Hackathon Day 3:

API Integration and Data Migration

Overview

On Day 3 of the Hackathon, I accomplished key technical milestones to advance my General E-Commerce Marketplace:

1. Designed and implemented schemas for **Products** in Sanity CMS.
2. Imported API data into Sanity CMS after appropriate transformations.
3. Integrated Sanity CMS with my Next.js application to fetch the data.
4. Displayed the data dynamically on my website for a seamless user experience.

Here is a detailed explanation of each step.

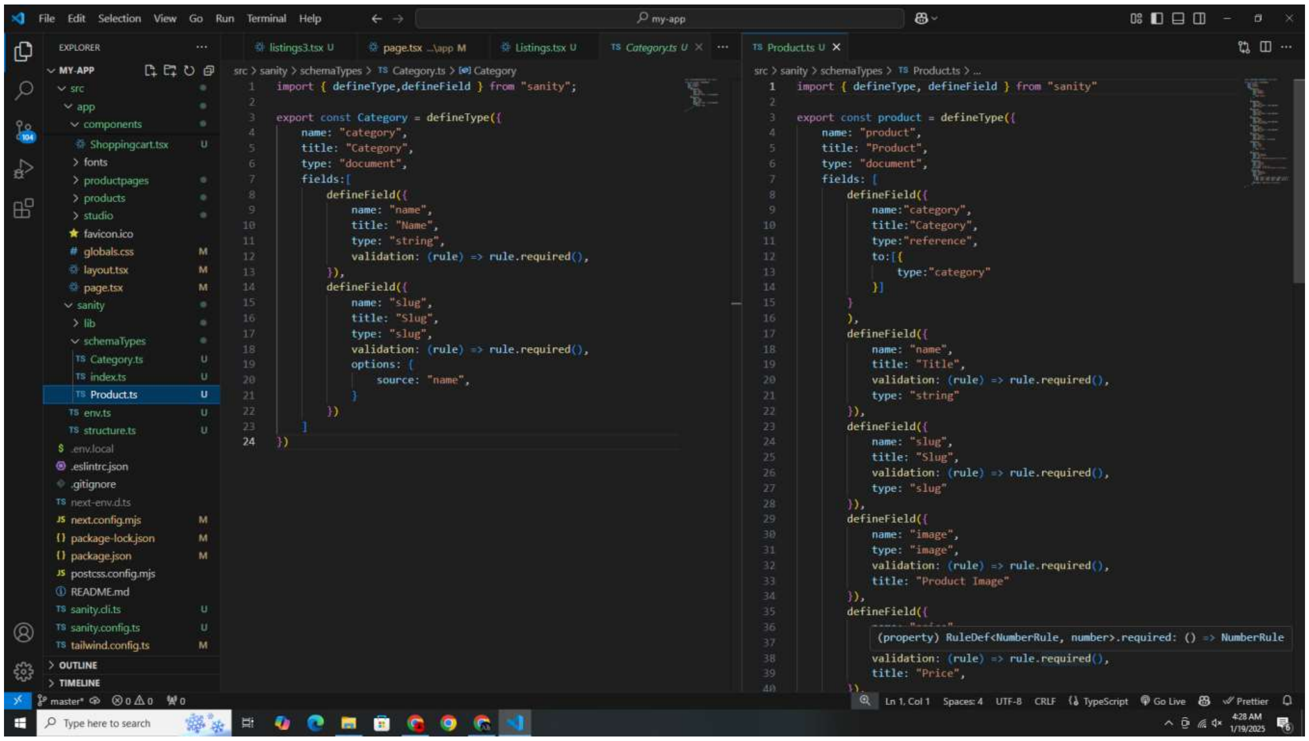
Step 1: Designing Sanity Schemas for Products

To efficiently manage product data, I created a schema in **Sanity CMS** tailored to the API structure. This schema organizes data for easy retrieval and updates.

