Document Title: "Day 3 - API Integration Report -Q-COMMERCE (FoodTuck)"

1. Introduction

This report outlines the activities and progress made on Day 3 of the project, specifically focusing on API Integration and Data Migration for the given template 9. The objective of this phase was to successfully integrate external APIs, adjust backend schemas, and migrate data into the Sanity CMS for further use on the frontend.

2. API Integration Process

2.1 API Selection

The first step in the API integration process involved selecting appropriate external APIs to fetch essential marketplace data. The key APIs chosen were responsible for retrieving information such as:

2.2 Setting up API Clients

The integration began with setting up API calls using the fetch() method. The calls were made asynchronous to ensure the smooth execution of the application without blocking other processes. Below is an example of the basic structure used for fetching API data: Name, Category, price, Description and inventory of Item.

2.3 Handling API Responses

Once the data was fetched from the external API, it was processed and stored in the application's state using React hooks (useState and useEffect). This allowed for dynamic updates to the frontend based on the fetched data.

2.4 Error Handling and Fallbacks

To ensure a smooth user experience, error handling was implemented. In case of any issues with the API call, fallback content was provided to maintain the integrity of the user interface.

3.No Adjustments Made to Schemas

Provided Schema is used to integrate the data fetched from the external API into the backend.

3.1 Frontend State Adjustment

In the frontend, I adjusted the state structure to store the fetched API data. The useState and useEffect hooks were used to handle the dynamic nature of the product data.
The following code snippet demonstrates the use of these hooks:

```
"use client";
import React, { useEffect, useState } from "react";
import Image from "next/image";
import Link from "next/link"; // Import Link from Next.js
import { Pagination, PaginationLink } from "@/components/ui/pagination";
import { Checkbox } from "@/components/ui/checkbox";
import SelectDemo from "@/components/Select";
import SearchBar from "@/components/SearchBar"; // Import the SearchBar component
import { client } from "@/sanity/lib/client";
import FoodCard from "@/components/FoodCard";
```

```
// Define the type for the food data from Sanity
type FoodItem = {
  _id: string;
  name: string;
  category: string;
  price: number;
  imageUrl: string;
};

const Shop = () => {
  const [foodItems, setFoodItems] = useState<FoodItem[]>([]);
  const [loading, setLoading] = useState(true);
  const [searchQuery, setSearchQuery] = useState("");
  const [selectedCategories, setSelectedCategories] = useState<string[]>([]);
  const [sortBy, setSortBy] = useState<string>("all"); // Default sort by "all"
  const [currentPage, setCurrentPage] = useState(1);
  const itemsPerPage = 9;

  // Fetch food items from Sanity
  useEffect(() => {
    const fetchFoodItems = async () => {
      const query = `*[_type == "food"]{
        _id,
        name,
        category,
        price,
        "imageUrl": image.asset->url
      }`;
      try {
        const data = await client.fetch(query);
        setFoodItems(data);
      } catch (error) {
        console.error("Failed to fetch food items:", error);
      } finally {
        setLoading(false);
      }
    };
    fetchFoodItems();
  }, []);

  // Sample categories (you can fetch these from Sanity as well)
  const categories = [
    "Sandwich",
    "Pizza",
    "Burger",
    "Drink",
    "Dessert",
    "Chicken Chup",
    "Salad",
    "Choclate Muffin",
  ];

  // Handle category selection
  const handleCategoryChange = (category: string) => {
    setSelectedCategories((prev) =>
      prev.includes(category)
        ? prev.filter((item) => item !== category) // Remove category if already selected
        : [...prev, category] // Add category if not selected
    );
    setCurrentPage(1); // Reset to first page when category changes
  };

  // Handle search query change
  const handleSearchChange = (query: string) => {
    setSearchQuery(query);
    setCurrentPage(1); // Reset to first page when search query changes
  };

  // Filter food items based on search query and selected categories
  const filteredItems = foodItems.filter((item) => {
    const matchesSearch = item.name.toLowerCase().includes(searchQuery.toLowerCase());
    const matchesCategory =
      selectedCategories.length === 0 || selectedCategories.includes(item.category);
