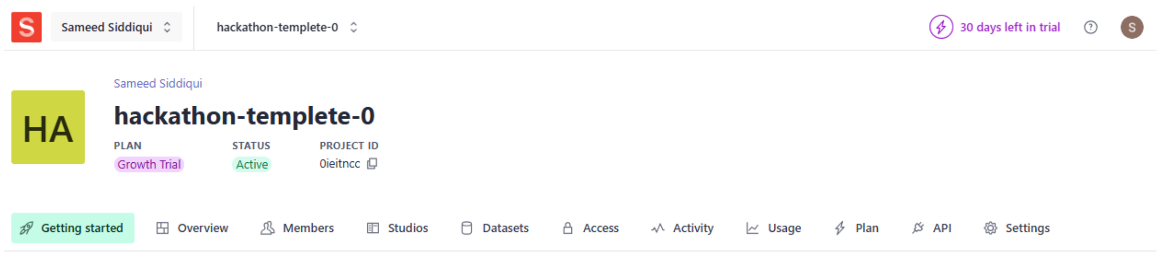
Day 3 - API Integration Report – Samspire Interiors

Date:19-1-2025

Step 1: Create Project in Sanity

1. **Sign Up/Log In** to Sanity.io:
   - Visit Sanity.io & log In
2. **Create a New Project**:
   - After logging in, clicked **"Create Project"**.
   - Enter a project name, e.g., "hackathon-template-0."
   - Click **"Create Project"** to proceed.



Sameed Siddiqui

hackathon-templete-0

PLAN: Growth Trial

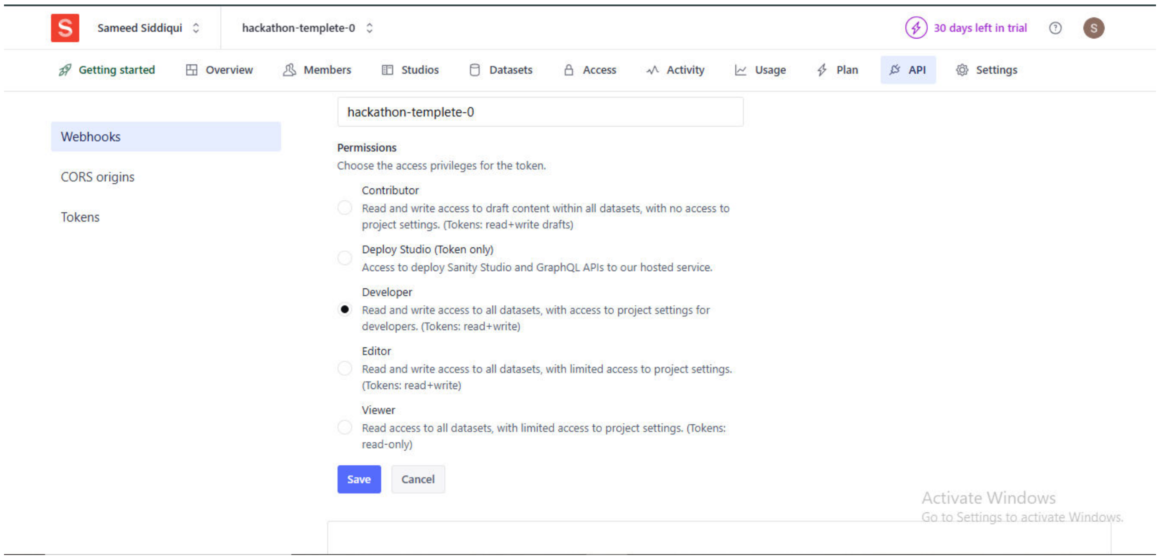
STATUS: Active

PROJECT ID: 0ieitncc

Step 2: Generate Token in Project

1. **Go to the Sanity Dashboard**:
   - Visit Sanity.io, log in, and open my project.
   - Navigate to **Settings > API > Tokens**.
   - Click **"Add API Token"**, select the appropriate permissions (e.g., Read, Write,Developer or Manage), and generated my token.

