

Day 3 - API Integration Report - EcoFurnish

Prepared By: Zija Yaseen

1.API Integration Process:

Provided API Key :<https://template-0-beta.vercel.app/api/product>

EcoFurnish began the API integration by setting up the Next.js application. The steps followed were:

1. Install Next.js

- A new Next.js project was created using the command:

- `npx create-next-app@latest`

2. Sanity CMS Setup Process

Sanity CMS was integrated to manage content and handle API data storage. The setup process included:

Install Sanity CLI

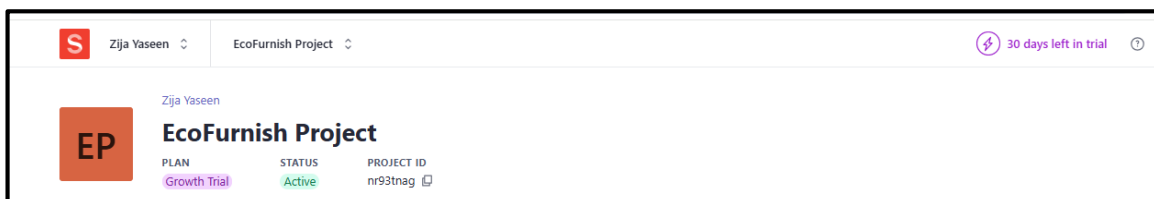
- Sanity was initialized by running the following commands:

- `npm create sanity@latest -- --template clean --create-project "learning-sanity-project" --dataset production`

- For a complete Sanity setup, refer to the documentation available at: [Sanity Setup Guide](#)

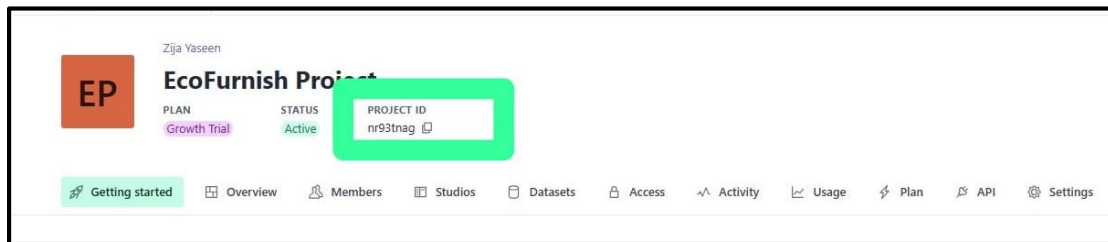
3. Create a Project in Sanity

- After setting up Sanity, a new project was created within the Sanity Studio to manage content effectively.



- **Define Project ID and Dataset in `.env.local`**

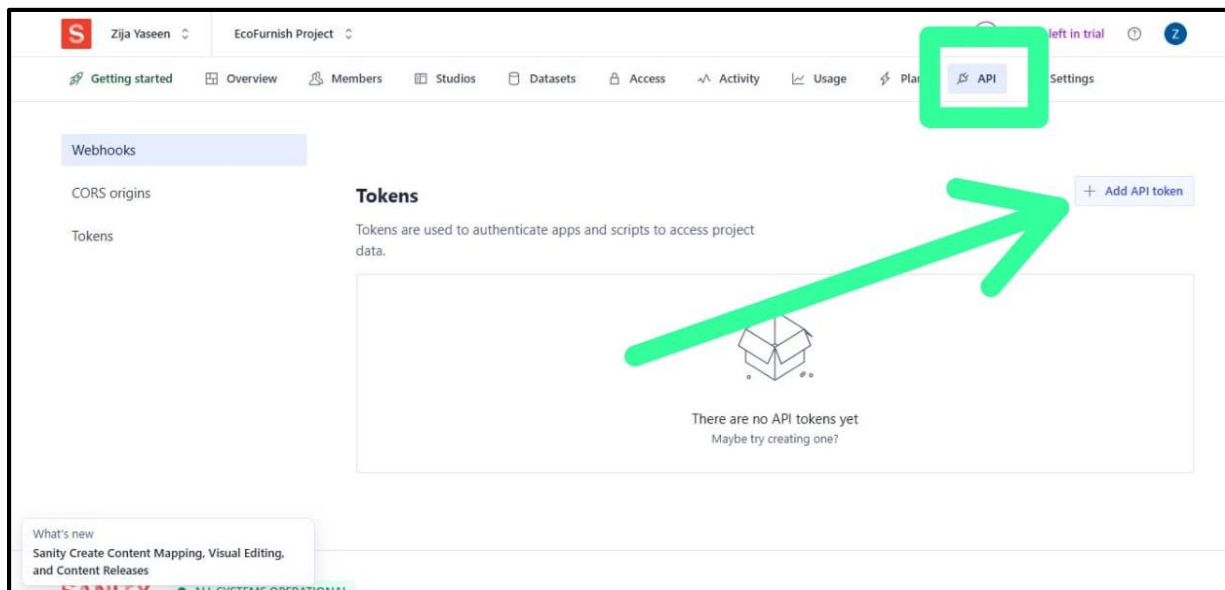
- The `NEXT_PUBLIC_SANITY_PROJECT_ID` and `NEXT_PUBLIC_SANITY_DATASET` were added to the `.env.local` file to securely manage the environment variables required for API integration.



