Submitted by:shan-e-zehra

Roll No:00386168

Days:sunday-09:00Am-12:00pm

Hackathone Template 8 Documentation :

Welcome to the Comforty hackathon! This document serves as a guide for integrating Sanity into the project to manage product data efficiently, extending the existing frontend functionality.

Overview

Comforty is a marketplace for chairs, designed to provide users with a seamless browsing and purchasing experience. In this hackathon, we aim to:

Set up a Sanity schema for product management.

Develop a migration script for transferring data between Sanity accounts.

Ensure seamless integration with the existing Next.js frontend.

Install Sanity Studio

You start by setting up your content editing environment. It’s called Sanity Studio, and you can configure and customize it with JavaScript. It runs in the browser. To develop locally, we need to run a development server so you can see your changes instantly.

To get started, run this in your command line:

npm create sanity@latest -- --template clean --create-project "learning-sanity-project" --dataset production

Sanity Schema:

Create two new files and add the following code:

products.ts

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

categories.ts

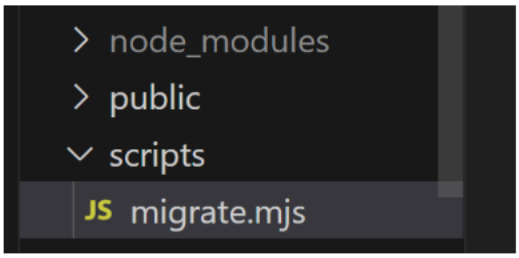
```
import { defineType } from "sanity";

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',
            }
        ],
});
```

Importing Schemas in index.ts

```
import { type SchemaTypeDefinition } from "sanity";
import { productSchema } from "./products";
import { categorySchema } from "./categories";

export const schema: { types: SchemaTypeDefinition[] } = {
    types: [productSchema, categorySchema],
};
```

Data Migration Script




```
// 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 {
        // 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


        // 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
            },
            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();
```

Setting Up Environment Variables

Create a .env file in the root of your project

```
NEXT_PUBLIC_SANITY_PROJECT_ID="<Project ID>" # Add your project Id
NEXT_PUBLIC_SANITY_DATASET="production"
NEXT_PUBLIC_SANITY_AUTH_TOKEN="<Auth Token>" # Add your token
```

Open `package.json` file and add the following code inside of scripts:

```
"scripts": {
  "dev": "next dev --turbopack",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "migrate": "node scripts/migrate.mjs"
},
```

