DAY - 3 API Integration Report Of - E-Commerce Marketplace:

API Integration Process :

1. Introduction

This document outlines integrating product data from an external API into the system. The integration was implemented using **Template 4**, which provided a structured setup for API calls and data management.

2. Tools and Technologies Used

Tool	Purpose
Template 4	Pre-built structure for seamless integration.
Node.js	JavaScript runtime environment for scripting.
Axios	Fetch data from the external API.
dotenv	Securely manage API credentials and environment variables.
Postman (optional)	Test API responses and endpoints.

4. API Integration Steps

Step 1: Cloning Template 4 :

1. Cloned the Template 4 repository to set up the integration process:

```
git clone https://github.com/anasseth/next-ecommerce-templa
te-4.git
cd next-ecommerce-template-4
```

2. Installed the required dependencies:

```
npm install
```

3. Configured environment variables in the `.env` file:

```
NEXT_PUBLIC_SANITY_PROJECT_ID={your-project-id}
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN={your-sanity-api-token}
```

Step 2: Fetch Product Data :

- Used the API endpoint from Template 4 to fetch product data:

  API Endpoint: `https://next-ecommerce-template-4.vercel.app/api/product`

- Axios was used to make GET requests to the API and retrieve product data:

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

Step 3: Testing the API

- Verified API responses using Postman for completeness and consistency.
- Ensure that all required fields are available in the response.

Schema Adjustment :

My Schema :

```
export default {
 name: 'product',
 title: 'Product',
 type: 'document',
 fields: [
```

```
 { name: 'productId', title: 'Product ID', type: 'string' },
 { name: 'name', title: 'Product Name', type: 'string' },
 { name: 'price', title: 'Price', type: 'number' },
 { name: 'tags', title: 'Tags', type:
 'array', of: [{ type: 'string' }] },
 { name: 'sizes', title: 'Sizes', type:
 'array', of: [{ type: 'string' }] },
 { name: 'color', title: 'Color Options',
 type: 'array', of: [{ type: 'string' }] },
 { name: 'dimension', title: 'Dimensions',
 type: 'object',
 fields: [
 { name: 'length', title: 'Length', type: 'number' },
  { name: 'width', title: 'Width', type: 'number' },
  { name: 'height', title: 'Height', type: 'number' },
  { name: 'unit', title: 'Unit', type: 'string',
   options: { list: ['inches', 'cm', 'mm'] }}
    ]
 },
    { name: 'stock', title: 'Stock Quantity', type: 'number'
```

Adjusted Schema :

```
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: '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'),
},
],
};
```

Migration steps and tools used :

1. Setting Up Sanity Client :

- Before we can interact with Sanity, we need to initialize the Sanity client by providing the necessary project credentials such as **Project ID**, **Dataset**, and **API Token**.
- This is accomplished using the `createClient` function from the `@sanity/client` library.

- The `createClient` function in the script establishes a connection between your application and **Sanity's API**. By providing **Sanity project credentials** such as the **Project ID**, **Dataset**, and **API Token**, you authenticate and configure the client to interact with your specific Sanity project.

```
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,   // 
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,       // Sa
  token: process.env.SANITY_API_TOKEN,                   // Sa
  apiVersion: '2025-01-15',                              // Sa
  useCdn: false,                                         // T
});
```

2. Setting Up Environment Variables:

- We load environment variables using the `dotenv` package. This ensures that sensitive information like API tokens and project IDs are not hard-coded and can be kept secure.

```
dotenv.config({ path: path.resolve(__dirname, '../../.env')
});
```

3. Fetching Data from External API :

As I mentioned above we fetch data by integrating a external API , we connect the external API end point and make a get request using axios.get() In this case, the data is fetched from a product API that returns information such as product names, descriptions, prices, and image paths etc .

4. Processing and Structuring Data :

- After fetching the product data, we loop through each product, process it, and prepare it for insertion into Sanity.

- For each product, we check if there's an associated image. If there is, then we pass the image path to the function uploadImageToSanity() function. then this function process to add the Image

Image Function :

```
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;
  }
}
```

Structuring the Data return from the API :

```
for (const item of products) {
  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,
  };
}
```

NOTE: Ensure that the **keys** in the `sanityItem` object exactly match the **field names** defined in your **Sanity schema**. These field names in the schema are like "variables" that your data will be inserted into.

Example :

