MARKETPLACE BUILDER HACKATHON 2025

DAY 3

API INTEGRATION AND DATA MIGRATION

API Integration Process

API Understanding:

I reviewed the API documentation for Template 2 (Sir Bilal Muhammad Khan and Sir Aneeq) and familiarized myself with the available endpoints, such as /products and /categories.

Key Endpoints:

- Products: API URL
- Categories: API URL

Adjustments Made to Schema:

I compared my existing Sanity CMS schema with the API data structure and made adjustments where necessary. For instance, the API used the field product title, which I mapped to the name field in the Sanity schema. I made similar adjustments for other fields to ensure smooth data migration.

anity > schemaTypes > TS product.ts

```ts
import { defineType, defineField } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        defineField({
            name:"category",
            title:"Category",
            type:"reference",
            to:[{
                type:"category"
            }]
        }
        ),
        defineField({
            name: "name",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        }),
        defineField({
            name: "slug",
            title: "Slug",
            validation: (rule) => rule.required(),
            type: "slug"
        }),
        defineField({
            name: "image",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        }),
        defineField({
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        }),
        defineField({
            name: "quantity",
            title: "Quantity",
            type: "number",
            validation: (rule) => rule.min(0),
          }),
```

```ts
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        }),
        defineField({
            name: "quantity",
            title: "Quantity",
            type: "number",
            validation: (rule) => rule.min(0),
          }),
        defineField({
            name: "tags",
            type: "array",
            title: "Tags",
            of:[{
                type: "string"
            }]
        }),
        defineField({
            name: 'description',
            title: 'Description',
            type: 'text',
            description: 'Detailed description of the product',
          }),
          defineField({
            name: 'features',
            title: 'Features',
            type: 'array',
            of: [{ type: 'string' }],
            description: 'List of key features of the product',
          }),
          defineField({
            name: 'dimensions',
            title: 'Dimensions',
            type: 'object',
            fields: [
              { name: 'height', title: 'Height', type: 'string' },
              { name: 'width', title: 'Width', type: 'string' },
              { name: 'depth', title: 'Depth', type: 'string' },
            ],
            description: 'Dimensions of the product',
          }),
    ]
})
```

```ts
import { defineType,defineField } from "sanity";

export const Category = defineType({
    name: "category",
    title: "Category",
    type: "document",
    fields:[
        defineField({
            name: "name",
            title: "Name",
            type: "string",
            validation: (rule) => rule.required(),
        }),
        defineField({
            name: "slug",
            title: "Slug",
            type: "slug",
            validation: (rule) => rule.required(),
            options: {
                source: "name",
            }
        })
    ]
})
```

MIGRATION STEPS

METHOD USED:

I chose the migration approach using the provided API.

- **SCRIPT USED**: Migration script
- **SCHEMA VALIDATION**: I validated all migrated fields in sanity CMS and ensured they matched the expected data types and relationships.
- **DATA IMPORT PROCESS**: The script fetched data from the API and transformed it into the required format for import into sanity CMS. I tested this with a subset of data before importing the full dataset.

ts > JS importSanityData.mjs > ...

```
import { createClient } from '@sanity/client';  324.3k (gzipped: 85.4k)
import axios from 'axios';  62.3k (gzipped: 23.1k)
import dotenv from 'dotenv';  6.8k (gzipped: 3k)
import { fileURLToPath } from 'url';
import path from 'path';
import slugify from 'slugify';  8.8k (gzipped: 3.8k)

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

const client = createClient({
    projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
    dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
    useCdn: false,
    token: process.env.SANITY_API_TOKEN,
    apiVersion: '2021-08-31'
});

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 createCategory(category, counter) {
    try {
        const categoryExist = await client.fetch(`*[_type=="category" && slug.current==$slug][0]`, { slug: category.slug });
        if (categoryExist) {
            return categoryExist._id;
        }
        const catObj = {
            _type: "category",
            _id: `${category.slug}-${counter}`,
            name: category.name,
            slug: {
                _type: 'slug',
                current: slugify(category.name || 'default-category', { lower: true, strict: true })
            }
        };
        const response = await client.createOrReplace(catObj);
        console.log('Category created successfully', response);
        return response._id;
    } catch (error) {
        console.error('Failed to create category:', category.name, error);
        return null;
    }
}

async function importData() {
    try {
        console.log('Fetching products from API...');
        const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
        const products = response.data;
        let counter = 1;

        for (const product of products) {
            console.log(`Processing product: ${product.name}`);
            let imageRef = null;
            let catRef = null;

            if (product.image) {
                imageRef = await uploadImageToSanity(product.image);
            }
            if (product.category?.name) {
                catRef = await createCategory(product.category, counter);
            }

            const sanityProduct = {
```

```
            catRef = await createCategory(product.category, counter);
        }

        const sanityProduct = {
            _id: `product-${counter}`,
            _type: 'product',
            name: product.name,
            slug: {
                _type: 'slug',
                current: slugify(product.name, { lower: true, strict: true })
            },
            price: product.price,
            category: catRef ? {
                _type: 'reference',
                _ref: catRef
            } : undefined,
            tags: product.tags || [],
            quantity: 50,
            image: imageRef ? {
                _type: 'image',
                asset: {
                    _type: 'reference',
                    _ref: imageRef
                }
            } : undefined,
            description: product.description || "Default product description",
            features: product.features || ["Premium material", "Handmade upholstery"],
            dimensions: product.dimensions || {
                _type: 'dimensions',
                height: "110cm",
                width: "75cm",
                depth: "50cm"
            }
        };

        await client.createOrReplace(sanityProduct);
        console.log(`✅ Imported product: ${sanityProduct.name}`);
        counter++;
      }
      console.log('✅ Data import completed!');
    } catch (error) {
      console.error('Error importing data:', error);
    }
}

importData();
```

