PHỤ LỤC BÁO CÁO CÁ NHÂN

Họ tên: Vũ Hồng Nhung

MSSV: 20221864

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: Tải lên tài liệu (file upload) cho chức năng đăng bài

- Tình huống cụ thể:

Trong dự án hệ thống chia sẻ tài liệu BKDocs, nhóm sử dụng middleware multer cho chức năng **tải lên tài liệu (file upload)**. Middleware này được chọn vì Node.js không hỗ trợ xử lý multipart/form-data (loại dữ liệu được gửi khi upload file) một cách tự động. Do đó, multer giúp tách và xử lý các phần dữ liệu là tệp đính kèm từ client.

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

- multer là middleware phổ biến, đáng tin cậy, hỗ trợ cấu hình chi tiết về nơi lưu file, đổi tên file, giới hạn dung lượng,...

- Có cộng đồng lớn, tài liệu rõ ràng, dễ tích hợp vào Express.

- So với các thư viện như formidable, busboy thì multer đơn giản hơn để tích hợp nhanh cho nhu cầu cơ bản đến trung bình.

**1.3. Middleware được chọn**

**- Tên:** multer

- **Vai trò:** Middleware xử lý multipart/form-data để lấy file từ request gửi từ client.

- **Nhiệm vụ:**

+ Đọc dữ liệu từ form.

+ Lưu file vào thư mục chỉ định trên server.

+ Trả về thông tin file để backend sử dụng (ví dụ: tên file, kích thước, đường dẫn,...)

**- Lý do chọn:** Dễ dùng, cấu hình rõ ràng, tương thích với Express, hiệu quả cho bài toán upload file kèm bài viết.

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

**- Gửi từ form trên giao diện:**  
Người dùng nhập tiêu đề, nội dung và chọn file đính kèm → nhấn "Đăng bài" → fetch() gửi POST request với FormData đến /api/posts.

**- Qua route:**  
Route Express nhận request:  
app.post('/api/posts', upload.single('file'), postController.createPost)

**- Middleware xử lý:**  
multer xử lý file upload → lưu vào thư mục uploads/ → đính kèm thông tin vào req.file.

**- Controller xử lý logic:**  
Đọc dữ liệu từ req.body và req.file → kiểm tra hợp lệ → chuẩn bị dữ liệu lưu CSDL.

**- Truy cập CSDL nếu cần:**  
Lưu thông tin bài viết và đường dẫn file vào bảng posts trong MySQL.

**- Controller trả kết quả:**  
Trả về JSON: { success: true, message: "Đăng thành công" }

**- Render view (hoặc API trả JSON):**  
API trả JSON về cho client.

**- Client nhận và hiển thị:**  
Giao diện hiển thị thông báo thành công và cập nhật danh sách bài viết.

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

Phần việc mình thực hiện bao gồm:

**3.1. Chức năng: Đăng ký tài khoản**

**Mô tả:**

* Người dùng nhập tên, email, mật khẩu và ngày sinh để tạo tài khoản mới.

**Frontend:**

* Giao diện: Dangky.html
* Dữ liệu được gửi qua fetch tới /auth/register
* Kiểm tra hợp lệ, hiển thị thông báo thành công hoặc lỗi

**Backend:**

* API /auth/register trong authController.js
* Kiểm tra email trùng
* Mã hoá mật khẩu (dùng bcrypt)
* Lưu thông tin vào CSDL

**CSDL:**

* Bảng nguoidung
* Truy vấn:

INSERT INTO nguoidung (ten\_nguoi\_dung, email, mat\_khau, ngay\_sinh) VALUES (?, ?, ?, ?)

**3.2. Chức năng: Đăng nhập tài khoản**

**Mô tả:**

* Người dùng nhập email và mật khẩu để đăng nhập, nhận token xác thực.

**Frontend:**

* Giao diện: Dangnhap.html
* Gửi dữ liệu qua fetch tới /auth/login
* Nếu thành công: lưu token vào localStorage

**Backend:**

* API /auth/login trong authController.js
* Kiểm tra email tồn tại
* So sánh mật khẩu bằng bcrypt.compare
* Tạo JWT token và gửi về client

**CSDL:**

* Bảng nguoidung
* Truy vấn:

SELECT \* FROM nguoidung WHERE email = ?

**3.3. Chức năng: Đăng bài viết**

**Mô tả:**

* Người dùng đăng nội dung (và file đính kèm nếu có).

**Frontend:**

* Giao diện: Baiviet.html
* Gửi nội dung và file qua fetch tới /posts
* Hiển thị thanh tiến trình tải

**Backend:**

* API POST /posts trong postController.js
* Kiểm tra tiêu đề, nội dung
* Lưu thông tin bài viết
* Nếu có file, lưu file và liên kết với tailieu

**CSDL:**

* Bảng baiviet, tailieu
* Truy vấn:

INSERT INTO baiviet (...) VALUES (...)

INSERT INTO tailieu (...) VALUES (...)

**3.4. Chức năng: Xoá bài viết**

**Mô tả:**

* Người dùng có thể xoá bài viết do mình đăng.

**Frontend:**

* Nút "Xoá" hiển thị nếu người dùng là chủ bài viết
* Gửi DELETE request tới /posts/:id

**Backend:**

* API DELETE /posts/:id
* Xác minh quyền (so sánh req.user.nguoi\_dung\_id)
* Nếu hợp lệ, xoá bài

**CSDL:**

* Bảng baiviet
* Truy vấn:

DELETE FROM baiviet WHERE bai\_viet\_id = ? AND nguoi\_dung\_id = ?

**3.5. Chức năng: Chỉnh sửa bài viết**

**Mô tả:**

* Người dùng sửa nội dung bài viết đã đăng.

**Frontend:**

* Nút "Chỉnh sửa" → hiển thị prompt → gửi PUT request
* Gửi dữ liệu tới /posts/:id

**Backend:**

* API PUT /posts/:id
* Kiểm tra quyền sở hữu
* Cập nhật nội dung bài viết trong DB

**CSDL:**

* Truy vấn:

UPDATE baiviet SET noi\_dung = ? WHERE bai\_viet\_id = ? AND nguoi\_dung\_id = ?

**3.6. Chức năng: Tìm kiếm bài viết theo từ khóa**

**Mô tả:**

* Người dùng nhập từ khóa để tìm bài viết theo tiêu đề hoặc nội dung.

**Frontend:**

* Giao diện: Search.html
* Nhập từ khóa, gửi POST request tới /posts/search
* Hiển thị danh sách bài viết phù hợp

**Backend:**

* API POST /posts/search
* Truy vấn bài viết có tieu\_de hoặc noi\_dung chứa từ khóa

**CSDL:**

* Truy vấn:

SELECT \* FROM baiviet WHERE tieu\_de LIKE ? OR noi\_dung LIKE ?