SUBMITTED BY: ASMA ABDUL QADIR
ROLL NO :230664

DAY-3:

API INTEGRATION & DATA MIGRATION REPORT IN GENERAL E-COMMERCE(HEKTO) WEBSITE:

PURPOSE:

The primary objective of this project is to develop a robust, scalable, and user-friendly e-commerce furniture website that offers seamless product browsing, efficient content management, and secure transaction handling. By integrating APIs through Sanity and leveraging modern web technologies like Next.js, TypeScript, and Tailwind CSS, the website aims to provide an optimized shopping experience for customers while ensuring easy content updates for administrators.

Project Overview

The project leverages Sanity as a headless CMS for managing and importing APIs, enabling efficient content management and seamless data flow.

Technologies Used:

Next.js: Framework for server-side rendering and static site generation.

TypeScript: Ensures type safety and improved code maintainability.

Tailwind CSS: Utility-first CSS framework for responsive and customizable UI design.

Sanity: Headless CMS for structured content management and API integration.

Core Features:

Dynamic Product Listings: Real-time updates of product information including images, descriptions, prices, and stock status.

Category Management: Efficient organization of products into categories for easy navigation.

Blog Integration: Dynamic blog posts managed through Sanity to enhance SEO and customer engagement.

User Management: Secure user registration, authentication, and profile management.

Order and Shipment Tracking: Comprehensive tracking of customer orders and shipment statuses.

Responsive Design: Fully responsive UI powered by Tailwind CSS for optimal viewing on all devices.

Real-time Content Updates: Instant content updates through Sanity webhooks and live query features.

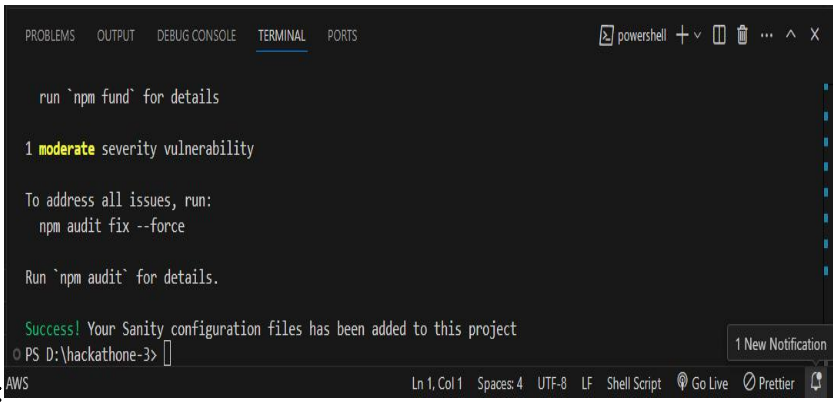
Secure API Access: Controlled API interactions with token-based authentication.

API Integration with Sanity:

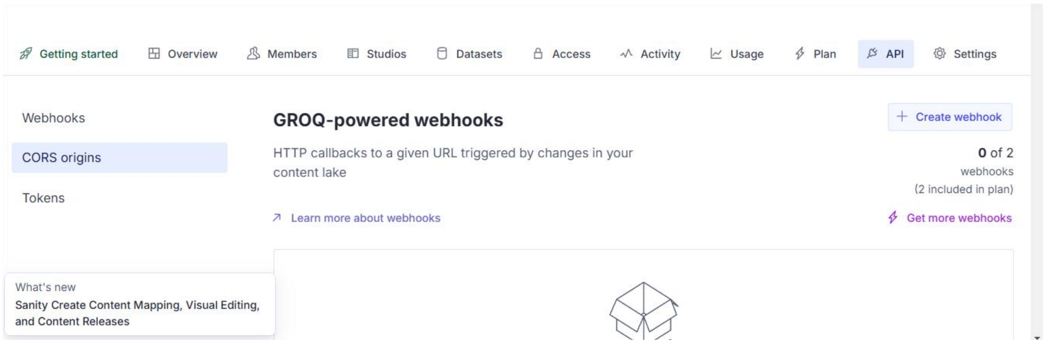
Sanity Setup

Created a Sanity project and defined schemas for products, categories, blogs, users, orders, and shipments.

