REPORT
HACKATHON Day 3 -MARKETPLACE BUILDER
COMFORTY CHAIR

Overview:

On Day 3 task,I focused on integerating APIs and migrating data in to Sanity (CMS) to build a backend of functional marketplace,

➔ Connecting APIs to my next.js project.
➔ Moved data APIs into Sanity.
➔ Use provided APIs of Template 8

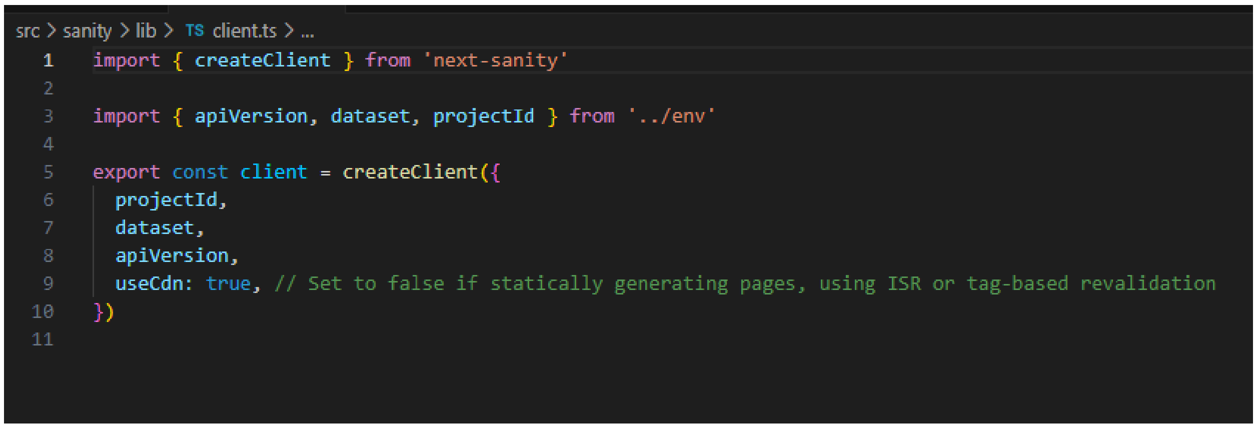
1: API INTERGRATION:

The integration of APIs was essential for fetching product data from an external source (in this case, the provided APIs) and populating it into our Sanity CMS for the marketplace. Below is the step-by-step process I followed for API integration in the Next.js project:

- **API Choosing:** I chose the API from Template 8 provided in the documentation.I used the provided API https://template-03- api.vercel.app/api/products was used to fetch product data. This endpoint provided essential details, including product titles, descriptions, prices, and category IDs.

- **API Documentation Review:** I reviewed the API documentation thoroughly to understand the endpoints.The documentation helped identify the necessary fields (e.g., product_title, price, category_id).

- **Setting Sanity API Call:** Here is the snip of Sanity/lib/client.ts



src > sanity > lib > TS client.ts > ...

```ts
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
})

```

- **Schema Revisions:** I updated the existing schema to ensure compatibility with the product data fetched from the API.

2. Adjustments Made to Schemas:

In order to store the product data in Sanity CMS, I had to adjust the existing schema to ensure compatibility with the data fetched from the API.

