API Integration and Migration Report – Luxury Home Decore

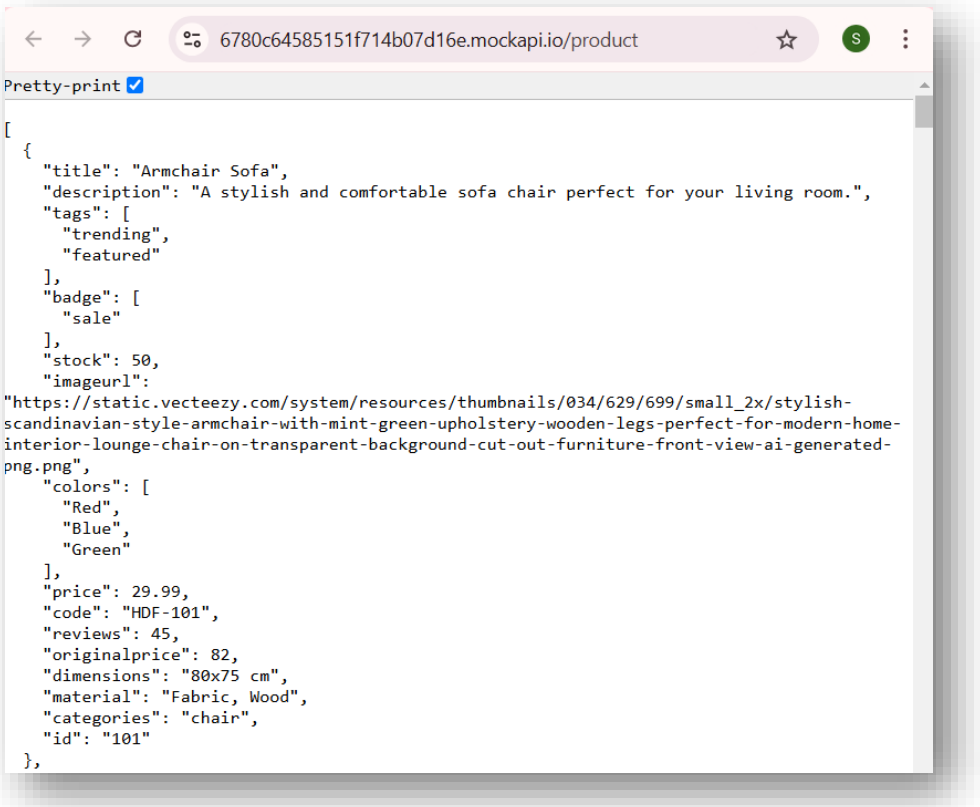
Overview: Our e-commerce platform aims to provide luxury home decor items, including vases, luxury chairs, fancy mirrors, and lamps. During the recent hackathon, I successfully implemented an API integration and data migration process that involved creating my own API, migrating data from this API to Sanity CMS, and finally fetching data from Sanity to display on the website. This document outlines the technical details, challenges faced, and the solutions implemented during the project.

1. Understanding the Custom API

Objective: Review and analyze the custom API designed for the project to identify key endpoints and data structures.

Steps Taken:

- Developed and documented the custom API of 50 products.
- Identified key endpoints such as:
    - **Product listings** (e.g., /products)
    - **Detailed product information** (e.g., /products/:id)



2. Validate and Adjust Your Schema

Objective: Compare the existing Sanity CMS schema with the custom API data structure and make necessary adjustments.

Steps Taken:

- Compared field names, types, and relationships to ensure compatibility.
- Adjusted schema fields for seamless integration.

```
src > sanity > schemaTypes > TS product.ts > default
import { defineType, defineField } from 'sanity';

export default defineType({
  name: 'product',
  title: 'Product', // Display title in Sanity Studio
  type: 'document', // Declares this as a document type
  fields: [
    defineField({
      name: 'title',
      title: 'Title',
      type: 'string',
      validation: (Rule) => Rule.required().min(3).max(100),
    }),
    defineField({
      name: 'description',
      title: 'Description',
      type: 'text',

    }),
    defineField({
      name: 'tags',
      title: 'Tags',
      type: 'array',
      of: [{ type: 'string' }],
    }),
    defineField({
      name: 'badge',
      title: 'Badge',
      type: 'array',
      of: [{ type: 'string' }],
    }),
    defineField({
      name: 'stock',
      title: 'Stock',
      type: 'number',

    }),
    defineField({
      name: 'imageurl',
      title: 'Image URL',
      type: 'image',
      options: {
        hotspot: true, // Allows you to crop and focus the image
      },
    }),
    defineField({
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],

    }),
```

```
TS product.ts M  JS importSanityData.mjs  page.tsx ...\[id]  header.tsx M  handlesearch.tsx
src > sanity > schemaTypes > TS product.ts > default
export default defineType({
  fields: [
    }),
    defineField({
      name: 'price',
      title: 'Price',
      type: 'number',

    }),
    defineField({
      name: 'code',
      title: 'Code',
      type: 'string',

    }),
    defineField({
      name: 'reviews',
      title: 'Reviews',
      type: 'number',

    }),
    defineField({
      name: 'originalprice',
      title: 'Original Price',
      type: 'number',
    }),
    defineField({
      name: 'dimensions',
      title: 'Dimensions',
      type: 'string',
    }),
    defineField({
      name: 'material',
      title: 'Material',
      type: 'string'
    }),
    defineField({
        name: 'categories',
        title: 'Categories',
        type: 'array',
        of: [{ type: 'string' }],
        validation: (Rule) => Rule.min(1).error('At least one category is required.'),
      }),
    defineField({
      name: 'id',
      title: 'Product ID',
      type: 'string',
      validation: (Rule) => Rule.required(),
    }),
  ],
});
```

3. Data Migration Process

Objective: Streamline the migration process using the custom API.

Methods Used:

Using the Custom API:

4. API Integration in Next.js

Objective: Integrate the custom API into the Next.js frontend project with robust error handling and modular coding.

Steps Taken:

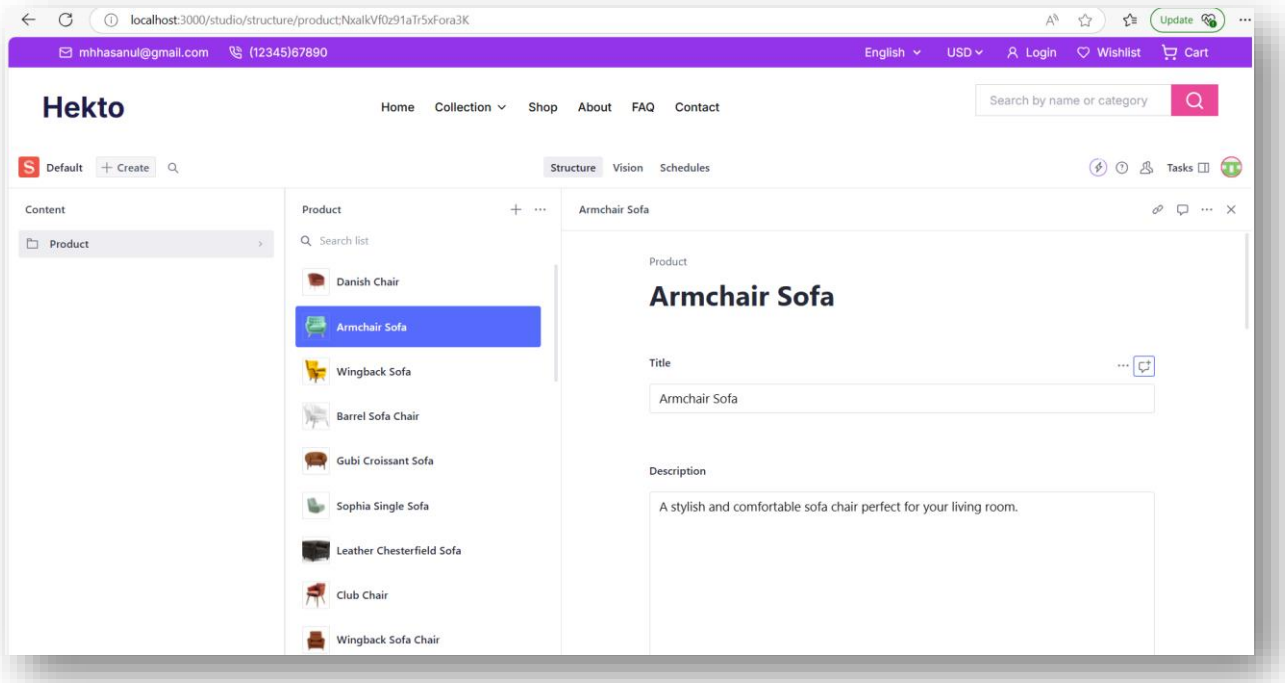
- **Step 1:** Created utility functions for API calls.
- **Step 2:** Rendered data dynamically in React components.
- **Step 3:** Tested API endpoints using Postman and browser developer tools.

