Day 3 Assignment: API Integration and Dynamic UI Development

Introduction

Today's task focused on integrating **Sanity CMS** into the application, fetching data dynamically through APIs, and rendering it in a responsive UI. The following sections highlight each stage of the process, supported by screenshots to illustrate progress and implementation.

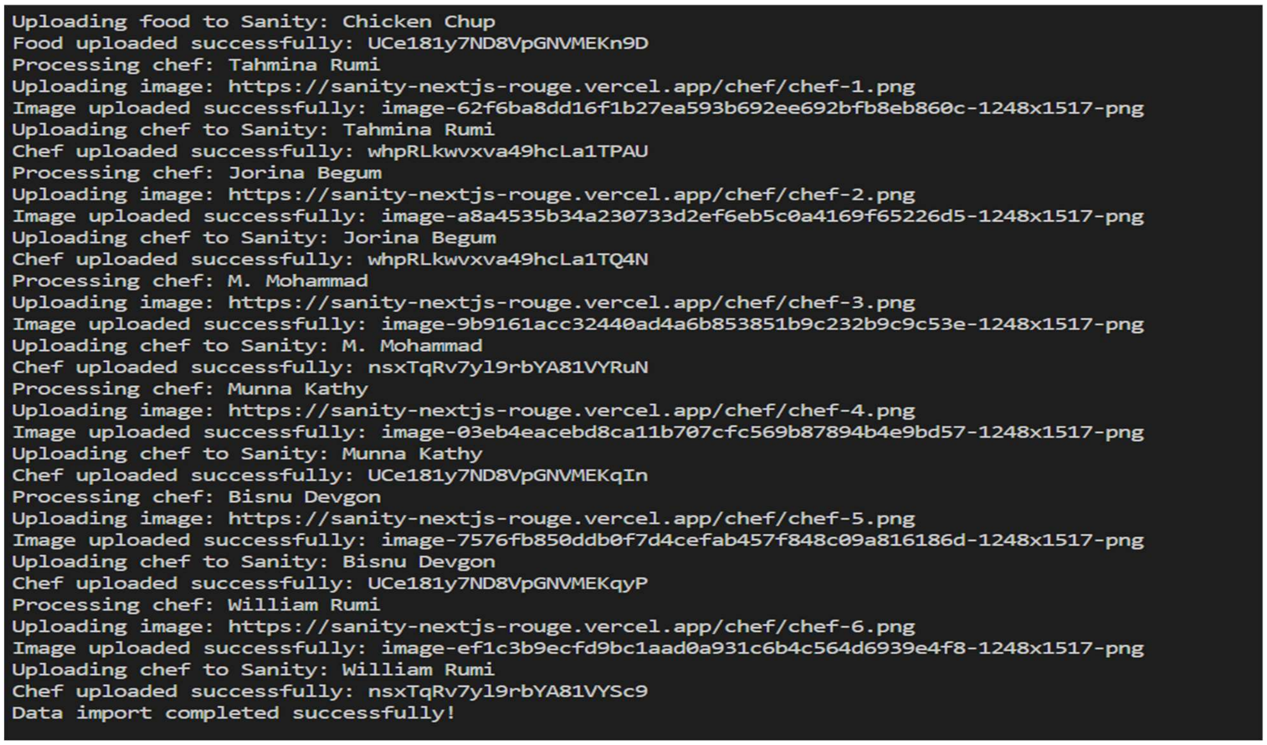
---

1. Fetching Data from API to Sanity

The first step involved migrating data from external APIs into **Sanity CMS**. This was achieved using scripts with the Sanity client.

- **Key Actions:**
  - Fetched data from both **Food** and **Chef** APIs.
  - Uploaded the data to Sanity CMS, including fields like `name`, `price`, `category`, `description`, and `image`.

