DAY 3 API INTEGRATION REPORT

OBJECTIVE

'The objective of Day 3 is to integrate APIs into the Next.js project and migrate data into Sanity CMS to populate the marketplace backend. This report outlines the steps followed for Template 4, which involves integrating the provided API into the Next.js project and ensuring that the Sanity CMS schema aligns with the data source.'

API Integration Steps

1. **Understand the API**:
   - Review the API documentation and identify key endpoints for products, categories, and other relevant data.
   - Note down field structures and data types for compatibility with the schema.
2. **Create Utility Functions**:
   - Develop reusable functions to fetch and process API data.
3. **Render Data in Components**:
   - Use fetched data to populate frontend components in Next.js.
4. **Test API Endpoints**:
   - Verify endpoints using tools like Postman or browser developer tools.
   - Log responses to ensure accurate data retrieval.
5. **Implement Error Handling**:
   - Log errors centrally and display fallback UI elements for a seamless user experience.

Data Migration Steps.

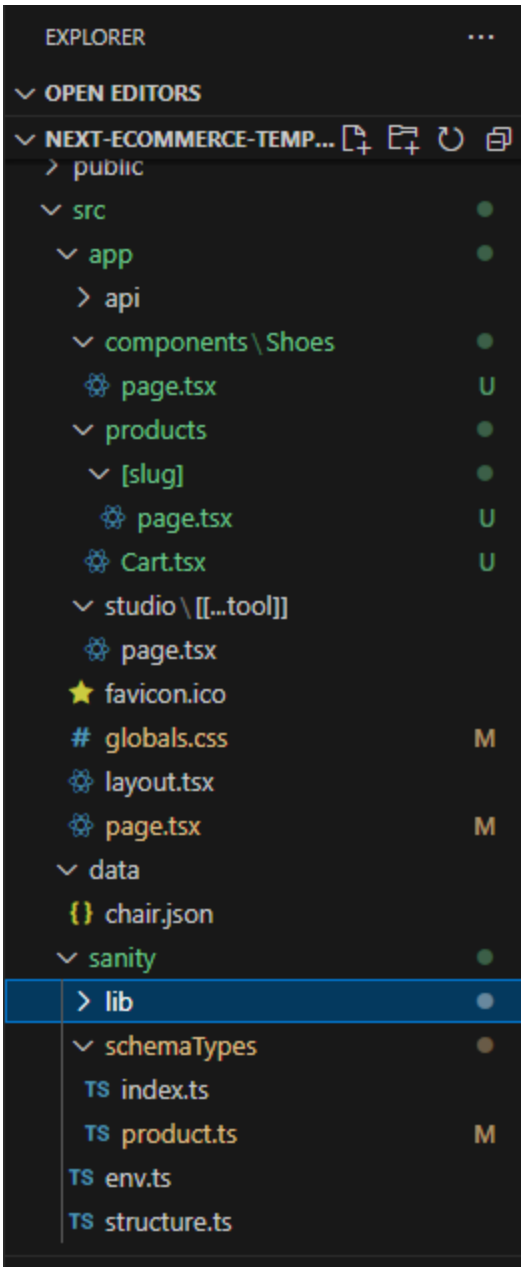
1. **Validate and Adjust Schema**:
   - Compare API field names and structures with the Sanity CMS schema.
   - Update schema fields for consistency, mapping API fields to schema fields if needed.
2. **Choose a Migration Method**:
   - **API-Based Migration**: Write scripts to fetch and insert data directly into Sanity CMS.
3. **Validate Imported Data**:
   - Ensure that all data is imported accurately and aligns with the schema.
   - Perform thorough checks to identify any errors or discrepancies.
4. **Backup and Error Resolution**:
   - Back up your Sanity project before large imports.
   - Address migration errors and rerun failed data batches if needed.
5. **Test Data in Frontend**:

- Populate frontend components with imported data to verify compatibility and correctness.

Tools Used for API Integration and Data Migration

1. **Sanity CMS**
   - Used as the backend content management system for storing and managing data.
   - Tools: Schema builder, import tools, and dataset management.
2. **Next.js**
   - Framework for building the frontend of the marketplace.
   - Used for rendering data fetched via APIs and creating reusable components.
3. **API Testing Tools**
   - **Postman**: Verified API endpoints and tested responses.
   - **Browser Developer Tools**: Used the network tab to inspect API calls and responses.
4. **Code Editors**
   - **Visual Studio Code (VS Code)**: Primary editor for writing and managing code, including utility functions, migration scripts, and schemas.
5. **Node.js**
   - Utilized for running scripts to fetch, transform, and migrate data.
6. **API Documentation and References**
   - Used documentation for understanding API structures, endpoints, and parameters.
7. **Data Validation Tools**

- Validation scripts: Ensured accuracy of imported data.
- Sanity Studio: Verified data alignment with schema fields.


