Day 3 - API Integration Report - [General E-Commerce]

API Integration Report – Chairs

API Integration Process

Step 1: API Endpoint Setup

- **API Endpoint**: https://docs.google.com/document/d/1tg3wdRvcGnEcyVRyzVpXzI8tbPClD_Z9mfFrq1vhkns/edit?tab=t.0
- **Data Structure**:

The API provides product data, including fields like `productName`, `category`, `price`, `inventory`, `colors`, `status`, `description`, and `image`.

- **Library Used**: Axios was used to fetch data from the API.

Step 2: Fetch Data from API

- **Code Snippet**:

```
import "dotenv/config";
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 = "https://giaic-hackathon-template-08.vercel.app", // API base URL for products and categories
} = process.env;
```

Changings Made in Schemas

Product Schema Modifications

1. **Field Updates**:
    a. **`colors`**: Added as an optional array of strings to accommodate multiple color options.
    b. **`image`**: Configured as a Sanity image field with hotspot enabled for better image cropping and scaling.
    c. TAGs: Added colorful TAGs and Strike Through on old prices.

```
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" },
        ],
      },
    },
  ],
});

export default productSchema;
```

Products Schema

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

export const categorySchema = defineType({
    name: 'categories',
    title: 'Categories',
    type: 'document',
    fields: [
        {
            name: 'title',
            title: 'Category Title',
            type: 'string',
        },
        {
            name: 'image',
            title: 'Category Image',
            type: 'image',
        },
        {
            title: 'Number of Products',
            name: 'products',
            type: 'number',
        }
    ],
});

export default categorySchema;
```

Categories Schema

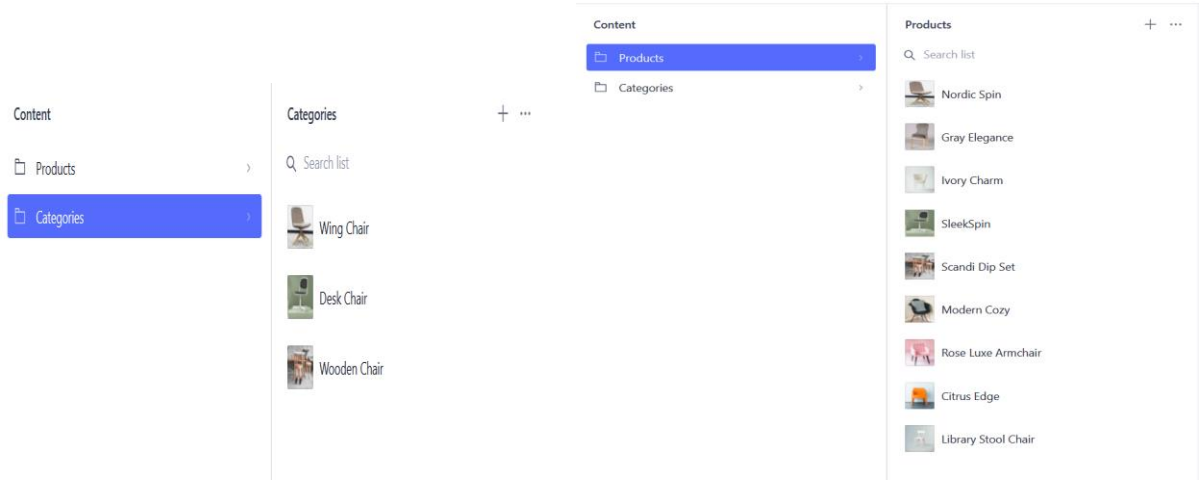
Migration Steps and Tools Used

Migration Steps

1. **Environment Setup:**

a. Installed required dependencies: `@sanity/client`, `axios`, `dotenv`.
b. Created a `.env.local` file to securely store environment variables.

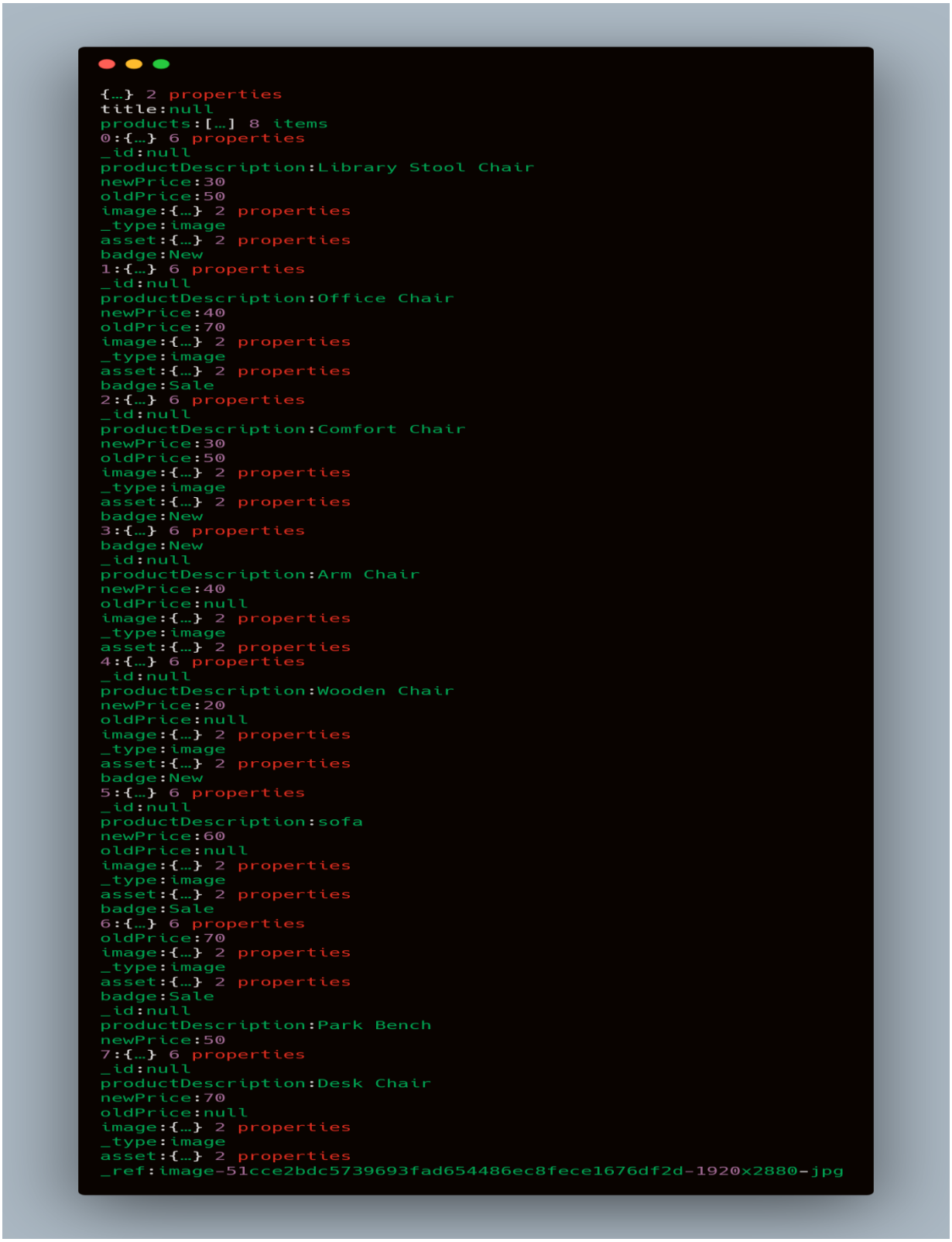
2. **Data Fetching**:
   a. Retrieved product data from the API endpoint using Axios.
   b. Parsed and logged the data to confirm its structure and integrity.
3. **Image Upload**:
   a. Downloaded images from the API's `image` field using Axios.
   b. Uploaded images to Sanity's Asset Manager using the Sanity client.
   c. **Code to upload images**:




```
// 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
    });
```

Document Creation:

- Created Sanity documents for each product by combining API data and uploaded image references.

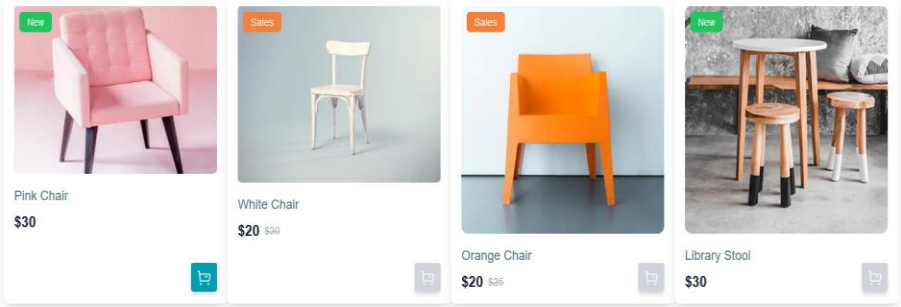


Products and Categories Output

```
import "dotenv/config";
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 = "https://glaic-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
}

// 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
  }
}

// Function to check and delete duplicate products
async function deleteDuplicateProductsByTitle(title) {
  try {
    // Query existing products with the same title
    const existingProducts = await targetClient.fetch(
      `*[_type == "products" && title == $title]`,
      { title }
    );

    // If duplicate products exist, delete them (excluding the first one)
    if (existingProducts.length > 1) {
      const productIdsToDelete = existingProducts.slice(1).map((product) => product._id);
      await targetClient.delete(productIdsToDelete);
      console.log(`Deleted ${productIdsToDelete.length} duplicate products with title: ${title}`);
    }
  } catch (error) {
    console.error("Error deleting duplicates:", error.message);
  }
}

// Main function to migrate data from REST API to Sanity
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}`);

      // Before inserting a product, check and delete existing duplicates
      await deleteDuplicateProductsByTitle(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();
```

Migration code

Featured Prod



Top Category

```tsx
"use client";

import Image from "next/image";
import { useEffect, useState } from "react";
import { ShoppingCart } from "lucide-react";
import { client } from "@/sanity/lib/client";
import { Badge } from "./ui/badge";
import { Button } from "./ui/button";
import { urlFor } from "@/sanity/lib/image";

interface Product {
  _id: string;
  productDescription: string;
  newPrice: number;
  oldPrice?: number;
  image: any;
  badge?: string;
}

export default function ProductGrid() {
  const [products, setProducts] = useState<Product[]>([]);
  const [sectionTitle, setSectionTitle] = useState<string>("");

  useEffect(() => {
    const fetchProducts = async () => {
      const data = await client.fetch(
        `*[_type == "HomeSection4"][0]{
          title,
          products[]{
            _id,
            productDescription,
            newPrice,
            oldPrice,
            image,
            badge
          }
        }`
      );
      setSectionTitle(data.title || "Our Products");
      setProducts(data.products || []);
    };

    fetchProducts();
  }, []);

  const handleAddToCart = (productId: string) => {
    console.log(`Added product ${productId} to cart`);
  };
```

API Integration	Schema Validation	Data Migration	API integration in Nextjs	Submission Preparation
Done	Done	Done	Done	Done