Schema Details

The schema includes the following fields:

- **Product Name (name):** The name of the product.
- **Price (price):** The cost per unit.
- **Stock (stock):** The number of items available.
- **Category (category):** The classification of the product (e.g., Men’s Wear, Electronics).
- **Description (description):** A concise description of the product.
- **Image URL (image):** A reference to the product image.



```ts
import { defineType,defineField } from "sanity";

export const Category = defineType({
    name: "category",
    title: "Category",
    type: "document",
    fields:[
        defineField({
            name: "name",
            title: "Name",
            type: "string",
            validation: (rule) => rule.required(),
        }),
        defineField({
            name: "slug",
            title: "Slug",
            type: "slug",
            validation: (rule) => rule.required(),
            options: {
                source: "name",
            }
        })
    ]
})
```

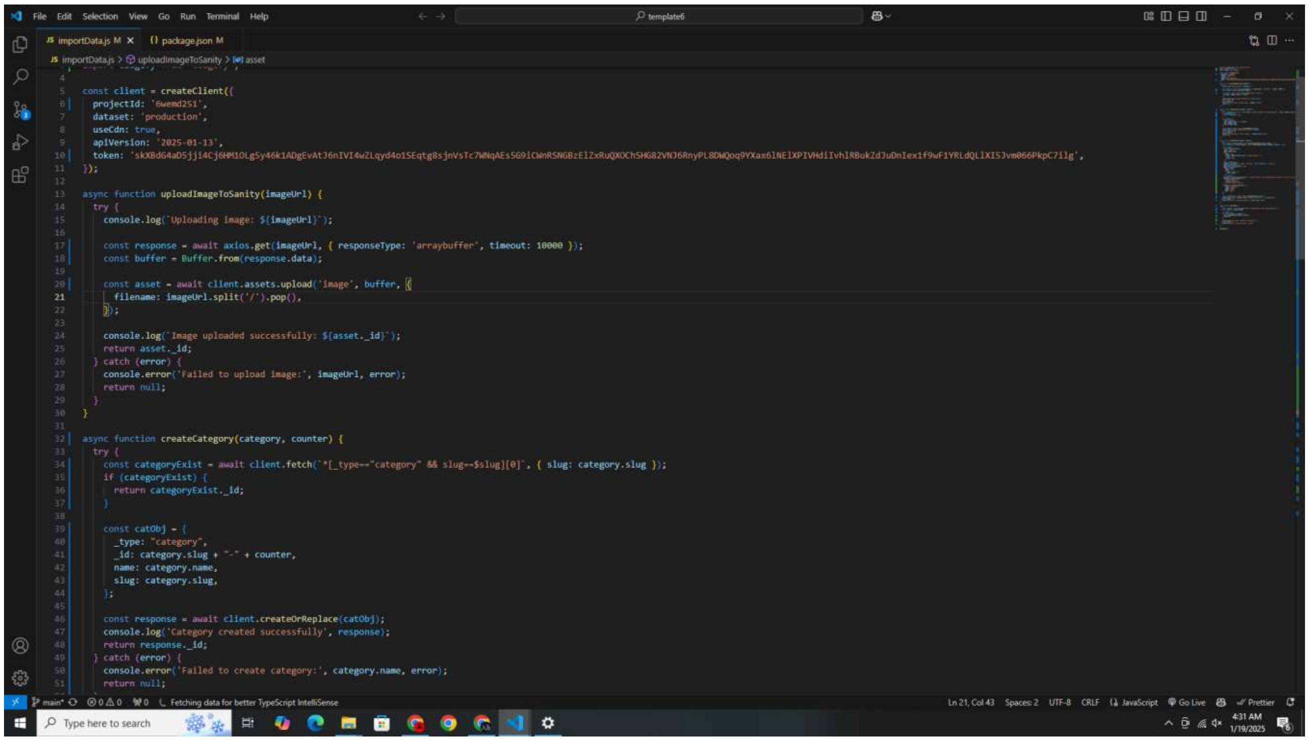
```ts
import { defineType, defineField } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        defineField({
            name:"category",
            title:"Category",
            type:"reference",
            to:[{
                type:"category"
            }]
        }
        ),
        defineField({
            name: "name",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        }),
        defineField({
            name: "slug",
            title: "Slug",
            validation: (rule) => rule.required(),
            type: "slug"
        }),
        defineField({
            name: "image",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        }),
        defineField({
            (property) RuleDef<NumberRule, number>.required: () => NumberRule
            validation: (rule) => rule.required(),
            title: "Price",
```

Step 2: Importing API Data into Sanity CMS

The next step involved transferring API data into Sanity CMS, ensuring the data matched the schema's structure.

