

Market Place Technical Foundation- Comforty(Template 8) Name: Abdullah Shaikh Day-3 API Integration and showing Sanity Data in Frontend.

Sanity Installation:

I install sanity to keep data of products and categories in it.

```
C:\Users\De11\Documents\Market_Place_Builder_Hackathon_2025_GIAIC\my-app>npm create sanity@latest
Need to install the following packages:
create-sanity@3.70.0
Ok to proceed? (y) y
✓ You are logged in as abdullahkamran6601@gmail.com using Google
✓ Fetching existing projects

? Create a new project or select an existing one Market_Place_Builder_Hackathon_2025_GIAIC (59wvk5e2)
? Select dataset to use production
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
It looks like you are using Next.js 15 and React 19
Please read our compatibility guide.
https://www.sanity.io/help/react-19
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@'
npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm WARN deprecated @humanwhododes/config-array@0.13.0: Use @eslint/config-array instead
npm WARN deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm WARN deprecated @humanwhododes/object-schema@2.0.3: Use @eslint/object-schema instead
npm WARN deprecated @sanity/block-tools@3.70.0: Renamed - use '@portabletext/block-tools' instead. '@sanity/block-tools' will no longer receive updates.
npm WARN deprecated eslint@8.57.1: This version is no longer supported. Please see https://eslint.org/version-support for other options.

added 1298 packages, and audited 1299 packages in 4m

244 packages are looking for funding
  run 'npm fund' for details

1 moderate severity vulnerability
```

```
To address all issues, run:  
npm audit fix --force
```

```
Run `npm audit` for details.
```

```
Success! Your Sanity configuration files has been added to this project
```

Making Schemas:

Then, I make schemas for products and categories:

Products:

```
my-app > src > sanity > schemaType > TS products.ts > productsSchema > fields > type  
1 import { defineArrayMember, defineField, defineType } from "sanity";  
2 export const productsSchema = defineType({  
3   title: "Products",  
4   name: "products",  
5   type: "document",  
6   fields: [  
7     defineField({  
8       title: "ID",  
9       name: "id",  
10      type: "string",  
11      description: "It should be Unique for every product so use UUID generator for it.",  
12      validation: Rule => Rule.required()  
13    }),  
14    defineField({  
15      title: "Title of Product",  
16      name: "title",  
17      type: "string",  
18      description: "Enter Title or name of product",  
19      validation: Rule => Rule.required()  
20    }),  
21    defineField({  
22      title: "Image of Product",  
23      name: "image",  
24      type: "image",  
25      description: "Upload Image of your Product",  
26      validation: Rule => Rule.required()  
27    }),  
28    defineField({  
29      title: "Price Without Discount",  
30      name: "priceWithoutDiscount",  
31      type: "number",  
32      description: "Write price without discount if you are not giving discount leave it to 0",  
33      initialValue: 0,  
34      validation: Rule => Rule.required()  
35    }),  
36    defineField({  
37      title: "Price of Product",
```

```

36     defineField({
37       title: "Price of Product",
38       name: "price",
39       type: "number",
40       description: "Write actual price of product",
41       validation: Rule => Rule.required()
42     }),
43     defineField({
44       title: "Badge?",
45       name: "badge",
46       type: "string",
47       validation: Rule => Rule.required()
48     }),
49     defineField({
50       title: "Description of product",
51       name: "description",
52       type: "text",
53       validation: Rule => Rule.required().max(600)
54     }),
55     defineField({
56       title: "Inventory Of Product",
57       name: "inventory",
58       type: "number",
59       validation: Rule => Rule.required()
60     }),
61     defineField({
62       name: "category",
63       title: "Category",
64       type: "reference",
65       to: [{ type: "category" }],
66       description: "Choose Category of product",
67       validation: Rule => Rule.required()
68     }),
69     defineField({

```

```

69     defineField({
70       title: "Tags of Product",
71       name: "tags",
72       type: "array",
73       of: [
74         defineArrayMember({
75           type: "string"
76         })
77       ],
78       options: {
79         list: [
80           { title: "Featured", value: "featured" },
81           {
82             title: "Follow products and discounts on Instagram",
83             value: "instagram",
84           },
85           { title: "Gallery", value: "gallery" },
86         ],
87       },
88       validation: Rule => Rule.required()
89     }),
90   ]
91 })

```

Categories:

```

my-app > src > sanity > schemaTypes > TS categories.ts > 60 categoriesSchema
1  import { defineField, defineType } from "sanity";
2
3  export const categoriesSchema = defineType({
4    title: "Categories",
5    name: "categories",
6    type: "document",
7    fields: [
8      defineField({
9        title: "ID",
10       name: "id",
11       type: "string",
12       description: "It should be Unique for every category so use UUID generator for it.",
13       validation: Rule => Rule.required()
14     }),
15     defineField({
16       title: "Title of Category",
17       name: "title",
18       type: "string",
19       description: "Enter Title or name of category",
20       validation: Rule => Rule.required()
21     }),
22     defineField({
23       title: "Image of Product",
24       name: "image",
25       type: "image",
26       description: "Upload Feature Image of your category",
27       validation: Rule => Rule.required()
28     })
29   ]
30 })

```

ENV:

I make env file and make environment variables in it

```
1. NEXT_PUBLIC_SANITY_PROJECT_ID
2. NEXT_PUBLIC_SANITY_DATASET
3. NEXT_PUBLIC_SANITY_AUTH_TOKEN
4. baseUrl
```



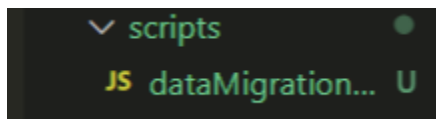
Data Migration:

For mock data migration I use these api's for it.

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

<https://giaic-hackathon-template-08.vercel.app/api/categories>

To migrate it I make migration file in scripts folder:



and then write this code in it

```
// 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: "category",
        title: category.title,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } :
undefined, // Add image if uploaded
      };

      // Save the category to Sanity
      // const result = await targetClient.create(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: "product",
  id: product._id,
  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();

```

and then add this in package.json:

```
"scripts": {  
  "dev": "next dev --turbo",  
  "build": "next build",  
  "start": "next start",  
  "lint": "next lint",  
  "migrateData": "node scripts/dataMigration.mjs"  
},
```

And then run `npm run migrateData` to migrate Data.