This document outlines a full-stack version of the maternity complaint portal using a traditional React frontend + Express backend + MongoDB setup.

**Recommended Folder & File Structure**

maternityportal/

├── client/ # React frontend (e.g., created with Vite or Create React App)

│ ├── public/

│ ├── src/

│ │ ├── assets/ # For static assets like images, global CSS

│ │ ├── components/ # Reusable UI components

│ │ │ ├── Navbar.jsx

│ │ │ ├── Footer.jsx

│ │ │ ├── Modal.jsx

│ │ │ └── ProtectedRoute.jsx

│ │ ├── contexts/ # For global state (e.g., AuthContext)

│ │ │ └── AuthContext.jsx

│ │ ├── pages/ # Top-level page components

│ │ │ ├── HomePage.jsx

│ │ │ ├── RegisterPage.jsx

│ │ │ ├── LoginPage.jsx

│ │ │ ├── RequestPasswordResetPage.jsx

│ │ │ ├── ResetPasswordPage.jsx

│ │ │ ├── DashboardPage.jsx

│ │ │ ├── NewComplaintPage.jsx

│ │ │ └── ViewComplaintsPage.jsx

│ │ ├── services/ # API call functions (optional, can be in context or components)

│ │ │ ├── authService.js // Example, if not handled in AuthContext

│ │ │ └── complaintService.js // Example

│ │ ├── App.jsx # Main application component with routing

│ │ └── main.jsx # Or index.js - Entry point for React app

│ ├── .env # Frontend environment variables (e.g., API base URL)

│ ├── package.json

│ └── vite.config.js # Or craco.config.js, etc. for Create React App

├── server/ # Express backend

│ ├── config/ # Configuration files (e.g., db connection)

│ │ └── db.js # (Optional: Mongoose connection logic)

│ ├── middleware/

│ │ └── verifyToken.js # JWT verification middleware

│ ├── models/

│ │ ├── User.js

│ │ └── Complaint.js

│ ├── routes/

│ │ ├── auth.js

│ │ └── complaints.js

│ ├── .env # Backend environment variables (MONGO\_URI, JWT\_SECRET, PORT, EMAIL\_USER, EMAIL\_PASS, CLIENT\_URL)

│ ├── index.js # Main server entry point

│ └── package.json

├── .gitignore

└── README.md

**Backend (server/)**

**server/.env (Example)**

Create this file in the server/ directory. **Do not commit this file to version control.**

PORT=5000

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_very\_strong\_jwt\_secret\_key\_at\_least\_32\_characters\_long

EMAIL\_USER=your\_email\_address\_for\_sending\_password\_resets@example.com

EMAIL\_PASS=your\_email\_password\_or\_app\_specific\_password

EMAIL\_SERVICE=gmail # Or your email provider service name (e.g., SendGrid, Mailgun)

CLIENT\_URL=http://localhost:5173 # Or your frontend's URL for password reset links

NODE\_ENV=development

**server/package.json (Example Dependencies)**

You'll need to install these: npm install express mongoose cors dotenv bcrypt jsonwebtoken nodemailer helmet express-rate-limit express-validator morgan For development: npm install -D nodemon

{

"name": "maternity-portal-server",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"start": "node index.js",

"dev": "nodemon index.js"

},

"dependencies": {

"bcrypt": "^5.1.1",

"cors": "^2.8.5",

"dotenv": "^16.3.1",

"express": "^4.18.2",

"express-rate-limit": "^7.1.5",

"express-validator": "^7.0.1",

"helmet": "^7.1.0",

"jsonwebtoken": "^9.0.2",

"mongoose": "^8.0.3",

"morgan": "^1.10.0",

"nodemailer": "^6.9.7"

},

"devDependencies": {

"nodemon": "^3.0.2"

}

}

**server/index.js**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const dotenv = require('dotenv');

const helmet = require('helmet');

const rateLimit = require('express-rate-limit');

const morgan = require('morgan');

const authRoutes = require('./routes/auth');

const complaintRoutes = require('./routes/complaints');

// Load environment variables

dotenv.config();

const app = express();

// Security Middlewares

app.use(helmet()); // Sets various security-related HTTP headers

// Rate Limiting - Apply to API routes

const apiLimiter = rateLimit({

windowMs: 15 \* 60 \* 1000, // 15 minutes

max: 100, // limit each IP to 100 requests per windowMs for API routes

standardHeaders: true, // Return rate limit info in the `RateLimit-\*` headers

legacyHeaders: false, // Disable the `X-RateLimit-\*` headers

message: 'Too many requests from this IP for API access, please try again later.'

});

app.use('/api/', apiLimiter); // Apply rate limiting specifically to /api routes

// Core Middlewares

app.use(cors({

origin: process.env.CLIENT\_URL || 'http://localhost:5173', // Allow requests from your client

optionsSuccessStatus: 200

}));

app.use(express.json()); // Parses incoming requests with JSON payloads

app.use(express.urlencoded({ extended: true })); // Parses incoming requests with URL-encoded payloads

// Logging Middleware

if (process.env.NODE\_ENV === 'development') {

app.use(morgan('dev')); // Logs requests to the console in 'dev' format

}

// API Routes

app.use('/api/auth', authRoutes);

app.use('/api/complaints', complaintRoutes);

// Basic route for testing server status

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

res.send('Maternity Benefit Portal API is alive and running!');

});

// Database Connection

mongoose.connect(process.env.MONGO\_URI)

.then(() => {

console.log('Successfully connected to MongoDB');

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

}).catch(err => {

console.error('MongoDB connection error. Please ensure MongoDB is running and MONGO\_URI is set correctly.');

console.error(err.message);

process.exit(1); // Exit process with failure

});

// Global Error Handler Middleware

// This should be the last piece of middleware added

app.use((err, req, res, next) => {

console.error('Unhandled Error:', err.stack || err);

res.status(err.status || 500).json({

error: err.message || 'An unexpected error occurred on the server.'

});

});

**server/models/User.js**

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const UserSchema = new mongoose.Schema({

email: {

type: String,

required: [true, 'Email is required.'],

unique: true,

trim: true,

lowercase: true,

// Simple regex for email validation, consider a more robust one or a library if needed

match: [/.+@.+\..+/, 'Please enter a valid email address.']

},

password: {

type: String,

required: [true, 'Password is required.'],

minlength: [6, 'Password must be at least 6 characters long.']

},

createdAt: {

type: Date,

default: Date.now

}

});

// Hash password before saving

UserSchema.pre('save', async function (next) {

// Only hash the password if it has been modified (or is new)

if (!this.isModified('password')) return next();

try {

const salt = await bcrypt.genSalt(10);

this.password = await bcrypt.hash(this.password, salt);

next();

} catch (error) {

next(error); // Pass errors to the next middleware

}

});

// Method to compare password for login

UserSchema.methods.comparePassword = async function(candidatePassword) {

try {

return await bcrypt.compare(candidatePassword, this.password);

} catch (error) {

throw error; // Or handle error appropriately

}

};

module.exports = mongoose.model('User', UserSchema);

**server/models/Complaint.js**

const mongoose = require('mongoose');

const ComplaintSchema = new mongoose.Schema({

userId: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User', // Reference to the User model

required: true

},

complainantName: { type: String, trim: true, required: [true, 'Complainant name is required.'] },

complainantContact: { type: String, trim: true, required: [true, 'Contact number is required.'] },

companyName: { type: String, trim: true, required: [true, 'Company name is required.'] },

companyAddress: { type: String, trim: true, required: [true, 'Company address is required.'] },

dateOfJoining: { type: Date, required: [true, 'Date of joining is required.'] },

employmentType: { type: String, required: [true, 'Employment type is required.'] },

expectedDeliveryDate: Date,

actualDeliveryDate: Date,

isAdoptiveOrCommissioning: { type: String, required: [true, 'Please specify if adoptive/commissioning mother.'] },

adoptionDate: Date, // Required if isAdoptiveOrCommissioning is 'yes'

numberOfSurvivingChildren: { type: Number, min: 0, required: [true, 'Number of surviving children is required.'] },

issueType: { type: String, required: [true, 'Issue type is required.'] },

issueDetails: { type: String, trim: true, required: [true, 'Issue details are required.'] },

chronologyOfEvents: { type: String, trim: true, required: [true, 'Chronology of events is required.'] },

supportingDocumentsInfo: { type: String, trim: true },

consentToShare: { type: Boolean, required: [true, 'Consent to share information is required.'] },

status: {

type: String,

default: 'submitted',

enum: ['submitted', 'in\_review', 'action\_taken', 'resolved', 'closed']

