

# Report Documenting

## 1.API Integration Steps:

- A new Sanity project was created via the Sanity dashboard.
- Added API token, project ID, and dataset to .env.local.
- The client was configured in fetch.ts

```
1  import { createClient } from "next-sanity";
2
3  const client = createClient({
4    projectId: "5sufnfsv",
5    dataset: "production",
6    useCdn: true,
7    apiVersion: "2023-10-10",
8  });
9
10 export async function sanityFetch({
11   query,
12   params = {},
13 }: {
14   query: string;
15   params?: any;
16 }) {
17   return await client.fetch(query, params);
18 }
```

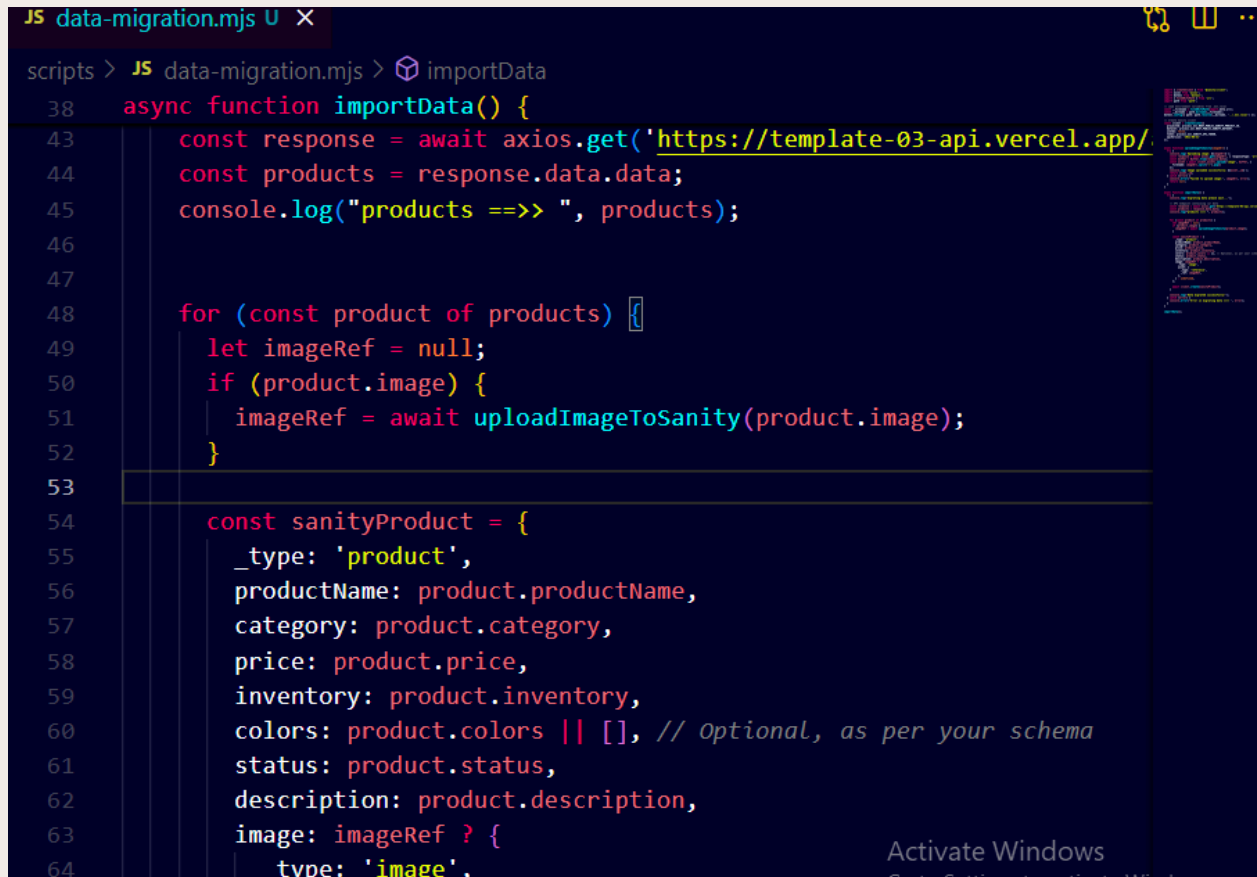
## 2. Schema Adjustments

- The Product.ts schema was created, defining fields like name, description, price, image, and category:

```
src > sanity > schemaTypes > TS product.ts > [🔗] productSchema
1  export const productSchema = {
