Day 3 - API Integration & Data Migration
Furniro E-Commerce

Overview:

This report documents the API integration and data management setup for the Furniro website using Sanity CMS. The goal is to design a seamless system to retrieve product data from the backend and display it accurately on the frontend. The report includes the integration process, schema customization, migration steps, and screenshots.

API Integration Process:

1: Data Source

Data was fetched from the given API endpoint: https://template6-six.vercel.app/api/products

The product data included the following fields: Title, Description, Price, Image URL, Discount Percentage, Tags, is New

2: Integration Steps

- **Schema Design:** A custom product schema was created in Sanity CMS to align with the structure of the imported data.

- **Import Script:** A migration script was written to fetch data from the API and upload it to Sanity CMS.

- **Image Handling:** The script was enhanced to handle images as arrays and generate unique references.

- **Data Validation:** Validation rules for fields like slug, price, and reviews were implemented.

- **API Endpoints:** Endpoints were created to retrieve data from Sanity CMS for use in the frontend.

3: Schema Adjustments

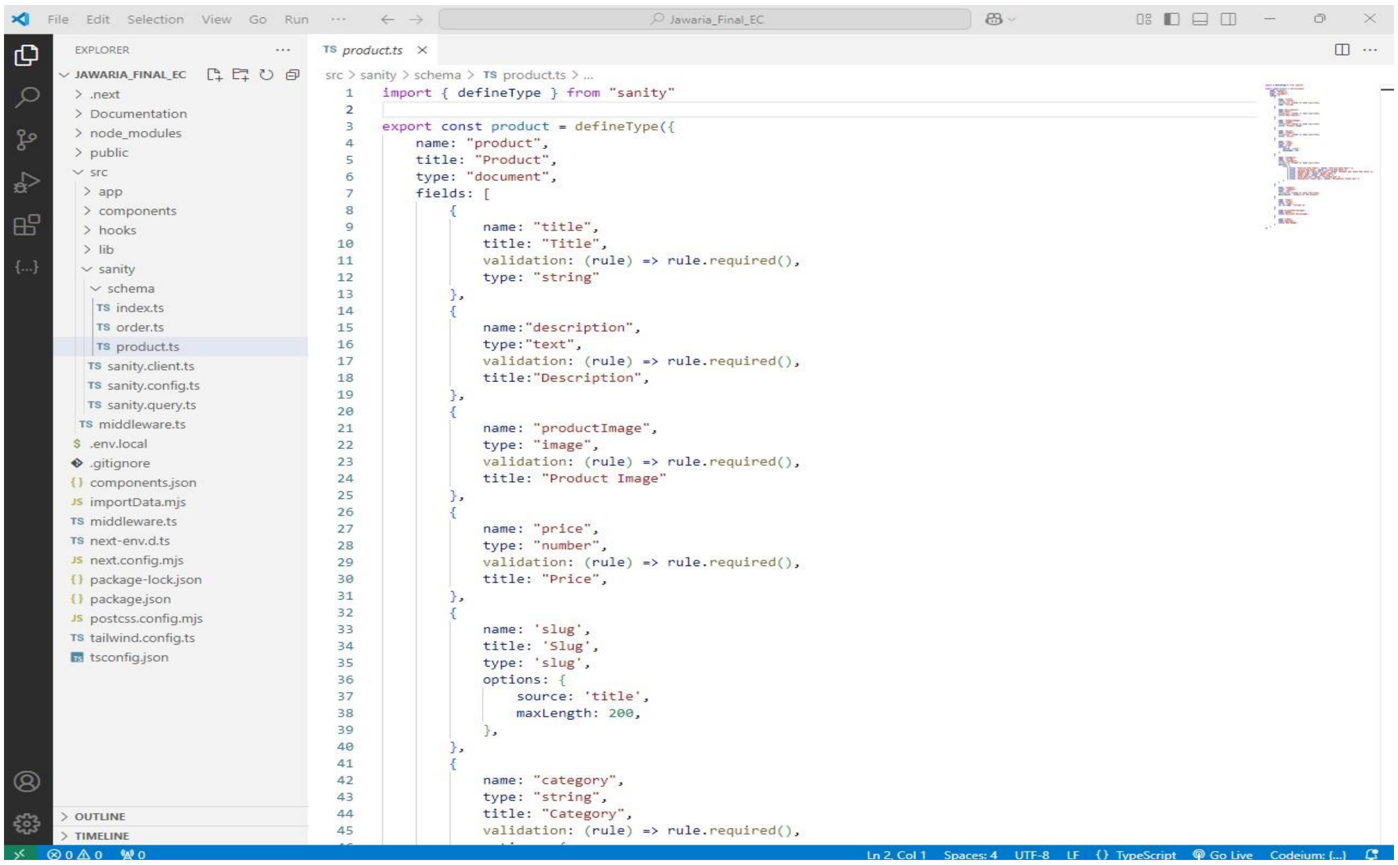
Product Schema Custom fields were added**:**

