

Day 3 - API Integration Report - Q-Commerce Food Tuck

API Integration Process

1. Understanding API Requirements

- a. Identified required endpoints for fetching and updating data.
- b. Reviewed API documentation for authentication and rate limits.

2. Setting Up API Calls in Next.js

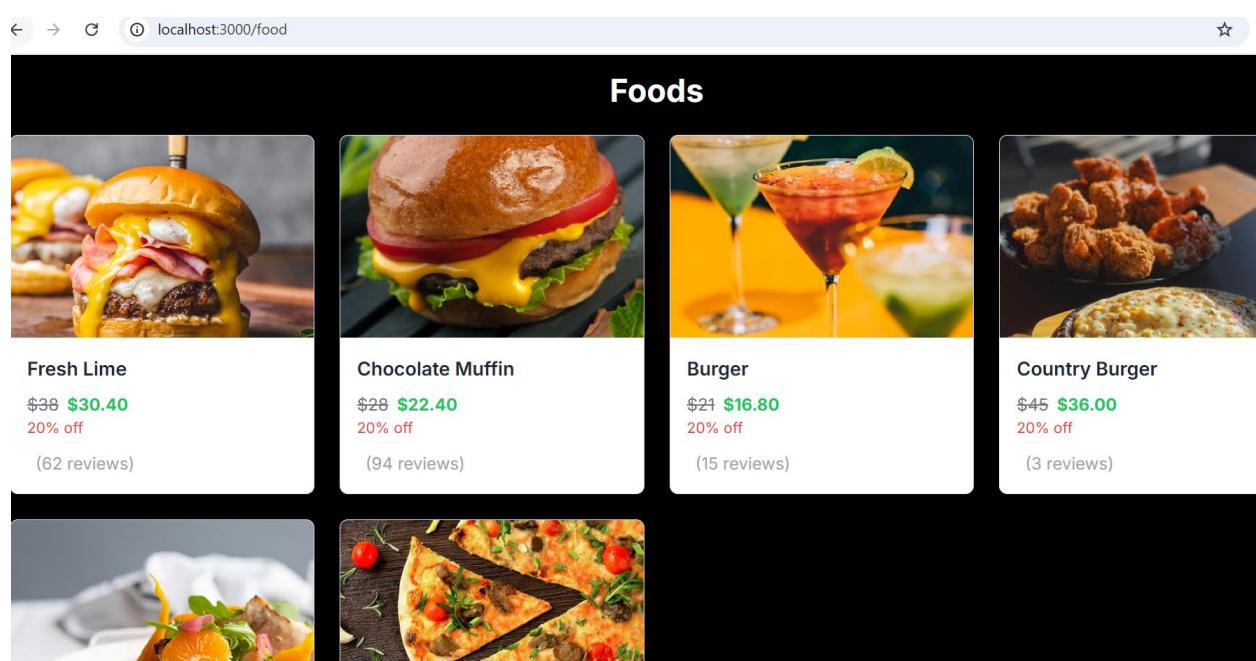
- a. Utilized fetch and axios for making API requests.
- b. Implemented API handling in api / folder for modularity.

