Hackathon Day 3: API Integration and Data Migration

Object : The focus for Day 3 is integrating APIs and migrating data into Sanity CMS to build a functional marketplace backend

1-Insatllation of Sanity

```
npm install -g @sanity/cli
```

2-Sanity Schema

Inside the src/schema folder create two files: product.ts and category.ts

- Product.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" },
        ],
      },
    },
  ],
});
```

- category.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',
        }
    ],
});
```

- Inside the schemas/index.js file, import the schemas and export them as an array

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

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

3-Data Migration

- First, create a scripts folder inside your project where you will store your migration scripts

- Inside this folder, create a file named migrate.mjs This file will contain the logic for your migration

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

import dotenv from 'dotenv';
dotenv.config(); // Load environment variables from .env.local file

console.log(process.env.NEXT_PUBLIC_SANITY_PROJECT_ID); // Check if the variable is loaded correctly


// 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})`);
  }
```

```
    // 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();
```

Add Scripts to package.json

Then, add the migration command to the scripts section of your package.json

```
"scripts": {
   "migrate": "node scripts/migrate.mjs"
}
```

Run the Command:

- Now, you can run the migration command by executing npm run migrate in your terminal. This will run the migration script and start the data migration
- Then install the dotenv command

```
npm install dotenv
```

- After installing all the required dependencies and setting up the environment, you can run the migration with

```
npm run migrate
```

Checklist

- API Understanding
- Schema Validation
- Data Migration
- API Integration In Next.js
- Submission

Sumbal Naz