Day 3

API INTEGRATION AND DATA SETUP FOR FURNIRO FURNITURE WEBSITE

Overview:

This document provides an overview of the progress achieved on Day 3 of the Furniro website development. The focus was on integrating APIs, creating custom schemas for furniture data, and utilizing GROQ queries to dynamically display content in a Next.js application. Since the data is manually added via the Sanity dashboard, this guide emphasizes schema setup and querying while excluding data migration processes.

Custom Schema Setup in Sanity CMS:

The custom schema in Sanity CMS defines how your furniture data, including product details, pricing, and category information, is organized. By setting up a schema, you ensure that the data is consistently structured and easily accessible within the CMS. Below is an example of how to set up a schema for Furniture Item data.

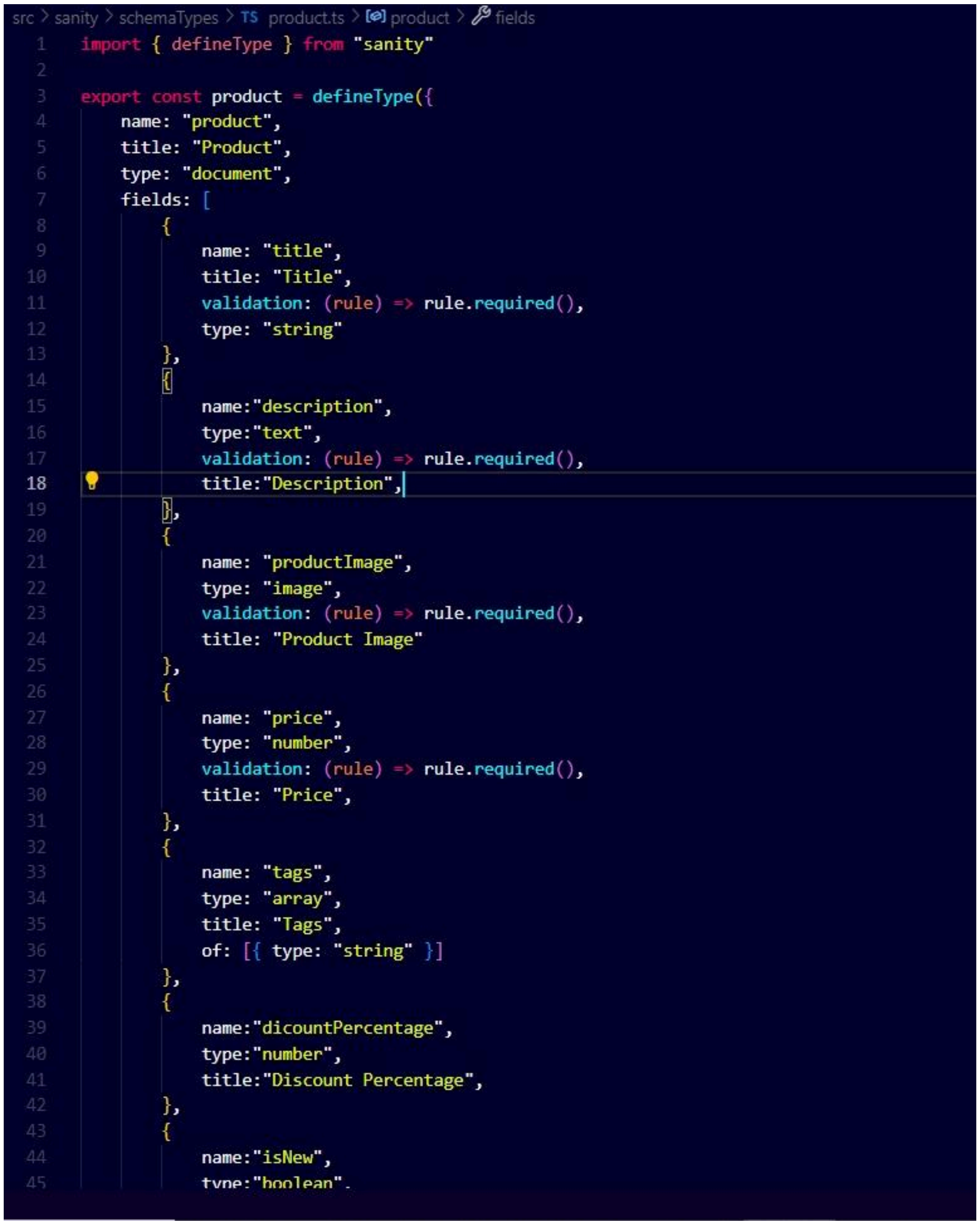
Key Components of the Furniture Item Schema:

Title: The name of the furniture item.
Description: A detailed description of the furniture item.
Price: The price of the furniture item.
Images: Images representing the furniture item.

