Day 03 HACKATHON DOCUMENTATION: API INTEGRATION AND DATA MIGRATION

API Integration Process

Overview

On Day 3 of the hackathon, we focused on integrating APIs to fetch product data and migrating existing data into our Sanity CMS backend. The process involved adjusting schemas, writing migration scripts, and ensuring the frontend displayed the data correctly.

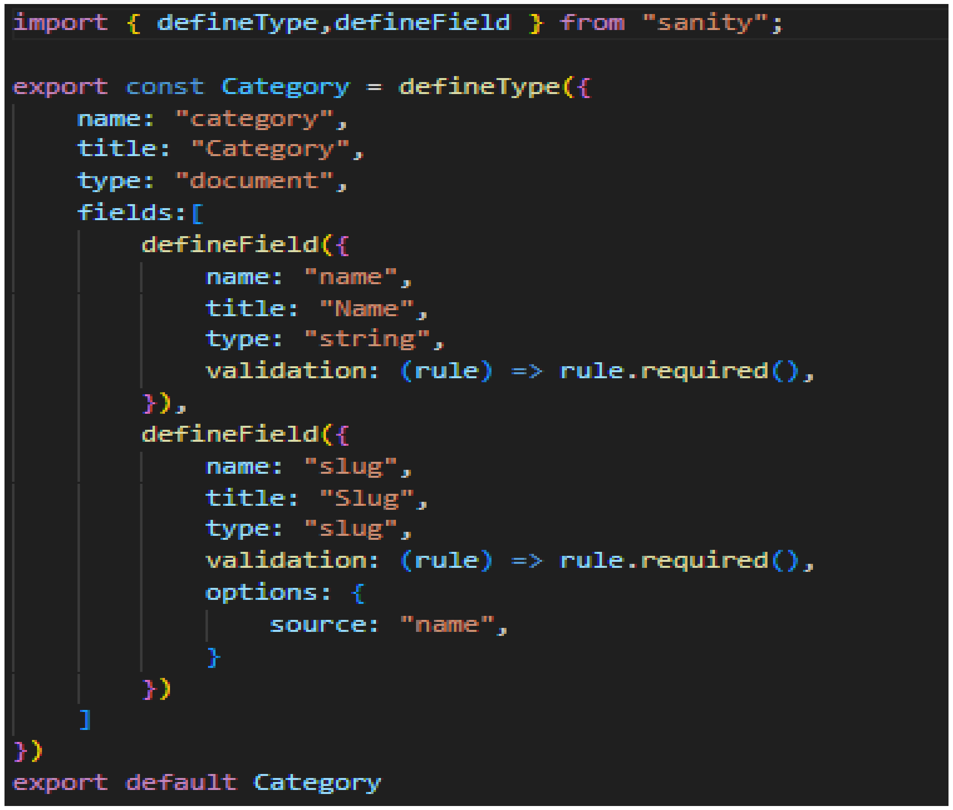
Prepared By Abdullah Kashif

Steps for API Integration

1. Import API Call Logic:
    - Implemented API call logic in the importSanityData.mjs script to fetch and push data to Sanity CMS.
2. Schema Adjustments:
    - Added a new category schema in the schemas/types folder.
    - Updated the product.ts schema to reference the category field.
    - Registered the new schema in the index.ts file within the schemas/types folder.
3. Frontend Integration:
    - Made API calls from the frontend to fetch and display data dynamically.
    - Ensured data binding with UI components built using Tailwind CSS.

