API integration process

Prepared by: Alishba Moin
Roll No: 00453212

---

1. API Selection and Setup

- API was provided to import product and category data for the project.
- Integrated the API using the fetch method in Next.js to streamline the data flow.
- Configured dynamic routes for fetching data based on category or slug parameters.
- Example API Endpoint:

```
Qodo Gen: Options | Test this function
async function importData() {
    try {
      // Fetch data from external API
      const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
      const products = response.data;
      //console.log(products)
      let counter=1;
      // Iterate over the products
      for (const product of products) {
        let imageRef = null;
        let catRef=null;

        // Upload image and get asset reference if it exists
        if (product.image) {
          //imageRef = await uploadImageToSanity(product.imageUrl);
          imageRef = await uploadImageToSanity(product.image);
        }

        if(product.category.name){
          catRef = await createCategory(product.category,counter)
        }
```

2. Frontend Integration

- Implemented useEffect and useState hooks to fetch and render data dynamically.

```
// Fetch product data from Sanity
useEffect(() => {
  const getData = async () => {
    if (!params.slug) return; // Handle null slug case
    const query = `*[_type == "product" && slug.current != '${params.slug}'][0]{
      "currentSlug": slug.current,
      name,
      price,
      description,
      dimensions,
      image {
        asset->{
          _id,
          url
        }
      },
      features
    }`;
    const product = await client.fetch(query);
    setData(product);
  };

  getData();
}, [params.slug]);
```

Adjustments Made to Schemas

1. Product Schema Adjustments

- Modified the schema to align with API structure, including fields for category, description, and slug.
- Final Product Schema:

```
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",
            type: "string"
        }),
        defineField({
            name: "slug",
            title: "Slug",
            type: "slug"
        }),
        defineField({
            name: "image",
            type: "image",
```

```
            title: "Product Image"
        }),
        defineField({
            name: "price",
            type: "number",
            title: "Price",
        }),
        defineField({
            name: "quantity",
            title: "Quantity",
            type: "number",
          }),
        defineField({
            name: 'description',
            title: 'Description',
            type: 'text',
            description: 'Detailed description of the product',
          }),
          defineField({
            name: 'features',
            title: 'Features',
            type: 'array',
            of: [{ type: 'string' }],
```

```
            description: 'List of key features of the product',
          }),
          defineField({
            name: 'dimensions',
            title: 'Dimensions',
            type: 'object',
            fields: [
              { name: 'height', title: 'Height', type: 'string' },
              { name: 'width', title: 'Width', type: 'string' },
              { name: 'depth', title: 'Depth', type: 'string' },
            ],
            description: 'Dimensions of the product',
          }),
    ]
})
```

2. Category Schema Adjustments

- Updated the schema to include slug generation for better routing and SEO.
- Final Category Schema:

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



export const Category = defineType({
    name: "category",
```

```
    title: "Category",
    type: "document",
    fields:[
        defineField({
            name: "name",
            title: "Name",
            type: "string",
        }),
        defineField({
            name: "slug",
            title: "Slug",
            type: "slug",
            options: {
                source: "name",
            }
        })
    ]
})
```

Migration Steps and Tools Used

1. Data Migration Steps

1. Received data through the provided API.
2. Exported data from the API in JSON format.
3. Transformed the data to match the new schema structure.
4. Imported the formatted data into Sanity CMS using their CLI tool.

2. Tools Used

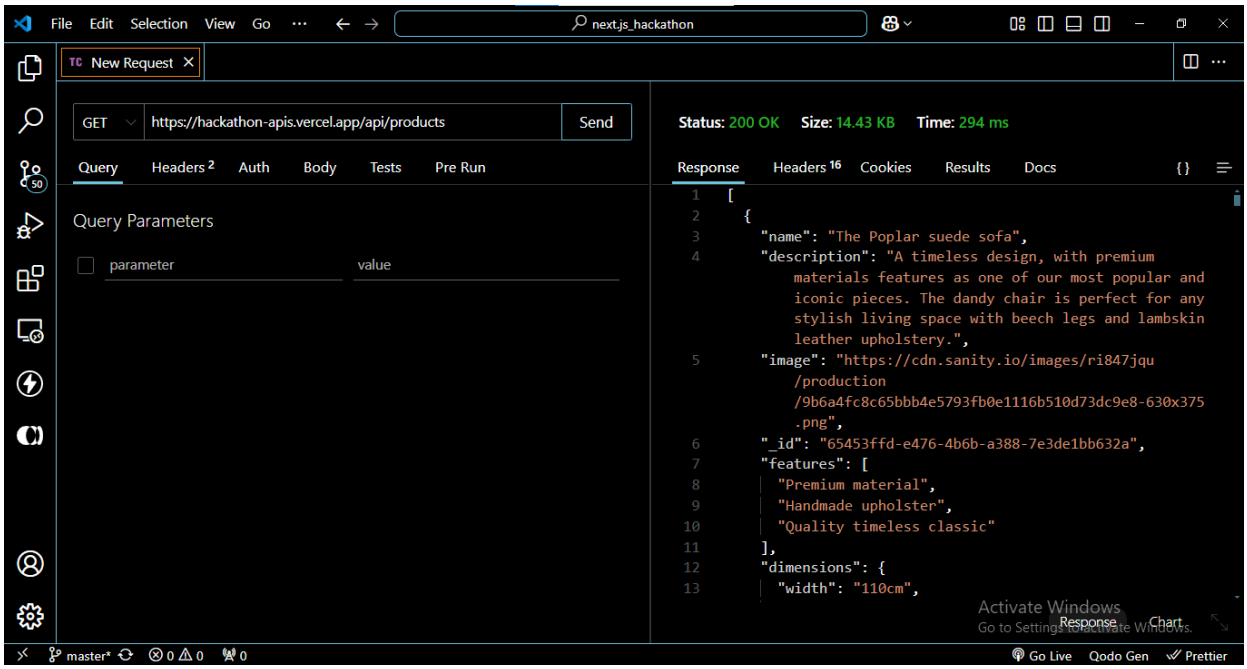
- **CLI**: For importing/exporting data.
- **ThunderClient**: To verify and test API endpoints.
- **Next.js**: For seamless frontend integration.

3. Migration Script Example

sanity dataset import exported-data.json production

---

1. API Calls

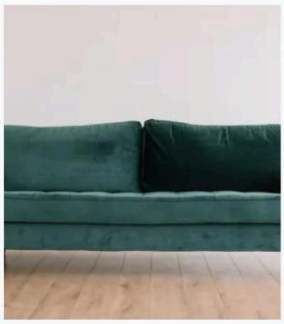


2. Data Displayed on Frontend

Product Listing

localhost:3000/AllProducts

All products



The Poplar suede sofa

$980

Tropical Vibe

$550

