Day 3 - API Integration Report – Hekto Furniture (Template 4)

Prepared by Owais Qazi

---

1. Introduction

This report details the API integration process for Hekto Furniture's marketplace project, including schema adjustments, migration steps, and tools used. Screenshots and code snippets are provided to illustrate the process.

---

2. API Integration Process

The API integration involved fetching product data from an external API and storing it in Sanity CMS. The following steps were taken:

1. API Endpoint Analysis:

- The provided API contained product details, including images, descriptions, and pricing.

2. Fetching Data:

- Next.js was used to fetch data using `fetch()`.
- Data was retrieved in JSON format and parsed accordingly.

3. Storing in Sanity CMS:

- A schema for 'products' was created in Sanity Studio.
- Data migration was performed using a script to populate Sanity with API data.

4. Fetching Data from Sanity to Frontend:

- The Next.js frontend was configured to pull product details dynamically from Sanity.

```tsx
import React, { useEffect, useState } from "react";
import Link from "next/link";
import Image from "next/image";
import { client } from "@/sanity/lib/client";
import imageUrlBuilder from "@sanity/image-url";

const builder = imageUrlBuilder(client);

interface Product {
  _id: string;
  name: string;
  image: any;
}

const urlFor = (source: any) => builder.image(source).width(300).height(
300).url();

const FeaturedProducts: React.FC = () => {
  const [products, setProducts] = useState<Product[]>([]);

  useEffect(() => {
    const fetchProducts = async () => {
      const query = `*[_type == "products" && isFeaturedProduct == true] {
        _id, name, image
      }`;
      const data = await client.fetch(query);
      setProducts(data);
    };

    fetchProducts();
  }, []);

  return (
    <div className="bg-white py-10">
      <h3 className="text-center text-2xl font-bold mb-6">
Featured Products</h3>
      <div className=
"grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-6 lg:mx-60">
        {products.map((product) => (
          <Link href={`/products/${product._id}`} key={product._id}>
            <div className=
"border rounded-lg p-2 shadow-lg text-center h-96 cursor-pointer">
              {product.image && (
                <Image
                  src={urlFor(product.image)}
                  alt={product.name}
                  width={300}
                  height={300}
                  className="w-full h-72 object-cover mb-4"
                />
              )}
              <h3 className="font-medium text-lg">{product.name}</h3>
            </div>
          </Link>
        ))}
      </div>
    </div>
  );
};

export default FeaturedProducts;
```

3. Schema Adjustments

The default schema required modifications to align with the API structure. The following changes were made:

- Added fields: `productName`, `description`, `price`, `imageURL`, `category`, `stockAvailability`.

- Ensured data types matched API response fields.

```
export default {
    name: 'products',
    type: 'document',
    title: 'Products',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Name',
        validation: (Rule: any) => Rule.required().
error('Name is required'),
      },
      {
        name: 'image',
        type: 'image',
        title: 'Image',
        options: {
          hotspot: true,
        },
        description:
'Upload an image of the product.',
      },
      {
        name: 'price',
        type: 'string',
        title: 'Price',
        validation: (Rule: any) => Rule.required().
error('Price is required'),
      },
      {
        name: 'description',
        type: 'text',
        title: 'Description',
        validation: (Rule: any) =>
          Rule.max(150).warning(
'Keep the description under 150 characters.'),
      },
      {
        name: 'discountPercentage',
        type: 'number',
        title: 'Discount Percentage',
        validation: (Rule: any) =>
          Rule.min(0).max(100).warning(
'Discount must be between 0 and 100.'),
      },
      {
        name: 'isFeaturedProduct',
        type: 'boolean',
        title: 'Is Featured Product',
      },
      {
        name: 'stockLevel',
        type: 'number',
        title: 'Stock Level',
        validation: (Rule: any) => Rule.min(0).error(
'Stock level must be a positive number.'),
      },
      {
        name: 'category',
        type: 'string',
        title: 'Category',
        options: {
          list: [
            { title: 'Chair', value: 'Chair' },
            { title: 'Sofa', value: 'Sofa' },
          ],
        },
        validation: (Rule: any) => Rule.required().
error('Category is required'),
      },
    ],
  };
```

---

4. Migration Steps and Tools Used

Tools Used:

- Next.js for frontend integration.
- Sanity CMS for structured content storage.
- Axios for API requests.
- Sanity CLI & Scripts for data migration.

Migration Steps:

1. Created a script to fetch and transform API data.
2. Used Sanity's client library to push data to CMS.
3. Verified successful migration using Sanity Studio.