Category

Slug

Summary

Key validation rules and slug uniqueness checks were implemented.



```
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: 'slug',
            title: 'Slug',
            type: 'slug',
            options: {
                source: 'title',
                maxLength: 200,
            },
        },
        {
            name: "category",
            type: "string",
            title: "Category",
            validation: (rule) => rule.required(),
```

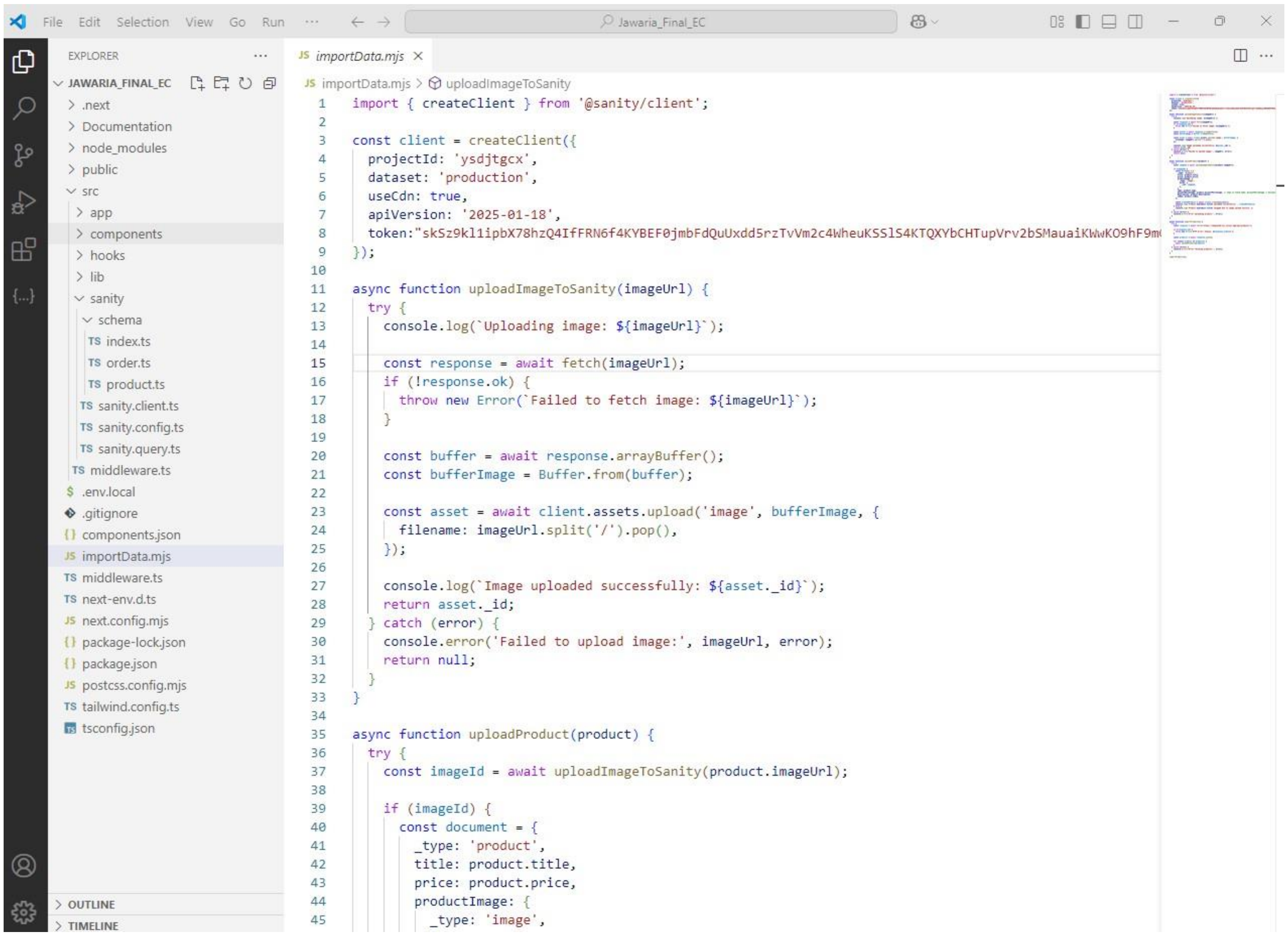
Migration Steps:

1. Tools Used

- Sanity Client: For uploading data.
- Fetch: For API calls.
- .env: For managing environment variables.

2. Migration Script

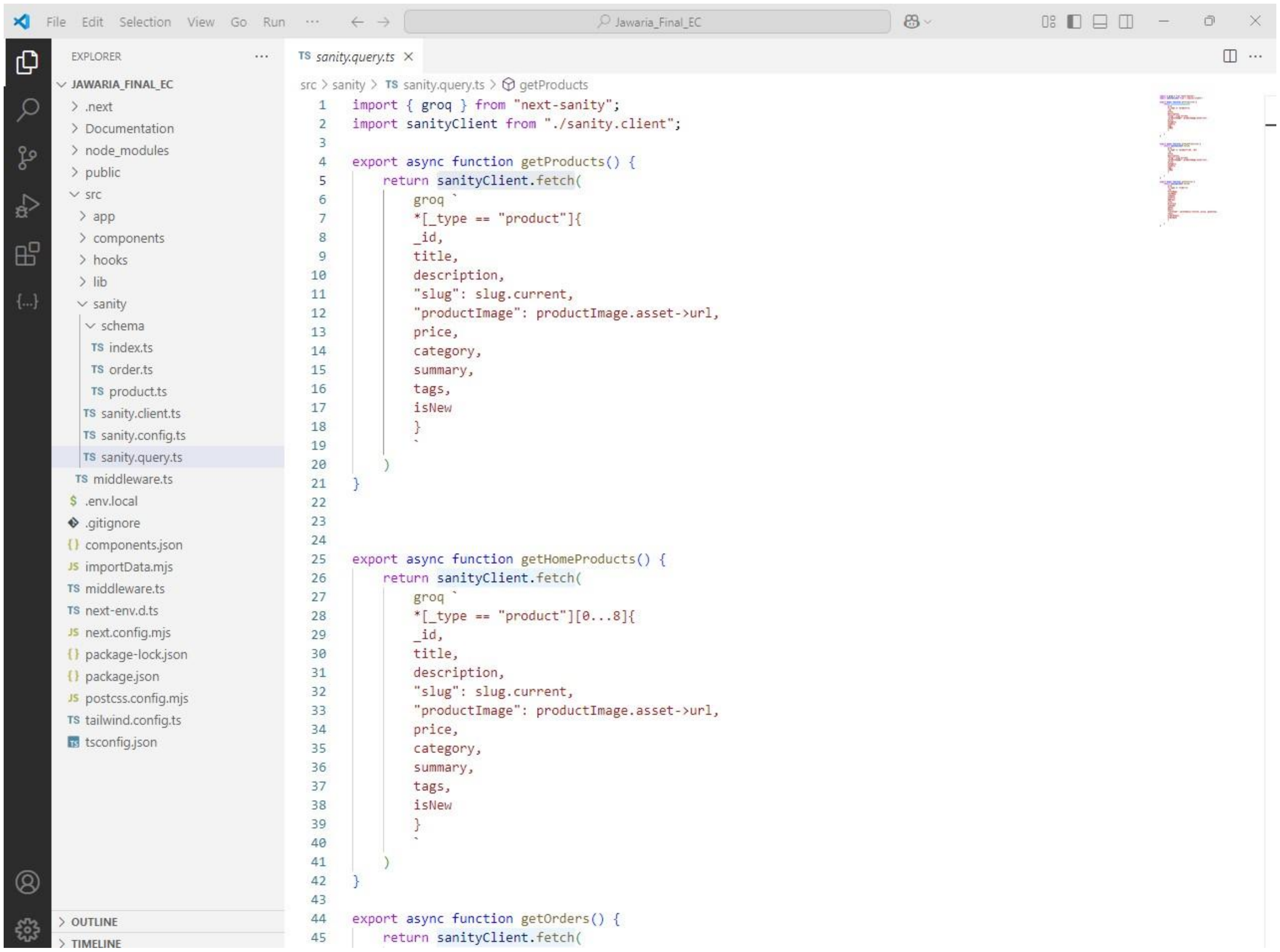
The script fetches API data and imports it into Sanity CMS while handling images.



