HACKATHON-03
DAY-03
My MarketPlace Name [Costco Club]

A report documenting
- API INTEGRATION PROCESS :
1. Install Sanity Client
First, install the Sanity client library in your project.
npm install @sanity/client
2. Configure the Sanity Client
Create a sanity.js file to configure the client with your Sanity project details (project ID, dataset, and API version).

```
import { createClient } from "next-sanity";

import { apiVersion } from "../env";

export const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: true,
  token: process.env.NEXT_PUBLIC_SANITY_TOKEN,
  apiVersion,
});
```

3. Fetch Data with GROQ Query
Write a function to fetch data using a GROQ query.

```
import { client } from "@/sanity/lib/client";
import { NextResponse } from "next/server";


export async function GET() {
  try {
    const data = await client.fetch(`*[_type=="product"]`);

    return  NextResponse.json(data, { status: 200 });
  } catch (error) {
    console.error('Error fetching data from Sanity:', error);
```

```
    return new NextResponse('Error fetching data', { status: 500 });
  }
}
```

- ADJUSTMENTS MADE TO SCHEMA :

```
import { defineType, defineField } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        defineField({
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        }),
        defineField({
            name: "slug",
            type: "slug",
            title: "Slug",
            options: {
              source: "title",
              maxLength: 96,
            },
          },),
        defineField({
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        }),
        defineField({
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        }),
        defineField({
```

```
            name: "quantity",
            title: "Quantity",
            type: "number",
            validation: (rule) => rule.min(0),
          }),
        defineField({
            name: "tags",
            type: "array",
            title: "Tags",
            of:[{
                type: "string"
            }]
        }),
        defineField({
            name: 'description',
            title: 'Description',
            type: 'text',
            description: 'Detailed description of the product',
          }),
          defineField({
            name: 'features',
            title: 'Features',
            type: 'array',
            of: [{ type: 'string' }],
            description: 'List of key features of the product',
          }),
          defineField({
            name: 'dimensions',
            title: 'Dimensions',
            type: 'object',
            fields: [
              { name: 'height', title: 'Height', type: 'string' },
              { name: 'width', title: 'Width', type: 'string' },
              { name: 'depth', title: 'Depth', type: 'string' },
            ],
            description: 'Dimensions of the product',
          }),
    ]
})
```

- MIGRATION STEPS AND TOOLS USED :

1. Sanity Installation

First we have to install sanity by:

npm create sanity@latest

2. Sanity Schema

After the installation Navigate to your schema folder: If you have a src folder, go to /src/sanity/schemaTypes. Otherwise, go to /sanity/schemaTypes.

Than place your sanity schema there.

Don't forget to import schema's in your index.ts file

3. Data Migration Script

- Create .env file and add the following variables:

```
NEXT_PUBLIC_SANITY_PROJECT_ID="Your Id"
NEXT_PUBLIC_SANITY_DATASET="Your DataSet"
NEXT_PUBLIC_SANITY_TOKEN="Your Token”
```

- Create migrate.mjs inside of the script folder and put your migarting data there

- Open `package.json` file and add the following code inside of script

```
    "import-data": "node ./scripts/importData.mjs"
```

- Install

npm install dotenv

- Now run the command

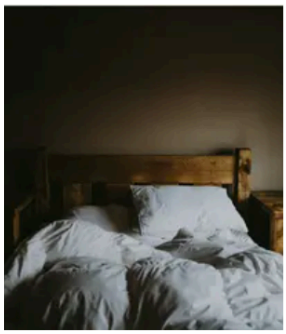
npm run import

Data successfully displayed in the frontend.

