Day 3 - API Integration Report - “FoodTuck"

API integration process:

Integration : Integrated an API to fetch food items dynamically in the React project.

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

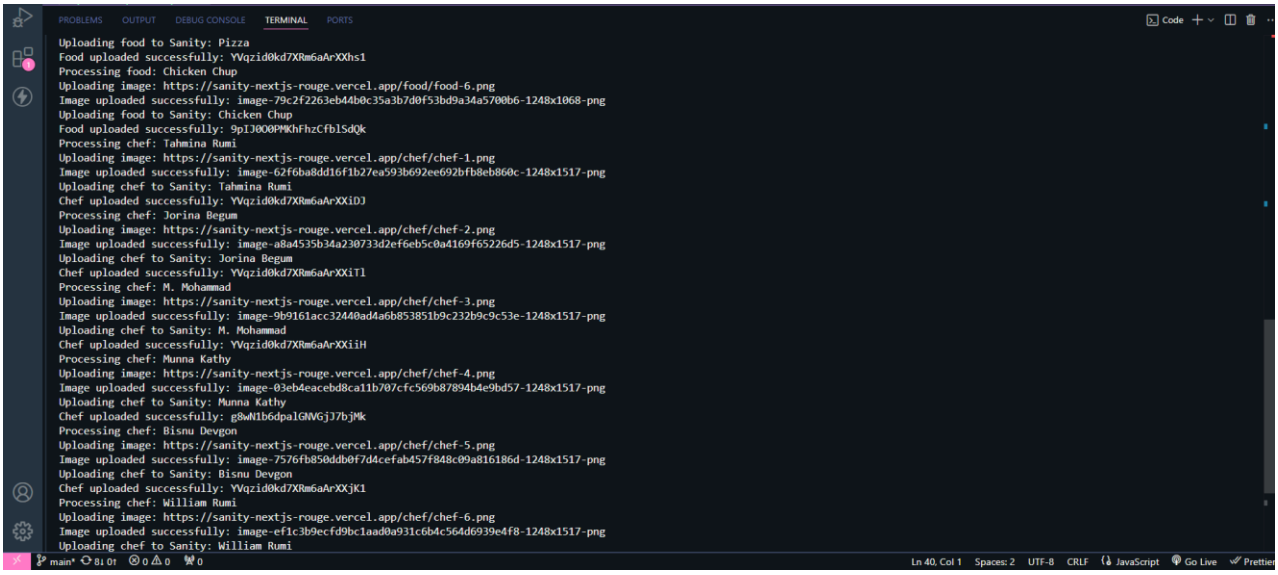
https://sanity-nextjs-rouge.vercel.app/api/chefs'

Used the `useEffect` hook to fetch data from the API on component mount.

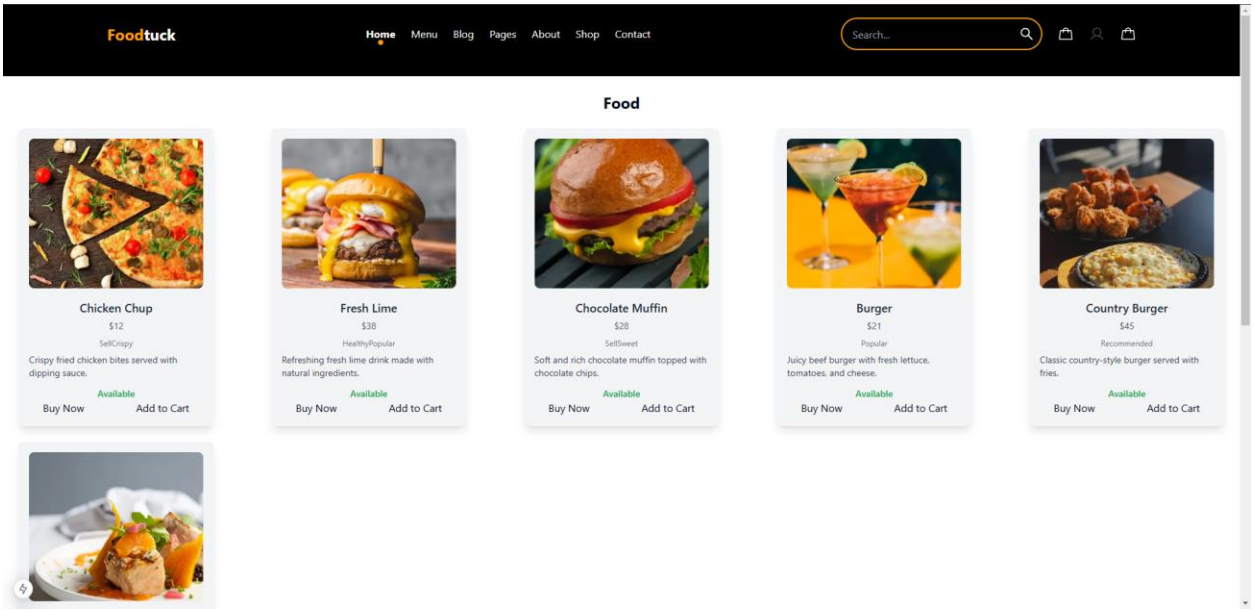
Managing State: Managed the API response and loading/error states using the `useState` hook.

