API Integration Report – General E Commerce

1. Understand the Provided API

- **Reviewed API Documentation:** Carefully examined the API documentation for the assigned template, focusing on:
  - **Endpoints:** Identified key endpoints such as /products, /singleProducts.
  - **Methods:** Determined the HTTP methods used for each endpoint (GET, POST, PUT, DELETE).
  - **Request/Response Formats:** Analyzed the expected request payloads and the structure of the API responses.

dataMigration.js U | route.ts ...\products U | productType.ts U | OurProductSection.tsx M

src > app > api > products > route.ts > GET

```
import { client } from "@/sanity/lib/client";
import { NextResponse } from "next/server";

export async function GET() {
  try {
    const data = await client.fetch(`*[_type=="product"]{
    _id,
    title,
    "imageUrl" :productImage.asset -> url,
    price,
    tags,
    dicountPercentage,
    description,
    isNew
}`);

    return  NextResponse.json(data, { status: 200 });
  } catch (error) {
    console.error('Error fetching data from Sanity:', error);
    return new NextResponse('Error fetching data', { status: 500 });
  }
}
```

2. Validate and Adjust Schema

- **Schema Comparison:** Compared the existing Sanity CMS schema defined on Day 2 with the data structure provided by the API.

```
ductType.ts ...\types U    productType.ts ...\schemaTypes M ×    OurProductSection.tsx M
src > sanity > schemaTypes > productType.ts > product > fields
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
```

```
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
            title:"New Badge",
        }
    ]
})
```

3. Data Migration

- **Chosen Method:** Selected **"Using the Provided API"** as the primary data migration method.

dataMigration.js U | route.ts ...\products U | productType.ts ...\types U | productType.ts ...\schemaTypes M

src > dataMigration.js > ...

```javascript
const { createClient } = require("next-sanity");

const client = createClient({
  projectId: "k904gve9",
  dataset: "production",
  apiVersion: "2025-01-04",
  useCdn: true,
  token: "skOZ62uDcOpKL5lovLx8zxO5sfXSj8BxvlDYUi9ZuJFNA59aHwsOTBHetz5oGyEP8F91bvCrE0gf5
})
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      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;
  }
}
```

```
34 async function uploadProduct(product) {
35   try {
36     const imageId = await uploadImageToSanity(product.imageUrl);
37
38     if (imageId) {
39       const document = {
40         _type: 'product',
41         title: product.title,
42         price: product.price,
43         productImage: {
44           _type: 'image',
45           asset: {
46             _ref: imageId,
47           },
48         },
49         tags: product.tags,
50         dicountPercentage: product.dicountPercentage,
51         description: product.description,
52         isNew: product.isNew,
53       };
54
55       const createdProduct = await client.create(document);
56       console.log(`Product ${product.title} uploaded successfully:`, createdProduct);
57     } else {
58       console.log(`Product ${product.title} skipped due to image upload failure.`);
59     }
60   } catch (error) {
61     console.error('Error uploading product:', error);
62   }
63 }
```

```
64
65 async function importProducts() {
66   try {
67     const response = await fetch('https://template6-six.vercel.app/api/products');
68
69     if (!response.ok) {
70       throw new Error(`HTTP error! Status: ${response.status}`);
71     }
72
73     const products = await response.json();
74
75     for (const product of products) {
76       await uploadProduct(product);
77     }
78   } catch (error) {
79     console.error('Error fetching products:', error);
80   }
81 }
82
83 importProducts();
```

- **Data Validation:**
  - Thoroughly validated the imported data in Sanity CMS to ensure accuracy and consistency.
  - Checked for missing fields, incorrect data types, and any other discrepancies.

