**Day 3**

# **Marketplace Name:** Women's Fashion E-Commerce

## API Integration Process

1. Developed a MockAPI to serve as a template for the integration.
2. Integrated the API with the application, ensuring smooth communication between components.
3. Tested the API endpoints using tools such as Postman to ensure correct functionality and data retrieval.

## Adjustments Made to Schemas

1. Modified the product schema to meet specific application requirements.
2. Updated validation rules to enhance data integrity.
3. Adjusted field types to align with the data structure and improve efficiency.

## Migration Steps and Tools Used

1. Updated migration scripts to move existing data to the new schema format.
2. Used different tools like AI and YouTube tutorials and documents.

## Product Schema Code:

```
import { defineType, defineField } from 'sanity';

export default defineType({
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    defineField({
      name: 'id',
      title: 'ID',
      type: 'number',
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: 'name',
      title: 'Name',
      type: 'string',
      validation: (Rule) => Rule.required().min(3).max(100),
    }),
    defineField({
      name: 'shortdesc',
      title: 'Short Description',
      type: 'string',
      validation: (Rule) => Rule.required().min(10).max(150),
    }),
```

```
    defineField({
      name: 'fulldesc',
      title: 'Full Description',
      type: 'text',
      validation: (Rule) => Rule.required().min(10).max(500),
    }),
    defineField({
      name: 'price',
      title: 'Price',
      type: 'number',
      validation: (Rule) => Rule.required().positive(),
    }),
    defineField({
      name: 'rating',
      title: 'Rating',
      type: 'number',
      validation: (Rule) => Rule.required().min(1).max(5),
    }),
    defineField({
      name: 'image',
      title: 'Image URL',
      type: 'url',
      validation: (Rule) =>
        Rule.required().uri({
          scheme: ['http', 'https'],
        }),
    }),
    defineField({
      name: 'category',
      title: 'Category',
      type: 'string',
      validation: (Rule) => Rule.required(),
    }),
  ],
});
```

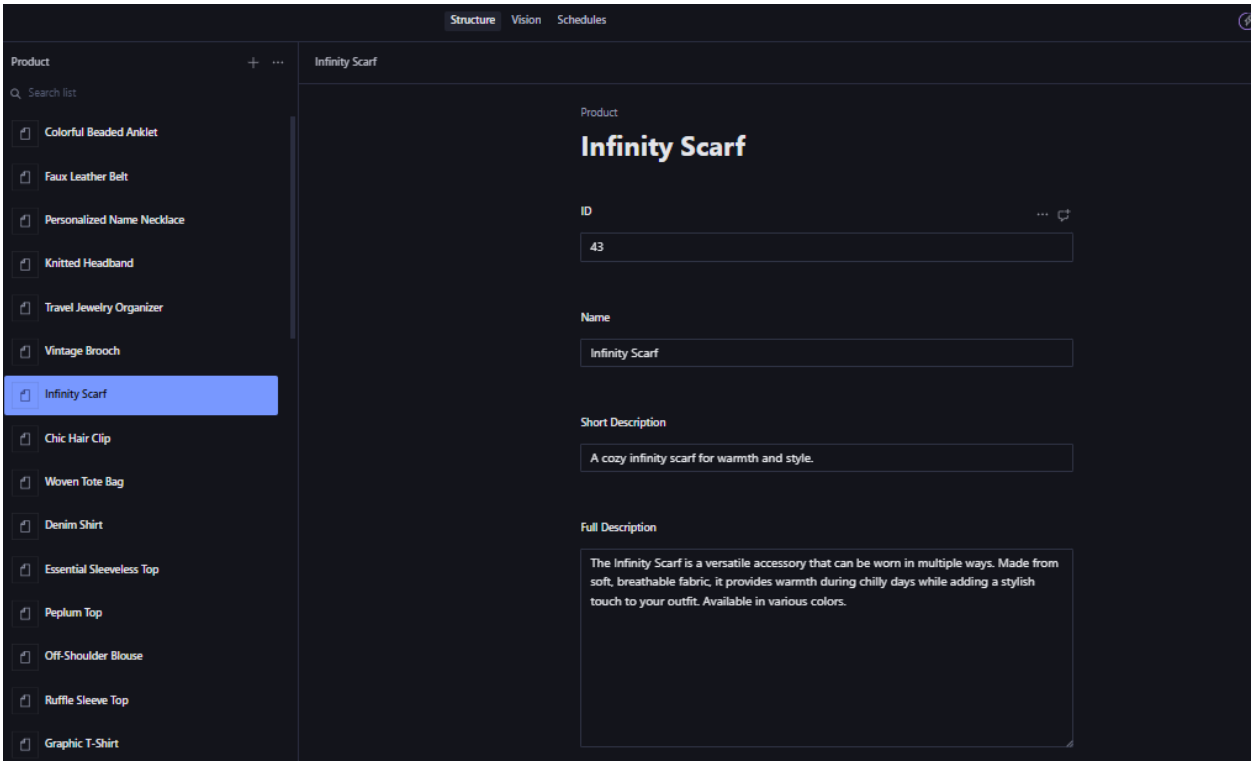
## Migration Script:

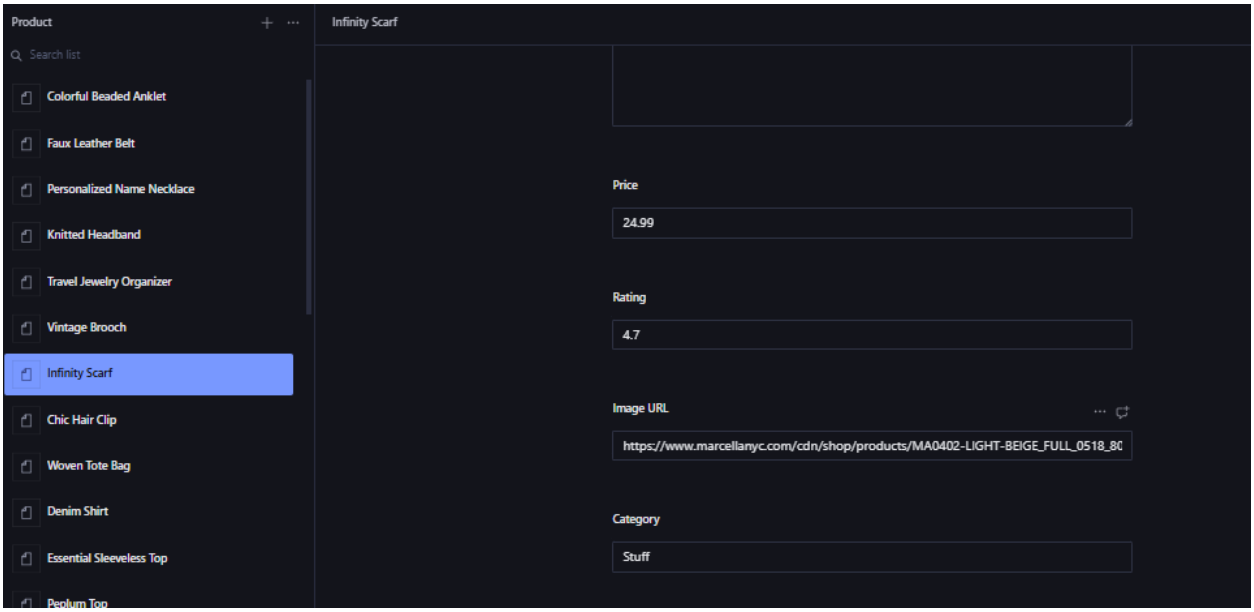
```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
```

```
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2023-05-25', // Use current date
});

async function importData() {
  try {
    console.log('Fetching products from API...');

    // Replace with your actual API endpoint
    const response = await
axios.get('https://678bc3461a6b89b27a2b5cd2.mockapi.io/products');
    const products = Array.isArray(response.data) ? response.data :
[response.data];

    console.log(`Fetched ${products.length} products`);

    for (const product of products) {
      try {
        console.log(`Processing product: ${product.name}`);

        // Directly use the product object as it matches the schema
        const sanityProduct = {
          _type: 'product',
          ...product  // Spread the entire product object
        };