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

const client = createClient({
  projectId: 'ysdjtgcx',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-18',
  token:"skSz9kl1ipbX78hzQ4IfFRN6f4KYBEF0jmbFdQuUxdd5rzTvVm2c4WheuKSSlS4KTQXYbCHTupVrv2bSMauaiKWwKO9hF9m
});

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',
```

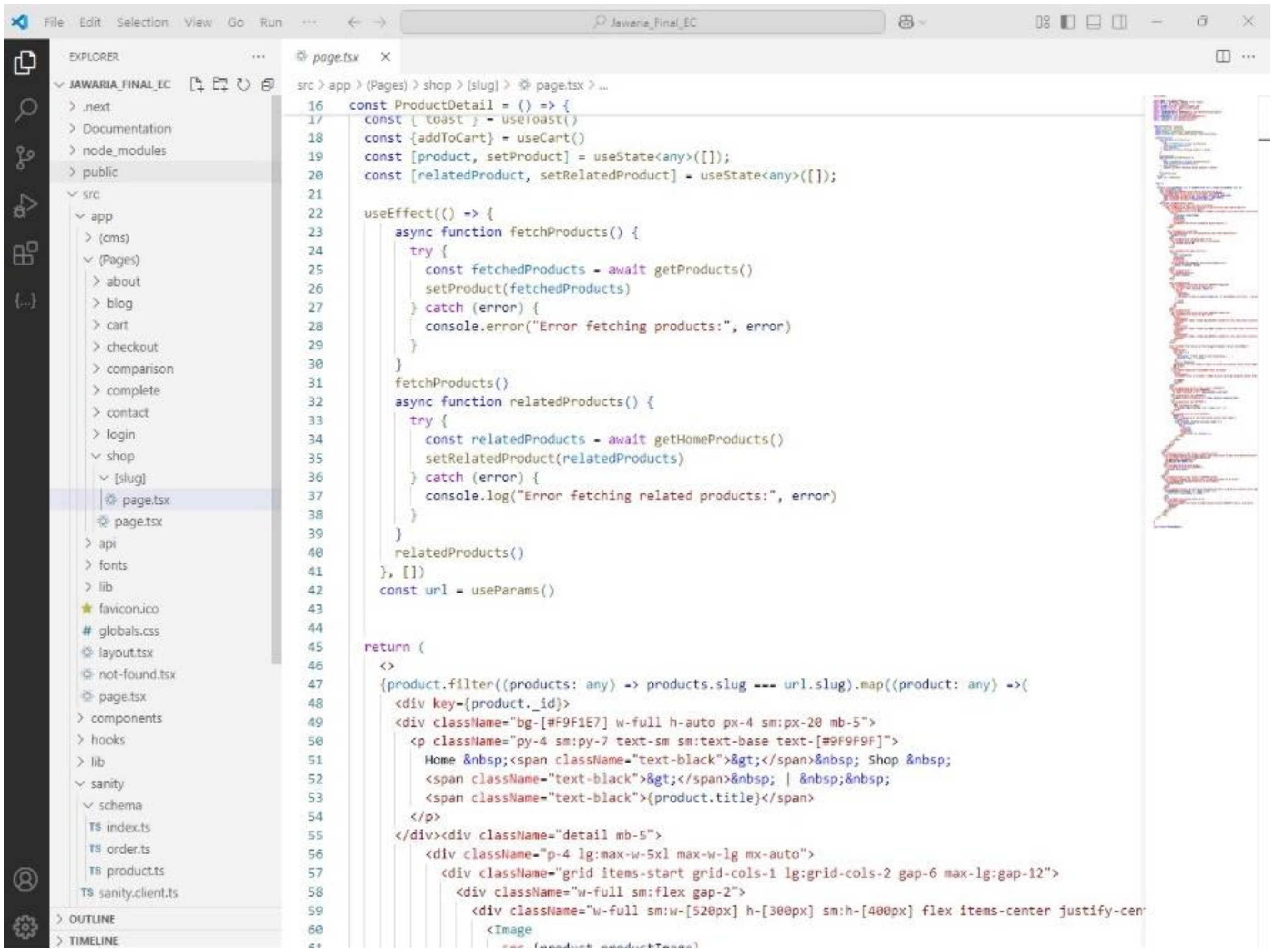
API Calls:

Fetch All Products



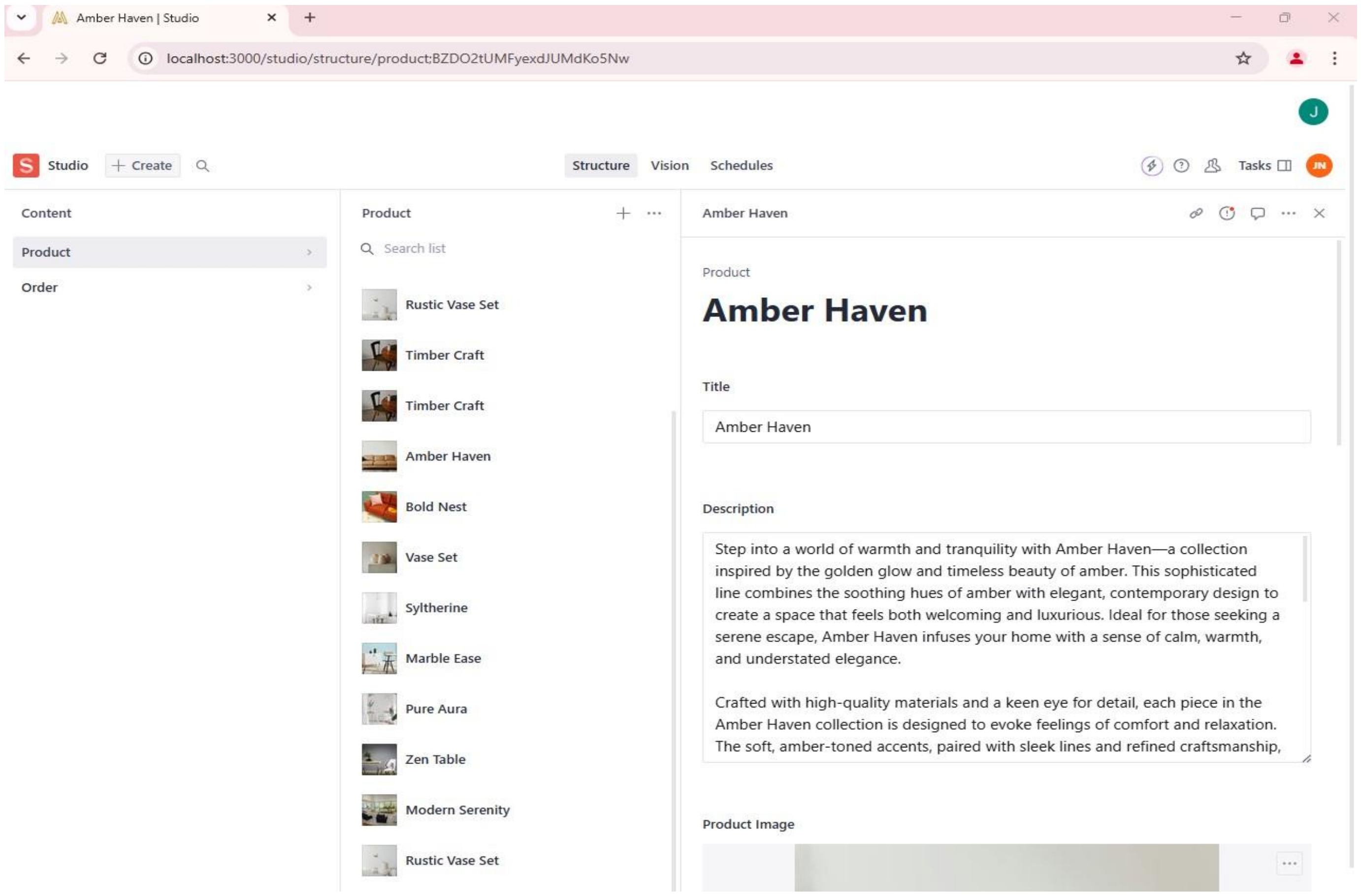
```ts
import { groq } from "next-sanity";
import sanityClient from "./sanity.client";

