**Objective:-** Create modern, scalable and high-speed Web Applications with React and Node.js +Express

**Code :-**

**ecom-client :-**

**App.jsx**

import { useState } from "react";

import "./App.css";

import CategoryPage from "./components/CategoryPage";

import ProductPage from "./components/ProductPage";

function App() {

  const [showCategoryPage, setShowCategoryPage] = useState(false);

  const toggleCategoryPage = () => {

    setShowCategoryPage(!showCategoryPage);

  };

  return (

    <div>

      <button onClick={toggleCategoryPage}>

        {showCategoryPage ? "Show Products" : "Add Category"}

      </button>

      {showCategoryPage ? <CategoryPage /> : <ProductPage />}

    </div>

  );

}

export default App;

**App.css**

#root {

  max-width: 1280px;

  margin: 0 auto;

  padding: 2rem;

  text-align: center;

}

.logo {

  height: 6em;

  padding: 1.5em;

  will-change: filter;

  transition: filter 300ms;

}

.logo:hover {

  filter: drop-shadow(0 0 2em #646cffaa);

}

.logo.react:hover {

  filter: drop-shadow(0 0 2em #61dafbaa);

}

@keyframes logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

@media (prefers-reduced-motion: no-preference) {

  a:nth-of-type(2) .logo {

    animation: logo-spin infinite 20s linear;

  }

}

.card {

  padding: 2em;

}

.read-the-docs {

  color: #888;

}

**index.css**

:root {

  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;

  line-height: 1.5;

  font-weight: 400;

  color-scheme: light dark;

  color: rgba(255, 255, 255, 0.87);

  background-color: #242424;

  font-synthesis: none;

  text-rendering: optimizeLegibility;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

a {

  font-weight: 500;

  color: #646cff;

  text-decoration: inherit;

}

a:hover {

  color: #535bf2;

}

body {

  margin: 0;

  display: flex;

  place-items: center;

  min-width: 320px;

  min-height: 100vh;

}

h1 {

  font-size: 3.2em;

  line-height: 1.1;

}

button {

  border-radius: 8px;

  border: 1px solid transparent;

  padding: 0.6em 1.2em;

  font-size: 1em;

  font-weight: 500;

  font-family: inherit;

  background-color: #1a1a1a;

  cursor: pointer;

  transition: border-color 0.25s;

}

button:hover {

  border-color: #646cff;

}

button:focus,

button:focus-visible {

  outline: 4px auto -webkit-focus-ring-color;

}

@media (prefers-color-scheme: light) {

  :root {

    color: #213547;

    background-color: #ffffff;

  }

  a:hover {

    color: #747bff;

  }

  button {

    background-color: #f9f9f9;

  }

}

**src/components/CategoryPage.jsx :-**

import { useEffect, useState } from "react";

import axios from "axios";

import "../css/CategoryPage.css"

const CategoryPage = () => {

  const [categories, setCategories] = useState([]);

  const [data, setData] = useState({});

  useEffect(() => {

    fetchAllCategory();

  }, []);

  const fetchAllCategory = () => {

    axios.get("http://localhost:3000/category").then((resp) => {

      setCategories(resp.data.data);

    });

  };

  const handleChange = (e) => {

    // console.log(e.target.name, e.target.value);

    // let name = e.target.name;

    // let value = e.target.value;

    setData({ ...data, [e.target.name]: e.target.value });

  };

  const handleEdit = (id) => {

    console.log(id);

  };

  const handleDelete = (id) => {

    console.log(id);

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    axios

      .post("http://localhost:3000/category", data)

      .then((resp) => {

        console.log(resp);

        fetchAllCategory();

      })

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

    e.target.reset();

  };

  return (

    <>

      <h2>CategoryPage</h2>

      <form onSubmit={(e) => handleSubmit(e)} className="category-form">

        <input

          type="text"

          name="category\_name"

          placeholder="Enter Category Name"

          onChange={handleChange}

          className="input-field"

        />

        <input

          type="text"

          name="category\_description"

          placeholder="Enter Category Description"

          onChange={handleChange}

          className="input-field"