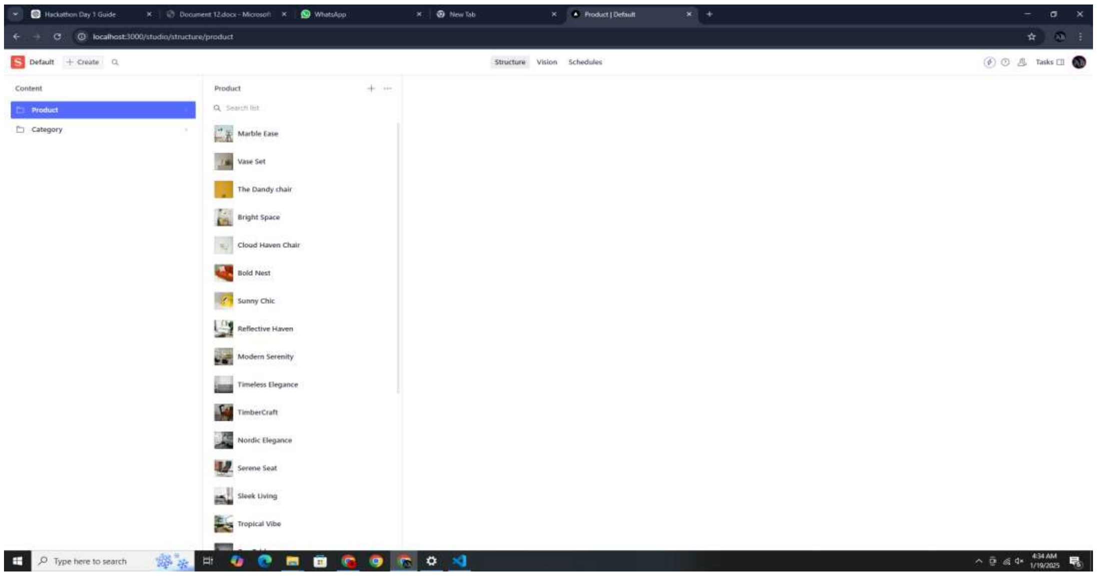
Migration Process

1. **Retrieve Data from API:**
   a. Queried the API to fetch product data in JSON format.
   b. Example API Endpoint: https://api.example.com/products.
2. **Map API Fields to Schema:**
   a. Transformed API field names to align with the schema in Sanity CMS.
   b. Example: api_product_name was mapped to name in the schema.
3. **Upload Data to Sanity CMS:**
   a. Used a custom Node.js script for automation.
   b. Migration script example:



```
const client = createClient({
  projectId: '6wemd251',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token: 'skXBdG4aD5jji4Cj6HM1OLg5y46k1ADgEvAtJ6nIVI4wZLqyd4o1SEqtg8sjnVsTc7WNqAEs5G9iCWnRSNGBzElZxRuQXOChSHG82VNJ6RnyPL8DWQoq9VXax6lNElXPIVHdiIvhlRBukZdJuDnIex1f9wF1YRLdQLlXI5Jvm066PkpC7ilg',
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await axios.get(imageUrl, { responseType: 'arraybuffer', timeout: 10000 });
    const buffer = Buffer.from(response.data);

    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

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);
    console.log('Category created successfully', response);
    return response._id;
  } catch (error) {
    console.error('Failed to create category:', category.name, error);
    return null;
```

- **Verify Data:**
- Checked the Sanity CMS dashboard to confirm data accuracy and completeness

