

DAY 3 – API INTEGRATION AND DATA MIGRATION

MIGRATE FILE

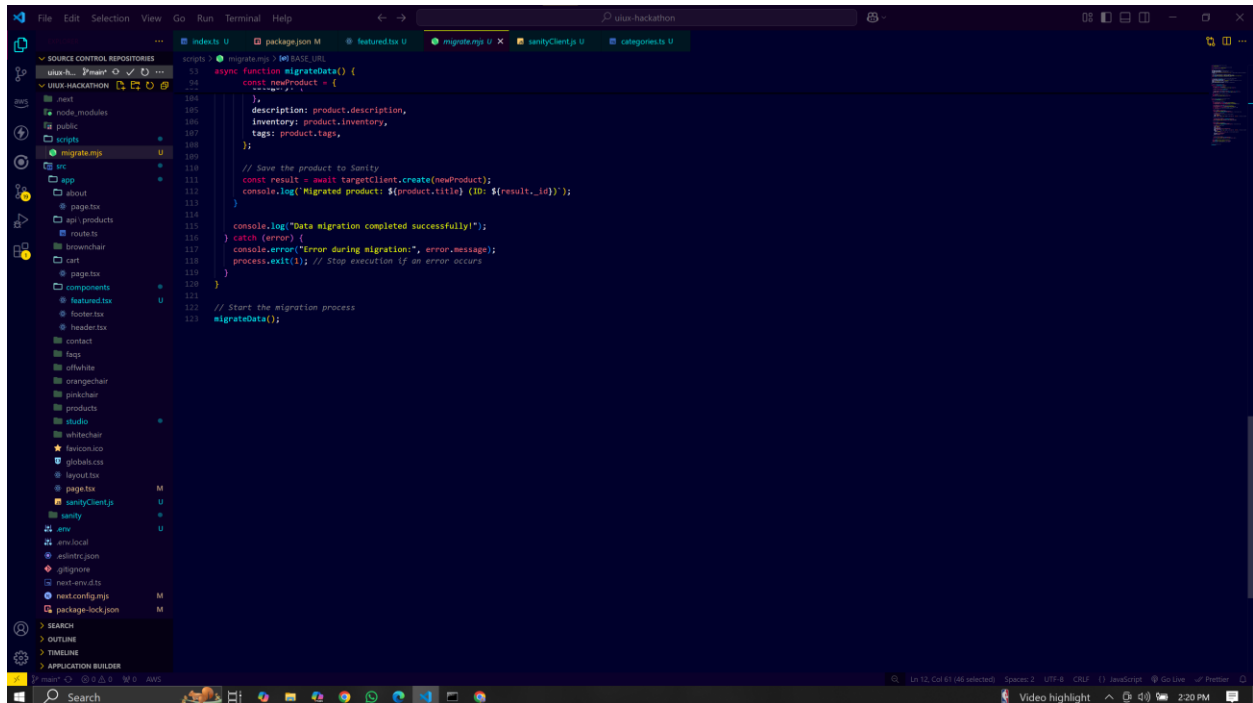
DAY 3 – API INTEGRATION AND DATA MIGRATION

The screenshot displays the VS Code editor with the `migrate.js` file open. The code implements the `migrateData` function, which is responsible for migrating data from a REST API to Sanity. The function follows these steps:

- Fetch Categories:** It uses `fetch` to retrieve categories from the REST API. If the response is successful, it parses the JSON and iterates over each category. For each category, it calls `uploadImageToSanity` to upload the category image and then uses `targetClient.createOrReplace` to create or update the category in Sanity.
- Fetch Products:** Similarly, it fetches products from the REST API. For each product, it calls `uploadImageToSanity` to upload the product image and then uses `targetClient.createOrReplace` to create or update the product in Sanity.
- Object Structure:** The `newCategory` and `newProduct` objects are structured to match the Sanity schema, including fields like `category_id`, `product_id`, `title`, `price`, `image`, and `badge`.

```
52 // Main function to migrate data from REST API to Sanity
53 async function migrateData() {
54   console.log("Starting data migration...");
55
56   try {
57     // Fetch categories from the REST API
58     const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
59     if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
60     const categoriesData = await categoriesResponse.json(); // Parse response to JSON
61
62     // Fetch products from the REST API
63     const productsResponse = await fetch(`${BASE_URL}/api/products`);
64     if (!productsResponse.ok) throw new Error("Failed to fetch products.");
65     const productsData = await productsResponse.json(); // Parse response to JSON
66
67     const categoryIdMap = {}; // Map to store migrated category IDs
68
69     // Migrate categories
70     for (const category of categoriesData) {
