Lab Day 4: Authentication & Authorization

Challenge

Add user authentication, authorization, and data validation to your Blog API from Day 3.

Setup

1. Continue with your project from Day 3 or create a new one
2. Install additional packages: `bcrypt`, `jsonwebtoken`, `joi`
3. Update your file structure:

```
blog-api/
├── server.js
├── models/
│   ├── User.js
│   └── Post.js
├── routes/
│   ├── userRoutes.js
│   ├── postRoutes.js
│   └── authRoutes.js
├── middleware/
│   ├── auth.js
│   └── validation.js
└── controllers/
    └── authController.js
```

Requirements

1. User Authentication

- Create a proper User model with secure password storage
- Implement user registration endpoint: `POST /api/auth/register`
- Implement login endpoint: `POST /api/auth/login` that returns a JWT token
- Create middleware to verify JWT token

2. Request Validation with Joi

- Create validation schemas for:
    - User registration

    - User login
    - Post creation/updating
- Implement middleware that validates requests before processing

3. Authorization

- Implement role-based authorization (regular users and admins)
- Restrict post editing/deletion to the author or admins
- Make certain routes available only to admins

4. Password Security

- Implement password hashing with bcrypt
- Add password complexity requirements using Joi
- Store only hashed passwords in the database

5. Mongoose Data Modeling (Choose at least 1)

- Implement virtual properties on your User model (e.g., fullName that combines firstName and lastName)
- Create pre/post hooks (middleware) on your schemas
- Add custom instance or static methods to your models
- Use populate to retrieve related documents efficiently

6. Testing Your Authentication

- Create a protected route that requires authentication
- Test your authentication flow with a tool like Postman or Insomnia

Tips & Resources

Authentication

- Look up "JWT authentication Express" for examples of implementing JSON Web Tokens
- Search for "bcrypt password hashing Node.js" to learn about secure password storage

Joi Validation

- Check "Joi schema validation Express" for integration examples
- Use Joi's built-in password complexity options

Mongoose Features

- Search for "Mongoose virtual properties" to learn about computed fields
- Look up "Mongoose middleware pre post hooks" for schema hooks

- Search "Mongoose populate" for efficient document relationships

JWT Security

- Keep your JWT secret key in an environment variable
- Set reasonable expiration times on tokens
- Consider using refresh tokens for better security

Remember to handle errors gracefully and provide meaningful error messages to users!