Hunar Bazaar: API Integration and Data Migration Report

1.0 Objective:

This document details the integration and data migration process for the Hunar Bazaar project. The data is entered manually into Sanity CMS, and queries are defined in a separate file for seamless retrieval and usage in the Next.js frontend. Additionally, the fetching methods demonstrate how data is accessed programmatically.

1. Manual Data Entry in Sanity CMS

1.1 Preparation:

To begin, I logged in to the Sanity CMS dashboard and ensured that the required schemas were deployed. This involved:

- Navigating to the "Content Studio" section.
- Checking the schema definitions for categories and products, sale-banner, orders.
- Verifying that all fields required by the frontend application were included.

1.2 Entering Data /Product, Sale-banner:

I manually entered the data of products, Categories, and sale banners, and for example, each product included fields like:

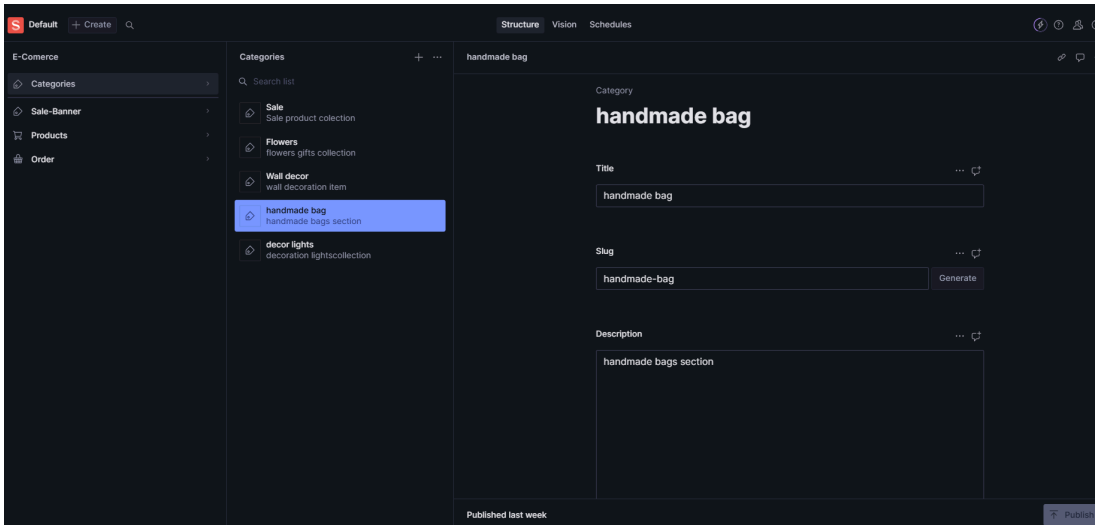
1. **Name:** "Embroidered Bag".
2. **Slug**: "embroidered-bag".
3. **Description:** "A finely crafted shawl with intricate embroidery."
4. **Price:** 35.00 ($).
5. **Discount Percent**: 30(%).
6. **Image:** Uploaded a high-resolution image make sure kbs, SVG format.
7. **Category::** Linked to an existing category (e.g., "handmade bags").
8. **Stock**: 30.
9. **Status**: 'NEW, HOT, SALE'

1.3 Schema Overview:

1

2

1.4 manually Data Entry:



2. Query Definition for Data Retrieval:

To fetch the manually entered data, I created queries using next-sanity. These queries were defined in a separate file for better modularity and reuse.

```
sanity > helpers > queries.ts > ...
You, 4 days ago | 2 authors (You and one other)
import { defineQuery } from "next-sanity";

Execute Query
export const SALE_QUERY =
  defineQuery(`*[_type == 'sale' && active == true] | order(name asc)`);

export const PRODUCTS_QUERY = defineQuery(
  `*[_type == 'products' ] | order(name asc)`
);

export const CATEGORIES_QUERY = defineQuery(
  `*[_type == 'category'] | order(name asc)`
);
      khadija-faisal, 5 days ago • my marketplace
export const PRODUCT_BY_SLUG = defineQuery(
  `*[_type == 'products' && slug.current == $slug] | order(name asc)[0]`
);

Execute Query
export const PRODUCT_SEARCH_QUERY = defineQuery(`*[_type == "products" && (
  lower(product) match lower($searchParam + "*") ||
  lower(description) match lower($searchParam + "*")
)]`);

export const PRODUCT_BY_CATEGORY = defineQuery(
  `*[_type == "products" && references(*[_type == "category" && slug.current == $CategorySlug]._id)] | order(product asc)`
);
```

