HACKATHON 3 DAY 3

API Integration and Data Migration

Overview

On Day 3 of the Hackathon, I successfully completed the following tasks to advance my General E-Commerce Marketplace

1. Created schemas for **Products** in Sanity CMS.
2. Migrated API data into the Sanity CMS.
3. Fetched data from Sanity CMS into my Next.js code.
4. Dynamically displayed the data on my website.

Below is a detailed breakdown of each step.

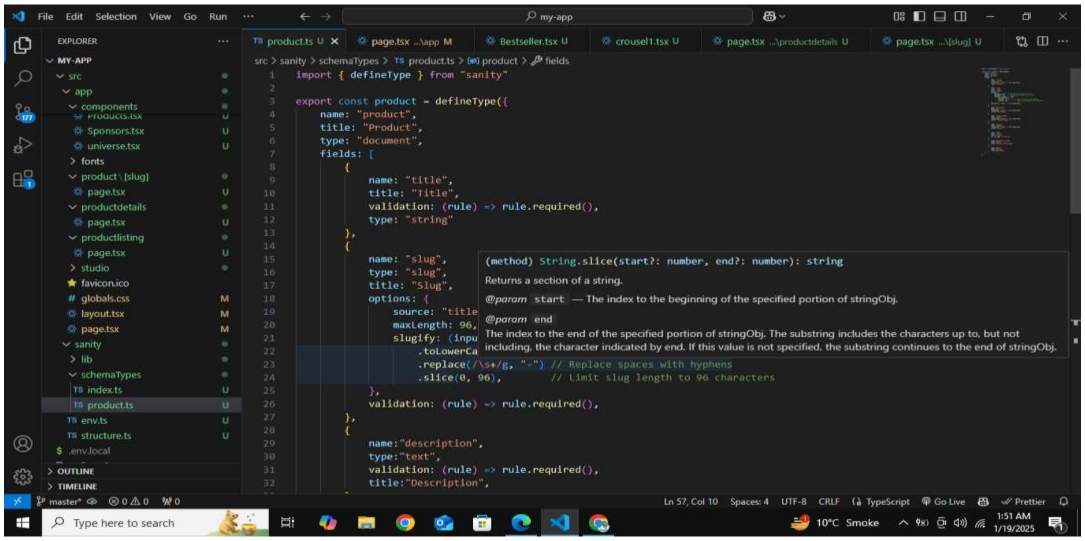
Step 1: Creating Sanity Schemas for Products

To store and manage product data effectively, I designed and implemented a schema in **Sanity CMS**. This schema allows for structured data storage and ensures compatibility with the data retrieved from the API.

Schema Details

The schema includes the following fields:

- **Product Name (name):** The title of the product.
- **Price (price):** The cost of the product.
- **Stock (stock):** The available quantity of the product.
- **Category (category):** The category to which the product belongs (e.g., Men's Wear, Electronics).
- **Description (description):** A brief description of the product.
- **Image URL (image):** A link to the product's image.



```
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name: "slug",
            type: "slug",
            title: "Slug",
            options: {
                source: "title
                maxLength: 96,
                slugify: (inpu
                    .toLowerCa
                    .replace(/\s+/g, "-") // Replace spaces with hyphens
                    .slice(0, 96),        // Limit slug length to 96 characters
            },
            validation: (rule) => rule.required(),
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
```