    return matchesSearch && matchesCategory;
  });

  // Sort filtered items
  const sortedItems = [...filteredItems]; // Create a copy of filtered items
  if (sortBy === "name") {
    sortedItems.sort((a, b) => a.name.localeCompare(b.name)); // Sort by name
  } else if (sortBy === "price") {
    sortedItems.sort((a, b) => a.price - b.price); // Sort by price
  }
  // If sortBy is "all", no sorting is applied

  // Pagination logic
  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;
  const currentItems = sortedItems.slice(indexOfFirstItem, indexOfLastItem);

  // Change page
  const paginate = (pageNumber: number) => setCurrentPage(pageNumber);

  // Show loading state while data is being fetched
  if (loading) {
    return <div className="text-center py-10">Loading foods...</div>;
  }

  return (
    <div className="container mx-auto px-4 py-2 md:pt-16 flex flex-wrap lg:flex-nowrap">
```

```
      {/* Left Content */}
      <div className="w-full lg:w-2/3 lg:mr-10">
        {/* Sort and Show Options */}
        <div className="flex gap-5 ml-14 mb-6">
          <span>Sort By:</span>
          <SelectDemo
            options={[
              { value: "all", label: "All" }, // Default option
              { value: "name", label: "Name" },
              { value: "price", label: "Price" },
            ]}
            onSelect={(value: string) => setSortBy(value)}
          />
        </div>

        {/* Food List */}
        <div className="space-y-4 grid grid-col-1 sm:grid-cols-2 md:grid-cols-3 gap-6">
          {currentItems.map((item) => (
            <Link key={item._id} href={`/shop/${item._id}`}> {/* Wrap FoodCard with Link */}
              <a>
                <FoodCard item={item} />
              </a>
            </Link>
          ))}
        </div>

        {/* Pagination */}
        <Pagination className="mt-6">
          {Array.from({ length: Math.ceil(sortedItems.length / itemsPerPage) }, (_, i) => (
            <PaginationLink
              key={i + 1}
              onClick={() => paginate(i + 1)}
              isActive={currentPage === i + 1}
            >
              {i + 1}
            </PaginationLink>
          ))}
        </Pagination>
      </div>

      {/* Right Sidebar */}
      <div className="w-full lg:w-1/6 mt-10 lg:mt-0">
        {/* Search Bar */}
        <SearchBar onSearch={handleSearchChange} />

        {/* Category Filter */}
        <div className="border rounded-lg mt-8 p-6 space-y-2 text-base">
          <h1 className="text-lg font-bold mb-4">Category</h1>
          {categories.map((category, index) => (
            <div key={index} className="flex gap-4 items-center">
            <Checkbox
              checked={selectedCategories.includes(category)}
              onCheckedChange={() => handleCategoryChange(category)}
            />
            <span>{category}</span>
          </div>
        ))}
      </div>

      {/* Latest Products */}
      <div className="bg-white border rounded-lg mt-8 p-6">
        <h1 className="text-lg font-bold mb-4">Latest Products</h1>
        {[...Array(5)].map((_, index) => (
          <Image
            key={index}
            src={`/chicken${index + 1}.png`}
            alt=""
            width={311}
            height={62}
            className="w-full h-auto cursor-pointer mb-4"
          />
        ))}
      </div>

      {/* Product Tags */}
      <div className="bg-white border rounded-lg mt-8 p-6">
        <h1 className="font-bold text-xl mb-4">Product Tags</h1>
        <div className="flex flex-wrap gap-2">
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Services
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Our Menu
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Pizza
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Cupcake
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Burger
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Cookies
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
            Our Shop
          </span>
          <span className="bg-gray-200 px-3 py-1 rounded-full text-sm">
```

```
              Tandoori Chicken
            </span>
          </div>
        </div>
      </div>
    </div>
  );
};

export default Shop;
```

4. Data Migration Steps and Tools Used

4.1 Migration Plan

The process of migrating data from the external API to the Sanity CMS involved several steps:

Data Structure Review: A thorough review of the API data structure was done to ensure compatibility with the existing Sanity schema.
Data Population: A custom Node.js script was written to fetch the API data and insert it into the Sanity CMS using the Sanity client.

