Day3 API INTEGRATION

API Integration Process

• API EndPoint Setup

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

• Provides product data with fields such as productName, category, price, inventory, colors, status, description, and image.

2- Categories Endpoint: https://giaic-hackathon-template-08.vercel.app/api/categories

• Provides category data with fields such as title, images, products

Library Used: Fetch API was used to fetch data from the API.

• Fetch Data From API

The API call retrieves an array of product objects from the endpoint, ensuring proper data handling and structure.

Code Snippet:

```
const ProductPage = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [cart, setCart] = useState<Product[]>([]);

  useEffect(() => {
    client.fetch(query).then((data) => {
      console.log(data);



      const formattedProducts = data.map((prod: {
        _id: string;
        title: string;
        price: number;
        priceWithoutDiscount: number;
        productImage: string;
        badge: string;
      }) => {
        const hasDiscount = prod.priceWithoutDiscount < prod.price;
        return {
          id: prod._id,
          img: prod.productImage,
          name: prod.title,
          RealPrice: `${prod.price.toFixed(2)}`,
          discPrice: hasDiscount
            ? `${(prod.price - (prod.price - prod.priceWithoutDiscount)).toFixed(2)}`
            : undefined,
          off: hasDiscount ? "Sale" : undefined,
          offColor: hasDiscount ? "#F5813F" : undefined,
        };
      });
      setProducts(formattedProducts);
    });
```

Prepared by Huzaifa Naeem

1. Product Schema

```
"": {
  "prefix": "",
  "body": [
    "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\" },",
    "        ],",
    "      },",
    "    },",
    "  ],",
    "});"
  ],
  "description": ""
}
```

2. Category 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',
        }
    ],
});
```

Migration Setup and Tool Used

1. Environment Setup:

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

2. Data Fetching:

- Retrieved product data from the API endpoint using the Fetch API.
- Parsed and logged the data to confirm its structure and integrity.

3. Image Upload:

- Downloaded images from the API's image field using the Fetch API.
- Uploaded images to Sanity's Asset Manager using the Sanity client.