EXPLORER
OPEN EDITORS
NEXT-ECOMMERCE-TEMP...
public
src
app
api
components \ Shoes
page.tsx U
products
[slug]
page.tsx U
Cart.tsx U
studio \ [[...tool]]
page.tsx
favicon.ico
globals.css M
layout.tsx
page.tsx M
data
chair.json
sanity
lib
schemaTypes
index.ts
product.ts M
env.ts
structure.ts

SANITY SCHEMA DEFINE:

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Name',
      validation: (Rule: any) => Rule.required().error('Name is required'),
    },
    {
     name:"slug",
     type:"slug",
     title:"slug",
     options:{
       source: "name",
      }
     },
     {
      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'),
    },
  ],
};

```

DATA MIGRATION:

```
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 client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  token: process.env.SANITY_API_TOKEN,
  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: 'product',
        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();
```

IMPORT SANITY DATA

```
"use client"
import Image from 'next/image'
import React ,{useState,useEffect} from 'react'

import { client } from '@/sanity/lib/client'
import { allProduct } from '@/sanity/lib/queries'
import { urlFor } from '@/sanity/lib/image'
import Link from 'next/link'
import {product} from "@/sanity/schemaTypes/product"


const Shoes = () => {
    const [product, setProduct] = useState<product[]>([])
    useEffect(() => {
      async function fetchproduct() {
        const fetchedproduct:product[]=await client.fetch(allProduct)
        setProduct(fetchedproduct);
      }
    fetchproduct()
    },[])

  return (
    <div className='mx-w-2xl mx-auto w-[1200px] h-[900px] px-4 py-8'>
      <h1 className=' text-2xl font-bold mb-6 text-center'>
        Our Latest Product
      </h1>
      <div className='grid grid-cols-1 sm:grid-cols-2 md-grid-cols-3 lg:grid-cols-4 gap-[30px]'>
        {product.map((product)=>(
            <div key={product._id}>
              <div className=' border rounded-lg  shadow-2xl p-4 hover:shadow-lg transition duration-200 '>
                <Link href={`/products/${product.slug?.current}`}>



{product.image && (
<Image
  src={urlFor(product.image).url()} alt='image' width={250} height={250} className='w-48 h-40 object-cover rounded-md ml-5'
/>

)}
<h2 className='text-lg ml-4  font-semibold mt-4'>
{product.name}

</h2>
<p className='text-gray-500 mt-2 ml-4'>
${product.price}

</p>
</Link>
 <button className='bg-blue-500 text-white px-14 py-1 rounded-sm mt-3 ml-4  '>Add to Cart</button>
</div>


