**CSS (styles.css)**

Basic styling for ProductDetail and EditProductForm.

css

Sao chép mã

.product-detail {

max-width: 600px;

margin: auto;

padding: 20px;

border: 1px solid #ddd;

border-radius: 8px;

text-align: center;

}

.product-title {

font-size: 24px;

margin-bottom: 10px;

}

.product-price {

color: #f60;

font-size: 18px;

margin-bottom: 10px;

}

.edit-button, .back-button {

display: inline-block;

margin: 10px;

padding: 10px 20px;

background-color: #007bff;

color: #fff;

text-decoration: none;

border-radius: 5px;

cursor: pointer;

}

.edit-product-form label {

display: block;

margin: 10px 0;

}

.edit-product-form input,

.edit-product-form textarea {

width: 100%;

padding: 8px;

margin-top: 5px;

box-sizing: border-box;

}

.save-button, .cancel-button {

margin: 5px;

padding: 10px 15px;

border: none;

border-radius: 5px;

cursor: pointer;

}

.save-button {

background-color: #28a745;

color: #fff;

}

.cancel-button {

background-color: #dc3545;

color: #fff;

}

**Updated Code**

**1. ProductDetail.js**

This updated version checks if an id is available from the URL. If not, it defaults to "add mode."

javascript

Sao chép mã

import { useState, useEffect } from "react";

import axios from "axios";

import { useParams } from 'react-router-dom';

import EditProductForm from "./EditProductForm";

const ProductDetail = () => {

const { id } = useParams();

const [product, setProduct] = useState({

id: id || 0, // Default to 0 for new items

title: "",

price: 0,

description: "",

category: "",

image: ""

});

const [isEditing, setIsEditing] = useState(!id); // If no id, we're in add mode

useEffect(() => {

if (id) {

// Only fetch if editing an existing product

const fetchProduct = async () => {

try {

const res = await axios.get(`https://fakestoreapi.com/products/${id}`);

setProduct(res.data);

} catch (err) {

console.log(err.message);

}

};

fetchProduct();

}

}, [id]);

const handleEditToggle = () => setIsEditing(!isEditing);

const handleSaveChanges = (updatedProduct) => {

setProduct(updatedProduct);

setIsEditing(false);

};

return (

<div className="product-detail">

{isEditing ? (

<EditProductForm

product={product}

onSave={handleSaveChanges}

onCancel={handleEditToggle}

isNew={!id} // Pass isNew prop to indicate add mode

/>

) : (

<>

<img src={product.image} alt={product.title} />

<h2 className="product-title">{product.title}</h2>

<p className="product-price">Giá: {product.price}</p>

<p className="product-description">{product.description}</p>

<button onClick={handleEditToggle} className="edit-button">Edit</button>

<a href="/" className="back-button">Quay lại Danh sách sản phẩm</a>

</>

)}

</div>

);

}

export default ProductDetail;

**2. EditProductForm.js (Updated)**

This component now includes logic for adding a new item. When isNew is true, it sends a POST request to create a new item instead of updating an existing one.

javascript

Sao chép mã

import { useState } from "react";

import axios from "axios";

const EditProductForm = ({ product, onSave, onCancel, isNew }) => {

const [editedProduct, setEditedProduct] = useState({ ...product });

const handleChange = (e) => {

const { name, value } = e.target;

setEditedProduct((prevProduct) => ({

...prevProduct,

[name]: value

}));

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

if (isNew) {

// If adding a new product, use POST

const response = await axios.post(`https://fakestoreapi.com/products`, editedProduct);

onSave(response.data); // Pass new product back to parent component

} else {

// If editing, use PUT

const response = await axios.put(`https://fakestoreapi.com/products/${product.id}`, editedProduct);

onSave(response.data); // Pass updated product back to parent component

}

} catch (err) {

console.log(err.message);

}

};

return (

<form onSubmit={handleSubmit} className="edit-product-form">

<label>Title:

<input type="text" name="title" value={editedProduct.title} onChange={handleChange} />

</label>

<label>Price:

<input type="number" name="price" value={editedProduct.price} onChange={handleChange} />

</label>

<label>Description:

<textarea name="description" value={editedProduct.description} onChange={handleChange} />

</label>

<label>Category:

<input type="text" name="category" value={editedProduct.category} onChange={handleChange} />

</label>

<label>Image URL:

<input type="text" name="image" value={editedProduct.image} onChange={handleChange} />

</label>

<button type="submit" className="save-button">Save</button>

<button type="button" onClick={onCancel} className="cancel-button">Cancel</button>

</form>

);

};

export default EditProductForm;

**3. App.js (or Parent Component with Routing)**

Ensure you have routes that differentiate between editing an existing product and adding a new one. For example:

javascript

Sao chép mã

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import ProductDetail from './ProductDetail';

function App() {

return (

<Router>

<Routes>

<Route path="/product/:id" element={<ProductDetail />} /> {/\* For editing \*/}

<Route path="/add-product" element={<ProductDetail />} /> {/\* For adding \*/}

</Routes>

</Router>

);

}

export default App;