Custom Validation:

To ensure data integrity and maintain accuracy, custom validation rules are incorporated into the schema:

Price: The price must always be a positive number. This ensures that invalid or negative prices are not entered.
Title: The title field cannot be left blank, ensuring that every product has a name.
Description: The description field cannot be empty, ensuring that there is sufficient information about each product.



src > sanity > schemaTypes > TS product.ts > product > fields

```
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",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
```

1. Sanity API Integration:

Sanity API integration connects the Furniro Furniture website to the Sanity CMS. It allows seamless retrieval, management, and display of data, ensuring the website stays updated with the latest product information.

2. Fetching Data from Sanity:

Data is fetched from Sanity CMS using GROQ (Graph-Relational Object Queries). These queries retrieve specific fields like product details, categories (e.g., living room, bedroom, office), prices, and images.

3. Example Query:

An example GROQ query to fetch furniture data:

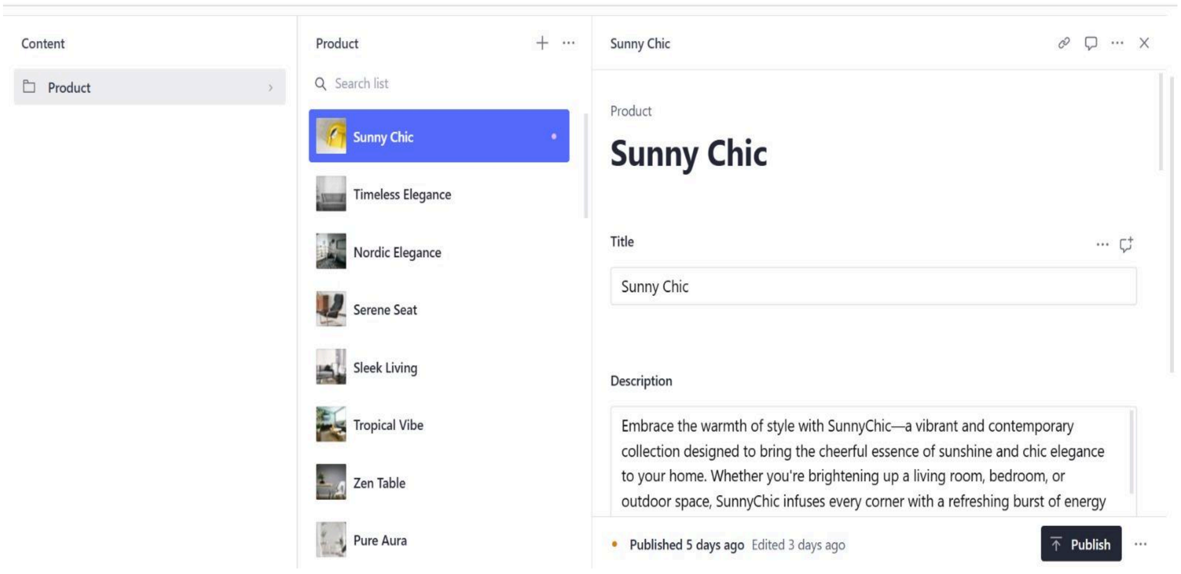
```
*[_type == "furnitureItem"]{
  _id,
  title,
  description,
  price,
  category->name,
  image{
    asset->{
      url
    }
  }
}
```

4. Mapping and Formatting:

The fetched data is processed to fit the website's UI. This includes grouping products by categories, formatting prices into currency format, and ensuring all required fields (e.g., title, description) are present.

5. Displaying Data:

The formatted data is displayed dynamically on the website. Each furniture item is presented with an image, title, description, price, and category, providing users with an intuitive browsing experience.



