Days-3

API Integration Report – [Furniro] By M.Anees

.

Furniro is a top-class furniture website dedicated to providing ease and convenience for both shopkeepers and customers.

For local sellers, Furniro offers a platform to reach a broader audience by leveraging the power of websites and the internet. This helps them showcase their furniture collections online, increasing visibility and sales opportunities.

On the other hand, customers can enjoy the convenience of browsing a wide variety of furniture options from the comfort of their homes. Furniro eliminates the hassle of visiting 30 to 50 shops by offering an extensive catalog, allowing users to compare prices, styles, and quality effortlessly.

Our goal is to bridge the gap between sellers and buyers, creating a seamless shopping experience that saves time, effort, and money.

Whether you're looking for budget-friendly options or premium furniture, Furniro is here to help you make the best choices without stepping out of your home.

Api Integration process

Here are the following steps that I use to integrate the given api data to my sanity .

To make my static website "Dynamic".

1) Setting up a new project in my sanity console with name "Fernero_marketplace"
2) Copy the given code from overview tab which include all setup command to make A new sanity project in Next js App .(eg . sanity –init , datasets and project id etc) Which we see later.

3) I configured my environment variables in my .env.local file I generate token from sanity console and paste it to my env.local file.
4) I check the api response using tool postman.
5) I make a folder and than I make a my importdata script .
6) I make my product schema acc to api response
7) Than I run my script command in terminal to import data from api to sanity.
8) Then I fetch my sanity data using query to integrate it to my products components.
9) Hurray !! my website become dynamic now
10) Finally my first step done . that I plan during day1 there are many challenges are waiting for me 😳.

Here are the little bit descripted form of my all steps of day 3 .
Enjoy!!!!!!!!!!!!!!!!!!!

Steps of API Integration in Next.js

1. Setting Up Sanity Studio

Run the command to initialize Sanity project in Next app:

npm create sanity@latest -- --project 57j2lf23 --dataset production --template clean

This creates a clean project and links it to your specific Sanity project ID and dataset (in this case, production).

2. Configuring Environment Variables

Add the following variables to your .env.local file

a) **SANITY_PROJECT_ID**: Unique identifier for your Sanity project.
b) **SANITY_DATASET**: The dataset name, such as production.
c) **SANITY_API_TOKEN**: An API token generated in the Sanity dashboard for

3. Verifying the API Response

Before starting integration, fetch data from the API in tool like postman that I'll be using to ensure it matches my schema.

4.Creating a Script for Data Migration

a) Create a folder named scripts in your project directory.

b) Inside this folder, create a file (e.g., dataImportingScript.mjs) to write the migration logic.

Word updated the entry for *Update when things change* from *page 2* to *page 3*.

5.Adjustment of Schema.

The schema in Sanity type was adjusted according to the API's response to ensure compatibility. This step guaranteed that the data would map correctly into the CMS.

