Day 3 - API Integration Report - MetroMart

Objective

This document outlines the steps and processes followed for API integration and data migration using **Template 06** in a Next.js project with Sanity CMS. It includes:

1. API integration process.
2. Schema validation and adjustments.
3. Data migration steps.
4. Marked positions for screenshots of the migration script, schema, and integration.
5. Best practices followed.

What is API Integration?

API integration refers to the process of connecting different software applications or systems using Application Programming Interfaces (APIs). This allows the systems to exchange data and interact seamlessly. In this project:

- The API provides product data for the marketplace.
- The frontend fetches and displays this data dynamically.
- Sanity CMS stores the data for better management.

API Integration

API Used

- **URL**: https://template6-six.vercel.app/api/products

Integration in Next.js

The API was integrated into the Next.js project to fetch product data and display it on the front end. The steps included:

1. **Creating API Utility Functions**: Functions were added to fetch data from the API and handle errors.
2. **Rendering Data in Components**: Fetched data was passed to a `productPage.tsx` component to display products.
3. **Testing**: Postman and browser developer tools were used to validate API responses and ensure integration worked as expected.

Sanity CMS Schema

Schema Validation and Adjustments

The provided schema for Template 06 was used as the base. Adjustments were made to ensure compatibility with the API data structure. Key changes:

- **Schema Reference**: https://github.com/developer-hammadrehman/template6/blob/main/src/sanity/schemaTypes/product.ts
- **Adjustments Made**:
  - Field `product_title` in API mapped to `title` in Sanity.
  - Field `discountPercentage` added to the schema to match the API data.

```
import { defineType } from "sanity"

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"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"discountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
            title:"New Badge",
        }
    ]
})
```

Data Migration

Migration Script

The `importData.js` script was used to fetch data from the API and upload it to Sanity CMS.

Key Steps:

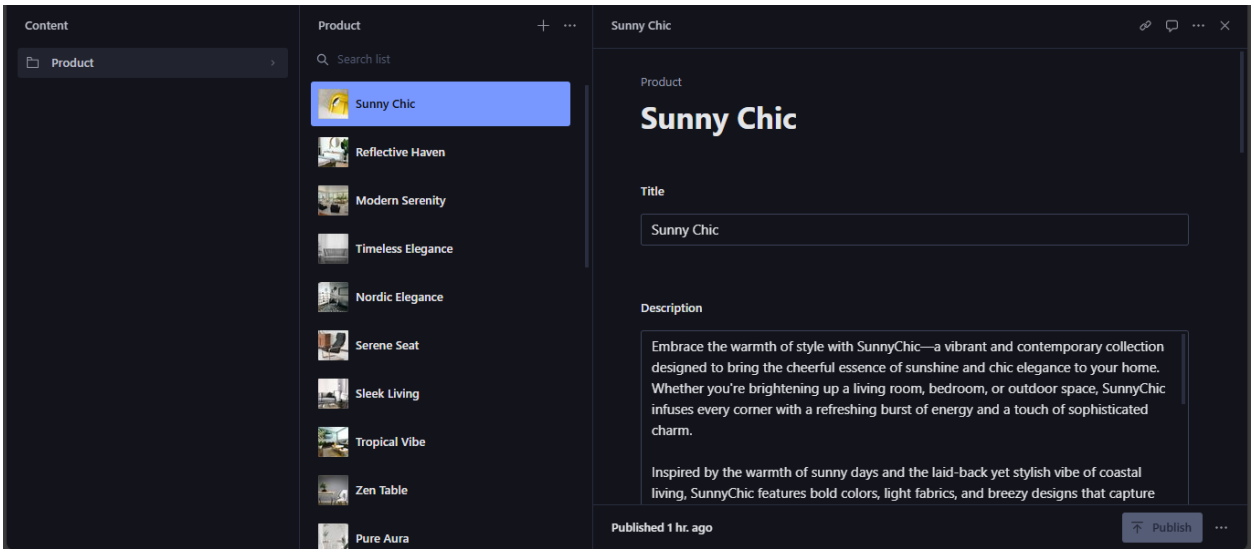
1. **API Call**: The script fetched product data from the API.
2. **Image Upload**: Images were uploaded to Sanity's asset library.

