

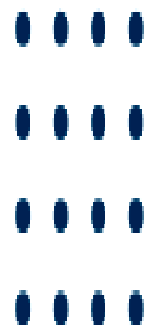


HACKATHON

Day3 API Integration and Data Migration

Created by: NADIA
nadiaawan400@gmail.com

Submitted to: AMEEN ALAM



Day 3 – API Integration and Data Migration Report – Bandage

Objective:

The goal for Day 3 was to integrate API data into Sanity CMS for the Bandage project, enabling dynamic content updates for the marketplace. Instead of manually entering data, the API integration provided a more efficient and salable solution.

1. Sanity CMS Schema Design:

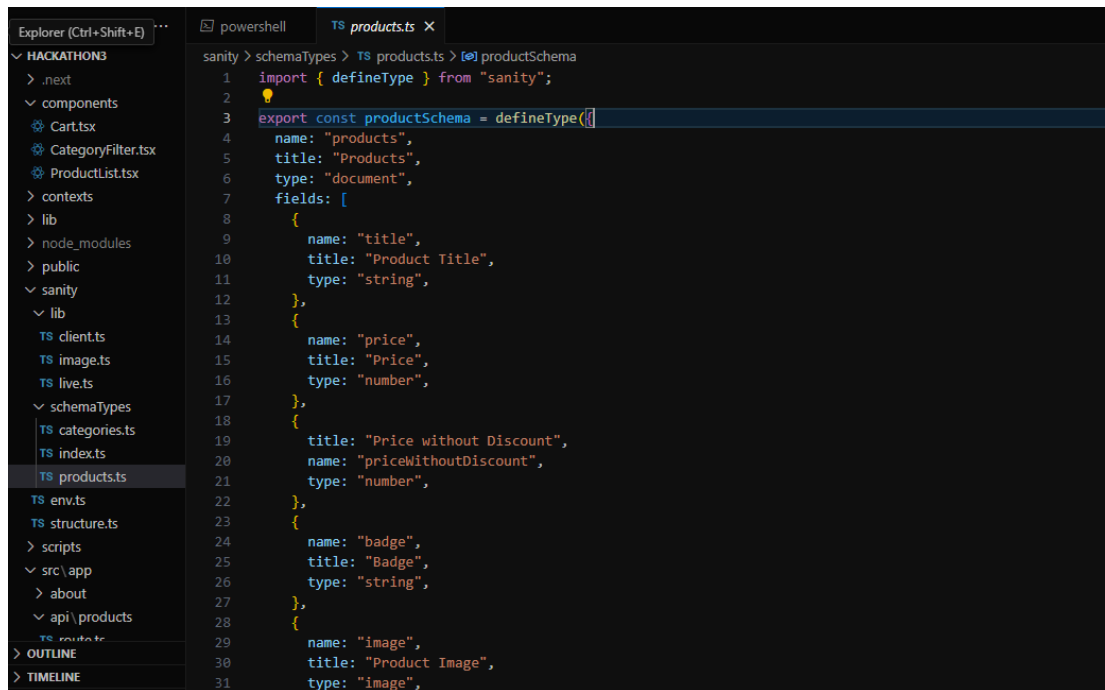
To ensure the seamless handling of product data, I designed a schema called `product` in Sanity CMS. The schema includes the following fields:

-

Main Fields:

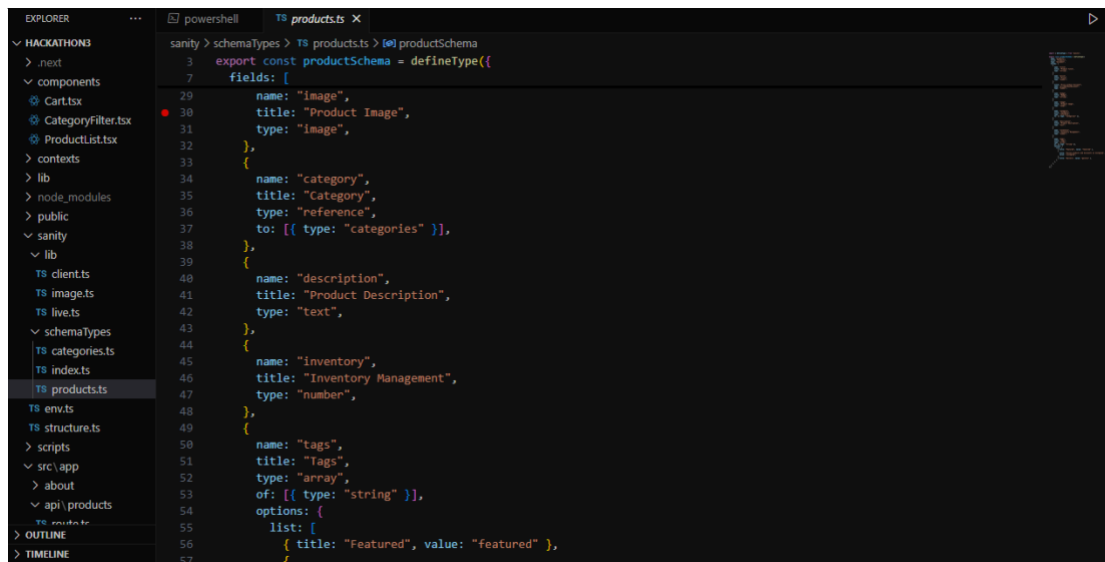
- o `title`: The product title (string type).
- o `description`: A detailed description of the product (text type).
- o `product_Image`: The main product image (image type).
- o `price`: The price of the product (number type).
- o `tags`: An array of tags to categorize the product (array of strings).
- o `discount_Percentage`: The discount percentage (number type).
- o `is_New`: A boolean flag indicating if the product is new (boolean type).

Code Snippet:



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'sanity' folder containing 'lib' and 'schemaTypes'. The 'products.ts' file is selected in the 'lib' folder. The code editor shows the definition of 'productSchema' using the 'defineType' function from 'sanity'. The schema defines fields for 'title', 'price', 'category', 'description', 'inventory', and 'tags'.

```
1 import { defineType } from "sanity";
2
3 export const productSchema = defineType({
4   name: "products",
5   title: "Products",
6   type: "document",
7   fields: [
8     {
9       name: "title",
10      title: "Product Title",
11      type: "string",
12    },
13    {
14      name: "price",
15      title: "Price",
16      type: "number",
17    },
18    {
19      title: "Price without Discount",
20      name: "priceWithoutDiscount",
21      type: "number",
22    },
23    {
24      name: "badge",
25      title: "Badge",
26      type: "string",
27    },
28    {
29      name: "image",
30      title: "Product Image",
31      type: "image",
32    },
33    {
34      name: "category",
35      title: "Category",
36      type: "reference",
37      to: [{ type: "categories" }],
38    },
39    {
40      name: "description",
41      title: "Product Description",
42      type: "text",
43    },
44    {
45      name: "inventory",
46      title: "Inventory Management",
47      type: "number",
48    },
49    {
50      name: "tags",
51      title: "Tags",
52      type: "array",
53      of: [{ type: "string" }],
54      options: {
55        list: [
56          { title: "Featured", value: "featured" },
57        ],
58      },
59    },
60  ],
61 });
```



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'sanity' folder containing 'lib' and 'schemaTypes'. The 'products.ts' file is selected in the 'lib' folder. The code editor shows the definition of 'productSchema' using the 'defineType' function from 'sanity'. The schema defines fields for 'title', 'price', 'category', 'description', 'inventory', and 'tags'.

```
3 export const productSchema = defineType({
7   fields: [
29     name: "image",
30     title: "Product Image",
31     type: "image",
32   },
33   {
34     name: "category",
35     title: "Category",
36     type: "reference",
37     to: [{ type: "categories" }],
38   },
39   {
40     name: "description",
41     title: "Product Description",
42     type: "text",
43   },
44   {
45     name: "inventory",
46     title: "Inventory Management",
47     type: "number",
48   },
49   {
50     name: "tags",
51     title: "Tags",
52     type: "array",
53     of: [{ type: "string" }],
54     options: {
55       list: [
56         { title: "Featured", value: "featured" },
57       ],
58     },
59   },
60 ],
61 });
```

```

sanity > schemaTypes > TS products.ts > productSchema
3 export const productSchema = defineType({
7   fields: [
44     {
45       name: "inventory",
46       title: "Inventory Management",
47       type: "number",
48     },
49     {
50       name: "tags",
51       title: "Tags",
52       type: "array",
53       of: [{ type: "string" }],
54       options: {
55         list: [
56           { title: "Featured", value: "featured" },
57           {
58             title: "Follow products and discounts on Instagram",
59             value: "instagram",
60           },
61           { title: "Gallery", value: "gallery" },
62         ],
63       },
64     },
65   ],
66 };