6. Dynamic Data Fetching

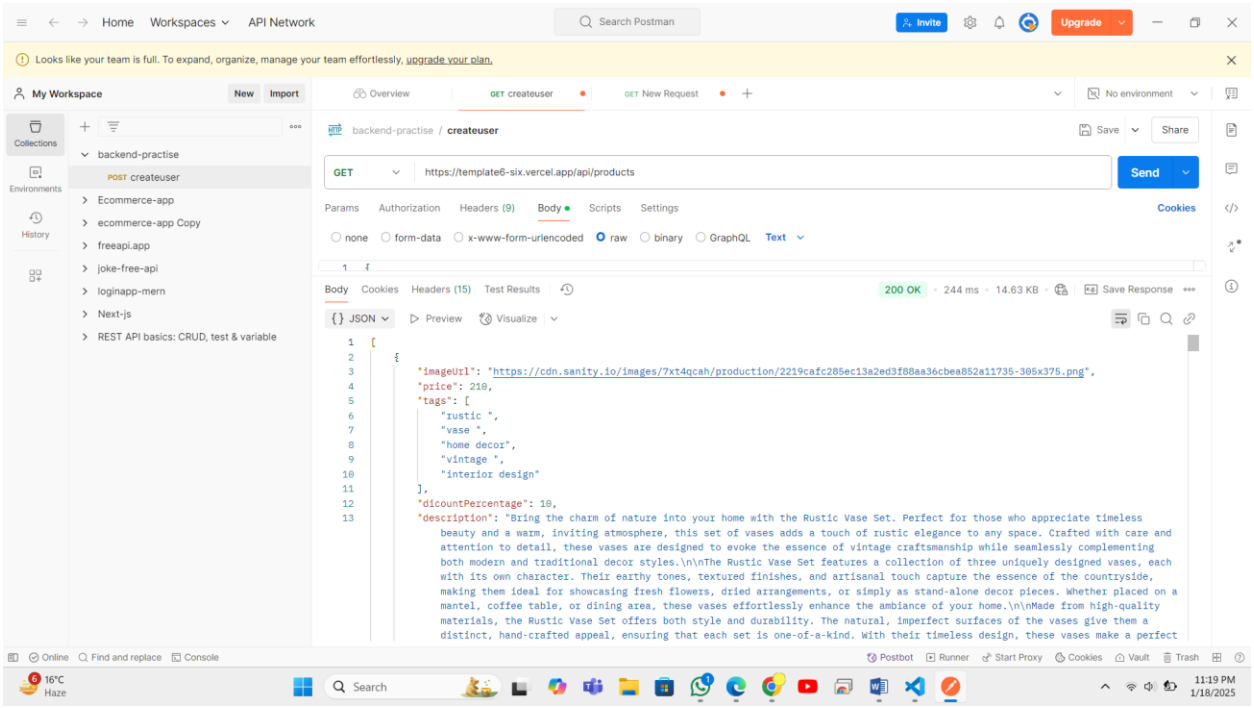
Once the data was successfully imported into Sanity CMS, a fetcher function was implemented in the frontend. This function retrieved data dynamically and passed it to multiple components to render dynamic content.

Adjustment to Schema

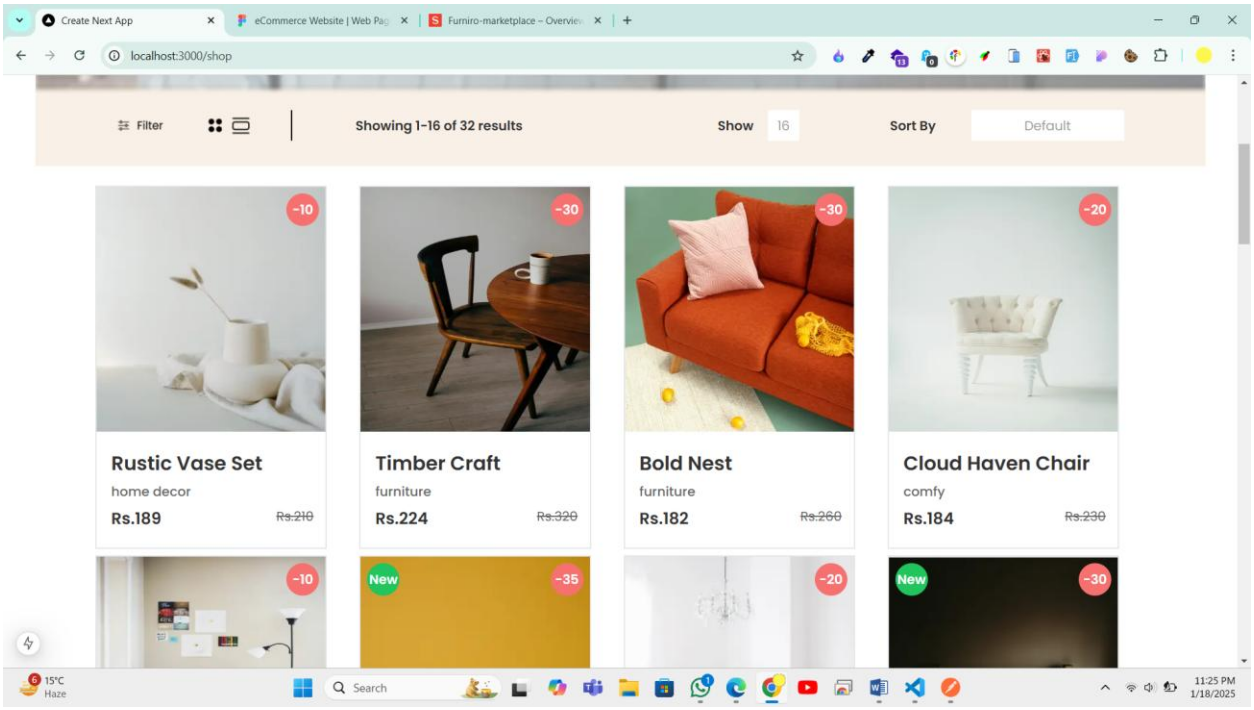
1) Change name to title for product name.
2) Change categories to tags
3) Make a new field called IsNew.

