



Roll No: 307318
Slot: FRIDAY morning (9am to 12pm)

HACKATHON 3 – DAY 03

API MIGRATION TO SANITY and INSERTING DATA TO FRONT-END FROM SANITY

API documentation Review:

- I read the documentation of my assigned template to understand its requirements and to implement it correctly
- I explored the whole document including schema files, migration file etc...

API testing:

- First of all I tested the APIs on browser if they are working properly or not.
- Then I used thunder client to test their endpoints and ensured the data was being returned correctly.

Adjustments in Sanity Schema:

- I reviewed the given schema with the schema I already created according to the requirements of my project.
- Then I have made some changes to it like Slug (as I wanted slug to be the endpoint of my product's dynamic page), inventory, and tags (for the products being fetched according to the tags)

Following is the updated sanity schema:

- The Products Schema

src/sanity/schemaTypes/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: "slug",
      title: "Slug",
      type: "slug",
      options: {
        source: "title",
        maxLength: 200,
      },
    },
    {
      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" },
        ],
      },
    },
  ],
})
```

- The Categories schema:

src/sanity/schemaTypes/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',
    }
  ],
});
```

Token and Keys placements:

- Placed an API key, data set inside env.local
- Generated a token at sanity.io on my existing project

API migration:

- Then I made a scripts folder at the root directory and inside that scripts I created a file “migrate.mjs” through which we can Migrate our data through an API to sanity
- Used the sanity keys and token in the migration.mjs

Below is the migration file:

```

scripts/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, // Sanity project ID
  NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
  NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
  BASE_URL = "https://giatic-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 imageUrl = 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: imageUrl ? { _type: "image", asset: { _ref: imageUrl } } : 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 imageUrl = 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: imageUrl ? { _type: "image", asset: { _ref: imageUrl } } : undefined, // Add image if
        uploaded
      };

      // Save the product to Sanity
      const result = await targetClient.create(newProduct);
      console.log("Migrated product: ${product.title} (ID: ${result._id})");
    }

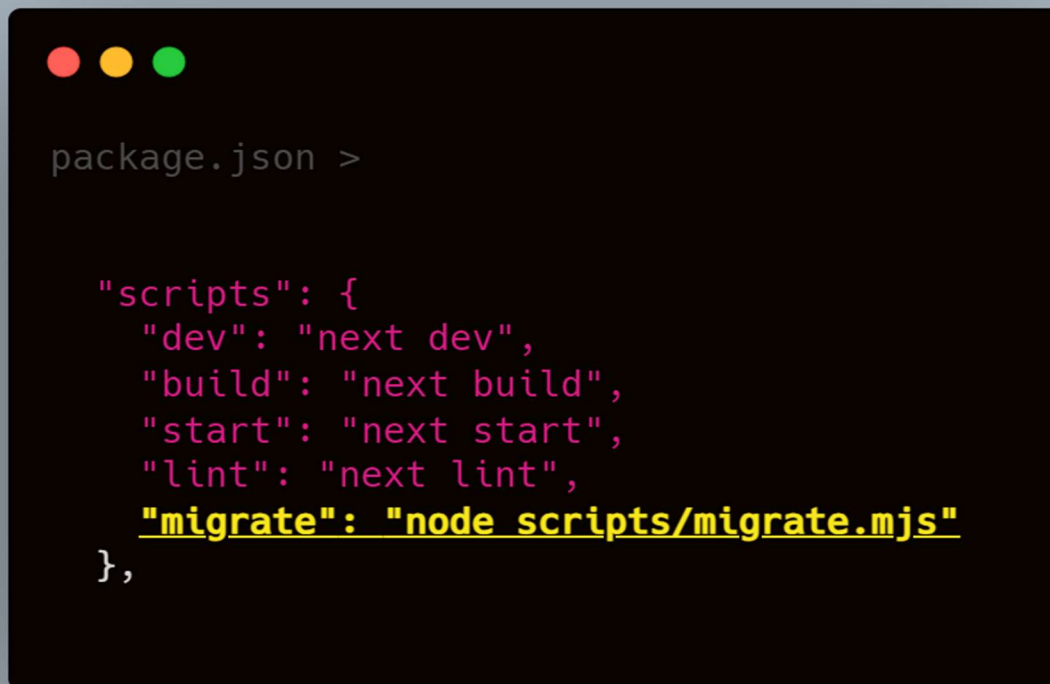
    console.log("Data migration completed successfully!");
  } catch (error) {
    console.error("Error during migration:", error.message);
    process.exit(1); // Stop execution if an error occurs
  }
}

// Start the migration process
migrateData();

```

Setting the command to run on the terminal:

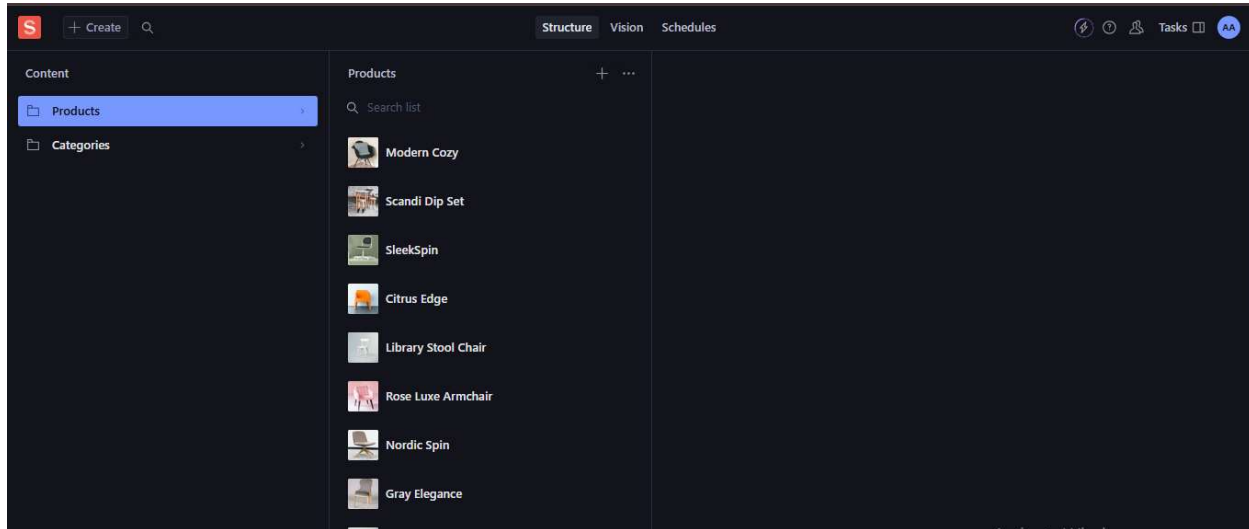
- To create a command for integrating data to sanity I followed a procedure
- Made some changes in “package.json” file
Below is the picture attached:



```
package.json >  
  
"scripts": {  
  "dev": "next dev",  
  "build": "next build",  
  "start": "next start",  
  "lint": "next lint",  
  "migrate": "node scripts/migrate.mjs"  
},
```

Through this process data had successfully imported to
sanity studio.

Here you can see it:



Final steps:

- Fetched data to Front-end
- By using GROQ query
- And finally I have fetched the data from sanity and used it in my front-end

THE CODE of few files:

src/app/products/page.tsx

```
"use client"

import { client } from "@/sanity/lib/client"
import Image from "next/image"
import Link from "next/link"
import { useState, useEffect } from "react"
import Toast from "react-hot-toast"
import AddToCart2 from "@/components/AddToCart2"

interface Items {
  _id: string
  title: string
  price: number
  price_id: string
  slug: string
  image_id: string
  inventory: number
  badge?: string
  priceWithoutDiscount?: number
}

async function getOurProducts() {
  const query = `*[_type == "products"] [0...8] {
    _id,
    title,
    price,
    price_id,
    slug,
    "image_id": image.asset->url,
    inventory,
    badge, // Fetch badge field
    priceWithoutDiscount // Fetch priceWithoutDiscount field
  }`
  const items = await client.fetch(query)
  return items
}

export default function OurProducts() {
  const [items, setItems] = useState<Items[]>([])

  useEffect(() => {
    const fetchProducts = async () => {
      const products = await getOurProducts()
      setItems(products)
    }
    fetchProducts()
  }, [])

  const handleAddToCart = (item: Items) => {
    if (item.inventory > 0) {
      Toast.success(`${item.title} has been added to your cart`, {
        duration: 3000,
      })
    } else {
      Toast.error(`${item.title} is out of stock`, {
        duration: 3000,
      })
    }
  }

  return (
    <div
      className="max-w-screen-2xl mx-auto overflow-x-hidden py-10 lg:py-20"
      >
      <div
        className="text-customBlue text-center xl:text-left lg:text-left xl:font-bold text-xl sm:text-2xl pb-3 lg:pb-10"
        >
        All Products
      </div>
      <div
        className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-x-4 gap-y-5 lg:gap-y-10 px-4 md:px-10 lg:px-0 overflow-hidden"
        >
        {items.map((item) => (
          <div
            key={item._id}
            className="group relative"
            >
            <div
              className="w-full overflow-hidden rounded-md bg-white transition-transform duration-200 hover:scale-105 group-hover:opacity-75"
              >
              <img
                src={item.image_id}
                alt={item.title}
                width={500}
                height={500}
                className="w-full h-full object-cover object-center"
              />
              </div>
              <div
                className="display flex justify-between"
                >
                <div
                  className="text-customBlue pt-2"
                  >
                  {item.title}
                </div>
                <div
                  className="flex items-center gap-2"
                  >
                  <div
                    className="text-lg font-medium"
                    >
                    {item.price}
                  </div>
                  <div
                    className="text-gray-500 line-through text-sm"
                    >
                    {item.priceWithoutDiscount}
                  </div>
                </div>
              </div>
              <div
                className="mt-4 flex justify-between"
                >
                <div
                  >
                  {item.badge}
                </div>
                <div
                  >
                  {item.inventory}
                </div>
              </div>
              <div
                className="relative"
                >
                <button
                  onClick={() => handleAddToCart(item)}
                  disabled={item.inventory === 0}
                  className={
                    item.inventory === 0 ? "opacity-50 cursor-not-allowed" : ""
                  }
                >
                  AddToCart2
                </button>
                <div
                  className="absolute bottom-0 left-0 w-full bg-black text-white text-xs py-3 text-center opacity-0 group-hover:opacity-100 transition-opacity duration-300"
                  >
                  out of Stock
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
    </div>
  )
}
```

THE FINAL FRONT-END UI:

Featured Products



Library Stool Chair
\$20



Rose Luxe Armchair
\$20 ~~\$38~~



Citrus Edge
\$20



Ivory Charm
\$20



Top Categories



All Products



Citrus Edge
\$20



Scandi Dip Set
\$40



Rose Luxe Armchair
\$20 ~~\$36~~



Modern Cozy
\$20



Library Stool Chair
\$20



SleekSpin
\$20



Nordic Spin
\$30



Gray Elegance
\$8

