“Day 3 - API Integration Report - [General E-Commerce]”

Introduction:

1. Understand how to integrate APIs into your Next.js project.
2. Learn to migrate data from APIs into Sanity CMS.
3. Explore how to use existing data from eCommerce platforms like Shopify, Magento, WooCommerce, WordPress, Salesforce, Custom Backend, Sanity, Mock APIs or others.
4. Develop skills to handle and validate schemas, ensuring alignment with data sources.

API Integration Process:

First, create a Sanity CMS project on their platform.Once the project is created, you’ll get access to **Sanity Studio**, where you can define your content types (e.g., products, categories, etc.).

Data Migration Process

Objective: The primary objective of the data migration process was to transfer and adjust the existing data to meet the requirements of the updated schema in the Sanity CMS. This process involved ensuring that the product data aligns with the new fields, such as stock availability, color, and other product attributes, ensuring the data is both consistent and accurate.

Steps Involved:

1. **Schema Adjustments:**
    - Initially, the product schema in the Sanity CMS was reviewed to identify any changes required for aligning with the new design. Some fields, such as "color" and "size," were added to the product schema to accommodate additional product information.
    - The "inStock" field was implemented to indicate product availability, and the image field was connected to Sanity’s asset management system to provide proper URLs.
2. **Data Preparation:**
    - A thorough audit of the existing product data was conducted to identify any missing or outdated information. This audit ensured that all the products were properly aligned with the new fields added to the schema.
    - Fields that were initially empty or incorrectly populated were updated manually or through API calls, ensuring that all data met the new schema requirements.
3. **Migration Script:**
    - A migration script was written to automatically migrate the existing data to match the updated schema. The script used the Sanity client to fetch old product data, adjusted it to the new format, and then pushed the updates back to the Sanity CMS.

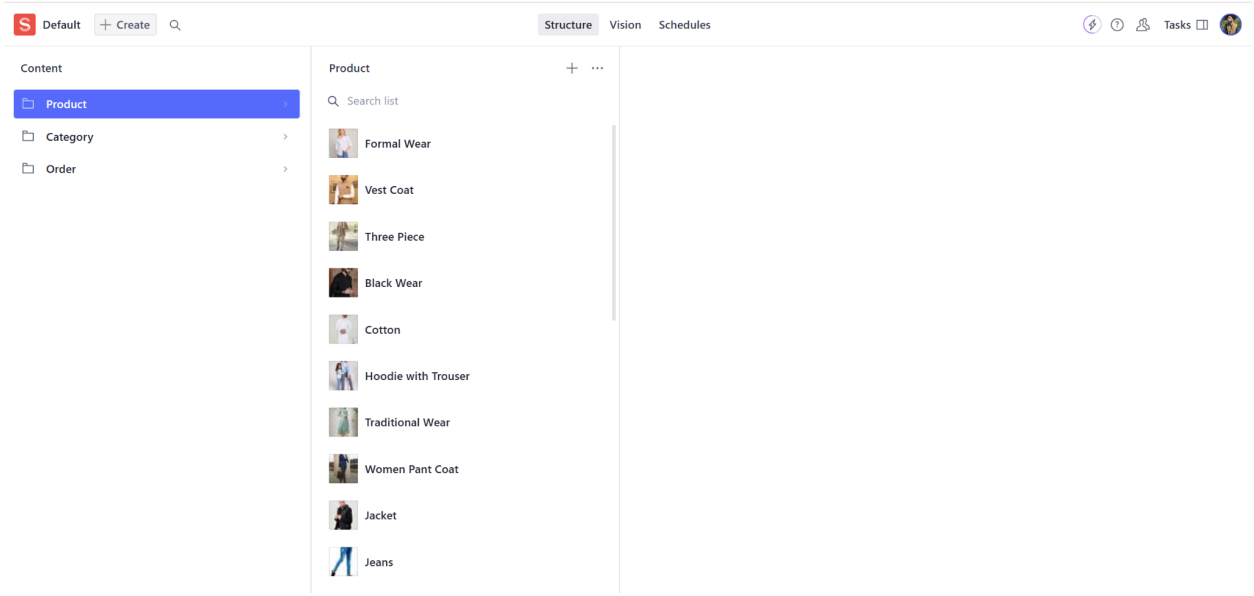
To fetch data from Sanity CMS, you need to install and configure the Sanity client in your Next.js project:

```
import { createClient } from '@sanity/client';

import { apiVersion, dataset, projectId } from '../env';

export const client = createClient({
  projectId,
  dataset,
  token:
  apiVersion,
  useCdn: true,
});
```

