

DAY 3 – API INTEGRATION REPORT – [CHAIRY]

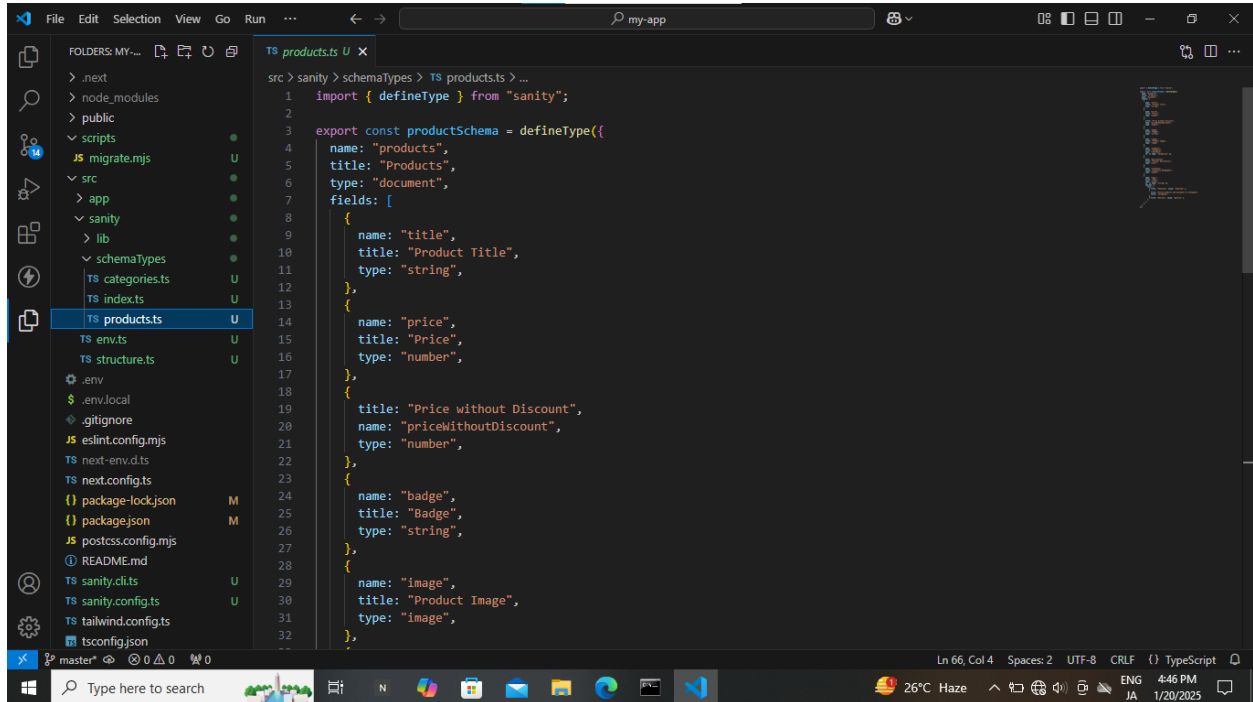
Project Created On Sanity

The screenshot shows the Sanity.io project overview page for a project named 'Temp-8'. The page is displayed in a web browser with the URL <https://www.sanity.io/manage/personal/project/pb8q4dnc>. The project is owned by 'CodeWithFozia' and has a 'Growth Trial' plan. The status is 'Active' and the project ID is 'pb8q4dnc'. The page includes a navigation bar with tabs for 'Getting started', 'Overview' (selected), 'Members', 'Studios', 'Datasets', 'Access', 'Activity', 'Usage', 'Plan', 'API', and 'Settings'. The 'Next steps' section provides instructions to initialize the project with the CLI, showing the command `npm create sanity@latest -- --project pb8q4dnc --dataset produ`. The 'Project members' section shows a button to 'Invite project members'. The 'Activity' section shows '132 / 250k API Requests'. The Windows taskbar at the bottom shows the date as 1/21/2025 and the time as 12:04 PM.

The screenshot shows the Sanity.io project API settings page for the same project 'Temp-8'. The URL is <https://www.sanity.io/manage/personal/project/pb8q4dnc/api>. The 'API' tab is selected in the navigation bar. The page is divided into three sections: 'Webhooks', 'CORS origins', and 'Tokens'. The 'CORS origins' section is expanded, showing a table of hosts that can connect to the project API. The table has three columns: 'ORIGIN', 'CREDENTIALS', and 'CREATED'. The 'Tokens' section is also visible, with a note that tokens are used to authenticate apps and scripts to access the project.

ORIGIN	CREDENTIALS	CREATED
http://localhost:3002	Allowed	6 hours
http://localhost:3000	Allowed	yesterday
http://localhost:3333	Allowed	yesterday

