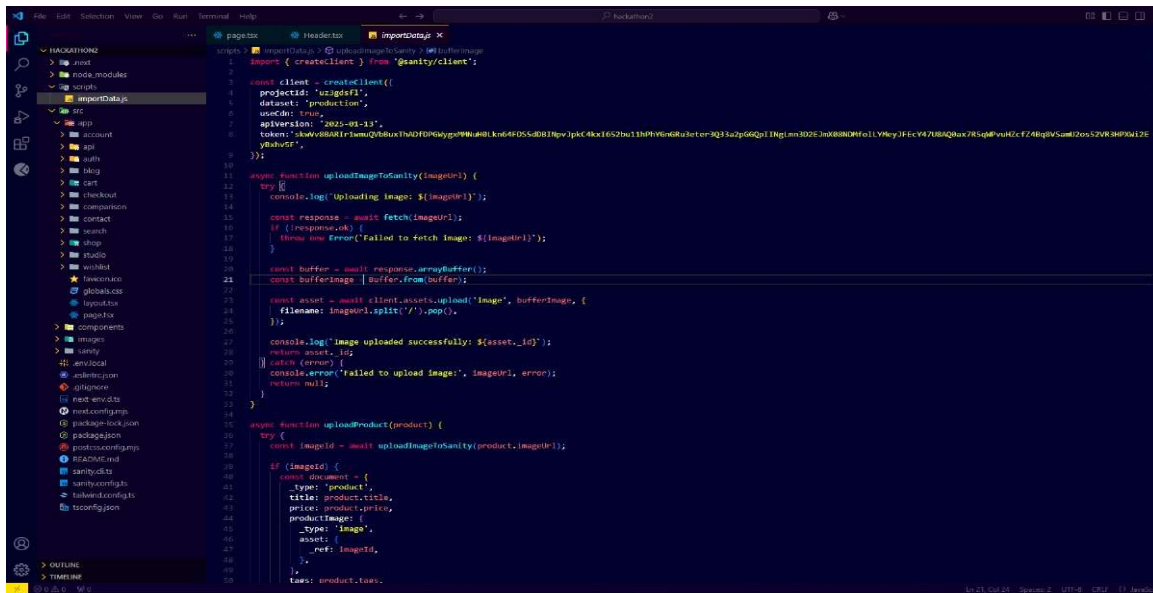


DAY 3 API INTEGRATION AND DATA MIGRATION

Objective:

- Integrate APIs within a Next.js application efficiently.
- Transfer data from external APIs to Sanity CMS seamlessly.
- Validate and synchronize schemas with data sources for consistency.
- Understand API integration strategies and their practical implementation.
 - Learn the process of migrating data from APIs to a CMS.
 - Customize and validate Sanity CMS schemas for smooth data compatibility.

