**Objective:-** Backend: Using NodeJS, build a job search platform or e-commerce platform api to search and filter.

**Code :-**

**ecom-client :-**

**App.jsx**

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

import Home from "./pages/Home";

import ProductDetails from "./pages/ProductDetails";

function App() {

  return (

    <Router>

      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="/product/:id" element={<ProductDetails />} />

      </Routes>

    </Router>

  );

}

export default App;

**index.css**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');

@tailwind base;

@tailwind components;

@tailwind utilities;

body {

  font-family: 'Poppins', sans-serif;

}

**src/pages/Home.jsx :-**

import { useEffect, useState } from "react";

import axios from "axios";

import { Link, useSearchParams } from "react-router-dom";

function Home() {

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

  // Display products in table rows

// Remove unused productRows mapping since the products are mapped directly in the table

  const [searchParams, setSearchParams] = useSearchParams();

  const [filters, setFilters] = useState({

    name: searchParams.get("name") || "",

    brand: searchParams.get("brand") || "",

    minPrice: searchParams.get("minPrice") || "",

    maxPrice: searchParams.get("maxPrice") || "" ,

    category: searchParams.get("category") || "" ,

    tags: searchParams.get("tags") || "" ,

  });

  useEffect(() => {

    const queryParams = new URLSearchParams();

    if (filters.name) queryParams.append("name", filters.name);

    if (filters.brand) queryParams.append("brand", filters.brand);

    if (filters.category) queryParams.append("category", filters.category);

    if (filters.minPrice) queryParams.append("minPrice", filters.minPrice);

    if (filters.maxPrice) queryParams.append("maxPrice", filters.maxPrice);

    if (filters.tags) queryParams.append("tags", filters.tags);

    const queryString = queryParams.toString();

    const url = `http://localhost:3000/api/products${queryString ? `?${queryString}` : ''}`;

    axios.get(url)

      .then(res => setProducts(res.data))

      .catch(err => console.error(err));

    setSearchParams(queryParams);

  }, [filters, setSearchParams]);

  const handleFilterChange = (e) => {

    setFilters({

      ...filters,

      [e.target.name]: e.target.value

    });

  };

  return (

    <div className="p-8 bg-gray-100 min-h-screen">

      <h1 className="text-4xl font-bold mb-10 text-center text-green-600">All Products</h1>

      {/\* Filter Form \*/}

      <div className="mb-8 bg-white p-6 rounded-xl shadow-md">

        <h2 className="text-xl font-semibold mb-4 text-green-600">Filter Products</h2>

        <div className="grid grid-cols-1 md:grid-cols-4 gap-4">

          <div>

            <label className="block text-sm font-medium text-gray-700 mb-1">Name</label>

            <input

              type="text"

              name="name"

              value={filters.name}

              onChange={handleFilterChange}

              className="w-full p-2 border border-gray-300 rounded-md focus:ring-green-500 focus:border-green-500"

              placeholder="Filter by name"

            />

          </div>

          <div>

            <label className="block text-sm font-medium text-gray-700 mb-1">Brand</label>

            <input

              type="text"

              name="brand"

              value={filters.brand}

              onChange={handleFilterChange}

              className="w-full p-2 border border-gray-300 rounded-md focus:ring-green-500 focus:border-green-500"

              placeholder="Filter by brand"

            />

          </div>

          <div>

            <label className="block text-sm font-medium text-gray-700 mb-1">Min Price</label>

            <input

              type="number"

              name="minPrice"

              value={filters.minPrice}

              onChange={handleFilterChange}

              className="w-full p-2 border border-gray-300 rounded-md focus:ring-green-500 focus:border-green-500"

              placeholder="Min price"

            />

          </div>

          <div>

            <label className="block text-sm font-medium text-gray-700 mb-1">Max Price</label>

            <input

              type="number"

              name="maxPrice"

              value={filters.maxPrice}

              onChange={handleFilterChange}

              className="w-full p-2 border border-gray-300 rounded-md focus:ring-green-500 focus:border-green-500"

              placeholder="Max price"

            />

          </div>

        </div>

      </div>

      {/\* Rest of your table code remains the same \*/}

      <div className="overflow-x-auto rounded-xl shadow-lg">

      <table className="min-w-full text-base border-collapse bg-white">

          <thead className="bg-green-500 text-white">

            <tr>

              <th className="py-4 px-6 text-center font-semibold uppercase tracking-wider">Name</th>

              <th className="py-4 px-6 text-center font-semibold uppercase tracking-wider">Brand</th>

              <th className="py-4 px-6 text-center font-semibold uppercase tracking-wider">Price</th>

              <th className="py-4 px-6 text-center font-semibold uppercase tracking-wider">Actions</th>

            </tr>

          </thead>

          <tbody className="divide-y divide-gray-200">

            {products.map((product, index) => (

              <tr

                key={product.id}

                className={`${index % 2 === 0 ? 'bg-white' : 'bg-gray-50'} hover:bg-gray-100 transition-colors duration-200`}

              >

                <td className="py-4 px-6 text-center font-medium">{product.name}</td>

                <td className="py-4 px-6 text-center">{product.brand}</td>

                <td className="py-4 px-6 text-center text-green-600 font-semibold">₹ {product.price}</td>

                <td className="py-4 px-6 text-center">

                  <Link

                    to={`/product/${product.id}`}

                    className="inline-block px-4 py-2 bg-green-600 text-white rounded-md text-sm font-medium hover:bg-green-700 transition-colors duration-200"

