API Integration and Data Migration Day 3

(Bandage Report)

Objective:

The focus of Day 3 was to implement API integration with Sanity CMS for the Bandage project. This integration aimed to automate dynamic content updates for the marketplace. By leveraging API data, we replaced the manual data entry process with a more efficient, scalable solution, streamlining content management and ensuring the marketplace remains up-to-date.

1. **Sanity CMS Schema Design:** To ensure the seamless handling of product data, I designed a schema called product in Sanity CMS. The schema includes the following fields:

- **MainFields**

o **title:** The product title (string type).
o **description:** A detailed description of the product (text type)
o **productImage:** The main product image (image type)
o **price:** The price of the product (number type).
o **tags:** An array of tags to categorize the product (array of strings)
o **discountPercentage**: The discount percentage (number type).
o **isNew:** A boolean flag indicating if the product is new (boolean type).

2 API Integration Enhancements:

- **Steps Taken:**

Environment Setup: Continued using `dotenv` for secure environment variables:

- `NEXT_PUBLIC_SANITY_PROJECT_ID`
- `NEXT_PUBLIC_SANITY_DATASET`
- `SANITY_TOKEN`

Sanity Client Creation:

- Used @sanity/client to establish a connection to the Sanity project.
- Configured the client with the project ID, dataset, API version, and authentication token.

Data Fetching:

- Made concurrent API calls using to fetch Product data
- Endpoints accessed:
  - https://template6-six.vercel.app/api/products

Data Processing:

- Iterated through the fetched data
- Uploaded images to Sanity's asset library using the

- client.assets.upload( ) method

.

Sanity Document Creation:

- Transformed fetched data into Sanity-compatible document
- structures. Uploaded each document using client.create( )

3 Error Handling:

- Implemented try-catch blocks for API calls and Sanity operations.
- Logged errors for debugging.

Migration Steps and Tools Used:

1. Migration Steps

Preparation:

- Analyzed API data structure and matched it with existing Sanity
- schemas. Created Sanity schemas for products with necessary field

Data Import Script:

- Built import-data.mjs script to automate data fetching, processing, an uploading.
- Utilized for API interaction and @sanity/client for CM operations

Image Handling:

- Uploaded images to Sanity, storing asset references in document fields

Document Creation

- Mapped API data fields to Sanity schema fields ♣
- Ensured fallback values for optional or missing fields

2. Tools Used

Node.js Modules:

- .dotenv file for environment variable management
- @sanity/client for CMS interactions

Sanity Features:

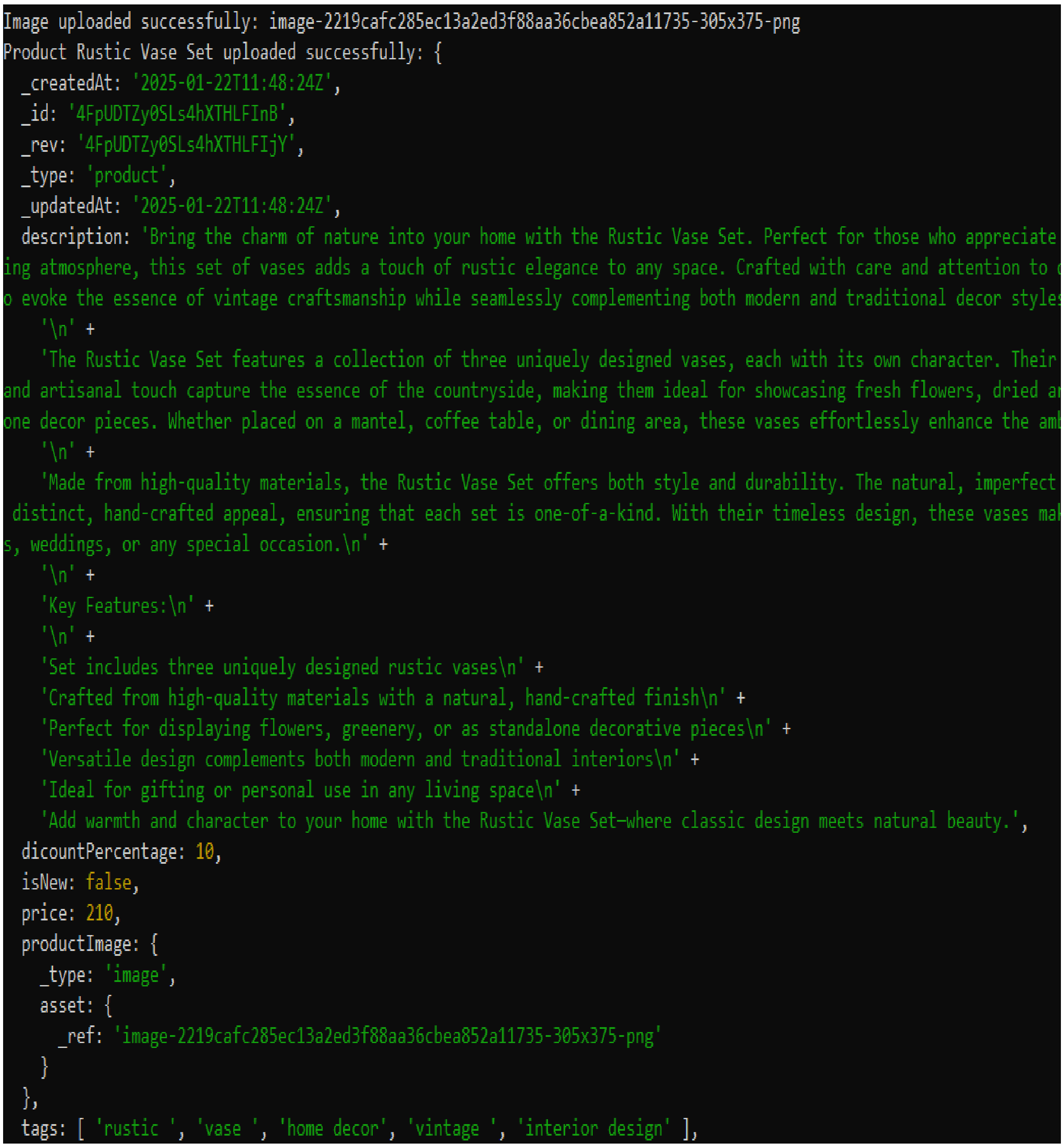
- Asset management for image uploads. API
- versioning for consistent schema support

Utilities:

- fileURLToPath and path for resolving file paths

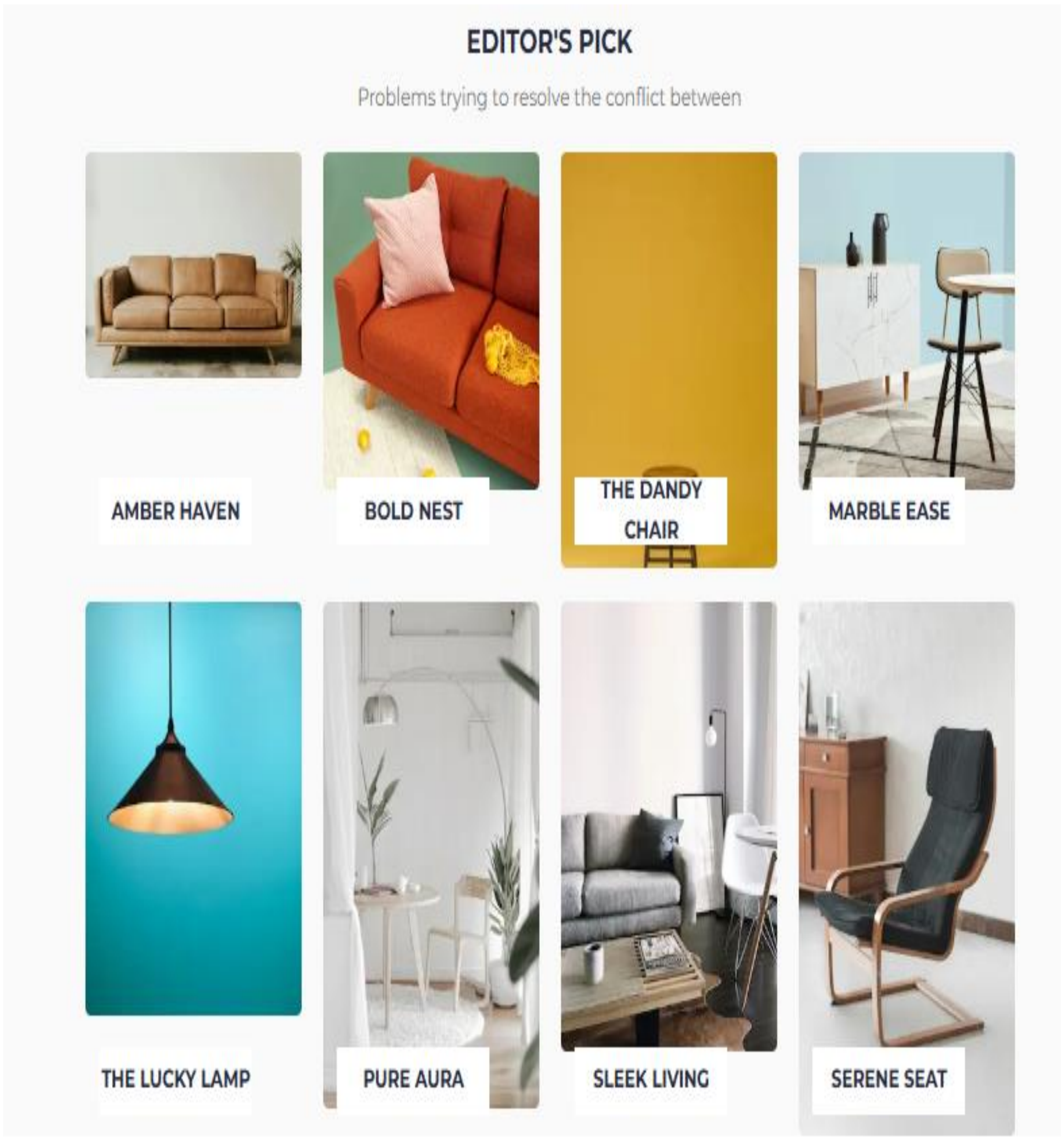
API Calls (Products)