4. API Integration in Next.js

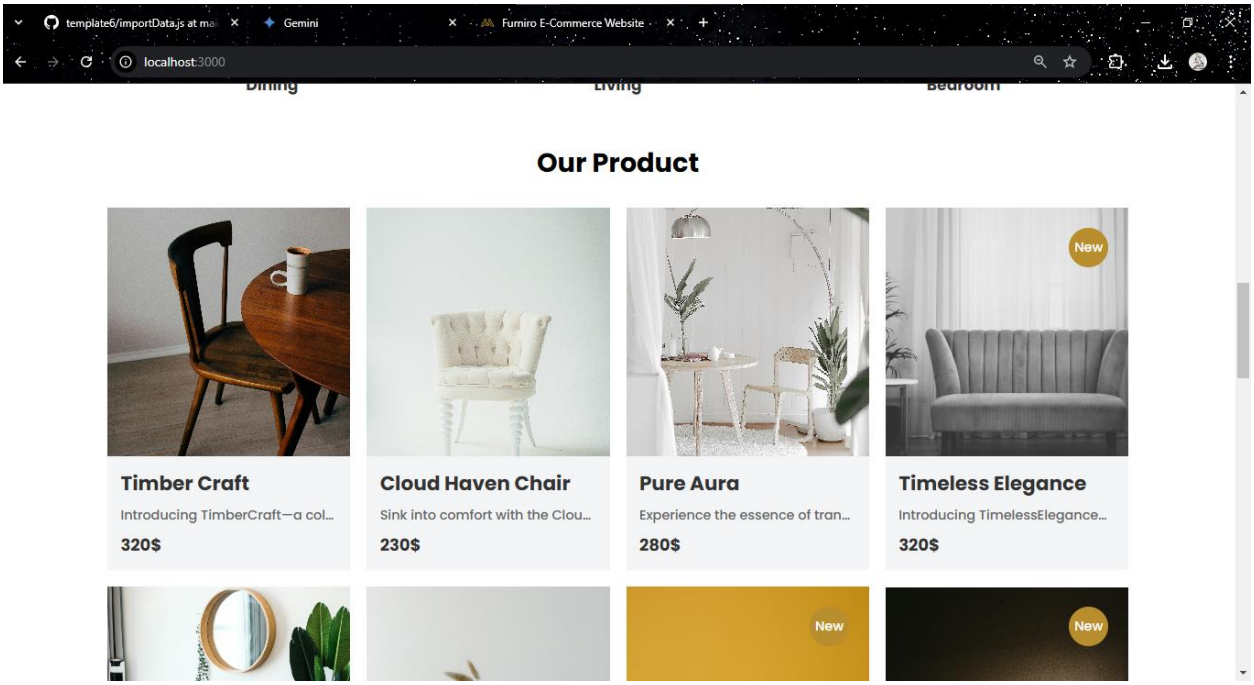
- **Utility Functions:** Created reusable utility functions in Next.js to:
  - Fetch data from the API endpoints.
  - Handle API requests and responses (e.g., error handling, data parsing).
  - Cache API responses to improve performance.

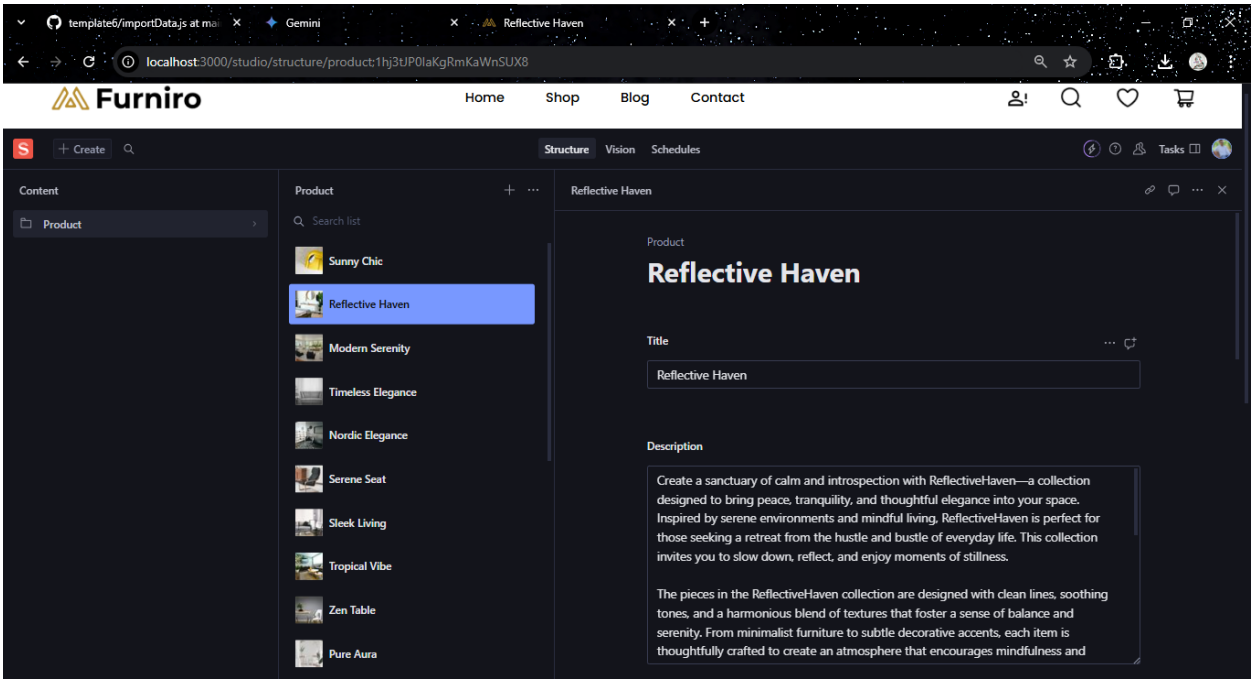
OurProductSection.tsx M | ProductCard.tsx M | page.tsx ...\shop | page.tsx ...\[product_id] M

