Day 3 Hackathon: API Integration and Data Migration Process

Objectives:

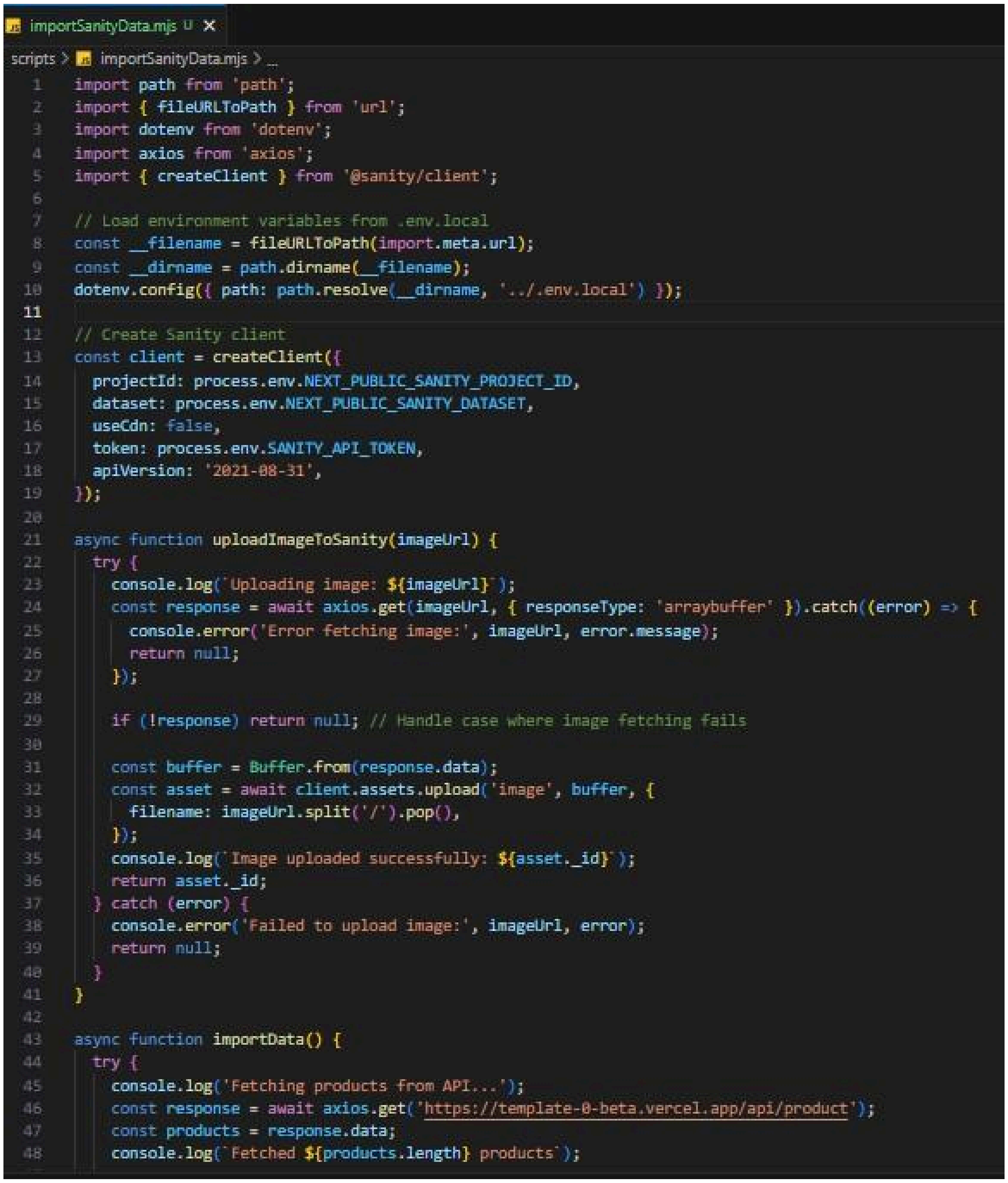
- **Implement API integration within a Next.js application.**
- **Transfer data from external APIs into Sanity CMS.**
- **Ensure schemas are validated and synchronized with data sources.**

Key Takeaways:

- **Grasp API integration strategies and their implementation.**
- **Learn the steps involved in migrating data from APIs to CMS.**
- **Customize and validate schemas in Sanity CMS for seamless data compatibility.**

