Day 3 - API Integration Report - Bandage

Brief Overview:

This document outlines the process of mastering API integration and data migration through the development of a functional backend for a marketplace. The primary goal of this task was to transfer data from a given API into Sanity CMS and seamlessly integrate it with a Next.js-based frontend. This approach simulates real-world development workflows, enabling us to better prepare for meeting diverse client needs.

API Integration Workflow:

The provided API for Template 5 is same as Template 6 which served as the basis for this integration. The API endpoint was https://template6-six.vercel.app/api/products. It delivered comprehensive product information, including titles, images, pricing details, tags and descriptions. This data was carefully migrated into Sanity CMS and later retrieved to create a dynamic display on the frontend.

For environment configuration, sensitive details were stored securely in the .env.local file, protecting critical information from exposure. Specific environment variables were defined to facilitate seamless integration.

The Sanity client was set up by incorporating project-specific identifiers such as the project ID and dataset into the Next.js application. Secure handling of sensitive details was ensured by utilizing environment variables throughout the process.

To fetch product details from Sanity CMS, GROQ queries were employed. These queries targeted key fields like id, titles, images, prices, discounts, and descriptions. Once retrieved, the data underwent a transformation process to align with the frontend's structural requirements.

Adjustments made to schemas:

The schema in Sanity CMS was carefully adjusted to align with the API's data fields, ensuring seamless data transfer and compatibility. This schema included fields for titles, prices, original prices, discount percentages, tags, and product descriptions, ensuring consistency between the API and CMS data.

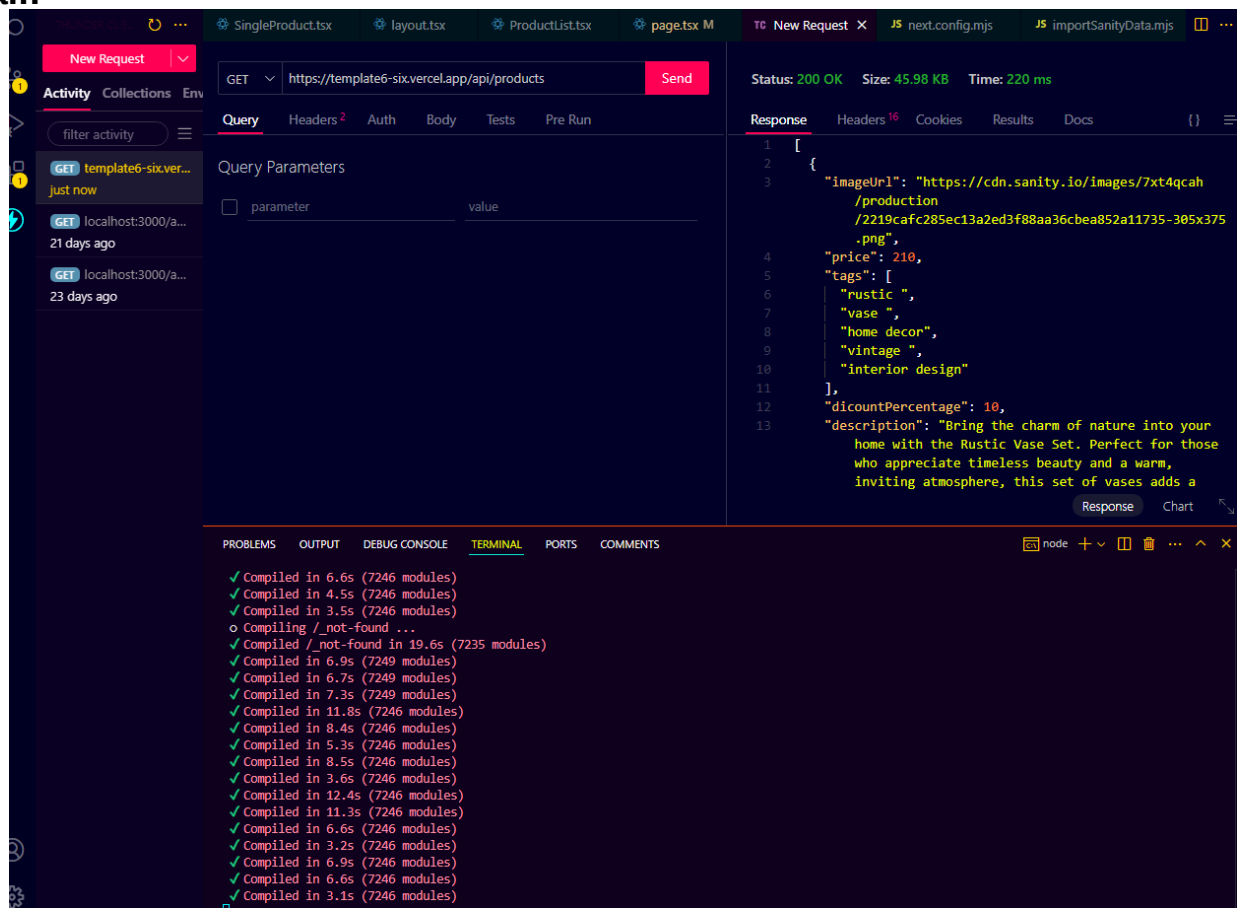
```
sanity > schemaTypes > TS product.ts > [e] product > fields > name
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
            title:"New Badge",
        }
    ]
})
```

