Hackathon Day 3

API Integration and Data Migration

Overview

On Day 3 of the Hackathon, I accomplished key technical milestones to advance my Quick E-Commerce Marketplace:

1. Designed and implemented schemas for **Products** in Sanity CMS.
2. Imported API data into Sanity CMS after appropriate transformations.
3. Integrated Sanity CMS with my Next.js application to fetch the data.
4. Displayed the data dynamically on my website for a seamless user experience.

Here is a detailed explanation of each step.

Step 1: Designing Sanity Schemas for Products

To efficiently manage product data, I created a schema in **Sanity CMS** tailored to the API structure. This schema organizes data for easy retrieval and updates.

Schema Details

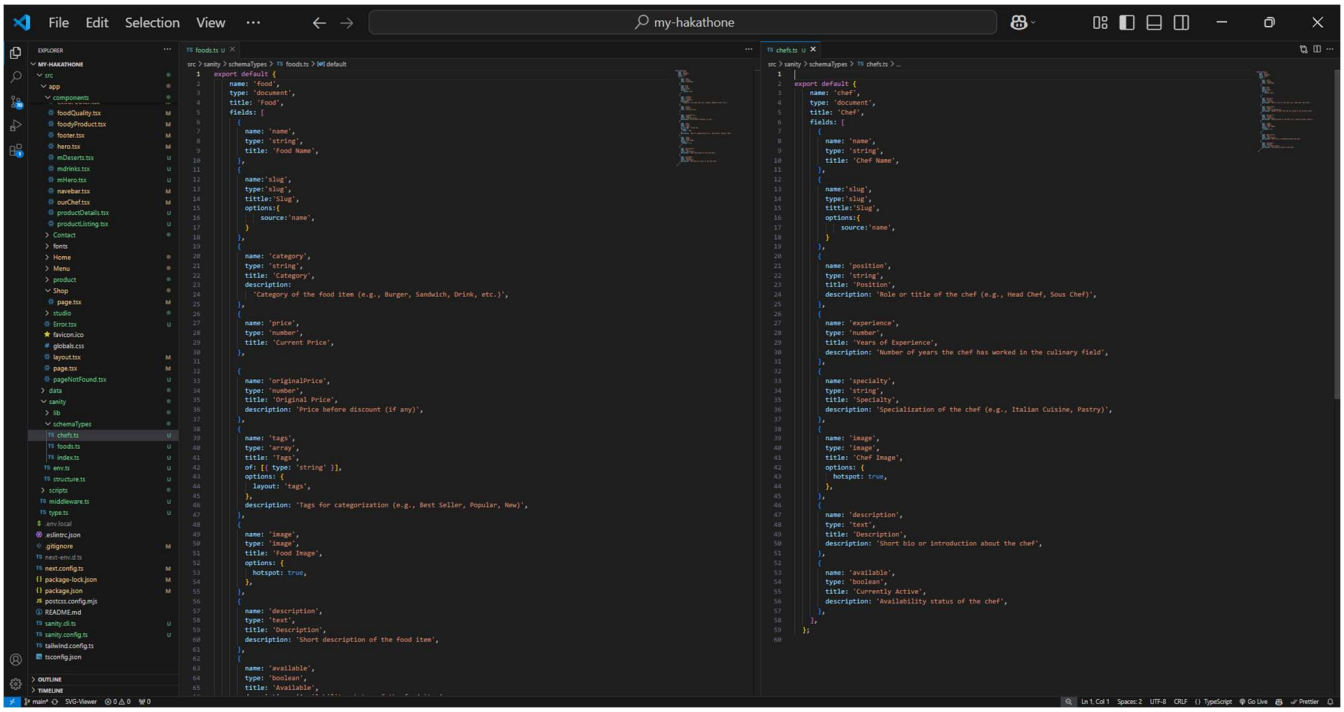
The schema includes the following fields:

Product Details:

- **Product Name (name):** The name of the product.
- **OriginalPrice :** The cost per unit.
- **DiscountPrice :** After discounted price.
- **Category :** The classification of the product (e.g., Drink, Burger).
- **Tags:** The products are healthy or unhealthy and poular or non-popular.
- **Available:** The product is available or not.
- **Description :** A concise description of the product.
- **Image URL (image):** A reference to the product image.

