Day 3 - API Integration and Data Migration

Final Report

Building **Comforty**, a furniture marketplace, and the focus is on:

1. Integrating **Sanity** to manage product and category data.
2. Setting up Sanity schemas for **products** and **categories**.
3. Migrating data to Sanity using a script.

1. Sanity Installation

Step-by-Step

1. Follow the Sanity Setup Guide to install Sanity.
2. Sanity project or create a new one for Comforty.
3. Decide the project name.
4. Select the **production** dataset.
5. Select sanity studio
6. Path selection for sanity studio is /studio

- Creating a New Token
- Created a new API token in the Sanity dashboard.
- Added the token to the .env & .env.local file.
- Working on Template 8: Template 8 - https://docs.google.com/document/d/1tg3wdRvcGnEcyVRyzVpXzI8tbPClD_Z9mfFrq1vhkns/edit?usp=sharing

Following codes are used on VS Code Terminal

npm create sanity@latest

npm install –g @sanity/cli

sanity init

2. Sanity Schema Setup

You'll create schemas for `products` and `categories`.

File Structure

- Create these files:

src/sanity/schemaTypes/products.ts

src/sanity/schemaTypes/categories.ts

Code for Schema (products.ts) :

```
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",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      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" },
      ],
    },
  },
],
});
```

Code for Schema (categories.ts) :

```
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',
      },
      {
         title: 'Number of Products',
         name: 'products',
         type: 'number',
      }
   ],
});
```

Import Schemas in index.ts :

```
import { type SchemaTypeDefinition } from "sanity";
import { productSchema } from "./products";
import { categorySchema } from "./categories";

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [productSchema, categorySchema],
};
```

3. Data Migration Script

Script to transfer data from REST APIs to Sanity.

Steps to Set Up

1. **Create** .env **&** .env.local **Files**:

In project root directory, create files named **.env & .env.local** and add:

```
NEXT_PUBLIC_SANITY_PROJECT_ID="wvuuekqg"
NEXT_PUBLIC_SANITY_DATASET="production"
NEXT_PUBLIC_SANITY_AUTH_TOKEN="token (Pivate & cofidential)”
```

2. **Create** migrate.mjs **File**:

Create a **scripts** folder in Root directory.

Inside the folder, create migrate.mjs

Code for migrate.mjs :

```
// 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();
```

1. **Add a Script to** package.json file

   Open package.json and add this under "scripts":

   "migrate": "node scripts/migrate.mjs

2. **Install** dotenv **Package**:

   npm install dotenv

3. **Run the Migration Script**:

   Execute the command to migrate data **npm run migrate**

Rest API endpoint for Details: (Provided in Document)

Products API

- Endpoint: https://giaic-hackathon-template-08.vercel.app/api/products

Categories API:

- Endpoint: https://giaic-hackathon-template-08.vercel.app/api/categories

Creating Files

1 – ProductListFromAPI.tsx (in the components folder)

2- api.ts (in the lib folder)

1 – ProductListFromAPI.tsx (in the components folder)

Coding:

```
import { useState, useEffect } from 'react'
import { fetchProducts, fetchCategories } from '../lib/api'
import Image from 'next/image'

interface Product {
  id: string;
  name: string;
  description: string;
  price: number;
  category: string;
```

```
  image: string;
  stock: number;
  rating: number;
}

interface Category {
  id: string;
  name: string;
  description: string;
}

export default function ProductListFromAPI() {
  const [products, setProducts] = useState<Product[]>([])
  const [categories, setCategories] = useState<Category[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)

  useEffect(() => {
    async function loadData() {
      try {
        const [productsData, categoriesData] = await Promise.all([
          fetchProducts(),
          fetchCategories()
        ])
        setProducts(productsData)
        setCategories(categoriesData)
      } catch (err) {
        setError('Failed to load data')
      } finally {
        setLoading(false)
      }
    }

    loadData()
  }, [])

  if (loading) return <div>Loading products and categories...</div>
  if (error) return <div>Error: {error}</div>

  return (
    <div>
      <h2 className="text-2xl font-bold mb-4">Categories</h2>
      <ul className="mb-8">
        {categories.map((category) => (
          <li key={category.id} className="mb-2">
```

```
        <h3 className="text-lg font-semibold">{category.name}</h3>
        <p className="text-gray-600">{category.description}</p>
      </li>
    ))}
  </ul>

  <h2 className="text-2xl font-bold mb-4">Products</h2>
  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
    {products.map((product) => (
      <div key={product.id} className="border rounded-lg p-4 shadow-md">
        <Image
          src={product.image || "/placeholder.svg"}
          alt={product.name}
          width={300}
          height={300}
          className="w-full h-48 object-cover mb-4 rounded"
        />
        <h2 className="text-xl font-semibold mb-2">{product.name}</h2>
        <p className="text-gray-600 mb-2">{product.description}</p>
        <p className="text-lg font-bold mb-2">${product.price.toFixed(2)}</p>
        <p className="text-sm text-gray-500 mb-2">Category: {product.category}</p>
        <p className="text-sm text-gray-500 mb-2">Stock: {product.stock}</p>
        <div className="flex items-center">
          <span className="text-yellow-400 mr-1">★</span>
          <span>{product.rating.toFixed(1)}</span>
        </div>
      </div>
    ))}
  </div>
</div>
)
}
```

2- api.ts (in the lib folder)

