

Day 3 - API Integration Report

E-Commerce - Selling Furniture Website

Installing Sanity into my Project
npm create sanity@latest

```
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon202 >npm create sanity@latest

> my-app@0.1.0 npx
> create-sanity

✓ You are logged in as ikramjan717@gmail.com using GitHub
✓ Fetching existing projects

? Create a new project or select an existing one Create new project
? Your project name: Final-Hackathon-Template-02
Your content will be stored in a dataset that can be public or private, depending on
whether you want to query your content with or without authentication.
The default dataset configuration has a public dataset named "production".
? Use the default dataset configuration? Yes
✓ Creating dataset
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes

Success! Your Sanity configuration files have been added to this project

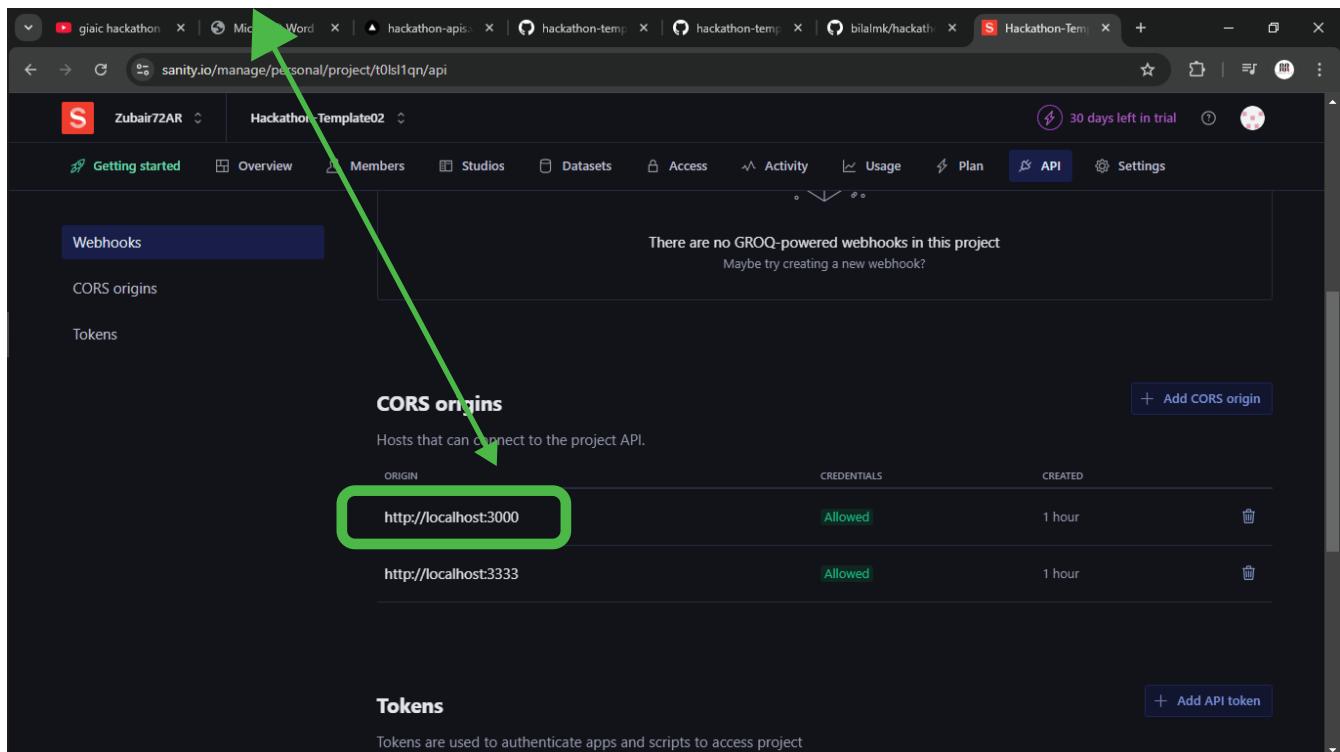
C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon202>
```

Success!

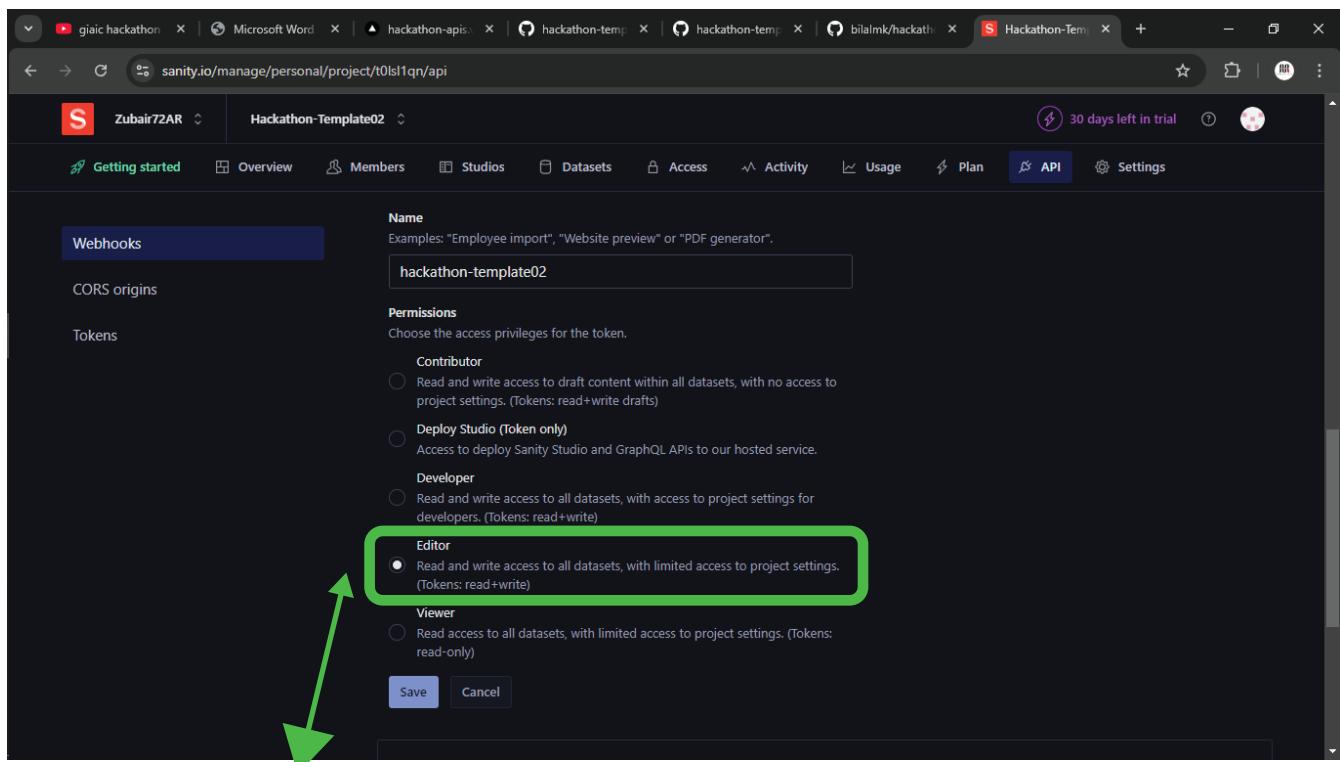
Sanity Installed Successfully

Zubair Ahmed
Saturday - 2 to 5

CORS origins is created
<https://http://localhost:3000>



A screenshot of a web browser showing the Sanity project settings for 'Hackathon-Template02'. The 'CORS origins' section is highlighted with a green arrow pointing from the top-left. It shows two entries: 'http://localhost:3000' and 'http://localhost:3333', both with the status 'Allowed'. The 'Tokens' section is also visible below it.



A screenshot of the Sanity project settings for 'Hackathon-Template02'. The 'Tokens' section is highlighted with a green arrow pointing from the bottom-left. A new token is being created with the name 'hackathon-template02'. In the 'Permissions' section, the 'Editor' option is selected, indicated by a green box and a green arrow pointing to it. The 'Save' button is at the bottom.