Chef Details:

- **Name:** The name of the chef.
- **Position :** Chefs post
- **Experience :** How many years to working it .
- **speciality :** What kind of food csn he cook?
- **Available:** The chef is available or not.
- **Description :** A concise description of the product.
- **Image URL (image):** A reference to the product image.

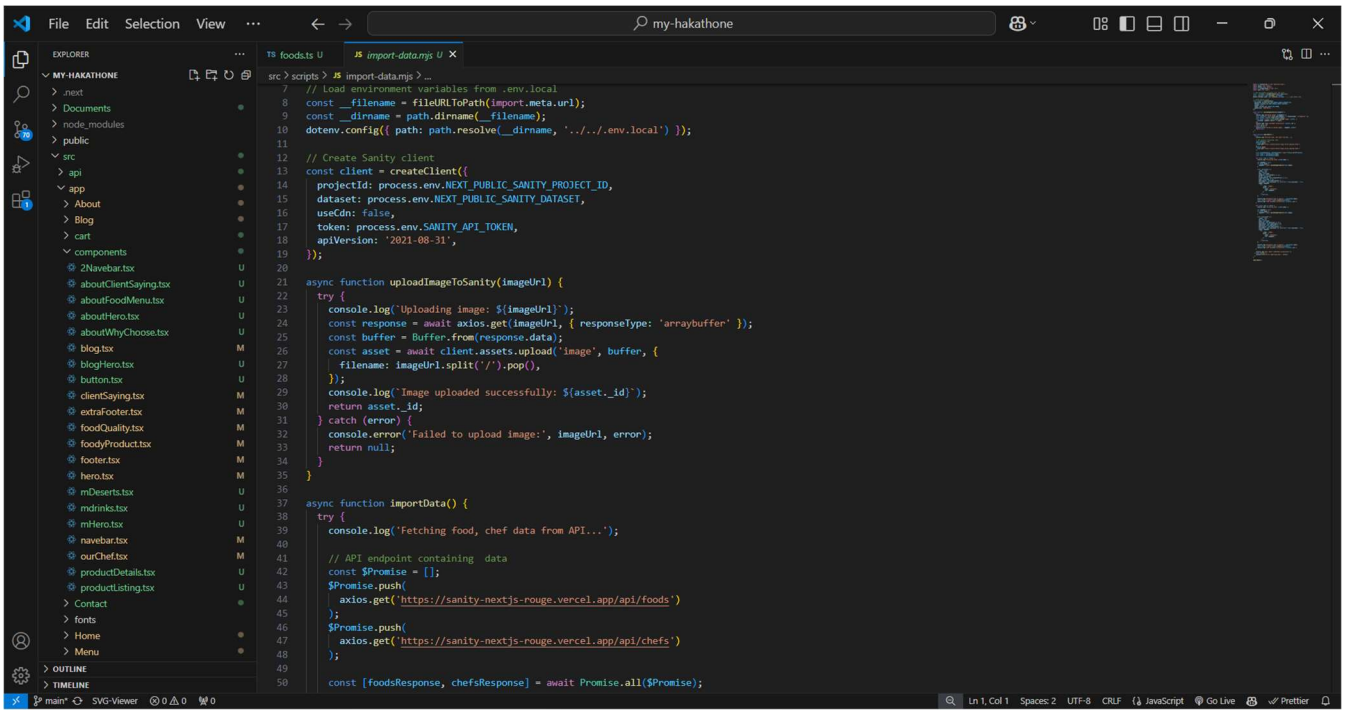


Step 2: Importing API Data into Sanity CMS

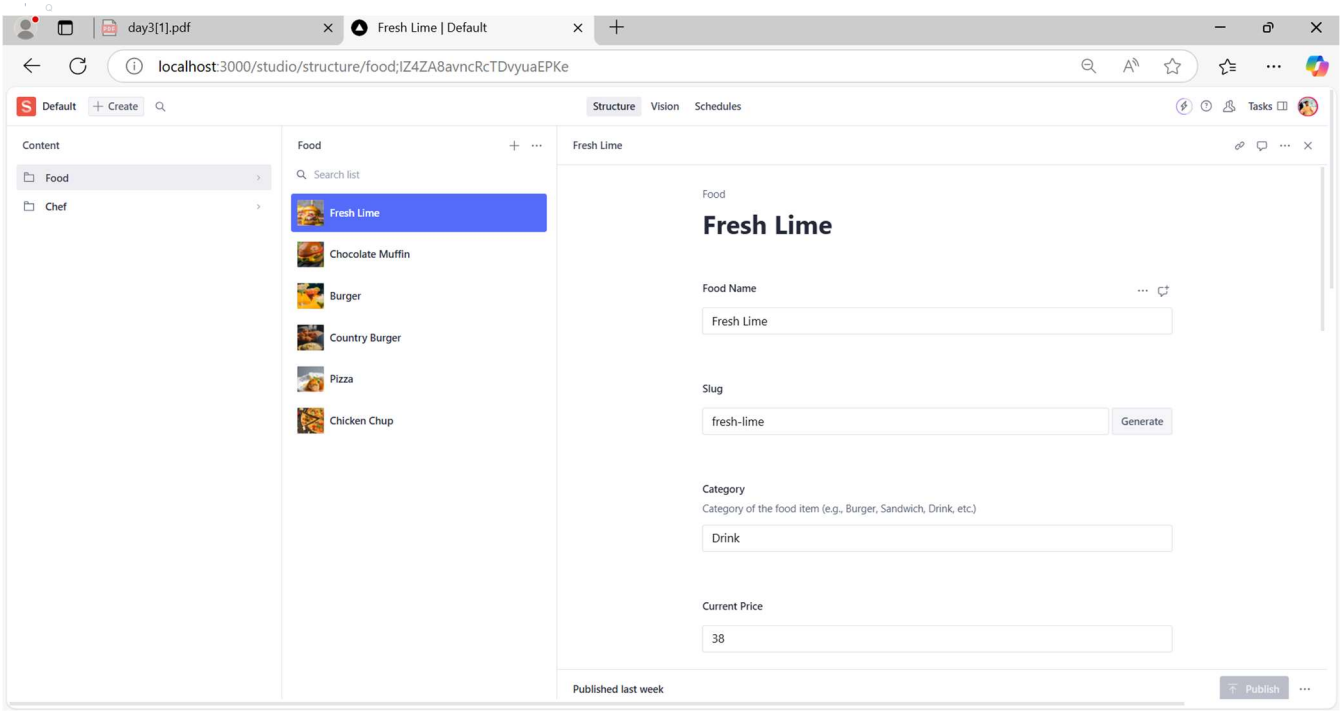
The next step involved transferring API data into Sanity CMS, ensuring the data matched the schema's structure.

Migration Process

1. **Retrieve Data from API:**
    a. Queried the API to fetch product data in JSON format.
    b. Example API Endpoint: https://api.example.com/products.
2. **Map API Fields to Schema:**
    a. Transformed API field names to align with the schema in Sanity CMS.
    b. Example: api_product_name was mapped to name in the schema.
3. **Upload Data to Sanity CMS:**
    a. Used a custom Node.js script for automation.
    b. Migration script example:

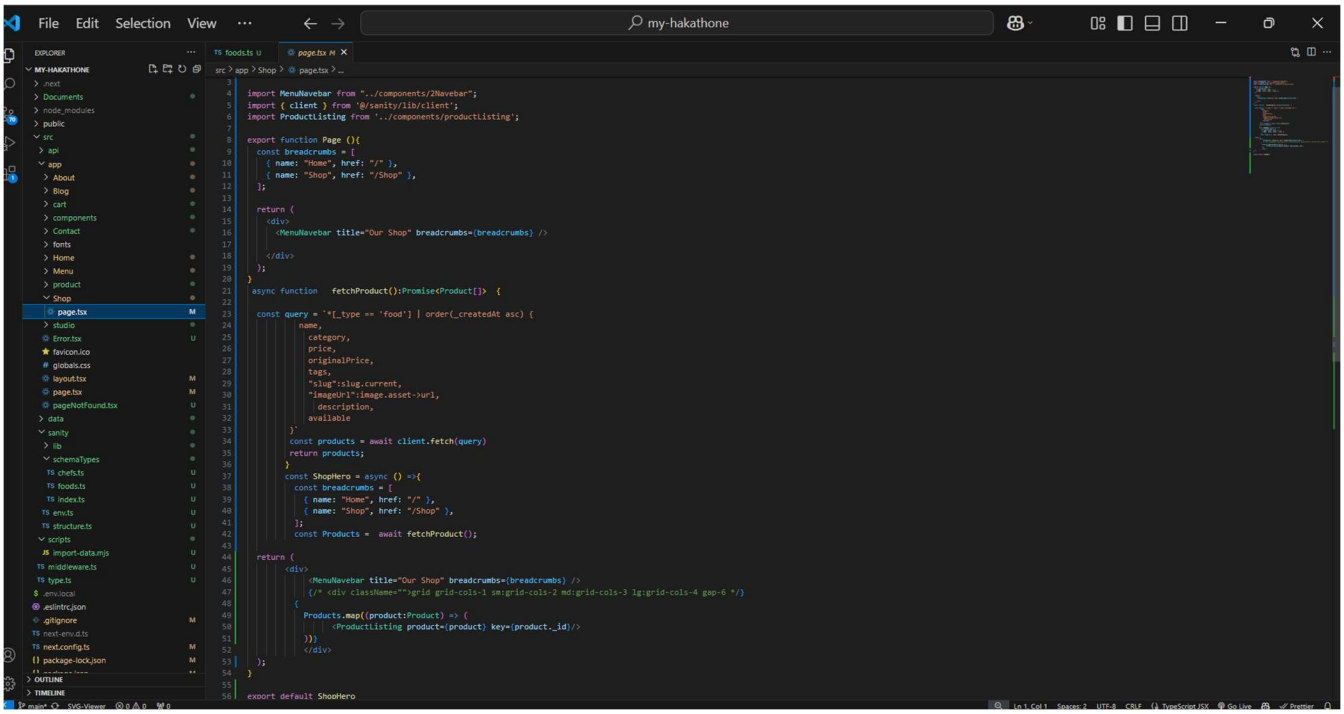


- **Verify Data:**
- Checked the Sanity CMS dashboard to confirm data accuracy and completeness



Step 3: Fetching Data in Next.js

With the data available in Sanity CMS, I integrated it into my Next.js application to fetch and display the information dynamically.

