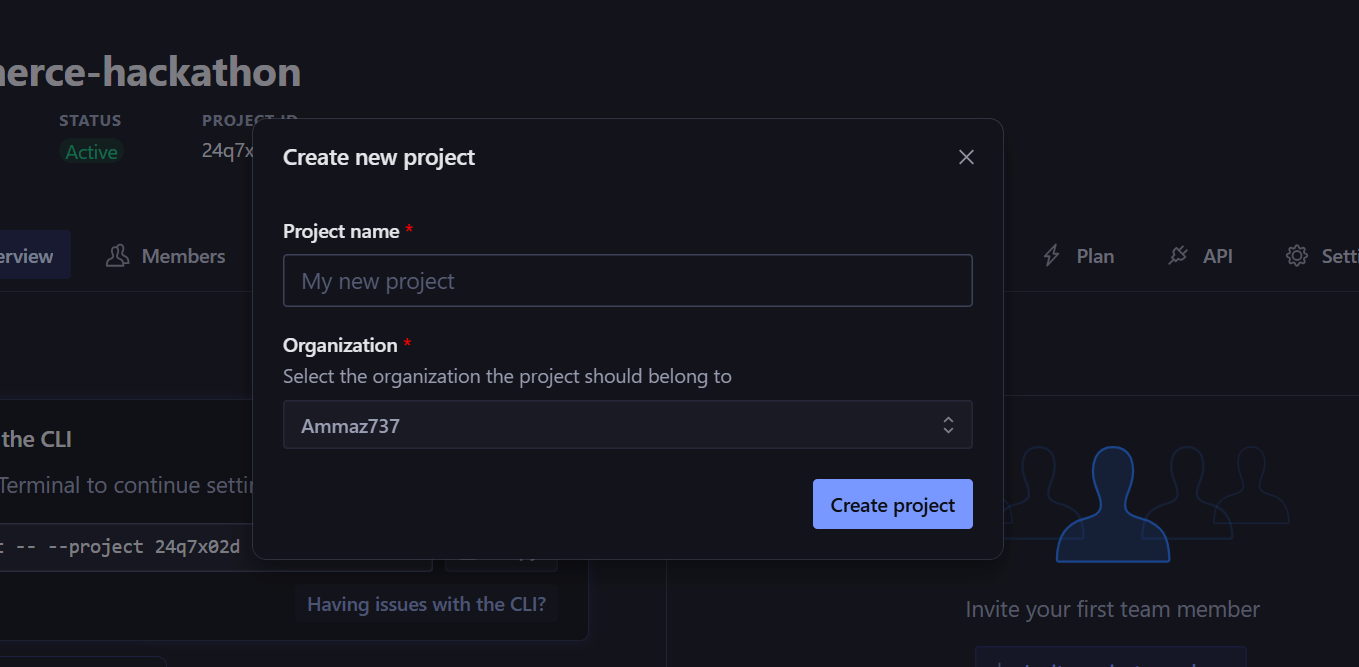
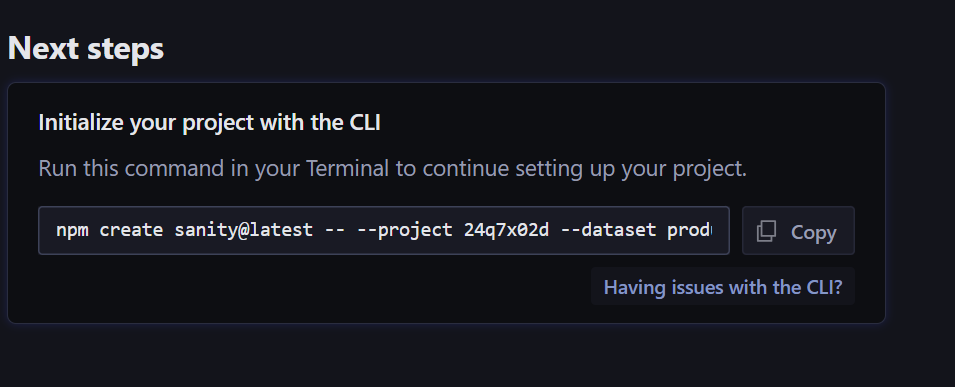
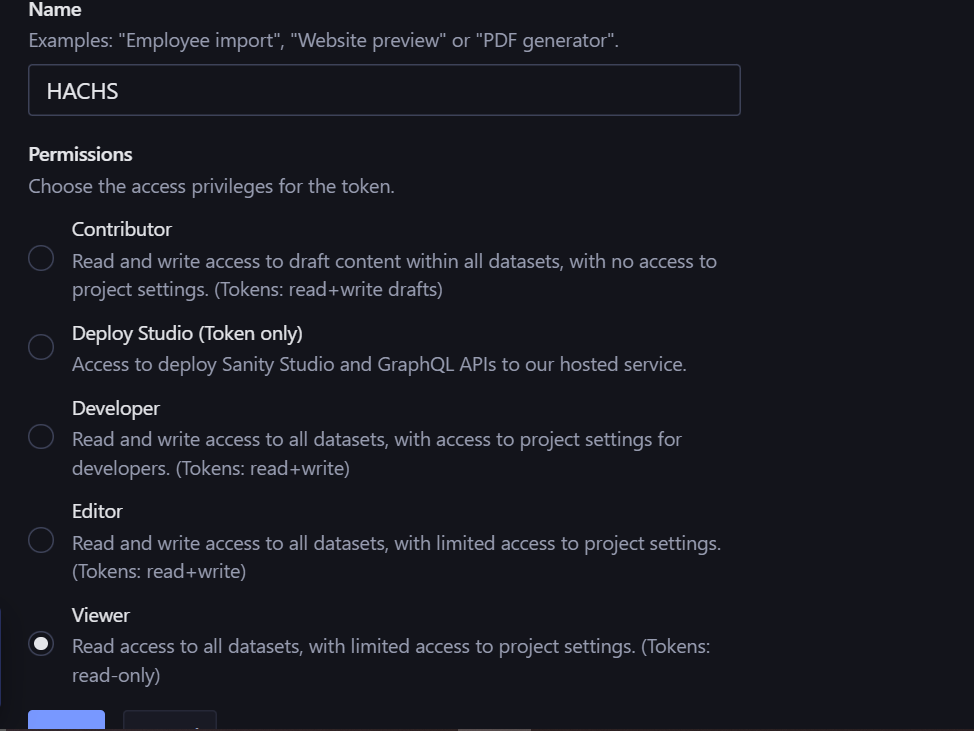
1. **Create Sanity Project**

