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

import { initializeApp } from 'firebase/app';

import {

    getAuth,

    createUserWithEmailAndPassword,

    signInWithEmailAndPassword,

    signOut,

    onAuthStateChanged,

    signInAnonymously,

    signInWithCustomToken

} from 'firebase/auth';

import {

    getFirestore,

    collection,

    addDoc,

    query,

    where,

    onSnapshot,

    doc,

    setDoc,

    getDoc,

    Timestamp,

    orderBy // Added orderBy

} from 'firebase/firestore';

import { setLogLevel } from 'firebase/app'; // Import setLogLevel

// --- Global Firebase Variables ---

// These would be provided by the Canvas environment in a real scenario

const \_\_app\_id = typeof window.\_\_app\_id !== 'undefined' ? window.\_\_app\_id : 'maternity-benefit-portal';

const \_\_firebase\_config\_str = typeof window.\_\_firebase\_config !== 'undefined' ? window.\_\_firebase\_config : JSON.stringify({

    apiKey: "YOUR\_API\_KEY", // Replace with your actual Firebase config

    authDomain: "YOUR\_AUTH\_DOMAIN",

    projectId: "YOUR\_PROJECT\_ID",

    storageBucket: "YOUR\_STORAGE\_BUCKET",

    messagingSenderId: "YOUR\_MESSAGING\_SENDER\_ID",

    appId: "YOUR\_APP\_ID"

});

const \_\_initial\_auth\_token = typeof window.\_\_initial\_auth\_token !== 'undefined' ? window.\_\_initial\_auth\_token : undefined;

const firebaseConfig = JSON.parse(\_\_firebase\_config\_str);

// --- Icon Components (Simple SVGs for now) ---

const HomeIcon = () => (

    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-6 h-6">

        <path strokeLinecap="round" strokeLinejoin="round" d="M2.25 12l8.954-8.955c.44-.439 1.152-.439 1.591 0L21.75 12M4.5 9.75v10.125c0 .621.504 1.125 1.125 1.125H9.75v-4.875c0-.621.504-1.125 1.125-1.125h2.25c.621 0 1.125.504 1.125 1.125V21h4.125c.621 0 1.125-.504 1.125-1.125V9.75M8.25 21h7.5" />

    </svg>

);

const DocumentPlusIcon = () => (

    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-6 h-6">

        <path strokeLinecap="round" strokeLinejoin="round" d="M19.5 14.25v-2.625a3.375 3.375 0 00-3.375-3.375h-1.5A1.125 1.125 0 0113.5 7.125v-1.5a3.375 3.375 0 00-3.375-3.375H8.25m3.75 9v6m3-3H9m1.5-12H5.625c-.621 0-1.125.504-1.125 1.125v17.25c0 .621.504 1.125 1.125 1.125h12.75c.621 0 1.125-.504 1.125-1.125V11.25a9 9 0 00-9-9z" />

    </svg>

);

const ListBulletIcon = () => (

    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-6 h-6">

        <path strokeLinecap="round" strokeLinejoin="round" d="M3.75 12h16.5m-16.5 3.75h16.5M3.75 19.5h16.5M5.625 4.5h12.75a1.875 1.875 0 010 3.75H5.625a1.875 1.875 0 010-3.75z" />

    </svg>

);

const UserCircleIcon = () => (

    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-6 h-6">

        <path strokeLinecap="round" strokeLinejoin="round" d="M17.982 18.725A7.488 7.488 0 0012 15.75a7.488 7.488 0 00-5.982 2.975m11.963 0a9 9 0 10-11.963 0m11.963 0A8.966 8.966 0 0112 21a8.966 8.966 0 01-5.982-2.275M15 9.75a3 3 0 11-6 0 3 3 0 016 0z" />

    </svg>

);

const ArrowLeftOnRectangleIcon = () => (

    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-6 h-6">

      <path strokeLinecap="round" strokeLinejoin="round" d="M15.75 9V5.25A2.25 2.25 0 0013.5 3h-6a2.25 2.25 0 00-2.25 2.25v13.5A2.25 2.25 0 007.5 21h6a2.25 2.25 0 002.25-2.25V15m3 0l3-3m0 0l-3-3m3 3H9" />

    </svg>

);

// --- Modal Component ---

const Modal = ({ message, type, onClose }) => {

    if (!message) return null;

    const bgColor = type === 'error' ? 'bg-red-100 border-red-400 text-red-700' : 'bg-green-100 border-green-400 text-green-700';

    const borderColor = type === 'error' ? 'border-red-500' : 'border-green-500';

    return (

        <div className="fixed inset-0 bg-gray-600 bg-opacity-50 overflow-y-auto h-full w-full z-50 flex justify-center items-center">

            <div className={`relative mx-auto p-5 border w-full max-w-md shadow-lg rounded-md ${bgColor} ${borderColor}`}>

                <div className="mt-3 text-center">

                    <h3 className="text-lg leading-6 font-medium">{type === 'error' ? 'Error' : 'Success'}</h3>

