Manahil Nadeem

friday 9-12(Morning)

API Integration Report - [AVION WEBSITE]

Reviewed API Documentation:

- I thoroughly reviewed the provided API documentation for my assigned template to gain a clear understanding of the available endpoint (/products).
- I analyzed the structure of the data returned by the API, including field names, data types, and their respective formats.

Set Up API Calls:

- I utilized Thunder Client to test the /products API endpoint, ensuring the data was returned correctly and matched the expected structure.
- I implemented utility functions in my Next.js project to handle data fetching from the API.
- Using the fetch API, I made GET requests to the API endpoints and stored the responses in variables for further processing.
- I logged the API responses to the console to verify the data structure and ensure accuracy before integrating it into the application.

Compared API Data with Sanity Schema:

- I carefully reviewed the API data structure and compared it with the existing schema in Sanity CMS to identify discrepancies in field names and data types.
- I updated the Sanity schema to align with the API data structure. For example:
- API Field: product_title → Sanity Field: name
- API Field: price → Sanity Field: price (ensuring the correct data type was used).
- API Field: description → Sanity Field: description (added to provide detailed product information).

Data Migration from API to Sanity CMS:

To migrate data from the API to Sanity CMS, I followed these steps:

1. Planning and Setup:

- I decided to use the provided API to fetch data and wrote a script to import it into Sanity CMS.

- **I created a scripts folder and added a migration file (importSanityData.mjs) to handle data fetching and transformation.**

2. Data Transformation and Upload:

- **The migration script fetched data from the API and transformed it into the format required by Sanity CMS.**
- **Using the Sanity client library, I uploaded the transformed data to the CMS, including product details, categories, and other relevant information.**

3. Verification and Validation:

- **After running the migration script, I verified the imported data by checking the Sanity dashboard to ensure all fields were correctly populated and matched the expected structure.**

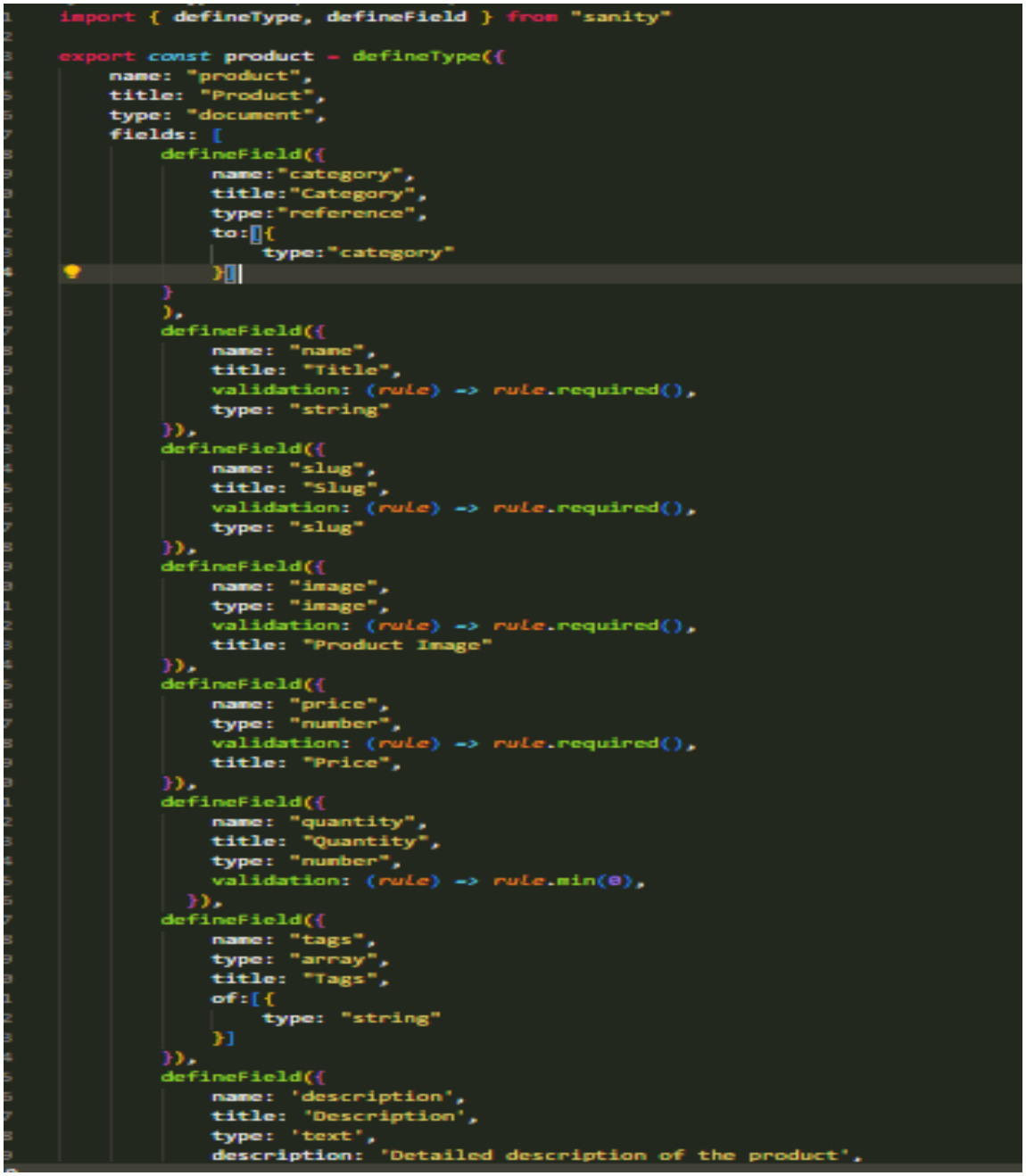
Outcome:

In this project, I successfully integrated the provided API into my Next.js frontend and migrated data into Sanity CMS. I adjusted the Sanity schema to align with the API data structure and ensured the data was accurately displayed on the frontend. This exercise provided me with hands-on experience in API integration, data migration, and schema validation, which are critical skills for building scalable and dynamic marketplaces.

CATEGORY PAGE:

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

export const Category = defineType({
    name: "category",
    title: "Category",
    type: "document",
    fields:[
        defineField({
            name: "name",
            title: "Name",
            type: "string",
            validation: (rule) => rule.required(),
        }),
        defineField({
            name: "slug",
            title: "Slug",
            type: "slug",
            validation: (rule) => rule.required(),
            options: {
                source: "name",
            }
        })
    ]
})
```

Product Page:



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

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        defineField({
            name:"category",
            title:"Category",
            type:"reference",
            to:[{
                type:"category"
            }]
        }
        ),
        defineField({
            name: "name",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        }),
        defineField({
            name: "slug",
            title: "Slug",
            validation: (rule) => rule.required(),
            type: "slug"
        }),
        defineField({
            name: "image",
            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',
```

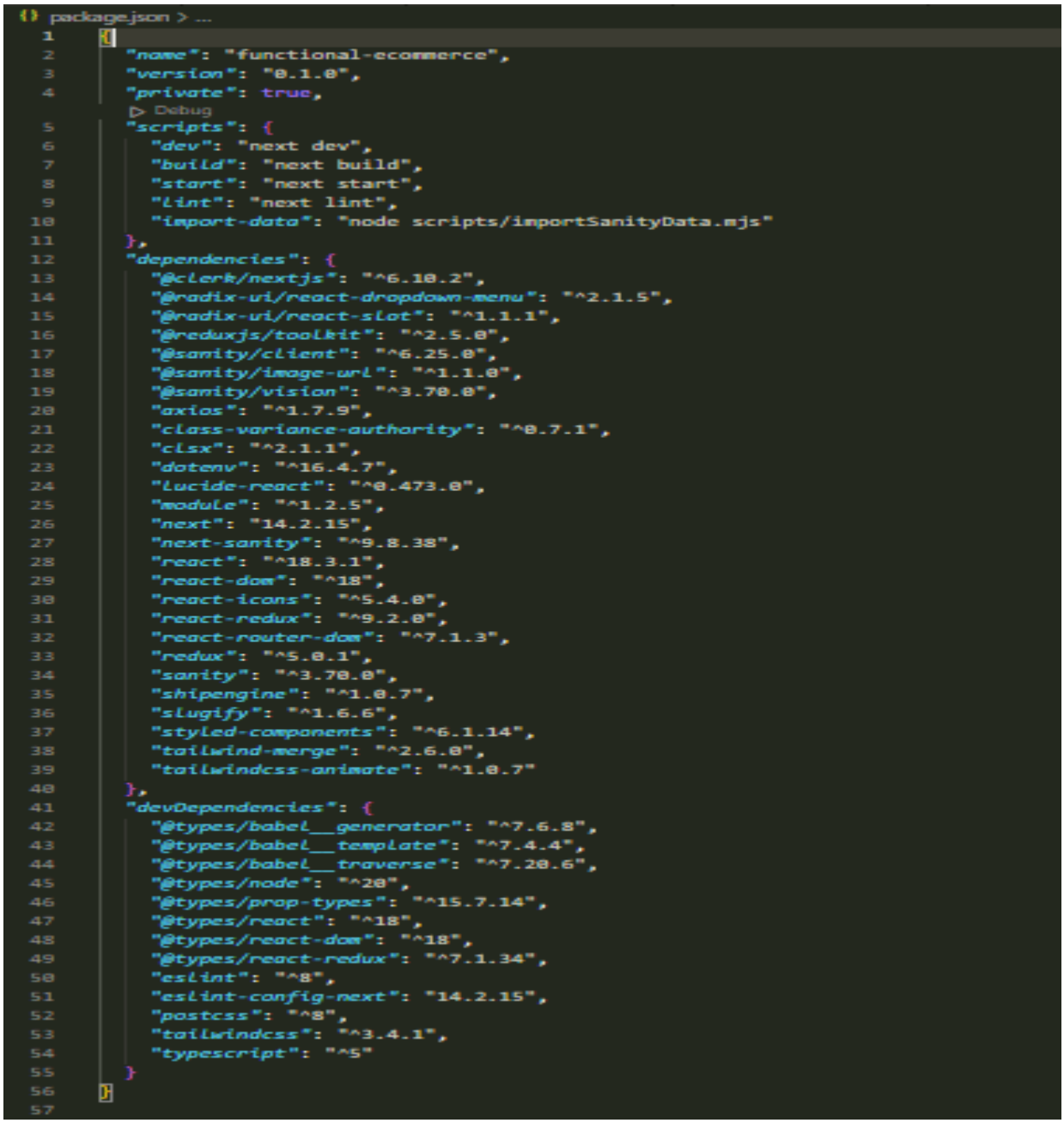
Import file

```
}

export const client = createClient({
    projectId:process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
    dataset:process.env.NEXT_PUBLIC_SANITY_DATASET,
    apiVersion:"v2025-01-18",
    token:process.env.SANITY_API_Token,
    useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
  })

async function uploadImageToSanity(imageUrl) {
    try {
        // Fetch the image from the URL and convert it to a buffer
        const response = await axios.get(imageUrl, { responseType: 'arraybuffer', timeout: 10000 });
        const buffer = Buffer.from(response.data);
        // Upload the image to Sanity
        const asset = await client.assets.upload('image', buffer, {
            filename: imageUrl.split('/').pop(), // Extract the filename from URL
        });
        // Debugging: Log the asset returned by Sanity
        console.log('Image uploaded successfully:', asset);
        return asset._id; // Return the uploaded image asset reference ID
    } catch (error) {
        console.error('❌ Failed to upload image:', imageUrl, error);
        return null;
    }
}

async function createCategory(category, counter) {
    try {
        const categoryExist = await client.fetch(`*[_type=="category" && slug.current==$slug][0]`, { slug: category.slug });
        if (categoryExist) {
            return categoryExist._id;
        }
        const catObj = {
            _type: "category",
            _id: `${category.slug}-${counter}`,
            name: category.name,
            slug: category.slug
        };
        const response = await client.createOrReplace(catObj);
        // Debugging: Log the asset returned by Sanity
        console.log('Category created successfully', response);
        return response._id; // Return the uploaded image asset reference ID
    } catch (error) {
        console.error('❌ Failed to create category:', category.name, error);
    }
}

async function importData() {
    try {
        // Fetch data from external API
        const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
        const products = response.data;
        let counter = 1;
        // Iterate over the products
        for (const product of products) {
            let imageRef = null;
            let catRef = null;
```