Migration Script:



```
import { createClient } from 'sanity/client';

const client = createClient({
  projectId: 'uzigsfl',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2023-01-13',
  token: 'shove8801rlm0qVd0ux7h0JTD0N0y0p9W0k064FDS5d0H1Np0pC4kx1G52bu11hP0v0d0u0e0r9Q1ka2p0G0p11Hgm0D253n0B0M0M0e11Y0eY3FicY47U0AQ0ux7R5p0vut0EcF24Hq0VSam0Zos12VR0M0M12E0h0u0C0F',
});

export async function uploadImageToSanity(imageUrl) {
  try {
    console.log('Uploading image: ${imageUrl}');

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error('Failed to fetch image: ${imageUrl}');
    }

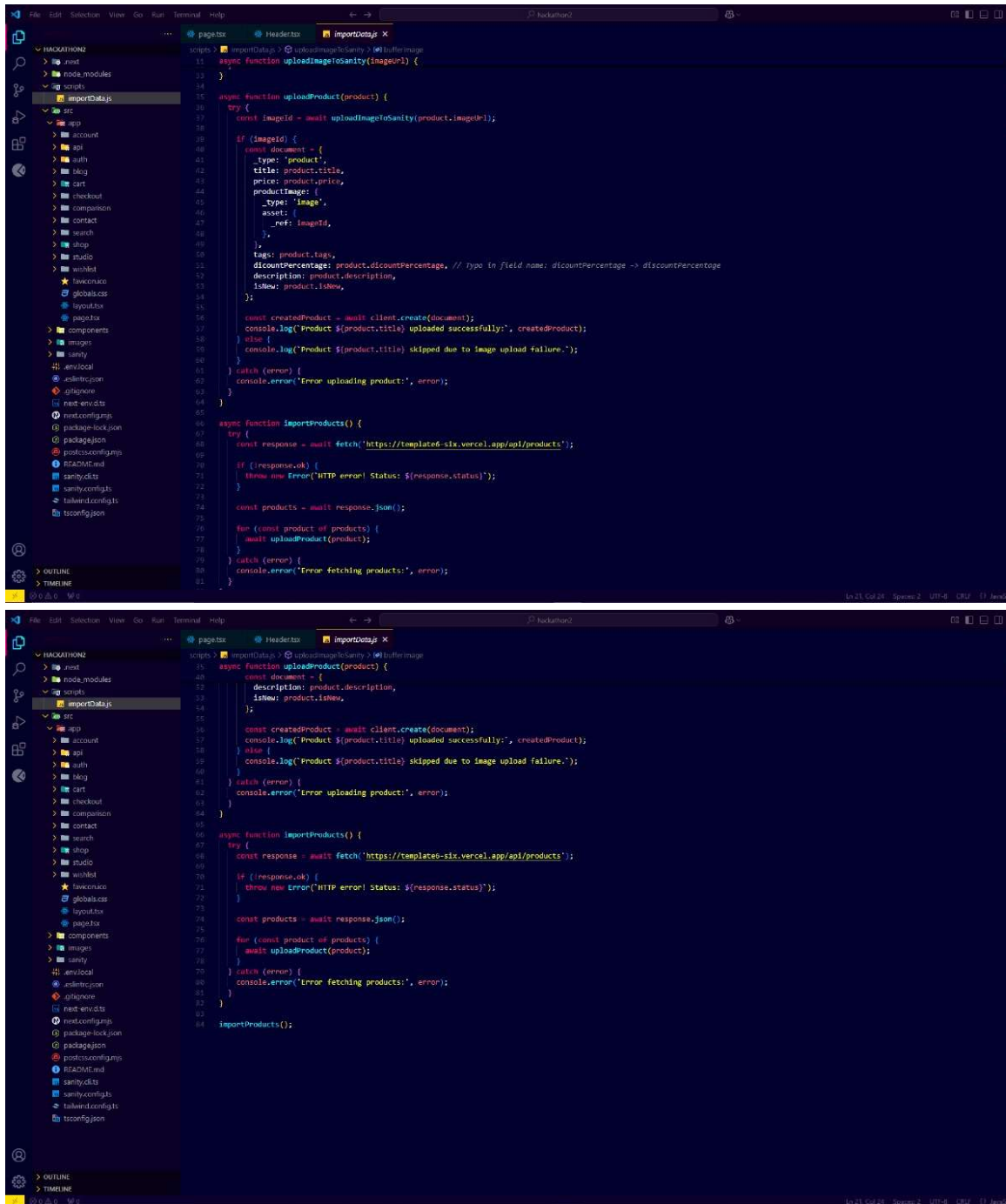
    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      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;
  }
}

export async function uploadProduct(product) {
  try {
    const imageUrl = await uploadImageToSanity(product.imageUrl);

    if (imageUrl) {
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageUrl,
          },
        },
        tags: product.tags,
      };
    }
  }
}
```



Sanity API Configuration:

- Connects to Sanity using a predefined dataset and project ID, authenticated via an API token.
- Sensitive data (project ID, dataset) stored in environment variables.

2. Data Retrieval from Sanity:

- Fetches content using GROQ queries.
- Retrieves product details like categories, descriptions, and pricing.

