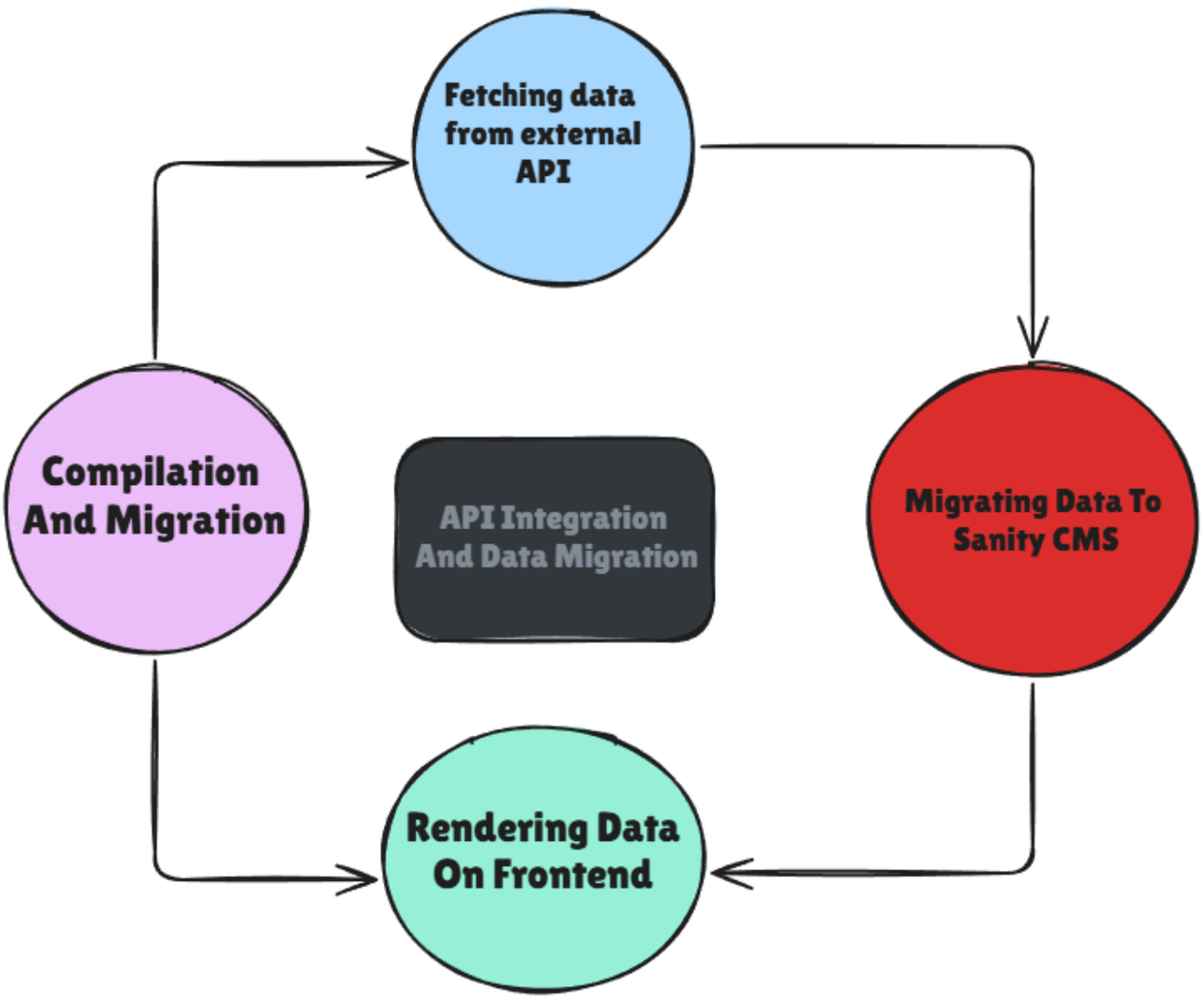
Day 3 - API Integration Report for FoodTuck

1. Introduction:

This report outlines the integration process of food product and category data from an external API into the backend system of **FoodTuck**, an online food ordering platform. The integration leverages **Sanity CMS** for content management and **Next.js** for frontend rendering.
The main objectives of this integration were:

1. Fetching food product and category data from an external API.
2. Storing and managing the data in Sanity CMS.
3. Dynamically displaying the fetched data on the frontend, enhancing user experience.



