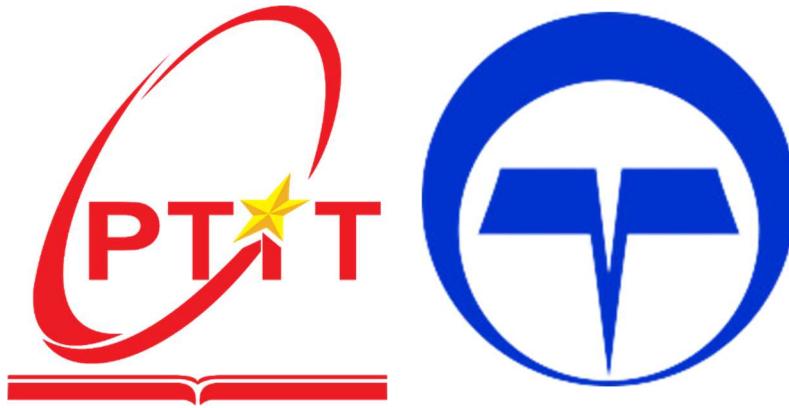


# HỌC VIỆN CÔNG NGHỆ BUƯU CHÍNH VIỄN THÔNG

## VIỆN KHOA HỌC KỸ THUẬT KINH TẾ BUƯU ĐIỆN



### BÀI TẬP LỚN MÔN : LẬP TRÌNH WEB ĐỀ TÀI: WEBSITE CHIA SẺ TÀI LIỆU

Học và tên: Nguyễn Anh Tuấn

Mã sinh viên: B23DCCC171

Mã lớp: RIPT1306-20241-16

Giáo viên giảng dạy: Vũ Văn Thương

# Mục Lục

<b>CHƯƠNG 1: MỞ ĐẦU .....</b>	4
<b>1.1. Lý do chọn đề tài.....</b>	4
<b>1.2. Nội dung và phạm vi đề tài.....</b>	4
<b>1.2.1    Nội dung đề tài.....</b>	4
<b>1.2.2    Phạm vi đề tài.....</b>	5
<b>1.3 Yêu cầu phi chức năng .....</b>	6
<b>CHƯƠNG 2: CÔNG NGHỆ SỬ DỤNG .....</b>	7
<b>2.1 ReactJS.....</b>	7
<b>2.2. Node.js.....</b>	7
<b>2.3 Cơ sở dữ liệu Mysql .....</b>	8
<b>2.4 SCSS.....</b>	10
<b>2.5 Rest API.....</b>	11
<b>2.6 Postman.....</b>	11
<b>CHƯƠNG 3: PHÂN TÍCH .....</b>	13
<b>3.1 Mô tả hệ thống.....</b>	13
<b>3.2 Biểu đồ ERD.....</b>	14
<b>CHƯƠNG 4: XÂY DỰNG DỰ ÁN .....</b>	19
<b>4.1 Xây dựng Cấu trúc Backend.....</b>	19
<b>4.1.1. Mô Hình Các Lớp .....</b>	19
<b>4.1.2. Các Middleware trong Dự án .....</b>	24
<b>4.1.3. Các Controller Backend.....</b>	27
<b>4.1.4. Các Routes Trong Dự Án .....</b>	35
<b>4.1.5. Cấu Hình Server Express.....</b>	39
<b>4.1.6. Cấu Hình Kết Nối MySQL và Kiểm Tra Kết Nối.....</b>	41
<b>4.2. Mô tả các giao diện chính.....</b>	43
<b>4.2.1. WEB dành cho Admin để quản lý .....</b>	43
<b>4.2.2. WEB dành cho người dùng .....</b>	46
<b>CHƯƠNG 5: KẾT LUẬN .....</b>	50
<b>TÀI LIỆU THAM KHẢO .....</b>	51

## **Lời Mở Đầu**

Trong bối cảnh phát triển mạnh mẽ của công nghệ thông tin và Internet, việc chia sẻ tài liệu học tập trực tuyến đã trở thành một xu hướng phổ biến và cần thiết trong ngành giáo dục. Việc cung cấp, chia sẻ và tiếp cận tài liệu học tập một cách nhanh chóng, dễ dàng và tiện lợi sẽ giúp học sinh, sinh viên và giảng viên tiết kiệm thời gian, nâng cao hiệu quả học tập và nghiên cứu. Tuy nhiên, việc quản lý và tổ chức các tài liệu học tập theo một hệ thống khoa học và hiệu quả vẫn là một vấn đề cần được giải quyết.

Nhận thấy sự cần thiết của việc phát triển một nền tảng chia sẻ tài liệu học tập, nhóm chúng em đã lựa chọn đề tài "Website chia sẻ tài liệu" với mục tiêu xây dựng một hệ thống trực tuyến cho phép người dùng dễ dàng tải lên, tải về và duyệt các tài liệu học tập. Hệ thống sẽ phân loại tài liệu theo các môn học, hỗ trợ nhiều định dạng tài liệu khác nhau như PDF, DOCX, và các tệp Microsoft Office, đồng thời cung cấp các tính năng quản lý tài liệu cho người dùng và admin.

Báo cáo này trình bày chi tiết về quá trình thiết kế, triển khai và vận hành website chia sẻ tài liệu, từ việc xác định yêu cầu hệ thống, lựa chọn công nghệ, đến các chức năng cụ thể như đăng tải tài liệu, tải về tài liệu, duyệt tài liệu và quản lý người dùng. Chúng em hy vọng rằng hệ thống này sẽ góp phần nâng cao chất lượng học tập và nghiên cứu, tạo ra một môi trường học tập linh hoạt, tiện lợi và hiệu quả cho cộng đồng người dùng.

# CHƯƠNG 1: MỞ ĐẦU

## 1.1 Lý do chọn đề tài

Trong thời đại cách mạng công nghiệp 4.0, sự phát triển mạnh mẽ của công nghệ thông tin đã tạo ra nhiều cơ hội và thách thức trong lĩnh vực giáo dục. Việc chia sẻ tài liệu học tập và nghiên cứu thông qua các nền tảng trực tuyến đang ngày càng phổ biến và cần thiết, đặc biệt đối với học sinh, sinh viên và giảng viên. Tuy nhiên, các nền tảng hiện có thường gặp một số hạn chế như:

- Giao diện phức tạp, gây khó khăn cho người sử dụng, đặc biệt là những người không am hiểu về công nghệ.
- Thiếu tính năng phân loại rõ ràng các loại tài liệu theo môn học hoặc định dạng, khiến việc tìm kiếm tài liệu mất nhiều thời gian.
- Không có hệ thống kiểm duyệt tài liệu chặt chẽ, dẫn đến việc chia sẻ các nội dung không phù hợp.

Những hạn chế này dẫn đến sự bất tiện và giảm hiệu quả trong việc tìm kiếm, trao đổi tài liệu giữa người dùng. Từ thực tế đó, tôi chọn đề tài “**Xây dựng website chia sẻ tài liệu học tập**” nhằm khắc phục các vấn đề trên.

Mục tiêu của đề tài là tạo ra một nền tảng trực tuyến với giao diện thân thiện, dễ sử dụng, cho phép người dùng tìm kiếm, tải xuống và chia sẻ tài liệu một cách nhanh chóng và hiệu quả. Đồng thời, đề tài này cũng hướng tới việc xây dựng một cộng đồng học tập trực tuyến, nơi người dùng có thể trao đổi và chia sẻ kiến thức, từ đó góp phần nâng cao hiệu quả học tập và nghiên cứu.

Ngoài ra, đề tài này còn giúp tôi rèn luyện và áp dụng các kiến thức đã học về lập trình web, quản lý cơ sở dữ liệu, và phát triển ứng dụng, qua đó nâng cao kỹ năng làm việc thực tế.

## 1.2 Nội dung và phạm vi đề tài

### 1.2.1 Nội dung đề tài

Đề tài tập trung vào việc thiết kế và phát triển một website chia sẻ tài liệu học tập với các chức năng cụ thể như sau:

- **Đối với người dùng (users):**
  - Xem danh sách tài liệu được chia theo các môn học để dễ dàng tìm kiếm.
  - Tải xuống các tài liệu cần thiết.
  - Đăng tài liệu lên hệ thống để chia sẻ với cộng đồng.
- **Đối với quản trị viên (admin):**
  - Duyệt và phê duyệt các tài liệu do người dùng đăng tải nhằm đảm bảo chất lượng nội dung.
  - Thực hiện các thao tác quản lý tài liệu như: thêm mới, chỉnh sửa hoặc xóa tài liệu không phù hợp.
- **Hệ thống phân loại tài liệu:**
  - Tài liệu được phân loại theo môn học như: Toán học, Vật lý, Hóa học, Ngữ văn, Lịch sử, v.v.
  - Phân loại theo định dạng tài liệu: file docx, pdf, pptx, và các file thuộc bộ Microsoft Office.
- **Cơ sở dữ liệu:**
  - Sử dụng hệ quản trị cơ sở dữ liệu MySQL để lưu trữ và quản lý thông tin về tài liệu, người dùng và các dữ liệu liên quan.

### **1.2.2 Phạm vi đề tài**

Phạm vi thực hiện đề tài bao gồm các khía cạnh sau:

- **Đối tượng sử dụng:**
  - Website hướng đến sinh viên, giảng viên và các cá nhân có nhu cầu tìm kiếm, chia sẻ tài liệu học tập.
- **Chức năng chính:**
  - Cung cấp giao diện cho phép người dùng xem danh sách tài liệu, tải xuống tài liệu, hoặc đăng tài liệu.
  - Cung cấp chức năng quản lý tài liệu dành cho quản trị viên, bao gồm: duyệt tài liệu, chỉnh sửa hoặc xóa tài liệu.
  - Tích hợp hệ thống phân loại tài liệu rõ ràng theo môn học và định dạng file.
- **Công nghệ áp dụng:**
  - Sử dụng **Node.js** làm nền tảng backend, đảm bảo xử lý các yêu cầu và giao tiếp với cơ sở dữ liệu.

- Sử dụng **React.js** với SCSS để thiết kế giao diện frontend, mang lại trải nghiệm người dùng tốt hơn.
- Sử dụng **MySQL** để lưu trữ dữ liệu một cách hiệu quả và có khả năng mở rộng.
- **Giới hạn:**
  - Website không tích hợp các tính năng phức tạp như: chat trực tuyến, đánh giá tài liệu, hoặc tính năng gợi ý thông minh.
  - Nội dung tài liệu chỉ được đăng tải sau khi được quản trị viên phê duyệt.

