Day 3 - API Integration Report - Comforty Marketplace

Prepared by: Ayesha Farooq
Date: [18-01-25]

---

1. Introduction

This report outlines the activities completed on Day 3 of the hackathon for Team Comforty. The main focus was on migrating product data to the Comforty Marketplace using Sanity CMS. The integration process, schema adjustments, and migration steps are detailed below.

---

2. API Integration Process

The API integration process aimed to ensure seamless data migration and interaction between the Sanity CMS backend and the marketplace frontend. Below are the main steps:

Sanity API Configuration

To initiate the migration process, the following steps were followed:

- **Product ID** and **API Token** were used to connect to the Sanity CMS API.
- These credentials were securely stored as environment variables to ensure the safety of sensitive data.

Schema Adjustments

Two new files were added in the Sanity CMS schema folder (under `sanity/schemas`):

1. **product.ts** - For storing product data, such as title, description, price, and image.
2. **categories.ts** - For categorizing products and linking them within the CMS.

These files ensure that the data is structured and easy to query.

---

3. Data Migration Process

The migration process involved creating a script to transfer product data from Sanity CMS to the local database. Here's how the migration was executed:

Creating the Migration Script

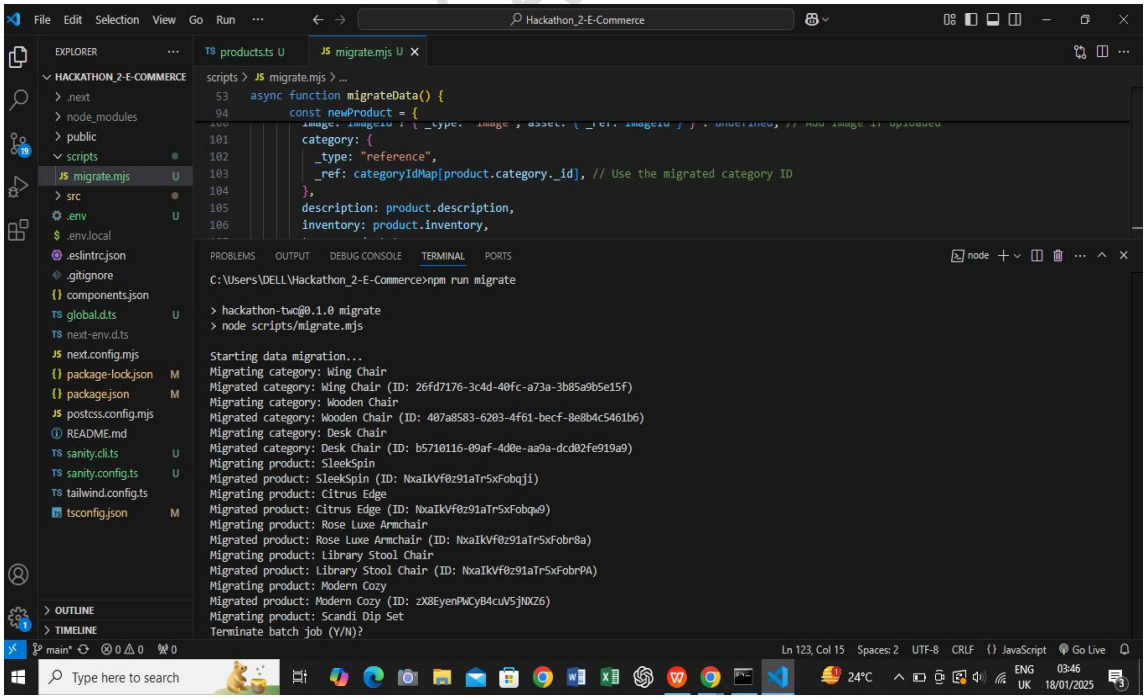
- A new folder named `scripts` was created in the root directory.
- A custom script called **migrate.mjs** was written to fetch data from Sanity CMS and migrate it into the local database.

