

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'](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.

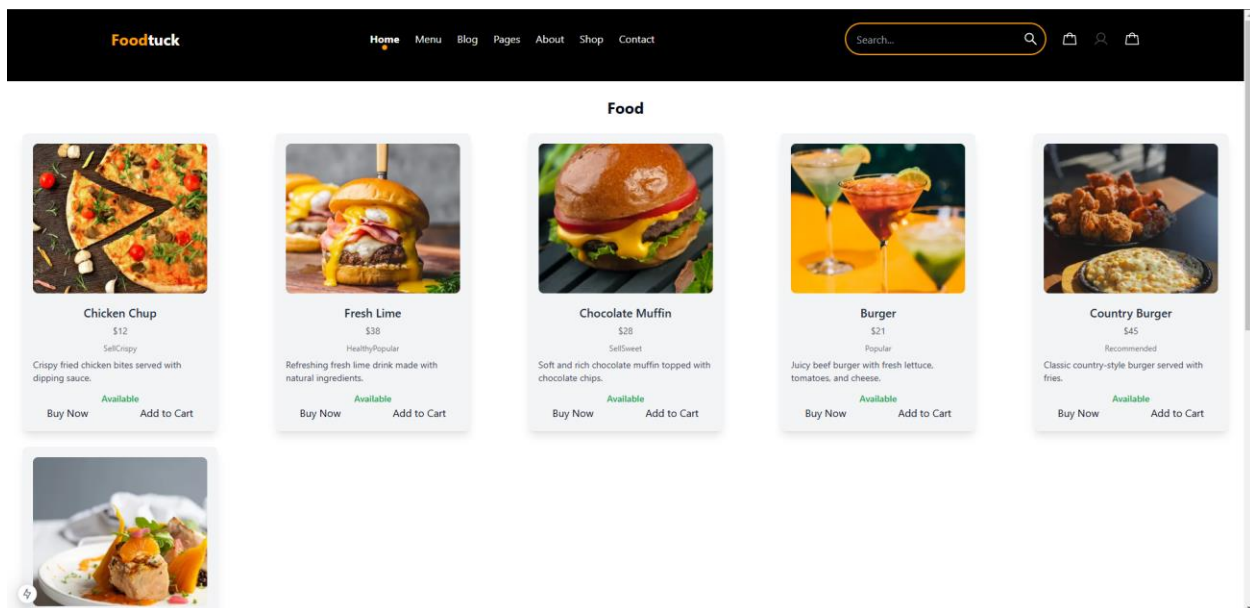
Error Handling: Implemented error handling to manage failed API requests or data issues.

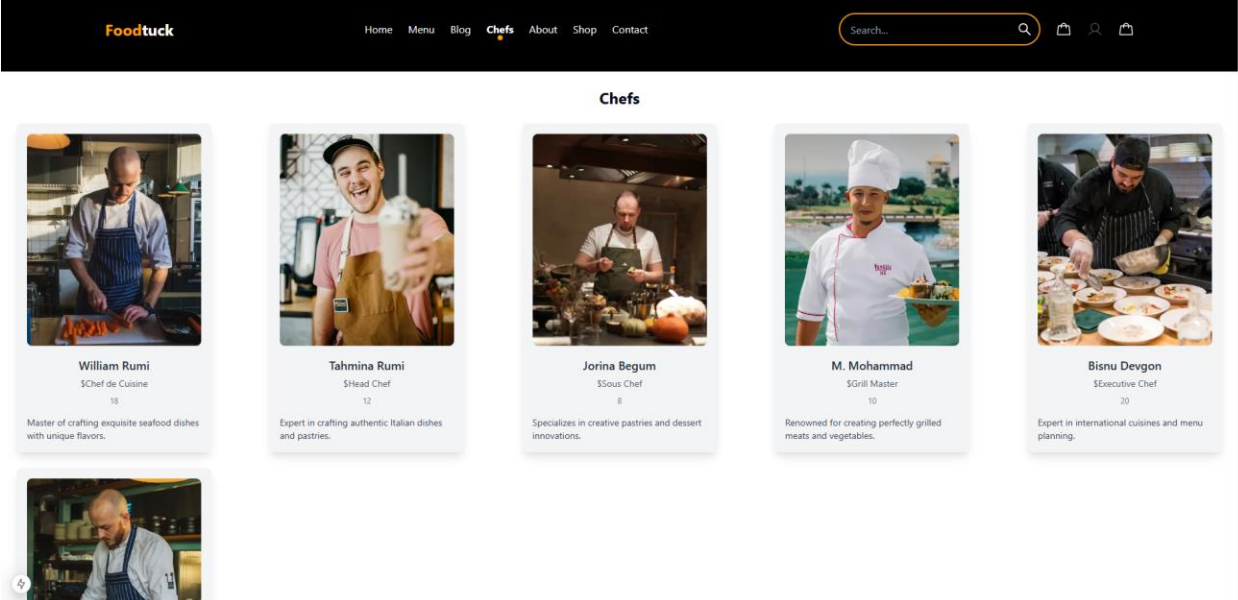
Responsive Layout: Ensured the layout is responsive, displaying three items per row and adjusting accordingly on different screen sizes.