The overview of the steps are here

:

```
  run `npm fund` for details

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
PS D:\hackathone-3> 
```

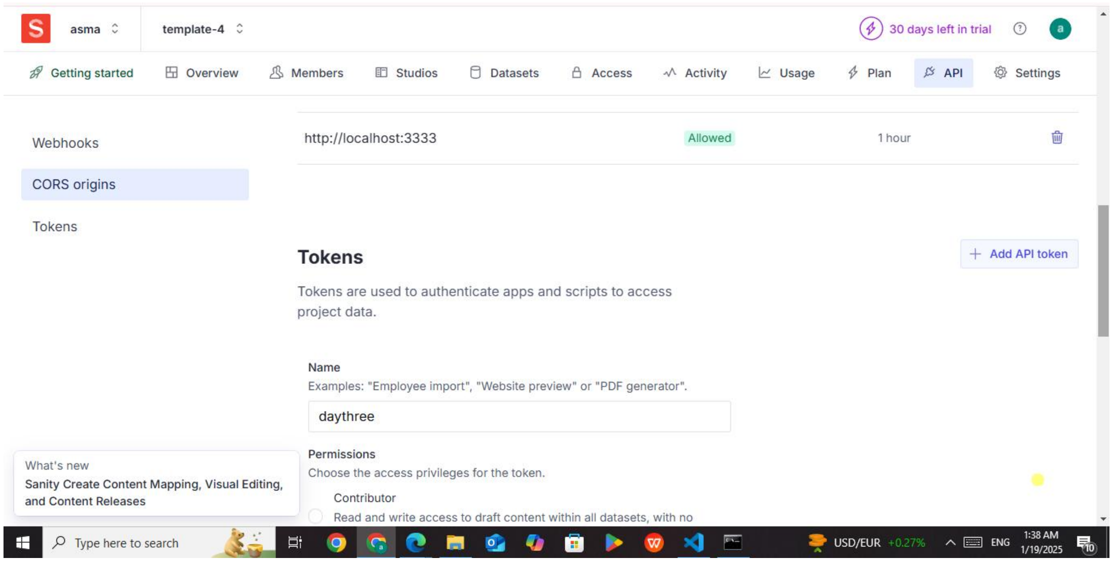


GROQ-powered webhooks

HTTP callbacks to a given URL triggered by changes in your content lake

0 of 2 webhooks (2 included in plan)

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases



http://localhost:3333	Allowed	1 hour

Tokens

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

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

Permissions
Choose the access privileges for the token.
Contributor
Read and write access to draft content within all datasets, with no

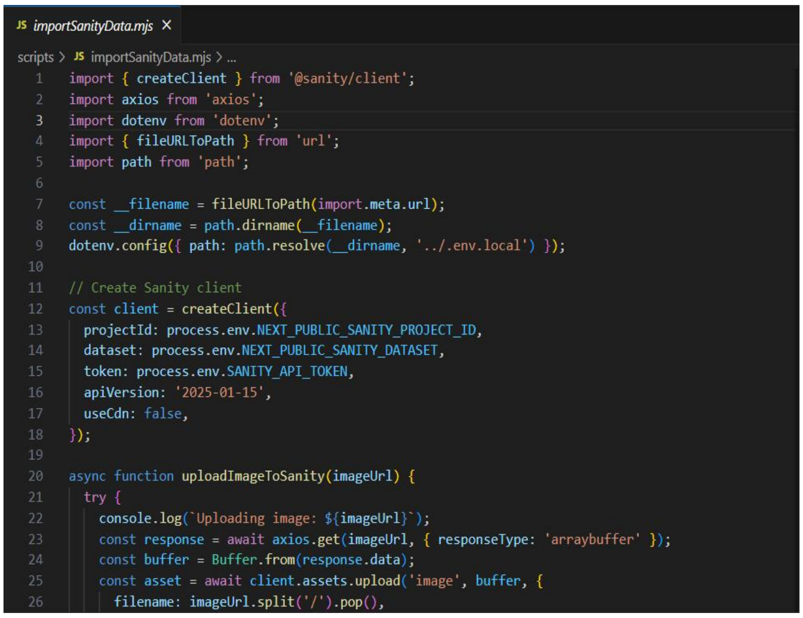
What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases

Imported Data into Sanity CMS

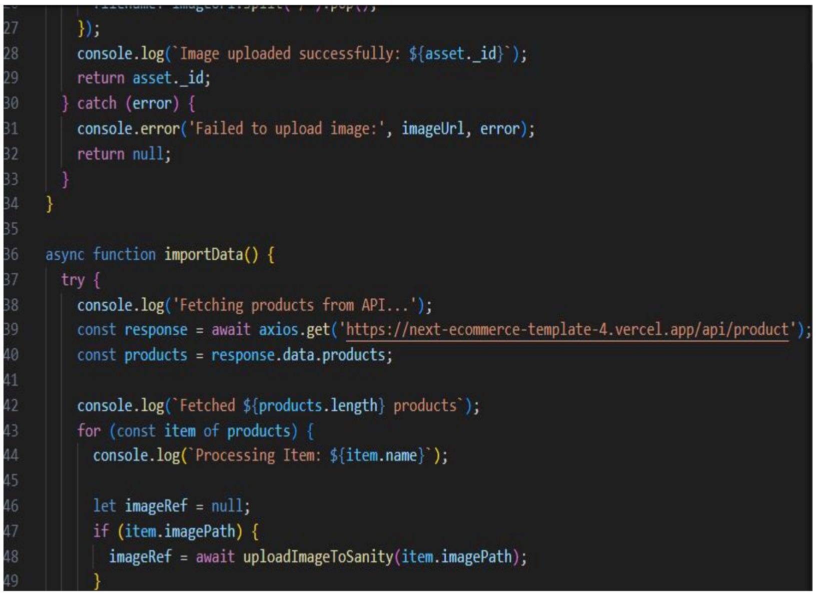
To populate Sanity CMS with API data, I used the provided data migration script. This script fetched data from the external API and imported it into Sanity CMS using its APIs.

Script Workflow

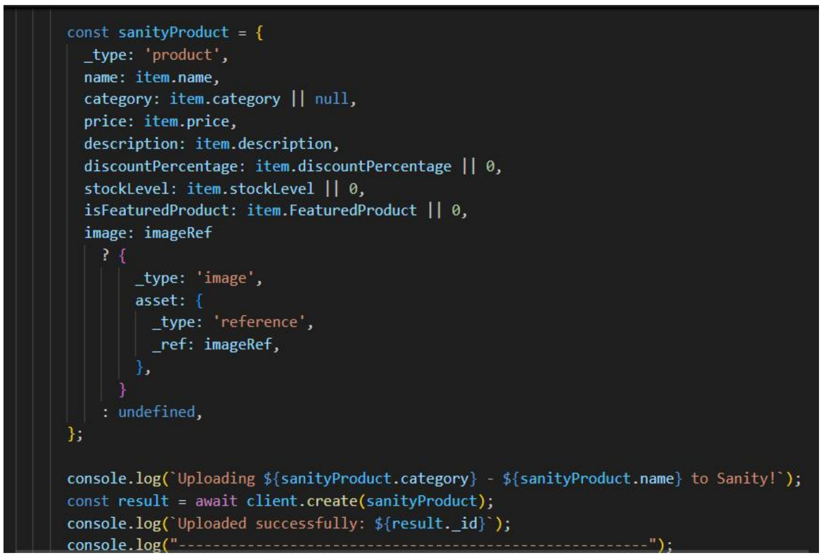
1. Fetch data from the external API.
2. Map the API fields to the Sanity schema field.
3. Push the transformed data into Sanity CMS.
4. During Import:
5. ● Ensured that API field names (like product_title) matched Sanity schema fields (name).
6. ● Validated data before pushing it into Sanity CMS.



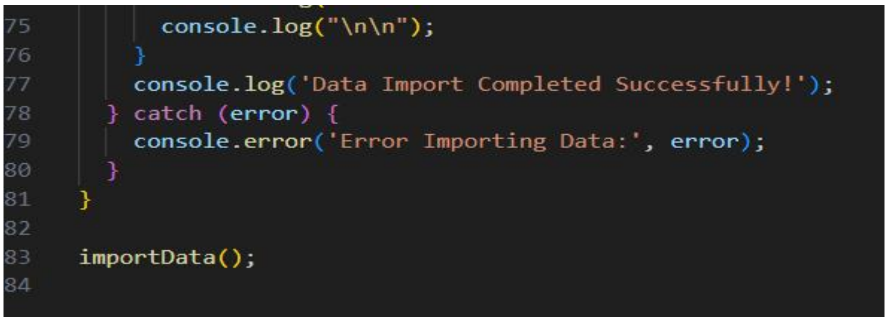
```js
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

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,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2025-01-15',
  useCdn: false,
});

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(),
```



```js
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function importData() {
  try {
    console.log('Fetching products from API...');
    const response = await axios.get('https://next-ecommerce-template-4.vercel.app/api/product');
    const products = response.data.products;

    console.log(`Fetched ${products.length} products`);
    for (const item of products) {
      console.log(`Processing Item: ${item.name}`);

      let imageRef = null;
      if (item.imagePath) {
        imageRef = await uploadImageToSanity(item.imagePath);
      }
```



```
const sanityProduct = {
  _type: 'product',
  name: item.name,
  category: item.category || null,
  price: item.price,
  description: item.description,
  discountPercentage: item.discountPercentage || 0,
  stockLevel: item.stockLevel || 0,
  isFeaturedProduct: item.FeaturedProduct || 0,
  image: imageRef
    ? {
        _type: 'image',
        asset: {
          _type: 'reference',
          _ref: imageRef,
        },
      }
    : undefined,
};

console.log(`Uploading ${sanityProduct.category} - ${sanityProduct.name} to Sanity!`);
const result = await client.create(sanityProduct);
console.log(`Uploaded successfully: ${result._id}`);
console.log("----------------------------------------------------");
```