Executing the Migration

To trigger the migration, the following command was run in the terminal:

```
npm run migrate
```

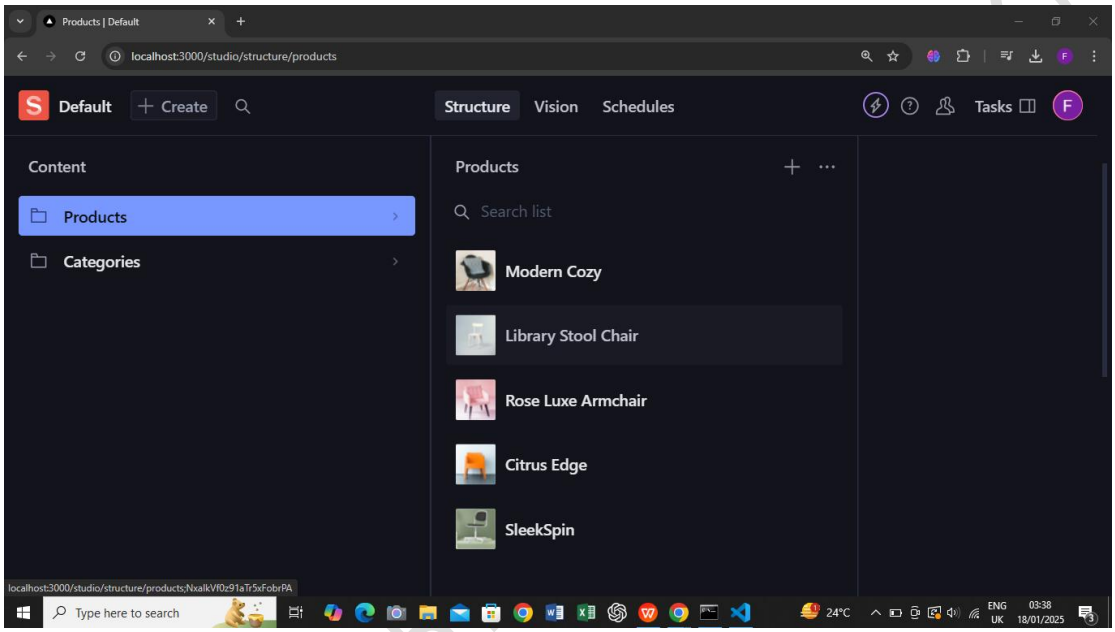
This command initiated the migration process and successfully transferred the data from Sanity CMS into the local database.



Verification

To verify the migration, the following steps were performed:

- The project was run locally.
- The `/studio/structure` endpoint was accessed on localhost to check if the data was successfully migrated.
- The same **Sanity account** used for the CMS was logged in to ensure consistency and correctness of the data.



4. Schema Adjustments

The data structure was organized using two schemas:

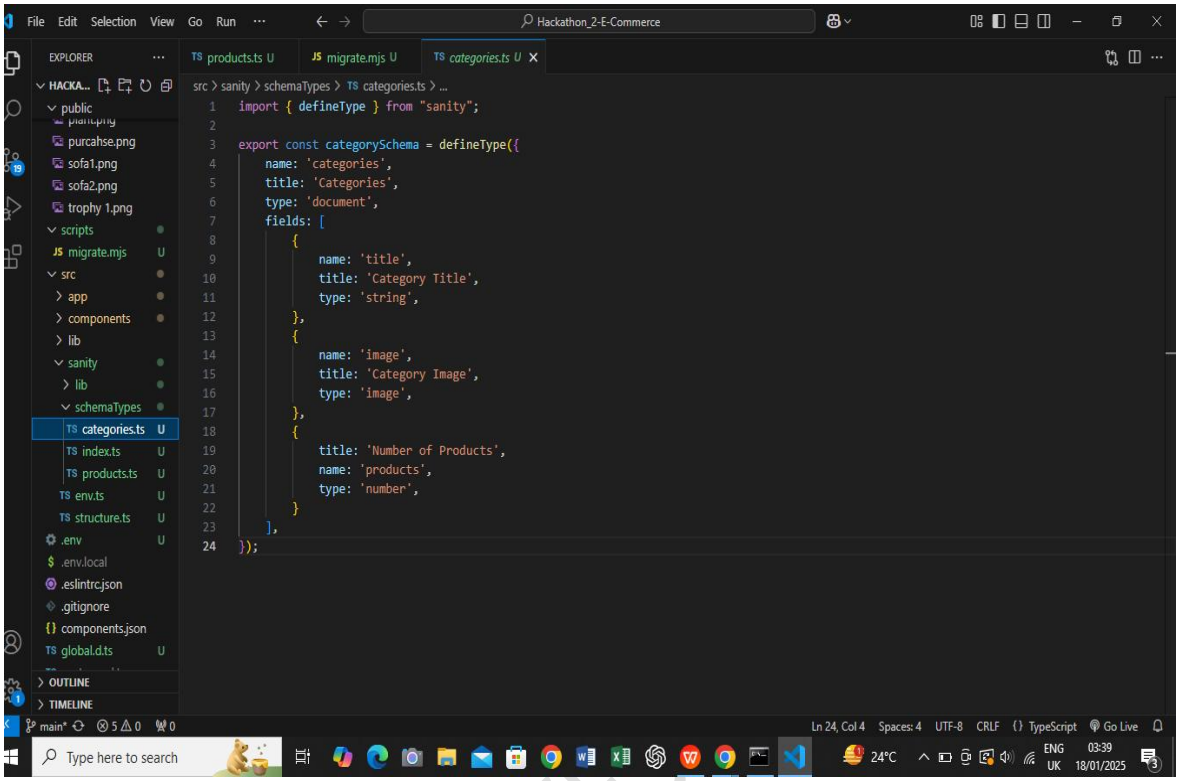
Product Schema (product.ts)

```typescript
import { defineType } from "sanity";

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Product Title",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      type: "number",
    },
    {
      name: "badge",
      title: "Badge",
      type: "string",
    },
    {
      name: "image",
      title: "Product Image",
      type: "image",
    },
    {
      name: "category",
      title: "Category",
      type: "reference",
      to: [{ type: "categories" }],
    },
    {
      name: "description",
      title: "Product Description",
      type: "text",
    },
    {
      name: "inventory",
      title: "Inventory Management",
      type: "number",
    },
    {
      name: "tags",
      title: "Tags",
      type: "array",
      of: [{ type: "string" }],
      options: {
        list: [
          { title: "Featured", value: "featured" },
          {
            title: "Follow products and discounts on Instagram",
            value: "instagram",
          },
          { title: "Gallery", value: "gallery" },
        ],
      },
    },
  ],
});
```