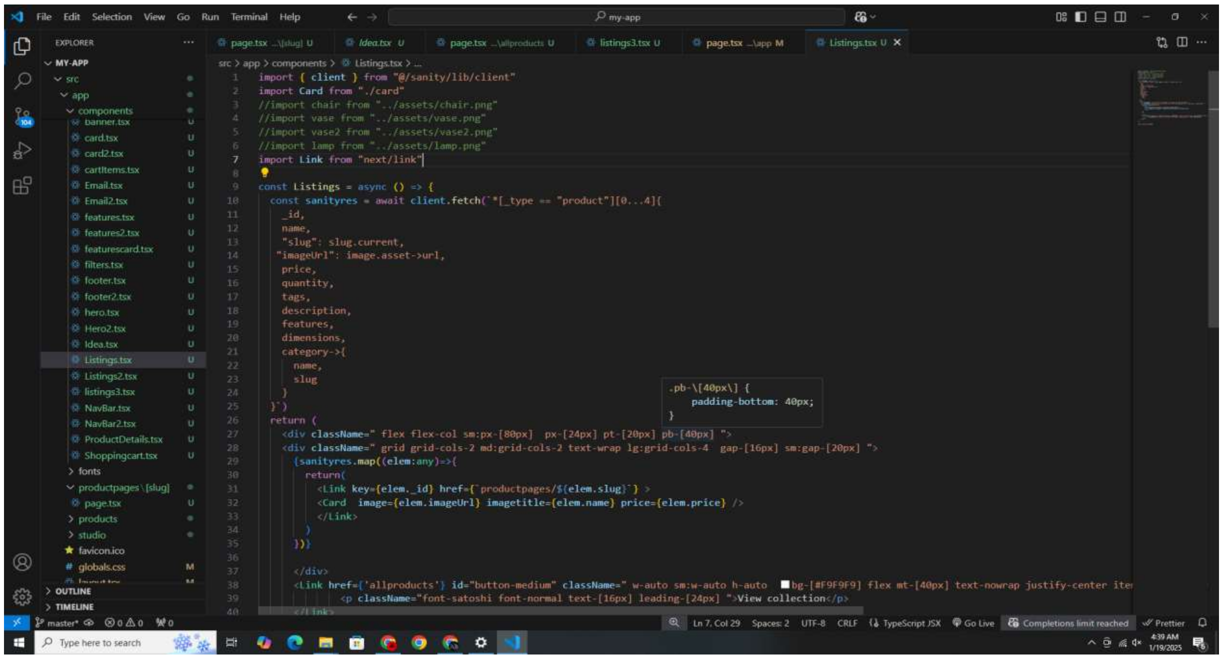


Marble Ease
Vase Set
The Dandy chair
Bright Space
Cloud Haven Chair
Bold Nest
Sunny Chic
Reflective Haven
Modern Serenity
Timeless Elegance
TimberCraft
Nordic Elegance
Serene Seat
Sleek Living
Tropical Vibe

Step 3: Fetching Data in Next.js

With the data available in Sanity CMS, I integrated it into my Next.js application to fetch and display the information dynamically.

Steps to Fetch Data



```
import { client } from "@/sanity/lib/client"
import Card from "./card"
//import chair from "../assets/chair.png"
//import vase from "../assets/vase.png"
//import vase2 from "../assets/vase2.png"
//import lamp from "../assets/lamp.png"
import Link from "next/link"

const Listings = async () => {
  const sanityres = await client.fetch(`*[_type == "product"][0...4]{
    _id,
    name,
    "slug": slug.current,
   "imageUrl": image.asset->url,
    price,
    quantity,
    tags,
    description,
    features,
    dimensions,
    category->{
      name,
      slug
    }
  }`)
  return (
    <div className=" flex flex-col sm:px-[80px]  px-[24px] pt-[20px] pb-[40px] ">
    <div className=" grid grid-cols-2 md:grid-cols-2 text-wrap lg:grid-cols-4  gap-[16px] sm:gap-[20px] ">
      {sanityres.map((elem:any)=>{
        return(
          <Link key={elem._id} href={`productpages/${elem.slug}`} >
          <Card  image={elem.imageUrl} imagetitle={elem.name} price={elem.price} />
          </Link>
        )
      })}

      </div>
      <Link href={'allproducts'} id="button-medium" className=" w-auto sm:w-auto h-auto  bg-[#F9F9F9] flex mt-[40px] text-nowrap justify-center ite
              <p className="font-satoshi font-normal text-[16px] leading-[24px] ">View collection</p>
```