4.2 Tools Used

The following tools were used during the data migration process:

Sanity CLI: To manage and push data into the Sanity CMS.

Node.js Migration Script: A provided script was utilized to automate the process of fetching data from the API and migrating it to the CMS.

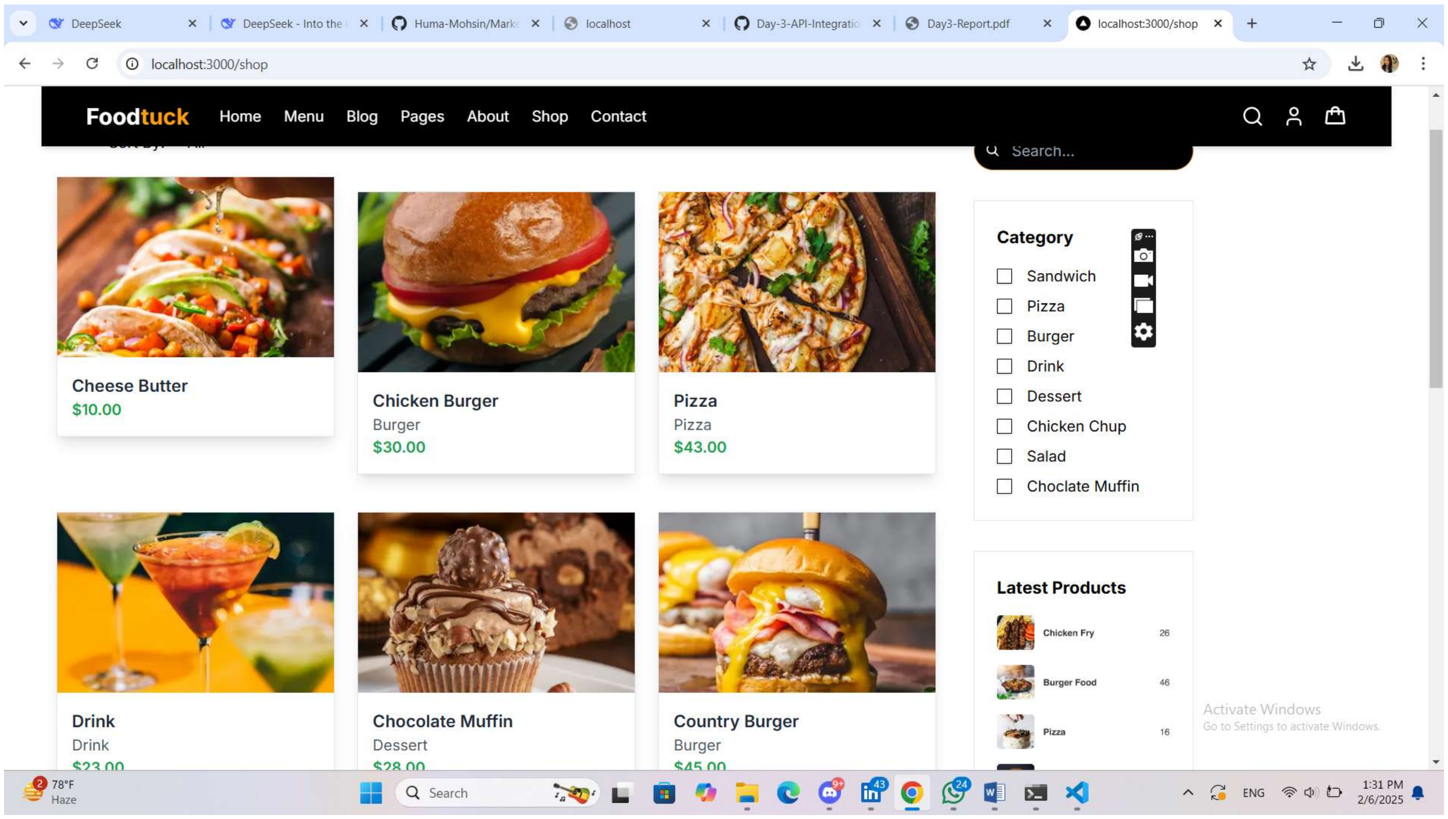
4.3 Verification of Data

Once the data migration was complete, I manually verified that the data had been correctly inserted into the Sanity CMS. This included confirming that all fields, such as product names, prices, and descriptions, were populated correctly.