export async function getProducts() {
    return sanityClient.fetch(
        groq `
        *[_type == "product"]{
        _id,
        title,
        description,
        "slug": slug.current,
        "productImage": productImage.asset->url,
        price,
        category,
        summary,
        tags,
        isNew
        }
        `
    )
}



export async function getHomeProducts() {
    return sanityClient.fetch(
        groq `
        *[_type == "product"][0...8]{
        _id,
        title,
        description,
        "slug": slug.current,
        "productImage": productImage.asset->url,
        price,
        category,
        summary,
        tags,
        isNew
        }
        `
    )
}

export async function getOrders() {
    return sanityClient.fetch(
```

Fetch Product by Slug



```tsx
const ProductDetail = () => {
  const { toast } = useToast()
  const {addToCart} = useCart()
  const [product, setProduct] = useState<any>([]);
  const [relatedProduct, setRelatedProduct] = useState<any>([]);

  useEffect(() => {
      async function fetchProducts() {
        try {
          const fetchedProducts = await getProducts()
          setProduct(fetchedProducts)
        } catch (error) {
          console.error("Error fetching products:", error)
        }
      }
      fetchProducts()
      async function relatedProducts() {
        try {
          const relatedProducts = await getHomeProducts()
          setRelatedProduct(relatedProducts)
        } catch (error) {
          console.log("Error fetching related products:", error)
        }
      }
      relatedProducts()
    }, [])
    const url = useParams()