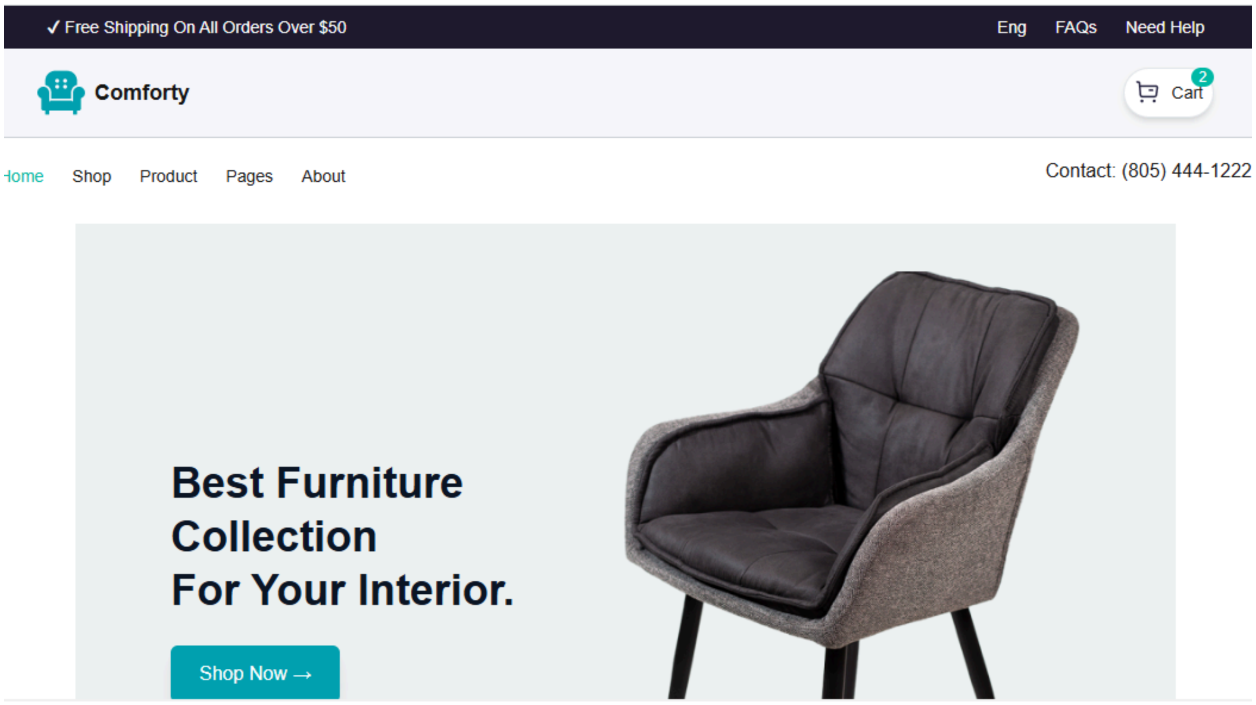
3.MIGRATION STEPS AND TOOLS :

Data Migration:
Migration Process: Using the provide migrating script to transferred the data of APIs into Sanity(CMS).The script fetched product data from the API, transformed it to match the Sanity schema, and then imported the data into the Sanity CMS.

Tools Used: Sanity/Client.

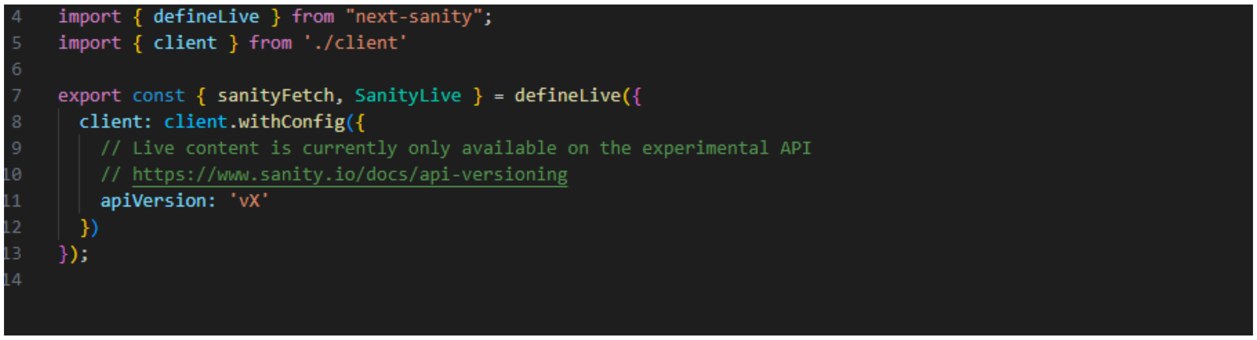
FRONTEND:

After migrating the into Sanity ,I created a dynamic responsive frontend in next.js project to display the data.

