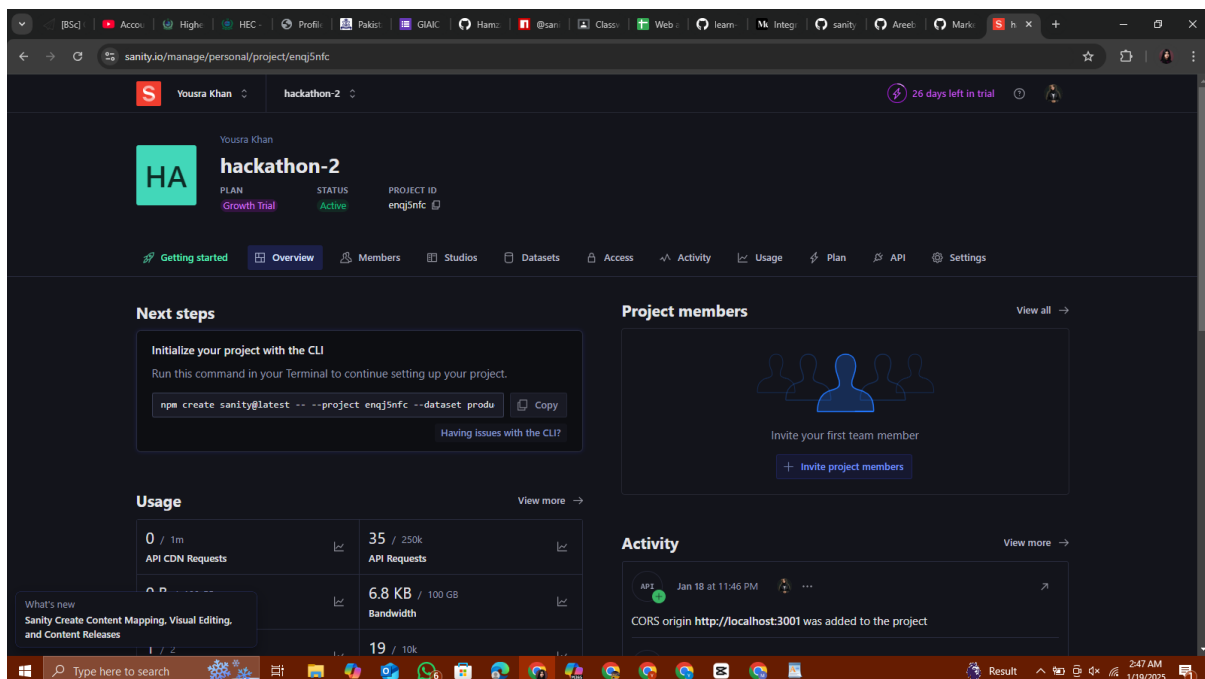


## DAY 3 - API INTEGRATION AND DATA MIGRATION- COMFORTY

### API integration process.

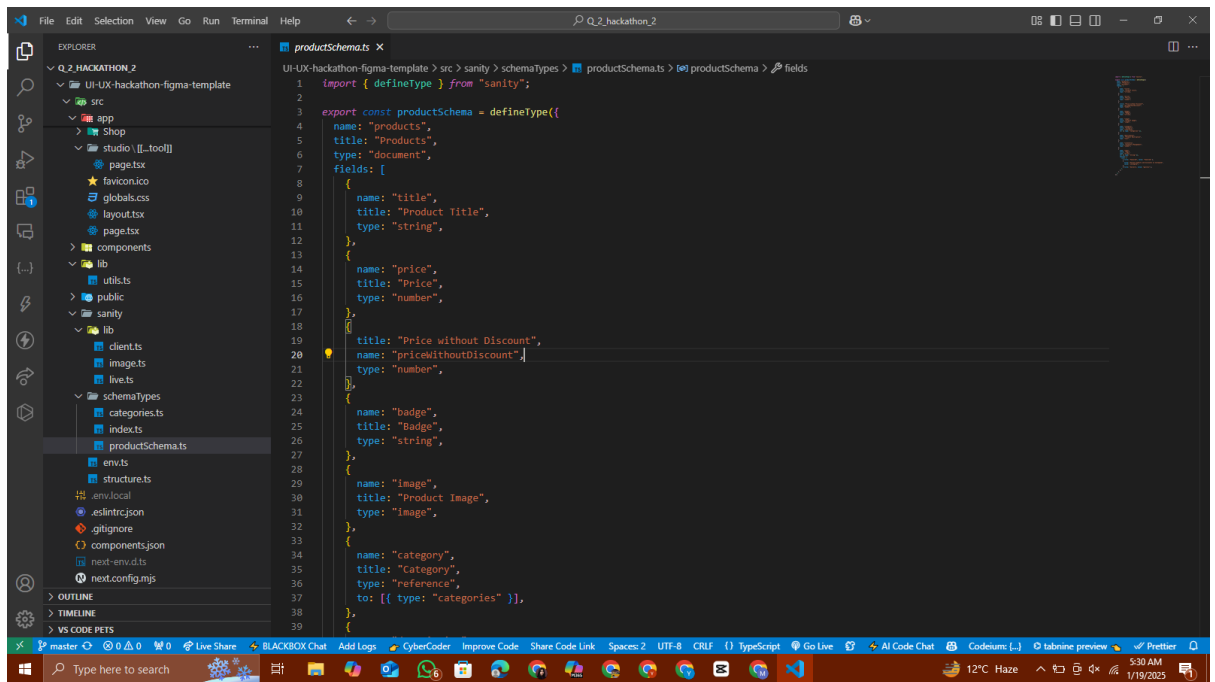
Seamlessly connected the backend services with the frontend by integrating REST APIs, enabling real-time data flow between the e-commerce platform and the Sanity CMS. This integration ensures accurate synchronization of product and category data for a dynamic user experience.