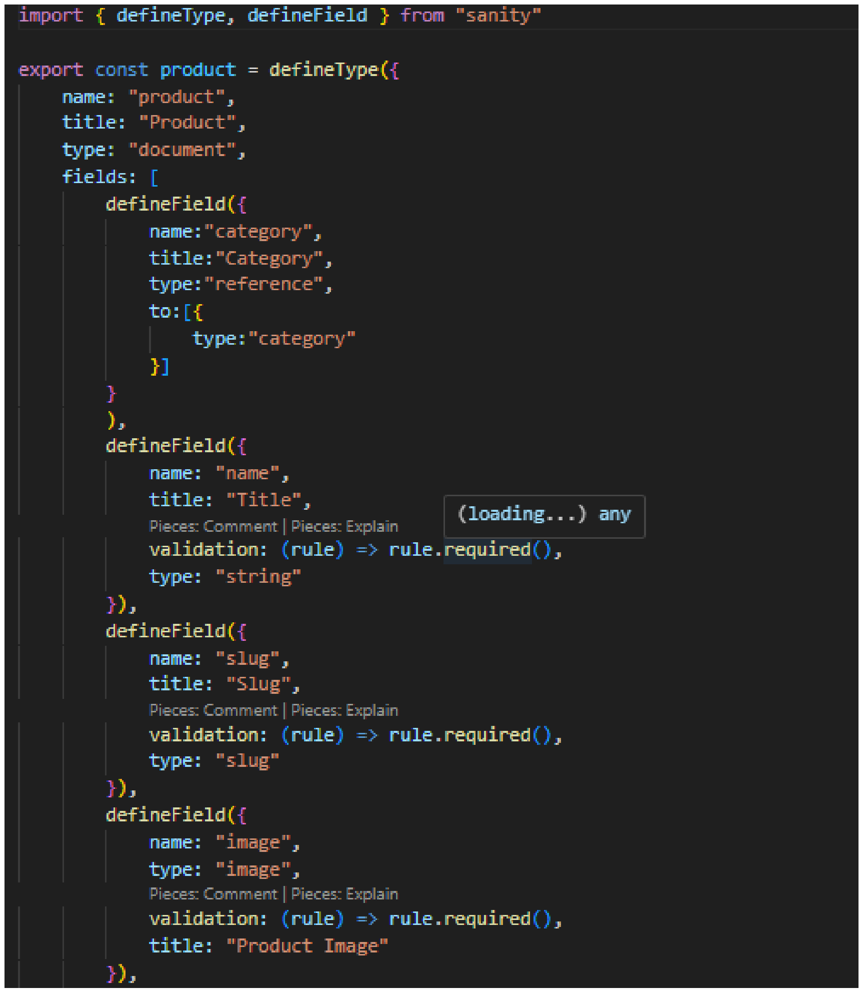
Adjustments Made to Schemas

New Category Schema

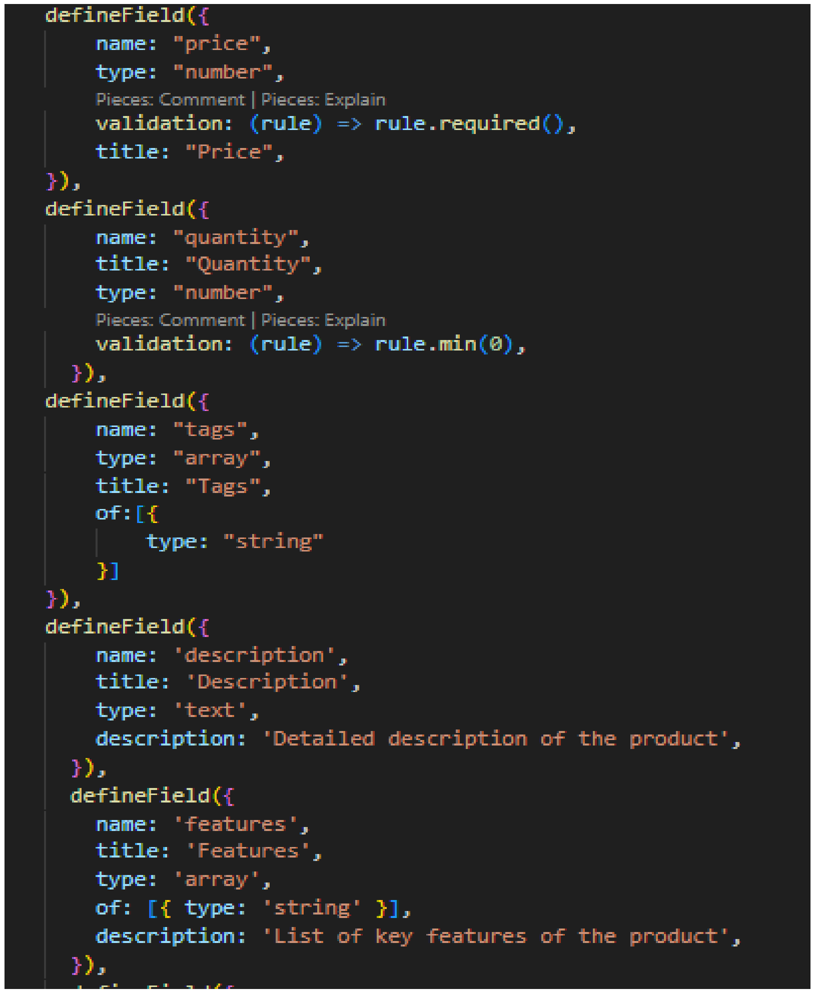


```
import { defineType,defineField } from "sanity";

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

Product Schema Update

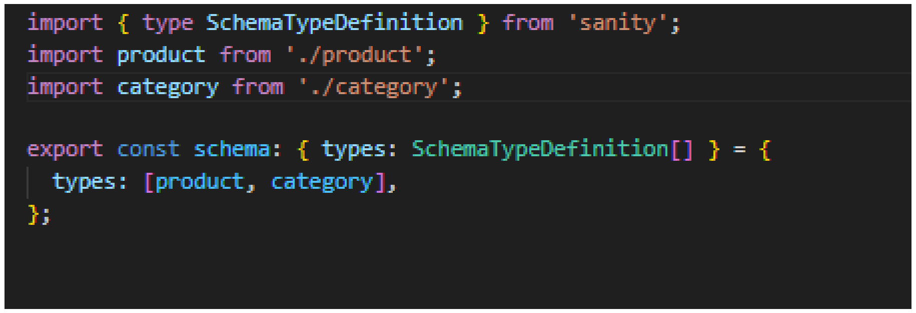


```
import { defineType, defineField } from "sanity"

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",
            Pieces: Comment | Pieces: Explain
            validation: (rule) => rule.required(),
            type: "string"
        }),
        defineField({
            name: "slug",
            title: "Slug",
            Pieces: Comment | Pieces: Explain
            validation: (rule) => rule.required(),
            type: "slug"
        }),
        defineField({
            name: "image",
            type: "image",
            Pieces: Comment | Pieces: Explain
            validation: (rule) => rule.required(),
            title: "Product Image"
        }),
```



```
  defineField({
      name: "price",
      type: "number",
      Pieces: Comment | Pieces: Explain
      validation: (rule) => rule.required(),
      title: "Price",
  }),
  defineField({
      name: "quantity",
      title: "Quantity",
      type: "number",
      Pieces: Comment | Pieces: Explain
      validation: (rule) => rule.min(0),
    }),
  defineField({
      name: "tags",
      type: "array",
      title: "Tags",
      of:[{
          type: "string"
      }]
  }),
  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',
    }),
```

Register Schemas:



```
import { type SchemaTypeDefinition } from 'sanity';
import product from './product';
import category from './category';