        />

        <input type="submit" name="btnSubmit" className="add-category-button" />

      </form>

      <div className="category-cards">

        {categories.map((category, index) => (

          <div className="category-card" key={index}>

            <h3>{category.category\_name}</h3>

            <p>{category.category\_description}</p>

            <p>{category.category\_status ? "Active" : "Inactive"}</p>

            <button onClick={() => handleEdit(category.id)}>Edit</button>

            <button onClick={() => handleDelete(category.id)}>Delete</button>

          </div>

        ))}

      </div>

    </>

  );

};

export default CategoryPage;

**src/css/CategoryPage.css :-**

.category-form {

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

    gap: 20px;

}

.category-form input {

    width: 250px;

    height: 35px;

    border: 1px solid #ccc;

    border-radius: 5px;

    padding: 10px;

    margin-bottom: 20px;

    margin-inline: 10px;

    box-sizing: border-box;

}

.add-category-button {

    background-color: #4CAF50;

    color: white;

    border: none;

    border-radius: 5px;

    padding: 10px;

    margin-inline: 10px;

    cursor: pointer;

    font-size: 16px;

}

.add-category-button:hover {

    background-color: #45a049;

}

.category-cards {

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

    gap: 20px;

}

.category-card {

    width: 250px;

    height: 175px;

    border: 1px solid #ccc;

    border-radius: 5px;

    overflow: hidden;

    padding: 10px;

    box-sizing: border-box;

}

.category-card h3,

.category-card p {

    margin: 0;

    margin-bottom: 10px;

    display: inline-block;

    overflow: hidden;

    white-space: nowrap;

    text-overflow: ellipsis;

    width: 100%;

}

.category-card button {

    background-color: #4CAF50;

    color: white;

    border: none;

    border-radius: 5px;

    padding: 5px 10px;

    cursor: pointer;

    font-size: 14px;

    margin-right: 10px;

}

.category-card button:hover {

    background-color: #45a049;

}

.category-card button:focus {

    outline: none;

}

**src/components/ProductPage.jsx :-**

import { useEffect, useState } from "react";

import axios from "axios";

import "../css/ProductPage.css"

import productimg from "../assets/product.jpeg"

const ProductPage = () => {

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

  const [categories, setCategories] = useState([]);

  const [selectedProductId, setSelectedProductId] = useState(null);

  const [data, setData] = useState({});

  useEffect(() => {

    axios.get("http://localhost:3000/product").then((resp) => {

      setProducts(resp.data.data);

    });

    fetchAllCategory();

  }, []);

  const fetchAllProduct = () => {

    axios.get("http://localhost:3000/product").then((resp) => {

      setProducts(resp.data.data);

    });

  };

  const fetchAllCategory = () => {

    axios.get("http://localhost:3000/category").then((resp) => {

      setCategories(resp.data.data);

    });

  };

  const handleChange = (e) => {

    setData({ ...data, [e.target.name]: e.target.value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (selectedProductId) {

      axios

        .put(`http://localhost:3000/product/${selectedProductId}`, data)

        .then((resp) => {

          console.log(resp.data);

          setSelectedProductId(null);

          setData({});

          fetchAllProduct();

        })

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

    } else {

      axios

        .post("http://localhost:3000/product", data)

        .then((resp) => {

          console.log(resp.data);

          setData({});

          fetchAllProduct();

        })

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

    }

    e.target.reset();

  };

  const handleEdit = (productId) => {

    setSelectedProductId(productId);

    const product = products.find((p) => p.id === productId);

    setData({

      product\_name: product.product\_name,

      product\_description: product.product\_description,

      price: product.price,

      categoryId: product.categoryId,

      product\_status: product.product\_status

    });

  };

  const handleDelete = (productId) => {

    axios

      .delete(`http://localhost:3000/product/del/${productId}`)

      .then((resp) => {

        console.log(resp.data);

        fetchAllProduct();

      })

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

  };

  return (

    <>

      <h2>ProductPage</h2>

      <form onSubmit={(e) => handleSubmit(e)}>

        <input

          className="input-field"

          type="text"

          name="product\_name"

          placeholder="Enter Product Name"

          onChange={handleChange}

          value={data.product\_name || ""}

        />

        <input

          className="input-field"

          type="text"

          name="product\_description"

          placeholder="Enter Product Description"

          onChange={handleChange}

          value={data.product\_description || ""}

        />

        <input

          className="input-field"

          type="number"

          name="price"

          placeholder="Enter Product Price"

          onChange={handleChange}

          value={data.price || ""}

        />

        <select className="input-field" name="categoryId" onChange={handleChange} value={data.categoryId || ""}>

