

"Day 3 - API Integration Report - General E-Commerce"

By Aqsa Saeed

API Integration Process

1. Sanity CMS Integration:

- Configured the Sanity CMS schema to include custom fields such as:
 - Tags: To categorize products.
 - Ingredients: For detailed product information.
 - Discounted Price: To display promotional offers.
- Used Sanity Studio to add data manually to these fields.

2. Fetching Data from Sanity:

- Integrated Sanity's API in the project using its client library.
- Created API calls to fetch data from Sanity, ensuring all fields (tags, ingredients, discounted price) were retrieved.

3. Category-Based Filtering:

- Implemented a category-based filtering feature on the frontend.
 - Used checkboxes to allow users to select categories, dynamically filtering and displaying relevant data on the UI.
-

Adjustments Made to Schemas

Overview

The following fields were added or modified in the Sanity schema:

- Tags:** To categorize products (e.g., "Spicy," "Vegan").
- Ingredients:** A list of ingredients for each product.
- Discounted Price:** To display promotional offers or sales.

Updated Sanity Schema

- **Product Schema:**

```
3 export const foodProductSchema = defineType({
4   name: 'Food Product',
5   title: 'Food Product',
6   type: 'document',
7   fields: [
8     defineField({
9       name: 'name',
10      title: 'Name',
11      type: 'string',
12      validation: (Rule) => Rule.required().min(1),
13    }),
14    defineField({
15      name: 'slug',
16      title: 'Slug',
17      type: 'slug',
18      options: {
19        source: 'name',
20        maxLength: 96,
21      },
22      validation: (Rule) => Rule.required(),
23    }),
24    defineField([
25      name: 'description',
26      title: 'Description',
27      type: 'array',
28      of: [{ type: 'block' }],
29    ]),
30    defineField({
31      name: 'price',
32      title: 'Price',
33      type: 'number',
34      validation: (Rule) => Rule.required().min(0),
35    }),
36    defineField({
37      name: 'discountPrice',
38      title: 'Discount Price',
39      type: 'number',
40      description: 'The discounted price of the product',
41      validation: (Rule) =>
42        Rule.min(0)
43          .precision(2)
44          .custom((discountPrice, context) => {
45            const document = context.document
46            if (!discountPrice && document.price)
47              return 'Discount price should be greater than or equal to the original price'
48            else
49              return true;
50          }),
51        }),
52        defineField({
53          name: 'isDiscounted',
54          title: 'Is Discounted?',
55          type: 'boolean',
56          description: 'Indicates if the product has a discounted price',
57          initialValue: false,
58        }),
59        defineField({
60          name: 'category',
61          title: 'Category',
62          type: 'string',
63        }),
64        defineField({
65          name: 'images',
66          title: 'Images',
67          type: 'array',
68          of: [
69            {
70              type: 'image',
71              options: {
72                hotspot: true,
73              },
74            },
75          ],
76        }),
77      ],
78    ),
79    defineField({
80      name: 'tags',
81      type: 'array',
82      title: 'Tags',
83      of: [{ type: 'string' }],
84      options: {
85        layout: 'tags',
86      },
87      description: 'Tags for categorization (e.g., Spicy, Vegan)'
88    }),
89    defineField({
90      name: 'ingredients',
91      title: 'Ingredients',
92      type: 'array',
93      of: [{ type: 'string' }],
94      validation: (Rule) => Rule.unique(),
95    }),
96    defineField({
97      name: 'nutritionalInfo',
98      title: 'Nutritional Information',
99      type: 'object',
100     fields: [
101       defineField({ name: 'calories', title: 'Calories' }),
102       defineField({ name: 'protein', title: 'Protein' }),
103       defineField({ name: 'fat', title: 'Fat' }),
104       defineField({ name: 'carbs', title: 'Carbohydrates' }),
105     ],
106     defineField({
107       name: 'available',
108       title: 'Available',
109       type: 'boolean',
110       initialValue: true,
111     }),
112   ],
113 })
```

Migration Steps and Tools Used

1. Manual Data Entry:

- Populated the new fields in Sanity Studio.
- Verified the data for consistency and correctness.