****

1. **Configure Sanity in Next.js Project:**

****

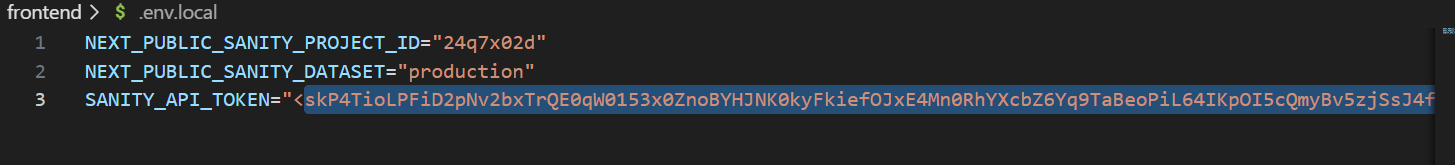
1. **Create Api Tokens**

****

1. **Api Token**

skP4TioLPFiD2pNv2bxTrQE0qW0153x0ZnoBYHJNK0kyFkiefOJxE4Mn0RhYXcbZ6Yq9TaBeoPiL64IKpOI5cQmyBv5zjSsJ4fco3Ai0zrLTVoUXtrAMqjIYW8GSlxrpqix89MA21nfcka8zjlpe3lMJf5Eyja2Fpm2rRZgyeICkIJ4e5yFT