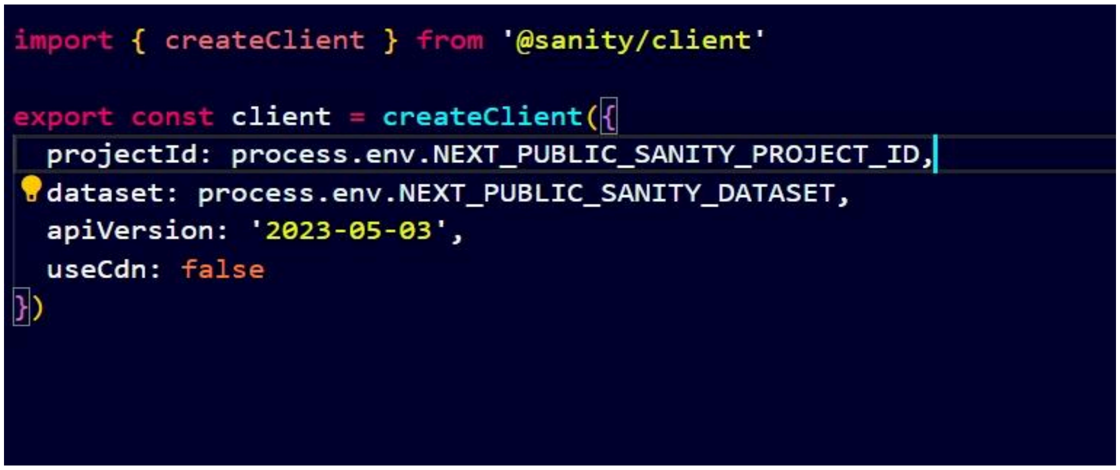
Sunny Chic

Title: Sunny Chic

Description: Embrace the warmth of style with SunnyChic—a vibrant and contemporary collection designed to bring the cheerful essence of sunshine and chic elegance to your home. Whether you're brightening up a living room, bedroom, or outdoor space, SunnyChic infuses every corner with a refreshing burst of energy

CLIENT-SIDE CODE:

The client-side code is responsible for rendering the Furniro data on a Next.js page. It handles the display of products, ensuring a seamless user experience by fetching and presenting data efficiently. The functionality of the code is explained in detail, providing insights into how it operates, from retrieving data to dynamically updating the user interface. This section highlights the key aspects of the implementation, making it easier to understand the flow and purpose of the code.