Creating API Token

Provide a name and Clicked on Editor Option

Zubair Ahmed
Saturday - 2 to 5

API Token is Created

Copying Token Code

The screenshot shows the Sanity project settings page for 'Hackathon-Template02'. In the 'Tokens' section, a new token named 'hackathon-template02' is listed with 'Editor' permissions and was created 'just now'. A callout box highlights the token code: `sk6EFDFZ9t0cv5f3dVWRcNFbx1RTSfnGk8oxy4XVs0hVksUqydhnnsc4y5UwoKq2m3gV42omZb1Du4xDcPc30zxgkZ4msyXkOKDtnBvNCiZfly5M9X9B76h7P4kSm0C7Ar8u4MbYC1F8p4s5ptsWxm83ezNNmIyRPfh2leHk9Qmd9mgHcq`. A green arrow points from the top-left towards this callout box.

The screenshot shows a Code Editor window with a dark theme. The file '.env.local' is open, containing the following environment variables:

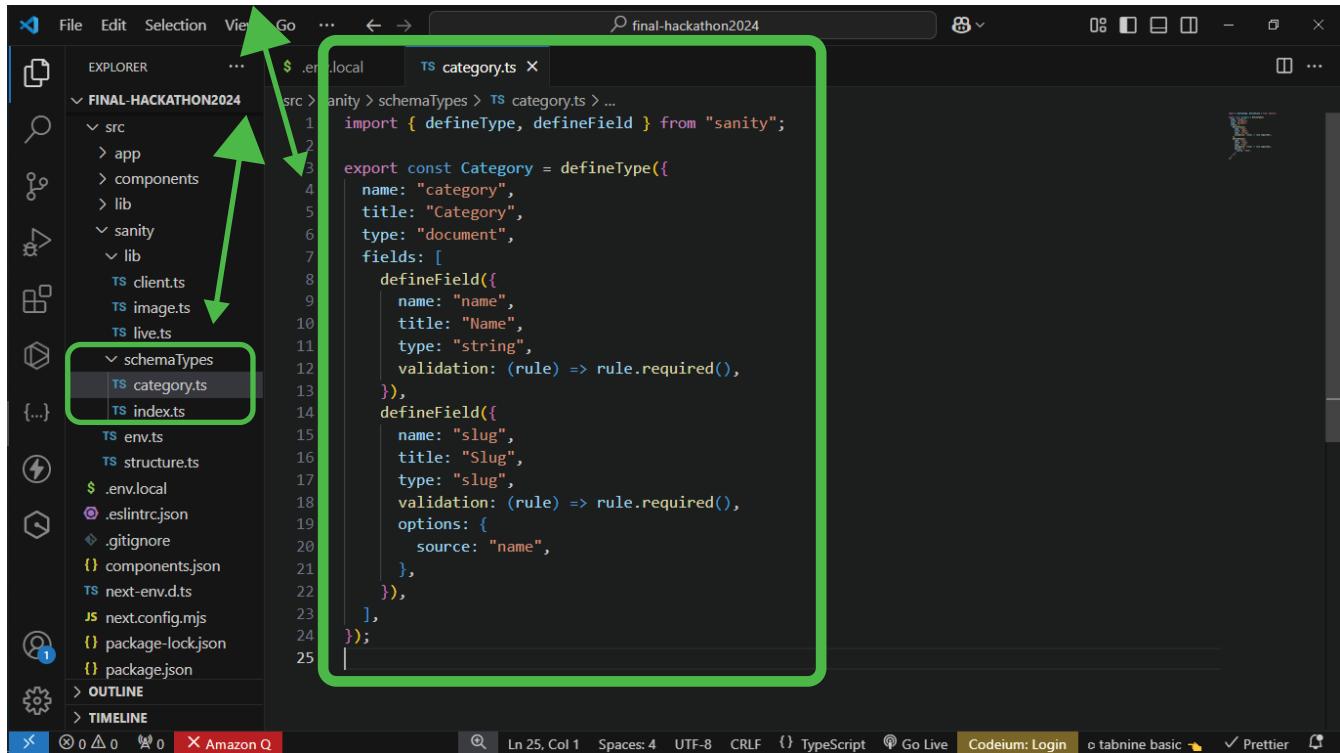
```
NEXT_PUBLIC_SANITY_PROJECT_ID="t0ls1qn"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="sk6EFDFZ9t0cv5f3dVWRcNFbx1RTSfnGk8oxy4XVs0hVksUqydhnnsc4y5UwoKq2m3gV42omZb1Du4xDcPc30zxgkZ4msyXkOKDtnBvNCiZfly5M9X9B76h7P4kSm0C7Ar8u4MbYC1F8p4s5ptsWxm83ezNNmIyRPfh2leHk9Qmd9mgHcq"
```

A green arrow points from the bottom of the previous screenshot towards the .env.local file in the editor. Another green arrow points upwards from the bottom of the editor towards the token code in the file.

