Day 3: API Integration & Data Migration

Process of API Integration

To begin the API integration, I followed the steps below:

- 1. Creating the Script Folder
- 2. First, I created a script folder at the root of my project directory.
- 3. Creating the importData.mjs File

Inside the script folder, I created a file named importData.mjs.

4. Adding the Script to package.json

In the package.json file, I added a new script under the "scripts" section:

"import-Data": "node script/importData.mjs".

5. Running the Script

After setting up the script, I ran it in the terminal using the command:

npm run import-Data.

6. Fetching Data from the API

Once the script was executed, it successfully fetched the data from the API.

7. Exporting Data to SanityThe script then exported the fetched data to Sanity one by one, completing the integration process.

Adjustments Made to Schema

During the integration process, I made the following adjustments to the schema:

1. Field Name Correction

I corrected the name of a field from new to isNew to ensure successful export of all API data to Sanity.

2. Adding Extra Fields

I added two extra fields—stars and rating—to match the requirements of the UI.

API Calls Data:

\$ npm run import-Data

> ecommerce-website@0.1.0 import-Data

> node script/importData.mjs

Uploading image:

https://cdn.sanity.io/images/7xt4qcah/production/4e2ed6a9eaa6e1413843e53f3113ccfd 2104c301-278x296.pngImage uploaded successfully: image-4e2ed6a9eaa6e1413843e53f3113ccfd2104c301-278x296-png

Product Casual Green Bomber Jacket uploaded successfully: {

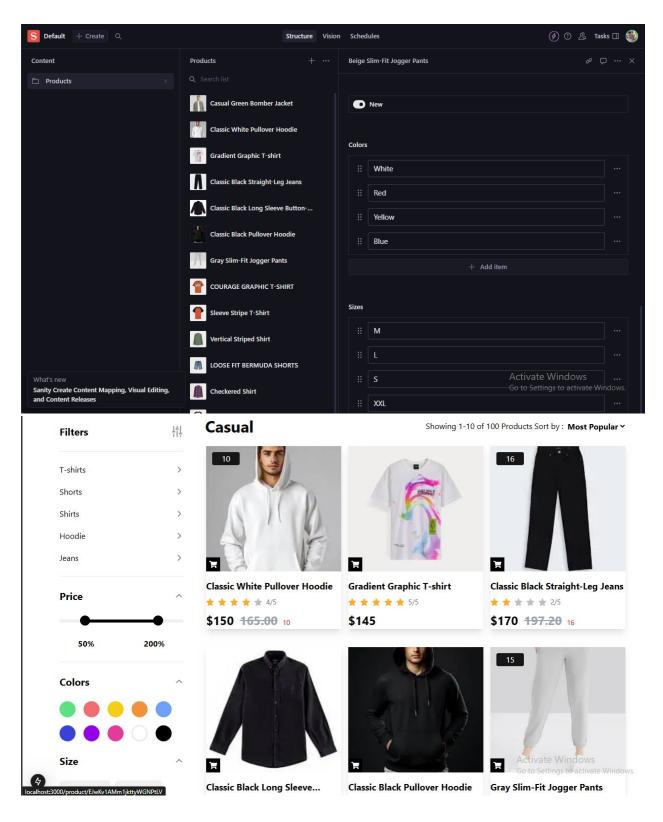
```
_createdAt: '2025-01-17T21:09:56Z',
_id: 'EJwKv1AMm1jkttyWGNCaHJ',
_rev: 'EJwKv1AMm1jkttyWGNCaEH',
_type: 'products',
_updatedAt: '2025-01-17T21:09:56Z',
category: 'hoodie',
colors: [ 'Blue', 'Red', 'Black', 'Yellow' ],
```

description: "This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a sporty yet refined look. The minimalist

style makes it perfect for layering over casual t-shirts or hoodies. Whether you're out with friends or just lounging, this jacket provides a laid-back yet fashionable vibe. Its muted green color adds a subtle, earthy tone that pairs well with a variety of outfits, making it a versatile addition to your casual wardrobe.",

```
discountPercent: 20,
image: {
   _type: 'image',
   asset: {
   _ref: 'image-4e2ed6a9eaa6e1413843e53f3113ccfd2104c301-278x296-png'
   }
},
isNew: true,
name: 'Casual Green Bomber Jacket',
price: 300,
sizes: [ 'S', 'XXL', 'XL', 'L' ]
}
```