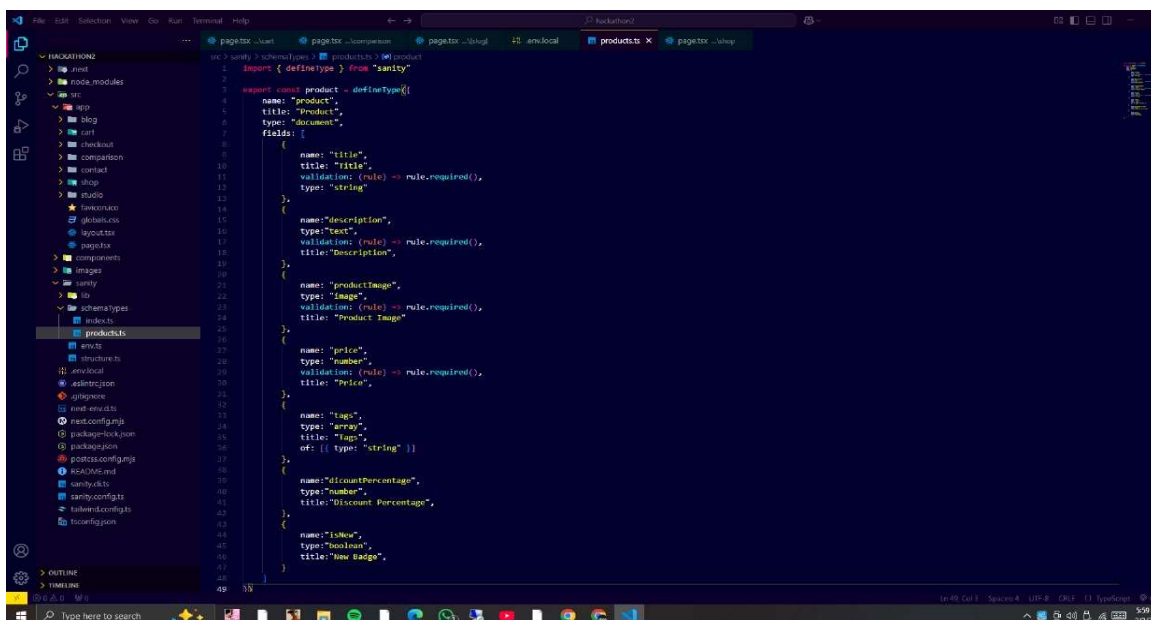
3. Data Mapping and Adjustment:

- Restructures fetched data to fit the application's schema.

4. Database Insertion:

- Inserts data into the database via REST API or commands.
- Error handling ensures smooth migration.

Schema Page Code



```
import { defineType } from 'sanity'

export const product = defineType({
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'title',
      title: 'Title',
      validation: (rule) => rule.required(),
      type: 'string',
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
      validation: (rule) => rule.required(),
    },
    {
      name: 'productImage',
      title: 'Product Image',
      type: 'image',
      validation: (rule) => rule.required(),
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
      validation: (rule) => rule.required(),
    },
    {
      name: 'tags',
      title: 'Tags',
      type: 'array',
      of: [{ type: 'string' }],
    },
    {
      name: 'discountPercentage',
      title: 'Discount Percentage',
      type: 'number',
    },
    {
      name: 'isNew',
      title: 'New Badge',
      type: 'boolean',
      default: true,
    },
  ],
})
```

