Day 3: API Integration and Schema Creation - Sports E-commerce Marketplace

1. Introduction

This document details the work completed on Day 3, focusing on integrating data from an external API into Sanity and creating the necessary Sanity schema for products. This integration enables populating the marketplace with product data.

2. Sanity Schema Creation

- **Product Schema (product.ts):**
  JavaScript
```javascript
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'description', type: 'string', title: 'Description' },
    { name: 'price', type: 'number', title: 'Product Price' },
    { name: 'discountPercentage', type: 'number', title: 'Discount
Percentage' },
    { name: 'priceWithoutDiscount', type: 'number', title: 'Price
Without Discount', description: 'Original price before discount' },
    { name: 'rating', type: 'number', title: 'Rating', description:
'Rating of the product' },
    { name: 'ratingCount', type: 'number', title: 'Rating Count',
description: 'Number of ratings' },
    { name: 'tags', type: 'array', title: 'Tags', of: [{ type:
'string' }], options: { layout: 'tags' }, description: 'Add tags like
"new arrival", "bestseller", etc.' },
    { name: 'sizes', type: 'array', title: 'Sizes', of: [{ type:
'string' }], options: { layout: 'tags' }, description: 'Add sizes
like S , M , L , XL , XXL' },
    { name: 'image', type: 'image', title: 'Product Image', options:
{ hotspot: true } }
  ]
};
```

- **Schema Integration (index.ts):**
  JavaScript
```javascript
import { type SchemaTypeDefinition } from 'sanity'
import product from './product'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product],
}
```

- **Explanation:** The product schema defines the structure of product documents in Sanity. It includes fields for name, description, price, discounts, ratings, tags, sizes, and product images. The index.ts file registers the product schema, making it available in the Sanity Studio.

3. **Data Import Script (importSanityData.mjs)**

- **Code:**

```
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: '2021-08-31',
});
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop(),  });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}
async function importData() {
  try {
    console.log('Fetching products from API...');
    const response = await axios.get('https://fakestoreapi.com/products');
    const products = response.data;
    console.log(`Fetched ${products.length} products`);
    for (const product of products) {
      console.log(`Processing product: ${product.title}`);
      let imageRef = null;
      if (product.image) {
        try { // Try-catch for image upload
          imageRef = await uploadImageToSanity(product.image);
        } catch (imageError) {
          console.error(`Error uploading image for ${product.title}:`, imageError);
          // You could set a placeholder image ref here if needed:
          // imageRef = "placeholder_image_id";  // Replace with your placeholder ID
        }
      }
      const sanityProduct = {
        _type: 'product',
        title: product.title, // Use title, not name
        description: product.description,
        price: product.price,
        discountPercentage: 0, // Or get it from the API if available
        // priceWithoutDiscount: product.price, // You might calculate this later
        tags: product.category ? [product.category] : [],
        // sizes: [], // Add sizes if available in the API
        productImage: imageRef ? { // Use productImage, not image
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : null, // Handle cases where imageRef is null
        slug: { // Generate slug
          _type: 'slug',
          current: product.title.toLowerCase().replace(/\s+/g, '-').replace(/[^a-zA-Z0-9-]/g, ''),
        },
        isNew: true, // Set to true or get value from the API
      };

      console.log('Uploading product to Sanity:', sanityProduct.title); // Use title here
      try { // Try-catch for product creation
        const result = await client.create(sanityProduct);
        console.log(`Product uploaded successfully: ${result._id}`);
      } catch (productError) {
        console.error(`Error creating product ${product.title}:`, productError);
      }
      // Add a small delay to respect rate limits (e.g., 500ms)
      await new Promise(resolve => setTimeout(resolve, 500));
    }
    console.log('Data import completed successfully!');
  } catch (error) {
    console.error('Error importing data:', error);
  }
}

importData();
```

- **Explanation:** This script performs the following actions:
  1. Sets up environment variables for Sanity.
  2. Creates a Sanity client instance.
  3. Fetches product data from the given API.

4. Iterates through each product:
    - Uploads the product image to Sanity's asset store (if an image URL is available).
    - Creates a new product document in Sanity, mapping the data from the API to the corresponding fields in the product schema.
5. Logs the progress and any errors encountered.

4. Environment Variables

- .env.local file:
```
NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id
NEXT_PUBLIC_SANITY_DATASET=production
SANITY_API_TOKEN=your_sanity_token
```

- **Explanation:** These environment variables store sensitive information (Sanity project ID and API token) and should *not* be committed to version control. The NEXT_PUBLIC_ prefix means the project ID and dataset are accessible on the client-side (important for Sanity integration).

5. Script Execution

- package.json script:
JSON
```json
"scripts": {
  "import-data": "node scripts/importSanityData.mjs"}
```

- **Execution command:** npm run import-data

7. Challenges and Solutions

- **Challenge 1: Data Not Importing:** *"Error: Cannot read properties of undefined (reading 'rate')".*
    - **Solution:** *The API sometimes returns products without a 'rating' object. Added a check:* `product.rating?.rate || 0` *to handle cases where the rating is missing*