Trường ĐH Công Thương TP.HCM Khoa: CNTT Bộ môn: Kỹ thuật phần mềm **LẬP TRÌNH MÃ NGUỒN MỞ**	**BÀI 9** **LARAVEL FRAMEWORK** **Validation, Middleware, Authentication & Authorization**	

A. MỤC TIÊU

- Vận dụng Validation ở mức thực tiễn (Form Request, custom Rule, thông điệp lỗi, xác thực mảng/tệp, điều kiện).
- Cấu hình và áp dụng Middleware theo chuẩn Laravel 12 (alias/group/global, CSRF, throttle).
- Tích hợp Authentication (Breeze) để bảo vệ các chức năng; nắm khái niệm guards / providers.
- Thực hiện Authorization bằng Gate/Policy, ràng buộc quyền ở cả route và Blade.
- Thiết lập tối thiểu các thực hành an toàn: CSRF, XSS (escape), hash mật khẩu, giới hạn tốc độ (rate limit).

B. NỘI DUNG THỰC HÀNH

1. Cơ sở lý thuyết

- Validation (Laravel 12)
  - Cách dùng: inline (`$request->validate()`), Form Request (`php artisan make:request`), Rule class (`php artisan make:rule`).
  - Rules thông dụng: `required`, `string`, `max`, `email`, `unique`, `exists`, `numeric`, `array`, `mimes`, `image`, `min`, `confirmed`, `sometimes`, `nullable`, `date`, `after`/`before`.
  - Kỹ thuật: bail (dừng sớm), xác thực mảng động (items.*.field), file upload (kích thước/định dạng), conditional validation.
  - Thông điệp lỗi và tùy biến thuộc tính (localization), hiển thị lỗi với `@error`.
- Middleware (Laravel 12)
  - Vai trò: lọc/tiền xử lý request/response; ví dụ: `auth`, `verified`, `throttle`, `CSRF`.
  - Cấu hình tập trung tại *bootstrap/app.php* qua `->withMiddleware(...)`:
  - Alias (đặt tên ngắn), append/prepend/remove theo group (web, api), global, CSRF exceptions.
  - Áp dụng ở route/route group bằng tên alias; kết hợp với nhóm web

(session/csrf/cookie).

- Authentication
  - Breeze: scaffolding đăng ký/đăng nhập, quên mật khẩu, email verification (tùy chọn).
  - Guards/providers: mặc định web (session) dùng users provider; có thể mở rộng.
  - Hash mật khẩu (bcrypt/argon), bảo vệ route bằng auth.
- Authorization
  - Gate: kiểm tra hành động bằng closure (nhanh, đơn giản).
  - Policy: ánh xạ quyền theo mô hình (model-based): `viewAny`, `view`, `create`, `update`, `delete`, `restore`, `forceDelete`.
  - Ràng buộc quyền ở Controller/Route/Blade: `Gate::allows()`, `@can`, middleware `can:`.
- An toàn & thực hành tốt
  - CSRF (`@csrf`), XSS (escape Blade), SQL Injection (Prepared Statement qua Eloquent), throttle đăng nhập, APP_DEBUG=false trên production.

2. Bài tập tại lớp

Chuẩn bị trước buổi học (yêu cầu sinh viên)

- Dự án Laravel 12 đã chạy ổn định; DB MySQL cấu hình .env.
- Có sẵn module Article (from Lab 7–8) hoặc skeleton tương đương (User đăng nhập được).

Bài tập 01: Chuẩn hóa Validation với Form Request

Yêu cầu:

- Tách logic validation hiện viết inline trong Controller (khi tạo/sửa Article) ra thành Form Request riêng biệt.
- Cấu hình rules, messages (thông điệp lỗi) và attributes (tên trường hiển thị thân thiện).
- Áp dụng các luật xác thực:
  - `required`, `string`, `max:255` cho `title`.
  - `unique:articles,title` để tiêu đề không trùng lặp.
  - `required|min:10` cho `body`.
  - `sometimes|nullable` cho các trường phụ (ví dụ: tags).
- Hiển thị lỗi rõ ràng tại view, giữ lại dữ liệu người dùng đã nhập (`old()`).

Mục tiêu đạt được:

- Hiểu và sử dụng Form Request để viết validation gọn gàng, có tổ chức.
- Biết cách tách logic kiểm tra dữ liệu ra khỏi Controller → Controller "sạch" và dễ bảo trì.
- Thực hành tuỳ biến thông điệp lỗi và tên trường.
- Giữ lại dữ liệu người dùng khi form có lỗi, đảm bảo trải nghiệm tốt.

Hướng dẫn:

- Tạo Form Request

```
php artisan make:request StoreArticleRequest
```

- Sửa file *app/Http/Requests/StoreArticleRequest.php*

```
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreArticleRequest extends FormRequest
{
    public function authorize(): bool
    {
        return true; // Cho phép tất cả (có thể thêm logic phân quyền sau)
    }

    public function rules(): array
    {
        return [
            'title' =>
['required','string','max:255','unique:articles,title'],
            'body'  => ['required','string','min:10'],
            'tags'  => ['sometimes','nullable','string'],
        ];
    }

    public function messages(): array
    {
        return [
            'title.required' => 'Tiêu đề không được để trống',
            'title.unique'   => 'Tiêu đề đã tồn tại, vui lòng chọn tiêu đề
khác',
            'body.required'  => 'Nội dung không được để trống',
            'body.min'       => 'Nội dung tối thiểu phải có :min ký tự',
        ];
    }
```

```
    public function attributes(): array
    {
        return [
            'title' => 'Tiêu đề',
            'body'  => 'Nội dung',
        ];
    }
}
```

- Sử dụng Form Request trong Controller

Trong `ArticleController@store` và `@update`, thay thế `$request->validate(...)` bằng type-hint Form Request:

```
use App\Http\Requests\StoreArticleRequest;

public function store(StoreArticleRequest $request)
{
    // Dữ liệu hợp lệ đã được validate
    $validated = $request->validated();
    // Tạm thời: giả lưu, thực tế sẽ lưu DB ở buổi sau
    return redirect()->route('articles.index')
        ->with('success', 'Tạo bài viết thành công (demo).');
}
```

Tương tự cho `update()`.

