React.js front-end design for a full-fledged e-commerce system, covering:

1. User Registration & Login

2. Customer Dashboard (Home, Cart, Payment, Orders)

3. Admin Dashboard (Order Management, User Management)

I'll break down the design for each page and provide a React.js + Tailwind CSS implementation.

1. User Authentication (Login & Registration)

Tech Stack

React.js (Frontend)

Firebase/Auth0 (Authentication)

React Hook Form & Yup (Form validation)

React Router (Navigation)

Login & Registration UI

Fields: Email, Password, Name (for registration)

Validation: Required fields, password strength

Login: Redirect to dashboard after success

Registration: Save user details in database

Code - AuthForm.jsx

import { useState } from "react";

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

import { auth } from "../firebase"; // Firebase auth instance

import { signInWithEmailAndPassword, createUserWithEmailAndPassword } from "firebase/auth";

export default function AuthForm({ isRegister }) {

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

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

const navigate = useNavigate();

const handleSubmit = async (e) => {

e.preventDefault();

try {

if (isRegister) {

await createUserWithEmailAndPassword(auth, email, password);

} else {

await signInWithEmailAndPassword(auth, email, password);

}

navigate("/dashboard"); // Redirect after login/register

} catch (err) {

setError(err.message);

}

};

return (

<div className="flex justify-center items-center h-screen">

<form onSubmit={handleSubmit} className="bg-white p-6 rounded shadow-lg">

<h2 className="text-xl font-semibold">{isRegister ? "Register" : "Login"}</h2>

{error && <p className="text-red-500">{error}</p>}

<input type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)}

className="border p-2 w-full my-2" required />

<input type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)}

className="border p-2 w-full my-2" required />

<button type="submit" className="bg-blue-500 text-white p-2 w-full rounded">

{isRegister ? "Sign Up" : "Login"}

</button>

</form>

</div>

);

}

Routing (App.js)

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

import AuthForm from "./components/AuthForm";

import Dashboard from "./pages/Dashboard";

function App() {

return (

<Router>

<Routes>

<Route path="/" element={<AuthForm isRegister={false} />} />

<Route path="/register" element={<AuthForm isRegister={true} />} />

<Route path="/dashboard" element={<Dashboard />} />

</Routes>

</Router>

);

}

export default App;

2. Customer Home Page

Features

Displays featured products

Search bar & filtering

Product categories

Code - HomePage.jsx

jsx

Copy

Edit

import { useState } from "react";

const products = [

{ id: 1, name: "Laptop", price: "$999", image: "/laptop.jpg" },

{ id: 2, name: "Headphones", price: "$199", image: "/headphones.jpg" },

];

export default function HomePage() {

const [search, setSearch] = useState("");

const filteredProducts = products.filter((p) => p.name.toLowerCase().includes(search.toLowerCase()));

return (

<div className="p-6">

<input type="text" placeholder="Search..." className="border p-2 w-full"

onChange={(e) => setSearch(e.target.value)} />

<div className="grid grid-cols-2 gap-4 mt-4">

{filteredProducts.map((product) => (

<div key={product.id} className="border p-4">

<img src={product.image} alt={product.name} className="h-32 w-full object-cover" />

<h2 className="text-lg">{product.name}</h2>

<p className="text-gray-600">{product.price}</p>

<button className="bg-blue-500 text-white p-2 w-full mt-2">Add to Cart</button>

</div>

))}

</div>

</div>

);

}

3. Shopping Cart

Features

Add/remove items

View total price

Code - CartPage.jsx

import { useState } from "react";

export default function CartPage() {

const [cart, setCart] = useState([{ name: "Laptop", price: 999, qty: 1 }]);

const removeItem = (index) => {

setCart(cart.filter((\_, i) => i !== index));

};

return (

<div className="p-6">

{cart.length === 0 ? <p>Your cart is empty.</p> : cart.map((item, index) => (

<div key={index} className="border p-4 flex justify-between">

<p>{item.name} - ${item.price}</p>

<button onClick={() => removeItem(index)} className="bg-red-500 text-white px-2">Remove</button>

</div>

))}

<h2 className="mt-4 text-lg">Total: ${cart.reduce((sum, item) => sum + item.price, 0)}</h2>

</div>

);

}

4. Payment Page

Features

Stripe payment integration

Code - PaymentPage.jsx

import { loadStripe } from "@stripe/stripe-js";

import { Elements, CardElement, useStripe, useElements } from "@stripe/react-stripe-js";

const stripePromise = loadStripe("your-stripe-public-key");

function CheckoutForm() {

const stripe = useStripe();

const elements = useElements();

const handleSubmit = async (e) => {

e.preventDefault();

const { error, paymentMethod } = await stripe.createPaymentMethod({ type: "card", card: elements.getElement(CardElement) });

if (!error) console.log("Payment Successful!", paymentMethod);

};

return (

<form onSubmit={handleSubmit} className="p-6">

<CardElement className="border p-2 w-full" />

<button type="submit" className="bg-blue-500 text-white p-2 w-full mt-4">Pay</button>

</form>

);

}

export default function PaymentPage() {

return (

<Elements stripe={stripePromise}>

<CheckoutForm />

</Elements>

);

}

5. Admin Dashboard

Features

View orders

Manage users

Code - AdminDashboard.jsx

const orders = [{ id: 1, user: "John Doe", amount: "$999", status: "Pending" }];

export default function AdminDashboard() {

return (

<div className="p-6">

<h2 className="text-xl">Orders</h2>

{orders.map((order) => (

<div key={order.id} className="border p-4 flex justify-between">

<p>{order.user} - {order.amount} - {order.status}</p>

</div>

))}

</div>

);

}