Contents

API Integration and Data Migration Report 2

1. **API Integration Process:** 2
   - API Setup: 2
   - Fetching Data: 2
   - Populating the Frontend: 2
2. **Adjustments Made to Schemas** 2
   - Orders Schema: 2
   - Food and Chef Schema Usage: 2
3. **Migration Steps and Tools Used:** 3
   - Exporting Data from API: 3
   - Data Formatting: 3
   - Inserting Data into Sanity: 3
   - Testing Data Integration: 4

Prepared by Misbah Nayab

Day 3: API Integration and Data Migration Report

1. API Integration Process:

To fetch data into the project, the provided API for Food and Chef was used. Here's how the API integration process was carried out:

- ### API Setup:

  The provided API endpoints for fetching food items and chef details were integrated into the project. These endpoints allowed retrieving data to be displayed dynamically. API keys were set up to ensure secure access to the data.

- ### Fetching Data:

  A fetch() method was used to make GET requests to the API. The responses were parsed to retrieve relevant data like food items and chef details. Proper error handling (e.g., try/catch blocks) was implemented to account for issues like network failures or invalid responses.

- ### Populating the Frontend:

  Retrieved data was displayed dynamically in the UI.

2. Adjustments Made to Schemas

The initial schema drafted on paper included basic entities like Customers, Orders, Products, Payments, and Shipments. However, after assessing the provided Food and Chef schemas, the following adjustments were made:

- ### Orders Schema:

  The Order schema was created to complement the existing Food and Chef schemas. Key adjustments made include: Added items (an array of objects), which specifies the list of foods ordered along with their quantity and price. Incorporated customer details as an object with fields like firstName, lastName, email, and phone for richer data representation. Added a shippingAddress field to store the customer's delivery details, including country, city, zipCode, and address lines. Simplified and refined naming conventions for consistency (total instead of TotalAmount).

- ### Food and Chef Schema Usage:

  The provided schemas for Food and Chef were directly used, requiring no modification. These schemas were sufficient for handling menu items and chef data.

3. Migration Steps and Tools Used:

The process of data migration was executed as follows:

- ### Initialize Sanity Project:

  I initialize a new sanity project to get environment variables. Those environment variablers were then stored in .env.local file in my project.

- ### Exporting Data from API:

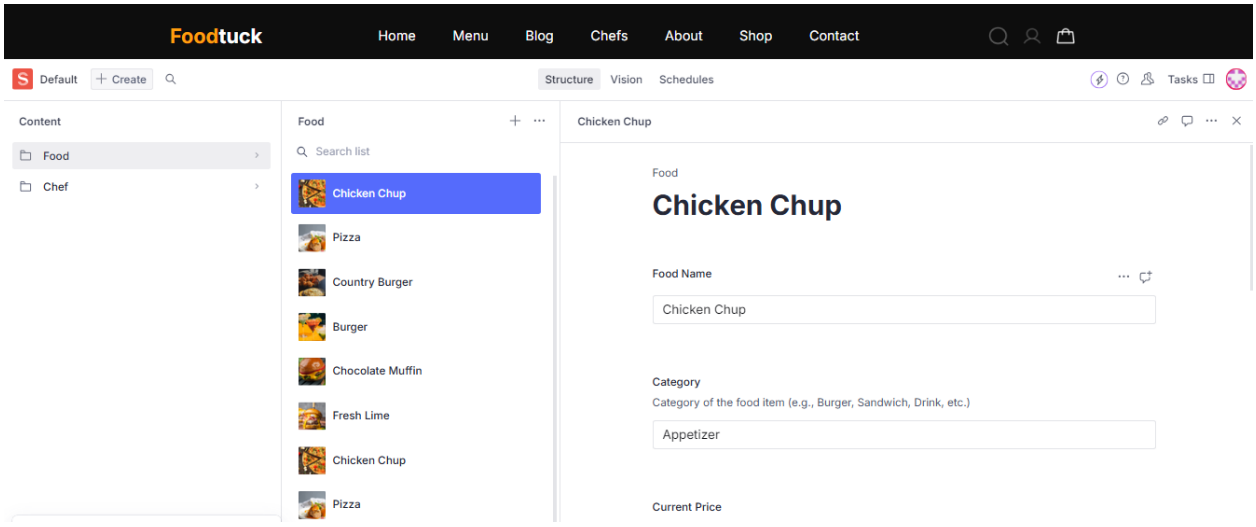
  Data from the provided API endpoints for food items and chef information was fetched and stored in the project.

- ### Data Formatting:

  JSON data received from the API was structured to align with the project's requirements.

- ### Inserting Data into Sanity:

  The @sanity/client library was used to interact with Sanity. The provided script was used to import data fetched from the API into Sanity.



- Testing Data Integration:

  The app was tested to ensure data flowed correctly from API to Sanity and then to the UI.