Rustic Vase Set

A timeless design, with premium materi…

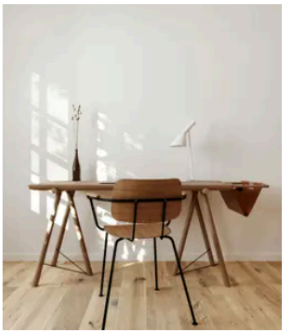
Price: $210



Bed

A timeless design, with premium materi…

Price: $250



Wood Chair

A timeless design, with premium materi…

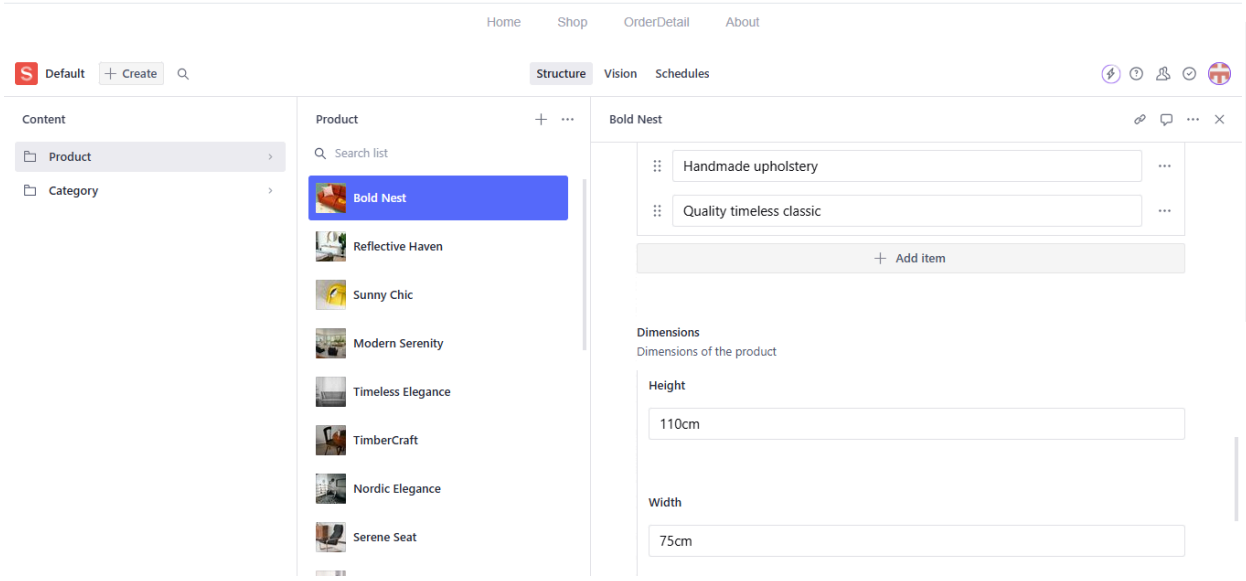
Price: $100

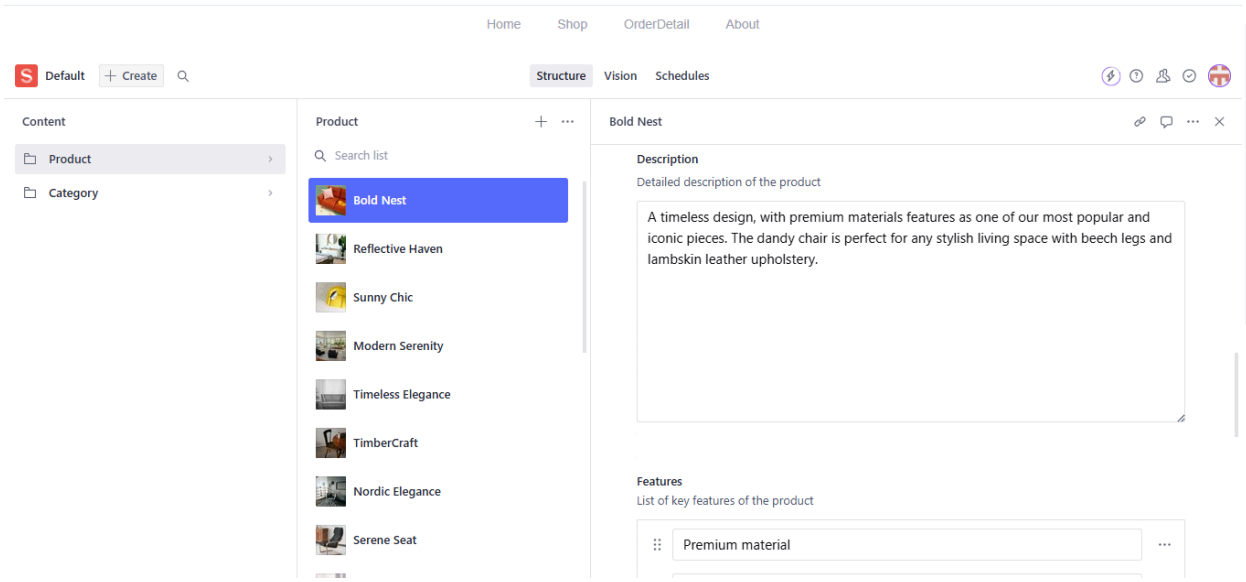
Pure Aura

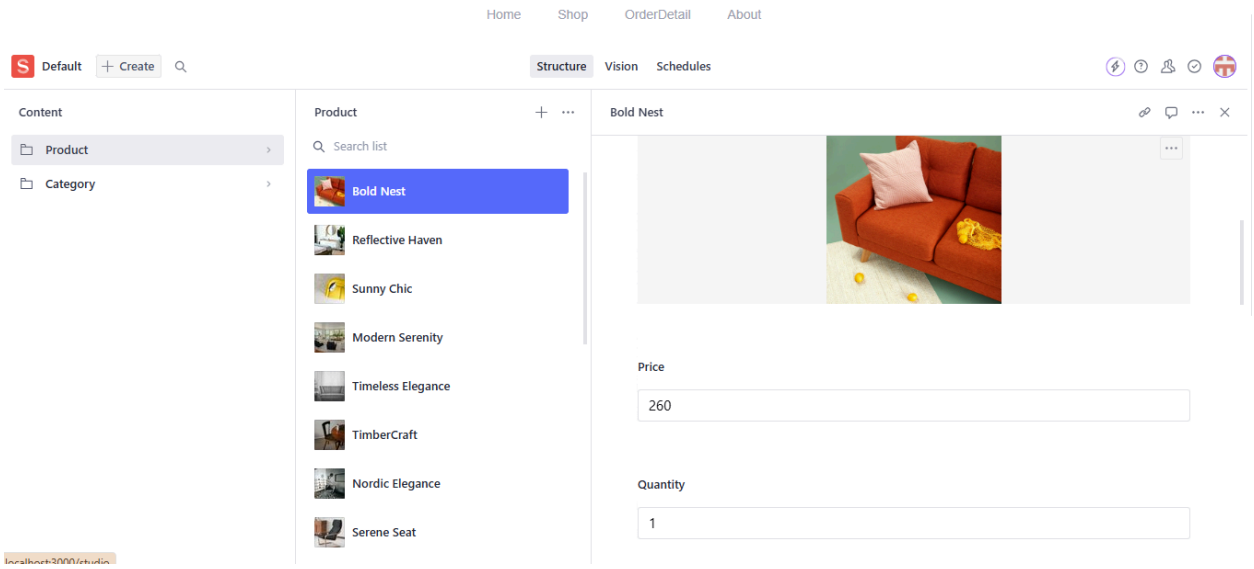
A timeless design, with premium materi…

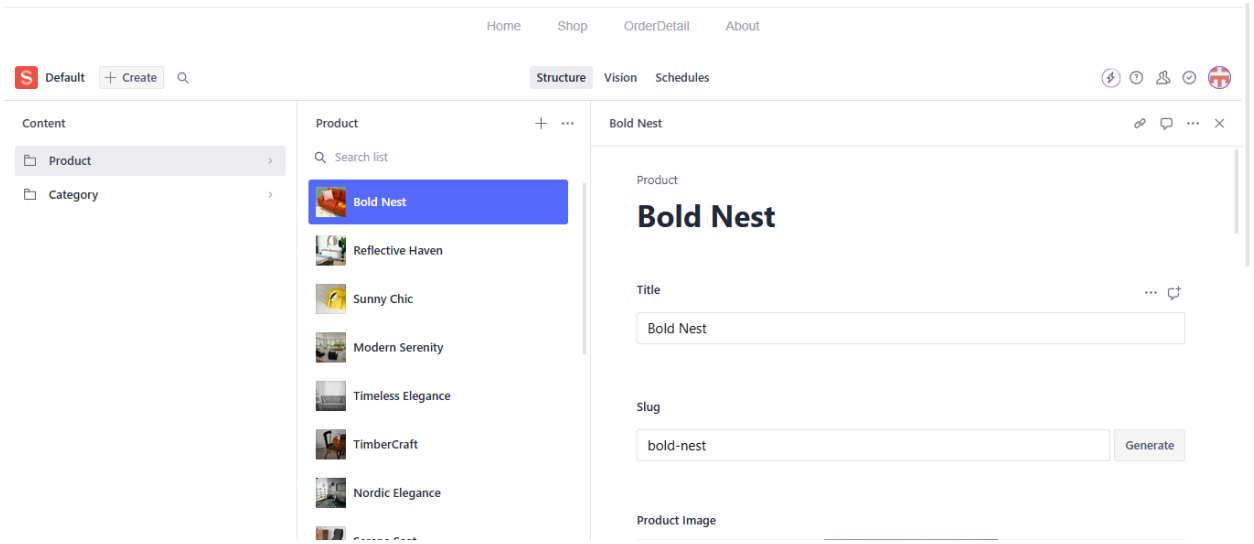
Price: $280

Populated Sanity CMS fields.


Home
Shop
OrderDetail
About
Default
Create
Structure
Vision
Schedules
Content
Product
Category
Product
Search list
Bold Nest
Reflective Haven
Sunny Chic
Modern Serenity
Timeless Elegance
TimberCraft
Nordic Elegance
Serene Seat
Bold Nest
Handmade upholstery
Quality timeless classic
Add item
Dimensions
Dimensions of the product
Height
110cm
Width
75cm


Home
Shop
OrderDetail
