# **Day 3: API Integration & Data Migration**

**API Integration Report**

**Process Overview**

The API integration connects an **external data source** providing information about **foods and chefs** to the **Sanity CMS**. This allows for seamless data management within the system.

## **1. API Integration Steps**

**🔹 Environment Setup**

* Configured environment variables using dotenv to securely store credentials.
* Essential environment variables included:
  + NEXT\_PUBLIC\_SANITY\_PROJECT\_ID → Sanity project identifier
  + NEXT\_PUBLIC\_SANITY\_DATASET → Dataset name
  + SANITY\_TOKEN → Authentication token for secure operations

**🔹 Establishing Sanity Client**

* Used the @sanity/client package to create a connection with Sanity CMS.
* Configured the client with:
  + Project ID
  + Dataset name
  + API version
  + Authentication token

**🔹 Fetching Data from API**

* Utilized **axios** to make **concurrent API requests** to retrieve food and chef data.
* Accessed endpoints:
  + https://sanity-nextjs-rouge.vercel.app/api/foods
  + https://sanity-nextjs-rouge.vercel.app/api/chefs

**🔹 Processing and Uploading Data**

* Iterated through the API response to extract relevant details.
* Uploaded images to **Sanity’s asset library** using client.assets.upload().
* Converted raw API data into **Sanity-compatible documents** and uploaded them using client.create().

## **2. Error Handling Strategies**

* Implemented **try-catch blocks** to capture errors during API calls and CMS operations.
* Logged errors in the console for **debugging and troubleshooting**.

## **3. Schema Enhancements**

**👨‍🍳 Chef Schema Modifications**

* **New fields added:**
  + **Role:** String field defining the chef's position (e.g., Head Chef, Sous Chef).
  + **Experience:** Numeric field representing years in the industry.
  + **Specialty:** String field listing culinary expertise.
  + **Availability:** Boolean field indicating work status.
* **Enabled hotspot cropping** for better image flexibility.

**🍔 Food Schema Modifications**

* **Essential fields included:**
  + **Category:** Defines food classification (e.g., Pizza, Beverages).
  + **Price:** Numeric field for selling price.
  + **Discounted Price:** Optional field for promotional pricing.
  + **Tags:** Array field to categorize food items.
  + **Stock Status:** Boolean field indicating availability.
* **Activated image hotspot** for adaptable display across different devices.

## **4. Migration Steps & Tools Used**

**📌 Data Migration Process**

**🔸 Step 1: Data Preparation**

* I analyzed the structure of the external API and compared it with Sanity’s schemas.
* Updated the **food and chef schemas** to accommodate all necessary fields.

**🔸 Step 2: Automating Data Import**

* Created a script (import-data.mjs) to:
  + Fetch data from the API.
  + Process and format data into Sanity-compatible structures.
  + Upload data into Sanity CMS automatically.
* Used axios for handling API requests and @sanity/client for CMS interactions.

**🔸 Step 3: Image Handling**

* Downloaded images from API using axios with responseType: 'arraybuffer'.
* Uploaded images to Sanity’s asset library, storing their references in document fields.

**🔸 Step 4: Document Creation & Data Validation**

* Mapped API fields to corresponding Sanity schema fields.
* Implemented **fallback values** for missing or optional fields to prevent errors.

## **5. Tools & Technologies Used**

**🔹 Node.js Modules**

* axios → For making API requests.
* dotenv → For managing environment variables securely.
* @sanity/client → For interacting with Sanity CMS.

**🔹 Sanity Features Utilized**

* **Asset Management** → For handling image uploads.
* **API Versioning** → To ensure compatibility with future schema updates.