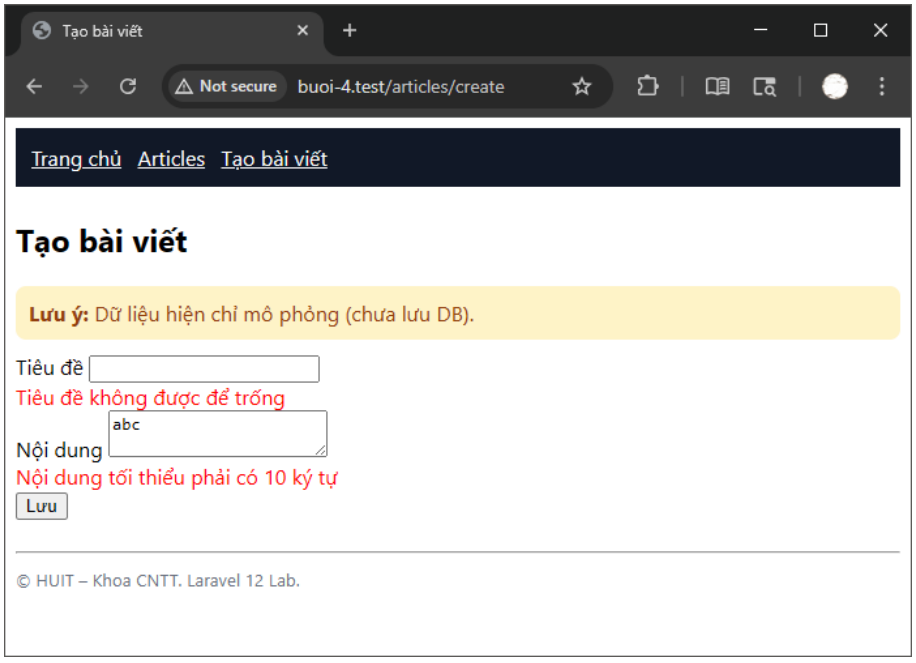
- Hiển thị lỗi trong View
  - Trong form tạo/sửa bài viết (*articles/create.blade.php*):

```
<form action="{{ route('articles.store') }}" method="post">
  @csrf

  <label>Tiêu đề</label>
  <input type="text" name="title" value="{{ old('title') }}">
  @error('title')
  <div style="color:red">{{ $message }}</div>
  @enderror

  <label>Nội dung</label>
  <textarea name="body">{{ old('body') }}</textarea>
  @error('body')
  <div style="color:red">{{ $message }}</div>
  @enderror

  <button type="submit">Lưu</button>
```

```
</form>
```

Kết quả mong đợi

- Khi nhập dữ liệu thiếu/không hợp lệ → hệ thống hiển thị lỗi dưới mỗi trường, giữ lại input đã nhập.
- Khi nhập hợp lệ → redirect về danh sách Articles kèm thông báo "Tạo bài viết thành công".
- Controller gọn hơn, không chứa logic validation chi tiết.



Hình 1 Kết quả bài 1

Lỗi thường gặp & khắc phục

- Validation không chạy: Quên type-hint StoreArticleRequest trong Controller.
- Thông điệp lỗi mặc định tiếng Anh: Chưa khai báo `messages()` hoặc chưa cấu hình localization.
- Lỗi unique: Chưa có bảng articles trong DB → cần migration trước (Lab 7).
- Không giữ dữ liệu cũ: Quên `old()` trong input.

Bài tập 02: Custom Rule & File Validation

Yêu cầu:

- Tạo Rule tuỳ chỉnh để kiểm soát hợp lệ của title (ví dụ: không chứa từ cấm hoặc không trùng tiêu đề trong ngày).
- Bổ sung upload ảnh minh hoạ cho bài viết (tuỳ chọn, không bắt buộc) với ràng buộc:
  - `image`, `mimes:jpg,jpeg,png`, `max:2048` (KB).
- Cập nhật Form Request của bài viết để sử dụng Rule và validate file.
- Lưu tệp ảnh (nếu có) vào *storage/app/public/articles/* và lưu đường dẫn vào CSDL.

- Hiển thị ảnh ở trang chi tiết/danh sách bài viết.

Mục tiêu đạt được

- Thành thạo xây dựng Custom Rule (class-based).
- Nắm cú pháp File Validation và quy trình upload–lưu đường dẫn–hiển thị.
- Tổ chức validation "sạch" trong Form Request và Controller.

Hướng dẫn:

- Chuẩn bị CSDL (nếu chưa có cột ảnh)
    - Tạo migration thêm cột để lưu đường dẫn ảnh (ví dụ image_path):

```
php artisan make:migration add_image_path_to_articles_table --table=articles
```

    - Sửa file migration:

```
public function up(): void
{
    Schema::table('articles', function (Blueprint $table) {
        $table->string('image_path')->nullable()->after('body');
    });
}
```

    - Chạy migration:

```
php artisan migrate
```

Lưu ý: Nếu chưa có bảng articles, có thể dùng bảng demo ở Buổi 2/3 hoặc tạo nhanh migration + model.