**Added Project ID, Dataset, and API Token
in the .env.Local File of my Project**

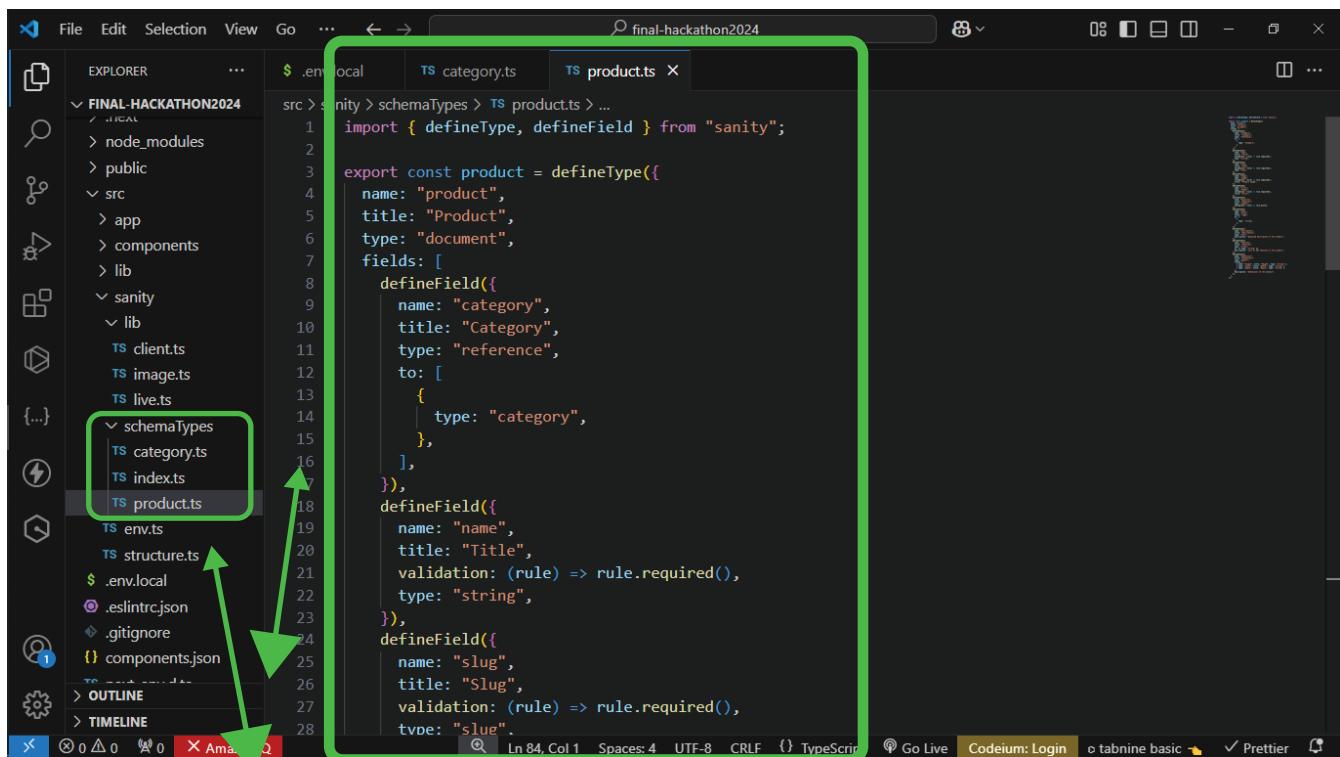
Zubair Ahmed
Saturday - 2 to 5

Added Category Schema in the SchemaTypes



```
File Edit Selection View Go ... $ .env.local ts category.ts ...
sanity > schemaTypes > TS category.ts > ...
import { defineType, defineField } from "sanity";

export const Category = defineType({
  name: "category",
  title: "Category",
  type: "document",
  fields: [
    defineField({
      name: "name",
      title: "Name",
      type: "string",
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: "slug",
      title: "Slug",
      type: "slug",
      validation: (rule) => rule.required(),
      options: {
        source: "name",
      },
    }),
  ],
});
```



```
File Edit Selection View Go ... $ .env.local ts category.ts ts products.ts ...
sanity > schemaTypes > TS products.ts > ...
import { defineType, defineField } from "sanity";

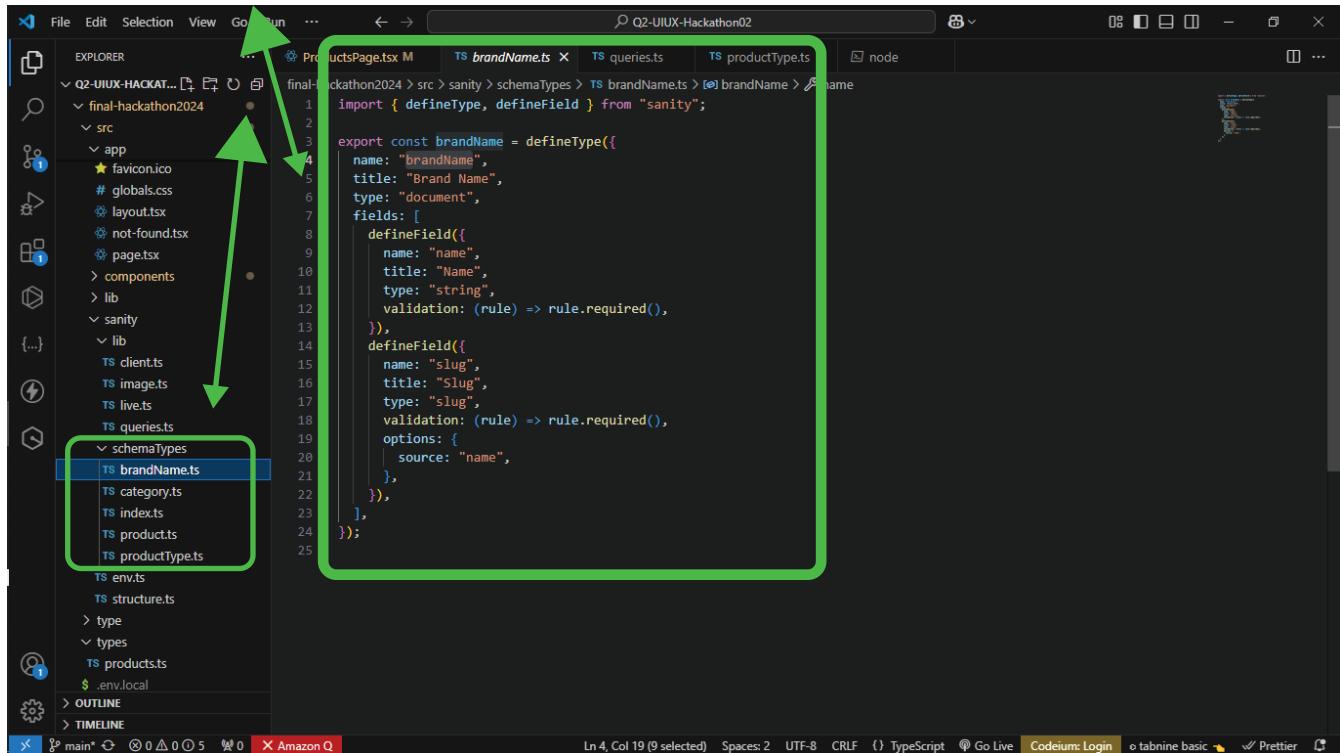
export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    defineField({
      name: "category",
      title: "Category",
      type: "reference",
      to: [
        {
          type: "category",
        },
      ],
    }),
    defineField({
      name: "name",
      title: "Title",
      validation: (rule) => rule.required(),
      type: "string",
    }),
    defineField({
      name: "slug",
      title: "Slug",
      validation: (rule) => rule.required(),
      type: "slug",
    }),
  ],
});
```

Added Products Schema in the SchemaTypes

Adjustments made to schemas

Filtrering by Brand Name

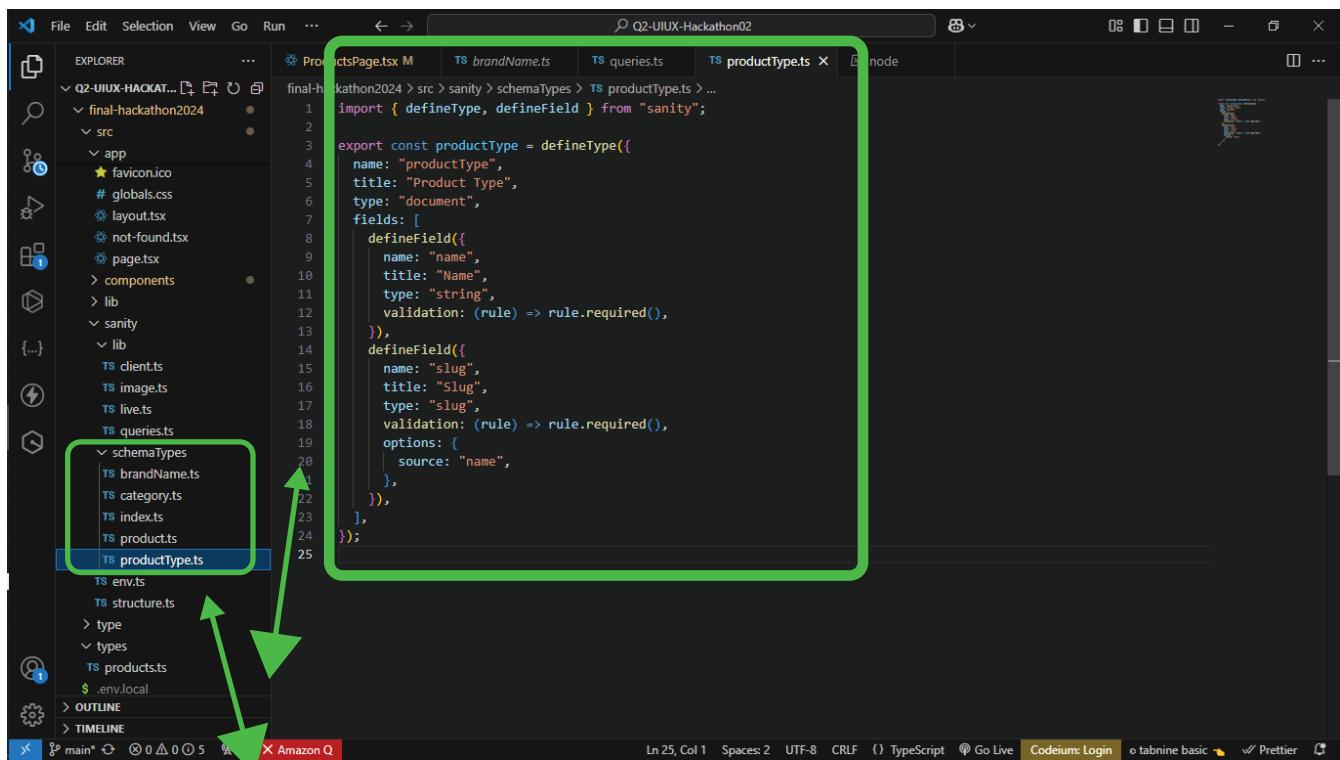
Added Brand Name Schema in the SchemaTypes



The screenshot shows the VS Code interface with the 'brandName.ts' file open in the center editor tab. The file content defines a 'brandName' schema type with fields for 'name' and 'slug'. A green box highlights the entire code block. On the left, the Explorer sidebar shows the project structure, including the 'schemaTypes' folder which contains 'brandName.ts' and other files like 'category.ts', 'index.ts', and 'product.ts'. A green arrow points from the top of the sidebar towards the 'brandName.ts' file.