Copy the token securely and add it to my environment variables (e.g., `.env.local`).



hackathon-templete-0

Permissions

Choose the access privileges for the token.

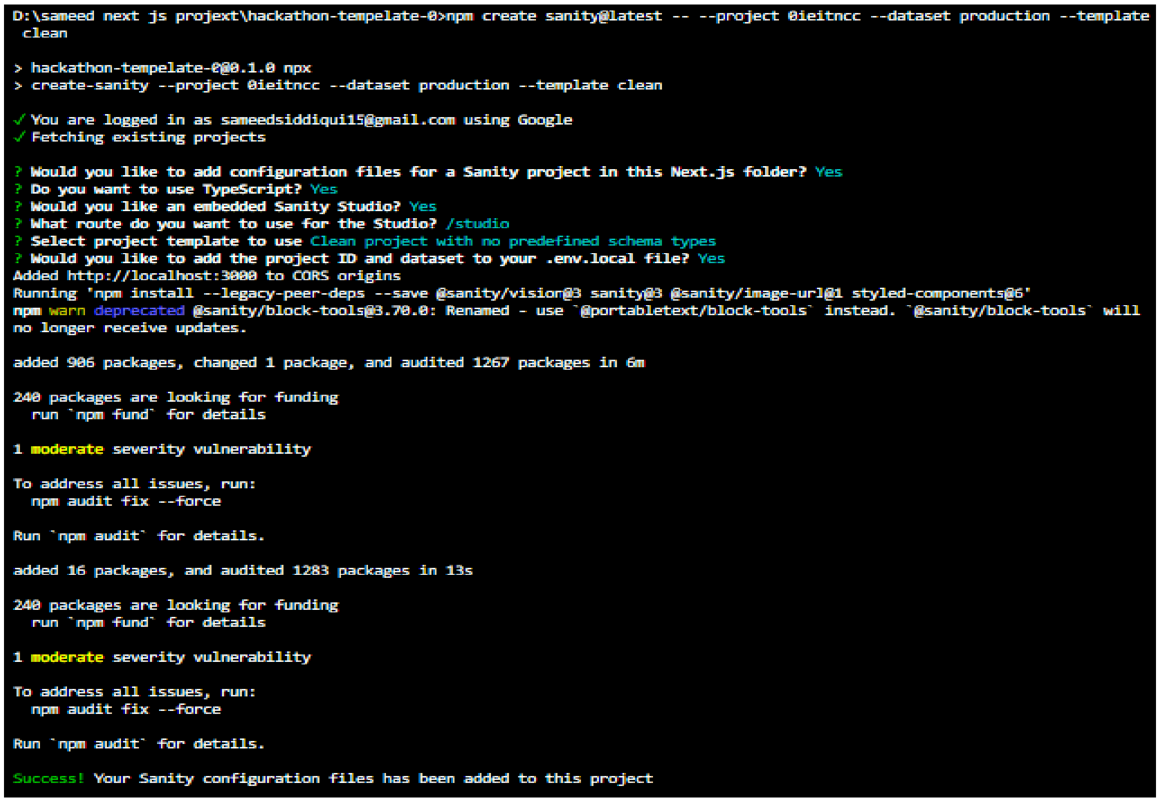
- Contributor
  Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)
- Deploy Studio (Token only)
  Access to deploy Sanity Studio and GraphQL APIs to our hosted service.
- ● Developer
  Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)
- Editor
  Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)
- Viewer
  Read access to all datasets, with limited access to project settings. (Tokens: read-only)

Step 3: Initialize Next JS Project with Sanity

1. **Run the Initialization Command**:

```
npm create sanity@latest -- --project 0ieitncc --dataset production --template clean
```

