"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

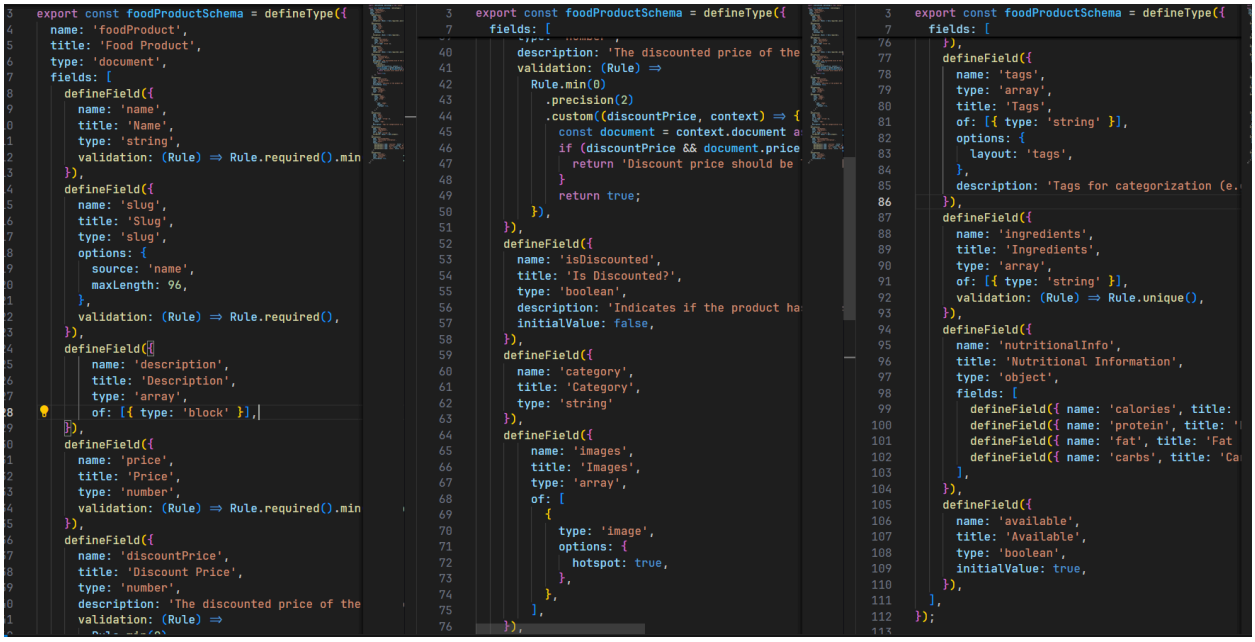
Overview

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

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

Updated Sanity Schema

- **Product Schema:**



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

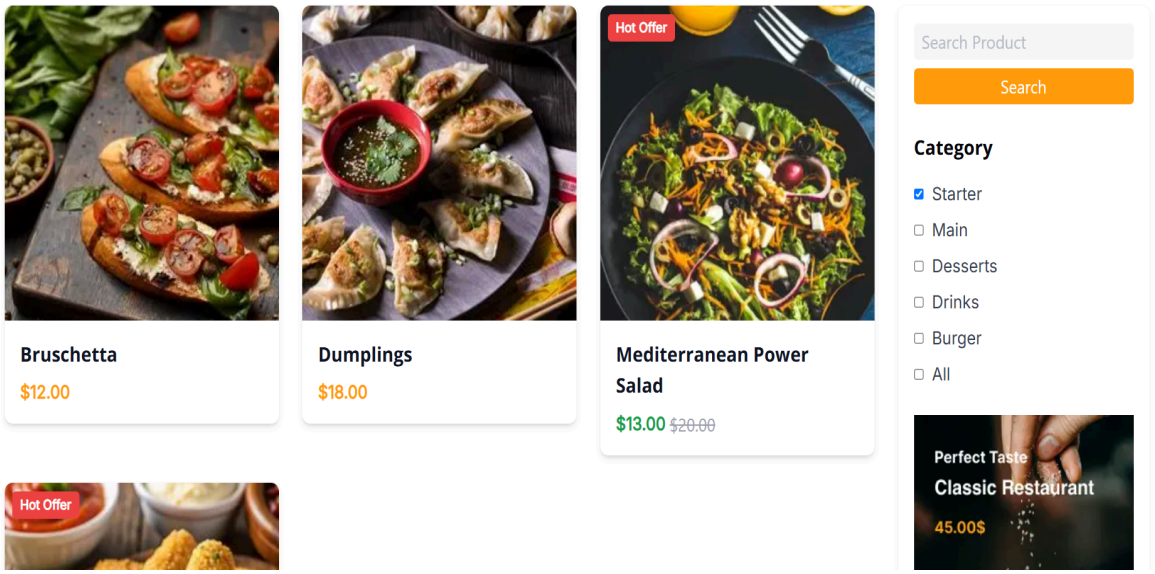
1. API Calls:

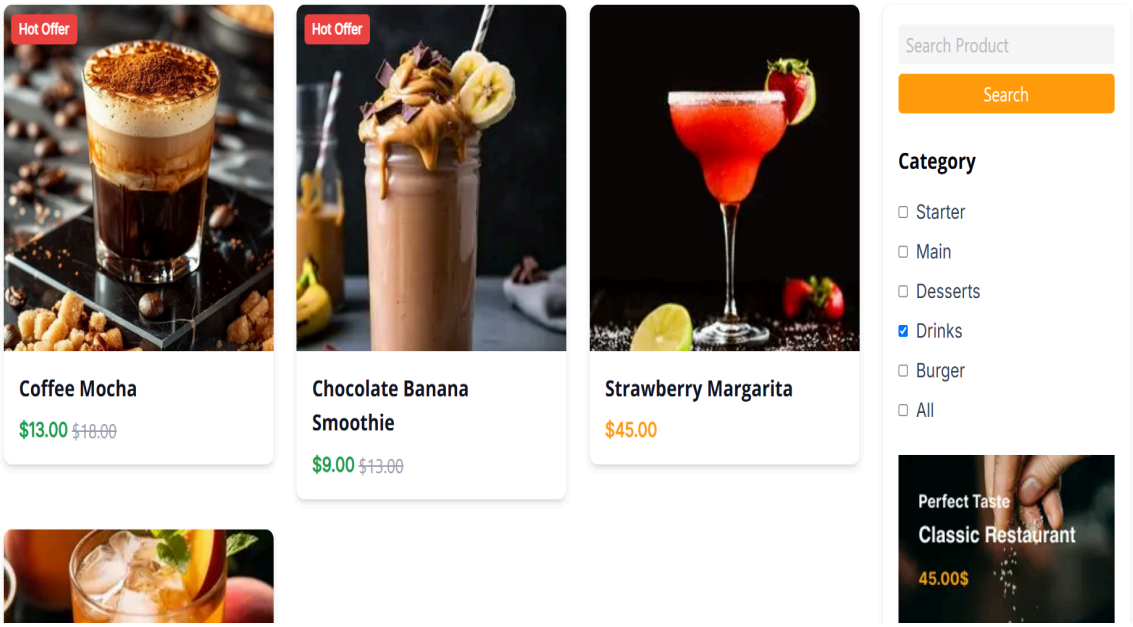
- Screenshot of successful API responses from Sanity.

```
{
  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 },
  isDiscounted: false,
  images: [ [Object], [Object], [Object] ],
  _id: '0bf33394-76a1-4dd6-b13f-31d87e7c8632',
  price: 18,
  ingredients: [ 'Spaghetti', 'Pancetta', 'Eggs', 'Pecorino romano', 'Olive Oil' ],
  discountPrice: null,
  category: 'main',
  description: [ [Object] ]
},
{
  name: 'Chocolate Banana Smoothie',
  price: 13,
  nutritionalInfo: { fat: 9, calories: 280, carbs: 40, protein: 25 },
  category: 'drinks',
  images: [ [Object], [Object], [Object] ],
  description: [ [Object] ],
  _id: '0bffd6b9-f28b-4785-b382-85984744e0e2',
  slug: { current: 'chocolate-banana-smoothie', _type: 'slug' },
  ingredients: [ 'Bananas', 'Cocoa Powder', 'Milk', 'Ice', 'Peanut Butter' ],
  discountPrice: 9,
  isDiscounted: true
},
```

2. Frontend Display:

- Screenshots showing filtered data based on selected categories.