},

submittedAt: { type: Date, default: Date.now },

updatedAt: { type: Date, default: Date.now }

});

// Middleware to update `updatedAt` field before saving

ComplaintSchema.pre('save', function(next) {

this.updatedAt = Date.now();

next();

});

// Conditional validation for adoptionDate

ComplaintSchema.path('adoptionDate').validate(function(value) {

if (this.isAdoptiveOrCommissioning === 'yes') {

return value != null; // adoptionDate is required if 'yes'

}

return true; // Not required otherwise

}, 'Adoption date is required if you are an adoptive or commissioning mother.');

module.exports = mongoose.model('Complaint', ComplaintSchema);

**server/middleware/verifyToken.js**

const jwt = require('jsonwebtoken');

function verifyToken(req, res, next) {

const authHeader = req.headers.authorization;

if (authHeader && authHeader.startsWith('Bearer ')) {

const token = authHeader.split(' ')[1]; // Extract token from "Bearer TOKEN"

if (!token) {

// This case should ideally not be hit if header.startsWith is true and split works,

// but as a safeguard:

return res.status(401).json({ error: 'Access denied. No token provided.' });

}

jwt.verify(token, process.env.JWT\_SECRET, (err, decoded) => {

if (err) {

if (err.name === 'TokenExpiredError') {

return res.status(401).json({ error: 'Token expired. Please log in again.' });

}

// For other errors like JsonWebTokenError (malformed token)

return res.status(401).json({ error: 'Invalid token. Authentication failed.' });

}

// If token is valid, attach decoded payload (which includes user ID) to request object

req.userId = decoded.id;

req.userEmail = decoded.email; // If you included email in JWT payload

next(); // Proceed to the next middleware or route handler

});

} else {

// No Authorization header or not in Bearer format

return res.status(401).json({ error: 'Access denied. Authorization header missing or malformed.' });

}

}

module.exports = verifyToken;

**server/routes/auth.js**

const express = require('express');

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

const jwt = require('jsonwebtoken');

// const bcrypt = require('bcrypt'); // Not directly used here as comparePassword is in User model

const { body, validationResult } = require('express-validator');

const crypto = require('crypto');

const nodemailer = require('nodemailer');

const router = express.Router();

// In-memory store for reset tokens.

// WARNING: This is NOT suitable for production. Use a database (e.g., Redis or a MongoDB collection) for this.

const resetTokens = new Map();

// POST /api/auth/register

router.post('/register', [

body('email').isEmail().withMessage('Please enter a valid email address.').normalizeEmail(),

body('password').isLength({ min: 6 }).withMessage('Password must be at least 6 characters long.')

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

try {

const { email, password } = req.body;

let existingUser = await User.findOne({ email });

if (existingUser) {

return res.status(400).json({ errors: [{ msg: 'User with this email already exists.' }] });

}

const newUser = new User({ email, password });

await newUser.save();

res.status(201).json({ message: 'User registered successfully. Please proceed to login.' });

} catch (err) {

next(err); // Pass to global error handler

}

});

// POST /api/auth/login

router.post('/login', [

body('email').isEmail().withMessage('Please enter a valid email address.').normalizeEmail(),

body('password').notEmpty().withMessage('Password is required.')

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

const { email, password } = req.body;

try {

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

if (!user) {

return res.status(401).json({ error: 'Invalid email or password.' });

}

const isMatch = await user.comparePassword(password);

if (!isMatch) {

return res.status(401).json({ error: 'Invalid email or password.' });

}

const payload = {

id: user.\_id,

email: user.email

// Add other non-sensitive info if needed

};

const token = jwt.sign(payload, process.env.JWT\_SECRET, { expiresIn: '1d' }); // Token expires in 1 day

res.json({

token,

userId: user.\_id,

email: user.email

});

} catch (err) {

next(err);

}

});

// POST /api/auth/request-reset

router.post('/request-reset', [

body('email').isEmail().withMessage('Please enter a valid email address.').normalizeEmail()

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

const { email } = req.body;

try {

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

if (!user) {

// It's often better practice not to reveal if an email exists or not for security reasons.

// You could send a generic success message regardless.

// For this example, we'll proceed assuming it's okay to inform the user.

return res.status(404).json({ error: 'No user found with that email address.' });

}

const token = crypto.randomBytes(32).toString('hex');

const expires = Date.now() + 3600000; // Token expires in 1 hour (3600000 ms)

resetTokens.set(token, { userId: user.\_id.toString(), expires }); // Store userId as string

const transporter = nodemailer.createTransport({

service: process.env.EMAIL\_SERVICE || 'gmail', // Ensure EMAIL\_SERVICE is set in .env

auth: {

user: process.env.EMAIL\_USER,

pass: process.env.EMAIL\_PASS

},

// Adding timeout for safety

connectionTimeout: 10000, // 10 seconds

greetingTimeout: 10000,

socketTimeout: 10000,

});

const resetUrl = `${process.env.CLIENT\_URL || 'http://localhost:5173'}/reset-password?token=${token}`;

await transporter.sendMail({

from: `"Maternity Justice Portal" <${process.env.EMAIL\_USER}>`,

to: email,

subject: 'Password Reset Request - Maternity Justice Portal',

text: `You are receiving this email because you (or someone else) have requested a password reset for your account on Maternity Justice Portal.\n\nPlease click on the following link, or paste it into your browser to complete the process within one hour:\n\n${resetUrl}\n\nIf you did not request this, please ignore this email and your password will remain unchanged.\n`,

html: `<p>You are receiving this email because you (or someone else) have requested a password reset for your account on Maternity Justice Portal.</p>

<p>Please click on the following link, or paste it into your browser to complete the process within one hour:</p>

<p><a href="${resetUrl}" target="\_blank" rel="noopener noreferrer">${resetUrl}</a></p>

<p>If you did not request this, please ignore this email and your password will remain unchanged.</p>`

});

res.json({ message: 'A password reset link has been sent to your email address. Please check your inbox (and spam folder).' });

} catch (err) {

console.error('Error in /request-reset:', err);

if (err.code === 'EAUTH' || err.command === 'CONN') { // Check for common nodemailer errors

return res.status(500).json({ error: 'Failed to send email due to server email configuration or connection issue. Please try again later or contact support.' });

}

next(err);

}

});

// POST /api/auth/reset-password

router.post('/reset-password', [

body('token').notEmpty().withMessage('Reset token is required.'),

body('newPassword').isLength({ min: 6 }).withMessage('New password must be at least 6 characters long.')

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

const { token, newPassword } = req.body;

try {

const tokenData = resetTokens.get(token);

if (!tokenData || tokenData.expires < Date.now()) {

if(tokenData) resetTokens.delete(token); // Clean up used/expired token from memory

return res.status(400).json({ error: 'Password reset token is invalid or has expired. Please request a new one.' });

}

const user = await User.findById(tokenData.userId);

if (!user) {

resetTokens.delete(token); // Clean up

return res.status(404).json({ error: 'User associated with this token could not be found.' });

}

user.password = newPassword; // The pre-save hook in User.js model will hash it

await user.save();

resetTokens.delete(token); // Token has been successfully used

res.json({ message: 'Your password has been successfully reset. You can now log in with your new password.' });

} catch (err) {

next(err);

}

});

module.exports = router;

**server/routes/complaints.js**

const express = require('express');

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

const verifyToken = require('../middleware/verifyToken'); // Import the middleware

const { body, validationResult, check } = require('express-validator');

const router = express.Router();