71       console.log("Migrating category: " + category.title);
72       const imageUrl = await uploadImageToSanity(category.imageUrl); // Upload category image
73
74       // Prepare the new category object
75       const newCategory = {
76         _id: category._id, // Use the same ID for reference mapping
77         _type: "categories",
78         title: category.title,
79         image: imageUrl ? { _type: "image", asset: { _ref: imageUrl } } : undefined, // Add image if uploaded
80       };
81
82       // Save the category to Sanity
83       const result = await targetClient.createOrReplace(newCategory);
84       categoryIdMap[category._id] = result._id; // Store the new category ID
85       console.log("Migrated category: " + category.title + " ID: " + result._id);
86     }
87
88     // Migrate products
89     for (const product of productsData) {
90       console.log("Migrating product: " + product.title);
91       const imageUrl = await uploadImageToSanity(product.imageUrl); // Upload product image
92
93       // Prepare the new product object
94       const newProduct = {
95         _type: "products",
96         title: product.title,
97         price: product.price,
98         priceWithoutDiscount: product.priceWithoutDiscount,
99         badge: product.badge,
100        image: imageUrl ? { _type: "image", asset: { _ref: imageUrl } } : undefined, // Add image if uploaded
101        category: {
102          _type: "reference",
103          _ref: categoryIdMap[product.category_id], // Use the migrated category ID
104        },
105      };
106    }
107  }
108 }
```

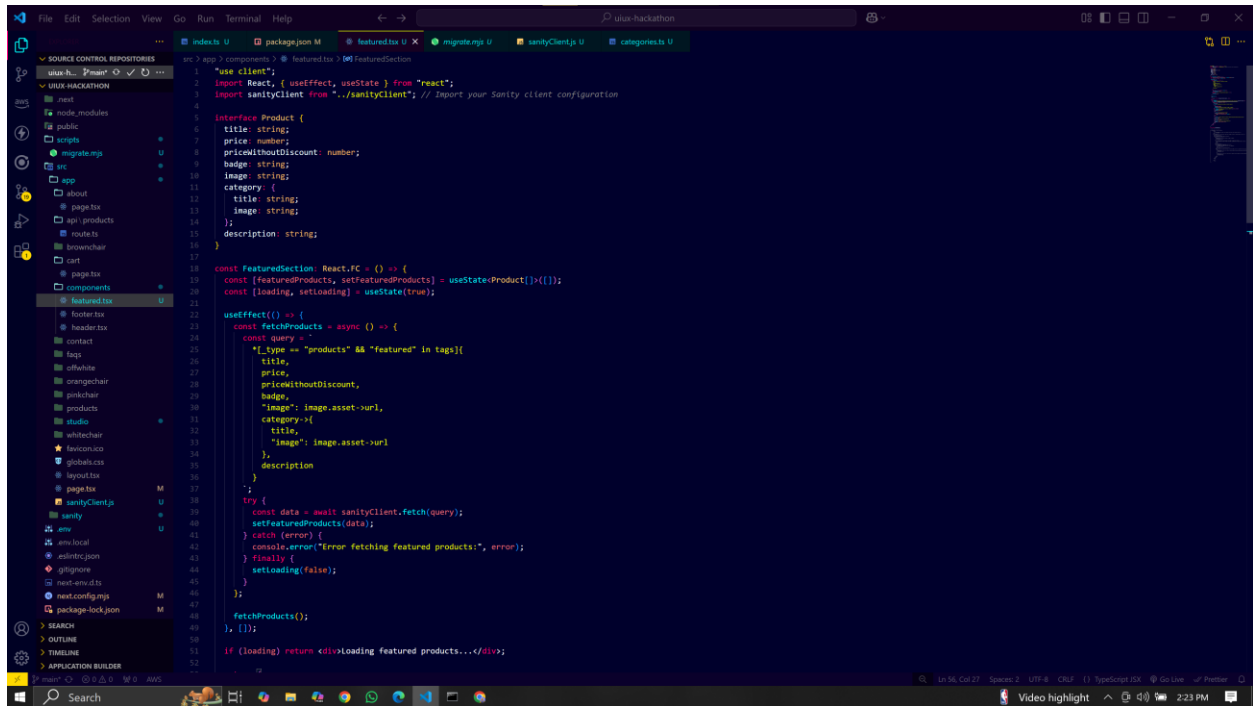
DAY 3 – API INTEGRATION AND DATA MIGRATION



