Day 3 - API Integration Report – “ مہک ”

API Integration Process :

The API integration involved migrating data from external APIs to Sanity CMS and then displaying it in the frontend. The following steps outline the process:

1. **API Data Source**: Data was migrated from an external API to Sanity CMS. The APIs were integrated for two key data types: `chefs` and `foods`.
2. **Sanity Setup**: Sanity CMS schemas were created to store this data. The schemas for both `chefs` and `foods` were adjusted to match the structure of the data from the API (refer to the schema section below)

Food.ts

```
export default {
  name: 'food',
  type: 'document',
  title: 'Food',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Food Name',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      description:
        'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
    },
    {
      name: 'discountPrice',
      type: 'number',
      title: 'Current Price',
    },
    {
      name: 'originalPrice',
      type: 'number',
      title: 'Original Price',
```

```
      description: 'Price before discount (if any)',
    },
    {
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }],
      options: {
        layout: 'tags',
      },
      description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
    },
    {
      name: 'image',
      type: 'image',
      title: 'Food Image',
      options: {
        hotspot: true,
      },
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
      description: 'Short description of the food item',
    },
    {
      name: 'available',
      type: 'boolean',
      title: 'Available',
      description: 'Availability status of the food item',
    },
  ],
};
```

Chefs.ts

```
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',
      description: 'Short bio or introduction about the chef',
    },
    {
```

```
      name: 'available',
      type: 'boolean',
      title: 'Currently Active',
      description: 'Availability status of the chef',
    },
  ],
};
```

3. **Data Fetching in Frontend**: Next.js was used on the frontend to fetch data from Sanity using GROQ queries. The data was then rendered into components like `ProductCard` and `ChefCard` on the UI.

```
4. ``'use client'``
5.
6. import { client } from '@/sanity/lib/client';
7. import React from 'react';
8. import ProductCard from './card';
9. import Header from './header';
10.import ChefCard from './chef';
11.
12.export default async function Home() {
13.
14.  const query = `*[_type == "food"]{
15.    id,
16.    "imageUrl": image.asset->url,
17.    name,
18.    category,
19.    description,
20.    discountPrice,
21.    originalPrice,
22.    tags,
23.    available
24.  }`;
25.  const query2 = `*[_type == "chef"]{
26.    name,
27.    position,
28.    experience,
29.    specialty,
30.    "imageUrl": image.asset->url,
31.    description,
```

```
    available
  }`;
  const food_data: Food[] = await client.fetch(query);
  const chef_data: Chefs[] = await client.fetch(query2);

  return (
    <div>
      <Header/>

     {
        food_data.map((data: Food) =>  (

          <ProductCard  data={data} />

        ))

      }

      {

        chef_data.map((chef: Chefs) => (


        <div id='chef' >
          <ChefCard chef={chef} />
        </div>

       ))}
    </div>
  );
}
```

4. **Rendering Process**: The fetched data was displayed dynamically based on user interactions (e.g., searching, browsing products, and viewing chefs).

Adjustments Made to Schemas

1. Chefs Schema

- Fields:
  - name: Name of the chef
  - position: Role or title of the chef (e.g., Head Chef, Sous Chef)
  - experience: Years of experience in the culinary field
  - specialty: Specialty in cooking (e.g., Italian cuisine, Pastry)
  - image: Chef image
  - description: Short bio or introduction about the chef
  - available: Availability status of the chef

2. Foods Schema

- Fields:
  - name: Name of the food item
  - category: Type of food (e.g., sushi, dessert)
  - description: Description of the food item
  - image: Food image
  - discountPrice: Price after discount
  - originalPrice: Original price before discount
  - tags: Tags to categorize and search for food
  - available: Whether the food item is available or out of stock

Adjustments:

- For both schemas, fields like available, description, and images were added to enhance data presentation on the frontend.

Migration Steps and Tools Used

1. **Data Migration to Sanity**:
   - The API data was imported into Sanity using Sanity's CLI tools and migration scripts.
   - A custom script was written to fetch data from the external API and create Sanity documents.
2. **Tools Used**:
   - **Sanity CLI**: Used for schema creation and migration.
   - **Sanity Studio**: To visualize and manage data post-migration.
   - **Custom Scripts**: Written in JavaScript to handle the fetching and importing of data from APIs into Sanity CMS.