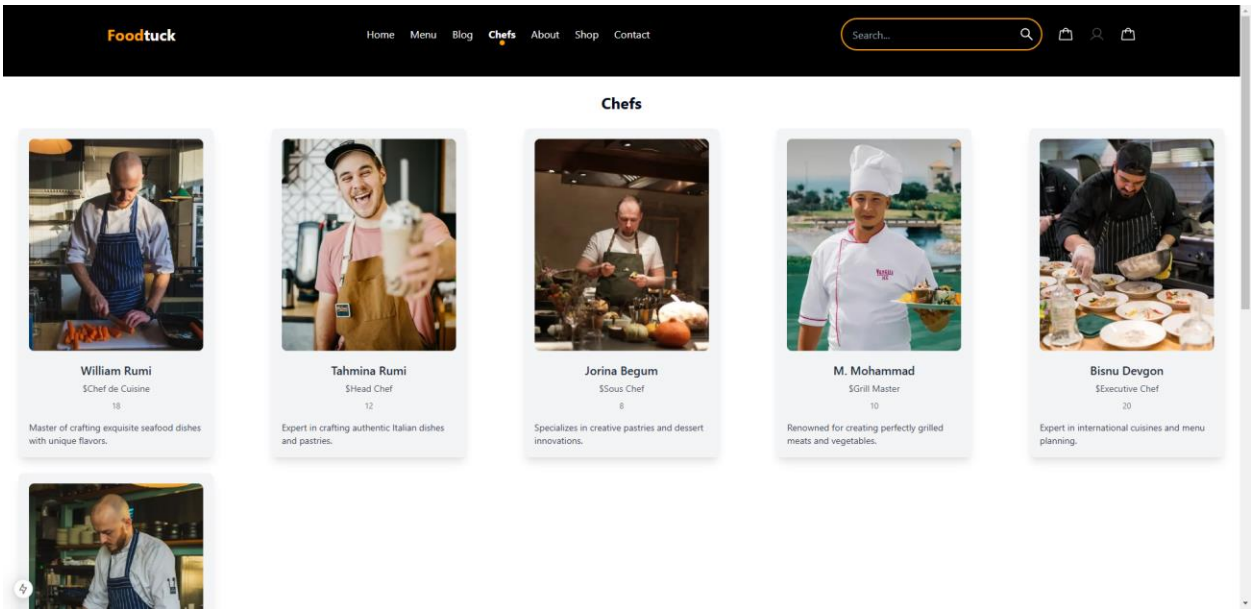
Displaying Data: Displayed the fetched food items in a responsive layout, with three items per row.
Rendering Item Details: For each food item, rendered its image, name, price, tags, description, and availability status.
Buttons: Added "Buy Now" and "Add to Cart" buttons for each food item.
API Call Method: Used `axios` (or `fetch`) to make the API call and handle the response.
Error Handling: Implemented error handling to manage failed API requests or data issues.
Responsive Layout: Ensured the layout is responsive, displaying three items per row and adjusting accordingly on different screen sizes.