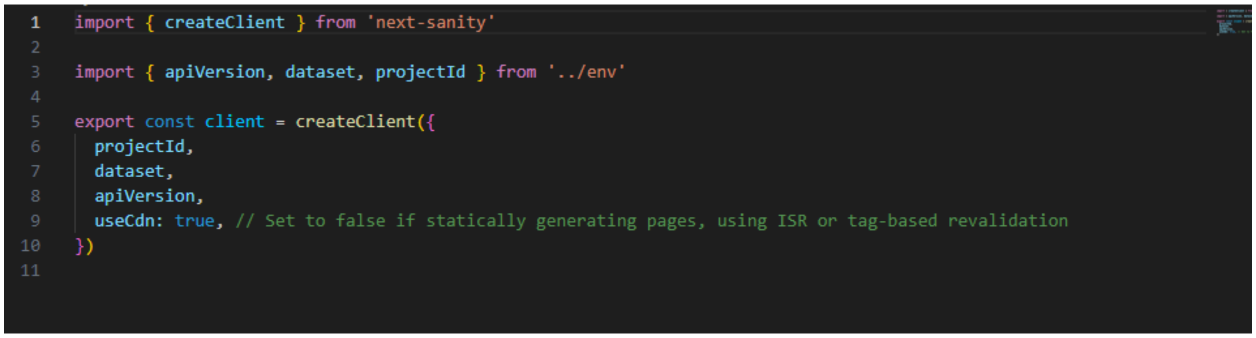


FETCHING PRODUCTS DATA:

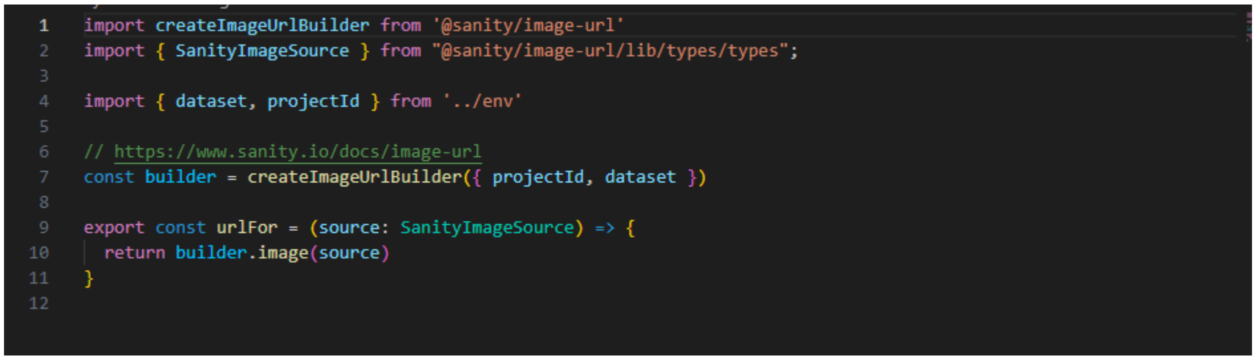
I used sanity GROQ queries to fetch data directly from sanity(CMS) to my next.js project.



```
import { defineLive } from "next-sanity";
import { client } from './client'

export const { sanityFetch, SanityLive } = defineLive({
  client: client.withConfig({
    // Live content is currently only available on the experimental API
    // https://www.sanity.io/docs/api-versioning
    apiVersion: 'vX'
  })
});
```