1. **Setup .env local**

****

1. **SchemaTypes**
2. **Product.ts**
3. import { defineType } from "sanity";
4. export const productSchema = defineType({
5. name: "products",
6. title: "Products",
7. type: "document",
8. fields: [
9. {
10. name: "title",
11. title: "Product Title",
12. type: "string",
13. },
14. {
15. name: "price",
16. title: "Price",
17. type: "number",
18. },
19. {
20. title: "Price without Discount",
21. name: "priceWithoutDiscount",
22. type: "number",
23. },
24. {
25. name: "badge",
26. title: "Badge",
27. type: "string",
28. },
29. {
30. name: "image",
31. title: "Product Image",
32. type: "image",
33. },
34. {
35. name: "category",
36. title: "Category",
37. type: "reference",
38. to: [{ type: "categories" }],
39. },
40. {
41. name: "description",
42. title: "Product Description",
43. type: "text",
44. },
45. {
46. name: "inventory",
47. title: "Inventory Management",
48. type: "number",
49. },
50. {
51. name: "tags",
52. title: "Tags",
53. type: "array",
54. of: [{ type: "string" }],
55. options: {
56. list: [
57. { title: "Featured", value: "featured" },
58. {
59. title: "Follow products and discounts on Instagram",
60. value: "instagram",
61. },
62. { title: "Gallery", value: "gallery" },
63. ],
64. },
65. },
66. ],
67. });

**2. 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',

        }

    ],

});

1. **Scripts**
2. **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 = "24q7x02d", // Sanity project ID

  NEXT\_PUBLIC\_SANITY\_DATASET = "production",  // Sanity dataset

  NEXT\_PUBLIC\_SANITY\_AUTH\_TOKEN = "skP4TioLPFiD2pNv2bxTrQE0qW0153x0ZnoBYHJNK0kyFkiefOJxE4Mn0RhYXcbZ6Yq9TaBeoPiL64IKpOI5cQmyBv5zjSsJ4fco3Ai0zrLTVoUXtrAMqjIYW8GSlxrpqix89MA21nfcka8zjlpe3lMJf5Eyja2Fpm2rRZgyeICkIJ4e5yFT", // Sanity API token

  BASE\_URL = "https://giaic-hackathon-template-08.vercel.app", // Base URL for APIs

} = 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);

}

// Create a Sanity client instance

const targetClient = createClient({

  projectId: NEXT\_PUBLIC\_SANITY\_PROJECT\_ID,

  dataset: NEXT\_PUBLIC\_SANITY\_DATASET,

  useCdn: false,

  apiVersion: "2023-01-01",

  token: NEXT\_PUBLIC\_SANITY\_AUTH\_TOKEN,

});

// Function to upload an image to Sanity

async function uploadImageToSanity(imageUrl) {

  try {

    const response = await fetch(imageUrl);

    if (!response.ok) throw new Error(`Failed to fetch image: ${imageUrl}`);

    const buffer = await response.arrayBuffer();

    const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {

      filename: imageUrl.split("/").pop(),

    });

    return uploadedAsset.\_id;

  } catch (error) {

    console.error("Error uploading image:", error.message);

    return null;

  }

}

// Main function to migrate data from REST API to Sanity

async function migrateData() {

  console.log("Starting data migration...");

  try {

    // Fetch categories

    const categoriesResponse = await fetch(`${BASE\_URL}/api/categories`);

    if (!categoriesResponse.ok) throw new Error(`Failed to fetch categories: ${categoriesResponse.statusText}`);

    const categoriesData = await categoriesResponse.json();

    // Fetch products

    const productsResponse = await fetch(`${BASE\_URL}/api/products`);

    if (!productsResponse.ok) throw new Error(`Failed to fetch products: ${productsResponse.statusText}`);

    const productsData = await productsResponse.json();

    const categoryIdMap = {};