### 1.3 Yêu cầu phi chức năng

Bên cạnh các chức năng chính, đề tài cũng đặt ra các yêu cầu phi chức năng nhằm đảm bảo chất lượng và hiệu quả hoạt động của website, bao gồm:

- **Hiệu suất (Performance):**
  - Website phải đảm bảo tốc độ tải trang nhanh, có khả năng xử lý đồng thời nhiều yêu cầu từ người dùng mà không gây gián đoạn.
- **Bảo mật (Security):**
  - Hệ thống phải bảo vệ thông tin cá nhân của người dùng thông qua các biện pháp như mã hóa dữ liệu.
  - Chỉ có quản trị viên mới có quyền phê duyệt hoặc chỉnh sửa tài liệu.
- **Thân thiện với người dùng (Usability):**
  - Giao diện được thiết kế đơn giản, dễ sử dụng, phù hợp với nhiều đối tượng người dùng.
  - Website hỗ trợ hiển thị tốt trên các thiết bị di động và máy tính.
- **Khả năng mở rộng (Scalability):**
  - Hệ thống phải được thiết kế linh hoạt, cho phép bổ sung các chức năng mới hoặc tăng cường hiệu suất khi có nhiều người dùng.
- **Tính ổn định (Stability):**
  - Website phải hoạt động liên tục, không xảy ra lỗi nghiêm trọng làm ảnh hưởng đến trải nghiệm người dùng.

## CHƯƠNG 2: CÔNG NGHỆ SỬ DỤNG

### 2.1 ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, chủ yếu sử dụng để xây dựng giao diện người dùng. Công nghệ này phù hợp với các ứng dụng có giao diện phức tạp và cần khả năng tương tác cao. Trong dự án website chia sẻ tài liệu, ReactJS được lựa chọn làm công cụ phát triển giao diện frontend với các lý do sau:

- **Khả năng tái sử dụng component:**  
ReactJS cho phép xây dựng các component độc lập, dễ dàng tái sử dụng trong các phần khác nhau của ứng dụng. Điều này giúp giảm thời gian phát triển và bảo trì giao diện.
- **Cập nhật nhanh nhờ Virtual DOM:**  
Công nghệ Virtual DOM của React giúp tối ưu hóa hiệu suất khi giao diện cần thay đổi dữ liệu, đảm bảo tốc độ xử lý nhanh chóng và trải nghiệm người dùng mượt mà.
- **Hỗ trợ mạnh mẽ cho SPA (Single Page Application):**  
Với ReactJS, website được phát triển dưới dạng SPA, cho phép chuyển đổi giữa các trang mà không cần tải lại toàn bộ, giúp tăng tốc độ và cải thiện trải nghiệm.
- **Cộng đồng lớn và thư viện phong phú:**  
ReactJS có một cộng đồng lớn và hệ sinh thái phong phú, giúp dễ dàng tìm kiếm giải pháp cho các vấn đề phát sinh và tích hợp thêm các thư viện như React Router, Redux.

Trong dự án, ReactJS được sử dụng để xây dựng các trang chính như:

- Trang hiển thị danh sách tài liệu.
- Trang chi tiết tài liệu.
- Trang quản lý tài liệu dành cho admin.

### 2.2. Node.js

Node.js là một nền tảng chạy JavaScript phía server, được xây dựng trên V8 JavaScript Engine của Google. Node.js nổi bật nhờ khả năng xử lý không đồng bộ và hiệu suất cao, phù hợp cho các ứng dụng web thời gian thực hoặc xử lý nhiều yêu cầu đồng thời. Trong dự án này, Node.js được sử dụng làm backend với các vai trò chính:

- **Xử lý API và giao tiếp với cơ sở dữ liệu:**  
Node.js cung cấp các API RESTful để kết nối giữa frontend (ReactJS) và cơ sở dữ liệu (MySQL). Các API này bao gồm các chức năng như: lấy danh sách tài liệu, thêm mới tài liệu, chỉnh sửa hoặc xóa tài liệu.
- **Xử lý không đồng bộ:**  
Với mô hình event-driven, Node.js cho phép xử lý nhiều yêu cầu đồng thời mà không làm giảm hiệu suất, đảm bảo website hoạt động ổn định ngay cả khi có nhiều người dùng truy cập.
- **Tích hợp các thư viện mạnh mẽ:**  
Node.js có hệ sinh thái phong phú với hàng ngàn gói thư viện qua npm (Node Package Manager), hỗ trợ các công việc như xác thực người dùng, bảo mật, quản lý session và gửi email.
- **Dễ dàng mở rộng:**  
Node.js giúp xây dựng các ứng dụng có khả năng mở rộng cao, sẵn sàng tích hợp thêm các tính năng mới trong tương lai.

Trong dự án, Node.js được sử dụng để xây dựng các chức năng backend như:

- Xử lý đăng nhập, đăng ký của người dùng.
- Lưu trữ và quản lý dữ liệu tài liệu.
- Quản lý quyền hạn giữa người dùng thông thường và quản trị viên.

## 2.3 Cơ sở dữ liệu Mysql

**MySQL** là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được phát triển và duy trì bởi Oracle Corporation. Nó sử dụng ngôn ngữ truy vấn SQL (Structured Query Language) để quản lý và thao tác dữ liệu. MySQL được sử dụng rộng rãi trong các ứng dụng web và các dự án có yêu cầu hiệu suất cao vì tính linh hoạt và khả năng mở rộng của nó.



### **Ưu điểm của MySQL:**

- Mã nguồn mở:** MySQL là phần mềm miễn phí và mã nguồn mở, cho phép người dùng tự do sử dụng, thay đổi và phân phối lại.
- Hiệu suất cao:** MySQL có khả năng xử lý lượng lớn dữ liệu với hiệu suất tốt, nhờ vào khả năng tối ưu hóa các truy vấn và hỗ trợ các chỉ mục (index).
- Dễ sử dụng:** MySQL có cú pháp đơn giản và dễ học, là sự lựa chọn phổ biến cho các lập trình viên mới bắt đầu.
- Tính linh hoạt:** Hỗ trợ nhiều hệ điều hành như Windows, Linux, macOS, giúp dễ dàng triển khai trên các nền tảng khác nhau.
- Cộng đồng lớn:** Là một công cụ phổ biến, MySQL có một cộng đồng hỗ trợ rộng lớn và tài liệu phong phú.

### **Nhược điểm của MySQL:**

- Hạn chế tính năng:** Mặc dù MySQL phát triển nhanh chóng, nhưng so với các hệ quản trị cơ sở dữ liệu như PostgreSQL hoặc Oracle, MySQL vẫn thiếu một số tính năng nâng cao như hỗ trợ các loại dữ liệu phức tạp và tính toán vạn dữ liệu cao cấp.

2. **Khả năng mở rộng hạn chế:** Dù MySQL có khả năng mở rộng tốt trong nhiều trường hợp, nhưng khi làm việc với các ứng dụng rất lớn và phức tạp, nó có thể gặp khó khăn trong việc xử lý tải nặng hoặc yêu cầu cao về khả năng phân tán dữ liệu.
3. **Quản lý sao lưu và phục hồi:** Một số người dùng cho rằng khả năng sao lưu và phục hồi của MySQL không linh hoạt và dễ sử dụng bằng một số hệ quản trị cơ sở dữ liệu khác.

Nhìn chung, MySQL là một lựa chọn tuyệt vời cho nhiều ứng dụng, đặc biệt là các ứng dụng web, nhờ vào tính linh hoạt, hiệu suất và cộng đồng hỗ trợ mạnh mẽ. Tuy nhiên, trong các tình huống yêu cầu tính năng phức tạp hoặc khả năng mở rộng vượt trội, người dùng có thể cần xem xét các giải pháp khác.

## 2.4 SCSS

SCSS (Sassy CSS) là một phần mở rộng của CSS, cung cấp cú pháp hiện đại và các tính năng bổ sung để cải thiện hiệu quả viết mã CSS. SCSS được lựa chọn trong dự án vì các lý do sau:

- **Cú pháp gọn gàng, dễ bảo trì:**  
SCSS hỗ trợ sử dụng biến, vòng lặp và hàm, giúp mã nguồn CSS dễ đọc, dễ bảo trì và tái sử dụng.
- **Khả năng phân chia file và modular hóa:**  
Với SCSS, các tệp CSS có thể được chia thành các module nhỏ hơn, dễ dàng quản lý và tránh xung đột trong mã nguồn.
- **Nested rules:**  
SCSS cho phép viết các quy tắc lồng nhau, giúp mã nguồn rõ ràng hơn, đặc biệt khi xử lý các thành phần giao diện phức tạp.
- **Hỗ trợ mixin và extend:**  
Các mixin và extend giúp tái sử dụng các đoạn mã CSS, giảm thiểu sự trùng lặp và đảm bảo tính nhất quán trong thiết kế giao diện.

Trong dự án, SCSS được sử dụng để:

- Thiết kế giao diện trang chủ, trang hiển thị tài liệu, và các trang quản trị với bố cục hiện đại, thân thiện với người dùng.

- Tối ưu hóa thiết kế responsive, đảm bảo website hiển thị tốt trên mọi thiết bị, bao gồm cả máy tính và điện thoại di động.
- Sử dụng biến SCSS để quản lý các màu sắc, kích thước font và khoảng cách, tạo sự thống nhất trong giao diện.

## 2.5 Rest API.

### 2.5.1 Khái niệm Rest API

- Rest API (Representational State Transfer Application Programming Interface) là một giao diện lập trình ứng dụng dựa trên kiến trúc REST, cho phép các hệ thống giao tiếp với nhau thông qua giao thức HTTP. Rest API tập trung vào việc định nghĩa và thao tác với tài nguyên (resource) bằng cách sử dụng các phương thức HTTP chuẩn như GET, POST, PUT, PATCH, DELETE

- Rest API tuân theo nguyên tắc cốt lõi như client – server, stateless (không lưu trạng thái giữa các request), giúp đảm bảo tính linh hoạt, dễ bảo trì và mở rộng. Rest API thường dùng định dạng JSON hoặc XML để trao đổi dữ liệu giữa client và server.

### 2.5.2 Chuẩn Restful API

- Restful API là một dạng chuẩn của kiến trúc Rest API kèm theo các yêu cầu :

+ Tên endpoint mô tả đúng tài nguyên ví dụ:

- /users thay vì /getuser
- /orders thay vì /getorder

+ Sử dụng đúng phương thức HTTP

- GET để lấy dữ liệu
- POST để gửi dữ liệu đi
- PUT/PATCH để cập nhật dữ liệu
- DELETE để xóa dữ liệu

## 2.6 Postman

**Postman** là một công cụ mạnh mẽ và phổ biến dùng để **thiết kế, kiểm thử và quản lý API** trong quá trình phát triển phần mềm. Với giao diện thân thiện, Postman giúp các nhà phát triển và kiểm thử viên dễ dàng làm việc với API, kiểm tra dữ liệu, đảm bảo API hoạt động chính xác và hiệu quả.

- Postman ban đầu là một tiện ích mở rộng trên trình duyệt **Chrome**. Hiện tại, nó đã phát triển thành một ứng dụng độc lập hỗ trợ **Windows, macOS, và Linux**.
- Postman giúp người dùng gửi các yêu cầu HTTP/HTTPS đến **API endpoints** và nhận phản hồi một cách trực quan.
- Công cụ này được sử dụng rộng rãi trong quá trình phát triển và kiểm thử các ứng dụng dựa trên **RESTful API** và **GraphQL API**.