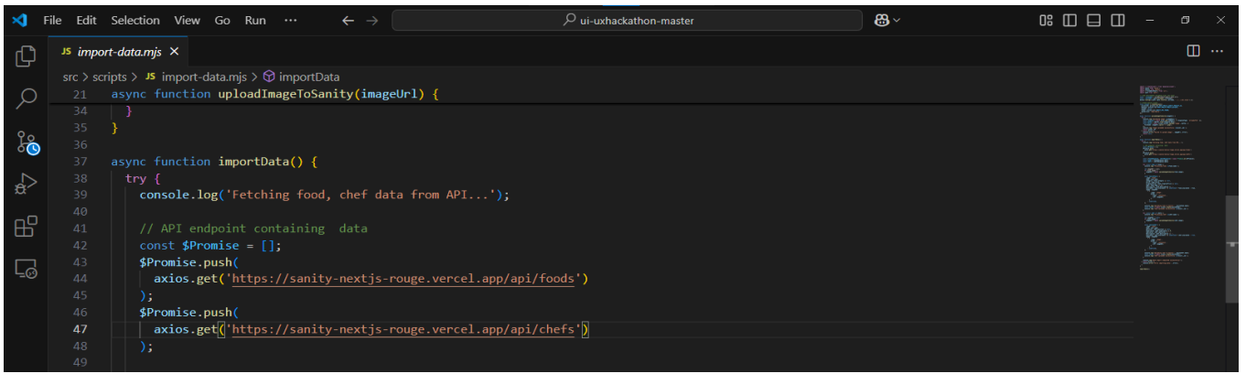
Step 1: Fetching Data from API

The API provided the following endpoints:

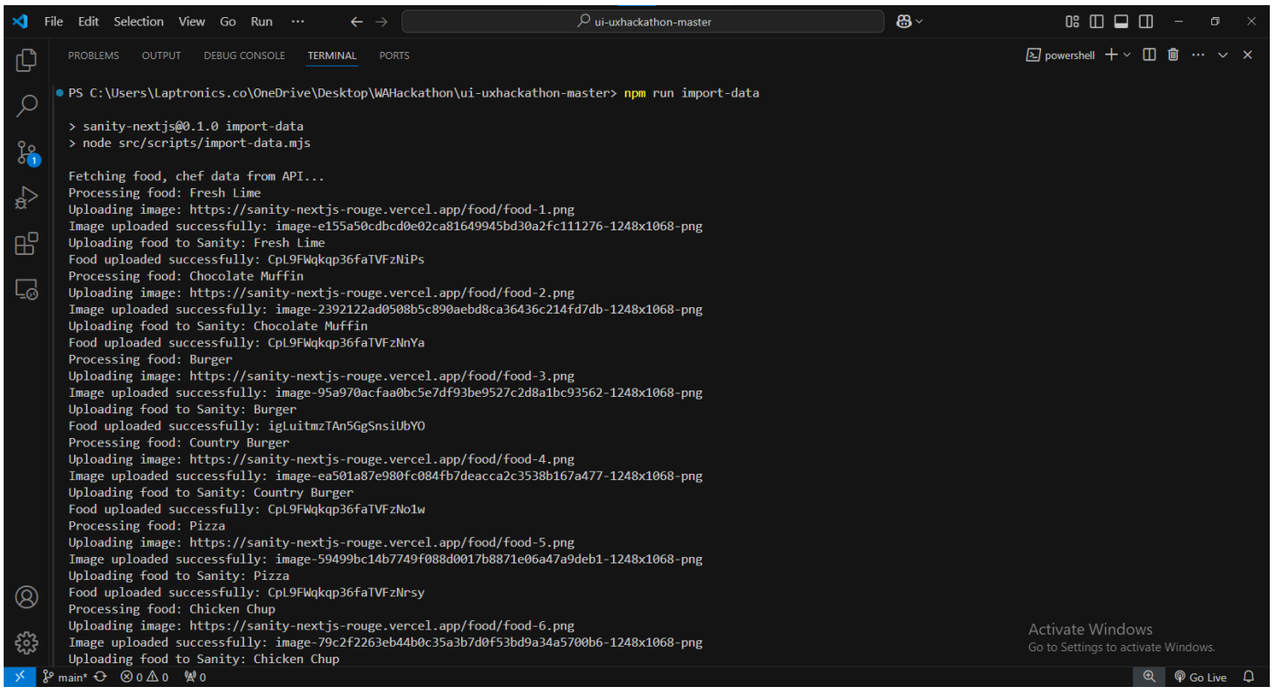
- **Products Endpoint:** Includes details such as titles, prices, descriptions, categories, inventory, and images.
    - **Base API URL (Updated):**

    - https://sanity-nextjs-rouge.vercel.app/api/foods

- **Chefs Endpoint:** Information related to chefs (like names, specialties, etc.)
    - **Base API URL (Updated):**
        - https://sanity-nextjs-rouge.vercel.app/api/chefs