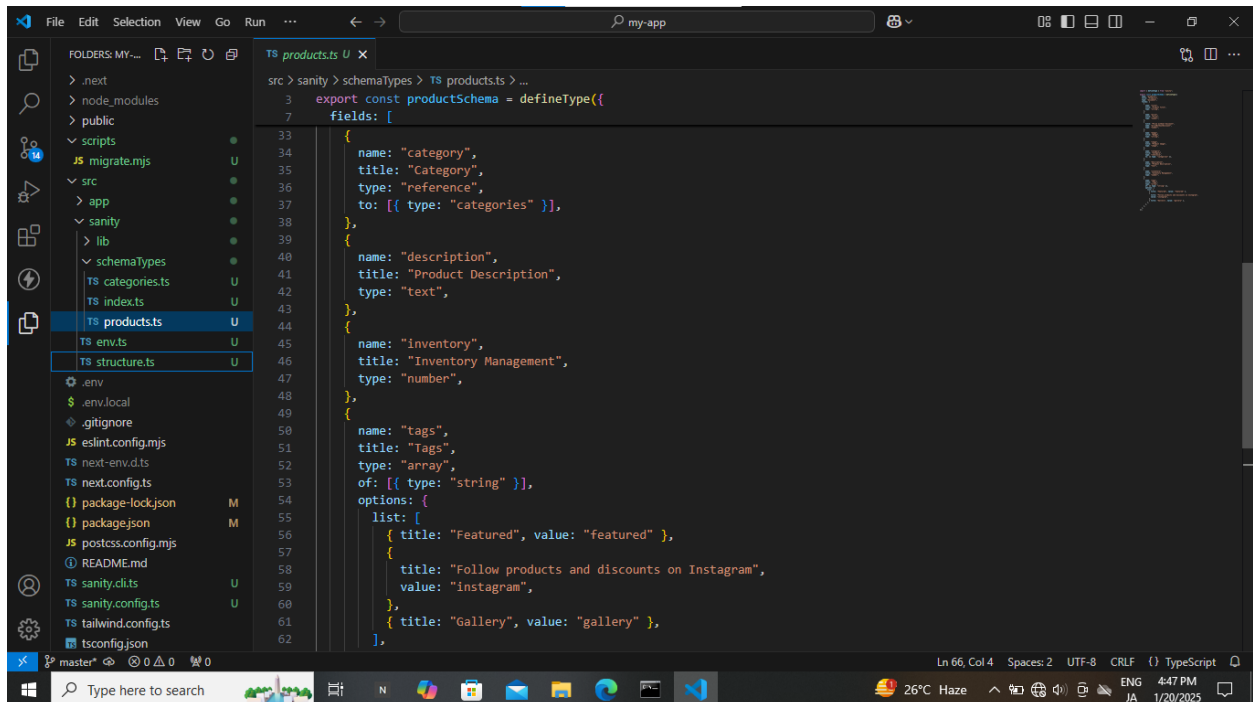
Product.ts File



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with folders like .next, node_modules, public, scripts, src, app, sanity, lib, schemaTypes, TS categories.ts, TS index.ts, TS products.ts (selected), TS env.ts, TS structure.ts, .env, .env.local, .gitignore, JS eslint.config.mjs, TS next-env.d.ts, TS next.config.ts, package-lock.json, package.json, postcss.config.mjs, README.md, TS sanity.d.ts, TS sanity.config.ts, TS tailwind.config.ts, and tsconfig.json. The main editor shows the content of TS products.ts, which is a TypeScript file defining a schema for products. The code is as follows:

```
src > sanity > schemaTypes > TS products.ts > ...
1  import { defineType } from "sanity";
2
3  export const productsSchema = 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 });
```

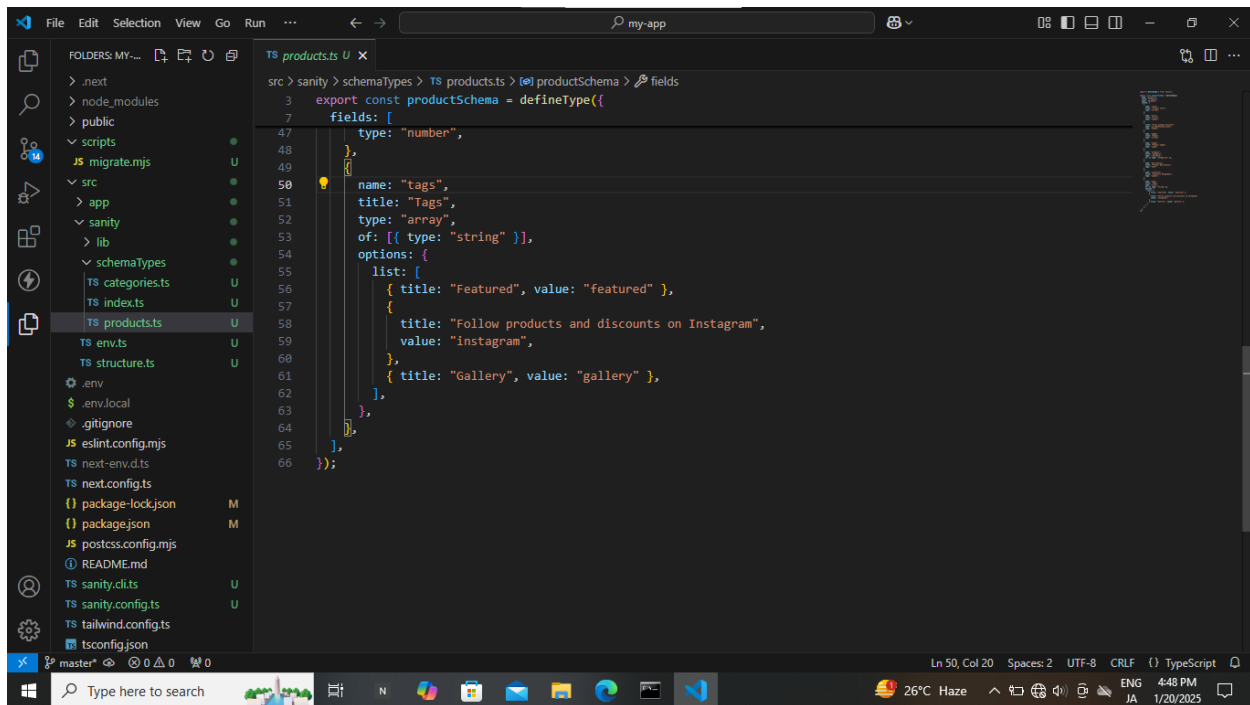
The status bar at the bottom shows the file is at line 66, column 4, with 2 spaces, UTF-8 encoding, CRLF line endings, and TypeScript language. The system tray at the bottom shows the date and time as 4:46 PM on 1/20/2025.