          <option className="option-item" value="unselected">Select Category</option>

          {categories.map((category, index) => (

            <option className="option-item" key={index} value={category.id}>

              {category.category\_name}

            </option>

          ))}

        </select>

        {selectedProductId ? <select

          className="input-field"

          name="product\_status"

          onChange={handleChange}

          value={data.product\_status ? "true" : "false" || ""}

        >

          <option className="option-item" value="true">Available</option>

          <option className="option-item" value="false">Not Available</option>

        </select> : <></>}

        {selectedProductId ? (

          <input type="submit" value="Update Product" className="add-product-button" />

        ) : (

          <input type="submit" value="Add Product" className="add-product-button" />

        )}

      </form>

      <br />

      <div className="product-cards">

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

          <div className="product-card" key={index}>

            <img src={productimg} alt={"Product Image"} />

            <div className="product-details">

              <h3>{product.product\_name}</h3>

              <p>{product.product\_description}</p>

              <div className="row">

                <p>{product.category && product.category.category\_name}</p>

                <p>{product.product\_status ? "Available" : "Not Available"}</p>

              </div>

              <p>Rs.{product.price}</p>

            </div>

            <div className="product-actions">

              <button onClick={() => handleEdit(product.id)}>Edit</button>

              <button onClick={() => handleDelete(product.id)}>Delete</button>

            </div>

          </div>

        ))}

      </div>

    </>

  );

};

export default ProductPage;

**src/css/ProductPage.css :-**

.product-cards {

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

    gap: 20px;

}

.product-card {

    width: 250px;

    height: 330px;

    border: 1px solid #ccc;

    border-radius: 5px;

    overflow: hidden;

}

.product-card img {

    width: 100%;

    height: 150px;

    object-fit: cover;

}

.product-details {

    padding: 15px;

    padding-top: 5px;

}

.product-details h3,

p {

    display: flex;

    margin: 0;

}

.product-details .row {

    display: flex;

    flex-direction: row;

    justify-content: space-between;

}

.product-actions {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px;

    padding-inline: 15px;

    border-top: 1px solid #ccc;

}

.product-actions button {

    background-color: #4CAF50;

    color: white;

    padding: 5px 10px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

.product-actions button:hover {

    background-color: #45a049;

}

.product-actions button:focus {

    outline: none;

}

.input-field {

    border: 1px solid #ccc;

    border-radius: 5px;

    padding: 10px;

    margin-bottom: 10px;

    margin-inline: 10px;

    box-sizing: border-box;

}

.option-item {

    padding: 10px;

}

.add-product-button {

    background-color: #4CAF50;

    color: white;

    border: none;

    border-radius: 5px;

    padding: 10px;

    margin-top: 10px;

    margin-inline: 10px;

    cursor: pointer;

    font-size: 16px;

}

.add-product-button:hover {

    background-color: #45a049;

}

**ecom-server**

**index.js**

const express = require('express')

const cors = require('cors')

const errors = require('./src/middleware/error')

const category = require('./src/category/category.routes')

const product = require('./src/product/product.routes')

const app = express()

const port = 3000

app.use(express.json())

app.use(cors())

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

app.use(errors.errorHandler);

app.use('/category', category);

app.use('/product', product);

app.listen(port, () => console.log(`Example app listening on port ${port}!`))

**src/config/db.config.js**

const MYSQL\_DB\_CONFIG = {

    HOST: "localhost",

    USER: "root",

    PORT: 3306,

    PASSWORD: "",

    DB: "awt\_ecom\_app\_db",

};

module.exports = {

    MYSQL\_DB\_CONFIG,

};

**src/helper/db.helper.js**

//for config

const { MYSQL\_DB\_CONFIG } = require("../config/db.config");

//for mysql connection

const mysql = require("mysql2/promise");

//for Sequelize ORM

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

module.exports = db = {};

initialize();

async function initialize() {

    const { HOST, USER, PORT, PASSWORD, DB } = MYSQL\_DB\_CONFIG;

    const connection = await mysql.createConnection({

        host: HOST,

        port: PORT,

        user: USER,

        password: PASSWORD,

    });

    await connection.query(`Create database if not exists \`${DB}\`;`);

