Project Documentation:

Login page: [scr/pages/login/Login.tsx]

```
import React, { useState, FormEvent } from 'react';
import { useRouter } from 'next/router';
import styles from '@/styles/Home.module.css'; // Importing CSS module for styling
import Button from '@mui/material/Button';
import TextField from '@mui/material/TextField';

function Login() {
  // State variables for username, password, and error message
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const router = useRouter(); // Next.js router for navigation

  // Handler function for form submission
  const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
    debugger; // Breakpoint for debugging
    e.preventDefault(); // Prevent the default form submission behavior

    try {
      // Send a POST request to the /api/login endpoint with the username and password
      const response = await fetch('/api/login', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ username, password }), // Send username and password in the request body
      });

      // If the response is not ok, throw an error
      if (!response.ok) {
        throw new Error('Login failed');
      }

      // Parse the response data as JSON
      const data = await response.json();
      // Navigate to the dashboard page upon successful login
      router.push('/dashboard');
    } catch (error) {
      // If there is an error, set the error message state
      setError('Invalid username or password');
      console.error(error); // Log the error to the console
    }
  };
```

```jsx
  return (
    <div className={styles.Logincontainer}> {/* Container div with a custom class for styling */}
      <h1>Login</h1>
      {error && <p style={{ color: 'red' }}>{error}</p>} {/* Display error message if there is any */}

      <form onSubmit={handleSubmit}> {/* Form element with onSubmit handler */}
        <div className={styles.FormInput}> {/* Input container div with a custom class for styling */}
          <TextField
            type="text"
            id="username"
            value={username}
            label="Username"
            variant="outlined"
            className={styles.MuiFormControlroot}             // {/* Custom class for styling MUI TextField */}
            onChange={(e) => setUsername(e.target.value)}     //  {/* Update username state on change */}
            required
          />
        </div>
        <div className={styles.FormInput}> {/* Input container div with a custom class for styling */}
          <TextField
            type="password"
            id="password"
            value={password}
            label="Password"
            variant="outlined"
            className={styles.MuiFormControlroot}             // {/* Custom class for styling MUI TextField */}
            onChange={(e) => setPassword(e.target.value)}       // {/* Update password state on change */}
            required
          />
        </div>
        <Button type="submit" variant="contained"> {/* MUI Button for form submission */}
          Login
        </Button>
      </form>
    </div>
  );
}

export default Login;
```

Products Page: [scr/pages/dashboard/Dashboard.tsx]

```tsx
import React, { useEffect, useState } from 'react';
import { DataGrid, GridColDef } from '@mui/x-data-grid';
import Sidenav from '../components/Sidenav/Sidenav'; // Importing the Sidenav component
import styles from '@/styles/Home.module.css'; // Importing CSS module for styling
import { Product } from './types'; // Importing the Product type
import { Dialog, DialogActions, DialogContent, DialogTitle, Button, Typography, CircularProgress } from '@mui/material'; // Importing MUI
components

const Dashboard: React.FC = () => {
  // State variables for products, selected product, dialog open state, and loading state
  const [products, setProducts] = useState<Product[]>([]);
  const [selectedProduct, setSelectedProduct] = useState<Product | null>(null);
  const [isDialogOpen, setIsDialogOpen] = useState(false);
  const [loading, setLoading] = useState(false);

  // Column definitions for the DataGrid
  const columns: GridColDef[] = [
    { field: 'id', headerName: 'ID', width: 90 },
    { field: 'title', headerName: 'Title', width: 150 },
    { field: 'description', headerName: 'Description', width: 250 },
    { field: 'category', headerName: 'Category', width: 150 },
    { field: 'price', headerName: 'Price', width: 120 },
    { field: 'discountPercentage', headerName: 'Discount %', width: 150 },
    { field: 'rating', headerName: 'Rating', width: 120 },
    { field: 'stock', headerName: 'Stock', width: 120 },
    { field: 'tags', headerName: 'Tags', width: 180 },
    { field: 'brand', headerName: 'Brand', width: 150 },
    {
      field: 'actions',
      headerName: 'Actions',
      width: 150,
      renderCell: (params) => (
        <div className={styles.fixedColumn}>
          <Button variant="contained" color="primary" onClick={() => handleViewReviews(params.row.id)}>
            View Reviews
          </Button>
        </div>
      ),
    },
  ];
```

```
// useEffect hook to fetch products when the component mounts
 useEffect(() => {
   async function fetchProducts() {
     try {
       const response = await fetch('https://dummyjson.com/products');
       const data = await response.json();
       setProducts(data.products); // Set the fetched products to the state
     } catch (error) {console.error('Error fetching products:', error);  }    }
   fetchProducts(); // Call the fetchProducts function
 }, []);
 // Function to handle viewing reviews of a product
 const handleViewReviews = async (productId: number) => {
   setLoading(true); // Set loading state to true
   setIsDialogOpen(true); // Open the dialog
   try {
     const response = await fetch(`https://dummyjson.com/products/${productId}`);
     const product = await response.json();
     setSelectedProduct(product); // Set the selected product to the state
   } catch (error) {
     console.error('Error fetching product details:', error);
   } finally {
     setLoading(false); // Set loading state to false
   } };
 // Function to handle closing the dialog
 const handleCloseDialog = () => {
   setIsDialogOpen(false); // Close the dialog
   setSelectedProduct(null); // Clear the selected product
 };
 return (
   <>
     <Sidenav /> {/* Render the Sidenav component */}
     <div className={styles.MainBody}> {/* Container div with a custom class for styling */}
       <div className={styles.MainBodyContainer}> {/* Inner container div with a custom class for styling */}
         <div style={{ height: 600, width: '100%' }}> {/* Container for the DataGrid with fixed height and width */}
           <h2 className={styles.TableTitle}>Products</h2> {/* Table title */}
           <DataGrid rows={products} columns={columns} /> {/* DataGrid with products and columns */}
         </div>
       </div>
     </div>
     <Dialog open={isDialogOpen} onClose={handleCloseDialog}> {/* Dialog for viewing product reviews */}
       <DialogTitle>Product Reviews</DialogTitle>
       <DialogContent>
         {loading ? (
           <CircularProgress /> // Show loading spinner if loading state is true
         ) : selectedProduct ? ( <>
             <Typography variant="h6">{selectedProduct.title}</Typography> {/* Display product title */}
             <Typography variant="body1">{selectedProduct.description}</Typography> {/* Display product description */}
           </> ) : (
           <Typography>No product selected</Typography> // Show message if no product is selected
         )}
       </DialogContent>
       <DialogActions> <Button onClick={handleCloseDialog} color="primary"> {/* Close button */} Close</Button>
       </DialogActions>  </Dialog>
   </>
 );
```

Run the project:

```
npm install


npm run dev
```