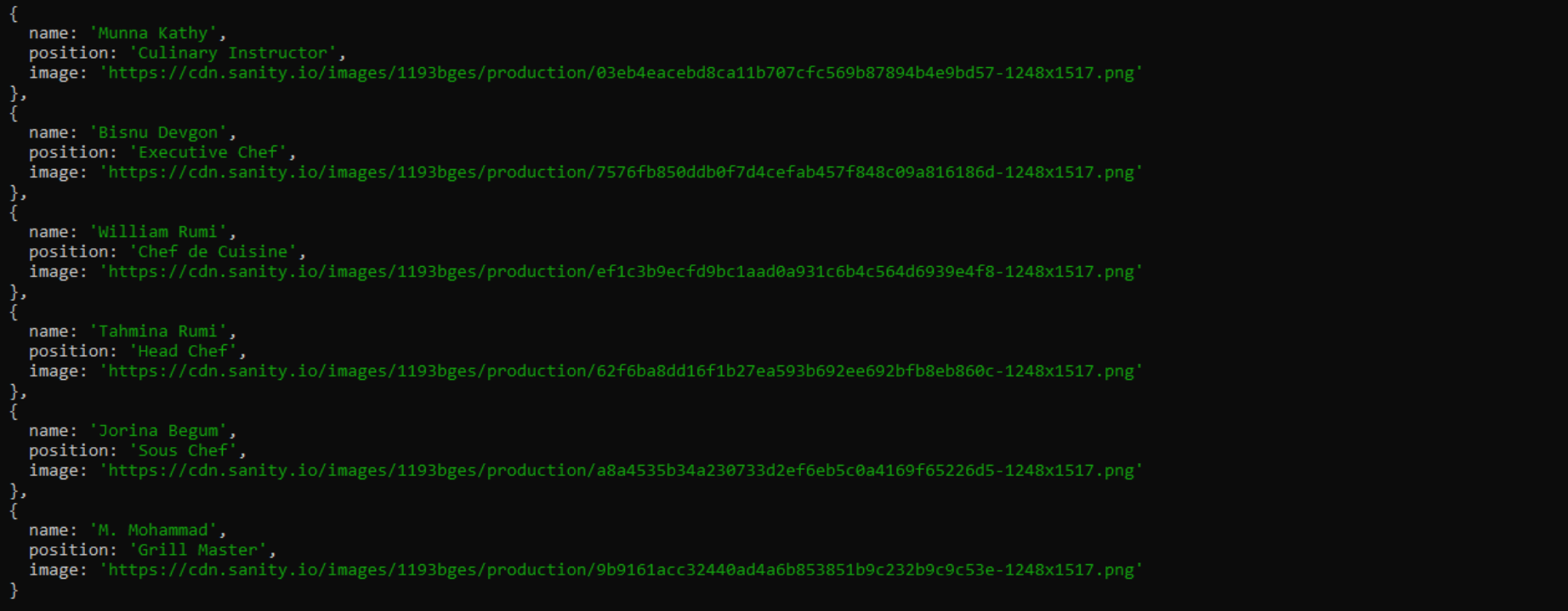
**🔹 Additional Utilities**

* fileURLToPath and path → Used for resolving file paths efficiently.

## **Overview & Key Achievements**

**📌 Chef API Calls**

Fetched chef-related data from the API, including details such as name, position, experience, specialty, and availability.



