Day 3 API Integration Report

(NIKE SHOES)

Day 3: API Integration Process

Overview

This report details the API integration process for the e-commerce project, highlighting both backend and frontend aspects. It includes Sanity CMS integration, schema design, migration script implementation, and API usage.

Backend Integration

API Details

- **Endpoint:** https://template-03-api.vercel.app/api/products
- **Data Structure**: The API provides product data, including fields such as:

o productName

o category

o price

o inventory

o colors

o status

o description

o image

Tools Used

- **Sanity Client**: For creating and managing content in Sanity CMS.
- **Axios:** For making HTTP requests to fetch data and images.
- **Dotenv:** To securely manage environment variables.

Sanity Schema

The product schema was designed as follows:

```
export const productSchema = {

  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'productName',
      title: 'Product Name',
      type: 'string',
    },
    {
      name: 'category',
      title: 'Category',
      type: 'string',
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
    {
      name: 'inventory',
      title: 'Inventory',
      type: 'number',
    },
    {
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],
    },
    {
```

```
      name: 'status',
      title: 'Status',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Image',
      type: 'image', // Using Sanity's image type for image field
      options: {
        hotspot: true,
      },
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
  ],
}
```

ct schema was designed as follows:

Migration Script

The following migration script was implemented to fetch data from the API and populate Sanity CMS:

```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
```

```
  apiVersion: '2021-08-31'
});


async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      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 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);
    }

    console.log('Data migrated successfully!');
  } catch (error) {
    console.error('Error in migrating data ==>> ', error);
  }
}

importData();
```

Frontend Integration

Nike Dunk Low Retro SE
Beige
MRP: 9695

Nike Blazer Mid '77 Vintage
WhiteBlack
MRP: 8495

Nike Air Force 1 Mid '07
White
MRP: 10795

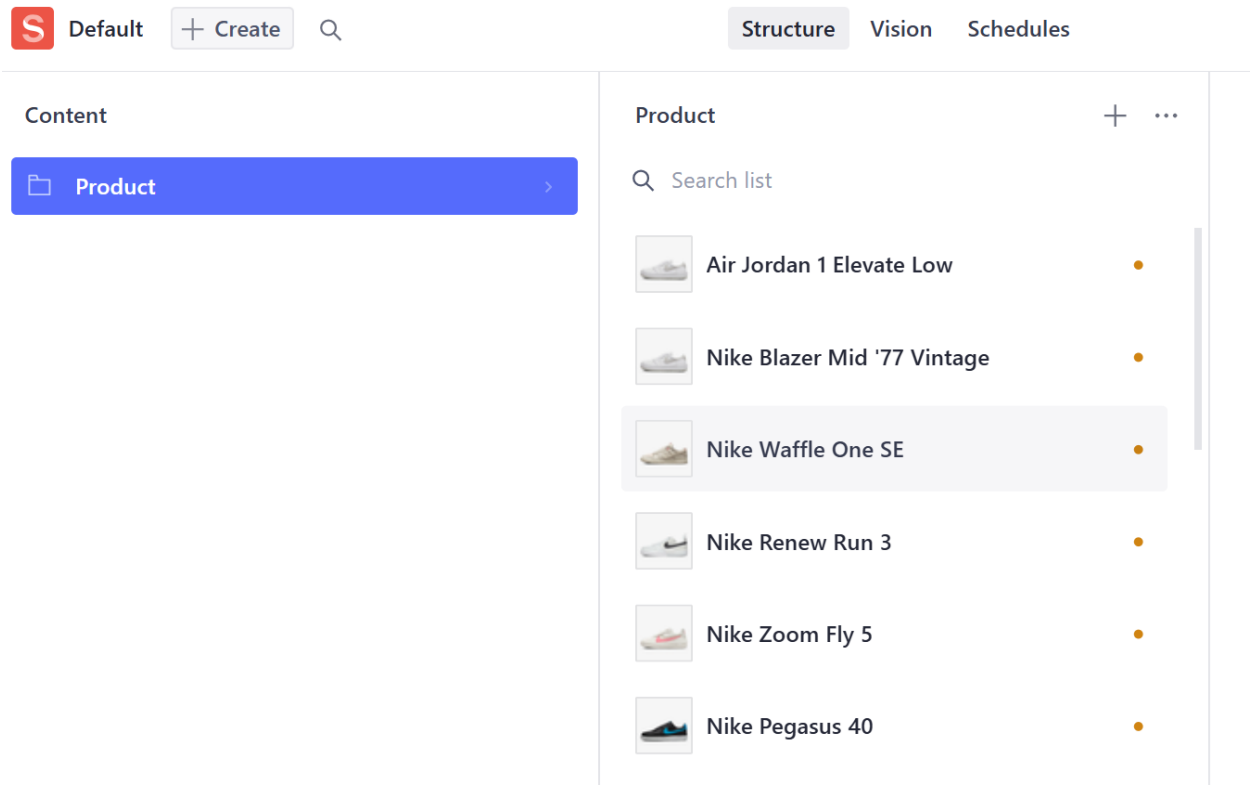
Nike Dri-FIT UV Hyverse
Teal
MRP: 2495

Data Fetching and Display

The frontend uses Axios to retrieve data from the API and displays the products on the website. It also pulls images and other fields directly from Sanity CMS, ensuring seamless integration.

Sanity Configuration

The project uses Sanity's asset manager to handle images and document references. Hotspot□enabled images improve user experience with better cropping and scaling.



Tools Used

- **Axios:** To fetch API data and integrate with Sanity.
- **Sanity Asset Manager:** For efficient image handling.
- **React Components:** For dynamic data rendering.

Conclusion

This integration successfully combines API data with Sanity CMS, providing a robust backend solution. The frontend dynamically displays data fetched from the backend, ensuring a seamless user experience. The combination of schema design, migration script, and API usage reflects a well-structured approach to modern e-commerce development.

Final CheckList

API Understanding	Schema Validation	Data Migration	API Integration in Next.js	Submission Preparation
✓	✓	✓	✓	✓