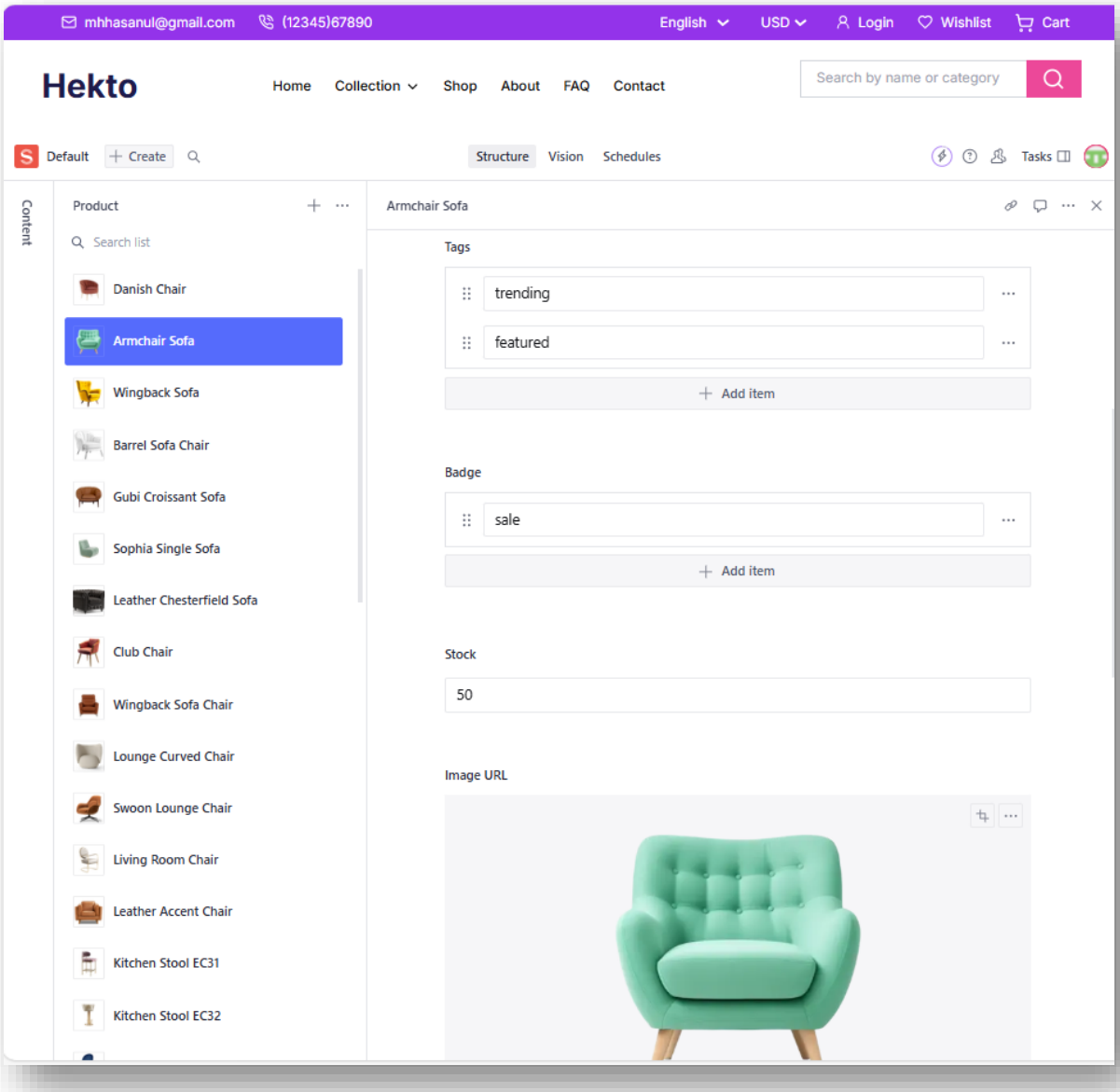
Code Example:

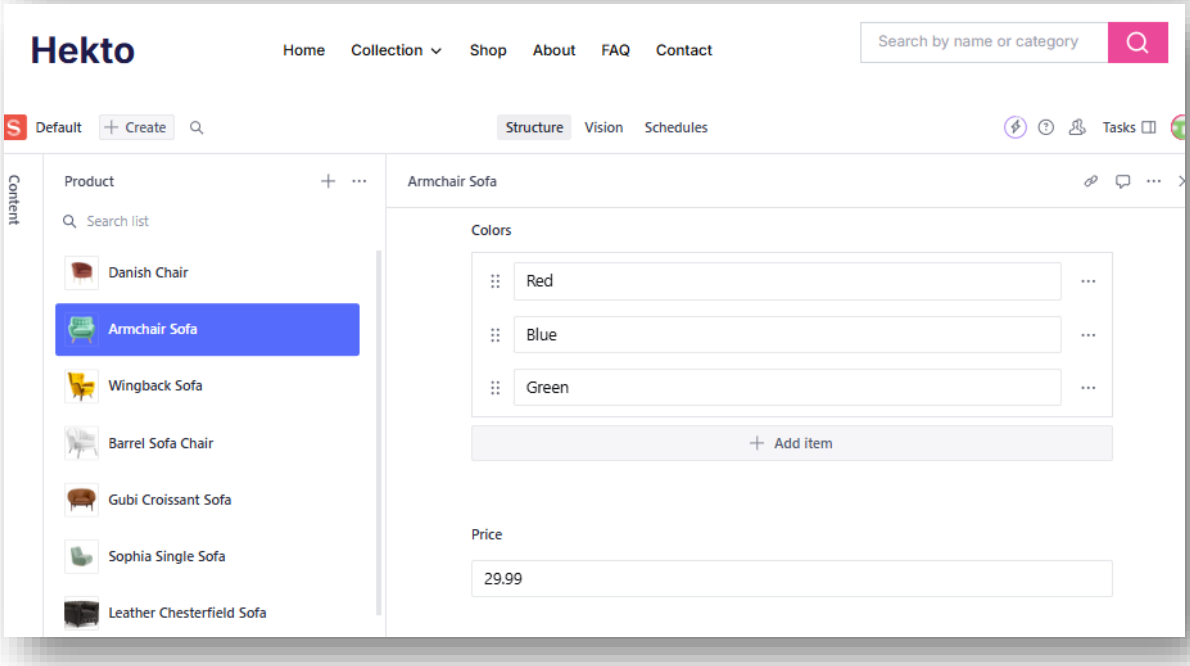
```
    console.log('Data import completed successfully!');
  } catch (error) {
    console.error('Error importing data:', error.message);
  }
}
```

