**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**

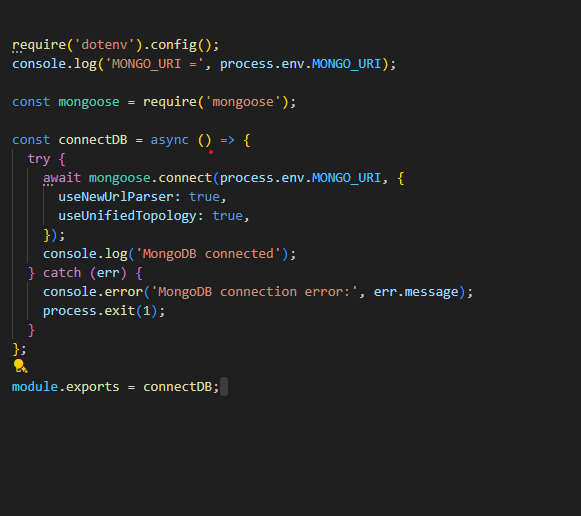
📄 .env

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

const express = require('express');

const path = require('path');

const connectDB = require('./config/db');

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

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

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

const expressLayouts = require('express-ejs-layouts');

require('dotenv').config();

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

// Set EJS as view engine dhe layout

app.use(expressLayouts);

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

app.set('layout', 'layout');

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

// Static folder për CSS, js, img

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

connectDB();

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

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

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

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

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

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

  try {

    const news = await News.find().sort({ createdAt: -1 }).lean();

    res.render('index', { title: 'Home', news });

  } catch (err) {

    console.error(err);

    res.status(500).send('Server error');

  }

});

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

  try {

    const { title, content } = req.body;

    if (!title || !content) {

      return res.status(400).send('Title and content are required');

    }

    const newNews = new News({

      title,

      content,

      createdAt: new Date(),

    });

    await newNews.save();

    res.redirect('/');

  } catch (err) {

    console.error(err); // e lë këtu në catch

    res.status(500).send('Server error');

  }

});

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

  try {

    const news = await News.findById(req.params.id).lean();

    if (!news) return res.status(404).send('News not found');

    const comments = await Comment.find({ newsId: news.\_id })

      .populate('authorId', 'username')

      .sort({ createdAt: -1 })

      .lean();

    res.render('news', { title: news.title, news, comments });

  } catch (err) {

    console.error(err);

    res.status(500).send('Server error');

  }

});

// Start server

const PORT = process.env.PORT || 5000;

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

**Postman API Testing**

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

1. **Request URL:** http://localhost:5000/api/auth/register
2. **Method:** POST
3. **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)**

1. **Request URL:** http://localhost:5000/api/auth/login
2. **Method:** POST
3. **Request Body (JSON):**

{

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

"password": "password123"

}

JSON containing a JWT token for authentication.

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

1. **Request URL:** http://localhost:5000/api/news
2. **Method:** POST
3. **Headers:**  
   Authorization: Bearer <JWT\_TOKEN>
4. **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)**

1. **Request URL:** http://localhost:5000/api/news/<newsId>
2. **Method:** GET

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

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

1. **Request URL:** http://localhost:5000/api/comments/news/<newsId>
2. **Method:** POST
3. **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)**

1. **Request URL:** http://localhost:5000/api/comments/news/<newsId>
2. **Method:** GET
3. **Expected Response:**  
   JSON array containing all comments related to the specified news item.