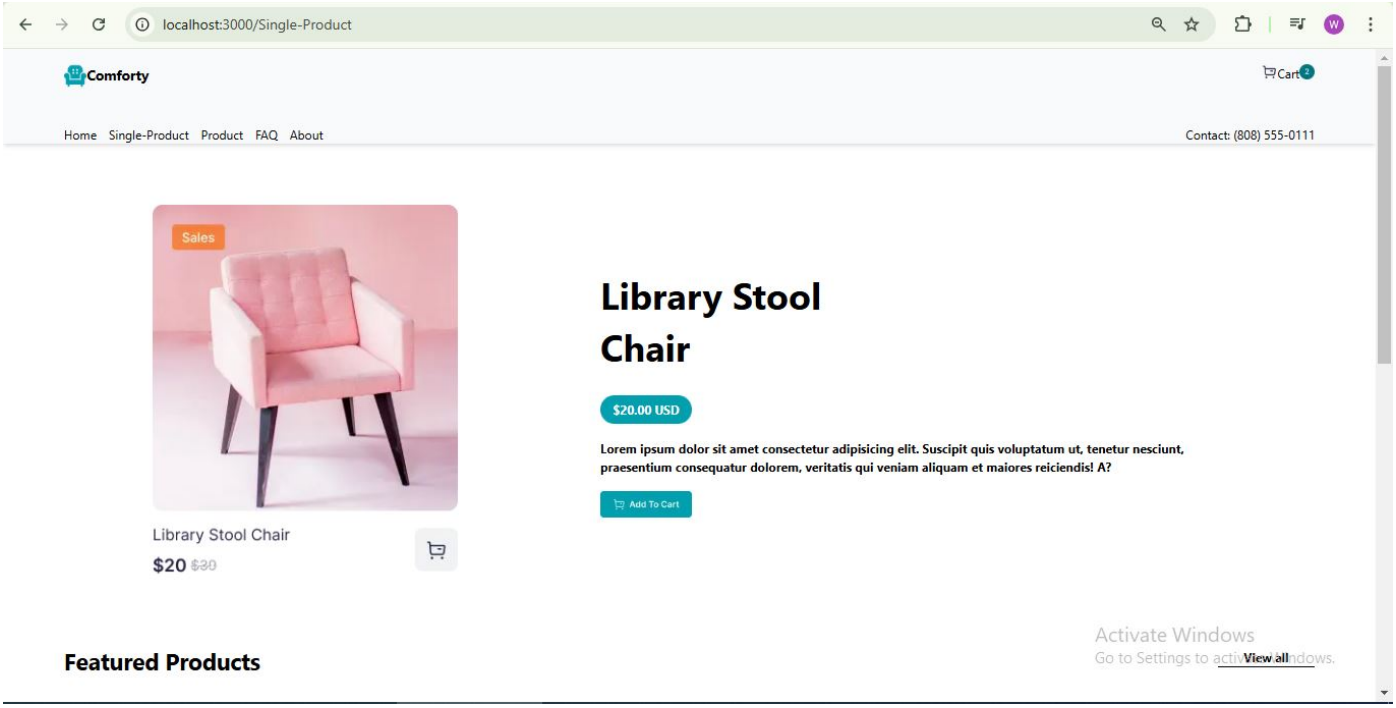
<u>Coding:</u>

```
export async function fetchProducts() {

  try {
```

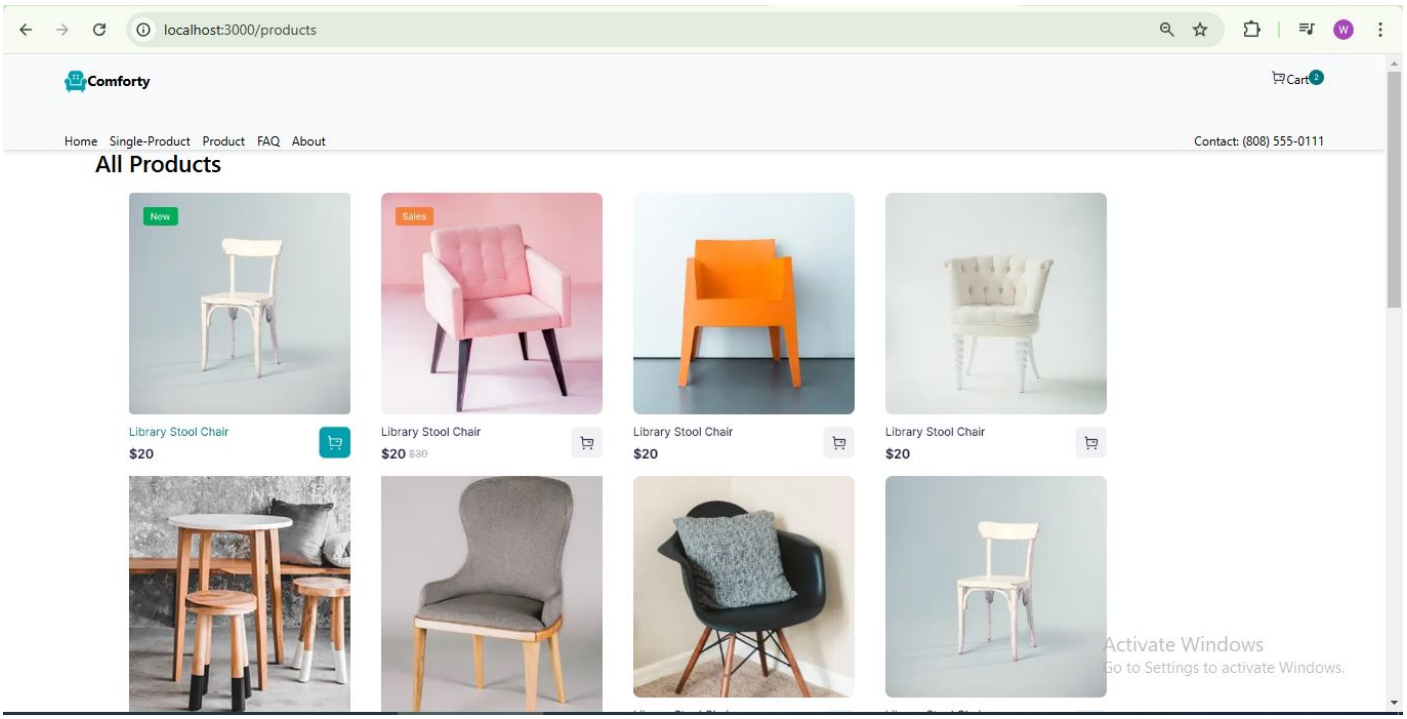
```
    const response = await fetch('https://giaic-hackathon-template-08.vercel.app/api/products');
    if (!response.ok) {
      throw new Error('Failed to fetch products');
    }
    return await response.json();
  } catch (error) {
    console.error('Error fetching products:', error);
    throw error;
  }
}

export async function fetchCategories() {
  try {
    const response = await fetch('https://giaic-hackathon-template-
08.vercel.app/api/categories');
    if (!response.ok) {
      throw new Error('Failed to fetch categories');
    }
    return await response.json();
  } catch (error) {
    console.error('Error fetching categories:', error);
    throw error;
  }
}
```