- **API Calls**:
  Card.tsx

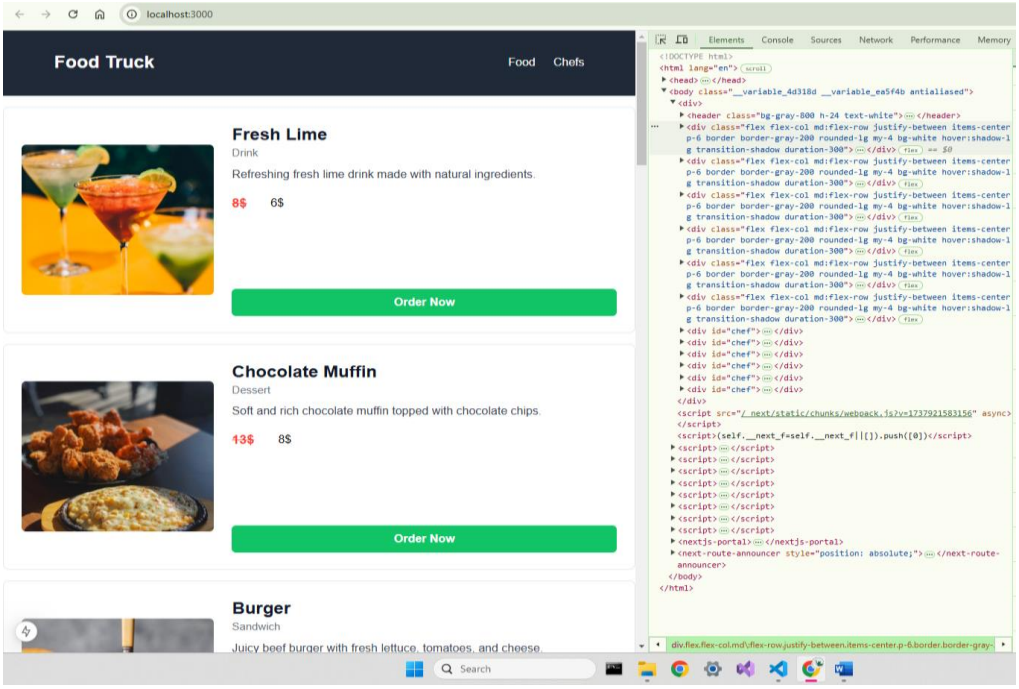
```
"use client";

import { urlFor } from '@/sanity/lib/image';
import React from 'react';

const ProductCard = ({ data }: { data: Food }) => {
  return (
    <div className="flex flex-col md:flex-row justify-between items-center p-6 border border-gray-200 rounded-lg my-4 bg-white hover:shadow-lg transition-shadow duration-300">
      <div className="w-full md:w-1/3">
        <img src={urlFor(data.imageUrl).url()}
          className="w-full h-auto object-cover rounded-md" />
      </div>

      <div className="flex flex-col justify-between mt-4 md:mt-0 md:ml-6 w-full md:w-2/3 text-center md:text-left">
        <h2 className="text-2xl font-bold text-gray-900">{data.name}</h2>
        <p className="text-gray-500 text-[15px] mb-2">{data.category}</p>
        <p className="text-gray-700 text-base mb-4">{data.description}</p>
        <div className="flex justify-center md:justify-start items-center mb-4">
          <span className="text-lg font-bold text-red-500 mr-8 line-through">{data.originalPrice}$</span>
          <span className="text-black ">{data.discountPrice}$</span>
        </div>

        <button className={`py-2 px-4 mt-24 rounded-md font-semibold transition-colors duration-300 ${data.available
              ? 'bg-green-500 text-white hover:bg-green-600'
              : 'bg-red-500 text-white cursor-not-allowed'
            }`} disabled={!data.available} >
          {data.available ? 'Order Now' : 'Out of Stock'}
        </button>
      </div>
    </div>
  );
};

export default ProductCard;
```