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with folders like .next, node_modules, public, scripts, src, app, sanity, lib, schemaTypes, TS categories.ts, TS index.ts, TS products.ts (selected), TS env.ts, TS structure.ts, .env, .env.local, .gitignore, JS eslint.config.mjs, TS next-env.d.ts, TS next.config.ts, package-lock.json, package.json, postcss.config.mjs, README.md, TS sanity.d.ts, TS sanity.config.ts, TS tailwind.config.ts, and tsconfig.json. The main editor shows the content of TS products.ts, which is a TypeScript file defining a schema for products. The code is as follows:

```
src > sanity > schemaTypes > TS products.ts > ...
3  export const productsSchema = defineType({
7    fields: [
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           { title: "Follow products and discounts on Instagram", value: "instagram" },
58           { title: "Gallery", value: "gallery" },
59         ],
60       },
61     },
62   ],
63 });
```

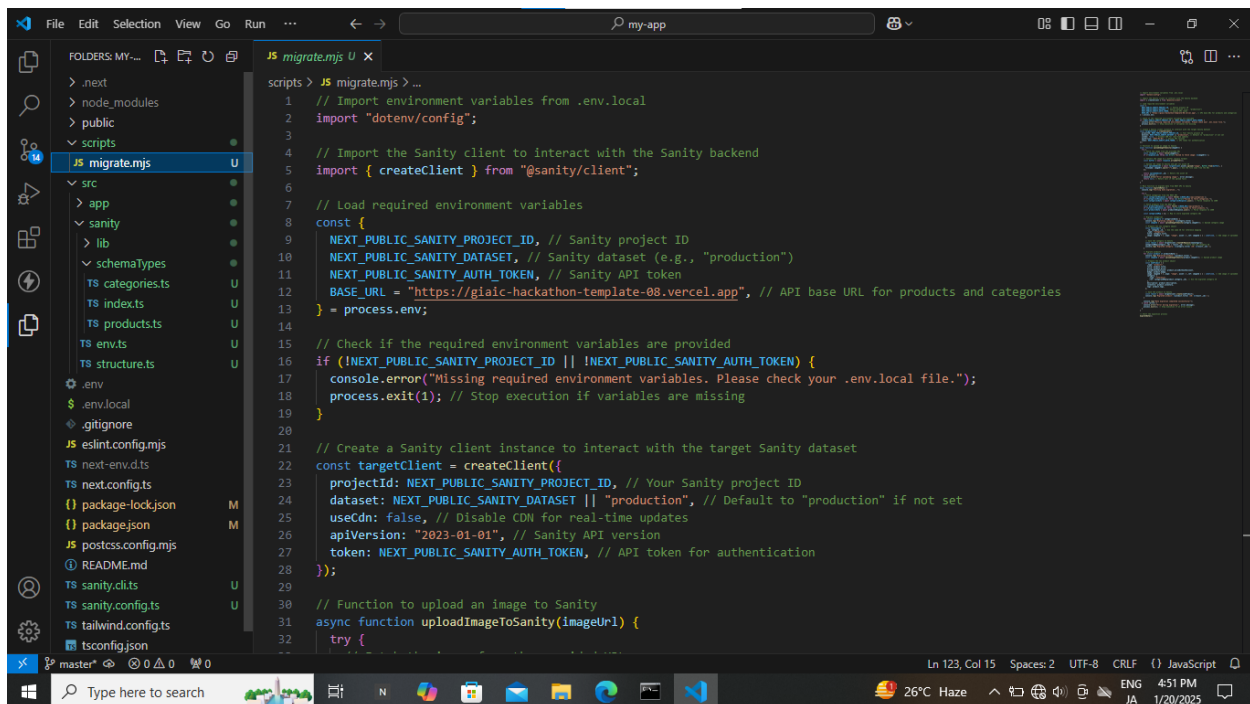
The status bar at the bottom shows the file is at line 66, column 4, with 2 spaces, UTF-8 encoding, CRLF line endings, and TypeScript language. The system tray at the bottom shows the date and time as 4:47 PM on 1/20/2025.



The screenshot shows a Visual Studio Code editor window with a project named "my-app". The file explorer on the left shows a directory structure with folders like ".next", "node_modules", "public", "scripts", "src", "sanity", and "lib". The "src" folder is expanded, showing files like "schemaTypes", "categories.ts", "index.ts", "products.ts", "env.ts", "structure.ts", ".env", ".env.local", ".gitignore", "eslint.config.mjs", "next-env.d.ts", "next.config.ts", "package-lock.json", "package.json", "postcss.config.mjs", "README.md", "sanity.d.ts", "sanity.config.ts", "tailwind.config.ts", and "tsconfig.json". The "products.ts" file is selected and open in the editor. The code defines a TypeScript interface for a product schema. It includes fields for "number", "tags", and "array". The "tags" field is an array of strings with specific options like "Featured", "Follow products and discounts on Instagram", and "Gallery".

```
src > sanity > schemaTypes > TS products.ts > [0] productSchema > fields
3 export const productSchema = defineType({
7   fields: [
47     {
48       type: "number",
49     },
50   ],
51   name: "tags",
52   title: "Tags",
53   type: "array",
54   of: [{ type: "string" }],
55   options: {
56     list: [
57       { title: "Featured", value: "featured" },
58       { title: "Follow products and discounts on Instagram", value: "instagram",
59       },
60       { title: "Gallery", value: "gallery" },
61     ],
62   },
63 },
64 ],
65 },
66 );
```

Migrate.mjs File



The screenshot shows the same Visual Studio Code editor window, but now the "migrate.mjs" file is selected and open in the editor. The code is a JavaScript script that imports environment variables from ".env.local" and the "dotenv/config" package. It also imports the "createClient" function from "@sanity/client". The script checks if the required environment variables (NEXT_PUBLIC_SANITY_PROJECT_ID, NEXT_PUBLIC_SANITY_DATASET, NEXT_PUBLIC_SANITY_AUTH_TOKEN, and BASE_URL) are provided. If any are missing, it logs an error and exits the process. If all are provided, it creates a Sanity client instance and defines an async function "uploadImageToSanity" that takes an "imageUrl" as an argument.

