API INTEGRATION AND DATA SETUP FOR FURNIRO FUNITURE WEBSITE

HACKATHON 3 DAY 3

AQSA KHASKHELI
REG NO: 00423336
TUESDAY 2 TO 5

OVERVIEW:

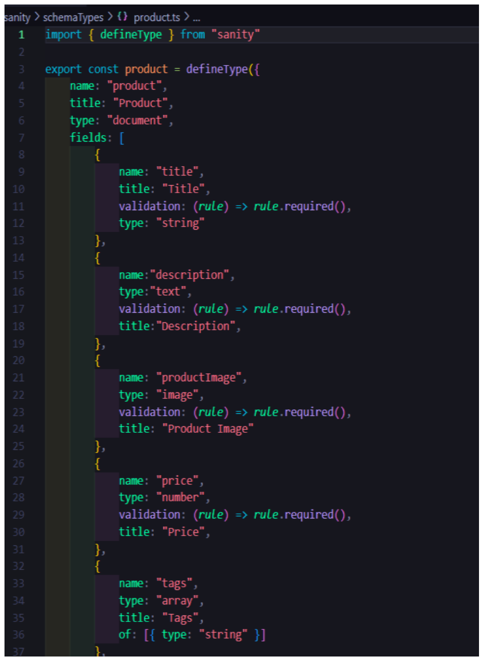
This documention outlines the progress made on DAY 3 of the furniro website development. It covers API integration, setting up custom schemas for furniture data, and using using GROQ queries to display content in a Next.js app. Since data is manually added through the sanity dashboard, this guide focuses on schema setup and querying,excluding data migration.

CUSTOM SCHEMA SETUP IN SANITY CMS

The custom schema define how the furniture data, such as product details, pricing and category information is structured in the sanity CMS. On the ridht is an example of how to set a schema for right is an example of how to set up a schema for the “Furniture Item” data.

Custom Validation

The schema incorporates validation rules to maintain data accuracy and integrity for insatance, the price must be a positive number, while the title and description fields cannot be left blank.



```
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" }]
        },
```

Sanity API Intergration:

The application connents to sanity"s API using a configured project ID and data. Authentication is handled securely via an API token. Environment variables(project ID, dataset) ensure sensitive information is protected.

Fetching Data From Sanity:

GROQ queries are employed to retrieve structurd content from sanity CMS, pulling essential furniture related data, such as product names, categories like sofas, chairs, tables, lighting, price, descriptions,and images. These queries allow deamless integration for furniture specific details into thw website.

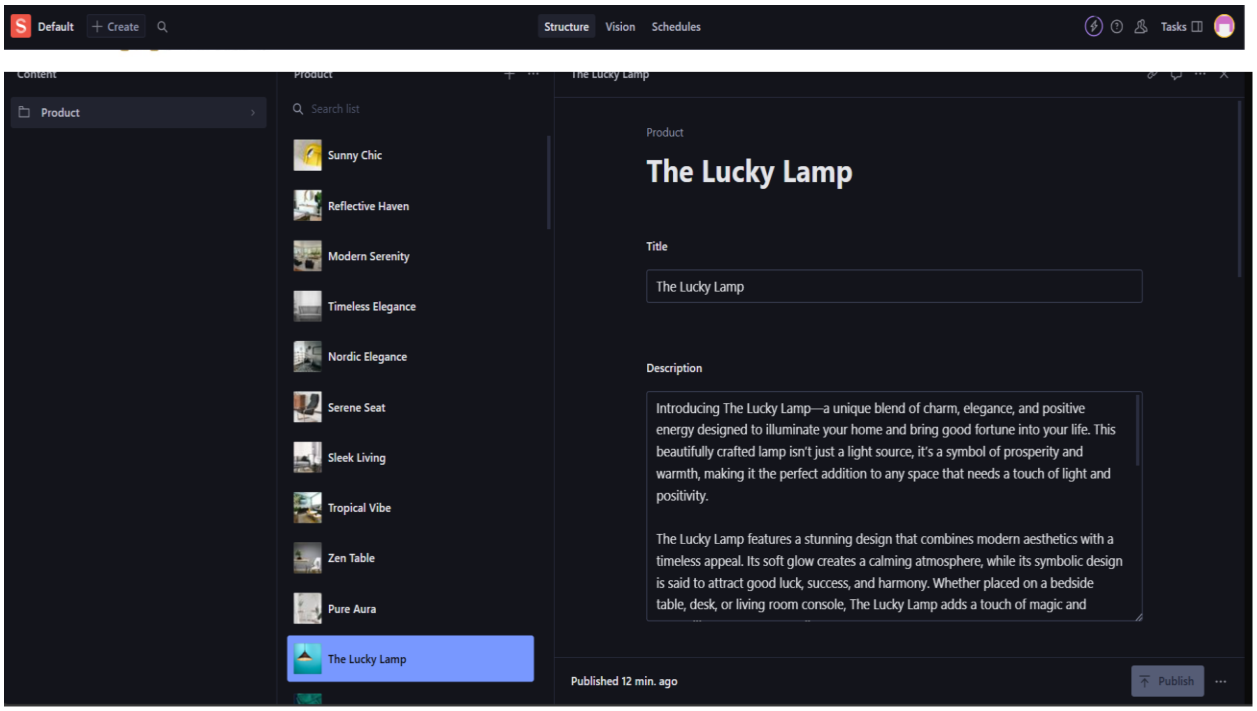
Mapping And Formatting:

After data is fetched, it is mapped and formatted to align with the website"s schema and design specifications. Each product record like a sofa or dining table is restructured to match the frontend display requirments of the Furniro website . For example: details such as the product title, description, price and images are organized for optinal rendering on the site.

Display data:

The structured data fetched from sanity is dynamically displayed on the Furniro Website. This data includes: Furnitue items categorized into relevant sections like: sofas, chairs, tables.

Since the data is oulled live from sanity through GROQ queries there is no need for database storage or manual API insertion. This ensures that the website reflects real time updates, providing the latest product information without requiring additional updates or maintenance.



Product

The Lucky Lamp

Title

The Lucky Lamp

Description

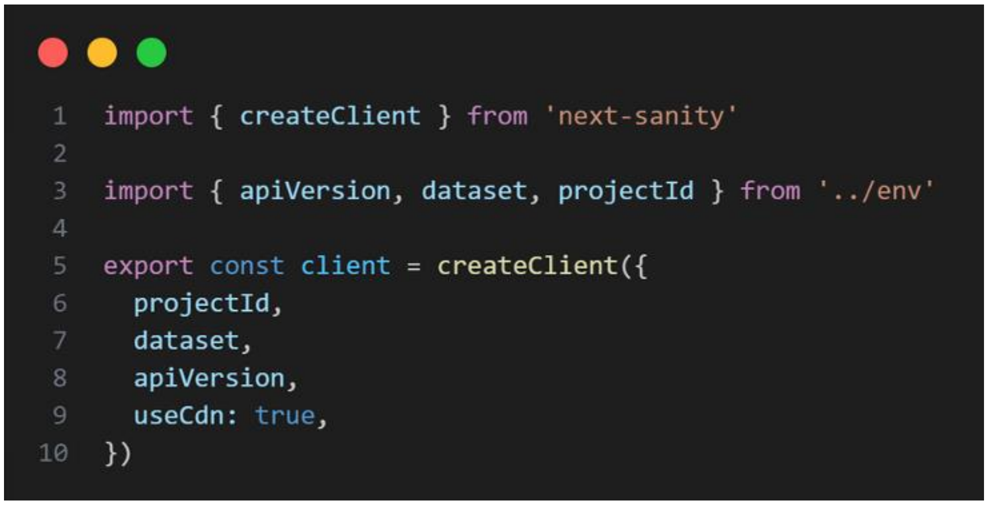
Introducing The Lucky Lamp—a unique blend of charm, elegance, and positive energy designed to illuminate your home and bring good fortune into your life. This beautifully crafted lamp isn't just a light source, it's a symbol of prosperity and warmth, making it the perfect addition to any space that needs a touch of light and positivity.

The Lucky Lamp features a stunning design that combines modern aesthetics with a timeless appeal. Its soft glow creates a calming atmosphere, while its symbolic design is said to attract good luck, success, and harmony. Whether placed on a bedside table, desk, or living room console, The Lucky Lamp adds a touch of magic and

Published 12 min. ago

CLIENT-SIDE CODE:

This section covers the client client-side code Responsible for rendering the Furniro data on a Next.js page. On the right is an explanation of how the code functions.



```
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true,
})
```

GROQ Query to Fetch Data:

The getServerSideProps function utilizes a 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 fast 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