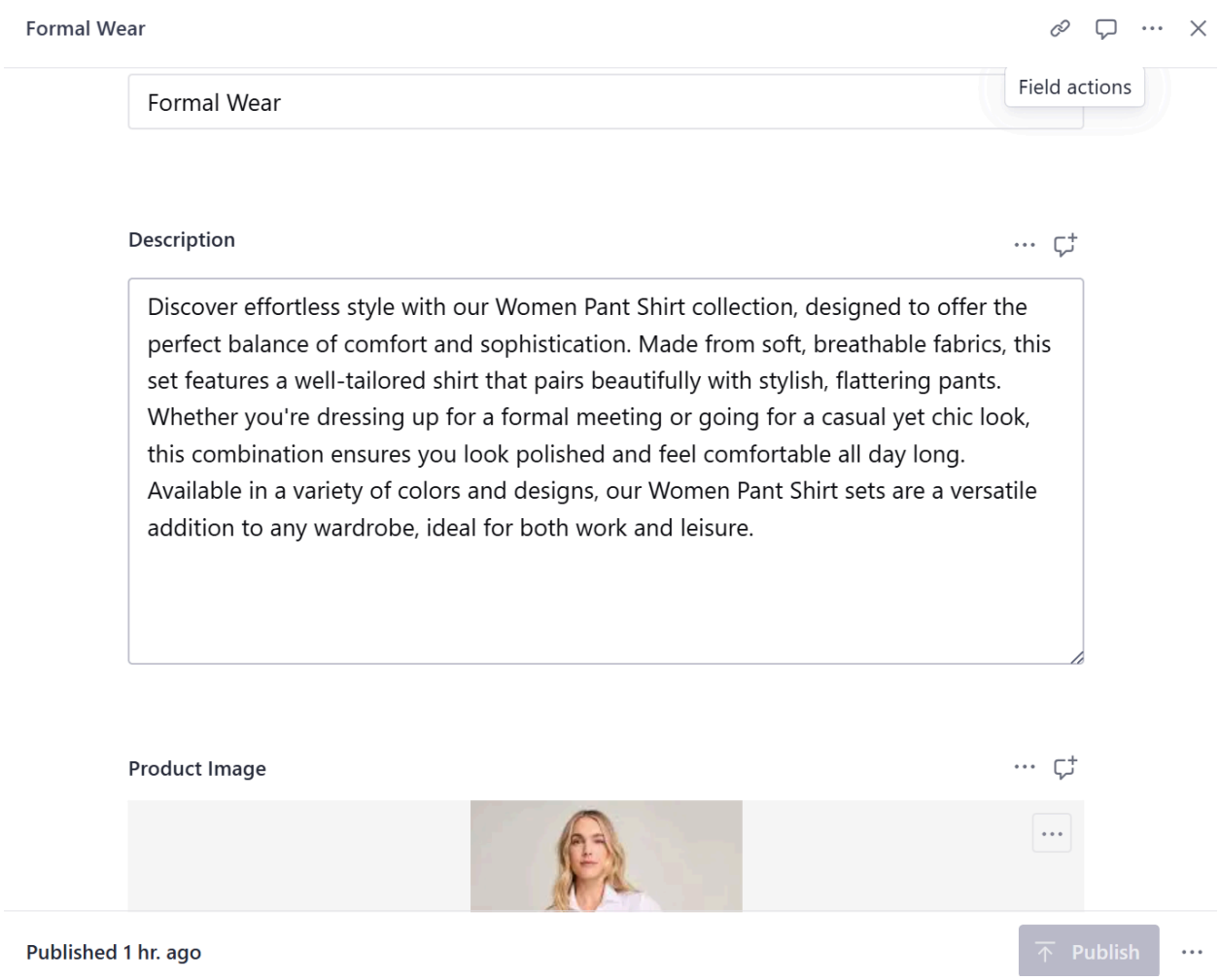
2. **Fetching Data**:
Sanity uses **Groq** (Graph-Relational Object Queries) to query data.

```
import { client as sanityClient } from "../lib/client";

export const fetchProducts = async () => {
  const query = `*[_type == "product"][0...8] {
    _id,
    title,
    description,
    productImage {
      asset -> {
        url
      }
    },
    price,
    tags,
    dicountPercentage,
    isNew,
    stock,
    category->{
      title,
      slug
    }
  }`;
```