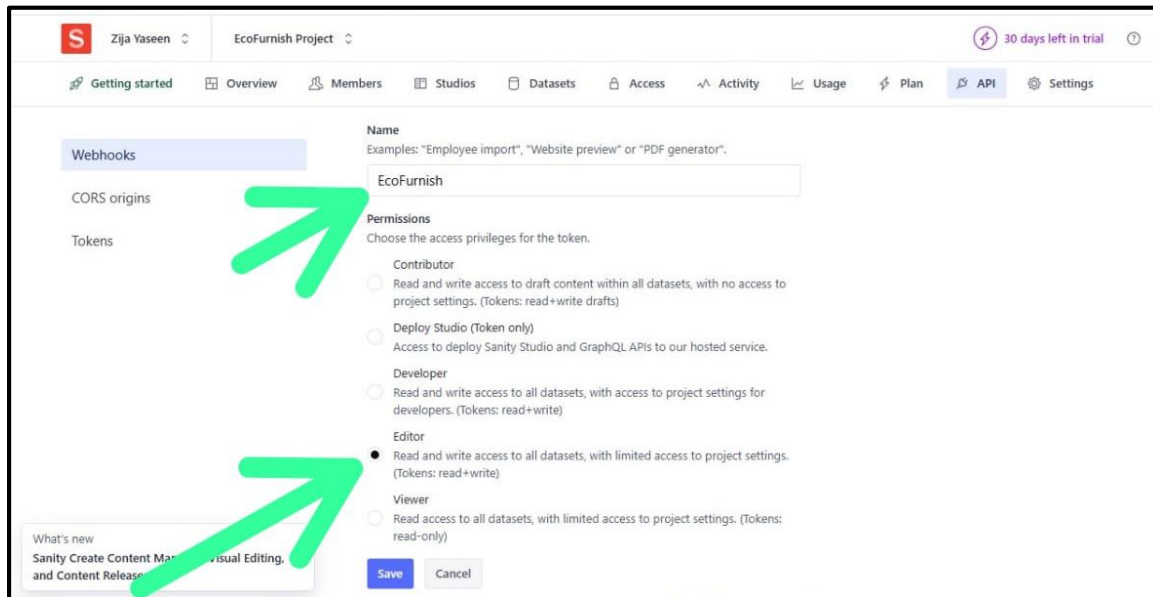
Copy your sanity projecID code and paste in `.env.local` file

Then generate API key:

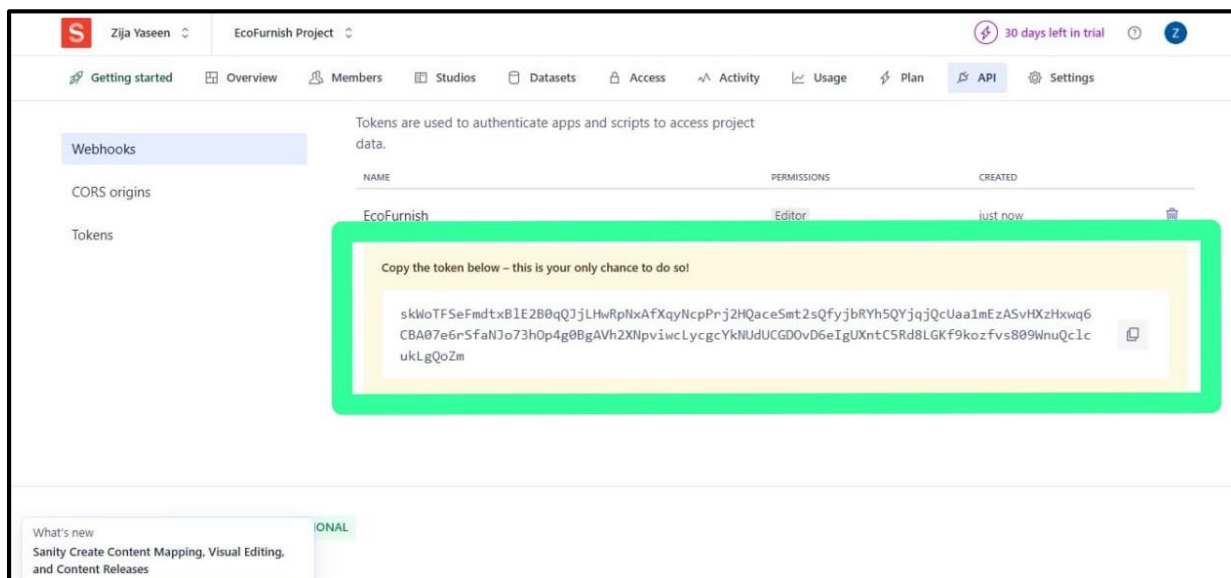
- Go to The API Tab
- Click on + Add API Token Button



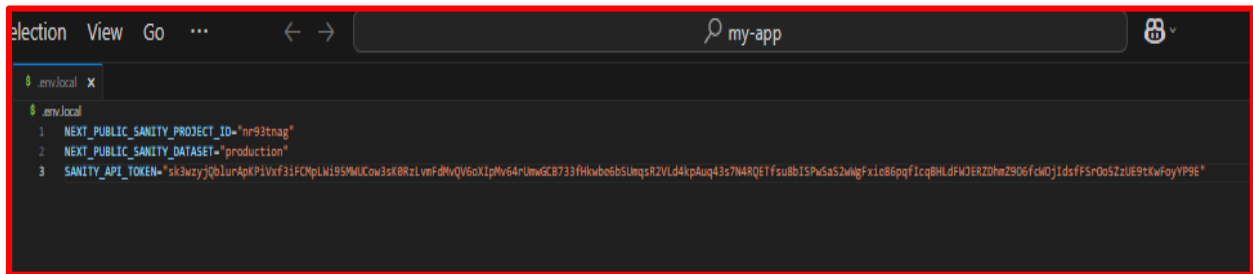
- Enter the name of your project, choose either **Developer Mode** or **Editor Mode**, and then click the **Save** button. Your API key will be generated.



- Copy the API key Token, And paste it on env.local file:**



Replace your sanity project ID with `NEXT_PUBLIC_SANITY_PROJECT_ID` and replace your Project Sanity API Token with `SANITY_API_TOKEN`.

