

DAY 3- API INTEGRATION REPORT- [FURNITURE]

Overview:

This document provides a detailed report of the API integration process for the E-commerce Furniture platform. The integration process was carried out using Sanity, a headless CMS, to streamline content management and improve the dynamic functionality of the website.

Purpose of API Integration:

The primary objectives of this API integration include:

- Enabling dynamic and efficient content updates for the furniture catalog.
- Improving data synchronization between the front-end and back-end systems.
- Ensuring seamless scalability to accommodate a growing inventory.
- Enhancing the user experience through real-time data updates.

Tools and Technologies Used:

- **Sanity CMS:** A headless content management system used as the primary backend for storing and managing furniture catalog data.
- **Next.js:** The front-end framework utilized for rendering dynamic content and pages.
- **Tailwind CSS:** For styling the front end.
- **REST APIs:** Provided by Sanity for data querying and manipulation.
- **React Query:** For efficient data fetching, caching, and synchronization.

API Integration Process:

1. Setting Up Sanity

1. Project Initialization:

- A new project was created in the Sanity dashboard.
- Schemas were defined to represent the structure of furniture data, including fields such as product name, description, category, price, tags, and images.

2. Dataset Configuration:

- Configured a dataset (e.g., `production`) to store all data.
- Enabled public read access for non-sensitive data, ensuring secure data queries.

3. GROQ Queries:

- Utilized Sanity's Graph-Relational Object Queries (GROQ) to fetch data efficiently.

2. Backend API Development

1. API Token Configuration:

- Generated API tokens with specific roles (e.g., read-only) for secure access.
- Stored the tokens securely using environment variables in the Next.js project.

2. Data Fetching Logic:

- Created utility functions to fetch data from Sanity's API endpoints.
- Example of a GROQ query for fetching furniture products:

```
const query = *[_type == "product"] { _id, title,  
  
  price,  
  
  description,  
  
  discountPercentage,  
  
  "imageUrl": productImage.asset->url,  
  
  tags };
```

3. Pagination and Filtering:

- Implemented server-side logic to handle pagination, filtering, and sorting of products.

4. Frontend Integration:

1. Dynamic Content Rendering:

- Integrated data fetched from Sanity into React components.
- Ensured that the data dynamically updated across the platform when changes were made in Sanity.

2. Optimized Image Delivery:

- Leveraged Sanity's built-in Image API for image transformation and optimization.

3. Real-Time Updates:

- Configured the front end to subscribe to real-time updates using Sanity's Webhooks, ensuring data reflects changes instantly.

Migration Steps and Tools Used:

1. Data Export from Legacy System:

- Extracted data from the legacy CMS or database in JSON format.
- Reviewed and cleaned data to ensure consistency and remove duplicates.

2. Data Transformation:

- Used scripts in Node.js to map legacy data to Sanity's schema structure.
- Verified data against the schema to ensure proper formatting and completeness.

3. Data Import into Sanity:

- Utilized Sanity's CLI tool (`sanity dataset import`) to upload the cleaned and transformed data into the configured dataset.
- Validated the imported data by performing sample queries in Sanity's Vision tool.

4. Tooling and Automation:

- Used tools such as Postman for testing API responses during migration.
- Automated repetitive tasks with custom scripts for batch processing and validation.

5. Final Validation:

- Conducted comprehensive testing to ensure the migrated data was accurately displayed on the front end.

Conclusion:

The API integration using Sanity for the E-commerce Furniture platform was successfully implemented. This integration has enhanced the overall functionality and scalability of the platform while providing a seamless user experience. Future improvements may include integrating additional third-party APIs and further optimizing the data delivery pipeline.

SCREENSHOTS:

SETTING UP SANITY

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

E:\Next.js\Hackathon 2\e-commerce-furniture>code .