Snippets:



```
刘 File Edit Selection View Go Run …
                                                                                    ∠ day-3
                                                                                                                                             ımportData.mjs × □ ···
Ð
       EXPLORER
                              script > ... importData.mjs > ③ importProducts
    import { createClient } from '@sanity/client';
     ∨ DAY-3
       > lo .next
       > node_modules
                                     const client = createClient({
  projectId: 'xvpo7quk',
  dataset: 'production',
       V 🐚 public
        > browse
        > le hero
                                       useCdn: true,
apiVersion: '2025-01-13',
token: 'skQMQxtNGUHmV3fM67NcDwAkFFoH6bvfR7cH7LeGLy3289wbLTDzFWnMy6mBoQujzm0A1hFXZgiITZUdjeKVl7wx8hshn0UtDpdyd
        > m products
        > recomended
        > topProducts
                                     async function uploadImageToSanity(imageUrl) {

√ Im script

        ■ importData.mjs
                                         console.log(`Uploading image: ${imageUrl}`);
       🗸 脯 арр
                                         const response = await fetch(imageUrl);
                                         if (!response.ok) {
  throw new Error(`Failed to fetch image: ${imageUrl}`);
         > 📭 cart
         page.tsx
         > 🙀 fonts
                                        const buffer = await response.arrayBuffer();

✓ Improduct \ fid1

            page.tsx
          > 🖿 studio
                                           filename: imageUrl.split('/').pop(),
            * favicon.ico
            ∃ globals.css
            layout.tsx
                                         console.log(`Image uploaded successfully: ${asset._id}`);
return asset._id;
            page.tsx

∨ I components

∨ ■ context

                                         console.error('Failed to upload image:', imageUrl, error);
          > skeletons
          > 📭 ui
                                     async function uploadProduct(product) {
                                         const imageId = await uploadImageToSanity(product.imageUrl);
                                         if (imageId) {
                                             _type: 'products',
name: product.name,
            # Header.tsx
> OUTLINE
                                             price: product.price,
image: {
                                                                                                         Ln 70, Col 87 Spaces: 2 UTF-8 CRLF {} JavaScript ♀ Go Live ✓ Prettier ♀
                                                                                                                                     参 14℃ ヘ 및 @ ⑴ 4:54 AM 1/19/2025

∠ Type here to search

                                              H 🥠 💽 🚱 🔚 🗊 🦺 💁 刘
                                                                                                          4
```

Fetch Data in UI

I successfully fetched the data in the UI using useContext(). Here's the process:

1. Creating Product Context

2. I created a productContext to hold the fetched data.

3. Adding Data to Context

The data fetched from the API was added to the productContext.

4. Using Context in UI

I utilized the context throughout the UI, ensuring the data was accessible across all components.

Product Context Component:

```
83 \
                                                                                                                       ▼ File Edit Selection View Go Run …
                                                                       D day-3
                                                                        ProductsContext.tsx X products.ts page.tsx ...\app
      EXPLORER

    ⊕ Header.tsx

                                         page.tsx ...\lidl

    ⇔ Hero.tsx

D
                         src > components > context > ∰ ProductsContext.tsx > ❷ ProductsContextProvider
     ∨ DAY-3
      > o .next
      > node modules
                               import type { Product } from "@/types/Product";

✓ Image public

       > browse
       > hero
       > 🦙 home
                                   setAllProducts : Dispatch<SetStateAction<Product[]>>
                                  newProducts : Product[];
setNewProducts : Dispatch<SetStateAction<Product[]>>
       > m products
       > recomended
       > topProducts
                                  setTopProducts : Dispatch<SetStateAction<Product[]>>
         importData.mjs
                                   setRecomendedProducts : Dispatch<SetStateAction<Product[]>>
       🗸 ធ арр
        > 📭 cart

✓ Total category

          page.tsx
                               export const productsContext = createContextproductsContextType | undefined>(undefined)
        > 🙀 fonts

✓ improduct \ [id]

                               export const useproductsContext = () => {
          page.tsx
                                 const context = useContext(productsContext);
        > studio
          ★ favicon.ico

    ∃ alobals.css

          layout.tsx
          page.tsx

∨ I components

                                 const [allProducts, setAllProducts] = useState<Product[]>([]);
const [newProducts, setNewProducts] = useState<Product[]>([]);
         const [topProducts, setTopProducts] = useState<Product[]>([]);
           ProductsCont...
                                 const [recomendedProducts, setRecomendedProducts] = useState<Product[]>([]);
         > skeletons
        > 📭 ui
                                                                      (property) productsContextType.allProducts: Product[]
          Brawse.tsx
          Cartitem.tsx
                                      {children}
          Header.tsx
                               export default ProductsContextProvider:
> OUTLINE
                                                                                                                  Activate Windows
                                                                                       클 12℃ Haze ヘ 및 연 예 5:02 AM 1/19/2025
                              ,++, 🖽 🐠 🥲 🧐 🔚 🖫 🤷 💁 📢
     Type here to search
```

Conclusion

In conclusion, the API integration process was successfully completed by creating a script to fetch data and export it to Sanity. Adjustments were made to the schema to ensure the correct data structure. Additionally, the fetched data was integrated seamlessly into the UI using useContext(), making it available across the application. These steps have laid a strong foundation for further development and ensured that the data flow between the backend and frontend is smooth and efficient.