Day 3 - API Integration Report - Food Marketplace

Prepared by: Amna Khalil **Date:** 18 Jan 2025

1. Introduction

This report outlines the accomplishments of Day 3 for the Food Marketplace project, developed by Team 9. The primary focus of this phase was the integration of the Sanity CMS API, defining schemas for food-related data, and executing a structured data migration process into the local database.

2. API Integration

The Sanity CMS API was integrated to enable seamless communication between the backend and the content management system. Key configurations included:

- Utilizing the **Project ID** and **API Token** for secure connection to Sanity CMS.
- Employing **environment variables** to securely store sensitive credentials, enhancing security and maintainability.

This integration established a robust foundation for structured data storage and retrieval.

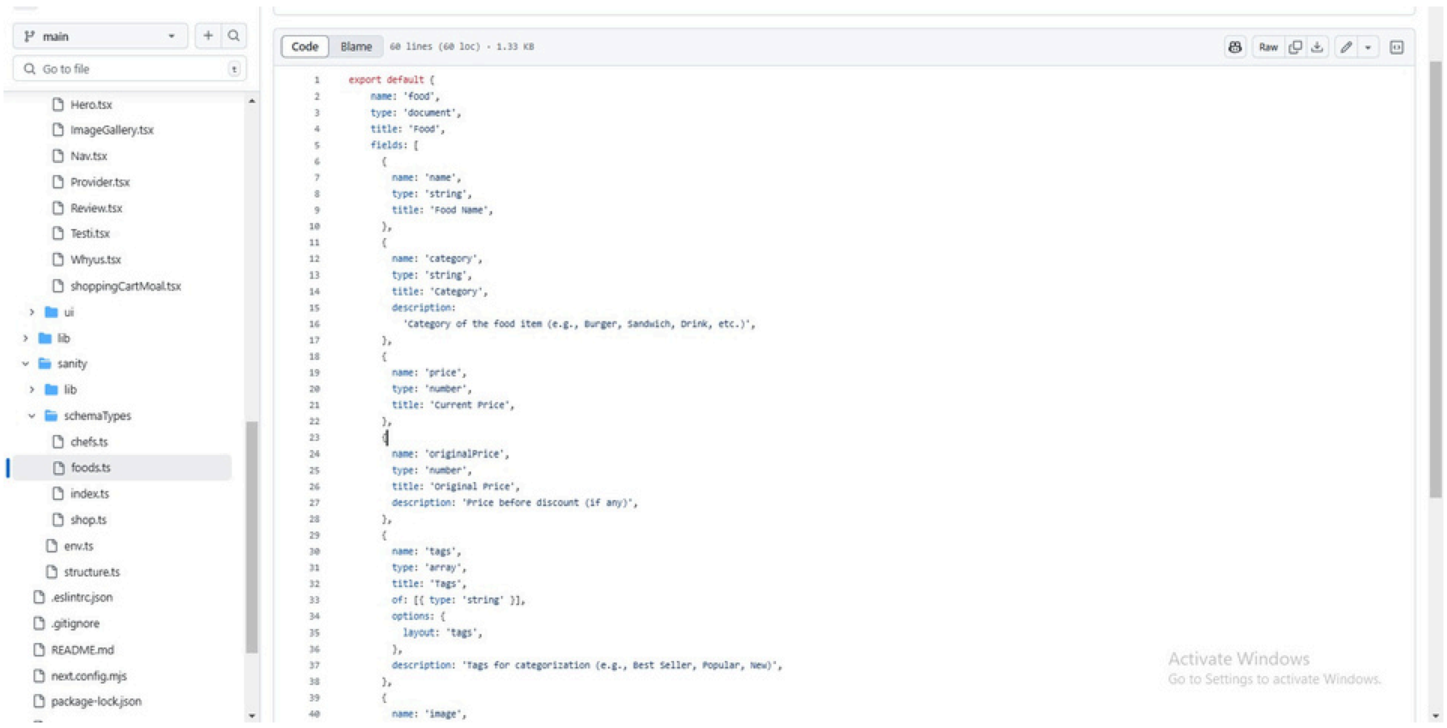
3. Schema Creation

Two schemas were created to e ectively organize and manage data:

Food Schema (food.ts)

Designed to store details about food items, this schema includes the following fields:

- **Name**: Name of the food item (String)
- **Description**: Detailed description of the food item (Text)
- **Price**: Price of the food item (Number)
- **Image**: Image representing the food item (Image)



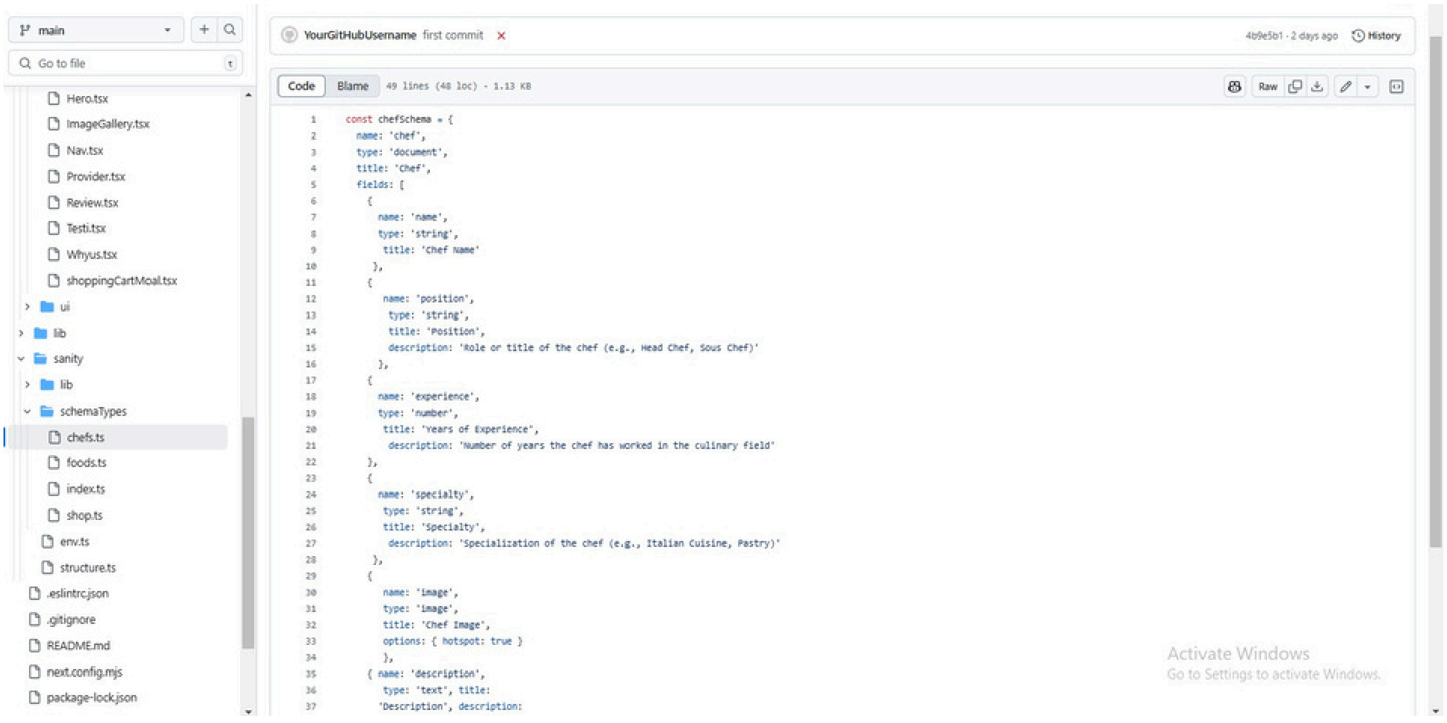
```
export default {
    name: 'food',
    type: 'document',
    title: 'Food',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Food Name',
      },
      {
        name: 'category',
        type: 'string',
        title: 'Category',
        description:
          'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
      },
      {
        name: 'price',
        type: 'number',
        title: 'Current Price',
      },
      {
        name: 'originalPrice',
        type: 'number',
        title: 'Original Price',
        description: 'Price before discount (if any)',
      },
      {
        name: 'tags',
        type: 'array',
        title: 'Tags',
        of: [{ type: 'string' }],
        options: {
          layout: 'tags',
        },
        description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
      },
      {
        name: 'image',
```