// Validation rules for creating/updating a complaint

const complaintValidationRules = [

body('complainantName').notEmpty().withMessage('Complainant name is required.').trim().escape(),

body('complainantContact').notEmpty().withMessage('Contact number is required.').trim().escape(), // Add specific phone validation if needed

body('companyName').notEmpty().withMessage('Company name is required.').trim().escape(),

body('companyAddress').notEmpty().withMessage('Company address is required.').trim().escape(),

body('dateOfJoining').isISO8601().toDate().withMessage('Valid date of joining is required.'),

body('employmentType').notEmpty().withMessage('Employment type is required.').trim().escape(),

body('expectedDeliveryDate').optional({ checkFalsy: true }).isISO8601().toDate().withMessage('Valid expected delivery date required if provided.'),

body('actualDeliveryDate').optional({ checkFalsy: true }).isISO8601().toDate().withMessage('Valid actual delivery date required if provided.'),

body('isAdoptiveOrCommissioning').isIn(['yes', 'no']).withMessage('Specify if adoptive/commissioning mother.'),

body('adoptionDate').custom((value, { req }) => {

if (req.body.isAdoptiveOrCommissioning === 'yes' && !value) {

throw new Error('Adoption date is required if adoptive/commissioning mother.');

}

if (value && !/^\d{4}-\d{2}-\d{2}$/.test(value)) { // Basic YYYY-MM-DD check

throw new Error('Adoption date must be a valid date (YYYY-MM-DD).');

}

return true;

}).optional({ checkFalsy: true }).toDate(),

body('numberOfSurvivingChildren').isInt({ min: 0 }).withMessage('Number of surviving children must be a non-negative integer.'),

body('issueType').notEmpty().withMessage('Issue type is required.').trim().escape(),

body('issueDetails').notEmpty().withMessage('Issue details are required.').trim().escape(),

body('chronologyOfEvents').notEmpty().withMessage('Chronology of events is required.').trim().escape(),

body('supportingDocumentsInfo').optional().trim().escape(),

body('consentToShare').isBoolean().withMessage('Consent to share must be true or false.').custom(value => {

if (value !== true) {

throw new Error('You must consent to share your information to proceed.');

}

return true;

}),

];

// POST /api/complaints/ - Create a new complaint

router.post('/', verifyToken, complaintValidationRules, async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

try {

// req.userId is attached by verifyToken middleware

const complaintData = { ...req.body, userId: req.userId };

const complaint = new Complaint(complaintData);

await complaint.save();

res.status(201).json({ message: 'Complaint submitted successfully!', complaintId: complaint.\_id, complaint });

} catch (err) {

next(err); // Pass to global error handler

}

});

// GET /api/complaints/ - Get all complaints for the logged-in user

router.get('/', verifyToken, async (req, res, next) => {

try {

// req.userId is attached by verifyToken middleware

const complaints = await Complaint.find({ userId: req.userId }).sort({ submittedAt: -1 });

// No need to check if !complaints, an empty array is a valid response if no complaints exist.

res.json(complaints);

} catch (err) {

next(err);

}

});

// GET /api/complaints/:id - Get a specific complaint by ID, ensuring it belongs to the user

router.get('/:id', verifyToken, [

check('id').isMongoId().withMessage('Invalid complaint ID format.')

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

try {

const complaint = await Complaint.findOne({ \_id: req.params.id, userId: req.userId });

if (!complaint) {

return res.status(404).json({ message: 'Complaint not found or you do not have permission to view it.' });

}

res.json(complaint);

} catch (err) {

// Catching specific Mongoose CastError for invalid ID format, though validator should catch it first

if (err.name === 'CastError' && err.kind === 'ObjectId') {

return res.status(400).json({ message: 'Invalid complaint ID format.' });

}

next(err);

}

});

// PUT /api/complaints/:id - Update a specific complaint

// Users should typically only be allowed to update certain fields or before a certain status.

// For simplicity, this allows updating fields defined in complaintValidationRules.

router.put('/:id', verifyToken, [

check('id').isMongoId().withMessage('Invalid complaint ID format.'),

...complaintValidationRules // Reuse validation, but ensure they are appropriate for update

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

try {

const complaint = await Complaint.findOne({ \_id: req.params.id, userId: req.userId });

if (!complaint) {

return res.status(404).json({ message: 'Complaint not found or you do not have permission to edit it.' });

}

// Prevent status updates by user directly through this route if needed

if (req.body.status && req.body.status !== complaint.status) {

// return res.status(403).json({ error: "User cannot change complaint status directly." });

// Or, simply ignore status changes from user input for this route:

delete req.body.status;

}

// Update allowed fields

Object.keys(req.body).forEach(key => {

if (key !== 'userId' && key !== '\_id') { // Prevent updating userId or \_id

complaint[key] = req.body[key];

}

});

complaint.updatedAt = Date.now(); // Manually set updatedAt

const updatedComplaint = await complaint.save();

res.json({ message: 'Complaint updated successfully.', complaint: updatedComplaint });

} catch (err) {

if (err.name === 'CastError' && err.kind === 'ObjectId') {

return res.status(400).json({ message: 'Invalid complaint ID format.' });

}

next(err);

}

});

// DELETE /api/complaints/:id - Delete a specific complaint

// Consider if users should be able to delete complaints or if it should be a soft delete.

router.delete('/:id', verifyToken, [

check('id').isMongoId().withMessage('Invalid complaint ID format.')

], async (req, res, next) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

try {

const complaint = await Complaint.findOneAndDelete({ \_id: req.params.id, userId: req.userId });

if (!complaint) {

return res.status(404).json({ message: 'Complaint not found or you do not have permission to delete it.' });

}

res.json({ message: 'Complaint deleted successfully.' });

} catch (err) {

if (err.name === 'CastError' && err.kind === 'ObjectId') {

return res.status(400).json({ message: 'Invalid complaint ID format.' });

}

next(err);

}

});

module.exports = router;

**Frontend (client/)**

**client/.env (Example for Vite)**

Create this file in the client/ directory.

VITE\_API\_BASE\_URL=http://localhost:5000/api

**client/package.json (Example Dependencies)**

Install: npm install react react-dom react-router-dom axios Dev Install: npm install -D tailwindcss postcss autoprefixer @vitejs/plugin-react vite (if using Vite)

{

"name": "maternity-portal-client",

"version": "0.1.0",

"private": true,

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"preview": "vite preview"

},

"dependencies": {

"axios": "^1.6.2",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-router-dom": "^6.21.1"

},

"devDependencies": {

"@types/react": "^18.2.43",

"@types/react-dom": "^18.2.17",

"@vitejs/plugin-react": "^4.2.1",

"autoprefixer": "^10.4.16",

"postcss": "^8.4.32",

"tailwindcss": "^3.4.0",

"vite": "^5.0.10"

}

}

**client/src/main.jsx (or index.js)**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App.jsx';

import './index.css'; // Your global styles, including Tailwind directives

import { BrowserRouter } from 'react-router-dom';

import { AuthProvider } from './contexts/AuthContext.jsx';

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<BrowserRouter>

<AuthProvider>

<App />

</AuthProvider>

</BrowserRouter>

</React.StrictMode>,

);

**client/src/App.jsx**

import React from 'react';

import { Routes, Route, Navigate } from 'react-router-dom';

import Navbar from './components/Navbar';

import Footer from './components/Footer';

import ProtectedRoute from './components/ProtectedRoute';

import HomePage from './pages/HomePage';

import LoginPage from './pages/LoginPage';

import RegisterPage from './pages/RegisterPage';

import RequestPasswordResetPage from './pages/RequestPasswordResetPage';

import ResetPasswordPage from './pages/ResetPasswordPage';

import DashboardPage from './pages/DashboardPage';

import NewComplaintPage from './pages/NewComplaintPage';

import ViewComplaintsPage from './pages/ViewComplaintsPage';

import { useAuth } from './contexts/AuthContext';

function App() {

const { currentUser, loading } = useAuth();

if (loading) {

return (

<div className="flex justify-center items-center min-h-screen bg-gradient-to-br from-indigo-100 via-purple-50 to-pink-100">

<p className="text-xl font-semibold text-indigo-600">Loading Application...</p>

</div>

);

}

return (

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

<Navbar /> {/\* Navbar will use useAuth internally \*/}

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

<Routes>

{/\* Public Routes \*/}

<Route path="/" element={currentUser ? <Navigate to="/dashboard" replace /> : <HomePage />} />

<Route path="/login" element={!currentUser ? <LoginPage /> : <Navigate to="/dashboard" replace />} />

<Route path="/register" element={!currentUser ? <RegisterPage /> : <Navigate to="/dashboard" replace />} />