Title: name of the product

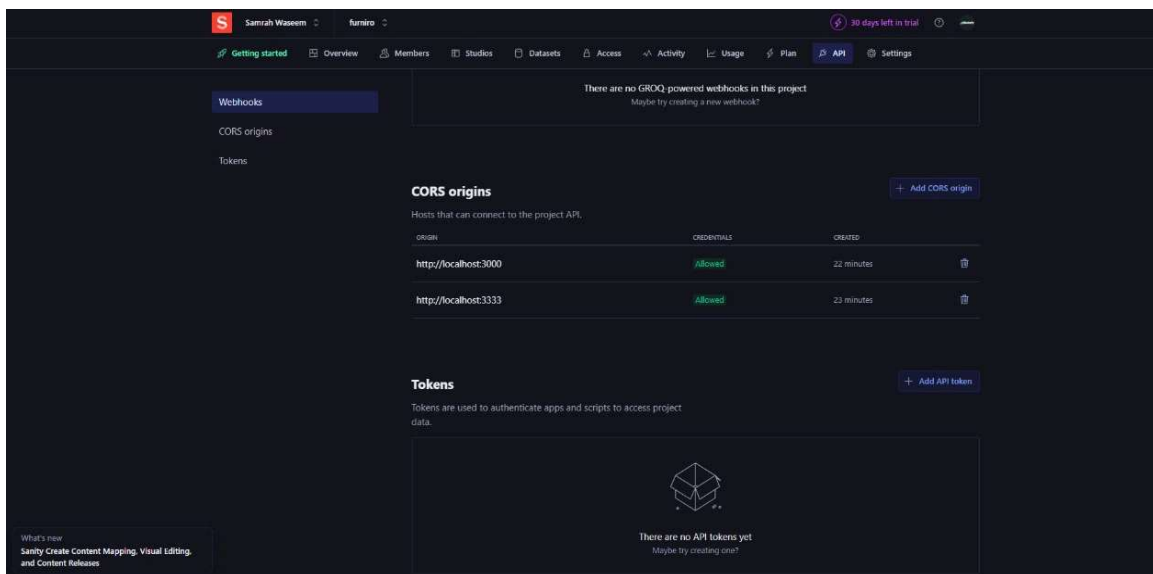
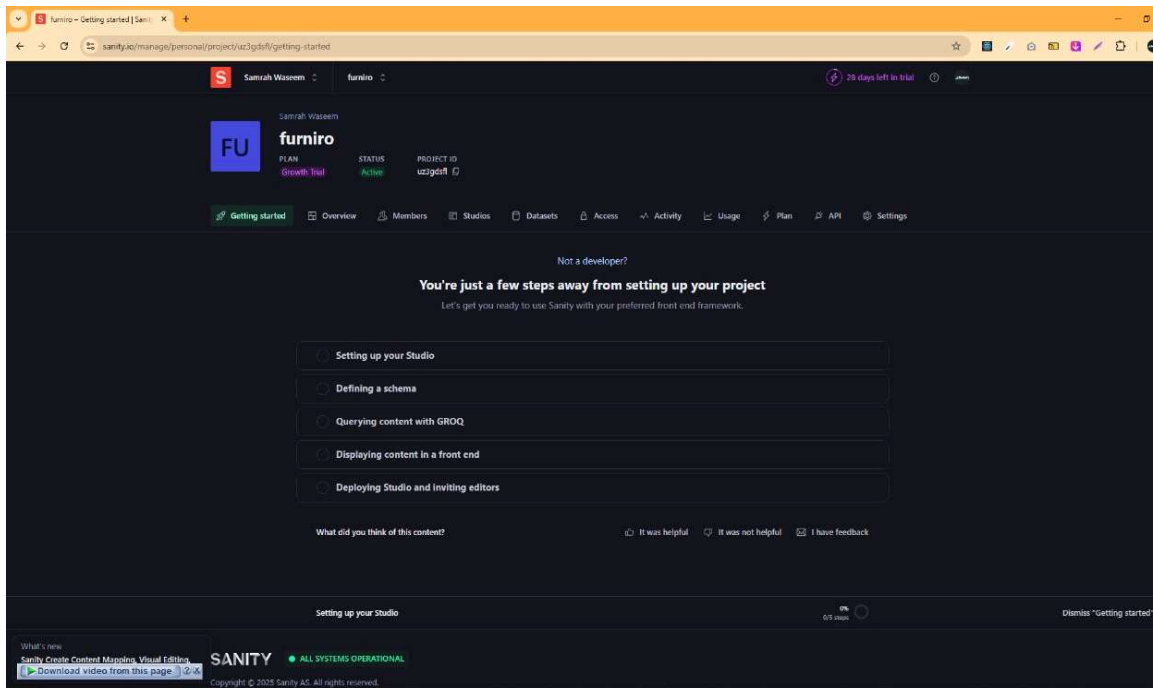
Description: detail of the products.