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product, category],
};
```

Migration Steps and Tools Used

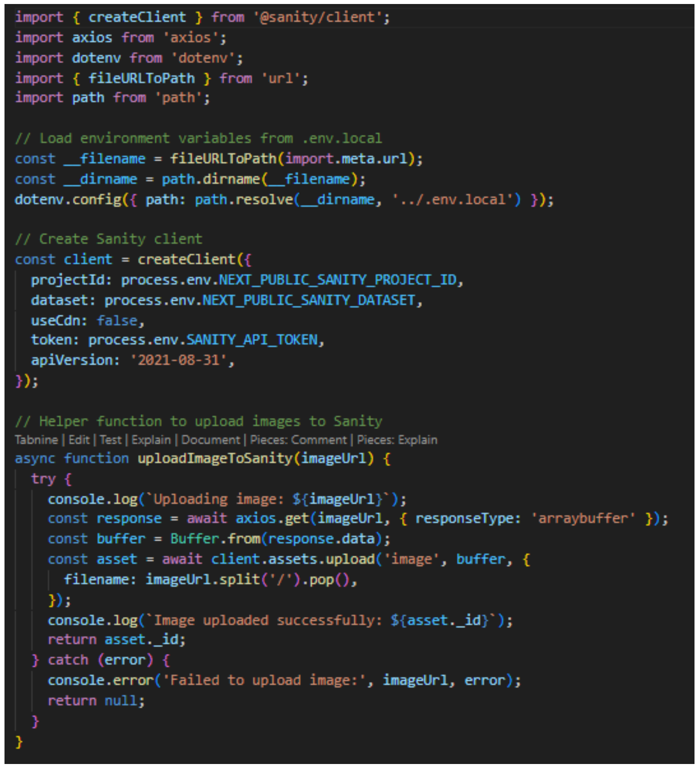
Migration Process

1. Preparation:
   - Created a migration script (importSanityData.mjs) to structure the data.
   - Tested API endpoints to verify data accuracy.
2. Data Migration:
   - Utilized the Sanity CLI tool to push data.
   - Verified data consistency and integrity in Sanity Studio.

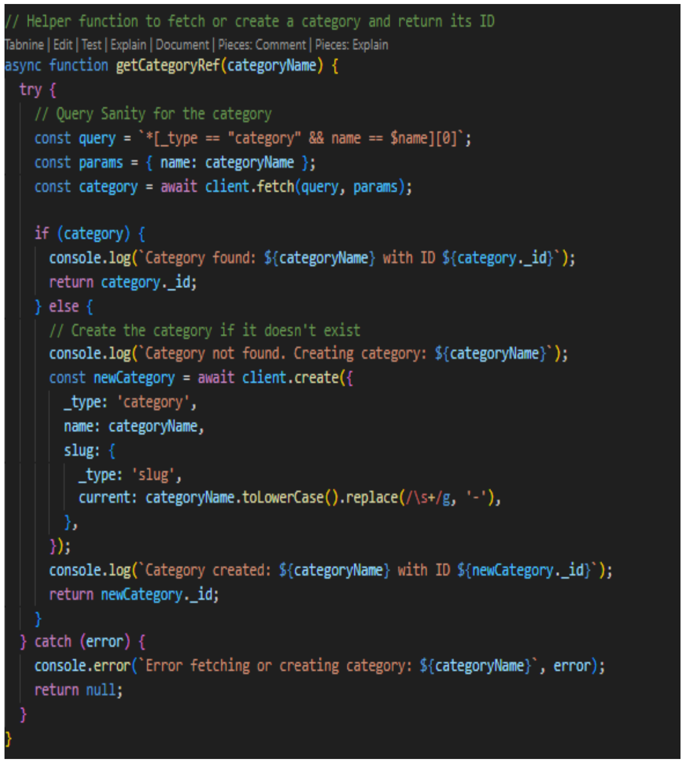
Tools Used

- Sanity CLI: To manage data migration and schema updates.
- Custom Script: Written in Node.js for bulk data migration.

Migration Script



```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31',
});