## CHƯƠNG 3: PHÂN TÍCH

### 3.1 Mô tả hệ thống

Hệ thống website chia sẻ tài liệu được thiết kế nhằm hỗ trợ người dùng tìm kiếm, tải xuống và đăng tài liệu một cách dễ dàng. Đồng thời, quản trị viên có quyền quản lý nội dung để đảm bảo chất lượng và phù hợp với mục tiêu sử dụng. Hệ thống gồm hai loại người dùng chính:

#### 1. Người dùng thông thường (Users):

- **Đăng ký và đăng nhập:** Người dùng phải đăng ký tài khoản để sử dụng các chức năng của hệ thống. Sau khi đăng nhập, người dùng có thể thực hiện các tác vụ như tải tài liệu, đăng tài liệu, hoặc tìm kiếm tài liệu theo môn học hoặc định dạng file.
- **Xem danh sách tài liệu:** Tài liệu được phân loại theo môn học và định dạng, giúp người dùng dễ dàng tìm kiếm và lựa chọn.
- **Tải xuống tài liệu:** Người dùng có thể tải xuống các tài liệu được hệ thống cung cấp.
- **Đăng tài liệu:** Người dùng có thể tải lên tài liệu cá nhân để chia sẻ với cộng đồng. Tài liệu sẽ chờ quản trị viên phê duyệt trước khi được hiển thị.

#### 2. Quản trị viên (Admins):

- **Quản lý tài liệu:** Quản trị viên có thể duyệt tài liệu, thêm mới, chỉnh sửa hoặc xóa tài liệu không phù hợp.
- **Kiểm duyệt tài liệu:** Mọi tài liệu do người dùng đăng tải cần được quản trị viên phê duyệt trước khi xuất hiện trên hệ thống.
- **Quản lý người dùng:** Có thể khóa tài khoản người dùng nếu phát hiện hành vi vi phạm.

#### 3. Cơ sở dữ liệu:

- Hệ thống sử dụng cơ sở dữ liệu MySQL để lưu trữ thông tin tài liệu, người dùng và các dữ liệu liên quan.

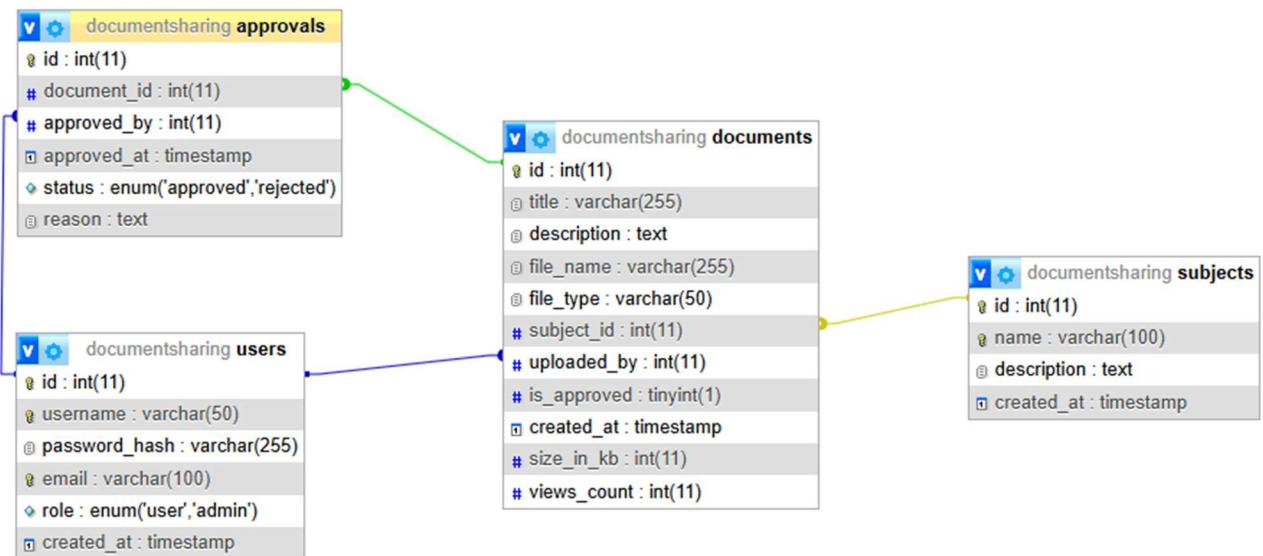
- Tài liệu được quản lý theo các thuộc tính như tên tài liệu, loại tài liệu, định dạng file, môn học, và ngày đăng.

#### 4. **Chức năng chính:**

- Phân loại tài liệu theo môn học và định dạng file.
- Hỗ trợ tìm kiếm nhanh tài liệu theo từ khóa.
- Quản lý trạng thái tài liệu: đã được duyệt hay chờ phê duyệt.

Hệ thống được thiết kế nhằm đáp ứng các yêu cầu phi chức năng như bảo mật thông tin, tốc độ xử lý nhanh, và khả năng mở rộng trong tương lai.

### 3.2 Biểu đồ ERD



Biểu đồ ERD của bạn gồm 4 bảng chính:

- **users** (người dùng)
- **documents** (tài liệu)
- **subjects** (môn học)

- **approvals** (phê duyệt)

## 1. Bảng users (Người dùng)

Bảng này lưu trữ thông tin của tất cả người dùng trên hệ thống, bao gồm cả người dùng thường và quản trị viên.

- **Thuộc tính:**

- id: Khóa chính, định danh duy nhất của mỗi người dùng.
- username: Tên đăng nhập của người dùng.
- password\_hash: Mật khẩu đã được mã hóa để bảo mật.
- email: Địa chỉ email của người dùng.
- role: Quyền của người dùng (user hoặc admin).
- created\_at: Thời điểm tài khoản được tạo.

- **Mối quan hệ:**

- Liên kết với bảng documents thông qua uploaded\_by để xác định người đã tải lên tài liệu.
- Liên kết với bảng approvals qua approved\_by để ghi nhận quản trị viên phê duyệt tài liệu.

## 2. Bảng documents (Tài liệu)

Bảng này lưu trữ tất cả các tài liệu được tải lên hệ thống.

- **Thuộc tính:**

- id: Khóa chính, định danh duy nhất của mỗi tài liệu.
- title: Tên tài liệu.
- description: Mô tả nội dung tài liệu.
- file\_name: Tên file của tài liệu được tải lên.

- file\_type: Loại file (PDF, DOCX, PPTX,...).
  - subject\_id: Khóa ngoại liên kết với bảng subjects, xác định môn học mà tài liệu thuộc về.
  - uploaded\_by: Khóa ngoại liên kết với bảng users, xác định người tải lên tài liệu.
  - is\_approved: Trạng thái tài liệu (đã được duyệt hay chưa).
  - created\_at: Thời điểm tài liệu được tải lên.
  - size\_in\_kb: Kích thước file (tính theo KB).
  - views\_count: Số lượt xem tài liệu.
- **Mối quan hệ:**

- Liên kết với bảng users qua uploaded\_by để xác định ai đã tải tài liệu lên.
- Liên kết với bảng subjects qua subject\_id để xác định môn học.
- Liên kết với bảng approvals qua document\_id để xác định trạng thái phê duyệt.

### 3. Bảng subjects (Môn học)

Bảng này chứa thông tin về các môn học mà tài liệu thuộc về.

- **Thuộc tính:**
  - id: Khóa chính, định danh duy nhất của mỗi môn học.
  - name: Tên môn học.
  - description: Mô tả chi tiết về môn học.
  - created\_at: Thời điểm môn học được thêm vào hệ thống.
- **Mối quan hệ:**

- Liên kết với bảng documents qua subject\_id để xác định tài liệu thuộc môn học nào.

#### **4. Bảng approvals (Phê duyệt)**

Bảng này quản lý trạng thái phê duyệt của tài liệu, do quản trị viên thực hiện.

- **Thuộc tính:**

- id: Khóa chính, định danh duy nhất của mỗi bản ghi phê duyệt.
- document\_id: Khóa ngoại liên kết với bảng documents, xác định tài liệu được phê duyệt.
- approved\_by: Khóa ngoại liên kết với bảng users, xác định quản trị viên phê duyệt.
- approved\_at: Thời điểm tài liệu được phê duyệt.
- status: Trạng thái phê duyệt (approved hoặc rejected).
- reason: Lý do từ chối (trường hợp bị từ chối).

- **Mối quan hệ:**

- Liên kết với bảng documents qua document\_id để xác định tài liệu nào được phê duyệt.
- Liên kết với bảng users qua approved\_by để ghi nhận quản trị viên nào đã thực hiện phê duyệt.

**Tổng kết mối quan hệ giữa các bảng:**

1. **users và documents:**

- Một người dùng (users) có thể tải lên nhiều tài liệu (documents), mỗi tài liệu sẽ có một trường uploaded\_by để xác định người dùng tải lên.
- Mỗi tài liệu chỉ thuộc về một người dùng tải lên, vì vậy mối quan hệ này là **1-n** (một người dùng có thể tải lên nhiều tài liệu, nhưng mỗi tài liệu chỉ thuộc về một người dùng).

2. **documents và subjects:**

- Mỗi tài liệu chỉ thuộc về một môn học, mỗi quan hệ này được xác định thông qua trường `subject_id` trong bảng `documents`.
- Một môn học có thể có nhiều tài liệu liên quan, vì vậy mỗi quan hệ này là **1-n** (một môn học có thể có nhiều tài liệu, nhưng mỗi tài liệu chỉ thuộc về một môn học).

### **3. approvals và documents:**

- Mỗi bản ghi phê duyệt (`approvals`) sẽ liên kết với một tài liệu duy nhất thông qua trường `document_id` trong bảng `approvals`.
- Mỗi tài liệu chỉ có một trạng thái phê duyệt duy nhất tại một thời điểm. Tuy nhiên, tài liệu có thể trải qua nhiều lần phê duyệt khác nhau theo thời gian. Do đó, mỗi quan hệ này là **1-n** (mỗi tài liệu có thể có nhiều lần phê duyệt nếu trạng thái thay đổi, nhưng mỗi bản ghi phê duyệt chỉ liên kết với một tài liệu duy nhất tại một thời điểm).

### **4. approvals và users:**

- Một quản trị viên (`users` có `role = admin`) có thể thực hiện nhiều lần phê duyệt (`approvals`), vì vậy mỗi quan hệ này là **1-n** (một quản trị viên có thể thực hiện nhiều lần phê duyệt).
- Mỗi lần phê duyệt (`approvals`) chỉ do một quản trị viên thực hiện, xác định qua trường `approved_by` trong bảng `approvals`.

## CHƯƠNG 4: XÂY DỰNG DỰ ÁN

### 4.1 Xây dựng Cấu trúc Backend

#### 4.1.1. Mô Hình Các Lớp

##### 1. Lớp Approval

```
class Approval {
    static create({ document_id, approved_by, status, reason }) {
        const query = 'INSERT INTO approvals (document_id, approved_by, status, reason) VALUES (?, ?, ?, ?)';
        return db.query(query, [document_id, approved_by, status, reason]);
    }
}
```

Lớp Approval dùng để quản lý các bản ghi phê duyệt tài liệu. Các phương thức trong lớp này chủ yếu là thao tác tạo mới một bản ghi phê duyệt trong cơ sở dữ liệu.