Console Output

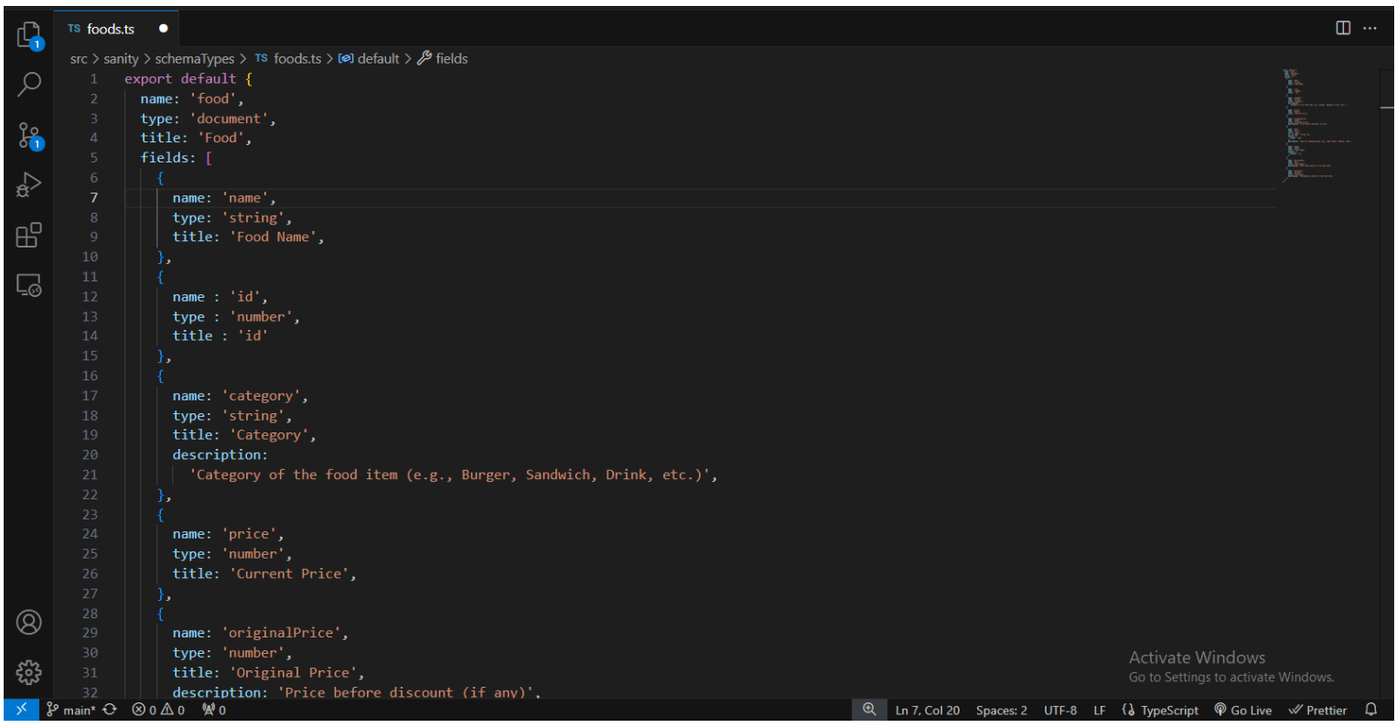


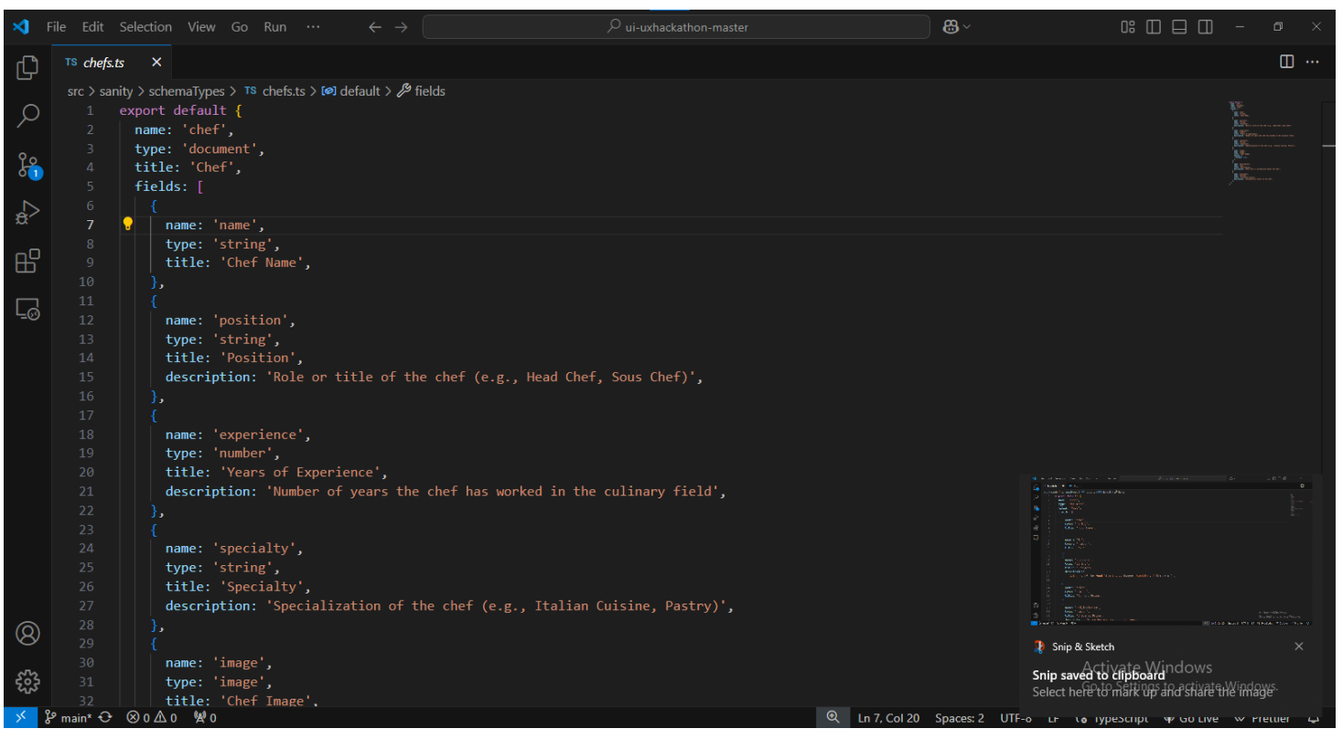
- **Step 2:** Migrating Data to Sanity CMS

  The fetched data was carefully structured and securely stored in **Sanity CMS** using its robust **JavaScript client library**. This powerful library facilitated seamless integration, ensuring the data fetched from multiple sources was accurately mapped to the relevant fields in the CMS. In the process, the **product schema** was significantly enhanced to include additional fields, providing more detailed data management capabilities. These updates were specifically designed to allow for a more granular approach to managing both

food and **chef** data, ensuring each element is categorized, tracked, and displayed in a way that is both comprehensive and user-friendly.

