SHOP.CO – Farwa Batool

API Integration and Data Migration Report - SHOP.CO

I. API Integration Process

Objective:

This report details the process of integrating a custom MockAPI and migrating data into Sanity CMS for SHOP.CO, ensuring seamless data handling and compatibility with the marketplace's frontend and backend systems.

1. APIs Used:
    - MockAPI for Initial Data: Endpoint: https://677fa0d60476123f76a7500c.mockapi.io/product Functionality: Provided initial product and category data for migration.

Steps Taken

2. Data Creation in MockAPI:

- MockAPI was used to create a dataset for products and categories.
- Example products fields: name, description, price, original price, images, colors, sizes, categories, tags

3. Manual Data Migration to Sanity:

- Data from MockAPI was manually exported and imported into Sanity CMS.
- Used Sanity Studio for field mapping and validation.

4. Testing API Integration:

- Verified data accuracy after migration using Sanity Studio and frontend rendering.

5. Frontend Integration:

- Rendered Sanity CMS data in com

6. Schema Adjustments in Sanity CMS

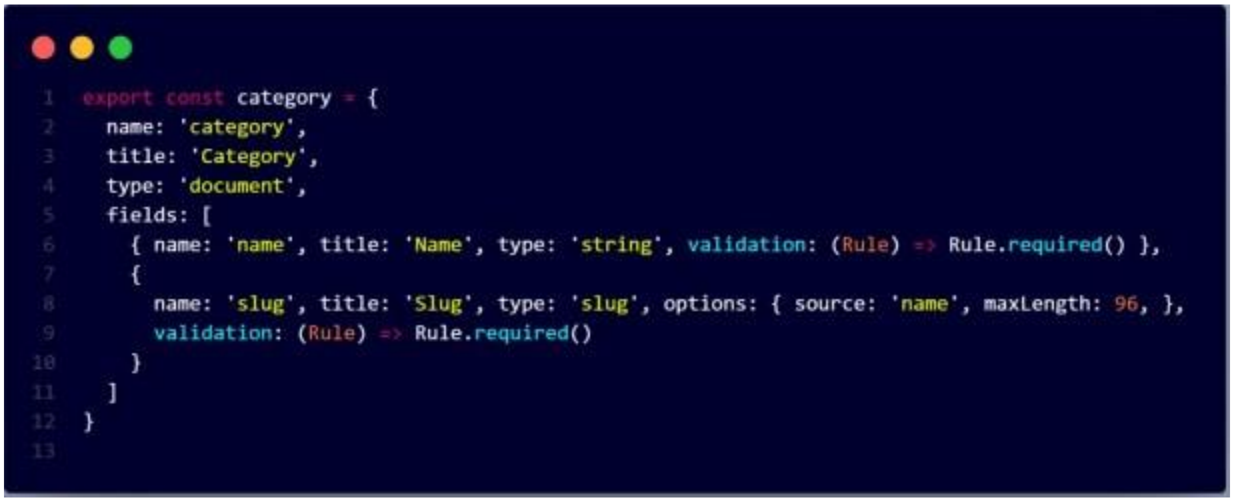
The following schema adjustments were made to align with MockAPI data:

• Products Schema:

Added fields: inventory, categories, images, tags etc.
Adjusted field names to match API, e.g., title to name.

• Categories Schema:

- Added a description field for better categorization.
- ponents like product cards and category filters.



```
export const category = {
  name: 'category',
  title: 'Category',
  type: 'document',
  fields: [
    { name: 'name', title: 'Name', type: 'string', validation: (Rule) => Rule.required() },
    {
      name: 'slug', title: 'Slug', type: 'slug', options: { source: 'name', maxLength: 96, },
      validation: (Rule) => Rule.required()
    }
  ]
}
```

II. Data Migration

Methodology

1. Manual Data Migration:

- Data from MockAPI was exported as JSON.
- Imported into Sanity CMS using the built-in Studio interface.

2. Data Validation:

- Ensured all imported fields matched the schema requirements in Sanity CMS.

III. Challenges and Resolutions

• **Field Mismatches:** Resolved by mapping MockAPI fields to Sanity schema

fields.

• **Duplicate Entries:** Validated unique fields (e.g., product_id) to prevent

duplicates.

Frontend Display

IV. Components Updated

1. Product Listing Page:

- Dynamically displayed products with stock status and category filters.

2. Category Filters:

- Integrated category data into dropdown menus for filtering.

3. Product Details Page:

- Showcased detailed product information including reviews and inventory.

Error Handling

1. Fallback Mechanisms:

- Displayed skeleton loaders and error messages in case of API failure:
  if (error) return <div>Failed to load products. Try again later. </div>;

2. Validation Checks:

- Ensured all data from MockAPI met schema requirements before inserting into Sanity CMS.

Conclusion

Day 3's tasks were successfully completed using MockAPI for initial data creation and Sanity CMS for robust backend handling. The marketplace is now equipped with a functional backend and ready for advanced features.

Dynamic Public Marketplace – SHOP.CO

By Farwa Batool