- **Phương thức create:**
  - **Mục đích:** Tạo một bản ghi phê duyệt tài liệu trong bảng approvals.
  - **Cách hoạt động:** Phương thức nhận một đối tượng chứa thông tin tài liệu, người phê duyệt, trạng thái và lý do phê duyệt, sau đó thực hiện truy vấn SQL để chèn dữ liệu vào bảng approvals.

## 2. Lớp Document

```
class Document {
    static create({ title, description, file_name, file_type, subject_id, uploaded_by }) {
        const query = 'INSERT INTO documents (title, description, file_name, file_type, subject_id, uploaded_by) VALUES (?, ?, ?, ?, ?, ?)';
        return db.query(query, [title, description, file_name, file_type, subject_id, uploaded_by]);
    }

    static getAll() {
        const query = `
            SELECT d.id, d.title, d.description, d.file_name, d.file_type,
            d.subject_id, s.name as subject_name, d.is_approved
            FROM documents d
            JOIN subjects s ON d.subject_id = s.id
            WHERE d.is_approved = 1
        `;
        return db.query(query).then(([rows]) => rows); // Trả về kết quả
    }

    static getBySubject(subject_id) {
        const query = 'SELECT * FROM documents WHERE subject_id = ? AND is_approved = 1';
        return db.query(query, [subject_id]).then(([rows]) => rows); // Trả về chỉ dữ liệu (rows)
    }

    static updateApprovalStatus(document_id, status) {
        const isApproved = status === 'approved' ? 1 : 0;
        const query = 'UPDATE documents SET is_approved = ? WHERE id = ?';
        return db.query(query, [isApproved, document_id]);
    }

    static async getUnapproved() {
        const query = `
            SELECT d.id, d.title, d.file_name, d.subject_id, s.name as subject_name, d.is_approved, d.file_type
            FROM documents d
            JOIN subjects s ON d.subject_id = s.id
            WHERE d.is_approved = 0
        `;
        try {
            const [rows] = await db.execute(query);
            return rows;
        } catch (err) {
            throw err;
        }
    }
}
```

Lớp Document được sử dụng để quản lý các tài liệu trong hệ thống. Nó bao gồm các phương thức giúp thêm tài liệu mới, lấy tài liệu, và cập nhật trạng thái phê duyệt của tài liệu.

- **Phương thức create:**
  - **Mục đích:** Tạo một bản ghi tài liệu mới trong bảng documents.
  - **Cách hoạt động:** Phương thức này nhận các thông tin như tiêu đề, mô tả, tên tệp, loại tệp, ID môn học và người tải lên tài liệu, sau đó thực hiện truy vấn SQL để thêm dữ liệu vào bảng documents.
- **Phương thức getAll:**

- **Mục đích:** Lấy tất cả tài liệu đã được phê duyệt (có trường `is_approved` bằng 1).
- **Cách hoạt động:** Truy vấn SQL sẽ lấy danh sách tài liệu kết hợp với thông tin môn học.
- **Phương thức `getBySubject`:**
  - **Mục đích:** Lấy tài liệu theo môn học.
  - **Cách hoạt động:** Truy vấn SQL lấy danh sách tài liệu của một môn học cụ thể, chỉ bao gồm những tài liệu đã được phê duyệt.
- **Phương thức `updateApprovalStatus`:**
  - **Mục đích:** Cập nhật trạng thái phê duyệt của tài liệu.
  - **Cách hoạt động:** Truy vấn SQL để thay đổi giá trị của trường `is_approved` trong bảng `documents` dựa trên ID tài liệu và trạng thái phê duyệt.
- **Phương thức `getUnapproved`:**
  - **Mục đích:** Lấy danh sách các tài liệu chưa được phê duyệt.
  - **Cách hoạt động:** Truy vấn SQL sẽ trả về danh sách các tài liệu chưa được phê duyệt (`is_approved = 0`).

#### 4. Lớp Subject

```
class Subject {
    static async create({ name, description }) {
        const query = 'INSERT INTO subjects (name, description) VALUES (?, ?)';
        try {
            const [result] = await db.execute(query, [name, description]);
            return result;
        } catch (err) {
            throw err;
        }
    }

    static async getAll() {
        const query = 'SELECT * FROM subjects';
        try {
            const [rows] = await db.execute(query);
            return rows;
        } catch (err) {
            throw err;
        }
    }
}
```

Lớp Subject dùng để quản lý các môn học trong hệ thống. Các phương thức trong lớp này giúp tạo mới môn học và lấy danh sách môn học.

- **Phương thức create:**
  - **Mục đích:** Tạo một môn học mới trong bảng subjects.
  - **Cách hoạt động:** Phương thức nhận tên và mô tả môn học, thực hiện truy vấn SQL để thêm dữ liệu vào bảng subjects.
- **Phương thức getAll:**
  - **Mục đích:** Lấy tất cả các môn học trong hệ thống.
  - **Cách hoạt động:** Truy vấn SQL lấy toàn bộ danh sách môn học.

## 5. Lớp User

```
class User {
    static async create(user) {
        const [result] = await db.query(
            'INSERT INTO users (username, password_hash, email, role) VALUES (?, ?, ?, ?)',
            [user.username, user.password_hash, user.email, user.role]
        );
        return result;
    }

    static async findByUsername(username) {
        const [rows] = await db.query('SELECT * FROM users WHERE username = ?', [username]);
        return rows[0];
    }
    static async findById(id) {
        const query = 'SELECT * FROM users WHERE id = ?';
        const [rows] = await db.query(query, [id]);
        return rows[0];
    }

    static async findByUsername(username) {
        const query = 'SELECT * FROM users WHERE username = ?';
        const [rows] = await db.query(query, [username]);
        return rows[0];
    }
}
```

Lớp User dùng để quản lý người dùng trong hệ thống. Các phương thức trong lớp này giúp tạo mới người dùng và tìm kiếm người dùng dựa trên tên đăng nhập hoặc ID.

- **Phương thức create:**
  - **Mục đích:** Tạo người dùng mới trong bảng users.
  - **Cách hoạt động:** Phương thức này nhận thông tin người dùng (tên đăng nhập, mật khẩu đã mã hóa, email, vai trò) và thực hiện truy vấn SQL để thêm dữ liệu vào bảng users.
- **Phương thức findByUsername:**
  - **Mục đích:** Tìm người dùng theo tên đăng nhập.
  - **Cách hoạt động:** Truy vấn SQL để tìm người dùng dựa trên tên đăng nhập.
- **Phương thức findById:**
  - **Mục đích:** Tìm người dùng theo ID.
  - **Cách hoạt động:** Truy vấn SQL để lấy người dùng dựa trên ID.

#### 4.1.2. Các Middleware trong Dự án

##### 1. Middleware Xác Thực Người Dùng (authMiddleware)

```
const jwt = require('jsonwebtoken');  53.3k (gzipped: 15.8k)

const authMiddleware = (req, res, next) => {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) {
    return res.status(401).json({ error: 'Token không hợp lệ hoặc không được cung cấp.' });
  }

  jwt.verify(token, process.env.JWT_SECRET || 'default_secret_key', (err, decoded) => {
    if (err) {
      return res.status(401).json({ error: 'Token không hợp lệ hoặc đã hết hạn.' });
    }
    req.user = decoded;
    next();
  });
};

Middleware này sử dụng JSON Web Token (JWT) để xác thực người dùng. Nó kiểm tra xem yêu cầu có chứa token hợp lệ trong header Authorization hay không và xác minh token đó.
```

- **Mục đích:** Đảm bảo rằng chỉ người dùng đã xác thực mới có thể truy cập các tài nguyên bảo mật của hệ thống.
- **Cách hoạt động:**
  - Token được lấy từ header Authorization theo định dạng Bearer <token>.
  - Nếu token không hợp lệ hoặc không tồn tại, hệ thống sẽ trả về mã lỗi 401 Unauthorized với thông báo lỗi thích hợp.
  - Nếu token hợp lệ, thông tin người dùng sẽ được giải mã và lưu vào req.user để sử dụng trong các middleware hoặc route tiếp theo.

## 2. Middleware Phân Quyền Truy Cập (roleMiddleware)

```
const roleMiddleware = (roles) => {
  return (req, res, next) => {
    if (!roles.includes(req.user.role)) {
      return res.status(403).json({ message: 'Forbidden' });
    }
    next();
  };
};
```

Middleware này được sử dụng để kiểm tra quyền hạn của người dùng dựa trên vai trò của họ. Chỉ những người dùng có vai trò hợp lệ mới có thể truy cập các tài nguyên yêu cầu quyền truy cập cao hơn.

- **Mục đích:** Đảm bảo rằng các tài nguyên chỉ được truy cập bởi những người có quyền hợp lệ.
- **Cách hoạt động:**
  - Middleware này nhận một mảng các vai trò (ví dụ: ['admin', 'moderator']).
  - Nếu vai trò của người dùng (được lấy từ req.user.role) không có trong mảng vai trò cho phép, người dùng sẽ nhận được phản hồi lỗi với mã trạng thái 403 Forbidden.
  - Nếu vai trò hợp lệ, middleware gọi next() để tiếp tục xử lý yêu cầu.

### 3. Middleware Xử Lý Tải Lên Tệp (upload)

```
const fs = require('fs');
const path = require('path');
const multer = require('multer');    241.4k (gzipped: 47.5k)
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    const uploadDir = path.resolve(__dirname, '../uploads');
    if (!fs.existsSync(uploadDir)) {
      fs.mkdirSync(uploadDir);
    }
    cb(null, uploadDir);
  },
  filename: (req, file, cb) => {
    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
    cb(null, `${file.fieldname}-${uniqueSuffix}${path.extname(file.originalname)})`);
  }
});

const fileFilter = (req, file, cb) => {
  const allowedTypes = [
    'application/pdf',
    'application/msword',
    'application/vnd.openxmlformats-officedocument.wordprocessingml.document',
    'application/vnd.ms-powerpoint',
    'application/vnd.openxmlformats-officedocument.presentationml.presentation',
    'application/x-zip-compressed',
    'application/vnd.rar'
  ];
  if (allowedTypes.includes(file.mimetype)) {
    cb(null, true);
  } else {
    cb(new Error('Invalid file type. Allowed: PDF, DOC, DOCX, PPT, PPTX, ZIP, RAR.'));
  }
};

const upload = multer({
  storage,
  fileFilter,
  limits: { fileSize: 50 * 1024 * 1024 }
});
```

Middleware này sử dụng thư viện multer để xử lý các tệp tải lên từ phía người dùng. Nó cấu hình cách lưu trữ tệp và kiểm tra loại tệp và kích thước tệp.

