Day 3 - API Integration and Data Migration

Objective:

Integrate APIs and migrate data into the Sanity CMS for the Q-commerce restaurant website. This involves populating your CMS with data such as food items, categories, and availability while ensuring seamless integration with the Next.js frontend. The process replicates real-world practices and prepares you to handle dynamic and scalable marketplaces.

Key Learning Outcomes:

1. **Integrate APIs into your Next.js project to fetch dynamic data.**
2. **Migrate data from APIs into the Sanity CMS.**
3. **Validate and align schemas to match the provided API or external data sources.**
4. **Develop error-handling mechanisms for robust API integration.**

API Overview:

Provided APIs:

For your restaurant template, use the following API references to populate the Sanity CMS:

- **Product Listings (Food Items): /api/food**
- **Categories: /api/categories**
- **Availability: /api/availability**

API Base URL:

https://restaurant-qcommerce.vercel.app

Steps for Day 3:

1. Understand the Provided API:

- **Review the API endpoints using tools like Postman or browser developer tools.**
- **Identify key fields:**
  - **Food Items: Name, price, original price, category, tags, image, description, availability.**
  - **Categories: Name, description, image.**
  - **Availability: Status (In Stock/Out of Stock), timings.**

2. Validate and Adjust Your Schema:

- **Compare the provided API data with your Sanity CMS schema created on Day 2.**
- **Adjust fields in the schema to match the API data. For example:**

API Field Mapping Example:

API Field	Schema Field	Adjustment Needed?
food_name	name	No
category	category	No
availability	status	Yes (map to boolean)

3. Data Migration Options:

Option 1: Using Migration Scripts

1. **Write a script in Node.js to fetch data from the API.**
2. **Transform the data to match your Sanity schema.**
3. **Use the Sanity client library to upload data to your CMS.**

Sample Script:

```
import sanityClient from '@sanity/client';

import axios from 'axios';


const client = sanityClient({

  projectId: 'your_project_id',

  dataset: 'production',

  useCdn: false,

  token: 'your_sanity_token',

});
```

```
async function migrateData() {
  try {
    const { data: foodItems } = await axios.get('https://restaurant-qcommerce.vercel.app/api/food');

    for (const item of foodItems) {
      await client.create({
        _type: 'food',
        name: item.food_name,
        category: item.category,
        price: item.price,
        originalPrice: item.original_price,
        tags: item.tags,
        image: item.image,
        description: item.description,
        availability: item.availability === 'In Stock',
      });
    }
    console.log('Data migration completed!');
  } catch (error) {
    console.error('Migration failed:', error);
  }
}

migrateData();
```

Option 2: Manual Import

1. **Export API data as JSON.**
2. **Use Sanity's built-in import tools to upload data.**

Option 3: Using External APIs

1. **Fetch data from platforms like Shopify or WooCommerce if applicable.**
2. **Transform the data and migrate it into Sanity.**

4. API Integration in Next.js:

Steps:

1. **Create Utility Functions: Write functions to fetch data from the provided API.**
2. **export const fetchFoodItems = async () => {**
3. **const response = await fetch('https://restaurant-qcommerce.vercel.app/api/food');**
4. **return response.json();**
5. **};**
6. **Render Data in Components: Fetch and display the data in your components.**
7. **import { fetchFoodItems } from '@/utils/api';**
8.
9. **export default function FoodList() {**
10. **const [foodItems, setFoodItems] = useState([]);**
11.
12. **useEffect(() => {**
13. **fetchFoodItems().then(setFoodItems);**
14. **}, []);**
15.
16. **return (**
17. ** **
18. **{foodItems.map((item) => (**
19. ** **
20. **<h2>{item.name}</h2>**
21. ** {item.price} **
22. ** **
23. **))}**
24. ** **
25. **);**
26. **}**
27. **Test API Integration:**
    - **Use tools like Postman to test endpoints.**
    - **Log API responses to ensure data consistency.**

Error Handling Tips:

- **Log errors in a centralized file for debugging.**

- **Display user-friendly error messages in the UI.**
- **Use fallback data or skeleton loaders for better user experience.**

Expected Output:

1. **Sanity CMS populated with:**
   - **Food items.**
   - **Categories.**
   - **Availability statuses.**
2. **Functional API integration displaying dynamic data on the frontend.**

Submission Requirements:

1. **A report titled: Day 3 - API Integration Report - [Your Restaurant Name]**
2. **Include:**
   - **API integration process.**
   - **Adjustments made to schemas.**
   - **Migration steps and tools used.**
3. **Screenshots of:**
   - **API calls.**
   - **Data displayed on the frontend.**
   - **Populated Sanity CMS fields.**
4. **Code snippets for API integration and migration scripts.**

Best Practices:

1. **Use .env files for storing sensitive data.**
2. **Follow clean coding practices:**
   - **Use descriptive variable names.**
   - **Modularize functions for reusability.**
   - **Add comments for complex logic.**
3. **Validate data during migration:**
   - **Check field types and constraints.**
   - **Log discrepancies.**
4. **Document every step thoroughly:**
   - **Include screenshots, scripts, and notes.**
5. **Test thoroughly:**
   - **Handle edge cases like empty responses or invalid data.**
   - **Use Postman for endpoint validation.**

Day 3 Checklist:

Task	Completed?
API Understanding	[ ]
Schema Validation	[ ]
Data Migration	[ ]
API Integration in Next.js	[ ]
Submission Preparation	[ ]

FAQs:

1. **Can we use other APIs or data sources? Yes, as long as they align with your requirements.**
2. **How do we handle schema mismatches? Adjust your schema in Sanity CMS to match the API fields or map the fields during migration.**
3. **What if we're new to API integration? Start with simple API calls using Postman or browser tools. Follow the example scripts for guidance.**
4. **Can we manually add data to Sanity? Yes, especially for small datasets. However, learning migration scripting is beneficial for larger datasets.**
5. **What should we submit? A report with API integration and migration details, screenshots, and code snippets.**

Umama Rajput