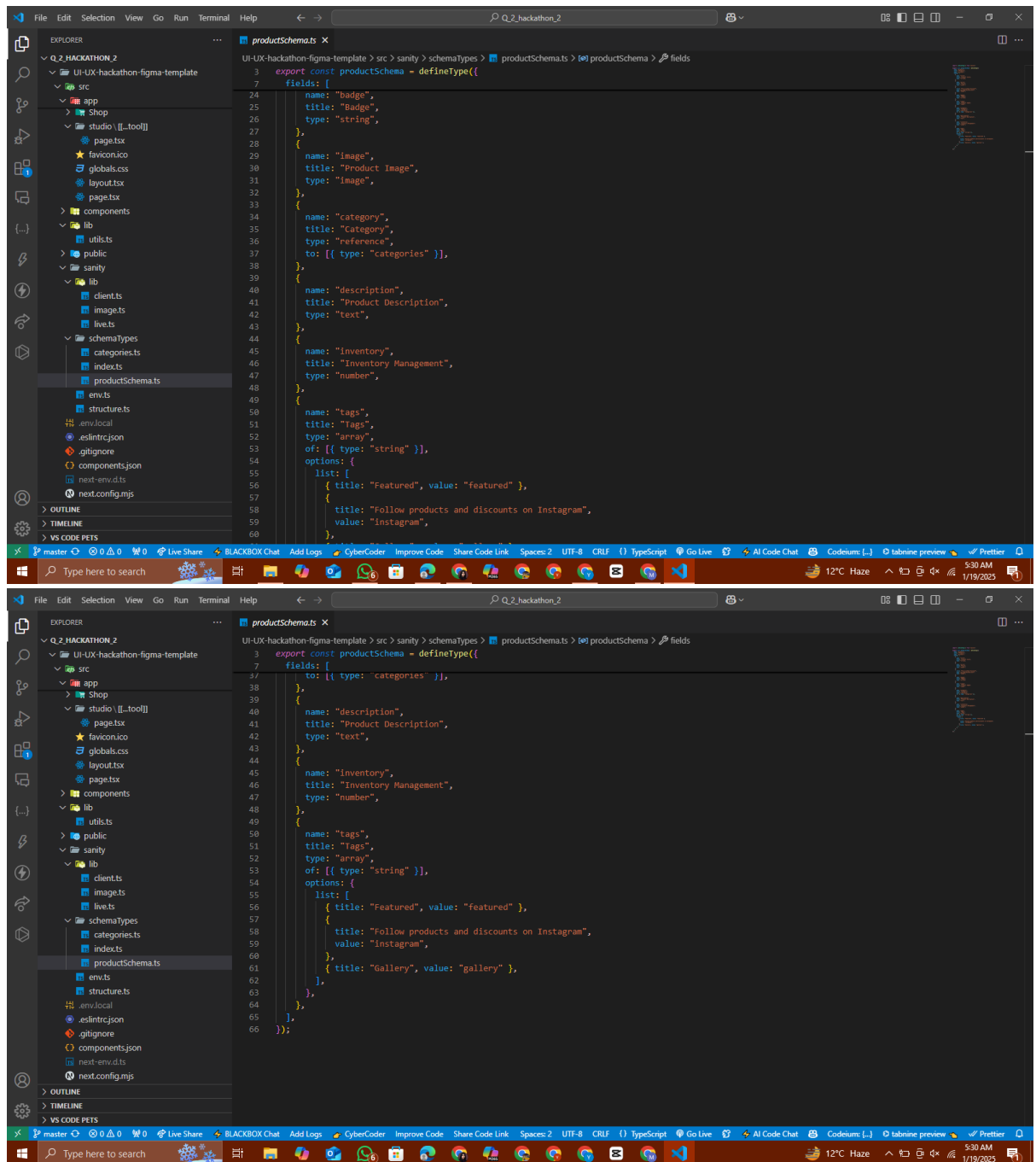
- **Sanity CMS dashboard**



- **Schema Adjustment:**

I have developed and refined 2 schemas productSchema.ts and categories.ts





- **Migration steps and tools used.**

First I created scripts folder, inside scripts folder make migrate.mjs file where I call the API , then run this command on the terminal (npm run migrate), where This will insert the data from the rest api to the sanity studio.

