API INTEGRATION AND DATA SETUP FOR SOFA WEBSITE

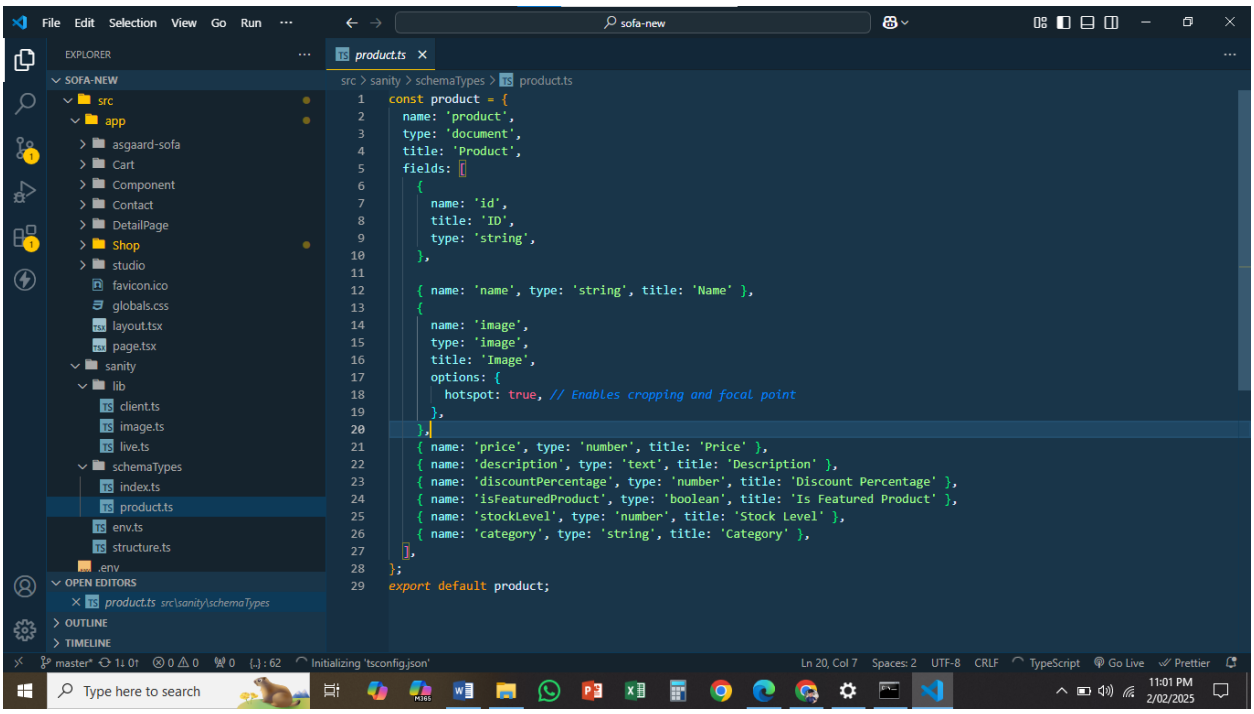
This documentation outlines the progress made on Day 3 of the sofa website development. It covers API integration, setting up custom schemas for furniture and couch data, and using GROQ queries to display content in a Next.js app. Since data is manually added through the Sanity dashboard, this guide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

CUSTOM SCHEMA SETUP IN SANITY CMS

The custom schema defines how the furniture and sofa data, such as product details, pricing, and category information, is structured in the Sanity CMS. On the right is an example of how to set up a schema for the sofa Item ' data.

Custom Validation

The schema incorporates validation rules to maintain data accuracy and integrity. For instance, the Price must be a positive number, while the Title and Description fields cannot be left blank.



Sanity API Integration:

The application connects to Sanity's API using a configured project ID and dataset. Authentication is handled securely via an API token. Environment variables (e.g., project ID, dataset) ensure sensitive information is protected.

Fetching Data from Sanity:

GROQ queries are employed to retrieve structured content from Sanity CMS, pulling essential furniturerelated data, such as product names, categories (e.g., sofas, chairs, tables, lighting), prices, descriptions, and images. These queries allow seamless integration of furniture-specific details into the website.

Example Query:

A typical query fetches data for individual furniture items, including product names, descriptions, prices, images, and categories (e.g., sofa, chairs, tables). This ensures all necessary details are accessible for display.