```

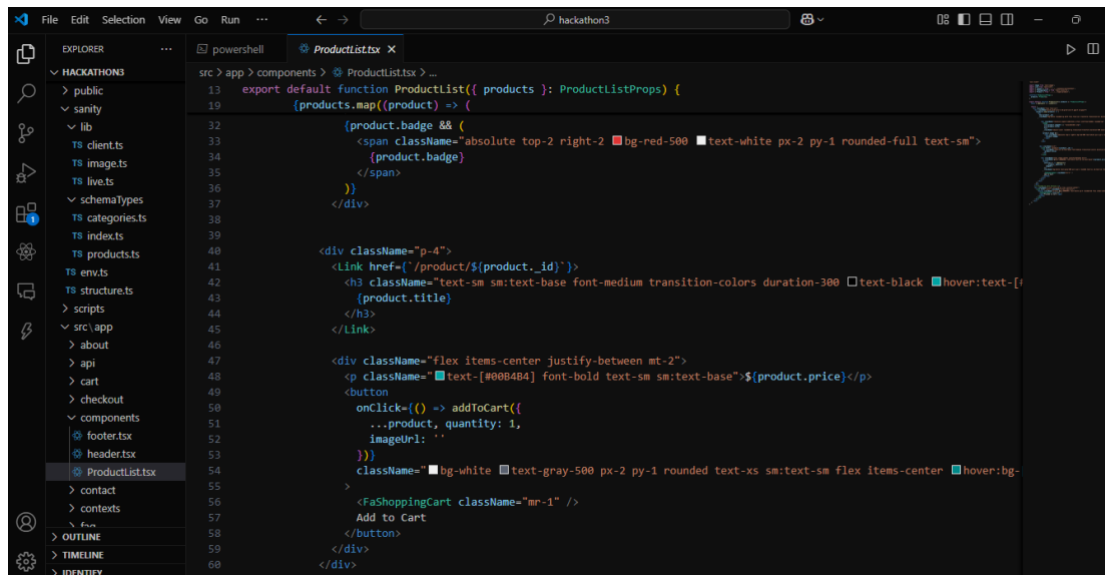
2. API Integration and Data Migration: API Data Fetching:

I fetched product data from an external API, which included details like images, titles, descriptions, prices, and tags. This data was then mapped to the corresponding fields in the Sanity CMS schema.

```

src > app > components > ProductList.tsx > ...
1 'use client'
2
3 import Image from 'next/image';
4 import Link from 'next/link';
5 import { useCart } from '../contexts/CartContext';
6 import { FaShoppingCart } from 'react-icons/fa';
7 import { Product } from '../types/product';
8
9 interface ProductListProps {
10   products: Product[];
11 }
12
13 export default function ProductList({ products }: ProductListProps) {
14   const { addToCart } = useCart();
15
16   return (
17     <div className="flex flex-col">
18       <div className="grid grid-cols-2 md:grid-cols-4 gap-4 sm:gap-6">
19         {products.map((product) => (
20           <div
21             key={product._id}
22             className="bg-white rounded-lg mb-6 flex flex-col transform transition-all duration-300 hover:scale-105 hover:
23           >
24             <div className="relative aspect-[262/312] w-full overflow-hidden rounded-lg">
25               <Image
26                 src={product.imageUrl || "/placeholder.svg"}
27                 alt={product.title}
28                 fill
29                 className="object-cover rounded-lg transition-transform duration-300 ease-in-out hover:scale-110"
30               />
31               <div>
32                 {product.badge && (