A screenshot of a code editor window with a dark theme. The editor shows a file named `.env.local` with three lines of environment variables. The first line is `NEXT_PUBLIC_SANITY_PROJECT_ID="nr93tnag"`, the second is `NEXT_PUBLIC_SANITY_DATASET="production"`, and the third is `SANITY_API_TOKEN="sk3wzyjQb1urApKp1Vvf31FCMplW195MMUCow3sx0RzLvmFdMvQV6oXIpmv64rUmwGCB733fHkwb6b5UmqsRZVLd4kpAuq43s7N4RQETfsu8b1SPw5a52WNgfx1c8B6pqf1cqBHLdFWJERZ0hmZ906fckOj1dsffs7Oo5ZzUE9tKwfoYVP9E"`. The editor has a search bar at the top right containing `my-app` and a sidebar on the left with a file explorer showing `.env.local` selected.

```
1 NEXT_PUBLIC_SANITY_PROJECT_ID="nr93tnag"
2 NEXT_PUBLIC_SANITY_DATASET="production"
3 SANITY_API_TOKEN="sk3wzyjQb1urApKp1Vvf31FCMplW195MMUCow3sx0RzLvmFdMvQV6oXIpmv64rUmwGCB733fHkwb6b5UmqsRZVLd4kpAuq43s7N4RQETfsu8b1SPw5a52WNgfx1c8B6pqf1cqBHLdFWJERZ0hmZ906fckOj1dsffs7Oo5ZzUE9tKwfoYVP9E"
```

3. Define The Schema For Products:

- Create a file named *product.ts* in the *sanity/schemaTypes* directory.

```
export default {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'id',
      title: 'ID',
      type: 'string',
    },
    {
      name: 'name',
      title: 'Name',
      type: 'string',
    },
    {
      name: 'imagePath',
      title: 'Image Path',
      type: 'url',
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
  ],
}
```

```

{
  name: 'description',
  title: 'Description',
  type: 'text',
},
{
  name: 'discountPercentage',
  title: 'Discount Percentage',
  type: 'number',
},
{
  name: 'isFeaturedProduct',
  title: 'Is Featured Product',
  type: 'boolean',
},
{
  name: 'stockLevel',
  title: 'Stock Level',
  type: 'number',
},
{
  name: 'category',
  title: 'Category',
  type: 'string',
},
],
};

```

- Then , update *index.ts* file in *sanity/schemaTypes* directory.
(*sanity/schemaTypes/index.ts*)

```

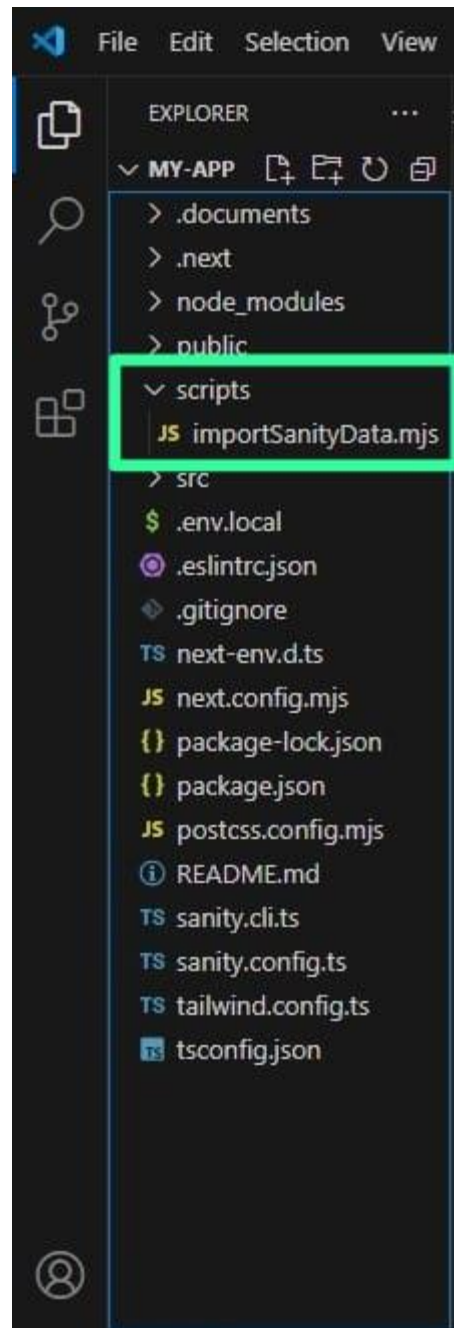
import { type SchemaTypeDefinition } from 'sanity'
import product from './product'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product],
}

```

4.Fetch Data from API and Insert into Sanity:

- Create a *scripts* folder in your project root and add a file named *importSanityData.mjs* inside it. (`scripts/importSanityData.mjs`).