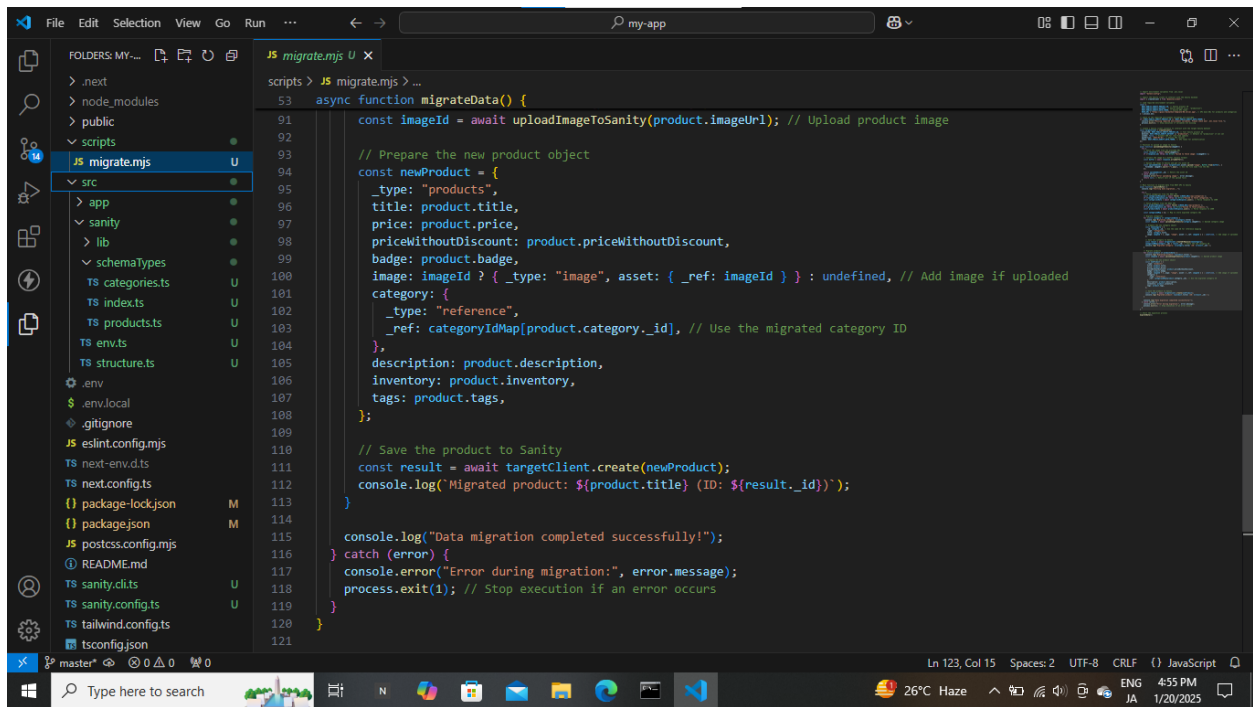
```
scripts > JS migrate.mjs > ...
1 // Import environment variables from .env.local
2 import "dotenv/config";
3
4 // Import the Sanity client to interact with the Sanity backend
5 import { createClient } from "@sanity/client";
6
7 // Load required environment variables
8 const {
9   NEXT_PUBLIC_SANITY_PROJECT_ID, // 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://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 {
```

This screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `migrate.mjs` file with the `uploadImageToSanity` function. The function is an async function that takes `imageUrl` as an argument. It attempts to fetch the image from the provided URL, converts it to a buffer, and uploads it to Sanity. If the upload fails, it throws an error. The function returns the `uploadedAsset_id` if successful, or null if it fails. The status bar at the bottom indicates the cursor is at line 123, column 15.

```
scripts > JS migrate.mjs > ...
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: ${imageUrl}');
36
37     // Convert the image to a buffer (binary format)
38     const buffer = await response.arrayBuffer();
39
40     // Upload the image to Sanity and get its asset ID
41     const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
42       filename: imageUrl.split("/").pop(), // Use the file name from the URL
43     });
44
45     return uploadedAsset.id; // Return the asset ID
46   } catch (error) {
47     console.error("Error uploading image:", error.message);
48     return null; // Return null if the upload fails
49   }
50 }
51
52 // Main function to migrate data from REST API to Sanity
53 async function migrateData() {
54   console.log("Starting data migration...");
55
56   try {
57     // Fetch categories from the REST API
58     const categoriesResponse = await fetch(`${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
```

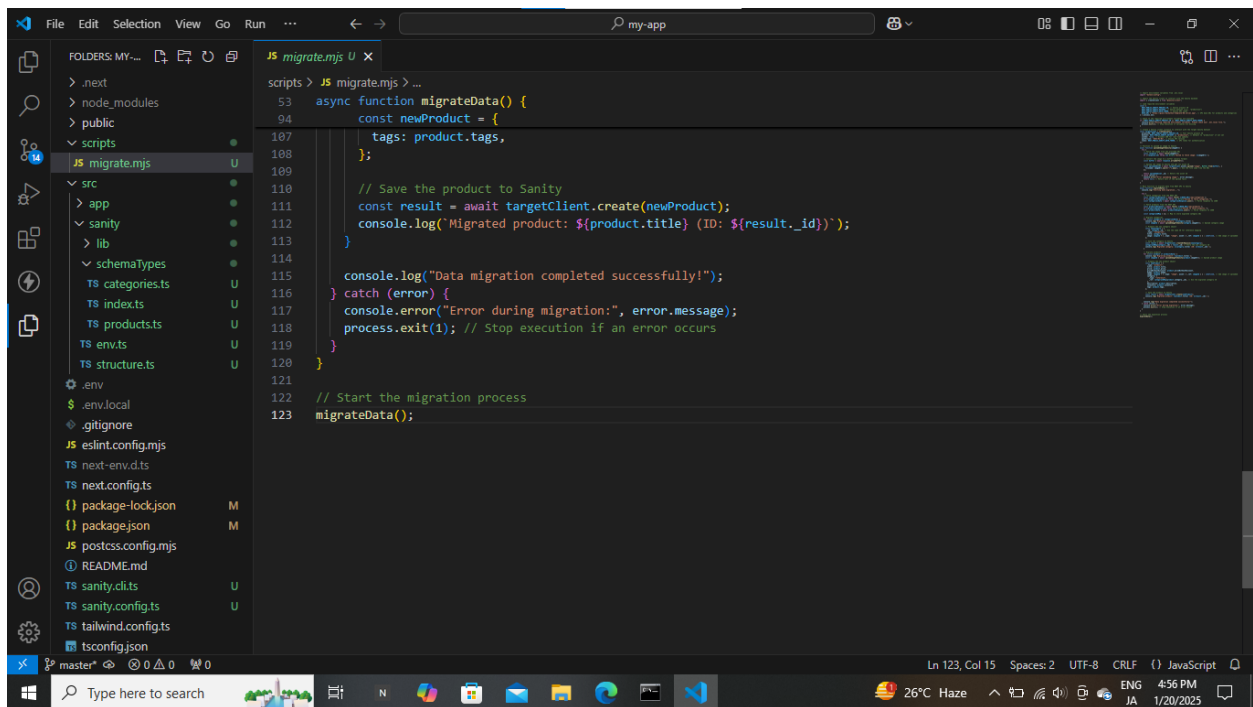
This screenshot shows the VS Code editor with the file explorer on the left. The main editor window shows the `migrate.mjs` file with the `migrateData` function. The function is an async function that fetches products from the REST API, uploads category images to Sanity, and then migrates the categories and products. It uses a `categoryIdMap` to store the new category IDs. The function logs the progress of the migration. The status bar at the bottom indicates the cursor is at line 123, column 15.

```
53 async function migrateData() {
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 imageId = 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: imageId ? { _type: "image", asset: { _ref: imageId } } : 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 imageId = await uploadImageToSanity(product.imageUrl); // Upload product image
```