                    <div className="mt-2 px-7 py-3">

                        <p className="text-sm">{message}</p>

                    </div>

                    <div className="items-center px-4 py-3">

                        <button

                            onClick={onClose}

                            className={`px-4 py-2 ${type === 'error' ? 'bg-red-500 hover:bg-red-700' : 'bg-green-500 hover:bg-green-700'} text-white text-base font-medium rounded-md w-full shadow-sm focus:outline-none focus:ring-2 focus:ring-offset-2 ${type === 'error' ? 'focus:ring-red-500' : 'focus:ring-green-500'}`}

                        >

                            OK

                        </button>

                    </div>

                </div>

            </div>

        </div>

    );

};

// --- Main App Component ---

function App() {

    const [page, setPage] = useState('home'); // home, register, login, dashboard, new-complaint, view-complaints

    const [user, setUser] = useState(null);

    const [userId, setUserId] = useState(null);

    const [isLoading, setIsLoading] = useState(true);

    const [db, setDb] = useState(null);

    const [auth, setAuth] = useState(null);

    const [isAuthReady, setIsAuthReady] = useState(false);

    const [modalMessage, setModalMessage] = useState('');

    const [modalType, setModalType] = useState(''); // 'success' or 'error'

    const appId = \_\_app\_id; // Use the global appId

    // Firebase Initialization and Auth State Listener

    useEffect(() => {

        try {

            const appInstance = initializeApp(firebaseConfig);

            const authInstance = getAuth(appInstance);

            const dbInstance = getFirestore(appInstance);

            setLogLevel('debug'); // Enable Firebase debug logging

            setAuth(authInstance);

            setDb(dbInstance);

            const unsubscribe = onAuthStateChanged(authInstance, async (currentUser) => {

                setIsLoading(true);

                if (currentUser) {

                    setUser(currentUser);

                    setUserId(currentUser.uid);

                    setPage('dashboard'); // Redirect to dashboard if logged in

                } else {

                    // If \_\_initial\_auth\_token is not defined, sign in anonymously

                    // This is for Canvas environment compatibility

                    if (typeof \_\_initial\_auth\_token === 'undefined' && authInstance) {

                        await signInAnonymously(authInstance).catch(error => {

                            console.error("Anonymous sign-in failed:", error);

                            setModalMessage(`Anonymous sign-in failed: ${error.message}`);

                            setModalType('error');

                        });

                    }

                    setUser(null);

                    setUserId(crypto.randomUUID()); // Generate a random ID for non-logged-in users

                    setPage('home'); // Default to home if not logged in

                }

                setIsAuthReady(true);

                setIsLoading(false);

            });

            // Perform custom token sign-in if token is available

            // This is for Canvas environment compatibility

            const performCustomSignIn = async () => {

                if (typeof \_\_initial\_auth\_token !== 'undefined' && authInstance) {

                    try {

                        await signInWithCustomToken(authInstance, \_\_initial\_auth\_token);

                        console.log("Successfully signed in with custom token.");

                    } catch (error) {

                        console.error("Custom token sign-in failed:", error);

                        setModalMessage(`Custom token sign-in failed: ${error.message}`);

                        setModalType('error');

                        // Fallback to anonymous sign-in if custom token fails

                        await signInAnonymously(authInstance).catch(anonError => {

                            console.error("Anonymous sign-in failed after custom token failure:", anonError);

                            setModalMessage(`Anonymous sign-in failed: ${anonError.message}`);

                            setModalType('error');

                        });

                    }

                }

            };

            performCustomSignIn();

            return () => unsubscribe();

        } catch (error) {

            console.error("Firebase initialization error:", error);

            setModalMessage(`Firebase initialization error: ${error.message}`);

            setModalType('error');

            setIsLoading(false);

            setIsAuthReady(true); // Still set auth ready to allow UI to render potentially

        }

    }, []);

    const handleLogout = async () => {

        if (!auth) {

            setModalMessage("Authentication service not available.");

            setModalType('error');

            return;

        }

        try {

            await signOut(auth);

            setUser(null);

            setUserId(crypto.randomUUID()); // Reset to a new random ID on logout

            setPage('home');

            setModalMessage("Logged out successfully.");

            setModalType('success');

        } catch (error) {

            console.error("Logout error:", error);

            setModalMessage(`Logout failed: ${error.message}`);

            setModalType('error');

        }

    };

    const closeModal = () => {

        setModalMessage('');

        setModalType('');

    };

    if (isLoading) {

        return (

            <div className="flex items-center justify-center min-h-screen bg-gray-100 font-sans">

                <div className="text-xl font-semibold text-indigo-600">Loading Portal...</div>

            </div>

        );

    }

    return (

        <div className="min-h-screen bg-gradient-to-br from-indigo-100 via-purple-50 to-pink-100 font-sans">

            <Modal message={modalMessage} type={modalType} onClose={closeModal} />

            <Navbar user={user} setPage={setPage} handleLogout={handleLogout} />

            <main className="container mx-auto px-4 py-8">

                {!isAuthReady && !isLoading && <p className="text-center text-yellow-600">Authenticating...</p>}

