**CODE :**

**Client**

**App.jsx**

import React, { useState } from 'react';

import ProductList from './components/ProductList';

import ProductForm from './components/ProductForm';

import ErrorBoundary from './components/ErrorBoundary.jsx';

function App() {

const [refresh, setRefresh] = useState(false);

return (

<div className="App">

<ErrorBoundary>

<header className="header">

<h1>Product Management</h1>

</header>

<div className="form-container">

<h2>Add New Product</h2>

<ProductForm onSuccess={() => setRefresh(!refresh)} />

</div>

<ErrorBoundary>

<div className="product-list">

<ProductList />

</div>

</ErrorBoundary>

</ErrorBoundary>

</div>

);

}

export default App;

**ProductForm**

import React, { useState } from 'react';

import { createProduct, updateProduct } from '../services/productService.jsx';

const ProductForm = ({ product, onSuccess }) => {

const [formData, setFormData] = useState({

name: product?.name || '',

price: product?.price || '',

description: product?.description || ''

});

const handleChange = (e) => {

setFormData({

...formData,

[e.target.name]: e.target.value

});

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

if (product) {

await updateProduct(product.id, formData);

} else {

await createProduct(formData);

}

onSuccess();

} catch (error) {

console.error('Error saving product:', error);

}

};

return (

<form onSubmit={handleSubmit}>

<div>

<label>Name:</label>

<input

type="text"

name="name"

value={formData.name}

onChange={handleChange}

required

/>

</div>

<div>

<label>Price:</label>

<input

type="number"

name="price"

value={formData.price}

onChange={handleChange}

step="0.01"

required

/>

</div>

<div>

<label>Description:</label>

<textarea

name="description"

value={formData.description}

onChange={handleChange}

/>

</div>

<button type="submit">Save</button>

</form>

);

};

export default ProductForm;

**ProductList**

import React, { useState, useEffect } from 'react';

import { getProducts, deleteProduct } from '../services/productService';

// import { deleteProduct, getProducts } from '../services/productService';

const ProductList = () => {

const [products, setProducts] = useState([]);

const [loading, setLoading] = useState(true);

const [error, setError] = useState('');

useEffect(() => {

fetchProducts();

}, []);

const fetchProducts = async () => {

try {

const response = await getProducts();

// Ensure we're working with an array

const productsData = Array.isArray(response?.data?.data)

? response.data.data

: Array.isArray(response?.data)

? response.data

: [];

setProducts(productsData);

setLoading(false);

} catch (err) {

setError('Failed to fetch products');

setLoading(false);

setProducts([]); // Ensure products is always an array

}

};

const handleDelete = async (id) => {

try {

await deleteProduct(id); // Make sure this matches your import

fetchProducts();

} catch (err) {

setError('Failed to delete product');

}

};

if (loading) return <div>Loading...</div>;

if (error) return <div className="error-message">{error}</div>;

return (

<>

<h2>Products</h2>

{products.length === 0 ? (

<p>No products found</p>

) : (

<ul style={{ listStyle: 'none' }}>

{products.map(product => (

<li key={product.\_id} className="product-item">

<div className="product-info">

<div className="product-name">{product.name}</div>

<div className="product-price">${product.price}</div>

{product.description && <p>{product.description}</p>}

</div>

<div className="product-actions">

<button

className="btn-delete"

onClick={() => handleDelete(product.\_id)}

>

Delete

</button>

</div>

</li>

))}

</ul>

)}

</>

);

};

export default ProductList;

**productService**

import axios from 'axios';

const API\_URL = 'http://localhost:5000/api/products';

// Get all products

export const getProducts = async () => {

try {

const response = await axios.get(API\_URL);

return response.data;

} catch (error) {

console.error('Error fetching products:', error);

throw error;

}

};

// Get single product

export const getProduct = async (id) => {

try {

const response = await axios.get(`${API\_URL}/${id}`);

return response.data;

} catch (error) {

console.error(`Error fetching product ${id}:`, error);

throw error;

}

};

// Create new product

export const createProduct = async (productData) => {

try {

const response = await axios.post(API\_URL, productData);

return response.data;

} catch (error) {

console.error('Error creating product:', error);

throw error;

}

};

// Update product

export const updateProduct = async (id, productData) => {

try {

const response = await axios.put(`${API\_URL}/${id}`, productData);

return response.data;

} catch (error) {

console.error(`Error updating product ${id}:`, error);

throw error;