Code Snippet:

```
export const apiVersion =
  process.env.NEXT_PUBLIC_SANITY_API_VERSION || '2025-01-18'

export const dataset = assertValue(
  process.env.NEXT_PUBLIC_SANITY_DATASET,
  'Missing environment variable: NEXT_PUBLIC_SANITY_DATASET'
)

export const projectId = assertValue(
  process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  'Missing environment variable: NEXT_PUBLIC_SANITY_PROJECT_ID'
)

export const token = assertValue(
  process.env.NEXT_PUBLIC_SANITY_API_TOKEN,
  'Missing environment variable: NEXT_PUBLIC_SANITY_API_TOKEN'
)

function assertValue<T>(v: T | undefined, errorMessage: string): T {
  if (v === undefined) {
    throw new Error(errorMessage)
  }

  return v
}
```

Client.ts File

```
import { createClient } from 'next-sanity'
import { apiVersion, dataset, projectId, token } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: false,
  token,

})
```

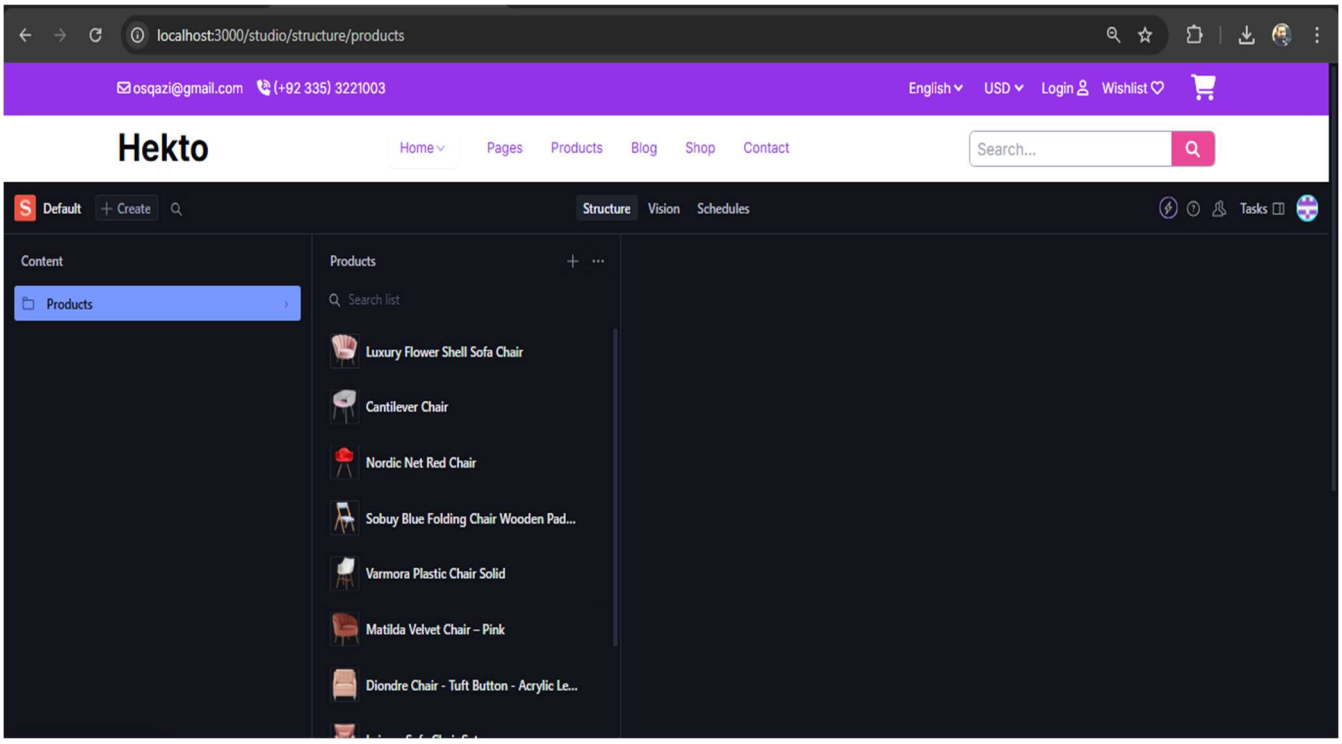
Import Script

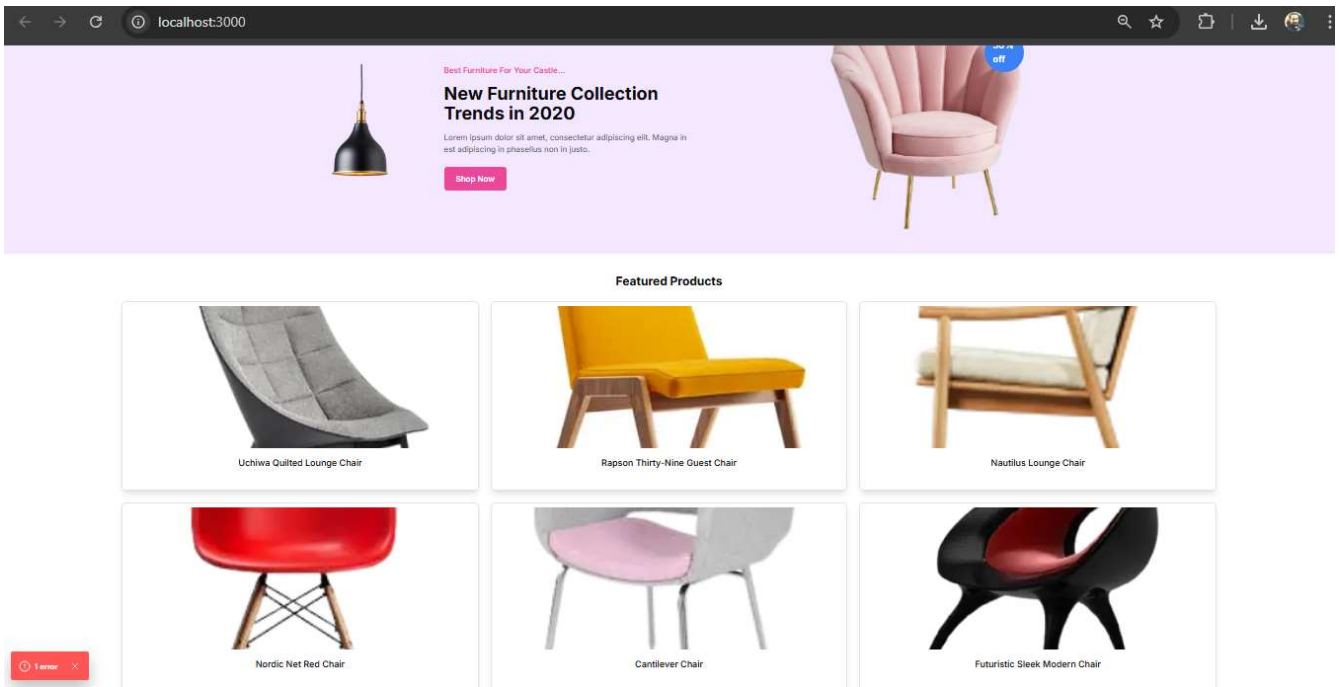
```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../../.env') });

const tokn = "xyz"

const client = createClient({
  projectId: "2iy1r2df",
  dataset: "production",
  token: tokn,
  apiVersion: '2025-01-15',
  useCdn: false,
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading Image : ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType:
'arraybuffer' });
    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 importData() {
  try {
    console.log('Fetching Product Data From API ...');

    const response = await axios.get(
"https://next-ecommerce-template-4.vercel.app/api/product")
