DAY 3 - API INTEGRATION REPORT - COMFORTY

Prepared by: Hafiza Alina Yasmeen
Date: January 17, 2025

---

1. API Integration Process

1.1 Overview

The API integration process for "Comforty" was focused on connecting the backend powered by Sanity CMS with the frontend to display and manage furniture products like sofas, chairs, and tables.

1.2 Steps Involved

1. **Sanity Client Setup**:
   Installed `@sanity/client` to fetch and manage data from Sanity CMS.
2. `npm install @sanity/client`
3. **API Development**:
   Created endpoints to handle core functionality:
   - Fetching products.
   - Managing orders.
   - Tracking shipments.
4. **Frontend Integration**:
   Integrated API calls into the frontend using React components to fetch and display data dynamically.

---

2. Adjustments Made to Schemas

2.1 Product Schema

- Added `tags` field for categorizing products (e.g., `"featured"`, `"sale"`).
- Adjusted `category` to reference pre-defined categories.
- Final Schema:
- ```
  {
  ```
- ```
    name: "products",
  ```

- `  title: "Products",`
- `  type: "document",`
- `  fields: [`
- `    { name: "title", title: "Product Title", type: "string", validation: (Rule) => Rule.required() },`
- `    { name: "price", title: "Price", type: "number", validation: (Rule) => Rule.required() },`
- `    { name: "image", title: "Image", type: "image", validation: (Rule) => Rule.required() },`
- `    { name: "tags", title: "Tags", type: "array", of: [{ type: "string" }] },`
- `    { name: "category", title: "Category", type: "reference", to: [{ type: "categories" }] },`
- `    { name: "inventory", title: "Inventory", type: "number", validation: (Rule) => Rule.required() },`
- `  ],`
- `}`

2.2 Order Schema

- Linked `products` as a reference array.
- Included `status` options: `"pending"`, `"processing"`, `"delivered"`.
- Final Schema:
- `{`
- `  name: "order",`
- `  title: "Order",`
- `  type: "document",`
- `  fields: [`
- `    { name: "user", title: "User", type: "reference", to: [{ type: "user" }] },`
- `    { name: "products", title: "Products", type: "array", of: [{ type: "reference", to: [{ type: "products" }] }] },`
- `    { name: "status", title: "Status", type: "string", options: { list: ["pending", "processing", "delivered"] } },`
- `  ],`
- `}`

---

3. Migration Steps and Tools Used

3.1 Tools

- **Sanity CLI**:
  Used `sanity dataset import/export` commands for migrating data.
- **Custom Scripts**:
  Wrote Node.js scripts to map existing furniture data into Sanity schema.

3.2 Migration Steps

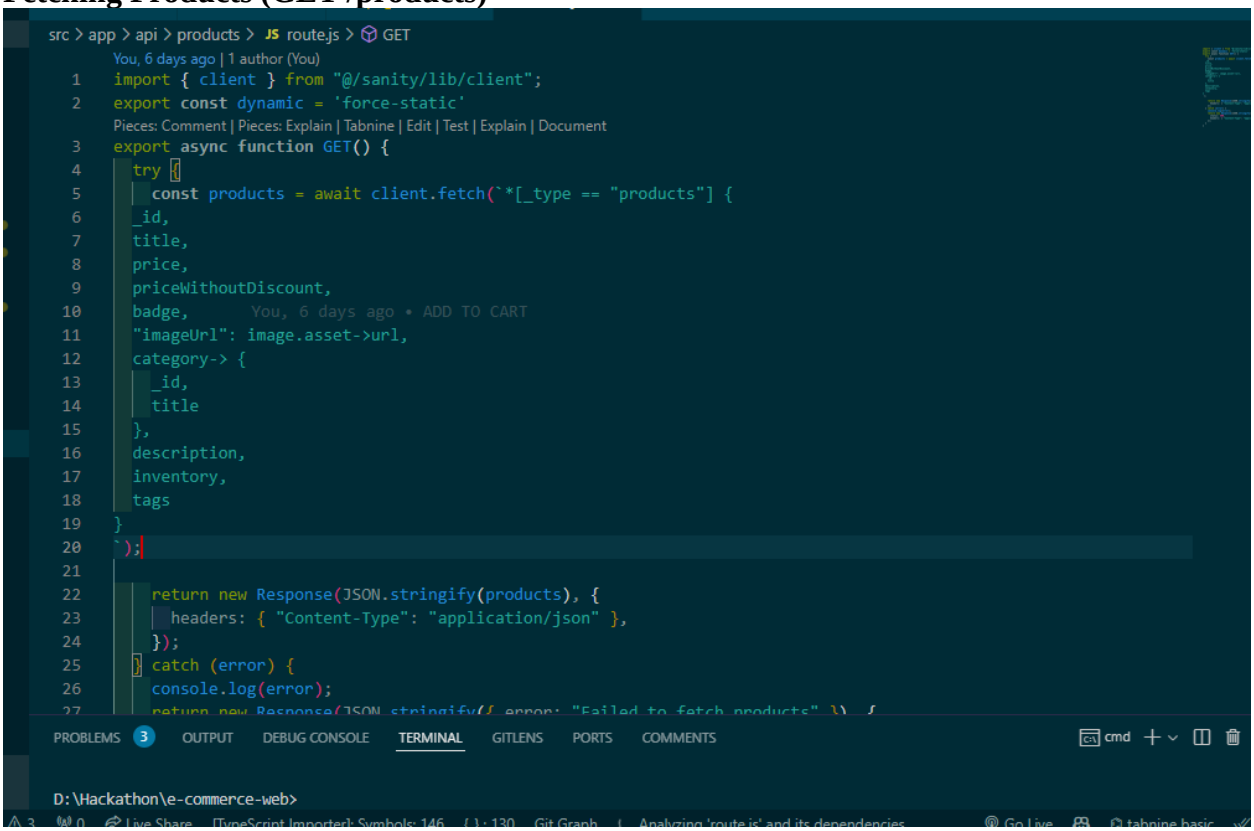
1. Exported existing datasets from previous storage:
2. `sanity dataset export production export-path`
3. Imported adjusted data into Sanity:
4. `sanity dataset import new-data.json production`
5. Verified data integrity in Sanity Vision.