  return (
    <>
    {product.filter((products: any) => products.slug === url.slug).map((product: any) =>(
      <div key={product._id}>
      <div className="bg-[#F9F1E7] w-full h-auto px-4 sm:px-20 mb-5">
        <p className="py-4 sm:py-7 text-sm sm:text-base text-[#9F9F9F]">
          Home &nbsp;<span className="text-black">&gt;</span>&nbsp; Shop &nbsp;
          <span className="text-black">&gt;</span>&nbsp; | &nbsp;&nbsp;
          <span className="text-black">{product.title}</span>
        </p>
      </div><div className="detail mb-5">
          <div className="p-4 lg:max-w-5xl max-w-lg mx-auto">
            <div className="grid items-start grid-cols-1 lg:grid-cols-2 gap-6 max-lg:gap-12">
              <div className="w-full sm:flex gap-2">
                <div className="w-full sm:w-[520px] h-[300px] sm:h-[400px] flex items-center justify-cen
                  <Image
```

1. Data Import



Product

Amber Haven

Title: Amber Haven

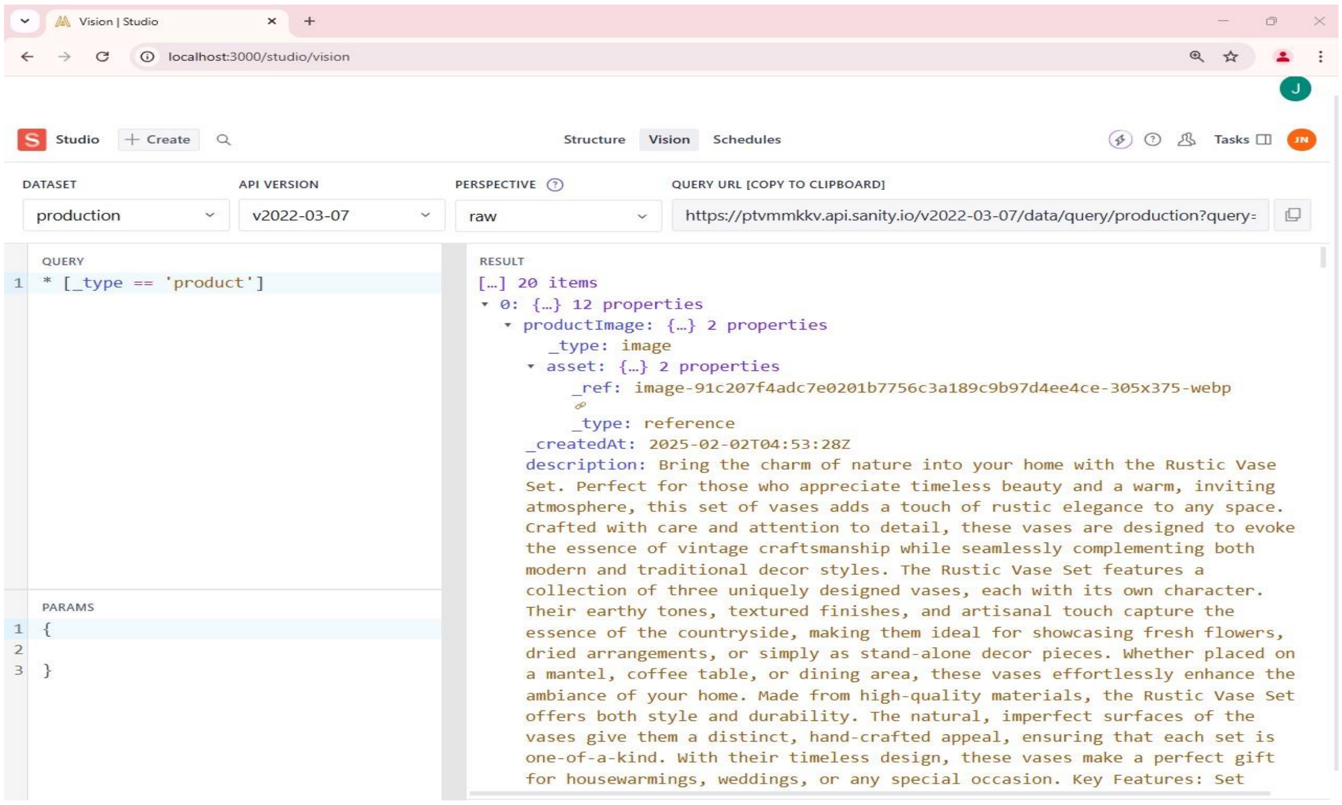