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

export const brandName = defineType({
  name: "brandName",
  title: "Brand Name",
  type: "document",
  fields: [
    defineField({
      name: "name",
      title: "Name",
      type: "string",
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: "slug",
      title: "Slug",
      type: "slug",
      validation: (rule) => rule.required(),
      options: {
        source: "name",
      },
    }),
  ],
});
```



The screenshot shows the VS Code interface with the 'productType.ts' file open in the center editor tab. The file content defines a 'productType' schema type with fields for 'name' and 'slug'. A green box highlights the entire code block. On the left, the Explorer sidebar shows the project structure, including the 'schemaTypes' folder which contains 'brandName.ts' and other files like 'category.ts', 'index.ts', and 'product.ts'. A green arrow points from the bottom of the sidebar towards the 'productType.ts' file.

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

export const productType = defineType({
  name: "productType",
  title: "Product Type",
  type: "document",
  fields: [
    defineField({
      name: "name",
      title: "Name",
      type: "string",
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: "slug",
      title: "Slug",
      type: "slug",
      validation: (rule) => rule.required(),
      options: {
        source: "name",
      },
    }),
  ],
});
```

Added Products Type Schema in the SchemaTypes

Filtrering by Products Type

Zubair Ahmed
Saturday - 2 to 5

Adjustments made to schemas

```
final-hackathon2024 > src > sanity > schemaTypes > ts product.ts > [s] product.ts X
1 import { defineType, defineField } from "sanity";
2
3 export const product = defineType({
4   name: "product",
5   title: "Product",
6   type: "document",
7   fields: [
8     defineField({
9       name: "category",
10      title: "Category",
11      type: "reference",
12      to: [
13        {
14          type: "category",
15        },
16      ],
17    }),
18
19    defineField({
20      name: "productType",
21      title: "Product Type",
22      type: "reference",
23      to: [
24        {
25          type: "productType",
26        },
27      ],
28    }),
29
30    defineField({
31      name: "brandName",
32      title: "Brand Name",
33      type: "reference",
34      to: [
35        {
36          type: "brandName",
37        },
38      ],
39    }),
40
41    defineField({
42      name: "name",
43      title: "Title",
44      validation: (rule) => rule.required(),
45      type: "string",
46    })
47  }
48});
```

Added Reference to the PRODUCT Schema File

Products Type and Brand Type Reference Added

Products Type and Brand Type Added and Imported new schemas to the INDEX.TS file

The screenshot shows the VS Code interface with the title bar "Q2-UIUX-Hackathon02". In the Explorer sidebar, under the "Q2-UIUX-HACKTHON02" project, there is a "schemaTypes" folder containing "brandName.ts", "category.ts", and "index.ts". The "index.ts" file is selected and open in the editor. A green box highlights the following code:

```
1 import { type SchemaTypeDefinition } from "sanity";
2 import { Category } from "./category";
3 import { product } from "./product";
4 import { productType } from "./productType";
5 import { brandName } from "./brandName";
6
7 export const schema: { types: SchemaTypeDefinition[] } = {
8   types: [Category, product, productType, brandName],
9 };
10
```

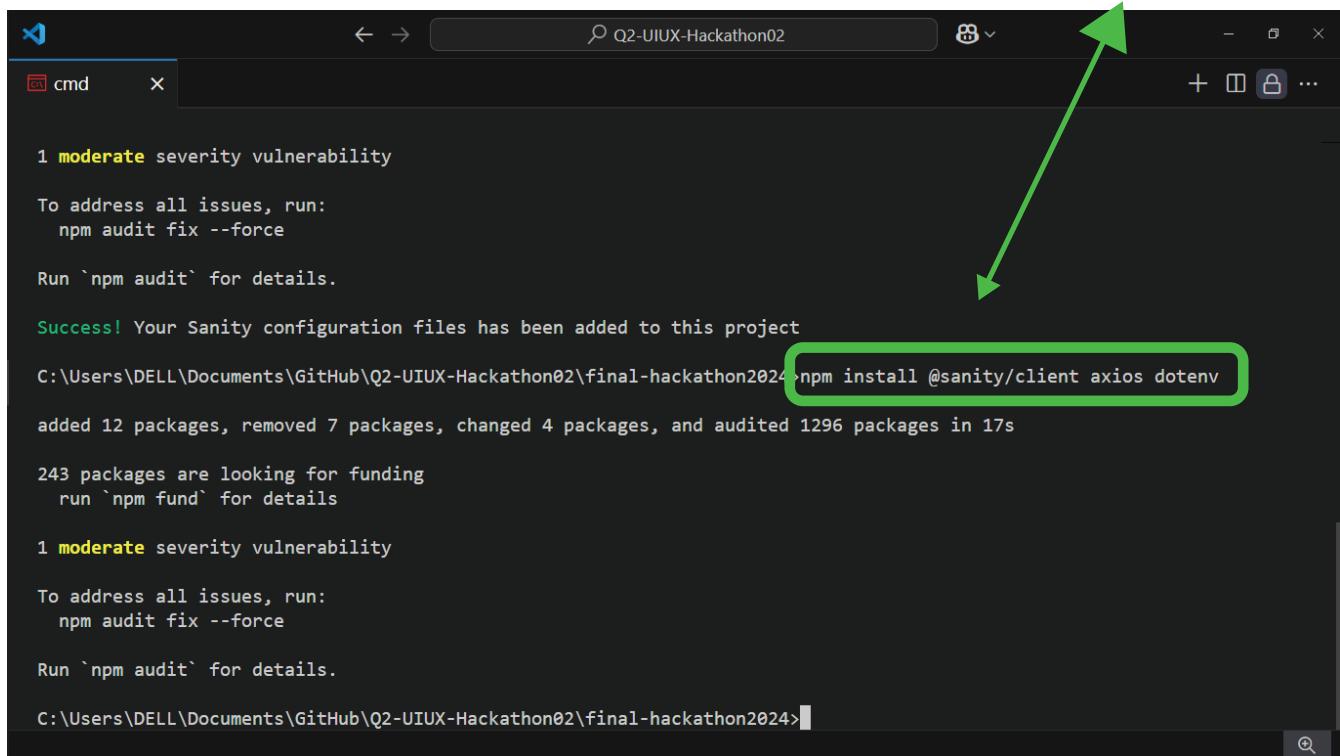
The screenshot shows the VS Code interface with the title bar "Q2-UIUX-Hackathon02". In the Explorer sidebar, under the "scripts" folder, there is a file named "importSanityData.mjs". The file is selected and open in the editor. A green box highlights the following code:

```
1 import { createClient } from "@sanity/client";
2 import axios from "axios";
3 import dotenv from "dotenv";
4 import { fileURLToPath } from "url";
5 import path from "path";
6 import slugify from "slugify";
7
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, "../.env.local") });
11
12 const client = createClient({
13   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
14   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
15   useCdn: false,
16   token: process.env.SANITY_API_TOKEN,
17   apiVersion: "2021-08-31",
18 });
19
20 async function uploadImageToSanity(imageUrl) {
21   try {
22     console.log(`Uploading image: ${imageUrl}`);
23     const response = await axios.get(imageUrl, { responseType: "arraybuffer" });
24     const buffer = Buffer.from(response.data);
25     const asset = await client.assets.upload("image", buffer, {
26       filename: imageUrl.split("/").pop(),
27     });
28   } catch (error) {
29     console.error(error);
30   }
31 }
```

