Marketplace Builder Hackathon Day 3

Api Integration And Data Migration In Sanity

Fetching Data From External Api:

To integrate an external API with Sanity and migrate data, I followed this process which involves several steps. First, set up your Sanity schema by defining the structure of the data you want to store, such as products. For this, you create a schema file (e.g., `products.ts`) specifying the fields like title, price, description, badge, inventory, tags, and image.

Next, create a migration script (e.g., `migrate.mjs`) that fetches data from the external API and formats it according to your schema before inserting it into Sanity. Use the Sanity client to connect to your dataset and perform mutations (data creation). Ensure the script handles the mapping of API fields to your Sanity schema fields.

After writing the script, update your `package.json` file to include a custom script command to execute the migration script (e.g., `"migrate": "node scripts/migrate.mjs"`). Install the necessary packages like the Sanity client and fetch library to enable API communication and data handling.

Finally, run the migration script using the custom command in your terminal (e.g., `npm run migrate`). This will fetch data from the external API and populate your Sanity dataset with the mapped data. Make sure your API responses align with your schema fields to avoid errors during the migration process.

Below is the code of migration.mjs

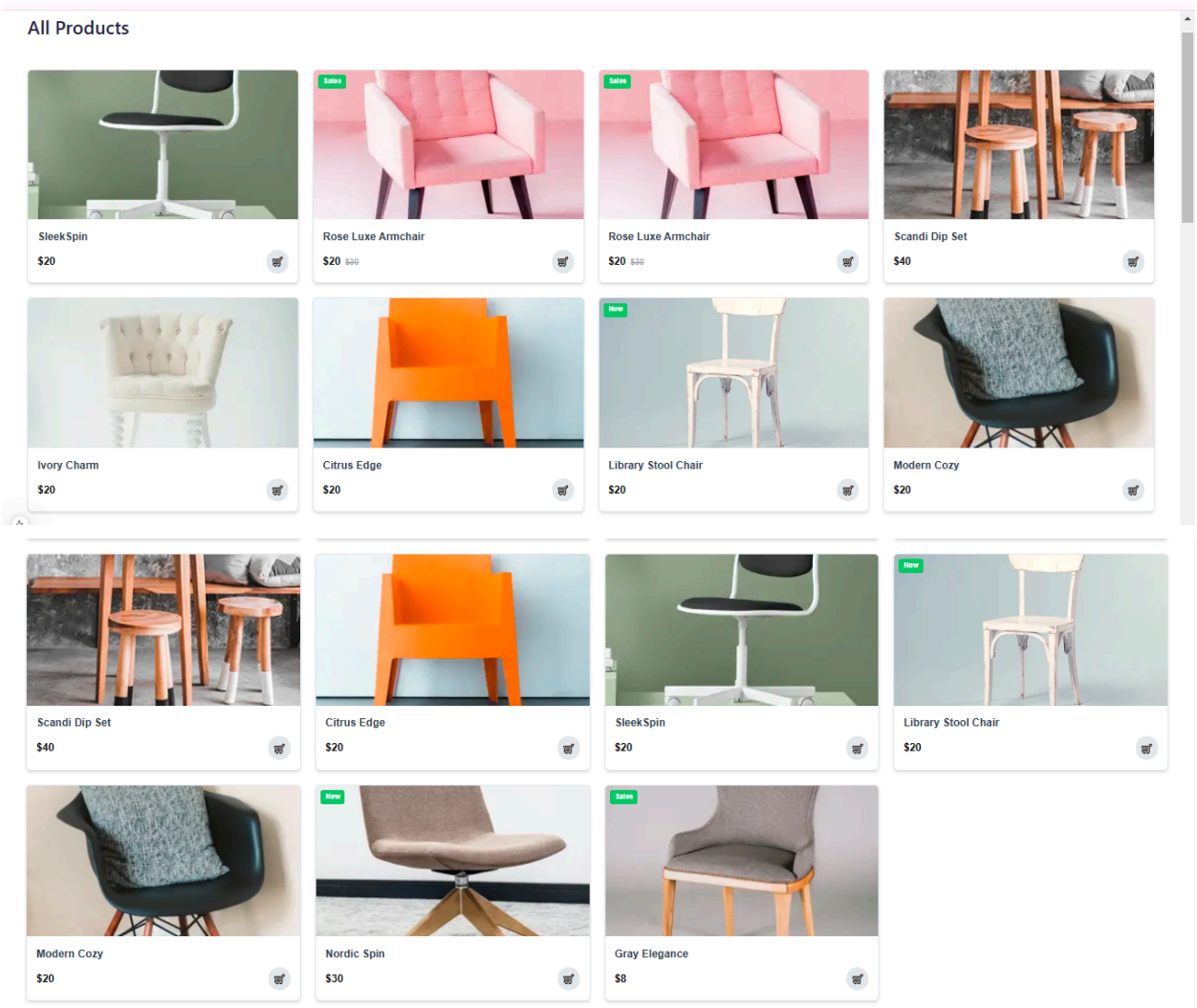
scripts > JS migrate.mjs > ...

```
// 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 {
```

```
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 migrate data from REST API to Sanity
async function migrateData() {
  console.log("Starting data migration...");

  try {

// 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
```

```
    // 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
        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();
```

Data Succesfully Displayed



Api Endpoint:

I use this api endpoint provided by sir hamza and make a little bit change in it according to my schema

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

Schema Structure:

```
src > sanity > schemaTypes > TS product.ts > ...
1  import { defineType } from "sanity";
2
3  export const productSchema = defineType({
4    name: "products",
5    title: "Products",
6    type: "document",
7    fields: [
8      {
9        name: "title",
10       title: "Product Title",
11       type: "string",
12     },
13     {
14       name: "price",
15       title: "Price",
16       type: "number",
17     },
18     {
19       title: "Price without Discount",
20       name: "priceWithoutDiscount",
21       type: "number",
22     },
23     {
24       name: "badge",
25       title: "Badge",
26       type: "string",
27     },
28     {
29       name: "image",
30       title: "Product Image",
31       type: "image",
32     },
```

```
    {
      name: "badge",
      title: "Badge",
      type: "string",
    },
    {
      name: "image",
      title: "Product Image",
      type: "image",
    },
    {
      name: "inventory",
      title: "Inventory Management",
      type: "number",
    },
      ],
    },
  );
```