            </div>
        ))}
        </div>
        </div>
  )
}

export default Shoes
```

Our Latest Product

Stylish Golden Metal Legs Mint Blue Fabric Velvet Sofa Leisure Armchair

$780

Add to Cart

High Quality Modern Customized Plastic Chair

$150

Add to Cart

Futuristic Sleek Modern Chair

$2000

Add to Cart

Liberty Wood 63' Floating Entertainment Center

$1100

Add to Cart

Leisure Sofa Chair Set

$1800

Add to Cart

Varmora Plastic Chair Solid

$100

Add to Cart

Cantilever Chair

$780

Add to Cart

Luxury Flower Shell Sofa Chair

$2500

Add to Cart

Uchiwa Quilted Lounge Chair

$1600

Add to Cart

Alpha Chair – Solid Ebonised Oak

$900

Add to Cart

Replica Hans Wegner Wishbone Chair

$750

Add to Cart

Matilda Velvet Chair – Pink

$600

Add to Cart

Nordic Net Red Chair

$320

Add to Cart

Tribù Elio Chair

$1200

Add to Cart

Armchair Tortuga

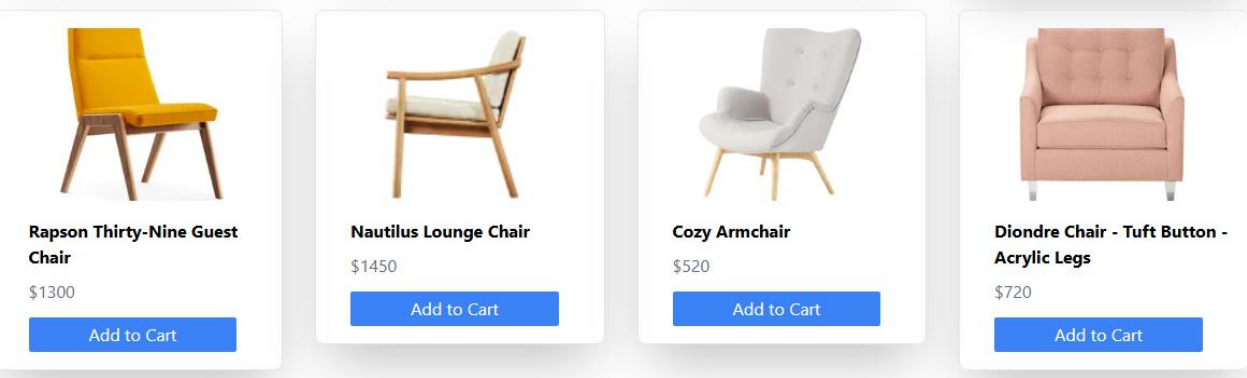
$850

Add to Cart

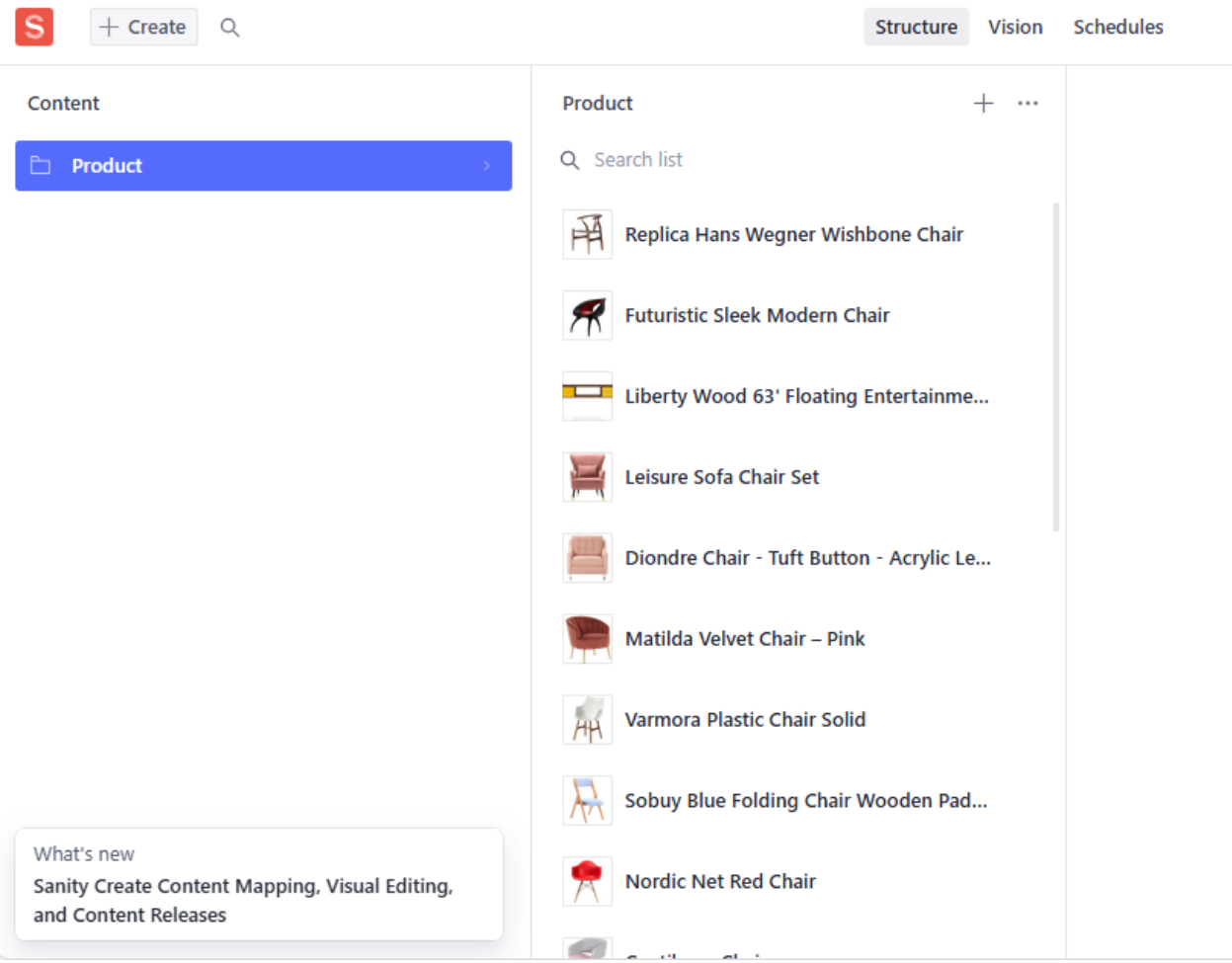
Hans Wegner Style Three-Legged Shell Chair

$990

Add to Cart



SANITY CMS FIELD PRODUCTS



SELF-VALIDATION CHECKLIST

Task	***status***
API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next.js	✓
Submission Preparation	✓