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

export const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  apiVersion: '2023-05-03',
  useCdn: false
})
```

GROQ Query to Fetch Data:

The getServerSide Props function utilizes GROQ query to fetch data from Sanity during server-side rendering (SSR). This approach ensures that the data is pre-fetched and injected into the page before it is served to the user, enabling seamless and fost content delivery.

Rendering Items:

The ClientPage component is responsible for rendering the list of furniture items passed via props. Using React's map() method, the fetched data is iterated over to create a list of Product cards, showcasing details like product name, description, and image.

Dynamic Routing:

The code incorporates dynamic routing links for individual Product items. When a user clicks on a product card, they are navigated to a detailed page (e.g./product/[id]), allowing them to explore more information about the selected furniture.

Code Highlights:

SSR Optimization: Server-side rendering improves loading times and enhances SEO, ensuring that search engines index the content effectively.

Responsive Design: The component structure is designed to be fully responsive, adapting seamlessly to various screen sizes, ensuring a smooth experience on mobile, tablet, and desktop devices.

Responsive Design

The Furniro furniture site is styled using modern CSS or Tailwind CSS, ensuring the layout adapts seamlessly to all devices while maintaining accessibility for all users.

Interactive User Experience

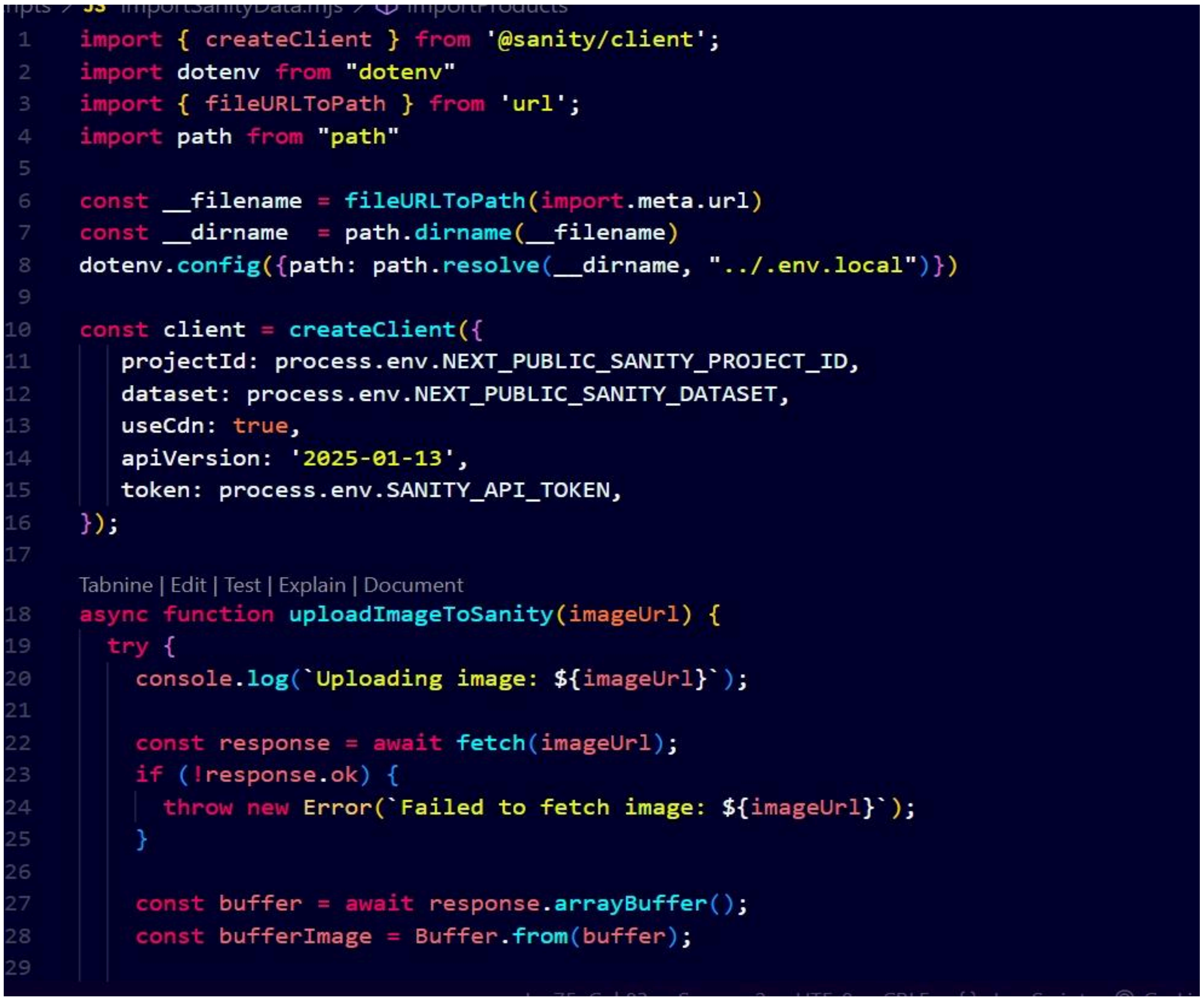
Dynamic features like an Add to Cart button and a View Details option make the site engaging. Images are optimized with next/image, ensuring fast loading and high-quality visuals.

Reusable Components:

The card component is designed to be versatile and reusable, allowing consistent use across the homepage, category pages, and promotional sections.

Secure Configuration

Sensitive data, such as API keys and database credentials, are securely managed using a env file tailored for the Furniro furniture application.



```javascript
import { createClient } from '@sanity/client';
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")})

const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: true,
  apiVersion: '2025-01-13',
  token: process.env.SANITY_API_TOKEN,
});