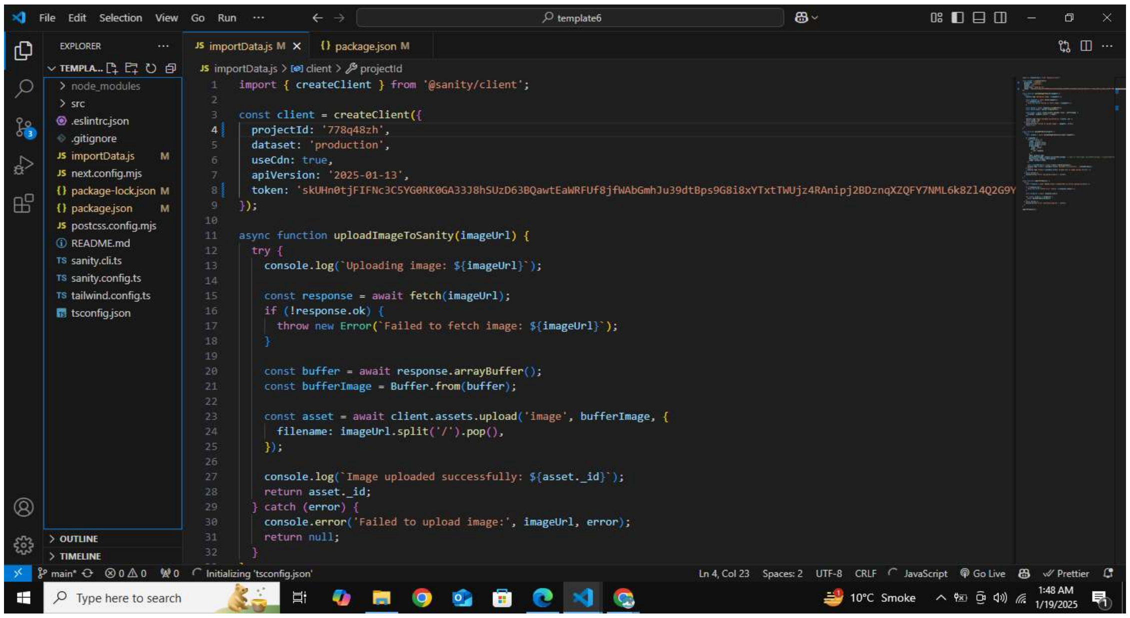
```
(method) String.slice(start?: number, end?: number): string
Returns a section of a string.
@param start — The index to the beginning of the specified portion of stringObj.
@param end
The index to the end of the specified portion of stringObj. The substring includes the characters up to, but not including, the character indicated by end. If this value is not specified, the substring continues to the end of stringObj.
```

Step 2: Migrating API Data into Sanity CMS

To populate the Sanity CMS with product data, I migrated the data retrieved from the API into the CMS. This step involved fetching data from the API and transforming it to match the schema created in Step 1.

Migration Process

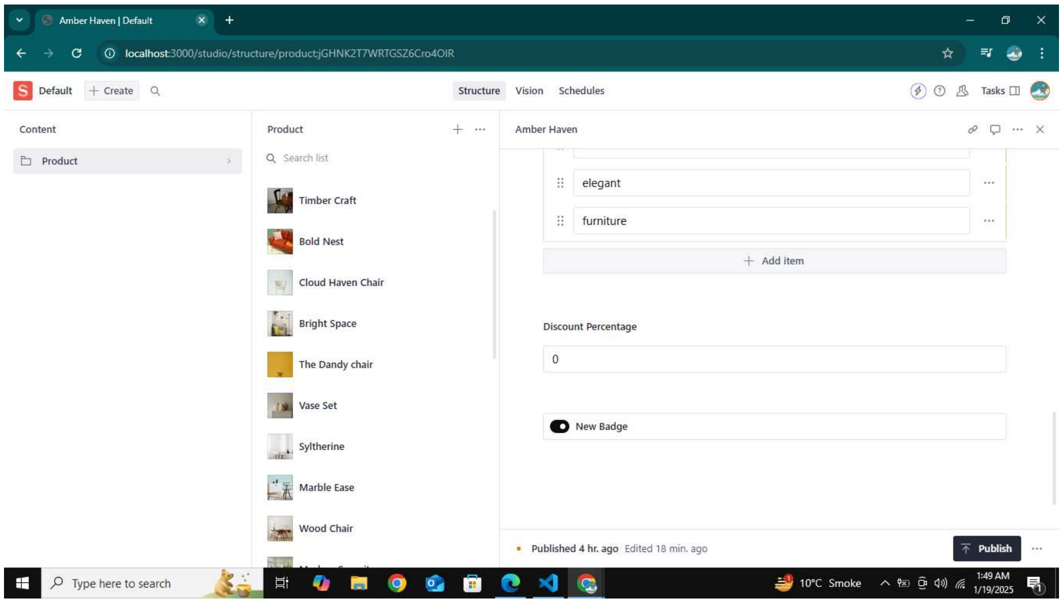
1. **Fetch Data from API:**
    a. Used the API endpoint to retrieve product data in JSON format.
    b. Example API Endpoint: `https://api.example.com/products`.



```
import { createClient } from '@sanity/client';

const client = createClient({
  projectId: '778q48zh',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token: 'skUHn0tjFIFNc3C5YG0RK0GA33J8hSUzD63BQawtEaWRFUf8jfWAbGmhJu39dtBps9G8i8xYTxtTWUjz4RAnipj2BDznqXZQFY7NML6k8Zl4Q2G9Y
});

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

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      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;
  }
```