Install the following package before running the script

npm install dotenv

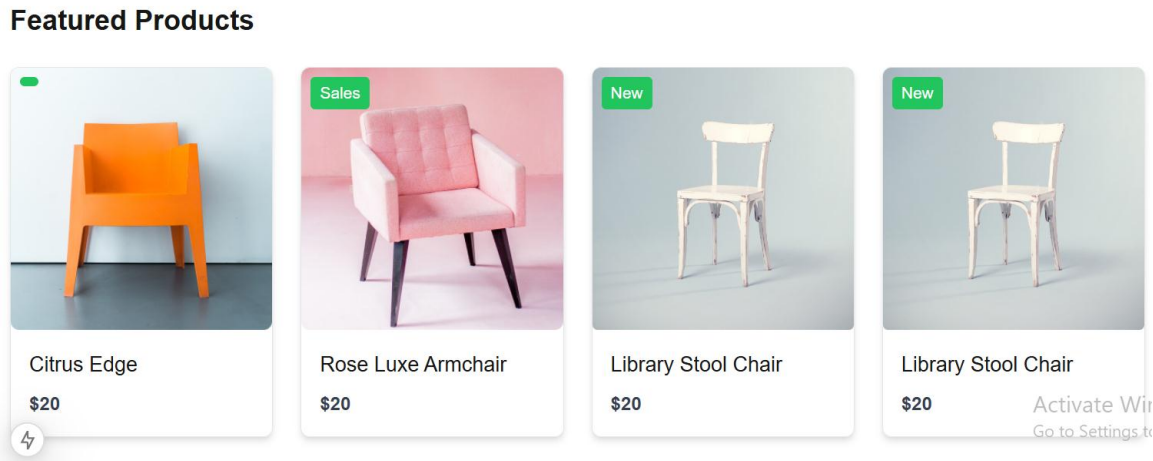
Now run the command npm run migrate

Featuredproduct:

```
import { sanityFetchProducts } from "@/sanity/lib/productFetch";
import { products } from "@/sanity/lib/queries";
import Image from "next/image";

type Products = {
    _id: string;
    title: string;
    description: string;
    price: number;
    category: string;
    badge: string;
    imageUrl: string;
};
export default async function FeaturedProducts() {
    const allProduct: Products[] = await sanityFetchProducts({
        query: products,
    });
    return (
```

```
<div className="py-10 px-4 bg-white">
<h2 className="text-2xl font-semibold mb-6">Featured Products</h2>
<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
{/* Product 1 */}
{
allProduct.map((product)=>(
<div key={product._id} className="border rounded-lg overflow-hidden shadow-md">
<div className="relative">
<Image src={product.imageUrl}alt={product.title} width={300} height={300} className=""/>
<span className="absolute top-2 left-2 bg-green-500 text-white px-2 py-1 text-sm rounded">
{
product.badge
}
</span>
</div>
<div className="p-4">
<h3 className="text-lg font-medium mb-2">{product.title}</h3>
<p className="text-gray-700
```

```
font-semibold">${product.price}</p>
                </div>
            </div>
                ))
            }


        </div>
    </div>
  );
}
```



Our Products:

```
"use client";

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

import Image from "next/image";
```

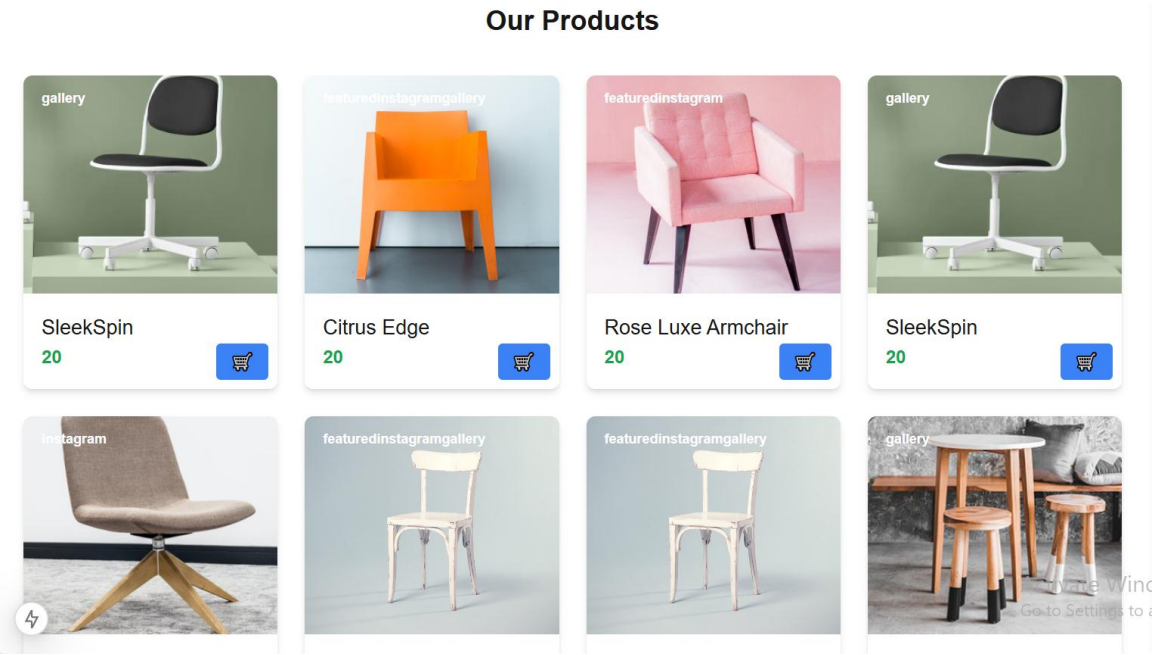
```
import { useEffect, useState } from "react";
import { urlFor } from "@/sanity/lib/image";

interface Products {
    _id: string;
    title: string;
    price: string;
    image: any;
    tags?: string;
}

export default function ProductGrid() {
    const [allproducts, setallproducts] = useState<Products[]>([]);

    useEffect(() => {
        const fetchProduct = async () => {
            try {
                const QueryProduct = `*[_type == "products"]`;
                const productsResponse = await client.fetch(QueryProduct);
                setallproducts(productsResponse);
            } catch (error) {
                console.error("Error fetching product data:", error);
```

```
        }
    };

    fetchProduct();

    return () => {
        setallproducts([]); // Reset state when component unmounts
    };
}, []);

console.log(allproducts);

return (
    <section className="px-6 py-12">
        <h2 className="text-2xl font-bold text-center mb-8">Our Products</h2>
        {allproducts.length > 0 ? (
            <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-6">
                {allproducts.map((product) => (
                    <div
                        key={product._id}
```

```
                    className="relative rounded-lg shadow-md
overflow-hidden"
                >
                    {/* Product Image */}
                    <Image
                        src={urlFor(product.image).url()}
                        alt={product.title}
                        width={300}
                        height={300}
                        className="w-full h-48 object-cover"
                    />

                    {/* Tag */}
                    {product.tags && (
                        <span
                            className={`absolute top-2 left-2 text-xs
font-bold text-white px-2 py-1 rounded`}
                            >
                            {product.tags}
                        </span>
                    )}
```

```
                    {/* Product Info */}
                    <div className="p-4 bg-white">
                        <h3 className="text-lg
font-medium">{product.title}</h3>
                        <div className="flex items-center space-x-2">
                            <span className="text-green-600 font-bold">
                                {product.price}
                            </span>
                        </div>
                    </div>

                    {/* Add to Cart */}
                    <button className="absolute bottom-2 right-2
bg-blue-500 text-white px-3 py-1 rounded hover:bg-blue-600">

                    </button>
                </div>
            ))}
        </div>
    ) : (
        <h1>Products not found</h1>
    )}
```

```
        </section>
    );
}
```

