

DAY 3 - API INTEGRATION AND DATA MIGRATION

Overview:

Day 3 revolved around the complete workflow from migrating data to integrating APIs, setting up schemas in Sanity CMS, and fetching/displaying data in a Next.js frontend using GROQ queries. This document details the steps followed and the results achieved.

Key Accomplishments:

API Integration:

- Integrated the provided API for Template One.
- Rendered product listings and categories in the Next.js frontend.

Data Migration:

- Migrated data into Sanity CMS using `importSanityData.mjs`.
- Adjusted schemas for compatibility with the API structure.

Schema Creation & Validation:

- Created and validated `product.ts` schema in Sanity CMS.

GROQ Queries & Data Display:

- Fetched dynamic data using GROQ queries.
- Displayed data in responsive Next.js components.

Error Handling:

- Implemented error handling in API utility functions for smooth data fetching.

Frontend Integration:

- Dynamically displayed API and GROQ data in Next.js components.
- Ensured a responsive and user-friendly interface.

Steps Taken:

Understanding the API:

The process began with understanding the API by reviewing and testing endpoints such as /products and /categories using Postman and browser developer tools. This helped analyze the data structure, required parameters, and potential errors.

Schema Validation:

Next, the API response fields were validated against the existing Sanity CMS schema. Adjustments were made to field names and data types to ensure compatibility, and additional fields were added where necessary.

Data Migration:

Once the schema was aligned, data migration was performed using the importSanityData.mjs script, successfully importing API data into Sanity CMS. The imported data was then verified to ensure accuracy.

Schema Creation:

Following this, a structured schema (product.ts) was created in Sanity CMS, defining essential fields like name, price, category, and description while also setting up relationships between different data entities for better organization.

GROQ Queries Implementation:

To efficiently retrieve and display data, GROQ queries were built and integrated into Next.js components, allowing dynamic fetching from Sanity CMS.

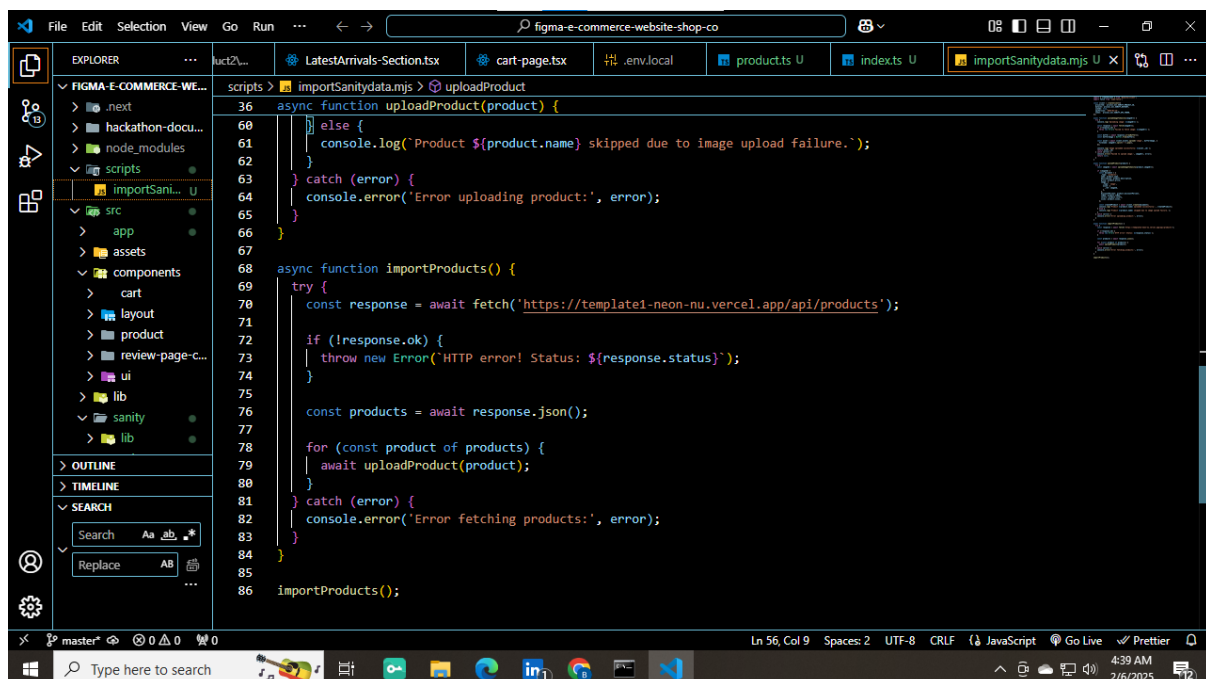
API Integration in Next.js:

API integration was then implemented by creating utility functions to fetch product data, passing it to frontend components, and ensuring seamless rendering.

Error Handling & Optimization:

Error handling was prioritized by centralizing error logging, displaying user-friendly messages in case of failures, and using fallback data along with skeleton loaders to improve user experience.

Function for Data Migration via API:



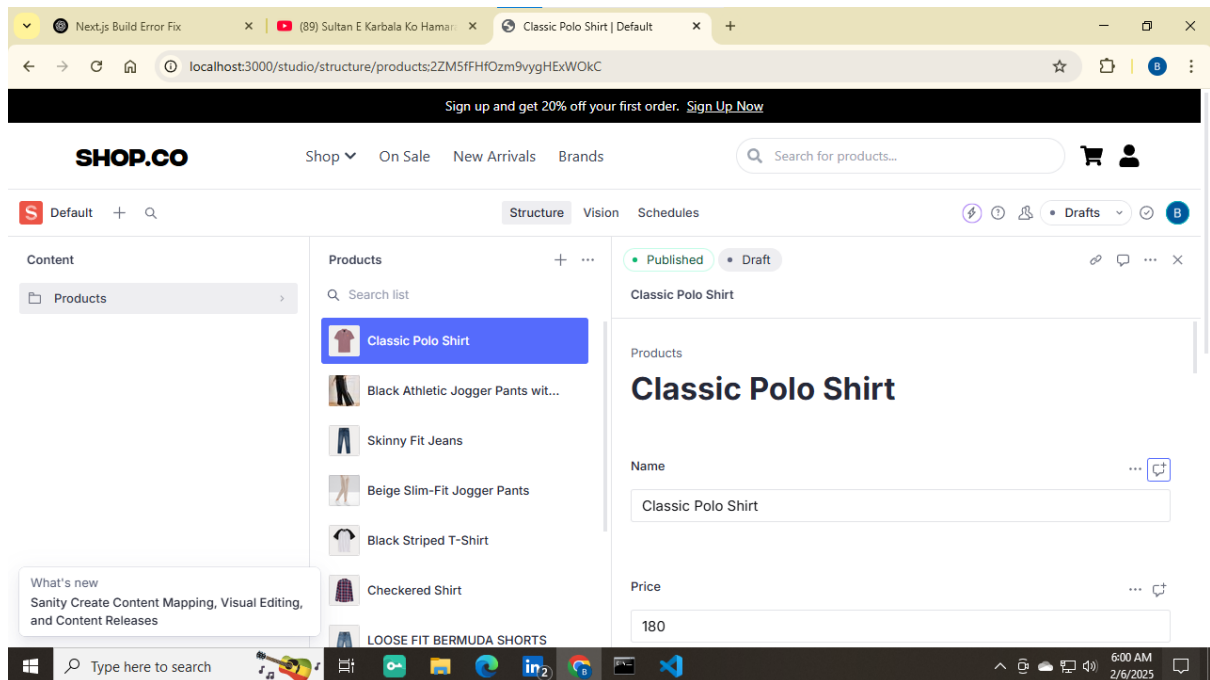
```
36 async function uploadProduct(product) {
60   } else {
61     console.log('Product ${product.name} skipped due to image upload failure.');
```

```
62   }
63 } catch (error) {
64   console.error('Error uploading product:', error);
65 }
66 }
67
68 async function importProducts() {
69   try {
70     const response = await fetch('https://template1-neon-nu.vercel.app/api/products');
```