E:\Next.js\Hackathon 2\e-commerce-furniture>npm create sanity@latest
Need to install the following packages:
create-sanity@3.71.1
Ok to proceed? (y) y
✔ You are logged in as ramshatahir23@gmail.com using Google
✔ Fetching existing projects

? Create a new project or select an existing one Create new project
? Your project name: day3api
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
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'

added 913 packages, and audited 1298 packages in 3m

244 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

added 9 packages, and audited 1307 packages in 17s

244 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Success! Your Sanity configuration files has been added to this project
```

API TOKEN GENERATION

S

Ramsha Tahir

day3api

30 days left in trial

Getting startedOverviewMembersStudiosDatasetsAccessActivityUsagePlanAPISettings

Webhooks

CORS origins

Tokens

Mapping, Visual Editing,

Tokens

Tokens are used to authenticate apps and scripts to access project data.

+ Add API token

Name

Examples: "Employee import", "Website preview" or "PDF generator".

day3api

Permissions

Choose the access privileges for the token.

Contributor

☐ Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)

☐ Deploy Studio (Token only)

Access to deploy Sanity Studio and GraphQL APIs to our hosted service.

Developer

☒ Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)

Editor

☐ Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)

Viewer

☐ Read access to all datasets, with limited access to project settings. (Tokens: read-only)

Save

Cancel

S

Ramsha Tahir

day3api

30 days left in trial

Getting startedOverviewMembersStudiosDatasetsAccessActivityUsagePlanAPISettings

Webhooks

CORS origins

Tokens

Tokens

Tokens are used to authenticate apps and scripts to access project data.

+ Add API token

NAME	PERMISSIONS	CREATED	
day3api	Developer	0 seconds	

Copy the token below – this is your only chance to do so!

skbrSD0jkbhpXD8yzsoHKgwJEaHZVMPcUNKrX00gIXH5TkWdxjMwkoXuW3NhLAoxE14u5R6z7VXDB7qRx1XVSe4YmFaBHQH98gbuEIZupQNj17P2o1Scn1ceG9gE01jzmrrcYVV2DdJXRRodEW9W9JeL1ahgU43RbzWxy5MAaxfuY50opEAr

DOTENV.LOCAL FILE:

```
$ .env.local
1 NEXT_PUBLIC_SANITY_PROJECT_ID="v3gec1xk"
2 NEXT_PUBLIC_SANITY_DATASET="production"
3 SANITY_API_TOKEN="skbRSD0jkbhpXD8yzsoHKgwJEaHZVMpcUNkrX00gIXH5TkW0xjMwkoXuw3NhLAoxE14u5R6z7VXDB7qRx1XVSe4YmFaBHQH98"
4
```

ADJUSTEMENT MADE IN SCHEMA:

```
src > sanity > schemaTypes > TS product.ts > [⌘] product
1 import { defineType } from "sanity"
2
3 export const product = defineType({
4   name: "product",
5   title: "Product",
6   type: "document",
7   fields: [
8     {
9       name: "title",
10      title: "Title",
11      validation: (rule) => rule.required(),
12      type: "string"
13    },
14    {
15      name: "description",
16      type: "text",
17      validation: (rule) => rule.required(),
18      title: "Description",
19    },
20    {
21      name: "productImage",
22      type: "image",
23      validation: (rule) => rule.required(),
24      title: "Product Image"
25    },
26    {
27      name: "price",
28      type: "number",
29      validation: (rule) => rule.required(),
30      title: "Price",
31    },
32    {
33      name: "tags",
34      type: "array",
35      title: "Tags",
36      of: [{ type: "string" }]
37    },
38    {
39      name: "dicountPercentage",
```

importData.js File:

```
script > JS importData.js > uploadProduct
1  import { createClient } from '@sanity/client';
2
3  const client = createClient({
4    projectId: 'v3gec1xk',
5    dataset: 'production',
6    useCdn: true,
7    apiVersion: '2025-01-13',
8    token: 'skbR5D0jkbhpXD8yzsoHKgwJEaHZVMPcUNkrXO0gIXH5TkW0xjMwkoXuW3NhLAoxE14u5R6z7VXDB7qRx1XVSe4YmFaBHQH98gbuEIZupQNj17F
9  });
10
11  async function uploadImageToSanity(imageUrl) {
12    try {
13      console.log(`Uploading image: ${imageUrl}`);
14
15      const response = await fetch(imageUrl);
16      if (!response.ok) {
17        throw new Error(`Failed to fetch image: ${imageUrl}`);
18      }
19
20      const buffer = await response.arrayBuffer();
21      const bufferImage = Buffer.from(buffer);
22
23      const asset = await client.assets.upload('image', bufferImage, {
24        filename: imageUrl.split('/').pop(),
25      });
26
27      console.log(`Image uploaded successfully: ${asset._id}`);
28      return asset._id;
29    } catch (error) {
30      console.error('Failed to upload image:', imageUrl, error);
31      return null;
32    }
33  }
34
35  async function uploadProduct(product) {
36    try {
37      const imageId = await uploadImageToSanity(product.imageUrl);
```

package.json file:

```
{ } package.json > { } scripts > import-data
1  {
2    "name": "e-commerce-furniture",
3    "version": "0.1.0",
4    "private": true,
5    "type": "module",
6    "scripts": {
7      "dev": "next dev",
8      "build": "next build",
9      "start": "next start",
10     "lint": "next lint",
11     "import-data": "node script/importData.js"
12   },
13   "dependencies": {
```

PRODUCT FETCHING IN TERMINAL:

```
C:\Windows\System32\cmd.exe

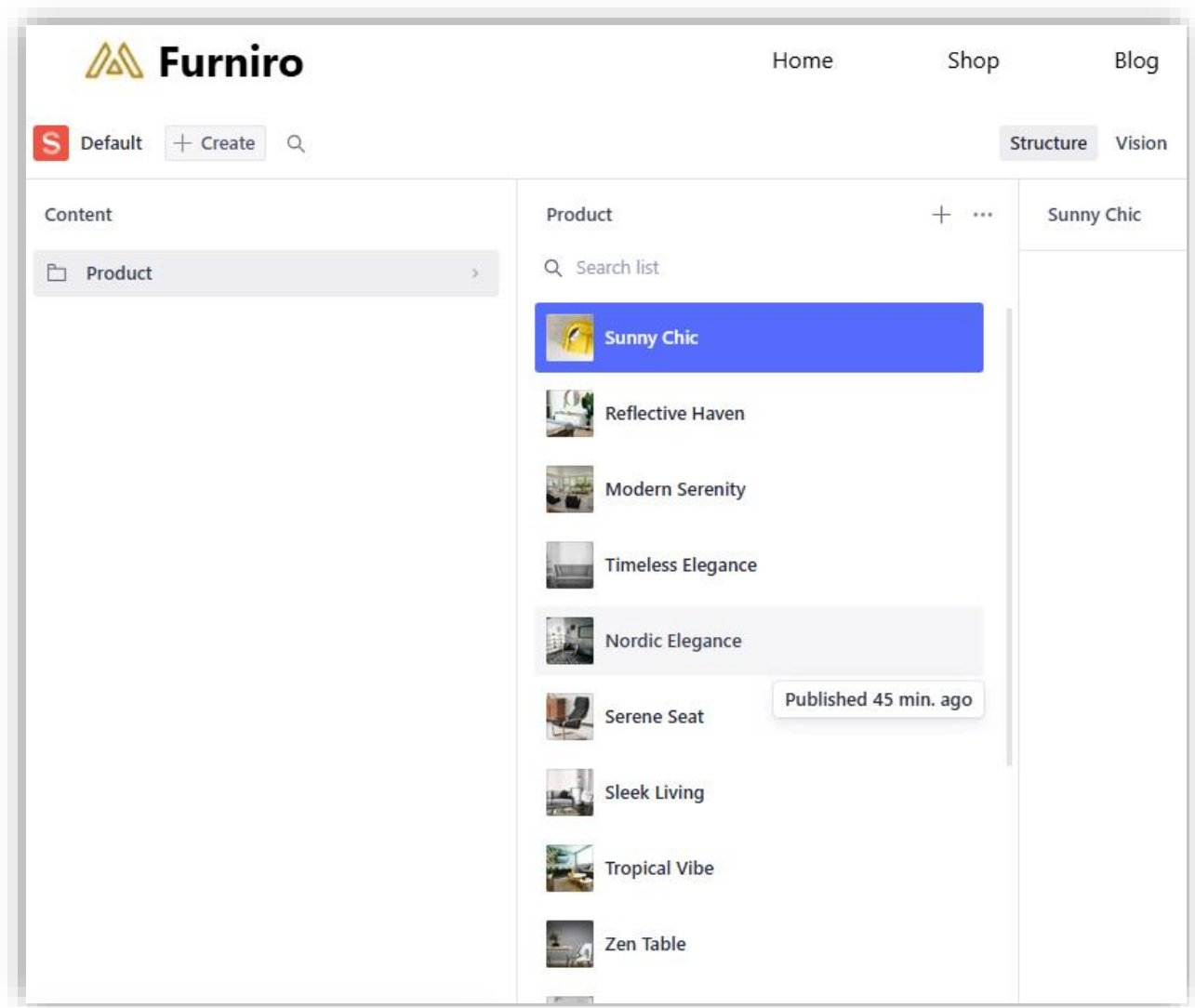
E:\Next.js\Hackathon 2\e-commerce-furniture>npm run import-data

