Name: Syed Arib Hassan
Roll No: 00474138
Class: Monnday (7 PM – 10 PM)

Day 3 - API Integration Report - Comforty

Table of Contents

1. Overview

Comforty is a marketplace platform designed to provide users with an engaging and seamless shopping experience. This project focuses on integrating external APIs and migrating product data into Sanity CMS to build a robust backend. The integration ensures that data is dynamically fetched and rendered on the frontend, offering scalability and flexibility. The process involved schema customization, data migration, and integration with Next.js components.

The Day 3 task focuses on integrating APIs and migrating data into the Sanity CMS to build the backend for Comforty, a marketplace platform. This report outlines the API integration process, schema adjustments, data migration steps, and testing results.

2. API Integration Process

2.1. Data Migration to Sanity CMS:

- The Template 8 API was utilized to fetch product and category data.
- Product and category data were migrated into Sanity CMS using a custom script to ensure alignment with the defined schema.

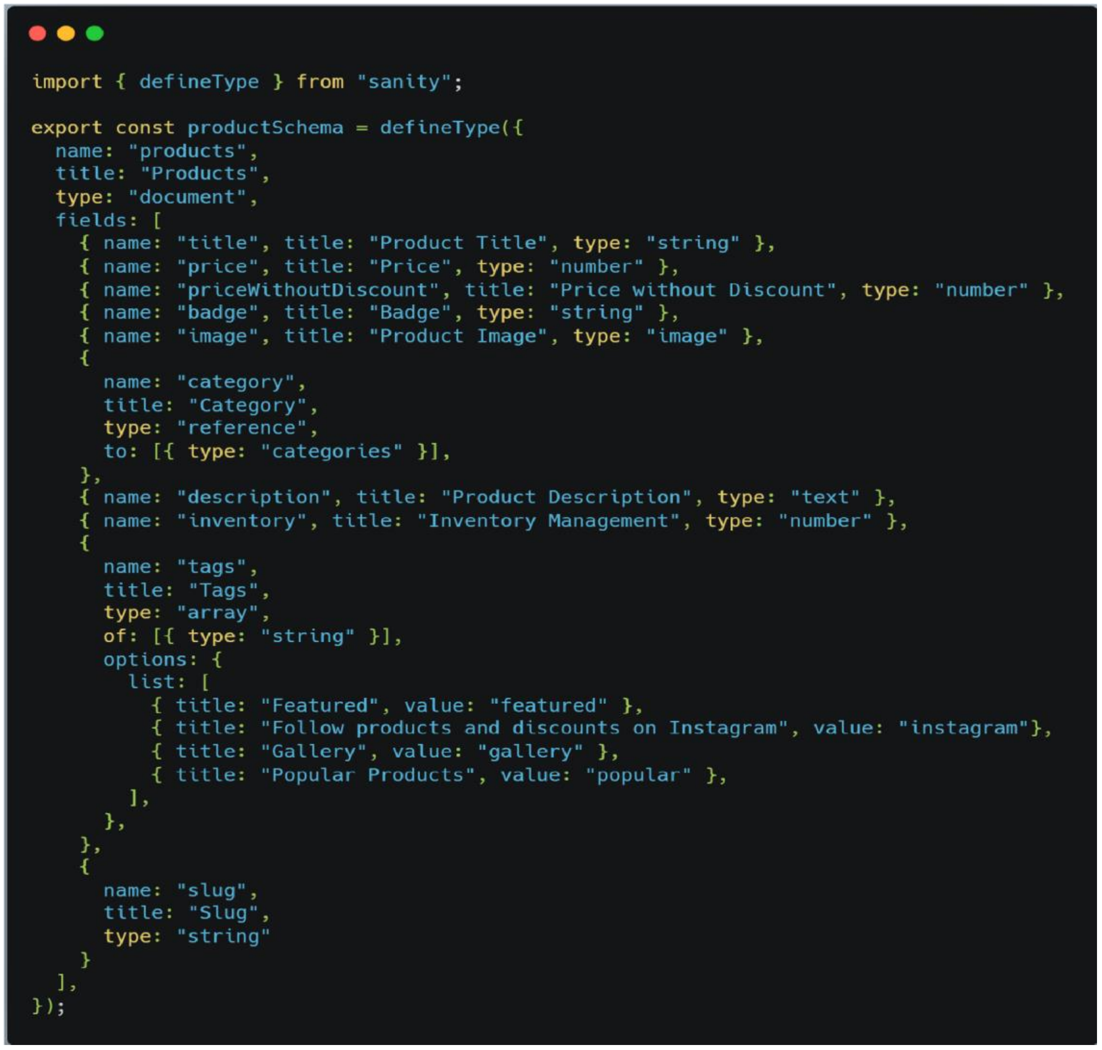
2.2. Frontend Integration

- Data from Sanity CMS was dynamically fetched and rendered in Next.js components.
- Incorporated error handling and fallback UI elements for a better user experience.