                {isAuthReady && page === 'home' && <HomePage setPage={setPage} />}

                {isAuthReady && page === 'register' && <RegisterPage setPage={setPage} auth={auth} setModalMessage={setModalMessage} setModalType={setModalType} />}

                {isAuthReady && page === 'login' && <LoginPage setPage={setPage} auth={auth} setModalMessage={setModalMessage} setModalType={setModalType} />}

                {isAuthReady && user && page === 'dashboard' && <DashboardPage setPage={setPage} user={user} userId={userId} />}

                {isAuthReady && user && page === 'new-complaint' && <ComplaintFormPage setPage={setPage} db={db} user={user} appId={appId} userId={userId} setModalMessage={setModalMessage} setModalType={setModalType} />}

                {isAuthReady && user && page === 'view-complaints' && <ViewComplaintsPage db={db} user={user} appId={appId} userId={userId} isAuthReady={isAuthReady} setModalMessage={setModalMessage} setModalType={setModalType} />}

            </main>

            <Footer />

        </div>

    );

}

// --- Navbar Component ---

const Navbar = ({ user, setPage, handleLogout }) => {

    return (

        <nav className="bg-white shadow-lg sticky top-0 z-40">

            <div className="container mx-auto px-6 py-3 flex justify-between items-center">

                <div onClick={() => setPage(user ? 'dashboard' : 'home')} className="cursor-pointer">

                    <h1 className="text-2xl font-bold text-indigo-600 hover:text-indigo-700 transition-colors">Maternity Justice Portal</h1>

                    <p className="text-xs text-gray-500">Empowering Working Mothers in India</p>

                </div>

                <div className="space-x-4">

                    {user ? (

                        <>

                            <button onClick={() => setPage('dashboard')} className="text-gray-700 hover:text-indigo-600 transition-colors px-3 py-2 rounded-md text-sm font-medium flex items-center space-x-1"> <HomeIcon /> <span>Dashboard</span></button>

                            <button onClick={() => setPage('new-complaint')} className="text-gray-700 hover:text-indigo-600 transition-colors px-3 py-2 rounded-md text-sm font-medium flex items-center space-x-1"> <DocumentPlusIcon /> <span>File Complaint</span></button>

                            <button onClick={() => setPage('view-complaints')} className="text-gray-700 hover:text-indigo-600 transition-colors px-3 py-2 rounded-md text-sm font-medium flex items-center space-x-1"> <ListBulletIcon /> <span>My Complaints</span></button>

                            <button onClick={handleLogout} className="bg-pink-500 hover:bg-pink-600 text-white px-4 py-2 rounded-md text-sm font-medium transition-colors flex items-center space-x-1"> <ArrowLeftOnRectangleIcon /> <span>Logout</span></button>

                        </>

                    ) : (

                        <>

                            <button onClick={() => setPage('home')} className="text-gray-700 hover:text-indigo-600 transition-colors px-3 py-2 rounded-md text-sm font-medium">Home</button>

                            <button onClick={() => setPage('login')} className="text-gray-700 hover:text-indigo-600 transition-colors px-3 py-2 rounded-md text-sm font-medium">Login</button>

                            <button onClick={() => setPage('register')} className="bg-indigo-500 hover:bg-indigo-600 text-white px-4 py-2 rounded-md text-sm font-medium transition-colors">Register</button>

                        </>

                    )}

                </div>

            </div>

        </nav>

    );

};

// --- Footer Component ---

const Footer = () => {

    return (

        <footer className="bg-gray-800 text-white py-8 mt-12">

            <div className="container mx-auto px-6 text-center">

                <p className="text-sm">&copy; {new Date().getFullYear()} Maternity Justice Portal. All rights reserved.</p>

                <p className="text-xs mt-1">Disclaimer: This portal provides a platform for grievance registration. Legal advice and actions are facilitated by independent legal professionals.</p>

            </div>

        </footer>

    );

};

// --- HomePage Component ---

const HomePage = ({ setPage }) => {

    return (

        <div className="text-center py-12 px-4">

            <header className="mb-12">

                <h2 className="text-5xl font-extrabold text-indigo-700 mb-4 animate-fade-in-down">Know Your Rights. Claim Your Dues.</h2>

                <p className="text-xl text-gray-700 mb-8 max-w-2xl mx-auto animate-fade-in-up">

                    Are you facing issues with receiving your rightful maternity benefits in India?

                    This portal helps you register your complaint and connect with legal assistance.