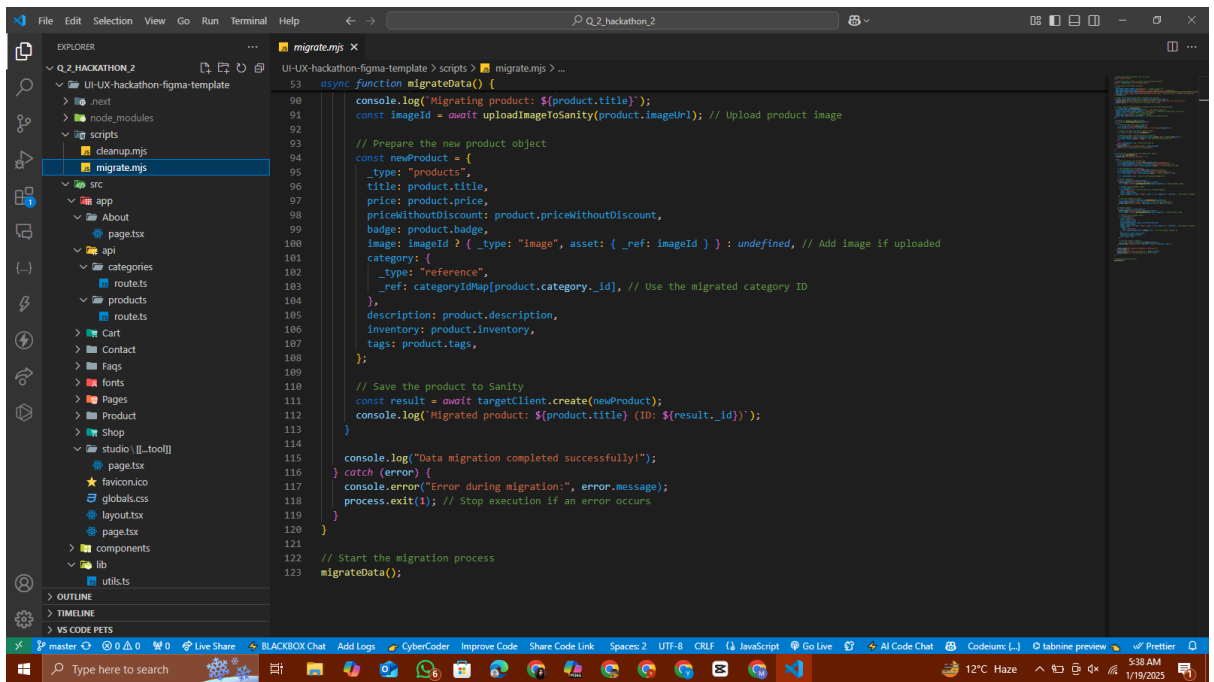
```
1 // Import environment variables from .env.local
2 import { config } from 'dotenv/config';
3
4 // Import the Sanity client to interact with the Sanity backend
5 import { createClient } from '@sanity/client';
6
7 // Load required environment variables
8 const {
9   NEXT_PUBLIC_SANITY_PROJECT_ID,
10   NEXT_PUBLIC_SANITY_DATASET,
11   NEXT_PUBLIC_SANITY_AUTH_TOKEN,
12   BASE_URL = "https://giaic-hackathon-template-88.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     // Fetch the image from the provided URL
34     const response = await fetch(imageUrl);
35     if (!response.ok) throw new Error("Failed to fetch image: ${imageUrl}");
36
37     // Convert the image to a buffer (binary format)
38     const buffer = await response.arrayBuffer();
```