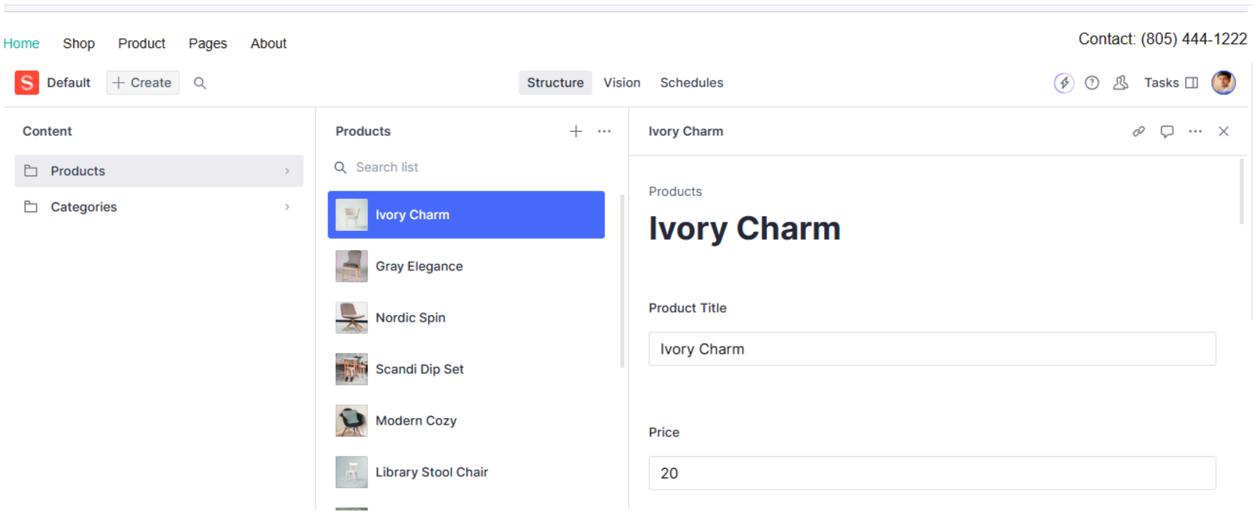
```
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
})
```



```
import createImageUrlBuilder from '@sanity/image-url'
import { SanityImageSource } from "@sanity/image-url/lib/types/types";

import { dataset, projectId } from '../env'

// https://www.sanity.io/docs/image-url
const builder = createImageUrlBuilder({ projectId, dataset })

export const urlFor = (source: SanityImageSource) => {
  return builder.image(source)
}
```

SANITY STUDIO INTERFACE:

After migrating the data script,the data appears in Sanity studio.