Screenshot:

- **API to Sanity Integration Log:**
- 

```
Uploading food to Sanity: Chicken Chup
Food uploaded successfully: UCe181y7ND8VpGNVMEKn9D
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: whpRLkwvxva49hcLa1TPAU
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: whpRLkwvxva49hcLa1TQ4N
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: nsxTqRv7yl9rbYA81VYRuN
Processing chef: Munna Kathy
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-4.png
Image uploaded successfully: image-03eb4eacebd8ca11b707cfc569b87894b4e9bd57-1248x1517-png
Uploading chef to Sanity: Munna Kathy
Chef uploaded successfully: UCe181y7ND8VpGNVMEKqIn
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: UCe181y7ND8VpGNVMEKqyP
Processing chef: William Rumi
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-6.png
Image uploaded successfully: image-ef1c3b9ecfd9bc1aad0a931c6b4c564d6939e4f8-1248x1517-png
Uploading chef to Sanity: William Rumi
Chef uploaded successfully: nsxTqRv7yl9rbYA81VYSc9
Data import completed successfully!
```

---

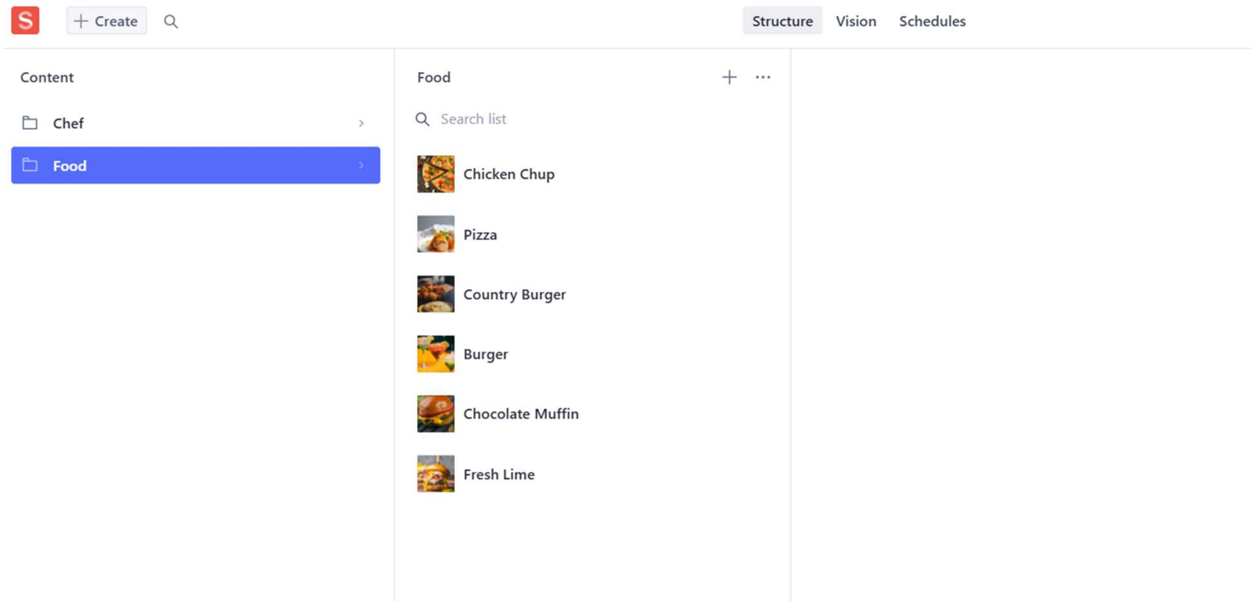
2. Verifying Data in Sanity CMS

After importing data, I verified its structure and accuracy in **Sanity Studio**.

- **Key Actions:**
  - Ensured schemas for **Food** and **Chef** were correctly configured.
  - Verified the fields (`name`, `slug`, `price`, `tags`, etc.) for each document.
  - Confirmed that images were uploaded and linked correctly.