Custom Data Migration Script

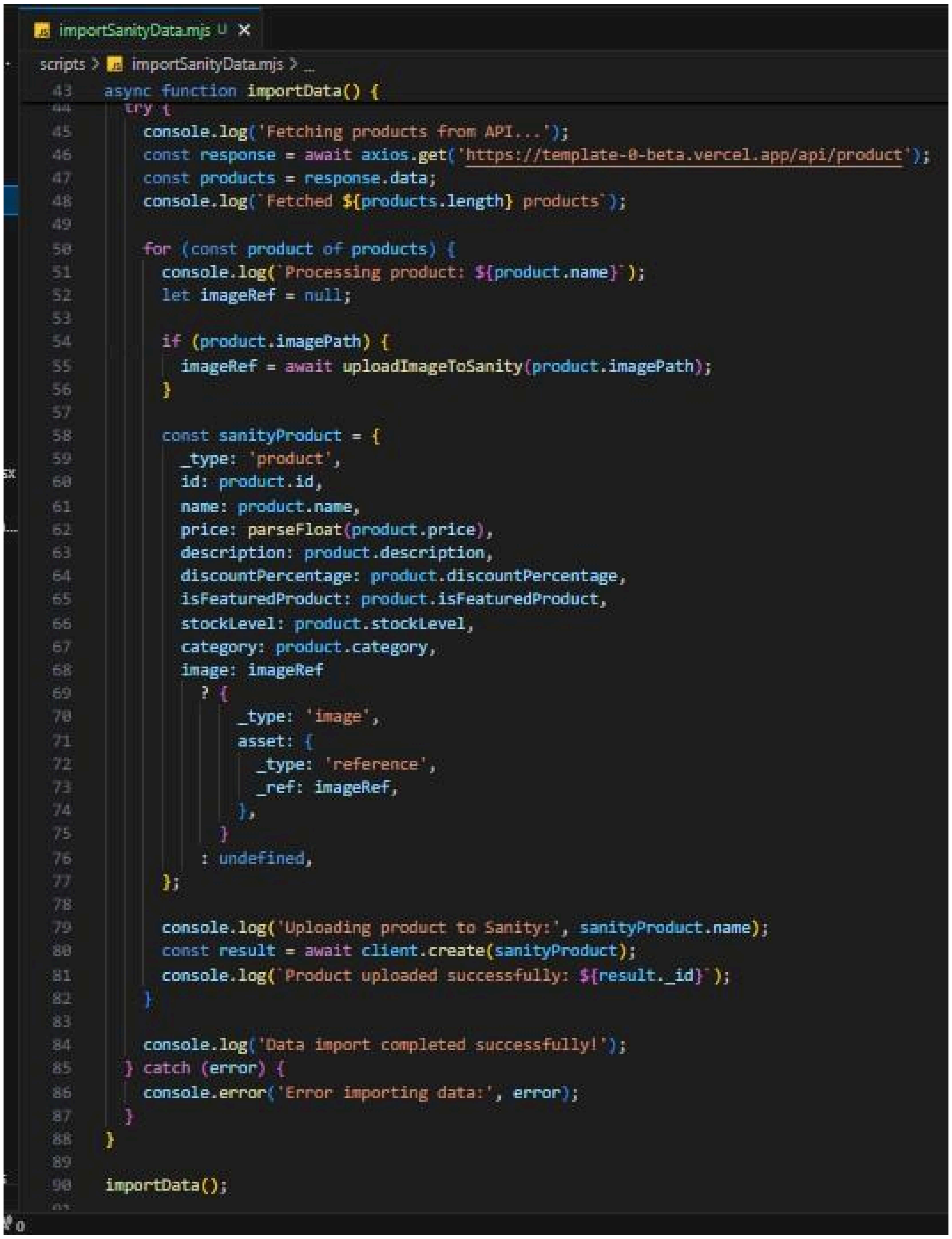
This script enables the seamless transfer of data from Sanity CMS to the E-Commerce Marketplace database. The migration workflow involves the following key steps:



importSanityData.mjs U

scripts > importSanityData.mjs > ...

