Made by: Muhammad Khalid Roll no: 172526

Day 3 - API Integration and Data Migration Report - [Furniro-For furniture E-Commerce Template-6]

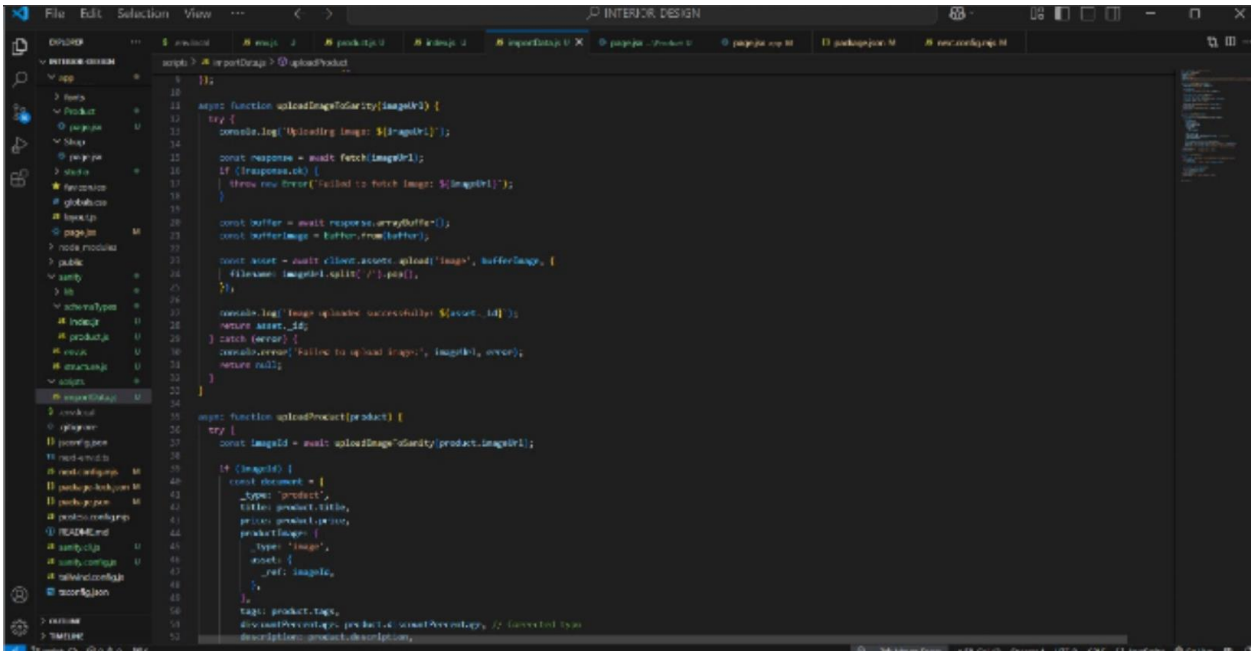
Executive Summary

This report encapsulates the meticulous process undertaken to seamlessly integrate APIs and migrate data into the Sanity CMS while building a fully functional Next.js frontend. The primary goal was to align schemas, ensure data accuracy, and deliver a robust and scalable frontend for displaying dynamic content.

Process Overview

1. Understanding the API Landscape

- **Comprehensive API Documentation Review:**
  - Key endpoints identified:
    - `/products`: Fetching product listings.
    - `/categories`: Retrieving product categories.
    - `/orders`: Accessing order history.
  - **Tools Utilized:** Postman and browser developer tools to rigorously test endpoints and verify response accuracy.



2. Schema Validation and Refinements

- **Alignment of Schema with API Structure:**
  - Example Refinement:
    - API Field: `product_title`
    - Schema Field: `name`
    - **Action Taken:** Adapted the schema to ensure precise field mapping and compatibility.
- Established relationships between categories and products to reflect structured and interconnected data.

