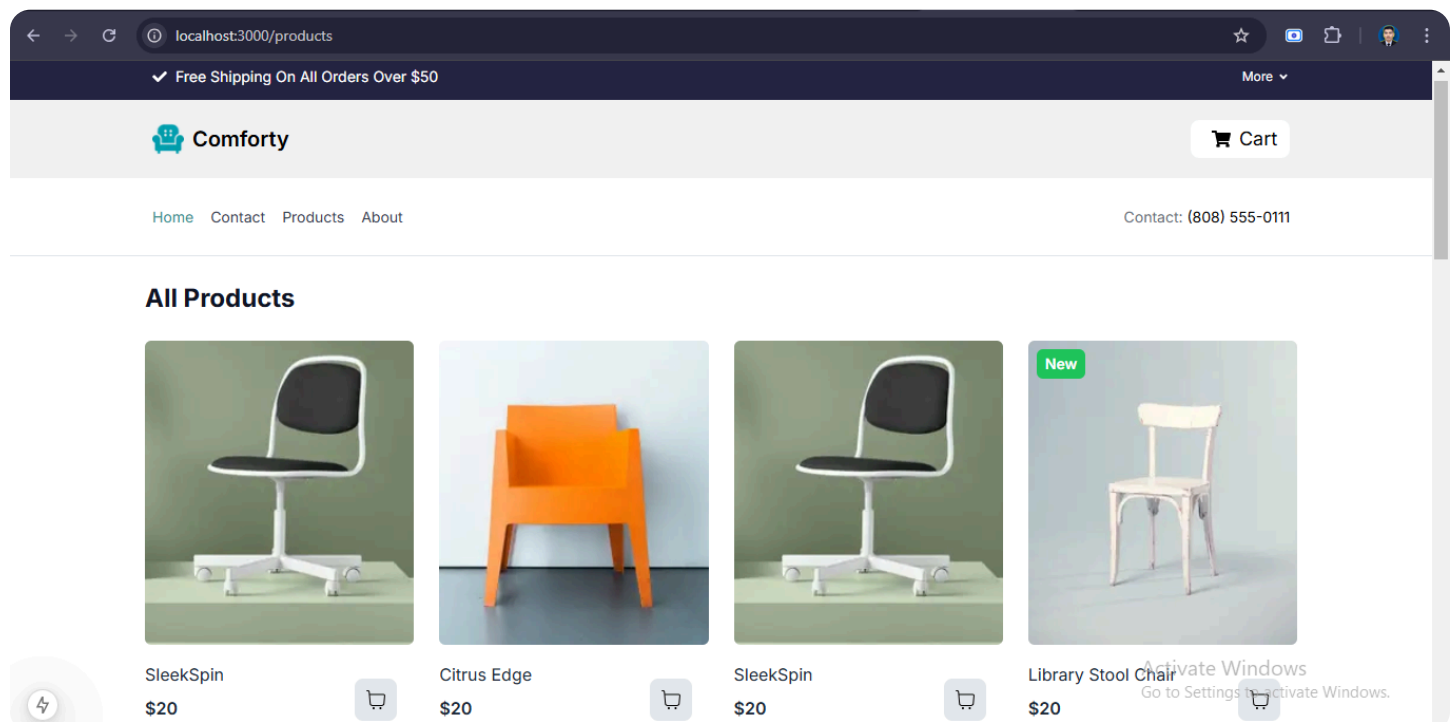


# Fetching and Displaying the Data in Front-End:

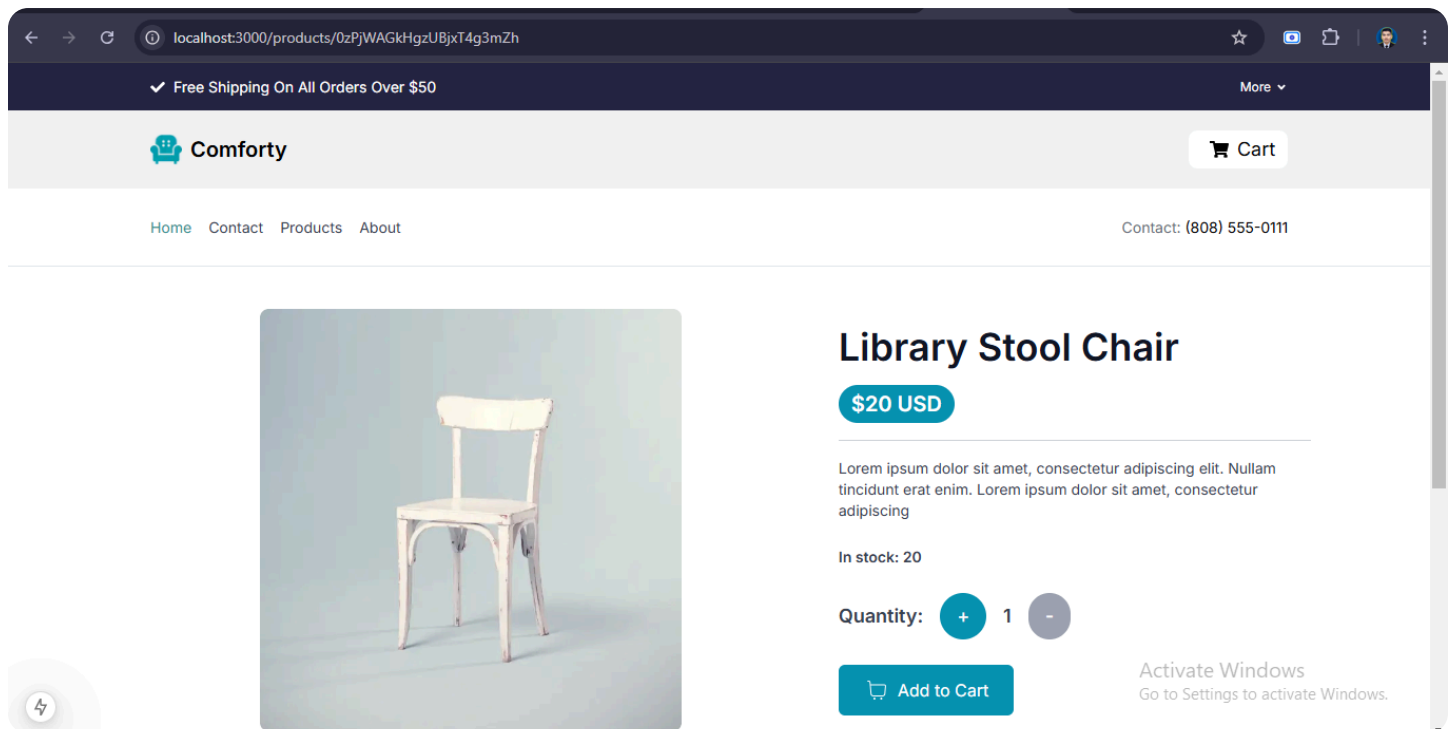
```
// sample code
import { groq } from "next-sanity";
import { sanityClient } from "../lib/sanity";

export async function getStaticProps() {
  const query = groq`*[_type == "products"] { title, price, image, category->title }`;
  const products = await sanityClient.fetch(query);
  return { props: { products } };
}
```