- This sets up Sanity Studio with a clean template in your Next.js project.



```
D:\sameed next js projext\hackathon-tempelate-0>npm create sanity@latest -- --project 0ieitncc --dataset production --template
 clean

> hackathon-tempelate-0@0.1.0 npx
> create-sanity --project 0ieitncc --dataset production --template clean

✓ You are logged in as sameedsiddiqui15@gmail.com using Google
✓ Fetching existing projects

? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'
npm warn deprecated @sanity/block-tools@3.70.0: Renamed - use `@portabletext/block-tools` instead. `@sanity/block-tools` will
no longer receive updates.

added 906 packages, changed 1 package, and audited 1267 packages in 6m

240 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

added 16 packages, and audited 1283 packages in 13s

240 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

Success! Your Sanity configuration files has been added to this project
```

Step 4: Define the Schema for Products

1. Created a schema in Sanity for the `product` document.

```
export default {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
        { name: 'id', title: 'ID', type: 'string' },
        { name: 'name', title: 'Name', type: 'string' },
        { name: 'image', title: 'Image', type: 'image' },
        { 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' },
    ],
};
```

- Fields included:
    - `id`: Unique identifier for each product.
    - `name`: Name of the product.
    - `image`: Image of the product, stored as a Sanity image asset.
    - `imagePath`: URL of the product image.
    - `price`: Price of the product.
    - `description`: Description of the product.
    - `discountPercentage`: Discount offered on the product.
    - `isFeaturedProduct`: Boolean to indicate if it's a featured product.
    - `stockLevel`: Quantity available in stock.
    - `category`: Category of the product.