2. Data Migration to Project:

- Fetched data using API calls.
- Transformed the fetched data (if needed) to match the frontend requirements.
- Ensured that all new fields (tags, ingredients, discounted price) were properly displayed on the UI.

Screenshots

1. API Calls:

- Screenshot of successful API responses from Sanity.

```
{  
  name: 'Coffee Mocha',  
  discountPrice: 13,  
  _id: '03aeda28-0404-4f99-8d5d-1def71dcbe6e',  
  slug: { current: 'coffee-mocha', _type: 'slug' },  
  price: 18,  
  ingredients: [ 'Espresso', 'Chocolate', 'Steamed Milk', 'Whipped cream' ],  
  nutritionalInfo: { fat: 25, calories: 280, carbs: 46, protein: 20 },  
  category: 'drinks',  
  isDiscounted: true,  
  images: [ [Object], [Object], [Object] ],  
  description: [ [Object] ]  
},  
{  
  name: 'Spaghetti Carbonara',  
  slug: { current: 'spaghetti-carbonara', _type: 'slug' },  
  nutritionalInfo: { carbs: 55, protein: 25, fat: 29, calories: 600 },  

```

2. Frontend Display:

- Screenshots showing filtered data based on selected categories.



Bruschetta
\$12.00



Dumplings
\$18.00



Mediterranean Power Salad
\$13.00 ~~\$20.00~~

Hot Offer

Search Product

Category

Starter

Main

Desserts

Drinks

Burger

All





Coffee Mocha
\$13.00 ~~\$10.00~~



Chocolate Banana Smoothie
\$9.00 ~~\$13.00~~



Strawberry Margarita
\$45.00

Hot Offer

Hot Offer

Search Product

Category

Starter

Main

Desserts

Drinks

Burger

All



-
- Screenshots displaying product details.



Coffee Mocha

A coffee mocha, often referred to as a **mocha latte** or simply **mocha**, is a rich, indulgent coffee beverage that combines the bold flavor of espresso with the smoothness of steamed milk and the sweetness of chocolate. It's a popular drink at coffee shops worldwide, offering a delightful balance of bitterness from the coffee, sweetness from the chocolate, and creaminess from the milk. Mocha is a versatile drink that can be enjoyed hot or iced, making it a year-round favorite for coffee lovers.

13.00\$ ~~18.00\$~~

- 1 + [Add to Cart](#)

Add to Wishlist Compare

Category: drinks

[Description](#) [Reviews](#)

A coffee mocha, often referred to as a **mocha latte** or simply **mocha**, is a rich, indulgent coffee beverage that combines the bold flavor of espresso with the smoothness of steamed milk and the sweetness of chocolate. It's a popular drink at coffee shops worldwide, offering a delightful balance of bitterness from the coffee, sweetness from the chocolate, and creaminess from the milk. Mocha is a versatile drink that can be enjoyed hot or iced, making it a year-round favorite for coffee lovers.

Ingredients:

Espresso
Chocolate
Steamed Milk
Whipped cream

3. Sanity CMS Fields:

- Screenshots of the populated Tags, Ingredients, and Discounted Price fields in Sanity Studio.

The screenshot shows the Sanity Studio interface for a document titled "Chocolate Banana Smoothie".

Tags
Tags for categorization (e.g., Best Seller, Popular, New)
Shakes × Smoothie × Popular ×

Ingredients

- ⋮ Bananas
- ⋮ Cocoa Powder
- ⋮ Milk
- ⋮ Ice
- ⋮ Peanut Butter

+ Add item

Published last week Edited just now

Publish

The screenshot shows the Sanity Studio interface for a document titled "Chocolate Banana Smoothie".