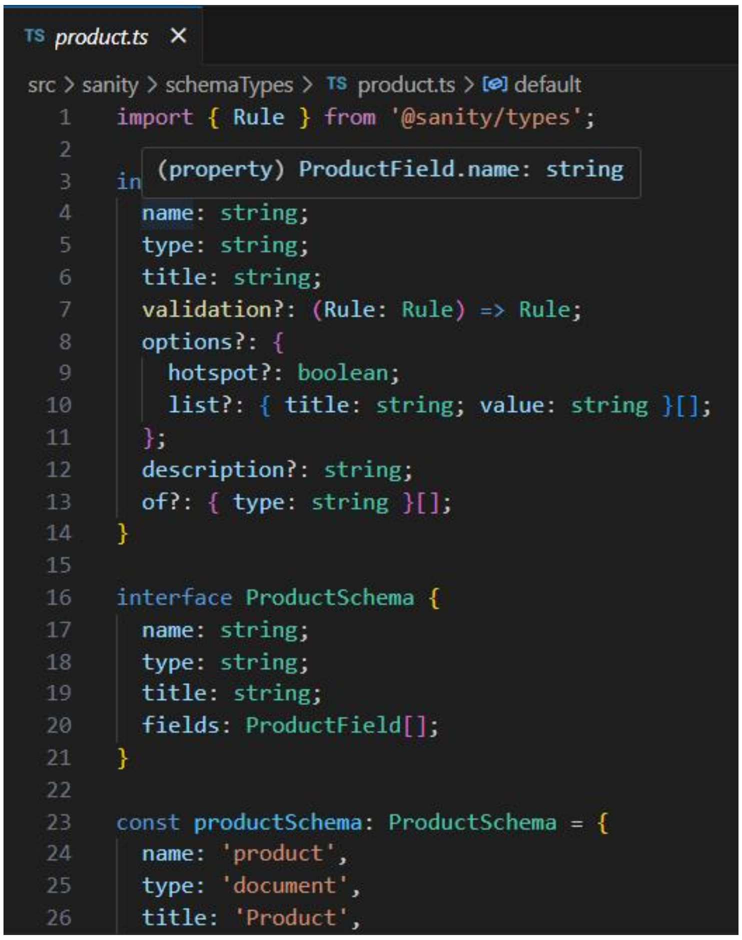
```
      console.log("\n\n");
    }
    console.log('Data Import Completed Successfully!');
  } catch (error) {
    console.error('Error Importing Data:', error);
  }
}

importData();
```

2. Updated the Sanity CMS Schema

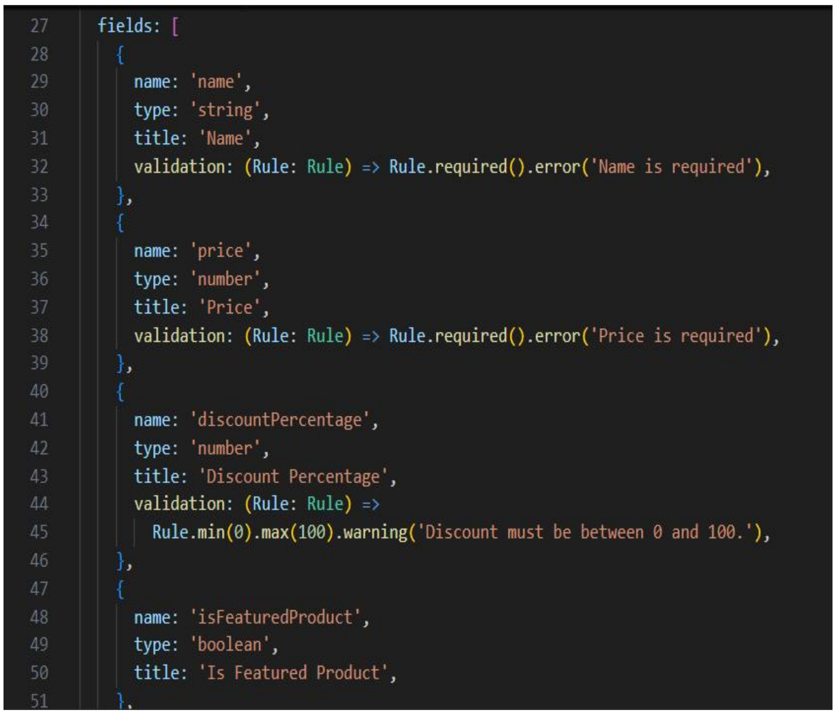
I made significant updates to the Product Schema in Sanity CMS to enhance its functionality and adaptability.