The screenshot shows the VS Code editor with the file `my-app` open. The file explorer on the left shows the project structure, including `scripts` and `src` folders. The `scripts` folder contains `migrate.mjs`, which is the active file. The code in `migrate.mjs` is as follows:

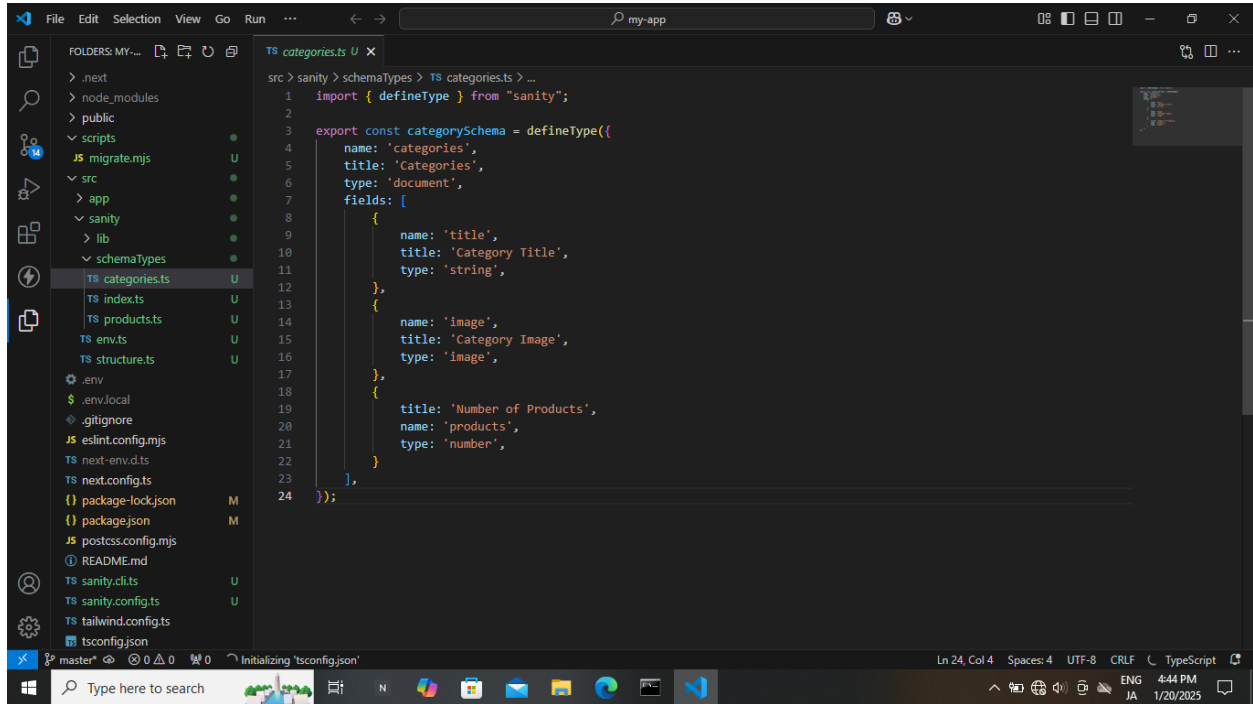
```
53 async function migrateData() {
91   const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image
92
93   // Prepare the new product object
94   const newProduct = {
95     _type: "products",
96     title: product.title,
97     price: product.price,
98     priceWithoutDiscount: 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
116  } catch (error) {
117    console.error("Error during migration:", error.message);
118    process.exit(1); // Stop execution if an error occurs
119  }
120
121 }
```



The screenshot shows the VS Code editor with the file `my-app` open. The file explorer on the left shows the project structure, including `scripts` and `src` folders. The `scripts` folder contains `migrate.mjs`, which is the active file. The code in `migrate.mjs` is as follows:

```
53 async function migrateData() {
94   const newProduct = {
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
116   } catch (error) {
117     console.error("Error during migration:", error.message);
118     process.exit(1); // Stop execution if an error occurs
119   }
120
121 }
122
123 // Start the migration process
124 migrateData();
```

