Day 3 - API Integration Report - Clothing E-commerce Store

1. Introduction

The objective of this task was to integrate APIs into the Sanity CMS and migrate data to build a functional marketplace backend. This process is vital in e-commerce development as it ensures that product and category data are dynamically fetched and displayed. In this documentation, I will describe the API integration and data migration steps followed, ensuring compatibility with my assigned template.

2. API Overview

I utilized the following API for data migration and integration:
- API Documentation:
https://github.com/developer-hammad-rehman/template1/blob/main/README.md
- Key Endpoints:
  - /products: Fetch product listings.
  - /categories: Fetch product categories.

3. Schema Validation

I validated and adjusted my Sanity CMS schema to ensure compatibility with the API. Several fields required mapping, for example:
- API field: product_title
- Schema field: name

The schema adjustments ensured that the API data matched the Sanity structure.

4. Data Migration Process

I used a script-based method to migrate data from the API into Sanity CMS. The steps followed include:
1. Fetching data from the API.
2. Transforming the data to match the Sanity schema.
3. Populating the Sanity CMS with the transformed data.

The migration script utilized can be found at:
https://github.com/developer-hammad-rehman/template1/blob/main/importData.js.

5. API Integration in Next.js

I integrated the API with my Next.js project by creating utility functions to fetch API data. The data is then rendered in the frontend components. Here are the steps followed:
1. Created utility functions to fetch data.
2. Rendered the API data in Next.js components.
3. Tested API calls using Postman and ensured data consistency.

The code for the utility functions and rendering components can be found in my project repository.

6. Error Handling

I handled potential errors during data migration and API integration by implementing error logging and fallback mechanisms. If the API failed to return data, a skeleton loader was used to enhance the user experience. The error handling code is part of my utility functions.

7. Output and Screenshots

Below are the outputs of the migration and API integration process:
- The Sanity CMS was populated with the product data from the API.
- The frontend displayed the product listings and categories fetched from the API.

8. Best Practices Followed

Throughout the project, I adhered to the following best practices:
- Used .env files to store sensitive API keys.
- Followed clean code principles and modularized utility functions for better reusability.
- Validated data during migration to ensure schema alignment.
- Used version control to track changes and commits throughout the project.

9. Conclusion

The API integration and data migration process were successfully completed for my e-commerce store project. I gained valuable skills in API integration, data handling, and schema validation. Going forward, I plan to enhance the project by implementing more dynamic features and refining the user interface.