Description:

Step into a world of warmth and tranquility with Amber Haven—a collection inspired by the golden glow and timeless beauty of amber. This sophisticated line combines the soothing hues of amber with elegant, contemporary design to create a space that feels both welcoming and luxurious. Ideal for those seeking a serene escape, Amber Haven infuses your home with a sense of calm, warmth, and understated elegance.

Crafted with high-quality materials and a keen eye for detail, each piece in the Amber Haven collection is designed to evoke feelings of comfort and relaxation. The soft, amber-toned accents, paired with sleek lines and refined craftsmanship,

Product Image

2. API Call Response



DATASET: production
API VERSION: v2022-03-07
PERSPECTIVE: raw
QUERY URL [COPY TO CLIPBOARD]: https://ptvmmkkv.api.sanity.io/v2022-03-07/data/query/production?query=

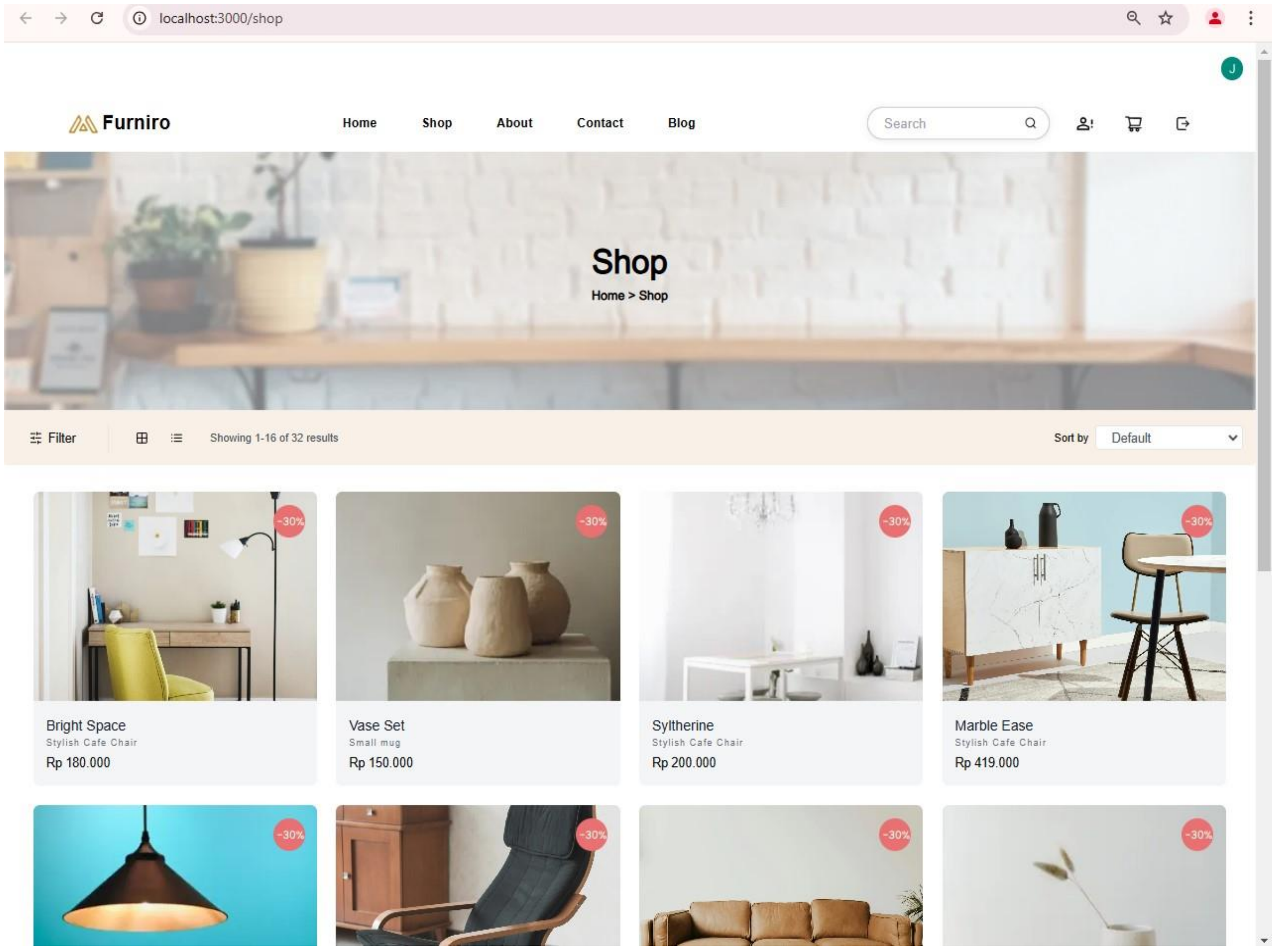
QUERY
```
* [_type == 'product']
```

PARAMS
```
{