2      name: 'product',
3      title: 'Product',
4      type: 'document',
5      fields: [
6          {
7              name: 'productName',
8              title: 'Product Name',
9              type: 'string',
10         },
11         {
12             name: 'category',
13             title: 'Category',
14             type: 'string',
15         },
16         {
17             name: 'price',
18             title: 'Price',
19             type: 'number',
20         },
21     ],
22 }
```

### 3. Migration Steps:

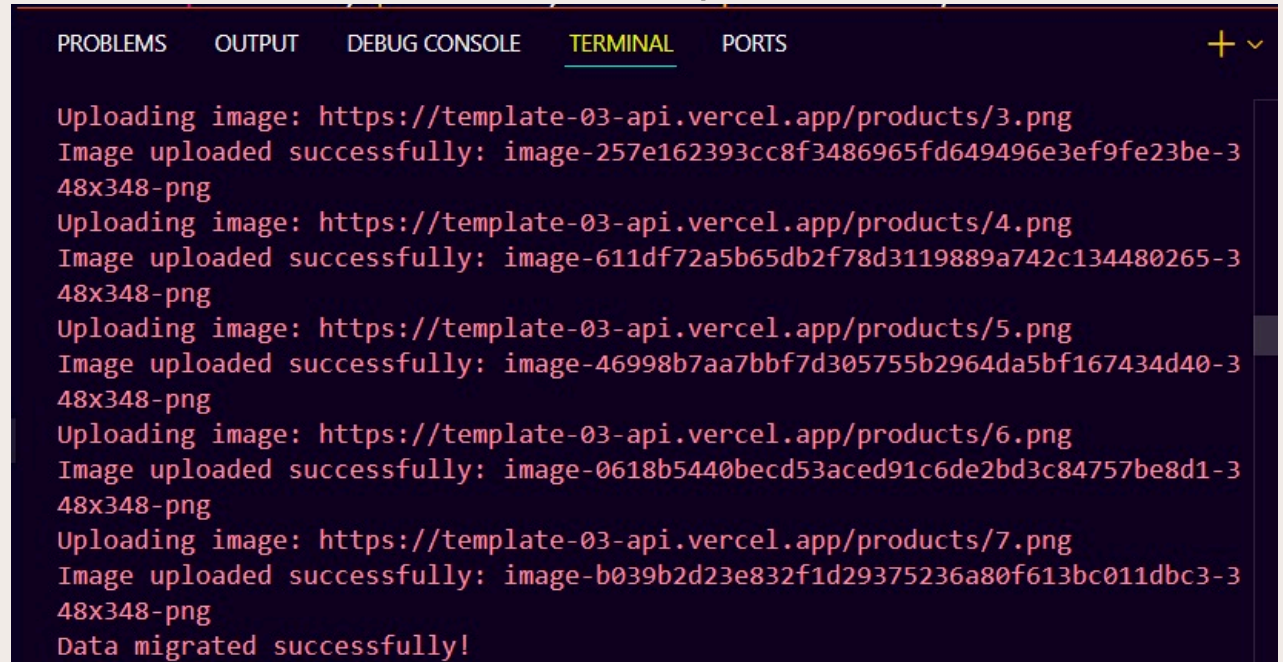
- Step 1: The data-migration.mjs script was created in the root directory, with provided API code pasted to migrate data.
- Step 2: Sanity Client was used to fetch and migrate existing data from Sanity.
- Step 3: API calls were made to fetch data and populate fields in the frontend.
- Step 4: Data was rendered in page.tsx.