// Helper function to upload images to Sanity
Tabnine | Edit | Test | Explain | Document | Pieces: Comment | Pieces: Explain
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop(),
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}
```

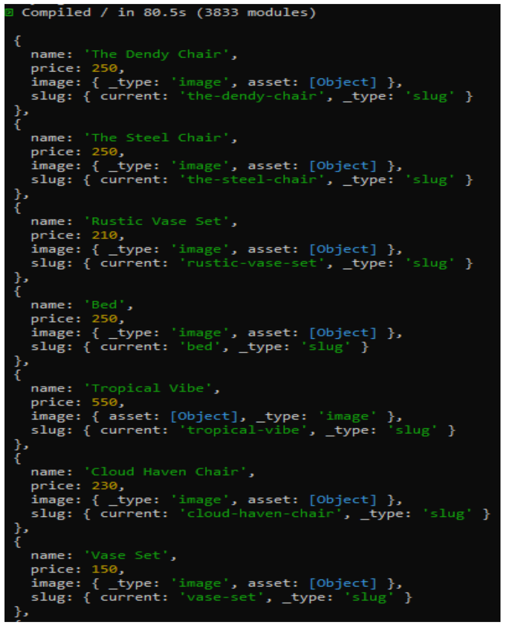


```
// Helper function to fetch or create a category and return its ID
Tabnine | Edit | Test | Explain | Document | Pieces: Comment | Pieces: Explain
async function getCategoryRef(categoryName) {
  try {
    // Query Sanity for the category
    const query = `*[_type == "category" && name == $name][0]`;
    const params = { name: categoryName };
    const category = await client.fetch(query, params);

    if (category) {
      console.log(`Category found: ${categoryName} with ID ${category._id}`);
      return category._id;
    } else {
      // Create the category if it doesn't exist
      console.log(`Category not found. Creating category: ${categoryName}`);
      const newCategory = await client.create({
        _type: 'category',
        name: categoryName,
        slug: {
          _type: 'slug',
          current: categoryName.toLowerCase().replace(/\s+/g, '-'),
        },
      });
      console.log(`Category created: ${categoryName} with ID ${newCategory._id}`);
      return newCategory._id;
    }
  } catch (error) {
    console.error(`Error fetching or creating category: ${categoryName}`, error);
    return null;
  }
}
```

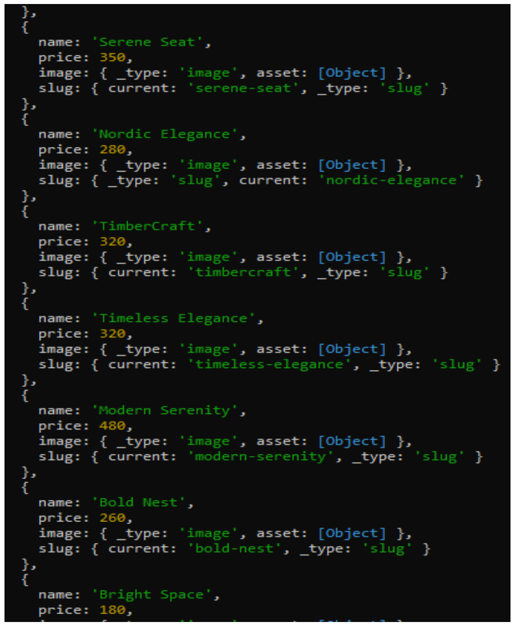
Screenshots

API Calls

- Screenshot showing successful API calls in the terminal/logs.



```
⊡ Compiled / in 80.5s (3833 modules)