- Tạo Custom Rule
    - Ví dụ Rule cấm từ nhạy cảm xuất hiện trong title (danh sách từ cấm tự định nghĩa).

```
php artisan make:rule NoForbiddenWords
```

    - Sửa *app/Rules/NoForbiddenWords.php*:

```
namespace App\Rules;

use Closure;
use Illuminate\Contracts\Validation\ValidationRule;

class NoForbiddenWords implements ValidationRule
{
    /**
     * Run the validation rule.
     *
     * @param  \Closure(string, ?string=):
\Illuminate\Translation\PotentiallyTranslatedString  $fail
     */
```

```
    public function validate(string $attribute, mixed $value, Closure $fail):
void
    {
        $forbidden = ['test', 'spam', 'xxx']; // ví dụ: danh sách từ cấm
        $lower = mb_strtolower((string)$value, 'UTF-8');

        foreach ($forbidden as $w) {
            if (str_contains($lower, $w)) {
                $fail('Trường :attribute chứa từ không được phép: "' . $w .
'".');
                return;
            }
        }
    }
}
```

Gợi ý biến thể nâng cao: Rule "duy nhất trong ngày" (kiểm tra tiêu đề đã tồn tại whereDate(created_at, today())).

- Cập nhật Form Request để dùng Rule + File Validation
  - Giả sử đang dùng `StoreArticleRequest`/`UpdateArticleRequest`. Ví dụ với `StoreArticleRequest`

```
use App\Rules\NoForbiddenWords;

class StoreArticleRequest extends FormRequest
{
    public function rules(): array
    {
        return [
            'title' => ['required', 'string', 'max:255',
'unique:articles,title', new NoForbiddenWords],
            'body'  => ['required', 'string', 'min:10'],
            // ảnh: tùy chọn
            'image' => ['sometimes', 'nullable', 'image',
'mimes:jpg,jpeg,png', 'max:2048'],
        ];
    }

    public function messages(): array
    {
        return [
            'title.required' => 'Tiêu đề không được để trống',
            'title.unique'   => 'Tiêu đề đã tồn tại, vui lòng chọn tiêu đề
khác',
```

```
            'body.required'  => 'Nội dung không được để trống',
            'body.min'       => 'Nội dung tối thiểu phải có :min ký tự',
            'image.image' => 'Tệp tải lên phải là hình ảnh.',
            'image.mimes' => 'Ảnh phải có định dạng: jpg, jpeg hoặc png.',
            'image.max'   => 'Kích thước ảnh tối đa là :max KB.',
        ];
    }

    public function attributes(): array
    {
        return [
            'title' => 'Tiêu đề',
            'body'  => 'Nội dung',
            'image' => 'Ảnh minh hoạ',
        ];
    }
}
```

Ghi chú: Với Update, thay `unique:articles,title` bằng `Rule::unique('articles','title')->ignore($this->route('article'))` để bỏ qua chính bản ghi đang sửa.

– Sửa Controller để xử lý upload & lưu đường dẫn

- Ví dụ `ArticleController@store`:

```
public function store(StoreArticleRequest $request)
{
    $data = $request->validated();

    // Xử lý ảnh (nếu có)
    if ($request->hasFile('image')) {
        // Lưu vào disk 'public' (đường dẫn: storage/app/public/articles/...)
        $path = $request->file('image')->store('articles', 'public');
        $data['image_path'] = $path; // lưu đường dẫn tương đối
    }

    Article::create($data);

    return redirect()->route('articles.index')
        ->with('success', 'Tạo bài viết thành công');
}
```

- Với `update()`:

```
use Illuminate\Support\Facades\Storage;
use App\Http\Requests\UpdateArticleRequest;
```

```
use App\Models\Article;

public function update(UpdateArticleRequest $request, Article $article)
{
    $data = $request->validated();

    if ($request->hasFile('image')) {
        // Xoá ảnh cũ (nếu có)
        if (!empty($article->image_path) && Storage::disk('public')-
>exists($article->image_path)) {
            Storage::disk('public')->delete($article->image_path);
        }
        $data['image_path'] = $request->file('image')->store('articles',
'public');
    }

    $article->update($data);

    return redirect()->route('articles.show', $article)
        ->with('success', 'Cập nhật bài viết thành công');
}
```

Lưu ý: đảm bảo đã chạy `php artisan storage:link` để có symlink *public/storage* trỏ đến *storage/app/public*.