Home Shop Product Pages About

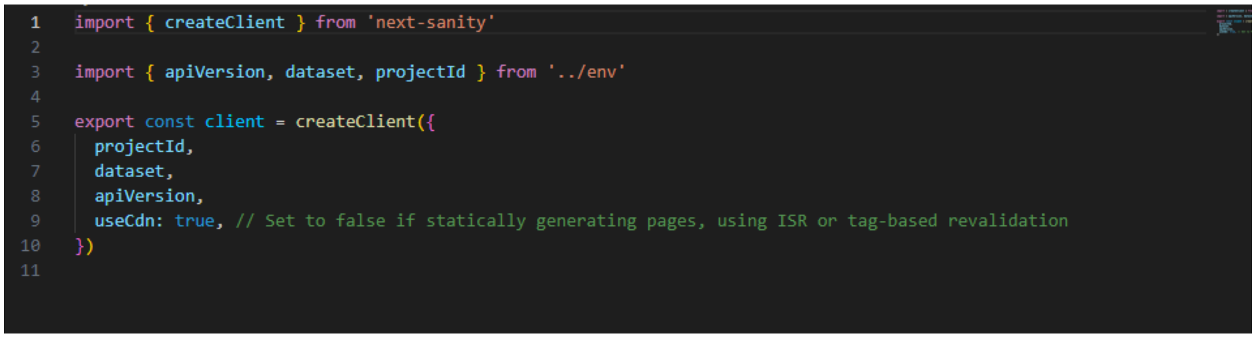
Contact: (805) 444-1222

Ivory Charm

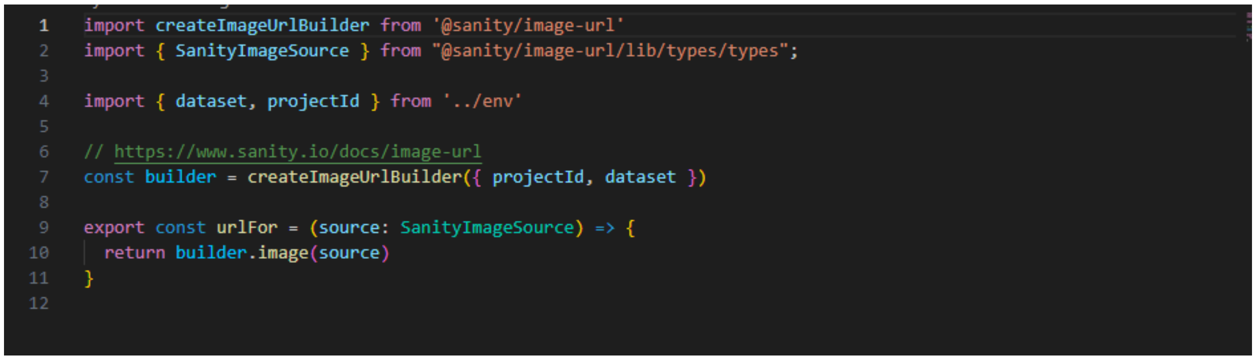
Product Title: Ivory Charm

Price: 20

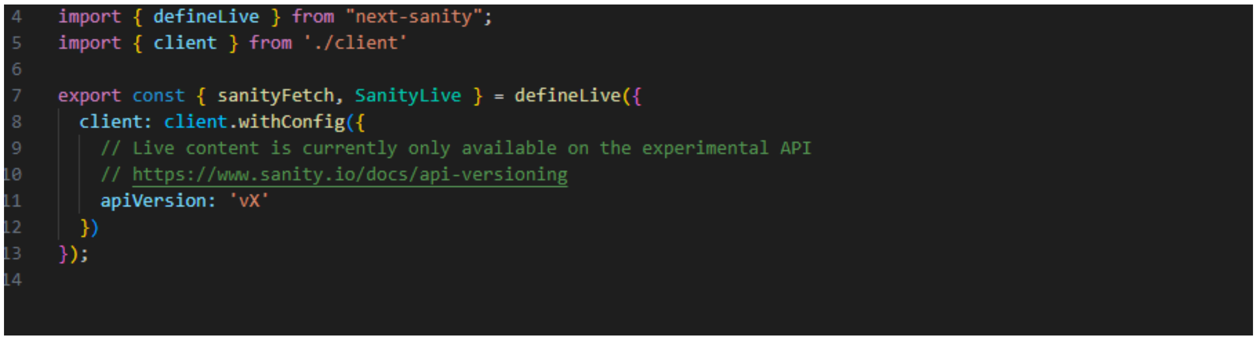
API CALLS:



```
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
})

```

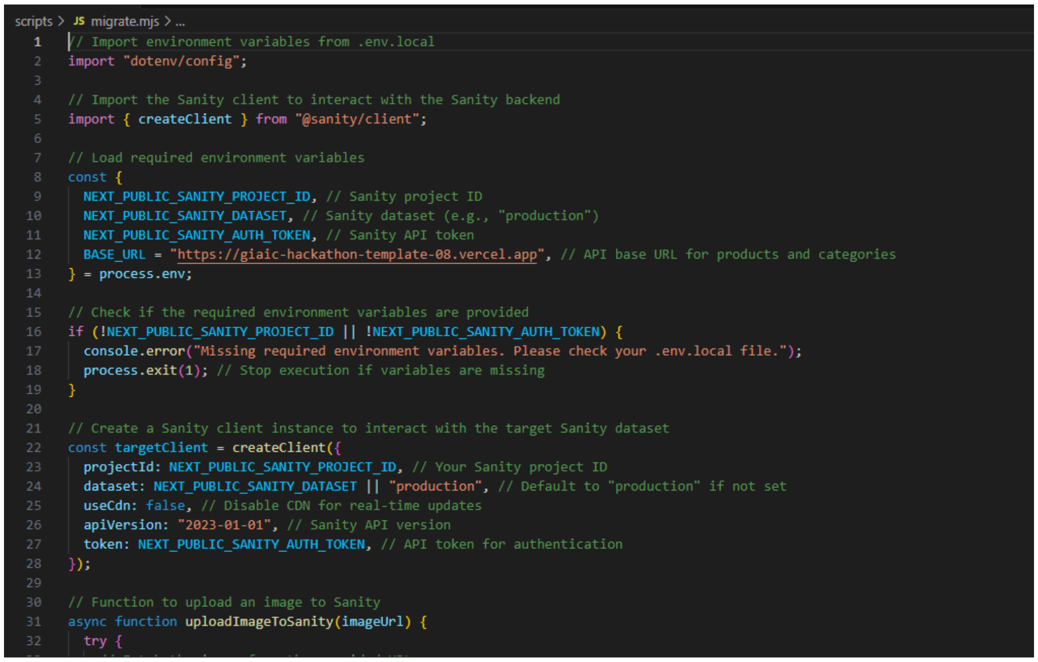


```
import createImageUrlBuilder from '@sanity/image-url'
import { SanityImageSource } from "@sanity/image-url/lib/types/types";

import { dataset, projectId } from '../env'

// https://www.sanity.io/docs/image-url
const builder = createImageUrlBuilder({ projectId, dataset })

export const urlFor = (source: SanityImageSource) => {
  return builder.image(source)
}