---

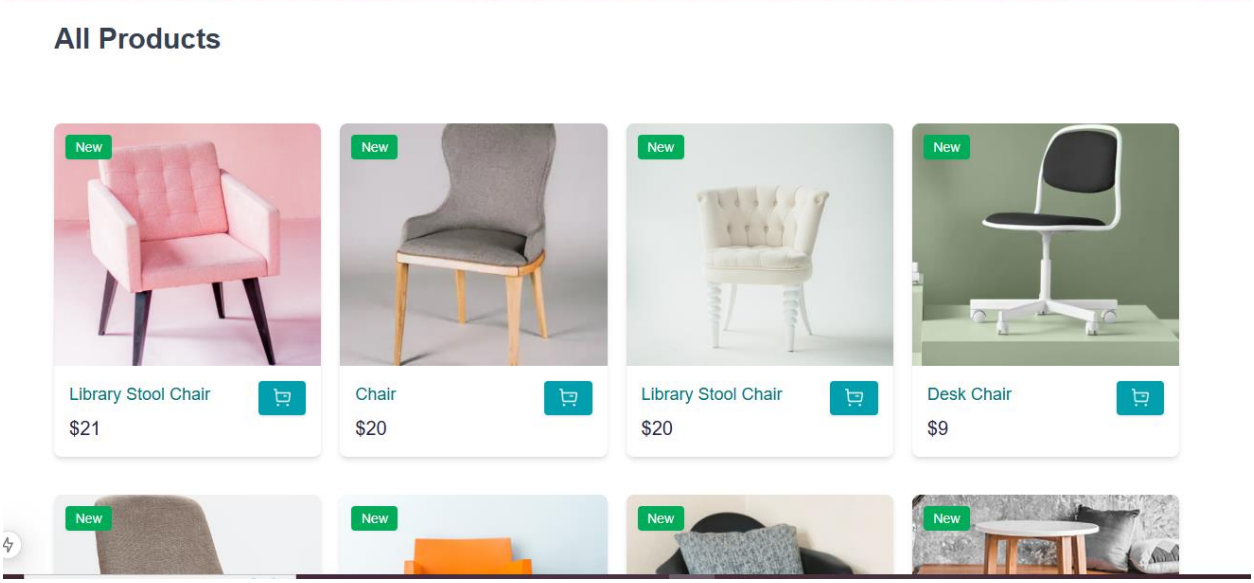
4. Screenshots

4.1 API Calls

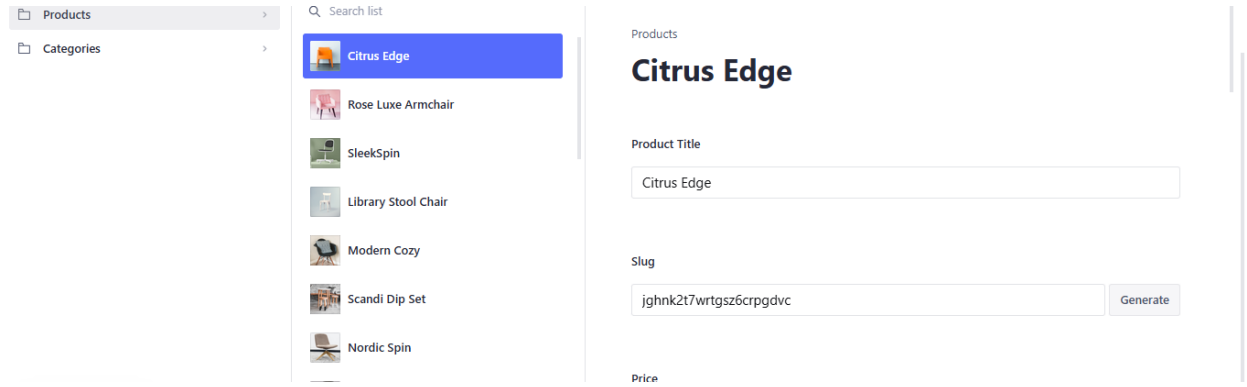
- **Fetching Products (GET /products)**



4.2 Data Displayed in Frontend



4.3 Populated Sanity CMS



5. Code Snippets

5.1 API Integration

Fetching Products:

```
import { client } from "@/lib/SanityClient";
```

```
   const products = await client.fetch(`*[_type == "products"] {
  _id,
  title,
  price,
  priceWithoutDiscount,
  badge,
  "imageUrl": image.asset->url,
  category-> {
    _id,
    title
  },
  description,
  inventory,
tags}`);
```

5.2 Migration Script

```
// Import environment variables from .env.local
import "dotenv/config";

// Import the Sanity client to interact with the Sanity backend
import { createClient } from "@sanity/client";

// Load required environment variables
const {
 NEXT_PUBLIC_SANITY_PROJECT_ID, // Sanity project ID
 NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
 NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
 BASE_URL = "https://giaic-hackathon-template-08.vercel.app", // API base URL for products and categories
} = process.env;

// Check if the required environment variables are provided
if (!NEXT_PUBLIC_SANITY_PROJECT_ID || !NEXT_PUBLIC_SANITY_AUTH_TOKEN) {
 console.error("Missing required environment variables. Please check your .env.local file.");
 process.exit(1); // Stop execution if variables are missing
}

// Create a Sanity client instance to interact with the target Sanity dataset
const targetClient = createClient({
 projectId: NEXT_PUBLIC_SANITY_PROJECT_ID, // Your Sanity project ID
 dataset: NEXT_PUBLIC_SANITY_DATASET || "production", // Default to "production" if not set
 useCdn: false, // Disable CDN for real-time updates
 apiVersion: "2023-01-01", // Sanity API version
 token: NEXT_PUBLIC_SANITY_AUTH_TOKEN, // API token for authentication
});

// Function to upload an image to Sanity
async function uploadImageToSanity(imageUrl) {
```

```
  try {
    // Fetch the image from the provided URL
    const response = await fetch(imageUrl);
    if (!response.ok) throw new Error(`Failed to fetch image: ${imageUrl}`);

    // Convert the image to a buffer (binary format)
    const buffer = await response.arrayBuffer();

    // Upload the image to Sanity and get its asset ID
    const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
      filename: imageUrl.split("/").pop(), // Use the file name from the URL
    });

    return uploadedAsset._id; // Return the asset ID
  } catch (error) {
    console.error("Error uploading image:", error.message);
    return null; // Return null if the upload fails
  }
}

