Day 3: API Integration & Data Migration Simplified

Today's task was all about integrating APIs and migrating data into Sanity CMS. The main goal was to import product data from an external API into Sanity CMS and then dynamically fetch it for the frontend. It involved a multi-step process that was both challenging and exciting. Every step brought its own learning, and I truly enjoyed the journey! 🚀

What I Did Today

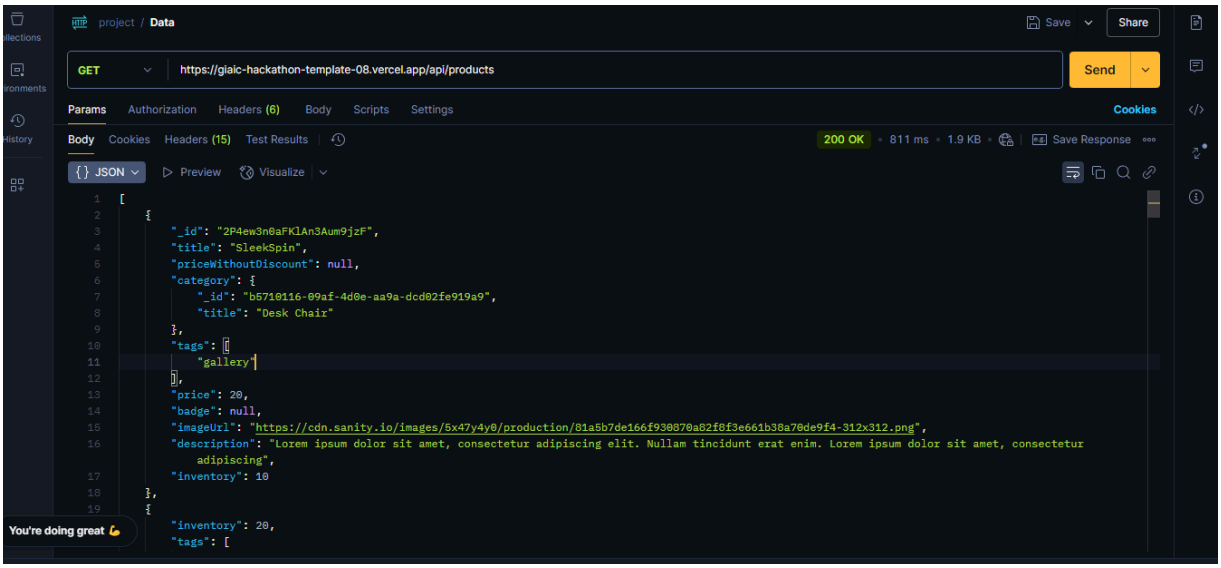
1. API Testing with Postman:

Before importing data, I tested the API endpoints using Postman to ensure everything was working correctly. This step helped in validating the responses and understanding the data structure.
Example Endpoint Tested : ***https://api.example.com/products***

- /products

Response Example:

(Include the Postman response screenshot or relevant JSON data here)



---

2. Imported Data into Sanity CMS

To populate Sanity CMS with API data, I used the provided data migration script. This script fetched data from the external API and imported it into Sanity CMS using its APIs.

Script Workflow:

1. Fetch data from the external API.
2. Map the API fields to the Sanity schema fields.
3. Push the transformed data into Sanity CMS.

During Import:

- Ensured that API field names (like product_title) matched Sanity schema fields (like name).
- Validated data before pushing it into Sanity CMS.

2. Updated the Sanity CMS Schema

I made significant updates to the `Product Schema` in Sanity CMS to enhance its functionality and adaptability.

Key Changes I Made:

- Introduced discountPercentage, isFeaturedProduct, tags, and color for added flexibility.
- Replaced static category options with dynamic references to a Category Schema.

3. Data Fetching from Sanity CMS

I created a utility function getAllProducts.ts in my project library to fetch data dynamically from Sanity CMS.