<Route path="/request-password-reset" element={<RequestPasswordResetPage />} />

<Route path="/reset-password" element={<ResetPasswordPage />} /> {/\* Token is in query param \*/}

{/\* Protected Routes \*/}

<Route path="/dashboard" element={<ProtectedRoute><DashboardPage /></ProtectedRoute>} />

<Route path="/new-complaint" element={<ProtectedRoute><NewComplaintPage /></ProtectedRoute>} />

<Route path="/complaints" element={<ProtectedRoute><ViewComplaintsPage /></ProtectedRoute>} />

{/\* Add specific complaint view route if needed: /complaints/:id \*/}

{/\* Fallback 404 Not Found route \*/}

<Route path="\*" element={

<div className="text-center py-10">

<h2 className="text-3xl font-bold text-indigo-700">404: Page Not Found</h2>

<p className="text-gray-600 mt-2">Sorry, the page you are looking for does not exist.</p>

<Link to="/" className="mt-4 inline-block bg-indigo-600 text-white px-6 py-2 rounded hover:bg-indigo-700">

Go to Homepage

</Link>

</div>

} />

</Routes>

</main>

<Footer />

</div>

);

}

export default App;

**client/src/contexts/AuthContext.jsx**

import React, { createContext, useContext, useState, useEffect, useCallback } from 'react';

import axios from 'axios';

const API\_BASE\_URL = import.meta.env.VITE\_API\_BASE\_URL || 'http://localhost:5000/api';

const AuthContext = createContext(null);

export function useAuth() {

return useContext(AuthContext);

}

export function AuthProvider({ children }) {

const [currentUser, setCurrentUser] = useState(null);

const [token, setToken] = useState(localStorage.getItem('token'));

const [loading, setLoading] = useState(true);

const authAxios = axios.create({

baseURL: API\_BASE\_URL,

});

authAxios.interceptors.request.use(

(config) => {

const storedToken = localStorage.getItem('token');

if (storedToken) {

config.headers['Authorization'] = `Bearer ${storedToken}`;

}

return config;

},

(error) => Promise.reject(error)

);

const verifyTokenAndSetUser = useCallback(async (userToken) => {

if (userToken) {

// Ideally, you'd have a /api/auth/me endpoint to verify token and get user details

// For simplicity, if token exists, assume it's valid and decode (not secure for production without verification)

// Or, retrieve stored user details.

const storedUserEmail = localStorage.getItem('userEmail');

const storedUserId = localStorage.getItem('userId');

if (storedUserEmail && storedUserId) {

setCurrentUser({ id: storedUserId, email: storedUserEmail, token: userToken });

} else {

// If details are missing, token might be stale or login was incomplete

console.warn("Token found but user details missing from localStorage. Logging out.");

localStorage.removeItem('token');

localStorage.removeItem('userEmail');

localStorage.removeItem('userId');

setToken(null);

setCurrentUser(null);

}

}

setLoading(false);

}, []);

useEffect(() => {

const storedToken = localStorage.getItem('token');

if (storedToken) {

setToken(storedToken); // Set token state

verifyTokenAndSetUser(storedToken); // Verify and set user

} else {

setLoading(false); // No token, not loading

}

}, [verifyTokenAndSetUser]);

const login = async (email, password) => {

try {

const response = await axios.post(`${API\_BASE\_URL}/auth/login`, { email, password });

const { token: receivedToken, userId, email: userEmail } = response.data;

localStorage.setItem('token', receivedToken);

localStorage.setItem('userId', userId);

localStorage.setItem('userEmail', userEmail);

setToken(receivedToken);

setCurrentUser({ id: userId, email: userEmail, token: receivedToken });

return response.data; // Return data for potential use in component

} catch (error) {

console.error("Login error in AuthContext:", error.response?.data || error.message);

throw error.response?.data || new Error(error.message || 'Login failed due to an unknown error.');

}

};

const register = async (email, password) => {

try {

const response = await axios.post(`${API\_BASE\_URL}/auth/register`, { email, password });

return response.data;

} catch (error) {

console.error("Registration error in AuthContext:", error.response?.data || error.message);

throw error.response?.data || new Error(error.message || 'Registration failed due to an unknown error.');

}

};

const logout = useCallback(() => {

localStorage.removeItem('token');

localStorage.removeItem('userId');

localStorage.removeItem('userEmail');

setToken(null);

setCurrentUser(null);

// Navigation to '/login' will be handled by ProtectedRoute or App logic

}, []);

const value = {

currentUser,

token,

loading,

login,

register,

logout,

authAxios // Provide the configured axios instance for authenticated requests

};

return (

<AuthContext.Provider value={value}>

{children}

</AuthContext.Provider>

);

}

**client/src/components/Navbar.jsx**

import React from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { useAuth } from '../contexts/AuthContext';

// Simple SVG Icons

const HomeIcon = () => <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-5 h-5 inline mr-1"><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-5 h-5 inline mr-1"><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-5 h-5 inline mr-1"><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 ArrowLeftOnRectangleIcon = () => <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-5 h-5 inline mr-1"><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>;

const Navbar = () => {

const { currentUser, logout } = useAuth();

const navigate = useNavigate();

const handleLogoutClick = () => {

logout();

navigate('/login'); // Ensure navigation after logout action completes

};

return (

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

<div className="container mx-auto px-4 sm:px-6 py-3 flex flex-wrap justify-between items-center">

<Link to={currentUser ? "/dashboard" : "/"} className="text-xl sm:text-2xl font-bold text-indigo-600 hover:text-indigo-700 transition-colors">

Maternity Justice Portal

</Link>

<div className="flex items-center space-x-1 sm:space-x-2 md:space-x-4 mt-2 sm:mt-0">

{currentUser ? (

<>

<span className="text-xs sm:text-sm text-gray-600 hidden md:inline">Welcome, {currentUser.email}!</span>

<Link to="/dashboard" className="text-gray-700 hover:text-indigo-600 px-2 py-2 rounded-md text-xs sm:text-sm font-medium flex items-center"> <HomeIcon />Dashboard</Link>

<Link to="/new-complaint" className="text-gray-700 hover:text-indigo-600 px-2 py-2 rounded-md text-xs sm:text-sm font-medium flex items-center"> <DocumentPlusIcon />File Complaint</Link>

<Link to="/complaints" className="text-gray-700 hover:text-indigo-600 px-2 py-2 rounded-md text-xs sm:text-sm font-medium flex items-center"> <ListBulletIcon />My Complaints</Link>

<button onClick={handleLogoutClick} className="bg-pink-500 hover:bg-pink-600 text-white px-2 sm:px-3 py-2 rounded-md text-xs sm:text-sm font-medium transition-colors flex items-center"> <ArrowLeftOnRectangleIcon />Logout</button>

</>

) : (

<>

<Link to="/" className="text-gray-700 hover:text-indigo-600 px-2 py-2 rounded-md text-xs sm:text-sm font-medium">Home</Link>

<Link to="/login" className="text-gray-700 hover:text-indigo-600 px-2 py-2 rounded-md text-xs sm:text-sm font-medium">Login</Link>

<Link to="/register" className="bg-indigo-500 hover:bg-indigo-600 text-white px-3 sm:px-4 py-2 rounded-md text-xs sm:text-sm font-medium transition-colors">Register</Link>

</>

)}

</div>

</div>

</nav>

);

};

export default Navbar;

**client/src/components/Footer.jsx**

import React from 'react';

const Footer = () => {

return (

<footer className="bg-gray-800 text-white py-8 mt-auto"> {/\* mt-auto pushes footer to bottom if content is short \*/}

<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>

);

};

export default Footer;

**client/src/components/Modal.jsx**

(A generic modal component for notifications)

import React from 'react';

const Modal = ({ title = "Notification", message, type = "info", onClose }) => {

if (!message) return null;

let bgColor, borderColor, textColor, buttonColor, buttonHoverColor;

switch (type) {

case 'success':

bgColor = 'bg-green-100';

borderColor = 'border-green-400';

textColor = 'text-green-700';

buttonColor = 'bg-green-500';

buttonHoverColor = 'hover:bg-green-700';

break;

case 'error':

bgColor = 'bg-red-100';

borderColor = 'border-red-400';

textColor = 'text-red-700';

buttonColor = 'bg-red-500';

buttonHoverColor = 'hover:bg-red-700';

break;

default: // info

bgColor = 'bg-blue-100';

borderColor = 'border-blue-400';

textColor = 'text-blue-700';

buttonColor = 'bg-blue-500';

buttonHoverColor = 'hover:bg-blue-700';