After running the command,

```
1 {
2   "name": "figma-hackathon",
3   "version": "0.1.0",
4   "private": true,
5   "scripts": {
6     "dev": "next dev",
7     "build": "next build",
8     "start": "next start",
9     "lint": "next lint",
10    "migrate": "node scripts/migrate.mjs"
11  },
12  "dependencies": {
13    "@next/font": "14.2.15",
14    "@radix-ui/react-dialog": "1.1.2",
15    "@radix-ui/react-dropdown-menu": "2.1.3",
16    "@radix-ui/react-slot": "1.1.0",
17    "@sanity/client": "0.25.0",
18    "@sanity/image-url": "1.1.0",
19    "@sanity/vision": "3.70.0",
20    "axios": "1.7.0",
21    "class-variance-authority": "0.7.1",
22    "clsx": "2.1.1",
23    "dotenv": "16.4.7",
24    "lucide-react": "0.468.0",
25    "next": "14.2.16",
26    "next-sanity": "0.8.35",
27    "react": "18",
28    "react-dom": "18",
29    "react-icons": "5.4.0",
30    "sanity": "3.70.0",
31    "styled-components": "6.1.14",
32    "tailwind-merge": "2.5.5",
33    "tailwindcss-animate": "1.0.7"
34  },
35  "devDependencies": {
36    "@types/node": "20",
37    "@types/react": "18",
38    "@types/react-dom": "18",
39  }
40 }
```

```
File Edit Selection View Go Run Terminal Help
Q2 HACKATHON 2
UI-UX-hackathon-figma-template > scripts > migrate.mjs > ...
22 const targetClient = createClient({
23   projectId: '...',
24   dataset: 'production',
25   useCdn: true,
26 });
27
28 // Function to upload an image to Sanity
29 Tabnine | Edit | Test | Explain | Document | Codeium Refactor | Explain | X
30 async function uploadImageToSanity(imageUrl) {
31   try {
32     // Fetch the image from the provided URL
33     const response = await fetch(imageUrl);
34     if (!response.ok) throw new Error("Failed to fetch image: ${imageUrl}");
35
36     // Convert the image to a buffer (binary format)
37     const buffer = await response.arrayBuffer();
38
39     // Upload the image to Sanity and get its asset ID
40     const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
41       filename: imageUrl.split("/").pop(), // Use the file name from the URL
42     });
43
44     return uploadedAsset.id; // Return the asset ID
45   } catch (error) {
46     console.error("Error uploading image:", error.message);
47     return null; // Return null if the upload fails
48   }
49 }
50
51 // Main function to migrate data from REST API to Sanity
52 Tabnine | Edit | Test | Explain | Document | Codeium Refactor | Explain | X
53 async function migrateData() {
54   console.log("Starting data migration...");
55
56   try {
57     // Fetch categories from the REST API
58     const categoriesResponse = await fetch(`${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     // Migrate categories
68     for (const category of categoriesData) {
69       const imageUrl = category.imageUrl;
70       const imageId = await uploadImageToSanity(imageUrl); // Upload category image
71
72       // Prepare the new category object
73       const newCategory = {
74         _id: category._id, // Use the same ID for reference mapping
75         _type: "categories",
76         title: category.title,
77         image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
78       };
79
80       // Save the category to Sanity
81       const result = await targetClient.createOrReplace(newCategory);
82       const categoryIdMap[category._id] = result._id; // Store the new category ID
83       console.log("Migrated category: ${category.title} (ID: ${result._id})");
84     }
85
86     // Migrate products
87     for (const product of productsData) {
88       const imageUrl = product.imageUrl;
89       const imageId = await uploadImageToSanity(imageUrl); // Upload product image
90
91       // Prepare the new product object
92       const newProduct = {
93         _type: "products",
94         title: product.title,
95         price: product.price,
96         priceWithoutDiscount: product.priceWithoutDiscount,
97         badge: product.badge,
98         image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
99         category: {
100           _type: "reference",
101           _ref: categoryIdMap[product.category._id], // Use the migrated category ID
102         },
103         description: product.description,
104         inventory: product.inventory,
105         tags: product.tags,
106       };
107
108       // Save the product to Sanity
109       const result = await targetClient.createOrReplace(newProduct);
110       const productIdMap[product._id] = result._id; // Store the new product ID
111       console.log("Migrated product: ${product.title} (ID: ${result._id})");
112     }
113
114     console.log("Data migration completed successfully.");
115   } catch (error) {
116     console.error("Error during data migration:", error.message);
117   }
118 }
119
120 // Execute the migrateData function
121 migrateData().catch(console.error);
```

```
File Edit Selection View Go Run Terminal Help
Q2 HACKATHON 2
UI-UX-hackathon-figma-template > scripts > migrate.mjs > ...
53 async function migrateData() {
54   // Fetch categories from the REST API
55   const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
56   if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
57   const categoriesData = await categoriesResponse.json(); // Parse response to JSON
58
59   // Fetch products from the REST API
60   const productsResponse = await fetch(`${BASE_URL}/api/products`);
61   if (!productsResponse.ok) throw new Error("Failed to fetch products.");
62   const productsData = await productsResponse.json(); // Parse response to JSON
63
64   // Migrate categories
65   for (const category of categoriesData) {
66     const imageUrl = category.imageUrl;
67     const imageId = await uploadImageToSanity(imageUrl); // Upload category image
68
69     // Prepare the new category object
70     const newCategory = {
71       _id: category._id, // Use the same ID for reference mapping
72       _type: "categories",
73       title: category.title,
74       image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
75     };
76
77     // Save the category to Sanity
78     const result = await targetClient.createOrReplace(newCategory);
79     const categoryIdMap[category._id] = result._id; // Store the new category ID
80     console.log("Migrated category: ${category.title} (ID: ${result._id})");
81   }
82
83   // Migrate products
84   for (const product of productsData) {
85     const imageUrl = product.imageUrl;
86     const imageId = await uploadImageToSanity(imageUrl); // Upload product image
87
88     // Prepare the new product object
89     const newProduct = {
90       _type: "products",
91       title: product.title,
92       price: product.price,
93       priceWithoutDiscount: product.priceWithoutDiscount,
94       badge: product.badge,
95       image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
96       category: {
97         _type: "reference",
98         _ref: categoryIdMap[product.category._id], // Use the migrated category ID
99       },
100       description: product.description,
101       inventory: product.inventory,
102       tags: product.tags,
103     };
104
105     // Save the product to Sanity
106     const result = await targetClient.createOrReplace(newProduct);
107     const productIdMap[product._id] = result._id; // Store the new product ID
108     console.log("Migrated product: ${product.title} (ID: ${result._id})");
109   }
110
111   console.log("Data migration completed successfully.");
112 }
113
114 // Execute the migrateData function
115 migrateData().catch(console.error);
```