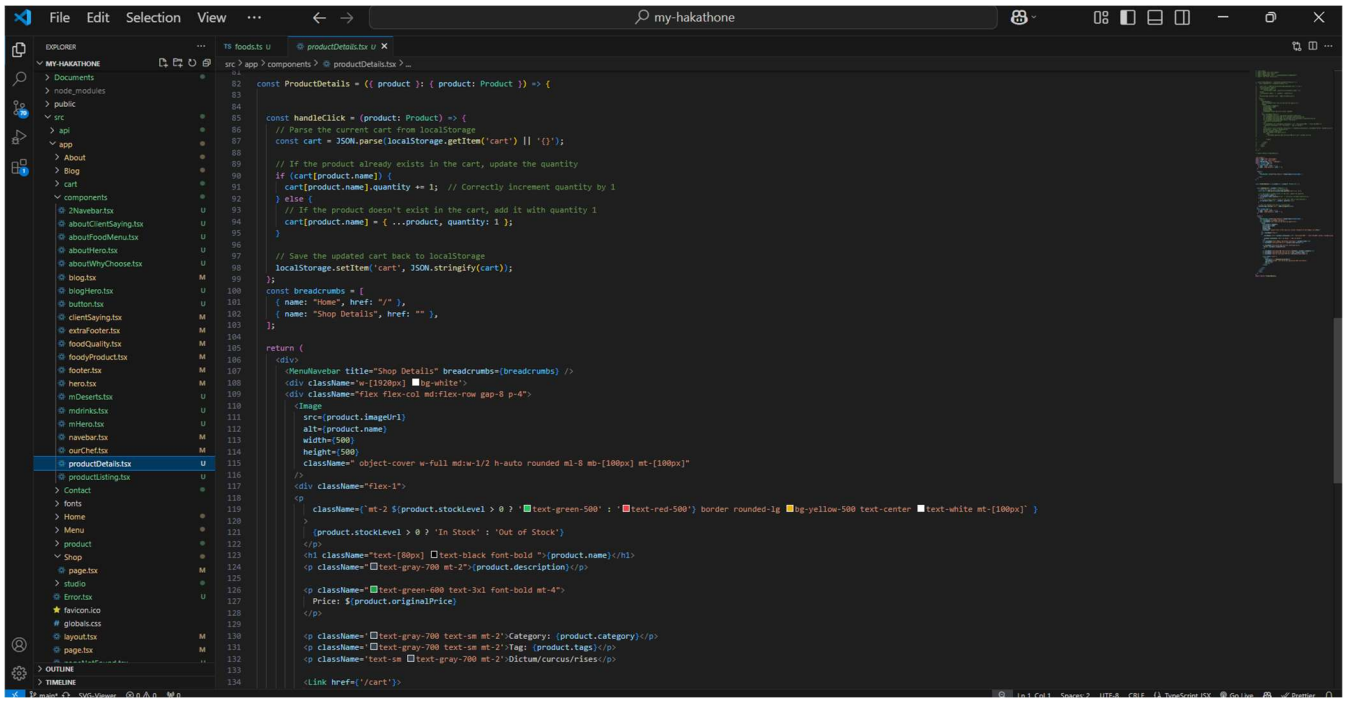


Step 4: Rendering Data Dynamically on the Website

After fetching the data, I displayed it dynamically on the website. This ensures that any updates made in Sanity CMS reflect immediately on the frontend.

Display Features

1. **Product Grid:**
    a. Rendered all products in a responsive grid layout.
    b. Included details like name, price, and image.
2. **Product Detail Page:**
    a. Showed more comprehensive details for individual products.


File Edit Selection View
my-hakathone
EXPLORER
MY-HAKATHONE
foods.ts U
productDetails.tsx U
src > app > components > productDetails.tsx > ...
const ProductDetails = ({ product }: { product: Product }) => {
const handleClick = (product: Product) => {
// Parse the current cart from localStorage
const cart = JSON.parse(localStorage.getItem('cart') || '{}');
// If the product already exists in the cart, update the quantity
if (cart[product.name]) {
cart[product.name].quantity += 1; // Correctly increment quantity by 1
} else {
// If the product doesn't exist in the cart, add it with quantity 1
cart[product.name] = { ...product, quantity: 1 };
}
// Save the updated cart back to localStorage
localStorage.setItem('cart', JSON.stringify(cart));
};
const breadcrumbs = [
{ name: "Home", href: "/" },
{ name: "Shop Details", href: "" },
];
return (

<MenuNavebar title="Shop Details" breadcrumbs={breadcrumbs} />

<Image
src={product.imageUrl}
alt={product.name}
width={500}
height={500}
className=" object-cover w-full md:w-1/2 h-auto rounded ml-8 mb-[100px] mt-[100px]"
/>

<p
className={`mt-2 ${product.stockLevel > 0 ? 'text-green-500' : 'text-red-500'} border rounded-lg bg-yellow-500 text-center text-white mt-[100px]` }
>
{product.stockLevel > 0 ? 'In Stock' : 'Out of Stock'}

<h1 className="text-[80px] text-black font-bold ">{product.name}</h1>
{product.description}

Price: ${product.originalPrice}

Category: {product.category}
Tag: {product.tags}
Dictum/curcus/rises
<Link href={'/cart'}>