Schema Update for Food and Chef

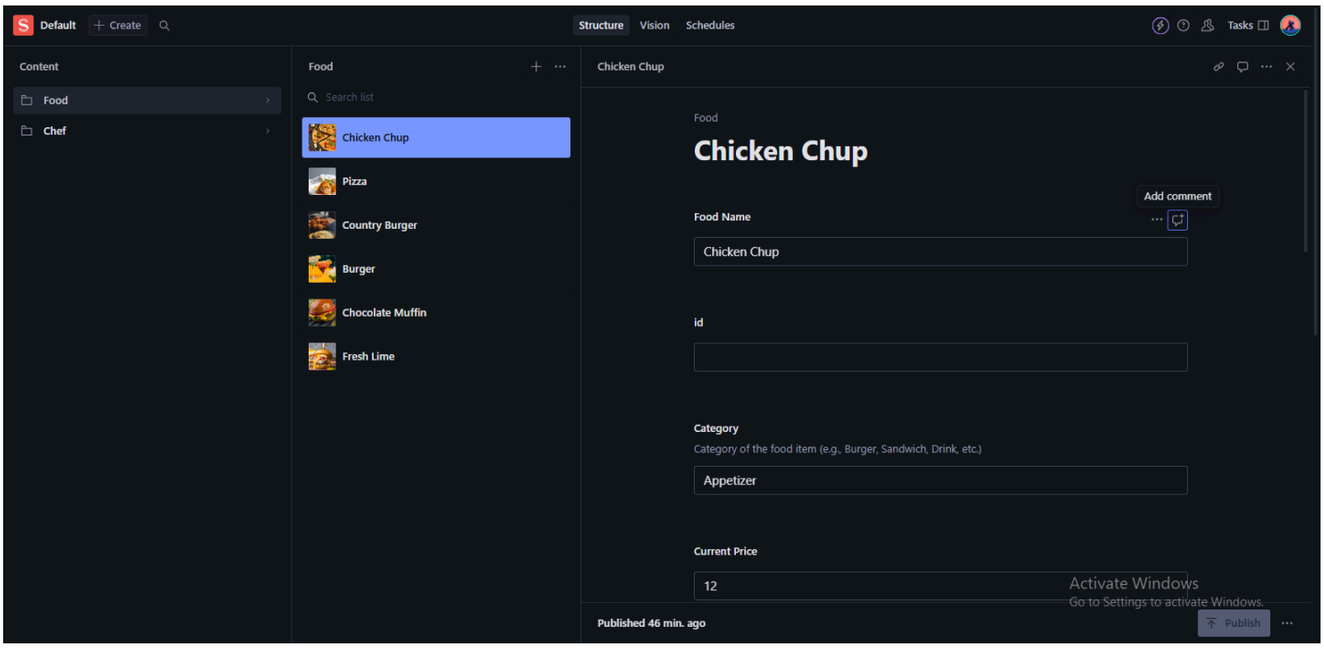


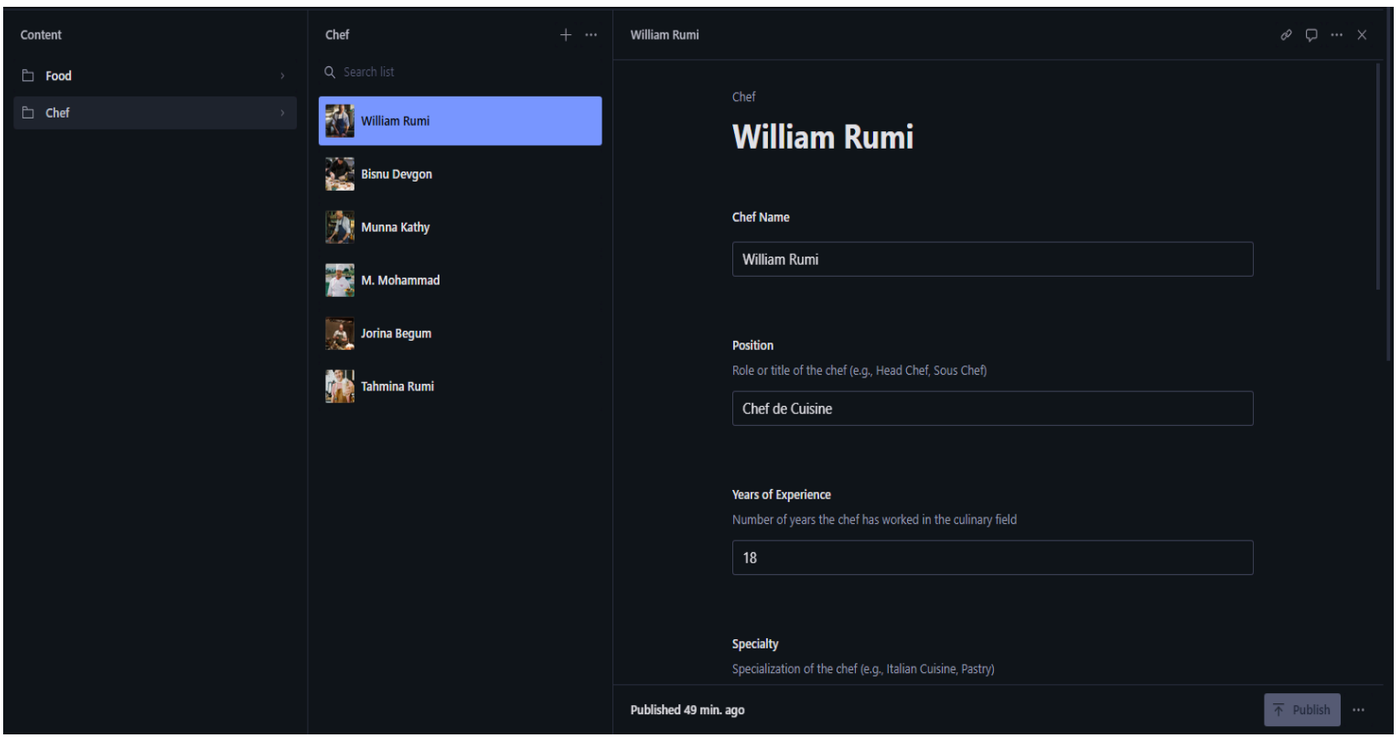


MIGRATION SCRIPT

```
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}`);
    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);
```