**📌 Food API Calls**

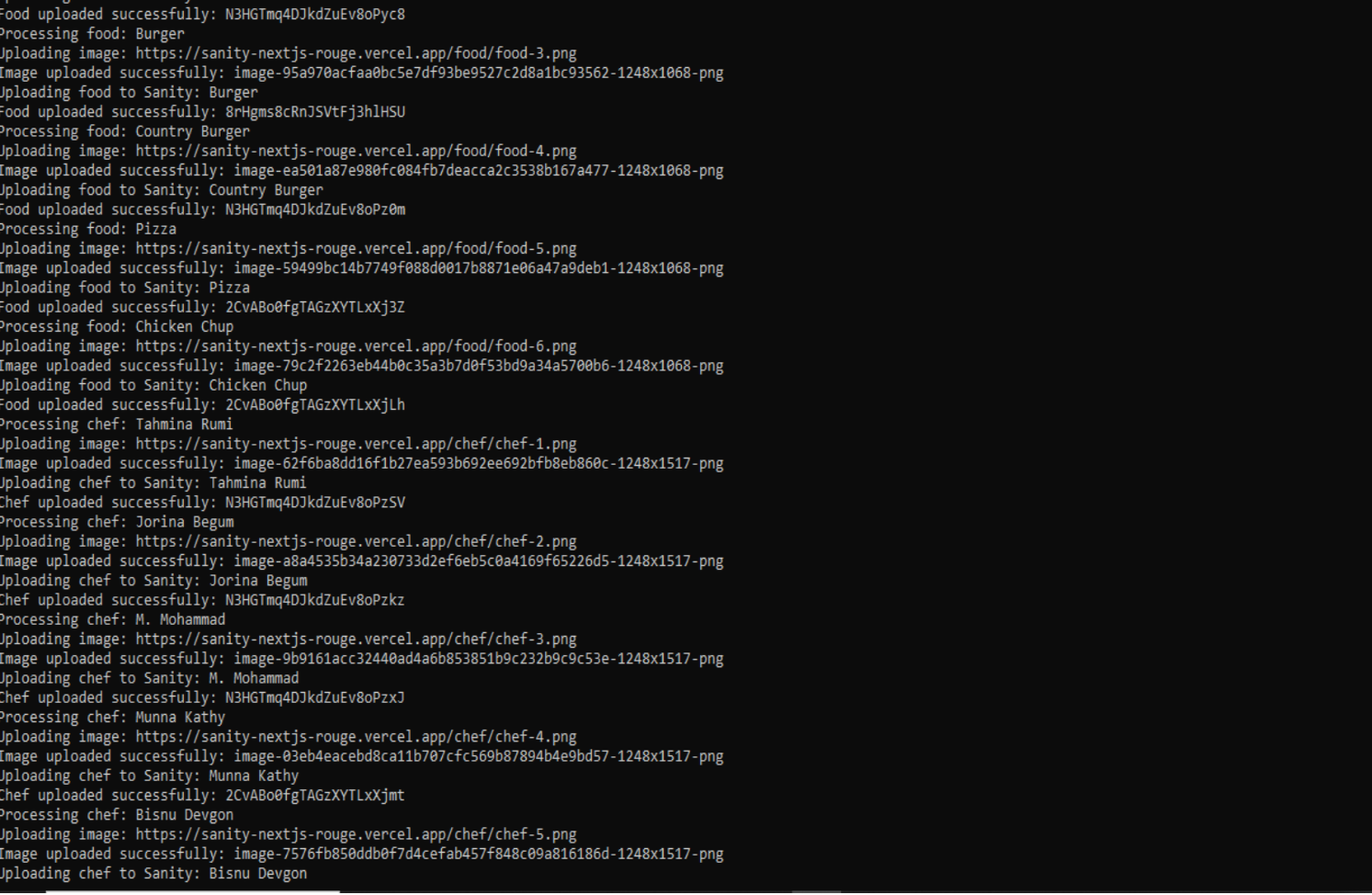
Successfully initiated API requests to fetch food-related data from the external API. The API responses contained structured information, including category, price, availability status, and images.



**📌 Data Successfully Uploaded to Sanity**

After processing the retrieved data, all records were **successfully stored** in **Sanity CMS**. This includes:

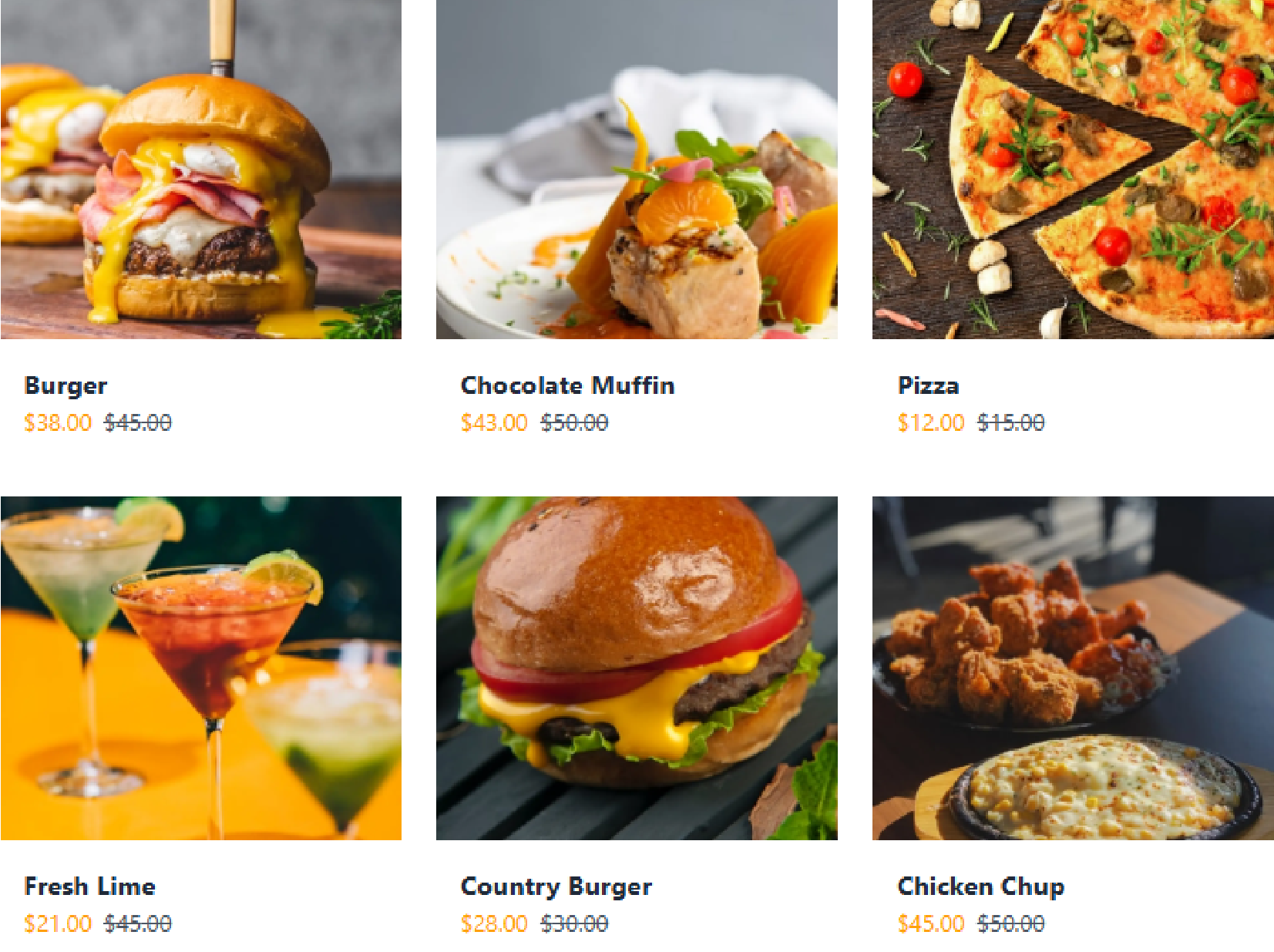
* Food items with structured metadata (e.g., category, pricing, tags).
* Chef profiles enriched with additional attributes (e.g., expertise, availability).
* Image assets uploaded and linked properly to the respective documents.