    // Migrate categories

    for (const category of categoriesData) {

      console.log(`Migrating category: ${category.title}`);

      const imageId = await uploadImageToSanity(category.imageUrl);

      const newCategory = {

        \_id: category.\_id,

        \_type: "categories",

        title: category.title,

        image: imageId ? { \_type: "image", asset: { \_ref: imageId } } : undefined,

      };

      const result = await targetClient.createOrReplace(newCategory);

      categoryIdMap[category.\_id] = result.\_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);

      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],

        },

        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);

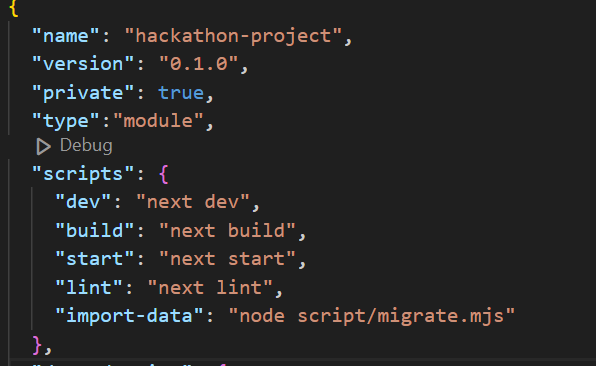
  }

}

// Start the migration process

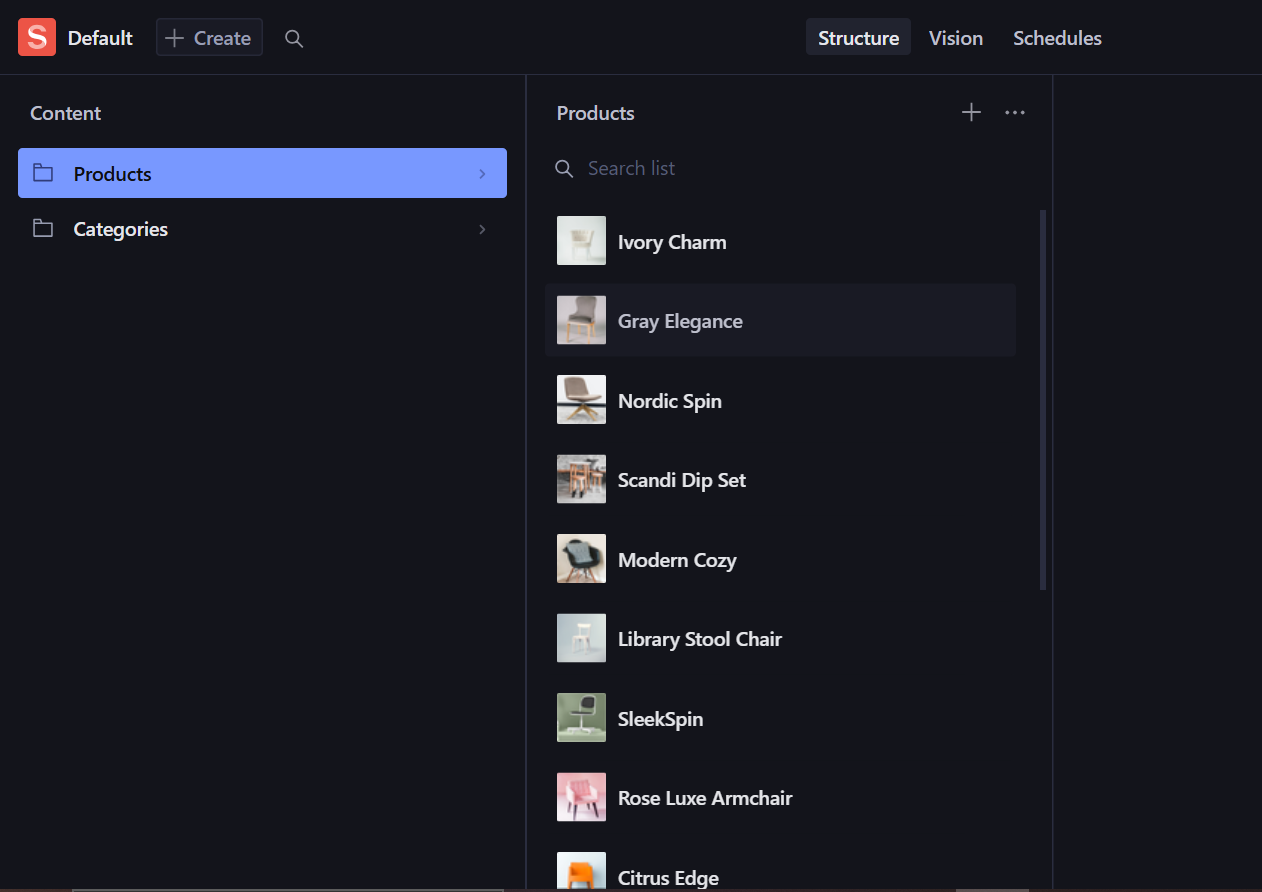
migrateData();