Index page

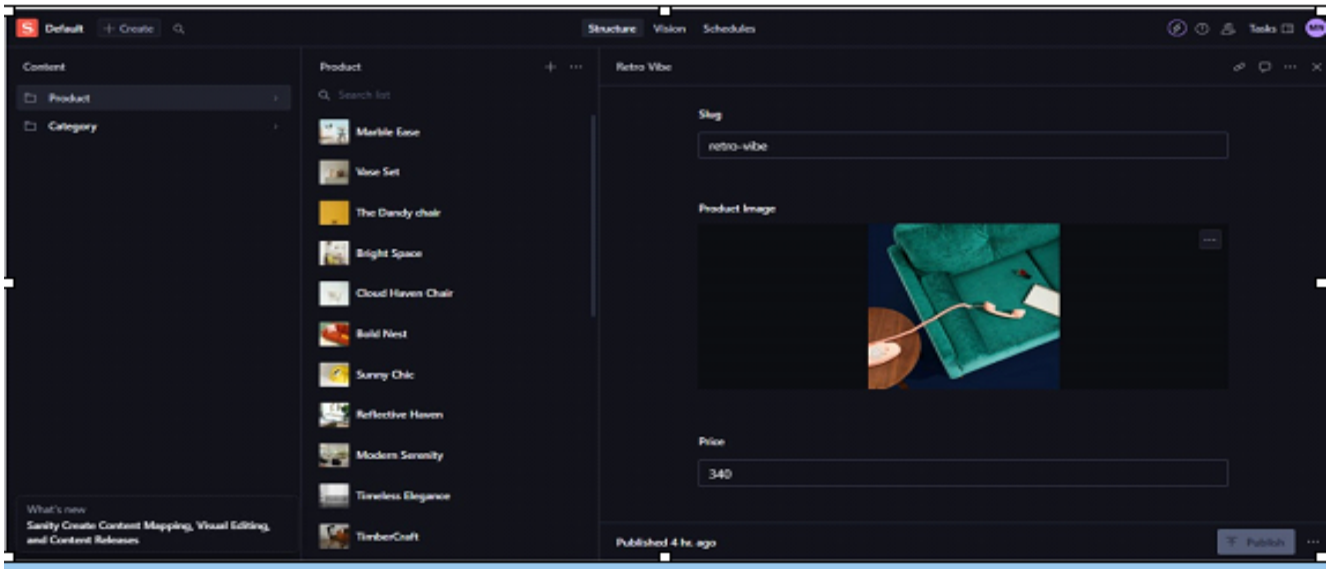
```
import { type SchemaTypeDefinition } from 'sanity'
import { product } from './product'
import { Category } from './category'

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

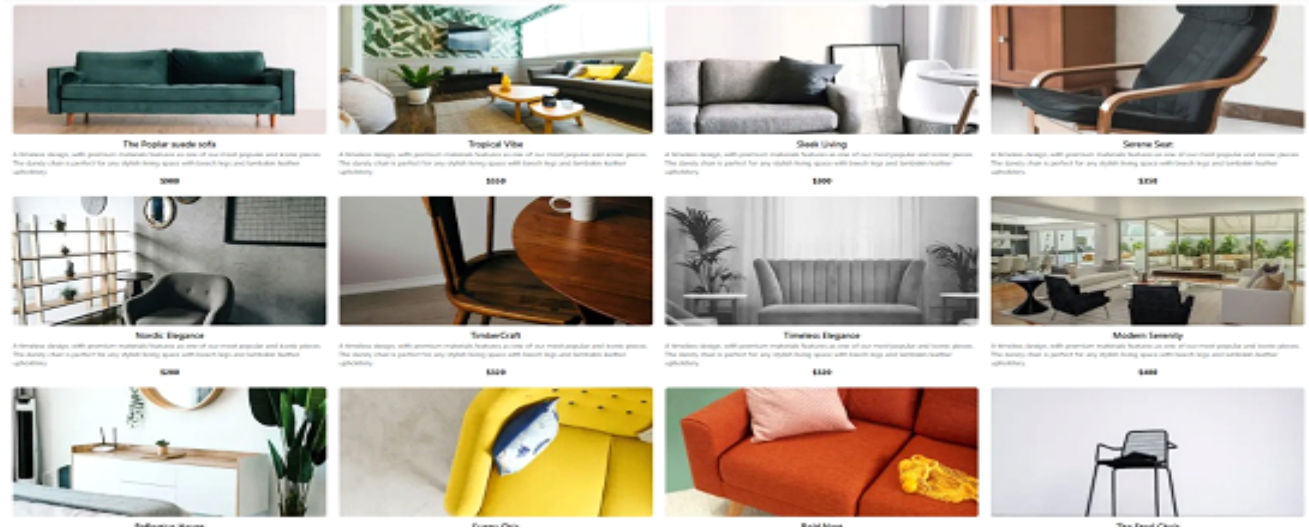
Package.json page