OUTPUT ON THE BROWSER

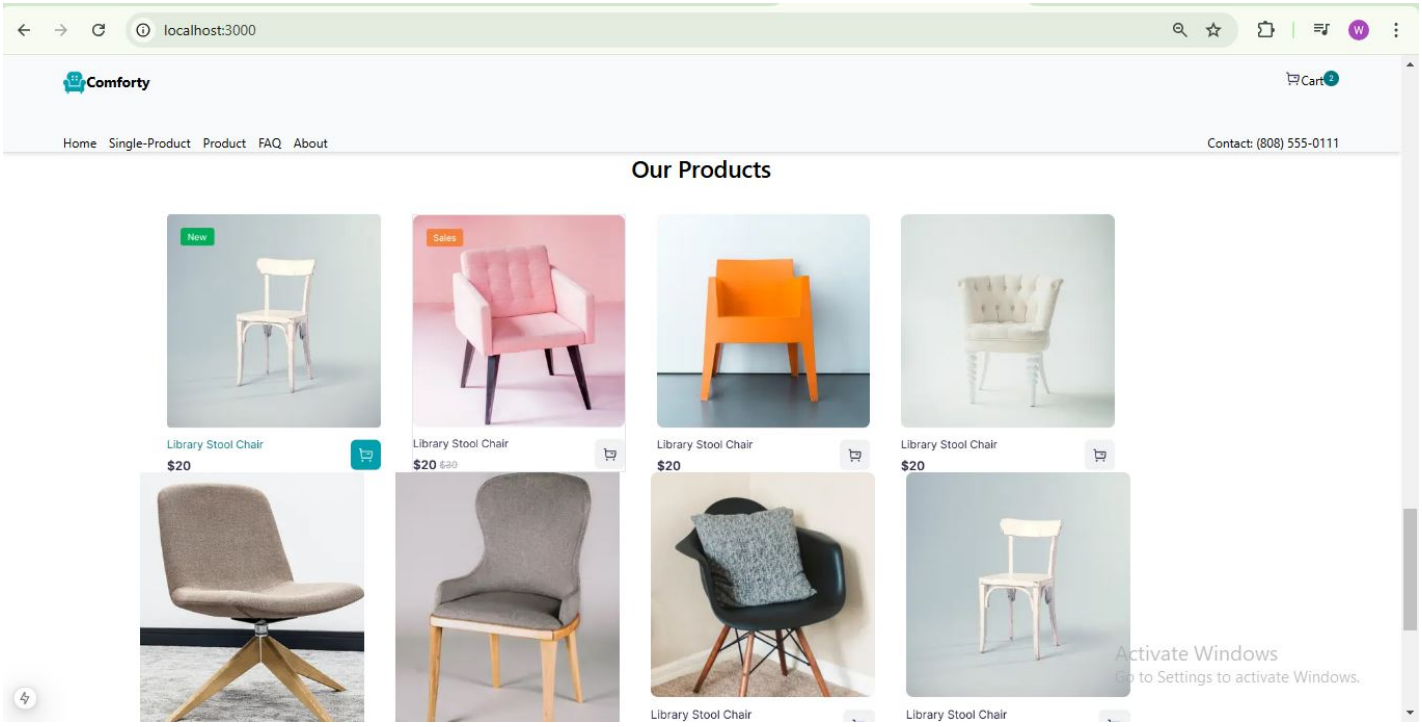
SINGLE PRODUCT PAGE



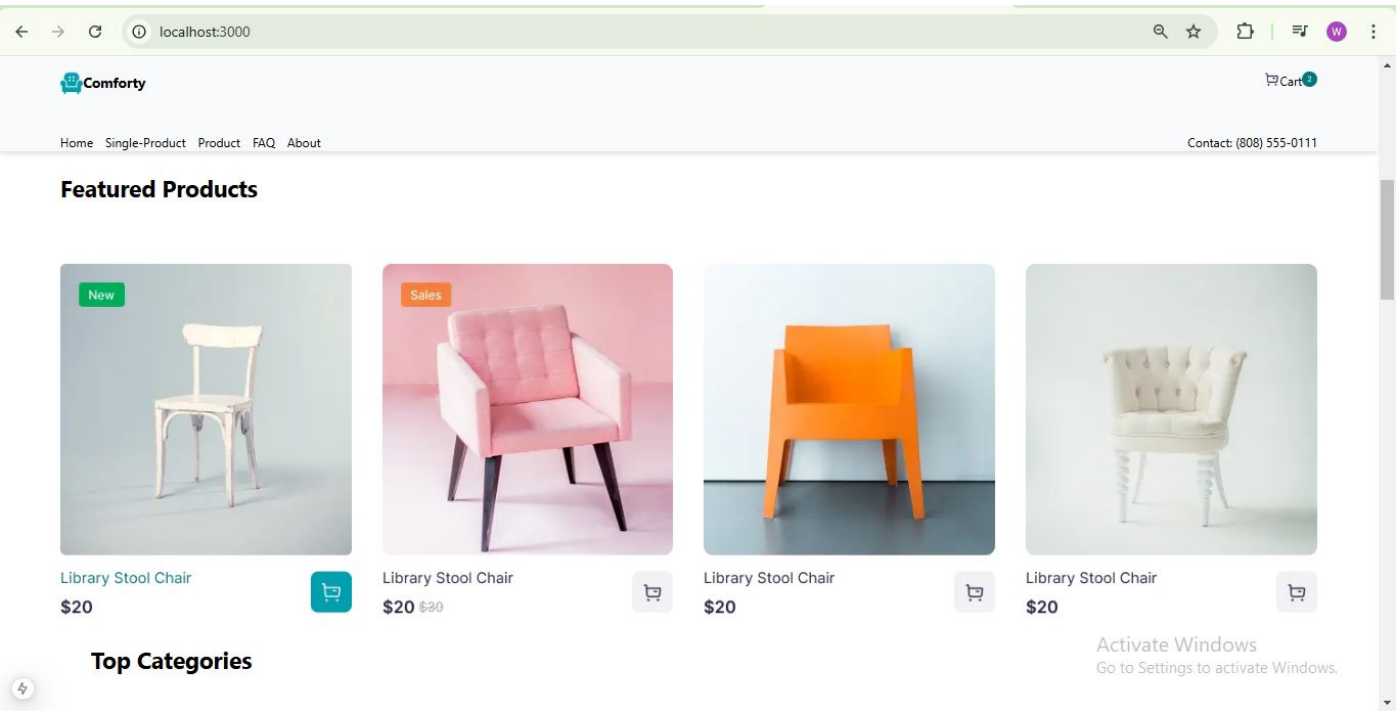
Product Page



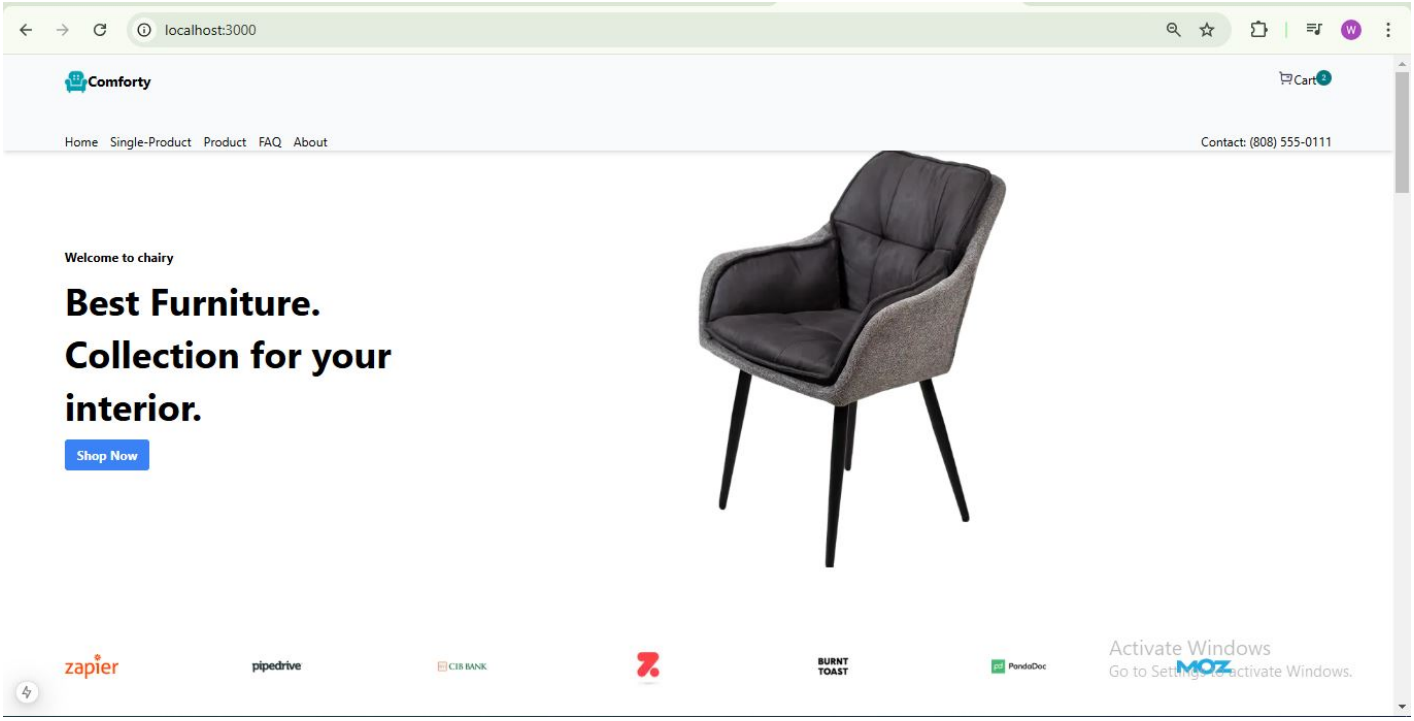
Home Page 3



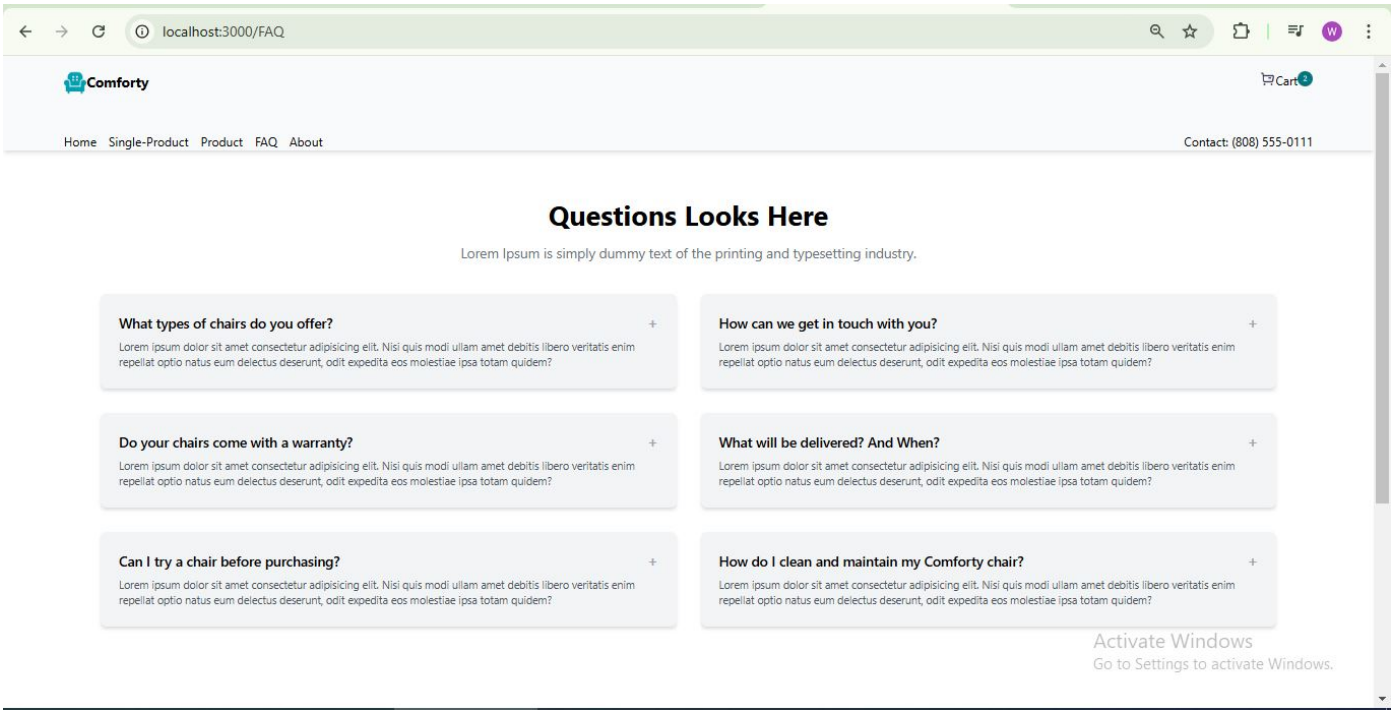
Home Page 2


localhost:3000
Comforty
Cart 2
Home Single-Product Product FAQ About
Contact: (808) 555-0111
Featured Products
New
Sales
Library Stool Chair
$20
Library Stool Chair
$20 $30
Library Stool Chair
$20
Library Stool Chair
$20
Top Categories
Activate Windows
Go to Settings to activate Windows.

