# Day 3 - API Integration Report - [E-commerce Website]

## Objective

This document outlines the process of API integration and data migration using Sanity CMS. The goal was to fetch product data via an API, migrate it into Sanity CMS, and display it dynamically on the frontend using Next.js.

---

## Table of Contents



---

## Overview

The assignment involved integrating an API into a Next.js project, fetching product data from Sanity CMS, and ensuring schema compatibility to display data seamlessly. The objective was to develop a dynamic and scalable marketplace.

---

## API Integration Process

### API Overview

- **API Endpoint Used**: [https://template-03-api.vercel.app/api/products]

- **Purpose**: To fetch product data including titles, images, prices, and descriptions.

### Fetching Data from Sanity API

- Sanity API was utilized to query product data using GROQ (Graph-Relational Object Queries).
- The query fetched product details such as name, price, category, and image.

---

# Schema Validation and Adjustments

- **Sanity Schema Fields**:
  - Name (String)
  - Price (Number)
  - Category (Reference)
  - Image (Image)
- **API Schema Fields**:
  - Product Name → Name
  - Product Price → Price
  - Product Category → Category
  - Product Image URL → Image
- **Adjustments**:
  - Field names were mapped appropriately.
  - Schema types were validated for compatibility (e.g., price as a number).

---

# Data Migration Process

- **Steps Taken**:
  - Data was fetched using an API.
  - Data fields were transformed to match Sanity schema.
  - Data was successfully uploaded to Sanity CMS.

---

# Frontend API Integration

- Reusable functions were created to fetch and display data dynamically in the frontend.
- The data was rendered in a structured layout for better user experience.

---

# Error Handling

- **Techniques Used**:
  - Implemented error logs for debugging.
  - Added fallback UI for seamless user experience in case of API failures.

---

# Expected Output

- A fully populated Sanity CMS with product data.
- A functional frontend dynamically displaying product listings.

---

# Screenshots and Evidence

1. Screenshot showing products populated in Sanity CMS.
2. Screenshot showing successful API responses.
3. Screenshot of the frontend displaying product data.

---

# Best Practices Followed

1. Securely stored sensitive data in environment variables.
2. Modularized code for better maintainability.
3. Validated schemas before migration to avoid compatibility issues.
4. Ensured version control for smooth collaboration.

---

# Self-Validation Checklist

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

---

1. Adjust your Sanity schema or transform data during migration.

**Prepared By:**

[Qareer-ul-Ain]