API Calls :

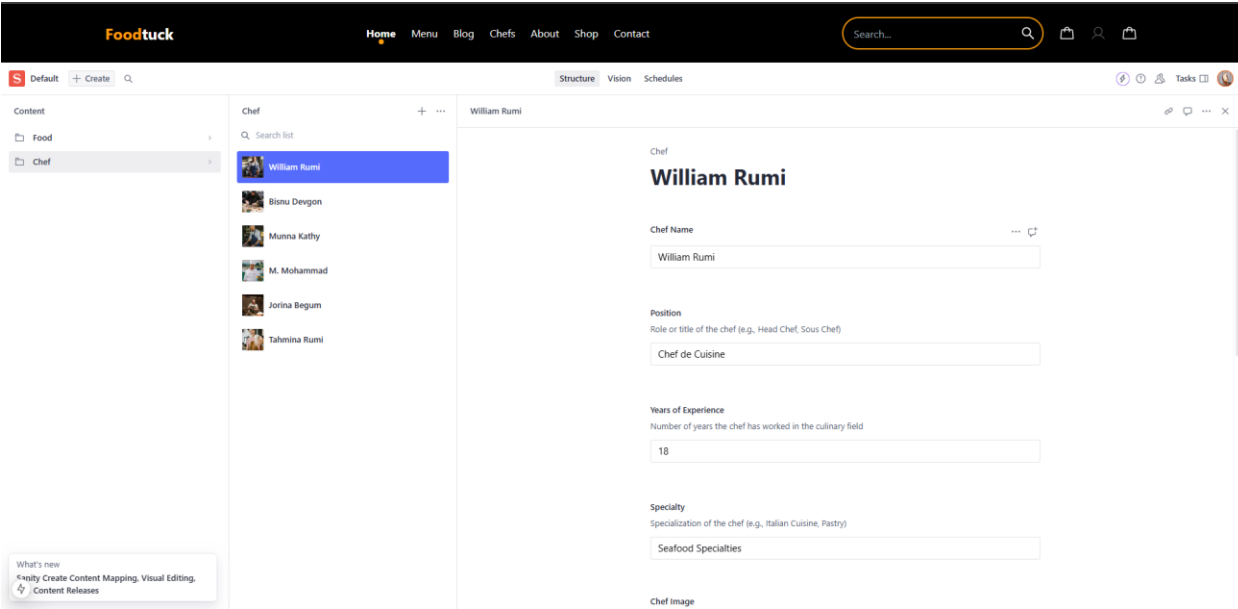
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Uploading food to Sanity: Pizza
Food uploaded successfully: VVqz1d0kd7XRm6aArXXhs1
Processing food: Chicken Chup
Uploading image: https://sanity-nextjs-rouge.vercel.app/food/food-6.png
Image uploaded successfully: image-79c2f2263eb44b0c35a3b7d0f53bd9a34a5700b6-1248x1068-png
Uploading food to Sanity: Chicken Chup
Food uploaded successfully: 9pIJ000P9KhFhzCfb15dQk
Processing chef: Tahmina Rumi
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-1.png
Image uploaded successfully: image-62f6ba8dd16f1b27ea593b692ee692bfb8eb860c-1248x1517-png
Uploading chef to Sanity: Tahmina Rumi
Chef uploaded successfully: VVqz1d0kd7XRm6aArXXiDj
Processing chef: Jorina Begum
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-2.png
Image uploaded successfully: image-a8a4535b34a230733d2ef6eb5c0a4169f65226d5-1248x1517-png
Uploading chef to Sanity: Jorina Begum
Chef uploaded successfully: VVqz1d0kd7XRm6aArXXi1l
Processing chef: M. Mohammad
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-3.png
Image uploaded successfully: image-9b9161acc32440ad4a6b853851b9c232b9c9c53e-1248x1517-png
Uploading chef to Sanity: M. Mohammad
Chef uploaded successfully: VVqz1d0kd7XRm6aArXXi1h
Processing chef: Munna Kathy
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-4.png
Image uploaded successfully: image-03eb4eacebd8c11b707cfc369b87894b4e9bd57-1248x1517-png
Uploading chef to Sanity: Munna Kathy
Chef uploaded successfully: g8wM1b6dpa1GwGj77bJmk
Processing chef: Bisnu Devgon
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-5.png
Image uploaded successfully: image-7576fb850ddb0f7d4cefab457f848c09a816186d-1248x1517-png
Uploading chef to Sanity: Bisnu Devgon
Chef uploaded successfully: VVqz1d0kd7XRm6aArXXjK1
Processing chef: William Rumi
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-6.png
Image uploaded successfully: image-ef1c3b9ecfd9bclaad0a931c6b4c564d6939e4f8-1248x1517-png
Uploading chef to Sanity: William Rumi
```

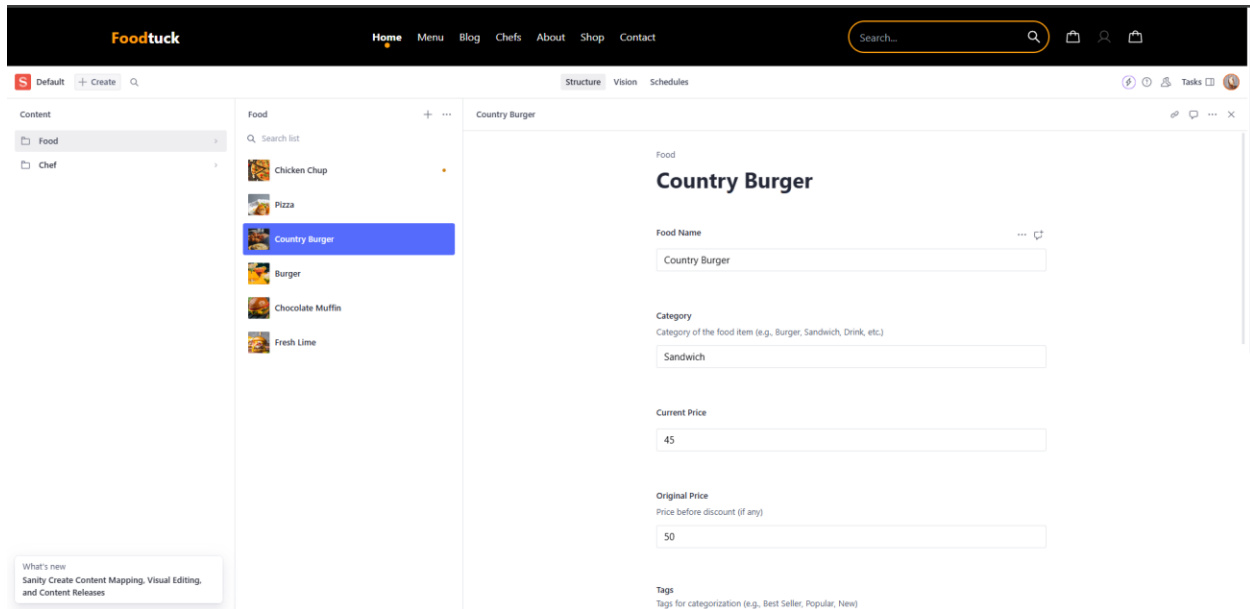
Data in Front-end :





Populated Sanity CMS fields:





```
src > sanity > schemaTypes > TS chefs.ts > default
1  export default {
2    name: 'chef',
3    type: 'document',
4    title: 'Chef',
5    fields: [
6      {
7        name: 'name',
8        type: 'string',
9        title: 'Chef Name',
10     },
11     {
12       name: 'position',
13       type: 'string',
14       title: 'Position',
15       description: 'Role or title of the chef (e.g., Head Chef, Sous Chef)',
16     },
17     {
18       name: 'experience',
19       type: 'number',
20       title: 'Years of Experience',
21       description: 'Number of years the chef has worked in the culinary field',
22     },
23     {
24       name: 'specialty',
25       type: 'string',
26       title: 'Specialty',
27       description: 'Specialization of the chef (e.g., Italian Cuisine, Pastry)',
28     },
29     {
30       name: 'image',
31       type: 'image',
32       title: 'Chef Image',
33       options: {
34         hotspot: true,
35       },
36     },
37     {
38       name: 'description',
39       type: 'text',
40       title: 'Description',
41     },
42   ],
43 }
```

```

src > sanity > schemaTypes > TS foods.ts > [0] default
1  export default {
5    fields: [
14      title: 'Category',
15      description:
16        'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
17    ],
18    {
19      name: 'price',
20      type: 'number',
21      title: 'Current Price',
22    },
23    {
24      name: 'originalPrice',
25      type: 'number',
26      title: 'Original Price',
27      description: 'Price before discount (if any)',
28    },
29    {
30      name: 'tags',
31      type: 'array',
32      title: 'Tags',
33      of: [{ type: 'string' }],
34      options: {
35        layout: 'tags',
36      },
37      description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
38    },
39    {
40      name: 'image',
41      type: 'image',
42      title: 'Food Image',
43      options: {
44        hotspot: true,
45      },
46    },
47    {
48      name: 'description',
49      type: 'text',
50      title: 'Description',
51      description: 'Short description of the food item',

```

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.