        console.log('Uploading product to Sanity:', sanityProduct.name);

        const result = await client.create(sanityProduct);
        console.log(`Product uploaded successfully: ${result._id}`);
      } catch (productError) {
        console.error(`Error processing product ${product.name}:`,
productError.message);
      }
    }

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

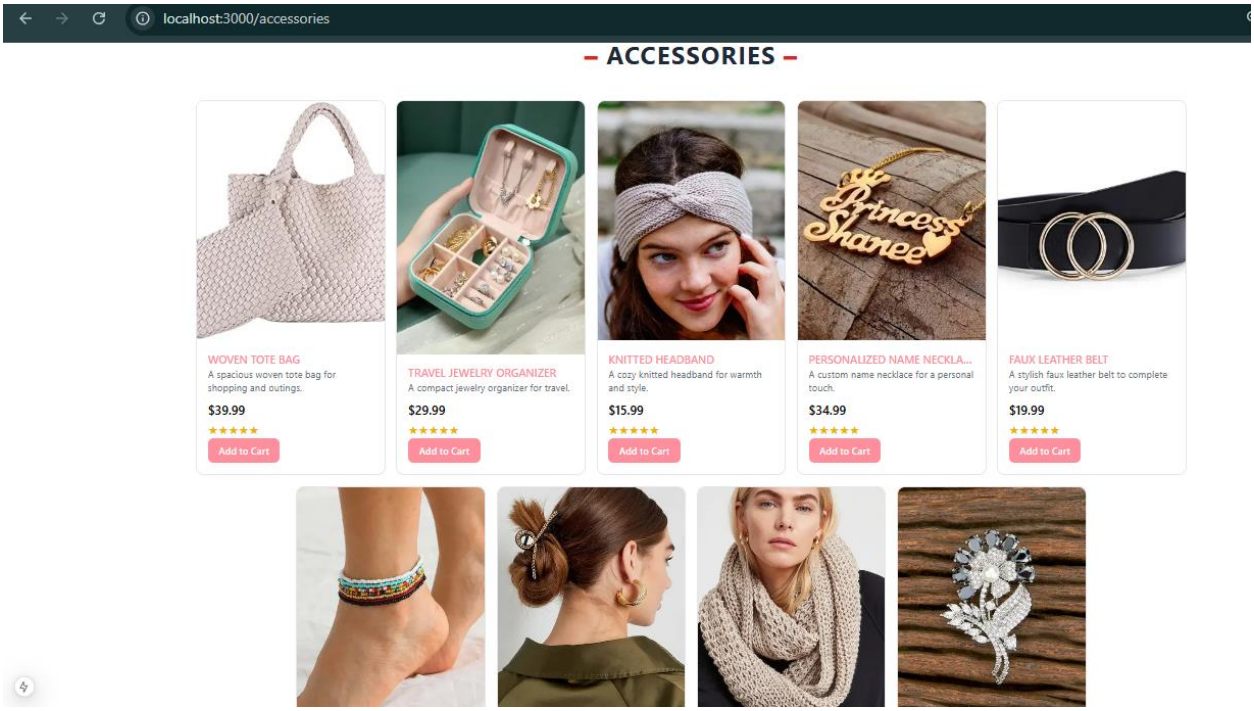
```
// Optional: Add a delay between imports to avoid rate limiting
async function importWithDelay() {
  try {
    await importData();
  } catch (error) {
    console.error('Import failed:', error);
  }
}

importWithDelay();
```

## Populated CMS field:






## Data Successfully Displayed:




## Day 3 Checklist:

### Self-Validation Checklist:

API Understanding: • ✔

Schema Validation: • ✔

Data Migration: • ✔

API Integration in Next.js: • ✔

Submission Preparation: • ✗