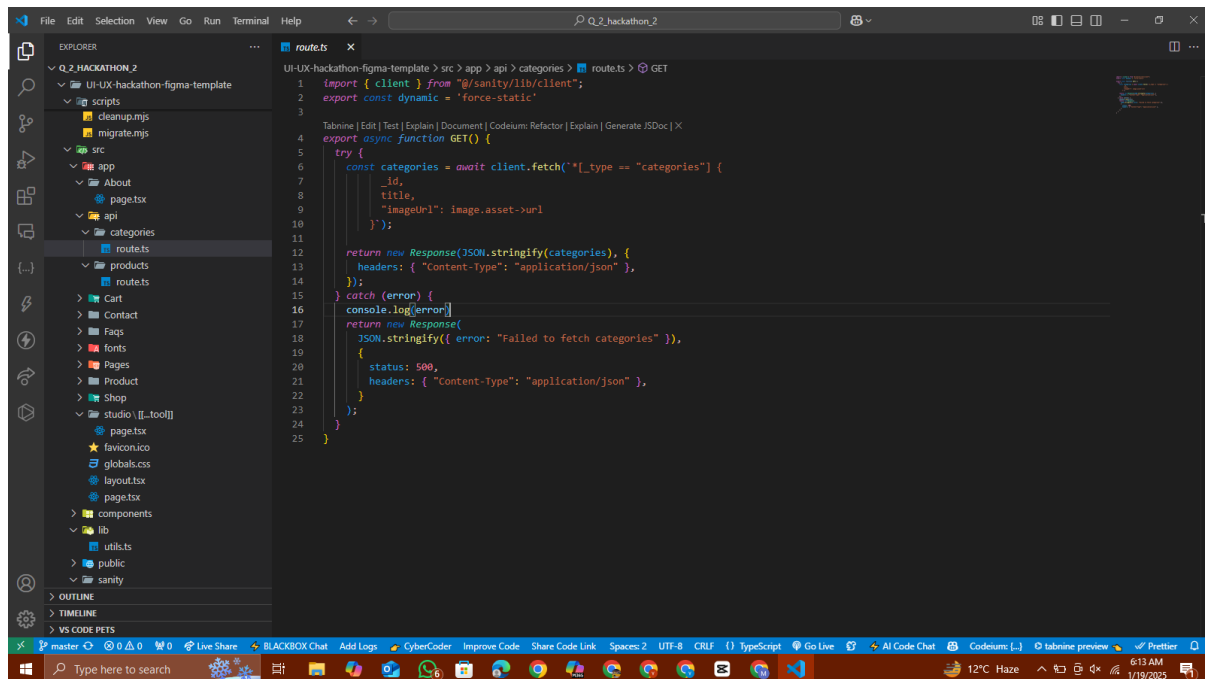
```
53 async function migrateData() {
90   console.log('Migrating product: ${product.title}');
91   const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image
92
93   // Prepare the new product object
94   const newProduct = {
95     _type: 'products',
96     title: product.title,
97     price: product.price,
98     priceWithoutDiscount: product.priceWithoutDiscount,
99     badge: product.badge,
100    image: imageId ? { _type: 'image', asset: { _ref: imageId } } : undefined, // Add image if uploaded
101    category: {
102      _type: 'reference',
103      _ref: categoryIdMap[product.category._id], // Use the migrated category ID
104    },
105    description: product.description,
106    inventory: product.inventory,
107    tags: product.tags,
108  };
109
110  // Save the product to Sanity
111  const result = await targetClient.create(newProduct);
112  console.log('Migrated product: ${product.title} (ID: ${result._id})');
113
114  console.log('Data migration completed successfully!');
115 } catch (error) {
116   console.error('Error during migration:', error.message);
117   process.exit(1); // Stop execution if an error occurs
118 }
119
120 // Start the migration process
121 migrateData();
```

- **Api calls:**

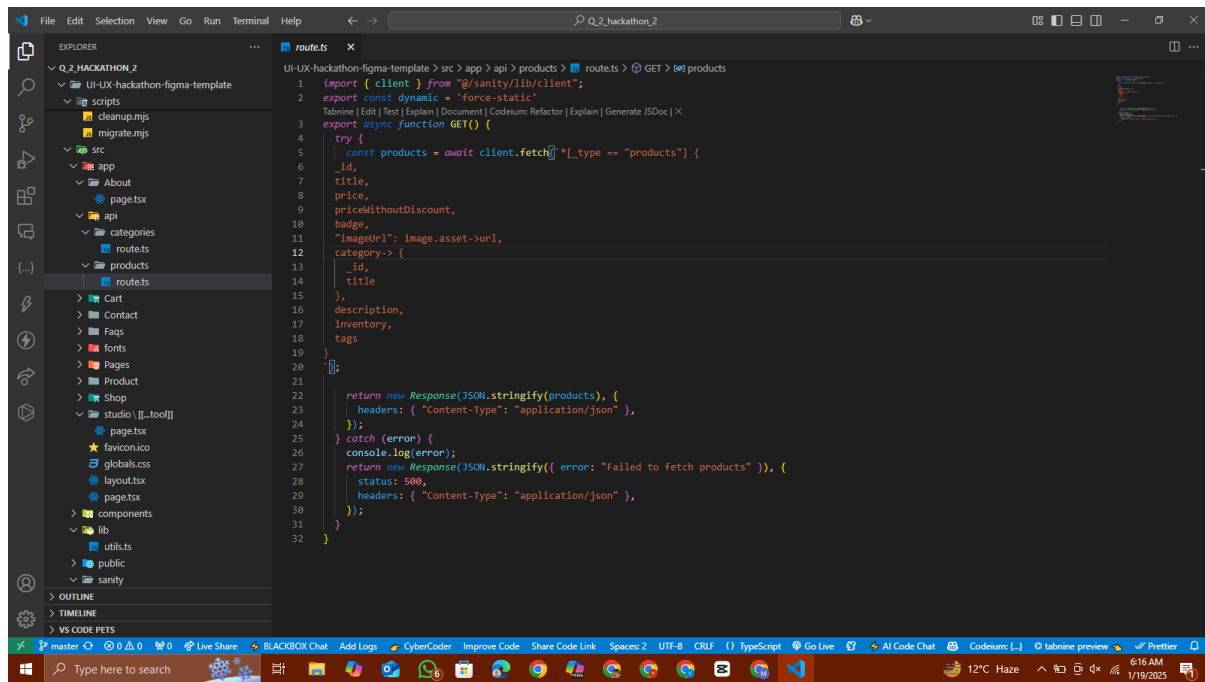
First i make a folder api inside src/app , Then i make 2 routes for fetching data through qroq query,