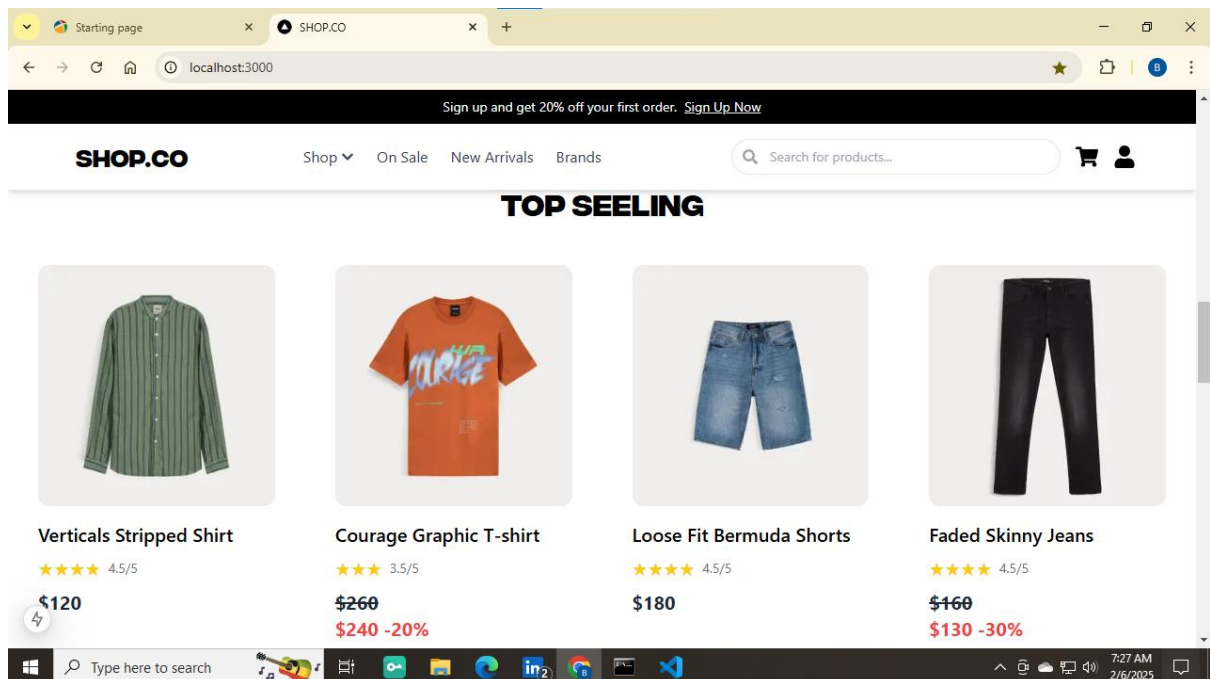
```
71   if (!response.ok) {
72     throw new Error('HTTP error! Status: ${response.status}');
```

```
73   }
74 }
75
76 const products = await response.json();
77
78 for (const product of products) {
79   await uploadProduct(product);
80 }
81 } catch (error) {
82   console.error('Error fetching products:', error);
83 }
84 }
85
86 importProducts();
```

Data in Sanity CMS:



Data Display in Frontend:



Prepared By Bilal Hassan

Adopted Best Practices:

- **Secure Storage of API Keys:** Used .env files to securely store API keys.
- **Modularized Code:** Organized utility functions into separate modules for better code reuse and maintainability.
- **Data Validation:** Validated data during migration to ensure its accuracy and integrity.
- **Documentation:** Documented each step of the process for future reference and ease of understanding.
- **Version Control:** Used version control to track progress and changes, ensuring easy collaboration and rollbacks if needed.

Self-Review Checklist:

Task	Status
Understanding API	✓
Schema Validation	✓
Migration Data	✓
Implement GROQ Queries	✓
Integration API in Next.js	✓
Submission Preparations	✓

Conclusion:

Day 3 tasks were successfully completed, showcasing expertise in API integration, data migration, schema creation, GROQ query implementation, and frontend rendering. This experience strengthened practical skills in building scalable marketplaces using Sanity CMS and Next.js, providing a solid foundation for future projects.