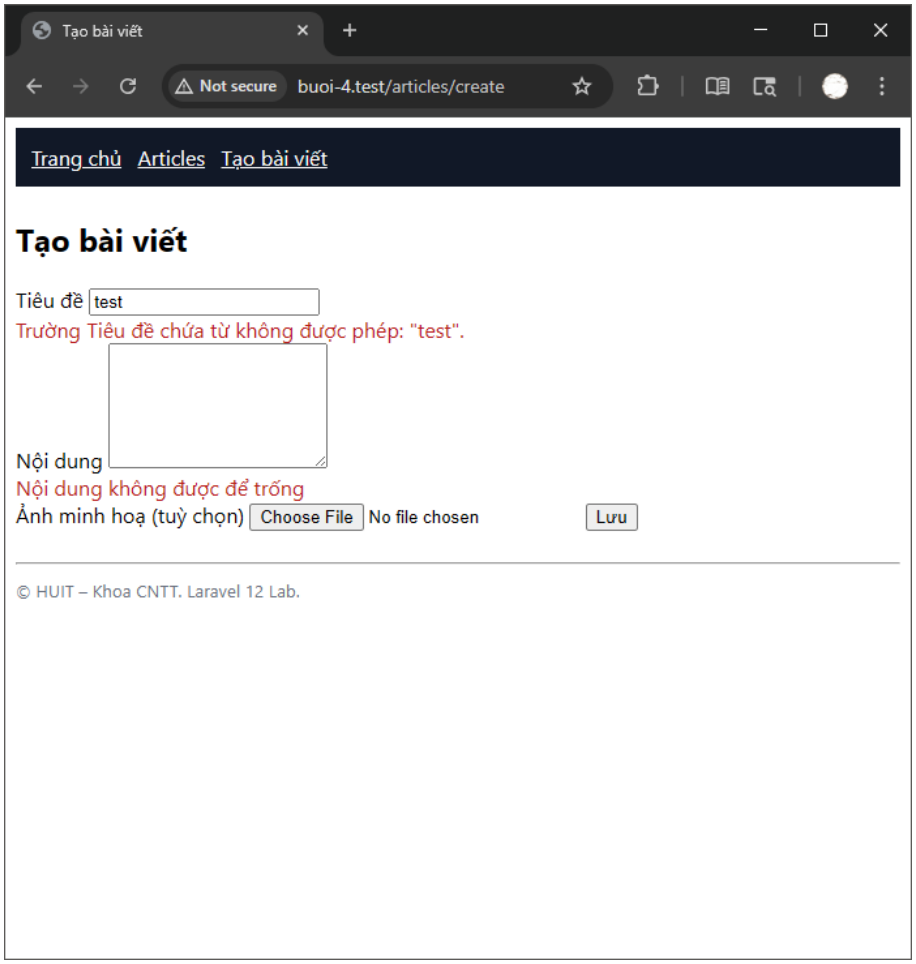
- Cập nhật Form và View hiển thị ảnh
  - Form tạo/sửa (*resources/views/articles/create.blade.php* & *edit.blade.php*):

```
<form action="{{ route('articles.store') }}" method="post"
enctype="multipart/form-data">
  @csrf

  <label>Tiêu đề</label>
  <input type="text" name="title" value="{{ old('title') }}">
  @error('title') <div style="color:#b91c1c">{{ $message }}</div> @enderror

  <label>Nội dung</label>
  <textarea name="body" rows="6">{{ old('body') }}</textarea>
  @error('body') <div style="color:#b91c1c">{{ $message }}</div> @enderror

  <label>Ảnh minh hoạ (tuỳ chọn)</label>
  <input type="file" name="image" accept=".jpg,.jpeg,.png">
  @error('image') <div style="color:#b91c1c">{{ $message }}</div> @enderror

  <button type="submit">Lưu</button>
```

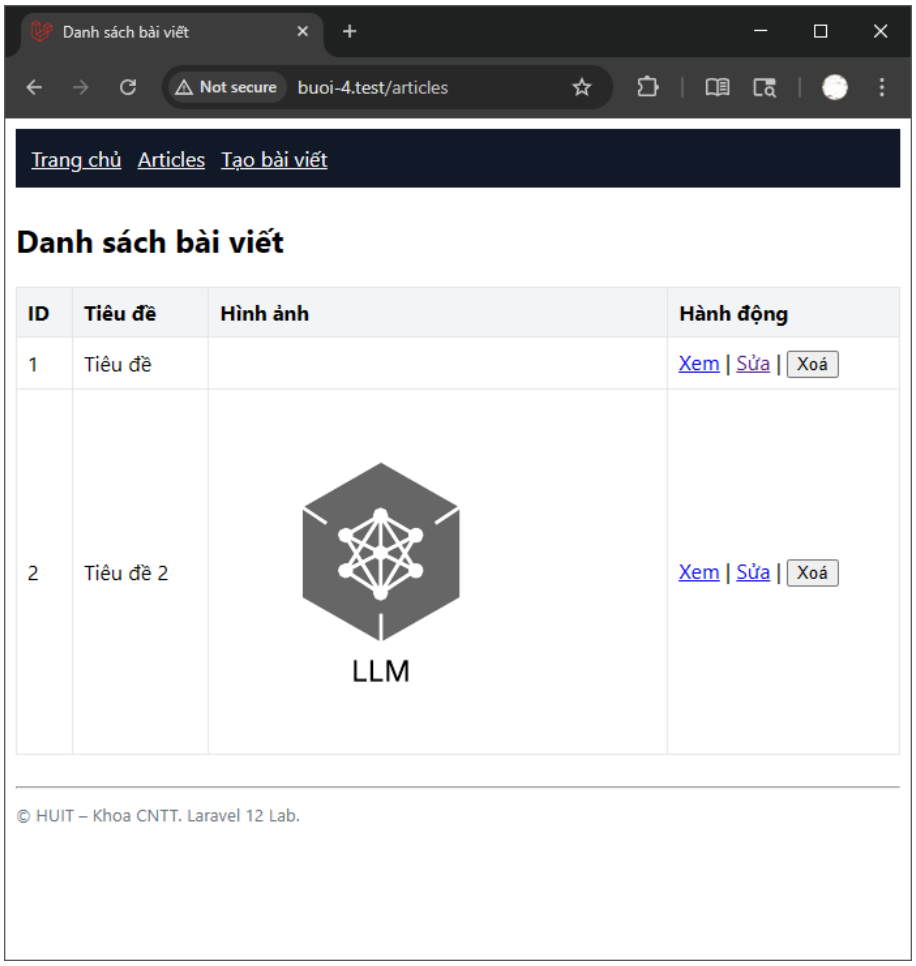
```
</form>
```

Kết quả mong đợi

- Khi tiêu đề chứa từ cấm, form báo lỗi từ Rule tuỳ chỉnh; khi hợp lệ thì chấp nhận.
- Ảnh upload đúng định dạng/kích thước; đường dẫn ảnh được lưu trong CSDL.
- Ảnh hiển thị bình thường qua `asset('storage/...')`.
- Mã nguồn gọn, validation nằm trong Form Request.