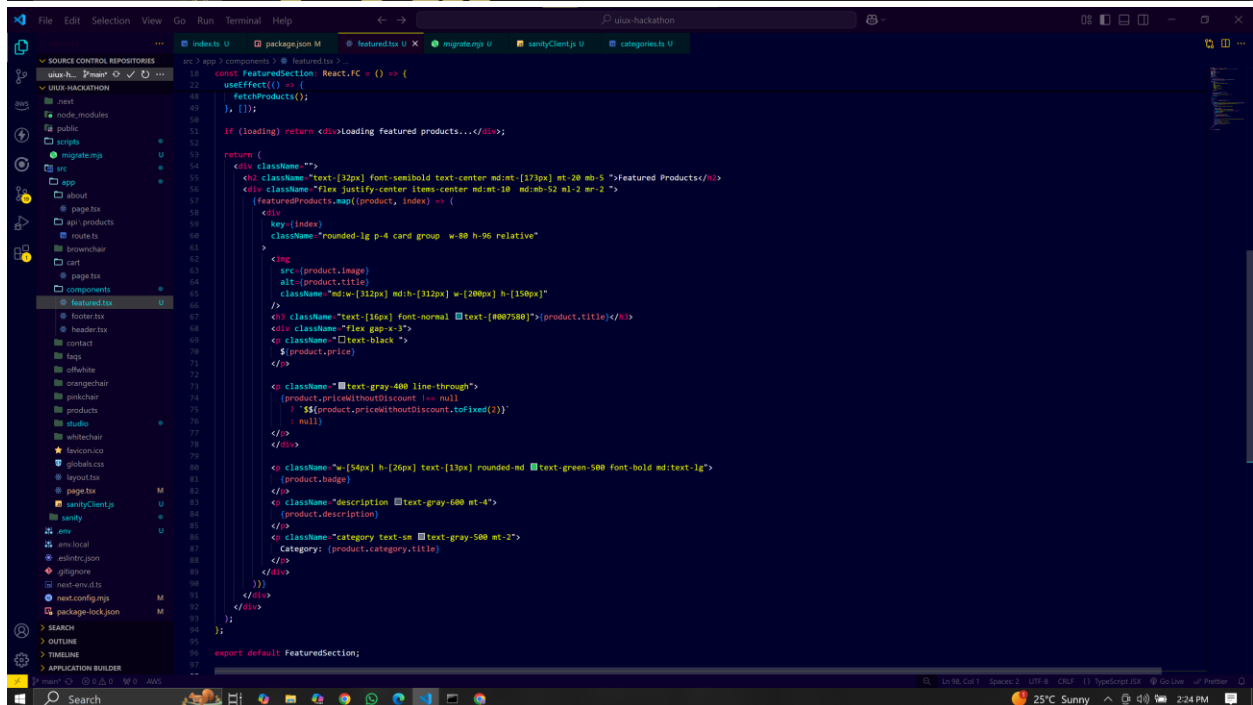
```
1 // migrate.js
2
3 // Import necessary modules
4 const { execSync } = require('child_process');
5 const fs = require('fs');
6
7 // Define the source and target URLs
8 const sourceUrl = 'http://localhost:3000/api/products';
9 const targetUrl = 'http://localhost:3001/api/products';
10
11 // Define the migrateData function
12 async function migrateData() {
13   // Get the list of products from the source URL
14   const products = await fetch(sourceUrl).then(res => res.json());
15
16   // Iterate through the products and create them in the target database
17   for (const product of products) {
18     const newProduct = {
19       title: product.title,
20       description: product.description,
21       inventory: product.inventory,
22       tags: product.tags,
23     };
24
25     // Save the product to Sanity
26     const result = await targetClient.create(newProduct);
27     console.log('Migrated product: ', result._id);
28   }
29
30   console.log('Data migration completed successfully!');
31 }
32
33 // Catch any errors
34 catch (error) {
35   console.error('Error during migration:', error.message);
36   process.exit(1); // Stop execution if an error occurs
37 }
38
39 // Start the migration process
40 migrateData();
```

