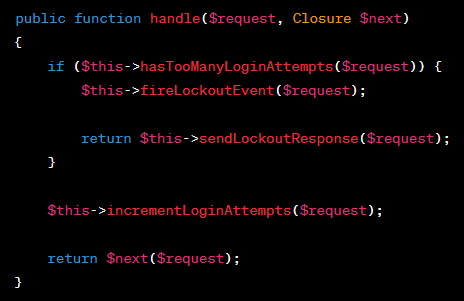
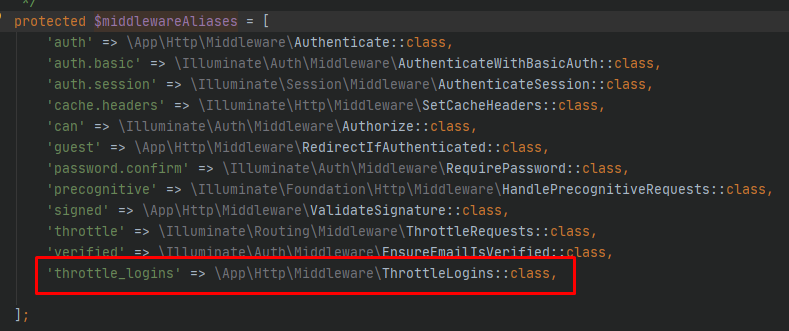
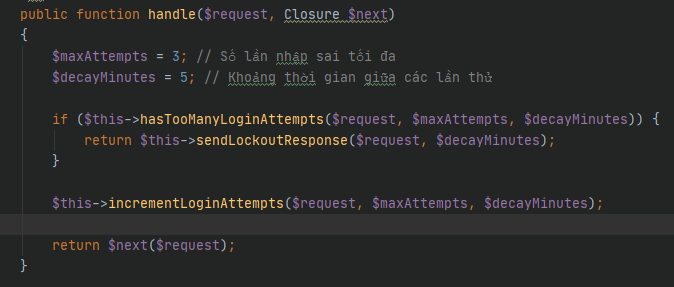
**Quản lý chia sẻ tài liệu ! ( Ghi chú các chức năng khó )**

1. **Đăng ký/ Đăng nhập người dùng:**
   1. Ngăn chặn dò mật khẩu :  
      - Tạo middleware : ***php artisan make:middleware ThrottleLogins   
      -*** Sau đó, mở tệp **app/Http/Middleware/ThrottleLogins.php** và cập nhật phương thức **handle:**
   2. Đăng ký Middleware  
      - Mở tệp app/Http/Kernel.php và thêm middleware vào mảng $routeMiddleware:  
        
      - Thêm Middleware vào Route Login :   
      
   3. Cấu hình số lần nhập sai và thời gian chờ  
      Mở tệp **app/Http/Middleware/ThrottleLogins.php** và cấu hình số lần nhập sai (**maxAttempts**) và khoảng thời gian (**decayMinutes**) trong phương thức **hasTooManyLoginAttempts**:  
      

=> Code hoàn chỉnh file ***ThrottleLogins.php*** :

<?php  
  
namespace App\Http\Middleware;  
  
use Closure;  
use Illuminate\Cache\RateLimiter;  
use Illuminate\Http\Request;  
use Symfony\Component\HttpFoundation\Response;  
  
class ThrottleLogins  
{  
 */\*\*  
 \* Handle an incoming request.  
 \*  
 \** ***@param*** *\Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next  
 \*/* public function handle($request, Closure $next)  
 {  
 $maxAttempts = 3; // Số lần nhập sai tối đa  
 $decayMinutes = 5; // Khoảng thời gian giữa các lần thử  
  
 if ($this->hasTooManyLoginAttempts($request, $maxAttempts, $decayMinutes)) {  
 return $this->sendLockoutResponse($request, $decayMinutes);  
 }  
  