Price
13

Discount Price
The discounted price of the product, if applicable.
9

Is Discounted?
Indicates if the product has a discount applied.

Code Snippets

1. API Integration:

```
1 import { client } from "@/sanity/lib/client"
2
3 export const allProductsQuery = `*[ _type == "foodProduct"]{
4   _id,
5   name,
6   slug,
7   price,
8   ingredients[],
9   nutritionalInfo,
10  discountPrice,
11  category,
12  isDiscounted,
13  images[] { asset→ { _id, url } },
14  description,
15}
16`
17
18 export const oneProduct = `*[ _type == "foodProduct" && slug.current == ${params.slug} ]{
19   _id,
20   name,
21   slug,
22   price,
23   ingredients[],
24   nutritionalInfo,
25   discountPrice,
26   category,
27   isDiscounted,
28   images[] { asset→ { _id, url } },
29   description,
30}
31`
```

```
32
33 export const menuProducts = async (category:string) => {
34
35   const query = `*[ _type == "foodProduct" && category == "${category}" ]{
36     _id,
37     name,
38     slug,
39     price,
40     ingredients[],
41     nutritionalInfo,
42     discountPrice,
43     category,
44     isDiscounted,
45     images[] { asset→ { _id, url } },
46     description,
47   }`
48
49   const products = await client.fetch(query);
50   return products;
51 }
```

```
const products = await client.fetch<FoodProduct[]>(allProductsQuery)
console.log(products);
```

```
const product = await client.fetch(oneProduct, { slug: params.slug })
```

2. Data Migration:

```
const filteredProducts = products.filter((product) => {
  const inCategory = selectedCategory === 'all' || product.category === selectedCategory;
  const inPriceRange = product.price >= priceRange[0] && product.price <= priceRange[1];
  return inCategory && inPriceRange;
});
```

3. Frontend Filtering:

```
<div className="flex-1">
  <div className="grid grid-cols-1 sm:grid-cols-2 xl:grid-cols-3 gap-4 sm:gap-6">
    {filteredProducts.map((product) =>
      <Link
        href={`/products/${product.slug.current}`}
        key={product._id}
      >
        <div className="group relative bg-white rounded-lg overflow-hidden shadow-md hover:shadow-xl transition-shadow duration-300">
          <div className="w-full aspect-square overflow-hidden">
            {product.images && product.images[0] && (
              <Image
                src={urlFor(product.images[0]).url() || ""}
                alt={product.name}
                height={400}
                width={400}
                className="w-full h-full object-cover object-center group-hover:scale-105 transition-transform duration-300"
              />
            )}
          </div>
          <div className="p-4">
            {product.isDiscounted && (
              <div className="absolute top-2 left-2 bg-red-500 inter text-white text-xs font-bold px-2 py-1 rounded">
                Hot Offer
              </div>
            )}
            <h3 className="text-lg lg:text-sm xl:text-lg font-semibold text-gray-900 openSans">
              {product.name}
            </h3>
          </div>
        </div>
      </Link>
    )}
  </div>
```

```
<h3 className="text-lg lg:text-sm xl:text-lg font-semibold text-gray-900 openSans">
  {product.name}
</h3>
<div className="flex flex-col sm:flex-row lg:flex-col xl:flex-row justify-between items-start sm:items-center gap-2 mt-2">
  <div>
    {product.isDiscounted && product.discountPrice ? (
      <div className="flex gap-1">
        <p className="text-green-600 font-bold inter">
          ${product.discountPrice.toFixed(2)}
        </p>
        <p className="text-gray-400 line-through">
          ${product.price.toFixed(2)}
        </p>
      </div>
    ) : (
      <p className="text-primYellow font-bold inter">
        ${product.price.toFixed(2)}
      </p>
    )}
  </div>
</div>
</div>
</Link>
))}>
</div>
</div>
```