DAY 3 – API INTEGRATION AND DATA MIGRATION

API CALL



```
1 use client;
2 import React, { useEffect, useState } from "react";
3 import sanityClient from "../sanityClient"; // Import your Sanity client configuration
4
5 interface Product {
6   title: string;
7   price: number;
8   priceWithoutDiscount: number;
9   badge: string;
10  image: string;
11  category: {
12    title: string;
13    image: string;
14  };
15  description: string;
16 }
17
18 const featuredSection: React.FC = () => {
19   const [featuredProducts, setFeaturedProducts] = useState<Product[]>([]);
20   const [loading, setLoading] = useState(true);
21
22   useEffect(() => {
23     const fetchProducts = async () => {
24       const query = `
25         *[_type == "products" && "featured" in tags][
26           title,
27           price,
28           priceWithoutDiscount,
29           badge,
30           "image": image.asset->url,
31           category->{
32             title,
33             "image": image.asset->url
34           },
35           description
36         ]
37       `;
38       try {
39         const data = await sanityClient.fetch(query);
40         setFeaturedProducts(data);
41       } catch (error) {
42         console.error("Error fetching featured products:", error);
43       } finally {
44         setLoading(false);
45       }
46     };
47     fetchProducts();
48   }, []);
49
50   if (loading) return <div>Loading featured products...</div>;
51 }
```

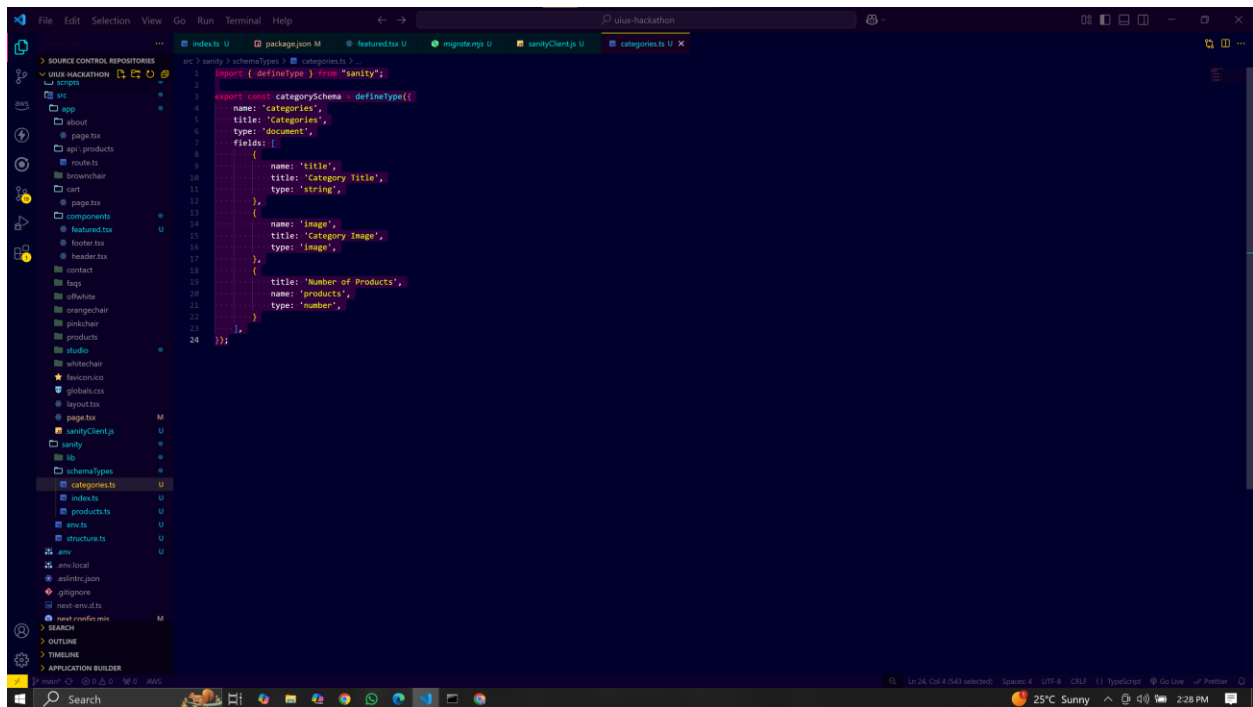


```
52 return (
53   <div className="text-[12px] font-semibold text-center md:text-[17px] mt-20 mb-5">Featured Products</div>
54   <div className="flex justify-center items-center md:mt-10 md:mb-12 md:mr-2">
55     {featuredProducts.map((product, index) => (
56       <div
57         key={index}
58         className="rounded-lg p-4 card group w-80 h-96 relative"
59       >
60         <img
61           src={product.image}
62           alt={product.title}
63           className="md:w-[312px] md:h-[312px] w-[200px] h-[150px]"
64         />
65         <div className="text-[16px] font-normal text-[#007580]>{product.title}</div>
66         <div className="flex gap-x-3">
67           <div className="text-black">
68             ${product.price}
69           </div>
70           <div className="text-gray-400 line-through">
71             {product.priceWithoutDiscount !== null
72               ? `${product.priceWithoutDiscount.toFixed(2)}`
73               : null}
74           </div>
75         </div>
76         <div className="w-[54px] h-[26px] text-[13px] rounded-md text-green-500 font-bold md:text-lg">
77           {product.badge}
78         </div>
79         <div className="description text-gray-600 mt-4">
80           {product.description}
81         </div>
82         <div className="category text-sm text-gray-500 mt-2">
83           Category: {product.category.title}
84         </div>
85       </div>
86     ))}
87   </div>
88 );
89
90 export default featuredSection;
```