 $this->incrementLoginAttempts($request, $maxAttempts, $decayMinutes);  
  
 return $next($request);  
 }  
  
 protected function hasTooManyLoginAttempts(Request $request, $maxAttempts, $decayMinutes)  
 {  
 return app(RateLimiter::class)->tooManyAttempts(  
 $this->resolveRequest($request), $maxAttempts, $decayMinutes  
 );  
 }  
  
 protected function resolveRequest(Request $request)  
 {  
 return sha1(  
 $request->getMethod().'|'.$request->ip().'|'.$request->path().'|'.$request->input('email')  
 );  
 }  
  
 protected function sendLockoutResponse($request, $decayMinutes)  
 {  
 $retryAfter = app(RateLimiter::class)->availableIn(  
 $this->resolveRequest($request), $decayMinutes \* 60  
 );  
  
 return back()->with('error', ' Bạn đã nhập quá lần thử, xin vui lòng đợi trong '.$retryAfter.' giây!');  
 }  
  
 protected function incrementLoginAttempts($request, $maxAttempts, $decayMinutes)  
 {  
 app(RateLimiter::class)->hit(  
 $this->resolveRequest($request), $decayMinutes \* 60  
 );  
 }  
}

2**. Đăng nhập bằng GOOGLE.**

Để thực hiện chức năng đăng nhập bằng tài khoản Google, bạn cần sử dụng OAuth 2.0 để xác thực người dùng và lấy thông tin từ tài khoản Google. Dưới đây là các bước tổng quan để thực hiện điều này:

1. **Tạo ứng dụng trên Google Developer Console:**
   * Đăng nhập vào [Google Developer Console](https://console.developers.google.com/).
   * Tạo một ứng dụng mới và cấu hình nó để sử dụng OAuth 2.0.
   * Nhận Client ID và Client Secret từ ứng dụng.
2. **Cài đặt các thư viện và gói mở rộng:**

Laravel cung cấp một số gói mở rộng để hỗ trợ đăng nhập bằng tài khoản Google. Một số gói phổ biến là **socialite** và **laravel/socialite**.

Để cài đặt **laravel/socialite**, chạy lệnh Composer:

composer require laravel/socialite

1. **Cấu hình ứng dụng Laravel:**

* Mở file **.env** và cung cấp thông tin OAuth 2.0 của Google:

GOOGLE\_CLIENT\_ID=your\_client\_id

GOOGLE\_CLIENT\_SECRET=your\_client\_secret

GOOGLE\_REDIRECT\_URI=your\_redirect\_uri

* Thêm thông tin cấu hình Google vào **config/services.php**:

'google' => [

'client\_id' => env('GOOGLE\_CLIENT\_ID'),

'client\_secret' => env('GOOGLE\_CLIENT\_SECRET'),

'redirect' => env('GOOGLE\_REDIRECT\_URI'),

],

**4.Sử dụng Socialite trong Laravel:**

* Sử dụng **Socialite** trong controller:

use Laravel\Socialite\Facades\Socialite;

* Tạo các route và phương thức xử lý cho đăng nhập bằng tài khoản Google.

Ví dụ:

Route::get('auth/google', 'AuthController@redirectToGoogle');

Route::get('auth/google/callback', 'AuthController@handleGoogleCallback');

**5. Controller xử lý:**

public function redirectToGoogle()

{

return Socialite::driver('google')->redirect();

}

public function handleGoogleCallback()

{

$user = Socialite::driver('google')->user();

// Xử lý dữ liệu người dùng và đăng nhập

}

**6. Giao diện người dùng:**

* Tạo một nút hoặc liên kết để người dùng có thể chọn đăng nhập bằng Google.
* Liên kết đến **/auth/google** để bắt đầu quy trình đăng nhập.

**7. Xử lý dữ liệu người dùng:**

* Trong phương thức **handleGoogleCallback**, bạn sẽ nhận được thông tin về người dùng từ Google.
* Từ đó, bạn có thể kiểm tra xem người dùng đã tồn tại trong hệ thống của bạn chưa và thực hiện đăng nhập.