3. Data Migration Strategies

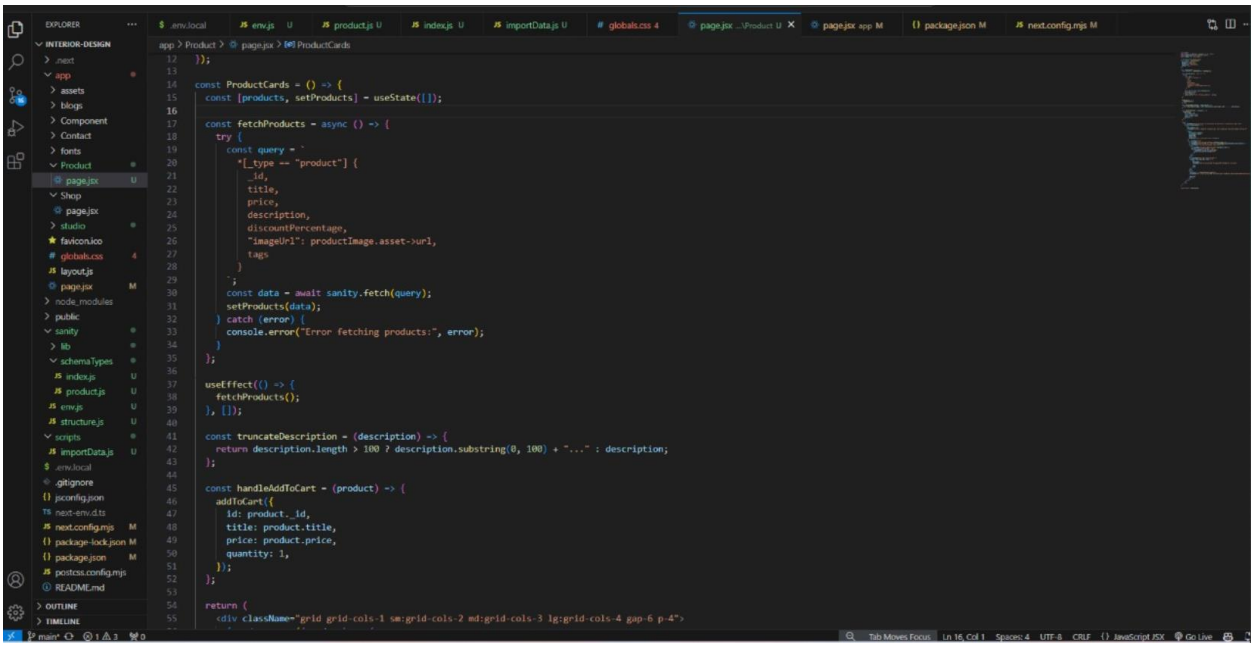
Approaches Implemented:

1. **Direct API Utilization:**
   a. Authored scripts to fetch, transform, and import API data directly into Sanity CMS.
2. **Manual Import:**
   a. Exported data as JSON/CSV files and imported them using Sanity's import tools—an ideal approach for smaller datasets.
3. **External Platform Integration:**

a. Leveraged Shopify API to extract data, restructured fields, and performed a seamless migration into the CMS.

Quality Assurance:

- Performed CMS backups prior to migration.
- Validated data consistency and alignment with the schema.
- Documented each migration step for traceability.



4. API Integration in Next.js

Key Implementation Steps:

1. **Development of Reusable Utility Functions:**
    a. Example:
```
export async function fetchProducts() {
  const response = await fetch("https://api.example.com/products");
  if (!response.ok) throw new Error("Failed to fetch products");
  return await response.json();
}
```