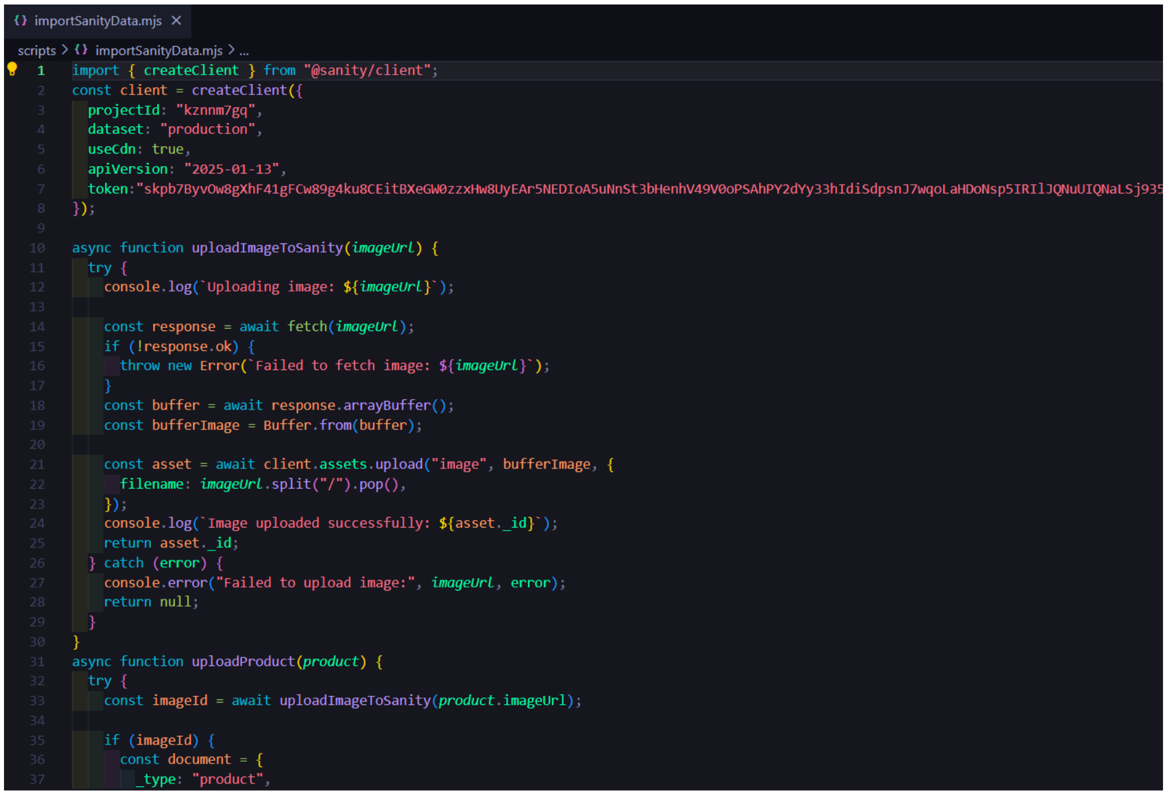
Dynamic features like an Add to Cart button and a View Details option make the site engaging. Images are optimized with next/ image, ensuring fast loading and high-quality visuals.

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";
const client = createClient({
  projectId: "kznnm7gq",
  dataset: "production",
  useCdn: true,
  apiVersion: "2025-01-13",
  token:"skpb7ByvOw8gXhF41gFCw89g4ku8CEitBXeGW0zzxHw8UyEAr5NEDIoA5uNnSt3bHenhV49V0oPSAhPY2dYy33hIdiSdpsnJ7wqoLaHDoNsp5IRIlJQNuUIQNaLSj935
});

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

    if (imageId) {
      const document = {
        _type: "product",
```

Environment Configuration

Project Identification:

The SANITY_PROJECT_ID acts as a unique identifier, ensuring all API interactions are directed to the correct Sanity project.

Dataset set 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 İS a highly secure credential used for authenticating requests to Sanity APIs. This token is never exposed to the frontend or unauthorized users, ensuring robust API 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.eny, making 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 breaches.

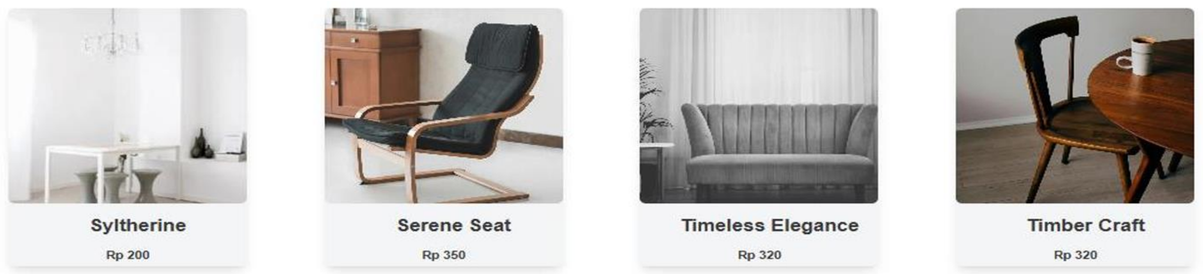
FRONTEND IMPLEMENTATION UPDATES

Day Achievements

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



THANK YOU ☺