```

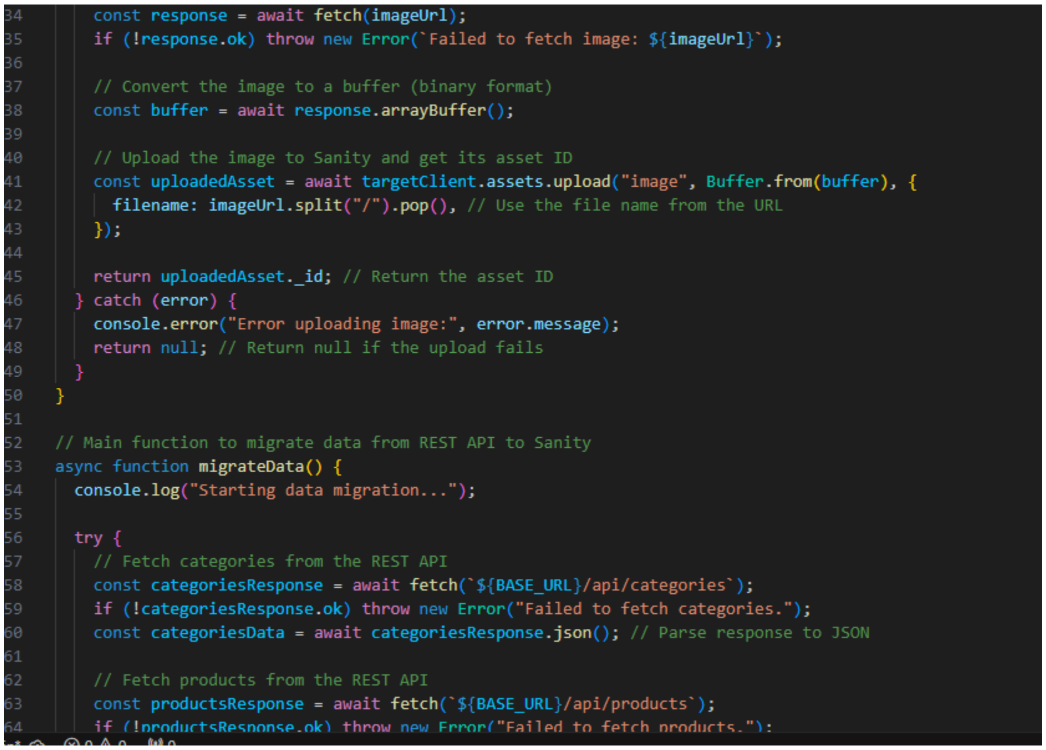


```
import { defineLive } from "next-sanity";
import { client } from './client'

export const { sanityFetch, SanityLive } = defineLive({
  client: client.withConfig({
    // Live content is currently only available on the experimental API
    // https://www.sanity.io/docs/api-versioning
    apiVersion: 'vX'
  })
});

