DAY 3 (HACKATHON)

Documentation: API Integration, Schema Adjustments, and Data Migration Process

Introduction

This documentation outlines the process of integrating external APIs with a Next.js project, adjusting schemas in Sanity CMS, and migrating product data to Sanity. The goal is to import product data, including product details and images, from an external API and populate a Sanity CMS database, which can then be used to render data on a front-end website.

Overview of Steps

1. **API Integration**: Fetching product data from an external API.
2. **Sanity Schema Adjustments**: Modifying the Sanity schema to store product data such as product name, category, price, and image.
3. **Data Migration**: Migrating product data from the external API to Sanity, including handling image uploads to Sanity's asset manager.

1. API Integration

API Endpoint and Fetching Product Data

The external API provides product data, which includes product names, categories, prices, descriptions, inventory, and image URLs. The process involves fetching this data and using it to populate our Sanity CMS.

API Request Setup:

We use **Axios** for making API requests to fetch the product data from an endpoint.

```
import axios from 'axios';


async function fetchData() {
  try {
    const response = await
axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;
    console.log("Products fetched:", products);
  } catch (error) {
    console.error('Error fetching product data:', error);
  }
}


fetchData();
```

This function sends a GET request to the API, retrieves product data, and logs it. This product data is later used to populate the Sanity CMS.

---

2. Sanity Schema Adjustments

Sanity uses schemas to define the structure of content. For this project, we need to create a schema for storing product data such as productName, category, price, description, and image.

Existing Schema:

Here is an example of how the schema for products might look in Sanity:

```
export default {
  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: 'description',
  title: 'Description',
  type: 'text',
},
{
  name: 'image',
  title: 'Image',
  type: 'image',
  options: {
    hotspot: true,
  },
    },
  ],
};
```

- **Product Name, Category, Price**: These are stored as strings and numbers in the schema.
- **Image**: The product image is of type `image` in Sanity, with additional options like `hotspot` to allow users to crop or focus on certain parts of the image.

We have ensured that the necessary fields are available in the schema for importing data like product names, descriptions, images, etc.

---

3. Data Migration Process

Migration Script Overview

The migration process involves two major tasks:

1. **Uploading images to Sanity's Asset Manager**: Images are fetched from external URLs and uploaded to Sanity.
2. **Creating product entries in Sanity**: For each product, we create a document in the Sanity CMS with the corresponding data.

Steps in the Migration Script

1. Setting Up Sanity Client

To interact with the Sanity API, we need to set up the Sanity client using the provided credentials in .env.local.

```
import { createClient } from '@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',
});
```

- projectId: Your Sanity project ID.
- dataset: The dataset within the Sanity project.
- useCdn: Set to false for real-time data fetching during development.

2. Uploading Images to Sanity

The images are fetched from the external URLs and uploaded to Sanity using the Sanity API.

```
import axios from 'axios';
```

```
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;
  }
}
```

- `axios.get(imageUrl)`: Fetches the image as an `arraybuffer`.
- `client.assets.upload`: Uploads the image as a `buffer` to the Sanity Asset Manager.

3. Creating Product Documents in Sanity

For each product, a new document is created in the `product` schema.

javascript

CopyEdit

```
async function importData() {
  try {
    const response = await
axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;

    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 || [],
        status: product.status,
        description: product.description,
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      await client.create(sanityProduct);
      console.log(`Product ${product.productName} created successfully.`);
    }

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

importData();
```

- **uploadImageToSanity**: First, we upload the product image to Sanity and retrieve its reference (_id).
- **client.create(sanityProduct)**: This creates a new product document in the Sanity dataset.