    const products = response.data.products;

    for (const item of products) {
      console.log(`Processing Item: ${item.name}`);

      let imageRef = null;
      if (item.imagePath) {
        imageRef = await uploadImageToSanity(item.imagePath);
      }

      const sanityItem = {
        _type: 'products',
        name: item.name,
        category: item.category || null,
        price: item.price,
        description: item.description || '',
        discountPercentage: item.discountPercentage || 0,
        stockLevel: item.stockLevel || 0,
        isFeaturedProduct: item.isFeaturedProduct,
        image: imageRef
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined,
      };

      console.log(`Uploading ${sanityItem.category} - ${sanityItem.name}
 to Sanity !`);
      const result = await client.create(sanityItem);
      console.log(`Uploaded Successfully: ${result._id}`);
      console.log(
"-------------------------------------------------------------")
      console.log("\n\n")
    }

    console.log('Data Import Completed Successfully !');
  } catch (error) {
    console.error('Error Importing Data : ', error);
  }
}

importData();
```

5. API Calls and Data Display

After successful migration, API calls were verified in the frontend:

- Fetched data dynamically from Sanity.
- Rendered product details on the marketplace.





6. Conclusion

The integration of API data into Sanity CMS and Next.js frontend was successfully completed. The marketplace now dynamically fetches and displays products stored in Sanity. The implementation ensures a scalable and flexible content management approach.