Hình 2 Kết quả bài 02 - Lỗi khi thêm mới



Hình 3 Kết quả bài 05 - Danh sách có hình ảnh

Lỗi thường gặp & khắc phục

- Validation không chạy
  - Quên type-hint Form Request trong Controller → thay tham số `$request` bằng `StoreArticleRequest`/`UpdateArticleRequest`.
- Ảnh không hiển thị
  - Chưa chạy php artisan storage:link.
  - Lưu sai disk/đường dẫn → đảm bảo dùng `store('articles','public')` và `asset('storage/'.$path)`.
- Lỗi unique khi update
  - Quên `ignore()` bản ghi hiện tại → dùng `Rule::unique(...)->ignore($article->id)`.
- File quá dung lượng/định dạng
  - Chưa khai báo max/mimes đúng hoặc vượt 2MB → điều chỉnh quy tắc, thông báo lỗi rõ ràng.
- Xoá ảnh cũ không thành công
  - Đường dẫn tồn tại nhưng sai disk → luôn kiểm tra bằng

`Storage::disk('public')->exists($article->image_path)` trước khi xoá.

Bài tập 03: Middleware (Laravel 12) & Throttle

Yêu cầu:

- Tạo middleware tùy biến CheckAdmin (chặn người không phải admin truy cập khu vực quản trị).
- Đăng ký middleware theo chuẩn Laravel 12 trong *bootstrap/app.php* với alias là admin.
- Áp dụng admin cho nhóm route quản trị (ví dụ /admin/* hoặc các route CRUD bài viết).
- Cấu hình throttle (giới hạn tần suất) cho một nhóm/route API hoặc cho route đăng nhập (minh họa).
- Minh chứng hoạt động: người không đủ quyền bị chặn; khi vượt ngưỡng throttle sẽ bị từ chối tạm thời.

Mục tiêu đạt được:

- Hiểu cơ chế middleware ở cấp hệ thống (đăng ký alias, add/remove vào group web/api, global).
- Biết cách áp dụng middleware ở cấp route và route group.
- Vận dụng rate limiting (throttle) để bảo vệ hệ thống khỏi lạm dụng.

Hướng dẫn:

- Tạo middleware `CheckAdmin`

```
php artisan make:middleware CheckAdmin
```

- Sửa file *app/Http/Middleware/CheckAdmin.php*:

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class CheckAdmin
{
    public function handle(Request $request, Closure $next)
    {
        // Ví dụ: ứng dụng có cột is_admin trong bảng users
        $user = $request->user();

        if (!$user || !$user->is_admin) {
            // Có thể redirect về trang chủ hoặc trả 403
```

```
            // return redirect('/')->with('error', 'Bạn không có quyền truy
cập khu vực quản trị.');
            abort(403, 'Bạn không có quyền truy cập khu vực quản trị.');
        }

        return $next($request);
    }
}
```

Ghi chú: nếu dự án dùng vai trò/permissions khác, thay điều kiện kiểm tra tương ứng.

- Đăng ký middleware theo chuẩn Laravel 12 trong *bootstrap/app.php*
    - Mở *bootstrap/app.php*, tìm/hoặc thêm cấu hình `withMiddleware`:

```
use App\Http\Middleware\CheckAdmin;

return Application::configure(basePath: dirname(__DIR__))
    // ...
    ->withMiddleware(function (Middleware $middleware): void {
        // 1) Đăng ký alias: dùng tên ngắn 'admin' trong route
        $middleware->alias([
            'admin' => CheckAdmin::class,
        ]);

        // 2) (Tuỳ chọn) Thao tác với nhóm 'web'/'api'
        // Ví dụ: thêm một middleware tuỳ biến vào group 'web'
        // $middleware->appendToGroup('web',
\App\Http\Middleware\AddSecurityHeaders::class);

        // 3) (Tuỳ chọn) Cấu hình ngoại lệ CSRF
        // $middleware->validateCsrfTokens(except: ['webhook/*']);
    })
    ->withExceptions(function (Exceptions $exceptions): void {
        //
    })->create();
```

Ưu tiên dùng alias để gắn nhanh vào route, đúng với chuẩn mới của Laravel 12.

- Áp dụng middleware admin cho khu vực quản trị
    - Ví dụ gom các route quản trị vào một route group có prefix admin:

```
use App\Http\Controllers\ArticleController;

Route::prefix('admin')
    ->middleware(['web', 'auth', 'admin']) // yêu cầu đăng nhập + admin
    ->group(function () {
```

```
        // Có thể tái dùng resource Articles cho khu quản trị
        Route::resource('articles', ArticleController::class);
    });
```

- Hoặc, nếu chỉ một vài route cần quyền admin, áp dụng trực tiếp:

```
Route::delete('/articles/{article}', [ArticleController::class, 'destroy'])
    ->middleware(['auth', 'admin'])
    ->name('articles.destroy');
```

Lưu ý: auth nên đặt trước admin để đảm bảo đã xác định người dùng.

– Cấu hình throttle (giới hạn tần suất)

Có 2 cách minh họa phổ biến:

- Cách A – Áp dụng throttle trên nhóm API:

```
use Illuminate\Support\Facades\Route;

Route::middleware(['throttle:60,1'])->group(function () {
    Route::get('/public-info', fn() => ['status'=>'ok']);
    // Các route API khác…
});
```

Ý nghĩa throttle:60,1: tối đa 60 yêu cầu/phút cho mỗi IP/user.

- Cách B – Áp dụng throttle cho route đăng nhập (web) để chống brute-force:

Nếu dùng Breeze/Fortify, route */login* có thể đã được giới hạn sẵn. Minh họa thêm:

```
Route::post('/login',
[\App\Http\Controllers\Auth\AuthenticatedSessionController::class, 'store'])
    ->middleware('throttle:5,1'); // chỉ cho phép 5 lần/phút
```

Ghi chú: Có thể tạo rate limiter tùy biến trong RouteServiceProvider (nâng cao). Ở phạm vi lab, dùng middleware throttle:xx,yy là đủ minh họa.

– Kiểm tra nhanh

- Kiểm tra admin:
  - Đăng nhập bằng user không phải admin → truy cập */admin/articles* → nhận 403 (hoặc redirect).
  - Đăng nhập bằng user admin → truy cập được bình thường.
- Kiểm tra throttle:
  - Gọi liên tiếp endpoint có throttle quá ngưỡng (ví dụ >60 lần/phút) → nhận phản hồi 429 Too Many Requests.

Lỗi thường gặp & khắc phục

– Route */login* không tìm thấy

- Chưa cài đặt Laravel Breeze, có thể tham khảo các bước cài đặt ở bài 4

- Middleware không hiệu lực
  - Quên đăng ký alias trong *bootstrap/app.php*.
  - Đặt sai thứ tự: thiếu auth trước admin khiến `$request->user()` rỗng → luôn 403.
- Luôn bị 403 dù là admin
  - Điều kiện kiểm tra sai (ví dụ chưa có cột is_admin, hoặc user chưa gán quyền).
  - Đăng nhập session không thành công → kiểm tra middleware web.
- Throttle không kích hoạt
  - Gắn sai middleware (không nằm trong chain của route).
  - Test quá chậm (không vượt ngưỡng) → dùng script/cURL để gọi nhanh liên tiếp.
- CSRF lỗi 419 khi test POST từ form
  - Quên @csrf trong form.
  - Đang gắn tuyến POST vào *api.php* (nhóm api mặc định stateless) nhưng test như web → đưa về web.php.

Bài tập 04: Tích hợp Authentication (Breeze) & Bảo vệ route

Yêu cầu:

- Cài Laravel Breeze (giao diện Blade) để có các trang: đăng ký, đăng nhập, quên mật khẩu, xác minh email (tùy chọn).
- Bảo vệ các ruote tạo/sửa/xóa bài viết bằng middleware auth; khách (guest) phải đăng nhập mới thao tác được.
- Hiển thị menu điều hướng khác nhau cho `@guest` và `@auth`; thêm nút Đăng xuất.
- (Tùy chọn) Tạo sẵn 01 tài khoản mẫu qua seeder để kiểm thử nhanh.

Mục tiêu đạt được

- Hiểu quy trình tích hợp Authentication tiêu chuẩn trong Laravel 12 bằng Breeze.
- Áp dụng middleware auth để bảo vệ tài nguyên; thao tác đúng luồng đăng nhập/đăng xuất.
- Tổ chức giao diện điều hướng theo trạng thái người dùng (guest vs. authenticated).

Hướng dẫn:

- Cài Breeze (Blade) và build giao diện

  Thực thi các lệnh trong thư mục dự án:

```
composer require laravel/breeze --dev
php artisan breeze:install blade
npm install
npm run build    # hoặc: npm run dev (khi phát triển)
```

```
php artisan migrate
```

Sau lệnh trên, dự án có sẵn route và view cho đăng ký/đăng nhập, quên mật khẩu, xác minh email (nếu bật).

- Bảo vệ các route quản trị/route thao tác Article

Ví dụ bảo vệ CRUD Articles (trừ xem danh sách/chi tiết):

```
// routes/web.php
use App\Http\Controllers\ArticleController;
use Illuminate\Support\Facades\Route;

// Xem danh sách & chi tiết: công khai
Route::resource('articles', ArticleController::class)-
>only(['index','show']);

// Các thao tác còn lại: yêu cầu đăng nhập
Route::middleware('auth')->group(function () {
    Route::resource('articles', ArticleController::class)-
>only(['create','store','edit','update','destroy']);
});
```

Nếu có khu vực quản trị:

```
Route::prefix('admin')
    ->middleware(['auth']) // có thể kèm 'admin' ở BT3
    ->group(function () {
        Route::resource('articles', ArticleController::class);
    });
```

- Hiển thị menu theo trạng thái người dùng & thêm nút Đăng xuất

Cập nhật *partials/nav.blade.php* (hoặc layout chung):

```
<nav style="padding:12px;background:#111827;color:white">
  <a href="{{ url('/') }}" style="color:#fff">Trang chủ</a>
  <a href="{{ route('articles.index') }}" style="color:#fff">Bài viết</a>

  @auth
    <a href="{{ route('articles.create') }}" style="color:#fff">Viết bài</a>
    <form method="POST" action="{{ route('logout') }}"
style="display:inline">
      @csrf
      <button type="submit"
style="background:none;border:none;color:#fff;cursor:pointer">Đăng
xuất</button>
    </form>
  @endauth
```

```
  @guest
    <a href="{{ route('login') }}" style="color:#fff">Đăng nhập</a>
    <a href="{{ route('register') }}" style="color:#fff">Đăng ký</a>
  @endguest
</nav>
```

Breeze đã đăng ký sẵn các route login, register, password.request, logout,…

- Seeder tạo tài khoản mẫu để kiểm thử

Tạo seeder:

```
php artisan make:seeder DemoUserSeeder
```

Sửa *database/seeders/DemoUserSeeder.php*:

```
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;
use App\Models\User;

class DemoUserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        User::updateOrCreate(
            ['email' => 'demo@huit.edu.vn'],
            [
                'name' => 'Demo User',
                'password' => Hash::make('password123'),
                // nếu có cột is_admin:
                // 'is_admin' => true,
            ]
        );
    }
}
```

Gọi trong `DatabaseSeeder`:

```
public function run(): void
{
    $this->call(DemoUserSeeder::class);
}
```

Chạy:

```
php artisan db:seed
```

- Kiểm tra luồng
    - Truy cập */articles/create* khi chưa đăng nhập → hệ thống redirect về */login*.
    - Đăng nhập (tài khoản tự đăng ký hoặc demo@huit.edu.vn / password123) → truy cập */articles/create* được phép.
    - Giao diện hiển thị Đăng xuất khi đã đăng nhập; ẩn Đăng ký/Đăng nhập.

Kết quả mong đợi

- Các route thao tác bài viết (create/store/edit/update/destroy) chỉ truy cập được sau khi đăng nhập.
- Điều hướng theo trạng thái người dùng hoạt động đúng (`@guest` vs. `@auth`).
- Có thể đăng ký/đăng nhập/đăng xuất hoàn chỉnh bằng Breeze.

Lỗi thường gặp & khắc phục

- Đăng nhập/đăng ký trả lỗi giao diện (Vite/asset)
    - Chưa chạy `npm run build` (hoặc `npm run dev`) sau khi cài Breeze.
    - Khắc phục: chạy lại build; làm mới cache trình duyệt.
- Form POST bị 419 (TokenMismatch)
    - Thiếu @csrf hoặc submit đến route *api.php* (stateless).
    - Khắc phục: đảm bảo form có @csrf và route trong web.php.
- Sau đăng nhập vẫn không vào được trang cần bảo vệ
    - Sai route name/URL redirect; kiểm tra RouteServiceProvider/middleware chain.
    - Kiểm tra session, domain, cấu hình APP_URL.
- Không tìm thấy route login/register
    - Chưa chạy php artisan breeze:install blade hoặc chưa migrate.
    - Khắc phục: cài lại Breeze, migrate DB, xem `php artisan route:list`.

Bài tập 05: Authorization bằng Policy: "Chỉ tác giả được sửa/xóa"

Yêu cầu:

- Thêm ràng buộc "chỉ tác giả (owner) được update/delete bài viết".
- Cài đặt Policy cho Article với ít nhất các phương thức: viewAny, view, create, update, delete.
- Ràng buộc quyền ở Controller (bằng `$this->authorize(...)` hoặc `authorize()`), Route (middleware `can:`), và Blade (`@can`).
- Ẩn/hiện nút Sửa/Xóa trên giao diện dựa trên quyền; truy cập trái phép phải nhận 403.

Mục tiêu đạt được:

- Hiểu mô hình Authorization theo Policy gắn với Model.

- Biết ba lớp ràng buộc quyền: Controller, Route, View.
- Thiết lập được ownership check (so sánh `article.user_id === auth()->id()`).

Hướng dẫn:

- Chuẩn bị (nếu chưa có cột user_id trên bảng articles)
  - Tạo migration thêm khóa ngoại `user_id`:

```
php artisan make:migration add_user_id_to_articles_table --table=articles
```

  - Sửa file *database/migrations/xxxx_add_user_id_to_articles_table.php*

```
public function up(): void
{
    Schema::table('articles', function (Blueprint $table) {
        $table->foreignId('user_id')->after('id')->constrained()-
>onDelete('cascade');
    });
}
```

Chạy lệnh:

```
php artisan migrate
```

Lưu ý: Khi tạo mới bài viết (store), cần gán `user_id = auth()->id()`, và điều chỉnh lại model để bổ sung trường dữ liệu mới.

- Tạo Policy cho Article

```
php artisan make:policy ArticlePolicy --model=Article
```

File tạo tại: *app/Policies/ArticlePolicy.php*.

- Cài đặt logic quyền trong Policy
  - Ví dụ triển khai owner-only cho update/delete:

```
namespace App\Policies;

use App\Models\Article;
use App\Models\User;
use Illuminate\Auth\Access\Response;

class ArticlePolicy
{
    /** Xem danh sách bài viết */
    public function viewAny(User $user): bool
    {
        return true; // công khai
    }
```

```
    /** Xem chi tiết bài viết */
    public function view(User $user, Article $article): bool
    {
        return true; // công khai
    }

    /** Tạo bài viết: yêu cầu đăng nhập */
    public function create(User $user): bool
    {
        return $user !== null;
    }

    /** Chỉ tác giả được sửa */
    public function update(User $user, Article $article): bool
    {
        return $article->user_id === $user->id;
    }

    /** Chỉ tác giả được xóa */
    public function delete(User $user, Article $article): bool
    {
        return $article->user_id === $user->id;
    }

    /**
     * Determine whether the user can restore the model.
     */
    public function restore(User $user, Article $article): bool
    {
        return false;
    }

    /**
     * Determine whether the user can permanently delete the model.
     */
    public function forceDelete(User $user, Article $article): bool
    {
        return false;
    }
}
```

Ghi chú: Laravel tự động auto-discover policy theo quy ước tên. Nếu dự án không theo quy ước, đăng ký thủ công trong *app/Providers/AuthServiceProvider.php* (mục "Policies").

- Ràng buộc quyền ở Controller
    - Trong ArticleController, thêm ủy quyền trước khi thực thi nghiệp vụ:

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use App\Http\Requests\StoreArticleRequest;
use App\Http\Requests\UpdateArticleRequest;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use App\Models\Article;

class ArticleController extends Controller
{
    use AuthorizesRequests;
    //...
    public function create()
    {
        $this->authorize('create', Article::class);
        return view('articles.create');
    }

    public function store(StoreArticleRequest $request)
    {
        $this->authorize('create', Article::class);
        //...

        Article::create($data);

        return redirect()->route('articles.index')
            ->with('success', 'Tạo bài viết thành công');
    }

    public function show($id)
    {
        $article = Article::findOrFail($id);
        return view('articles.show', compact('article'));
    }

    public function edit(Request $request, Article $article)
    {
        $this->authorize('update', $article);
        return view('articles.edit', compact('article'));
```

```
    }

    public function update(UpdateArticleRequest $request, Article $article)
    {
        $this->authorize('update', $article);
        //...
        $article->update($data);

        return redirect()->route('articles.show', $article)
            ->with('success', 'Cập nhật bài viết thành công');
    }

    public function destroy(Article  $article)
    {
        $this->authorize('delete', $article);
        $article->delete();
        return redirect()->route('articles.index')->with('success', 'Đã xóa
bài viết');
    }
}
```