3. Sanity Schema Overview

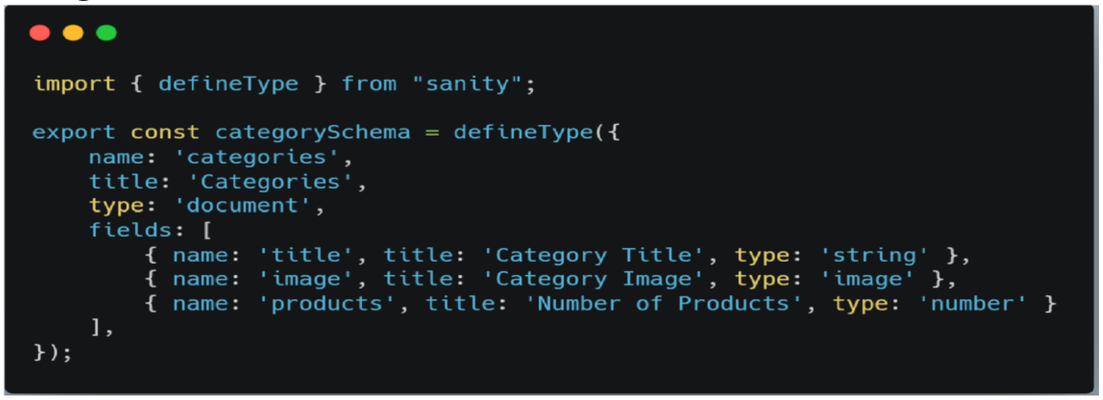
The following schemas were used:

3.1. Products Schema



```
import { defineType } from "sanity";

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    { name: "title", title: "Product Title", type: "string" },
    { name: "price", title: "Price", type: "number" },
    { name: "priceWithoutDiscount", title: "Price without Discount", type: "number" },
    { name: "badge", title: "Badge", type: "string" },
    { name: "image", title: "Product Image", type: "image" },
    {
      name: "category",
      title: "Category",
      type: "reference",
      to: [{ type: "categories" }],
    },
    { name: "description", title: "Product Description", type: "text" },
    { name: "inventory", title: "Inventory Management", type: "number" },
    {
      name: "tags",
      title: "Tags",
      type: "array",
      of: [{ type: "string" }],
      options: {
        list: [
          { title: "Featured", value: "featured" },
          { title: "Follow products and discounts on Instagram", value: "instagram"},
          { title: "Gallery", value: "gallery" },
          { title: "Popular Products", value: "popular" },
        ],
      },
    },
    {
      name: "slug",
      title: "Slug",
      type: "string"
    }
  ],
});
```