```



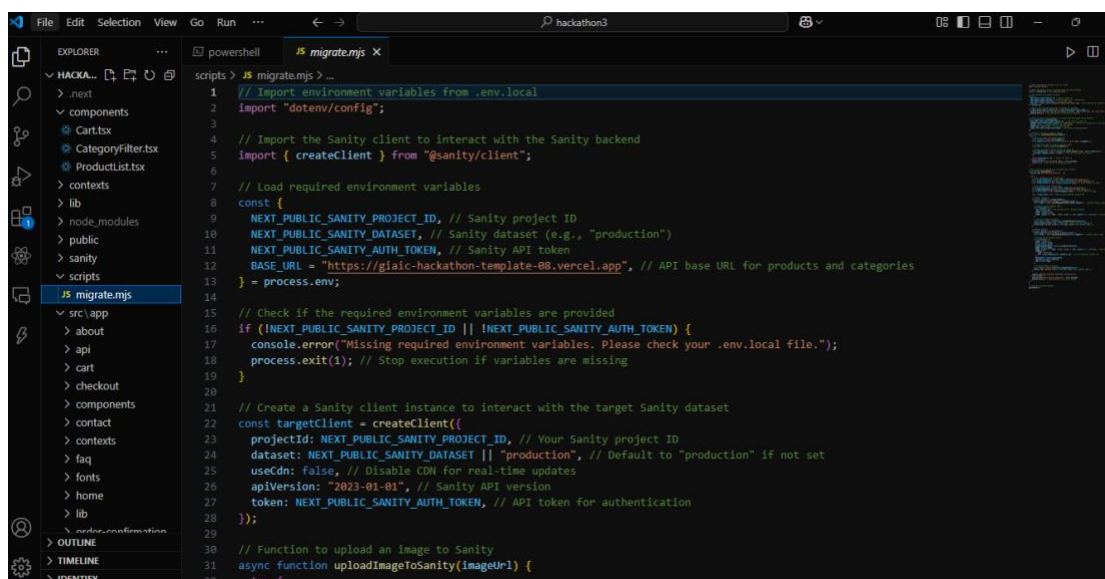
```
src > app > components > ProductList.tsx > ...
13 export default function ProductList({ products }: ProductListProps) {
19   {products.map((product) => {
32     (product.badge && (
33       <span className="absolute top-2 right-2 bg-red-500 text-white px-2 py-1 rounded-full text-sm">
34         {product.badge}
35       </span>
36     ))
37   </div>
38
39   <div className="p-4">
40     <Link href={`/product/${product.id}`}>
41       <h3 className="text-sm sm:text-base font-medium transition-colors duration-300 text-black hover:text-[
42         (product.title)
43       </h3>
44     </Link>
45
46     <div className="flex items-center justify-between mt-2">
47       <p className="text-[#008484] font-bold text-sm sm:text-base">${product.price}</p>
48     </div>
49     <button
50       onClick={() => addToCart({
51         ...product, quantity: 1,
52         imageUrl: ''
53       })}
54       className="bg-white text-gray-500 px-2 py-1 rounded text-xs sm:text-sm flex items-center hover:bg-
55     >
56     <FaShoppingCart className="mr-1" />
57     Add to Cart
58   </button>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
```

Data Population in Sanity CMS:

After fetching the API data, I populated the product fields in Sanity CMS dynamically. This allowed for the automated population of product information, ensuring consistency and accuracy across the platform.

Data Migration:

Using the Sanity CLI, I exported the data-set from Sanity CMS for backup purposes and later re-imported it for testing. This migration ensured that all data was properly structured and displayed as intended on the front end.