Product Image: field to save image of the products

Price: field for the product cost

Tags: A unique name for products helps with searching and filtering products. Discount % represents the discount value as a percentage.

Sanity Output:



Getting started

Overview

Members

Studios

Datasets

Access

Activity

Usage

Plan

API

Settings

Webhooks

CORS origins

Tokens

Origin

CREDENTIALS

CREATED

http://localhost:3000

Allowed

23 minutes

http://localhost:3333

Allowed

24 minutes

Tokens

+ Add API token

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

NAME

PERMISSIONS

CREATED

Day3

Developer

just now

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

skwVv8BARiC-JamuQVbRuxThADFDGglygzP9Nuh0Lkm64FD55d0BILpv2pkC4ksIG52bu1lhPhYGnGRu3eter3Q33a2p6GQp11Ng1km3D2E7mX08R0HfclLY9ey3FE-Y47UBAQ0hxc7R5q0PvuH2cF248q8VSamlJ2os52VR3HPXMI2EyDxhv5F

What's new

Sanity Create Content Mapping, Visual Editing, and Content Releases

SANITY

ALL SYSTEMS OPERATIONAL

Copyright © 2025 Sanity AS. All rights reserved.

Token created

Getting started

Overview

Members

Studios

Datasets

Access

Activity

Usage

Plan

API

Settings

Webhooks

CORS origins

Tokens

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

NAME

PERMISSIONS

Day3

Developer

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

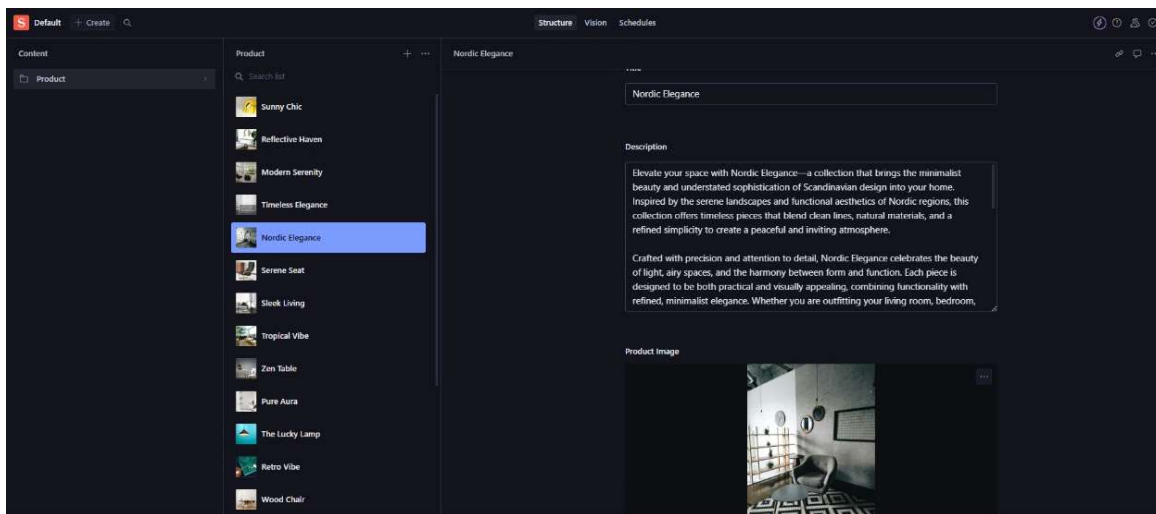
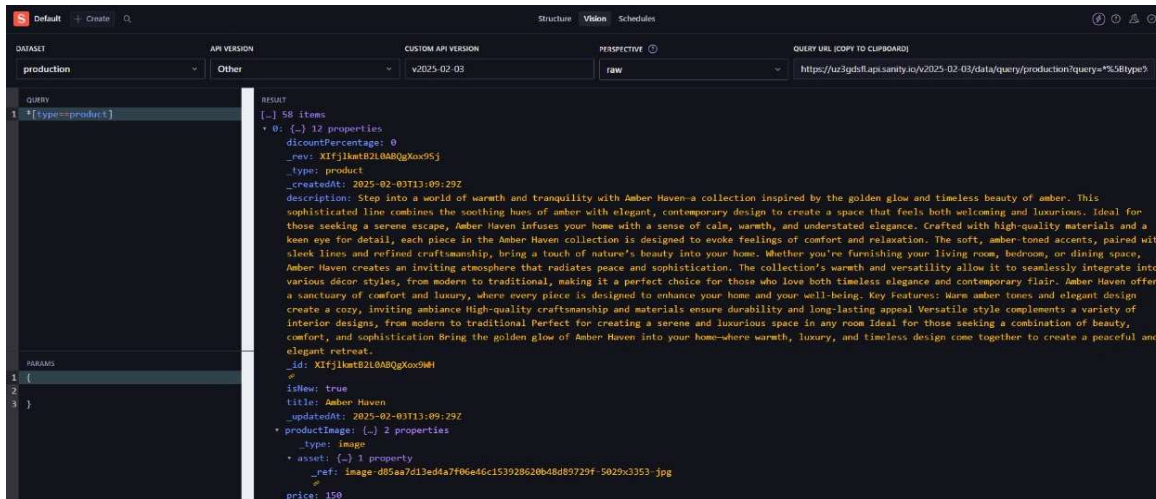
Read access to all datasets, with limited access to project settings. (tokens: read-only)