- Copy the following code and paste it into the `importSanityData.mjs` file inside the `scripts` folder:

```
import { createClient } from '@sanity/client'
import axios from 'axios'
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') })
// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
})

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`)
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' })
    const buffer = Buffer.from(response.data)
    const asset = await client.assets.upload('image', buffer, {
      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 importData() {
  try {
    console.log('Fetching products from API...')
    const response = await axios.get('https://template-0-beta.vercel.app/api/product')
    const products = response.data
    console.log(`Fetched ${products.length} products`)
    for (const product of products) {
```

```

    console.log(`Processing product: ${product.name}`)
    let imageRef = null
    if (product.imagePath) {
      imageRef = await uploadImageToSanity(product.imagePath)}

    const sanityProduct = {
      _type: 'product',
      id: product.id,
      name: product.name,
      description: product.description,
      price: parseFloat(product.price),
      discountPercentage: product.discountPercentage,
      isFeaturedProduct: product.isFeaturedProduct,
      stockLevel: product.stockLevel,
      category: product.category,
      imagePath: product.imagePath,
    }
    console.log('Uploading product to Sanity:', sanityProduct.name)
    const result = await client.create(sanityProduct)
    console.log(`Product uploaded successfully: ${result._id}`)
  }
  console.log('Data import completed successfully!')
} catch (error) {
  console.error('Error importing data:', error)
}
}
importData()

```

To install the necessary packages for the project, run the following command in your terminal:

- `npm install @sanity/client axios dotenv`

This will install the **Sanity Client**, **Axios**, and **Dotenv** packages, which are required for the API integration.

5. Adding the Import Script to `package.json`

```
{
  "name": "my-app",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importSanityData.mjs"
  },
}
```

In the `package.json` file, under the `scripts` section, add the following line to define the script for importing data:

```
"scripts": {
  "import-data": "node scripts/importSanityData.mjs"
}
```

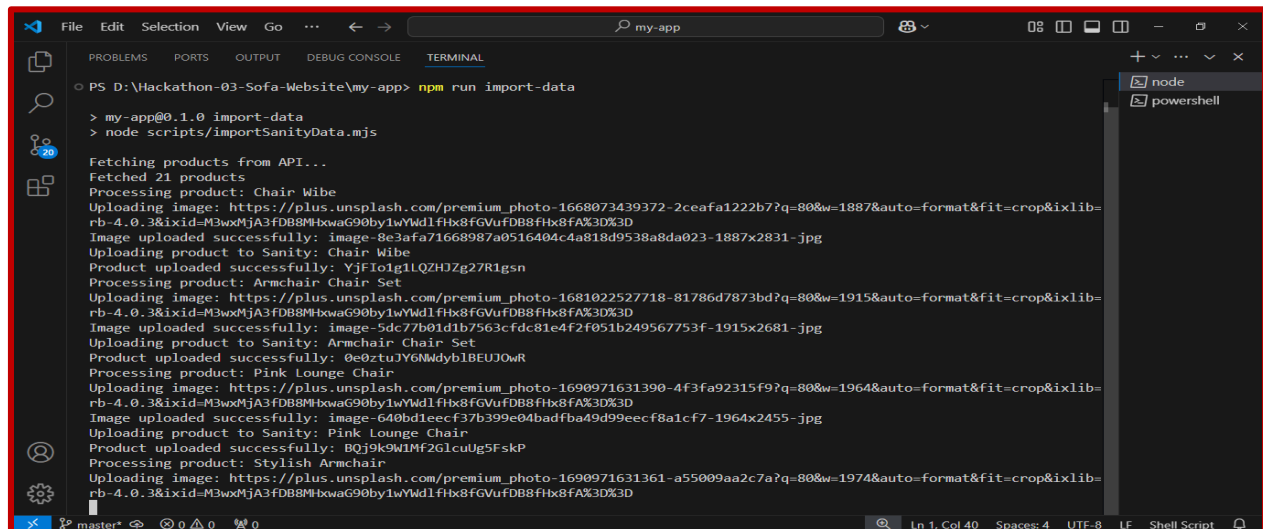
Run the following command in your terminal:

- `npm run import-data`

This command will execute the `importSanityData.mjs` script, which is designed to import data from your Sanity CMS into your project. It will initiate the process of fetching the data and integrating it into your application.