Screenshot:

- **Sanity Studio View**



---

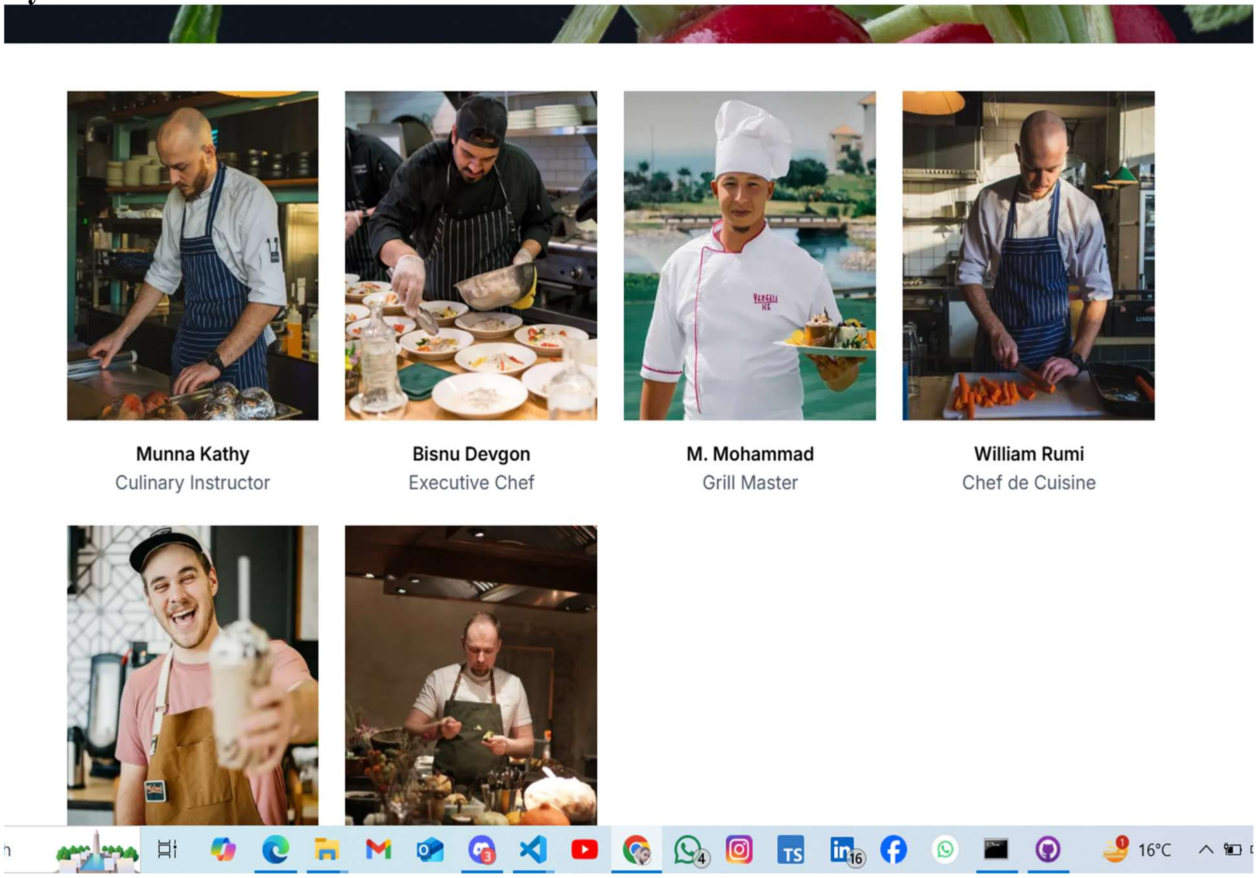
3. Displaying Data Dynamically on the UI

The data from Sanity CMS was fetched dynamically using GROQ queries and rendered in the UI.

- **Key Actions:**
  - Used **client-side fetching** in Next.js with GROQ queries to retrieve data.
  - Rendered dynamic details like name and images for both Food and Chef entities.
  - Added conditional rendering for availability and enhanced user experience.