**II. PHÂN QUYỀN**

1. **Tạo middleware**

* *Chạy lệnh : php artisan make:middleware CheckLevel*

1. Sau đó, mở file **CheckLevel.php** trong thư mục **app/Http/Middleware** và cập nhật hàm **handle** như sau:

*public function handle($request, Closure $next)*

*{*

*if (auth()->check() && auth()->user()->level == 0) {*

*return redirect('/');*

*}*

*return $next($request);*

*}*

1. Thêm Middleware mới vào kernel (**app/Http/Kernel.php**). Thêm tên middleware mới vào phần **$routeMiddleware**:

*protected $routeMiddleware = [*

*// ...*

*'checkLevel' => \App\Http\Middleware\CheckLevel::class,*

*];*

1. Chỉnh Route:   
   

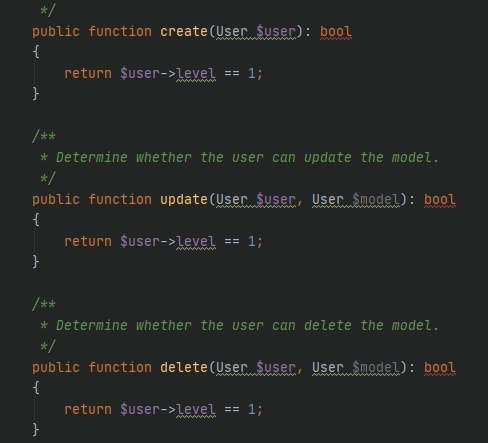
**III. PHÂN QUYỀN ĐỂ CHỈ ADMIN MỚI CÓ THỂ THÊM SỬA XÓA CÁC TÀI KHOẢN**

1. **Tạo một policy cho model User**

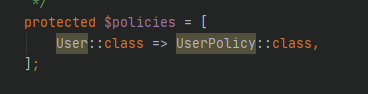
*php artisan make:policy UserPolicy --model=User*

* + Sau khi chạy lệnh này, Laravel sẽ tạo một file **UserPolicy.php** trong thư mục **app/Policies**.

1. Mở file **UserPolicy.php** và thêm các hàm kiểm tra quyền:

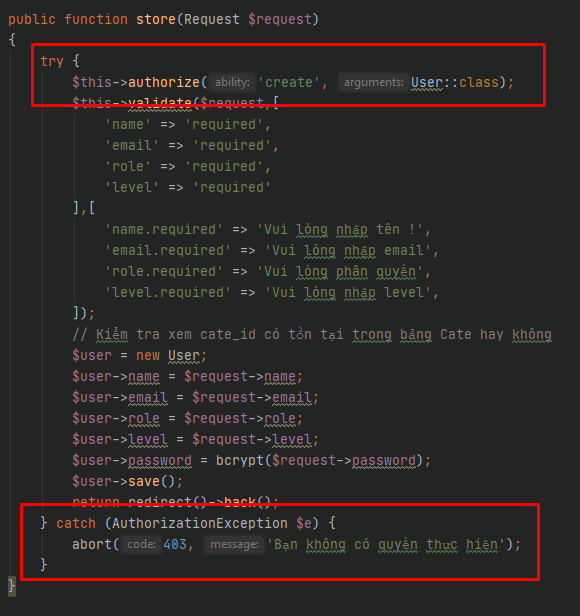


1. Đăng ký policy trong file **Providers**/**AuthServiceProvider.php**:

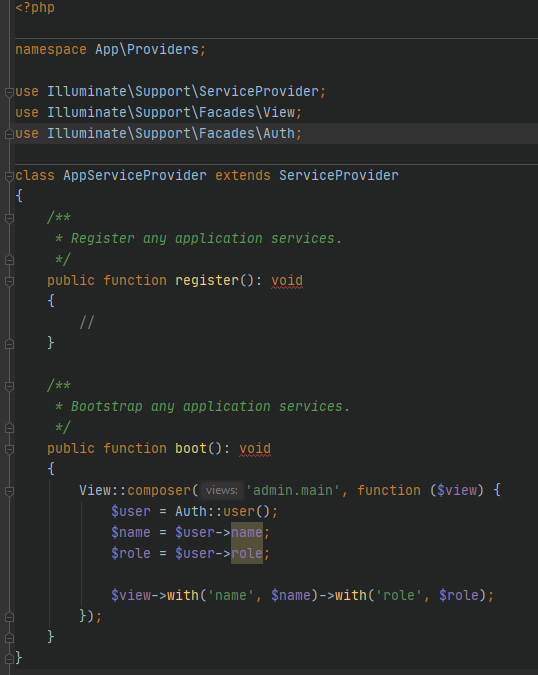


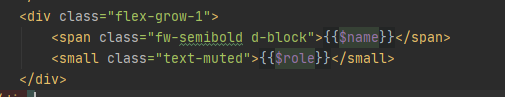
1. Sử dụng policy trong controller **UsersController**:





**IV. CÁCH HIỂN THỊ 1 SỐ INFO USER MÀ KHÔNG CẦN CONTROLLER RIÊNG**

* Vào file ***Providers/AppServiceProvider.php*** và code thế này :   
  
  + File ***main.blade.php*** :



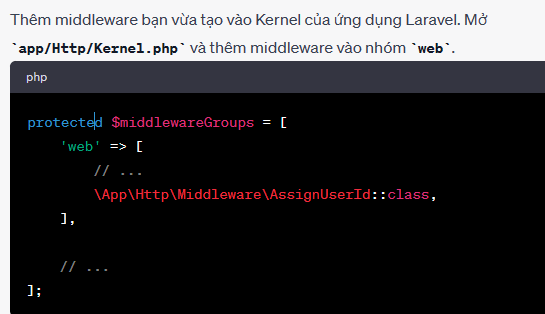
**V. Tự động gán user\_id khi thêm mới một tài liệu :**

**1. Tạo middleware**

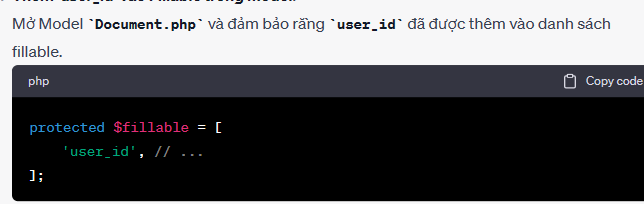
**php artisan make:middleware AssignUserId**



1. **Đăng ký middleware**



1. **Thêm 'user\_id' vào Fillable trong Model**



1. **Controller - Thêm mới Document:**

* Trong hàm trong Controller, bạn không cần phải cung cấp **user\_id** bởi vì Middleware sẽ làm điều đó.  
   **$document->user\_id = Auth::id();**

**VI. Xem chi tiết các file docx, pdf, xlsx, pptx :**

1. ***Cài đặt gói :****composer require vish4395/laravel-file-viewer*
2. **Xuất bản Tài nguyên (Assets)**

*php artisan vendor:publish --provider="Vish4395\LaravelFileViewer\LaravelFileViewerServiceProvider" --tag=assets*

1. **Xuất bản View**

*php artisan vendor:publish --provider="Vish4395\LaravelFileViewer\LaravelFileViewerServiceProvider" --tag=views*

1. **Controller Show View**



**VII. Chuyển Files sang PDF:**

1. **Cài đặt gói :**

*composer require convertapi/convertapi-php*

1. **Lấy API – SECRET :**

*use \ConvertApi\ConvertApi;*

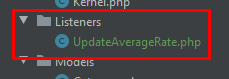
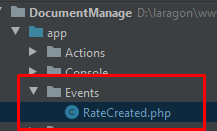
*ConvertApi::setApiSecret('your-api-secret');*