4. Frontend Integration

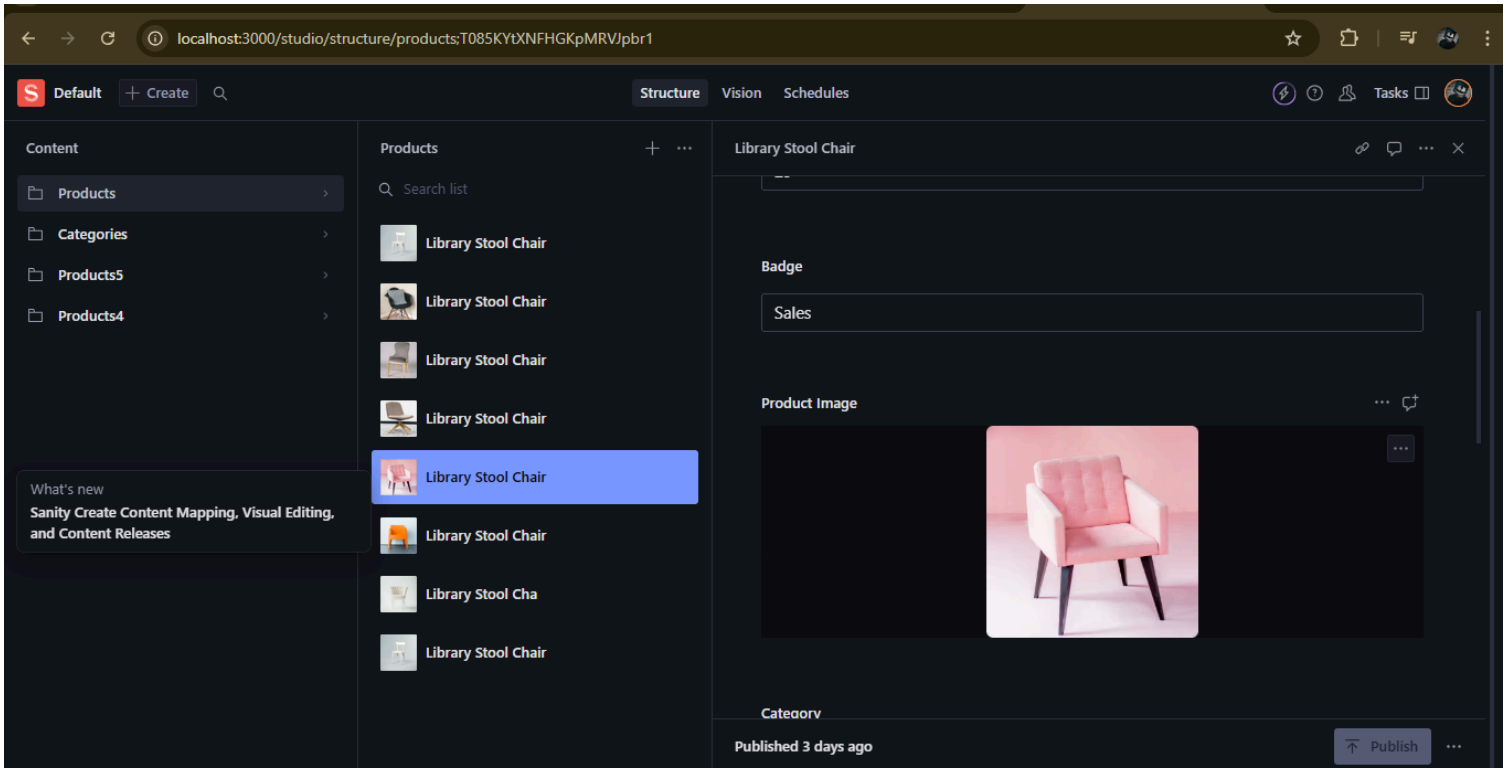
I integrated the getAllProducts function into various components and pages, such as:

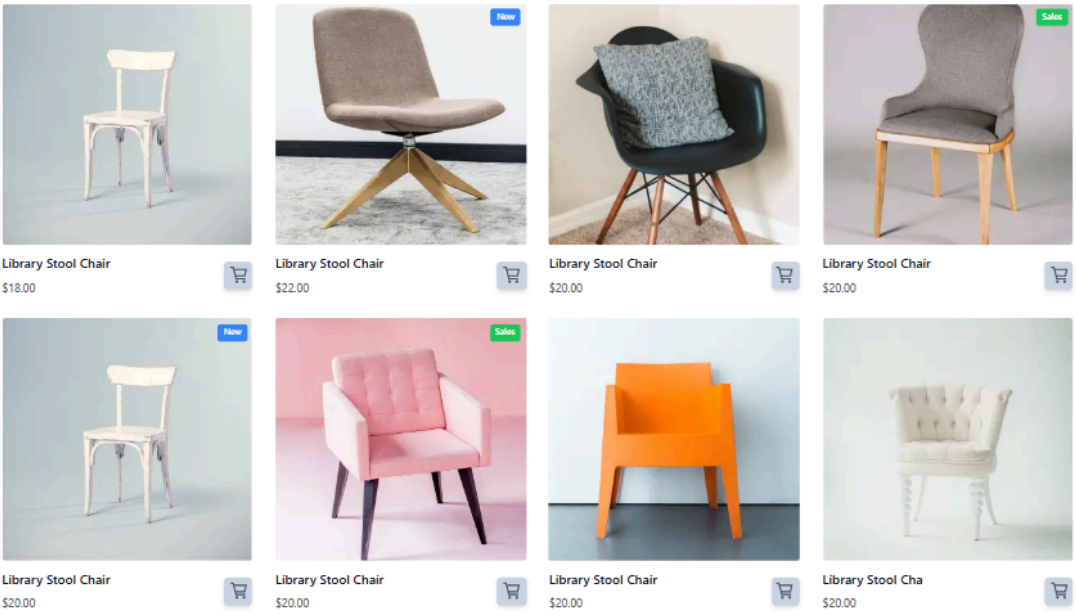
- Featured Products Section
- Latest Products on the Shop Page
- Dynamic Product Details Page using Routing

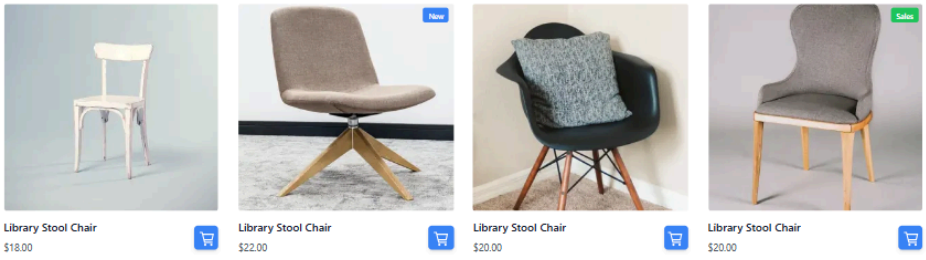
Example for Dynamic Routing (Single Product Page):

What's Next:

Moving forward, I'll focus on enhancing the user experience by adding features like product filtering, sorting, and SEO optimizations. The journey so far has been incredibly fulfilling, and I'm excited to see what Day 4 brings! 🚀
Let me know your thoughts, feedback, or suggestions. 😊







Top Categories

Home Shop Products Pages About

Contact: (808) 555-0111

Desk Chair

Total Products:

Home Shop Products Pages About

Contact: (808) 555-0111

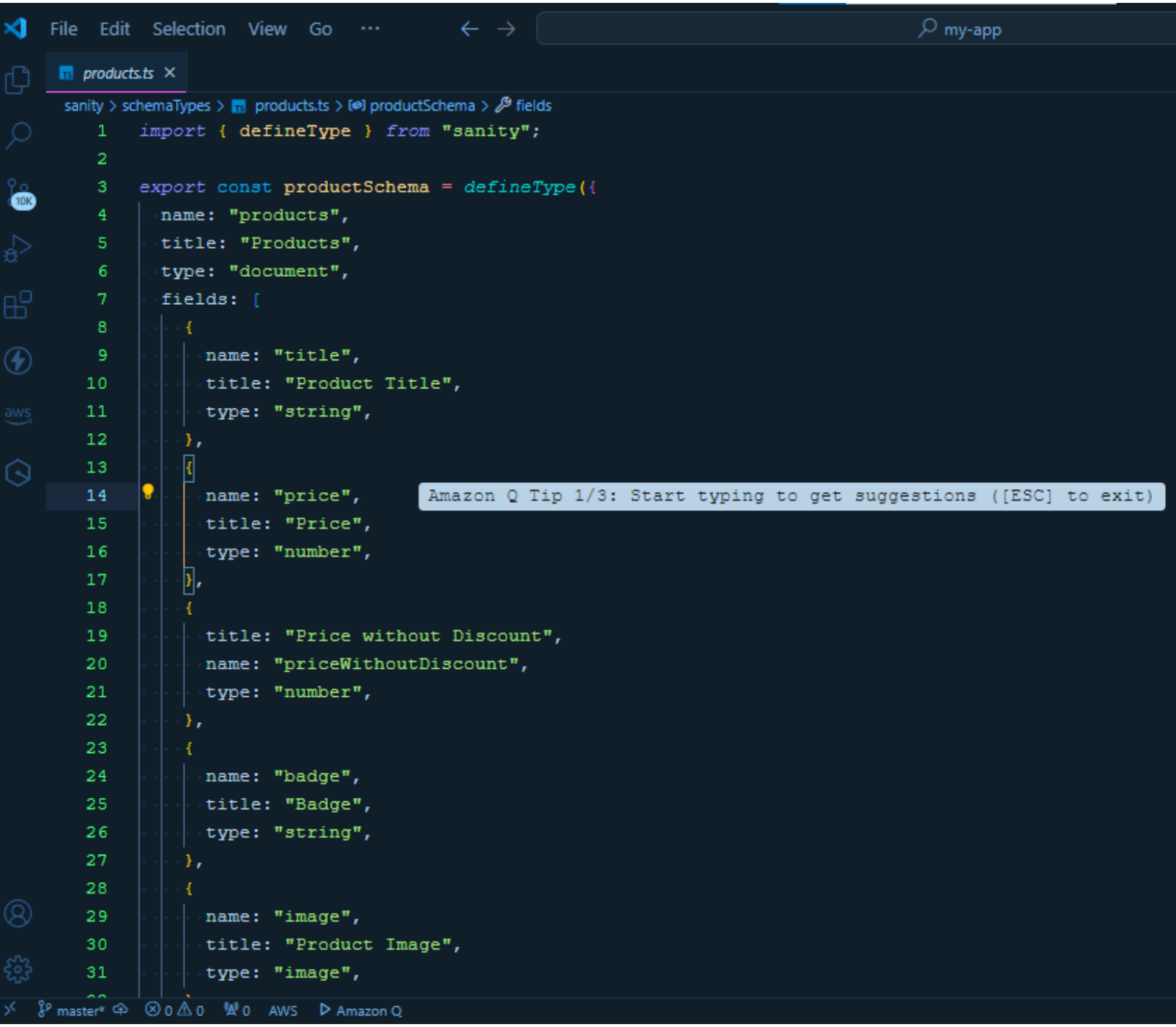
Library Stool Cha

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

$20

Inventory: 20 available

Add to Cart


File Edit Selection View Go ···
my-app
products.ts
sanity > schemaTypes > products.ts > productSchema > fields
import { defineType } from "sanity";
export const productSchema = defineType({
name: "products",
title: "Products",
type: "document",
fields: [
{
name: "title",
title: "Product Title",
type: "string",
},
{
name: "price",
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)
title: "Price",
type: "number",
},
{
title: "Price without Discount",
name: "priceWithoutDiscount",
type: "number",
},
{
name: "badge",
title: "Badge",
type: "string",
},
{
name: "image",
title: "Product Image",
type: "image",
master*
AWS
Amazon Q

components > Itam.tsx > Product

```tsx
};

const ProductPage = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);
  const router = useRouter(); // Hook to navigate between pages

  useEffect(() => {
    const fetchData = async () => {
      try {
        const query = `*[_type == "products"] {
          _id,
          title,
          price,
          priceWithoutDiscount,
          category -> {
            _id,
            title
          },
          tags,
          badge,
          "imageUrl": image.asset->url,
          description,
          inventory,
          coloredIcon,
          newLabel,
          salesLabel
        }`;

        const result = await client.fetch(query);
```

Ln 19, Col 21 Spaces: 2 UTF-8 CRLF