{
  name: 'The Dendy Chair',
  price: 250,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'the-dendy-chair', _type: 'slug' }
},
{
  name: 'The Steel Chair',
  price: 250,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'the-steel-chair', _type: 'slug' }
},
{
  name: 'Rustic Vase Set',
  price: 210,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'rustic-vase-set', _type: 'slug' }
},
{
  name: 'Bed',
  price: 250,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'bed', _type: 'slug' }
},
{
  name: 'Tropical Vibe',
  price: 550,
  image: { asset: [Object], _type: 'image' },
  slug: { current: 'tropical-vibe', _type: 'slug' }
},
{
  name: 'Cloud Haven Chair',
  price: 230,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'cloud-haven-chair', _type: 'slug' }
},
{
  name: 'Vase Set',
  price: 150,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'vase-set', _type: 'slug' }
},
```

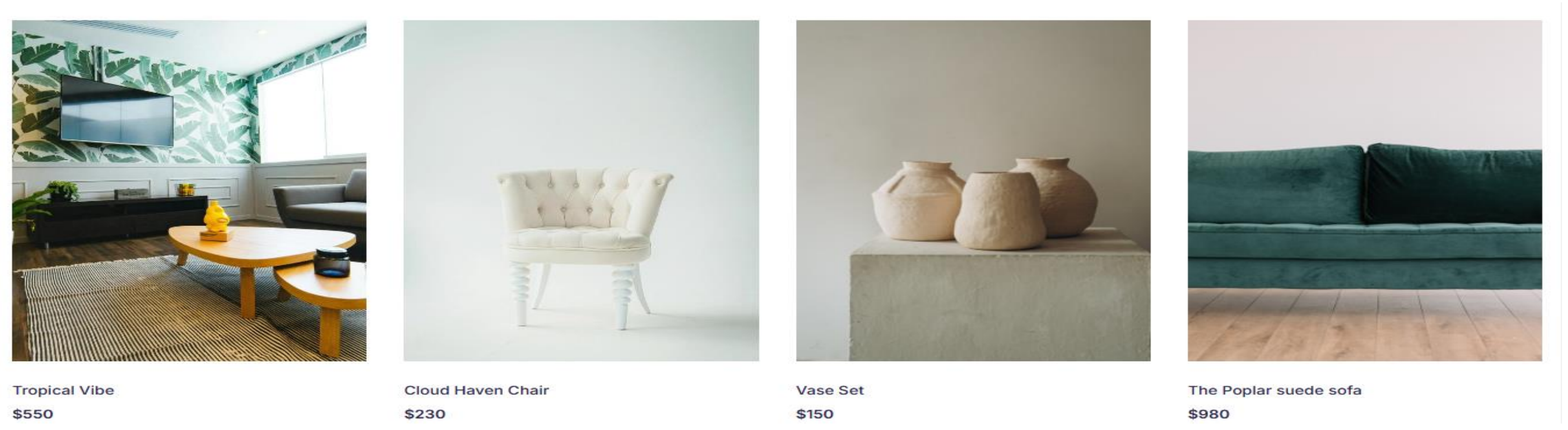