ScreenShort



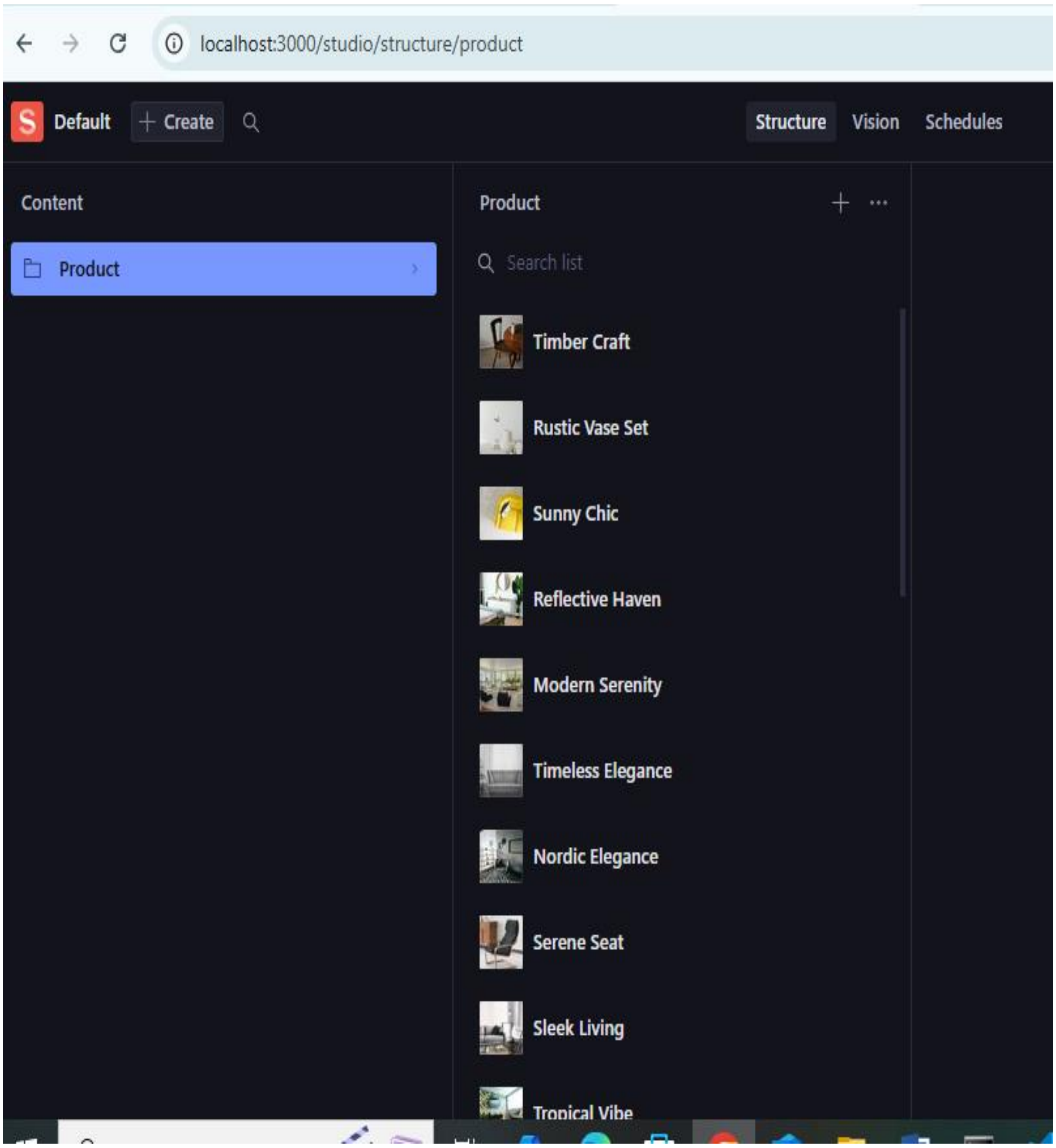
```
Image uploaded successfully: image-2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375-png
Product Rustic Vase Set uploaded successfully: {
  _createdAt: '2025-01-22T11:48:24Z',
  _id: '4FpUDTZy0SLs4hXTHLFInB',
  _rev: '4FpUDTZy0SLs4hXTHLFIjY',
  _type: 'product',
  _updatedAt: '2025-01-22T11:48:24Z',
  description: 'Bring the charm of nature into your home with the Rustic Vase Set. Perfect for those who appreciate
ing atmosphere, this set of vases adds a touch of rustic elegance to any space. Crafted with care and attention to
o evoke the essence of vintage craftsmanship while seamlessly complementing both modern and traditional decor styles
    '\n' +
    'The Rustic Vase Set features a collection of three uniquely designed vases, each with its own character. Their
and artisanal touch capture the essence of the countryside, making them ideal for showcasing fresh flowers, dried a
one decor pieces. Whether placed on a mantel, coffee table, or dining area, these vases effortlessly enhance the amb
    '\n' +
    'Made from high-quality materials, the Rustic Vase Set offers both style and durability. The natural, imperfect
 distinct, hand-crafted appeal, ensuring that each set is one-of-a-kind. With their timeless design, these vases mak
s, weddings, or any special occasion.\n' +
    '\n' +
    'Key Features:\n' +
    '\n' +
    'Set includes three uniquely designed rustic vases\n' +
    'Crafted from high-quality materials with a natural, hand-crafted finish\n' +
    'Perfect for displaying flowers, greenery, or as standalone decorative pieces\n' +
    'Versatile design complements both modern and traditional interiors\n' +
    'Ideal for gifting or personal use in any living space\n' +
    'Add warmth and character to your home with the Rustic Vase Set—where classic design meets natural beauty.',
  dicountPercentage: 10,
  isNew: false,
  price: 210,
  productImage: {
    _type: 'image',
    asset: {
      _ref: 'image-2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375-png'
    }
  },
  tags: [ 'rustic ', 'vase ', 'home decor', 'vintage ', 'interior design' ],
```

Data successfully displayed in the frontend:



Sanity CMS field:

(Product):



localhost:3000/studio/structure/product

Timber Craft

Rustic Vase Set

Sunny Chic

Reflective Haven

Modern Serenity

Timeless Elegance

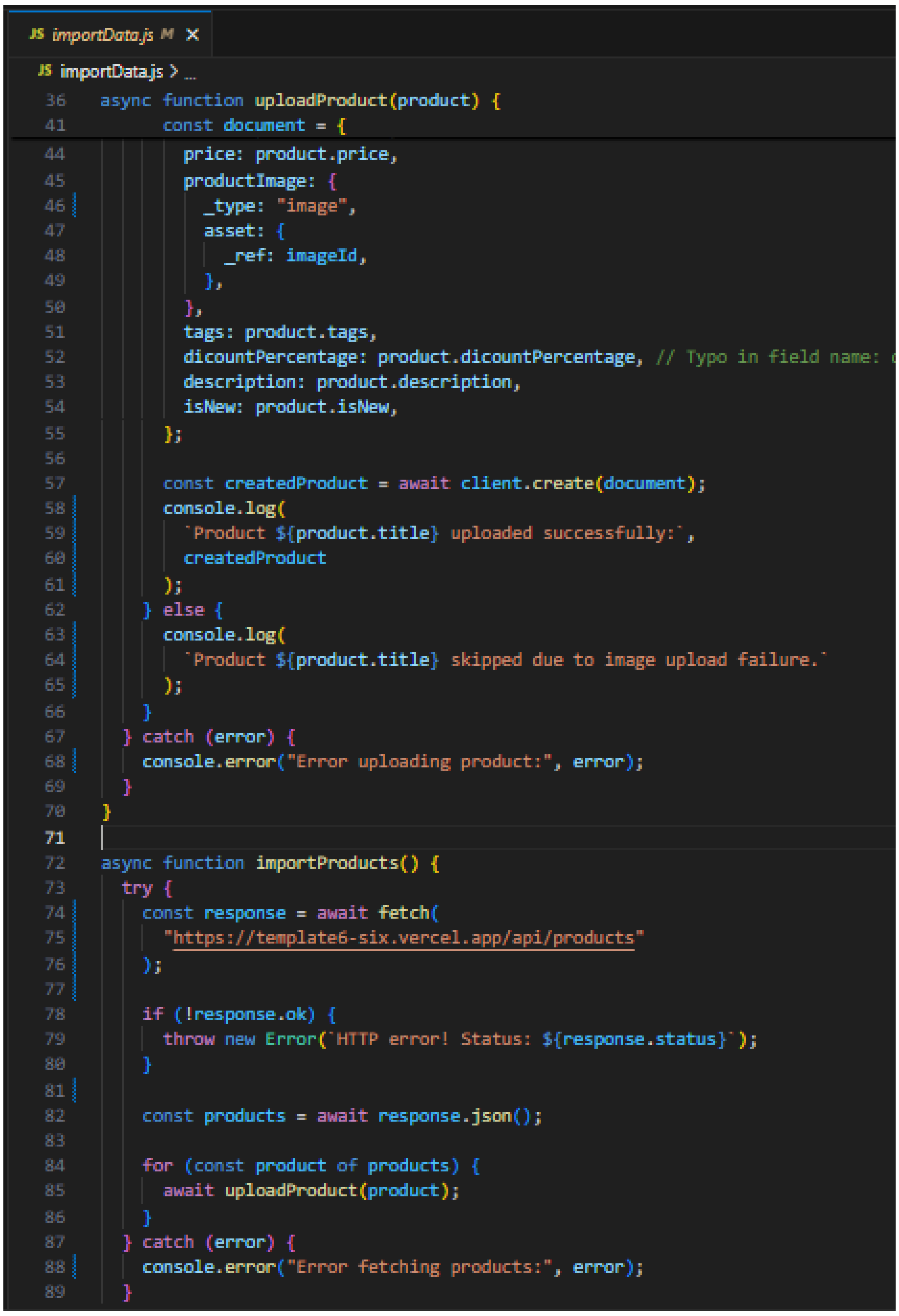
Nordic Elegance

Serene Seat

Sleek Living

Tropical Vibe

Code Snippents for API integration and migration scripts:



```
async function uploadProduct(product) {
      const document = {
        price: product.price,
        productImage: {
          _type: "image",
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo in field name:
        description: product.description,
        isNew: product.isNew,
      };

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

async function importProducts() {
  try {
    const response = await fetch(
      "https://template6-six.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);
  }
```