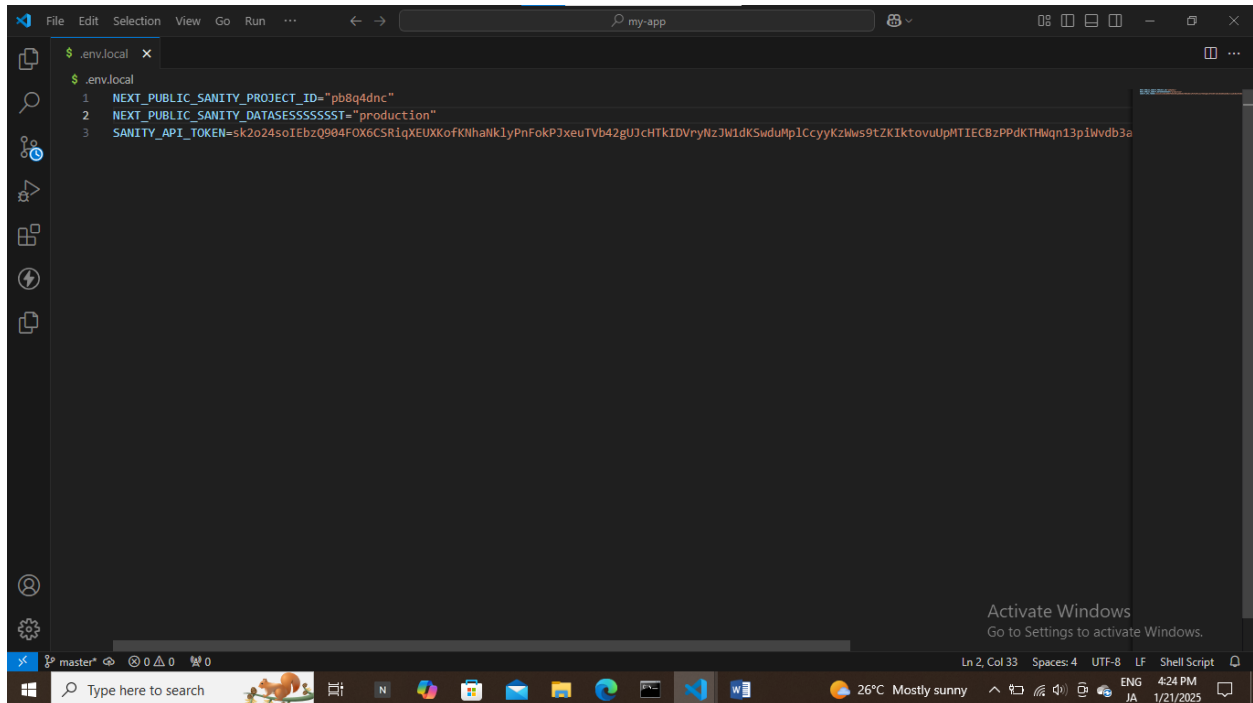
category.ts File



The screenshot shows the VS Code editor with the file `category.ts` open. The file is located in the `src > sanity > schemaTypes` directory. The code defines a Sanity schema for categories.

```
src > sanity > schemaTypes > TS categories.ts > ...
1  import { defineType } from "sanity";
2
3  export const categorySchema = defineType({
4    name: 'categories',
5    title: 'Categories',
6    type: 'document',
7    fields: [
8      {
9        name: 'title',
10       title: 'Category Title',
11       type: 'string',
12     },
13     {
14       name: 'image',
15       title: 'Category Image',
16       type: 'image',
17     },
18     {
19       title: 'Number of Products',
20       name: 'products',
21       type: 'number',
22     },
23   ],
24 });
```

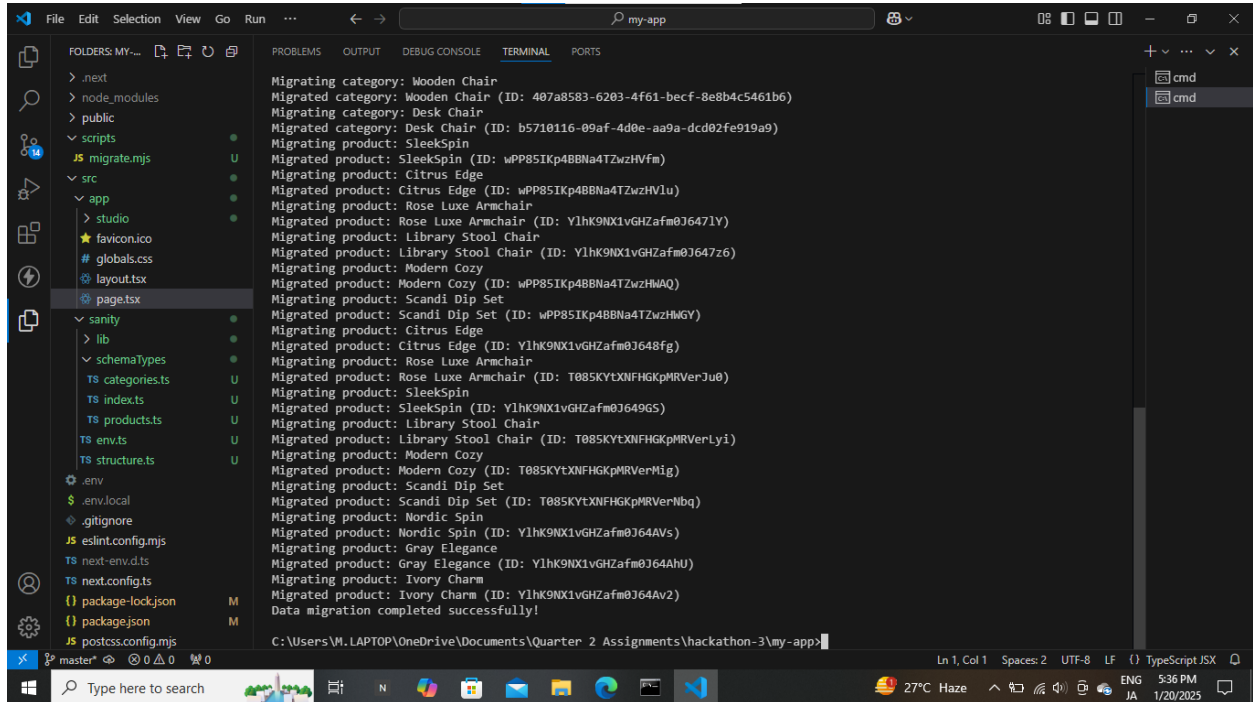
.env File



The screenshot shows the VS Code editor with the `.env.local` file open. The file contains environment variables for the application.

```
$.env.local
$.env.local
1  NEXT_PUBLIC_SANITY_PROJECT_ID="pb8q4dnc"
2  NEXT_PUBLIC_SANITY_DATASESSSSST="production"
3  SANITY_API_TOKEN=sk2o24soIEbzQ904FOX6CSRIqXEUkKofKNhaNklyPnFokPJxeuTVb42GujCHTKIDVryNzJW1dKSwdUmp1CyyKzIws9tZKIKtovuUpMTIECBzPPdKTHmqn13p1Wvdb3a
```

