Hackathon MarketPlace Day 3: API Integration Report - [Furniro Furniture website]

By Muhammad fuzail

API INTEGRATION PROCESS

1. Reviewed API Documentation:

- I carefully read the provided API documentation to understand the available endpoint. In my case template 6 there is only one available endpoint (/products).
- I have identified the structure of the data returned by the API, including field names and data types. It is an array of objects so an easy task to understand the structure.
- **I don't need any changes to be made in the schema I have done earlier. Because I first documented the data need then crafted schema so it has helped me alot.**

2. Set Up API Calls

- I have used browser developer tools to test API endpoint (/product) and ensure that data was returned correctly.
- I have created importData.mjs file to fetch data from API.
- Added script to **package.json** to run command "import-data" to fetch data from API and post it to Sanity.

```
"scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importData.mjs"
  },
```

- Now it is time to create product schema in /sanity/schemaTypes/product.ts.

```
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",
        }
    ]
})
```

- Here is the next step to create Sanity client with the token to developer access (read, write, management).
- Environment variables are set in .env file and here is the code to create client.

```
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: true,
  apiVersion: '2025-01-26',
  token: process.env.SANITY_API_TOKEN,
});
```

- Now here is the actual code to post data to sanity with image and product data posting.

```
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', bufferI
      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 uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUr

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo
        description: product.description,
        isNew: product.isNew,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.title} uploaded successf
    } else {
      console.log(`Product ${product.title} skipped due to im
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template6-six.verce

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

Now run the command to import data.

```
npm run import-data
```

Now the data should be inserted in the sanity.

In this project, I successfully integrated the provided API into Next.js frontend and migrated data into Sanity CMS. I don’t need any adjustment to the schema to match the API data structure. This exercise helped me gain practical experience in API integration, data migration, and schema validation, which are essential skills for building scalable marketplaces.

3. Data Access

- I have made a separate file products.ts to which includes all the functions about data access about the products and other data related functionalities. Here is the code of it.

```
import { client } from "@/sanity/client";

export async function getData(query:string) {
  // const query = `*[_type == 'product']`;
  return await client.fetch(query);
}

// const query = `*[_type == 'product']`;
// const result = await client.fetch(query);
// console.log(result)


export async function getSingleProduct(productId: string) {
  const query = `[_type == "product" && id == ${productId}][0
  return await client.fetch(query)
}

export function makeDescriptionShort(desc:string){
  const trimmed = desc.substring(0,50);
  return trimmed + "...";
}
```

- I have to define image type to get images from sanity in proper way.

```
export interface ProductImageType {
    asset:{
      _ref:string
    }
    _type?:"image"
  }
```

- Then I have write logic to convert the image data to actual url to get images.

```

import imageUrlBuilder from "@sanity/image-url"
const builder = imageUrlBuilder(client);

export function urlFor(source:ProductImageType){
    return builder.image(source).url();
}
```

- **Then I have configured the domain of images in next.config.js so Next.js can identify proper source and allow image access.**

```
import type { NextConfig } from "next";