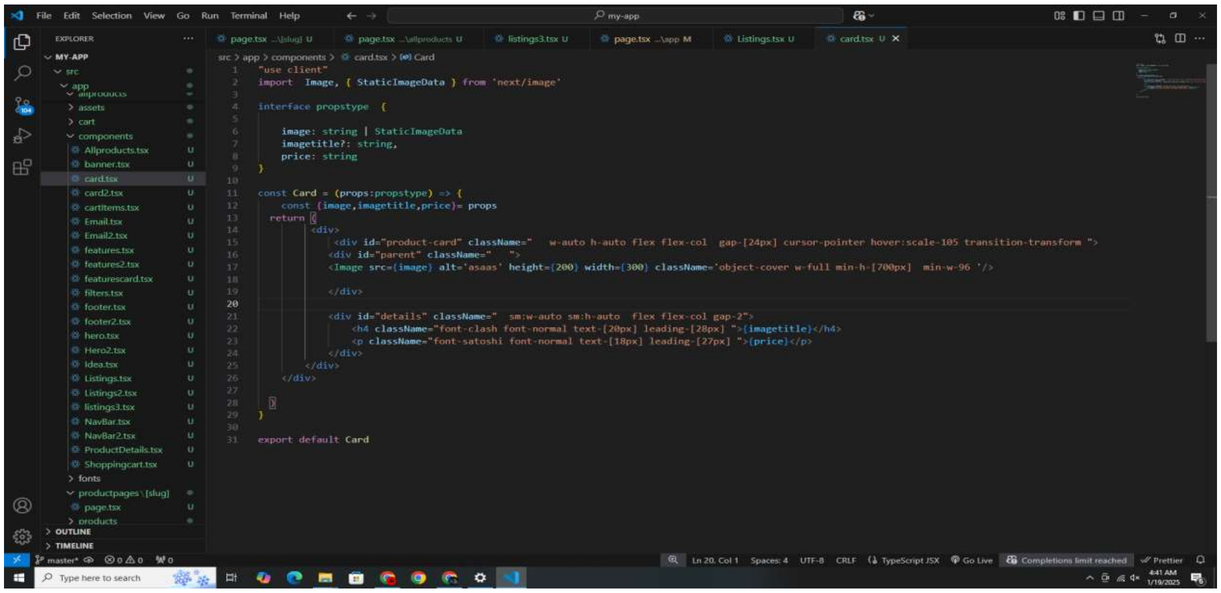
```
.pb-\[40px\] {
    padding-bottom: 40px;
}
```

Step 4: Rendering Data Dynamically on the Website

After fetching the data, I displayed it dynamically on the website. This ensures that any updates made in Sanity CMS reflect immediately on the frontend.

Display Features

1. **Product Grid:**
   a. Rendered all products in a responsive grid layout.
   b. Included details like name, price, and image.
2. **Product Detail Page:**
   a. Showed more comprehensive details for individual products.
3. **Live Updates:**
   a. The website reflected changes in Sanity CMS without requiring manual updates.



```
"use client"
import  Image, { StaticImageData } from 'next/image'

interface propstype  {

    image: string | StaticImageData
    imagetitle?: string,
    price: string
}

const Card = (props:propstype) => {
    const {image,imagetitle,price}= props
  return (
        <div>
            <div id="product-card" className="   w-auto h-auto flex flex-col  gap-[24px] cursor-pointer hover:scale-105 transition-transform ">
           <div id="parent" className="    ">
           <Image src={image} alt='asaas' height={200} width={300} className='object-cover w-full min-h-[700px]  min-w-96 '/>

           </div>

           <div id="details" className="  sm:w-auto sm:h-auto  flex flex-col gap-2">
               <h4 className="font-clash font-normal text-[20px] leading-[28px] ">{imagetitle}</h4>
               <p className="font-satoshi font-normal text-[18px] leading-[27px] ">{price}</p>
           </div>
        </div>
    </div>

  )
}

export default Card
```

Conclusion

By completing the tasks for Day 3, I:

- Designed an optimized schema for product management in Sanity CMS.
- Imported API data into the CMS seamlessly.
- Integrated and displayed this data dynamically using Next.js.