Sleek Living

$300

localhost:3000/AllProducts

Timeless Elegance

$320

Modern Serenity

$480

Reflective Haven

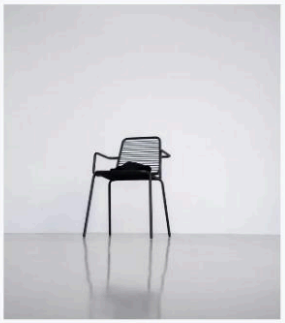
$300

Sunny Chic

$400

Bold Nest

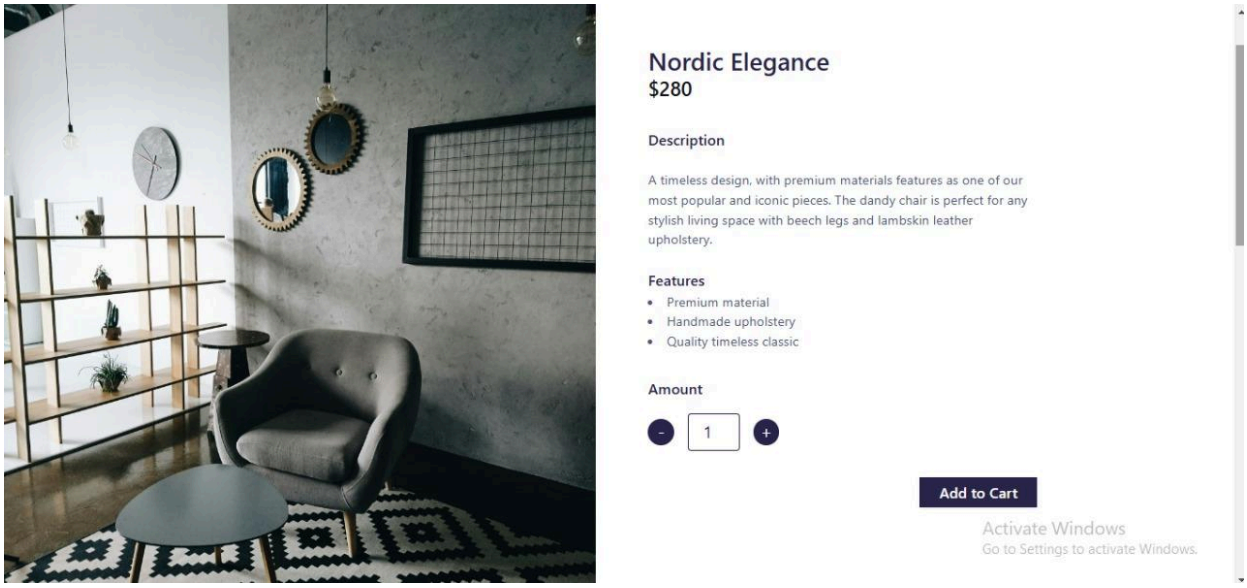
$260



The Steel Chair

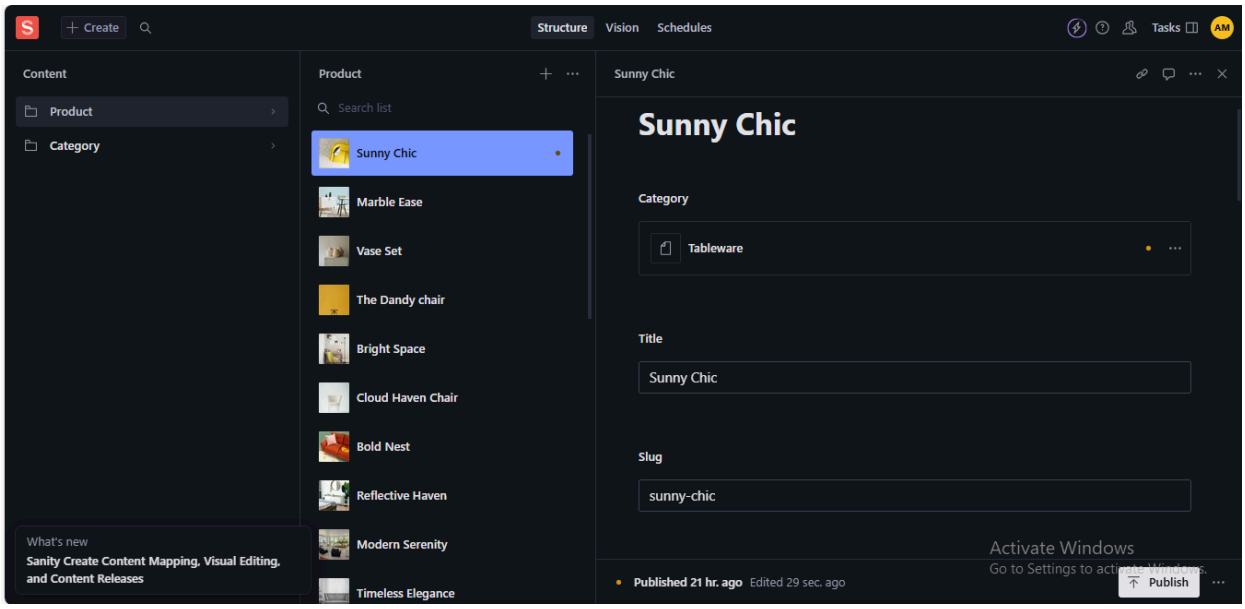
$250

Product Details

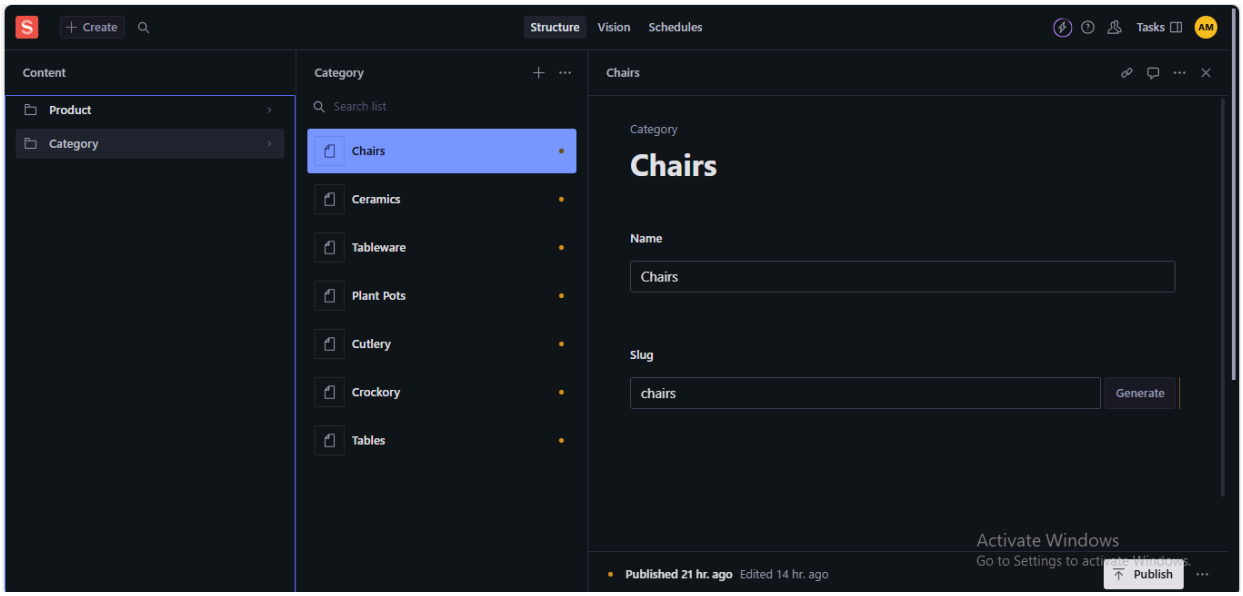


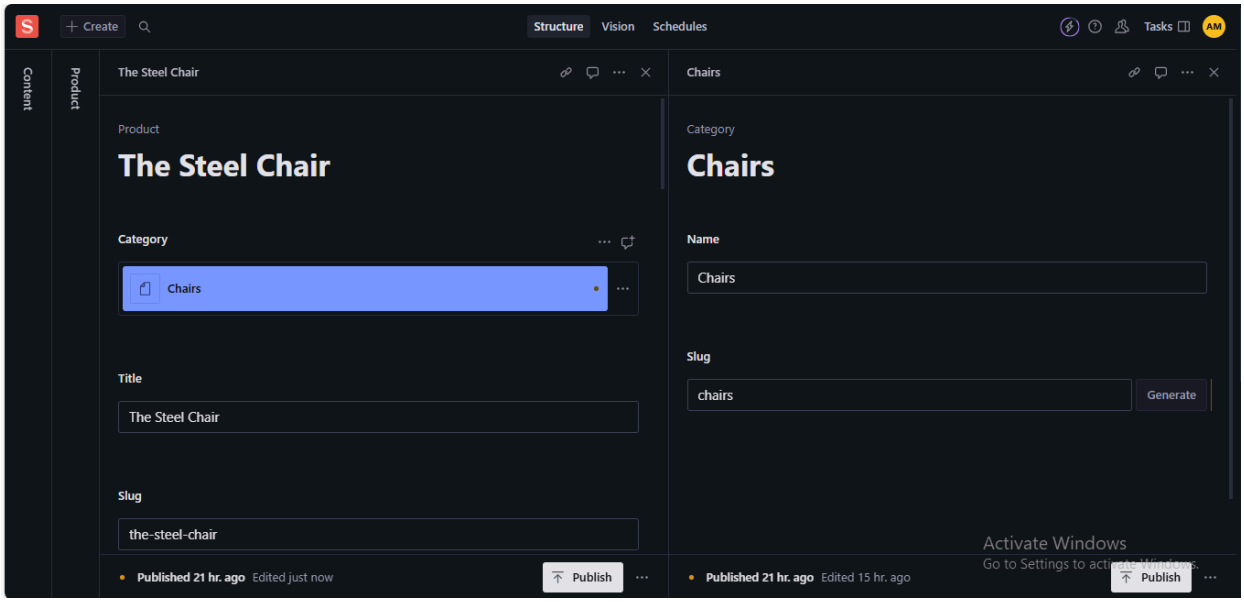
3. Populated Sanity CMS Fields

Products Sanity CMS



Category Sanity CMS





Day 3 Checklist:

Task	Status
API Understanding	✔

Schema Validation	✔
Data Migration	✔
API Integration In Next.js	✔
Submission Preparation	✔