const nextConfig: NextConfig = {
  images: {
    remotePatterns: [
      {
        protocol: "https",
        hostname: "cdn.sanity.io",
        port: "", // Optional: Leave empty if not using a spe
        pathname: "/**", // Allow all paths under this hostna
      },
    ],
  },
};
```

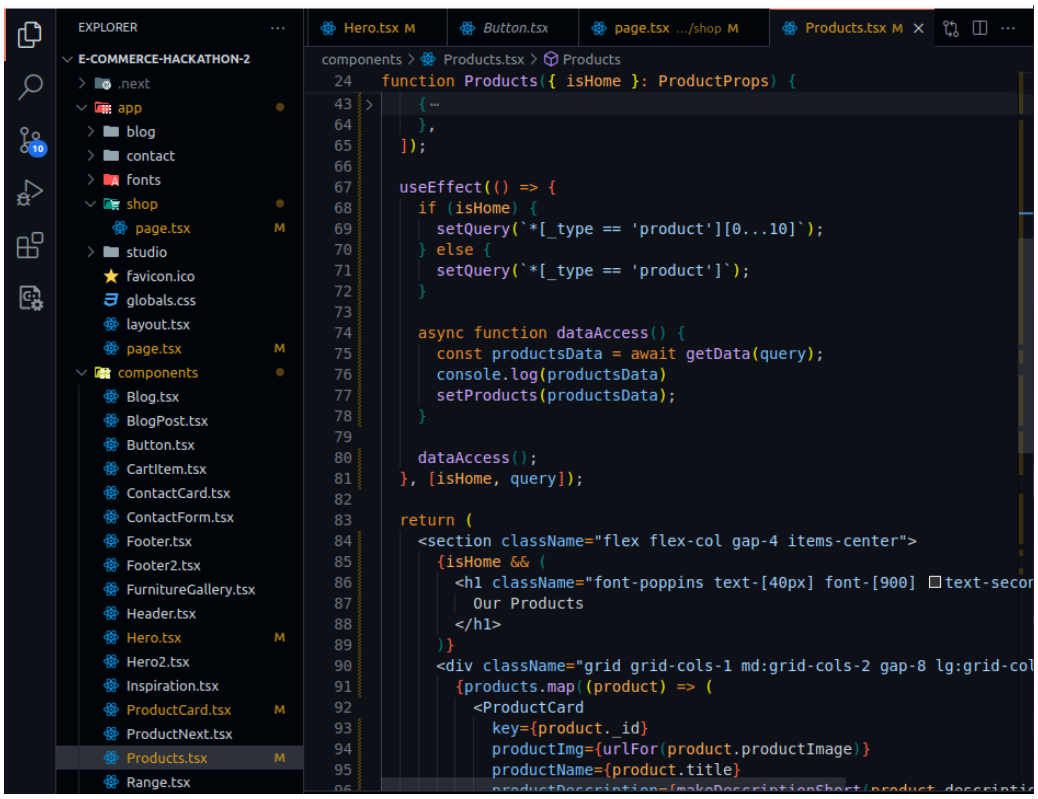
```
export default nextConfig;
```

Here is the table with overview of tasks and their status.

Tasks	**Status**
API Understanding	√√
Schema Validation	√√
Data Migration	√√
API Integration in Next.js	√√
Submission Preparation	√√

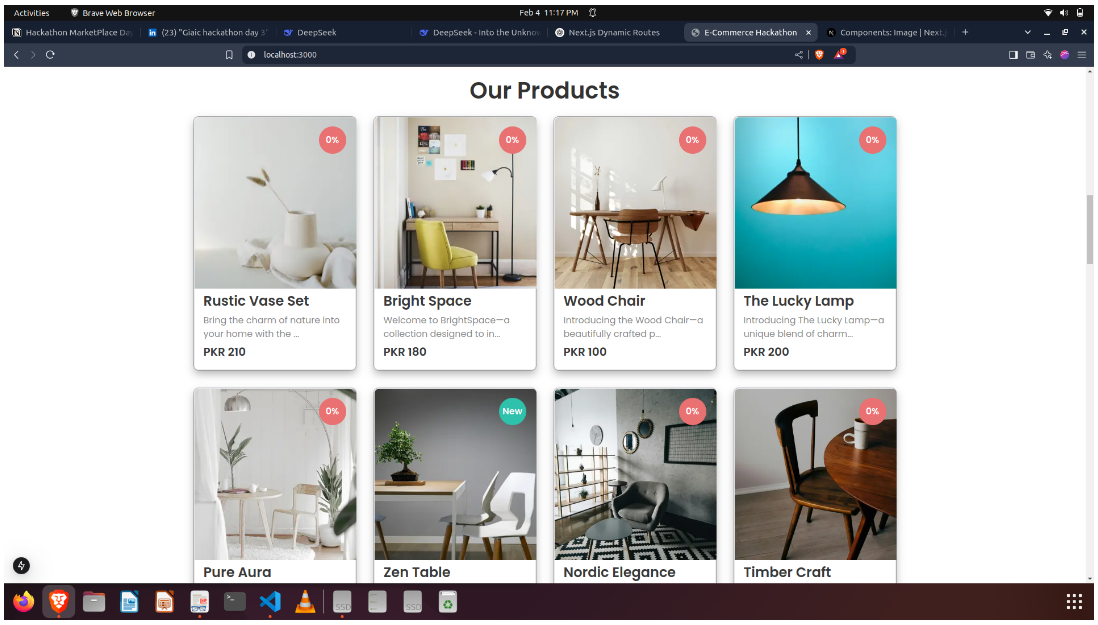
4. Screenshots

- **API Calls**