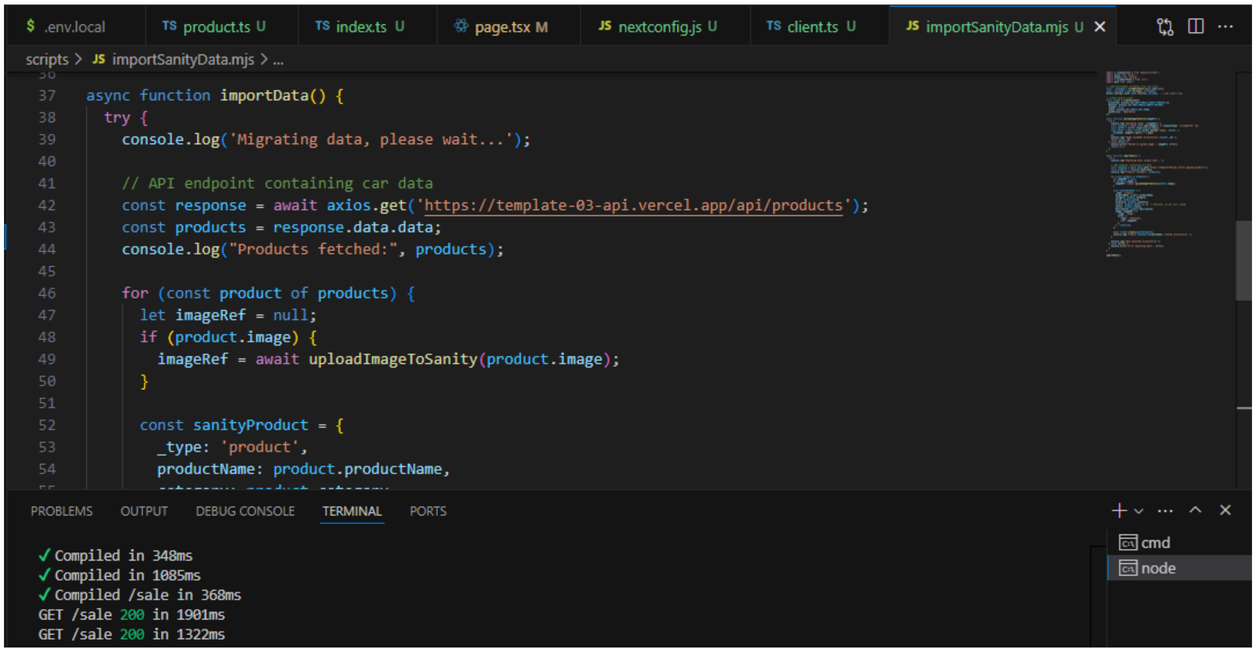
Tools and Technologies Used

- **Sanity CMS**: Headless CMS used for managing and storing product data.
- **Axios**: For fetching data from the external API.

- **Sanity API**: For interacting with the Sanity project and uploading assets.
- **Next.js**: The React-based framework for building the front-end application.
- **dotenv**: For managing environment variables like API keys securely.

SCREENSHOTS OF :

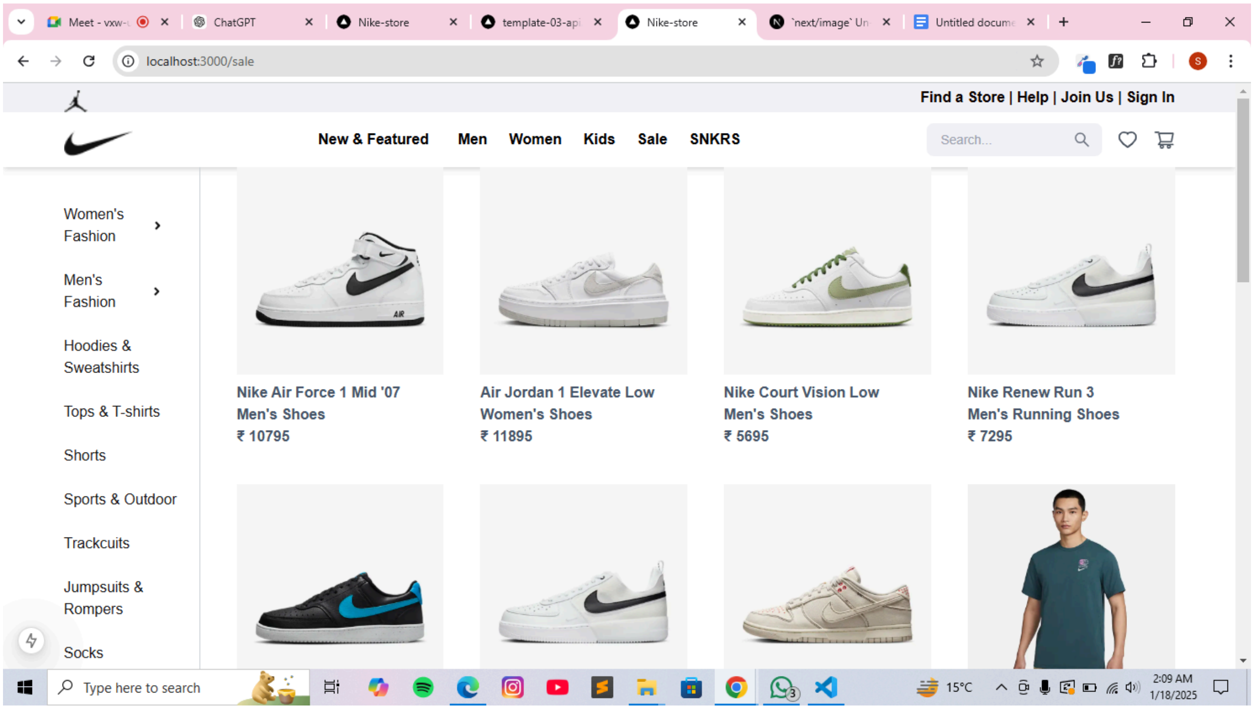
API calls:



```
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 fetched:", products);

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

      const sanityProduct = {
        _type: 'product',
        productName: product.productName,
```

```
✓ Compiled in 348ms
✓ Compiled in 1085ms
✓ Compiled /sale in 368ms
GET /sale 200 in 1901ms
GET /sale 200 in 1322ms
```