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

// 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' }).catch((error) => {
      console.error('Error fetching image:', imageUrl, error.message);
      return null;
    });

    if (!response) return null; // Handle case where image fetching fails

    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`);
```



```javascript
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,
        price: parseFloat(product.price),
        description: product.description,
        discountPercentage: product.discountPercentage,
        isFeaturedProduct: product.isFeaturedProduct,
        stockLevel: product.stockLevel,
        category: product.category,
        image: imageRef
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined,
      };

      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();
```

1. **Sanity API Configuration:**

   a. Connects to Sanity's API using a predefined dataset and project ID, authenticated via an API token.

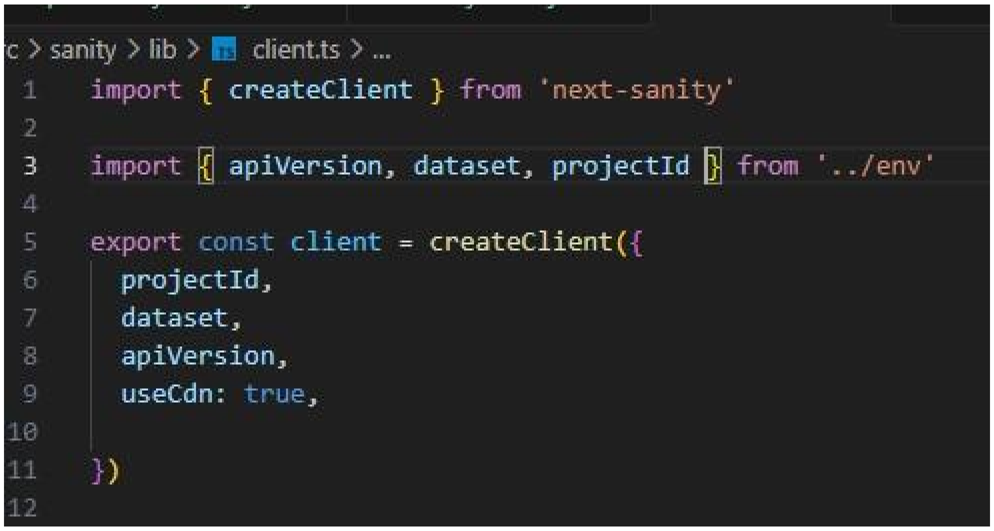
b. Environment variables securely store sensitive data like project ID and dataset.

2. **Data Retrieval from Sanity:**
   a. GROQ queries fetch structured content from Sanity CMS.
   b. Retrieves product details such as categories, descriptions, and pricing.

3. **Data Mapping and Adjustment:**
   a. Restructures the fetched data to match the application's schema for compatibility.

4. **Database Insertion:**
   a. Inserts data into the database via REST API or direct commands.
   b. Error handling ensures smooth migration without interruptions.

CLIENT PAGE CODE



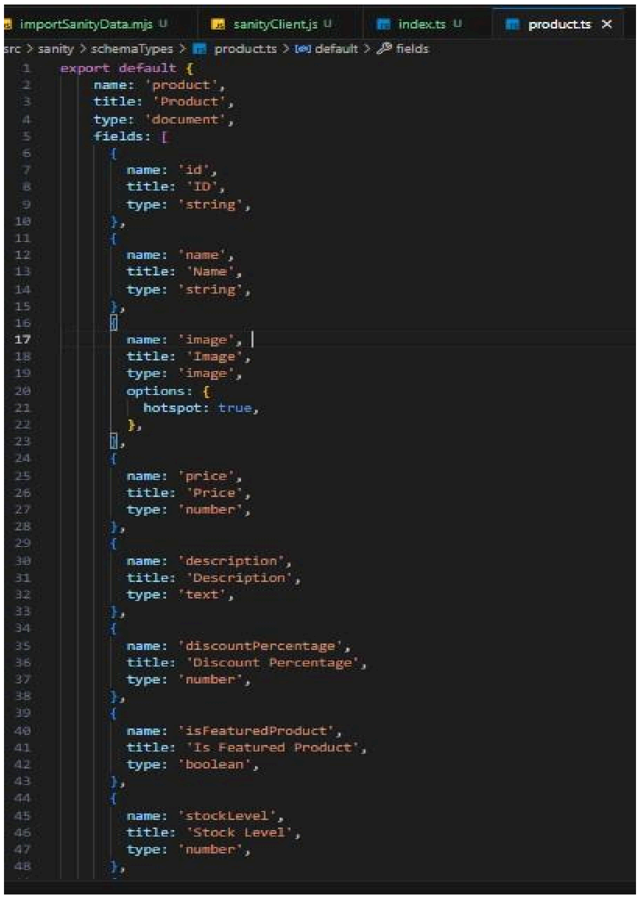
c > sanity > lib > client.ts > ...