1. **Controller Document :**
2. if ($request->hasFile('file')) {  
    $file = $request->file('file');  
    $fileSize = $file->getSize();  
    $fileSizeKB = $fileSize / 1024;  
    $document->size = round($fileSizeKB, 2) . ' KB';  
    $fileName = $file->getClientOriginalName(); // Lấy tên gốc của file  
    $fileExtension = $file->getClientOriginalExtension(); // Lấy phần đuôi của file  
    $file->storeAs('public/filesOrigin', $fileName);  
     
    // Lưu tệp tin tạm thời  
    $file->storeAs('public/files', $fileName);  
     
    // Đường dẫn tệp tin tạm thời  
    $tempFilePath = storage\_path('app/public/files/' . $fileName);  
     
    // Chuyển đổi tệp thành PDF  
    ConvertApi::*setApiSecret*('6QSfRhA7Nr905F3u'); // Thay 'your-api-secret' bằng API Secret của bạn  
    $result = ConvertApi::*convert*('pdf', ['File' => $tempFilePath]);  
     
    // Lưu tệp PDF  
    $pdfFileName = pathinfo($fileName, *PATHINFO\_FILENAME*) . '.pdf';  
    $pdfFilePath = storage\_path('app/public/files/' . $pdfFileName);  
    $result->getFile()->save($pdfFilePath);  
     
    // Gán tên tệp PDF vào thuộc tính 'file'  
    $document->file = $pdfFileName;  
     
    // Xóa tệp tin tạm thời  
    unlink($tempFilePath);  
   }

**VIII. Tự động cập nhật điểm Đánh giá trung bình cho mỗi tài liệu:**

* 1. *Tạo Listeners và Events :*

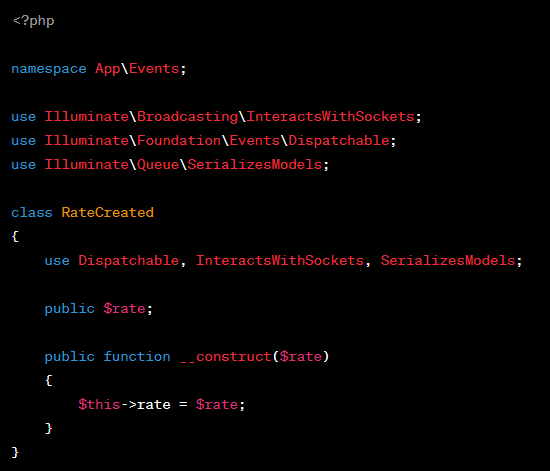




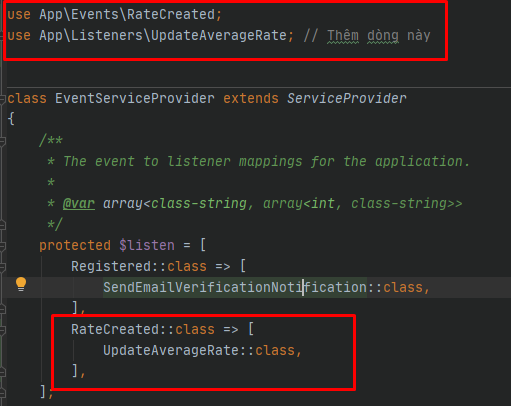
* 1. *Mở file* ***UpdateAverageRate.php*** *và cập nhật phương thức* ***handle*** *để tính toán lại điểm trung bình và cập nhật cột* ***rate*** *trong bảng* ***documents****:*



* 1. *File* ***RateCreated.php*** *định nghĩa sự kiện* ***RateCreated****. Trong trường hợp của bạn, sự kiện này được sử dụng để tạo ra một sự kiện khi một đánh giá mới được tạo.*



* 1. Sau đó, đặt file của bạn vào đó và tiếp theo bạn cần phải đảm bảo rằng bạn đã đăng ký sự kiện và listener của mình đúng cách. Điều này thường được thực hiện trong file **EventServiceProvider** trong **app/Providers**.



* 1. Ở function **Rate** để đánh giá ta thêm dòng code sau :