Script Folder / ImportSanityData.mjs File Created

Zubair Ahmed
Saturday - 2 to 5

Installing Axios npm install @sanity/client axios dotenv

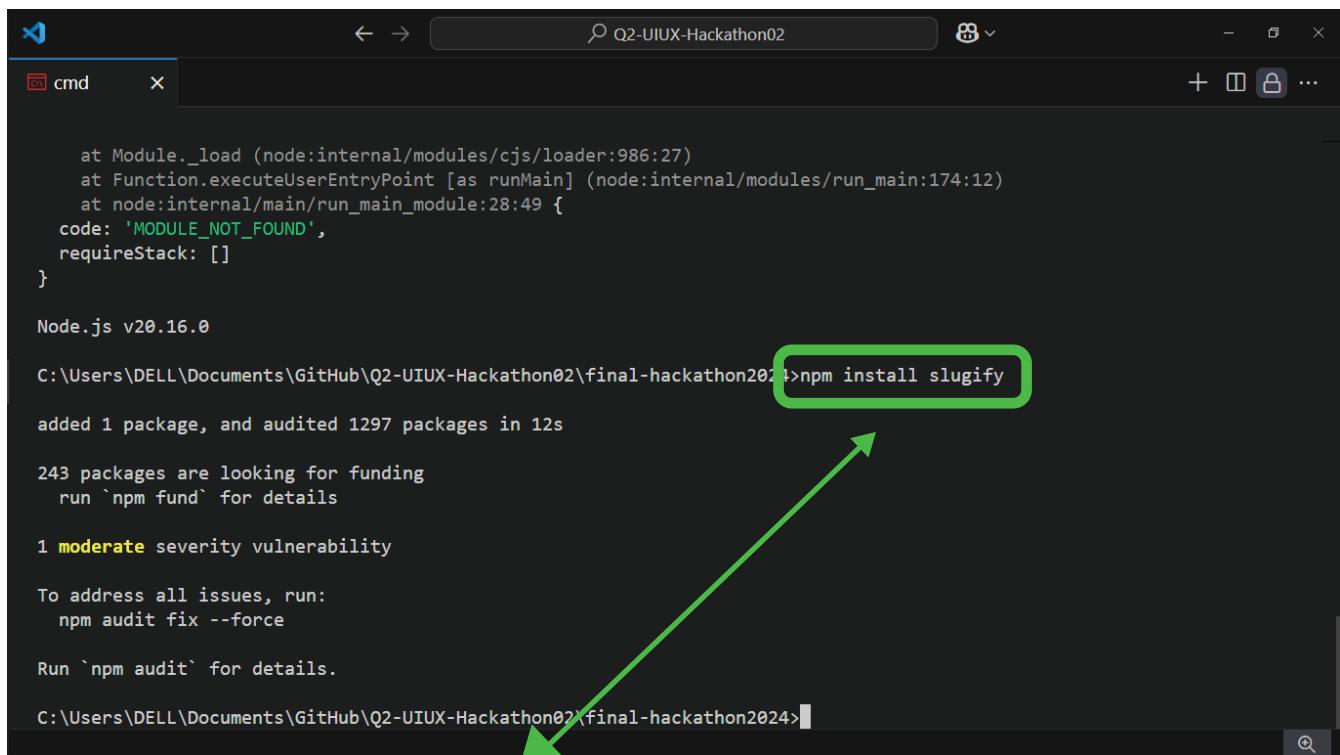


```
1 moderate severity vulnerability
To address all issues, run:
  npm audit fix --force
Run `npm audit` for details.

Success! Your Sanity configuration files has been added to this project
C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon2024>npm install @sanity/client axios dotenv
added 12 packages, removed 7 packages, changed 4 packages, and audited 1296 packages in 17s
243 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability
To address all issues, run:
  npm audit fix --force
Run `npm audit` for details.

C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon2024>
```



```
at Module._load (node:internal/modules/cjs/loader:986:27)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:174:12)
at node:internal/main/run_main_module:28:49 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js v20.16.0
C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon2024>npm install slugify
added 1 package, and audited 1297 packages in 12s
243 packages are looking for funding
  run `npm fund` for details

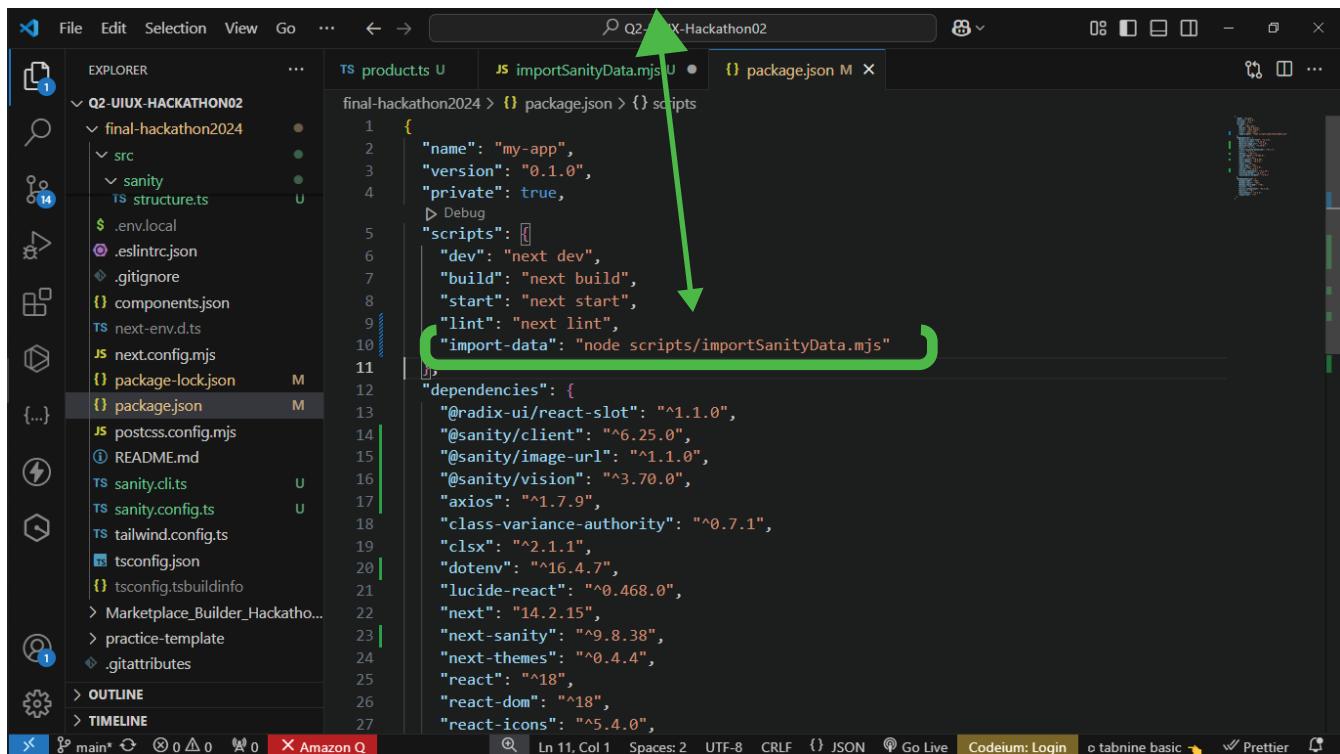
1 moderate severity vulnerability
To address all issues, run:
  npm audit fix --force
Run `npm audit` for details.

C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon2024>
```