About
Default
Create
Structure
Vision
Schedules
Content
Product
Category
Product
Search list
Bold Nest
Reflective Haven
Sunny Chic
Modern Serenity
Timeless Elegance
TimberCraft
Nordic Elegance
Serene Seat
Bold Nest
Description
Detailed description of the product
A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.
Features
List of key features of the product
Premium material





Code snippets for API integration and migration scripts.

```
import { createClient } from '@sanity/client';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
```

```
dotenv.config({ path: path.resolve(__dirname, './.env.local') });


const client = createClient({
  projectId: "vlsu19wp",
  dataset: "production",
  useCdn: false,

token:"skCId6mBzzpblu9A6fhJSz82tqkvw5tLQJzPxyTPXdOmg0SZ5GThiII1EZHW9A9JxlF
iBPl8uqC78wk0Xhup7bmZx9m5R0NwmYsEJj6iqYvrI49qc4E61PqM3BuyG7IKFqdiOZIlTfUOE
TTqTbdh9gtIVHYt2QGp6qMnMZBO1EmkssPk1bkt",
  apiVersion: '2021-08-31',
});

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.image);

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.name,
        description: product.description,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        features: product.features,
        dimensions: product.dimensions,
        category: product.category,
        price: product.price,
        tags: product.tags,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.title} uploaded successfully:`,
createdProduct);
    } else {
      console.log(`Product ${product.title} skipped due to image upload
failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await
fetch('https://hackathon-apis.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
```

```
    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```