

# Day 3 - API Integration Report - Marketplace E-Commerce Website

## 1. API Integration Process

For our marketplace e-commerce website, we successfully integrated third-party APIs to enhance functionality and ensure seamless data flow. The API was used to fetch product listings, user information, and order details. The integration process involved:

- Fetching data from the API endpoints using Next.js server-side fetching methods.
- Storing and managing fetched data within Sanity CMS.
- Displaying data dynamically on the frontend with Tailwind CSS styling.

## 2. Schema Adjustments

To match the API data with our marketplace structure, we modified the Sanity CMS schema as follows:

- **Product Schema:** Added fields for productID, title, description, price, category, image, and stock.

## 3. Migration Steps and Tools Used

### Migration Steps:

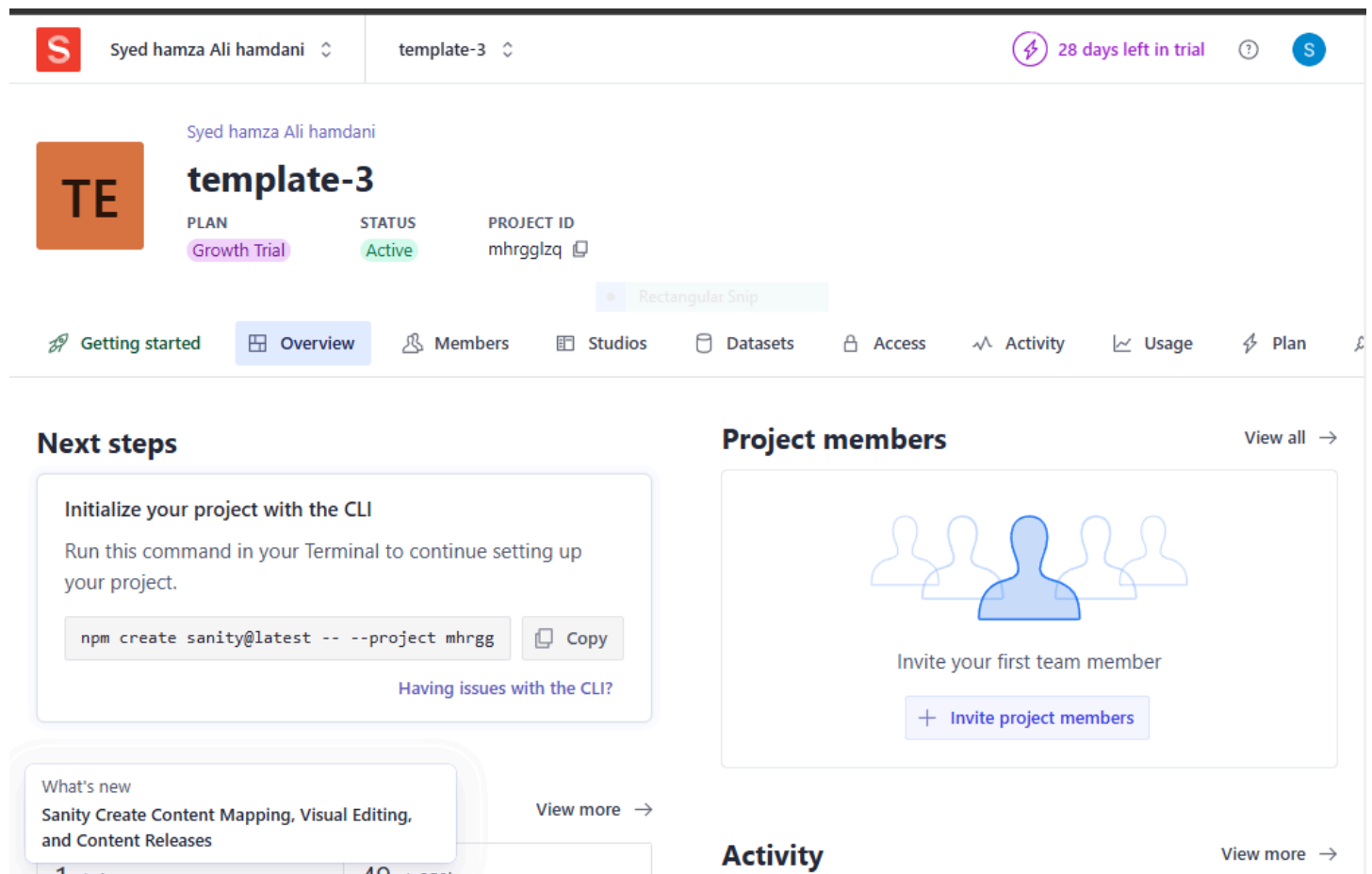
1. Extracted existing data from the API using Axios in Next.js.
2. Processed and formatted data to match our Sanity CMS structure.
3. Uploaded data into Sanity CMS using Sanity's dataset migration script.