Key Changes I Made:

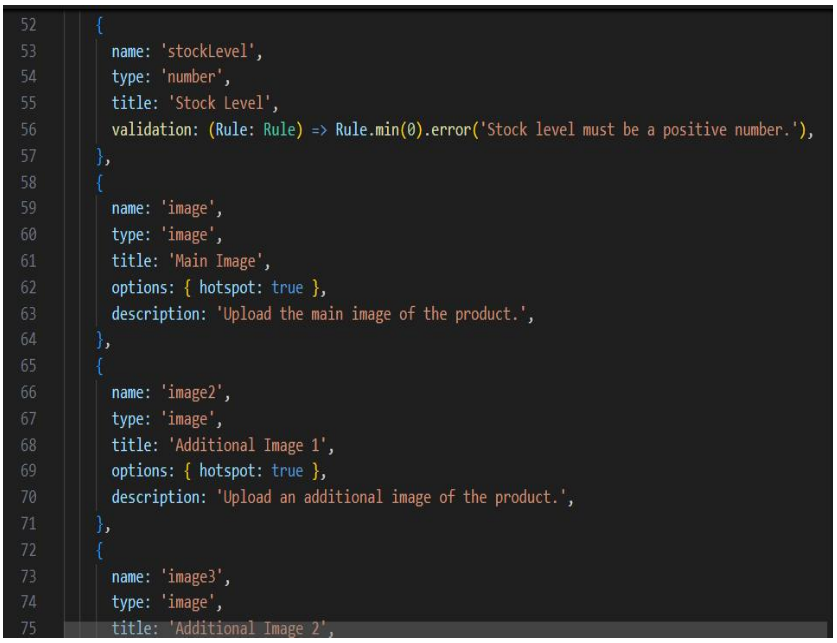
● Introduced discountPercentage, isFeaturedProduct, tags, and color for added flexibilit ● Replaced static category options with dynamic references to a Category Schema.



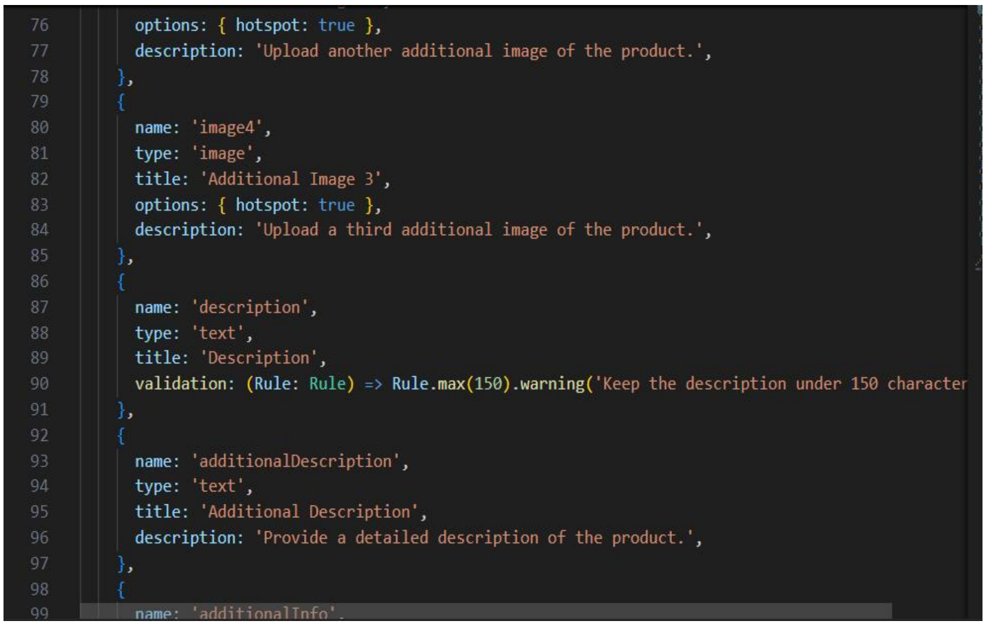
```
import { Rule } from '@sanity/types';

in (property) ProductField.name: string
  name: string;
  type: string;
  title: string;
  validation?: (Rule: Rule) => Rule;
  options?: {
    hotspot?: boolean;
    list?: { title: string; value: string }[];
  };
  description?: string;
  of?: { type: string }[];
}

interface ProductSchema {
  name: string;
  type: string;
  title: string;
  fields: ProductField[];
}

const productSchema: ProductSchema = {
  name: 'product',
  type: 'document',
  title: 'Product',
```