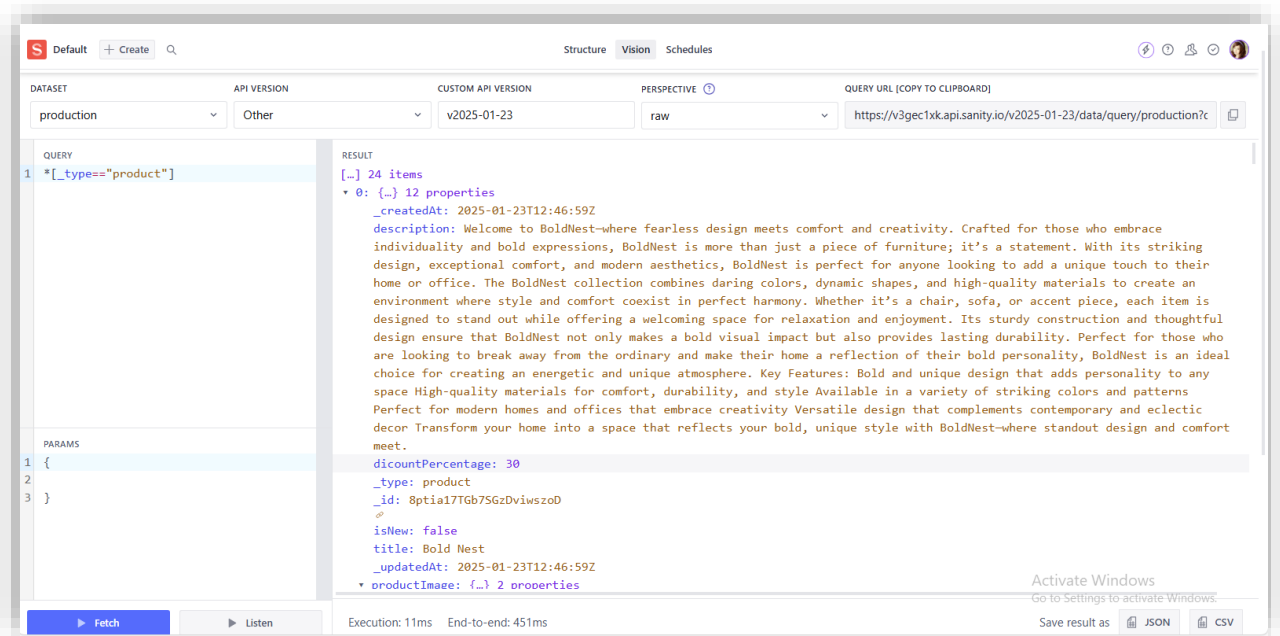
e-commerce-furniture@0.1.0 import-data
node script/importData.js