- **Chef**: Reference to the chef associated with the food item (Reference)

Chef Schema (chefs.ts)

Designed to manage information about chefs, this schema includes:

- **Name**: Name of the chef (String)
- **Bio**: Short biography or description of the chef (Text)
- **Photo**: Chef's photograph (Image)

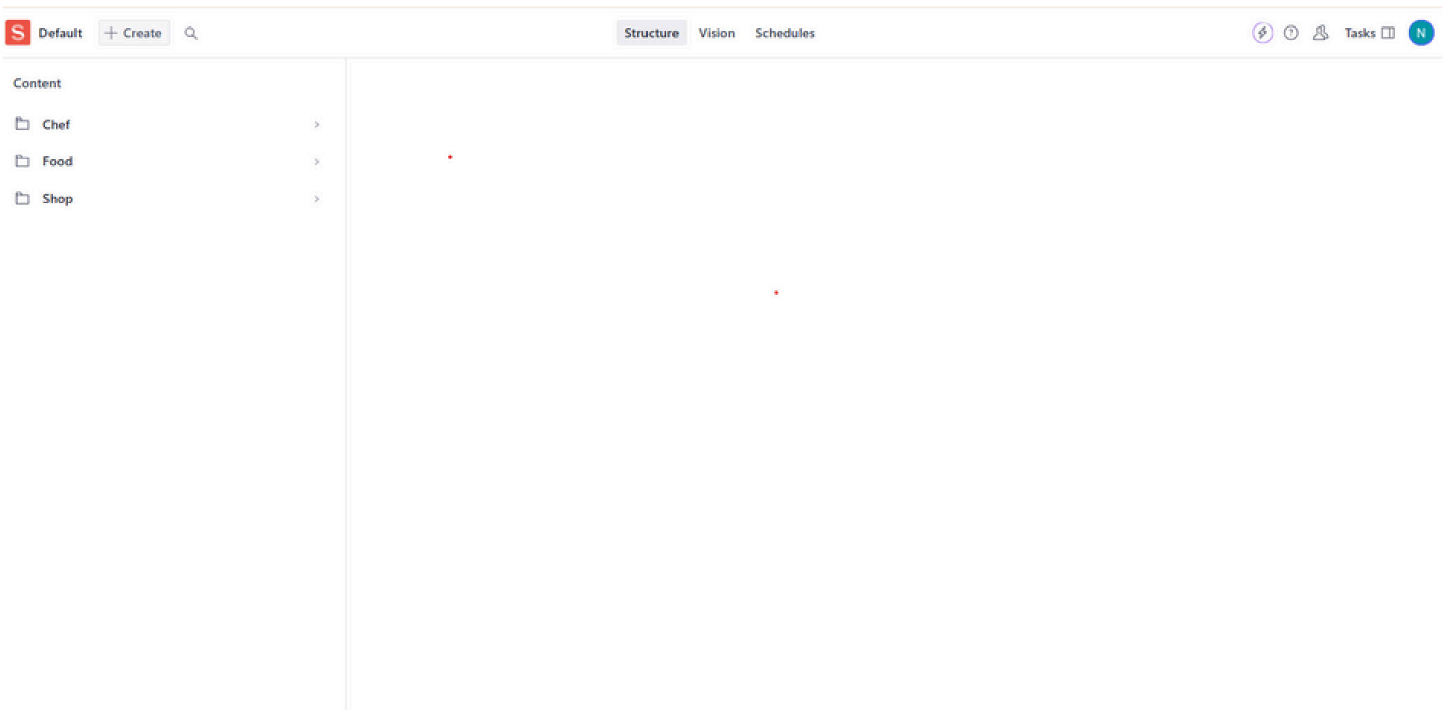


```
const chefSchema = {
  name: 'chef',
  type: 'document',
  title: 'Chef',
  fields: [
    {
      name: 'name',
      type: 'string',
       title: 'Chef Name'
     },
    {
       name: 'position',
        type: 'string',
        title: 'Position',
        description: 'Role or title of the chef (e.g., Head Chef, Sous Chef)'
      },
    {
      name: 'experience',
      type: 'number',
       title: 'Years of Experience',
        description: 'Number of years the chef has worked in the culinary field'
    },
    {
      name: 'specialty',
       type: 'string',
       title: 'Specialty',
        description: 'Specialization of the chef (e.g., Italian Cuisine, Pastry)'
     },
    {
       name: 'image',
       type: 'image',
       title: 'Chef Image',
       options: { hotspot: true }
       },
    { name: 'description',
       type: 'text', title:
      'Description', description:
```