```
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true,

})

```

Schema Code:

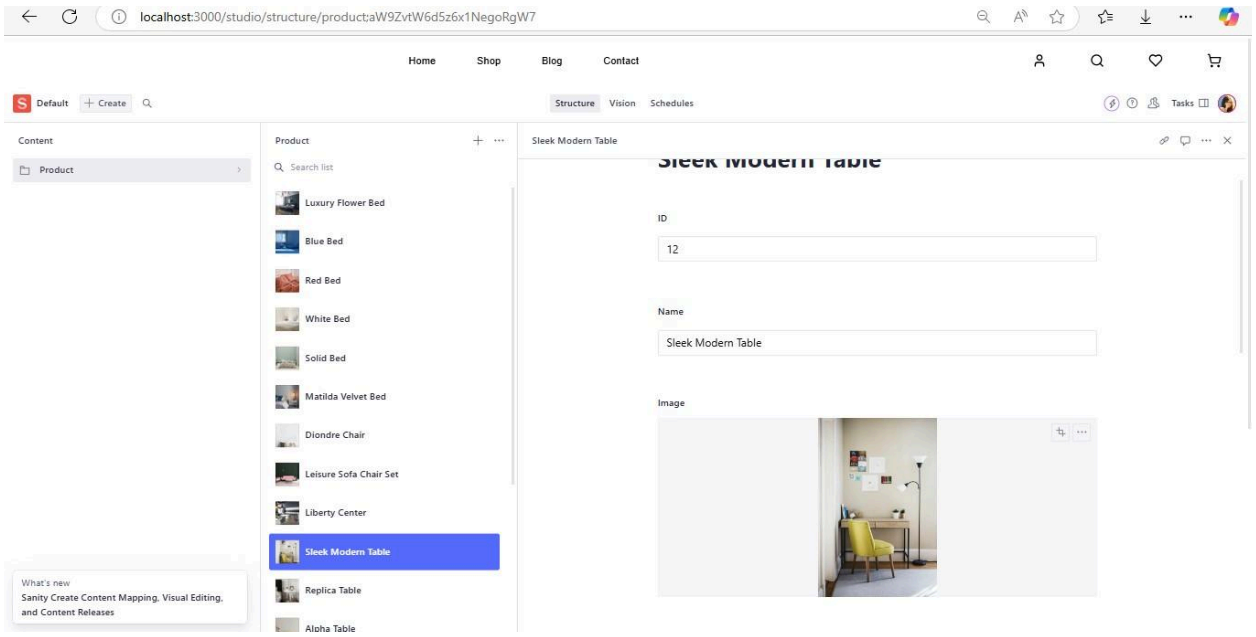
The schema outlines the structure of content in Sanity CMS. Key components include:

- **Title: Name of the product.**
- **Slug: Unique identifier for dynamic routing.**
- **Description: Details about the product.**
- **Price: Numeric field for product cost.**
- **Image: Field to store product images.**



```
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',
        options: {
          hotspot: 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',
      },
```

Sanity Output :


localhost:3000/studio/structure/product;aW9ZvtW6d5z6x1NegoRgW7

ID

12

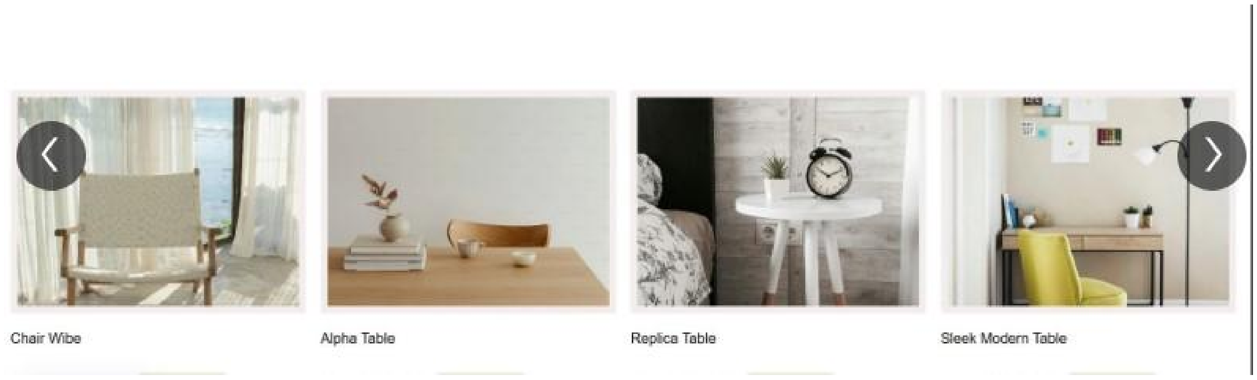
Name

Sleek Modern Table

Image

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases

Products on UI :



Day 3 Completed Successfully:

The API integration and data migration process was successfully completed, including schema setup, data fetching from Sanity, and dynamic routing implementation for the marketplace.