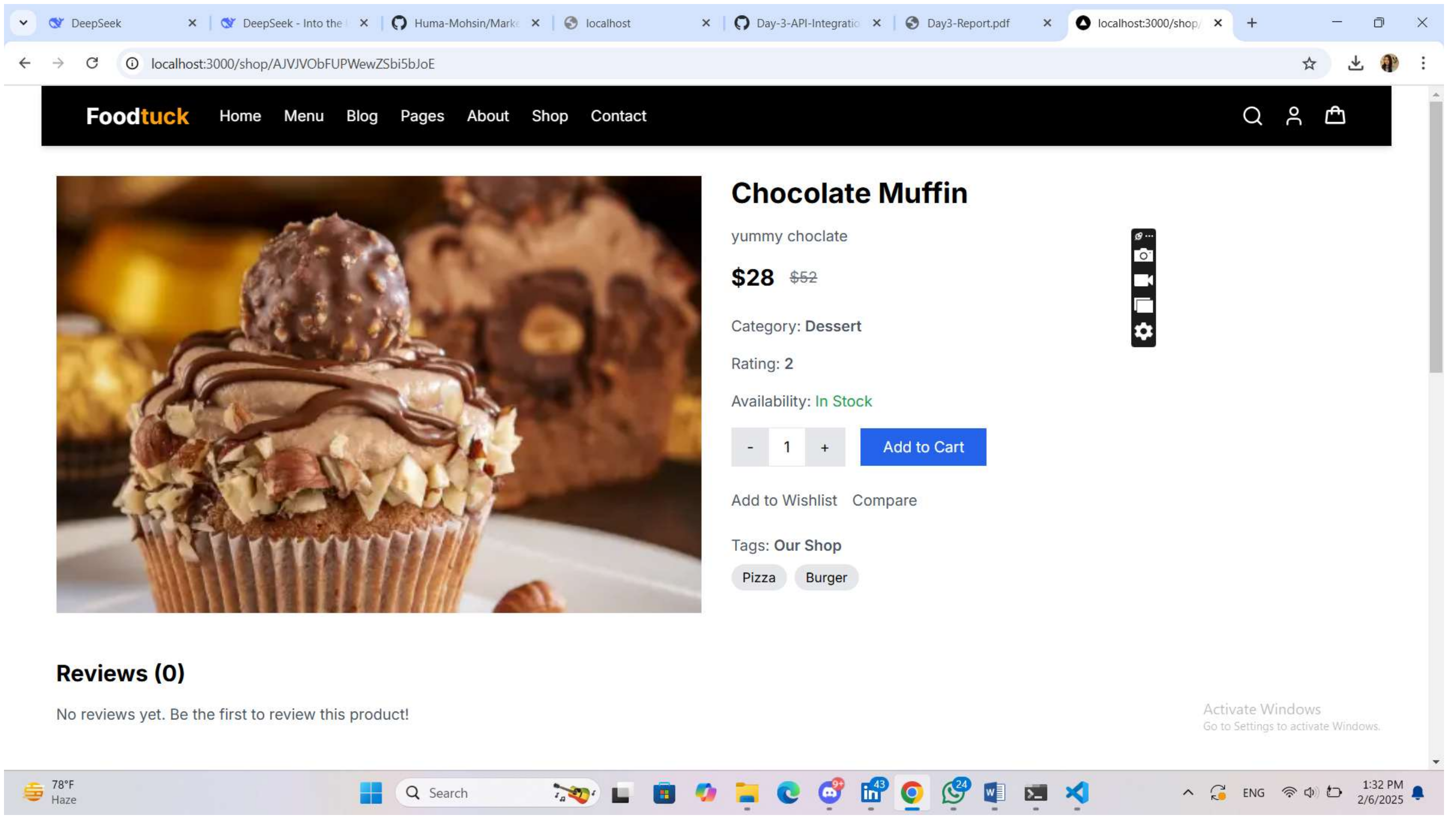
5. Screenshots

API Calls:

-Screenshot from the network tab showing the successful API call and response. Data Displayed in the Frontend:



-Screenshot showing the dynamic display of product data on the marketplace frontend. Populated Sanity CMS Fields:



Chocolate Muffin

yummy choclate

$28 ~~$52~~

Category: Dessert

Rating: 2

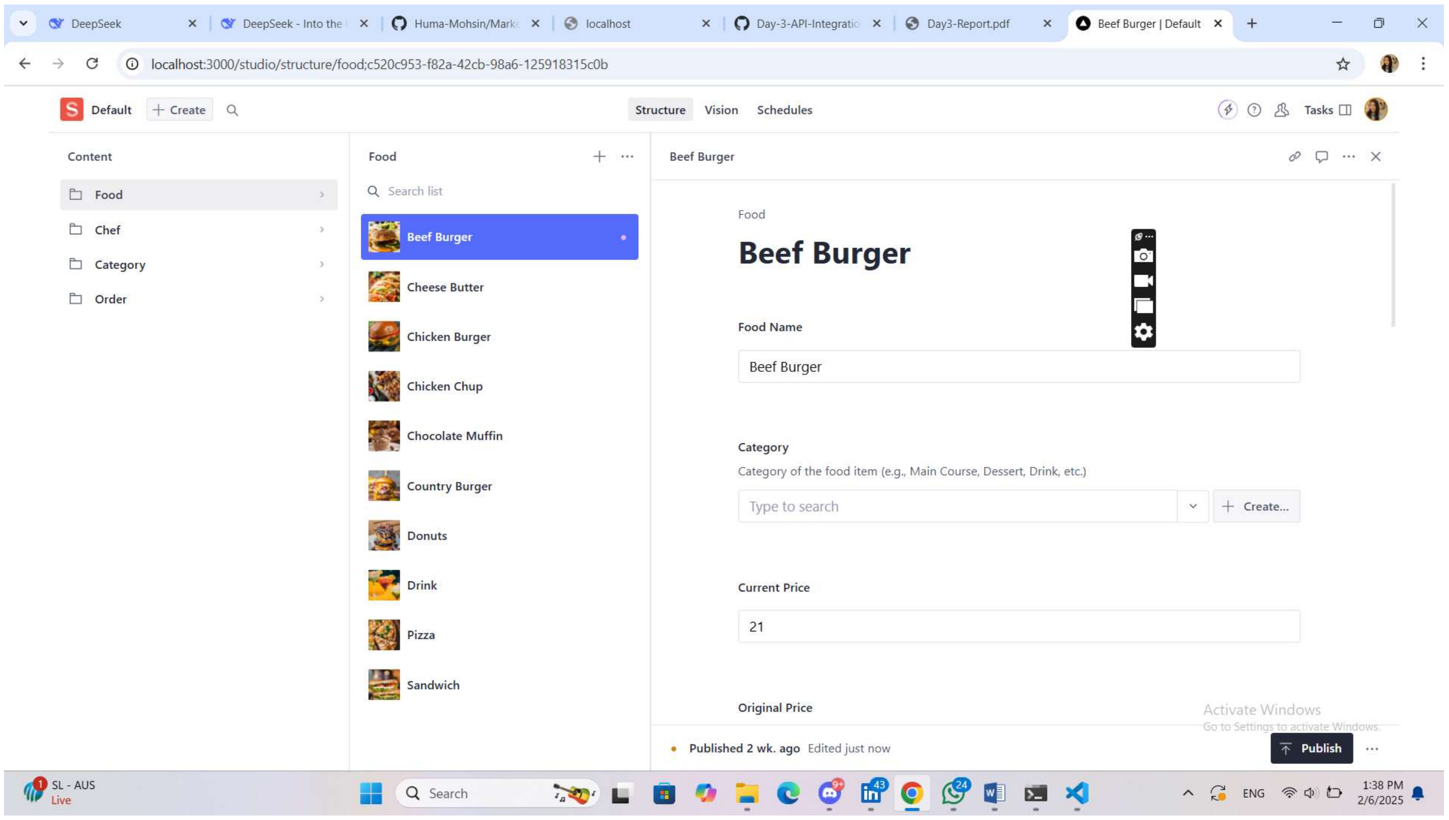
Availability: In Stock

Tags: Our Shop

Reviews (0)

No reviews yet. Be the first to review this product!

-Screenshot from the Sanity dashboard showing the populated fields for the Food Items document.


localhost:3000/studio/structure/food;c520c953-f82a-42cb-98a6-125918315c0b

Content: Food, Chef, Category, Order

Food: Beef Burger, Cheese Butter, Chicken Burger, Chicken Chup, Chocolate Muffin, Country Burger, Donuts, Drink, Pizza, Sandwich

Food

Beef Burger

Food Name: Beef Burger

Category
Category of the food item (e.g., Main Course, Dessert, Drink, etc.)
Type to search

Current Price: 21

Original Price

Published 2 wk. ago Edited just now

6. Code Snippets for API Integration and Migration Scripts

```javascript
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}`);
    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;
  }
}

async function checkIfFoodExists(name) {
  const query = `*[_type == "food" && name == $name][0]`;
  const params = { name };
  const existingFood = await client.fetch(query, params);
  return existingFood ? existingFood._id : null;
}

async function importData() {
  try {
    console.log('Fetching food data from API...');

    // API endpoint containing food data
    const foodsResponse = await axios.get('https://sanity-nextjs-
rouge.vercel.app/api/foods');
    const foods = foodsResponse.data;

    for (const food of foods) {
      console.log(`Processing food: ${food.name}`);

      // Check if food already exists
      const existingFoodId = await checkIfFoodExists(food.name);

      let imageRef = null;
      if (food.image) {
        imageRef = await uploadImageToSanity(food.image);
      }

      const sanityFood = {
        _type: 'food',
        _id: existingFoodId || undefined, // Use existing ID if available
        name: food.name,
        category: food.category || null,
        price: food.price,
        originalPrice: food.originalPrice || null,
        tags: food.tags || [],
        description: food.description || '',
        available: food.available !== undefined ? food.available : true,
        image: imageRef
```

```
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined,
      };

      console.log('Uploading food to Sanity:', sanityFood.name);
      const result = await client.createOrReplace(sanityFood);
      console.log(`Food processed successfully: ${result._id}`);
    }

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

importData();
```

7. Conclusion

The API integration and data migration process for Day 3 was successfully completed, with all required data fetched from the external API and migrated into the Sanity CMS. The frontend was updated to dynamically display the product information, and the system was thoroughly tested to ensure smooth functionality.