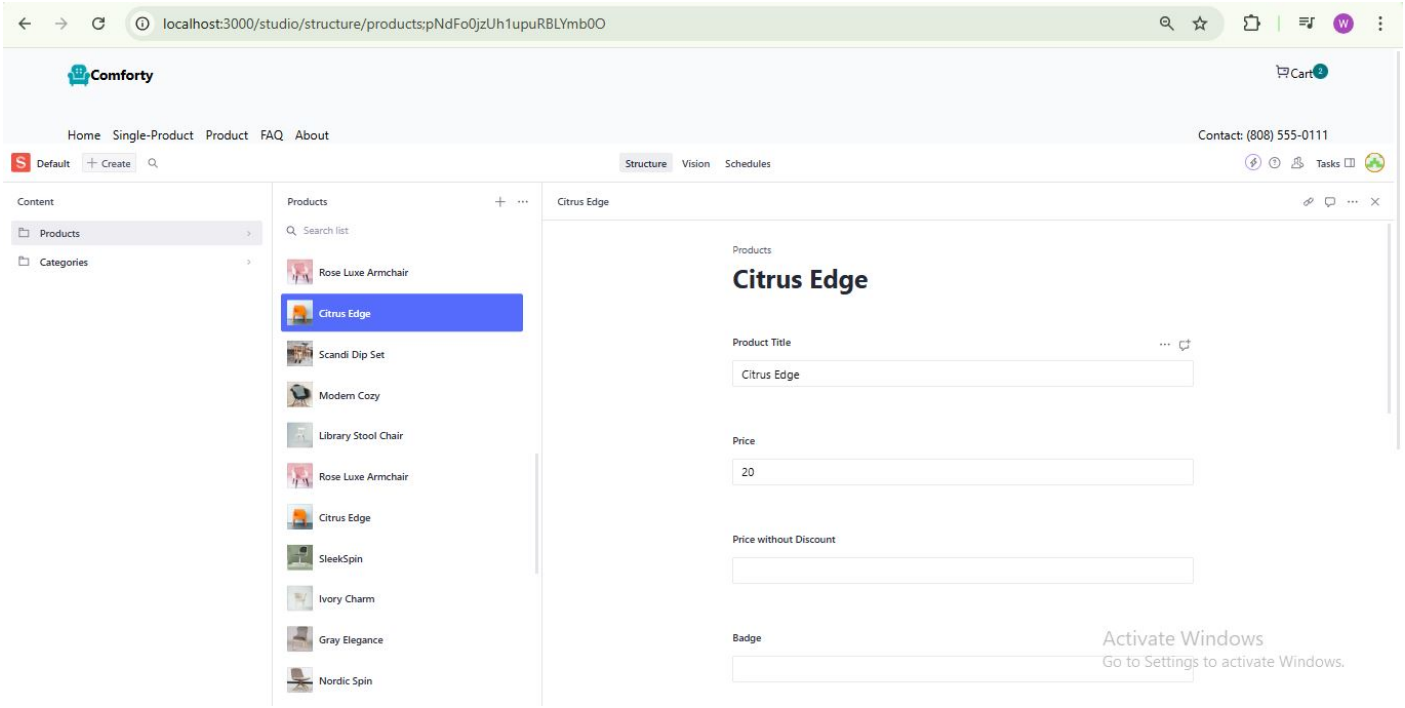
Home Page 1



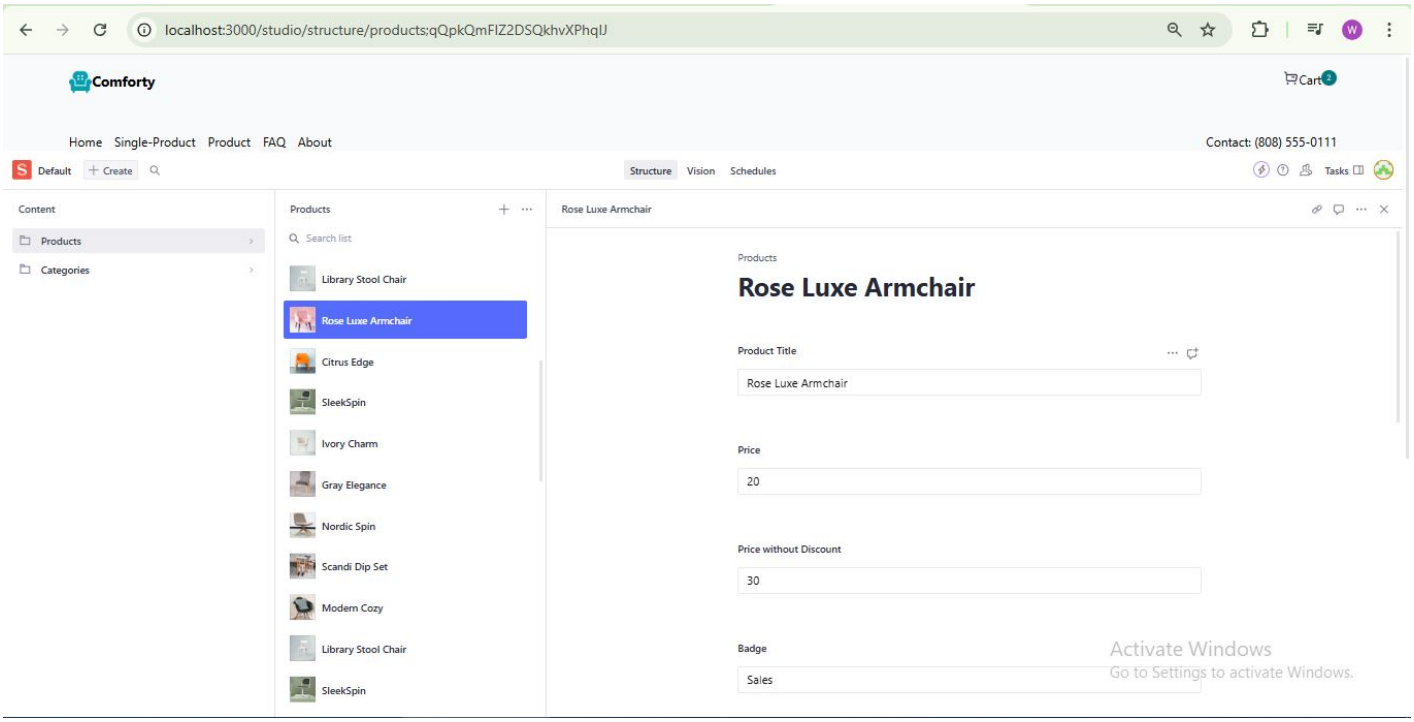
FAQ Page



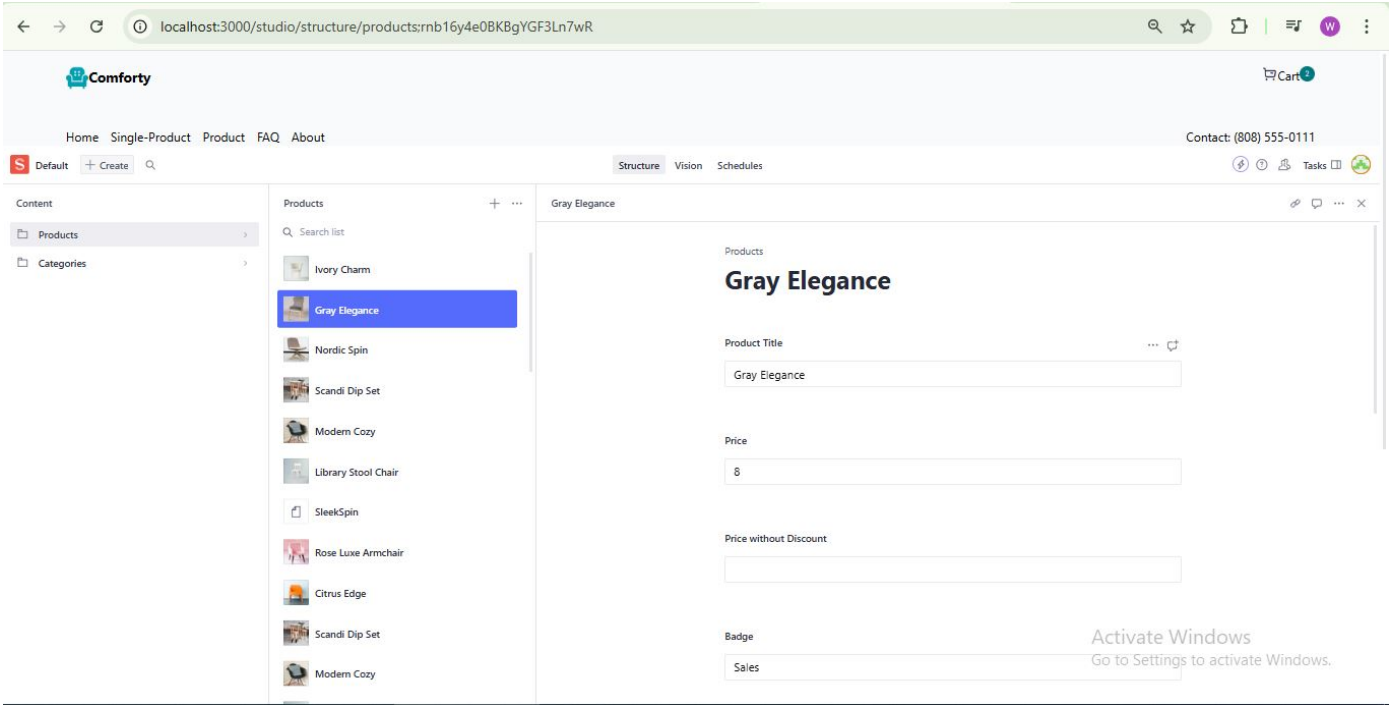
Content Products Page 3



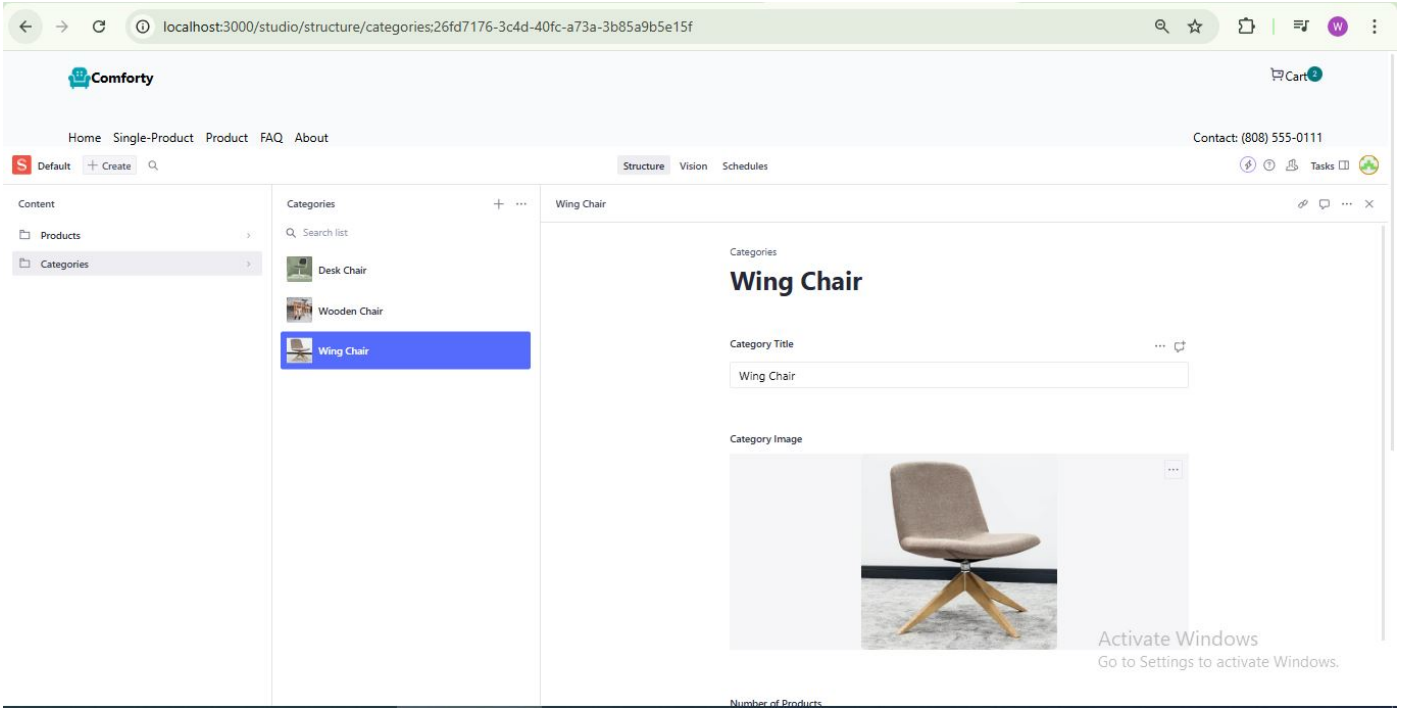
Content Products Page 2



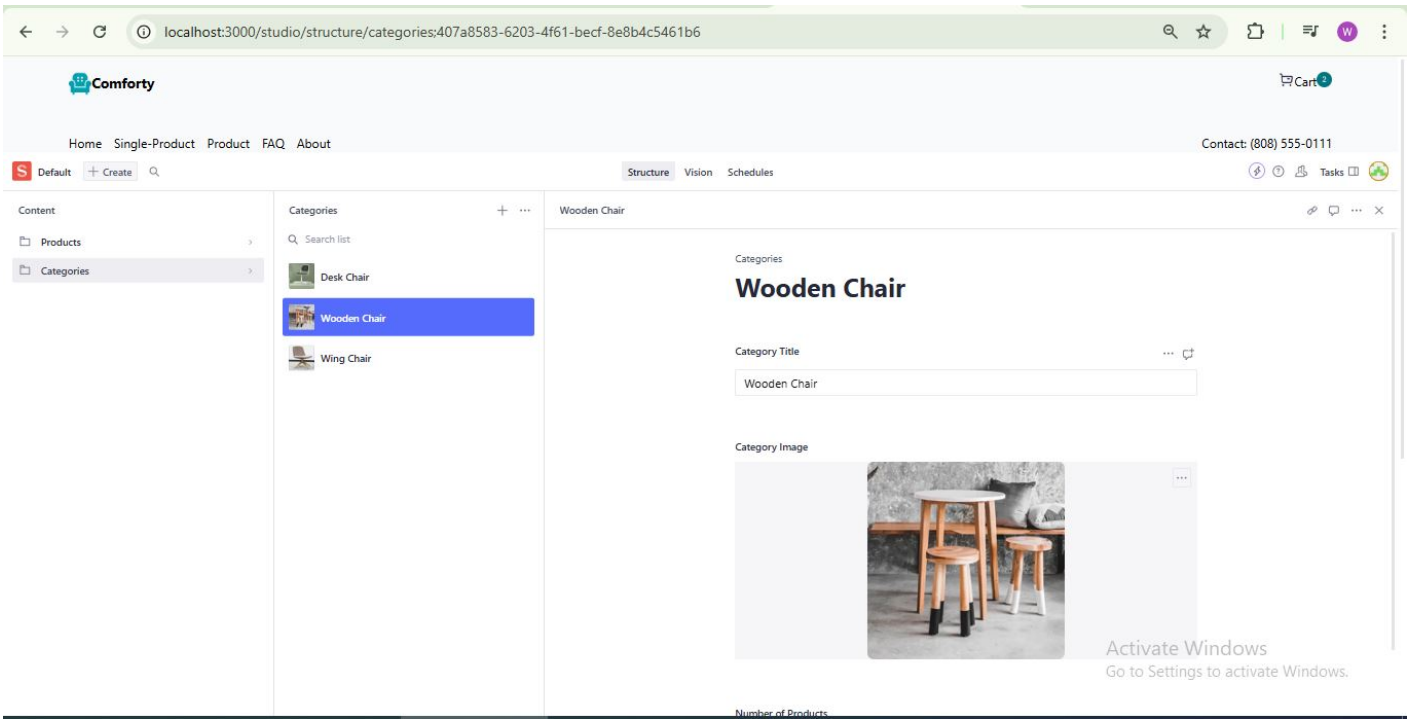