                </p>

                <div className="space-x-4 animate-fade-in">

                    <button onClick={() => setPage('register')} className="bg-indigo-600 hover:bg-indigo-700 text-white font-semibold py-3 px-8 rounded-lg shadow-lg transform hover:scale-105 transition-all duration-300 text-lg">

                        Register Your Complaint

                    </button>

                    <button onClick={() => setPage('login')} className="bg-pink-500 hover:bg-pink-600 text-white font-semibold py-3 px-8 rounded-lg shadow-lg transform hover:scale-105 transition-all duration-300 text-lg">

                        Login to Your Account

                    </button>

                </div>

            </header>

            <section className="mt-16 grid md:grid-cols-3 gap-8 text-left">

                <div className="bg-white p-6 rounded-xl shadow-xl hover:shadow-2xl transition-shadow duration-300">

                    <h3 className="text-2xl font-semibold text-indigo-600 mb-3">What is the Maternity Benefit Act?</h3>

                    <p className="text-gray-600 text-sm">

                        The Maternity Benefit Act, 1961 (amended in 2017) provides pregnant women and new mothers with paid leave, crèche facilities, and protection from dismissal.

                        Key benefits include 26 weeks of paid leave for the first two children.

                    </p>

                </div>

                <div className="bg-white p-6 rounded-xl shadow-xl hover:shadow-2xl transition-shadow duration-300">

                    <h3 className="text-2xl font-semibold text-indigo-600 mb-3">Common Issues Faced</h3>

                    <p className="text-gray-600 text-sm">

                        Many women are denied full benefits, face pressure to resign, or are not provided with adequate facilities. This portal is here to help you address such injustices.

                    </p>

                </div>

                <div className="bg-white p-6 rounded-xl shadow-xl hover:shadow-2xl transition-shadow duration-300">

                    <h3 className="text-2xl font-semibold text-indigo-600 mb-3">How We Help</h3>

                    <p className="text-gray-600 text-sm">

                        Register your complaint securely. Our system will help organize your information, which can then be used by empaneled lawyers to prepare and send a legal notice to your employer.

                    </p>

                </div>

            </section>

        </div>

    );

};

// --- RegisterPage Component ---

const RegisterPage = ({ setPage, auth, setModalMessage, setModalType }) => {

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

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

    const [confirmPassword, setConfirmPassword] = useState('');

    const [error, setError] = useState(''); // Local error for form validation

    const [isLoading, setIsLoading] = useState(false);

    const handleRegister = async (e) => {

        e.preventDefault();

        setError('');

        if (!auth) {

            setModalMessage("Authentication service not available. Please try again later.");

            setModalType('error');

            return;

        }

        if (password !== confirmPassword) {

            setError("Passwords do not match.");

            return;

        }

        if (password.length < 6) {

            setError("Password should be at least 6 characters long.");

            return;

        }

        setIsLoading(true);

        try {

            await createUserWithEmailAndPassword(auth, email, password);

            setModalMessage("Registration successful! Please login.");

            setModalType('success');

            setPage('login');

        } catch (err) {

            console.error("Registration error:", err);

            setError(err.message); // Show Firebase specific error if any

            setModalMessage(`Registration failed: ${err.message}`);

            setModalType('error');

        } finally {

            setIsLoading(false);

        }

    };

    return (

        <div className="flex flex-col items-center justify-center py-8">

            <div className="w-full max-w-md bg-white p-8 rounded-xl shadow-2xl">

                <h2 className="text-3xl font-bold text-center text-indigo-600 mb-8">Create Your Account</h2>

                {error && <p className="bg-red-100 text-red-700 p-3 rounded-md mb-4 text-sm">{error}</p>}

                <form onSubmit={handleRegister} className="space-y-6">

                    <div>

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

                        <input

                            type="email"

                            value={email}

                            onChange={(e) => setEmail(e.target.value)}

                            required

                            className="w-full px-4 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            placeholder="you@example.com"

                        />

                    </div>

                    <div>

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

                        <input

                            type="password"

                            value={password}

                            onChange={(e) => setPassword(e.target.value)}

                            required

                            className="w-full px-4 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            placeholder="••••••••"

                        />

                         <p className="text-xs text-gray-500 mt-1">Password must be at least 6 characters.</p>

                    </div>

                    <div>

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

                        <input

                            type="password"

                            value={confirmPassword}

                            onChange={(e) => setConfirmPassword(e.target.value)}

                            required

                            className="w-full px-4 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            placeholder="••••••••"

                        />

                    </div>

                    <button

                        type="submit"

                        disabled={isLoading}

                        className="w-full bg-indigo-600 hover:bg-indigo-700 text-white font-semibold py-3 px-4 rounded-lg shadow-md focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-offset-2 transition-all duration-300 disabled:opacity-50"

                    >

                        {isLoading ? 'Registering...' : 'Register'}

                    </button>

                </form>

                <p className="text-center text-sm text-gray-600 mt-8">

                    Already have an account?{' '}

                    <button onClick={() => setPage('login')} className="font-medium text-indigo-600 hover:text-indigo-500 transition-colors">

                        Login here

                    </button>

                </p>

            </div>

        </div>

    );

};

// --- LoginPage Component ---

const LoginPage = ({ setPage, auth, setModalMessage, setModalType }) => {

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

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

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

    const [isLoading, setIsLoading] = useState(false);

    const handleLogin = async (e) => {

        e.preventDefault();

        setError('');

        if (!auth) {

            setModalMessage("Authentication service not available. Please try again later.");

            setModalType('error');

            return;