```
},
{
  name: 'Serene Seat',
  price: 350,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'serene-seat', _type: 'slug' }
},
{
  name: 'Nordic Elegance',
  price: 280,
  image: { _type: 'image', asset: [Object] },
  slug: { _type: 'slug', current: 'nordic-elegance' }
},
{
  name: 'TimberCraft',
  price: 320,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'timbercraft', _type: 'slug' }
},
{
  name: 'Timeless Elegance',
  price: 320,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'timeless-elegance', _type: 'slug' }
},
{
  name: 'Modern Serenity',
  price: 480,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'modern-serenity', _type: 'slug' }
},
{
  name: 'Bold Nest',
  price: 260,
  image: { _type: 'image', asset: [Object] },
  slug: { current: 'bold-nest', _type: 'slug' }
},
{
  name: 'Bright Space',
  price: 180,
```

.

Data Displayed on Frontend

- Screenshot demonstrating data successfully fetched and displayed on the frontend.

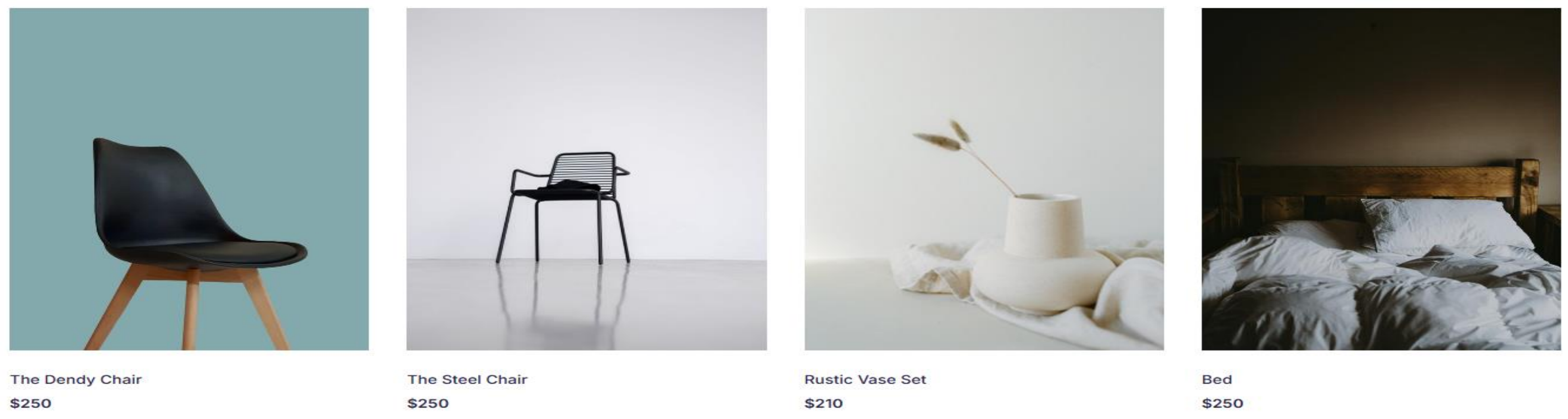


Tropical Vibe
$550

Cloud Haven Chair
$230

Vase Set
$150

The Poplar suede sofa
$980



The Dendy Chair
$250

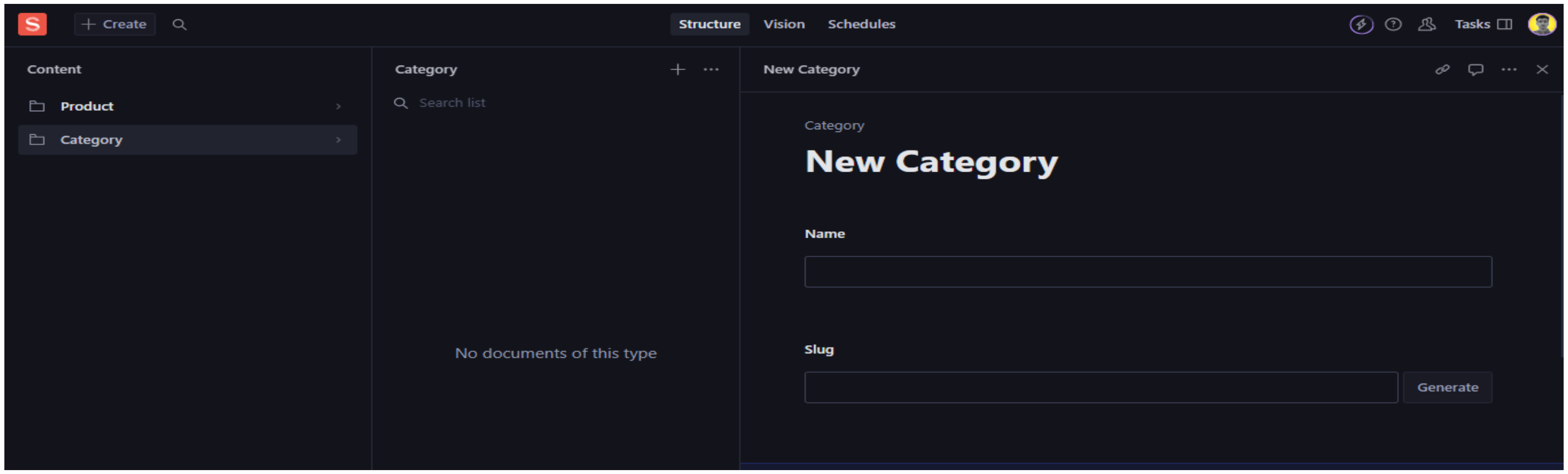
The Steel Chair
$250

Rustic Vase Set
$210

Bed
$250

Sanity Studio Fields

- Screenshot of populated fields in Sanity Studio, including Product and Category documents.



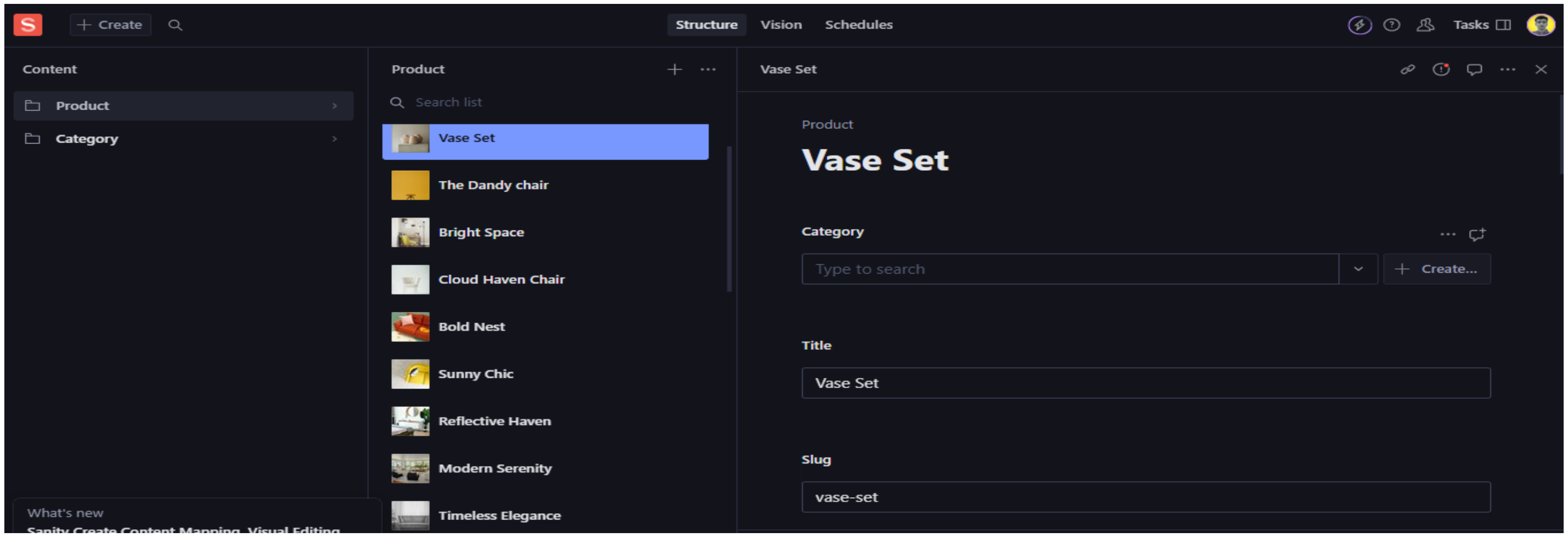
No documents of this type

Category

New Category

Name

Slug



Vase Set
The Dandy chair
Bright Space
Cloud Haven Chair
Bold Nest
Sunny Chic
Reflective Haven
Modern Serenity
Timeless Elegance

Product

Vase Set

Category: Type to search

Title: Vase Set

Slug: vase-set

Conclusion

Day 3 involved integrating APIs, updating schemas, and migrating data into Sanity CMS. The successful implementation ensured that data was displayed seamlessly on the frontend and stored accurately in the backend.

Connect with Me