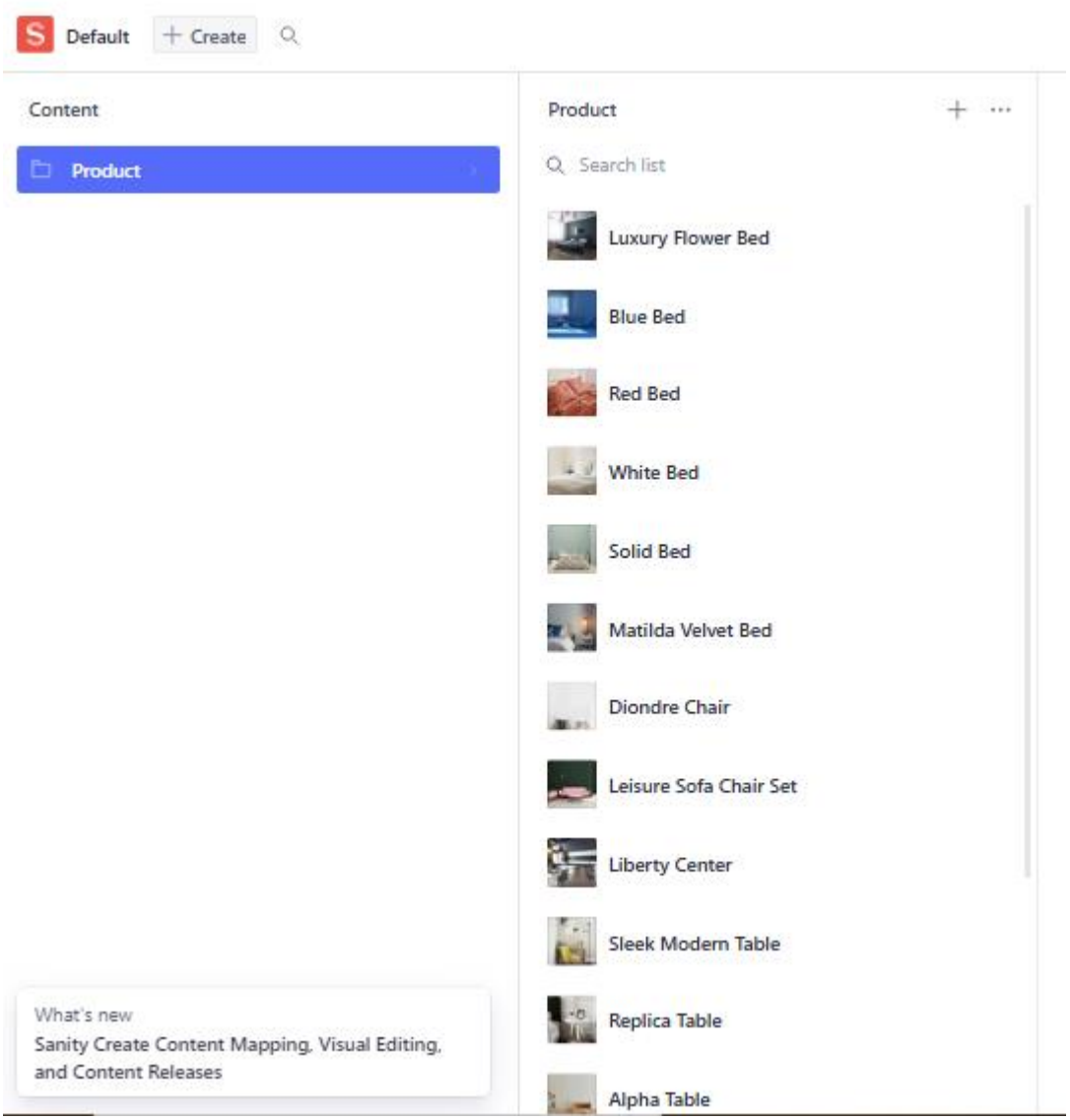
Mapping and Formatting:

After data is fetched, it is mapped and formatted to align with the website's schema and design specifications. Each product record (e.g., a sofa or dining table) is restructured to match the frontend display requirements of the Sofa website. For example, details such as the product title, description, price, and images are organized for optimal rendering on the site.

Displaying Data:

The structured data fetched from Sanity is dynamically displayed on the Sofa website. This data includes: Furniture items categorized into relevant sections (e.g., sofas, chairs, tables). Since the data is pulled live from Sanity through GROQ queries, there's 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



CLIENT-SIDE CODE:

This section covers the client-side code responsible for rendering the sofas data on a Next.js page. At the bottom is an explanation of how the code functions.

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., /productDetail/[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 Sofa 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 Sofa application.

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 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 is 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.env, 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.

```ts
export const apiVersion =
  process.env.NEXT_PUBLIC_SANITY_API_VERSION || '2025-01-29'

export const dataset = assertValue(
  process.env.NEXT_PUBLIC_SANITY_DATASET,
  'Missing environment variable: NEXT_PUBLIC_SANITY_DATASET'
)

export const projectId = assertValue(
  process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  'Missing environment variable: NEXT_PUBLIC_SANITY_PROJECT_ID'
)

function assertValue<T>(v: T | undefined, errorMessage: string): T {
  if (v === undefined) {
    throw new Error(errorMessage)
  }

  return v
}
```

FRONTEND IMPLEMENTATION HIGHLIGHTS

Day 3 Achievements

The focus was on backend setup and dynamic integration for the Sofa 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

Cozy Sofa

520$ Item: 15

Replica Table

750$ Item: 25

Matilda Velvet Bed

600$ Item: 12