Hoor Furniture E-Commerce Website
Online Furniture Market Business

Introduction

This document outlines the steps and process used to set up an e-commerce furniture website using **Next.js** and **Sanity**. The process includes schema creation, API integration, data fetching into Sanity, and displaying the products on the shop page of the website.

Step 1: Project Setup

1. Install and configure **Sanity** in the project.
2. Set up the basic structure of the e-commerce website using **Next.js**.
3. Prepare the project environment to work with **Sanity** as the backend for product data.

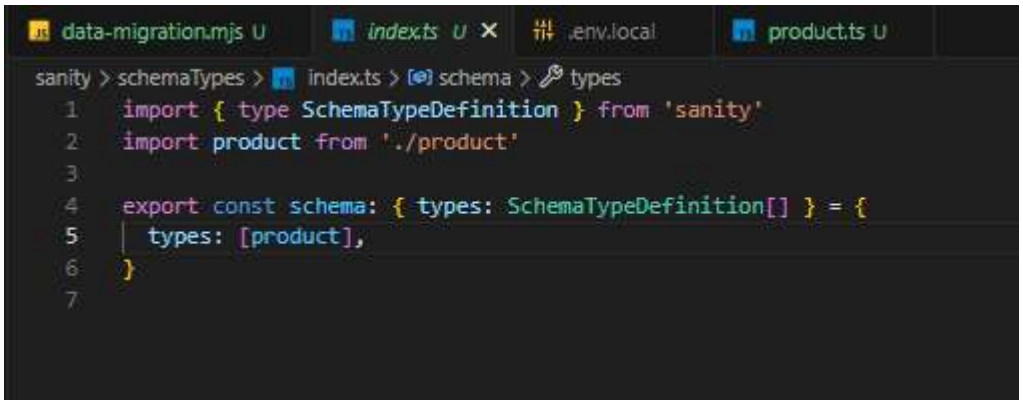
```
? Create a new project or select an existing one Create new project
? Your project name: marketplace
Your content will be stored in a dataset that can be public or private, depending on
whether you want to query your content with or without authentication.
The default dataset configuration has a public dataset named "production".
? Use the default dataset configuration? Yes
⠋ Creating dataset
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'
npm warn deprecated @sanity/block-tools@3.70.0: Renamed - use `@portabletext/block-tools` instead. `@sanity/block-tools`
 will no longer receive updates.

added 843 packages, changed 4 packages, and audited 1363 packages in 4m

265 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix --force
```

Step 2: Schema Definition

1. Create a product schema for the e-commerce website in **Sanity**.
2. The schema should define the structure of the product data such as title, description, price, image, etc.
3. The schema is defined in a file named **product.ts**.
4. This schema allows for easy management and updating of product information.

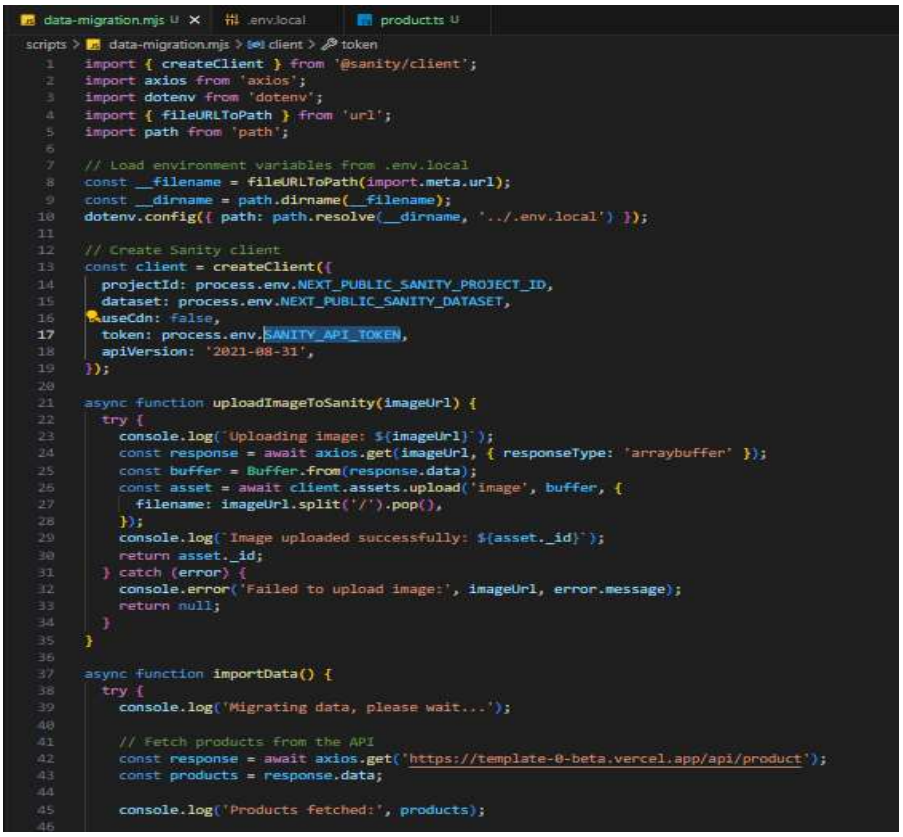
Step 3: Importing the schema

1. Import the product schema into the main application file **index.ts**.
2. Ensure that the project can interact with the **Sanity** backend by correctly importing the schema.



Step 4: Data fetching script

1. Create a **data-migration.mjs** script in the **scripts/** folder.
2. The script will fetch product data from an external API and migrate it to **Sanity**.
3. This script helps to automate the process of getting product data into the **Sanity** backend.



```
    for (const product of products) {
      let imageRef = null;

      if (product.imagePath) {
        imageRef = await uploadImageToSanity(product.imagePath);
      }

      const sanityProduct = {
        _type: 'product',
        _id: product.id,
        name: product.name,
        category: product.category,
        description: product.description,
        discountPercentage: product.discountPercentage,
        isFeaturedProduct: product.isFeaturedProduct,
        stockLevel: product.stockLevel,
        price: parseFloat(product.price),
        image: imageRef
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined,
        imagePath: product.imagePath, // Store original image URL
      };

      await client.create(sanityProduct);
      console.log(`Product created in Sanity: ${sanityProduct.id}`);
    }

    console.log('Data migrated successfully!');
  } catch (error) {
    console.error('Error in migrating data:', error.message);
  }
}

importData();
```

Step 5: Package Configuration

1. Modify the **package.json** file to add a custom script for running the data migration script.
2. The script is added under the **"scripts"** section:

```
Debug
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "Data": "node scripts/data-migration.mjs"
},
```

3. This allows you to run the data migration with the command npm run Data.

Step 6: Running the data migration command

1. Run the following command in the terminal to trigger the data migration:

```
npm run Data
```

2. The data is fetched from the API and migrated into **Sanity**, making it available in the backend.

Step 7: Displaying products on the shop page

1. The products that were added to **Sanity** are now displayed dynamically on the **shop page** of the website.
2. The product data is fetched from **Sanity** and shown on the front end using **Next.js**.
3. This integration ensures that the shop page is automatically updated whenever the product data is modified in **Sanity**.