        }

        setIsLoading(true);

        try {

            await signInWithEmailAndPassword(auth, email, password);

            setModalMessage("Login successful!");

            setModalType('success');

            setPage('dashboard'); // This will be handled by onAuthStateChanged in App.js too

        } catch (err) {

            console.error("Login error:", err);

            setError(err.message);

            setModalMessage(`Login failed: ${err.message}`);

            setModalType('error');

        } finally {

            setIsLoading(false);

        }

    };

    return (

        <div className="flex flex-col items-center justify-center py-8">

            <div className="w-full max-w-md bg-white p-8 rounded-xl shadow-2xl">

                <h2 className="text-3xl font-bold text-center text-indigo-600 mb-8">Welcome Back</h2>

                {error && <p className="bg-red-100 text-red-700 p-3 rounded-md mb-4 text-sm">{error}</p>}

                <form onSubmit={handleLogin} className="space-y-6">

                    <div>

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

                        <input

                            type="email"

                            value={email}

                            onChange={(e) => setEmail(e.target.value)}

                            required

                            className="w-full px-4 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            placeholder="you@example.com"

                        />

                    </div>

                    <div>

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

                        <input

                            type="password"

                            value={password}

                            onChange={(e) => setPassword(e.target.value)}

                            required

                            className="w-full px-4 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            placeholder="••••••••"

                        />

                    </div>

                    <button

                        type="submit"

                        disabled={isLoading}

                        className="w-full bg-indigo-600 hover:bg-indigo-700 text-white font-semibold py-3 px-4 rounded-lg shadow-md focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-offset-2 transition-all duration-300 disabled:opacity-50"

                    >

                        {isLoading ? 'Logging in...' : 'Login'}

                    </button>

                </form>

                <p className="text-center text-sm text-gray-600 mt-8">

                    Don't have an account?{' '}

                    <button onClick={() => setPage('register')} className="font-medium text-indigo-600 hover:text-indigo-500 transition-colors">

                        Register here

                    </button>

                </p>

            </div>

        </div>

    );

};

// --- DashboardPage Component ---

const DashboardPage = ({ setPage, user, userId }) => {

    return (

        <div className="bg-white p-8 rounded-xl shadow-2xl">

            <h2 className="text-3xl font-bold text-indigo-700 mb-6">Welcome, {user.email}!</h2>

            <p className="text-gray-600 mb-2">This is your personal dashboard. From here, you can manage your complaints and access resources.</p>

            <p className="text-xs text-gray-500 mb-8">Your User ID (for reference): {userId}</p>

            <div className="grid md:grid-cols-2 gap-6">

                <div

                    onClick={() => setPage('new-complaint')}

                    className="bg-indigo-50 p-6 rounded-lg hover:bg-indigo-100 transition-colors cursor-pointer shadow-md hover:shadow-lg"

                >

                    <div className="flex items-center text-indigo-600 mb-3">

                        <DocumentPlusIcon />

                        <h3 className="text-xl font-semibold ml-3">File a New Complaint</h3>

                    </div>

                    <p className="text-gray-700 text-sm">If you're facing issues with your maternity benefits, start here to submit the details of your grievance.</p>

                </div>

                <div

                    onClick={() => setPage('view-complaints')}

                    className="bg-pink-50 p-6 rounded-lg hover:bg-pink-100 transition-colors cursor-pointer shadow-md hover:shadow-lg"

                >

                     <div className="flex items-center text-pink-600 mb-3">

                        <ListBulletIcon />

                        <h3 className="text-xl font-semibold ml-3">View My Complaints</h3>

                    </div>

                    <p className="text-gray-700 text-sm">Track the status of your submitted complaints and review the information you've provided.</p>

                </div>

            </div>

            <div className="mt-10 border-t pt-6">

                <h3 className="text-xl font-semibold text-gray-800 mb-3">Important Information</h3>

                <ul className="list-disc list-inside text-gray-600 space-y-2 text-sm">

                    <li>Ensure all information provided in your complaints is accurate and truthful.</li>

                    <li>Keep copies of all relevant documents (employment contract, communications, medical certificates).</li>

                    <li>The Maternity Benefit Act, 2017, provides significant protections. Familiarize yourself with its key provisions.</li>

                </ul>

            </div>

        </div>

    );

};

// --- ComplaintFormPage Component ---

const ComplaintFormPage = ({ setPage, db, user, appId, userId, setModalMessage, setModalType }) => {

    const [formData, setFormData] = useState({

        complainantName: '',

        complainantContact: '',

        companyName: '',

        companyAddress: '',

        dateOfJoining: '',

        employmentType: 'permanent', // permanent, contract, temporary

        expectedDeliveryDate: '',

        actualDeliveryDate: '',

        isAdoptiveOrCommissioning: 'no', // yes, no

        adoptionDate: '', // if yes

        numberOfSurvivingChildren: '0',

        issueType: 'denial\_of\_leave', // denial\_of\_leave, partial\_leave, non\_payment, crèche\_denial, termination, other

        issueDetails: '',

        chronologyOfEvents: '',

        supportingDocumentsInfo: '', // Text area for now, actual upload later

        consentToShare: false,

    });

    const [isSubmitting, setIsSubmitting] = useState(false);

    const handleChange = (e) => {

        const { name, value, type, checked } = e.target;

        setFormData(prev => ({

            ...prev,

            [name]: type === 'checkbox' ? checked : value

        }));

