"Day 3 - API Integration Report - [Muniba e-shop]"

Understand the Provided API:

The given api for template 2 is: https://hackathon-apis.vercel.app/api/products

I test the api through thunderstorm extension

Identified given end points:

o Product listings (i.e, /products)

Validate and Adjust Your Schema:

• Compare your existing Sanity CMS schema (created on Day 2) with the API data structure.

• Adjust field names, types, and relationships to ensure compatibility and remove extra fields like discountpercentage and replace id with slug.

API integration process:

Here, I use the given api of template 2 with the process of importing from writing script file in scripts folder: scripts>importData.mjs (i convert typescript given file to mjs format).

Adjustments made to schemas:

I am using the given schemas, where I make changes in slug field:

```
 defineField({
      name: "slug",
      title: "Slug",
      type: "slug",
      options: {
       source: "name", // Replace 'name' with the field you want to use as the source for slug generation
        maxLength: 200, // Optional: Limit the length of the generated slug
        slugify: (input) =>
         input
           .toLowerCase()
           .replace(/\s+/g, '-') // Replace spaces with dashes
```

```
      .replace(/[^a-z0-9-]/g, ''), // Remove special characters
    },
    validation: (rule) => rule.required(),
  }),
```

And make it able to generate slug from name field.

I also add another field of price_id , because I am using stripe for payment, stripe give us price_id for developers, so we work on it to get price from our data.

Migration steps and tools used:

With the help of this blog I migrate given api data to my sanity schema:

How to integrate External Data on Sanity in NextJS project.

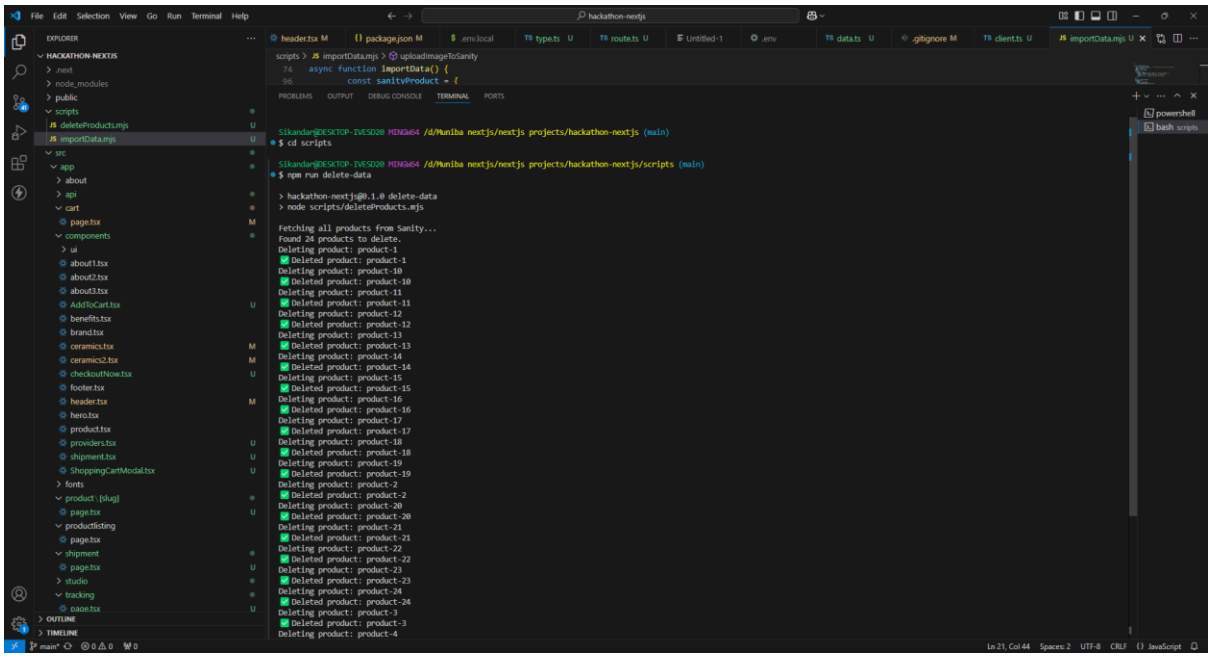
A step-by-step guide:

https://medium.com/@huzaifa3108hassan/integrating-sanity-with-next-js-a-guide-to-data-import-and-environment-setup-760eb41ea2a2

I also make deleteProduct.mjs file. If i need to delete all products because of any reason like to import new products in my sanity than in my nextjs project.