    //connect to db

    const sequelize = new Sequelize(DB, USER, PASSWORD, {

        dialect: "mysql",

        host: HOST,

    });

    db.Category = require("../category/category.model")(

        sequelize

    );

    db.Product = require("../product/product.model")(

        sequelize

    );

    db.Varient = require("../varient/varient.model")(

        sequelize

    );

    // Define associations after all models have been defined

    db.Category.hasMany(db.Product);

    db.Product.belongsTo(db.Category);

    db.Product.hasMany(db.Varient);

    db.Varient.belongsTo(db.Product);

    await sequelize.sync({ alter: false });

}

**src/middleware/error.js**

function errorHandler(err, req, res, next) {

    switch (true) {

        case typeof err === "string": {

            // custom application error

            const is404 = err.toLowerCase().endsWith("not found");

            const statusCode = is404 ? 404 : 400;

            return res.status(statusCode).json({ message: err });

        }

        default:

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

    }

}

module.exports = {

    errorHandler,

};

**src/category/category.controller.js**

const category\_service = require("../category/category.service");

exports.create = (req, res, next) => {

    category\_service

        .create(req.body)

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.findAll = (req, res, next) => {

    category\_service

        .getAll()

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.findOne = (req, res, next) => {

    category\_service.getById(req.params.id, (error, response) => {

        if (error) {

            return next(error);

        } else {

            return res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            });

        }

    });

};

exports.update = (req, res, next) => {

    category\_service

        .update(req.params.id, req.body)

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.delete = (req, res, next) => {

    category\_service

        .changeStatus(req.params.id)

        .then((response) =>

            res.status(200).send({ message: "Success", data: response })

        )

        .catch(next);

};

exports.search = (req, res, next) => {

    category\_service

        .searchByKeyword(req.params.keyword)

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.del = (req, res, next) => {

    category\_service

        .del(req.params.id)

        .then((response) =>

            res.status(200).send({ message: "Success", data: response })

        )

        .catch(next);

};

**src/category/category.model.js**

const DataTypes = require('sequelize');

module.exports = model;

function model(sequelize) {

    const attributes = {

        category\_name: { type: DataTypes.STRING, allowNull: false },

        category\_description: { type: DataTypes.STRING, allowNull: true },

        category\_status: { type: DataTypes.BOOLEAN, allowNull: false, defaultValue: true },

    };

    return sequelize.define("category", attributes);

}

**src/category/category.routes.js**

const categoryController = require("../category/category.controller");

const express = require("express");

const router = express.Router();

router.post("/", categoryController.create);

router.get("/", categoryController.findAll);

router.get("/:id", categoryController.findOne);

router.get("/search/:keyword", categoryController.search);

router.put("/:id", categoryController.update);

router.delete("/:id", categoryController.delete);

router.delete("/del/:id", categoryController.del);

module.exports = router;

**src/category/category.service.js**

const db = require("../helper/db.helper");

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

module.exports = {

    getAll,

    getById,

    create,

    update,

    del,

    changeStatus,

    searchByKeyword,

};

async function getAll() {

    return await db.Category.findAll();

}

async function create(params) {

    if (await db.Category.findOne({ where: { category\_name: params.category\_name } })) {

        return "Category " + params.category\_name + " is already exists";

    }

    const category = new db.Category(params);

    await category.save();

    return category;

}

async function getById(id, callback) {

    getCategory(id)

        .then((response) => {

            return callback(null, response);

        })

        .catch((error) => {

            return callback(error);

        });

}

async function update(id, params) {

    const category = await getCategory(id);

    const nameChanged = params.category\_name && params.category\_name !== category.category\_name;

    if (

        nameChanged &&

        (await db.Category.findOne({ where: { category\_name: params.category\_name } }))

    ) {

        return "Category with name " + params.category\_name + " is already exists";

    }

    Object.assign(category, params);

    await category.save();

    return category;

}

async function changeStatus(id) {

    const category = await getCategory(id);

    //    const ret\_msg = '';

    if (category.category\_status) {

        category.category\_status = false;

        // ret\_msg = 'Camp Inactivated';

        console.log("from true");

    } else {

        category.category\_status = true;

        console.log("from false");

        // ret\_msg = 'Camp Activated';

    }

    await category.save();

    return category;

}

async function searchByKeyword(searchKeyword) {

    const category = await db.Category.findAll({

        where: { category\_name: { [Op.like]: "%" + searchKeyword + "%" } },

    });

    if (!category || category == []) return "no category found";

    return category;