    };

    const handleSubmit = async (e) => {

        e.preventDefault();

        if (!db || !user || !userId) {

            setModalMessage("Database or user information is not available. Cannot submit complaint.");

            setModalType('error');

            return;

        }

        if (!formData.consentToShare) {

            setModalMessage("You must consent to share your information with legal professionals to proceed.");

            setModalType('error');

            return;

        }

        setIsSubmitting(true);

        try {

            // Path: /artifacts/{appId}/users/{userId}/complaints/{complaintId}

            const complaintsCollectionPath = `artifacts/${appId}/users/${userId}/complaints`;

            await addDoc(collection(db, complaintsCollectionPath), {

                ...formData,

                userId: user.uid, // Store the authenticated user's UID

                userEmail: user.email,

                status: 'submitted', // Initial status

                submittedAt: Timestamp.now()

            });

            setModalMessage("Complaint submitted successfully!");

            setModalType('success');

            setPage('view-complaints'); // Navigate to view complaints page

        } catch (error) {

            console.error("Error submitting complaint:", error);

            setModalMessage(`Failed to submit complaint: ${error.message}. Ensure you are logged in and have a stable connection.`);

            setModalType('error');

        } finally {

            setIsSubmitting(false);

        }

    };

    // Form field structure

    const formFields = [

        { label: "Your Full Name", name: "complainantName", type: "text", required: true },

        { label: "Your Contact Number", name: "complainantContact", type: "tel", required: true },

        { label: "Company Name", name: "companyName", type: "text", required: true },

        { label: "Company Full Address", name: "companyAddress", type: "textarea", required: true },

        { label: "Date of Joining Company", name: "dateOfJoining", type: "date", required: true },

        { label: "Type of Employment", name: "employmentType", type: "select", options: [

            {value: "permanent", label: "Permanent"},

            {value: "contract", label: "Contract"},

            {value: "temporary", label: "Temporary"},

            {value: "probation", label: "Probation"},

            {value: "other", label: "Other"}

        ], required: true },

        { label: "Expected Date of Delivery (EDD)", name: "expectedDeliveryDate", type: "date" },

        { label: "Actual Date of Delivery (if occurred)", name: "actualDeliveryDate", type: "date" },

        { label: "Are you an adoptive or commissioning mother?", name: "isAdoptiveOrCommissioning", type: "select", options: [{value:"no", label:"No"}, {value:"yes", label:"Yes"}], required: true },

    ];

    const conditionalFields = formData.isAdoptiveOrCommissioning === 'yes' ? [

        { label: "Date of Adoption / Child Handover", name: "adoptionDate", type: "date", required: true }

    ] : [];

    const remainingFields = [

        { label: "Number of Surviving Children (excluding current pregnancy/adoption)", name: "numberOfSurvivingChildren", type: "number", min: "0", required: true },

        { label: "Primary Issue Faced", name: "issueType", type: "select", options: [

            {value: "denial\_of\_leave", label: "Complete Denial of Maternity Leave"},

            {value: "partial\_leave", label: "Partial Leave Granted (Less than entitled)"},

            {value: "non\_payment\_salary", label: "Non-payment or Partial Payment of Salary During Leave"},

            {value: "crèche\_denial", label: "Denial or Lack of Crèche Facility (if applicable)"},

            {value: "termination\_dismissal", label: "Termination or Dismissal Due to Pregnancy/Maternity"},

            {value: "work\_from\_home\_denial", label: "Denial of Work From Home Option (if applicable post leave)"},

            {value: "coercion\_to\_resign", label: "Coercion to Resign"},

            {value: "unfavorable\_treatment", label: "Unfavorable Treatment Post Resuming Work"},

            {value: "other", label: "Other (Please specify in details)"}

        ], required: true },

        { label: "Detailed Description of the Issue", name: "issueDetails", type: "textarea", rows: 4, required: true, placeholder: "Explain the problem in detail. What happened, when, and who was involved?" },

        { label: "Chronology of Events", name: "chronologyOfEvents", type: "textarea", rows: 4, required: true, placeholder: "List the key dates and events in order. E.g., Applied for leave on DD/MM/YYYY, Received response on DD/MM/YYYY..." },

        { label: "Information about Supporting Documents", name: "supportingDocumentsInfo", type: "textarea", rows: 3, placeholder: "List any documents you have (e.g., employment contract, leave application, emails, medical certificates). Actual document upload will be handled separately if required by lawyers." },

    ];

    return (

        <div className="bg-white p-6 md:p-8 rounded-xl shadow-2xl max-w-3xl mx-auto">

            <h2 className="text-2xl md:text-3xl font-bold text-indigo-700 mb-6 text-center">File a New Complaint</h2>