}

};

// Delete product - MAKE SURE THIS EXISTS AND IS EXPORTED

export const deleteProduct = async (id) => {

try {

const response = await axios.delete(`${API\_URL}/${id}`);

return response.data;

} catch (error) {

console.error(`Error deleting product ${id}:`, error);

throw error;

}

};

// Export all functions as named exports

export default {

getProducts,

getProduct,

createProduct,

updateProduct,

deleteProduct

};

**SERVER**

**Config.db**

const mongoose = require('mongoose');

const connectDB = async () => {

try {

const conn = await mongoose.connect(process.env.MONGODB\_URI || 'mongodb://127.0.0.1:27017/crudDB', {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log(`MongoDB Connected: ${conn.connection.host}`);

} catch (error) {

console.error(`Error: ${error.message}`);

process.exit(1);

}

};

module.exports = connectDB;

**Index**

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const productRoutes = require('./routes/productRoutes');

const connectDB = require('./config/db');

const app = express();

// Connect to Database

connectDB();

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Routes

app.use('/api/products', productRoutes);

// Error handling middleware

app.use((err, req, res, next) => {

console.error(err.stack);

res.status(500).json({

success: false,

error: 'Server Error'

});

});

module.exports = app;

**Model**

const mongoose = require('mongoose');

const ProductSchema = new mongoose.Schema({

name: {

type: String,

required: [true, 'Please add a name'],

trim: true,

maxlength: [50, 'Name cannot be more than 50 characters']

},

price: {

type: Number,

required: [true, 'Please add a price'],

min: [0, 'Price must be at least 0']

},

description: {

type: String,

maxlength: [500, 'Description cannot be more than 500 characters']

},

createdAt: {

type: Date,

default: Date.now

}

});

module.exports = mongoose.model('Product', ProductSchema);

**Controller**

// Use either ES module exports (if you're using type: module in package.json)

// Or CommonJS exports (standard Node.js)

// Option 1: CommonJS exports (recommended for Node.js)

const Product = require('../models/Product');

// Get all products

const getProducts = async (req, res) => {

try {

const products = await Product.find();

res.status(200).json({

success: true,

count: products.length,

data: products

});

} catch (err) {

res.status(500).json({

success: false,

error: 'Server Error'

});

}

};

// Get single product

const getProduct = async (req, res) => {

try {

const product = await Product.findById(req.params.id);

if (!product) {

return res.status(404).json({

success: false,

error: 'No product found'

});

}

res.status(200).json({

success: true,

data: product

});

} catch (err) {

res.status(500).json({

success: false,

error: 'Server Error'

});

}

};

// Export as an object

module.exports = {

getProducts,

getProduct,

createProduct: async (req, res) => {

try {

const product = await Product.create(req.body);

res.status(201).json({

success: true,

data: product

});

} catch (err) {

if (err.name === 'ValidationError') {

const messages = Object.values(err.errors).map(val => val.message);

return res.status(400).json({

success: false,

error: messages

});

} else {

res.status(500).json({

success: false,

error: 'Server Error'

});

}

}

},

updateProduct: async (req, res) => {

try {

const product = await Product.findByIdAndUpdate(req.params.id, req.body, {

new: true,

runValidators: true

});

if (!product) {

return res.status(404).json({

success: false,

error: 'No product found'

});

}

res.status(200).json({

success: true,

data: product

});

} catch (err) {

if (err.name === 'ValidationError') {

const messages = Object.values(err.errors).map(val => val.message);

return res.status(400).json({

success: false,

error: messages

});

} else {

res.status(500).json({

success: false,

error: 'Server Error'

});

}

}

},

deleteProduct: async (req, res) => {

try {

const product = await Product.findByIdAndDelete(req.params.id);

if (!product) {

return res.status(404).json({

success: false,

error: 'No product found'

});

}

res.status(200).json({

success: true,

data: {}

});

} catch (err) {

res.status(500).json({

success: false,

error: 'Server Error'

});

}

}

};

**Route**

const express = require('express');

const router = express.Router();

const {

getProducts,

getProduct,

createProduct,

updateProduct,

deleteProduct

} = require('../controllers/productController');

// Properly attach the functions to the routes

router.get('/', getProducts);

router.get('/:id', getProduct);

router.post('/', createProduct);

router.put('/:id', updateProduct);

router.delete('/:id', deleteProduct);

module.exports = router;

**App**

const app = require('./app');

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

**OUTPUT**