                  >

                    View Details

                  </Link>

                </td>

              </tr>

            ))}

            {products.length === 0 && (

              <tr>

                <td colSpan="4" className="text-center text-gray-500 py-10">No products found.</td>

              </tr>

            )}

          </tbody>

        </table>

      </div>

    </div>

  );

}

export default Home;

**src/pages/ProductDetails.jsx :-**

import { useEffect, useState } from "react";

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

import axios from "axios";

function ProductDetails() {

  const { id } = useParams();

  const [product, setProduct] = useState(null);

  useEffect(() => {

    axios

      .get(`http://localhost:3000/api/products/${id}`)

      .then((res) => setProduct(res.data))

      .catch((err) => console.error(err));

  }, [id]);

  if (!product) return <div className="p-6 text-center text-lg">Loading...</div>;

  return (

    <div className="p-6 bg-gradient-to-br from-gray-100 to-white min-h-screen">

      <h1 className="text-4xl font-extrabold mb-10 text-center text-indigo-700">📦 Product Details</h1>

      <div className="max-w-2xl mx-auto bg-white rounded-2xl shadow-xl overflow-hidden border border-gray-200">

        <table className="w-full table-auto">

          <tbody

            <tr className="hover:bg-gray-50 transition">

              <td className="px-6 py-5 font-bold text-gray-700 w-1/3 bg-gray-100">📝 Product Name</td>

              <td className="px-6 py-5 text-gray-800">{product.name}</td>

            </tr>

            <tr className="hover:bg-gray-50 transition">

              <td className="px-6 py-5 font-bold text-gray-700 bg-gray-100">🏷️ Brand</td>

              <td className="px-6 py-5 text-gray-800">{product.brand}</td>

            </tr>

            <tr className="hover:bg-gray-50 transition">

              <td className="px-6 py-5 font-bold text-gray-700 bg-gray-100">💰 Price</td>

              <td className="px-6 py-5 text-green-600 font-semibold text-lg">₹ {product.price}</td>

            </tr>

            <tr className="hover:bg-gray-50 transition">

              <td className="px-6 py-5 font-bold text-gray-700 bg-gray-100">🧾 Description</td>

              <td className="px-6 py-5 text-gray-800">{product.description}</td>

            </tr>

          </tbody>

        </table>

      </div>

    </div>

  );

}

export default ProductDetails;

**ecom\_api**

**index.js**

require('dotenv').config();

const express = require('express');

const cors = require('cors');

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

// Database and Routes

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

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

const app = express();

// Middleware

app.use(cors());

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Routes

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

app.get('/', (req, res) => {

    res.send('Welcome to the Product API!');

  });

// DB sync and start server

sequelize.sync().then(() => {

    console.log('MySQL synced  and Connected Sucessfully');

    app.listen(process.env.PORT, () => {

        console.log(`Server running at http://localhost:${process.env.PORT}`);

    });

}).catch(err => {

    console.error('Unable to connect to DB:', err);

});

**config/db.js**

const { Sequelize } = require('sequelize');

require('dotenv').config();

const sequelize = new Sequelize(

    process.env.DB\_NAME,

    process.env.DB\_USER,

    process.env.DB\_PASSWORD,

    {

        host: process.env.DB\_HOST,

        dialect: 'mysql',

        port: process.env.DB\_PORT,

        logging: false,

    }

);

module.exports = sequelize;

**controllers/productController.js**

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

const { Op } = require('sequelize');

// Create a new product

exports.createProduct = async (req, res) => {

    try {

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

        res.status(201).json(product);

    } catch (err) {

        res.status(400).json({ error: err.message });

    }

};

// Get all products with optional filters

exports.getProducts = async (req, res) => {

    try {

        const { name, brand, category, minPrice, maxPrice, tags } = req.query;

        let filters = {};

        if (name) filters.name = { [Op.like]: `%${name}%` };

        if (brand) filters.brand = { [Op.like]: `%${brand}%` };

        if (category) filters.category = category;

        if (minPrice || maxPrice) filters.price = {};

        if (minPrice) filters.price[Op.gte] = parseFloat(minPrice);

        if (maxPrice) filters.price[Op.lte] = parseFloat(maxPrice);

        if (tags) {

            filters.tags = { [Op.like]: `%${tags}%` };

        }

        const products = await Product.findAll({ where: filters });

        res.json(products);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Get product by ID

exports.getProductById = async (req, res) => {

    try {

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

        if (!product) return res.status(404).json({ message: 'Product not found' });

        res.json(product);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

**models/Product.js**

const { DataTypes } = require('sequelize');

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

const Product = sequelize.define('Product', {

    name: {

        type: DataTypes.STRING,

        allowNull: false

    },

    brand: {

        type: DataTypes.STRING

    },

    category: {

        type: DataTypes.STRING

    },

    price: {

        type: DataTypes.FLOAT,

        allowNull: false

    },

    description: {

        type: DataTypes.TEXT

    },

    stock: {

        type: DataTypes.INTEGER,

        defaultValue: 0

    },

    tags: {

        type: DataTypes.STRING, // comma separated string (e.g., "tech,electronics")

    }

},{

    timestamps: false

  });

module.exports = Product;

**routes/productRoutes.js**

const express = require('express');

const router = express.Router();

const {

    createProduct,

    getProducts,

    getProductById

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

router.post('/', createProduct);

router.get('/', getProducts);

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

module.exports = router;

**.env**

DB\_HOST = "localhost"

DB\_USER = "root"

DB\_PASSWORD = ""

DB\_NAME = "ecommerce"

DB\_PORT = 3306

PORT = 3000

**Output :-**







