DAY 3 - API INTEGRATION AND DATA MIGRATION

Overview :

The focus for Day 3 was to cover the entire process from data migration to API integration, schema creation in Sanity CMS, and displaying data using GROQ queries in a Next.js frontend. This document outlines the steps taken to achieve these objectives and highlights the outcomes.

Key Tasks Completed:

- **API Integration**:
  - Successfully integrated the provided API for Template one.
  - Rendered data from the API in the Next.js frontend, including product listings and categories.
- **Data Migration**:
  - Utilized the migration script importSanityData.mjs to migrate data into Sanity CMS.
  - Adjusted schemas for compatibility with API data structure.
- **Schema Creation and Validation**:
  - Created and validated a schema file product.ts in Sanity CMS to align with the API fields.
- **GROQ Queries and Data Display**:
  - Implemented GROQ queries to fetch data dynamically from Sanity CMS.
  - Displayed fetched data in the Next.js frontend with responsive components.
- **Error Handling**:
  - Implemented error handling mechanisms in API utility functions to ensure smooth data fetching and display.

- **Frontend Integration**:
  - Displayed API and GROQ data dynamically in the Next.js frontend components.
  - Ensured responsive design and user-friendly data visualization.

Steps Taken :

Understand the API: Reviewed and tested endpoints (e.g., /products, /categories) using Postman and browser tools.

Validate Schema: Compared API response fields with Sanity CMS schema, adjusted field names/types, and added extra fields.

Data Migration: Used a migration script to import API data into Sanity CMS, verifying the import.

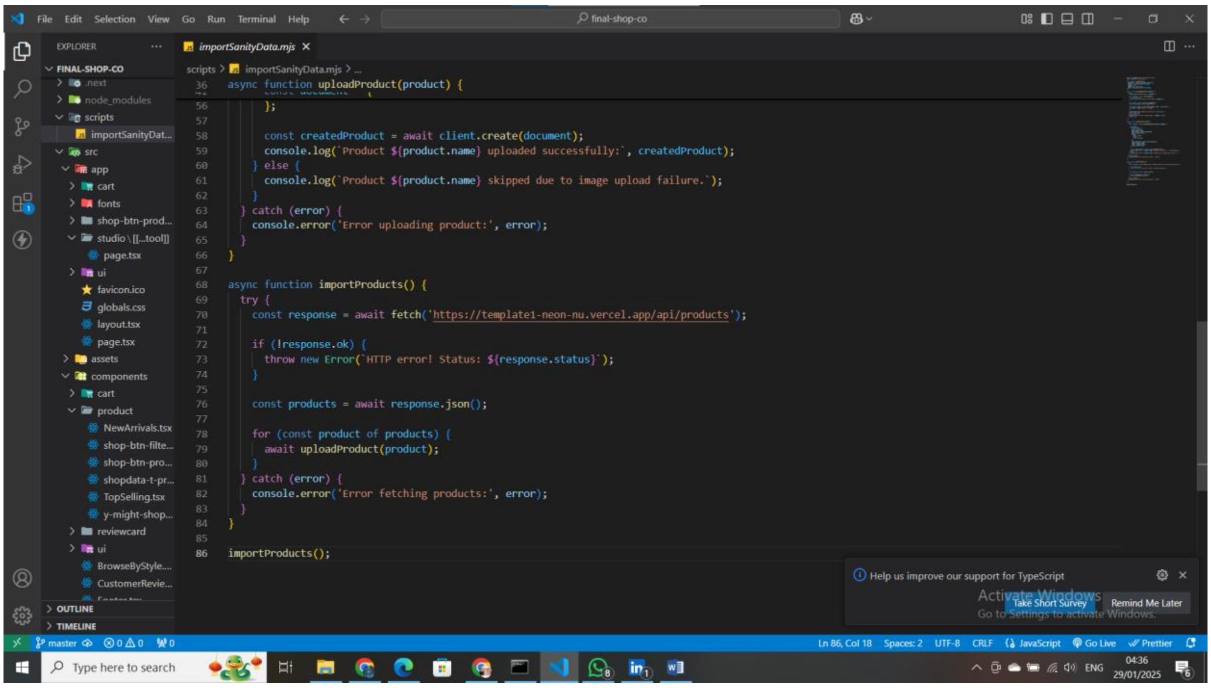
Schema Creation: Defined schema in Sanity with necessary fields (name, price, category, description etc) and set relationships.