}

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 p-4 transition-opacity duration-300 ease-in-out">

<div className={`relative mx-auto p-5 border ${borderColor} w-full max-w-sm sm:max-w-md shadow-xl rounded-lg ${bgColor} transform transition-all duration-300 ease-in-out scale-95 opacity-0 animate-modal-appear`}>

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

<h3 className={`text-lg leading-6 font-medium ${textColor}`}>{title}</h3>

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

<p className={`text-sm ${textColor}`}>{message}</p>

</div>

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

<button

onClick={onClose}

className={`px-4 py-2 ${buttonColor} ${buttonHoverColor} text-white text-base font-medium rounded-md w-full shadow-sm focus:outline-none focus:ring-2 focus:ring-offset-2`}

>

OK

</button>

</div>

</div>

</div>

{/\* Basic CSS for modal animation (add to your index.css or App.css) \*/}

<style jsx global>{`

@keyframes modal-appear {

0% {

opacity: 0;

transform: scale(0.95);

}

100% {

opacity: 1;

transform: scale(1);

}

}

.animate-modal-appear {

animation: modal-appear 0.3s ease-out forwards;

}

`}</style>

</div>

);

};

export default Modal;

**client/src/components/ProtectedRoute.jsx**

import React from 'react';

import { Navigate, useLocation } from 'react-router-dom';

import { useAuth } from '../contexts/AuthContext';

const ProtectedRoute = ({ children }) => {

const { currentUser, loading } = useAuth();

const location = useLocation();

if (loading) {

// You can show a global loading spinner here if you prefer

return (

<div className="flex justify-center items-center min-h-[calc(100vh-200px)]"> {/\* Adjust height as needed \*/}

<p className="text-xl font-semibold text-indigo-600">Authenticating...</p>

</div>

);

}

if (!currentUser) {

// User not authenticated, redirect to login page

// Pass the current location so we can redirect back after login

return <Navigate to="/login" state={{ from: location }} replace />;

}

return children; // User is authenticated, render the children components

};

export default ProtectedRoute;

**client/src/pages/HomePage.jsx**

import React from 'react';

import { Link } from 'react-router-dom';

import { useAuth } from '../contexts/AuthContext';

const HomePage = () => {

const { currentUser } = useAuth();

return (

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

<header className="mb-12">

<h2 className="text-4xl sm:text-5xl font-extrabold text-indigo-700 mb-4 animate-fade-in-down">

Know Your Rights. Claim Your Dues.

</h2>

<p className="text-lg sm: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 will facilitate a legal notice to your employer.

</p>

<div className="space-y-4 sm:space-y-0 sm:space-x-4 animate-fade-in">

{currentUser ? (

<Link

to="/dashboard"

className="w-full sm:w-auto inline-block 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"

>

Go to Your Dashboard

</Link>

) : (

<>

<Link

to="/register"

className="w-full sm:w-auto inline-block 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 mb-3 sm:mb-0"

>

Register Your Complaint

</Link>

<Link

to="/login"

className="w-full sm:w-auto inline-block 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

</Link>

</>

)}

</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-xl sm:text-2xl font-semibold text-indigo-600 mb-3">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 (if applicable), 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-xl sm:text-2xl font-semibold text-indigo-600 mb-3">Common Issues</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 by facilitating a formal legal notice.

</p>

</div>

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

<h3 className="text-xl sm: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. Based on your complaint, empaneled lawyers will prepare and send a legal notice to your employer.

</p>

</div>

</section>

</div>

);

};

export default HomePage;

**client/src/pages/LoginPage.jsx**

import React, { useState } from 'react';

import { Link, useNavigate, useLocation } from 'react-router-dom';

import { useAuth } from '../contexts/AuthContext';

import Modal from '../components/Modal'; // Import the Modal

const LoginPage = () => {

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

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

const [error, setError] = useState(''); // For form-specific errors

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

const [modalMessage, setModalMessage] = useState(''); // For modal notifications

const navigate = useNavigate();

const location = useLocation();

const { login } = useAuth();

const from = location.state?.from?.pathname || "/dashboard"; // Redirect path after login

const handleLogin = async (e) => {

e.preventDefault();

setError('');

setModalMessage('');

setIsLoading(true);

try {

await login(email, password);

// AuthContext now handles setting user and token.

// Success navigation is handled by App.jsx or ProtectedRoute logic.

// We can show a success message if needed, then navigate.

// setModalMessage('Login successful! Redirecting...'); // Optional success message

navigate(from, { replace: true });

} catch (err) {

const errorMessage = err.error || err.errors?.map(e => e.msg).join(', ') || err.message || 'Login failed. Please check your credentials.';

setError(errorMessage); // Show error directly on the form

// setModalMessage(errorMessage); // Or use modal

} finally {

setIsLoading(false);

}

};

return (

<>

<Modal message={modalMessage} type={error ? "error" : "success"} onClose={() => setModalMessage('')} />

<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 text-center">{error}</p>}

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

<div>

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

<input

id="login-email"

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"

autoComplete="email"

/>

</div>

<div>

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

<input

id="login-password"

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="••••••••"

autoComplete="current-password"

/>

</div>

<div className="flex items-center justify-end">

<Link to="/request-password-reset" className="text-sm font-medium text-indigo-600 hover:text-indigo-500">

Forgot your password?

</Link>

</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?{' '}

<Link to="/register" className="font-medium text-indigo-600 hover:text-indigo-500 transition-colors">

Register here

</Link>

</p>

</div>

</div>

</>

);

};

export default LoginPage;

**client/src/pages/RegisterPage.jsx**

import React, { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { useAuth } from '../contexts/AuthContext';

import Modal from '../components/Modal';

const RegisterPage = () => {

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

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

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

const [formError, setFormError] = useState(''); // For form-specific errors

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

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

const [modalType, setModalType] = useState('');

const navigate = useNavigate();

const { register } = useAuth();

const handleRegister = async (e) => {

e.preventDefault();

setFormError('');

setModalMessage('');

setModalType('');

if (password !== confirmPassword) {

setFormError('Passwords do not match.');

return;

}

if (password.length < 6) {

setFormError('Password must be at least 6 characters long.');

return;

}

setIsLoading(true);

try {

const data = await register(email, password);

setModalMessage(data.message || 'Registration successful! Please log in.');

setModalType('success');

setEmail('');

setPassword('');

setConfirmPassword('');

setTimeout(() => {

setModalMessage(''); // Clear modal

navigate('/login');

}, 2500);

} catch (err) {

const errorMessage = err.error || err.errors?.map(e => e.msg).join(', ') || err.message || 'Registration failed. Please try again.';

setFormError(errorMessage); // Show error directly on the form

// setModalMessage(errorMessage); // Or use modal for critical server errors

// setModalType('error');

} finally {

setIsLoading(false);

}

};

return (

<>

<Modal message={modalMessage} type={modalType} onClose={() => setModalMessage('')} />

<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>

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

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

<div>

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

<input

id="register-email"

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"

autoComplete="email"

/>

</div>

<div>

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

<input

id="register-password"

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="••••••••"

autoComplete="new-password"

/>

<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" htmlFor="register-confirm-password">Confirm Password</label>

<input

id="register-confirm-password"

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="••••••••"

autoComplete="new-password"

/>

</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?{' '}

<Link to="/login" className="font-medium text-indigo-600 hover:text-indigo-500 transition-colors">

Login here

</Link>

</p>

</div>

</div>

</>

);

};

export default RegisterPage;

**client/src/pages/RequestPasswordResetPage.jsx**

import React, { useState } from 'react';

import { Link } from 'react-router-dom';

import axios from 'axios'; // Using axios for consistency

import Modal from '../components/Modal';

const API\_BASE\_URL = import.meta.env.VITE\_API\_BASE\_URL || 'http://localhost:5000/api';

const RequestPasswordResetPage = () => {

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

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

const [modalType, setModalType] = useState('');

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

const handleSubmit = async (e) => {

e.preventDefault();

setModalMessage('');

setModalType('');

setIsLoading(true);

try {

const response = await axios.post(`${API\_BASE\_URL}/auth/request-reset`, { email });

setModalMessage(response.data.message || 'If an account with that email exists, a password reset link has been sent.');

setModalType('success');

setEmail(''); // Clear email field on success

} catch (err) {

const errorMsg = err.response?.data?.error || err.response?.data?.errors?.map(e => e.msg).join(', ') || 'Failed to request password reset. Please try again.';

setModalMessage(errorMsg);

setModalType('error');

} finally {

setIsLoading(false);

}

};

return (

<>

<Modal message={modalMessage} type={modalType} onClose={() => setModalMessage('')} title={modalType === 'error' ? "Error" : "Success"}/>

<div className="flex flex-col items-center justify-center min-h-[calc(100vh-150px)] bg-gradient-to-br from-indigo-100 via-purple-50 to-pink-100 font-sans p-4">

<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-6">Forgot Your Password?</h2>

<p className="text-center text-sm text-gray-600 mb-6">

No worries! Enter your email address below, and we'll send you a link to reset your password.