}
```

RESULT
```
[…] 20 items
▾ 0: {…} 12 properties
  ▾ productImage: {…} 2 properties
      _type: image
    ▾ asset: {…} 2 properties
        _ref: image-91c207f4adc7e0201b7756c3a189c9b97d4ee4ce-305x375-webp
        _type: reference
  _createdAt: 2025-02-02T04:53:28Z
  description: Bring the charm of nature into your home with the Rustic Vase
  Set. Perfect for those who appreciate timeless beauty and a warm, inviting
  atmosphere, this set of vases adds a touch of rustic elegance to any space.
  Crafted with care and attention to detail, these vases are designed to evoke
  the essence of vintage craftsmanship while seamlessly complementing both
  modern and traditional decor styles. The Rustic Vase Set features a
  collection of three uniquely designed vases, each with its own character.
  Their earthy tones, textured finishes, and artisanal touch capture the
  essence of the countryside, making them ideal for showcasing fresh flowers,
  dried arrangements, or simply as stand-alone decor pieces. Whether placed on
  a mantel, coffee table, or dining area, these vases effortlessly enhance the
  ambiance of your home. Made from high-quality materials, the Rustic Vase Set
  offers both style and durability. The natural, imperfect surfaces of the
  vases give them a distinct, hand-crafted appeal, ensuring that each set is
  one-of-a-kind. With their timeless design, these vases make a perfect gift
  for housewarmings, weddings, or any special occasion. Key Features: Set
```

3. Frontend Display

Screenshot of Furniro products rendered dynamically on the website.



Conclusion:

The integration established a seamless process for importing, validating, and displaying Furniro product data from the API into Sanity CMS. Custom schemas and migration scripts ensured data consistency and project alignment. Future steps include enhancing API queries and implementing advanced features like search and filtering.