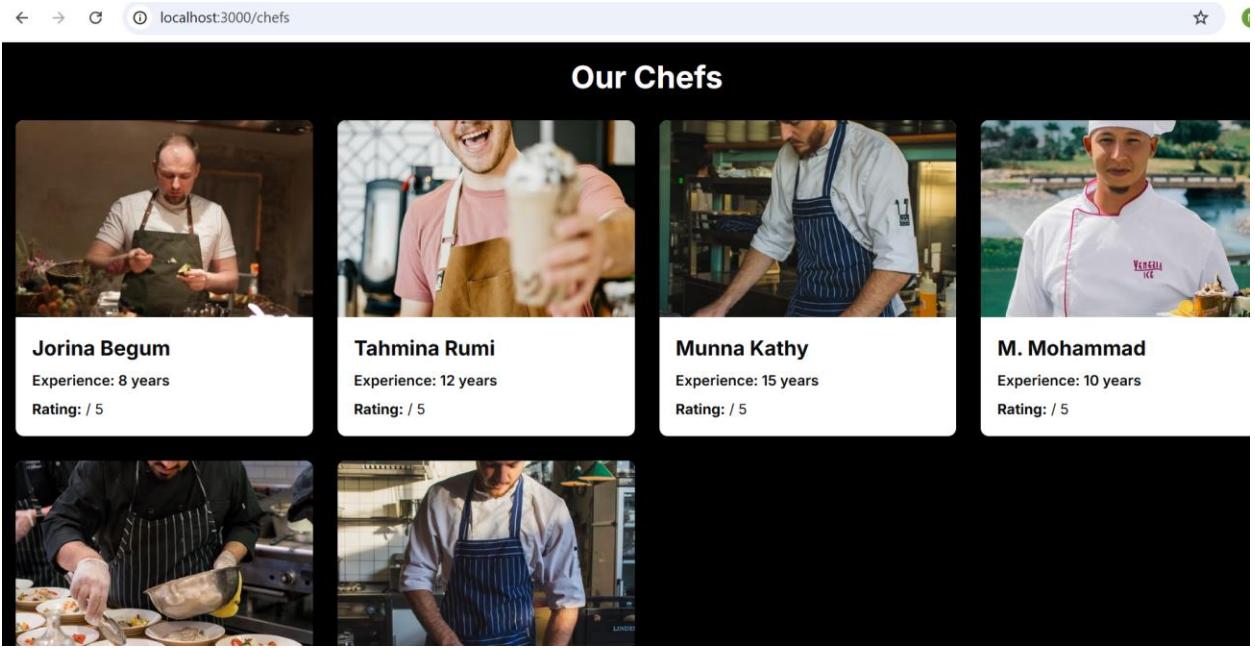
3. Authentication & Security

- a. Stored API keys securely in .env.local.
- b. Used token-based authentication where required.

4. Error Handling & Response Management

- a. Implemented error handling for timeout and invalid responses.
- b. Displayed appropriate messages for different HTTP status codes.





Adjustments Made to Schemas

- 1. Sanity CMS Adjustments**
 - a. Added missing fields for product and order schemas.
 - b. Modified field types to align with API response data.
- 2. Database Updates**
 - a. Migrated existing schema to include `inventory` and `order_status` fields.
 - b. Ensured backward compatibility for previous data.

The screenshot shows the Sanity CMS interface. On the left, there's a sidebar with 'Content' and sections for 'Chef', 'Food', and 'products'. Under 'Food', there are several items: 'burger' (selected), 'Chocolate Muffin', 'Fresh Lime', 'Chicken Chup', 'Pizza', 'Country Burger' (highlighted with a blue background), 'Burger', 'Chocolate Muffin', and 'Fresh Lime'. To the right, the 'Country Burger' item is detailed. It has a 'Tags' section with 'Tags for categorization (e.g., Best Seller, Popular, New)' and a 'Recommended' section with an 'X'. Below that is a 'Food Image' section showing two images: one of fried chicken wings and another of a dish with melted cheese. At the bottom right, there are buttons for 'Activate Window', 'Go to Settings to act', and other options.

Migration Steps and Tools Used

- 1. Database Backup**
 - a. Created a backup of existing data before migration.
- 2. Data Transformation**
 - a. Used scripts to convert old data formats to new structures.
 - b. Validated data types and constraints.
- 3. Migration Execution**
 - a. Used `Sanity CLI` for importing adjusted schema.
 - b. Verified data consistency post-migration.

Screenshots

- API calls tested in Postman.
- Successfully displayed data on the frontend.
- Populated fields in Sanity CMS.

```

Chef uploaded successfully: sDoE4Hn6nkBPnPnCufhlpzPr
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: xaIbDd1g3ehznCa7cmJA6
Processing chef: Munna Kathy
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-4.png
Image uploaded successfully: image-03eb4eacebd8ca11b707cfcc569b87894b4e9bd57-1248x1517.png
Uploading chef to Sanity: Munna Kathy
Chef uploaded successfully: xaIbDd1g3ehznCa7cmJUQ
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: sDoE4Hn6nkBPnPnCufhlpzJb
Processing chef: William Rumi
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-6.png
Image uploaded successfully: image-e1c3b9ecfd9bc1aa0a931c6b4c564d6939e4f8-1248x1517.png
Uploading chef to Sanity: William Rumi
Chef uploaded successfully: xaIbDd1g3ehznCa7cmKsY
Data import completed successfully!

```

Best Practices Followed

- 1. Use .env for sensitive data storage.**
- 2. Clean coding practices:**
 - a. Descriptive variable names.
 - b. Modularized functions.
 - c. Added comments for clarity.
- 3. Validated data during migration:**
 - a. Ensured field type consistency.
 - b. Logged errors for further debugging.
- 4. Detailed Documentation:**
 - a. Included screenshots, scripts, and testing notes.
 - b. Maintained a changelog for schema updates.
- 5. Version Control:**
 - a. Frequent commits with meaningful messages.
 - b. Tagged major milestones.
- 6. Comprehensive Testing:**
 - a. Tested API endpoints with Postman.
 - b. Handled edge cases for missing or invalid data.
- 7. Peer Review:**
 - a. Shared code for feedback.

- b. Incorporated improvements.

Self-Validation Checklist

Task	Status
API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next.js	✓
Submission Preparation	✓

- API Integration and Data Migration for the Q-Commerce Food Tuck project. 

BY: UROOJ MEMON