ScreenShot :-

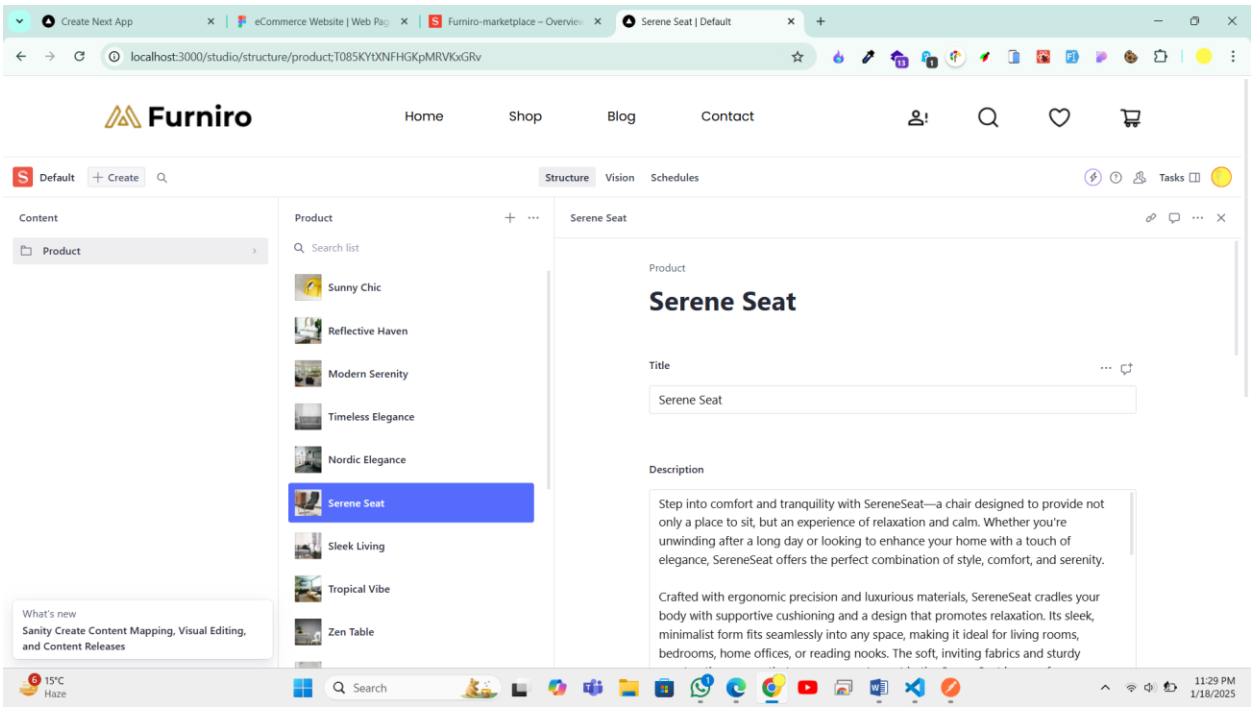
1) API Response using Postman



Data successfully displayed in the frontend.