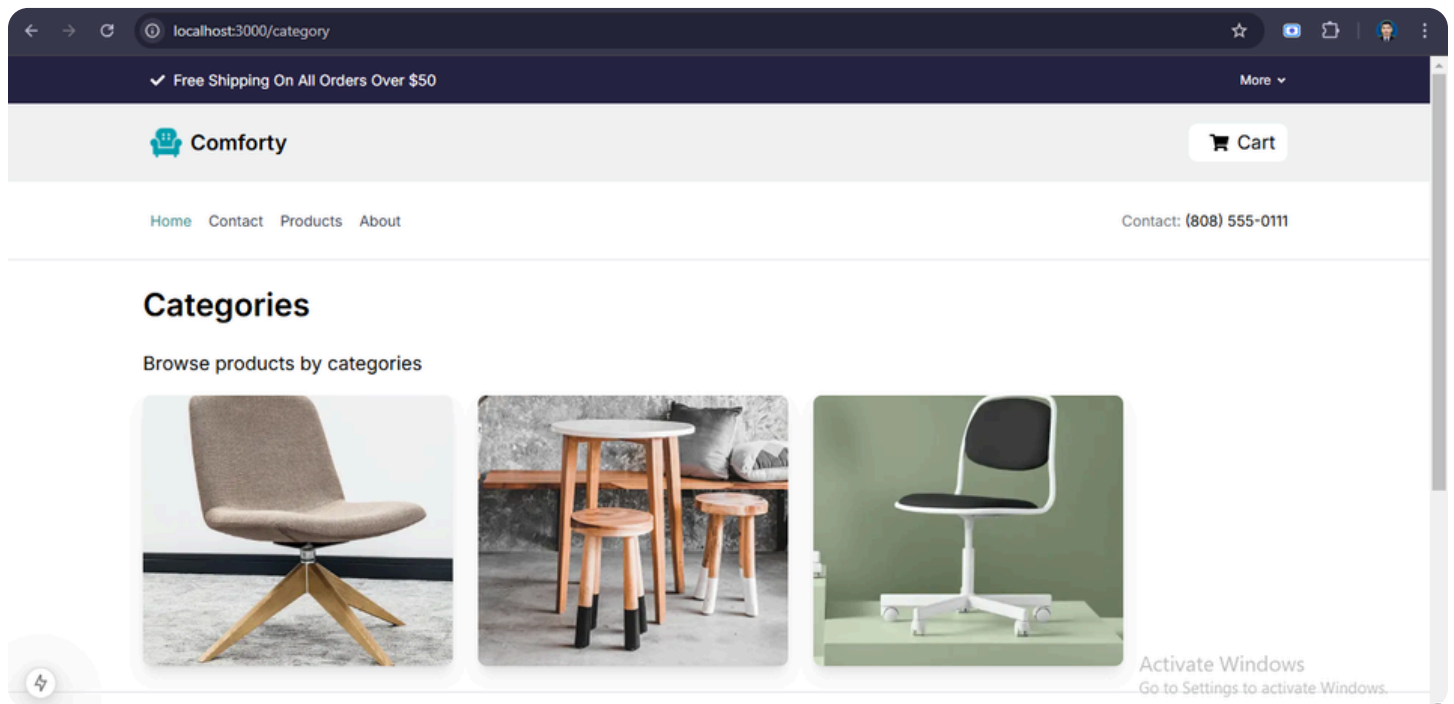
## Products Listing Page:



## Products Detail Page:



## Categories Page:



## Categories Detail Page

## 7. Conclusion

This documentation outlines the successful integration of an external API into the Next.js Sanity-based marketplace project. The process involved:

- Testing and integrating the API.
- Modifying Sanity schemas to align with the API structure.
- Implementing a migration script to import data into Sanity CMS.
- Displaying the migrated data on the frontend using GROQ queries.

**The implementation ensures smooth data management and dynamic content updates within the marketplace.**