```
function Products({ isHome }: ProductProps) {
    {…
    },
  ]);

  useEffect(() => {
    if (isHome) {
      setQuery(`*[_type == 'product'][0...10]`);
    } else {
      setQuery(`*[_type == 'product']`);
    }

    async function dataAccess() {
      const productsData = await getData(query);
      console.log(productsData)
      setProducts(productsData);
    }

    dataAccess();
  }, [isHome, query]);

  return (
    <section className="flex flex-col gap-4 items-center">
      {isHome && (
        <h1 className="font-poppins text-[40px] font-[900] ☐text-secor
          Our Products
        </h1>
      )}
      <div className="grid grid-cols-1 md:grid-cols-2 gap-8 lg:grid-col
        {products.map((product) => (
          <ProductCard
            key={product._id}
            productImg={urlFor(product.productImage)}
            productName={product.title}
```

- **Data Displayed in frontend.**
  - **Home Page**



Our Products

Rustic Vase Set
0%
Bring the charm of nature into your home with the ...
PKR 210

Bright Space
0%
Welcome to BrightSpace—a collection designed to in...
PKR 180

Wood Chair
0%
Introducing the Wood Chair—a beautifully crafted p...
PKR 100

The Lucky Lamp
0%
Introducing The Lucky Lamp—a unique blend of charm...
PKR 200

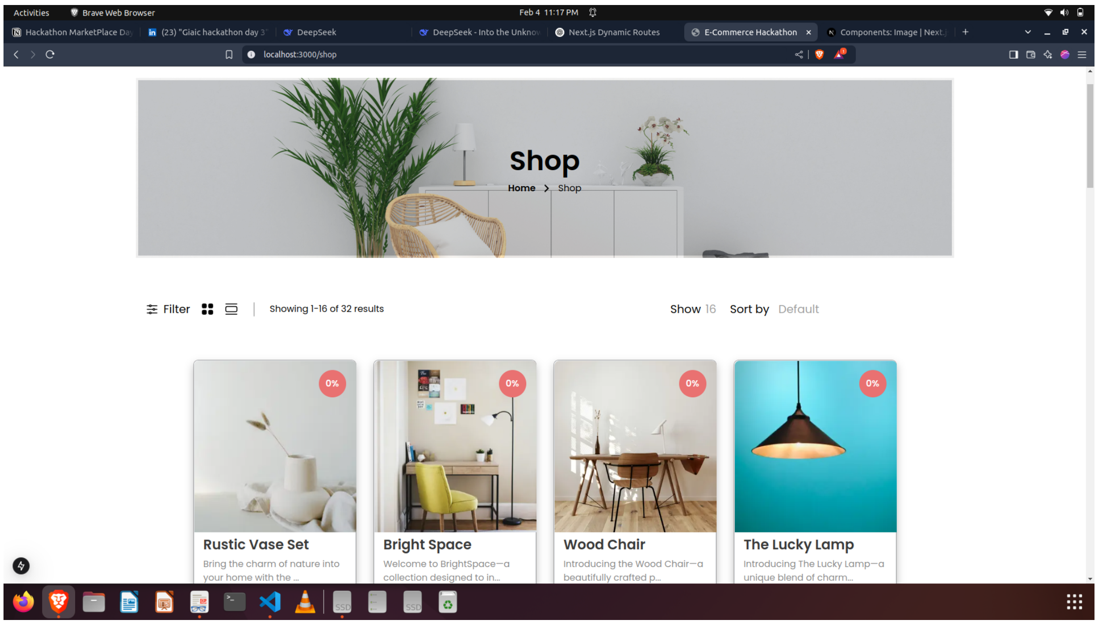