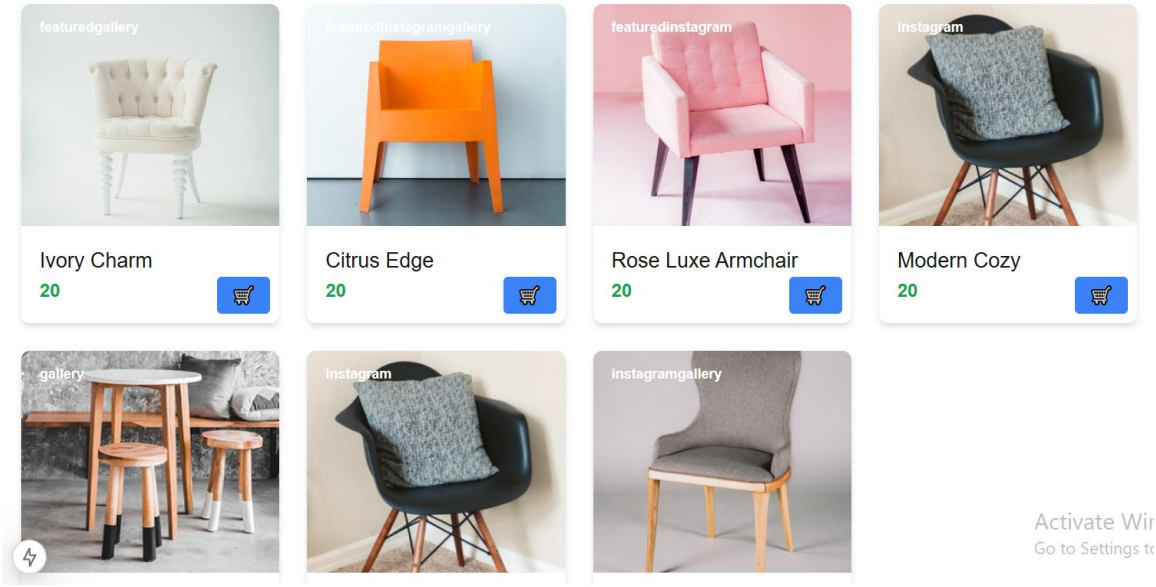


TopCategories:

```
import { sanityFetchProducts } from "@/sanity/lib/productFetch";
import { category } from "@/sanity/lib/queries";
import Image from "next/image";


type Category = {
    _id: string;
    title: string;
    products: string;
```