}

async function getCategory(id) {

    const category = await db.Category.findByPk(id);

    if (!category) return "category not found";

    return category;

}

async function del(did) {

    return await db.Category.destroy({

        where: {

            id: did

        }

    });

}

**src/product/product.controller.js**

const product\_service = require("../product/product.service");

exports.create = (req, res, next) => {

    product\_service

        .create(req.body)

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.findAll = (req, res, next) => {

    product\_service

        .getAll()

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.findOne = (req, res, next) => {

    product\_service.getById(req.params.id, (error, response) => {

        if (error) {

            return next(error);

        } else {

            return res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            });

        }

    });

};

exports.update = (req, res, next) => {

    product\_service

        .update(req.params.id, req.body)

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.delete = (req, res, next) => {

    product\_service

        .changeStatus(req.params.id)

        .then((response) =>

            res.status(200).send({ message: "Success", data: response })

        )

        .catch(next);

};

exports.search = (req, res, next) => {

    product\_service

        .searchByKeyword(req.params.keyword)

        .then((response) =>

            res.status(200).send({

                message: typeof response === "string" ? "Error" : "Success",

                data: response,

            })

        )

        .catch(next);

};

exports.del = (req, res, next) => {

    product\_service

        .del(req.params.id)

        .then((response) =>

            res.status(200).send({ message: "Success", data: response })

        )

        .catch(next);

};

**src/product/product.model.js**

const DataTypes = require('sequelize');

module.exports = model;

function model(sequelize) {

    const { INTEGER, BOOLEAN, STRING, TEXT } = DataTypes;

    const attributes = {

        product\_name: { type: STRING, allowNull: false },

        product\_description: { type: TEXT, allowNull: true },

        price: { type: INTEGER, allowNull: false },

        categoryId: { type: INTEGER, allowNull: false },

        product\_status: { type: BOOLEAN, allowNull: false, defaultValue: true },

    };

    return sequelize.define("product", attributes);

}

**src/product/product.routes.js**

const productController = require("../product/product.controller");

const express = require("express");

const router = express.Router();

router.post("/", productController.create);

router.get("/", productController.findAll);

router.get("/:id", productController.findOne);

router.get("/search/:keyword", productController.search);

router.put("/:id", productController.update);

router.delete("/:id", productController.delete);

router.delete("/del/:id", productController.del);

module.exports = router;

**src/product/product.service.js**

const db = require("../helper/db.helper");

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

module.exports = {

    getAll,

    getById,

    create,

    update,

    del,

    changeStatus,

    searchByKeyword,

};

async function getAll() {

    return await db.Product.findAll(

        {

            include: [

                {

                    model: db.Category

                }

            ],

        },

    );

}

async function create(params) {

    if (await db.Product.findOne({ where: { product\_name: params.product\_name } })) {

        return "Product " + params.product\_name + " is already exists";

    }

    const product = new db.Product(params);

    await product.save();

    return product;

}

async function getById(id, callback) {

    getProduct(id)

        .then((response) => {

            return callback(null, response);

        })

        .catch((error) => {

            return callback(error);

        });

}

async function update(id, params) {

    const product = await getProduct(id);

    const nameChanged = params.product\_name && params.product\_name !== product.product\_name;

    if (

        nameChanged &&

        (await db.Product.findOne({ where: { product\_name: params.product\_name } }))

    ) {

        return "Product with name " + params.product\_name + " is already exists";

    }

    Object.assign(product, params);

    await product.save();

    return product;

}

async function changeStatus(id) {

    const product = await getProduct(id);

    //    const ret\_msg = '';

    if (product.product\_status) {

        product.product\_status = false;

        // ret\_msg = 'Camp Inactivated';

        console.log("from true");

    } else {

        product.product\_status = true;

        console.log("from false");

        // ret\_msg = 'Camp Activated';

    }

    await product.save();

    return product;

}

async function searchByKeyword(searchKeyword) {

    const product = await db.Product.findAll({

        where: { product\_name: { [Op.like]: "%" + searchKeyword + "%" } },

    });

    if (!product || product == []) return "no product found";

    return product;

}

async function getProduct(id) {

    const product = await db.Product.findByPk(id);

    if (!product) return "product not found";

    return product;

}

async function del(did) {

    return await db.Product.destroy({

        where: {

            id: did

        }

    });

}

**Output :-**