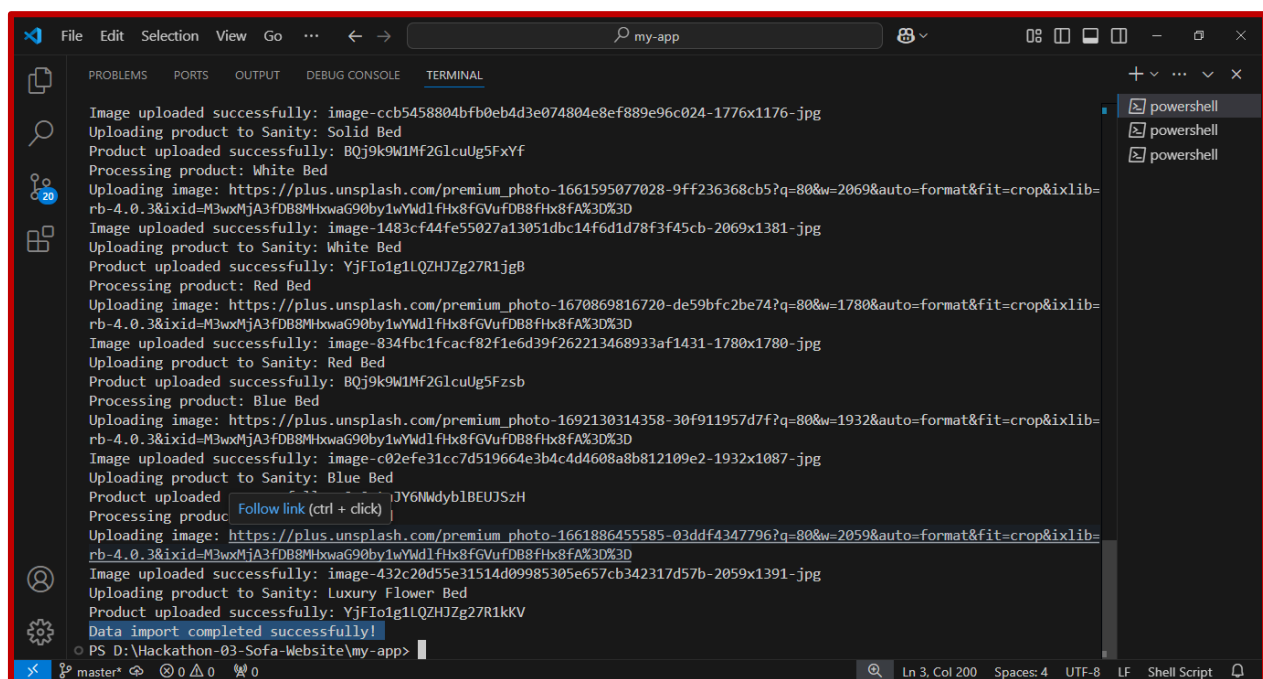
6:Successfully Imported Data:

Once the data is successfully uploaded, the terminal will display a confirmation message. Below is a screenshot of the terminal output for reference:



```
PS D:\Hackathon-03-Sofa-Website\my-app> npm run import-data
> my-app@0.1.0 import-data
> node scripts/importSanityData.mjs

Fetching products from API...
Fetched 21 products
Processing product: Chair Wibe
Uploading image: https://plus.unsplash.com/premium_photo-1668073439372-2ceafa1222b7?q=80&w=1887&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-8e3afa71668987a0516404c4a818d9538a8da023-1887x2831-jpg
Uploading product to Sanity: Chair Wibe
Product uploaded successfully: YjFIoIg1LQZHJZg27R1gsn
Processing product: Armchair Chair Set
Uploading image: https://plus.unsplash.com/premium_photo-1681022527718-81786d7873bd?q=80&w=1915&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-5dc77b01dd1b7563cfdc81e4f2f051b249567753f-1915x2681-jpg
Uploading product to Sanity: Armchair Chair Set
Product uploaded successfully: 0e0ztuYGNWdyb1BEUJOWr
Processing product: Pink Lounge Chair
Uploading image: https://plus.unsplash.com/premium_photo-1690971631390-4f3fa92315f9?q=80&w=1964&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-640bd1eefc37b399e04badfba49d99eefc8a1cf7-1964x2455-jpg
Uploading product to Sanity: Pink Lounge Chair
Product uploaded successfully: BQj9k9W1MF2GlcUg5FskP
Processing product: Stylish Armchair
Uploading image: https://plus.unsplash.com/premium_photo-1690971631361-a55009aa2c7a?q=80&w=1974&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
```

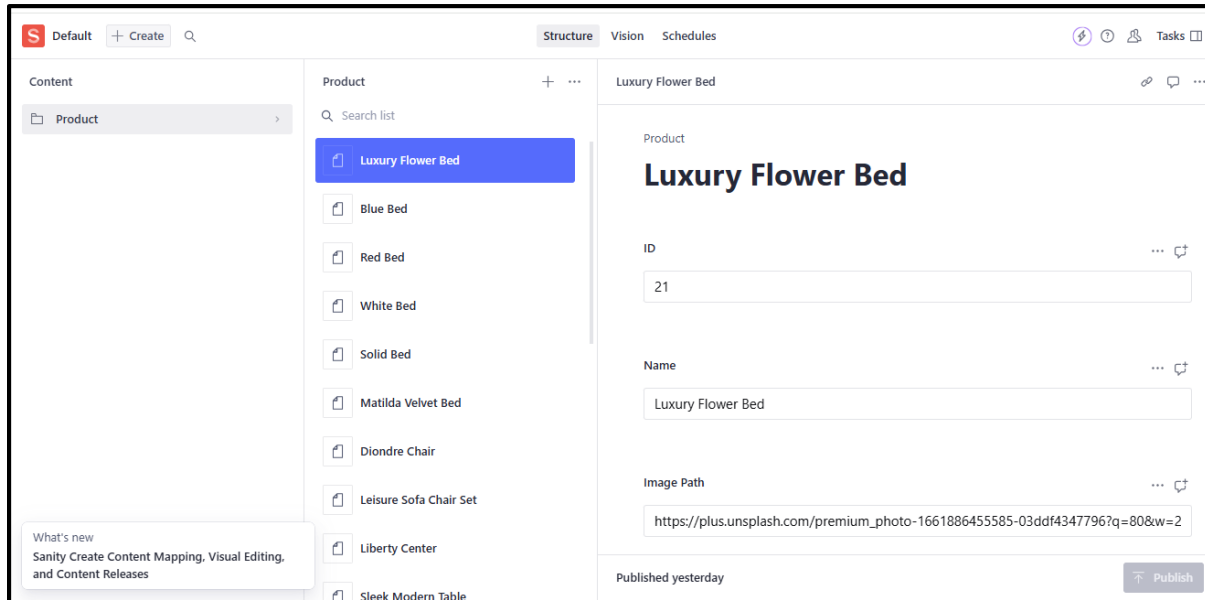


```
Image uploaded successfully: image-ccb5458804bfb0eb4d3e074804e8ef889e96c024-1776x1176-jpg
Uploading product to Sanity: Solid Bed
Product uploaded successfully: BQj9k9W1MF2GlcUg5Fxyf
Processing product: White Bed
Uploading image: https://plus.unsplash.com/premium_photo-1661595077028-9ff236368cb5?q=80&w=2069&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-1483cf44fe55027a13051dbc14f6d1d78f3f45cb-2069x1381-jpg
Uploading product to Sanity: White Bed
Product uploaded successfully: YjFIoIg1LQZHJZg27R1jgB
Processing product: Red Bed
Uploading image: https://plus.unsplash.com/premium_photo-1670869816720-de59bfc2be74?q=80&w=1780&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-834fbc1fcacf82f1e6d39f262213468933af1431-1780x1780-jpg
Uploading product to Sanity: Red Bed
Product uploaded successfully: BQj9k9W1MF2GlcUg5Fzsb
Processing product: Blue Bed
Uploading image: https://plus.unsplash.com/premium_photo-1692130314358-30f911957d7f?q=80&w=1932&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-c02efe31cc7d519664e3b4c4d4608a8b812109e2-1932x1087-jpg
Uploading product to Sanity: Blue Bed
Product uploaded successfully: JY6NMdyb1BEUJSzH
Processing product: Follow link (ctrl + click)
Uploading image: https://plus.unsplash.com/premium_photo-1661886455585-03ddf4347796?q=80&w=2059&auto=format&fit=crop&iixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MhBwaG90by1wYWdlFHx8fGVuFDB8fhx8fA%3D%3D
Image uploaded successfully: image-432c20d55e31514d09985305e657cb342317d57b-2059x1391-jpg
Uploading product to Sanity: Luxury Flower Bed
Product uploaded successfully: YjFIoIg1LQZHJZg27R1kkV
Data import completed successfully!
PS D:\Hackathon-03-Sofa-Website\my-app>
```