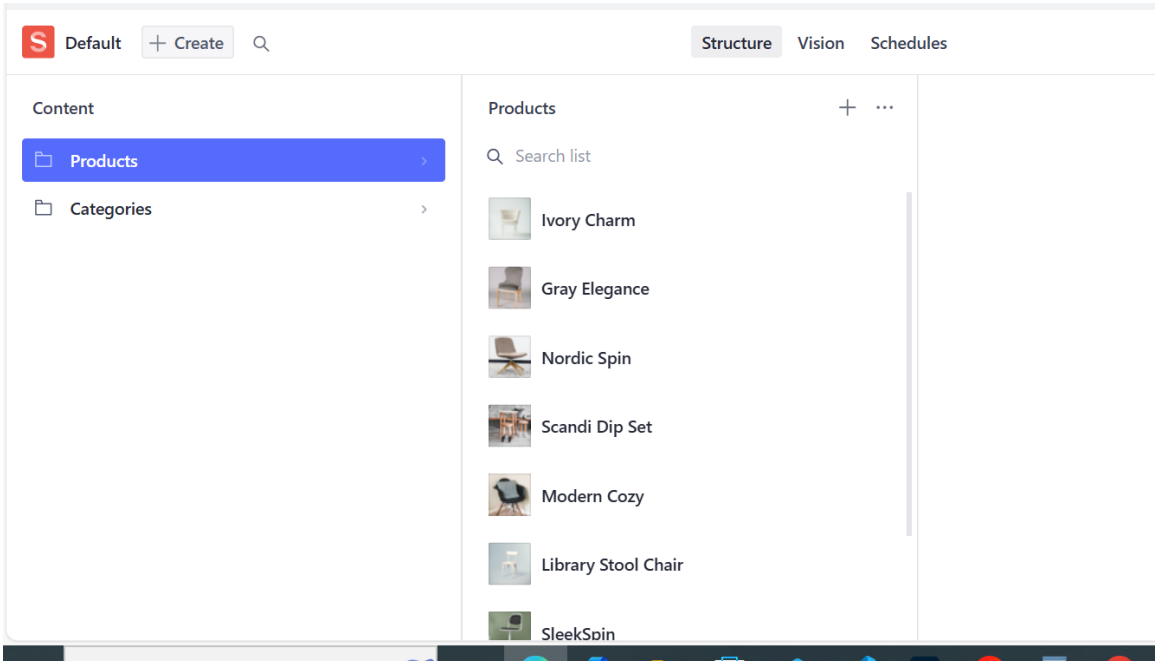
```
    imageUrl: string;
};

export default async function TopCategories() {
    const allCategory: Category[] = await sanityFetchProducts({
        query: category,
    });
    return (
        <section className="px-6 py-12">
            <h2 className="text-2xl font-bold mb-6">Top Categories</h2>
            <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
                {allCategory.map((category) => (
                    <div
                        key={category._id}
                        className="relative overflow-hidden rounded-lg
shadow-md"
                    >
                        <Image
                            src={category.imageUrl}
                            alt={category.title}
                            width={400}
                            height={300}
```

```
                    className="object-cover w-full h-64"
                  />
                <div className="h-[80px] w-full bg-black opacity-70
absolute bottom-0 rounded-b-md shadow-md">
                    <div className="text-white my-[18px] ml-4
hover:cursor-pointer ">
                      <p className="font-bold">{category.title}</p>
                      <p className="font-extralight text-[14px] ">
                        {category.products} <span
className="ml-1">Products</span>{" "}
                      </p>
                    </div>
                  </div>
              </div>
            ))}
          </div>
        </section>
    );
}
```

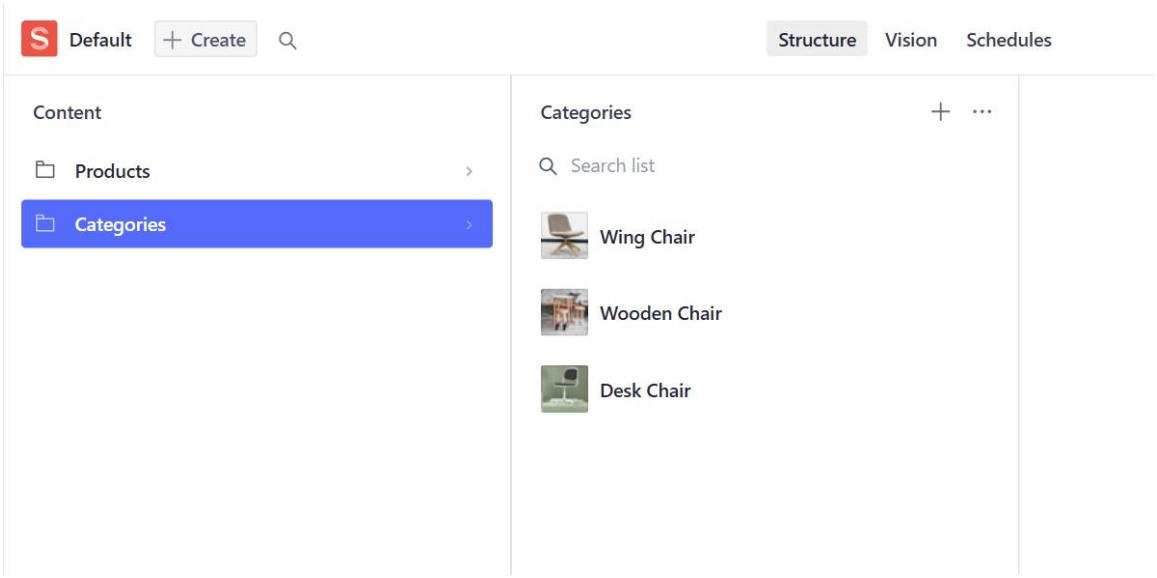


Sanity Studio:

Products:



Categories:


Default
Create
Structure
Vision
Schedules
Content
Products
Categories
Categories
Search list
Wing Chair
Wooden Chair
Desk Chair