→ categories/routes.ts

→ Product/routes.ts

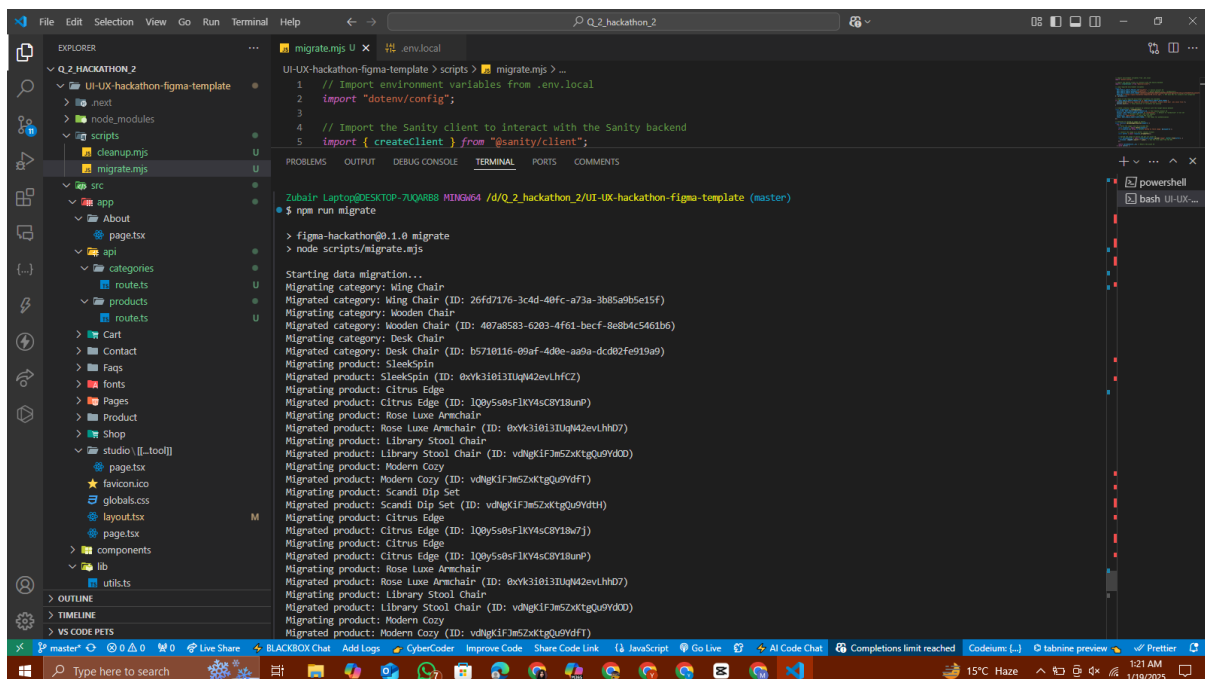


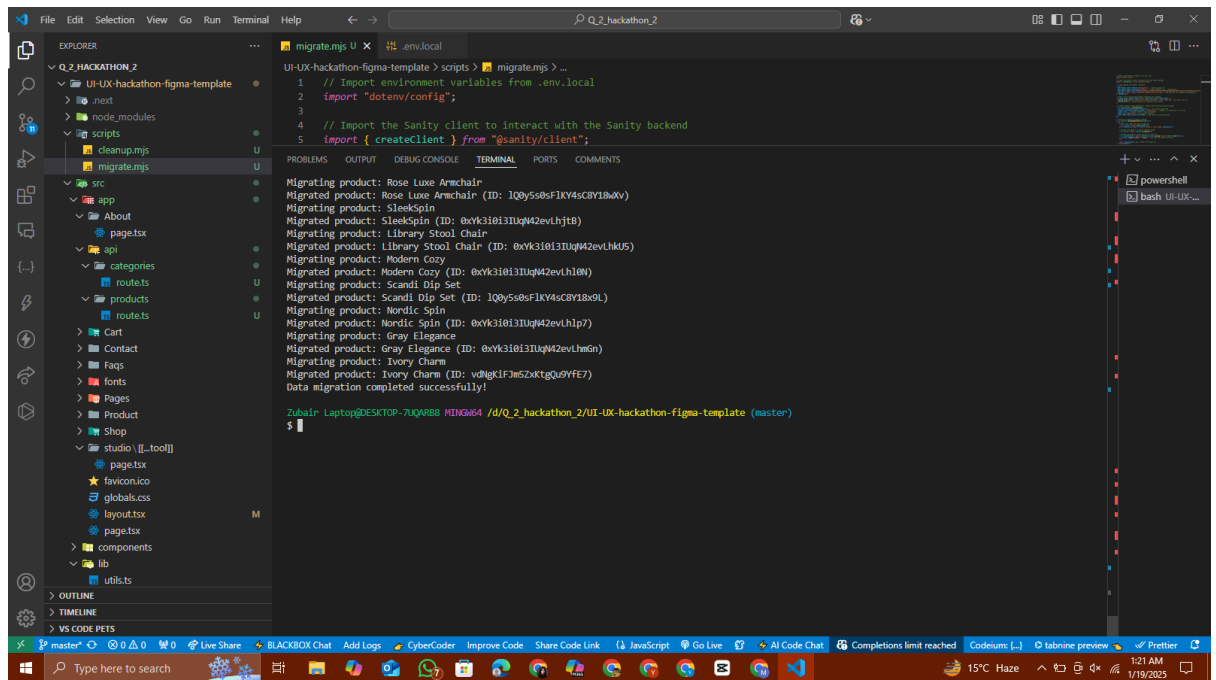
```
1 import { client } from '@sanity/lib/client';
2 export const dynamic = 'force-static';
3
4 export async function GET() {
5   try {
6     const categories = await client.fetch('*[_type == "categories"] {
7       _id,
8       title,
9       imageUrl: image.asset->url
10     }');
11
12     return new Response(JSON.stringify(categories), {
13       headers: { 'Content-Type': 'application/json' },
14     });
15   } catch (error) {
16     console.log(error);
17     return new Response(
18       JSON.stringify({ error: 'Failed to fetch categories' }),
19       {
20         status: 500,
21         headers: { 'Content-Type': 'application/json' },
22       }
23     );
24   }
25 }
```