1. **package.json**

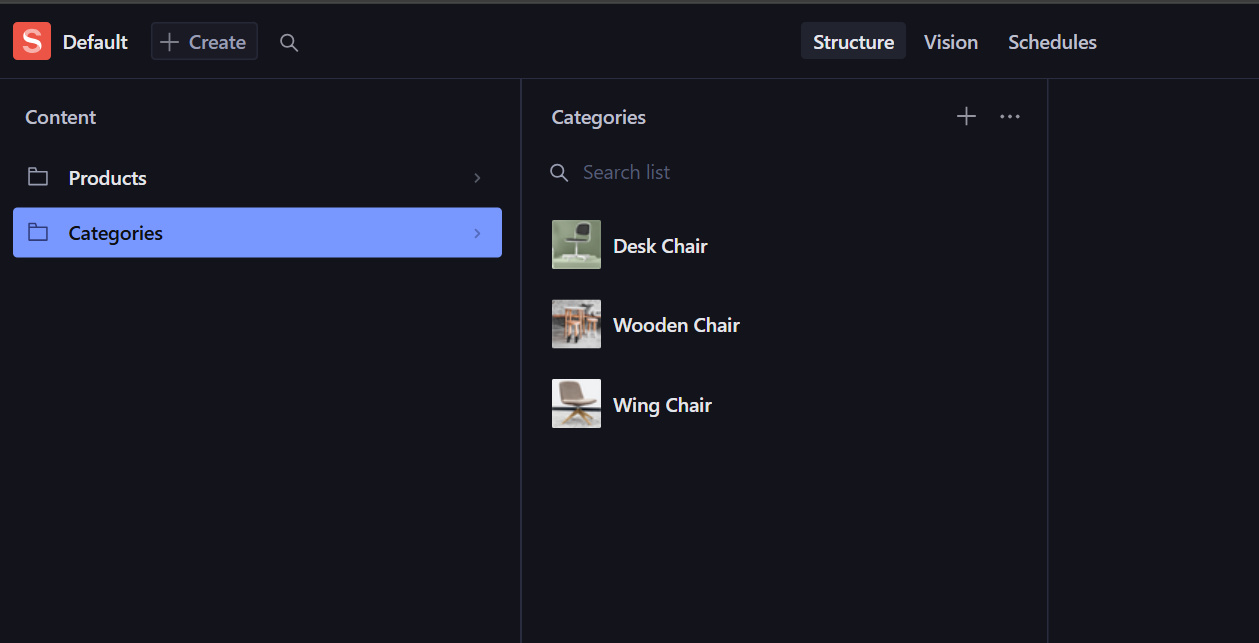
****

1. **Sanity Products**

**1.**

****

**2.**

****