Day 3: API Integration Report – Comforty

1. Introduction

This report highlights the process of integrating external data from a REST API into Sanity CMS using the Sanity client to enable seamless content management. The goal is to automate the addition and management of products and categories in the backend, ensuring they are displayed efficiently on the Next.js frontend.

Key Steps:

1. Fetching product and category data from an external API.
2. Migrating the data into Sanity CMS.
3. Ensuring smooth synchronization for the frontend.

---

2. API Integration Overview

API Details

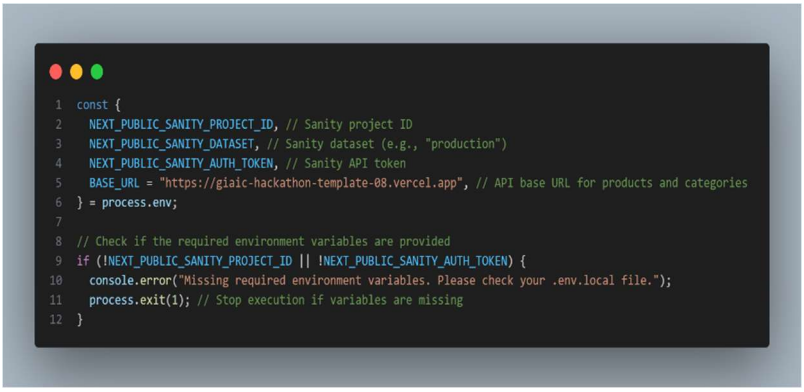
- **API Name:** External Marketplace API
- **Base URI:** https://giaic-hackathon-template-08.vercel.app

Data Fetched

1. **Products**: Includes titles, prices, descriptions, categories, images, and inventory.
2. **Categories**: Associated with products for proper organization.

Sanity Client

The Sanity client is used to interact with Sanity CMS. It enables uploading and managing fetched data within corresponding schemas, making it easy to structure and display the data in the frontend.



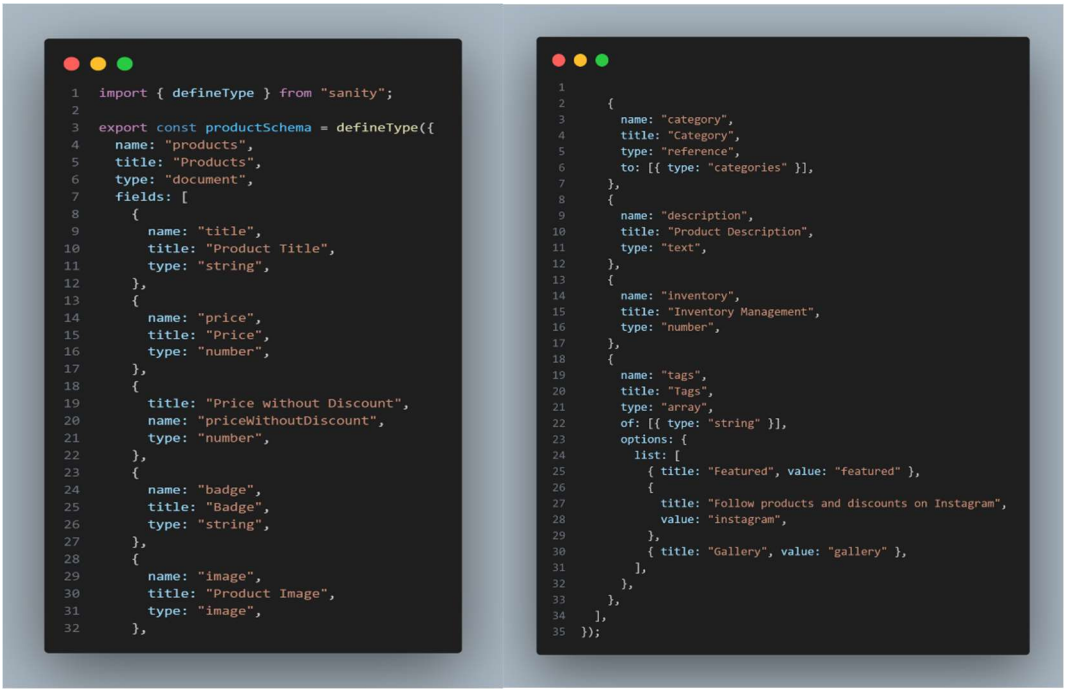
```
const {
  NEXT_PUBLIC_SANITY_PROJECT_ID, // Sanity project ID
  NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
  NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
  BASE_URL = "https://giaic-hackathon-template-08.vercel.app", // API base URL for products and categories
} = process.env;

// 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
}
```

3. Schema Structure Changes

The original product schema was basic, containing only essential fields like name, price, stock, and description. The updated schema has been significantly expanded to enhance product management capabilities and includes the following fields:

New Schema Fields:

1. **title (string)**: Product's title (formerly name in the old schema).
2. **priceWithoutDiscount (number)**: Price of the product without discounts.
3. **badge (string)**: Tags such as "New", "Sale", or other promotional labels.
4. **image (image)**: Allows uploading a product image for each product.
5. **category (reference)**: References category documents to organize products.
6. **inventory (number)**: Manages stock levels for products (replaces the stock field).
7. **tags (array)**: Includes predefined tags like "Featured", "Instagram", or "Gallery".