npm install slugify
Installing Slugify

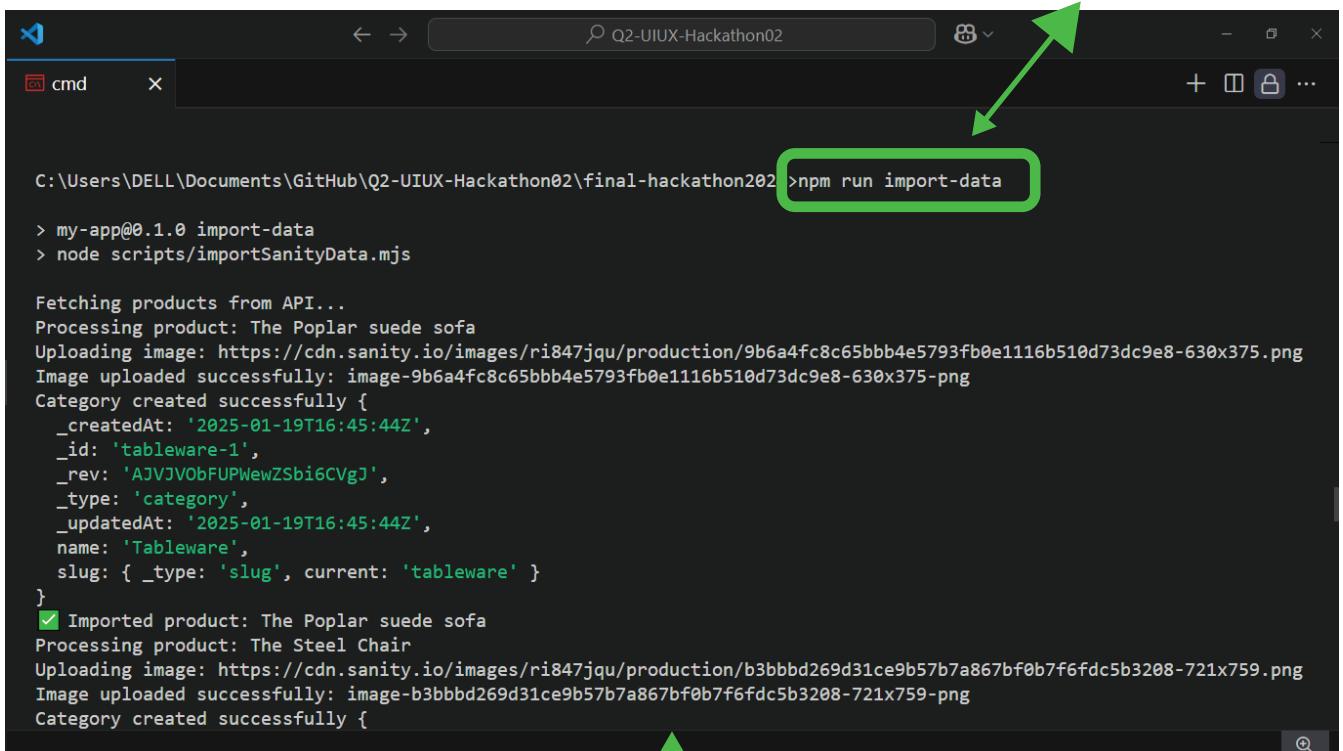
Zubair Ahmed
Saturday - 2 to 5

Updating PACKAGE.JSON
"import-data": "node scripts/importSanityData.mjs"



```
1  {
2    "name": "my-app",
3    "version": "0.1.0",
4    "private": true,
5    > Debug
6    "scripts": [
7      "dev": "next dev",
8      "build": "next build",
9      "start": "next start",
10     "lint": "next lint",
11     "import-data": "node scripts/importSanityData.mjs"
12   ]
13   "dependencies": {
14     "@radix-ui/react-slot": "^1.1.0",
15     "@sanity/client": "^6.25.0",
16     "@sanity/image-url": "^1.1.0",
17     "@sanity/vision": "^3.70.0",
18     "axios": "^1.7.9",
19     "class-variance-authority": "^0.7.1",
20     "clsx": "^2.1.1",
21     "dotenv": "^16.4.7",
22     "lucide-react": "^0.468.0",
23     "next": "14.2.15",
24     "next-sanity": "9.8.38",
25     "next-themes": "^0.4.4",
26     "react": "^18",
27     "react-dom": "^18",
28     "react-icons": "^5.4.0"
29   }
30 }
```

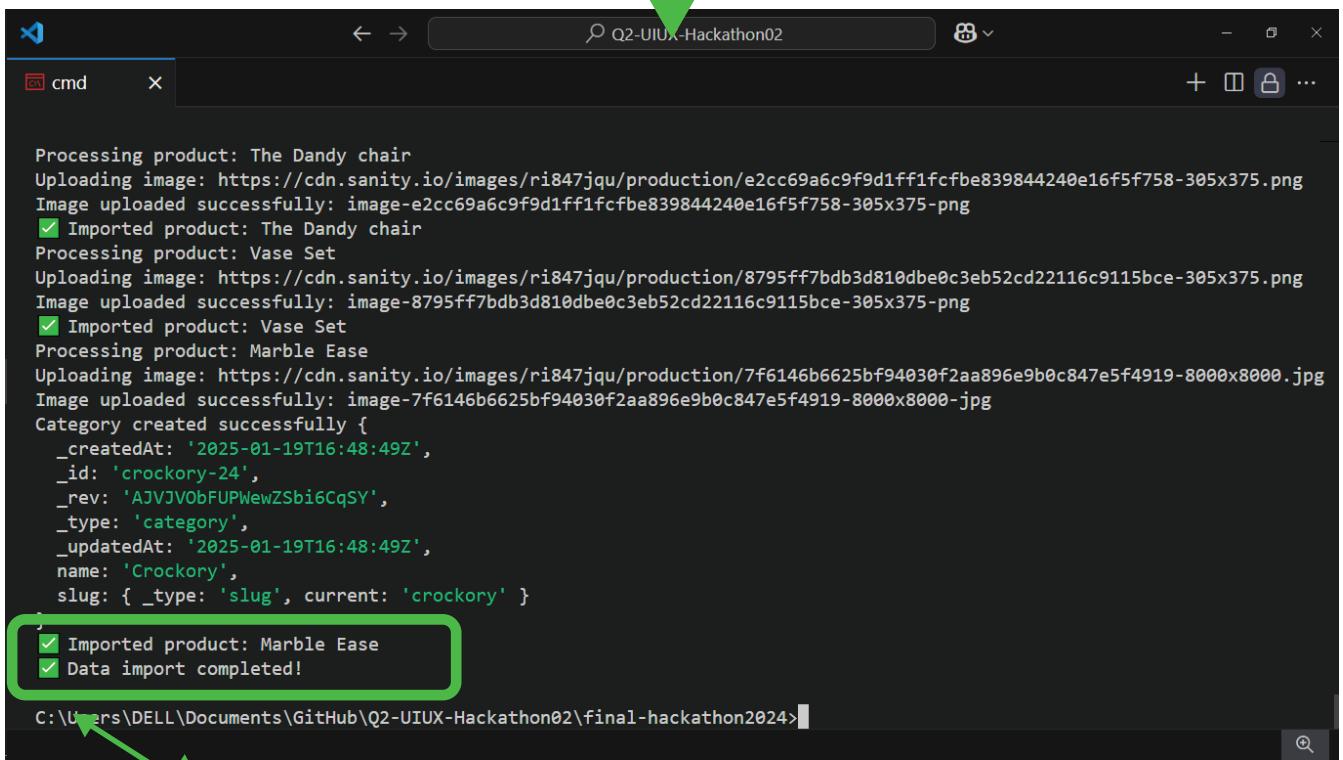
API integration process is Ready
npm run import-data



```
C:\Users\DELL\Documents\GitHub\Q2-UIUX-Hackathon02\final-hackathon202 >npm run import-data

> my-app@0.1.0 import-data
> node scripts/importSanityData.mjs

Fetching products from API...
Processing product: The Poplar suede sofa
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/9b6a4fc8c65bbb4e5793fb0e1116b510d73dc9e8-630x375.png
Image uploaded successfully: image-9b6a4fc8c65bbb4e5793fb0e1116b510d73dc9e8-630x375.png
Category created successfully {
  _createdAt: '2025-01-19T16:45:44Z',
  _id: 'tableware-1',
  _rev: 'AJVJVObFUPWewZSbi6CVgJ',
  _type: 'category',
  _updatedAt: '2025-01-19T16:45:44Z',
  name: 'Tableware',
  slug: { _type: 'slug', current: 'tableware' }
}
✓ Imported product: The Poplar suede sofa
Processing product: The Steel Chair
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/b3bbbd269d31ce9b57b7a867bf0b7f6fdc5b3208-721x759.png
Image uploaded successfully: image-b3bbbd269d31ce9b57b7a867bf0b7f6fdc5b3208-721x759.png
Category created successfully {
```



```
Processing product: The Dandy chair
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/e2cc69a6c9f9d1ff1fcfbe839844240e16f5f758-305x375.png
Image uploaded successfully: image-e2cc69a6c9f9d1ff1fcfbe839844240e16f5f758-305x375.png
✓ Imported product: The Dandy chair
Processing product: Vase Set
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/8795ff7bdb3d810dbe0c3eb52cd22116c9115bce-305x375.png
Image uploaded successfully: image-8795ff7bdb3d810dbe0c3eb52cd22116c9115bce-305x375.png
✓ Imported product: Vase Set
Processing product: Marble Ease
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg
Image uploaded successfully: image-7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg
Category created successfully {
  _createdAt: '2025-01-19T16:48:49Z',
  _id: 'crockory-24',
  _rev: 'AJVJVObFUPWewZSbi6CqSY',
  _type: 'category',
  _updatedAt: '2025-01-19T16:48:49Z',
  name: 'Crockory',
  slug: { _type: 'slug', current: 'crockory' }
}
✓ Imported product: Marble Ease
✓ Data import completed!
```