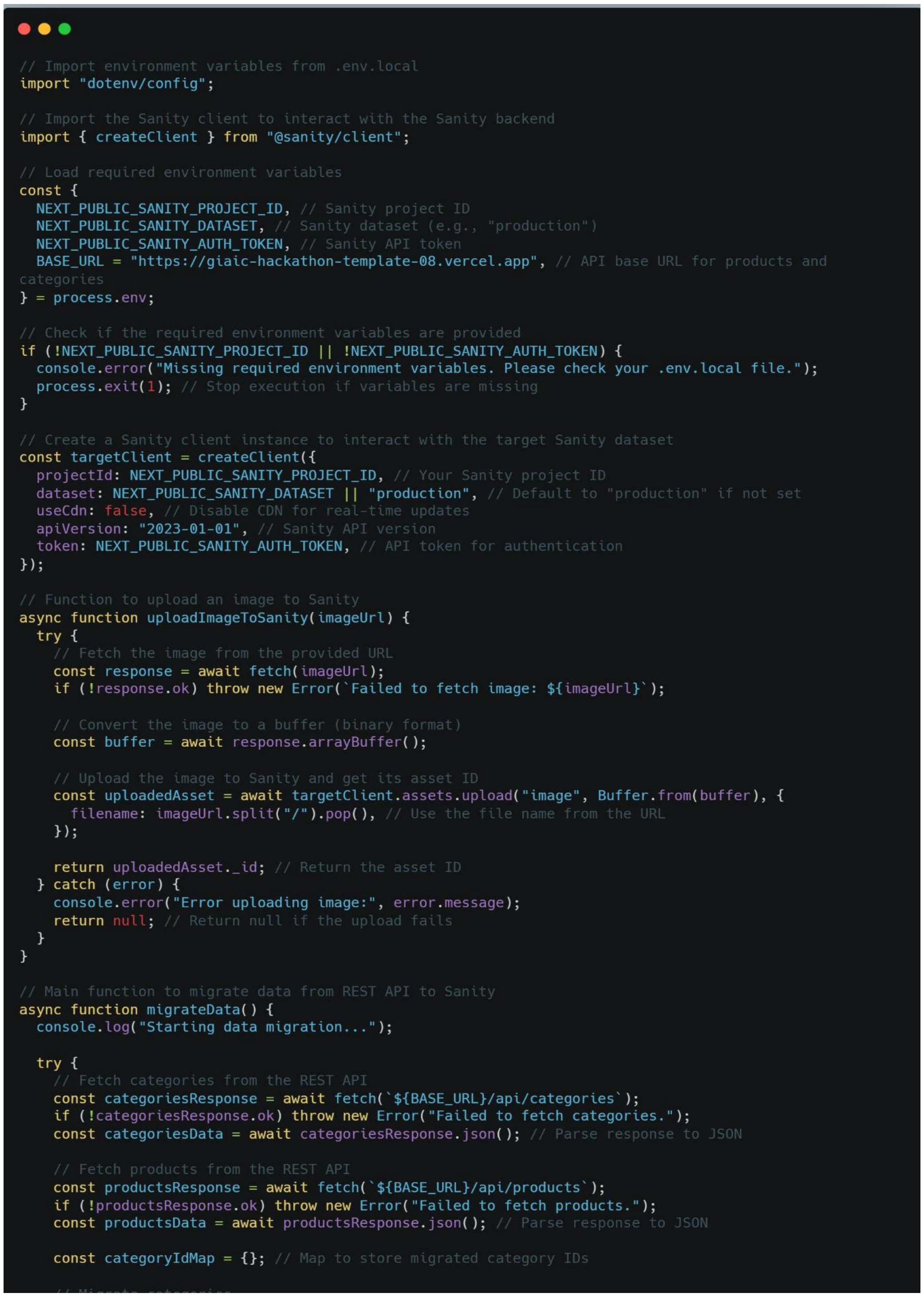
3.2. Categories Schema



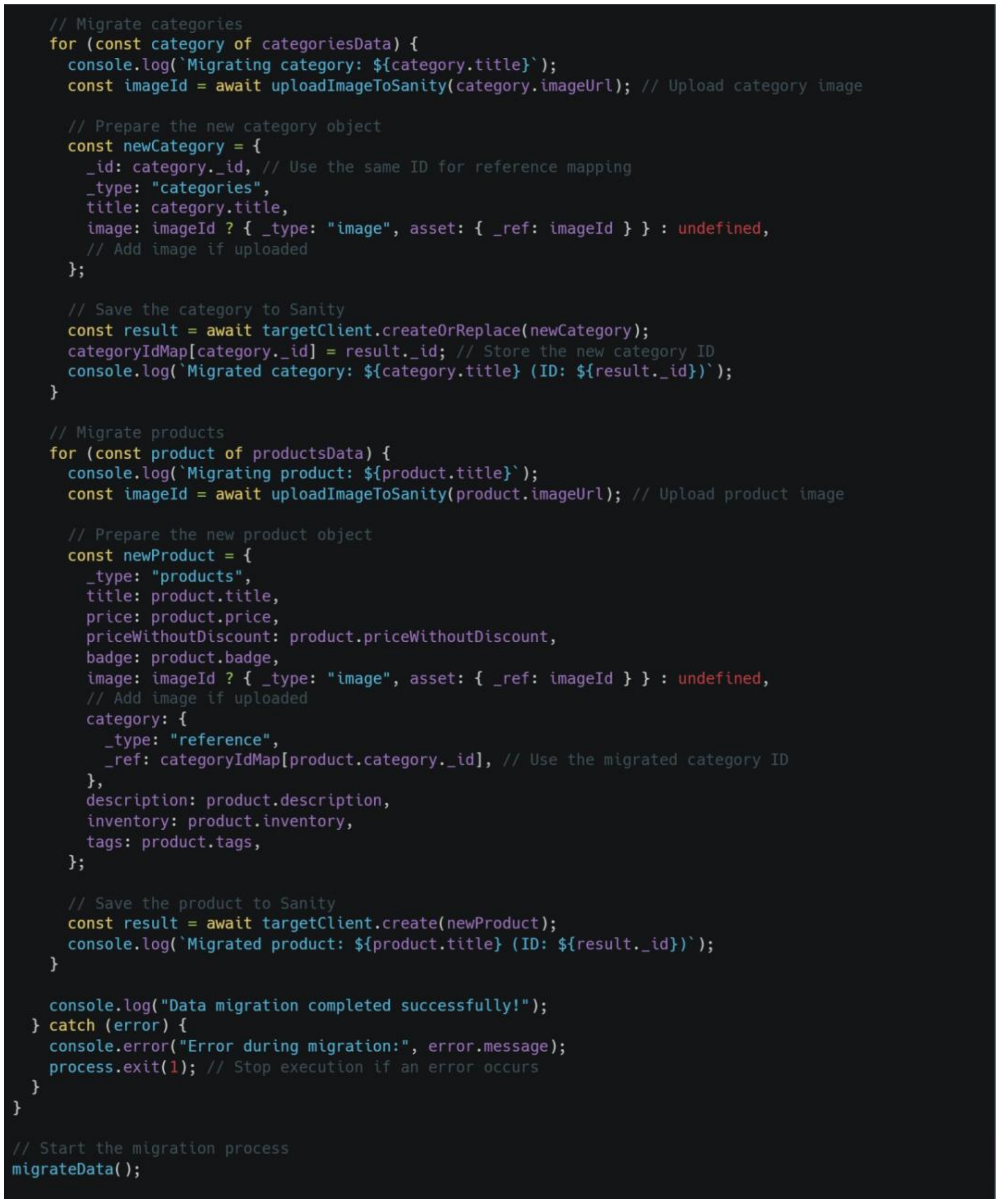
```
import { defineType } from "sanity";

export const categorySchema = defineType({
    name: 'categories',
    title: 'Categories',
    type: 'document',
    fields: [
        { name: 'title', title: 'Category Title', type: 'string' },
        { name: 'image', title: 'Category Image', type: 'image' },
        { name: 'products', title: 'Number of Products', type: 'number' }
    ],
});
```