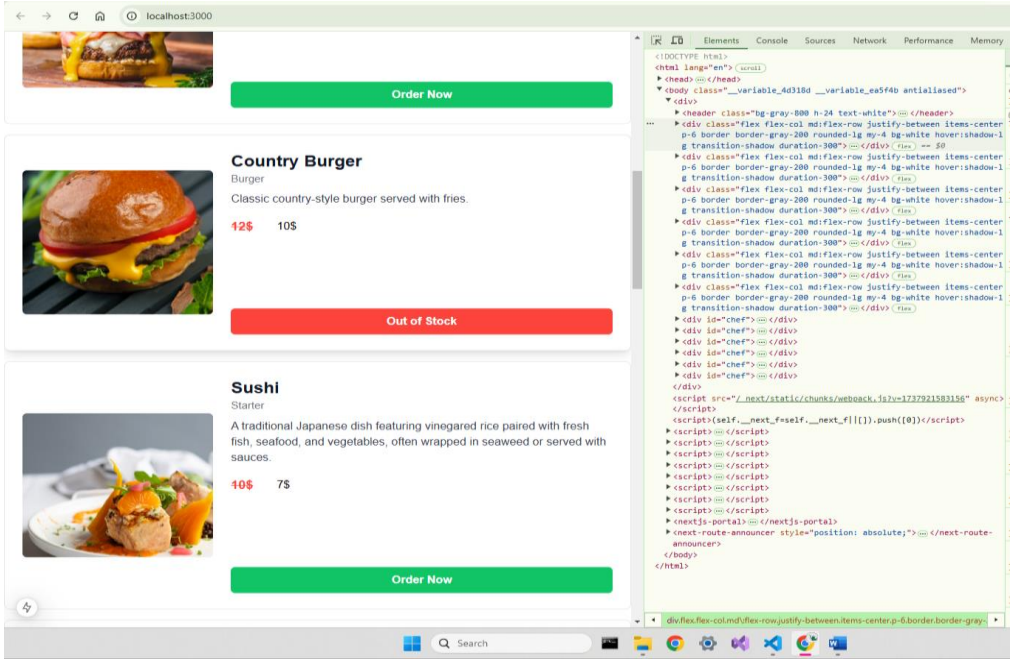
Chef.tsx

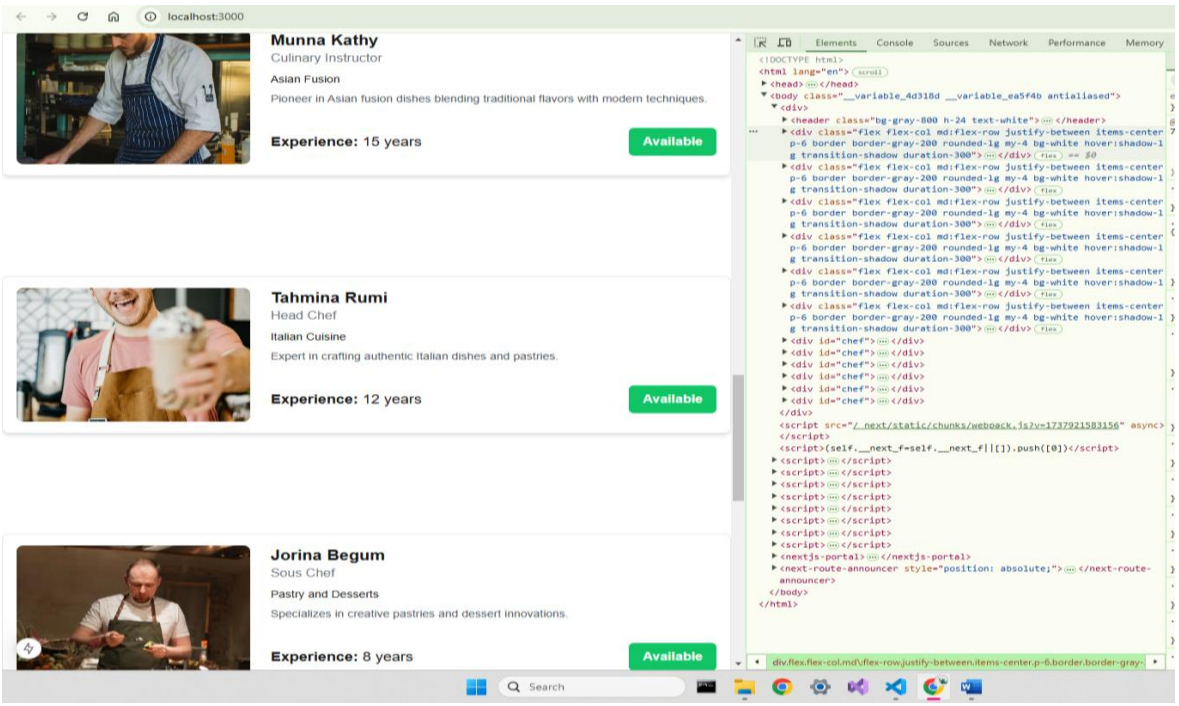
```tsx
"use client";

import React from 'react';
import { urlFor } from '@/sanity/lib/image';

const ChefCard = ({ chef }: { chef: Chefs }) =>  {

  return (
    <div
      className="flex flex-col md:flex-row items-start justify-between border
rounded-lg p-4 shadow-md max-w-4xl mx-auto mt-36 ">
      <div className="w-full md:w-1/3 mb-4 md:mb-0">
        <img
          src={urlFor(chef.imageUrl).url()}
          alt={chef.name}
          className="w-full h-48 object-cover rounded-lg"
        />
      </div>
      <div className="flex flex-col justify-between w-full md:w-2/3 md:pl-6">
        <div className="mb-4">
          <h2 className="text-xl font-bold">{chef.name}</h2>
          <p className="text-gray-500">{chef.position}</p>
          <p className="mt-2 text-sm">{chef.specialty}</p>
          <p className="mt-2 text-sm text-gray-600">{chef.description}</p>
        </div>
        <div className="flex items-center justify-between mt-4">
          <div>
            <span className="text-lg font-semibold">Experience: </span>
            <span className="text-lg">{chef.experience} years</span>
          </div>
          <span
            className={`py-2 px-4 rounded-md font-semibold transition-colors
duration-300 ${
              chef.available
                ? 'bg-green-500 text-white hover:bg-green-600'
                : 'bg-red-500 text-white cursor-not-allowed opacity-50'
            }`}
          >
            {chef.available ? 'Available' : 'Not Available'}
          </span>
        </div>
      </div>
    </div>
```

```
  );};
export default ChefCard;
```