3. **Document Creation**: Each product was created as a Sanity document with the appropriate schema.

Script Reference: https://github.com/developer-hammadrehman/template6/blob/main/importData.js

Result

Products were successfully migrated to Sanity CMS.

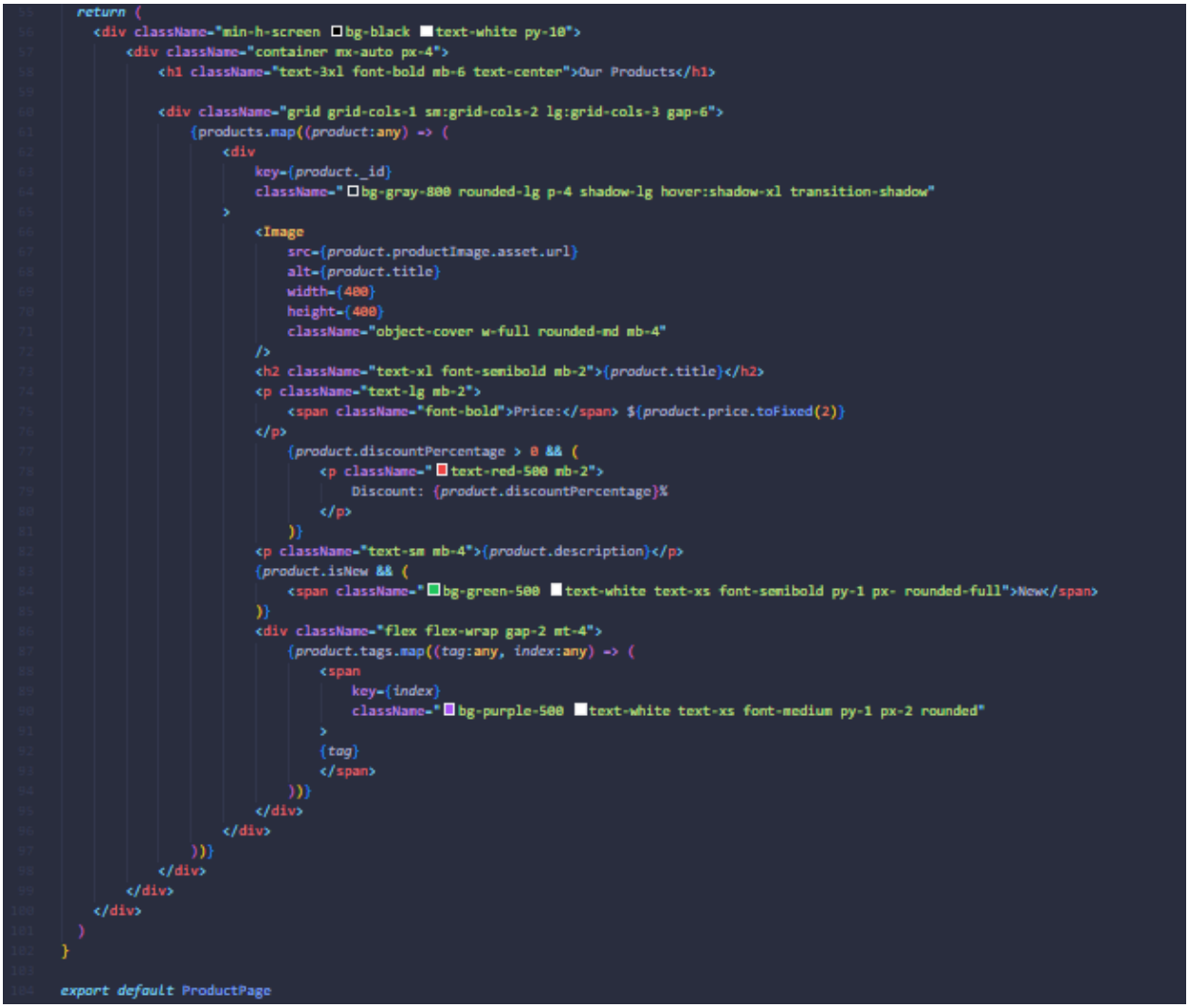


Frontend Display

The `productPage.tsx` component was created to render the product data fetched from Sanity CMS. Tailwind CSS was used for styling.