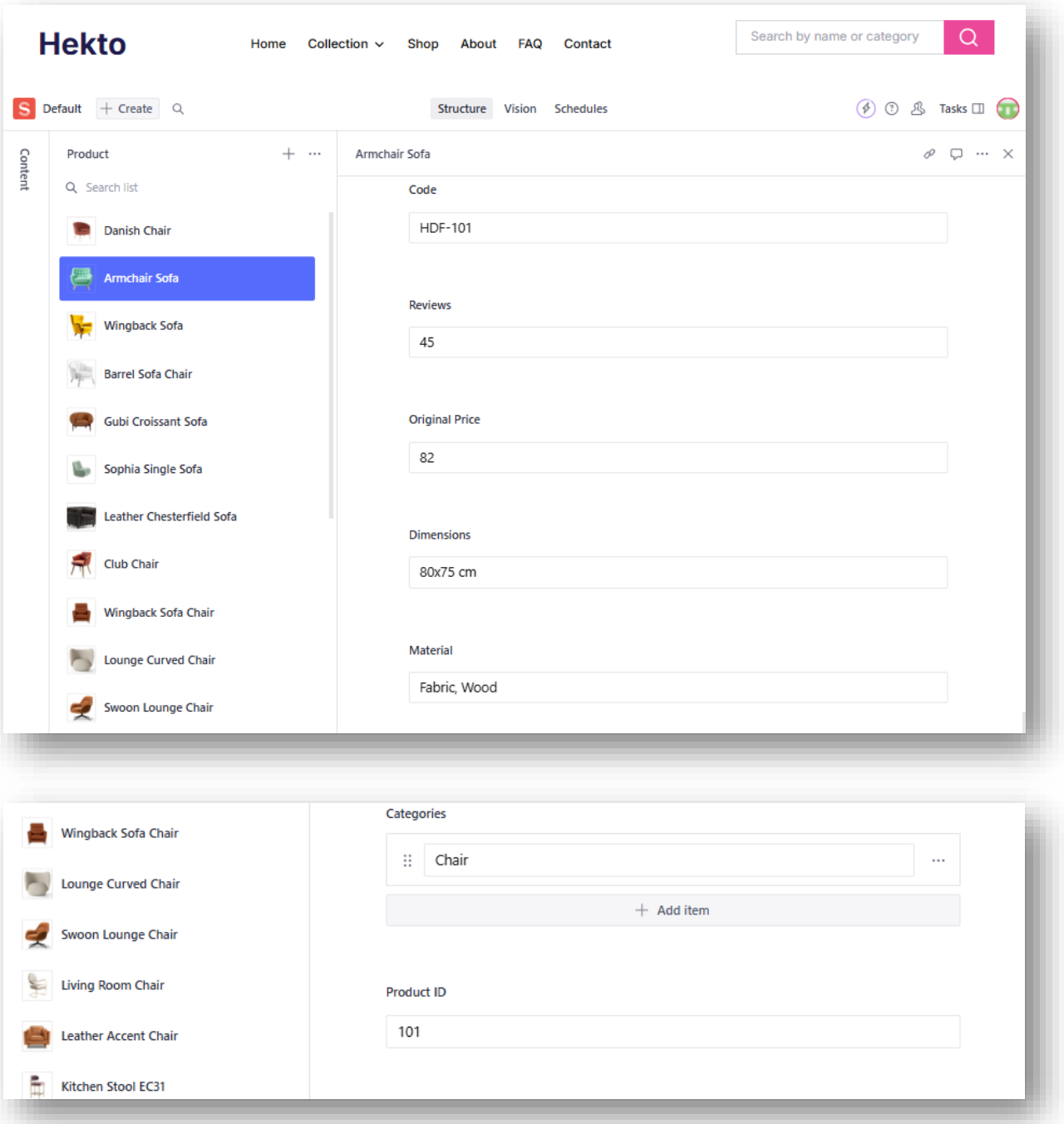
- Log errors in a centralized file for debugging.
- Display user-friendly error messages in the UI.
- Use fallback data or skeleton loaders for better UX.

Conclusion This project demonstrated the seamless integration of a custom API with a CMS like Sanity and highlighted the importance of efficient data migration and fetching techniques. The final solution not only enhanced the data management capabilities but also ensured a dynamic and user-friendly web experience.




mhhasanul@gmail.com
(12345)67890
English
USD
Login
Wishlist
Cart
Hekto
Home
Collection
Shop
About
FAQ
Contact
Search by name or category
Default
Create
Structure
Vision
Schedules
Tasks
Content
Product
Search list
Danish Chair
Armchair Sofa
Wingback Sofa
Barrel Sofa Chair
Gubi Croissant Sofa
Sophia Single Sofa
Leather Chesterfield Sofa
Club Chair
Wingback Sofa Chair
Lounge Curved Chair
Swoon Lounge Chair
Living Room Chair
Leather Accent Chair
Kitchen Stool EC31
Kitchen Stool EC32
Armchair Sofa
Tags
trending
featured
Add item
Badge
sale
Add item
Stock
50
Image URL


Hekto
Home
Collection
Shop
About
FAQ
Contact
Search by name or category
Default
Create
Structure
Vision
Schedules
Tasks
Content
Product
Search list
Danish Chair
Armchair Sofa
Wingback Sofa
Barrel Sofa Chair
Gubi Croissant Sofa
Sophia Single Sofa
Leather Chesterfield Sofa
Armchair Sofa
Colors
Red
Blue
Green
Add item
Price
29.99



Future Enhancements

- Automating the migration process to handle periodic updates.
- Enhancing the API with additional endpoints for more complex queries.
- Exploring advanced GROQ queries for optimized data fetching.