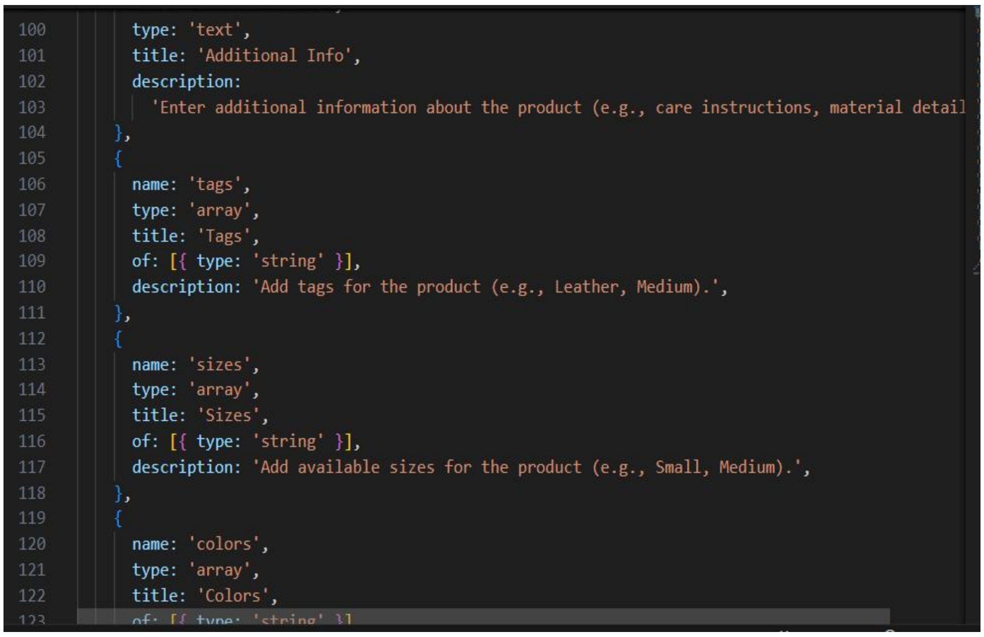
```
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Name',
      validation: (Rule: Rule) => Rule.required().error('Name is required'),
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price',
      validation: (Rule: Rule) => Rule.required().error('Price is required'),
    },
    {
      name: 'discountPercentage',
      type: 'number',
      title: 'Discount Percentage',
      validation: (Rule: Rule) =>
        Rule.min(0).max(100).warning('Discount must be between 0 and 100.'),
    },
    {
      name: 'isFeaturedProduct',
      type: 'boolean',
      title: 'Is Featured Product',
    },
```