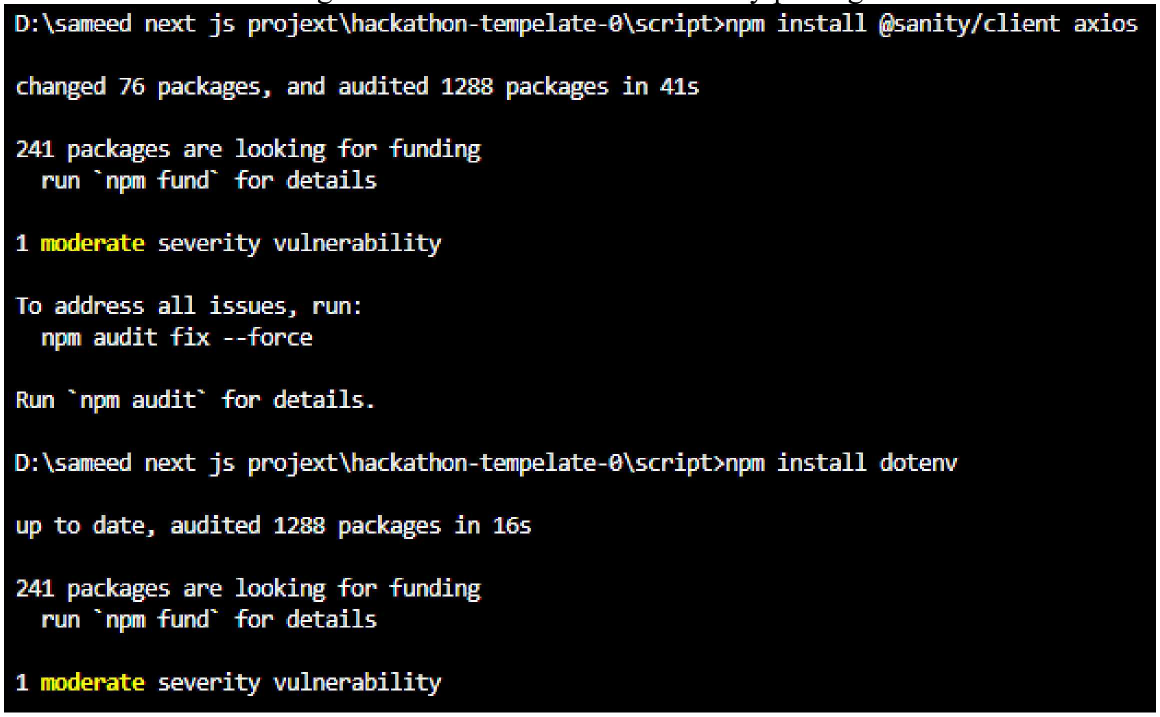
Step 5: Set Up a Environment Variables

1. Add the following variables to your `.env.local` file in the project root directory:

```
NEXT_PUBLIC_SANITY_PROJECT_ID="0ieitncc"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="skbfZ8u1Nmn1gwmcatG6vKHeJkfoegar9FblMyQeKJPrf5NGX4fY4UKM7ZXoHs4
pfyURus5MVOIFTWt3k2kgYDchQWaa3unv26u93kqcZTDRXSsDEaRf0ES9WVizy0i5BPNxG4m5X69myvBK
TVFI5zJ90o4aj7sdlBUNZsk0pc6LgFkzx5Tv"
```

Step 6: Install Required Dependencies

1. Run the following command to install the necessary packages:



```
D:\sameed next js projext\hackathon-tempelate-0\script>npm install @sanity/client axios

changed 76 packages, and audited 1288 packages in 41s

241 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

D:\sameed next js projext\hackathon-tempelate-0\script>npm install dotenv

up to date, audited 1288 packages in 16s

241 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability
```

Step 7: Script/migrate.mjs

1. Created a folder of script in root folder.
2. In script create file of migrate.mjs.
3. In migrate.mjs
    1. Set Up the Sanity Client

- Import and configure the `createClient` function in your migration script as follows:

```
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',
});
```

2. Create Image Upload Function:

   - Use the provided uploadImageToSanity function to handle the uploading of images to Sanity's asset management:

```
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.message);
 return null;
 }
}
```

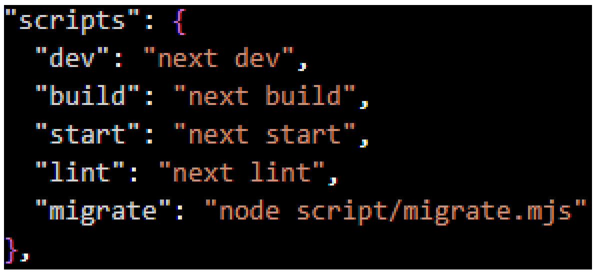
3. Write the Migration Script:

- Use the importData function to fetch data from the external API, upload images, and create documents in Sanity:

```
async function importData() {
 try {
 console.log('Migrating data, please wait...');
 // Fetch products from the API
 const response = await
axios.get('https://template-0-beta.vercel.app/api/product');
 const products = response.data;

 console.log('Products fetched:', products);

 for (const product of products) {
 let imageRef = null;

 if (product.imagePath) {
 imageRef = await uploadImageToSanity(product.imagePath);
 }

 const sanityProduct = {
 _type: 'product',
 id: product.id,
 name: product.name,
 category: product.category,
 description: product.description,
 discountPercentage: product.discountPercentage,
 isFeaturedProduct: product.isFeaturedProduct,
 stockLevel: product.stockLevel,
 imagePath: parseFloat(product.price),
 image: imageRef ? {
 _type: 'image',
 asset: {
 _type: 'reference',
 _ref: imageRef,
 },} : undefined,
 imagePath: product.imagePath, // Store original image URL
 };

 await client.create(sanityProduct);
 console.log(`Product created in Sanity: ${sanityProduct.id}`);
 }

 console.log('Data migrated successfully!');
 } catch (error) {
 console.error('Error in migrating data:', error.message);
 }
}

importData();
```