SCHEMA

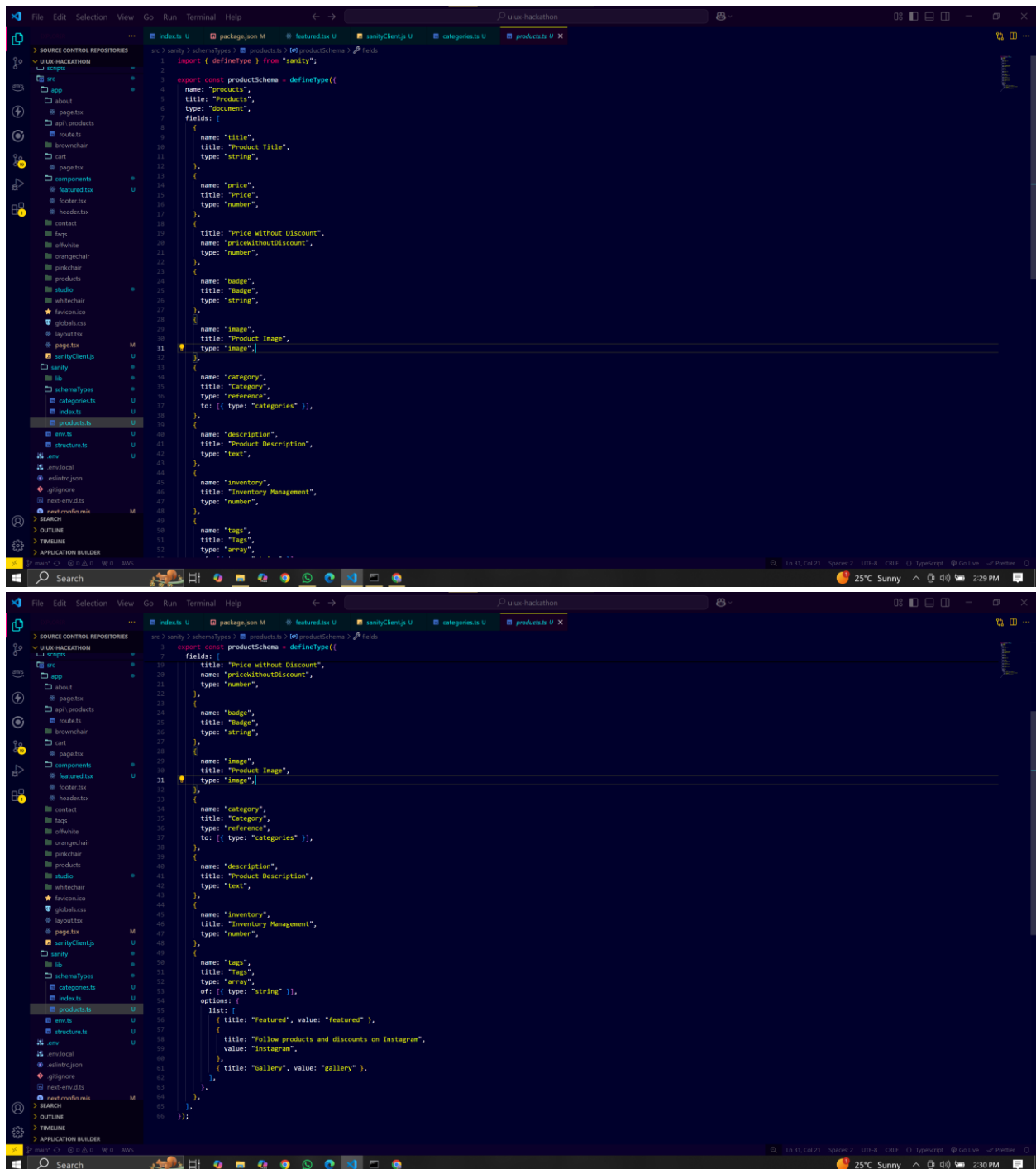
CATEGORIES.TS

DAY 3 – API INTEGRATION AND DATA MIGRATION



PRODUCTS.TS

DAY 3 – API INTEGRATION AND DATA MIGRATION



The image displays two screenshots of a VS Code editor interface, showing the development of a Sanity schema for a product. The left sidebar shows the project structure, including folders like 'src', 'api', 'components', 'routes', and 'sanity'. The main editor area shows the 'products.js' file in the 'sanity' directory.

Top Screenshot: Shows the initial schema definition for the 'products' type. The schema includes fields for 'title', 'price', 'priceWithoutDiscount', 'badge', 'image', 'category', 'description', 'inventory', and 'tags'.

```
import { defineType } from "sanity";

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Product Title",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      type: "number",
    },
    {
      name: "badge",
      title: "Badge",
      type: "string",
    },
    {
      name: "image",
      title: "Product Image",
      type: "image",
    },
    {
      name: "category",
      title: "Category",
      type: "reference",
      to: [{ type: "categories" }],
    },
    {
      name: "description",
      title: "Product Description",
      type: "text",
    },
    {
      name: "inventory",
      title: "Inventory Management",
      type: "number",
    },
    {
      name: "tags",
      title: "Tags",
      type: "array",
    },
  ],
});
```