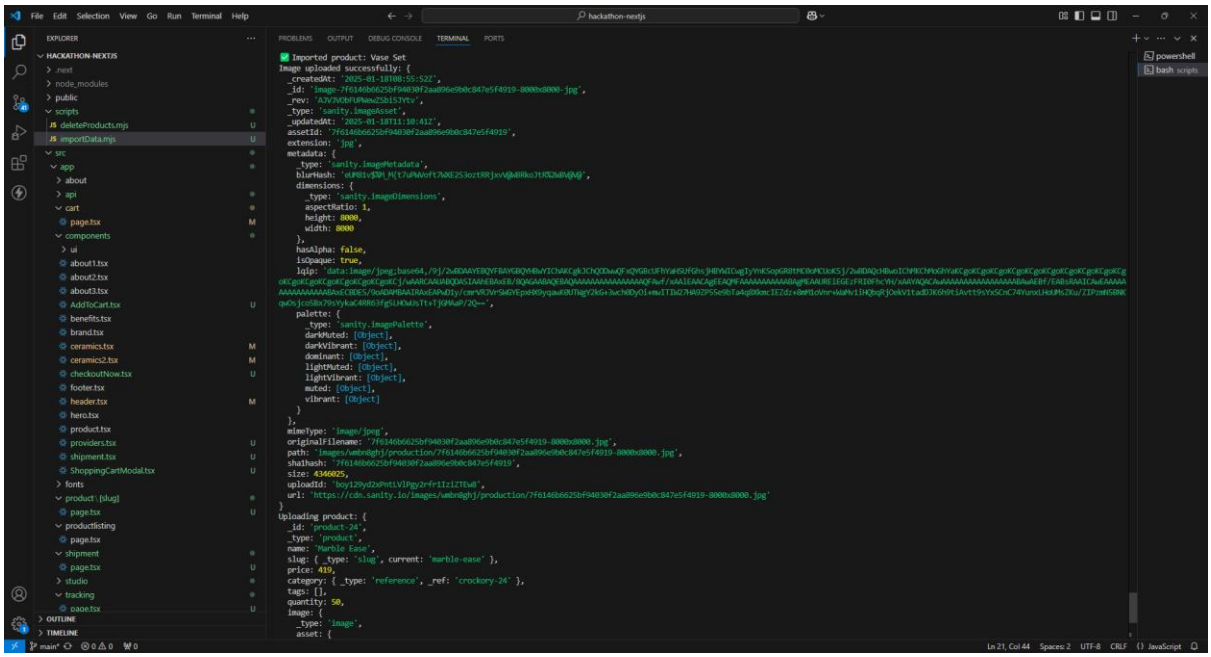
Screenshot of deleting products:

Command: npm run delete-data

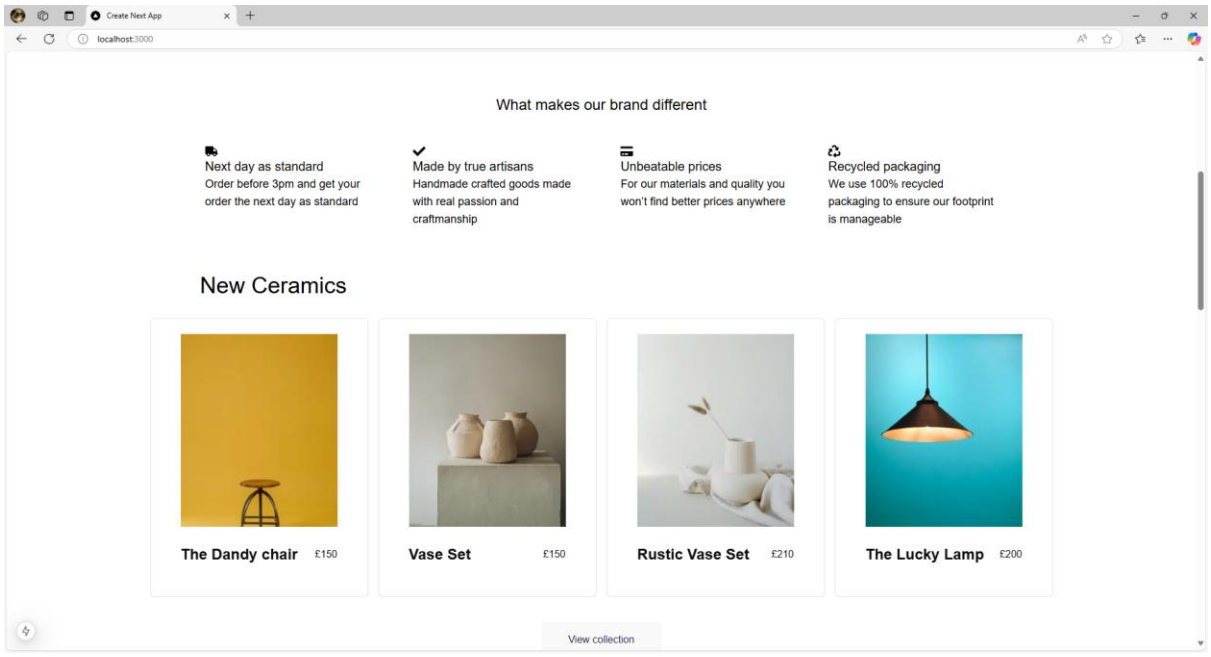


Screenshot of importing products:

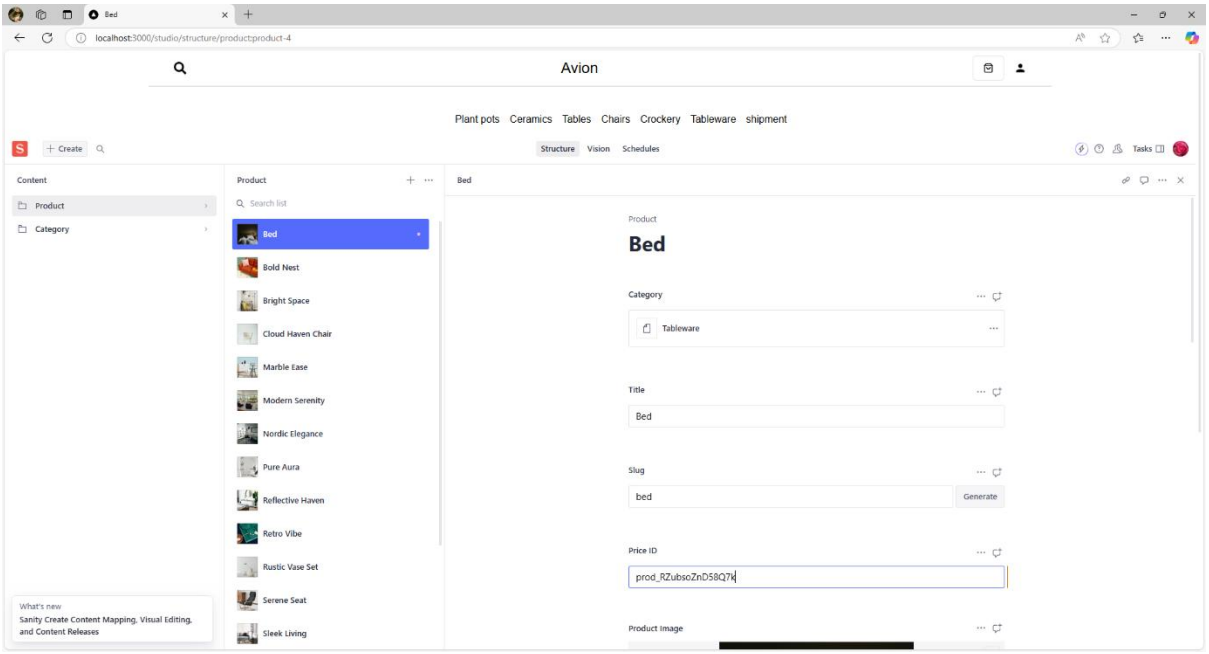
Command: npm run import-data



Data successfully displayed in the frontend:



Screenshot of Populated Sanity CMS fields.



