const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const jwt = require('jsonwebtoken');

const app = express();

const PORT = process.env.PORT || 3000;

const JWT\_SECRET = 'your\_jwt\_secret';

// Middleware

app.use(bodyParser.json());

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/doctor\_db', {

useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => {

console.log('Connected to MongoDB');

}).catch(err => {

console.error('Error connecting to MongoDB:', err);

});

// Review Model

const Review = mongoose.model('Review', {

patientName: String,

doctorName: String,

rating: { type: Number, required: true, min: 1, max: 5 },

feedback: { type: String, required: true, maxlength: 500 },

response: String,

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

});

// Middleware to verify JWT token

const verifyToken = (req, res, next) => {

const token = req.headers.authorization;

if (!token) {

return res.status(401).json({ error: 'Unauthorized' });

}

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

if (err) {

return res.status(401).json({ error: 'Unauthorized' });

}

req.user = decoded;

next();

});

};

// Route to submit a review

app.post('/api/reviews', verifyToken, async (req, res) => {

try {

const { patientName, doctorName, rating, feedback } = req.body;

const review = await Review.create({ patientName, doctorName, rating, feedback });

res.status(201).json(review);

} catch (err) {

console.error('Error submitting review:', err);

res.status(500).json({ error: 'Internal server error' });

}

});

// Route to get reviews for a specific doctor

app.get('/api/reviews/:doctorName', async (req, res) => {

try {

const doctorName = req.params.doctorName;

const reviews = await Review.find({ doctorName });

res.json(reviews);

} catch (err) {

console.error('Error fetching reviews:', err);

res.status(500).json({ error: 'Internal server error' });

}

});

// Route for doctors to respond to patient feedback

app.post('/api/respond', verifyToken, async (req, res) => {

try {

const { reviewId, response } = req.body;

const review = await Review.findById(reviewId);

if (!review) {

return res.status(404).json({ error: 'Review not found' });

}

review.response = response;

await review.save();

res.json({ message: 'Response added successfully', review });

} catch (err) {

console.error('Error adding response:', err);

res.status(500).json({ error: 'Internal server error' });

}

});

// Route for user authentication

app.post('/api/login', (req, res) => {

const { username, password } = req.body;

// Check credentials and generate JWT token

if (username === 'patient' && password === 'password') {

const token = jwt.sign({ username: 'patient' }, JWT\_SECRET);

res.json({ token });

} else if (username === 'doctor' && password === 'password') {

const token = jwt.sign({ username: 'doctor' }, JWT\_SECRET);

res.json({ token });

} else {

res.status(401).json({ error: 'Invalid credentials' });

}

});

// Start the server

app.listen(PORT, () => {

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

});