</p>

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

<div>

<label htmlFor="email-request-reset" className="block text-sm font-medium text-gray-700 mb-1">

Email Address

</label>

<input

id="email-request-reset"

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"

autoComplete="email"

/>

</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 ? 'Sending Link...' : 'Send Password Reset Link'}

</button>

</form>

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

Remembered your password?{' '}

<Link to="/login" className="font-medium text-indigo-600 hover:text-indigo-500 transition-colors">

Login

</Link>

</p>

</div>

</div>

</>

);

};

export default RequestPasswordResetPage;

**client/src/pages/ResetPasswordPage.jsx**

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

import { useNavigate, useLocation, Link } from 'react-router-dom';

import axios from 'axios';

import Modal from '../components/Modal';

const API\_BASE\_URL = import.meta.env.VITE\_API\_BASE\_URL || 'http://localhost:5000/api';

const ResetPasswordPage = () => {

const [newPassword, setNewPassword] = useState('');

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

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

const [modalType, setModalType] = useState('');

const [formError, setFormError] = useState(''); // For form-specific errors

const [token, setToken] = useState('');

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

const navigate = useNavigate();

const location = useLocation();

useEffect(() => {

const params = new URLSearchParams(location.search);

const t = params.get('token');

if (t) {

setToken(t);

} else {

setModalMessage('Invalid or missing password reset token in URL. Please use the link from your email.');

setModalType('error');

}

}, [location.search]);

const handleReset = async (e) => {

e.preventDefault();

setFormError('');

setModalMessage('');

setModalType('');

if (!token) {

setModalMessage('Reset token is missing. Please use the link from your email again.');

setModalType('error');

return;

}

if (newPassword !== confirmPassword) {

setFormError('Passwords do not match.');

return;

}

if (newPassword.length < 6) {

setFormError('Password must be at least 6 characters long.');

return;

}

setIsLoading(true);

try {

const response = await axios.post(`${API\_BASE\_URL}/auth/reset-password`, { token, newPassword });

setModalMessage(response.data.message || 'Password reset successful! You can now log in.');

setModalType('success');

setNewPassword('');

setConfirmPassword('');

setTimeout(() => {

setModalMessage(''); // Clear modal

navigate('/login');

}, 3000);

} catch (err) {

const errorMsg = err.response?.data?.error || err.response?.data?.errors?.map(e => e.msg).join(', ') || 'Password reset failed. The link may be invalid or expired.';

setModalMessage(errorMsg);

setModalType('error');

} finally {

setIsLoading(false);

}

};

return (

<>

<Modal message={modalMessage} type={modalType} onClose={() => setModalMessage('')} title={modalType === 'error' ? "Error" : "Success"}/>

<div className="flex flex-col items-center justify-center min-h-[calc(100vh-150px)] bg-gradient-to-br from-indigo-100 via-purple-50 to-pink-100 font-sans p-4">

<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">Set New Password</h2>

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

{!token && !modalMessage && <p className="text-center text-yellow-600 p-3 rounded-md mb-4">Verifying reset link...</p>}

{token && !modalMessage.includes('successful') && ( // Only show form if token is present and no success message yet

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

<div>

<label className="block text-sm font-medium text-gray-700 mb-1" htmlFor="reset-new-password">New Password</label>

<input

id="reset-new-password"

type="password"

placeholder="Enter new password"

value={newPassword}

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

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"

required

autoComplete="new-password"

/>

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

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-1" htmlFor="reset-confirm-password">Confirm New Password</label>

<input

id="reset-confirm-password"

type="password"

placeholder="Confirm new password"

value={confirmPassword}

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

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"

required

autoComplete="new-password"

/>

</div>

<button

type="submit"

disabled={isLoading || !token}

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 ? 'Resetting...' : 'Reset Password'}

</button>

</form>

)}

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

<Link to="/login" className="font-medium text-indigo-600 hover:text-indigo-500 transition-colors">

Back to Login

</Link>

</p>

</div>

</div>

</>

);

};

export default ResetPasswordPage;

**client/src/pages/DashboardPage.jsx**

import React from 'react';

import { Link } from 'react-router-dom';

import { useAuth } from '../contexts/AuthContext';

// Icons

const DocumentPlusIcon = () => <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-8 h-8 text-indigo-500"><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-8 h-8 text-pink-500"><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 DashboardPage = () => {

const { currentUser } = useAuth();

return (

<div className="bg-white p-6 md:p-8 rounded-xl shadow-2xl animate-fade-in">

<h2 className="text-3xl font-bold text-indigo-700 mb-2">Welcome to Your Dashboard!</h2>

{currentUser && <p className="text-gray-600 mb-1">Logged in as: <span className="font-medium">{currentUser.email}</span></p>}

{currentUser?.id && <p className="text-xs text-gray-500 mb-8">User ID: {currentUser.id}</p>}

<p className="text-gray-700 mb-8">

This is your personal space to manage your maternity benefit grievances.

You can file a new complaint or track the status of previously submitted ones.

</p>

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

<Link

to="/new-complaint"

className="group block bg-indigo-50 p-6 rounded-lg hover:bg-indigo-100 transition-all duration-300 ease-in-out cursor-pointer shadow-md hover:shadow-lg focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:ring-opacity-50 transform hover:-translate-y-1"

>

<div className="flex flex-col items-center text-center md:flex-row md:text-left">

<DocumentPlusIcon />

<div className="mt-3 md:mt-0 md:ml-4">

<h3 className="text-xl font-semibold text-indigo-700 group-hover:text-indigo-800">File a New Complaint</h3>

<p className="text-gray-700 text-sm mt-1">

If you're facing issues with your maternity benefits, start here to submit the details.

</p>

</div>

</div>

</Link>

<Link

to="/complaints"

className="group block bg-pink-50 p-6 rounded-lg hover:bg-pink-100 transition-all duration-300 ease-in-out cursor-pointer shadow-md hover:shadow-lg focus:outline-none focus:ring-2 focus:ring-pink-500 focus:ring-opacity-50 transform hover:-translate-y-1"

>

<div className="flex flex-col items-center text-center md:flex-row md:text-left">

<ListBulletIcon />

<div className="mt-3 md:mt-0 md:ml-4">

<h3 className="text-xl font-semibold text-pink-700 group-hover:text-pink-800">View My Complaints</h3>

<p className="text-gray-700 text-sm mt-1">

Track the status of your submitted complaints and review the information provided.