Step 8: Steps to Add the Migration Script

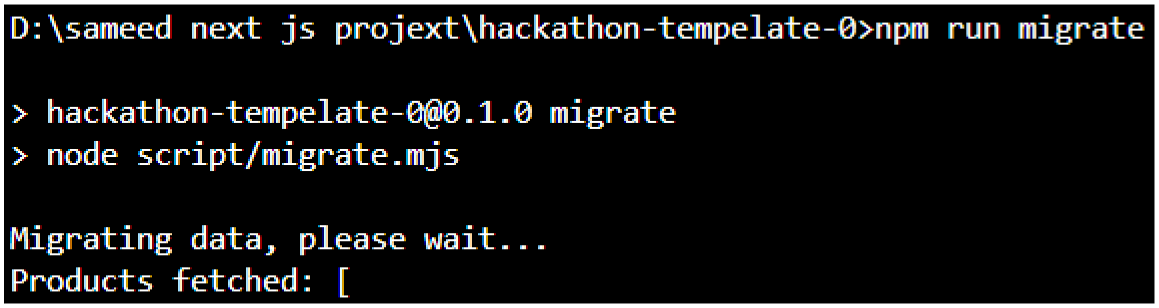
1. Locate package.json File:
   - Open the package.json file located in the root of your Next.js project
2. Add a Custom Script.
   - Under the "scripts" section, add a new script entry for the migration script.



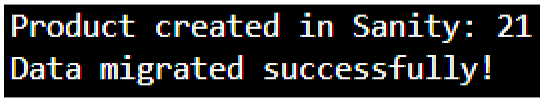
```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "migrate": "node script/migrate.mjs"
},
```

Step 9: Run the Migration Script

1. Execute the migration script by running:



```
D:\sameed next js projext\hackathon-tempelate-0>npm run migrate

> hackathon-tempelate-0@0.1.0 migrate
> node script/migrate.mjs

Migrating data, please wait...
Products fetched: [
```



```
Product created in Sanity: 21
Data migrated successfully!
```

Step 10: Fetch Products from Sanity

1. Wrote a GROQ query to fetch data from the Sanity dataseti n `ShopTopPicks` component in Next.js.
   - Retrieved fields: `id`, `name`, `image.asset->url` (image URL), and `price`.

```
import Link from "next/link";
import { client } from "@/sanity/lib/client";

const getData = async () => {
  const res = await client.fetch(
    `*[_type == "product"]{
    id,
    name,
    "imageUrl":image.asset->url,
    price
    }`
  )
  return (res)
}
```