```
{} package.json > ...
{
  "name": "functional-ecommerce",
  "version": "0.1.0",
  "private": true,
  ▷ Debug
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importSanityData.mjs"
  },
  "dependencies": {
    "@clerk/nextjs": "^6.10.2",
    "@radix-ui/react-dropdown-menu": "^2.1.5",
    "@radix-ui/react-slot": "^1.1.1",
    "@reduxjs/toolkit": "^2.5.0",
    "@sanity/client": "^6.25.0",
    "@sanity/image-url": "^1.1.0",
    "@sanity/vision": "^3.70.0",
    "axios": "^1.7.9",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "dotenv": "^16.4.7",
    "lucide-react": "^0.473.0",
    "module": "^1.2.5",
    "next": "14.2.15",
    "next-sanity": "^9.8.38",
    "react": "^18.3.1",
    "react-dom": "^18",
    "react-icons": "^5.4.0",
    "react-redux": "^9.2.0",
    "react-router-dom": "^7.1.3",
    "redux": "^5.0.1",
    "sanity": "^3.70.0",
    "shipengine": "^1.0.7",
    "slugify": "^1.6.6",
    "styled-components": "^6.1.14",
    "tailwind-merge": "^2.6.0",
    "tailwindcss-animate": "^1.0.7"
  },
  "devDependencies": {
    "@types/babel__generator": "^7.6.8",
    "@types/babel__template": "^7.4.4",
    "@types/babel__traverse": "^7.20.6",
    "@types/node": "^20",
    "@types/prop-types": "^15.7.14",
    "@types/react": "^18",
    "@types/react-dom": "^18",
    "@types/react-redux": "^7.1.34",
    "eslint": "^8",
    "eslint-config-next": "14.2.15",
    "postcss": "^8",
    "tailwindcss": "^3.4.1",
    "typescript": "^5"
  }
}
```

Web page


Default
Create
Structure
Vision
Schedules
Tasks
Content
Product
Category
Product
Search list
Marble Ease
Vase Set
The Dandy chair
Bright Space
Cloud Haven Chair
Bold Nest
Sunny Chic
Reflective Haven
Modern Serenity
Timeless Elegance
TimberCraft
What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases
Retro Vibe
Slug
retro-vibe
Product Image
Price
340
Published 4 hr. ago
Publish


The Poplar suede sofa
Tropical Vibe
Sleek Living
Serene Seat
Nordic Elegance
TimberCraft
Timeless Elegance
Modern Serenity