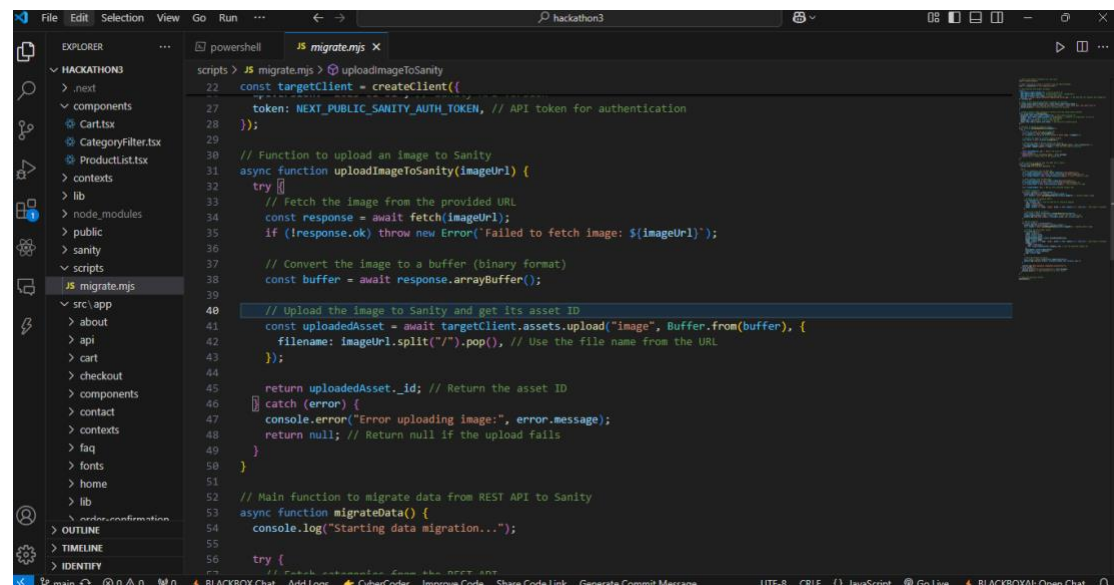
```
File Edit Selection View Go Run ... migrate.mjs
1 // Import environment variables from .env.local
2 import "dotenv/config";
3
4 // Import the Sanity client to interact with the Sanity backend
5 import { createClient } from "@sanity/client";
6
7 // Load required environment variables
8 const {
9   NEXT_PUBLIC_SANITY_PROJECT_ID, // Sanity project ID
10  NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
11  NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
12  BASE_URL = "https://gialc-hackathon-template-08.vercel.app", // API base URL for products and categories
13 } = process.env;
14
15 // Check if the required environment variables are provided
16 if (!NEXT_PUBLIC_SANITY_PROJECT_ID || !NEXT_PUBLIC_SANITY_AUTH_TOKEN) {
17   console.error("Missing required environment variables. Please check your .env.local file.");
18   process.exit(1); // Stop execution if variables are missing
19 }
20
21 // Create a Sanity client instance to interact with the target Sanity dataset
22 const targetClient = createClient({
23   projectId: NEXT_PUBLIC_SANITY_PROJECT_ID, // Your Sanity project ID
24   dataset: NEXT_PUBLIC_SANITY_DATASET || "production", // Default to "production" if not set
25   useCdn: false, // Disable CDN for real-time updates
26   apiVersion: "2023-01-01", // Sanity API version
27   token: NEXT_PUBLIC_SANITY_AUTH_TOKEN, // API token for authentication
28 });
29
30 // Function to upload an image to Sanity
31 async function uploadImageToSanity(imageUrl) {
32   try {
33     // ...
34   }
35 }
36
37 // ...
38
39 // ...
40
41 // ...
42
43 // ...
44
45 // ...
46
47 // ...
48
49 // ...
50
51 // ...
52
53 // ...
54
55 // ...
56
57 // ...
58
59 // ...
60
61 // ...
62
63 // ...
64
65 // ...
66
67 // ...
68
69 // ...
70
71 // ...
72
73 // ...
74
75 // ...