Data import completed!

API Process Done Successfully

Zubair Ahmed
Saturday - 2 to 5

Sanity Received Data

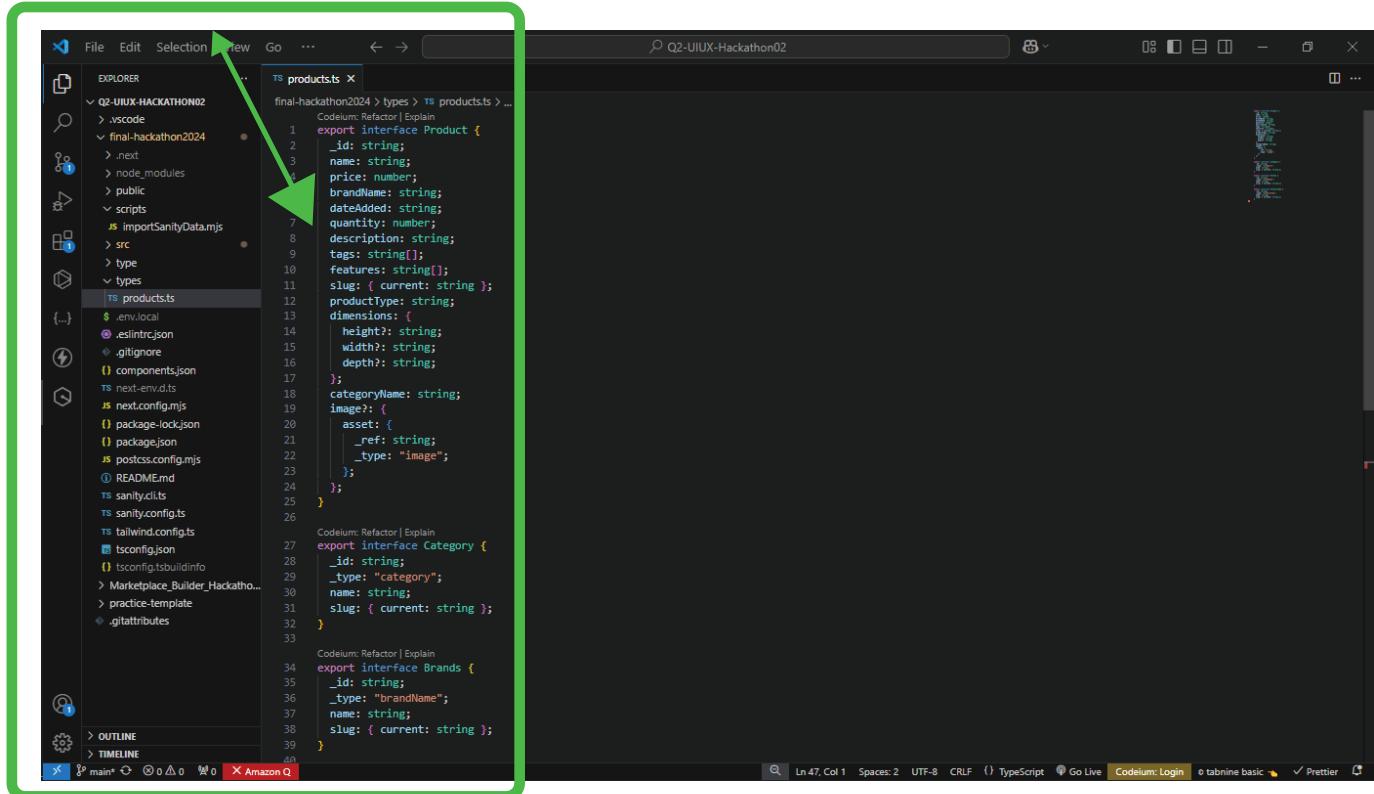
The screenshot shows the Contentful Studio interface for managing assets. On the left, there's a sidebar with a tree view of content types: Category, Product, Product Type, and Brand Name. The 'Product Type' node under Product is highlighted with a green box and has a green arrow pointing from the sidebar to it. The 'Brand Name' node under Product is also highlighted with a green box and has a green arrow pointing from the sidebar to it. The main content area displays an asset for 'Rustic Vase Set'. It includes a preview image, the title 'Rustic Vase Set', and several fields: 'Category' set to 'Plant Pots', 'Product Type' set to 'Vase', and 'Brand Name' set to 'Luminex'. There are also status indicators like 'Published 19 hr. ago' and a 'Publish' button.