2. **Transform Data:**
    a. Mapped the API fields to match the schema fields in Sanity CMS.
    b. For instance, `api_product_name` was mapped to `name` in the schema.
3. **Upload Data to Sanity:**
    a. Used the Sanity CLI and custom scripts to upload the data.
4. **Validation:**

Migration CMS screenshot for validation:



Product

Timber Craft
Bold Nest
Cloud Haven Chair
Bright Space
The Dandy chair
Vase Set
Syltherine
Marble Ease
Wood Chair

Amber Haven

elegant
furniture

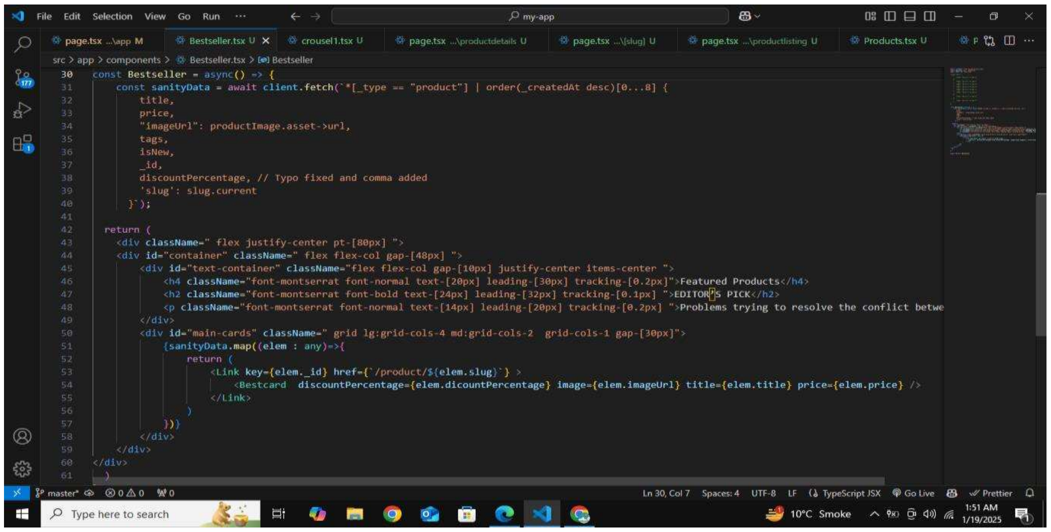
Discount Percentage
0

New Badge

Published 4 hr. ago Edited 18 min. ago Publish

Step 3: Fetching Data in Next.js

After migrating the data to Sanity CMS, I fetched it into my Next.js application to display it dynamically on the website.



```
const Bestseller = async() => {
    const sanityData = await client.fetch(`*[_type == "product"] | order(_createdAt desc)[0...8] {
        title,
        price,
        "imageUrl": productImage.asset->url,
        tags,
        isNew,
        _id,
        discountPercentage, // Typo fixed and comma added
        'slug': slug.current
      }`);

  return (
    <div className=" flex justify-center pt-[80px] ">
    <div id="container" className=" flex flex-col gap-[48px] ">
        <div id="text-container" className="flex flex-col gap-[10px] justify-center items-center ">
            <h4 className="font-montserrat font-normal text-[20px] leading-[30px] tracking-[0.2px]">Featured Products</h4>
            <h2 className="font-montserrat font-bold text-[24px] leading-[32px] tracking-[0.1px] ">EDITOR'S PICK</h2>
            <p className="font-montserrat font-normal text-[14px] leading-[20px] tracking-[0.2px] ">Problems trying to resolve the conflict betwe
        </div>
        <div id="main-cards" className=" grid lg:grid-cols-4 md:grid-cols-2  grid-cols-1 gap-[30px]">
            {sanityData.map((elem : any)=>{
                return (
                    <Link key={elem._id} href={`/product/${elem.slug}`} >
                        <Bestcard  discountPercentage={elem.dicountPercentage} image={elem.imageUrl} title={elem.title} price={elem.price} />
                    </Link>
                )
            })}
        </div>
    </div>
</div>
)
```