```

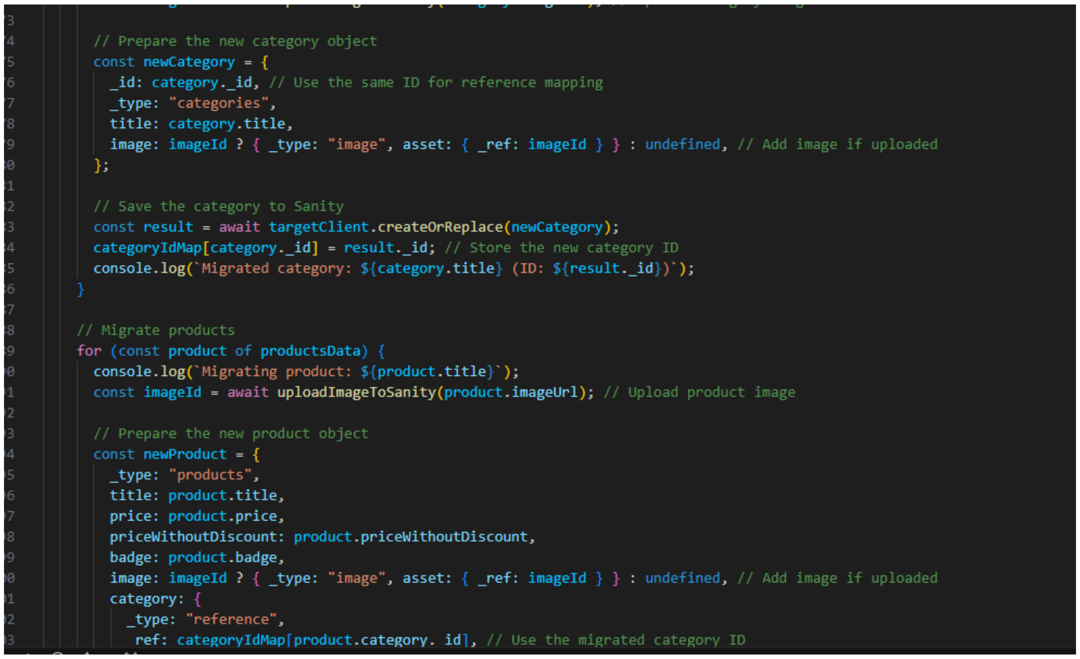
MIGRATING SCRIPT:



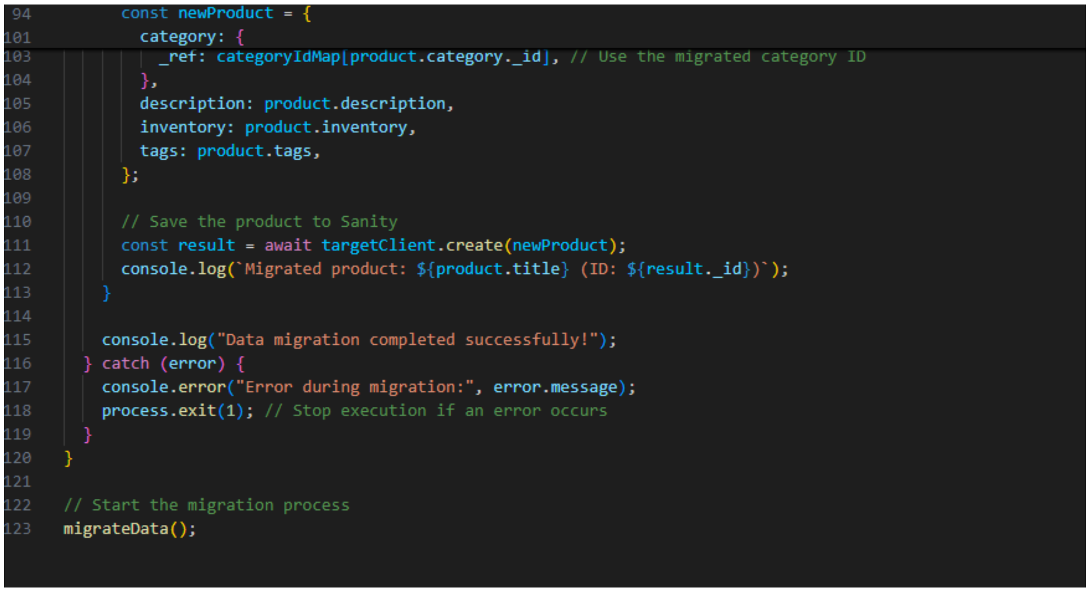
```javascript
// Import environment variables from .env.local
import "dotenv/config";

// Import the Sanity client to interact with the Sanity backend
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;

// 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 {
```



```javascript
    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 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.");
```



```javascript

      // 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
```



```javascript
      const newProduct = {
        category: {
          _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();
```

Conclusion:

In this report, I have documented the process of integrating an external API into a Next.js project, adjusting the Sanity CMS schema, and migrating product data into the CMS. The project successfully fetched product data from the API, displayed it on the frontend, and populated the Sanity CMS with the data