**Backend - World News Application**

1. Project Setup

First, I initialized the project and installed the necessary packages:

**npm init -y**

**npm install express mongoose bcryptjs jsonwebtoken dotenv**

Then, I created the folder structure:

backend/

│

├── config/

│ └── db.js

├── middleware/

│ └── authMiddleware.js

├── models/

│ ├── User.js

│ ├── News.js

│ └── Comment.js

├── routes/

│ ├── authRoutes.js

│ ├── newsRoutes.js

│ └── commentRoutes.js

├── .env

├── server.js

└── package.jsonpackage.json

**2. Environment Configuration**

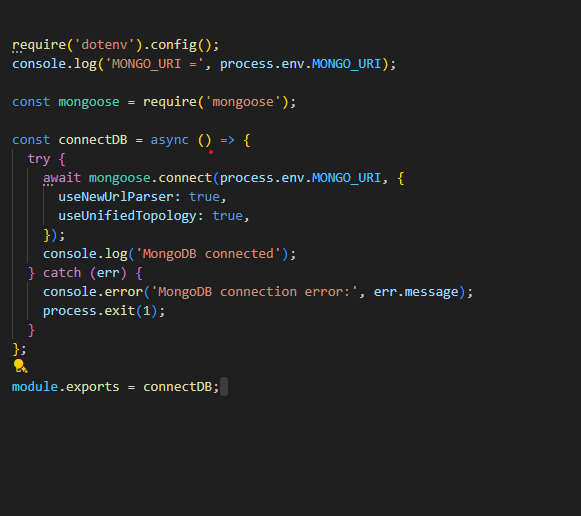
📄 .env

PORT=5000

MONGO\_URI=mongodb://localhost:27017/worldNewsDB

JWT\_SECRET=myverysecretkey

**3. Connecting to MongoDB**



**4. Models**

📄 models/User.js

constmongoose=require('mongoose');

constUserSchema=newmongoose.Schema({

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

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

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

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

});

module.exports=mongoose.model('User', UserSchema);

**📄 models/News.js**

//models/News.js

constmongoose=require('mongoose');

constNewsSchema=newmongoose.Schema({

authorId:{ type:mongoose.Schema.Types.ObjectId, ref:'User', required:true},

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

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

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

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

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

});

module.exports=mongoose.model('News', NewsSchema);

📄 models/Comment.js

constmongoose=require('mongoose');

constCommentSchema=newmongoose.Schema({

newsId:{ type:mongoose.Schema.Types.ObjectId, ref:'News', required:true},

authorId:{ type:mongoose.Schema.Types.ObjectId, ref:'User', required:true},

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

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

});

module.exports=mongoose.model('Comment', CommentSchema);

**5. Authentication Middleware**

📄 middleware/authMiddleware.js

***~~//authMiddleware.js~~***

***~~constjwt=require('jsonwebtoken');~~***

***~~constauthMiddleware=(req, res, next) =>{~~***

***~~consttoken=req.header('Authorization')?.replace('Bearer ', '');~~***

***~~if(!token) {~~***

***~~returnres.status(401).json({ message:'Authorization token missing'});~~***

***~~}~~***

***~~try{~~***

***~~constdecoded=jwt.verify(token, process.env.JWT\_SECRET);~~***

***~~req.user=decoded; // ruan info të user-it në request~~***

***~~next();~~***

***~~} catch(err) {~~***

***~~returnres.status(401).json({ message:'Invalid token'});~~***

***~~}~~***

***~~};~~***

***~~module.exports=authMiddleware;~~***

**6. Routes**

📄 routes/authRoutes.js

constexpress=require('express');

constbcrypt=require('bcryptjs');

constjwt=require('jsonwebtoken');

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

constrouter=express.Router();

// Register

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

try{

const{ username, email, password} =req.body;

constexistingUser=awaitUser.findOne({ email});

if(existingUser) returnres.status(400).json({ message:'Email already in use'});

consthashedPassword=awaitbcrypt.hash(password, 10);

constuser=newUser({ username, email, passwordHash:hashedPassword});

awaituser.save();

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

} catch(err) {

res.status(500).json({ message:'Server error'});

}

});

// Login

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

try{

const{ email, password} =req.body;

constuser=awaitUser.findOne({ email});

if(!user) returnres.status(400).json({ message:'Invalid credentials'});

constisMatch=awaitbcrypt.compare(password, user.passwordHash);

if(!isMatch) returnres.status(400).json({ message:'Invalid credentials'});

consttoken=jwt.sign({ id:user.\_id, username:user.username}, process.env.JWT\_SECRET, {

expiresIn:'1h',

});

res.json({ token});

} catch(err) {

res.status(500).json({ message:'Server error'});

}

});

module.exports=router;

📄 routes/newsRoutes.js

constexpress=require('express');

constNews=require('../models/News');

constauthMiddleware=require('../middleware/authMiddleware');

constrouter=express.Router();

// CREATE news

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

try{

const{ title, content, category} =req.body;

if(!title||!content||!category) {

returnres.status(400).json({ message:'Please provide title, content, and category'});

}

constnews=newNews({

authorId:req.user.id, // nga tokeni

title,

content,

category,

});

awaitnews.save();