Screenshot:

- **Dynamic Data on UI:**



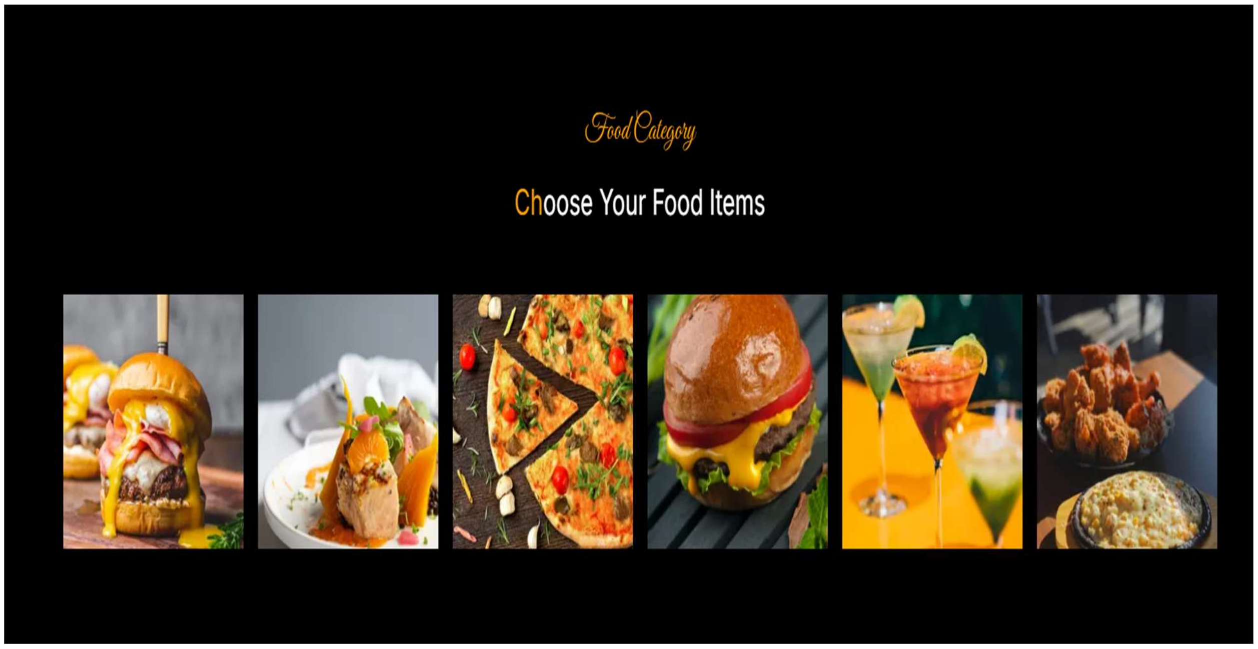
Munna Kathy
Culinary Instructor

Bisnu Devgon
Executive Chef

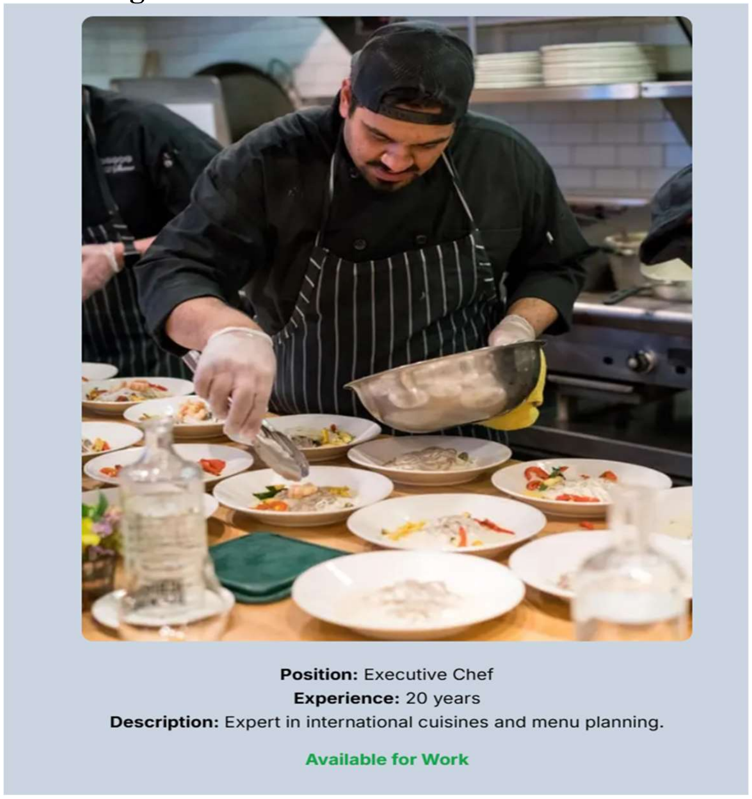
M. Mohammad
Grill Master

William Rumi
Chef de Cuisine

-
-

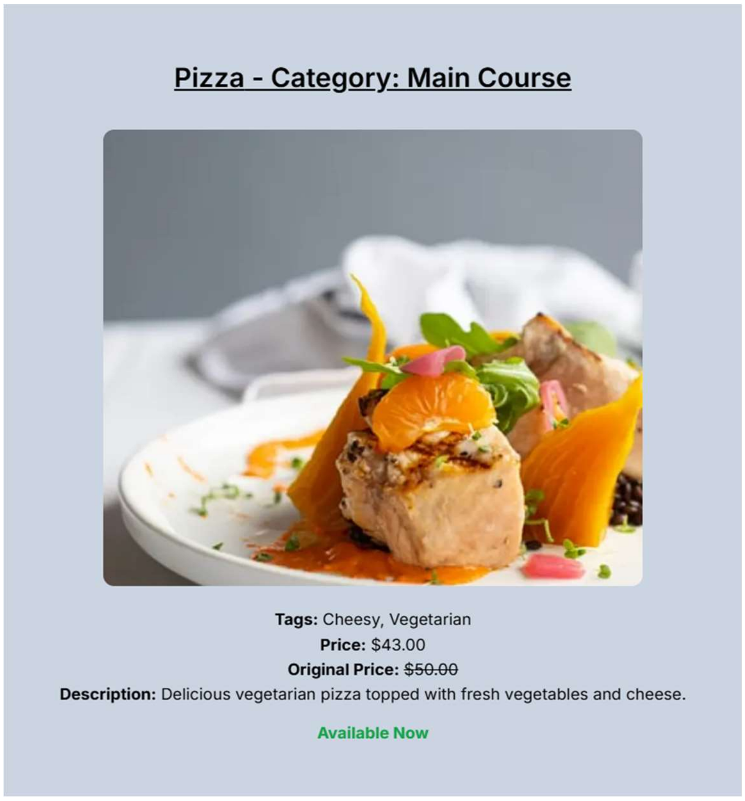


- **Detail Page View:**



Position: Executive Chef
Experience: 20 years
Description: Expert in international cuisines and menu planning.

Available for Work



Pizza - Category: Main Course

Tags: Cheesy, Vegetarian
Price: $43.00
Original Price: ~~$50.00~~
Description: Delicious vegetarian pizza topped with fresh vegetables and cheese.

Available Now

4. Implementing Dynamic Routing

Dynamic routing was achieved using **[slug].tsx** files and the `params` object in Next.js.

- **Key Actions:**
  - Used `params` to retrieve the `slug` and fetch the corresponding data from Sanity CMS.
  - Rendered dynamic pages for each Food or Chef item based on the slug.

Adding Screenshots

To provide better clarity and documentation, screenshots have been attached for each stage of the process:

1. **API Integration Logs:** Demonstrating the successful fetching of data from APIs and uploading it to Sanity.
2. **Sanity CMS Verification:** Showing schema setup, datasets, and document details in Sanity Studio.
3. **UI Rendering:** Highlighting dynamic data fetched from Sanity displayed on the application's UI.
4. **Dynamic Routes:** Illustrating slug-based routing and rendering for specific items.

Conclusion

By using Sanity CMS, GROQ queries, and dynamic routes, I successfully integrated APIs into the application and rendered the data in real-time. Each stage, from fetching API data to UI rendering, is supported by screenshots for verification and documentation.

How to Attach Screenshots

You can paste the screenshots directly into the document in your preferred tool (e.g., Microsoft Word, Google Docs). Ensure each screenshot is labeled and positioned under its corresponding section for clarity.

Let me know if you need help formatting this further!