</p>

</div>

</div>

</Link>

</div>

<div className="mt-10 border-t border-gray-200 pt-6">

<h3 className="text-xl font-semibold text-gray-800 mb-4">Important Information & Next Steps</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 to the best of your knowledge.</li>

<li>Keep digital or physical copies of all relevant documents (e.g., employment contract, communications with HR/manager, medical certificates, leave applications).</li>

<li>After submitting a complaint, a legal notice will be prepared based on the details you provide and sent to your organization.</li>

<li>You will be updated on the status of your complaint, including when the notice is sent.</li>

<li>Familiarize yourself with the key provisions of the Maternity Benefit Act, 2017, to understand your rights better.</li>

</ul>

</div>

{/\* Basic CSS for fade-in animation (add to your index.css or App.css) \*/}

<style jsx global>{`

@keyframes fadeIn {

from { opacity: 0; transform: translateY(10px); }

to { opacity: 1; transform: translateY(0); }

}

.animate-fade-in {

animation: fadeIn 0.5s ease-out forwards;

}

@keyframes fade-in-down {

from { opacity: 0; transform: translateY(-20px); }

to { opacity: 1; transform: translateY(0); }

}

.animate-fade-in-down { animation: fade-in-down 0.6s ease-out forwards; }

@keyframes fade-in-up {

from { opacity: 0; transform: translateY(20px); }

to { opacity: 1; transform: translateY(0); }

}

.animate-fade-in-up { animation: fade-in-up 0.6s ease-out forwards; }

`}</style>

</div>

);

};

export default DashboardPage;

**client/src/pages/NewComplaintPage.jsx**

import React, { useState } from 'react';

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

import { useAuth } from '../contexts/AuthContext';

import Modal from '../components/Modal';

const NewComplaintPage = () => {

const { authAxios } = useAuth();

const navigate = useNavigate();

const [formData, setFormData] = useState({

complainantName: '',

complainantContact: '',

companyName: '',

companyAddress: '',

dateOfJoining: '',

employmentType: 'permanent',

expectedDeliveryDate: '',

actualDeliveryDate: '',

isAdoptiveOrCommissioning: 'no',

adoptionDate: '',

numberOfSurvivingChildren: '0',

issueType: 'denial\_of\_leave',

issueDetails: '',

chronologyOfEvents: '',

supportingDocumentsInfo: '',

consentToShare: false,

});

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

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

const [modalType, setModalType] = useState('');

const [formErrors, setFormErrors] = useState({});

const validateForm = () => {

const errors = {};

if (!formData.complainantName.trim()) errors.complainantName = "Complainant name is required.";

if (!formData.complainantContact.trim()) errors.complainantContact = "Contact number is required.";

// Basic Indian mobile number validation

else if (!/^(?:\+91[\-\s]?)?[6-9]\d{9}$/.test(formData.complainantContact.trim())) errors.complainantContact = "Please enter a valid Indian mobile number.";

if (!formData.companyName.trim()) errors.companyName = "Company name is required.";

if (!formData.companyAddress.trim()) errors.companyAddress = "Company address is required.";

if (!formData.dateOfJoining) errors.dateOfJoining = "Date of joining is required.";

if (!formData.employmentType) errors.employmentType = "Employment type is required.";

if (formData.isAdoptiveOrCommissioning === 'yes' && !formData.adoptionDate) errors.adoptionDate = "Adoption date is required for adoptive/commissioning mothers.";

if (formData.numberOfSurvivingChildren === '' || parseInt(formData.numberOfSurvivingChildren) < 0) errors.numberOfSurvivingChildren = "Number of surviving children must be 0 or more.";

if (!formData.issueType) errors.issueType = "Issue type is required.";

if (!formData.issueDetails.trim()) errors.issueDetails = "Issue details are required.";

if (!formData.chronologyOfEvents.trim()) errors.chronologyOfEvents = "Chronology of events is required.";

if (!formData.consentToShare) errors.consentToShare = "You must consent to share your information.";

setFormErrors(errors);

return Object.keys(errors).length === 0;

};

const handleChange = (e) => {

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

setFormData(prev => ({

...prev,

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

}));

// Clear specific error when user starts typing

if (formErrors[name]) {

setFormErrors(prev => ({...prev, [name]: ''}));

}

};

const handleSubmit = async (e) => {

e.preventDefault();

setModalMessage('');

setModalType('');

if (!validateForm()) {

setModalMessage("Please correct the errors in the form.");

setModalType('error');

// Focus first field with error

const firstErrorField = Object.keys(formErrors).find(key => formErrors[key]);

if (firstErrorField) {

const fieldElement = document.getElementsByName(firstErrorField)[0];

fieldElement?.focus();

}

return;

}

setIsSubmitting(true);

try {

const response = await authAxios.post('/complaints', formData);

setModalMessage(response.data.message || "Complaint submitted successfully!");

setModalType('success');

setFormData({ // Reset form

complainantName: '', complainantContact: '', companyName: '', companyAddress: '',

dateOfJoining: '', employmentType: 'permanent', expectedDeliveryDate: '', actualDeliveryDate: '',

isAdoptiveOrCommissioning: 'no', adoptionDate: '', numberOfSurvivingChildren: '0',

issueType: 'denial\_of\_leave', issueDetails: '', chronologyOfEvents: '',

supportingDocumentsInfo: '', consentToShare: false,

});

setFormErrors({}); // Clear errors

setTimeout(() => {

setModalMessage(''); // Clear modal

navigate('/complaints');

}, 2500);

} catch (error) {

console.error("Error submitting complaint:", error.response?.data || error.message);

const errorMsg = error.response?.data?.error || error.response?.data?.errors?.map(err => err.msg).join(', ') || "Failed to submit complaint. Please try again.";

setModalMessage(errorMsg);

setModalType('error');

} finally {

setIsSubmitting(false);

}

};

const closeModal = () => {

setModalMessage('');

setModalType('');

};

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

const errorTextClass = "text-red-500 text-xs mt-1";

const formStructure = [

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

{ label: "Your Contact Number", name: "complainantContact", type: "tel", required: true, placeholder: "+91-XXXXXXXXXX" },

{ 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 },

...(formData.isAdoptiveOrCommissioning === 'yes' ? [{ label: "Date of Adoption / Child Handover", name: "adoptionDate", type: "date", required: true }] : []),

{ 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 (Key dates and actions)", name: "chronologyOfEvents", type: "textarea", rows: 4, required: true, placeholder: "E.g., Applied for leave on DD/MM/YYYY, Received response on DD/MM/YYYY..." },

{ label: "Information about Supporting Documents You Possess", name: "supportingDocumentsInfo", type: "textarea", rows: 3, placeholder: "List documents like employment contract, leave application copy, emails, medical certificates. You may be asked to provide these later." },

];

return (

<>

<Modal message={modalMessage} type={modalType} onClose={closeModal} title={modalType === 'error' ? "Form Submission Error" : "Success"}/>

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

<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.</p>

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

{formStructure.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} placeholder={field.placeholder} className={`${commonInputClass} ${formErrors[field.name] ? 'border-red-500' : 'border-gray-300'}`} />

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

<select id={field.name} name={field.name} value={formData[field.name]} onChange={handleChange} className={`${commonInputClass} bg-white ${formErrors[field.name] ? 'border-red-500' : 'border-gray-300'}`}>

{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} min={field.min} pattern={field.pattern} placeholder={field.placeholder} className={`${commonInputClass} ${formErrors[field.name] ? 'border-red-500' : 'border-gray-300'}`} />

)}

{formErrors[field.name] && <p className={errorTextClass}>{formErrors[field.name]}</p>}

</div>

))}

<div className="pt-4">

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

<input type="checkbox" id="consentToShare" name="consentToShare" checked={formData.consentToShare} onChange={handleChange} 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>

{formErrors.consentToShare && <p className={errorTextClass}>{formErrors.consentToShare}</p>}

</div>

<div className="flex flex-col sm:flex-row justify-end space-y-3 sm:space-y-0 sm:space-x-4 pt-6">

<button type="button" onClick={() => navigate(-1)} className="w-full sm:w-auto 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="w-full sm:w-auto 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>

<style jsx global>{`

@keyframes fadeIn {

from { opacity: 0; transform: translateY(10px); }

to { opacity: 1; transform: translateY(0); }

}

.animate-fade-in {

animation: fadeIn 0.5s ease-out forwards;

}

`}</style>

</>

);

};

export default NewComplaintPage;

**client/src/pages/ViewComplaintsPage.jsx**

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

import { useAuth } from '../contexts/AuthContext';

import { Link } from 'react-router-dom';

import Modal from '../components/Modal'; // Using the shared Modal

const ViewComplaintsPage = () => {

const { authAxios, currentUser } = useAuth();

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

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

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

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

const fetchComplaints = useCallback(async () => {

if (!currentUser) { // Ensure currentUser is available from AuthContext

setIsLoading(false);

setError("Please log in to view your complaints.");

return;

}

setIsLoading(true);

setError('');

try {

const response = await authAxios.get('/complaints'); // authAxios includes the token

setComplaints(response.data);