Code snippets for API integration and migration scripts:

1- For importing data:

```
import { createClient } from '@sanity/client'
import axios from 'axios'
import dotenv from 'dotenv'
import { fileURLToPath } from 'url'
import path from 'path'
import slugify from 'slugify';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url)
const __dirname = path.dirname(__filename)
dotenv.config({ path: path.resolve(__dirname, '../.env.local') })
// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
})
async function uploadImageToSanity(imageUrl) {

  try {
    // Fetch the image from the URL and convert it to a buffer
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer',timeout: 10000 });
    const buffer = Buffer.from(response.data);

    // Upload the image to Sanity
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop(), // Extract the filename from URL
    });

    // Debugging: Log the asset returned by Sanity
    console.log('Image uploaded successfully:', asset);

    return asset._id; // Return the uploaded image asset reference ID
  } catch (error) {
    console.error('❌ Failed to upload image:', imageUrl, error);
    return null
    //throw error;
  }
}


async function createCategory(category,counter) {

  try {
    const categoryExist = await client.fetch(`*[_type=="category" && slug==$slug][0]`,
{slug:category.slug})
    if(categoryExist)
    {
      return categoryExist._id
    }
    const catObj = {
      _type:"category",
      _id:category.slug+"-"+counter,
      name:category.name,
      slug:category.slug
    }
    const response = await client.createOrReplace(catObj)

    // Debugging: Log the asset returned by Sanity
    console.log('Category created successfully', response);

    return response._id; // Return the uploaded image asset reference ID
  } catch (error) {
    console.error('❌ Failed to category:', category.name, error);
    //throw error;
  }
}

async function importData() {
    try {
      // Fetch data from external API
      const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
      const products = response.data;
      //console.log(products)
      let counter=1;
      // Iterate over the products
      for (const product of products) {
        let imageRef = null;
        let catRef=null;

        // Upload image and get asset reference if it exists
        if (product.image) {
          //imageRef = await uploadImageToSanity(product.imageUrl);
          imageRef = await uploadImageToSanity(product.image);
        }

        if(product.category.name){
          catRef = await createCategory(product.category,counter)
        }

        const sanityProduct = {
          _id: `product-${counter}`, // Prefix the ID to ensure validity
          _type: 'product',
          name: product.name,
          slug: {
            _type: 'slug',
            current: slugify(product.name || 'default-product', {
              lower: true, // Ensure the slug is lowercase
              strict: true, // Remove special characters
            }),
          },
          price: product.price,
          category:{
            _type: 'reference',
            _ref:catRef?catRef:undefined
          },
          tags: product.tags?product.tags:[],
          quantity:50,
          image: imageRef ? {
            _type: 'image',
            asset: {
              _type: 'reference',
              _ref: imageRef, // Set the correct asset reference ID
            },
          } : undefined,
          description: product.description?product.description: "A timeless design, with premium
materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any
stylish living space with beech legs and lambskin leather upholstery.",
          features: product.features?product.features: [
            "Premium material",
            "Handmade upholstery",
            "Quality timeless classic",
          ],
          dimensions: product.dimensions?product.dimensions : {
            _type: 'dimensions', // Custom object type for dimensions
            height: "110cm",
            width: "75cm",
            depth: "50cm",
          }
        };
        counter++
        // Log the product before attempting to upload it to Sanity
        console.log('Uploading product:', sanityProduct);

        // Import data into Sanity
        await client.createOrReplace(sanityProduct);
        console.log(`✅ Imported product: ${sanityProduct.name}`);
      }

      console.log('✅ Data import completed!');
    } catch (error) {
      console.error('❌ Error importing data:', error);
    }
}

importData();
```

2- For deleting data:

```
import { createClient } from '@sanity/client';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31',
});

async function deleteAllProducts() {
  try {
    console.log('Fetching all products from Sanity...');
    const products = await client.fetch('*[_type == "product"]{_id}');
    console.log(`Found ${products.length} products to delete.`);

    for (const product of products) {
      console.log(`Deleting product: ${product._id}`);
      await client.delete(product._id);
      console.log(`✅ Deleted product: ${product._id}`);
    }

    console.log('All products deleted successfully!');
  } catch (error) {
    console.error('Error deleting products:', error);
  }
}

deleteAllProducts();
```