Content Products Page 1



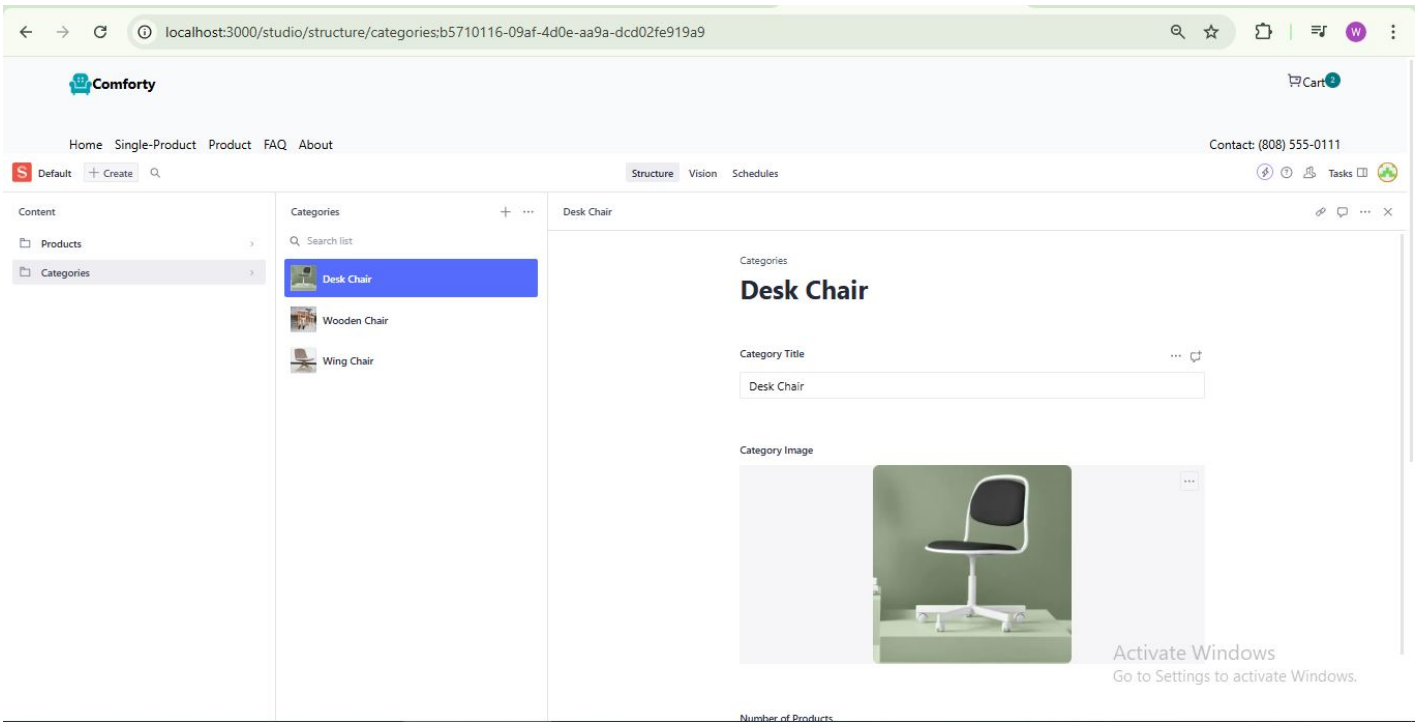
Content Categories Page 3



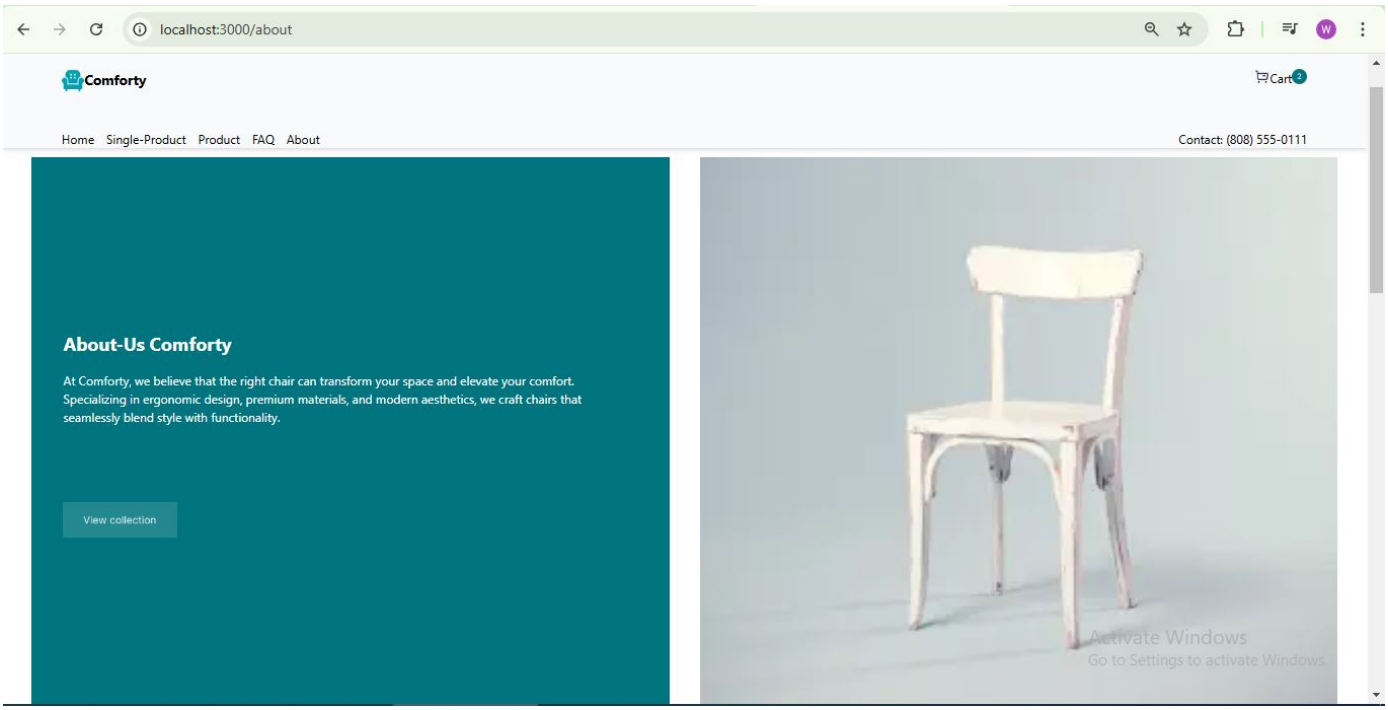
Content Categories Page 2



Content Categories Page 1



About Page


localhost:3000/about
Comforty
Cart 2
Home Single-Product Product FAQ About
Contact: (808) 555-0111
About-Us Comforty
At Comforty, we believe that the right chair can transform your space and elevate your comfort. Specializing in ergonomic design, premium materials, and modern aesthetics, we craft chairs that seamlessly blend style with functionality.
View collection
Activate Windows
Go to Settings to activate Windows.

Day 3 Checklist: Self-Validation

Checklist: API Understanding: • ✔

Schema Validation: • ✔

Data Migration: • ✔

API Integration in Next.js: • ✔

Submission Done