GROQ Queries: Built GROQ queries to fetch data efficiently and integrated them into Next.js components.

API Integration: Created utility functions for fetching product data, passed it to frontend components, and added error handling.

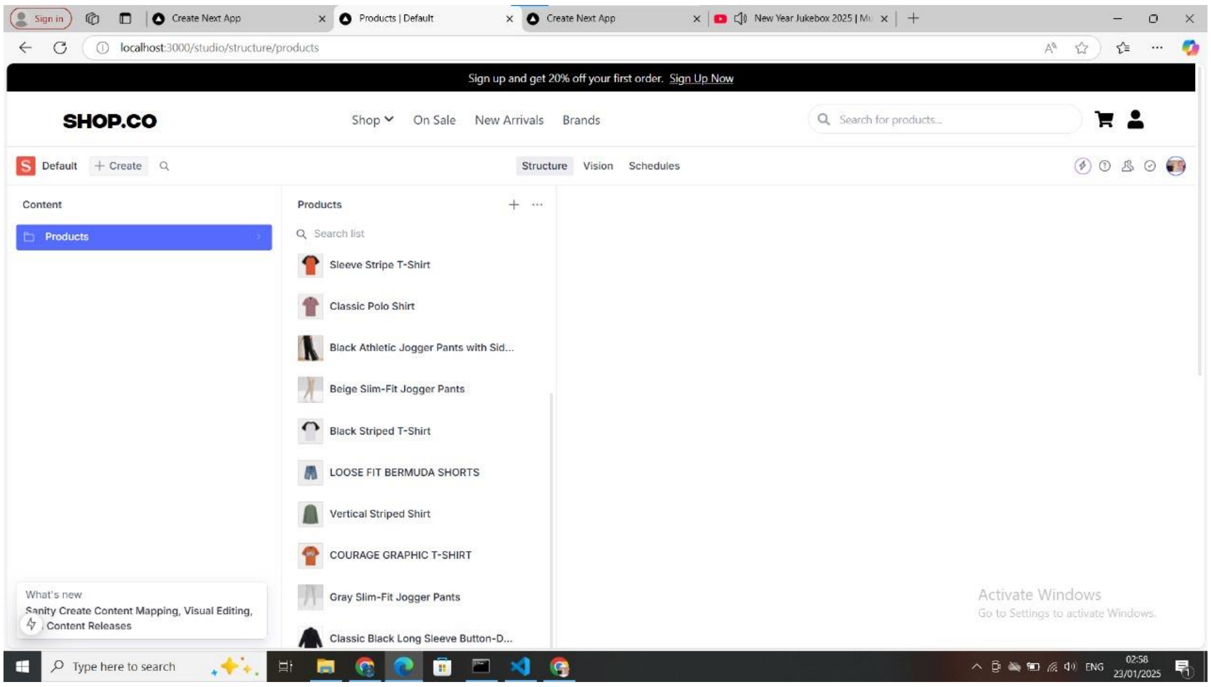
Error Handling: Centralized error logging, displayed user-friendly error messages, and used fallback data and skeleton loaders.