- **Mục đích:** Xử lý các yêu cầu tải lên tệp một cách hiệu quả và an toàn.
- **Cách hoạt động:**
  - **Lưu trữ tệp:** Các tệp được lưu vào thư mục uploads. Nếu thư mục này không tồn tại, nó sẽ được tạo tự động.
  - **Kiểm tra loại tệp:** Middleware kiểm tra loại MIME của tệp và chỉ chấp nhận các tệp với các loại như PDF, DOC, DOCX, PPT, PPTX, ZIP, và RAR.
  - **Giới hạn kích thước tệp:** Kích thước tối đa của tệp được giới hạn là 50 MB.

### 4.1.3. Các Controller Backend

#### 1. Admin Document Controller (`adminDocumentController.js`)

Controller này chịu trách nhiệm quản lý các tài liệu từ phía admin, bao gồm chức năng lấy danh sách tài liệu.

```
const Document = require('../models/Document');

exports.getAllDocuments = async (req, res) => {
  try {
    const documents = await Document.getAll();
    res.status(200).json(documents);
  } catch (error) {
    res.status(500).json({ error: 'Lỗi khi lấy danh sách tài liệu', details: error.message });
  }
};
```

- **getAllDocuments:**
  - **Chức năng:** Lấy danh sách tất cả các tài liệu.
  - **Phân tích:** Hàm này gọi phương thức `getAll()` từ mô hình `Document`, sau đó trả về danh sách tài liệu dưới dạng JSON với mã HTTP 200 nếu thành công. Nếu gặp lỗi, mã lỗi 500 sẽ được trả về cùng với thông báo lỗi chi tiết..

## 2. Authentication Controller (authController.js)

Controller này xử lý chức năng đăng ký, đăng nhập và lấy thông tin người dùng.

```
const bcrypt = require('bcryptjs'); 21.6k (gzipped: 9.8k)
const jwt = require('jsonwebtoken'); 53.3k (gzipped: 15.8k)
const User = require('../models/User');
require('dotenv').config(); 6.3k (gzipped: 2.7k)

exports.register = async (req, res) => {
  const { username, password, email, role } = req.body;

  if (!username || !password || !email) {
    return res.status(400).json({ error: 'Vui lòng cung cấp đầy đủ thông tin.' });
  }

  try {
    const existingUser = await User.findByUsername(username);
    if (existingUser) {
      return res.status(400).json({ error: 'Tên người dùng đã tồn tại.' });
    }

    const hashedPassword = await bcrypt.hash(password, 10);

    await User.create({
      username,
      password_hash: hashedPassword,
      email,
      role: role || 'user'
    });

    res.status(201).json({ message: 'Đăng ký người dùng thành công.' });
  } catch (err) {
    console.error('Lỗi đăng ký người dùng:', err);
    res.status(500).json({ error: 'Không thể đăng ký người dùng.', details: err.message });
  }
};
```

```

exports.login = async (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) {
    return res.status(400).json({ error: 'Vui lòng cung cấp tên đăng nhập và mật khẩu.' });
  }

  try {
    const user = await User.findByUsername(username);

    if (!user || !(await bcrypt.compare(password, user.password_hash))) {
      return res.status(401).json({ error: 'Tên đăng nhập hoặc mật khẩu không hợp lệ.' });
    }

    const token = jwt.sign(
      { id: user.id, role: user.role },
      process.env.JWT_SECRET || 'default_secret_key',
      { expiresIn: '1h' }
    );

    res.json({
      message: 'Đăng nhập thành công.',
      token
    });
  } catch (err) {
    console.error('Lỗi đăng nhập:', err);
    res.status(500).json({ error: 'Không thể đăng nhập.' });
  }
};

exports.getProfile = async (req, res) => {
  console.log('User from Token:', req.user);
  try {
    const user = await User.findById(req.user.id);
    if (!user) {
      return res.status(404).json({ error: 'Không tìm thấy người dùng' });
    }
    res.status(200).json({ user });
  } catch (error) {
    res.status(500).json({ error: 'Lỗi lấy thông tin người dùng', details: error.message });
  }
};

```

- **register:**
  - **Chức năng:** Đăng ký người dùng mới.
  - **Phân tích:** Hàm này kiểm tra tính hợp lệ của các trường thông tin (username, password, email) và kiểm tra xem tên người dùng có bị trùng hay không. Nếu không có lỗi, mật khẩu sẽ được mã hóa và người dùng mới sẽ được tạo ra trong cơ sở dữ liệu. Sau khi đăng ký thành công, một thông báo sẽ được trả về.
- **login:**

- **Chức năng:** Đăng nhập và cấp token.
- **Phân tích:** Hàm này nhận thông tin đăng nhập, kiểm tra tính hợp lệ của tên người dùng và mật khẩu. Nếu đúng, một JWT (JSON Web Token) sẽ được tạo và trả về để sử dụng trong các yêu cầu tiếp theo. Mã lỗi 401 sẽ được trả về nếu thông tin đăng nhập không hợp lệ.
- **getProfile:**
  - **Chức năng:** Lấy thông tin người dùng từ token.
  - **Phân tích:** Hàm này lấy thông tin người dùng từ token đã được xác thực. Nếu người dùng không tồn tại, mã lỗi 404 sẽ được trả về, ngược lại, thông tin người dùng sẽ được trả về dưới dạng JSON.

### 3. Dashboard Controller (dashboardController.js)

Controller này cung cấp các thống kê tổng quan về tài liệu, người dùng và môn học trong hệ thống.

```
const db = require('../config/db');
exports.getStats = async (req, res) => {
  try {
    const totalDocumentsResult = await db.query('SELECT COUNT(*) AS totalDocuments FROM documents');
    console.log('Total Documents:', totalDocumentsResult);
    const totalDocuments = totalDocumentsResult[0][0].totalDocuments;

    const unapprovedDocumentsResult = await db.query('SELECT COUNT(*) AS unapprovedDocuments FROM documents WHERE is_approved = 0');
    console.log('Unapproved Documents:', unapprovedDocumentsResult);
    const unapprovedDocuments = unapprovedDocumentsResult[0][0].unapprovedDocuments;

    const approvedDocumentsResult = await db.query('SELECT COUNT(*) AS approvedDocuments FROM documents WHERE is_approved = 1');
    console.log('Approved Documents:', approvedDocumentsResult);
    const approvedDocuments = approvedDocumentsResult[0][0].approvedDocuments;

    const totalUsersResult = await db.query('SELECT COUNT(*) AS totalUsers FROM users');
    console.log('Total Users:', totalUsersResult);
    const totalUsers = totalUsersResult[0][0].totalUsers;

    const totalSubjectsResult = await db.query('SELECT COUNT(*) AS totalSubjects FROM subjects');
    console.log('Total Subjects:', totalSubjectsResult);
    const totalSubjects = totalSubjectsResult[0][0].totalSubjects;

    res.status(200).json({
      totalDocuments,
      totalUsers,
      totalSubjects,
      approvedDocuments,
      unapprovedDocuments,
    });
  } catch (error) {
    console.error('Error fetching stats:', error);
    res.status(500).json({ error: 'Lỗi khi lấy thống kê', details: error.message });
  }
}
```

- **getStats:**
  - **Chức năng:** Lấy các thống kê về tài liệu, người dùng, môn học.

- **Phân tích:** Hàm này thực hiện các truy vấn SQL để lấy số lượng tài liệu (tổng số tài liệu, tài liệu đã phê duyệt, tài liệu chưa phê duyệt), số lượng người dùng và số lượng môn học. Các thông kê này sẽ được trả về dưới dạng JSON. Nếu có lỗi xảy ra trong quá trình truy vấn, mã lỗi 500 và thông báo lỗi sẽ được trả về.

#### 4. Document Controller (`documentController.js`)

Controller này quản lý các chức năng liên quan đến tài liệu, bao gồm tải lên, xem danh sách tài liệu, tải tài liệu và phân loại tài liệu theo môn học.

```
const documentController = {
  uploadDocument: async (req, res) => {
    const { title, description, subject_id } = req.body;
    const uploaded_by = req.user.id;
    const file = req.file;

    if (!title || !subject_id) {
      return res
        .status(400)
        .json({ message: "Title and Subject ID are required" });
    }
    if (!file) {
      return res.status(400).json({ message: "No file uploaded" });
    }

    try {
      const file_name = file.filename;
      const file_type = path.extname(file.originalname);
      const [result] = await Document.create({
        title,
        description: description || "",
        file_name,
        file_type,
        subject_id,
        uploaded_by,
      });

      res
        .status(200)
        .json({ message: "Document uploaded successfully", result });
    } catch (error) {
      console.error("Error uploading document:", error);
      res.status(500).json({ message: "Error uploading document", error });
    }
  },
  getAllDocuments: async (req, res) => {
    try {
      const documents = await Document.getAll();
      res.status(200).json(documents);
    } catch (error) {
      console.error("Error fetching documents:", error);
      res.status(500).json({ message: "Error fetching documents", error });
    }
  },
};
```

```

getDocumentsBySubject: async (req, res) => {
  const { subject_id } = req.params;
  try {
    const documents = await Document.getBySubject(subject_id);
    res.status(200).json(documents);
  } catch (error) {
    console.error("Error fetching documents by subject:", error);
    res
      .status(500)
      .json({ message: "Error fetching documents by subject", error });
  }
},
downloadDocument: async (req, res) => {
  const { id } = req.params;

  try {
    const [document] = await db.query("SELECT * FROM documents WHERE id = ?", [
      id,
    ]);
    if (!document) {
      return res.status(404).json({ message: "Document not found" });
    }

    const filePath = path.resolve(__dirname, "../uploads", document.file_name);
    if (fs.existsSync(filePath)) {
      return res.download(filePath);
    } else {
      return res.status(404).json({ message: "File not found on server" });
    }
  } catch (error) {
    console.error("Error downloading document:", error);
    res.status(500).json({ message: "Error downloading document", error });
  }
};

```

- **uploadDocument:**

- **Chức năng:** Tải lên tài liệu mới.
- **Phân tích:** Hàm này nhận thông tin tài liệu từ form (tiêu đề, mô tả, id môn học) và tệp tài liệu. Sau khi kiểm tra tính hợp lệ của dữ liệu, tài liệu sẽ được lưu vào cơ sở dữ liệu, bao gồm tên tệp, loại tệp và id môn

học. Tệp tài liệu được lưu trữ trong thư mục uploads, và sau khi tải lên thành công, thông báo và thông tin tài liệu sẽ được trả về.

- **getAllDocuments:**
  - **Chức năng:** Lấy tất cả tài liệu.
  - **Phân tích:** Hàm này gọi phương thức `getAll()` từ mô hình `Document` và trả về danh sách tất cả các tài liệu. Nếu có lỗi xảy ra, mã lỗi 500 và thông báo lỗi sẽ được trả về.
- **getDocumentsBySubject:**
  - **Chức năng:** Lấy tài liệu theo môn học.
  - **Phân tích:** Hàm này nhận `subject_id` từ tham số URL và gọi phương thức `getBySubject()` từ mô hình `Document` để lấy tất cả tài liệu thuộc môn học đó. Dữ liệu sẽ được trả về dưới dạng JSON.
- **downloadDocument:**
  - **Chức năng:** Tải về tài liệu.
  - **Phân tích:** Hàm này nhận `id` tài liệu từ tham số URL và kiểm tra xem tài liệu có tồn tại trong cơ sở dữ liệu không. Nếu tài liệu tồn tại, nó sẽ gửi tệp tài liệu về client để tải xuống. Nếu không, mã lỗi 404 sẽ được trả về.