Products Type and Brand Type Reference Added

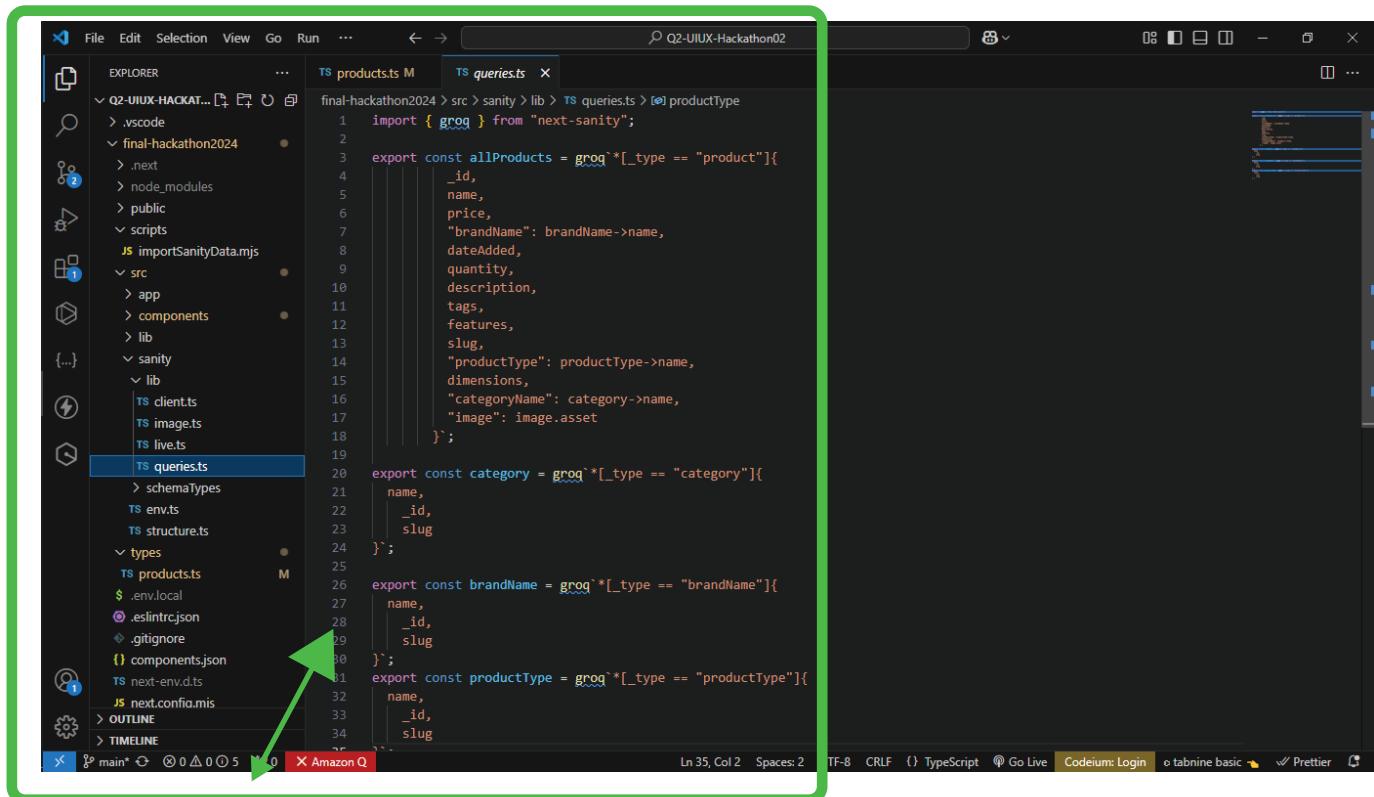
Zubair Ahmed
Saturday - 2 to 5

Frontend Work

Define interface Type



```
File Edit Selection View Go ... Q2-UIUX-Hackathon02 types products.ts final-hackathon2024 > types > TS products.ts ... Codeium: Refactor | Explain
1 export interface Product {
2   _id: string;
3   name: string;
4   price: number;
5   brandName: string;
6   dateAdded: string;
7   quantity: number;
8   description: string;
9   tags: string[];
10  features: string[];
11  slug: { current: string };
12  productType: string;
13  dimensions: {
14    height?: string;
15    width?: string;
16    depth?: string;
17  };
18  categoryName: string;
19  image?: {
20    asset: {
21      _ref: string;
22      _type: "image";
23    };
24  };
25 }
26
Codeium: Refactor | Explain
27 export interface Category {
28   _id: string;
29   _type: "category";
30   name: string;
31   slug: { current: string };
32 }
33
Codeium: Refactor | Explain
34 export interface Brands {
35   _id: string;
36   _type: "brandName";
37   name: string;
38   slug: { current: string };
39 }
```



```
File Edit Selection View Go Run ... Q2-UIUX-Hackathon02 src sanity lib queries.ts productType
1 import { groq } from "next-sanity";
2
3 export const allProducts = groq`*[_type == "product"]{
4   _id,
5   name,
6   price,
7   "brandName": brandName->name,
8   dateAdded,
9   quantity,
10  description,
11  tags,
12  features,
13  slug,
14  "productType": productType->name,
15  dimensions,
16  "categoryName": category->name,
17  "image": image.asset
18 }`;
19
20 export const category = groq`*[_type == "category"]{
21   name,
22   _id,
23   slug
24 }`;
25
26 export const brandName = groq`*[_type == "brandName"]{
27   name,
28   _id,
29   slug
30 }`;
31 export const productType = groq`*[_type == "productType"]{
32   name,
33   _id,
34   slug
35 }`;
```

GROG Quiries Type

Zubair Ahmed
Saturday - 2 to 5

Frontend Work

Fetching Data from Sanity

Displaying Data for Frontend

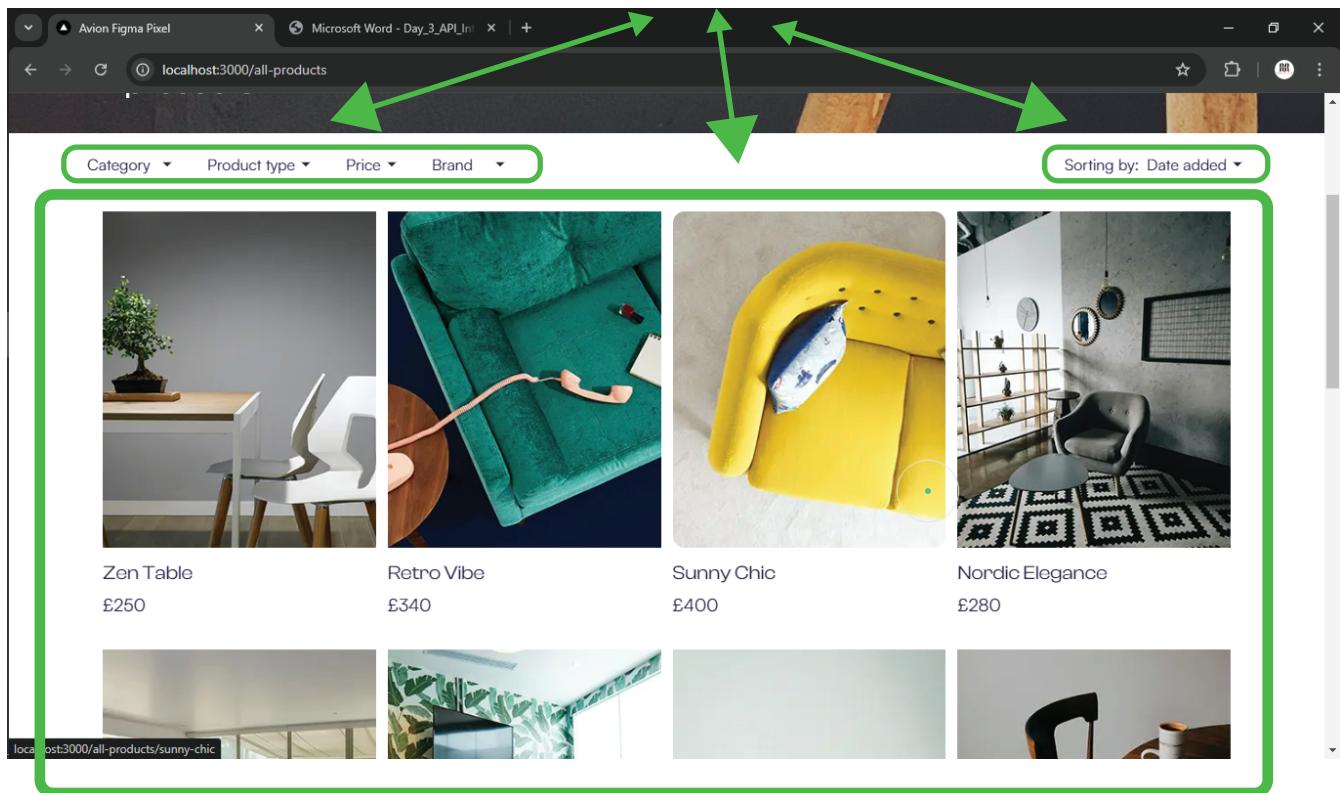
Zubair Ahmed

Saturday - 2 to 5

Day 3 - API Integration Report

E-Commerce - Selling Furniture Website

Product Display



- API integration process.
- Data Migration
- Adjustments made to schemas.
- Fetching Data
- Filtering, Sorting and Dynamic Display,
- Styles and Responsive.
- Testing Done.