PHỤ LỤC BÁO CÁO CÁ NHÂN

---

Họ và tên: Vũ Ngọc Ánh

MSSV: 20221836

Nhóm: group08

Tên đề tài: Diễn đàn hỏi đáp và chia sẻ tài liệu học tập

-----------------------------------------------------------------------------------

1. Lý do lựa chọn middleware trong một trường hợp cụ thể (1 điểm)

1.1. Miêu tả trường hợp

* Chức năng cụ thể cần xử lý: Kiểm tra xác thực người dùng (Authentication) khi người dùng truy cập các route yêu cầu đã đăng nhập.
* Tình huống cụ thể:  
  Trong dự án hệ thống chia sẻ tài liệu BKDocs, nhóm sử dụng middleware auth.js cho chức năng kiểm tra token JWT kèm theo trong request. Một số chức năng như đăng bài, chỉnh sửa bài, tải file, bình luận… yêu cầu người dùng phải đăng nhập.

Node.js/Express không tự động xử lý JWT token và không có sẵn cơ chế kiểm tra người dùng từ token → do đó nhóm sử dụng middleware kết hợp với thư viện jsonwebtoken để xử lý logic này.

1.2. Lý do chọn middleware / thư viện bên thứ 3

* jsonwebtoken là thư viện phổ biến, đáng tin cậy, hỗ trợ chuẩn JWT.
* Có cộng đồng lớn, tài liệu rõ ràng, dễ tích hợp vào Express.
* Đã được kiểm chứng về tính an toàn và hiệu quả.
* Tự code giải mã token rất dễ bị lỗi hoặc mất thời gian → dùng thư viện giúp tăng độ an toàn, tiết kiệm thời gian.

1.3. Middleware được chọn

* Tên: middlewares/auth.js
* Vai trò: Middleware kiểm tra token JWT, xác minh người dùng, đảm bảo các request đến các route quan trọng phải có token hợp lệ.
* Nhiệm vụ:
  + Đọc JWT token từ header Authorization của request.
  + Giải mã token bằng jsonwebtoken.
  + Kiểm tra user có tồn tại trong database không (nguoidung table).
  + Nếu hợp lệ → gán thông tin user vào req.user, cho phép tiếp tục.
  + Nếu không hợp lệ → trả về lỗi 401 Unauthorized.
* Lý do chọn:
  + Cách triển khai đơn giản, rõ ràng, dễ đọc và dễ bảo trì.
  + jsonwebtoken tương thích tốt với Express middleware.
  + Giải pháp hiệu quả để bảo vệ các API cần quyền hạn (authorization).

2. Giải thích đường đi từ request đến response (1 điểm)

* Gửi từ giao diện (client):  
  Người dùng đăng nhập thành công → nhận được JWT token → các request sau (ví dụ: đăng bài, upload file, xóa bài) sẽ gửi token kèm theo trong header Authorization: Bearer <token>.
* Qua route:  
  Các route trong Express có cấu hình middleware:

app.post('/api/posts', authMiddleware, postController.createPost);

* Middleware xử lý:  
  auth.js middleware thực hiện:  
  → Kiểm tra header Authorization → lấy token → giải mã bằng jwt.verify()  
  → Kiểm tra user trong DB → nếu hợp lệ gán vào req.user → gọi next().  
  → Nếu lỗi → trả về 401 Unauthorized.
* Controller xử lý logic:  
  Nếu next() được gọi → controller thực hiện logic chính như: đọc dữ liệu từ req.body, req.user, xử lý nghiệp vụ → lưu vào CSDL.
* Truy cập CSDL:  
  Ví dụ khi đăng bài → INSERT INTO posts (title, content, author\_id, ...)  
  author\_id lấy từ req.user.nguoi\_dung\_id.
* Controller trả kết quả:  
  Trả về JSON: { success: true, message: "Đăng bài thành công" }
* Client nhận và hiển thị:  
  Giao diện nhận response → hiển thị thông báo thành công hoặc lỗi.

3. Giải thích vận hành website mình làm, tập trung vào phần việc được phân (1 điểm)

Phần việc mình thực hiện bao gồm:

Chức năng: Đăng ký, đăng nhập người dùng; đăng bài viết; chỉnh sửa/xóa bài viết; kiểm tra xác thực người dùng (middleware auth.js).

Mô tả chi tiết cách phần đó hoạt động, tương tác với các thành phần khác:

• Frontend sử dụng gì:

Giao diện người dùng được xây dựng bằng HTML, CSS, JavaScript.  
Các form như đăng ký, đăng nhập, tạo bài viết đều được thiết kế tại đây.  
Khi người dùng thao tác (nhập dữ liệu, nhấn nút), frontend gửi request đến backend thông qua API bằng fetch() hoặc axios.

• Backend xử lý gì:

Backend sử dụng Node.js với Express.

* Đăng ký/đăng nhập:
  + Backend kiểm tra thông tin, mã hóa mật khẩu bằng bcrypt, lưu vào CSDL, hoặc kiểm tra và trả về JWT token.
* Đăng bài, chỉnh sửa, xóa bài:
  + Backend sử dụng middleware auth.js để kiểm tra token.
  + Kiểm tra quyền người dùng → xử lý logic chính → lưu thông tin vào CSDL.

• CSDL truy vấn ra sao:

Sử dụng MySQL.

* Khi đăng ký:
* INSERT INTO nguoidung (username, password\_hash, email, ...) VALUES (...);
* Khi đăng nhập:
* SELECT \* FROM nguoidung WHERE email = ?;
* Khi đăng bài:
* INSERT INTO posts (title, content, author\_id, created\_at) VALUES (...);
* Khi chỉnh sửa/xóa bài:
* UPDATE posts SET ... WHERE post\_id = ? AND author\_id = ?;

Tóm lại:  
Middleware auth.js giúp bảo vệ các API yêu cầu quyền hạn, đảm bảo chỉ người dùng đã xác thực mới có thể thực hiện các hành động như đăng bài, chỉnh sửa, xóa bài.  
Việc kết hợp jsonwebtoken giúp việc xử lý JWT dễ dàng, bảo mật và hiệu quả.