**FULL STACK WEB DEVELOPMENT**

**E-COMMERCE CART SYSTEM**

**Frontend:** React + Axios

**Backend:** Express.js with RESTful APIs

**Database:** MongoDB with Mongoose ORM

**Security:** JWT authentication + bcrypt for password hashing

**FRONT END (jsx)**

**// components/ProductList.js**

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

**import axios from '../api';**

function ProductList({ addToCart }) {

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

useEffect(() => {

axios.get('/products').then(res => setProducts(res.data));

}, []);

return (

<div>

{products.map(product => (

<div key={product.\_id}>

<img src={product.image} alt={product.name} width={100} />

<h4>{product.name}</h4>

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

<button onClick={() => addToCart(product)}>Add to Cart</button>

</div>

))}

</div>

);

}

CARD MANAGEMENT (jsx)

// components/Cart.js

function Cart({ cart, updateQuantity, removeFromCart }) {

const total = cart.reduce((sum, item) => sum + item.price \* item.quantity, 0);

return (

<div>

{cart.map(item => (

<div key={item.\_id}>

<h4>{item.name}</h4>

<p>Qty:

<input

type="number"

value={item.quantity}

onChange={(e) => updateQuantity(item.\_id, e.target.value)}

/>

</p>

<button onClick={() => removeFromCart(item.\_id)}>Remove</button>

</div>

))}

<h3>Total: ${total.toFixed(2)}</h3>

</div>

);

}

**BACKEND**

**PRODUCT MODEL**

// models/Product.js

const mongoose = require('mongoose');

const ProductSchema = new mongoose.Schema({

name: String,

price: Number,

image: String,

description: String

});

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

**USER AUTHENTICATION**

// routes/users.js

const express = require('express');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const router = express.Router();

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

router.post('/register', async (req, res) => {

const hashed = await bcrypt.hash(req.body.password, 10);

const user = new User({ email: req.body.email, password: hashed });

await user.save();

res.sendStatus(201);

});

router.post('/login', async (req, res) => {

const user = await User.findOne({ email: req.body.email });

if (user && await bcrypt.compare(req.body.password, user.password)) {

const token = jwt.sign({ id: user.\_id }, 'secret');

res.json({ token });

} else {

res.sendStatus(401);

}

});

**ORDER HANDLING**

// routes/orders.js

router.post('/', authMiddleware, async (req, res) => {

const order = new Order({

userId: req.user.id,

items: req.body.items,

total: req.body.total

});

await order.save();

res.status(201).send(order);

});

**SECURITY SESSION**

function authMiddleware(req, res, next) {

const token = req.headers.authorization?.split(" ")[1];

if (!token) return res.sendStatus(401);

jwt.verify(token, 'secret', (err, user) => {

if (err) return res.sendStatus(403);

req.user = user;

next();

});

}