```
// 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
  }
}
```

Tool Used:

- Sanity Client: For interacting with the Sanity CMS.
- Fetch API: For making HTTP requests to fetch API data and images.
- Dotenv: To load environment variables from .env.local.

Prepared by Huzaifa Naeem

ScreenShot:

1. API Call Output

- Output showing the retrieved product data.

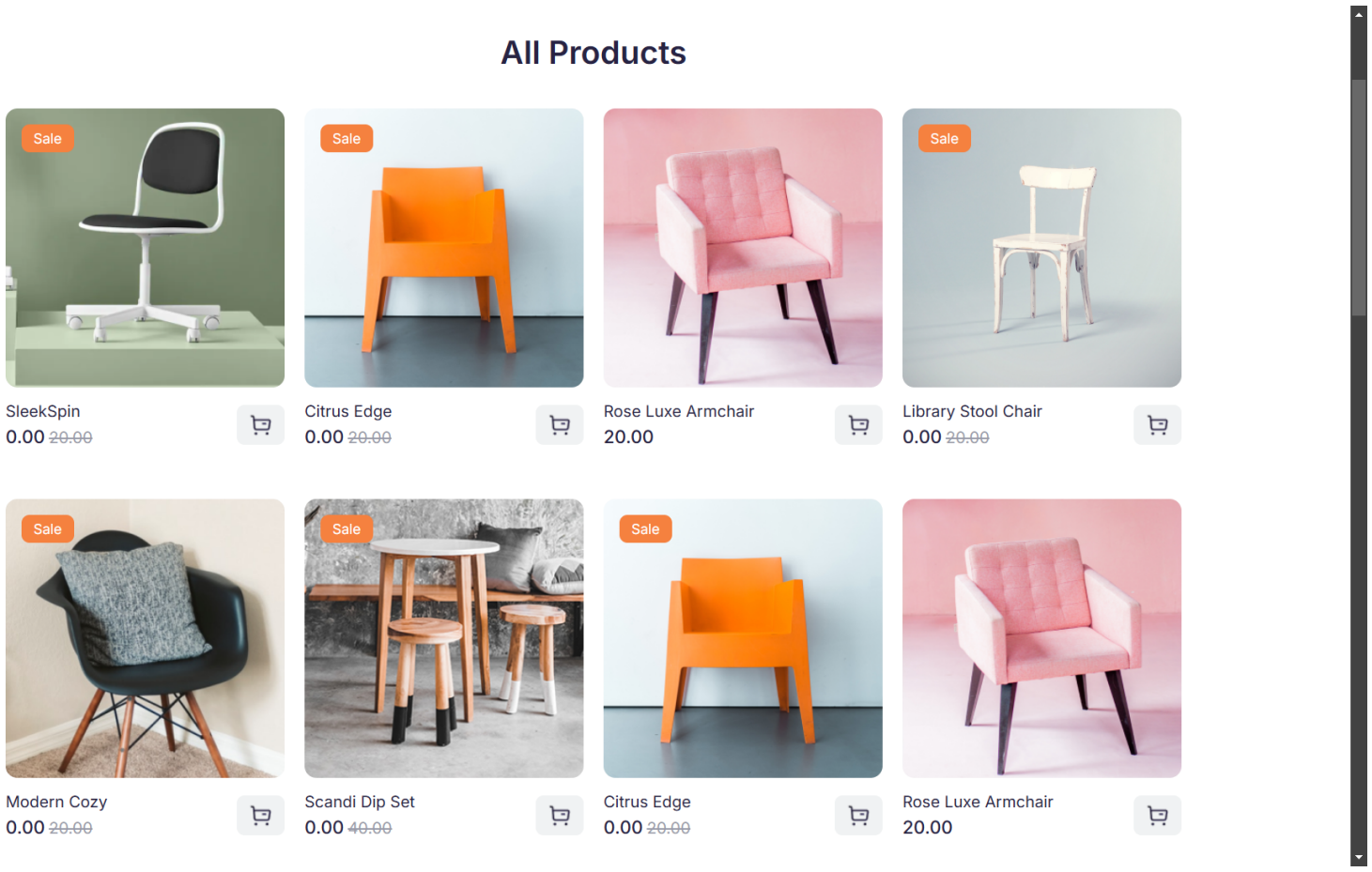
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Migrating product: SleekSpin
Migrated product: SleekSpin (ID: pNdFo0jzUh1upuRBLLliLN)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: qQpkQmFIZ2DSQkhvXCkxmn)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: pNdFo0jzUh1upuRBLLljXD)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: qQpkQmFIZ2DSQkhvXCkyLC)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: pNdFo0jzUh1upuRBLLllZn)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: qQpkQmFIZ2DSQkhvXCkz54)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: pNdFo0jzUh1upuRBLLlnhe)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: qQpkQmFIZ2DSQkhvXCkzS0)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: rnb16y4e0BKBgYGF3FNpOz)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: rnb16y4e0BKBgYGF3FNpgH)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: qQpkQmFIZ2DSQkhvXCl2AC)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: qQpkQmFIZ2DSQkhvXCl2u4)
Migrating product: Nordic Spin
Migrated product: Nordic Spin (ID: qQpkQmFIZ2DSQkhvXCl3H0)
Migrating product: Gray Elegance
Migrated product: Gray Elegance (ID: rnb16y4e0BKBgYGF3FNr2Z)
Migrating product: Ivory Charm
Migrated product: Ivory Charm (ID: qQpkQmFIZ2DSQkhvXCl4VS)
Data migration completed successfully!

D:\hackathon-3\Hackathon2-ecommerce-figma-to-nextjs>
```