- Screenshots displaying product details.

Coffee Mocha

A **coffee mocha**, often referred to as a **mocha latte** or simply **mocha**, rich, indulgent coffee beverage that combines the bold flavor of espr with the smoothness of steamed milk and the sweetness of chocolat a popular drink at coffee shops worldwide, offering a delightful balar bitterness from the coffee, sweetness from the chocolate, and creaminess from the milk. Mocha is a versatile drink that can be enjc 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

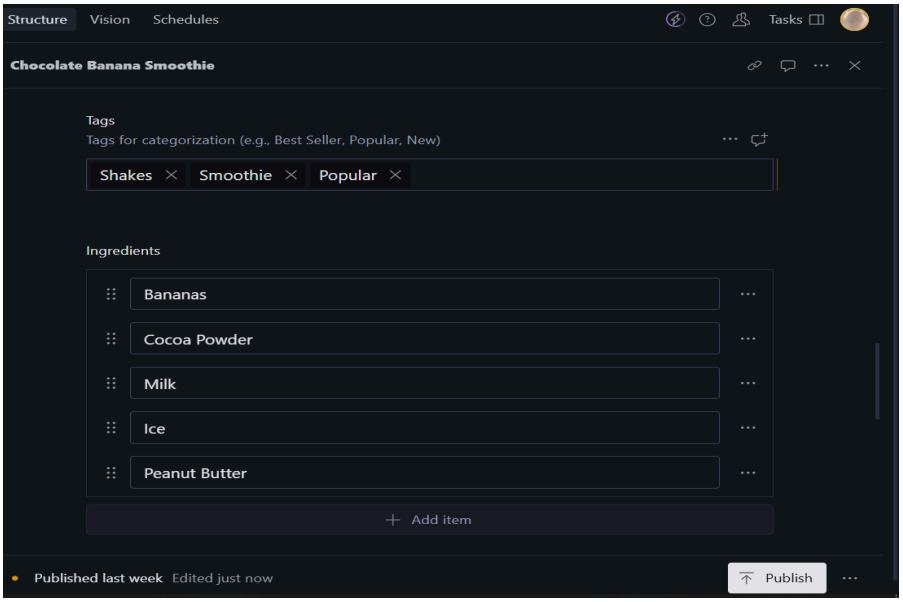
A **coffee mocha**, often referred to as a **mocha latte** or simply **mocha**, is a rich, indulgent coffee beverage that combines the bold flavor of espresso with the smoothness of steamed milk and the sweetness of chocolate. It's a popular drink at coffee shops worldwide, offering a delightful balance 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.

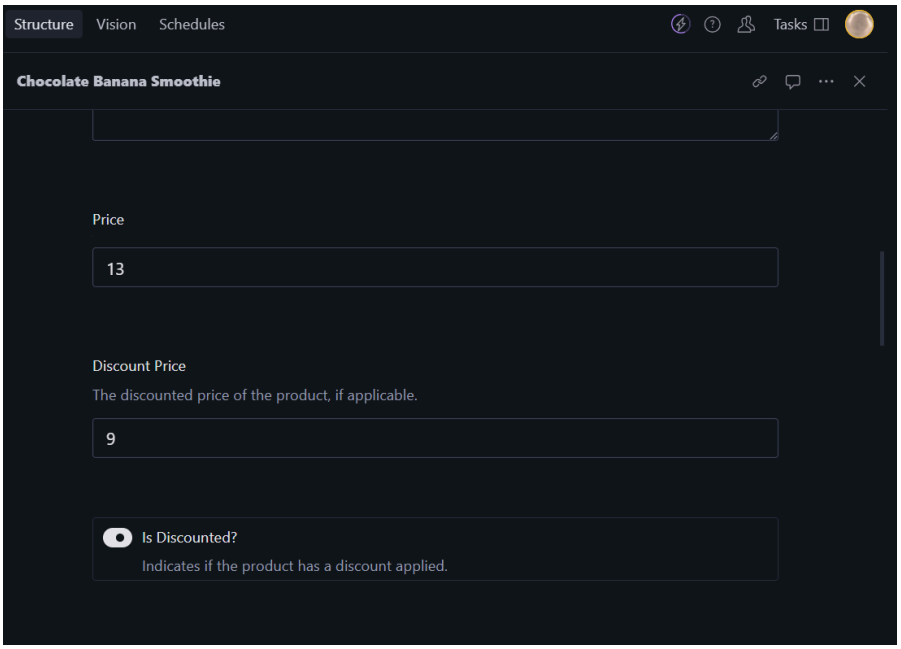
Ingredients:

Espresso
Chocolate
Steamed Milk
Whipped cream

3. Sanity CMS Fields:

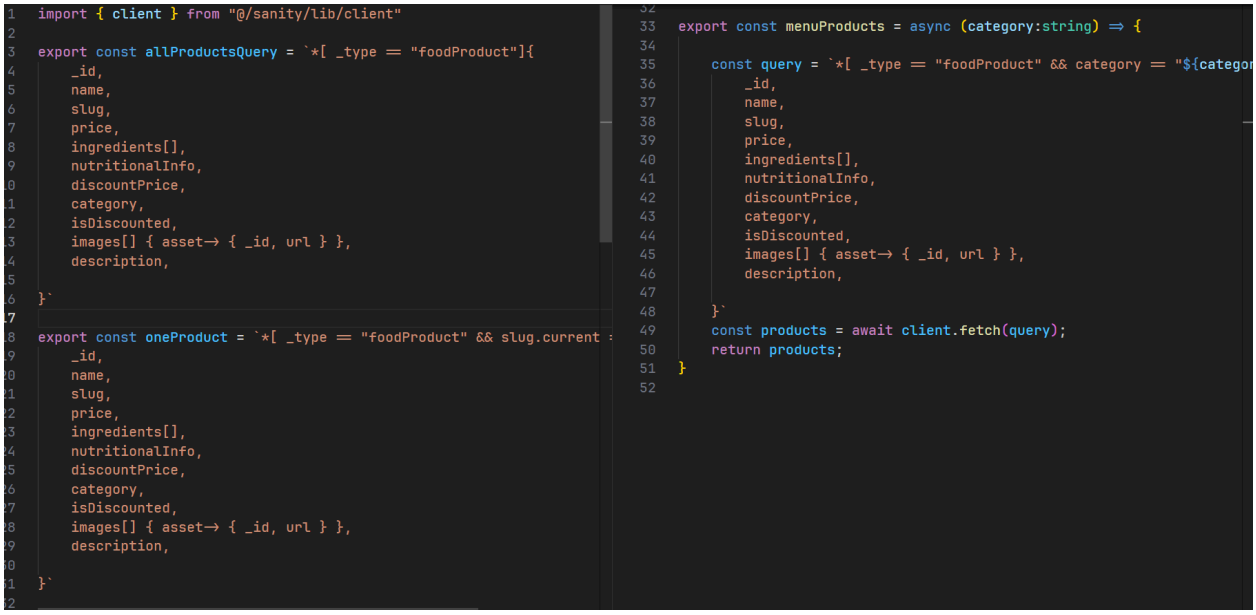
- Screenshots of the populated Tags, Ingredients, and Discounted Price fields in Sanity Studio.





Code Snippets

1. API Integration:



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

export const allProductsQuery = `*[ _type == "foodProduct"]{
    _id,
    name,
    slug,
    price,
    ingredients[],
    nutritionalInfo,
    discountPrice,
    category,
    isDiscounted,
    images[] { asset-> { _id, url } },
    description,

}`

export const oneProduct = `*[ _type == "foodProduct" && slug.current
    _id,
    name,
    slug,
    price,
    ingredients[],
    nutritionalInfo,
    discountPrice,
    category,
    isDiscounted,
    images[] { asset-> { _id, url } },
    description,

}`
```

```
export const menuProducts = async (category:string) => {

    const query = `*[ _type == "foodProduct" && category == "${categor
        _id,
        name,
        slug,
        price,
        ingredients[],
        nutritionalInfo,
        discountPrice,
        category,
        isDiscounted,
        images[] { asset-> { _id, url } },
        description,

    }`
    const products = await client.fetch(query);
    return products;
}
```

```
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>
```

```
            <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>
        </div>
      </Link>
    ))}
  </div>
</div>
```