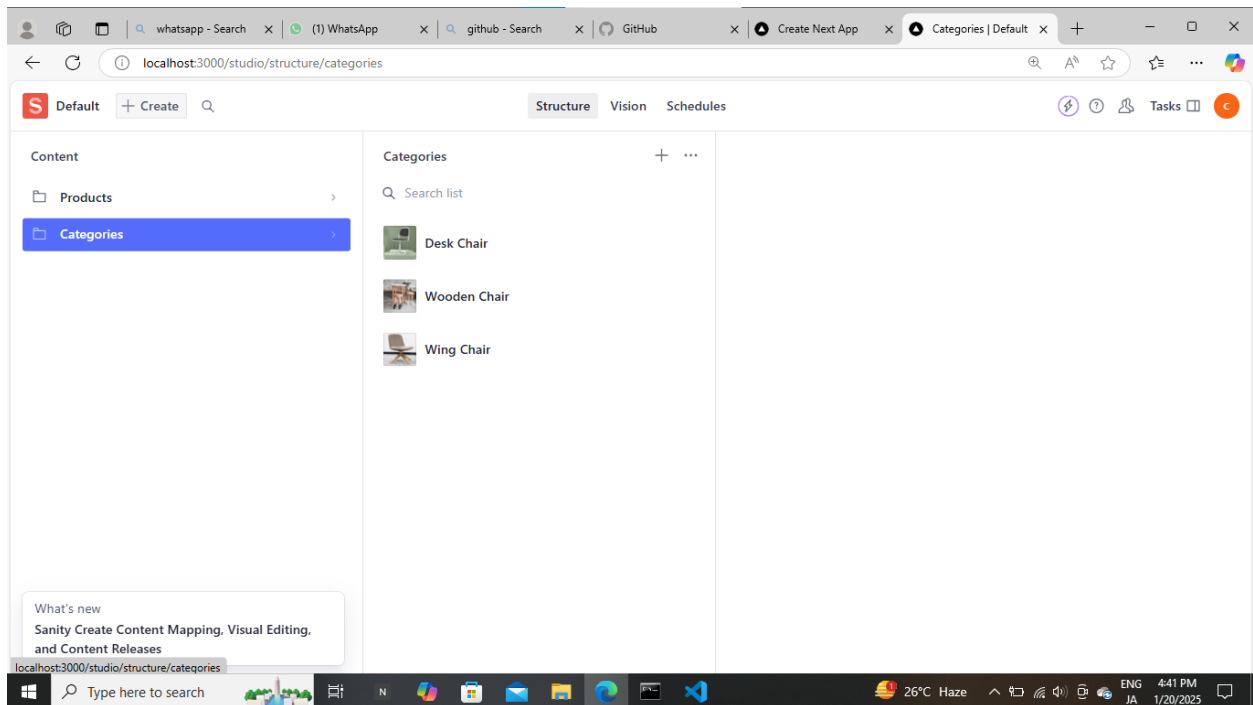
Data Migration

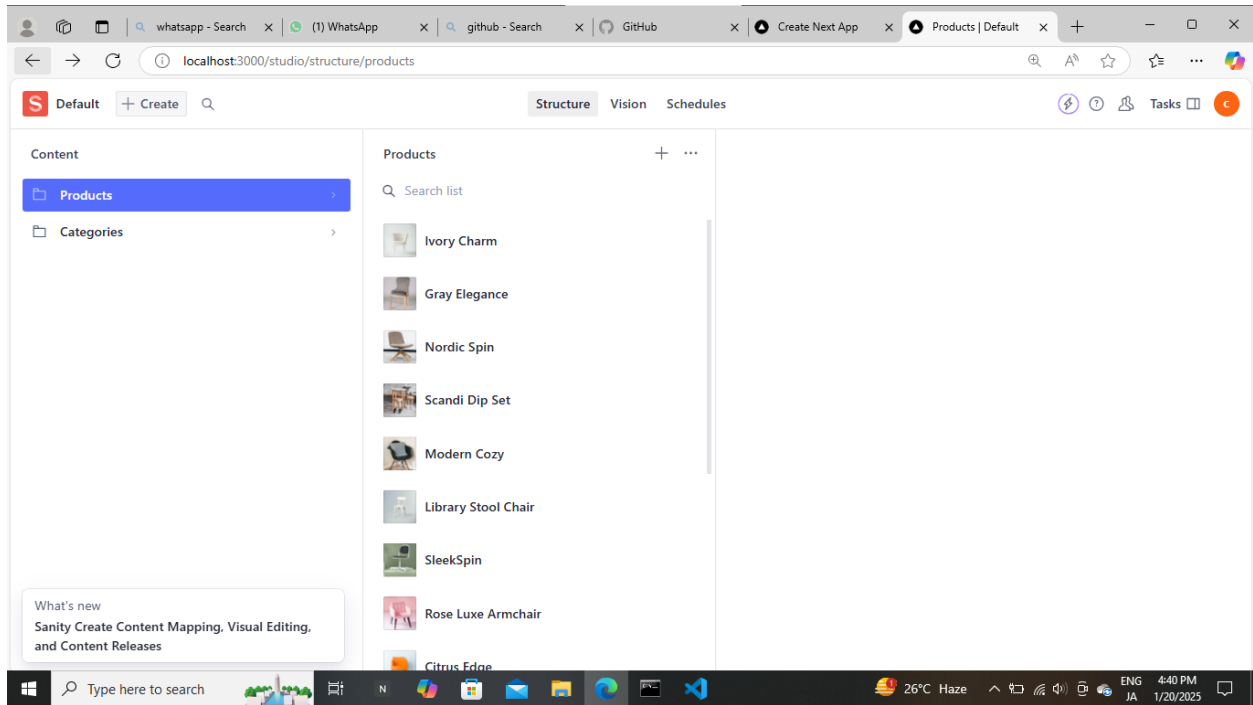


The screenshot shows a Visual Studio Code editor with a terminal window open. The terminal displays a series of migration logs for a Sanity CMS project. The logs show the migration of categories and products, including their IDs and names. The migration process is completed successfully.

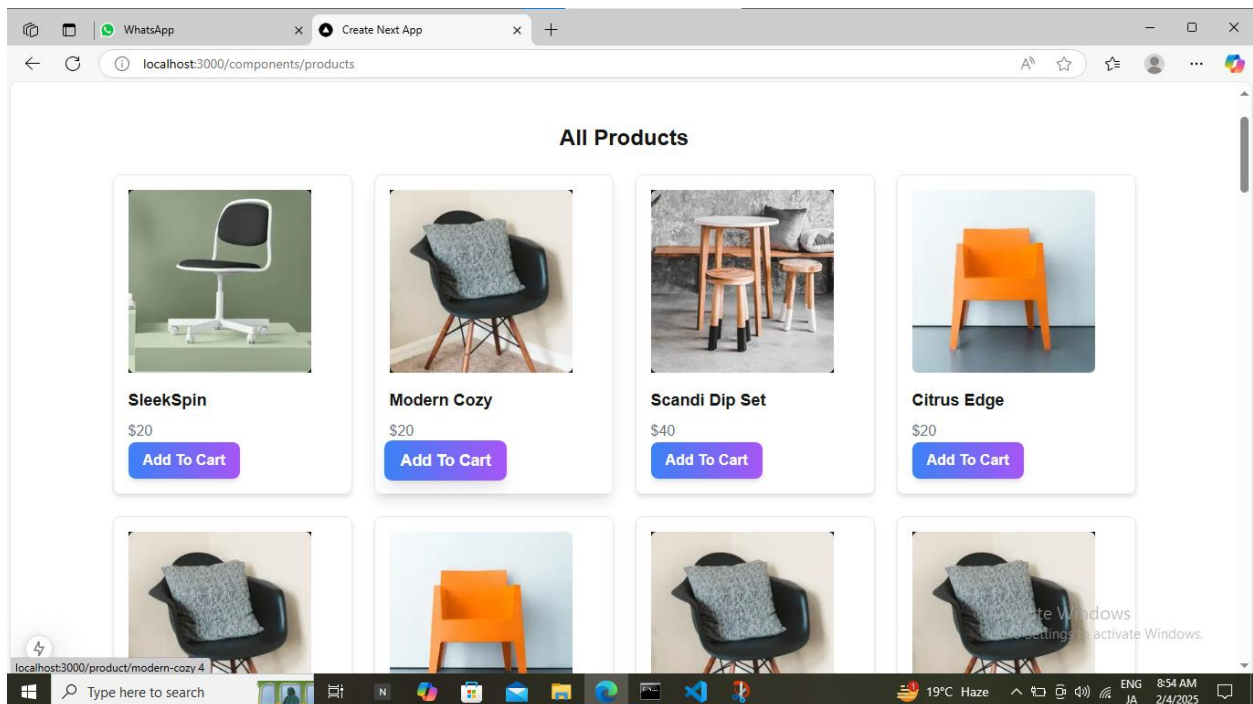
```
File Edit Selection View Go Run ... my-app
FOLDERS: MY-...
> .next
> node_modules
> public
> scripts
  migrate.mjs
  src
    app
      studio
        faviconico
        # globals.css
        layout.tsx
        page.tsx
      sanity
        lib
        schemasTypes
          categories.ts
          indexes
          products.ts
          env.ts
          structure.ts
        .env
        .env.local
        .gitignore
        JS eslint.config.mjs
        TS next-env.d.ts
        TS next.config.ts
        package-lock.json
        package.json
        JS postcss.config.mjs
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Migrating category: Wooden Chair
Migrated category: Wooden Chair (ID: 407a8583-6203-4f61-becf-8e8b4c5461b6)
Migrating category: Desk Chair