2. **Dynamic Data Rendering:**

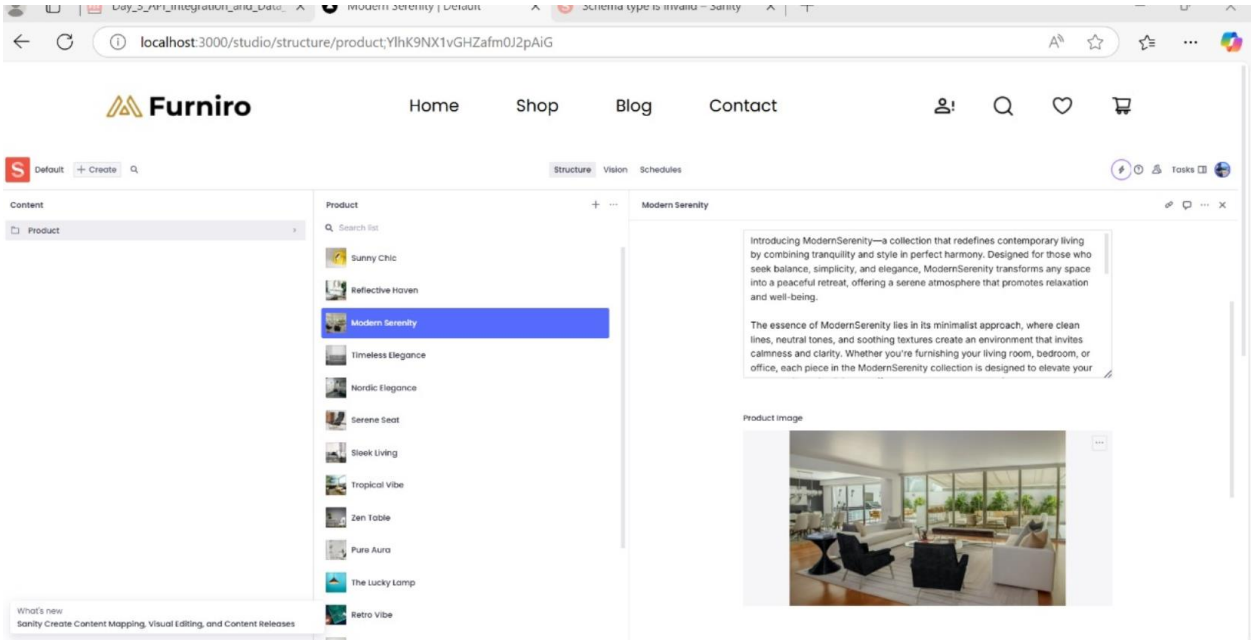
a. Incorporated components to display data fetched from APIs: `function`

```
ProductList({ products }) {
  return (
    <div>
      {products.map((product) => (
        <div key={product.id}>{product.name}</div>
      ))}
    </div>
  );
}
```

3. **Rigorous Testing and Debugging:**
    a. Verified endpoints with Postman.
    b. Logged API responses to identify and resolve discrepancies effectively.

Error Handling:

- Centralized error logging for streamlined debugging.
- User-friendly UI messages to enhance the user experience.
- Implemented skeleton loaders to manage data loading gracefully.



Achievements

1. **Sanity CMS:** Successfully populated with data from the API, external sources, and manual uploads, with relationships between entities accurately established.
2. **Next.js Frontend:** Fully integrated API-driven functionality showcasing product listings and categories, supported by fallback mechanisms for enhanced reliability.

Best Practices Observed

- **Secure Data Management:** Safeguarded sensitive credentials in `.env` files.
- **Clean Coding Standards:**
  - Modularized functions for reusability.
  - Used meaningful variable names and added explanatory comments for clarity.
- **Data Validation:** Verified data integrity during migration and documented discrepancies for resolution.
- **Version Control:**
  - Committed frequently with descriptive messages.
  - Tagged major milestones to facilitate collaboration and tracking.
- **Comprehensive Testing:** Addressed edge cases, validated responses, and ensured smooth performance.

Submission Checklist

1. **Documentation:**
   a. Detailed steps for API integration and schema refinements.
   b. Comprehensive migration strategies.
2. **Visual Evidence:**
   a. Screenshots of API responses.
   b. Sanity CMS populated fields.
   c. Data displayed dynamically in the frontend.
3. **Code Artifacts:**
   a. Scripts showcasing data migration processes.
   b. Utility functions and components for API integration.

Conclusion

By adhering to a structured approach and best practices, I successfully delivered a scalable and robust API-integrated system. This initiative underscores the value of precision, validation, and thorough testing in ensuring the success of modern web development projects.