**Frontend**

1.views

createNews.js

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title><%= title %></title>

  <style>

    body { font-family: Arial, sans-serif; max-width: 700px; margin: auto; padding: 20px; }

    input, textarea { width: 100%; padding: 8px; margin: 5px 0; }

    button { padding: 10px 15px; background-color: #007bff; color: white; border: none; cursor: pointer; }

    button:hover { background-color: #0056b3; }

  </style>

</head>

<body>

  <h1>Create a News Item</h1>

  <form action="/news/create" method="POST">

    <input type="text" name="title" placeholder="Title" required />

    <textarea name="content" rows="6" placeholder="Content" required></textarea>

    <button type="submit">Create</button>

  </form>

  <p><a href="/">Back to Home</a></p>

</body>

</html>

Index.ejs

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>World News</title>

  <style>

    body {

  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

  background-color: #eef2f7;

  margin: 0;

  padding: 0;

  color: #333;

}

.container {

  max-width: 900px;

  margin: 30px auto;

  background: #fff;

  padding: 25px 30px;

  border-radius: 10px;

  box-shadow: 0 6px 15px rgba(0,0,0,0.1);

}

h1, h2 {

  color: #2c3e50;

  margin-bottom: 20px;

  font-weight: 700;

}

.news-item {

  border: 1px solid #ddd;

  border-radius: 8px;

  padding: 18px 22px;

  margin-bottom: 18px;

  background: #f9fafc;

  transition: transform 0.25s ease, box-shadow 0.25s ease;

  cursor: pointer;

}

.news-item:hover {

  background: #fff;

  box-shadow: 0 8px 20px rgba(44, 62, 80, 0.12);

  transform: translateY(-3px);

}

.news-item a {

  font-size: 1.1rem;

  font-weight: 600;

  color: #2980b9;

  text-decoration: none;

  transition: color 0.3s ease;

}

.news-item a:hover {

  color: #1c5980;

  text-decoration: underline;

}

.news-date {

  display: block;

  margin-top: 6px;

  font-size: 0.85rem;

  color: #7f8c8d;

}

footer {

  text-align: center;

  padding: 15px 0;

  font-size: 0.9rem;

  color: #95a5a6;

  border-top: 1px solid #ddd;

  margin-top: 40px;

}

.news-form {

  margin-bottom: 30px;

  display: flex;

  flex-direction: column;

  gap: 12px;

}

.news-form input,

.news-form textarea {

  padding: 10px 15px;

  border: 1px solid #ccc;

  border-radius: 6px;

  font-size: 1rem;

  resize: none;

  font-family: inherit;

}

.news-form button {

  width: 140px;

  padding: 10px 0;

  background-color: #2980b9;

  color: white;

  font-weight: 600;

  border: none;

  border-radius: 6px;

  cursor: pointer;

  transition: background-color 0.3s ease;

}

.news-form button:hover {

  background-color: #1c5980;

}

  </style>

</head>

<body>

  <div class="container">

  <form action="/news/create" method="POST" class="news-form">

  <input type="text" name="title" placeholder="News Title" required />

  <textarea name="content" placeholder="News Content" rows="4" required></textarea>

  <button type="submit">Create News</button>

</form>

  <h2>Latest News</h2>

  <% if (news && news.length) { %>

    <% news.forEach(n => { %>

      <div class="news-item" onclick="window.location.href='/news/<%= n.\_id %>'">

        <a href="/news/<%= n.\_id %>"><%= n.title %></a>

        <small class="news-date"><%= n.createdAt ? new Date(n.createdAt).toLocaleDateString() : 'No Date' %></small>

      </div>

    <% }) %>

  <% } else { %>

    <p>No news found.</p>

  <% } %>

  <footer>Copyright © 2025</footer>

</div>

</body>

</html>

Layout.ejs

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <head>

  <title><%= title || 'World News' %></title>

  <link rel="stylesheet" href="../public/style.css" />

</head>

</head>

<body>

  <header>

    <h1></h1>

  </header>

  <main>

    <%- body %>

  </main>

</body>

</html>

News.ejs

<h2><%= news.title %></h2>

<p><%= news.content %></p>

<p><small>Category: <%= news.category %></small></p>

<h3>Comments</h3>

<% if (comments && comments.length) { %>

  <ul>

    <% comments.forEach(c => { %>

      <li><strong><%= c.authorId.username %></strong>: <%= c.text %></li>

    <% }) %>

  </ul>

<% } else { %>

  <p>No comments yet.</p>

<% } %>