A screenshot of a code editor with a dark theme. The file name 'data-migration.mjs' is visible in the top left. The code is written in JavaScript and includes comments. It defines an 'importData' function that fetches data from an API, logs it, and then iterates over the products to upload images to Sanity and create Sanity product objects. The code is partially obscured by a vertical scrollbar on the right.

```
JS data-migration.mjs U X
scripts > JS data-migration.mjs > importData
38  async function importData() {
43      const response = await axios.get('https://template-03-api.vercel.app/');
44      const products = response.data.data;
45      console.log("products ==>> ", products);
46
47
48      for (const product of products) {
49          let imageRef = null;
50          if (product.image) {
51              imageRef = await uploadImageToSanity(product.image);
52          }
53
54          const sanityProduct = {
55              _type: 'product',
56              productName: product.productName,
57              category: product.category,
58              price: product.price,
59              inventory: product.inventory,
60              colors: product.colors || [], // Optional, as per your schema
61              status: product.status,
62              description: product.description,
63              image: imageRef ? {
64                  type: 'image',
```

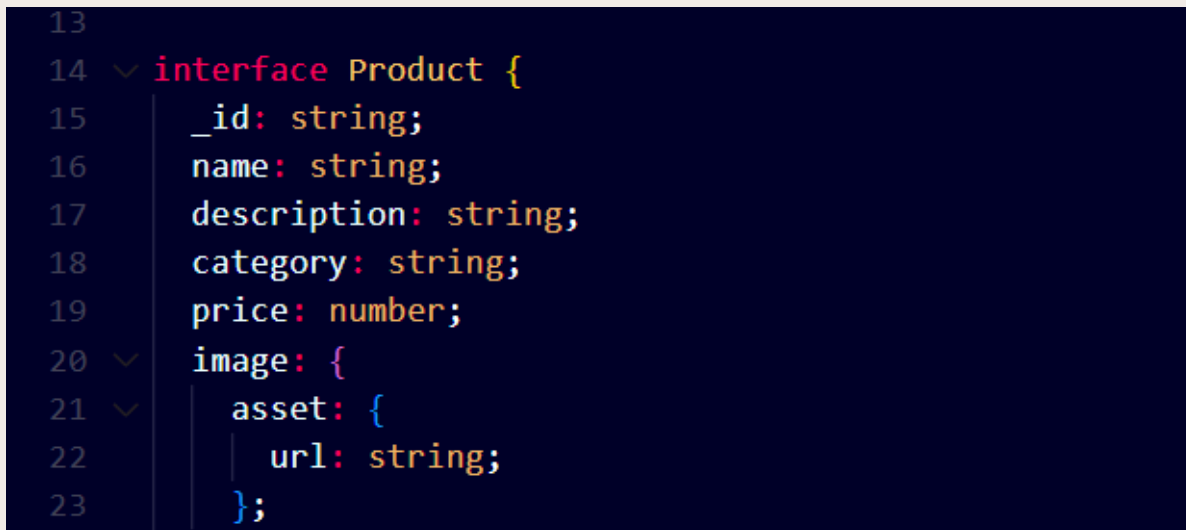
## 4. Screenshot of API Calls

- Provide a screenshot where fetch data successfully from the API.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v