4. Data Migration Steps

4.1 Migration Script

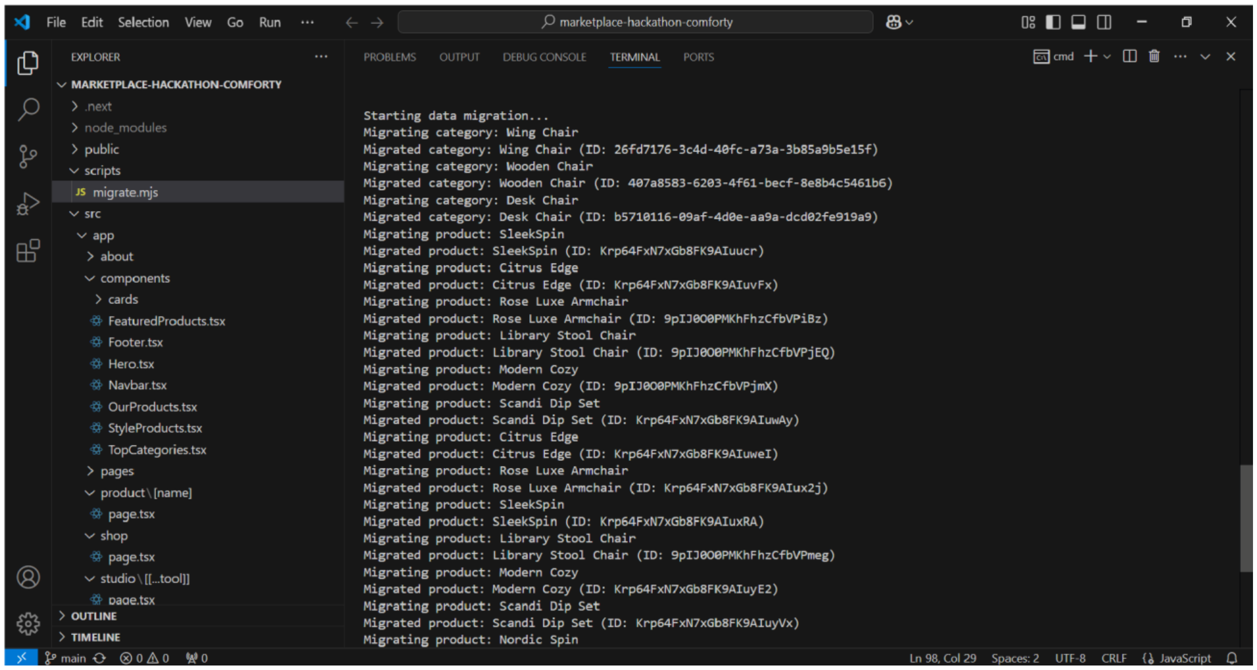


```javascript
// Import environment variables from .env.local
import "dotenv/config";

// 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
```



```javascript
    // 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();
```

