Day 3 - API Integration Report – BI Structure Ecommerce

This document involves integrating APIs and migrating product data into Sanity CMS. The aim is to fetch product data using Axios, store it in Sanity, and render the data on the frontend built with Next.js.

Objectives

1. Integrate an external API to fetch product data.
2. Migrate the fetched data to Sanity CMS.
3. Retrieve data from Sanity CMS and display it dynamically on the frontend.
4. Ensure robust error handling and clean code practices.

API Integration Steps

Fetch Data Using Axios:

- Utilize Axios to make GET requests to the external API.
- Validate and transform the response data to match the Sanity schema.

Migrate Data to Sanity CMS:

- Use the Sanity client to create or update product documents in the CMS.
- Ensure schema compatibility between the API data and Sanity.

Create API Endpoints (/api/products):

- Fetch data from Sanity using GROQ queries.
- Return the product data in response.

Render Data in Frontend:

- Fetch data from API endpoint like /api/produtcs.
- Display product data dynamically in the frontend using Next.js.

Testing and Debugging:

- Use Thunder Client to test API responses.
- Validate data in Sanity CMS to ensure accuracy.
- Test the frontend rendering for proper integration.

Code Snippets:

Data-migration.mjs

```
async function importData() {
  try {
    console.log('migrating data please wait...');

    // API endpoint containing car data
    const response = await axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;
    console.log("products ==>> ", products);


    for (const product of products) {
      let imageRef = null;
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }

      const sanityProduct = {
        _type: 'product',
        productName: product.productName,
        category: product.category,
        price: product.price,
        inventory: product.inventory,
        colors: product.colors || [], // Optional, as per your schema
        status: product.status,
        description: product.description,
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      await client.create(sanityProduct);
    }
```

API endpoint (/api/products)

```ts
import { client } from '@/sanity/lib/client';
import { NextRequest, NextResponse } from 'next/server';

export async function GET() {
  try {
    const data = await client.fetch(`*[_type == "product"]`);

    return NextResponse.json({ data }, { status: 200 });
  } catch (error) {
    console.error('Error fetching posts:', error);
    return NextResponse.json({ error: 'Failed to fetch posts' }, { status: 500 });
  }
}

export async function POST(req:NextRequest) {
  const { slug } = await req.json();
  try {
    const data = await client.fetch(`*[_type == "product" && slug.current == "${slug}"][0]`);

    return NextResponse.json({ data }, { status: 200 });
  } catch (error) {
    console.error('Error fetching posts:', error);
    return NextResponse.json({ error: 'Failed to fetch posts' }, { status: 500 });
  }
}
```

ProductDetails Page

```ts
const ProductDetail = ({ params }: { params: { productdetails: string } }) => {
  const { productdetails } = params;
  const [productData, setProductData] = useState<Product|undefined>(undefined)
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);
  const [quantity, setQuantity] = useState<number>(1)
  const { addToCart,cart } = useCart();

  useEffect(()=>{
    const fetchData = async() =>{
      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products`,{
          cache:'no-cache',
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify({ slug: productdetails })

        })
        if (!response.ok) {
          throw new Error(`Failed to fetch data: ${response.status}`);
        }
        const data = await response.json()
        setProductData(data.data);
      } catch (error: unknown) {

        if (error instanceof Error) {
          setError(error.message);
          console.error(error)
        } else {
          setError('An unexpected error occurred.');
        }
      }finally {
        setIsLoading(false);
      }
    }
```

This document provides a comprehensive guide to the API integration and data migration process for the Nike Ecommerce project, showcasing the technical foundation and implementation roadmap.