Uploading image: https://cdn.sanity.io/images/7xt4qcah/production/2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375.png
Image uploaded successfully: image-2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375-png
Product Rustic Vase Set uploaded successfully: {
  _createdAt: '2025-01-23T12:46:39Z',
  _id: 'BsnR1UxX7Cp0BctRefz6f0',
  _rev: 'BsnR1UxX7Cp0BctRefz6f0',
  _type: 'product',
  _updatedAt: '2025-01-23T12:46:39Z',
  description: 'Bring the charm of nature into your home with the Rustic Vase Set. Perfect for those who appreciate timeless beauty and a warm, inviting atmosphere, this set of vases adds a touch of rustic elegance to any space. Crafted with care and attention to detail, these vases are designed to evoke the essence of vintage craftsmanship while seamlessly complementing both modern and traditional decor styles.',
  title: 'The Rustic Vase Set features a collection of three uniquely designed vases, each with its own character. Their earthy tones, textured finishes, and artisanal touch capture the essence of the countryside, making them ideal for showcasing fresh flowers, dried arrangements, or simply as stand-alone decor pieces. Whether placed on a mantel, coffee table, or dining area, these vases effortlessly enhance the ambiance of your home.',
  keyFeatures: [
    'Made from high-quality materials, the Rustic Vase Set offers both style and durability. The natural, imperfect surfaces of the vases give them a distinct, hand-crafted appeal, ensuring that each set is one-of-a-kind. With their timeless design, these vases make a perfect gift for housewarmings, weddings, or any special occasion.',
    'Key Features:',
    'Set includes three uniquely designed rustic vases',
    'Crafted from high-quality materials with a natural, hand-crafted finish',
    'Perfect for displaying flowers, greenery, or as standalone decorative pieces',
    'Versatile design complements both modern and traditional interiors',
    'Ideal for gifting or personal use in any living space',
    'Add warmth and character to your home with the Rustic Vase Set-where classic design meets natural beauty.'
  ],
  discountPercentage: 10,
  isNew: false,
  price: 210,
  productImage: {
    _type: 'image',
    asset: {
      _ref: 'image-2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375-png'
    }
  },
  tags: [ 'rustic ', 'vase ', 'home decor', 'vintage ', 'interior design' ],
  title: 'Rustic Vase Set'
}
Uploading image: https://cdn.sanity.io/images/7xt4qcah/production/cbf4d42e1e3a58e25763a1d84efe10a8c25aed7-3264x4928.jpg
Image uploaded successfully: image-35a8072e40a8b80142ff8e96e0f3af21811f5b9f-3264x4928-jpg
Product Timber Craft uploaded successfully: {
  _createdAt: '2025-01-23T12:46:43Z',
  _id: 'BsnR1UxX7Cp0BctRefz6gm',
  _rev: 'BsnR1UxX7Cp0BctRefz6gm',
  _type: 'product',
```

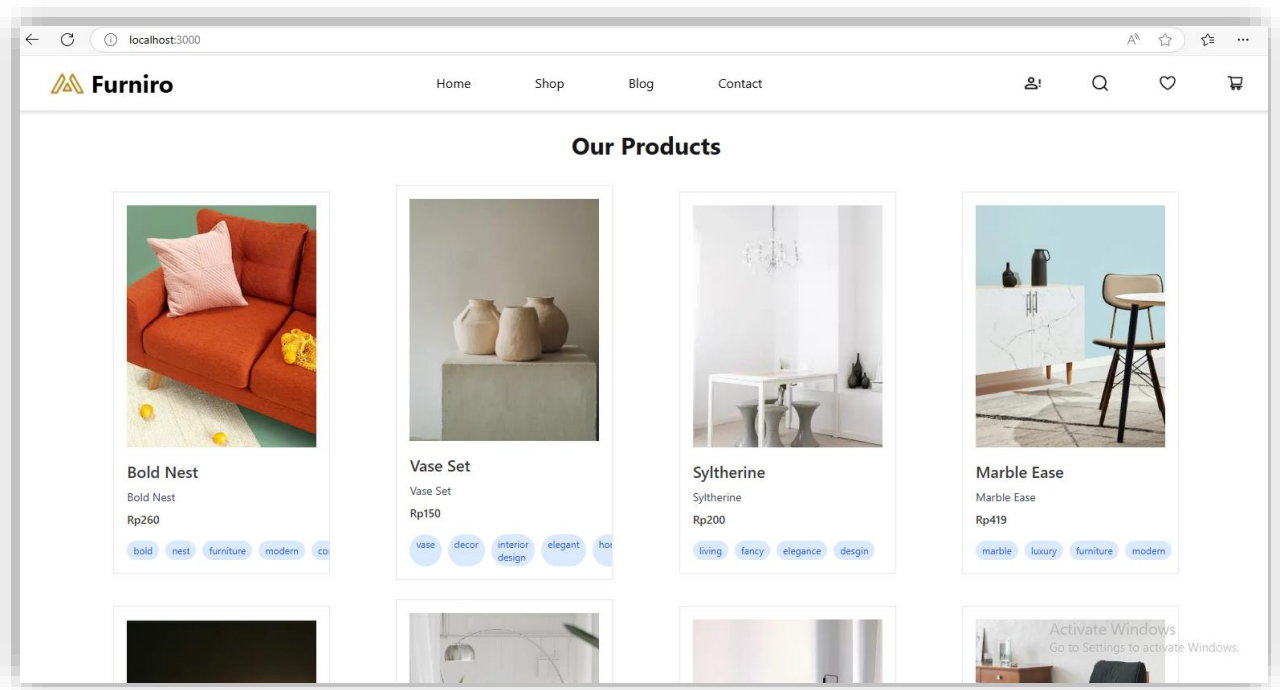
Activate Windows
Go to Settings to activate Windows.

POPULATED SANITY FIELD:





FRONTEND DISPLAY OF DATA:



SELF VALIDATION CHECKLIST:

API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next.js	✓
Submission Preparation	✓