Marketplace-Builder-Hackathon

Day-3: API Integration and Data Migration in Sanity

EXPLANATION OF MIGRATING DATA FROM API TO SANITY IN NEXTJS

The focus for Day 3 of the hackathon was on API Integration and Data Migration. The task was to integrate external APIs into the marketplace project. implemented a solution that allows us to migrate and import data from an external API into sanity CMS. By the end of the day, the objective was to have a fully integrated marketplace with dynamic data fetched from APIs and correctly displayed on the frontend. Below is step by step explanation how I achieved it:

Created by Emad Ahmed

Roll Num => 227065

1. Fetching Data from External APIs

The first step in this process was to fetch data from the external API. In my Next.js application, where the frontend code was already present, I created a folder named "scripts" and within that, I created a file called "data-migration.mjs" where I wrote the code to fetch data from the API.

Afterward, I tested it both in the application and through Postman API, and my data was successfully fetched.

```
import sanityClient from '@sanity/client';

const client = sanityClient({
    projectId: '8p9t123f',
    dataset: 'production',
    useCdn: true,
    apiVersion: '2025-01-13',
    token:
    'skZmy1300QiIxde8zb8sqH40I2D0usUaQOaqsthKGotYwgHWP568Z1ujRLDHIfle44oevfr4R6jLR618c
    50fgGltRXSwJdiydpuqN0LLb9SlKTsH3qsfXjbLrRy4jiISmvLpoTfD1jNMhUWrUTSSKuBbpwuH4DWZqEf
    1Labmue50Y4T1KYUV',
});

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}`);
        }
}
```

2. Comparing API data with Sanity Schema

After the data was fetched, the next task was to compare the data structure of the API with the Sanity schema. The Sanity schema will be used to handle the product data. The main purpose of this step was to ensure that the data coming from the API matches the Sanity schema. This step ensures that the data from the API perfectly matches the Sanity schema.

3. Data Migration Script

After confirming the schema and the API data structure, the next step was to write a migration script in existing file data-migration.mjs that wood automate the process of importing the fetched data into Sanity. A script was created that would:

- 1. Fetch data from API
- 2. Validate data with sanity schema
- 3. Use the sanity client to push data into sanity

```
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(),
     });
```

4. Setting Up The Environment

Once the script was written, the next step was to configure Sanity. For the configuration, a token for the project needs to be generated from Sanity's official website, and by looking at the ID, DATA, and TOKEN, it needs to be written in the .env.local file, and also provided in the data-migration.mjs file.

NEXT_PUBLIC_SANITY_PROJECT_ID="8p9t123f"
NEXT_PUBLIC_SANITY_DATASET="production"
NEXT_PUBLIC_SANITY_API_TOKEN="skZmy1300QiIxde8zb8sqH40I2D0usUaQOaqsthKGotYwgHWP568Z1
ujRLDHIfle44oevfr4R6jLR618c50fgGltRXSwJdiydpuqN0LLb9SlKTsH3qsfXjbLrRy4jiISmvLpoTfD1j
NMhUWrUTSSKuBbpwuH4DWZqEf1Labmue50Y4T1KYUV"
NEXT_PUBLIC_SANITY_API_VERSION="2025-01-13"

5. Modifying package.json

To run the migration script, the necessary npm package was edit to the package json file under the scripts section. This made it easier to run the script through the command line.

```
"name": "day3",
  "version": "0.1.0",
  "private": true,
  "type": "module",
  Debug
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "importData": "node importData.js"
},
```

6. Importing Data into Sanity

With everything setup, I executed the migration script to send data into sanity by running the following command:

npm run migrate

```
import { client } from "@/sanity/lib/client";
import { NextResponse } from "next/server";
export async function GET() {
  try {
   const data = await client.fetch(`*[_type=="product"]{
   _id,
   title,
    "imageUrl" :productImage.asset -> url,
   price,
   tags,
   dicountPercentage,
   description,
   isNew
}`);
   return NextResponse.json(data, { status: 200 });
 } catch (error) {
   console.error('Error fetching data from Sanity:', error);
   return new NextResponse('Error fetching data', { status: 500 });
```

7. Displaying data in frontend

After fetching data from Sanity, I mapped it and printed it in cards, and successfully my data was displayed on the frontend.



Timber Craft

Introducing TimberCraft—a collection that celebrates the timeless beauty of wood craftsmanship and the art of nature's finest materials. Inspired by the rustic charm and durability of natural wood, TimberCraft brings warmth, character, and a touch of handcrafted elegance to any space. Perfect for those who value authenticity and sustainability, this collection combines the strength of timber with refined design, making it the ideal choice for modern and traditional homes alike. Each piece in the TimberCraft collection is meticulously crafted to highlight the unique grains and textures of the wood, bringing an organic, earthy feel to your space. Whether you're looking for furniture, decor or accents, TimberCraft offers a variety of beautifully designed pieces that blend rustic appeal with contemporary sophistication. The collection is designed to stand the test of time, with materials



Cloud Haven Chair

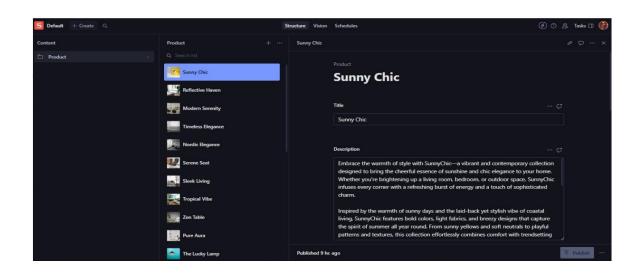
Sink into comfort with the Cloud Haven Chairwhere softness meets support in a beautifully designed piece that feels like a personal retreat. Inspired by the gentle, calming presence of clouds, this chair offers an ultra-comfortable, inviting experience that lets you relax and unwind, making it the perfect addition to any living room, bedroom, or cozy nook. The Cloud Haven Chair is crafted with premium materials that create a plush, cloud-like seating experience. Its luxurious cushioning provides gentle support, cradling you in comfort while the ergonomic design ensures the perfect balance between relaxation and posture. The chair's sleek, modern lines and soft, neutral tones blend seamlessly into any decor, adding a touch of sophistication and tranquility to your space. Whether you're enjoying a good book, meditating, or simply relaxing after a long day, the Cloud Haven Chair creates a peaceful sanctuary where you can



Bright Space

Products

Welcome to BrightSpace—a collection designed to infuse your home with light, energy, and vibrancy. Inspired by the power of natural light and open spaces, BrightSpace brings a fresh, airy feel to any room. Whether you're looking to brighten your living area, create an inspiring workspace, or transform your bedroom into a peaceful retreat, BrightSpace offers the perfect blend of style and function to make your environment feel open and inviting. Featuring clean lines, light hues, and functional designs, BrightSpace is all about creating a positive, uplifting atmosphere. Each piece is carefully crafted to enhance the flow of light, with materials that reflect brightness and encourage a sense of openness. The collection brings a modern minimalist approach that allows your space to feel more expansive, whether through large windows, reflective surfaces, or light furniture choices. Ideal for those who appreciate a modern, fresh aesthetic



Conclusion

This process successfully automated the migration of data from an external API to Sanity CMS. The Key steps included:

- √ Fetching Data from External APIs
- ✓ Comparing API data with Sanity Schema
- ✓ Data Migration Script
- ✓ Setting Up The Environment
- ✓ Modifying package.json
- ✓ Importing Data into Sanity
- ✓ Displaying data in frontend