76
77 // ...
78
79 // ...
80
81 // ...
82
83 // ...
84
85 // ...
86
87 // ...
88
89 // ...
90
91 // ...
92
93 // ...
94
95 // ...
96
97 // ...
98
99 // ...
100 // ...
```

3. Steps Taken for Data Migration:

Exporting Data:

The first step was exporting the data from Sanity CMS using the Sanity CLI.

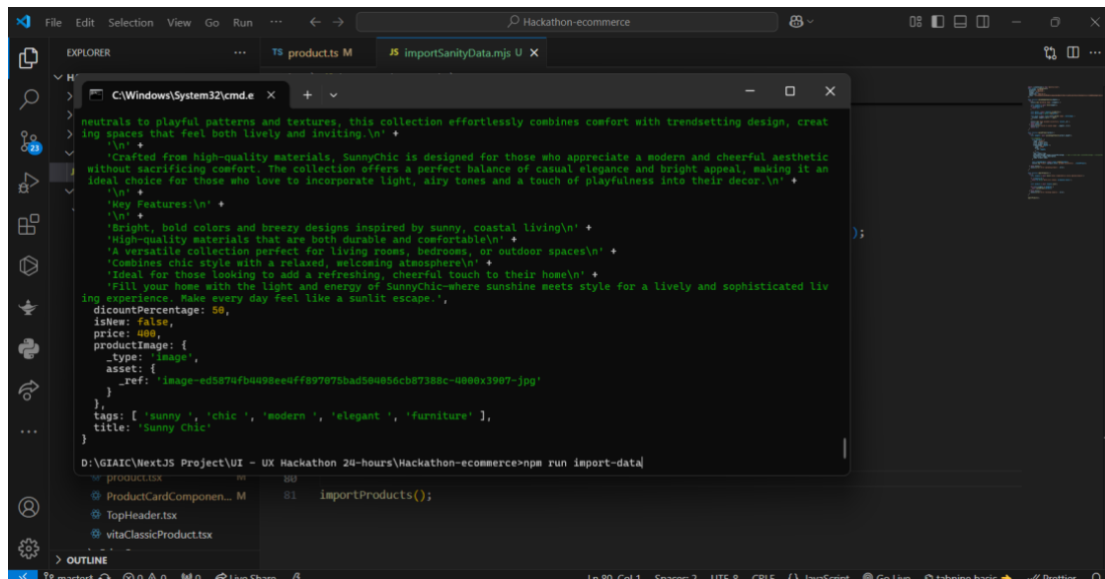
This ensured that all product data was safely backed up before any further operations.



```
scripts > .\ migrate.mjs > uploadImageToSanity
22 const targetClient = createClient({
23   token: NEXT_PUBLIC_SANITY_AUTH_TOKEN, // API token for authentication
24 });
25
26 // Function to upload an image to Sanity
27 async function uploadImageToSanity(imageUrl) {
28   try {
29     // Fetch the image from the provided URL
30     const response = await fetch(imageUrl);
31     if (!response.ok) throw new Error("Failed to fetch image: ${imageUrl}");
32
33     // Convert the image to a buffer (binary format)
34     const buffer = await response.arrayBuffer();
35
36     // Upload the image to Sanity and get its asset ID
37     const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
38       filename: imageUrl.split("/").pop(), // Use the file name from the URL
39     });
40
41     return uploadedAsset.id; // Return the asset ID
42   } catch (error) {
43     console.error("Error uploading image:", error.message);
44     return null; // Return null if the upload fails
45   }
46 }
47
48 // Main function to migrate data from REST API to Sanity
49 async function migrateData() {
50   console.log("Starting data migration...");
51
52   try {
53     // Fetch categories from the REST API
54     const categories = await fetchCategories();
55     // ... (rest of the code) ...
56   } catch (error) {
57     console.error("Error migrating data:", error.message);
58   }
59 }
```

