// This document will outline a full-stack version of your maternity complaint portal

// using a traditional React frontend + Express backend + MongoDB setup.

// Let's start with the Express backend server (Node.js).

// --- server/index.js ---

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const dotenv = require('dotenv');

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

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

const app = express();

dotenv.config();

app.use(cors());

app.use(express.json());

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

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

mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => {

console.log('Connected to MongoDB');

app.listen(process.env.PORT || 5000, () => {

console.log(`Server running on port ${process.env.PORT || 5000}`);

});

}).catch(err => console.error(err));

// --- server/models/User.js ---

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const UserSchema = new mongoose.Schema({

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

});

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

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

this.password = await bcrypt.hash(this.password, 10);

next();

});

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' },

complainantName: String,

complainantContact: String,

companyName: String,

companyAddress: String,

dateOfJoining: Date,

employmentType: String,

expectedDeliveryDate: Date,

actualDeliveryDate: Date,

isAdoptiveOrCommissioning: String,

adoptionDate: Date,

numberOfSurvivingChildren: Number,

issueType: String,

issueDetails: String,

chronologyOfEvents: String,

supportingDocumentsInfo: String,

consentToShare: Boolean,

status: { type: String, default: 'submitted' },

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

});

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

// --- server/routes/auth.js ---

const express = require('express');

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

const jwt = require('jsonwebtoken');

const bcrypt = require('bcrypt');

const router = express.Router();

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

try {

const user = new User(req.body);

await user.save();

res.status(201).send({ message: 'User registered successfully' });

} catch (err) {

res.status(400).send({ error: err.message });

}

});

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

const { email, password } = req.body;

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

if (!user) return res.status(401).send({ error: 'Invalid credentials' });

const match = await bcrypt.compare(password, user.password);

if (!match) return res.status(401).send({ error: 'Invalid credentials' });

const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET, { expiresIn: '1d' });

res.send({ token });

});

module.exports = router;

// --- server/routes/complaints.js ---

const express = require('express');

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

const jwt = require('jsonwebtoken');

const router = express.Router();

function verifyToken(req, res, next) {

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

if (!token) return res.status(403).send({ error: 'Token missing' });

try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

req.userId = decoded.id;

next();

} catch {

return res.status(401).send({ error: 'Invalid token' });

}

}

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

try {

const complaint = new Complaint({ ...req.body, userId: req.userId });

await complaint.save();

res.status(201).send({ message: 'Complaint submitted' });

} catch (err) {

res.status(400).send({ error: err.message });

}

});

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

try {

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

res.send(complaints);

} catch (err) {

res.status(400).send({ error: err.message });

}

});

module.exports = router;