Sanity Dashboard Screenshot Where Our Products is stored

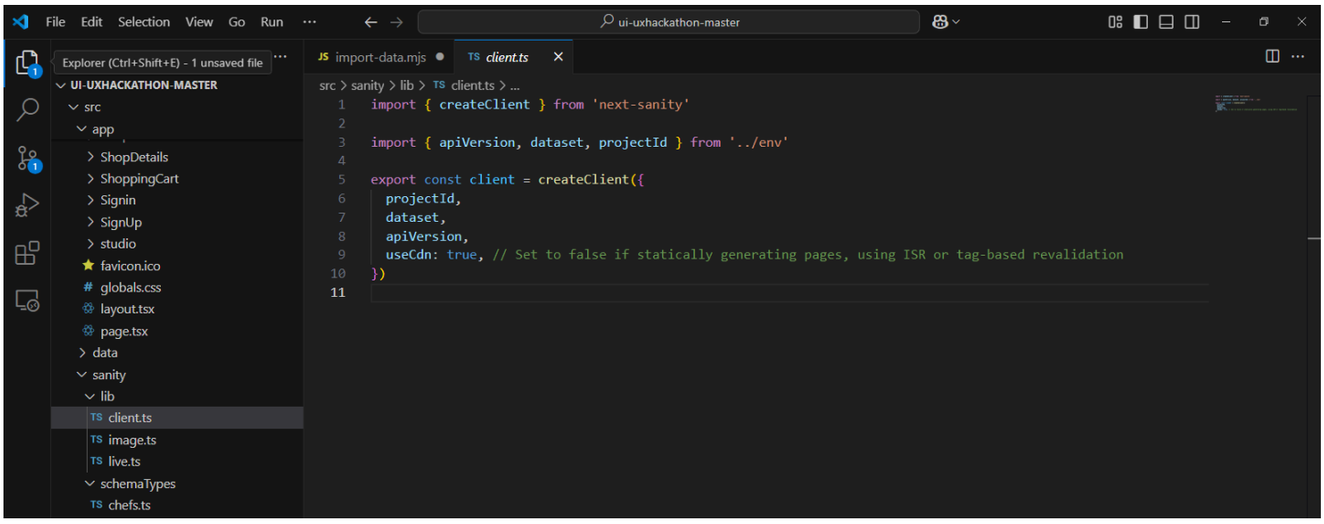




Step 3: Rendering Data on Frontend

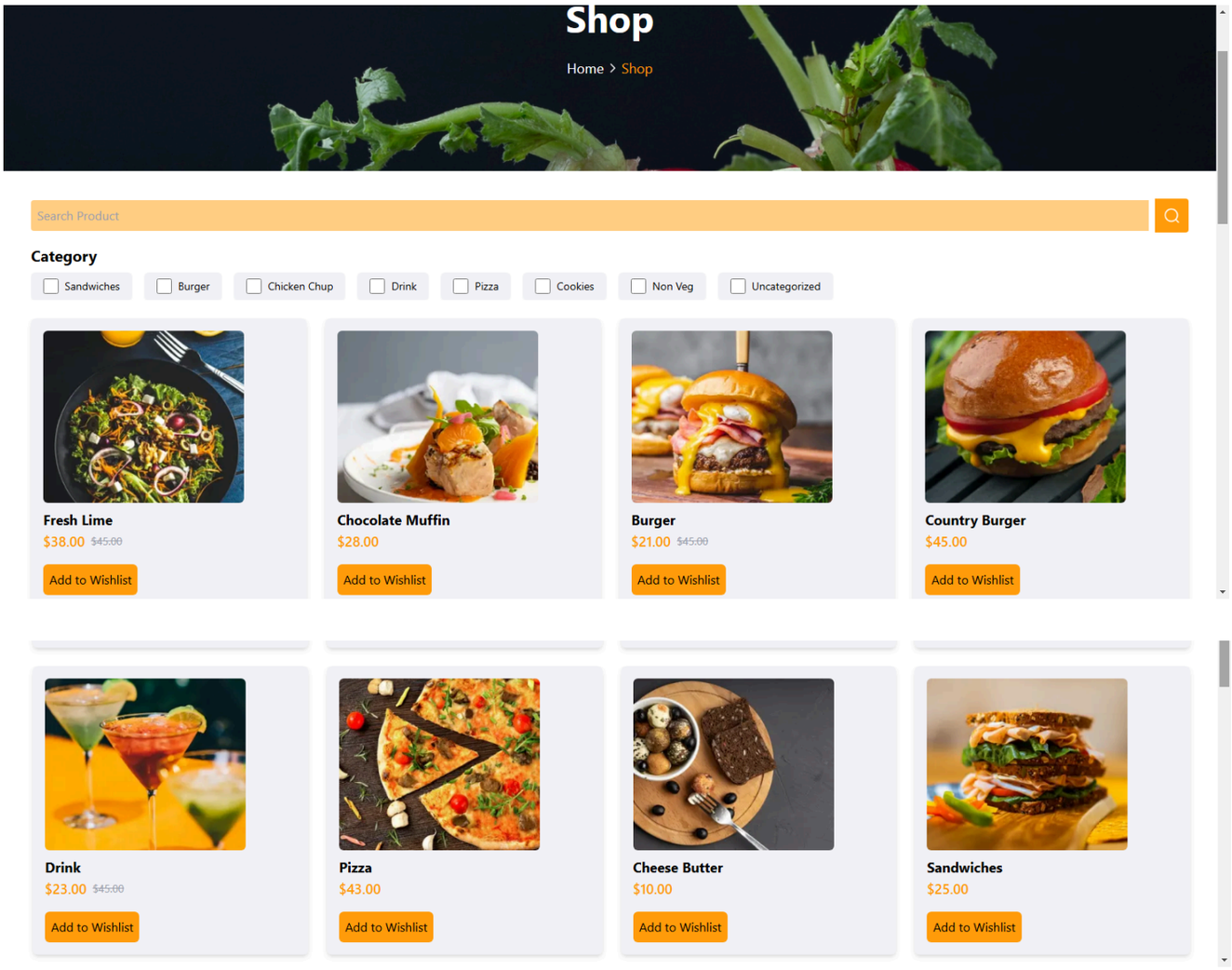
The next step was to fetch the stored data from **Sanity CMS** and dynamically display it on the **FoodTuck** frontend. Using **Next.js**, we were able to fetch the data from Sanity CMS and render it into the food listing UI in a highly dynamic and efficient way. This approach ensures that the food items and related data are seamlessly integrated into the user interface, creating an engaging experience for customers.