7: Verify Data in Sanity Studio

To confirm that the data has been successfully imported, open your browser and navigate to: **`http://localhost:3000/sanity`**

You should see the list of products displayed in the Sanity Studio interface.



7. Fetching Product Data from Sanity CMS and Displaying on the Frontend

The following code snippet demonstrates how to fetch product data from Sanity CMS and display it on the frontend.

Code Snippet:

Sanity Query and API Integration

First, the `GetProductsData` function retrieves data from Sanity CMS using the `client.fetch()` method:

```
import { client } from './client';

export async function GetProductsData() {
  const query = `
    *[_type == 'product'] {
      _id,
      name,
      imagePath,
      description,
      price,
      category,
      stockLevel,
      isFeaturedProduct
    }`;
  return await client.fetch(query);}
```

Defining the Product Interface

The `Product` interface ensures type safety for the product data:

```
interface Product {
  _id: string;
  name: string;
  imagePath?: string;
  description?: string;
  price?: number;
  category?: string;
  stockLevel?: number;
  isFeaturedProduct?: boolean;
}
```

Fetching and Rendering Products on the Frontend

The `Shop` component fetches products and renders them in a simple layout:

```
import { GetProductsData } from '@sanity/lib/queries';

const Shop = async () => {
  const products: Product[] = await GetProductsData();

  return (
    <div>
      <h1>Shop</h1>
      <div style={{ display: 'flex', flexWrap: 'wrap', gap: '1rem' }}>
        {products.map((product) => (
          <div
            key={product._id}
            style={{
              border: '1px solid #ddd',
              padding: '1rem',
              borderRadius: '8px',
              width: '200px',
            }}
          >
            {product.imagePath && (
              <img
                src={product.imagePath}
                alt={product.name}
                style={{ width: '100%', borderRadius: '8px' }}
              />
            )}
            <h3>{product.name}</h3>
            <p>{product.description}</p>
            <p>Price: ${product.price}</p>
            <p>Category: {product.category}</p>
            <p>Stock: {product.stockLevel}</p>
            {product.isFeaturedProduct && <strong>Featured Product!</strong>}
          </div>
        ))}
      </div>
    </div>
  );
}; export default Shop;
```

Explanation:

1. Data Fetching:

The `GetProductsData` function queries all products with the specified fields from Sanity CMS.

2. Frontend Rendering:

- Products are displayed in a card-style layout.
- Each card includes the product's image, name, description, price, category, stock level, and a badge if it's a featured product.

Displaying Product Data on the Frontend:

After successfully fetching the product data from Sanity CMS, it is displayed on the frontend in a visually structured format. The product details, such as the name, image, description, price, category, stock level, and a featured badge (if applicable), are presented in a frontend web page.