Data successfully displayed in the frontend:



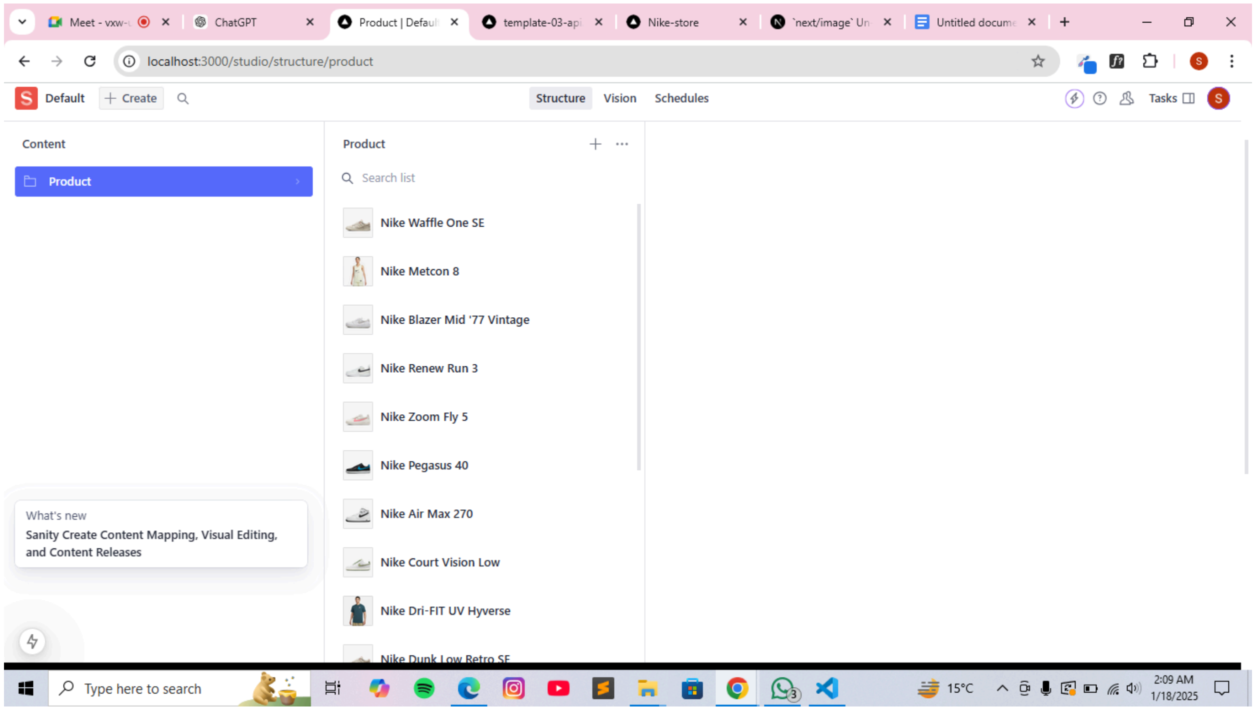
Nike Air Force 1 Mid '07
Men's Shoes
₹ 10795

Air Jordan 1 Elevate Low
Women's Shoes
₹ 11895

Nike Court Vision Low
Men's Shoes
₹ 5695

Nike Renew Run 3
Men's Running Shoes
₹ 7295

Populated Sanity CMS fields:



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

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