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

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Product Title",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      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: "description",
      title: "Product Description",
      type: "text",
    },
    {
      name: "inventory",
      title: "Inventory Management",
      type: "number",
    },
    {
      name: "tags",
      title: "Tags",
      type: "array",
      of: [{ type: "string" }],
      options: {
        list: [
          { title: "Featured", value: "featured" },
          {
            title: "Follow products and discounts on Instagram",
            value: "instagram",
          },
          { title: "Gallery", value: "gallery" },
        ],
      },
    },
  ],
});
```

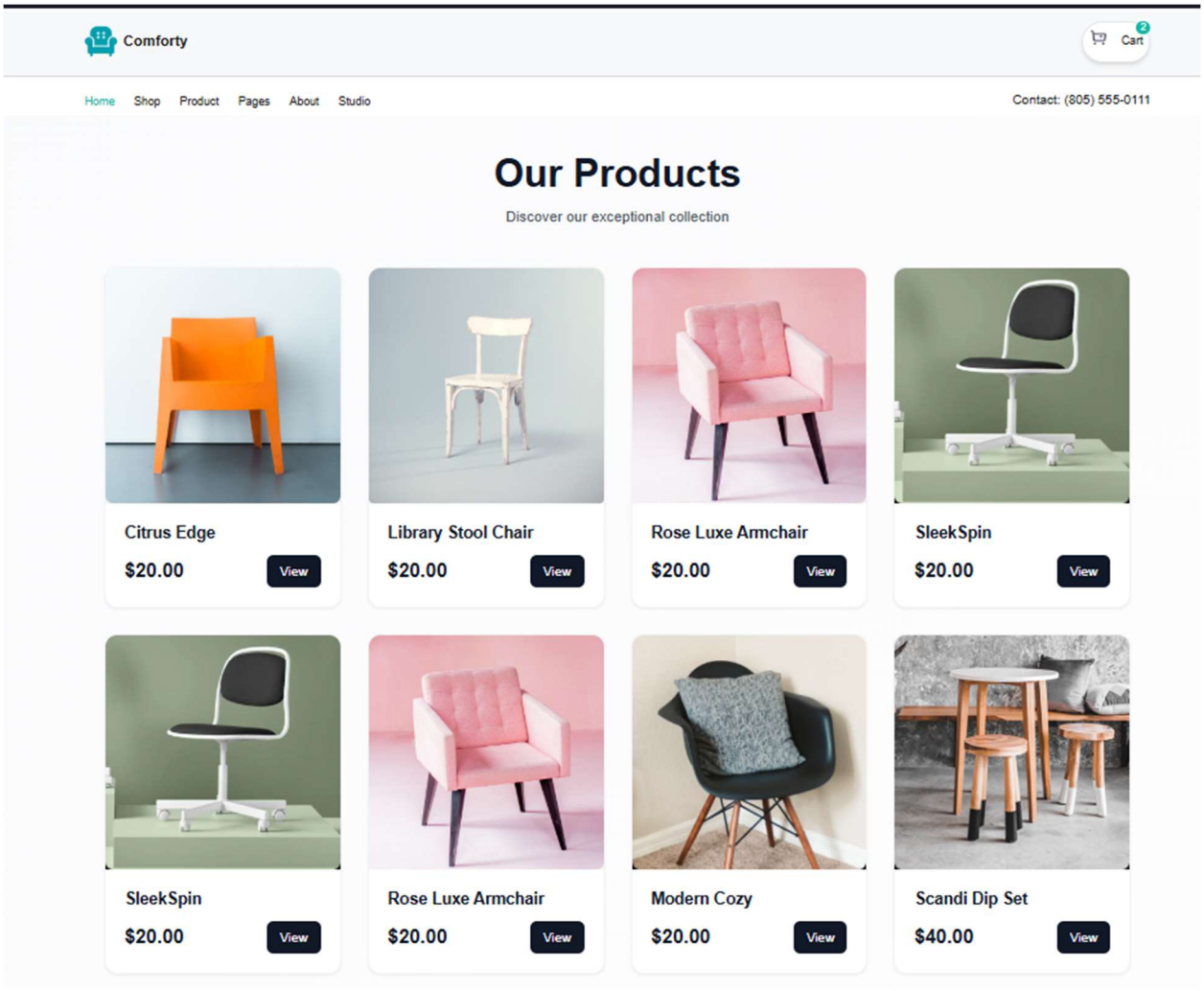
4. Migration Process and Tools Used

Tools Utilized:

1. **Sanity Client**: For interacting with the Sanity CMS (@sanity/client).
2. **Node.js & Fetch API**: For fetching data from the external API and uploading it to Sanity.
3. **Environment Variables**: To securely store sensitive data such as API tokens (dotenv).

Migration Steps:

1. **Set Up Environment**:
    - Install dependencies (@sanity/client, dotenv).
    - Configure .env.local for secure access to project-specific keys.
2. **Sanity Client Setup**:
    - Create a targetClient instance using the Sanity project ID and API token.
    - Configure it to interact with the correct Sanity dataset.
3. **Fetch Data from API**:
    - Use the fetch method to retrieve product and category data.
4. **Upload Images to Sanity**:
    - If the API provides image URLs, utilize a helper function to upload the images to Sanity and retrieve their asset IDs.
5. **Migrate Categories**:
    - Iterate through the categories and upload them to Sanity, linking associated images if available.
6. **Migrate Products**:
    - Iterate through the product data, upload their images, and reference their categories in Sanity.
7. **Verify Data**:
    - After migration, manually verify the data within Sanity to ensure accuracy.



6. Conclusion

This integration automates the process of migrating external product and category data into Sanity CMS, enhancing backend management and improving frontend synchronization. The newly structured schema ensures better scalability and usability for both administrators and users.

Key Achievements:

- API integration with Sanity CMS.
- Improved schema for detailed product management.
- Automation of image uploads and category referencing.

This project showcases the seamless integration of modern tools and efficient backend systems to create a robust eCommerce platform.