4. Verified and tested data consistency in both frontend and backend.

### Tools Used:

- **Axios** for API calls
- **Sanity CLI** for schema adjustments and dataset migrations
- **Postman** for API testing
- **Next.js & Tailwind CSS** for frontend display

## 4. Screenshots



EXPLORER

src > sanity > lib > TS fetch.ts > ...

```
1 import { createClient } from "next-sanity";
2
3 const client = createClient({
4   projectId: "mhrggglzq",
5   dataset: "production",
6   useCdn: false,
7   apiVersion: "2025-01-17",
8 });
9
10 export async function sanityFetch({ query, params = {} }: { query: string, par
11   return await client.fetch(query, params);
12 }
13
```

Codeium: Refactor | Explain | Generate JSDoc | X

EXPLORER

- SECOND-GIAIC...
- node\_modules
- public
- scripts
  - importSanityData.mjs
- src
  - app
  - components
  - lib
    - sanity
      - lib
        - TS client.ts
        - TS fetch.ts
        - TS image.ts
        - TS live.ts
        - TS queries.ts
      - schemaTypes
        - TS index.ts
        - TS product.ts
        - TS env.ts
        - TS structure.ts
  - .env.local
  - .eslintrc.json
  - .gitignore
  - components.json
  - next-env.d.ts
  - next.config.mjs

second-giaic-hakathon-final

src > sanity > lib > TS queries.ts > allproducts

```
1 import { defineQuery } from "next-sanity";
2
3 export const allproducts = defineQuery(`
4   *[_type == "product"]{
5     _id,
6     productName,
7     category,
8     price,
9     inventory,
10    status,
11    description,
12    "imageUrl": image.asset->url,
13    colors
14  }
15 `);
16
```

```
src > app > day-3-work-H3 > page.tsx > Home
17  };
18
19  Codeium: Refactor | Explain | Generate JSDoc | X
20  export default async function Home(){
21    const products : Product[] = await sanityFetch({query : allproducts})
22
23    return(
24      <div>
25        <h1>
26          Products
27        </h1>
28        <div className="grid grid-cols-3 gap-4">
29          {
30            products.map((product)=>(
31              <div className="border p-4 rounded-lg shadow-sm flex flex-col items-
32                {
33                  product.imageUrl && (
34                    <Image
35                      src={urlFor(product.imageUrl).url()}
36                      alt={product.name}
37                      className="w-60"
38                      width={500}
39                      height={500}
40                      unoptimized
41                    />
42                  )
43                }
44              <h2 className="text-xl font-bold text-center">
45                {product.name}
46              </h2>