API Calls :

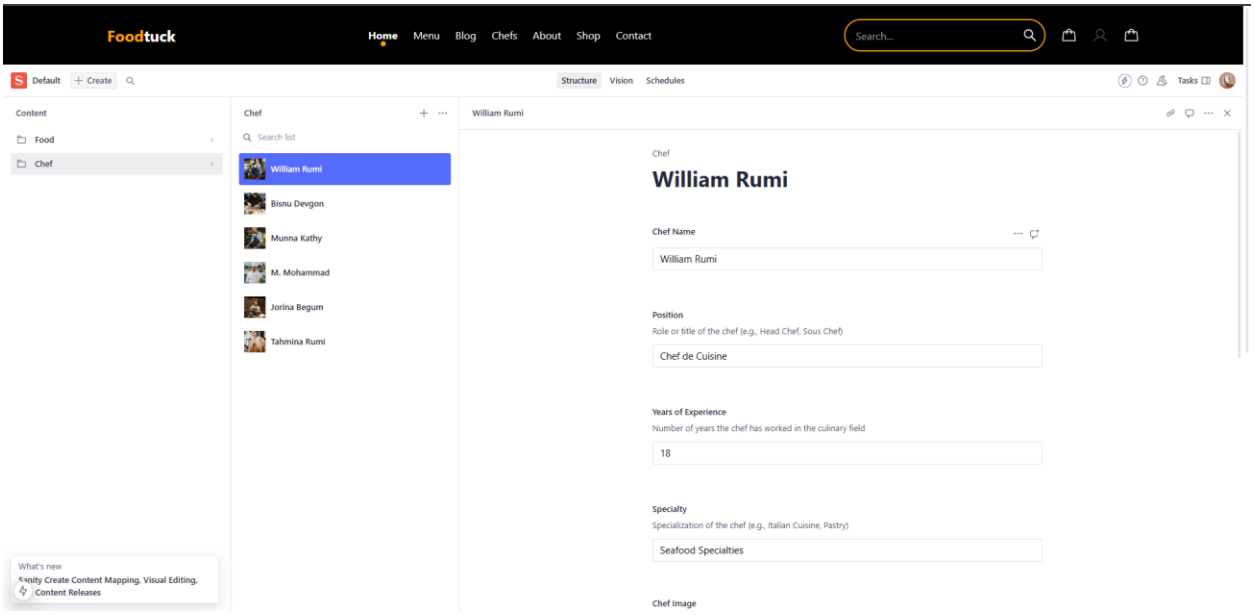


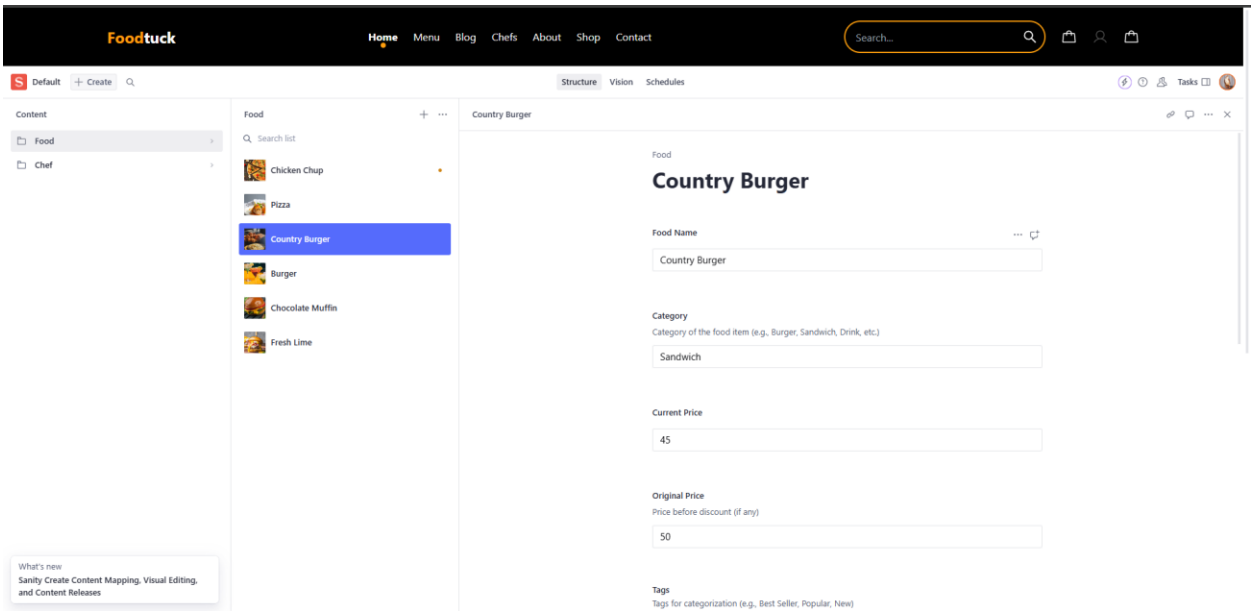
Data in Front-end :