Step 11: Display Products on the Frontend

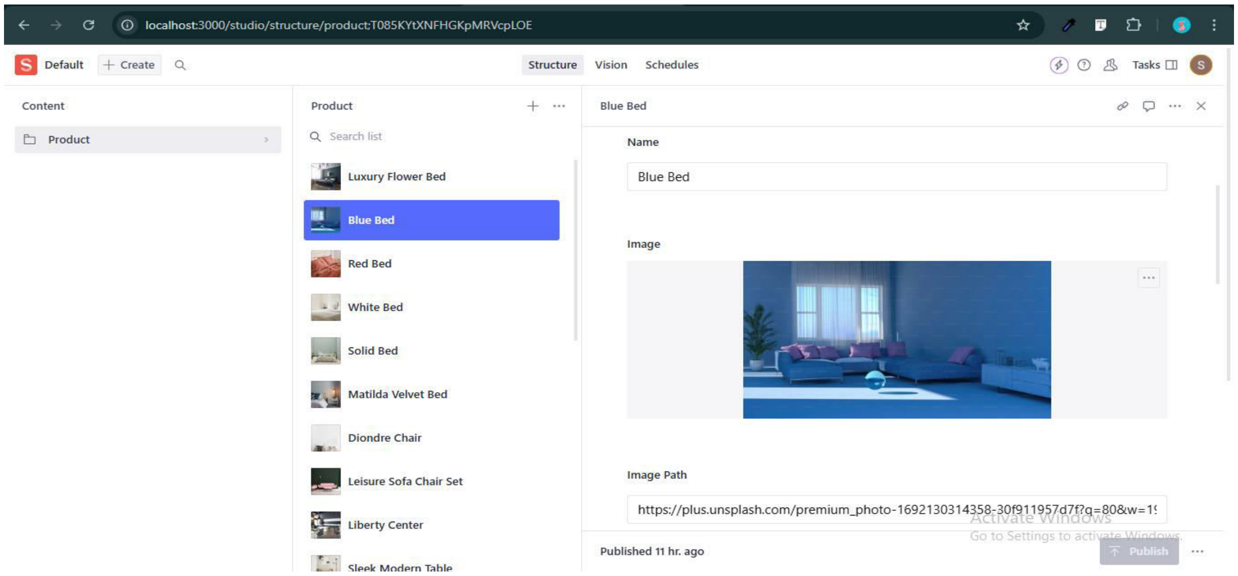
1. Created a `ShopTopPicks` component in Next.js.
   - Called the data-fetching function to get products.
   - Rendered a grid layout displaying product images, names, and prices.
   - Included pagination controls with navigation buttons.

```

export default async function ShopTopPicks() {
    const products = await getData();

    return (
        <section className="py-12 bg-white">
            <div className="max-w-[1440px] mx-auto px-6">
                <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4
gap-6">
```

```
                    {products.map((product:any) => (
                        <div
                            key={product.id}
                            className="text-center p-4 flex flex-col items-center
justify-center hover:shadow-md transition-shadow"
                        >
                            <img
                                src={product.imageUrl}
                                alt={product.name}
                                className="w-full h-80 rounded-md mb-4 object-cover
object-center"
                            />
                            <h3 className="text-lg
font-medium">{product.name}</h3>
                            <p className="text-sm font-semibold text-gray-700">
                                ${product.price}
                            </p>
                        </div>
                    ))}
                </div>

                <div className="flex justify-center mt-8">
                    <nav className="flex gap-4">
                        <button className="px-4 py-2  bg-[#FBEBB5] rounded-md
hover:bg-gray-100">1</button>
                        <button className="px-4 py-2  rounded-md
hover:bg-[#FBEBB5]"><Link href={'/Blog'}>2</Link></button>
                        <button className="px-4 py-2  rounded-md
hover:bg-[#FBEBB5]"><Link href={'/Blog'}>3</Link></button>
                        <button className="px-4 py-2  rounded-md
hover:bg-[#FBEBB5]"><Link href={'/Blog'}>Next</Link></button>
                    </nav>
                </div>
            </div>
        </section>
    );
}
```

View of Sanity Studio

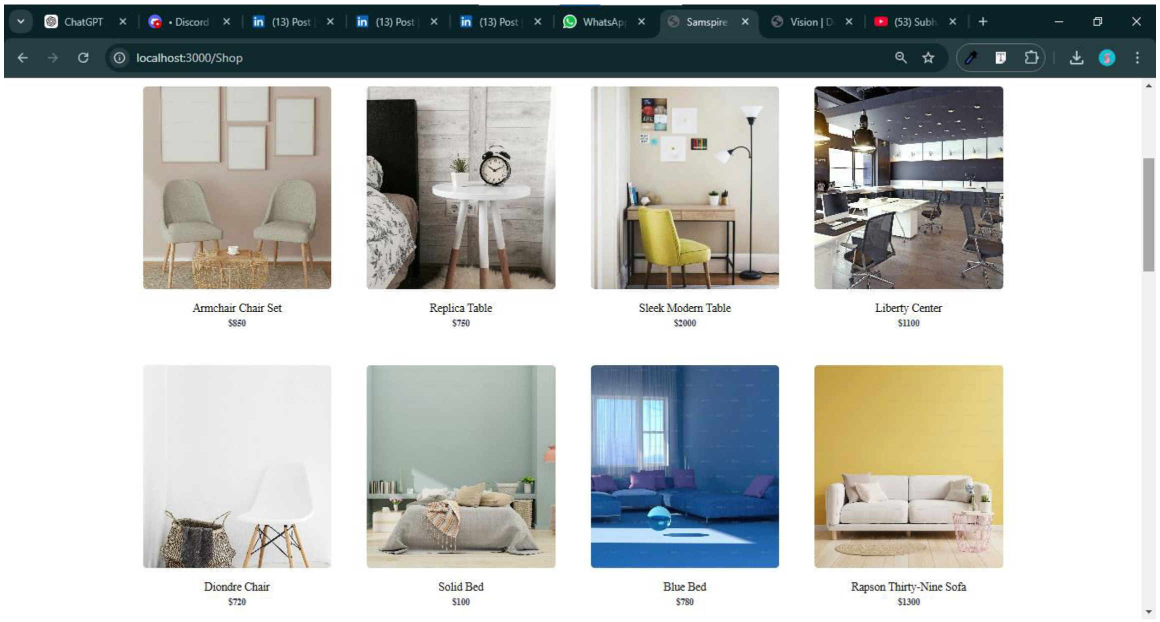
Once the data is imported, your Sanity Studio interface will look like this:



View of Frontend/Website

The frontend page showcases the furniture product details fetched from Sanity. Below is an example of how the page appears:

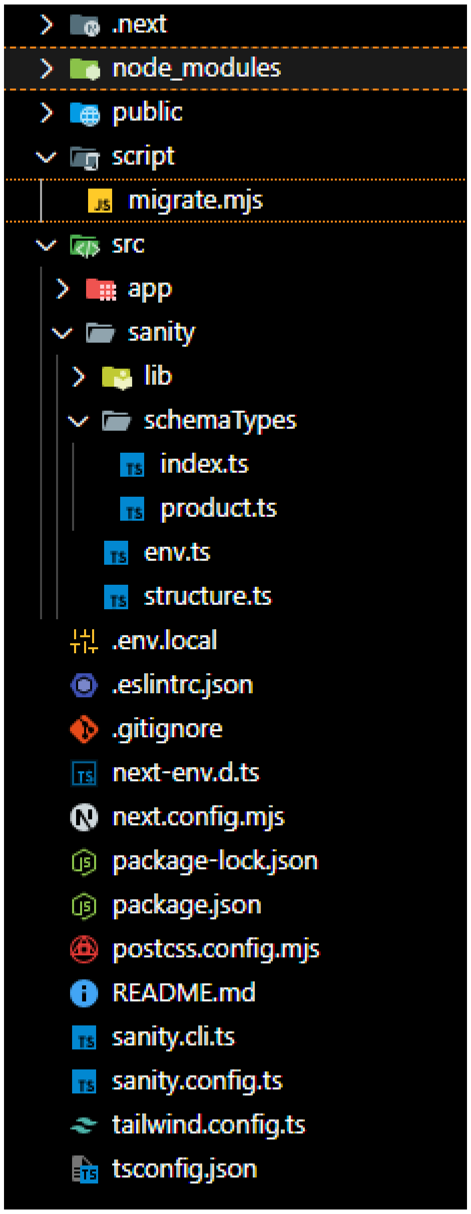
In this refernece we can clearly see images, product title, price and discount.



Conclusion

The integration of the e-commerce API with the frontend application through Sanity has been successfully implemented. The API dynamically retrieves product details and images, while Sanity acts as the backend for storing and managing this data. This setup streamlines the handling of product information and elevates the user experience by providing an intuitive, flexible, and engaging frontend for seamless browsing and shopping.

Folder Structure



```
> .next
> node_modules
> public
v script
    migrate.mjs
v src
  > app
  v sanity
    > lib
    v schemaTypes
        index.ts
        product.ts
      env.ts
      structure.ts
  .env.local
  .eslintrc.json
  .gitignore
  next-env.d.ts
  next.config.mjs
  package-lock.json
  package.json
  postcss.config.mjs
  README.md
  sanity.cli.ts
  sanity.config.ts
  tailwind.config.ts
  tsconfig.json
```

Prepared by : Sameed Siddiqui