- Data successfully displayed in the frontend

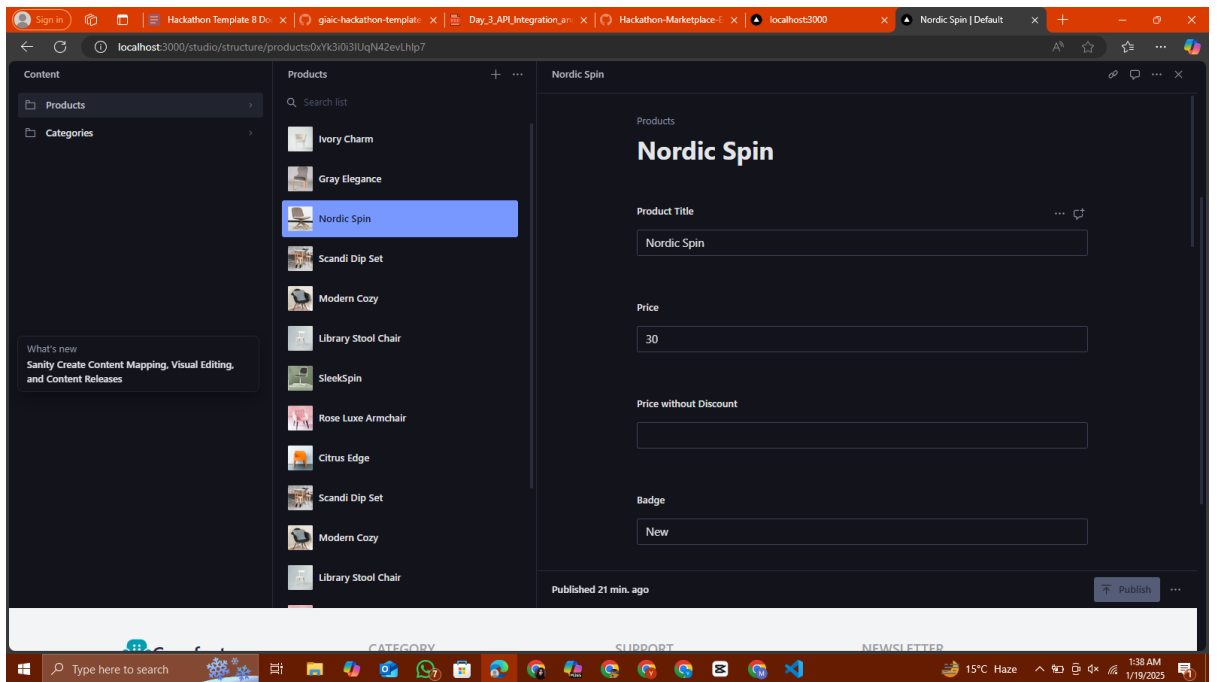
When I run `npm run migrate`, so the data successfully fetch in the terminal.