            <p className="text-sm text-gray-600 mb-6 text-center">Please fill out this form with as much detail as possible. All information will be kept confidential and used solely for the purpose of addressing your grievance.</p>

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

                {[...formFields, ...conditionalFields, ...remainingFields].map(field => (

                    <div key={field.name}>

                        <label htmlFor={field.name} className="block text-sm font-medium text-gray-700 mb-1">

                            {field.label} {field.required && <span className="text-red-500">\*</span>}

                        </label>

                        {field.type === "textarea" ? (

                            <textarea

                                id={field.name}

                                name={field.name}

                                value={formData[field.name]}

                                onChange={handleChange}

                                rows={field.rows || 3}

                                required={field.required}

                                placeholder={field.placeholder}

                                className="w-full px-3 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            />

                        ) : field.type === "select" ? (

                            <select

                                id={field.name}

                                name={field.name}

                                value={formData[field.name]}

                                onChange={handleChange}

                                required={field.required}

                                className="w-full px-3 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors bg-white"

                            >

                                {field.options.map(opt => <option key={opt.value} value={opt.value}>{opt.label}</option>)}

                            </select>

                        ) : (

                            <input

                                type={field.type}

                                id={field.name}

                                name={field.name}

                                value={formData[field.name]}

                                onChange={handleChange}

                                required={field.required}

                                min={field.min}

                                placeholder={field.placeholder}

                                className="w-full px-3 py-2 border border-gray-300 rounded-lg shadow-sm focus:ring-indigo-500 focus:border-indigo-500 transition-colors"

                            />

                        )}

                    </div>

                ))}

                <div className="pt-4">

                    <label htmlFor="consentToShare" className="flex items-start">

                        <input

                            type="checkbox"

                            id="consentToShare"

                            name="consentToShare"

                            checked={formData.consentToShare}

                            onChange={handleChange}

                            required

                            className="h-5 w-5 text-indigo-600 border-gray-300 rounded focus:ring-indigo-500 mt-0.5"

                        />

                        <span className="ml-2 text-sm text-gray-700">

                            I consent to share the information provided in this form with empaneled legal professionals for the purpose of preparing a legal notice and assisting with my maternity benefit claim. I understand this information will be handled confidentially. <span className="text-red-500">\*</span>

                        </span>

                    </label>

                </div>

                <div className="flex justify-end space-x-4 pt-6">

                     <button

                        type="button"

                        onClick={() => setPage('dashboard')}

                        className="px-6 py-2 border border-gray-300 rounded-lg text-sm font-medium text-gray-700 hover:bg-gray-50 transition-colors"

                    >

                        Cancel

                    </button>

                    <button

                        type="submit"

                        disabled={isSubmitting}

                        className="px-8 py-3 bg-indigo-600 hover:bg-indigo-700 text-white font-semibold rounded-lg shadow-md focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-offset-2 transition-all duration-300 disabled:opacity-50"

                    >

                        {isSubmitting ? 'Submitting...' : 'Submit Complaint'}

                    </button>

                </div>

            </form>

        </div>

    );

};

// --- ViewComplaintsPage Component ---

const ViewComplaintsPage = ({ db, user, appId, userId, isAuthReady, setModalMessage, setModalType }) => {

    const [complaints, setComplaints] = useState([]);

    const [isLoading, setIsLoading] = useState(true);

    const [selectedComplaint, setSelectedComplaint] = useState(null); // For modal view

    useEffect(() => {

        if (!db || !user || !userId || !isAuthReady) {

            if (isAuthReady) setIsLoading(false); // Only set loading false if auth is ready but other deps are missing

            return;

        }

        setIsLoading(true);

        // Path: /artifacts/{appId}/users/{userId}/complaints

        const complaintsCollectionPath = `artifacts/${appId}/users/${userId}/complaints`;

        const q = query(collection(db, complaintsCollectionPath), orderBy("submittedAt", "desc")); // Order by submission time

        const unsubscribe = onSnapshot(q, (querySnapshot) => {

            const complaintsData = [];

            querySnapshot.forEach((doc) => {

                complaintsData.push({ id: doc.id, ...doc.data() });

            });

            setComplaints(complaintsData);

            setIsLoading(false);

        }, (error) => {

            console.error("Error fetching complaints: ", error);

            setModalMessage(`Error fetching complaints: ${error.message}`);

            setModalType('error');

            setIsLoading(false);

        });

        return () => unsubscribe();

    }, [db, user, appId, userId, isAuthReady, setModalMessage, setModalType]); // Added missing dependencies

    const formatDate = (timestamp) => {

        if (!timestamp) return 'N/A';

        return timestamp.toDate().toLocaleDateString('en-IN', { year: 'numeric', month: 'long', day: 'numeric', hour: '2-digit', minute: '2-digit' });