Re-importing Data:

After verification, the data-set was re-imported into Sanity CMS. This confirmed that the data migration was successful and the system was working as expected.

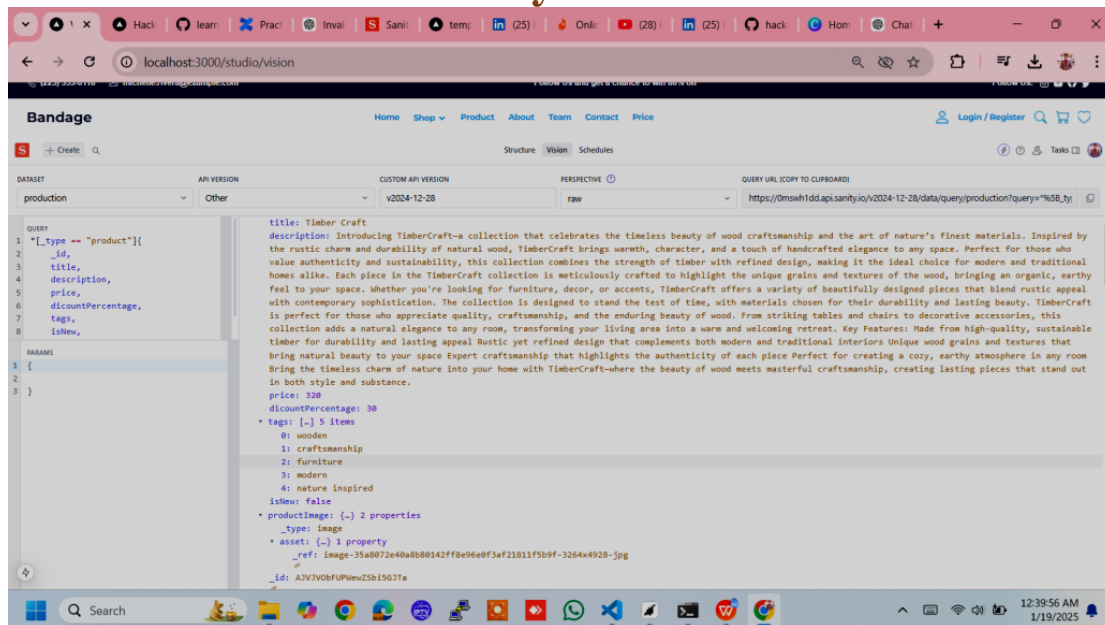


4. Tools Used:

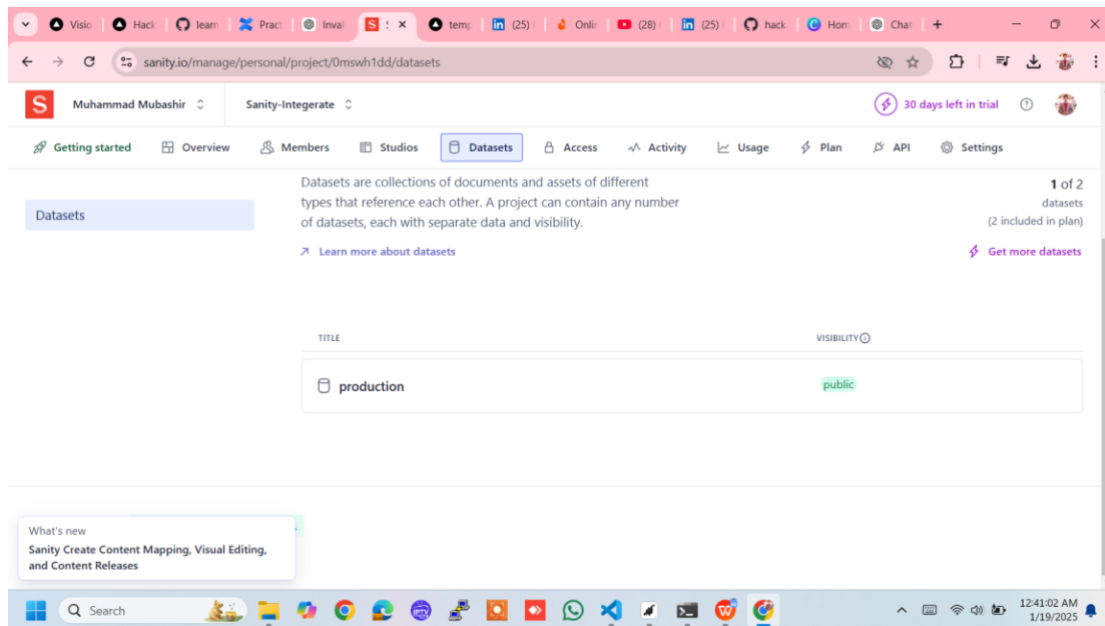
Sanity Studio:

Used for schema creation, content management, and displaying product data.

Sanity Vision



Sanity Database:



Sanity CLI:

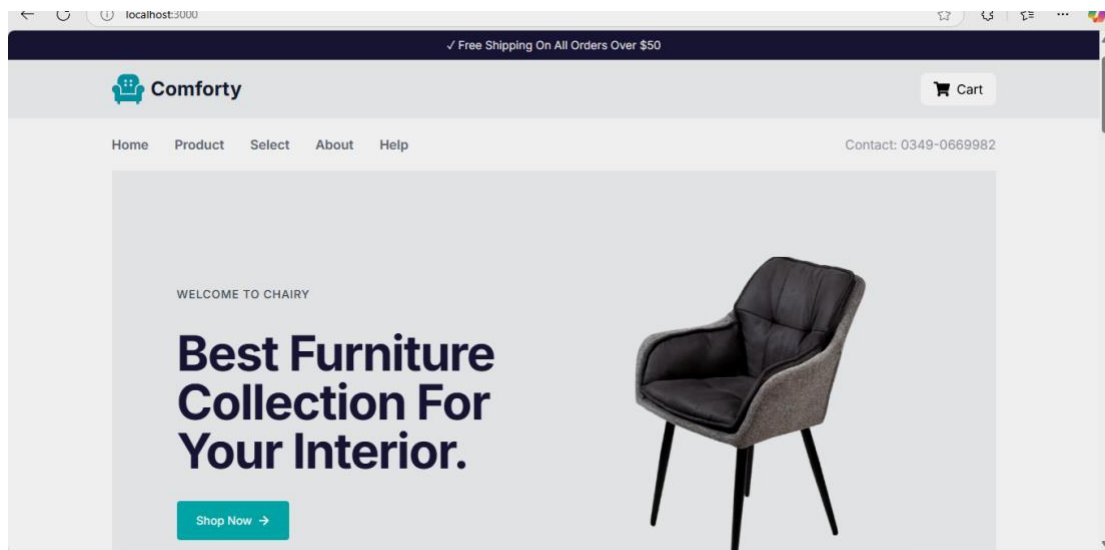
Utilized for exporting and importing the data-set, ensuring data consistency and backup.

