DAY 3 - API INTEGRATION AND DATA MIGRATION

API integration process.

Adjustments Made to Schemas:

1.Identify New Data Requirements:

- . Analyzed the API data structure and determined necessary fields for the schema.
- Added or modified fields in the schema to accommodate new data.:
- *Tested the updated schema by adding new documents through the Sanity Studio or programmatically.*
- *Mapped API response fields to the updated schema to ensure proper data storage in Sanity.*

Code Snippet

```
const productSchema = {
  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',
  options: {
     isHighlighted: true,
       },
},
{
  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',
    },
  ],
};

export default productSchema;
```

Fetch Data from API:

Set Up Axios: Install Axios using npm install axios.

Import Axios into my script or component: import axios from 'axios';.

. Use Axios to make a GET request to the API endpoint:

Data Migration Script

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

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,
      price: 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);
```

}}

mportData()

---

3. Display Data in Project:

Connect and Render:

I. Combine the API fetching, pushing, and querying in my sanity studio to display the data dynamically in my hackathon project. Use state management or props to reflect changes on the UI.

```
import React from 'react'
import Navbar from '../components/navbar'
import Mainbanner from '../components/pagebanner'
import Offers from '../components/offers'
import Footer from '../components/footer'
import { client } from '../../sanity/lib/client'
import Link from 'next/link'
export default async function Shoppage() {
  const query = `*[_type == "product"]{
    id,
    name,
    description,
    price,
    "imageUrl": image.asset->url,
    discountPercentage,
  }`;
  const products = await client.fetch(query);
  return (
    <div>
      <Navbar />
      <Mainbanner pageName="Shop" />
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4 p-4">
```

```
{products.map((product:any) => (
 <Link href={`/shop/${product.id}`}
  key={product.id}
  className="border rounded-lg shadow-lg p-4 bg-white"
 >
  {product.imageUrl && (
   <img
    src={product.imageUrl}
    alt={product.title}
    className="h-40 w-full object-cover rounded-t-lg"
   />
  )}
  <h2 className="text-lg font-bold mt-2">{product.title}</h2>
  <p className="text-sm text-gray-600">{product.description}</p>
  <p className="text-lg font-semibold text-green-600 mt-2">
   ${product.price}
  </p>
  {/* {product.discountPercentage && (
   <p className="text-sm text-gray-600 line-through mt-2">
    ${product.price} (-{product.discountPercentage}%){" "}
   </p>
  )} */}
   {product.discountPercentage > 0 ? (
    <>
     <p className="text-md text-gray-500 line-through">
      ${product.price.toFixed(2)}
     </p>
     <p className="text-lg font-semibold text-green-600">
      ${(product.price * (1 - product.discountPercentage / 100)).toFixed(2)}{' '}
      <span className="bg-red-100 text-red-600 text-sm font-bold px-2 py-1 rounded-full">
       {product.discountPercentage}% off
```

```
            </span>
</p>
          </>
        ) : (
          <p className="text-lg font-semibold text-black">
            ${product.price.toFixed(2)}
          </p>
        )} </Link>
    ))} </div>
     <Offers/>
     <Footer/>       </div>    ) }
```

4.Test Integration

Verified the changes by querying and displaying the data from Sanity in the project. Adjusted frontend rendering to reflect the new schema fields.

Migration Steps and Tools Used:

1. Analyze Existing Schema:

- Reviewed the current schema to identify necessary updates based on API data or project requirements.

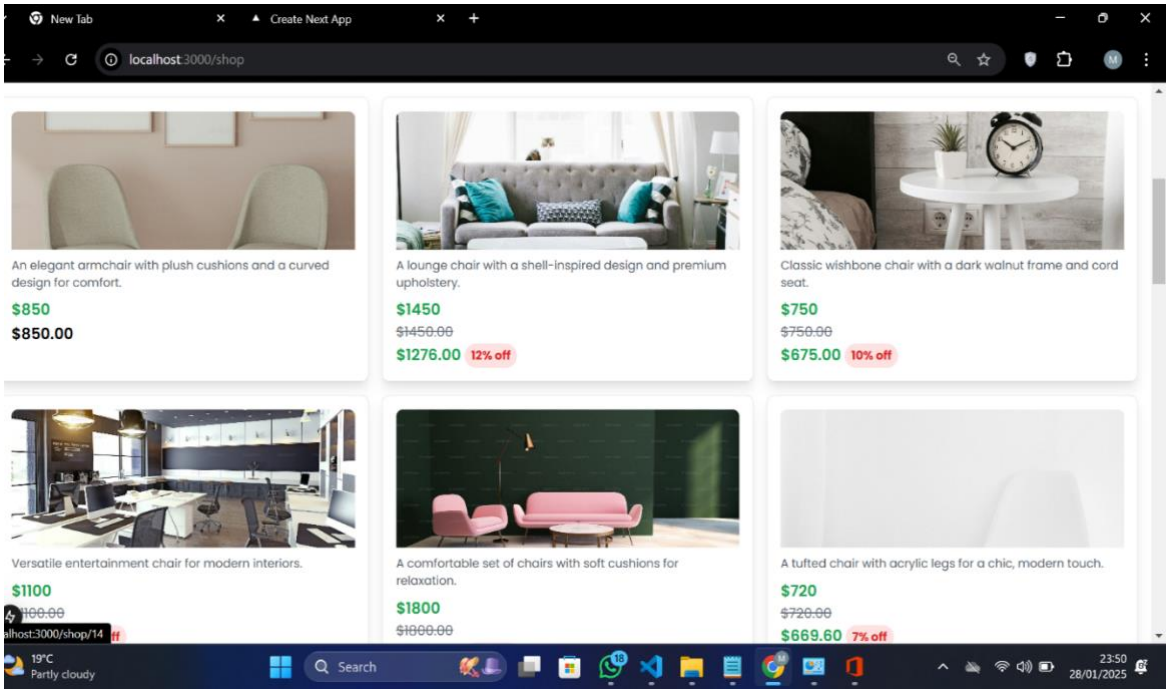
2. Update Schema in Sanity:

- Made adjustments to the schema by modifying or adding new fields using Sanity's schema definition files.

3. Tools Used:

- Sanity CLI: For exporting and importing datasets.
- Sanity Studio: To manage and test schema changes.
- Axios: To fetch data from APIs for migration.
- Sanity Client: For programmatically inserting or updating data.

Product showing to my project



Api Response