- **Fields**:
  - `Title`: The name of the product.
  - `Description`: Detailed product description.
  - `Price`: The cost of the product.
  - `Image`: Product image(s).

Categories Schema (categories.ts)

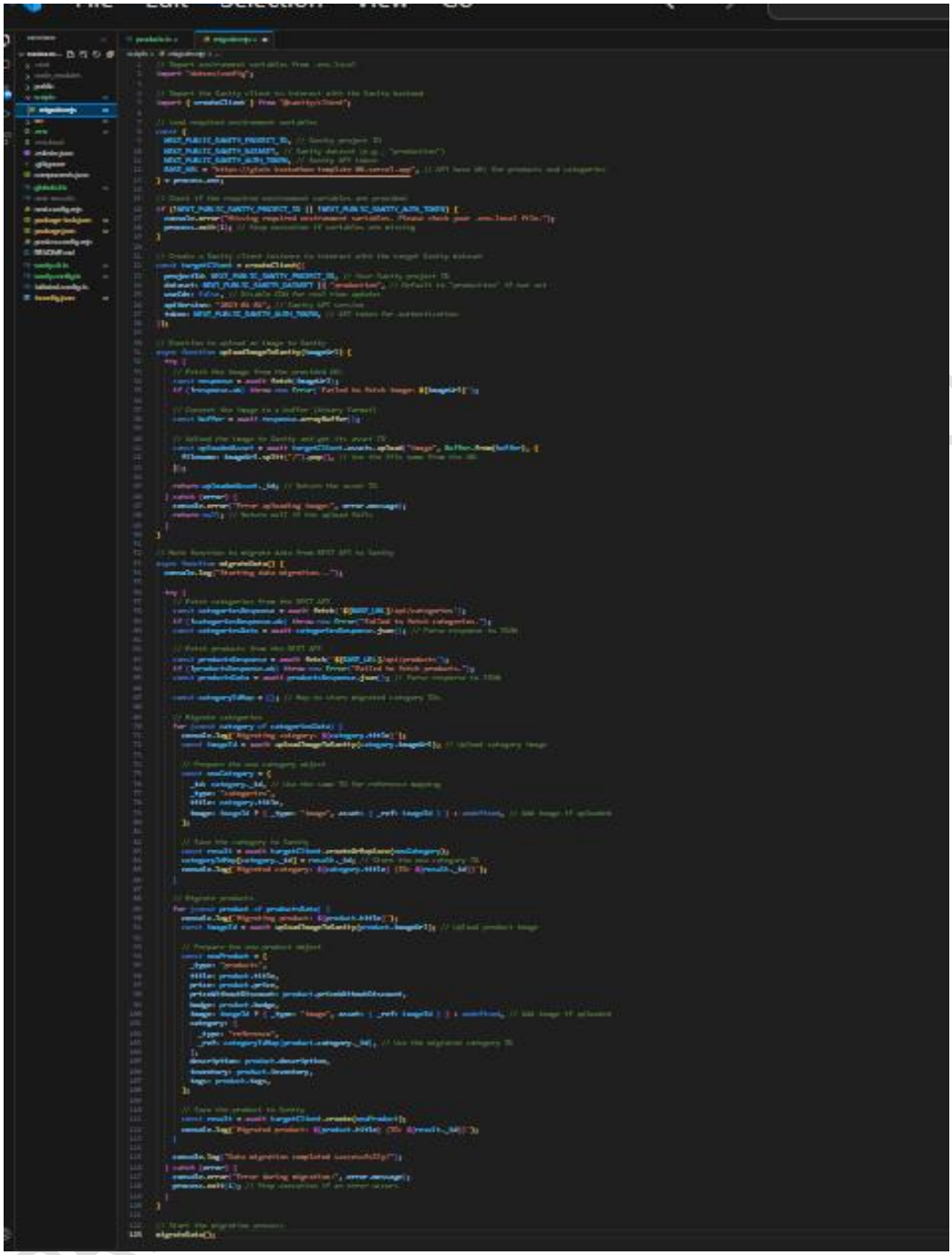


- **Fields**:
  - `Name`: The name of the category.
  - `Products`: A reference to products belonging to this category.

---

5. Migration Script

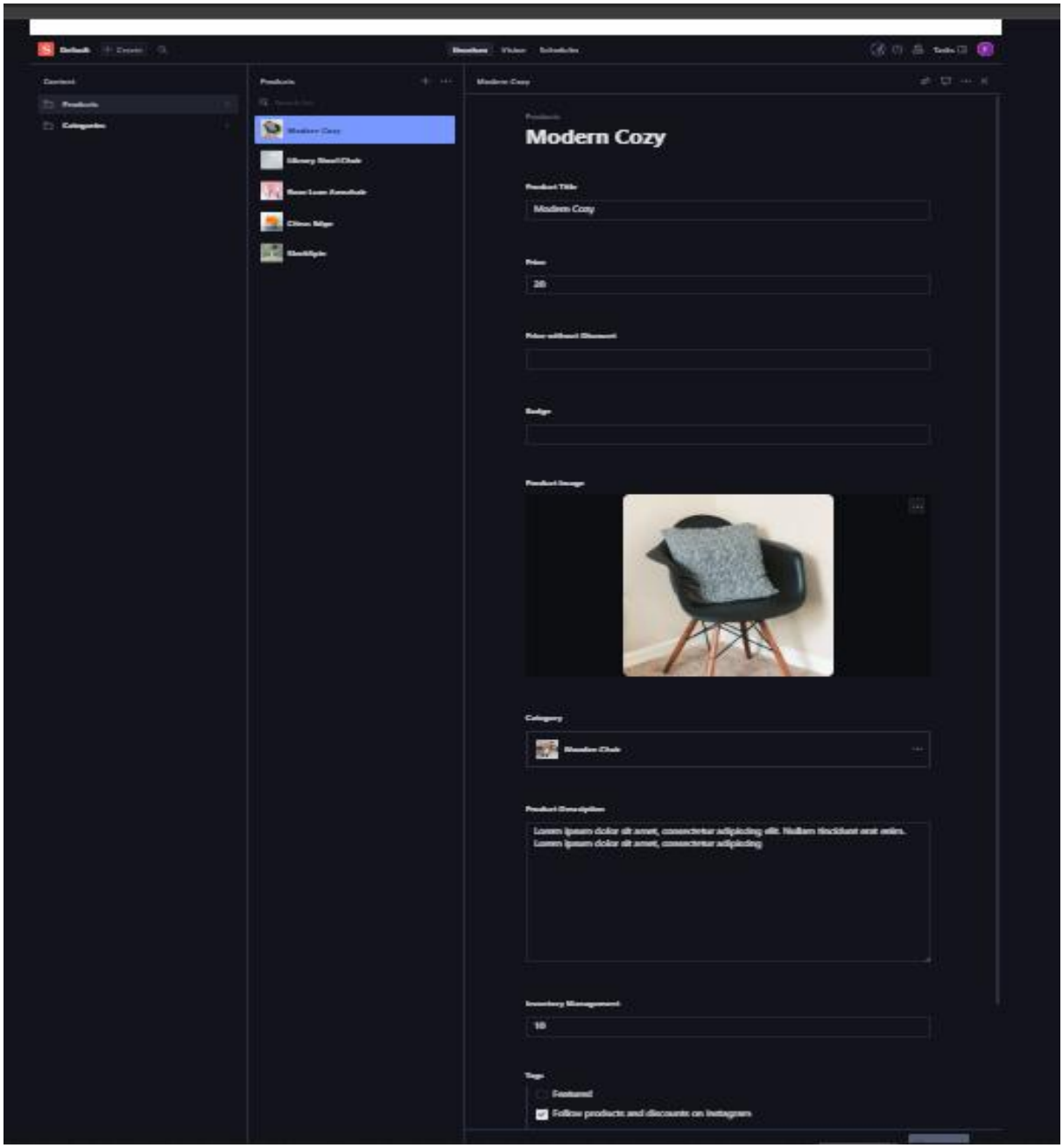
The migration script (`migrate.mjs`) was created to automate the transfer of data from Sanity CMS to the local database.