Code Snippet

```tsx
"use client"

import React from "react";
import { client } from "@/sanity/lib/client";
import { useEffect, useState } from "react";
import Image from "next/image";

type Product= {
    _id: string;
    title: string;
    price: number;
    productImage: {
        assets: {
            url:string;
        };
    };
    tags: string[];
    discountPercentage: string;
    description: string;
    isNew: boolean;
}

const ProductPage = () => {

    const [products, setProducts] = useState<Product[]>([]);

    useEffect(() => {
        const fetchProducts = async () => {
            try {
                const query = `*[_type == "product"] {
                    _id,
                    title,
                    price,
                    productImage {
                        asset -> {
                            url
                        }
                    },
                    tags,
                    discountPercentage,
                    description,
                    isNew,
                }`;

                const result = await client.fetch(query);
                setProducts(result);
            } catch (error) {
                console.error('Cannot fetch Data. Please try again later!', error);
            }
        };

        fetchProducts();
    }, []);
```



```
    return (
      <div className="min-h-screen bg-black text-white py-10">
        <div className="container mx-auto px-4">
          <h1 className="text-3xl font-bold mb-6 text-center">Our Products</h1>

          <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
            {products.map((product:any) => (
              <div
                key={product._id}
                className="bg-gray-800 rounded-lg p-4 shadow-lg hover:shadow-xl transition-shadow"
              >
                <Image
                  src={product.productImage.asset.url}
                  alt={product.title}
                  width={400}
                  height={400}
                  className="object-cover w-full rounded-md mb-4"
                />
                <h2 className="text-xl font-semibold mb-2">{product.title}</h2>
                <p className="text-lg mb-2">
                  <span className="font-bold">Price:</span> ${product.price.toFixed(2)}
                </p>
                  {product.discountPercentage > 0 && (
                    <p className="text-red-500 mb-2">
                      Discount: {product.discountPercentage}%
                    </p>
                  )}
                <p className="text-sm mb-4">{product.description}</p>
                {product.isNew && (
                  <span className="bg-green-500 text-white text-xs font-semibold py-1 px- rounded-full">New</span>
                )}
                <div className="flex flex-wrap gap-2 mt-4">
                  {product.tags.map((tag:any, index:any) => (
                    <span
                      key={index}
                      className="bg-purple-500 text-white text-xs font-medium py-1 px-2 rounded"
                    >
                    {tag}
                    </span>
                  ))}
                </div>
              </div>
            ))}
          </div>
        </div>
      </div>
    )
  }

  export default ProductPage
```