Uploading image: https://template-03-api.vercel.app/products/3.png
Image uploaded successfully: image-257e162393cc8f3486965fd649496e3ef9fe23be-3
48x348-png
Uploading image: https://template-03-api.vercel.app/products/4.png
Image uploaded successfully: image-611df72a5b65db2f78d3119889a742c134480265-3
48x348-png
Uploading image: https://template-03-api.vercel.app/products/5.png
Image uploaded successfully: image-46998b7aa7bbf7d305755b2964da5bf167434d40-3
48x348-png
Uploading image: https://template-03-api.vercel.app/products/6.png
Image uploaded successfully: image-0618b5440becd53aced91c6de2bd3c84757be8d1-3
48x348-png
Uploading image: https://template-03-api.vercel.app/products/7.png
Image uploaded successfully: image-b039b2d23e832f1d29375236a80f613bc011dbc3-3
48x348-png
Data migrated successfully!
```

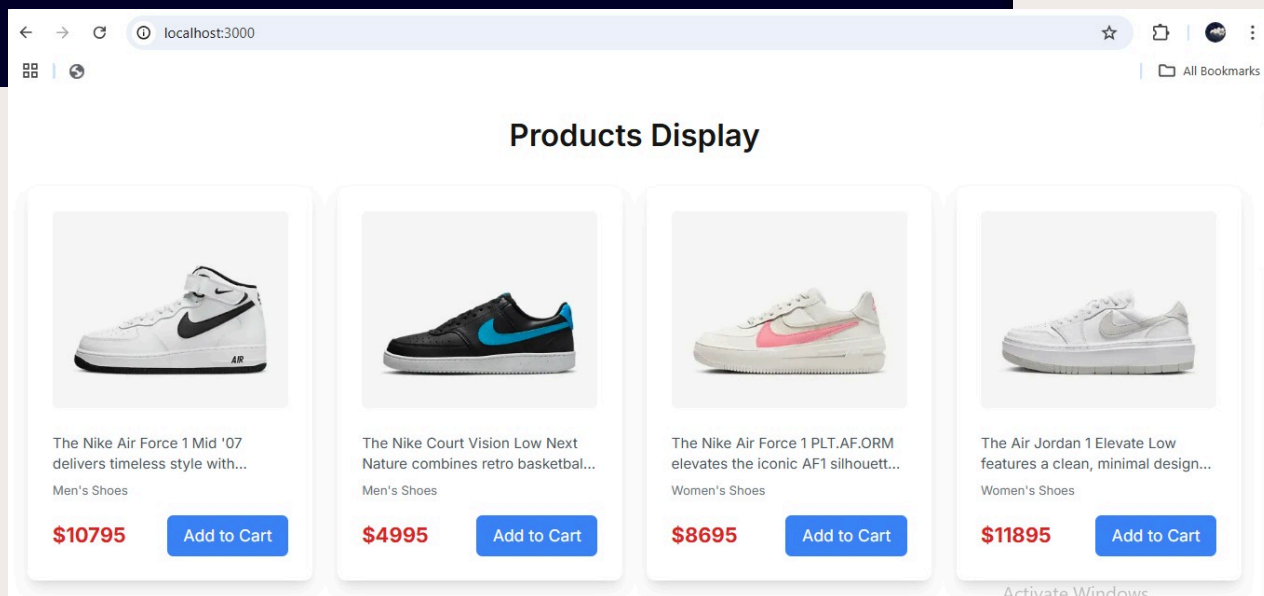


```
13
14 ✓ interface Product {
15   _id: string;
16   name: string;
17   description: string;
18   category: string;
19   price: number;
20 ✓ image: {
21   ✓   asset: {
22     url: string;
23   };
24 }
```

## 5. Data Successfully Displayed on Frontend

- screenshot showing the product data being displayed on the frontend correctly.

```
src > app > components > Sanity > ProductCard.tsx > ...
17  ard: React.FC<{ product: Product }> = ({ product }) => {
33
34
35  uct Name */}
36  sName="text-lg font-semibold □ text-gray-900 truncate">{product.name}</h3
37
38  uct Description */}
39  Name="□ text-gray-600 text-sm mt-2 line-clamp-2">{product.description}</p
40
41  uct Category */}
42  assName="block □ text-gray-500 text-xs mt-2">{product.category}</span>
43
44  e and Button */}
```



## 6. Populated Sanity CMS Fields

- screenshots of the populated fields in the Sanity dashboard (Product, Image, Price, etc.).

