**Project**

This project is a Simple Blog System built using Node.js, Express, MongoDB, and EJS. It allows administrators to manage blog posts by adding, editing, deleting, and uploading associated images. The platform also implements basic authentication using JWT (JSON Web Tokens) for secure access.

**Features Implemented**

**1. Authentication**

Login System:

* Admin users can log in using a username and password.
* Passwords are securely hashed using bcrypt.
* JWT is used to manage user sessions with tokens stored in cookies.

Middleware for Authorization:

* Ensures only authenticated users can access sensitive routes like the admin panel or manage posts.

**2. Post Management**

Add New Posts:

* Admins can add new posts with:
  + Title
  + Body/content
  + Image (uploaded via a form from the local system).

Edit Posts:

* Admins can update existing posts, including replacing or adding images.

Delete Posts:

* Admins can delete posts by ID.

**3. Image Upload**

File Handling:

* Implemented using Multer, which allows file uploads.
* Images are stored in a local uploads/ folder.
* Images are given unique names to prevent overwriting.

**4. Dynamic Routing**

Routes are set up for:

* /admin (login page).
* /dashboard (admin panel for managing posts).
* /add-post (adding new posts).
* /edit-post/:id (editing specific posts).
* /delete-post/:id (deleting specific posts).
* /logout (clearing session cookies).

**Code Summary**

**1. Authentication Middleware**

The following middleware ensures only authenticated users can access protected routes:

javascript

Копировать код

const authMiddleware = (req, res, next) => {

const token = req.cookies.token;

if (!token) {

return res.status(401).json({ message: 'Unauthorized' });

}

try {

const decoded = jwt.verify(token, jwtSecret);

req.userId = decoded.userId;

next();

} catch (error) {

res.status(401).json({ message: 'Unauthorized' });

}

};

**2. Image Upload Configuration**

Using Multer for handling file uploads:

javascript

Копировать код

const storage = multer.diskStorage({

destination: (req, file, cb) => {

cb(null, 'uploads/');

},

filename: (req, file, cb) => {

cb(null, `${Date.now()}-${file.originalname}`);

}

});

const upload = multer({ storage });

**3. Post Model**

The schema for blog posts in MongoDB:

javascript

Копировать код

const PostSchema = new mongoose.Schema({

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

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

imagePath: { type: String, default: null },

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

updatedAt: { type: Date }

});

**4. Routes**

**Add a New Post**

javascript

Копировать код

router.post('/add-post', authMiddleware, upload.single('image'), async (req, res) => {

try {

const newPost = new Post({

title: req.body.title,

body: req.body.body,

imagePath: req.file ? `/uploads/${req.file.filename}` : null

});

await Post.create(newPost);

res.redirect('/dashboard');

} catch (error) {

console.log(error);

}

});

**Edit an Existing Post**

javascript

Копировать код

router.put('/edit-post/:id', authMiddleware, upload.single('image'), async (req, res) => {

try {

const updateData = {

title: req.body.title,

body: req.body.body,

updatedAt: Date.now()

};

if (req.file) {

updateData.imagePath = `/uploads/${req.file.filename}`;

}

await Post.findByIdAndUpdate(req.params.id, updateData);

res.redirect(`/edit-post/${req.params.id}`);

} catch (error) {

console.log(error);

}

});

**Delete a Post**

javascript

Копировать код

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

try {

await Post.deleteOne({ \_id: req.params.id });

res.redirect('/dashboard');

} catch (error) {

console.log(error);

}

});

**Setup Instructions**

**1. Dependencies**

Install the necessary packages:

bash

Копировать код

npm install express ejs bcrypt jsonwebtoken mongoose multer dotenv cookie-parser method-override express-session connect-mongo

**2. Environment Variables**

Create a .env file with the following content:

plaintext

Копировать код

MONGODB\_URI=mongodb://localhost:27017/blogdb

JWT\_SECRET=your\_jwt\_secret

**3. Folder Structure**

Ensure the following folder structure:

arduino

Копировать код

project/

│

├── models/

│ ├── Post.js

│ └── User.js

│

├── views/

│ ├── admin/

│ │ ├── dashboard.ejs

│ │ ├── add-post.ejs

│ │ ├── edit-post.ejs

│ │ └── index.ejs

│ └── layouts/

│ └── admin.ejs

│

├── public/

│ └── uploads/

│

├── server/

│ ├── config/

│ │ └── db.js

│ ├── helpers/

│ │ └── routeHelpers.js

│ └── routes/

│ ├── main.js

│ └── admin.js

│

├── .env

├── app.js

├── package.json

└── README.md

**Conclusion**

This project provides a robust starting point for a content management system. It incorporates secure login, file handling, and CRUD operations for posts, making it an excellent foundation for a scalable blog platform.