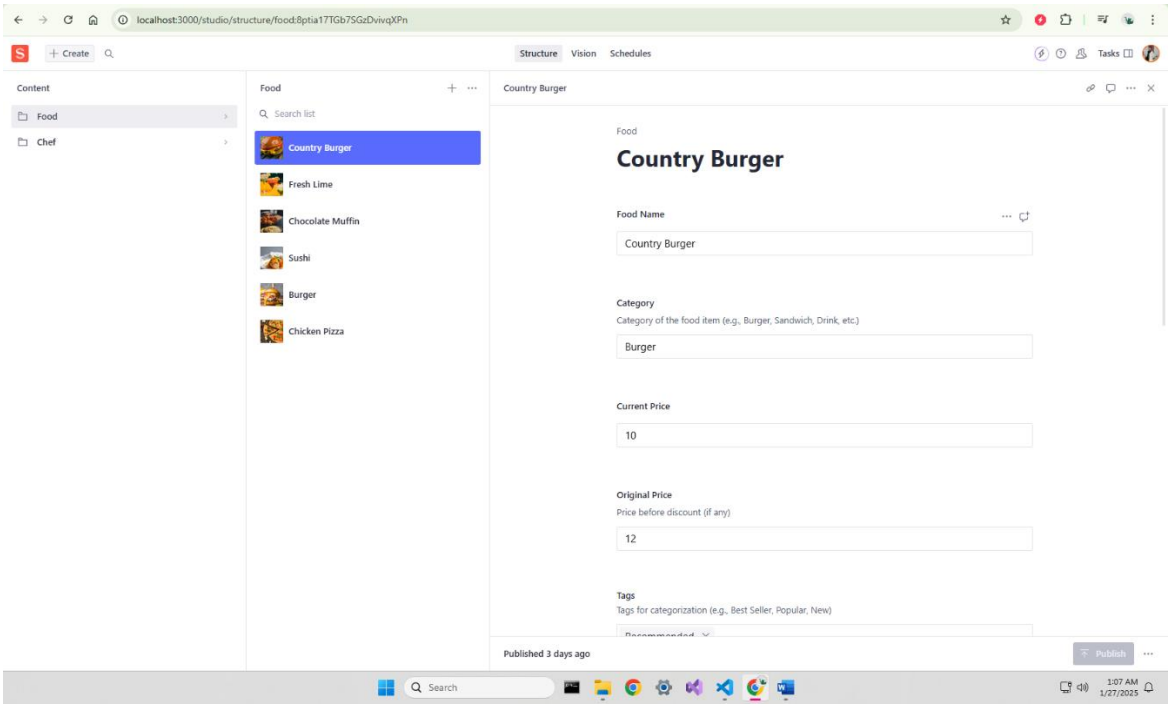
- **Data Display in Frontend**:

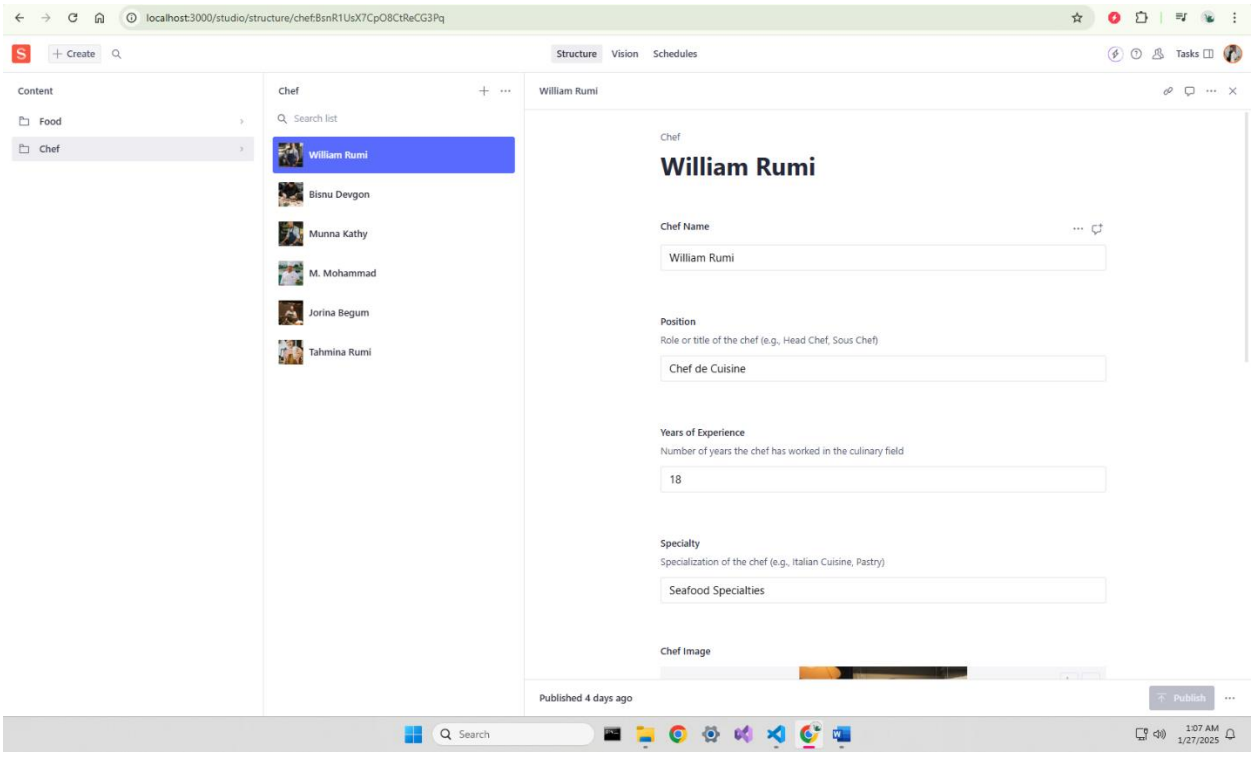






- **Populated Sanity CMS Fields**:




localhost:3000/studio/structure/chef;BsnR1UsX7CpO8CtReCG3Pq
Create
Structure
Vision
Schedules
Tasks
Content
Food
Chef
Chef
Search list
William Rumi
Bisnu Devgon
Munna Kathy
M. Mohammad
Jorina Begum
Tahmina Rumi
William Rumi
Chef
William Rumi
Chef Name
William Rumi
Position
Role or title of the chef (e.g., Head Chef, Sous Chef)
Chef de Cuisine
Years of Experience
Number of years the chef has worked in the culinary field
18
Specialty
Specialization of the chef (e.g., Italian Cuisine, Pastry)
Seafood Specialties
Chef Image
Published 4 days ago
Publish
Search
1:07 AM
1/27/2025