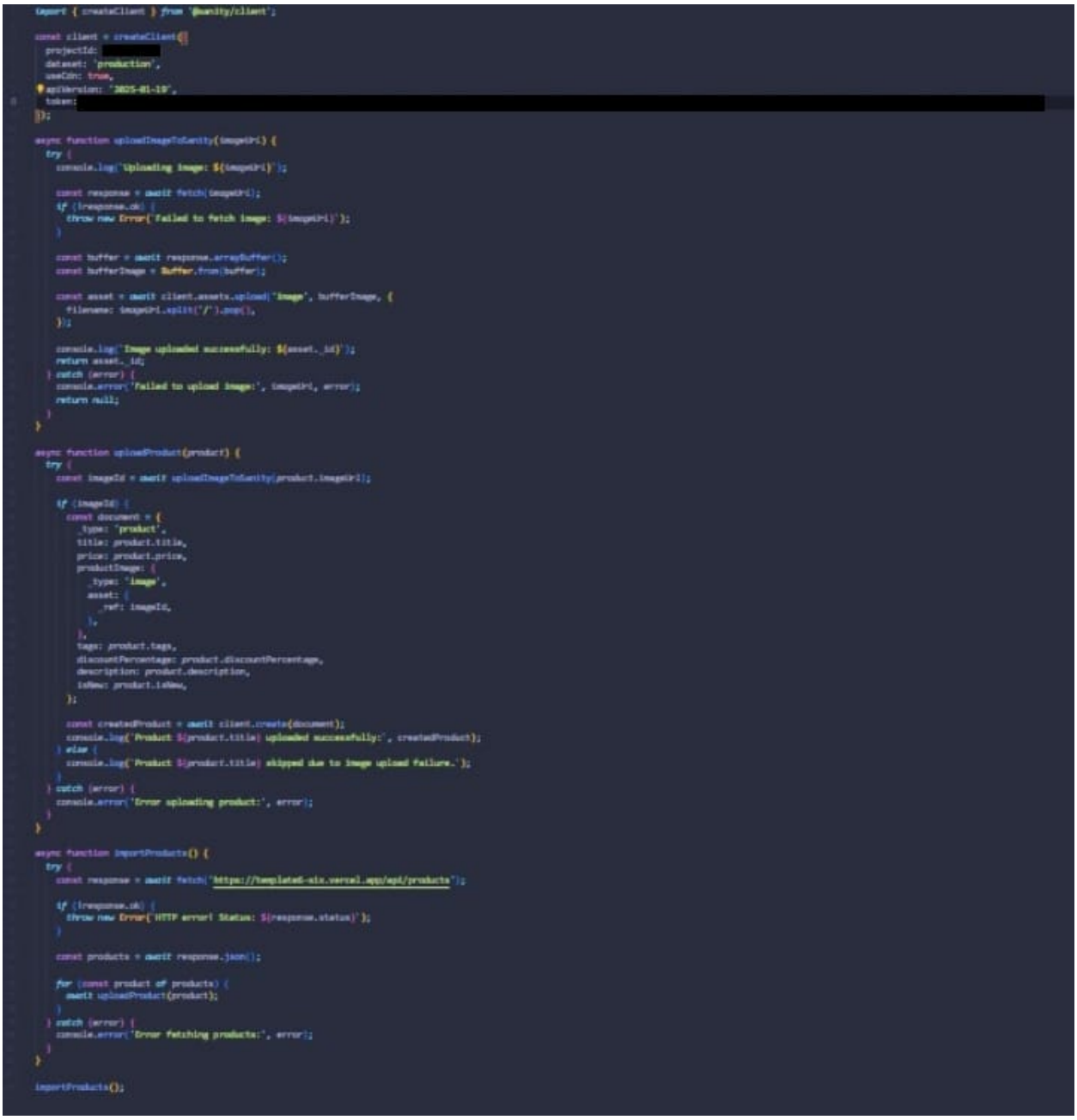
Best Practices Followed

1. **Environment Variables**:
    - API tokens stored securely in `.env` files.
2. **Error Handling**:
    - Logging and fallback mechanisms in the migration script.
3. **Modular Code**:
    - Utility functions for reusability.
4. **Validation**:

- Ensured data alignment with the schema before migration.

Screenshots

1. **Migration Script (`importData.js`)**:

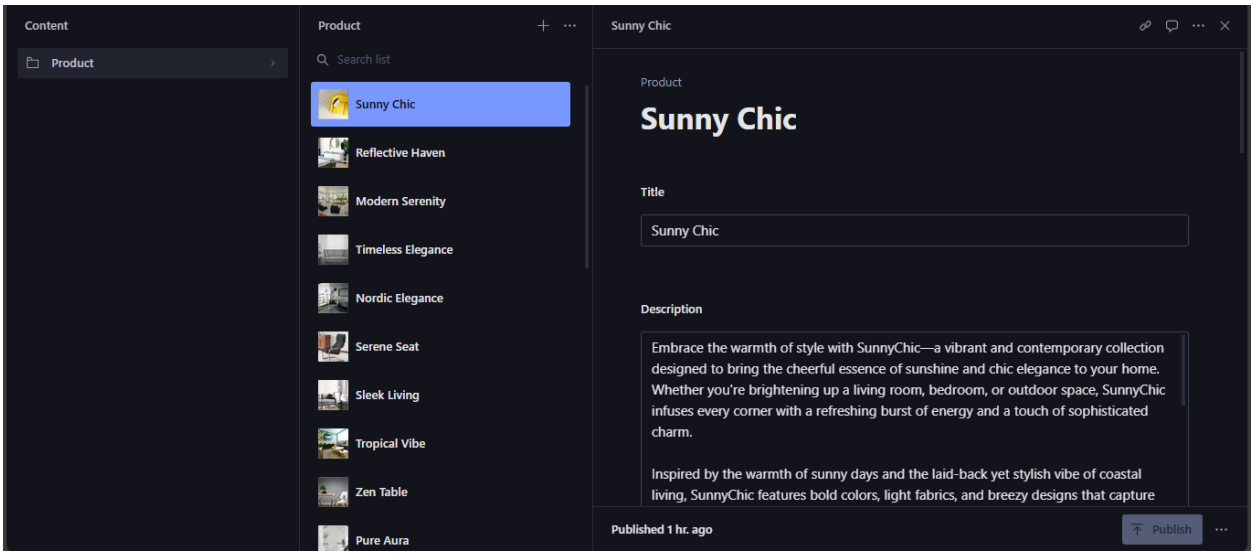


```
import { createClient } from '@sanity/client';

const client = createClient({
  projectId: [redacted]
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-19',
  token: [redacted]
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      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 uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        discountPercentage: product.discountPercentage,
        description: product.description,
        isNew: product.isNew,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.title} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.title} skipped due to image upload failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template6-six.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

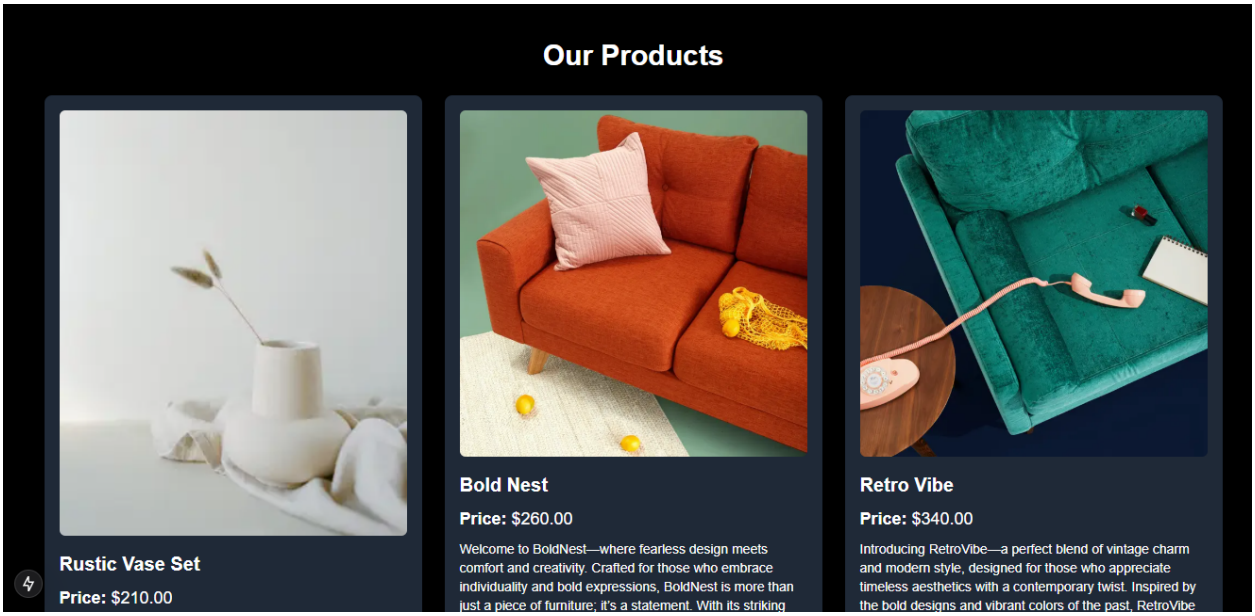
2. **Sanity Schema (`product.ts`):**

```ts
import { defineType } from "sanity"

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"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"discountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
            title:"New Badge",
        }
    ]
})
```

3. **Populated Sanity CMS**:



4. **Frontend Display**:



Submission Requirements

- API integration process.
- Adjustments made to schemas.
- Migration steps and scripts.

- Marked positions for screenshots of:
  - Migration scripts.
  - Populated CMS.
  - Frontend display.

Prepared by Abdul Rehman Rajpoot