## 5. Subject Controller (`subjectController.js`)

Controller này cung cấp chức năng lấy danh sách các môn học.

```
const subjectController = {
  getAllSubjects: async (req, res) => {
    try {
      const subjects = await Subject.getAll();
      res.status(200).json(subjects);
    } catch (error) {
      console.error(error);
      res.status(500).json({ message: 'Error fetching subjects', error: error.message });
    }
  };
};
```

- **getAllSubjects:**
  - **Chức năng:** Lấy danh sách môn học.
  - **Phân tích:** Hàm này gọi phương thức `getAll()` từ mô hình `Subject` và trả về tất cả các môn học trong hệ thống dưới dạng JSON. Nếu có lỗi xảy ra, mã lỗi 500 và thông báo lỗi sẽ được trả về.

## 6. Approval Document Controller (approvalController.js)

Controller này chịu trách nhiệm quản lý các tài liệu từ phía admin chức năng phê duyệt tài liệu.

```
const approvalController = {
    approveDocument: async (req, res) => {
        const { document_id, status, reason } = req.body;
        const approved_by = req.user.id;

        try {
            const updateResult = await Document.updateApprovalStatus(document_id, status);

            if (updateResult[0].affectedRows === 0) {
                return res.status(404).json({ message: 'Document not found or status unchanged' });
            }

            const approval = await Approval.create({ document_id, approved_by, status, reason });

            res.status(200).json({
                message: 'Document approval updated successfully',
                approval: approval[0],
            });
        } catch (error) {
            res.status(500).json({ message: 'Error approving document', error });
        }
    },

    getUnapprovedDocuments: async (req, res) => {
        try {
            const documents = await Document.getUnapproved();
            res.status(200).json(documents);
        } catch (error) {
            res.status(500).json({ message: 'Error fetching unapproved documents', error });
        }
    },
};
```

- **approveDocument:**
  - **Chức năng:** Phê duyệt hoặc từ chối tài liệu.
  - **Phân tích:** Hàm này nhận thông tin từ body của yêu cầu (id tài liệu, trạng thái phê duyệt, lý do). Sau đó, nó cập nhật trạng thái phê duyệt của tài liệu trong cơ sở dữ liệu và lưu thông tin phê duyệt vào bảng approvals. Nếu tài liệu không tìm thấy hoặc không thay đổi trạng thái, hàm sẽ trả về mã lỗi 404. Nếu thành công, trả về thông báo và thông tin phê duyệt.
- **getUnapprovedDocuments:**
  - **Chức năng:** Lấy danh sách các tài liệu chưa được phê duyệt.

**Phân tích:** Hàm này gọi phương thức getUnapproved() từ mô hình Document để lấy các tài liệu chưa phê duyệt. Dữ liệu được trả về dưới dạng JSON

#### 4.1.4. Các Routes Trong Dự Án

##### 1. Routes Admin - `adminRoutes.js`

Tuyến đường này dành riêng cho các hành động của quản trị viên (admin), bao gồm việc lấy danh sách tài liệu.

```
const express = require('express');
const router = express.Router();
const { getAllDocuments } = require('../controllers/adminDocumentController');
const authMiddleware = require('../middleware/authMiddleware');
const roleMiddleware = require('../middleware/roleMiddleware');

router.get('/documents', authMiddleware, roleMiddleware(['admin']), getAllDocuments);

module.exports = router;
```

- **GET /documents:**

- **Mục đích:** Lấy tất cả tài liệu trong hệ thống (chỉ dành cho admin).
- **Middleware:**
  - authMiddleware: Kiểm tra xác thực người dùng.
  - roleMiddleware ([ 'admin' ]): Kiểm tra nếu người dùng có vai trò là admin.
- **Controller:** getAllDocuments từ adminDocumentController.

##### 2. Routes Phê Duyệt - `approvalRoutes.js`

Tuyến đường này xử lý các hành động liên quan đến phê duyệt tài liệu, chỉ cho phép admin thực hiện các thao tác phê duyệt.

```
const express = require('express');
const approvalController = require('../controllers/approvalController');
const authMiddleware = require('../middleware/authMiddleware');
const roleMiddleware = require('../middleware/roleMiddleware')
const router = express.Router();

router.post('/approve', authMiddleware, roleMiddleware('admin'), approvalController.approveDocument);
router.get('/unapproved', authMiddleware, roleMiddleware('admin'), approvalController.getUnapprovedDocuments);
```

- **POST /approve:**

- **Mục đích:** Phê duyệt tài liệu.
- **Middleware:**

- authMiddleware: Kiểm tra xác thực người dùng.
  - roleMiddleware ('admin'): Kiểm tra nếu người dùng có vai trò admin.
- **Controller:** approveDocument từ approvalController.
- **GET /unapproved:**
  - **Mục đích:** Lấy các tài liệu chưa được phê duyệt.
  - **Middleware:**
    - authMiddleware: Kiểm tra xác thực người dùng.
    - roleMiddleware ('admin'): Kiểm tra nếu người dùng có vai trò admin.
  - **Controller:** getUnapprovedDocuments từ approvalController.

### 3. Routes Đăng Ký và Đăng Nhập - authRoutes.js

Tuyến đường này bao gồm các hành động liên quan đến đăng ký, đăng nhập và lấy thông tin người dùng.

```
const express = require('express');
const { register, login } = require('../controllers/authController');
const router = express.Router();
const authMiddleware = require('../middleware/authMiddleware');
const { getProfile } = require('../controllers/authController');

router.post('/register', register);
router.post('/login', login);
router.get('/profile', authMiddleware, getProfile);

module.exports = router;
```

- **POST /register:**
  - **Mục đích:** Đăng ký người dùng mới.
  - **Controller:** register từ authController.
- **POST /login:**
  - **Mục đích:** Đăng nhập và nhận token xác thực.
  - **Controller:** login từ authController.
- **GET /profile:**
  - **Mục đích:** Lấy thông tin hồ sơ người dùng đã đăng nhập (cần xác thực qua token).
  - **Middleware:** authMiddleware để xác thực người dùng.

## 4. Routes Bảng Điều Khiển - `dashboardRoutes.js`

Tuyến đường này cung cấp các thông tin thống kê liên quan đến hệ thống, chỉ dành cho admin.

```
const express = require('express');
const router = express.Router();
const { getStats } = require('../controllers/dashboardController');
const authMiddleware = require('../middleware/authMiddleware');
const roleMiddleware = require('../middleware/roleMiddleware');

router.get('/stats', authMiddleware, roleMiddleware(['admin']), getStats);
```

- **GET /stats:**

- **Mục đích:** Lấy thông tin thống kê của hệ thống (chỉ dành cho admin).
- **Middleware:**
  - authMiddleware: Kiểm tra xác thực người dùng.
  - roleMiddleware ( ['admin' ] ): Kiểm tra nếu người dùng có vai trò admin.
- **Controller:** `getStats` từ `dashboardController`.

## 5. Routes Tài Liệu - `documentRoutes.js`

Tuyến đường này xử lý các hành động liên quan đến tài liệu như tải lên, lấy tài liệu, và tải xuống.

```
const express = require('express');
const documentController = require('../controllers/documentController');
const authMiddleware = require('../middleware/authMiddleware');
const upload = require('../middleware/uploadMiddleware');
const router = express.Router();

router.post('/upload', authMiddleware, upload.single('file'), documentController.uploadDocument);
router.get('/', documentController.getAllDocuments);
router.get('/:subject_id', documentController.getDocumentsBySubject);
router.get('/download/:id', authMiddleware, documentController.downloadDocument);
```

- **POST /upload:**

- **Mục đích:** Tải lên tài liệu mới.
- **Middleware:**
  - authMiddleware: Kiểm tra xác thực người dùng.
  - `upload.single('file')`: Middleware xử lý file tải lên.

- **Controller:** uploadDocument từ documentController.
- **GET /:**
  - **Mục đích:** Lấy tất cả tài liệu trong hệ thống.
  - **Controller:** getAllDocuments từ documentController.
- **GET /:subject\_id:**
  - **Mục đích:** Lấy tài liệu theo môn học.
  - **Controller:** getDocumentsBySubject từ documentController.
- **GET /download/:id:**
  - **Mục đích:** Tải xuống tài liệu theo ID.
  - **Middleware:** authMiddleware để xác thực người dùng.
  - **Controller:** downloadDocument từ documentController.

## 6. Routes Môn Học - `subjectRoutes.js`

Tuyến đường này xử lý các hành động liên quan đến môn học, bao gồm việc lấy tất cả các môn học.

```
const express = require('express');
const subjectController = require('../controllers/subjectController');
const router = express.Router();

router.get('/', subjectController.getAllSubjects);
```

- **GET /:**
  - **Mục đích:** Lấy danh sách tất cả môn học.
  - **Controller:** getAllSubjects từ subjectController.

#### 4.1.4. Cấu Hình Server Express

```
const express = require('express');
const bodyParser = require('body-parser'); 487.3k (gzipped: 212k)
const authRoutes = require('./routes/authRoutes');
const subjectRoutes = require('./routes/subjectRoutes');
const documentRoutes = require('./routes/documentRoutes');
const approvalRoutes = require('./routes/approvalRoutes');
const dashboardRoutes = require('./routes/dashboardRoutes');
const adminRoutes = require('./routes/adminRoutes');
const cors = require('cors'); 4.5k (gzipped: 1.9k)

const corsOptions = {
  origin: ['http://localhost:3001', 'http://localhost:3002'],
  methods: 'GET,HEAD,PUT,PATCH,POST,DELETE',
  credentials: true,
};

const app = express();

app.use(cors(corsOptions));
app.use(bodyParser.json());

app.use('/api/auth', authRoutes);
app.use('/api/subjects', subjectRoutes);
app.use('/api/documents', documentRoutes);
app.use('/api/approvals', approvalRoutes);
app.use('/api/dashboard', dashboardRoutes);
app.use('/api/admin', adminRoutes);

app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

## 1. Cấu Hình Server

- **Express.js**: Thư viện chính để xây dựng server và xử lý các tuyến đường API.
- **body-parser**: Được sử dụng để phân tích cú pháp của dữ liệu JSON trong request body.
- **cors**: Cấu hình Cross-Origin Resource Sharing (CORS) để cho phép các ứng dụng frontend từ các nguồn khác nhau giao tiếp với backend.

## 2. Cấu Hình CORS

- Cấu hình cho phép các nguồn (origin) từ `http://localhost:3001` và `http://localhost:3002` (ứng dụng React) được phép truy cập vào API.
- **CORS Options:**
  - **origin:** Xác định các nguồn được phép truy cập vào API.
  - **methods:** Cung cấp các phương thức HTTP được phép (GET, POST, PUT, DELETE, v.v.).
  - **credentials:** Cho phép gửi cookie từ phía client (nếu sử dụng).

### 3. Sử Dụng Body Parser

- `bodyParser.json()` giúp phân tích dữ liệu JSON từ body của yêu cầu (request), cho phép backend nhận và xử lý dữ liệu từ client dưới dạng JSON.

### 4. Các Routes API

Ứng dụng chia thành nhiều tuyến đường API, mỗi tuyến đường thực hiện một chức năng cụ thể.

1. **Routes Authentication - `authRoutes.js`:**
  - Đảm nhiệm các chức năng liên quan đến đăng ký (`/register`), đăng nhập (`/login`), và lấy thông tin hồ sơ người dùng đã đăng nhập (`/profile`).
2. **Routes Subject - `subjectRoutes.js`:**
  - Cung cấp các API liên quan đến môn học, ví dụ như lấy tất cả môn học (/).
3. **Routes Document - `documentRoutes.js`:**
  - Xử lý các hành động liên quan đến tài liệu, bao gồm tải lên tài liệu mới (`/upload`), lấy danh sách tài liệu (/), và tải xuống tài liệu (`/download/:id`).
4. **Routes Approval - `approvalRoutes.js`:**
  - Cung cấp các API để phê duyệt tài liệu, ví dụ như phê duyệt tài liệu (`/approve`) và lấy các tài liệu chưa được phê duyệt (`/unapproved`).
5. **Routes Dashboard - `dashboardRoutes.js`:**
  - Cung cấp thông tin thống kê hệ thống, chỉ dành cho admin (`/stats`).
6. **Routes Admin - `adminRoutes.js`:**
  - Xử lý các chức năng admin, ví dụ như lấy danh sách tài liệu (`/documents`).

#### 4.1.5. Cấu Hình Kết Nối MySQL và Kiểm Tra Kết Nối