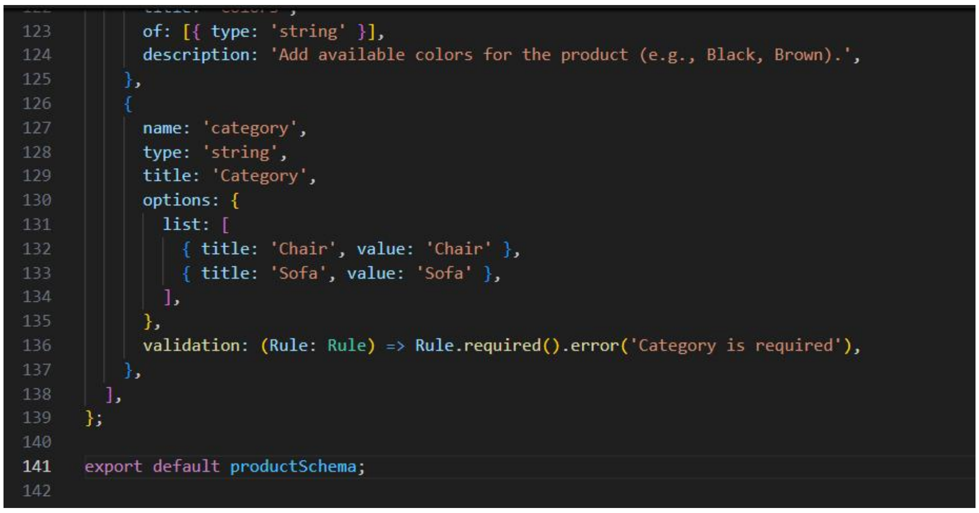
```
    {
      name: 'stockLevel',
      type: 'number',
      title: 'Stock Level',
      validation: (Rule: Rule) => Rule.min(0).error('Stock level must be a positive number.'),
    },
    {
      name: 'image',
      type: 'image',
      title: 'Main Image',
      options: { hotspot: true },
      description: 'Upload the main image of the product.',
    },
    {
      name: 'image2',
      type: 'image',
      title: 'Additional Image 1',
      options: { hotspot: true },
      description: 'Upload an additional image of the product.',
    },
    {
      name: 'image3',
      type: 'image',
      title: 'Additional Image 2',
```



```
      options: { hotspot: true },
      description: 'Upload another additional image of the product.',
    },
    {
      name: 'image4',
      type: 'image',
      title: 'Additional Image 3',
      options: { hotspot: true },
      description: 'Upload a third additional image of the product.',
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
      validation: (Rule: Rule) => Rule.max(150).warning('Keep the description under 150 character
    },
    {
      name: 'additionalDescription',
      type: 'text',
      title: 'Additional Description',
      description: 'Provide a detailed description of the product.',
    },
    {
      name: 'additionalInfo',
```



```
      type: 'text',
      title: 'Additional Info',
      description:
        'Enter additional information about the product (e.g., care instructions, material detail
    },
    {
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }],
      description: 'Add tags for the product (e.g., Leather, Medium).',
    },
    {
      name: 'sizes',
      type: 'array',
      title: 'Sizes',
      of: [{ type: 'string' }],
      description: 'Add available sizes for the product (e.g., Small, Medium).',
    },
    {
      name: 'colors',
      type: 'array',
      title: 'Colors',
```



```
      of: [{ type: 'string' }],
      description: 'Add available colors for the product (e.g., Black, Brown).',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      options: {
        list: [
          { title: 'Chair', value: 'Chair' },
          { title: 'Sofa', value: 'Sofa' },
        ],
      },
      validation: (Rule: Rule) => Rule.required().error('Category is required'),
    },
  ],
};

export default productSchema;
```

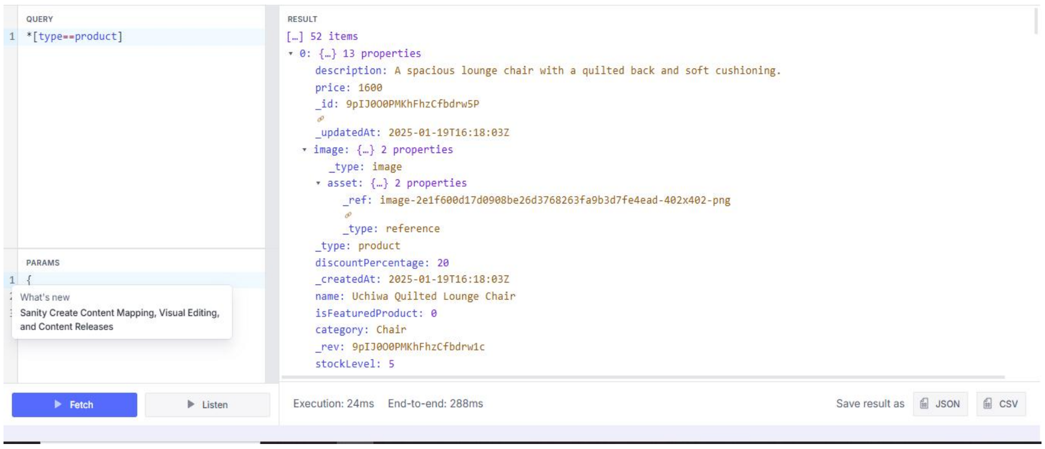
Key Benefits of the Integration

Scalability: Easy to add new content types and expand the API without major code changes.

Efficiency: Centralized content management with Sanity reduces manual data handling.

Flexibility: Strong integration with Next.js enables both static and dynamic rendering options.

Security: API access is controlled via tokens and permissions in Sanity.