5. Screenshots and Frontend Display:

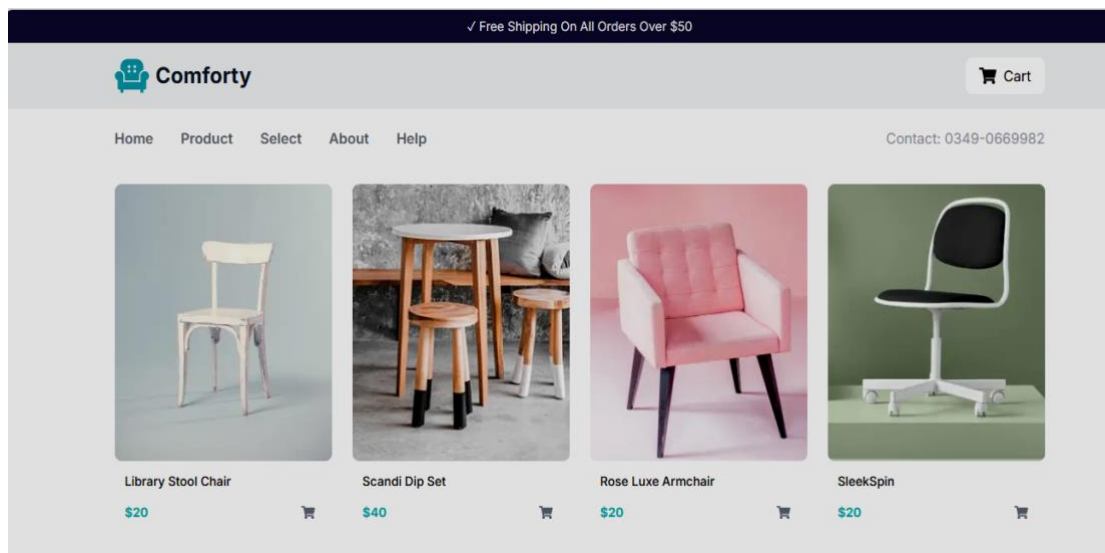
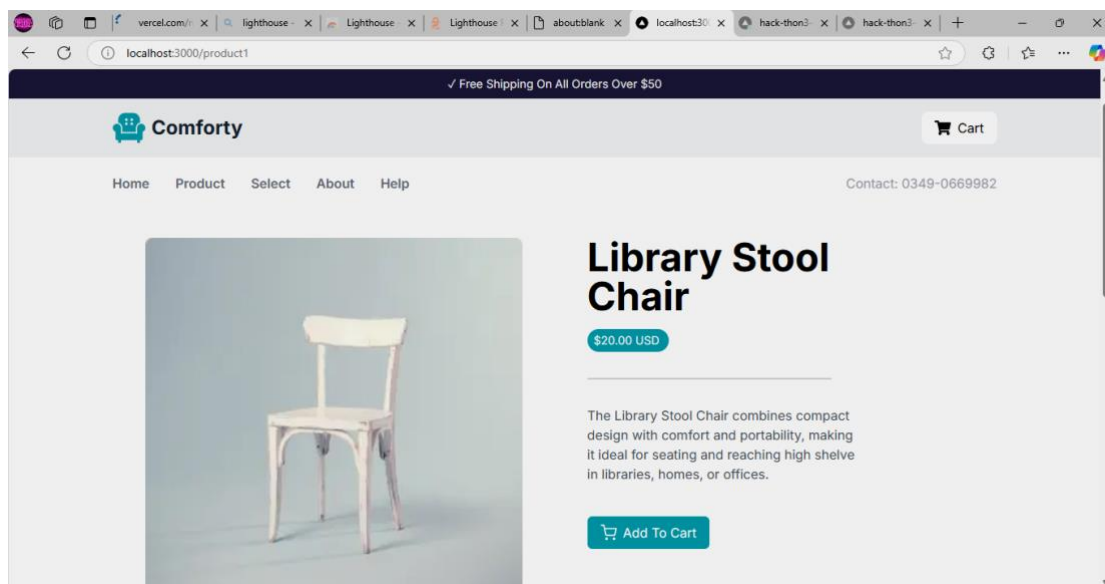
Sanity CMS Fields:

Frontend Display:

Screenshot demonstrating how the product data is displayed dynamically on the front end of the marketplace.



Product Detail Page Display:



Conclusion:

The API integration and data migration were successfully completed, enhancing the efficiency and scalability of the Bandage project. This integration streamlined the process of adding and updating product data in the marketplace, while the migration steps ensured data consistency and accuracy across the system. With this setup, the Bandage project is now more dynamic and easier to maintain.

Checklist

API Understanding



Schema Validation



Data Migration



API Integration in Next.js



Submission Preparation