```
const mysql = require('mysql2/promise');  780.3k (gzipped: 346.3k)
require('dotenv').config();  6.3k (gzipped: 2.7k)

const db = mysql.createPool({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'documentssharing',
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0,
});

async function testConnection() {
  try {
    const connection = await db.getConnection();
    console.log('Kết nối thành công đến MySQL');
    connection.release();
  } catch (err) {
    console.error('Kết nối thất bại: ', err);
  }
}

testConnection();
```

##### 1. Cấu Hình Kết Nối MySQL

- **Sử dụng mysql2/promise:** Thư viện mysql2/promise cho phép sử dụng cú pháp async/await, giúp xử lý các kết nối và truy vấn MySQL một cách dễ dàng và hiệu quả hơn.
- **Tạo Pool Kết Nối:**
  - **createPool** được sử dụng để tạo pool kết nối, giúp quản lý và tái sử dụng các kết nối đến cơ sở dữ liệu thay vì tạo mới mỗi khi cần kết nối.
  - **Cấu hình Pool:**
    - **host:** Địa chỉ máy chủ MySQL (ở đây là localhost).
    - **user:** Tên người dùng MySQL (ở đây là root).
    - **password:** Mật khẩu của người dùng MySQL (ở đây là chuỗi rỗng, trong thực tế cần được bảo mật).

- **database**: Tên cơ sở dữ liệu cần kết nối (ở đây là documentssharing).
- **waitForConnections**: Cấu hình này đảm bảo ứng dụng đợi các kết nối sẵn có khi pool đầy.
- **connectionLimit**: Số lượng kết nối tối đa trong pool (ở đây là 10).
- **queueLimit**: Giới hạn số lượng yêu cầu kết nối khi pool đầy (ở đây là không giới hạn).

## 2. Kiểm Tra Kết Nối

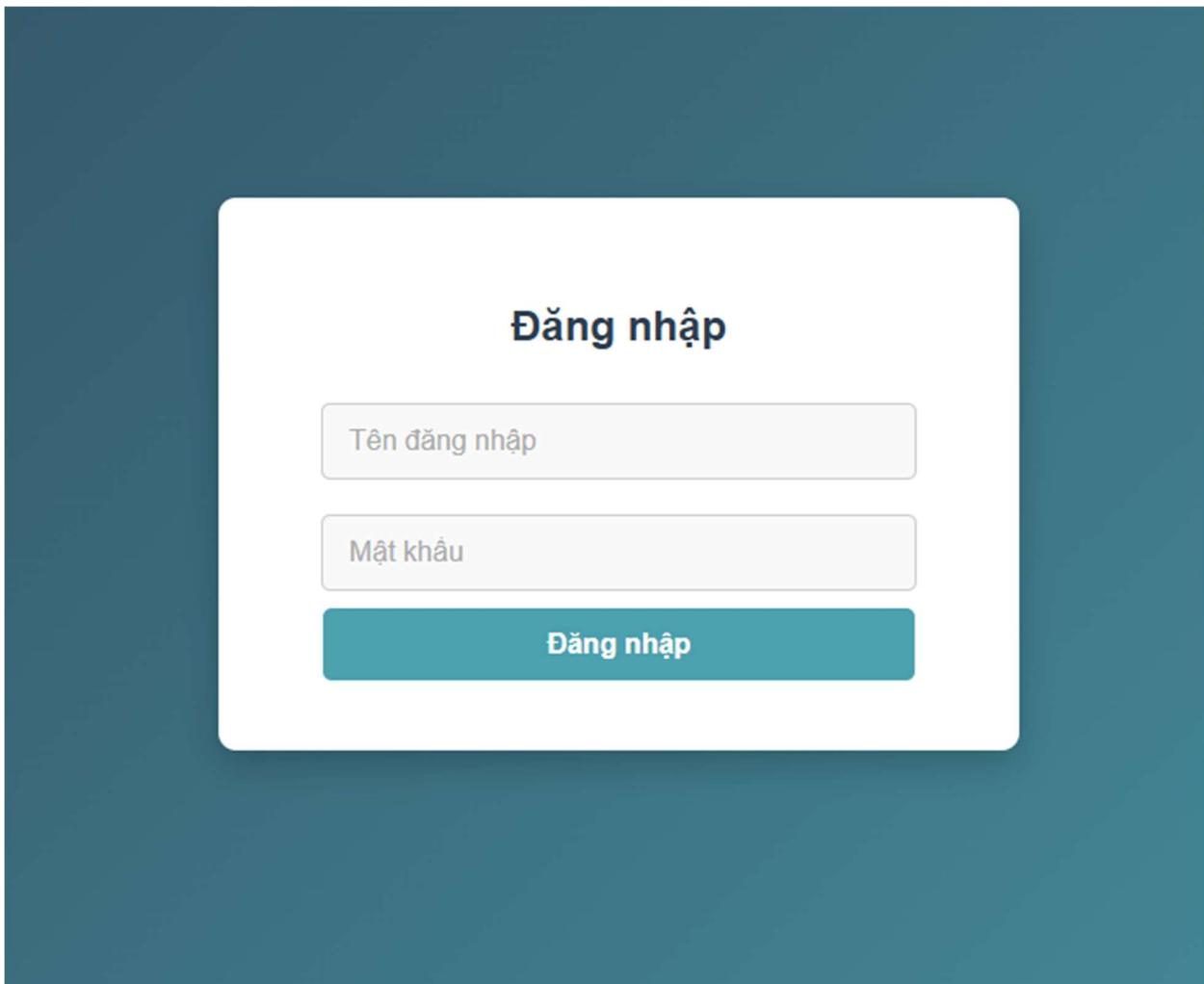
- Phương thức **testConnection** được sử dụng để kiểm tra kết nối tới cơ sở dữ liệu.
  - **await db.getConnection()** : Lấy một kết nối từ pool.
  - **connection.release()** : Sau khi thực hiện kết nối thành công, phương thức này trả lại kết nối vào pool để có thể sử dụng lại.
  - Nếu kết nối thành công, sẽ hiển thị thông báo "Kết nối thành công đến MySQL", còn nếu thất bại, thông báo lỗi sẽ được ghi lại trong console.

## 4.2 Mô tả các giao diện chính

### 4.2.1. WEB dành cho Admin để quản lý

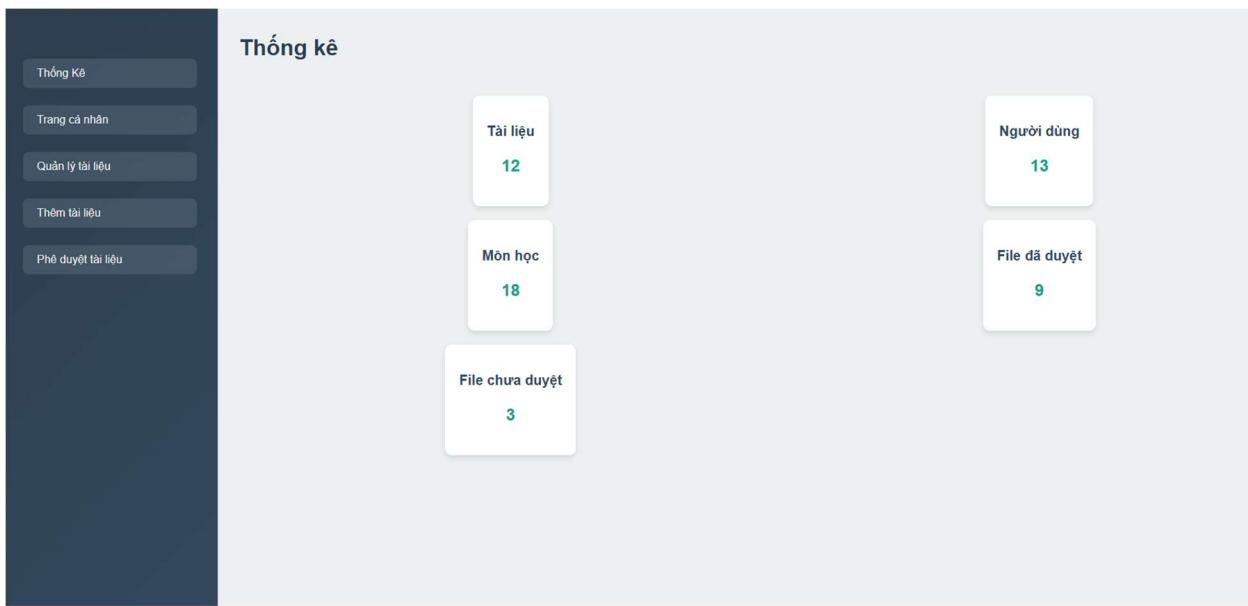
#### 1. Trang đăng nhập

Trang đăng nhập là nơi người dùng nhập thông tin đăng nhập để truy cập vào hệ thống và xác thực có phải admin không.



#### 2. Trang Thống kê

Đây là giao diện thống kê các số liệu document



### 3. Trang Quản Lý Tài Liệu

Là nơi người dùng có thể theo dõi và quản lý danh sách tài liệu đã được duyệt trong hệ thống.

**Chức năng chính:** Thêm tìm kiếm theo nội dung tài liệu và theo môn học, có cả sắp xếp môn học.

The screenshot shows a sidebar with the same navigation items as the previous page. The main content area is titled 'DANH SÁCH TÀI LIỆU' and features a table with the following columns: #, Tên tài liệu, Môn học, and Trạng thái. The table contains 9 rows of document information.

#	Tên tài liệu	Môn học	Trạng thái
1	Tài liệu về Node.js	Toán học 1	Đã duyệt
2	Tài liệu về Node.js	Toán học 1	Đã duyệt
3	Tài liệu về Node.js	Toán học 1	Đã duyệt
4	Tài liệu về Node.js	Triết học	Đã duyệt
5	Báo cáo nhóm gió tai	Kinh tế học đại cương	Đã duyệt
6	Hoàn dz	Giáo dục thể chất	Đã duyệt
7	Tuan-DCC171	Toán học 1	Đã duyệt
8	Anh Tuấn	Kỹ thuật lập trình web	Đã duyệt
9	Minh Hoàn	Lập trình di động	Đã duyệt

## 4. Trang tải tài liệu mới

Chức năng chính là thêm document.

TÀI TÀI LIỆU MỚI

Tên tài liệu

Mô tả tài liệu

Chọn môn học

Choose File No file chosen

Tải lên

## 5. Trang Phê duyệt đề thi

Trang để admin có thể phê duyệt các tài liệu được đăng bởi user hoặc admin.

PHÊ DUYỆT TÀI LIỆU

#	Tên tài liệu	Tên file	Đuôi file	Môn học	Hành động
1	Tài liệu thử nghiệm	file-123.pdf	.pdf	Toán học 1	<button>Duyệt</button> <button>Tù chối</button>
2	báo cáo nhóm gió tai	file-1734954116650-292681916.docx	.docx	Kinh tế học đại cương	<button>Duyệt</button> <button>Tù chối</button>
3	Tờ vawn huấn	file-1734955868004-416525742.docx	.docx	Lý thuyết xác suất	<button>Duyệt</button> <button>Tù chối</button>

## 6. Trang Cá nhân

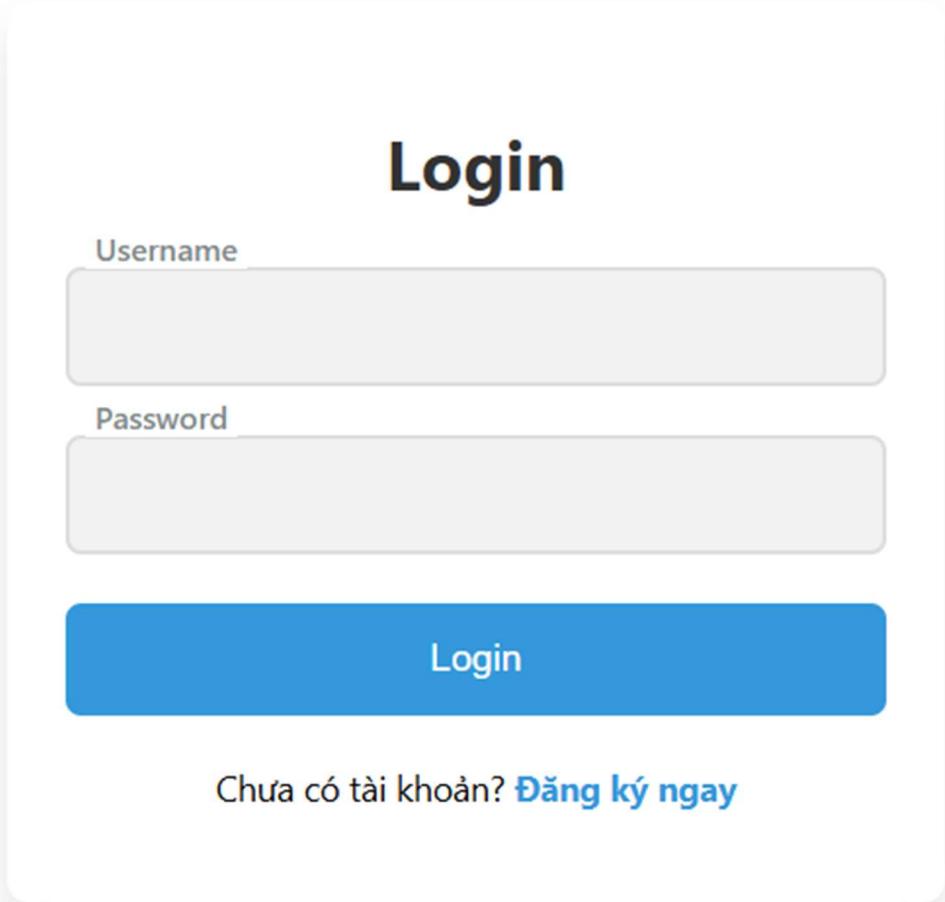
Thông tin cá nhân

Tên: NATTuan16012005  
Email: NATTuan16012005@gmail.com  
Vai trò: admin

#### 4.2.2. WEB dành cho người dùng

##### 1. Trang Đăng ký , Đăng nhập

Trang này admin và user có thể đăng nhập



The image shows a login form interface. At the top center is the word "Login" in a large, bold, dark blue font. Below it are two input fields: the first is labeled "Username" and the second is labeled "Password", both in a smaller, gray font. Both input fields have a light gray rounded rectangular background. Below these fields is a large blue button with the word "Login" in white. At the bottom left of the form, there is a link in blue text that reads "Chưa có tài khoản? [Đăng ký ngay](#)". The entire form is set against a white background with a soft shadow.