src > components > sections > OurProductSection.tsx > OurProductSection

```
7   const OurProductSection = () => {
8
9       const [products, setProducts] = useState<Product[]>([]);
10      const [isLoading, setIsLoading] = useState(true);
11
12      useEffect(() => {
13          const fetchData = async () => {
14              try {
15                  const response = await fetch('/api/products');
16                  if (!response.ok) {
17                      throw new Error('Network response was not ok');
18                  }
19                  const data = await response.json();
20                  console.log(data)
21                  setProducts(data.slice(0,12));
22              } catch (error) {
23                  console.error('Error fetching products:', error);
24              } finally {
25                  setIsLoading(false)
26              }
27          };
28
29          fetchData();
30      }, []);
31
32      if (isLoading) {
33          return <div>Loading...</div>;
34      }
35
36      if (!products) {
37          return <div>Error loading products</div>;
38      }
```

- **Component Rendering:** Integrated the API utility functions into the frontend components.
    - Used the fetched data to dynamically render product listings, category pages, and other components.
    - Implemented data fetching and loading states to provide a smooth user experience.


template6/importData.js at ma
Gemini
Furniro E-Commerce Website
localhost:3000
Our Product
Timber Craft
Introducing TimberCraft—a col...
320$
Cloud Haven Chair
Sink into comfort with the Clou...
230$
Pure Aura
Experience the essence of tran...
280$
New
Timeless Elegance
Introducing TimelessElegance...
320$
New
New


template6/importData.js at ma
Gemini
Reflective Haven
localhost:3000/studio/structure/product;1hj3tJP0IaKgRmKaWnSUX8
Furniro
Home
Shop
Blog
Contact
Create
Structure
Vision
Schedules
Tasks
Content
Product
Product
Search list
Sunny Chic
Reflective Haven
Modern Serenity
Timeless Elegance
Nordic Elegance
Serene Seat
Sleek Living
Tropical Vibe
Zen Table
Pure Aura
Reflective Haven
Product
Reflective Haven
Title
Reflective Haven
Description
Create a sanctuary of calm and introspection with ReflectiveHaven—a collection designed to bring peace, tranquility, and thoughtful elegance into your space. Inspired by serene environments and mindful living, ReflectiveHaven is perfect for those seeking a retreat from the hustle and bustle of everyday life. This collection invites you to slow down, reflect, and enjoy moments of stillness.
The pieces in the ReflectiveHaven collection are designed with clean lines, soothing tones, and a harmonious blend of textures that foster a sense of balance and serenity. From minimalist furniture to subtle decorative accents, each item is thoughtfully crafted to create an atmosphere that encourages mindfulness and

- **API Testing:**
    - Utilized tools like Postman and browser developer tools to test API endpoints and verify data integrity.
    - Logged API responses to identify and resolve any issues.