Migrated category: Desk Chair (ID: b5710116-09af-4d0e-aa9a-dcd02fe919a9)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: wPP85IKp48BNA4TzwzHVfm)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: wPP85IKp48BNA4TzwzHVlu)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: YlhK9NX1vGHZafm0J6471Y)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: YlhK9NX1vGHZafm0J647z6)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: wPP85IKp48BNA4TzwzHWAQ)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: wPP85IKp48BNA4TzwzHMGY)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: YlhK9NX1vGHZafm0J648fg)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: T085KYtXNFGKpMRVerJu0)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: YlhK9NX1vGHZafm0J649G5)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: T085KYtXNFGKpMRVerLy1)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: T085KYtXNFGKpMRVerMig)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: T085KYtXNFGKpMRVerNbq)
Migrating product: Nordic Spin
Migrated product: Nordic Spin (ID: YlhK9NX1vGHZafm0J64AVs)
Migrating product: Gray Elegance
Migrated product: Gray Elegance (ID: YlhK9NX1vGHZafm0J64AHU)
Migrating product: Ivory Charm
Migrated product: Ivory Charm (ID: YlhK9NX1vGHZafm0J64Av2)
Data migration completed successfully!
C:\Users\VM.LAPTOP\OneDrive\Documents\Quarter 2 Assignments\hackathon-3\my-app>
```

Data Import In Sanity





Data fetch to Frontend



Made by Fouzia Bibi!