    };

    const getStatusColor = (status) => {

        switch (status?.toLowerCase()) {

            case 'submitted': return 'bg-blue-100 text-blue-800';

            case 'in review': return 'bg-yellow-100 text-yellow-800';

            case 'action taken': return 'bg-green-100 text-green-800';

            case 'closed': return 'bg-gray-100 text-gray-800';

            default: return 'bg-gray-100 text-gray-800';

        }

    };

    if (isLoading) {

        return <div className="text-center py-10 text-indigo-600">Loading your complaints...</div>;

    }

    if (!user && isAuthReady) { // Check if user is null explicitly when auth is ready

         return <div className="text-center py-10 text-red-600">Please log in to view your complaints.</div>;

    }

    return (

        <div className="bg-white p-6 md:p-8 rounded-xl shadow-2xl max-w-4xl mx-auto">

            <h2 className="text-2xl md:text-3xl font-bold text-indigo-700 mb-6 text-center">My Submitted Complaints</h2>

            {complaints.length === 0 ? (

                <p className="text-gray-600 text-center py-8">You have not submitted any complaints yet.</p>

            ) : (

                <div className="space-y-6">

                    {complaints.map((complaint) => (

                        <div key={complaint.id} className="border border-gray-200 rounded-lg p-4 md:p-6 shadow-sm hover:shadow-md transition-shadow">

                            <div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-3">

                                <h3 className="text-lg font-semibold text-indigo-600">Complaint ID: ...{complaint.id.slice(-8)}</h3>

                                <span className={`text-xs font-medium mr-2 px-2.5 py-0.5 rounded-full ${getStatusColor(complaint.status)}`}>

                                    Status: {complaint.status || 'N/A'}

                                </span>

                            </div>

                            <p className="text-sm text-gray-500 mb-1"><strong>Submitted:</strong> {formatDate(complaint.submittedAt)}</p>

                            <p className="text-sm text-gray-700 mb-1"><strong>Company:</strong> {complaint.companyName}</p>

                            <p className="text-sm text-gray-700 mb-3"><strong>Issue Type:</strong> {complaint.issueType?.replace(/\_/g, ' ').replace(/\b\w/g, l => l.toUpperCase())}</p>

                            <button

                                onClick={() => setSelectedComplaint(complaint)}

                                className="text-sm text-indigo-600 hover:text-indigo-800 font-medium transition-colors"

                            >

                                View Details

                            </button>

                        </div>

                    ))}

                </div>

            )}

            {selectedComplaint && (

                <ComplaintDetailsModal complaint={selectedComplaint} onClose={() => setSelectedComplaint(null)} formatDate={formatDate} />

            )}

        </div>

    );

};

// --- ComplaintDetailsModal Component ---

const ComplaintDetailsModal = ({ complaint, onClose, formatDate }) => {

    return (

        <div className="fixed inset-0 bg-gray-600 bg-opacity-75 overflow-y-auto h-full w-full z-50 flex justify-center items-center p-4">

            <div className="relative bg-white rounded-lg shadow-xl w-full max-w-2xl max-h-[90vh] flex flex-col">

                <div className="flex justify-between items-center p-4 border-b">

                    <h3 className="text-xl font-semibold text-indigo-700">Complaint Details (ID: ...{complaint.id.slice(-8)})</h3>

                    <button onClick={onClose} className="text-gray-400 hover:text-gray-600">

                        <svg className="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M6 18L18 6M6 6l12 12"></path></svg>

                    </button>

                </div>

                <div className="p-6 space-y-4 overflow-y-auto">

                    {Object.entries(complaint).map(([key, value]) => {

                        if (key === 'id' || key === 'userId' || key === 'userEmail') return null; // Skip internal IDs

                        let displayValue = value;

                        if (key === 'submittedAt' || key === 'expectedDeliveryDate' || key === 'actualDeliveryDate' || key === 'dateOfJoining' || key === 'adoptionDate') {

                            displayValue = value ? (value.toDate ? formatDate(value) : value.toString()) : 'N/A';

                        } else if (typeof value === 'boolean') {

                            displayValue = value ? 'Yes' : 'No';

                        } else if (value === '' || value === null || typeof value === 'undefined') {

                            displayValue = 'N/A';

                        } else if (typeof value === 'object' && value !== null && !value.toDate) { // Avoid trying to stringify Timestamps again

                            displayValue = JSON.stringify(value);

                        }

                        const formattedKey = key.replace(/([A-Z])/g, ' $1').replace(/^./, str => str.toUpperCase());

                        return (

                            <div key={key} className="grid grid-cols-3 gap-2 border-b border-gray-100 pb-2">

                                <p className="text-sm font-medium text-gray-600 col-span-1">{formattedKey}:</p>

                                <p className="text-sm text-gray-800 col-span-2 break-words">{String(displayValue)}</p>

                            </div>

                        );

                    })}

                </div>

                 <div className="p-4 border-t flex justify-end">

                    <button

                        onClick={onClose}

                        className="px-6 py-2 bg-indigo-600 hover:bg-indigo-700 text-white text-sm font-medium rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

                    >

                        Close

                    </button>

                </div>

            </div>

        </div>

    );

};

export default App;