**DAY5: AUTH**

ADD JWT AUTH  
  
**1)Add user model to backend**  
(in models/user.js or index.js)

**2)install AUTH packages**npm install bcryptjs jsonwebtoken

**3)create auth endpoints**  
**4)protect todo endpoints with jwt**

CODE:

//user model and schema

require('dotenv').config();

const JWT\_SECRET = process.env.JWT\_SECRET;

const express=require('express');

const cors=require('cors');

const mongoose=require('mongoose');

const bcrypt=require('bcryptjs');

const jwt=require('jsonwebtoken');

const app=express();

app.use(cors({

  origin: ['http://localhost:3000', 'https://your-frontend.vercel.app'], // or your React app's URL

  methods: ['GET', 'POST', 'PUT', 'DELETE'],

  credentials: true // if using cookies or auth headers

}));

app.use(express.json());

**//mongo**

mongoose.connect('mongodb+srv://nehagade715:iDnC2Wq9BN1qOWU2@cluster0.vvf0s6k.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0',{

    useNewUrlParser:true,

    useUnifiedTopology:true,

})

.then(()=>console.log('mongodb connected'))

.catch(err=>console.error('error connection:', err));

**//schema-mongo**

const todoSchema=new mongoose.Schema({

    text:String,

    done:Boolean,

    userId:mongoose.Schema.Types.ObjectId,             //add this

});

const Todo=mongoose.model('Todo',todoSchema);

**//new user schema for auth**

const userSchema = new mongoose.Schema({

  username: { type: String, unique: true },

  password: String, // Store hashed password!

});

const User = mongoose.model('User', userSchema);

**//auth**

app.get('/debug-my-todos', async (\_req, res) => {

  const todos = await Todo.find({});

  res.json(todos);

});

**//create auth endpoints**

**//register**

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

    const{username,password}=req.body;

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

try{

    const user=await User.create({username,password:hashed});

    res.status(201).json({message:'User created'});

}

catch(e)

{

    res.status(400).json({error:'username taken'});

}

});

**//login**

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

    const {username,password}=req.body;

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

    if(!user)return res.status(400).json(

        {error:'/invalid username'}

    );

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

    if(!isMatch) return res.status(400).json(

        {error:'invalid pwd'}

    );

**// CHANGE 1: Add userId to response**

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

  res.json({ token, userId: user.\_id }); // <-- Added userId

});

**//protect todo endpoints with jwt**

**//add a middleware fnc auth**

function auth(req,res,next){

    const authHeader=req.headers.authorization;

    if(!authHeader)

        return res.status(401).json({error:'no token'});

    const token=authHeader.split(' ')[1];

    try{

        const payload=jwt.verify(token,JWT\_SECRET);

        req.user=payload;

        next();

    }

    catch{

        res.status(401).json({error:'invalid token'});

    }

}

**//use this middleware in your todo routes**

app.get('/',auth,async(\_req,res)=>{

    res.send('welcome to todo api');

});

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

    const todos = await Todo.find({ userId: req.user.userId });

    res.json(todos);

});

app.post('/todos',auth,async(req,res)=>{

    const todo = new Todo({

    text: req.body.text,

    done: req.body.done,

    userId: req.user.userId // Convert string to ObjectId //set jwt to payload

  });

    await todo.save();

    res.status(201).json(todo);

});

app.delete('/todos/:id',auth,async(req,res)=>{

    // await Todo.findByIdAndDelete(req.params.id);

    await Todo.deleteOne({ \_id: req.params.id, userId: req.user.userId });

    res.status(204).end();

});

app.put('/todos/:id',auth,async(req,res)=>{

    const todo=await Todo.findById(req.params.id);

    if(!todo) return res.status(404).json({error:'Todo not found'});

    todo.done=!todo.done;

    await todo.save();

    res.json(todo);

});

app.listen(5000,()=>{

    console.log('Server running on http://localhost:5000 ');

});

6)

**//add login nregister form in react and update for auth by sending the token in headers**

//for build

// At the end of your Express index.js

const path = require('path');

app.use(express.static(path.join(\_\_dirname, 'build')));

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

  res.sendFile(path.join(\_\_dirname, 'build', 'index.html'));

});