API Function :



```
async function uploadProduct(product) {
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.name} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.name} skipped due to image upload failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template1-neon-nu.vercel.app/api/products');

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

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

Sanity CMS Data :



localhost:3000/studio/structure/products

Sign up and get 20% off your first order. Sign Up Now

SHOP.CO

Content

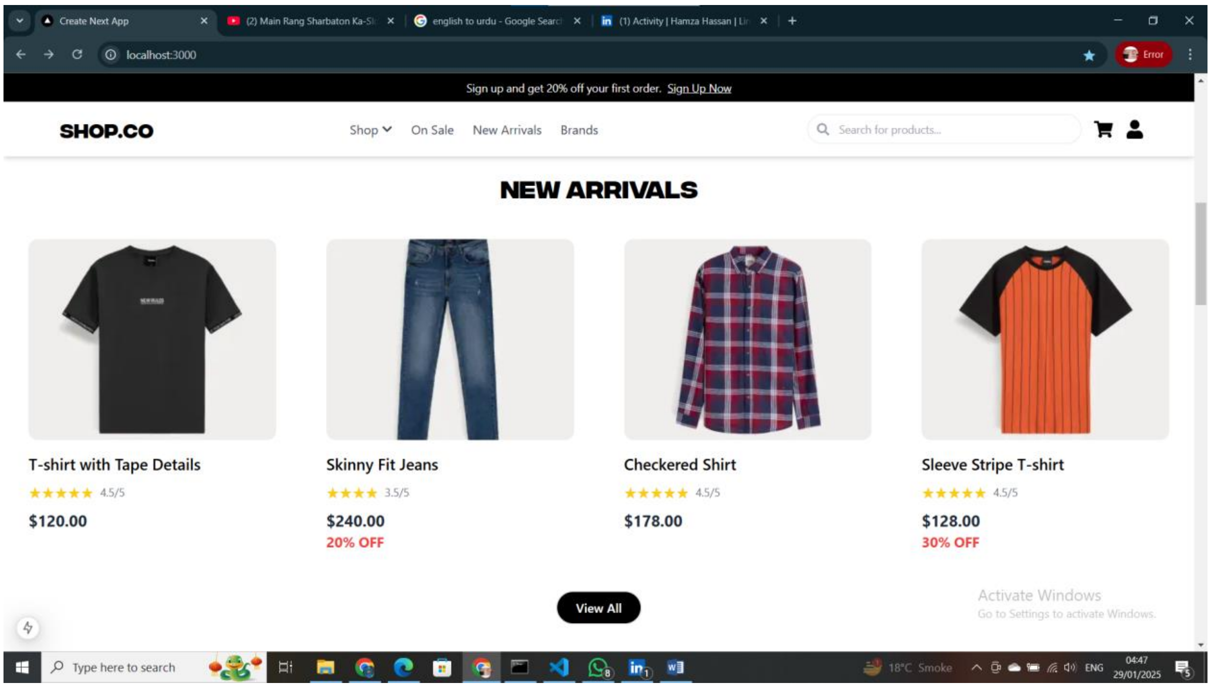
Products

- Sleeve Stripe T-Shirt
- Classic Polo Shirt
- Black Athletic Jogger Pants with Sid...
- Beige Slim-Fit Jogger Pants
- Black Striped T-Shirt
- LOOSE FIT BERMUDA SHORTS
- Vertical Striped Shirt
- COURAGE GRAPHIC T-SHIRT
- Gray Slim-Fit Jogger Pants
- Classic Black Long Sleeve Button-D...

What's new
Sanity Create Content Mapping, Visual Editing, Content Releases

Prepared By Hamza Hassan

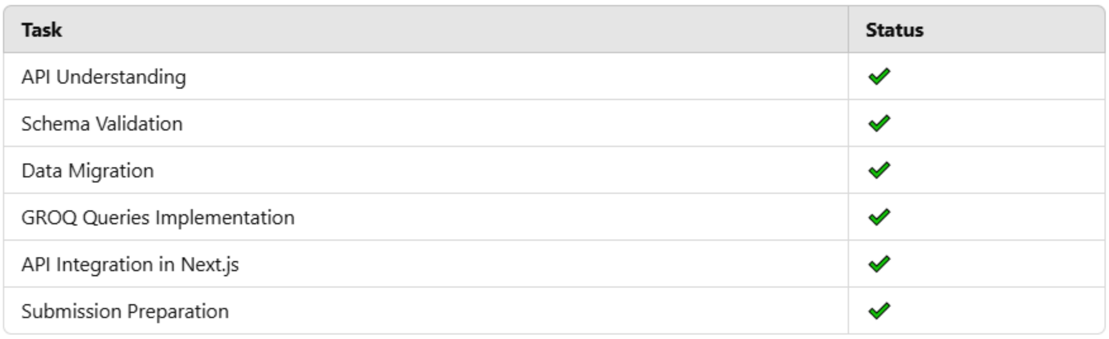
Front-end Data Display :



Best Practices Followed :

1. Used `.env` files to securely store API keys.
2. Modularized utility functions for better code reuse.
3. Validated data during migration to ensure accuracy.
4. Documented every step for future reference.
5. Used version control to track progress and changes.

Self-Validation Checklist :



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

Conclusion :

The Day 3 tasks were successfully completed, demonstrating proficiency in API integration, data migration, schema creation, GROQ query implementation, and frontend rendering. This experience reinforces practical skills for building scalable marketplaces using Sanity CMS and Next.js.