These schemas ensured structured, queryable data to power the Food Marketplace application.

4. Data Migration

The data migration process involved manually transferring food and chef data from Sanity CMS to the local application. This process ensured a deeper understanding of the data structure and its alignment with the application's requirements. The key steps were as follows:



Manual Data Fetch and Processing

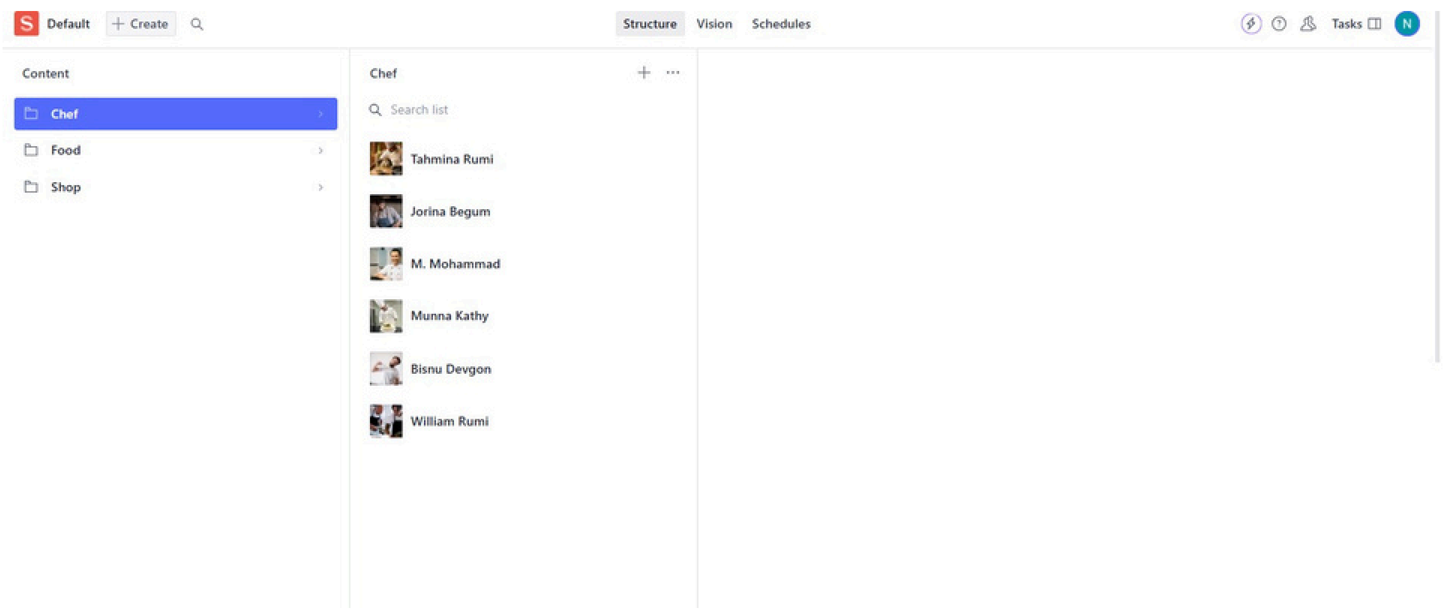
- **Connection Setup**: The Sanity CMS API was configured using the **Project ID** and **API Token** to establish secure communication.
- **Data Querying**: Data was fetched by executing a query manually within the development environment.
- **Data Analysis**: The fetched data was manually inspected and formatted as per the application's schema requirements.

