from flask import Flask, request, jsonify, g

from functools import wraps

import jwt

from pymongo import MongoClient

from bson.objectid import ObjectId

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'your\_secret\_key'

# MongoDB Configuration

client = MongoClient('mongodb://localhost:27017/')

db = client['task\_management']

users\_collection = db['users']

tasks\_collection = db['tasks']

# User roles

ROLES = {

'admin': 0,

'manager': 1,

'regular': 2

}

# Authentication Middleware

def token\_required(f):

@wraps(f)

def decorated(\*args, \*\*kwargs):

token = request.headers.get('Authorization')

if not token:

return jsonify({'message': 'Token is missing!'}), 401

try:

data = jwt.decode(token, app.config['SECRET\_KEY'], algorithms=["HS256"])

g.user = data['user']

except:

return jsonify({'message': 'Token is invalid!'}), 401

return f(\*args, \*\*kwargs)

return decorated

# Routes

@app.route('/login', methods=['POST'])

def login():

auth = request.authorization

if not auth or not auth.username or not auth.password:

return jsonify({'message': 'Could not verify'}), 401

user = users\_collection.find\_one({'username': auth.username})

if not user:

return jsonify({'message': 'User not found'}), 401

if user['password'] == auth.password:

token = jwt.encode({'user': user['username'], 'role': user['role']}, app.config['SECRET\_KEY'], algorithm="HS256")

return jsonify({'token': token})

return jsonify({'message': 'Could not verify'}), 401

@app.route('/tasks', methods=['GET'])

@token\_required

def get\_tasks():

user\_role = g.user['role']

if user\_role == ROLES['admin']:

tasks = tasks\_collection.find()

elif user\_role == ROLES['manager']:

tasks = tasks\_collection.find({'assigned\_to': g.user['user']})

else:

tasks = tasks\_collection.find({'assigned\_to': g.user['user']})

output = []

for task in tasks:

output.append({'id': str(task['\_id']), 'title': task['title'], 'description': task['description'], 'status': task['status']})

return jsonify({'tasks': output})

@app.route('/tasks/<task\_id>', methods=['PUT'])

@token\_required

def update\_task(task\_id):

task = tasks\_collection.find\_one({'\_id': ObjectId(task\_id)})

if not task:

return jsonify({'message': 'Task not found'}), 404

user\_role = g.user['role']

if user\_role == ROLES['admin'] or (user\_role == ROLES['manager'] and task['assigned\_to'] == g.user['user']) or (user\_role == ROLES['regular'] and task['assigned\_to'] == g.user['user']):

data = request.get\_json()

tasks\_collection.update\_one({'\_id': ObjectId(task\_id)}, {'$set': data})

return jsonify({'message': 'Task updated successfully'})

else:

return jsonify({'message': 'You are not authorized to perform this action'}), 403

# Example for other CRUD operations (Create and Delete tasks) can be similarly implemented.

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)