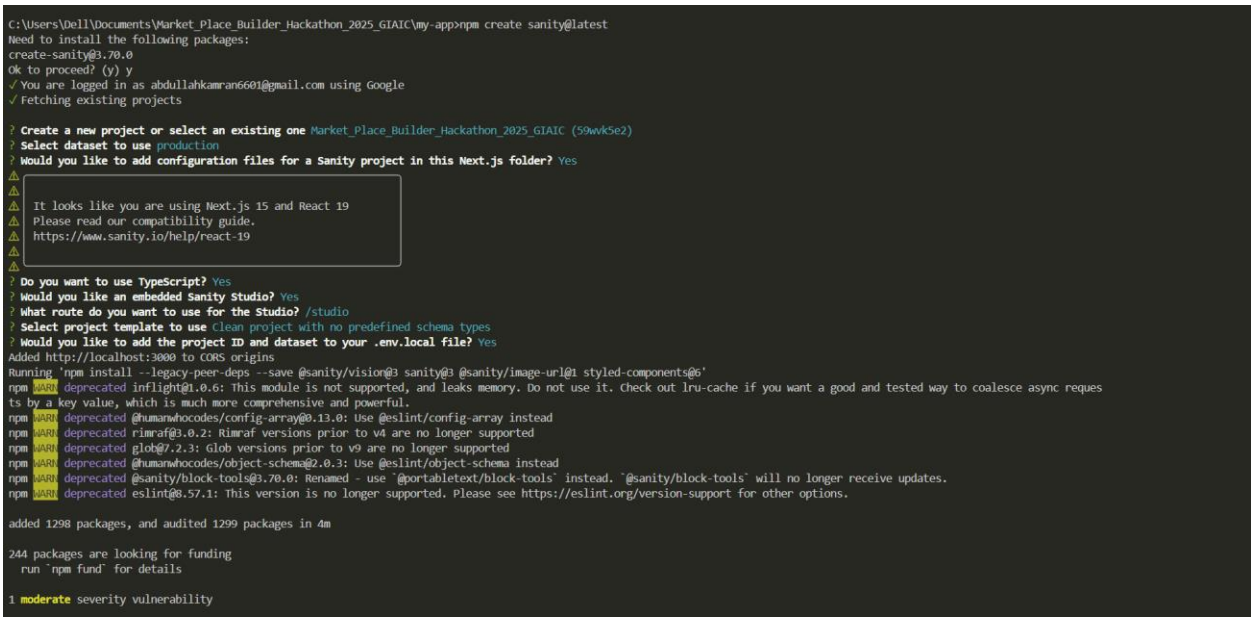
Market Place Technical Foundation- Comforty(Template 8) Name: Abdullah Shaikh Day-3 API Integration and showing Sanity Data in Frontend.

Sanity Installation:

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

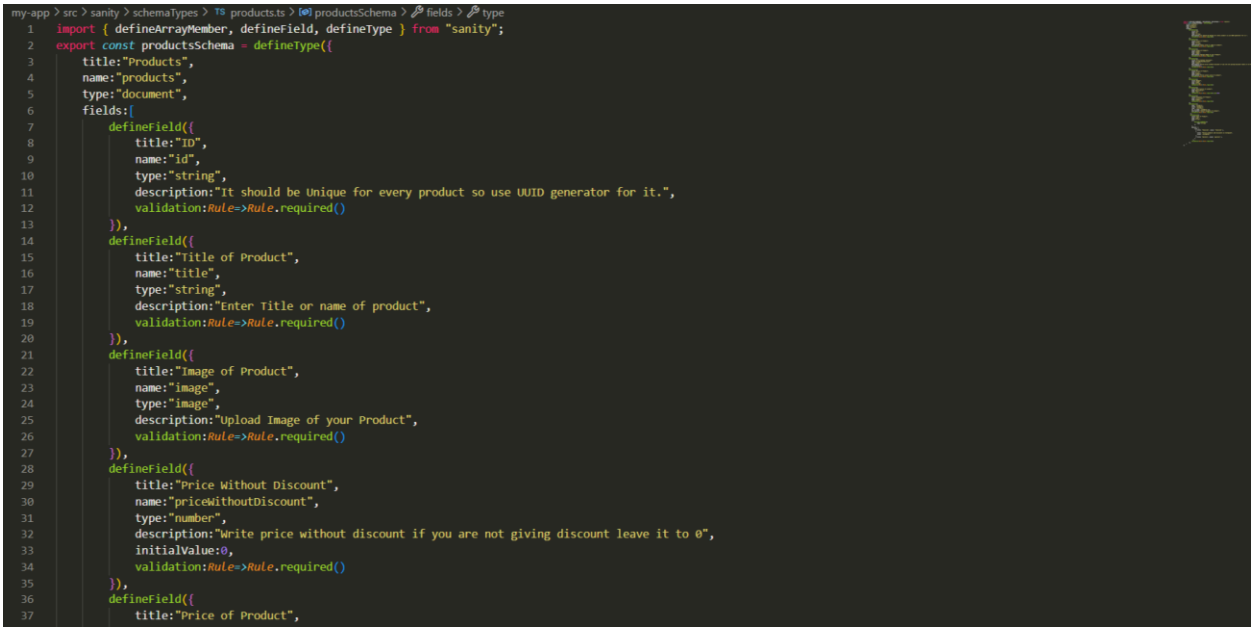


```
To address all issues, run:
  npm audit fix --force

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 > schemaTypes > products.ts > productsSchema > fields > type
import { defineArrayMember, defineField, defineType } from "sanity";
export const productsSchema = defineType({
    title:"Products",
    name:"products",
    type:"document",
    fields:[
        defineField({
            title:"ID",
            name:"id",
            type:"string",
            description:"It should be Unique for every product so use UUID generator for it.",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Title of Product",
            name:"title",
            type:"string",
            description:"Enter Title or name of product",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Image of Product",
            name:"image",
            type:"image",
            description:"Upload Image of your Product",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Price Without Discount",
            name:"priceWithoutDiscount",
            type:"number",
            description:"Write price without discount if you are not giving discount leave it to 0",
            initialValue:0,
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Price of Product",
```

```
        defineField({
            title:"Price of Product",
            name:"price",
            type:"number",
            description:"Write actual price of product",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Badge?",
            name:"badge",
            type:"string",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Description of product",
            name:"description",
            type:"text",
            validation:Rule=>Rule.required().max(600)
        }),
        defineField({
            title:"Inventory Of Product",
            name:"inventory",
            type:"number",
            validation:Rule=>Rule.required()
        }),
        defineField({
            name: "category",
            title: "Category",
            type: "reference",
            to: [{ type: "category" }],
            description:"Choose Category of product",
            validation:Rule=>Rule.required()
          }),
          defineField({
```

```
          defineField({
            title:"Tags of Product",
            name:"tags",
            type:"array",
            of:[
                defineArrayMember({
                    type:"string"
                })
            ],
            options: {
                list: [
                  { title: "Featured", value: "featured" },
                  {
                    title: "Follow products and discounts on Instagram",
                    value: "instagram",
                  },
                  { title: "Gallery", value: "gallery" },
                ],
              },
            validation:Rule=>Rule.required()
        }),
    ]
})
```

Categories:

```
my-app > src > sanity > schemaTypes > TS categories.ts > [o] categoriesSchema
import { defineField, defineType } from "sanity";

export const categoriesSchema = defineType({
    title:"Categories",
    name:"categories",
    type:"document",
    fields:[
        defineField({
            title:"ID",
            name:"id",
            type:"string",
            description:"It should be Unique for every category so use UUID generator for it.",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Title of Category",
            name:"title",
            type:"string",
            description:"Enter Title or name of category",
            validation:Rule=>Rule.required()
        }),
        defineField({
            title:"Image of Product",
            name:"image",
            type:"image",
            description:"Upload Feature Image of your category",
            validation:Rule=>Rule.required()
        }),
    ]
})
```

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
```

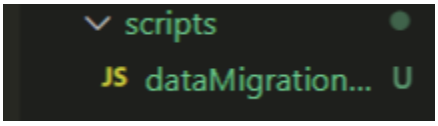
Data Migration:

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

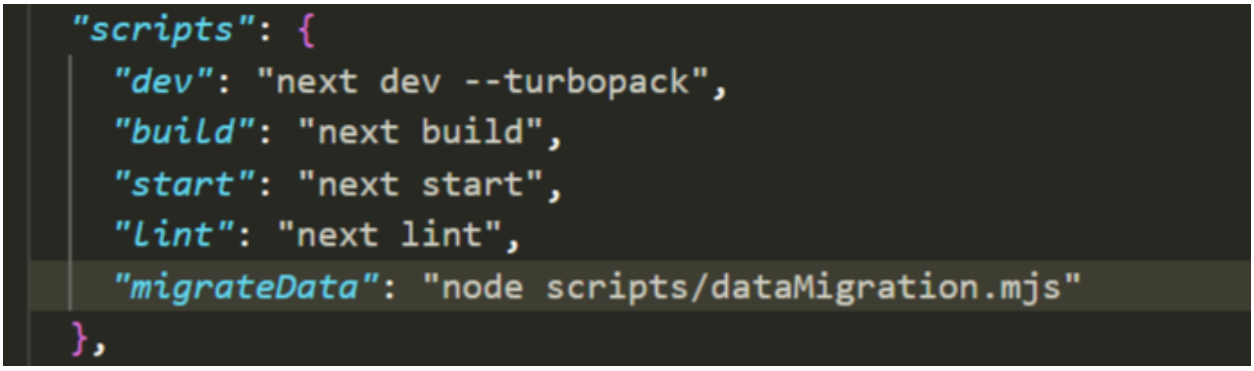
```
// Import environment variables from .env.local
import "dotenv/config";

// 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:



And then run npm run migrateData to migrate Data.