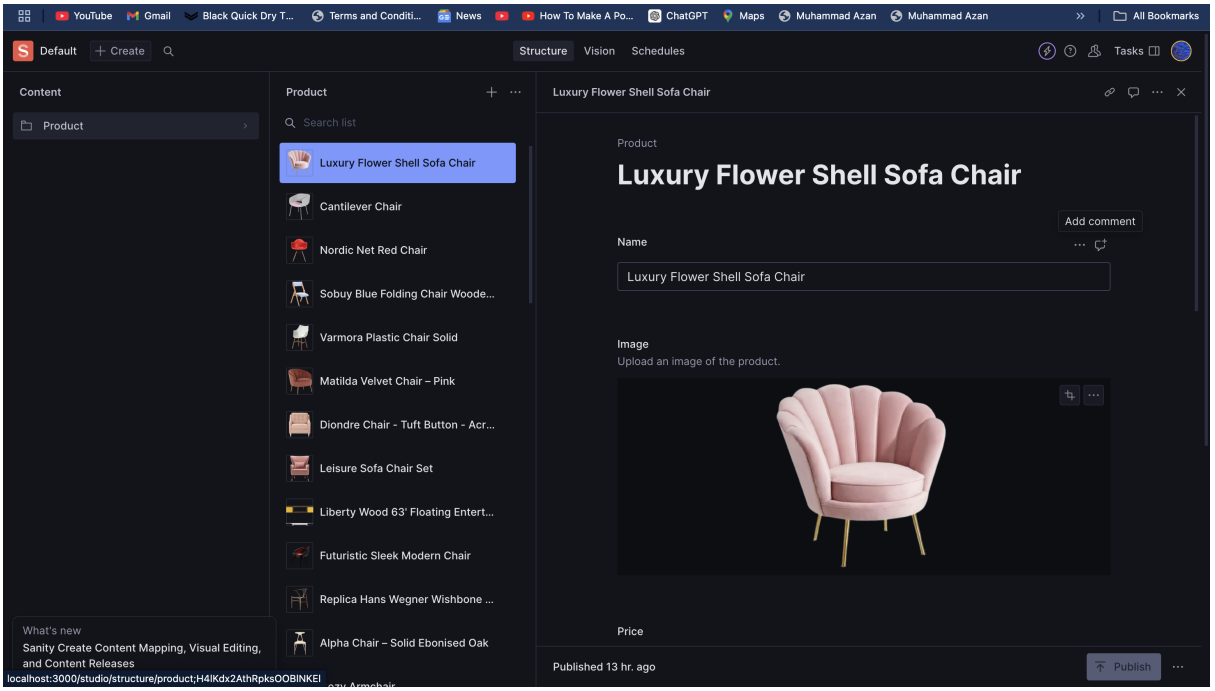
```
export default {
name: 'product',
title: 'Product',
type: 'document',
fields: [

{
name: 'price', // the value of the field will be saved in the
title: 'Price',
type: 'number',
},

]
```

5. Inserting Data into Sanity :

Once the data is structured, we use `client.create()` to insert each product into Sanity. This function creates a new document in the specified dataset.

```
const result = await client.create(sanityItem);
console.log(`Uploaded Successfully: ${result._id}`);
```

Populated Sanity CMS fields :



Data successfully displayed in the frontend :

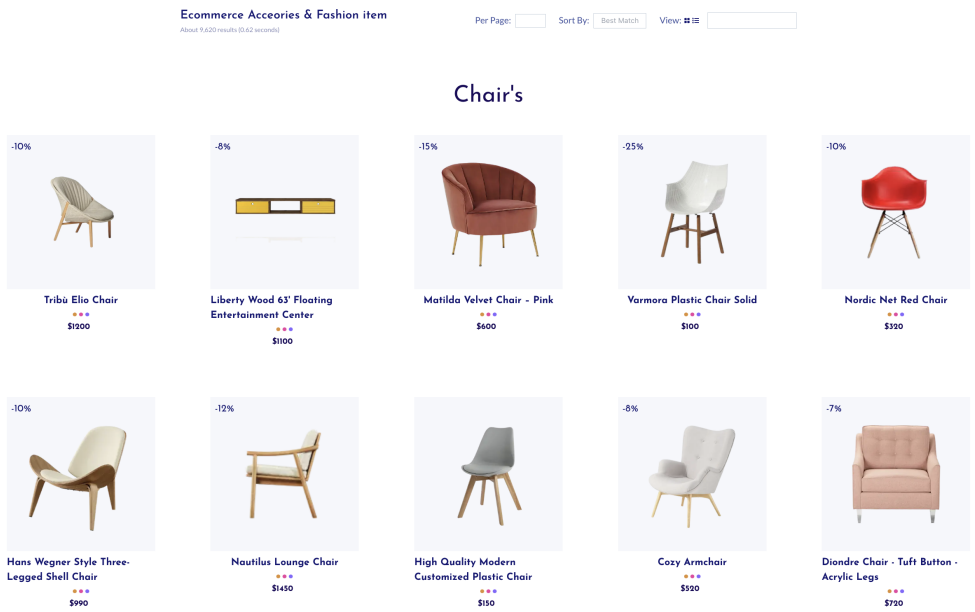

Ecommerce Acceories & Fashion item
About 9,620 results (0.62 seconds)
Per Page:
Sort By:
Best Match
View:
Chair's
-10%
Tribù Elio Chair
$1200
-8%
Liberty Wood 63' Floating Entertainment Center
$1100
-15%
Matilda Velvet Chair - Pink
$600
-25%
Varmora Plastic Chair Solid
$100
-10%
Nordic Net Red Chair
$320
-10%
Hans Wegner Style Three-Legged Shell Chair
$990
-12%
Nautilus Lounge Chair
$1450
High Quality Modern Customized Plastic Chair
$150
-8%
Cozy Armchair
$520
-7%
Diondre Chair - Tuft Button - Acrylic Legs
$720

IMAGE 2 :


Sofa's
-18%
Leisure Sofa Chair Set
$1800 $2000
Luxury Flower Shell Sofa Chair
$2500
Stylish Golden Metal Legs Mint Blu...
$780