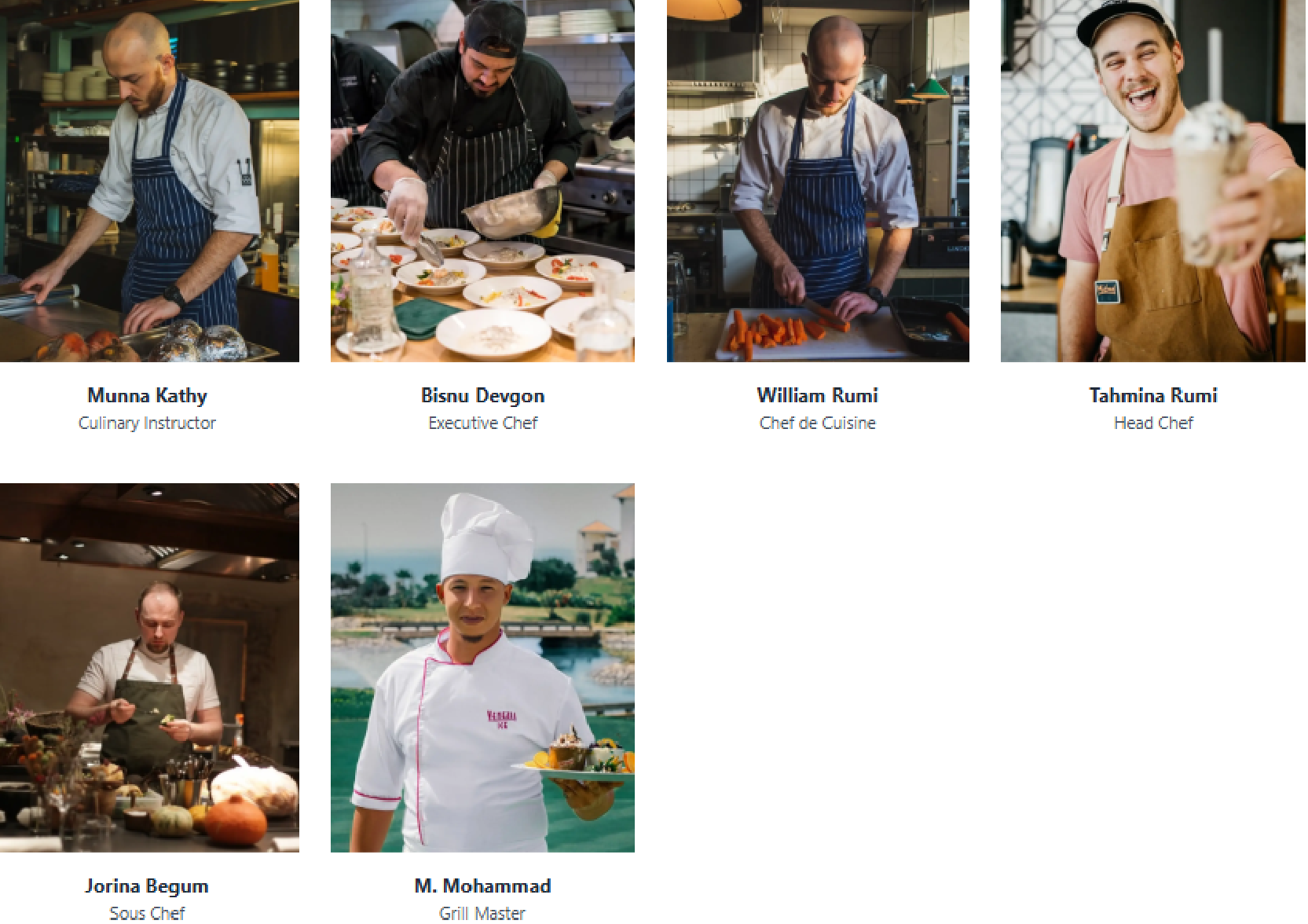
**📌 Data Successfully Rendered in Frontend**

Verified that the frontend correctly retrieves and displays data from **Sanity CMS**, ensuring a seamless user experience. The following sections were tested:

* **Food Data:** Displaying food items with accurate details, including name, price, category, and images.



* **Chef Data:** Showcasing chef profiles with relevant attributes like experience and specialization.



## **Task Breakdown & Connections**

| **🏗 Step** | **🔍 What You Did** | **🔗 How It Connects to API & Sanity** |
| --- | --- | --- |
| 📥 **Fetching Food Data** | Used axios to get food data from the API. | Provided fresh food data for CMS storage. |
| 📥 **Fetching Chef Data** | Retrieved chef details via API request. | Ensured chef profiles were up-to-date. |
| 🔄 **Processing Data** | Formatted API response & structured documents. | Transformed raw data into a Sanity-compatible format. |
| 📤 **Uploading to Sanity** | Used client.create() to store documents. | Saved food and chef details into the CMS. |
| 🖼 **Uploading Images** | Sent images to Sanity using client.assets.upload(). | Ensured images were linked correctly to data. |
| ✅ **Confirming Data in CMS** | Checked if data appeared in Sanity. | Verified the migration was successful. |
| 🎨 **Displaying on Frontend** | Rendered food and chef data in UI. | Showed real-time information on the website. |