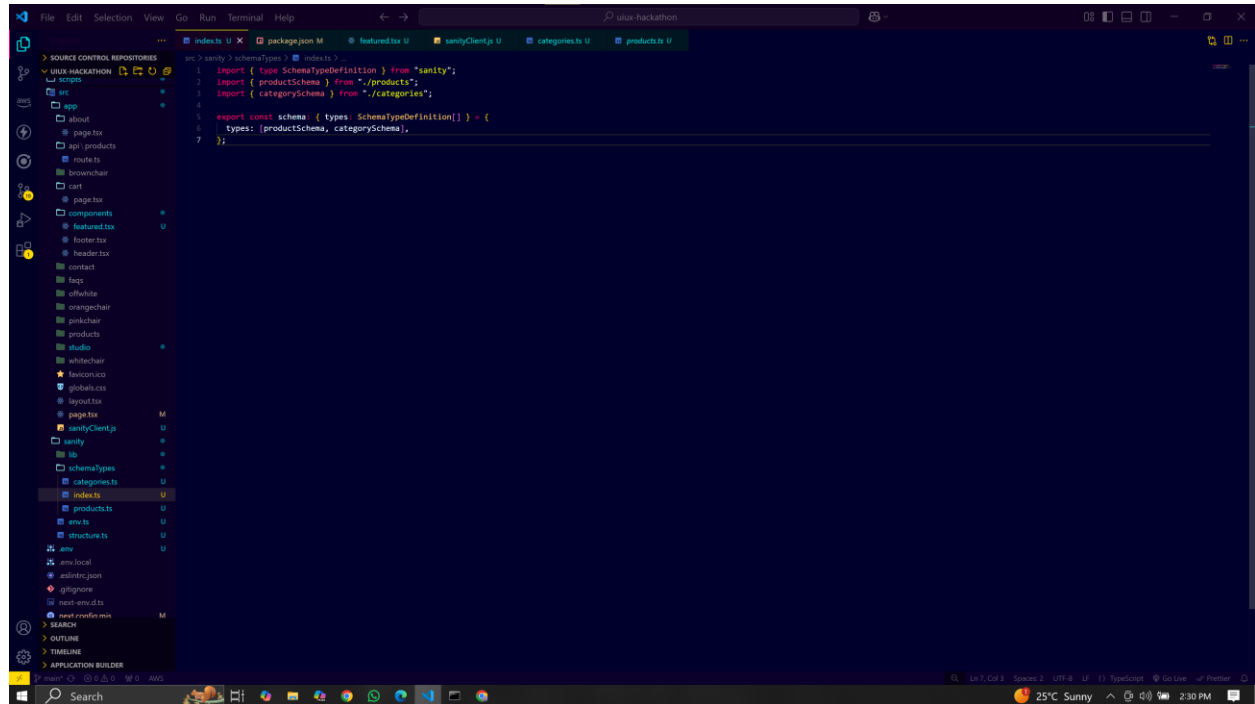
Bottom Screenshot: Shows the same schema definition, but with additional options for the 'tags' field, including a list of predefined tags: 'features', 'discounts', 'instagram', and 'gallery'.

```
import { defineType } from "sanity";

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Product Title",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      type: "number",
    },
    {
      name: "badge",
      title: "Badge",
      type: "string",
    },
    {
      name: "image",
      title: "Product Image",
      type: "image",
    },
    {
      name: "category",
      title: "Category",
      type: "reference",
      to: [{ type: "categories" }],
    },
    {
      name: "description",
      title: "Product Description",
      type: "text",
    },
    {
      name: "inventory",
      title: "Inventory Management",
      type: "number",
    },
    {
      name: "tags",
      title: "Tags",
      type: "array",
      of: [{ type: "string" }],
      options: {
        list: [
          { title: "features", value: "features" },
          { title: "Follow products and discounts on Instagram", value: "instagram" },
          { title: "Gallery", value: "gallery" },
        ],
      },
    },
  ],
});
```

DAY 3 – API INTEGRATION AND DATA MIGRATION

INDEX.TS



```
src > sanity > schemaTypes > index.ts ...
1 import { type SchemaTypeDefinition } from "sanity";
2 import { productSchema } from "../products";
3 import { categorySchema } from "../categories";
4
5 export const schema: { types: SchemaTypeDefinition[] } = {
6   types: [productSchema, categorySchema],
7 }
```