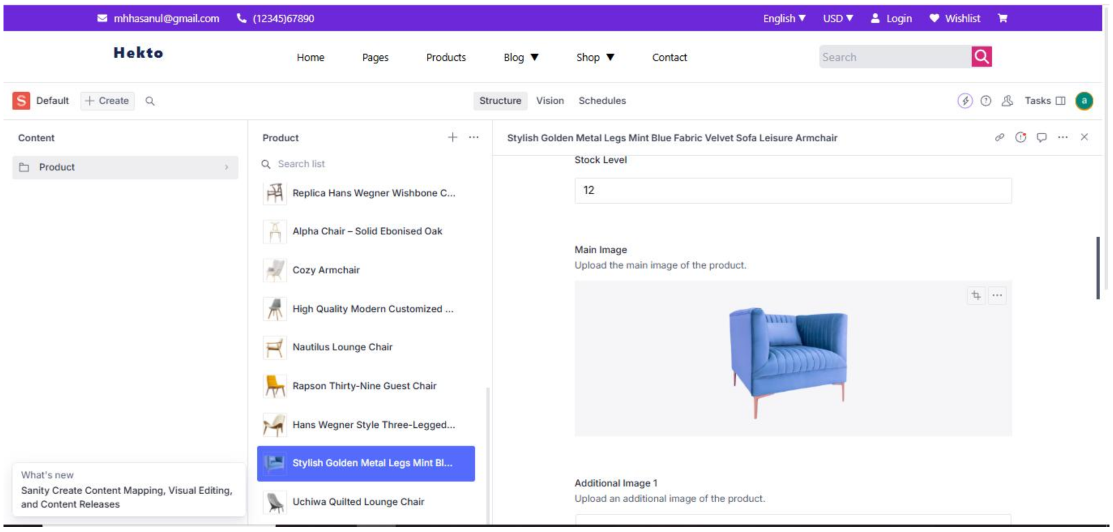


```
*[type==product]
```

```
{
```

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases
RESULT

```
[…] 52 items
▾ 0: {…} 13 properties
    description: A spacious lounge chair with a quilted back and soft cushioning.
    price: 1600
    _id: 9pIJ0O0PMKhFhzCfbdrw5P
    _updatedAt: 2025-01-19T16:18:03Z
  ▾ image: {…} 2 properties
      _type: image
    ▾ asset: {…} 2 properties
        _ref: image-2e1f600d17d0908be26d3768263fa9b3d7fe4ead-402x402-png
        _type: reference
    _type: product
    discountPercentage: 20
    _createdAt: 2025-01-19T16:18:03Z
    name: Uchiwa Quilted Lounge Chair
    isFeaturedProduct: 0
    category: Chair
    _rev: 9pIJ0O0PMKhFhzCfbdrw1c
    stockLevel: 5
```



Product

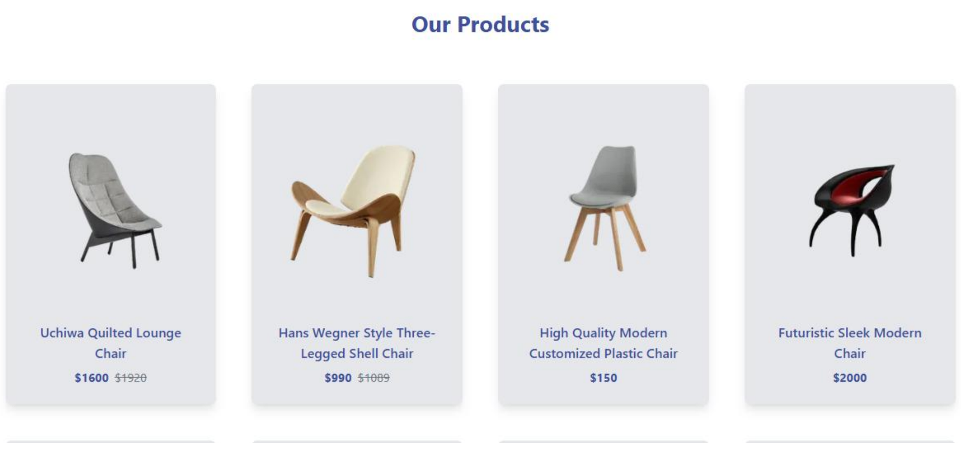
- Replica Hans Wegner Wishbone C...
- Alpha Chair – Solid Ebonised Oak
- Cozy Armchair
- High Quality Modern Customized ...
- Nautilus Lounge Chair
- Rapson Thirty-Nine Guest Chair
- Hans Wegner Style Three-Legged...
- Stylish Golden Metal Legs Mint Bl...
- Uchiwa Quilted Lounge Chair

Stylish Golden Metal Legs Mint Blue Fabric Velvet Sofa Leisure Armchair

Stock Level: 12

Main Image
Upload the main image of the product.

Additional Image 1
Upload an additional image of the product.


Our Products

Uchiwa Quilted Lounge Chair	Hans Wegner Style Three-Legged Shell Chair	High Quality Modern Customized Plastic Chair	Futuristic Sleek Modern Chair
$1600 ~~$1920~~	$990 ~~$1089~~	$150	$2000

Conclusion

The integration of Sanity APIs with Next.js, TypeScript, and Tailwind CSS has significantly enhanced the e-commerce furniture website's performance, maintainability, and user experience. The API-driven architecture ensures that the platform is robust, flexible, and ready for future scalability.