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

} catch(err) {

console.error("Error in POST /news:", err);

res.status(500).json({ message:'Server error'});

}

});

// UPDATE news

router.put('/:id', authMiddleware, async(req, res) =>{

try{

constnews=awaitNews.findById(req.params.id);

if(!news) returnres.status(404).json({ message:'News not found'});

if(!news.authorId) {

returnres.status(400).json({ message:'News is missing authorId'});

}

if(news.authorId.toString() !==req.user.id) {

returnres.status(403).json({ message:'Not authorized'});

}

const{ title, content, category} =req.body;

news.title=title||news.title;

news.content=content||news.content;

news.category=category||news.category;

news.updatedAt=newDate();

awaitnews.save();

res.json(news);

} catch(err) {

console.error("Error in PUT /news/:id:", err);

res.status(500).json({ message:'Server error'});

}

});

// DELETE news

router.delete('/:id', authMiddleware, async(req, res) =>{

try{

constnews=awaitNews.findById(req.params.id);

if(!news) returnres.status(404).json({ message:'News not found'});

if(!news.authorId) {

returnres.status(400).json({ message:'News is missing authorId'});

}

if(news.authorId.toString() !==req.user.id) {

returnres.status(403).json({ message:'Not authorized'});

}

awaitnews.remove();

res.json({ message:'News deleted successfully'});

} catch(err) {

console.error("Error in DELETE /news/:id:", err);

res.status(500).json({ message:'Server error'});

}

});

// GET all news

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

try{

constnews=awaitNews.find().sort({ createdAt:-1});

res.json(news);

} catch(err) {

res.status(500).json({ message:'Server error'});

}

});

// GET one news by ID

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

try{

constnews=awaitNews.findById(req.params.id);

if(!news) returnres.status(404).json({ message:'News not found'});

res.json(news);

} catch(err) {

res.status(500).json({ message:'Server error'});

}

});

module.exports=router;

📄 routes/commentRoutes.js

constexpress=require('express');

constComment=require('../models/Comment');

constauthMiddleware=require('../middleware/authMiddleware');

constrouter=express.Router();

// Marr të gjitha komentet për një lajm

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

try{

constcomments=awaitComment.find({ newsId:req.params.newsId})

.populate('authorId', 'username')

.sort({ createdAt:-1});

res.json(comments);

} catch(err) {

console.error('Error in GET /comments/news/:newsId:', err);

res.status(500).json({ message:'Server error'});

}

});

// POST një koment të ri për një lajm

router.post('/news/:newsId', authMiddleware, async(req, res) =>{

try{

const{ text} =req.body;

if(!text) {

returnres.status(400).json({ message:'Comment text is required'});

}

constcomment=newComment({

newsId:req.params.newsId,

authorId:req.user.id,

text,

});

awaitcomment.save();

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

} catch(err) {

console.error('Error in POST /comments/news/:newsId:', err);

res.status(500).json({ message:'Server error'});

}

});

module.exports=router;

**7. Main Server File**

//server.js

constexpress=require('express');

constconnectDB=require('./config/db');

constauthRoutes=require('./routes/authRoutes');

constnewsRoutes=require('./routes/newsRoutes');

constcommentRoutes=require('./routes/commentRoutes');

require('dotenv').config();

constapp=express();

// Middleware

app.use(express.json());

// Connect DB

connectDB();

// Routes

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

app.use('/api/news', newsRoutes);

app.use('/api/comments', commentRoutes);

constPORT=process.env.PORT||5000;

app.listen(PORT, () =>console.log(`Server running on port ${PORT}`));

**Postman API Testing**

1. User Registration (POST /api/auth/register)

**Request URL:** http://localhost:5000/api/auth/register

**Method:** POST

**Request Body (JSON):**

**{**

**"username": "testuser",**

**"email": "testuser@example.com",**

**"password": "password123"**

**}**

Status code 201 with message "User registered successfully".

**2. User Login (POST /api/auth/login)**

**Request URL:** http://localhost:5000/api/auth/login

**Method:** POST

**Request Body (JSON):**

**{**

**"email": "testuser@example.com",**

**"password": "password123"**

**}**

JSON containing a JWT token for authentication.

**3. Create News (POST /api/news)**

**Request URL:** http://localhost:5000/api/news

**Method:** POST

**Headers:**  
Authorization: Bearer <JWT\_TOKEN>

**Request Body (JSON):**

**{**

**"title": "My first news",**

**"content": "This is the content of the news.",**

**"category": "Politics"**

**}**

JSON with the newly created news object including the \_id.

**4. Get News by ID (GET /api/news/:newsId)**

**Request URL:** http://localhost:5000/api/news/<newsId>

**Method:** GET

**Expected Response:**  
JSON object with news details matching the provided ID.

**5. Add Comment to News (POST /api/comments/news/:newsId)**

**Request URL:** http://localhost:5000/api/comments/news/<newsId>

**Method:** POST

**Headers:**  
Authorization: Bearer <JWT\_TOKEN>

**{**

**"text": "This is a comment."**

**}**

JSON with the newly created comment object.

**6. Get Comments for News (GET /api/comments/news/:newsId)**

**Request URL:** http://localhost:5000/api/comments/news/<newsId>

**Method:** GET

**Expected Response:**  
JSON array containing all comments related to the specified news item.