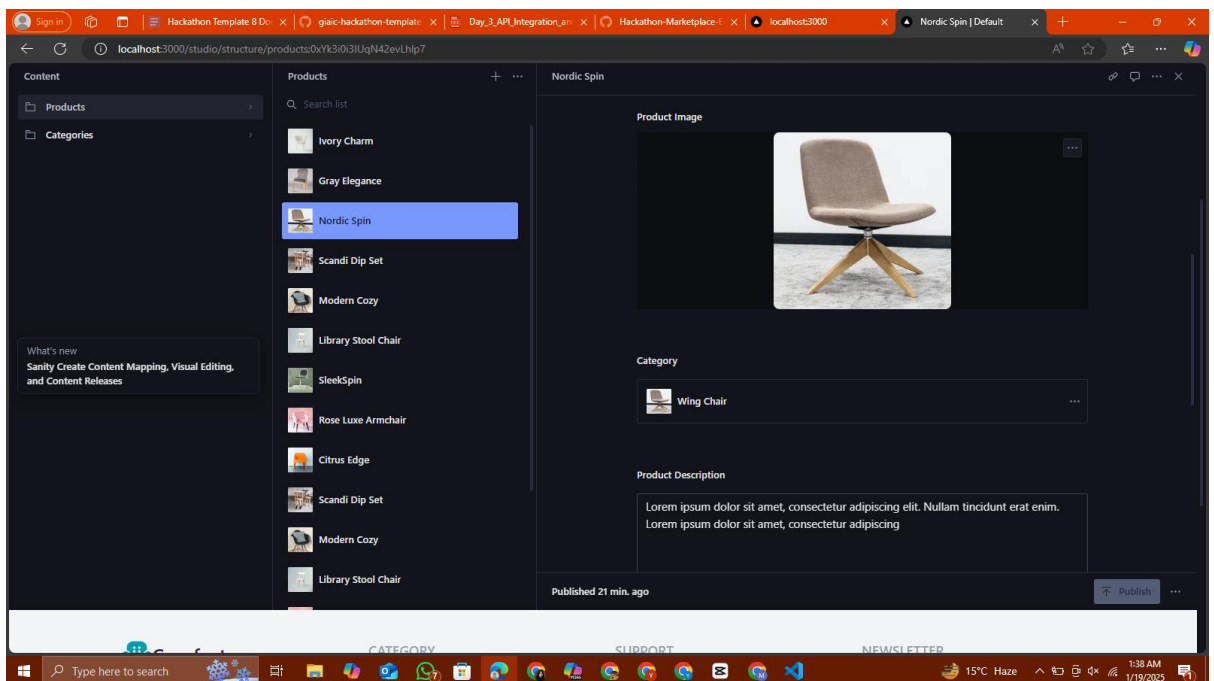
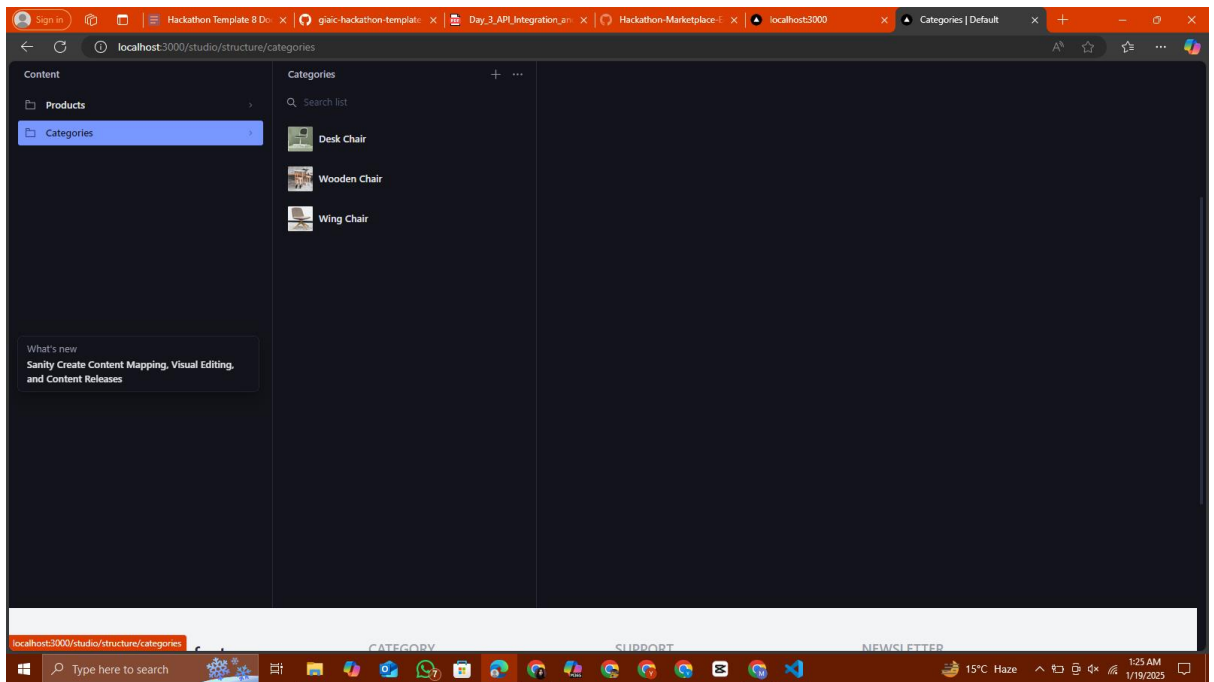




- The output was seamlessly rendered in Sanity Studio and reflected on the frontend.







## Day 3 Checklist: Self-Validation Checklist:

☐ API Understanding: ✓

- ❑ Schema Validation: ✓
- ❑ Data Migration: ✓
- ❑ API Integration in Next.js: ✓
- ❑ Submission Preparation: ✓

### What I learned:

Day 3 of the hackathon has been a transformative experience, teaching me the critical importance of API integration and data migration in modern web development. I gained hands-on experience in working with REST APIs, creating schemas in Sanity CMS, and developing scripts for seamless data transfer. This journey not only enhanced my technical skills but also highlighted the significance of efficient workflows, debugging, and real-time problem-solving in achieving project goals.