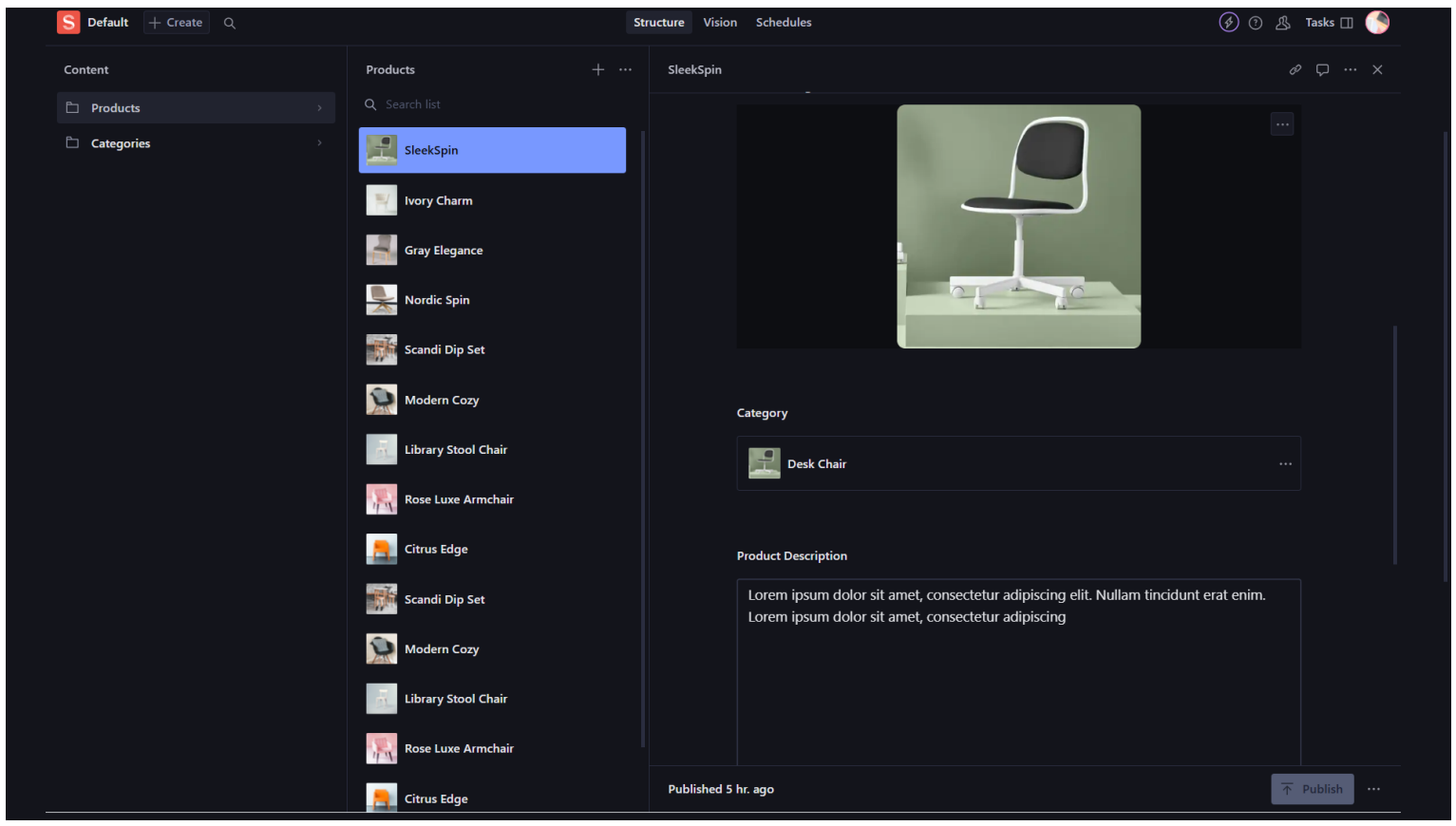
2. Frontend Display

- The data displayed on the front-end.



3. Populated Sanity CMS Fields

- Sanity's CMS populated with the product and image data.



MigrationScript:

```
import "dotenv/config";

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

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",
} = process.env;

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
}

const targetClient = createClient({
  projectId: 'ie4h9yk3', // Your Sanity project ID
  dataset: 'production', // Default to "production" if not set
  useCdn: false, // Disable CDN for real-time updates
  apiVersion: "2023-01-01", // Sanity API version
  token:
'sk9YHNBufBhKqOBCo9QX2tt7lAE9LDCdBsbdc1EdHXvHnQ1Z1YCsWwy666QnRugAaglGtJMowTzkdKL7laUUgCjZGNqE9uhSGGrY8d
vTE6j7WuGXVxesB4ogke09JOkg3LWYfseqEvYVARXEGaHxDx2NoyM5VTJ9CuqASkIhpKpxSOjCM9ld', // 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 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

      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
      };

      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})`);
    }

    for (const product of productsData) {
      console.log(`Migrating product: ${product.title}`);
      const imageId = await uploadImageToSanity(product.imageUrl);
      const newProduct = {
        _type: "products",
        title: product.title,
        price: product.price,
        priceWithoutDiscount: product.priceWithoutDiscount,
        badge: product.badge,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined,
        category: {
          _type: "reference",
          _ref: categoryIdMap[product.category._id], // Use the migrated category ID
        },
        description: product.description,
        inventory: product.inventory,
        tags: product.tags,
      };
      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);
  }
}

migrateData();
```

CheckList:

Self-Validation Check-list:

API Understanding:

- ✔

Schema Validation:

- ✔

Data Migration:

- ✔

API Integration in Next.js:

- ✔

Submission Preparation:

- ✔