API INTEGRATION IN NEXT.JS

- **UTILITY FUNCTIONS**: I created utility functions to fetch data from the API endpoints and display it within my next.Js application.
- **RENDERING DATA IN COMPONENTS**: I created components to display products and categories on the frontend.
- **TESTING API INTEGRATION**: I used postman to test the API endpoints and ensure the data returned matched the expected structure. All endpoints were successfully integrated into the next.Js app, displaying the products and categories.

```tsx
"use client";

import React, { useEffect, useState } from "react";  7.8k (gzipped: 3k)
import client from "../../sanity/lib/client";
import { urlFor } from "../../sanity/lib/image";
import { four } from "../../sanity/lib/queries";
import { productdetail } from "../../../types/products";
import Image from "next/image";  17.2k (gzipped: 6.3k)
import { Button } from "../ui/button";
import Link from "next/link";  37.5k (gzipped: 11.3k)

const ProductCard = () => {
  const [products, setProducts] = useState<productdetail[]>([]);

  useEffect(() => {
    async function fetchProducts() {
      const fetchedProducts: productdetail[] = await client.fetch(four);
      setProducts(fetchedProducts);
    }
    fetchProducts();
  }, []);

  return (
```

```tsx
const ProductCard = () => {
  const [products, setProducts] = useState<productdetail[]>([]);

  useEffect(() => {
    async function fetchProducts() {
      const fetchedProducts: productdetail[] = await client.fetch(four);
      setProducts(fetchedProducts);
    }
    fetchProducts();
  }, []);

  return (
    <div>
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-[48px] mx-[86px] mt-20">
        {products.map((product) => (
          <div
            key={product._id}
            className="flex flex-col justify-center place-items-start mb-4 gap-[16px] w-[270px]"
          >
            <Link href={`/productdetail/${product.slug.current}`}>
              <div className="flex flex-col justify-center place-items-start mb-4 gap-[16px] w-[270px]">
                {product.image && (
                  <Image
                    src={urlFor(product.image).url()}
                    alt={product.name}
                    width={270}
                    height={270}
                    className="w-[270px] h-[270px]"
                  />
                )}
                <div>
                  <h3 className="font-bold text-lg">{product.name}</h3>
                  <p className="text-gray-600">{product.price}</p>
                </div>
                <Button>Add to cart</Button>
              </div>
            </Link>
          </div>
        ))}
      </div>
    </div>
  );
};

export default ProductCard;
```

EXPECTED OUTPUT

- **SANITY CMS POPULATION:**
  - The data was successfully imported into sanity CMS from the provided API. The products and categories were populated correctly and aligned with the schema.
- **FRONTEND DISPLAY:**
  - The product listings and categories were displayed correctly in the frontend after integrating the data from the API into the next.Js components.

Avion

Ceramics Chairs Crockory Cutlery Plant Pots Tables Tableware

Default Create

Structure Vision Schedules

Tasks

Content

Product

Category

What's new

Sanity Create Content Mapping, Visual Editing, and Content Releases

Product

Search list

The Dandy chair

Bright Space

Cloud Haven Chair

Bold Nest

Sunny Chic

Reflective Haven

Modern Serenity

Timeless Elegance

TimberCraft

Nordic Elegance

Serene Seat

Sleek Living

Tropical Vibe

Zen Table

Pure Aura

The Lucky Lamp

Retro Vibe

Wood Chair

The Dandy chair

Product

The Dandy chair

Category

Chairs

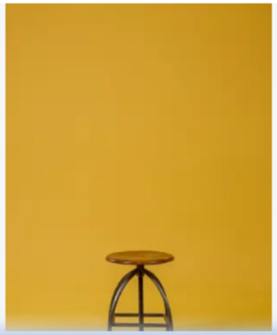
Title

The Dandy chair

Slug

the-dandy-chair

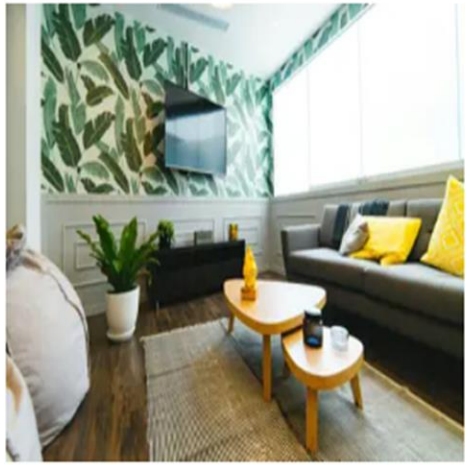
Product Image



The Poplar suede sofa

980

Add to cart



Tropical Vibe

550

Add to cart

Sleek Living

300

Add to cart

Serene Seat

350

Add to cart

BEST PRACTICES FOLLOWED

- .ENV FILES FOR SENSITIVE DATA:
  - All sensitive API keys were stored securely using environment variables.
- CLEAN CODING PRACTICES:
  - Used descriptive variable names and modularized functions for reusability.
  - Added comments to explain the logic behind complex code.
- DATA VALIDATION:
  - Ensured all fields in the migrated data matched the schema requirements in sanity CMS.
  - Implemented validation checks in the migration scripts to handle any discrepancies.
- VERSION CONTROL:
  - Regularly committed changes to GitHub with meaningful commit messages.
  - Tagging significant milestones in the repository.

Peer Review and Testing

FEEDBACK INCORPORATED:

I shared my code and documentation with peers for feedback, which I used to enhance the quality of the project.

CONCLUSION

I have completed the API integration and data migration tasks for day 3. The provided API data has been successfully integrated into my next.Js frontend, and the sanity CMS has been populated with the required data. The project is now ready for the next steps.