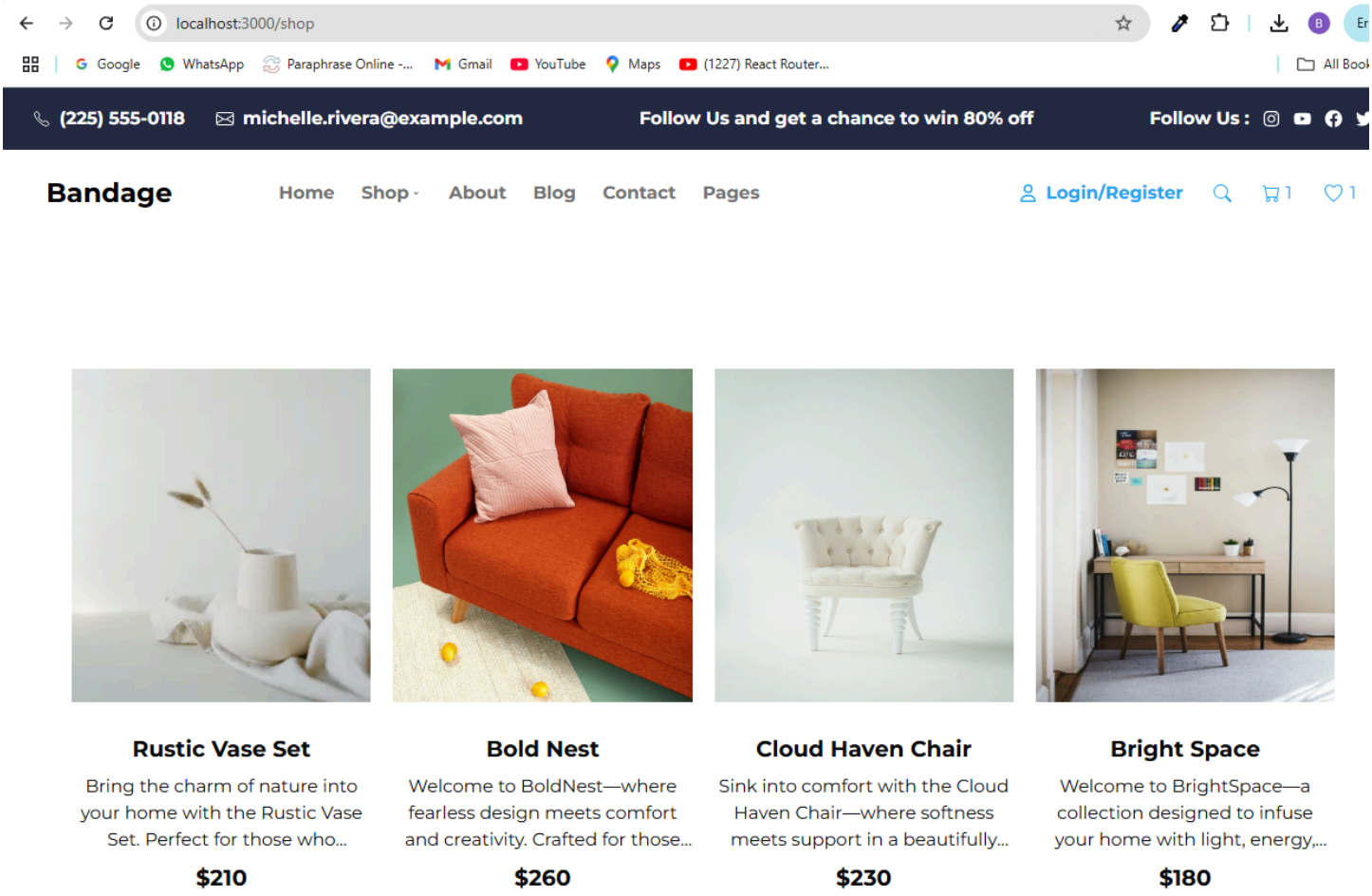
Migration Techniques and Tools:

To automate the migration of data from the API to Sanity CMS, a custom script was developed. This script fetched data programmatically and ensured its accuracy by implementing validation mechanisms during the migration process. The frontend was integrated with the migrated data using dynamic rendering techniques in Next.js.

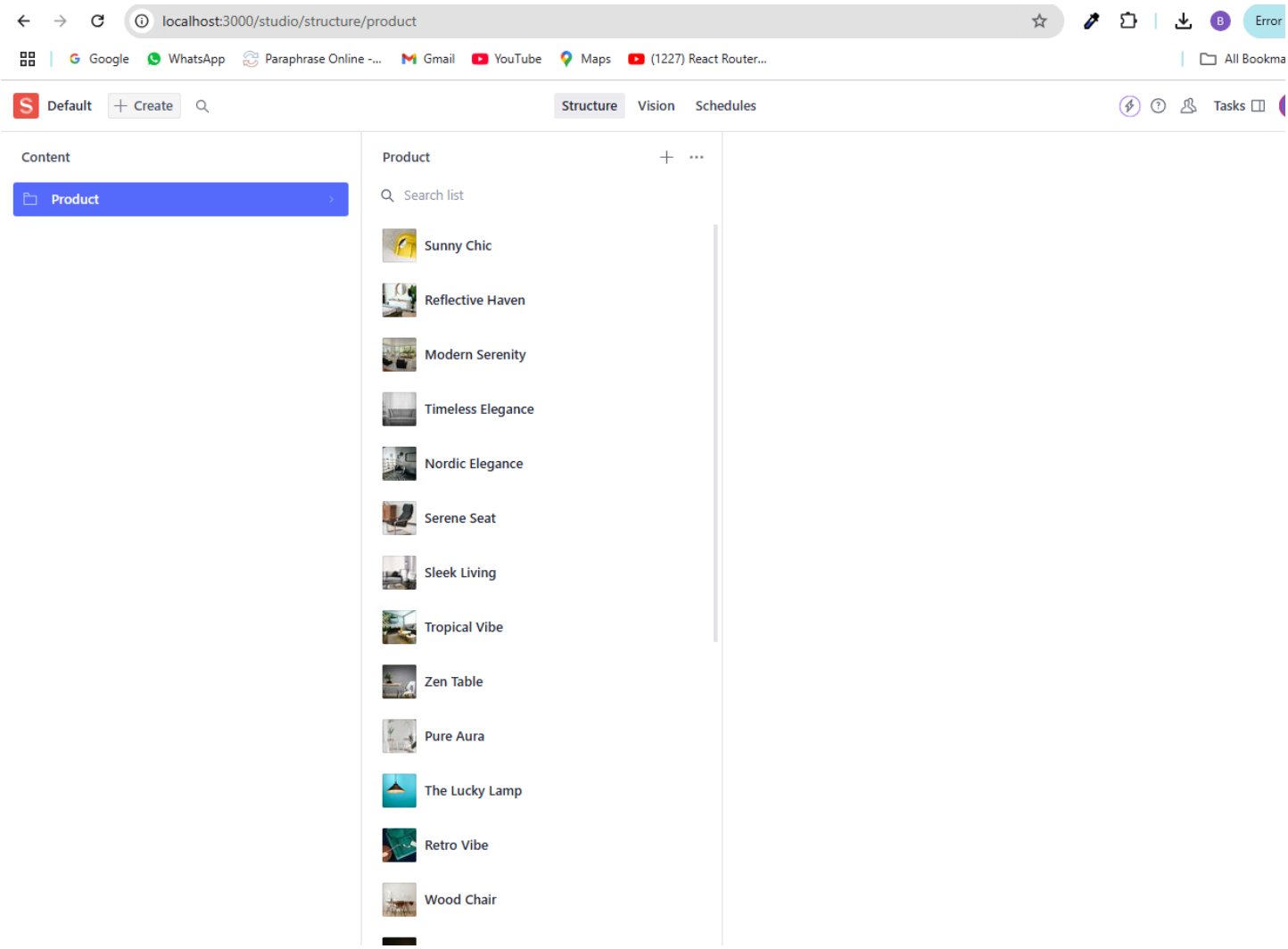
Screenshots:
API Call:



Data on Front-End:



Populated Sanity Fields:



Code Snippets:
API Integration:

```
async function importProducts() {
  try {
    const response = await fetch('https://template6-six.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

```
SingleProduct.tsx | layout.tsx | ProductList.tsx | page.tsx M | New Request | next.config.mjs | importSanityData.mjs
app > shop > page.tsx > page

import { Montserrat } from "next/font/google";
import { urlFor } from '@/sanity/lib/image';
import Header from '@/components/Header';
import Footer from '@/components/Footer';

const Montserratfont = Montserrat({
  weight: ['400', '500', '600', '700'],
  style: "normal",
  subsets: ["latin"]
});

export default async function page() {
  const query = `
    *[_type == "product"]{
      _id,
      title,
      description,
      productImage,
      price,
      tags,
      discountPercentage,
      isNew
    }
  `;

  const products = await client.fetch(query);

  return (
    <>
    <Header/>
      <div className={`${Montserratfont.className} container w-full max-w-[2000px]`}>
        <div className="w-[90%] mx-auto grid grid-cols-1 md:grid-cols-4 gap-x-5 gap-y-12 my-20">
          {products.map((product: any) => (
            <div
              key={product._id}
              className="flex flex-col justify-center gap-y-2 h-[550px] cursor-pointer"
            >
              <div className="h-[300px] w-full">
                {product.productImage ? (
                  <Image
                    src={urlFor(product.productImage).url()}
                    alt={product.title || 'Product image'}
                    width={4000}
                    height={4000}
```

Migration Script:

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

const client = createClient({
  projectId: 'dtt2k3eb',
  dataset: 'production',
  useCdn: false,
  apiVersion: '2025-01-13',
  token: 'skDzkpMY6gDtfiRJwJSwAtbtXulmcrdGsaiPW82qwaA0usXUAVoyigXJn62GSbQgMIPPWWlnQnZV6OpuORD4tQsJBAFx5YJQi3QPfv
});

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;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
```

Self-Validation Checklist:

Task	Progress
Understanding the API structure and endpoints	Done
Validating and adjusting the schema in Sanity CMS	Done
Automating the migration process using a script	Done
Integrating API data into the Next.js frontend	Done
Conducting error handling and debugging processes	Done

Conclusion:

The API integration and data migration tasks for this project provided valuable insights into handling real-world scenarios involving external data sources and CMS platforms. By securely storing sensitive information, designing a schema that mirrors API data, and implementing dynamic rendering techniques, we successfully created a seamless flow from backend to frontend. This project reinforced the importance of meticulous planning, robust error handling, and attention to detail in ensuring a smooth user experience. The hands-on experience gained from this exercise has enhanced our readiness to tackle similar challenges in future projects.

This report reflects the streamlined workflow of integrating an external API with Sanity CMS and successfully rendering the data on a frontend application built with Next.js. The hands-on experience gained through this task reinforces our readiness to tackle real-world projects with similar requirements.