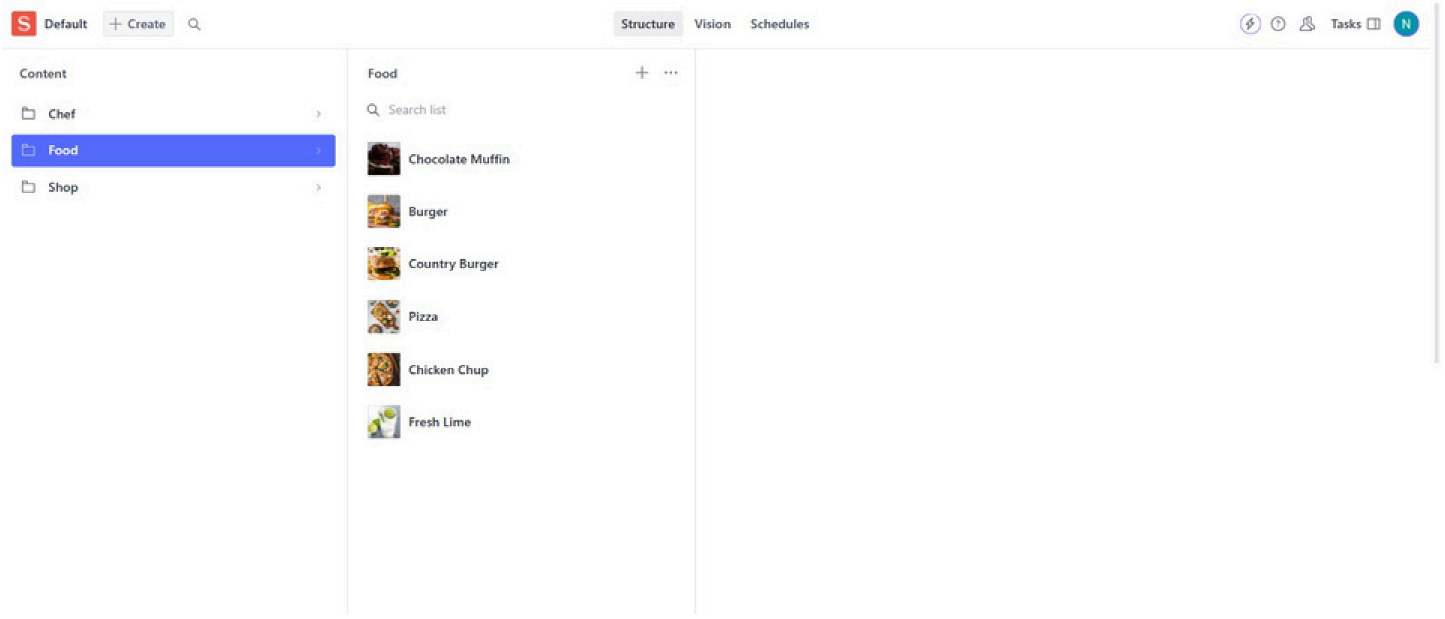
Verification

- The data was logged and reviewed in the console to verify its accuracy and completeness.
- After processing, the data was tested within the local environment to ensure compatibility with the application.

Conclusion

This manual approach provided valuable insights into the data structure and helped fine-tune the schemas (food.ts and chefs.ts) before automation. It also confirmed that the Sanity CMS API was functioning as expected for data retrieval.





6. Screenshots

Include the following screenshots in the MS Word version of this report:

1. **Schema Folder**: Displaying food.ts and chefs.ts files.
2. **Migration Output**: Output of the npm run importData command.
3. **Sanity Studio**: Showing the migrated food and chef data.

7. Conclusion

On Day 3, we successfully:

- Integrated the Sanity CMS API.
- Developed the food.ts and chefs.ts schemas for e cient data organization.
- Automated the data migration process using the importData.mjs script.
- Verified successful migration by inspecting both local and Sanity Studio structures.

These accomplishments form a robust backend setup, paving the way for frontend integration and enhanced user interaction in subsequent development stages.