// Main function to migrate data from REST API to Sanity
async function migrateData() {
  console.log("Starting data migration...");

  try {
    // Fetch categories from the REST API
    const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
    if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
    const categoriesData = await categoriesResponse.json(); // Parse response to JSON

    // Fetch products from the REST API
    const productsResponse = await fetch(`${BASE_URL}/api/products`);
    if (!productsResponse.ok) throw new Error("Failed to fetch products.");
    const productsData = await productsResponse.json(); // Parse response to JSON

    const categoryIdMap = {}; // Map to store migrated category IDs

    // Migrate categories
    for (const category of categoriesData) {
      console.log(`Migrating category: ${category.title}`);
      const imageId = await uploadImageToSanity(category.imageUrl); // Upload category image

      // Prepare the new category object
      const newCategory = {
        _id: category._id, // Use the same ID for reference mapping
        _type: "categories",
```

```
      title: category.title,
      image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
    };

    // Save the category to Sanity
    const result = await targetClient.createOrReplace(newCategory);
    categoryIdMap[category._id] = result._id; // Store the new category ID
    console.log(`Migrated category: ${category.title} (ID: ${result._id})`);
  }

  // Migrate products
  for (const product of productsData) {
    console.log(`Migrating product: ${product.title}`);
    const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image

    // Prepare the new product object
    const newProduct = {
      _type: "products",
      title: product.title,
      price: product.price,
      priceWithoutDiscount: product.priceWithoutDiscount,
      badge: product.badge,
      image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
      category: {
        _type: "reference",
        _ref: categoryIdMap[product.category._id], // Use the migrated category ID
      },
      description: product.description,
      inventory: product.inventory,
      tags: product.tags,
    };

    // Save the product to Sanity
    const result = await targetClient.create(newProduct);
    console.log(`Migrated product: ${product.title} (ID: ${result._id})`);
  }

  console.log("Data migration completed successfully!");
 } catch (error) {
  console.error("Error during migration:", error.message);
  process.exit(1); // Stop execution if an error occurs
 }
}

// Start the migration process
migrateData();
```