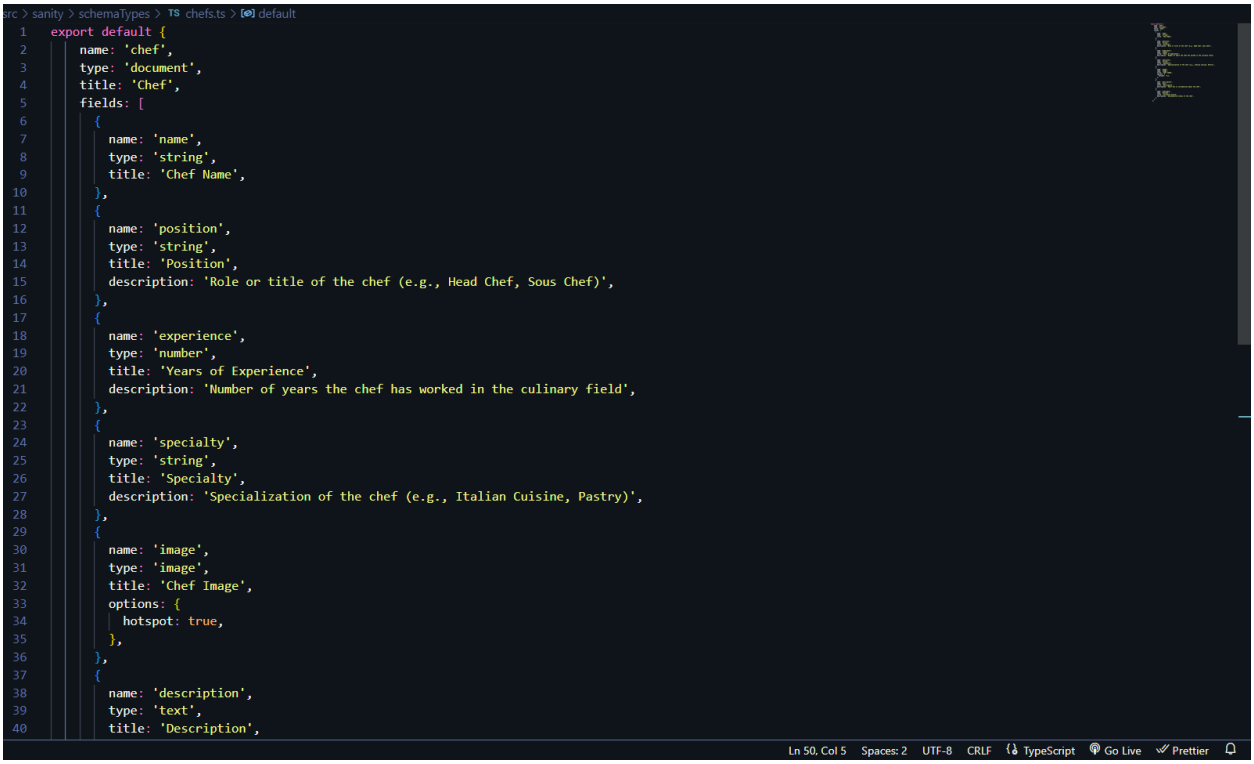


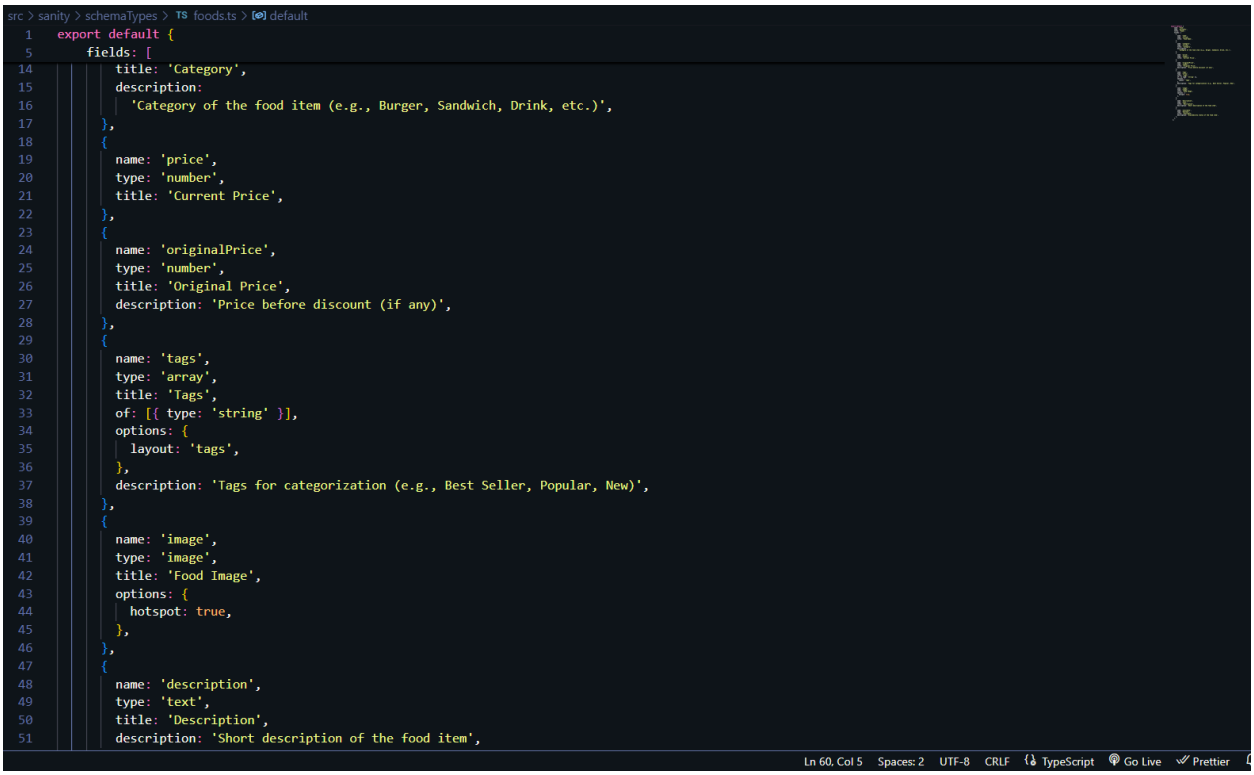


Populated Sanity CMS fields:




Foodtuck
Home
Menu
Blog
Chefs
About
Shop
Contact
Search...
Default
Create
Structure
Vision
Schedules
Tasks
Content
Food
Chef
Food
Search list
Chicken Chup
Pizza
Country Burger
Burger
Chocolate Muffin
Fresh Lime
Country Burger
Food
Country Burger
Food Name
Country Burger
Category
Category of the food item (e.g., Burger, Sandwich, Drink, etc.)
Sandwich
Current Price
45
Original Price
Price before discount (if any)
50
Tags
Tags for categorization (e.g., Best Seller, Popular, New)
What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases


src > sanity > schemaTypes > chefs.ts > default
export default {
name: 'chef',
type: 'document',
title: 'Chef',
fields: [
{
name: 'name',
type: 'string',
title: 'Chef Name',
},
{
name: 'position',
type: 'string',
title: 'Position',
description: 'Role or title of the chef (e.g., Head Chef, Sous Chef)',
},
{
name: 'experience',
type: 'number',
title: 'Years of Experience',
description: 'Number of years the chef has worked in the culinary field',
},
{
name: 'specialty',
type: 'string',
title: 'Specialty',
description: 'Specialization of the chef (e.g., Italian Cuisine, Pastry)',
},
{
name: 'image',
type: 'image',
title: 'Chef Image',
options: {
hotspot: true,
},
},
{
name: 'description',
type: 'text',
title: 'Description',
Ln 50, Col 5
Spaces: 2
UTF-8
CRLF
TypeScript
Go Live
Prettier



Conclusion :

In this project, we successfully integrated data from **Sanity CMS** into a frontend application using **API calls**. We populated the relevant fields in Sanity Studio, including chef details such as `name`, `position`, `speciality`, and `imageUrl`. These fields were defined through schema files in Sanity, ensuring a structured content model.

The **Sanity API** was then used to fetch the data, which was displayed in the frontend, confirming the smooth flow of data from the CMS to the user interface. This integration not only makes the application dynamic but also allows easy management and updating of content directly from the **Sanity Studio** without needing to modify the frontend code.

Through this process, we learned how to populate content in Sanity CMS, define schema structures, and retrieve the data via API calls to display it efficiently in a React or Next.js application.