} catch (err) {

console.error("Error fetching complaints:", err.response?.data || err.message);

setError(err.response?.data?.error || "Failed to fetch your complaints. Please try refreshing the page.");

} finally {

setIsLoading(false);

}

}, [authAxios, currentUser]); // Add currentUser as dependency

useEffect(() => {

fetchComplaints();

}, [fetchComplaints]);

const formatDate = (dateString) => {

if (!dateString) return 'N/A';

try {

return new Date(dateString).toLocaleDateString('en-IN', {

year: 'numeric', month: 'long', day: 'numeric',

// hour: '2-digit', minute: '2-digit' // Uncomment if time is needed

});

} catch (e) {

return 'Invalid Date';

}

};

const getStatusClass = (status) => {

const s = status?.toLowerCase();

if (s === 'submitted') return 'bg-blue-100 text-blue-800';

if (s === 'in\_review') return 'bg-yellow-100 text-yellow-800';

if (s === 'action\_taken') return 'bg-teal-100 text-teal-800';

if (s === 'resolved') return 'bg-green-100 text-green-800';

if (s === 'closed') return 'bg-gray-200 text-gray-700';

return 'bg-gray-100 text-gray-600';

};

const renderComplaintDetailsInModal = () => {

if (!selectedComplaint) return null;

const detailsToShow = [

{ label: 'Complaint ID', value: selectedComplaint.\_id?.slice(-8) || 'N/A' },

{ label: 'Status', value: selectedComplaint.status?.replace(/\_/g, ' ').replace(/\b\w/g, l => l.toUpperCase()) || 'N/A' },

{ label: 'Submitted On', value: formatDate(selectedComplaint.submittedAt) },

{ label: 'Complainant Name', value: selectedComplaint.complainantName },

{ label: 'Contact', value: selectedComplaint.complainantContact },

{ label: 'Company Name', value: selectedComplaint.companyName },

{ label: 'Company Address', value: selectedComplaint.companyAddress },

{ label: 'Date of Joining', value: formatDate(selectedComplaint.dateOfJoining) },

{ label: 'Employment Type', value: selectedComplaint.employmentType },

{ label: 'Expected Delivery Date', value: formatDate(selectedComplaint.expectedDeliveryDate) },

{ label: 'Actual Delivery Date', value: formatDate(selectedComplaint.actualDeliveryDate) },

{ label: 'Adoptive/Commissioning', value: selectedComplaint.isAdoptiveOrCommissioning },

...(selectedComplaint.isAdoptiveOrCommissioning === 'yes' ? [{ label: 'Adoption Date', value: formatDate(selectedComplaint.adoptionDate) }] : []),

{ label: 'Surviving Children', value: selectedComplaint.numberOfSurvivingChildren },

{ label: 'Issue Type', value: selectedComplaint.issueType?.replace(/\_/g, ' ').replace(/\b\w/g, l => l.toUpperCase()) },

{ label: 'Issue Details', value: selectedComplaint.issueDetails, pre: true },

{ label: 'Chronology of Events', value: selectedComplaint.chronologyOfEvents, pre: true },

{ label: 'Supporting Documents Info', value: selectedComplaint.supportingDocumentsInfo || 'N/A', pre: true },

{ label: 'Consent to Share', value: selectedComplaint.consentToShare ? 'Yes' : 'No' },

];

return (

detailsToShow.map(item => item.value || item.value === 0 ? ( // Check for 0 as valid value

<div key={item.label} className="grid grid-cols-3 gap-2 border-b border-gray-100 py-2.5 last:border-b-0">

<p className="text-sm font-semibold text-gray-700 col-span-1">{item.label}:</p>

<p className={`text-sm text-gray-800 col-span-2 ${item.pre ? 'whitespace-pre-wrap break-words' : 'break-words'}`}>{String(item.value)}</p>

</div>

) : null)

);

};

if (isLoading) {

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

}

return (

<>

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

<h2 className="text-2xl md:text-3xl font-bold text-indigo-700 mb-8 text-center">My Submitted Complaints</h2>

{error && <Modal message={error} type="error" onClose={() => setError('')} title="Error Fetching Complaints"/>}

{!isLoading && !error && complaints.length === 0 && (

<div className="text-center py-8">

<svg className="mx-auto h-12 w-12 text-gray-400" fill="none" viewBox="0 0 24 24" stroke="currentColor" aria-hidden="true">

<path vectorEffect="non-scaling-stroke" strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M9 13h6m-3-3v6m-9 1V7a2 2 0 012-2h6l2 2h6a2 2 0 012 2v8a2 2 0 01-2 2H5a2 2 0 01-2-2z" />

</svg>

<h3 className="mt-2 text-lg font-medium text-gray-900">No complaints filed yet</h3>

<p className="mt-1 text-sm text-gray-500">Get started by filing your first complaint.</p>

<div className="mt-6">

<Link

to="/new-complaint"

className="inline-flex items-center px-4 py-2 border border-transparent shadow-sm text-sm font-medium rounded-md text-white bg-indigo-600 hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

>

<svg className="-ml-1 mr-2 h-5 w-5" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20" fill="currentColor" aria-hidden="true">

<path fillRule="evenodd" d="M10 3a1 1 0 011 1v5h5a1 1 0 110 2h-5v5a1 1 0 11-2 0v-5H4a1 1 0 110-2h5V4a1 1 0 011-1z" clipRule="evenodd" />

</svg>

File New Complaint

</Link>

</div>

</div>

)}

{complaints.length > 0 && (

<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-lg transition-shadow duration-300 ease-in-out">

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

<h3 className="text-lg font-semibold text-indigo-700 mb-2 sm:mb-0">

Complaint vs: <span className="font-bold">{complaint.companyName || 'N/A'}</span>

</h3>

<span className={`text-xs font-semibold mr-2 px-3 py-1 rounded-full ${getStatusClass(complaint.status)}`}>

{complaint.status?.replace(/\_/g, ' ').replace(/\b\w/g, l => l.toUpperCase()) || 'N/A'}

</span>

</div>

<p className="text-sm text-gray-500 mb-1"><strong>ID:</strong> ...{complaint.\_id?.slice(-8) || 'N/A'}</p>

<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>Issue:</strong> {complaint.issueType?.replace(/\_/g, ' ').replace(/\b\w/g, l => l.toUpperCase()) || 'N/A'}</p>

<p className="text-sm text-gray-700 mb-3 truncate"><strong>Details:</strong> {complaint.issueDetails || 'N/A'}</p>

<button

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

className="text-sm text-indigo-600 hover:text-indigo-800 font-medium transition-colors focus:outline-none hover:underline"

>

View Full Details

</button>

</div>

))}

</div>

)}

{selectedComplaint && (

<Modal title={`Complaint Details`} onClose={() => setSelectedComplaint(null)}>

{renderComplaintDetailsInModal()}

</Modal>

)}

</div>

<style jsx global>{`

@keyframes fadeIn {

from { opacity: 0; transform: translateY(10px); }

to { opacity: 1; transform: translateY(0); }

}

.animate-fade-in {

animation: fadeIn 0.5s ease-out forwards;

}

`}</style>

</>

);

};

export default ViewComplaintsPage;

This completes the set of components you requested. Remember to:

1. Place each code block into its corresponding file within the client/src/ directory structure.
2. Ensure you have axios and react-router-dom installed in your client/package.json.
3. Set up Tailwind CSS correctly in your client project.
4. The App.jsx uses react-router-dom for routing; ensure your main.jsx (or index.js) wraps the <App /> component with <BrowserRouter>.
5. The AuthContext provides a global way to manage authentication. Wrap your App component with <AuthProvider> in main.jsx.
6. The ProtectedRoute component can be used to protect routes that require authentication.
7. You'll need to create the client/src/index.css file and include Tailwind directives:
8. @tailwind base;
9. @tailwind components;
10. @tailwind utilities;
11. /\* Optional: Add global styles or font imports here \*/
12. body {
13. font-family: 'Inter', sans-serif; /\* Example font \*/
14. }
15. Ensure your backend server is running and accessible at the URL specified in VITE\_API\_BASE\_URL (for Vite) or REACT\_APP\_API\_BASE\_URL (for Create React App) in your client/.env file.

This provides a very comprehensive starting point for your full-stack application's frontend.