Pure Aura
0%

Zen Table
New

Nordic Elegance
0%

Timber Craft
0%

- ◦ **Shop Page**



Shop
Home > Shop

Filter | Showing 1–16 of 32 results   Show 16   Sort by Default

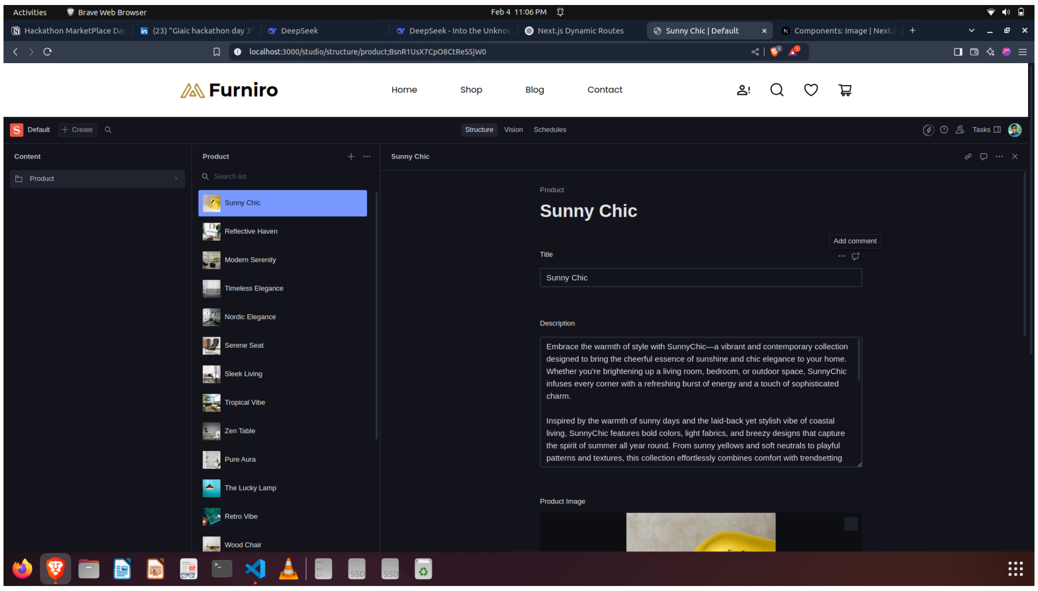
Rustic Vase Set
0%
Bring the charm of nature into your home with the ...

Bright Space
0%
Welcome to BrightSpace—a collection designed to in...

Wood Chair
0%
Introducing the Wood Chair—a beautifully crafted p...

The Lucky Lamp
0%
Introducing The Lucky Lamp—a unique blend of charm...

- **Sanity with populated data.**



Product

Sunny Chic

Title

Sunny Chic

Description

Embrace the warmth of style with SunnyChic—a vibrant and contemporary collection designed to bring the cheerful essence of sunshine and chic elegance to your home. Whether you're brightening up a living room, bedroom, or outdoor space, SunnyChic infuses every corner with a refreshing burst of energy and a touch of sophisticated charm.

Inspired by the warmth of sunny days and the laid-back yet stylish vibe of coastal living, SunnyChic features bold colors, light fabrics, and breezy designs that capture the spirit of summer all year round. From sunny yellows and soft neutrals to playful patterns and textures, this collection effortlessly combines comfort with trendsetting

Product Image