Sanity Studio:



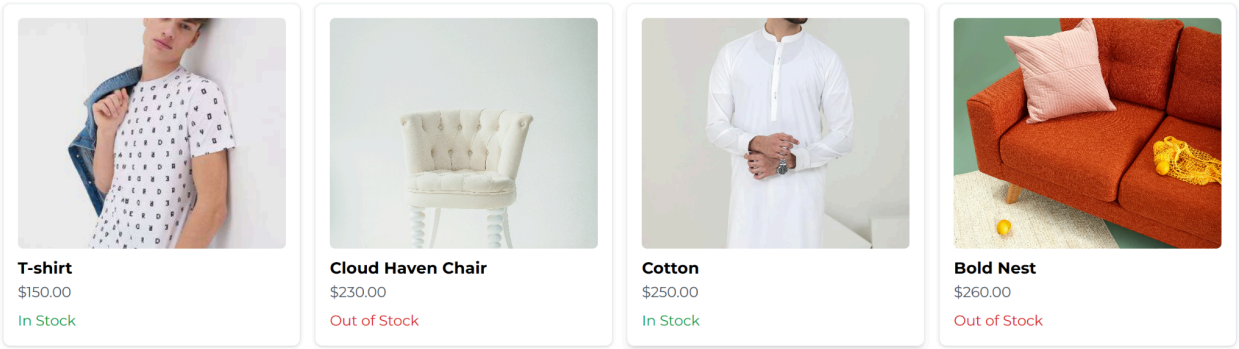
Sanity Product Details:



Sanity Product Schema:

```
export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Title",
      validation: (rule) => rule.required(),
      type: "string",
    },
    {
      name: "description",
      type: "text",
      validation: (rule) => rule.required(),
      title: "Description",
    },
    {
      name: "productImage",
      type: "image",
      validation: (rule) => rule.required(),
      title: "Product Image",
    },
    {
```

UI Fetch from Sanity Schema:



Conclusion:

In this report, we have explored the process of integrating APIs and migrating data into Sanity CMS for a general e-commerce project using Next.js. The key steps involved understanding how to connect external APIs, particularly Sanity CMS, and how to handle and validate data within a structured schema.

The API Integration Process provided an overview of how to configure the Sanity client in the Next.js application and how data from Sanity can be fetched using Groq queries. This process helped ensure that product data, such as pricing, description, availability, and images, were efficiently retrieved and displayed on the frontend.

In the Data Migration Process, we focused on adjusting the existing product schema in Sanity CMS to accommodate new fields, such as color, size, and inStock. The migration script was created to automatically update the existing product data and align it with the new schema. This step was crucial for ensuring that all product information was accurate and consistent with the updated schema requirements.

Overall, this project has provided a comprehensive understanding of how to effectively integrate external APIs, migrate and manipulate data, and structure it within a CMS like Sanity. This process is crucial for building scalable and efficient e-commerce platforms that require accurate and consistent data handling.

Name: Muneeb Jawed

Roll Number: 00355255

Class: Sunday 2 to 5