3. Fetching Data from Sanity CMS:

Once the queries were set up, I implemented functions to fetch data programmatically. This ensured smooth integration between the backend and front-end. Below are examples of how the queries were used:

1. **Error Handling:** In case of a failure (e.g., network error, no response from Sanity), the try-catch block ensures the error is logged, and the application doesn't crash. It returns an empty array so that the front-end can handle it gracefully.
2. **Data Retrieval:** Each function uses the respective query to fetch the relevant data (sale banners, products, or categories). The fetched data is returned to be used in the front-end.
3. **Consistency:** Each function follows the same structure, which makes it easy to maintain and understand.

Author: Khadija
Date: 23-January 2025

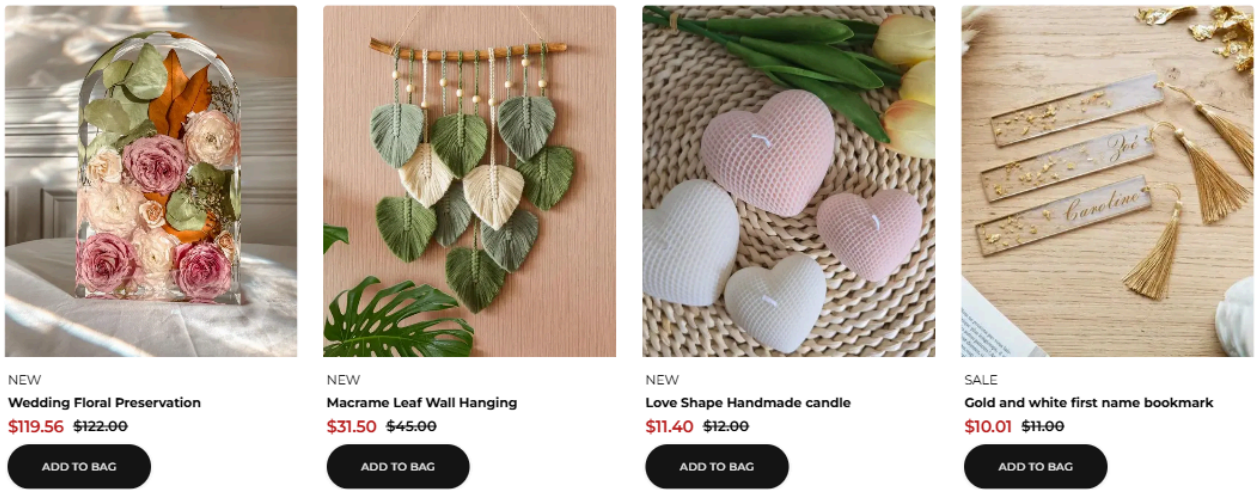
3.1 Fetching Functions (sanity to front-end):

```ts
export const getallCategories = async () => {
  try {
    const categories = await sanityFetch({
      query: CATEGORIES_QUERY,
    });
    return categories?.data || [];
  } catch (error) {
    console.error("Error to fetch categories Data", error);
    return [];
  }
};

Complexity is 6 It's time to do something...
export const getProductbySlug = async (slug: string) => {
  try {
    const productbySlug = await sanityFetch({
      query: PRODUCT_BY_SLUG,
      params: { slug },
    });
    return productbySlug?.data || null;
  } catch (error) {
    console.error("Error to fetch Product by Slug", error);
    return null;
  }
};
Complexity is 6 It's time to do something...
export const getProductbyName = async (searchParam: string) => {
  try {
    console.log("Searching with parameter:", searchParam);
    const productbyName = await sanityFetch({
      query: PRODUCT_SEARCH_QUERY,
      params: {
        searchParam: searchParam.toLowerCase(),
      },
    });
    console.log("Search results:", productbyName);
    return productbyName?.data || [];
  } catch (error) {
    console.error("Error to fetch Product by Name", error);
    return [];
  }
```

4. Displaying Data on the Frontend ：

Once the data was fetched from Sanity CMS, it was displayed on the frontend using React components in Next.js.



Author: Khadija
Date: 23-January 2025