```


Content


Product


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


Product


Search list


 Nike Waffle One SE


 Nike Metcon 8


 Nike Blazer Mid '77 Vintage


 Nike Renew Run 3


 Nike Zoom Fly 5

 Nike Pegasus 40

 Nike Air Max 270

 Nike Court Vision Low

 Nike Dri-FIT UV Hyverse

 Nike Dunk Low Retro SE

```

        inventory,
        status,
        description,
        "imageUrl": image.asset->url,
        colors
    }
  });

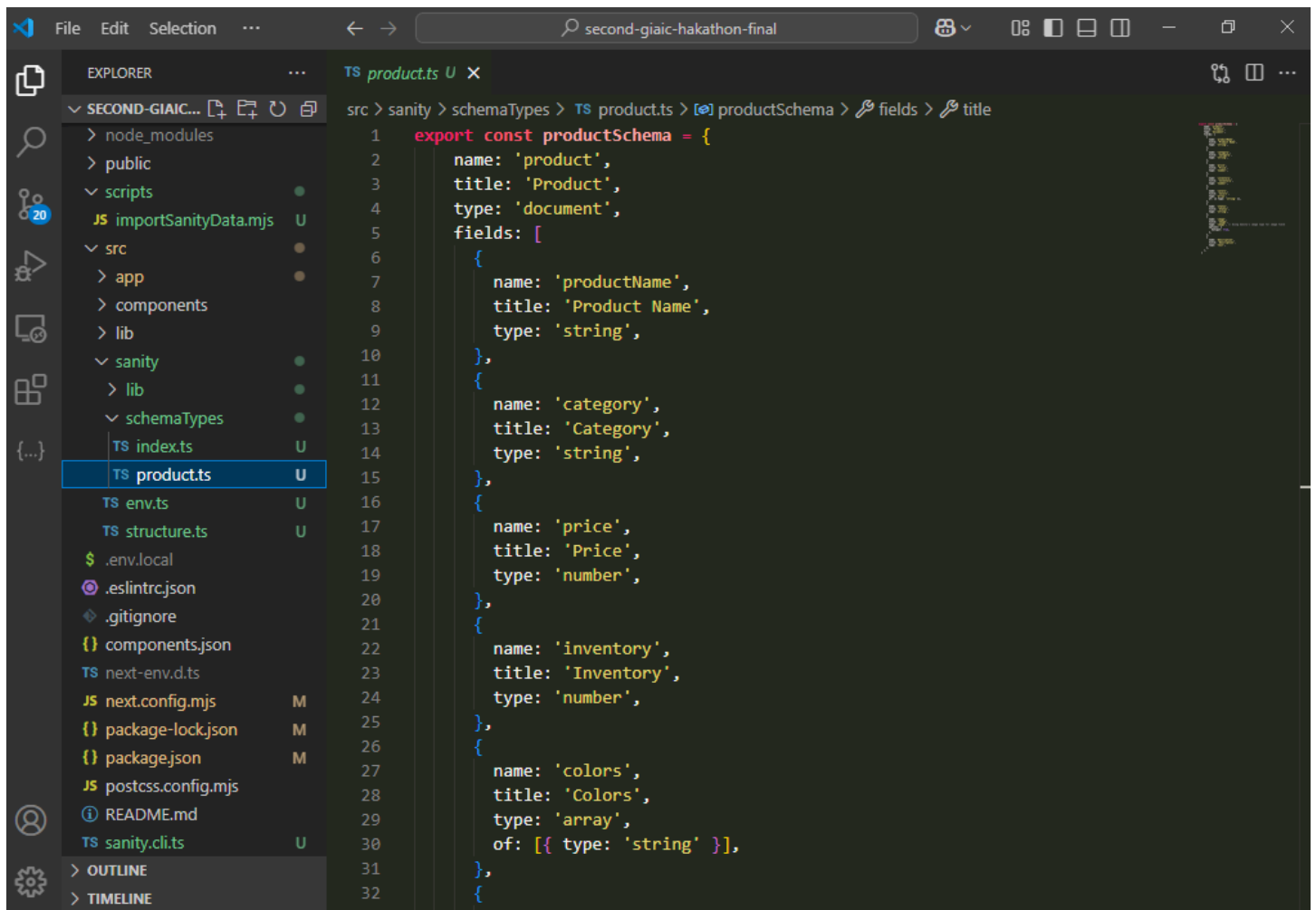
export const fourproducts = defineQuery(`
  *[_type == "product"][0..3]{
    _id,
    productName,
    category,
    price
  }
`);

```

- API call responses in Postman.
- Product listings displayed dynamically on the frontend.
- Sanity CMS populated with API data.

## 5. Code Snippets

API Integration Code:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'src' directory containing 'sanity' and 'schemaTypes'. The 'schemaTypes' directory contains 'product.ts'. The code editor shows the following TypeScript code:

```
1 export const productSchema = {
2   name: 'product',
3   title: 'Product',
4   type: 'document',
5   fields: [
6     {
7       name: 'productName',
8       title: 'Product Name',
9       type: 'string',
10    },
11    {
12      name: 'category',
13      title: 'Category',
14      type: 'string',
15    },
16    {
17      name: 'price',
18      title: 'Price',
19      type: 'number',
20    },
21    {
22      name: 'inventory',
23      title: 'Inventory',
24      type: 'number',
25    },
26    {
27      name: 'colors',
28      title: 'Colors',
29      type: 'array',
30      of: [{ type: 'string' }],
31    },
32  ],
33 }
```

## Conclusion

The API integration was successfully completed, ensuring real-time product updates and seamless backend management. Our e-commerce website now dynamically fetches and displays data from the API while maintaining structured storage in Sanity CMS.

**Submitted by:** Syed Hamza Ali Hamdani

**Project Name:** Marketplace E-Commerce Website