**Implement Authentication:**

**Create src/routes/userRoutes.ts:**

import { Router } from 'express';

import bcrypt from 'bcryptjs';

import jwt from 'jsonwebtoken';

import User from '../models/User';

const router = Router();

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

const { email, password } = req.body;

try {

const passwordHash = await bcrypt.hash(password, 10);

const user = new User({ email, passwordHash });

await user.save();

res.status(201).send('User created');

} catch (err) {

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

}

});

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

const { email, password } = req.body;

try {

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

if (!user || !(await bcrypt.compare(password, user.passwordHash))) {

return res.status(401).send('Unauthorized');

}

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

res.json({ token });

} catch (err) {

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

}

});

export default router;

**Create src/routes/postRoutes.ts:**

import { Router } from 'express';

import jwt from 'jsonwebtoken';

import Post from '../models/Post';

import User from '../models/User';

const router = Router();

const authenticate = (req: any, res: any, next: any) => {

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

if (!token) return res.status(401).send('Unauthorized');

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

if (err) return res.status(401).send('Unauthorized');

req.user = decoded;

next();

});

};

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

const { title, content } = req.body;

try {

const post = new Post({ title, content, authorId: req.user.id });

await post.save();

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

} catch (err) {

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

}

});

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

const { author } = req.query;

try {

const posts = author ? await Post.find({ authorId: author }) : await Post.find();

res.json(posts);

} catch (err) {

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

}

});

export default router;