Populated Sanity CMS fields.



Code snippets for API integration and migration scripts.

1) Migration Script

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

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

    // Create Sanity client
    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}
`)

        // Fetch image from the external URL
        const response = await axios.get(imageUrl
, { responseType: 'arraybuffer' })
        const buffer = Buffer.from(response.data)

        // Upload the image to Sanity
        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 products from API...'
)
      const response = await axios.get('
https://template6-six.vercel.app/api/products')
      const products = response.data

      console.log(`Fetched ${products.length}
 products`)

      for (const product of products) {
        console.log(`Processing product: ${
product.title}`)

        let imageRef = null
        if (product.imageUrl) {
          imageRef = await uploadImageToSanity(
product.imageUrl)
        } else {
          imageRef = null
          console.log('
No image found for product:', product.title)
        }

        // Prepare product data for Sanity
        const sanityProduct = {
          _type: 'product',
          title: product.title,
          description: product.description,
          price: product.price,
          discountPercentage: product.
dicountPercentage || 0,
          tags: product.tags || [],
          imageUrl: product.imageUrl,
          isNew: product.isNew || false,
          image: imageRef
            ? {
                _type: 'image',
                asset: {
                  _type: 'reference',
                  _ref: imageRef,
                },
              }
            : null,
        }

        console.log('Uploading product to Sanity:
', sanityProduct.title)
        console.log('
Uploading product image to Sanity:',
sanityProduct.imageUrl)


        const result = await client.create(
sanityProduct)

        console.log(`
Product uploaded successfully: ${result._id}`)
      }

      console.log('
Data import completed successfully!')
    } catch (error) {
      console.error('Error importing data:',
 error)
    }
  }

importData()
```

2) Fetcher Function :

```
import { client } from '@/sanity/lib/client'



    const res = async () => {
        const response = await client.fetch(`
*[_type == "product"]{
            _id,
            title,
            price,
            description,
            imageUrl,
            isNew,
            tags,
            discountPercentage
        }`)
        const data = response
        return data
        // console.log(data)
    }

  const data =  res()



export default data
```

3) Integration in Products Section Card Components as Prop;

```
"use client"
import { Button } from '@/components/ui/button'
import React, { useState } from 'react'
import CardProd from '
@/components/shop/prodcard/CardProd'
import ProductFetch from '../../shop/ProductFetch
'

const products = await ProductFetch

const Products = () => {
  const [visibleProducts, setVisibleProducts] =
useState(2);

  const productsPerRow = 4;
  const handleShowMore = () => {
    setVisibleProducts((prev) => prev + 2);
  };

  return (
    <section className='
 flex py-10 items-center justify-center  bg-white
'>
      <div className='
 w-full md:w-[1236px] flex flex-col items-center
justify-between py-1 bg-white
'>
        <h1 className='
text-[40px] font-bold text-myblack'>Our Products
</h1>
        <div className='
 w-[90%] md:w-[1236px] md:px-0 px-12 grid grid-co
ls-1 md:grid-cols-4 gap-3 mb-4 bg-white
'>
          <CardProd Product={products.slice(0,
visibleProducts * productsPerRow)} />
        </div>
        {visibleProducts * productsPerRow <
products.length && (
          <Button variant='products' size='sx'
onClick={handleShowMore}>
            Show More
          </Button>
        )}
      </div>
    </section>
  );
};

export default Products;
```

Self-Validation Checklist:

API Understanding:	Schema Validation:	Data Migration:	API Integration in Next.js:	Submission Preparation:
☑	☑	☑	☑	☑