# Register

Username

Email

Password

Confirm Password

Register

Đã có tài khoản? [Đăng nhập ngay](#)

## 2. Trang Chủ

Trang này cho phép người dùng xem tài liệu và tải tài liệu, có thêm chức năng tìm kiếm tài liệu.

The screenshot shows the Home page with a dark sidebar on the left containing a user profile (Tuan1601) and navigation links (Trang chủ, Trang cá nhân, Thêm tài liệu). The main content area is titled "Trang chủ" and "Danh sách tài liệu". It features a search bar and a list of four items, each representing a document about Node.js:

- Tài liệu về Node.js (Tài liệu học về Node.js)  
Buttons: Tải về (blue), Xem trước (pink)
- Tài liệu về Node.js (Tài liệu học về Node.js)  
Buttons: Tải về (blue), Xem trước (pink)
- Tài liệu về Node.js (Tài liệu học về Node.js)  
Buttons: Tải về (blue), Xem trước (pink)
- Tài liệu về Node.js (Tài liệu học về Node.js)  
Buttons: Tải về (blue), Xem trước (pink)

Pagination at the bottom shows page 1 of 3.

## 3. Trang Cá nhân

The screenshot shows the Personal Information page with a dark sidebar on the left containing a user profile (Tuan1601) and navigation links (Trang chủ, Trang cá nhân, Thêm tài liệu). The main content area is titled "THÔNG TIN CÁ NHÂN" and displays the following user details:

**Tên:** Tuan1601  
**Email:** Tuan1601@gmail.com  
**Vai trò:** user

## 4. Trang Thêm tài liệu

The screenshot shows a user interface for adding a new document. At the top left, there is a vertical sidebar with the text "NATTuan16012005" and a red button labeled "Đăng xuất" (Logout). The main content area has a title "TÀI LIỆU MỚI" (New Document). Below the title are four input fields: "Tên tài liệu" (Document name), "Mô tả tài liệu" (Description), "Chọn môn học" (Select subject), and a file upload field with the placeholder "Choose File | No file chosen". A large blue button at the bottom right is labeled "Tải lên" (Upload).

## CHƯƠNG 5: KẾT LUẬN

Hệ thống chia sẻ tài liệu trực tuyến là một giải pháp hiệu quả nhằm hỗ trợ các cơ sở giáo dục và người học trong việc chia sẻ, quản lý và truy cập tài liệu học tập một cách dễ dàng và tiện lợi. Bằng việc ứng dụng công nghệ vào việc quản lý tài liệu, hệ thống giúp giảm thiểu các thủ tục phức tạp và rút ngắn thời gian tìm kiếm tài liệu, từ đó nâng cao hiệu quả học tập và giảng dạy.

Website không chỉ cung cấp các chức năng cơ bản như quản lý tài liệu, đăng tải tài liệu mà còn hỗ trợ các tính năng quan trọng như duyệt tài liệu, phân loại theo môn học, và cho phép người dùng tải về hoặc xem nội dung tài liệu trực tiếp. Việc ứng dụng công nghệ vào quản lý tài liệu giúp đảm bảo rằng người dùng có thể dễ dàng tìm thấy tài liệu cần thiết và chia sẻ thông tin một cách hiệu quả.

Web đã đáp ứng đầy đủ các chức năng chính sau:

- Quản lý tài liệu chia sẻ.
- Quản lý người dùng và quyền truy cập.
- Duyệt tài liệu và phân loại theo môn học.
- Tải lên tài liệu.
- Cung cấp báo cáo tổng quan về hoạt động chia sẻ tài liệu.

Website hoạt động ổn định, đáp ứng đầy đủ các yêu cầu đã đề ra. Các tính năng quản lý tài liệu và người dùng đều hỗ trợ các thao tác như thêm và tìm kiếm tài liệu một cách dễ dàng.

Sản phẩm đạt được:

- Một website chia sẻ tài liệu trực tuyến với đầy đủ các tính năng cần thiết.
- Hỗ trợ người dùng dễ dàng truy cập và chia sẻ tài liệu học tập.
- Cung cấp các công cụ quản lý tài liệu hiệu quả cho người quản trị.
- Tăng cường trải nghiệm người dùng với tính năng tải lên, duyệt và tìm kiếm tài liệu dễ dàng.

**Ưu điểm:**

- Giao diện thân thiện, dễ sử dụng.
- Đơn giản nhưng đáp ứng đầy đủ nhu cầu chia sẻ tài liệu.
- Có tính ứng dụng thực tế cao.
- Áp dụng các công nghệ hiện đại.
- Tính năng xác thực và phân quyền giúp bảo mật dữ liệu.

## **Khuyết điểm:**

- Chưa tối ưu hóa cơ sở dữ liệu.
- Chưa hỗ trợ report với nhiều loại file khác nhau hoặc cấu trúc file phức tạp.
- Gặp 1 số vấn đề khi tải tài liệu và xem tài liệu.

## **Hướng phát triển:**

Hiện tại, website đã hỗ trợ tốt cho việc quản lý tài liệu, và trong tương lai có thể phát triển thêm:

- Mở rộng thêm các tính năng hỗ trợ việc quản lý hiệu quả hơn.
- Cải thiện chức năng tải xuống và xem tài liệu trực tiếp trên web.
- Nâng cấp giao diện nhìn hiện đại hơn.
- Nâng cao tính bảo mật cho hệ thống.
- Tối ưu hóa cơ sở dữ liệu.
- Cải thiện tính năng report, hỗ trợ nhiều loại file với cấu trúc đơn giản hơn.

## **TÀI LIỆU THAM KHẢO**

1. <https://www.youtube.com/>
2. <https://chatgpt.com/>
3. <https://vu-van-thuong.gitbook.io/web>