Save

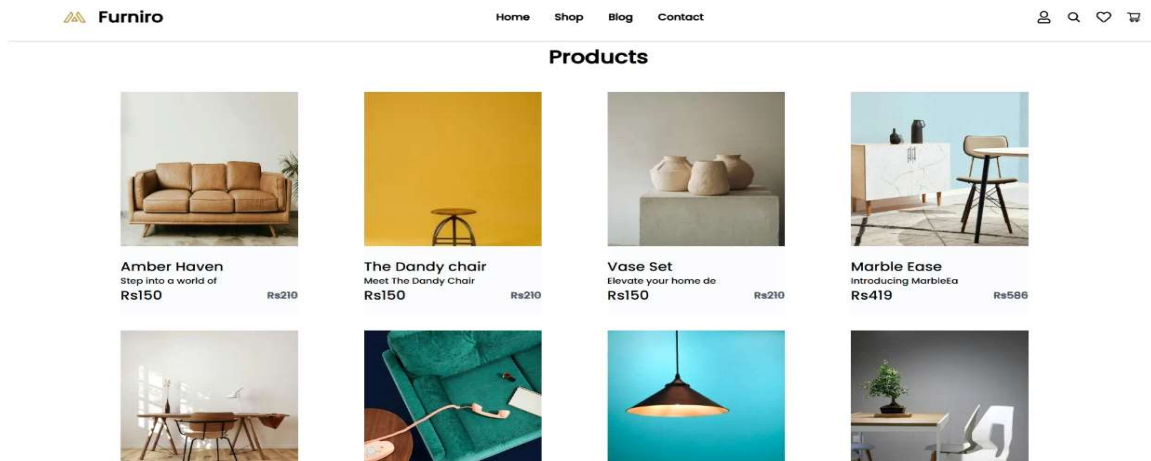
Cancel

There are no API tokens yet

Click here to create a token



Fetch Products on UI



Day 3 completed successfully.

The API integration and data migration process were finished, which included setting up the schema, fetching data from Sanity, and implementing dynamic routing for the marketplace.