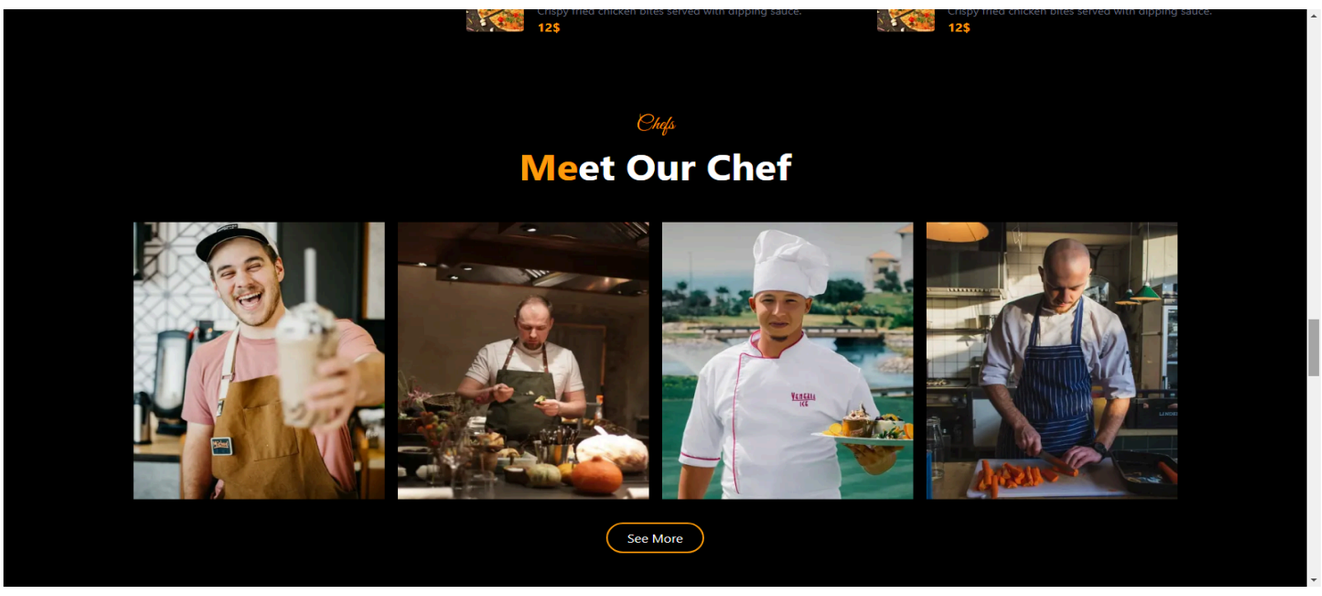
Code Example for Fetching Data in Frontend:



Product Display

Below is a screenshot of the product listing page,showing products dynamically rendered on the frontend:





Conclusion

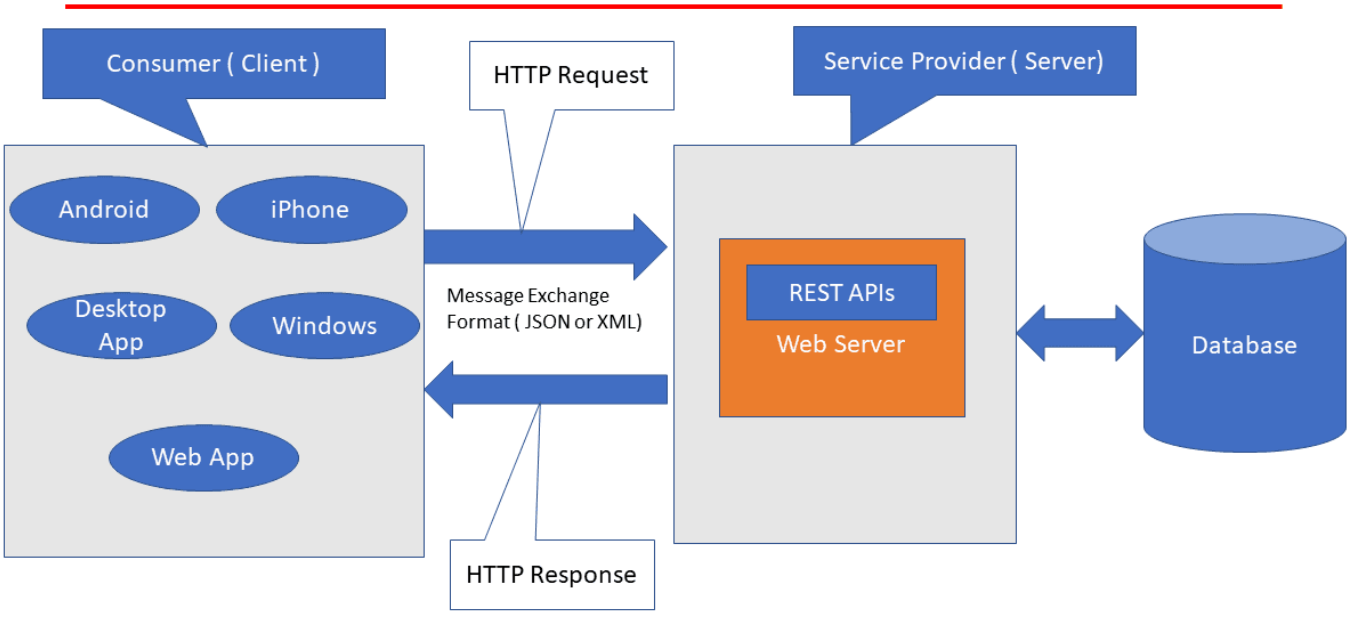
The API integration for FoodTuck has been successfully completed, ensuring seamless synchronization between the Sanity CMS and the frontend of the application. The process encompassed several critical steps, including the fetching of data from Sanity CMS, storing the data in the CMS, and dynamically rendering the content on the frontend using Next.js.

This integration has not only streamlined the process of managing food items and chef profiles, but it also provides a robust system for efficiently handling product details, categories, pricing, and promotions. By leveraging Sanity CMS as the content management system, the admin panel is now able to update, manage, and display product data in real-time with minimal effort, ensuring consistency and accuracy across the platform.

The dynamic rendering of food data on the FoodTuck frontend enhances the user experience by ensuring the menu is always up-to-date and responsive, adapting in real-time to changes made in the CMS. This seamless flow of data from the backend to the user interface simplifies both administration and user interaction, promoting smooth operations and enhancing the overall efficiency of managing the food offerings.

To validate the implementation, screenshots and data outputs are attached, which highlight the successful completion of each step, from data fetching to the final frontend display. These visual validations serve as proof of the integration's effectiveness and functionality, ensuring that both the administrative tasks and the user-facing elements are functioning flawlessly.

Ultimately, this integration provides a streamlined management experience, empowering administrators to efficiently update products and categories while delivering an engaging, real-time experience to users. The FoodTuck platform is now fully equipped to handle dynamic product and category management with ease, contributing to better operational efficiency and a more intuitive user experience.



Documentation Author: Wajahat Ali

Slot: (Saturday 9am to 12pm)

Task Given By : Sir Ameen Alam

Class Teacher : Sir Bilal Muhammad Khan & Sir Aneeq Khatri