FETCHED FROM SANITY

DAY 3 – API INTEGRATION AND DATA MIGRATION

Featured Products



Citrus Edge
\$20

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Category: Desk Chair



Rose Luxe Armchair
\$20 ~~\$36.00~~
Sales

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Category: Wing Chair



Scandi Dip Set
\$40

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Category: Wooden Chair

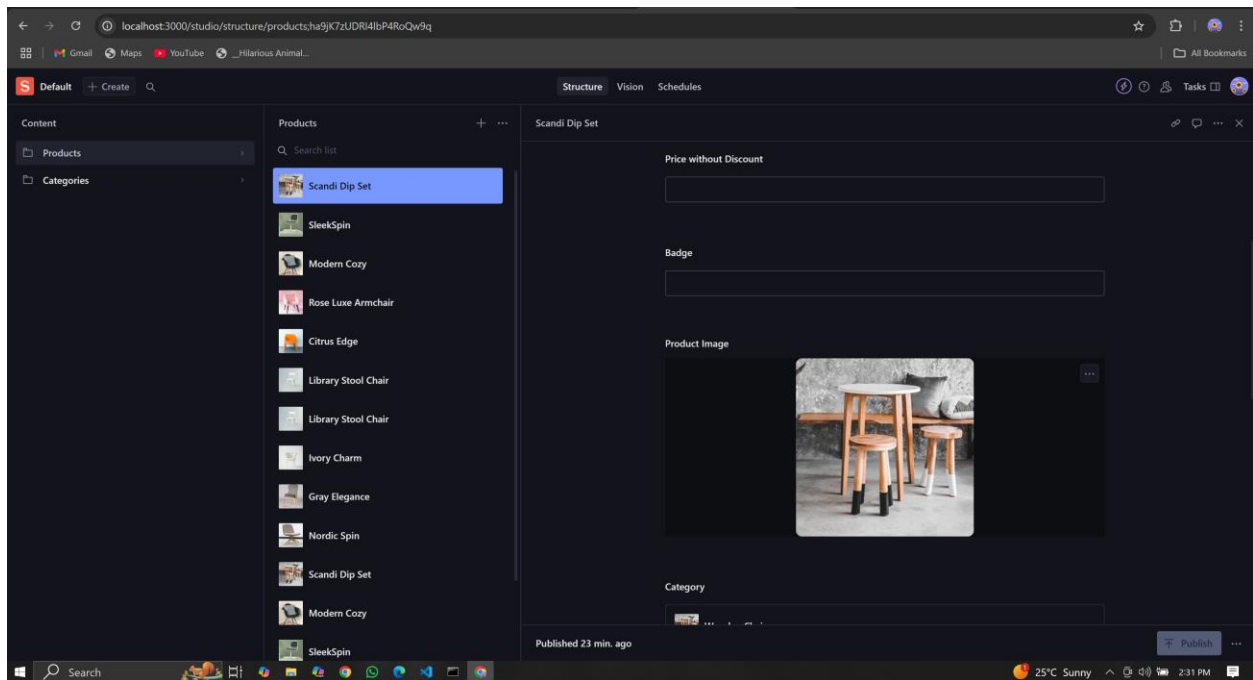


Ivory Charm
\$20

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Category: Wing Chair

SANITY MIGRATION



DAY 3 – API INTEGRATION AND DATA MIGRATION

The screenshot displays the Studio Vision API client interface in a web browser. The browser address bar shows the URL `localhost:3000/studio/vision`. The interface includes a top navigation bar with tabs for 'Structure', 'Vision', and 'Schedules'. Below this, there are dropdown menus for 'DATASET' (set to 'production'), 'API VERSION' (set to 'Other'), 'CUSTOM API VERSION' (set to 'v2025-01-20'), and 'PERSPECTIVE' (set to 'raw'). A 'QUERY URL (COPY TO CLIPBOARD)' field contains the URL `https://3r58cz83.api.sanity.io/v2025-01-20/data/query/production?query=%5B%5D`.

The main area is divided into two panels. The left panel, labeled 'QUERY', contains a JSON query:

```
1 [{"_type" == "products" && "featured" in tag
2   title,
3   price,
4   tags
5 }
6 ]
```

The right panel, labeled 'RESULT', displays the JSON response from the API:

```
[{"_type": "products", "title": "Citrus Edge", "price": 20, "tags": [{"_type": "featured"}, {"_type": "instagram"}, {"_type": "gallery"}]}, {"_type": "products", "title": "Rose Luxe Armchair", "price": 20, "tags": [{"_type": "featured"}, {"_type": "instagram"}]}, {"_type": "products", "title": "Scandi Dip Set", "price": 40, "tags": [{"_type": "featured"}, {"_type": "gallery"}]}, {"_type": "products", "title": "Ivory Charm", "price": 20, "tags": [{"_type": "featured"}, {"_type": "gallery"}]}]
```

At the bottom of the interface, there are buttons for 'Fetch' and 'Listen', and a status bar showing 'Execution: 2ms End-to-end: 171ms'. The bottom of the browser window shows a Windows taskbar with a search bar and system tray icons.