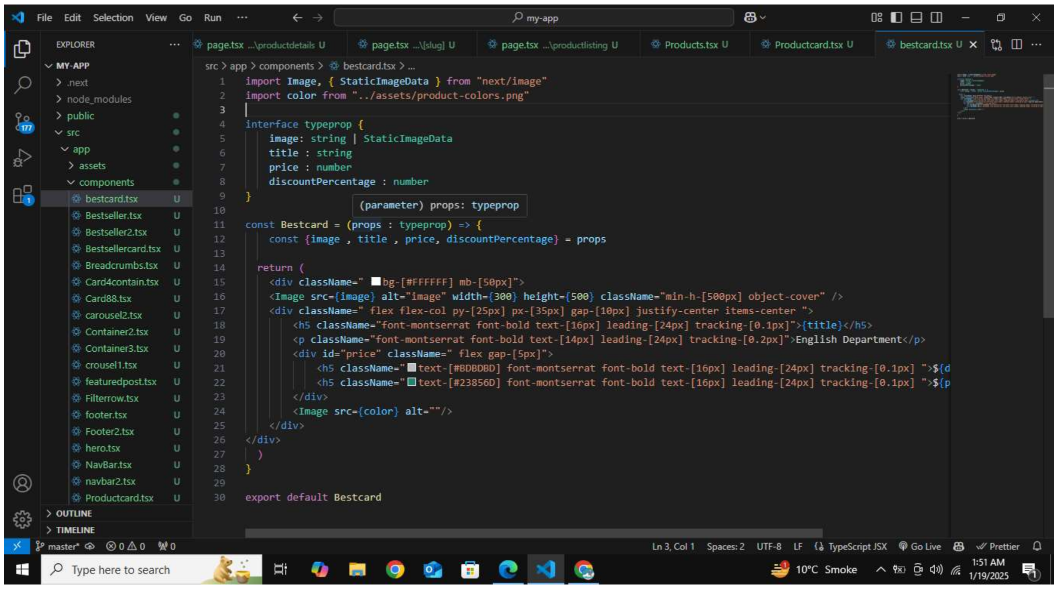
Step 4: Dynamically Displaying Data on the Website

With the data successfully fetched, I dynamically displayed it on the website. This ensures that product information updates automatically when changes are made in Sanity CMS.

Display Features

1. **Product Listing Page:**
    a. Displayed all products in a grid format.
    b. Included product name, price, and image.
2. **Product Details Page:**
    a. Displayed detailed information about a selected product.

3. **Dynamic Updates:**
   a. Any updates in Sanity CMS automatically reflected on the website due to live fetching.



```tsx
import Image, { StaticImageData } from "next/image"
import color from "../assets/product-colors.png"

interface typeprop {
    image: string | StaticImageData
    title : string
    price : number
    discountPercentage : number
}

const Bestcard = (props : typeprop) => {
    const {image , title , price, discountPercentage} = props

  return (
    <div className=" bg-[#FFFFFF] mb-[50px]">
    <Image src={image} alt="image" width={300} height={500} className="min-h-[500px] object-cover" />
    <div className=" flex flex-col py-[25px] px-[35px] gap-[10px] justify-center items-center ">
        <h5 className="font-montserrat font-bold text-[16px] leading-[24px] tracking-[0.1px]">{title}</h5>
        <p className="font-montserrat font-bold text-[14px] leading-[24px] tracking-[0.2px]">English Department</p>
        <div id="price" className=" flex gap-[5px]">
            <h5 className=" text-[#BDBDBD] font-montserrat font-bold text-[16px] leading-[24px] tracking-[0.1px] ">${d
            <h5 className=" text-[#23856D] font-montserrat font-bold text-[16px] leading-[24px] tracking-[0.1px] ">${p
        </div>
        <Image src={color} alt=""/>
    </div>
</div>
  )
}

export default Bestcard
```

Conclusion

By completing the tasks for Day 3, I:

- Created a robust schema in Sanity CMS for product data.
- Successfully migrated API data into the CMS.
- Dynamically fetched and displayed this data in my Next.js application.