This script fetches data from an array and uses the `client.create()` method to insert it into Sanity CMS.

6. API Integration Screenshots

Sanity Product Schema:



Title Field

- **Type:** String
- **Purpose:** Stores the name of the product (e.g., "Comfortable Chair").
- **Validation:** Ensures this field is not left empty, as it's a required field for identifying the product.

Image Field

- **Type:** Image
- **Purpose:** Stores a single image of the product.
- **Options:**
  - Supports high-resolution images.
  - Allows alternative text for accessibility purposes.

Description Field

- **Type:** Text
- **Purpose:** Provides a detailed description of the product, explaining its features, dimensions, and other details.

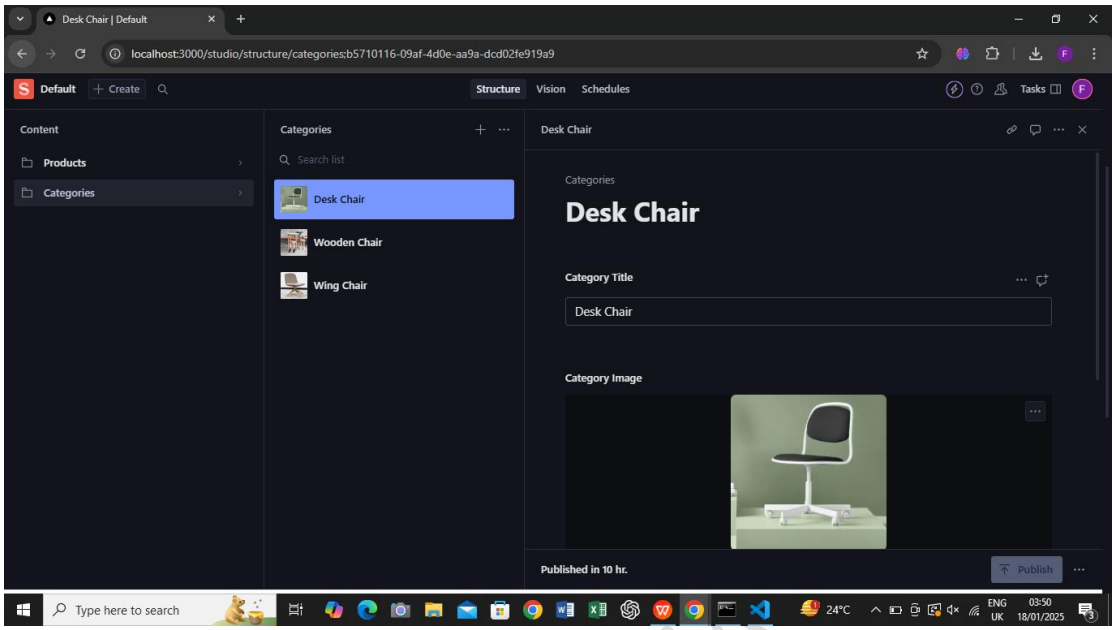
Price Field

- **Type:** Number
- **Purpose:** Represents the cost of the product.
- **Validation:** Ensures the price is a positive value (e.g., `greaterThan(0)`).

Category Field

- **Type:** Reference or String
- **Purpose:** Associates the product with a category (e.g., "Sofas", "Sectionals").
- **Benefits:** Helps in filtering and organizing products based on their categories.

Sanity Categories Schema:



- **Fields:**
  - **Name:** The name of the category (e.g., "Sofas").
  - **Products:** A reference to products belonging to this category.

---

7. Conclusion

On Day 3 of the hackathon, significant progress was made in building the backend of the Comforty Marketplace:

- The **data migration process** was successfully executed, allowing product and category data to be transferred from Sanity CMS into the local database.
- **Schemas** for product and category were created and implemented.
- The **API integration** with the frontend was successful, enabling the marketplace to fetch and display the data.

This sets a solid foundation for the next stages of development, where the focus will shift to the frontend and feature enhancements.