4.2. Migration Process

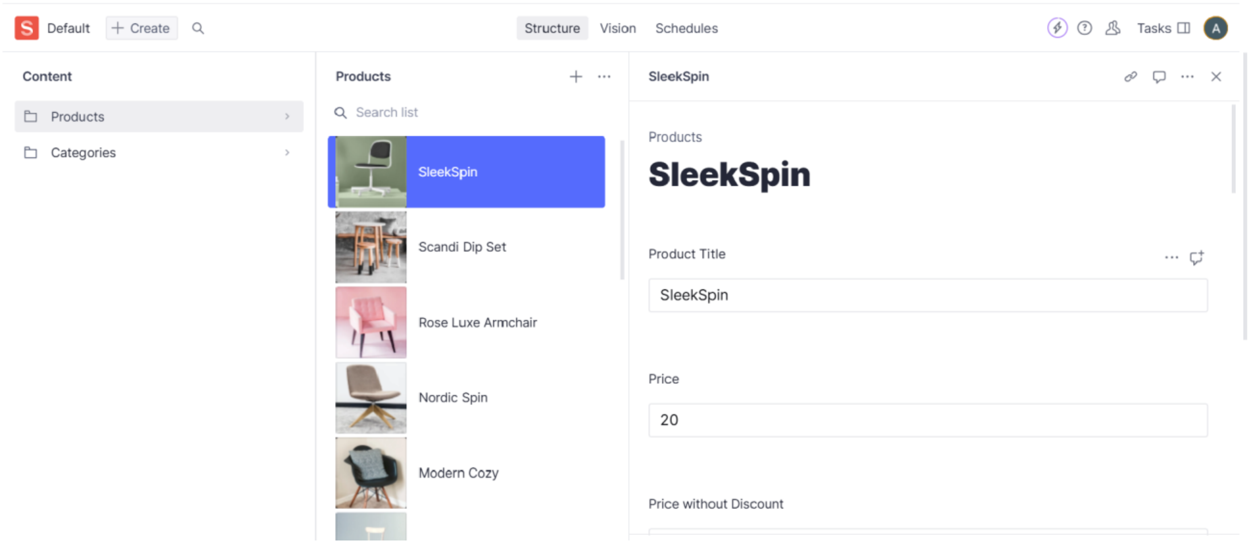


```
Starting data migration...
Migrating category: Wing Chair
Migrated category: Wing Chair (ID: 26fd7176-3c4d-40fc-a73a-3b85a9b5e15f)
Migrating category: Wooden Chair
Migrated category: Wooden Chair (ID: 407a8583-6203-4f61-becf-8e8b4c5461b6)
Migrating category: Desk Chair
Migrated category: Desk Chair (ID: b5710116-09af-4d0e-aa9a-dcd02fe919a9)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: Krp64FxN7xGb8FK9AIuucr)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: Krp64FxN7xGb8FK9AIuvFx)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: 9pIJ0O0PMKhFhzCfbVPiBz)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: 9pIJ0O0PMKhFhzCfbVPjEQ)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: 9pIJ0O0PMKhFhzCfbVPjmX)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: Krp64FxN7xGb8FK9AIuwAy)
Migrating product: Citrus Edge
Migrated product: Citrus Edge (ID: Krp64FxN7xGb8FK9AIuweI)
Migrating product: Rose Luxe Armchair
Migrated product: Rose Luxe Armchair (ID: Krp64FxN7xGb8FK9AIux2j)
Migrating product: SleekSpin
Migrated product: SleekSpin (ID: Krp64FxN7xGb8FK9AIuxRA)
Migrating product: Library Stool Chair
Migrated product: Library Stool Chair (ID: 9pIJ0O0PMKhFhzCfbVPmeg)
Migrating product: Modern Cozy
Migrated product: Modern Cozy (ID: Krp64FxN7xGb8FK9AIuyE2)
Migrating product: Scandi Dip Set
Migrated product: Scandi Dip Set (ID: Krp64FxN7xGb8FK9AIuyVx)
Migrating product: Nordic Spin
```

5. Expected Output and Results

5.1. Sanity CMS

- Populated with products from the Template 8 API.
- Fields such as name, price, description, and image correctly mapped and displayed.
- Screenshot of populated Sanity CMS:



SleekSpin

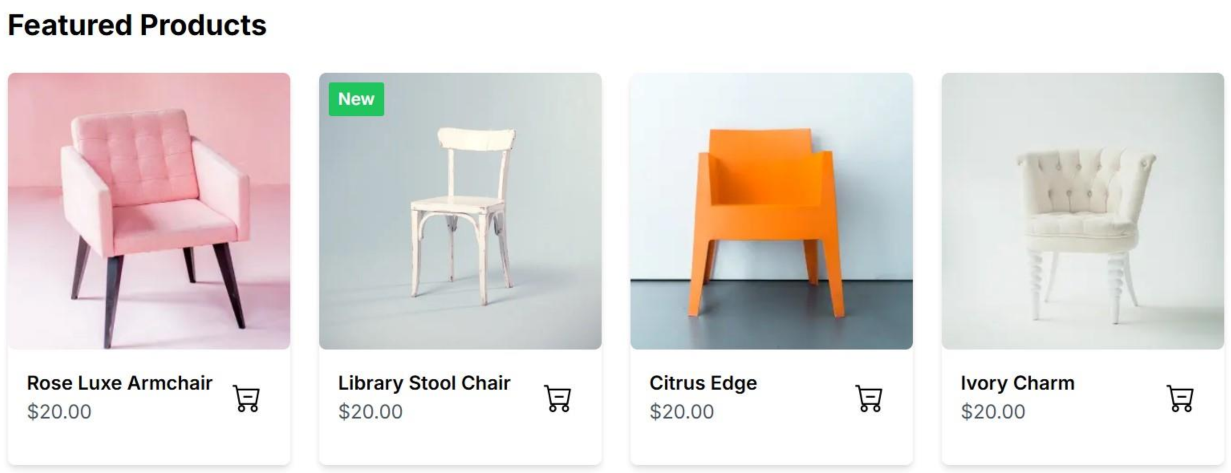
Product Title: SleekSpin

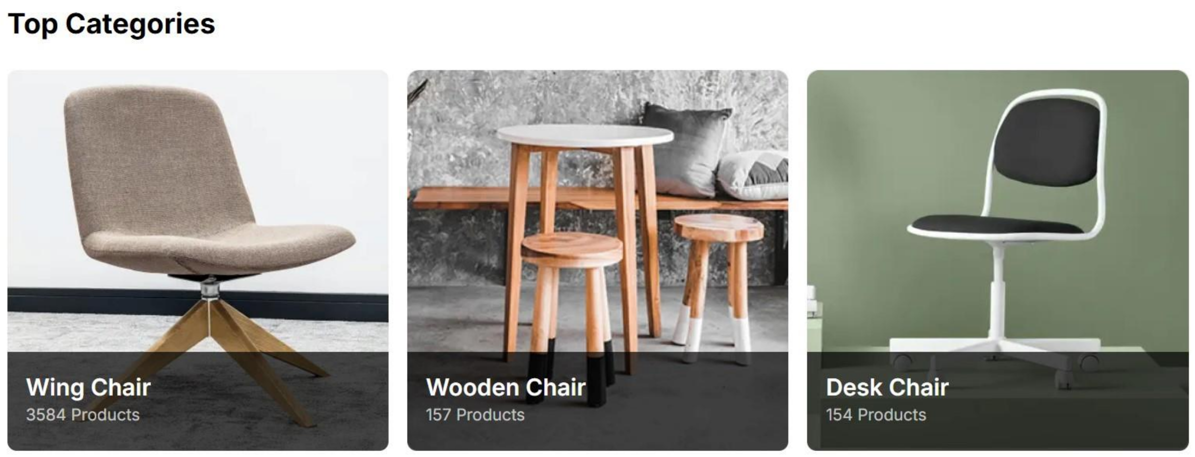
Price: 20

Price without Discount

5.2. Frontend Display

- Products listed dynamically using the data from Sanity.
- Categories and additional metadata rendered accurately. ● Screenshots of frontend display:





6. Conclusion

This report demonstrates the successful integration of the Template 8 API and data migration into the Sanity CMS for Comforty. All tasks were completed in compliance with the Day 3 requirements, ensuring a robust and functional marketplace backend.