**-----------------Dynamic Product Level adding specifications -----------------**

**Xampp MySql - DB**

CREATE TABLE categories ( id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL );

CREATE TABLE subcategories ( id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL, category\_id INT NOT NULL, FOREIGN KEY (category\_id) REFERENCES categories(id) );

CREATE TABLE products ( id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL, subcategory\_id INT NOT NULL, FOREIGN KEY (subcategory\_id) REFERENCES subcategories(id) );

CREATE TABLE specifications ( id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, product\_id INT NOT NULL, FOREIGN KEY (product\_id) REFERENCES products(id) );

CREATE TABLE selections ( id INT AUTO\_INCREMENT PRIMARY KEY, product\_id INT NOT NULL, specs\_json TEXT NOT NULL, FOREIGN KEY (product\_id) REFERENCES products(id) );

**Node.js - Backend API’s**

const express = require('express'); const mysql = require('mysql2'); const cors = require('cors');

const app = express(); app.use(cors()); app.use(express.json());

const db = mysql.createConnection({ host: 'localhost', user: 'root', password: '', // Set your MySQL password if any database: 'sbs\_crm\_specs' });

db.connect(err => { if (err) throw err; console.log('MySQL Connected'); });

// GET /categories - Get main categories app.get('/categories', (req, res) => { db.query('SELECT \* FROM categories', (err, results) => { if (err) throw err; res.json(results); }); });

// GET /subcategories/:categoryId app.get('/subcategories/:categoryId', (req, res) => { const categoryId = req.params.categoryId; db.query('SELECT \* FROM subcategories WHERE category\_id = ?', [categoryId], (err, results) => { if (err) throw err; res.json(results); }); });

// GET /products/:subcategoryId app.get('/products/:subcategoryId', (req, res) => { const subcategoryId = req.params.subcategoryId; db.query('SELECT \* FROM products WHERE subcategory\_id = ?', [subcategoryId], (err, results) => { if (err) throw err; res.json(results); }); });

// GET /specs/:productId app.get('/specs/:productId', (req, res) => { const productId = req.params.productId; db.query('SELECT \* FROM specifications WHERE product\_id = ?', [productId], (err, results) => { if (err) throw err; res.json(results); }); });

// POST /specs - Add new specification app.post('/specs', (req, res) => { const { product\_id, name } = req.body; if (!product\_id || !name) return res.status(400).json({ error: 'Missing fields' }); db.query('INSERT INTO specifications (name, product\_id) VALUES (?, ?)', [name, product\_id], (err, results) => { if (err) throw err; res.json({ id: results.insertId }); }); });

// POST /selections - Save configuration app.post('/selections', (req, res) => { const { product\_id, specs\_json } = req.body; if (!product\_id || !specs\_json) return res.status(400).json({ error: 'Missing fields' }); db.query('INSERT INTO selections (product\_id, specs\_json) VALUES (?, ?)', [product\_id, JSON.stringify(specs\_json)], (err, results) => { if (err) throw err; res.json({ id: results.insertId }); }); });

// GET /selections/:productId - Get saved selections app.get('/selections/:productId', (req, res) => { const productId = req.params.productId; db.query('SELECT \* FROM selections WHERE product\_id = ?', [productId], (err, results) => { if (err) throw err; res.json(results.map(r => ({ ...r, specs\_json: JSON.parse(r.specs\_json) }))); }); });

app.listen(5000, () => { console.log('Server running on port 5000'); });

**React – Frontend**

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

function App() { const [categories, setCategories] = useState([]); const [selectedCategory, setSelectedCategory] = useState(''); const [subcategories, setSubcategories] = useState([]); const [selectedSubcategory, setSelectedSubcategory] = useState(''); const [products, setProducts] = useState([]); const [selectedProduct, setSelectedProduct] = useState(''); const [specs, setSpecs] = useState([]); const [specValues, setSpecValues] = useState({}); const [newSpec, setNewSpec] = useState(''); const [savedSelections, setSavedSelections] = useState([]);

useEffect(() => { axios.get('http://localhost:5000/categories') .then(res => setCategories(res.data)) .catch(err => console.error(err)); }, []);

useEffect(() => { if (selectedCategory) { axios.get(`http://localhost:5000/subcategories/${selectedCategory}`) .then(res => { setSubcategories(res.data); setSelectedSubcategory(''); setProducts([]); setSelectedProduct(''); setSpecs([]); setSpecValues({}); setSavedSelections([]); }) .catch(err => console.error(err)); } }, [selectedCategory]);

useEffect(() => { if (selectedSubcategory) { axios.get(`http://localhost:5000/products/${selectedSubcategory}`) .then(res => { setProducts(res.data); setSelectedProduct(''); setSpecs([]); setSpecValues({}); setSavedSelections([]); }) .catch(err => console.error(err)); } }, [selectedSubcategory]);

const fetchSpecsAndSelections = () => { if (selectedProduct) { axios.get(`http://localhost:5000/specs/${selectedProduct}`) .then(res => setSpecs(res.data)) .catch(err => console.error(err));

axios.get(`http://localhost:5000/selections/${selectedProduct}`) .then(res => setSavedSelections(res.data)) .catch(err => console.error(err));

setSpecValues({}); } };

useEffect(() => { fetchSpecsAndSelections(); }, [selectedProduct]);

const handleSpecChange = (name, value) => { setSpecValues(prev => ({ ...prev, [name]: value })); };

const handleSave = () => { axios.post('http://localhost:5000/selections', { product\_id: selectedProduct, specs\_json: specValues }) .then(() => { alert('Saved successfully!'); fetchSpecsAndSelections(); // Refresh saved list setSpecValues({}); // Clear form }) .catch(err => console.error(err)); };

const handleAddSpec = () => { if (!newSpec) return; axios.post('http://localhost:5000/specs', { product\_id: selectedProduct, name: newSpec }) .then(() => { setNewSpec(''); fetchSpecsAndSelections(); // Refresh specs }) .catch(err => console.error(err)); };

return (

# Product Specifications Selector

Select Category:

setSelectedCategory(e.target.value)}>

--Choose--

{categories.map(cat => (

{cat.name}

))}

{selectedCategory && (

Select Subcategory:

setSelectedSubcategory(e.target.value)}>

--Choose--

{subcategories.map(sub => (

{sub.name}

))}

)}

{selectedSubcategory && (

Select Product:

setSelectedProduct(e.target.value)}>

--Choose--

{products.map(prod => (

{prod.name}

))}

)}

{selectedProduct && specs.length > 0 && (

## **Enter Specifications:**

{specs.map(spec => (

{spec.name}:

handleSpecChange(spec.name, e.target.value)} />

))} Save Configuration

)}

{selectedProduct && (

## **Add New Specification:**

setNewSpec(e.target.value)} placeholder="New spec name" /> Add

)}

{selectedProduct && savedSelections.length > 0 && (

## **Saved Configurations:**

{savedSelections.map(sel => (

**ID: {sel.id}**

{Object.entries(sel.specs\_json).map(([key, value]) => (

* {key}: {value}

))}

))}

)}

); }

export default App;