Tabnine | Edit | Test | Explain | Document
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);

```

Environment Configuration :

Project identification:
The SANITY PROJECT ID acts as a unique identifier, ensuring all API interactions directed to the correct Sanity project

Dataset Management:
The SANITY DATASET specifies the environment type, such as production or development, enabling seamless content management tailored to different stages of the application lifecycle.

API Authentication and Security

Secure API Token:
The SANITY API TOKEN a highly secure credential used for authenticating requests to Sanity APIs. This token is never exposed to the frontend or unauthorized users, ensuring robust Apt security.

Backend Connections:
Additional environment variables handle secure connections to databases and backend services, ensuring seamless communication between various system components.

Best Practices for Security

Protected Access:
All environment variables are securely stored in the env file and accessed using process.env, moking them inaccessible to the client-side application.

Mitigating Risks:
By adhering to security best practices, sensitive information like tokens, keys, and database configurations are shielded from potential vulnerabilities or breache.

FRONTEND IMPLEMENTATION HIGHLIGHTS:

The focus was on backend setup and dynamic integration for the Fumiro website.

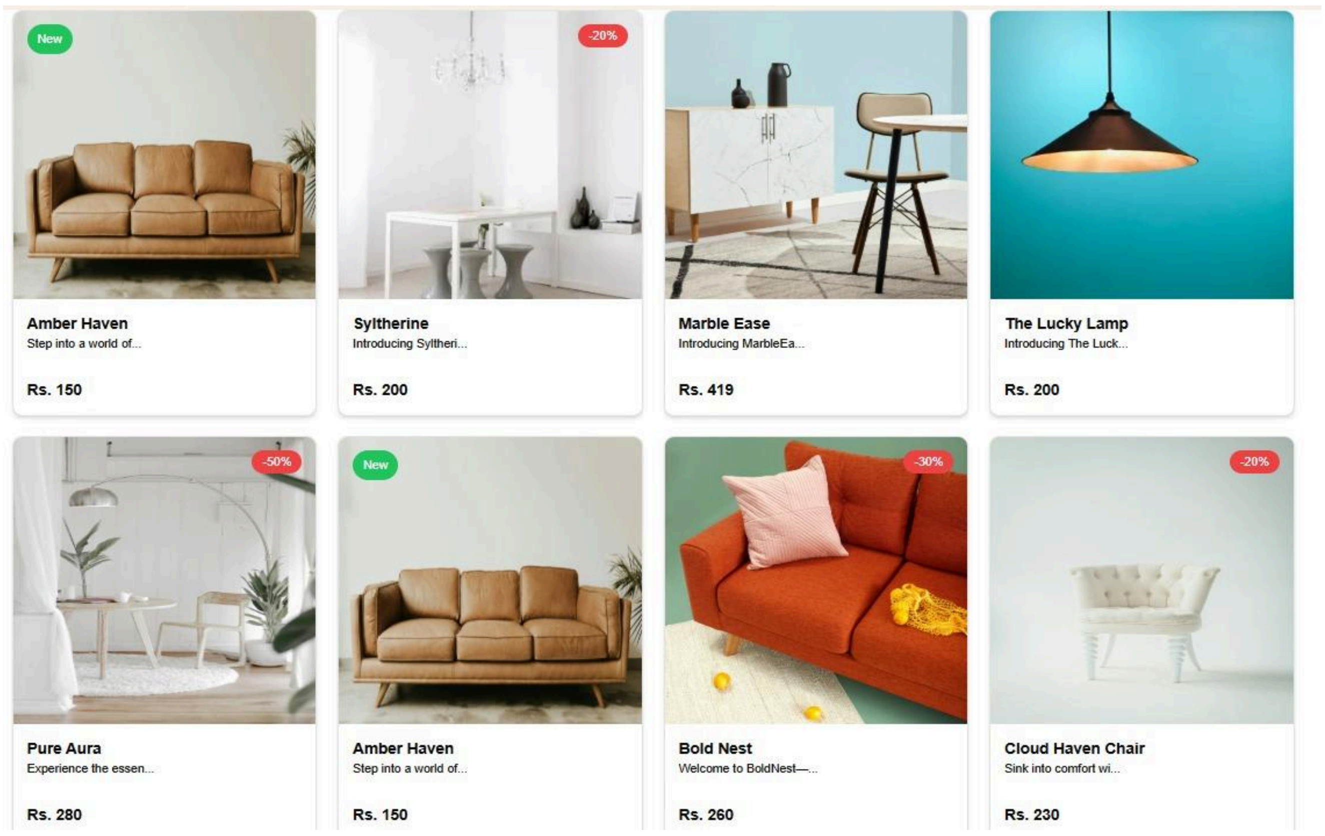
Key highlights:

Schema Design: Developed a robust structure for product data in Sanity.

Dynamic Content: Integrated GROQ queries for fetching and rendering data.

Responsive: UI Designed a mobile-friendly layout for menus and restaurant details

Secure Setup: Managed sensitive configurations with environment variables.


New
Amber Haven
Step into a world of...
Rs. 150
-20%
Syltherine
Introducing Syltheri...
Rs. 200
Marble Ease
Introducing MarbleEa...
Rs. 419
The Lucky Lamp
Introducing The Luck...
Rs. 200
-50%
Pure Aura
Experience the essen...
Rs. 280
New
Amber Haven
Step into a world of...
Rs. 150
-30%
Bold Nest
Welcome to BoldNest—...
Rs. 260
-20%
Cloud Haven Chair
Sink into comfort wi...
Rs. 230