- Ràng buộc quyền ở Route (bổ trợ)
  - Có thể tăng cường bảo vệ bằng middleware can::

```
use App\Http\Controllers\ArticleController;

// Công khai
Route::resource('articles', ArticleController::class)-
>only(['index','show']);

// Cần đăng nhập
Route::middleware('auth')->group(function () {
    Route::get('/articles/create', [ArticleController::class, 'create'])
        ->name('articles.create')
        ->middleware('can:create,App\Models\Article');

    Route::post('/articles', [ArticleController::class, 'store'])
        ->name('articles.store')
        ->middleware('can:create,App\Models\Article');

    Route::get('/articles/{article}/edit', [ArticleController::class,
'edit'])
        ->name('articles.edit')
        ->middleware('can:update,article');
```

```
    Route::put('/articles/{article}', [ArticleController::class, 'update'])
        ->name('articles.update')
        ->middleware('can:update,article');

    Route::delete('/articles/{article}', [ArticleController::class,
'destroy'])
        ->name('articles.destroy')
        ->middleware('can:delete,article');
});
```

Lưu ý: Ở Laravel, Route Model Binding giúp article trong middleware chính là thực thể Article cần kiểm tra.

– Ràng buộc quyền ở View (Blade)

- Ẩn/hiện nút thao tác theo quyền:

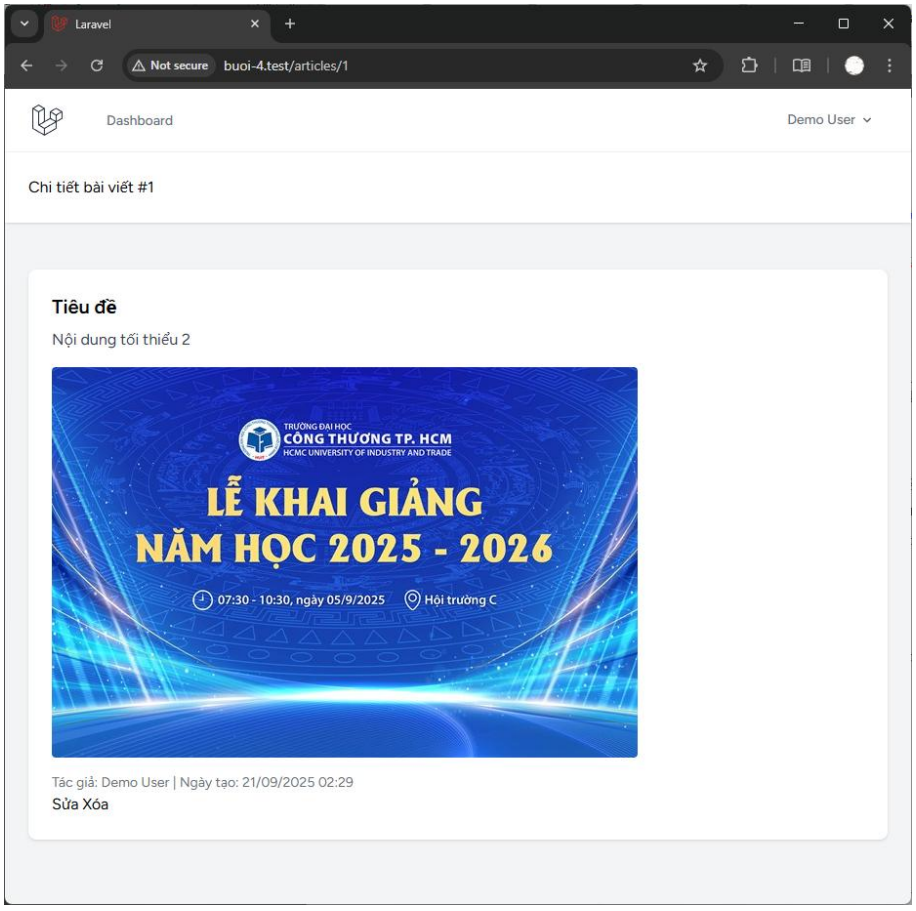
```
<div class="py-12">
    <div class="max-w-3xl mx-auto sm:px-6 lg:px-8">
        <div class="p-6 bg-white shadow sm:rounded-lg">
            <h3 class="text-xl font-bold mb-2">{{ $article->title }}</h3>
            <div class="mb-4 text-gray-700">{{ $article->body }}</div>
            @if($article->image_path)
                <div class="mb-4">
                    <img src="{{ asset('storage/' . $article->image_path) }}"
alt="Ảnh minh hoạ"
                         class="max-w-full h-auto rounded">
                </div>
            @endif
            <div class="text-sm text-gray-500">Tác giả: {{ $article->user-
>name }} | Ngày tạo:
                {{ $article->created_at->format('d/m/Y H:i') }}</div>

            @can('update', $article)
                <a href="{{ route('articles.edit', $article) }}">Sửa</a>
            @endcan

            @can('delete', $article)
                <form action="{{ route('articles.destroy', $article) }}"
method="post" style="display:inline">
                    @csrf
                    @method('DELETE')
                    <button type="submit" onclick="return confirm('Xóa bài
viết này?')">Xóa</button>
                </form>
```

```
            @endcan

            @cannot('update', $article)
                <div class="mt-4 p-4 bg-blue-100 border border-blue-300
rounded">
                    <p class="text-blue-700">Bạn không phải tác giả.</p>
                </div>
            @endcannot
        </div>
    </div>
</div>
```

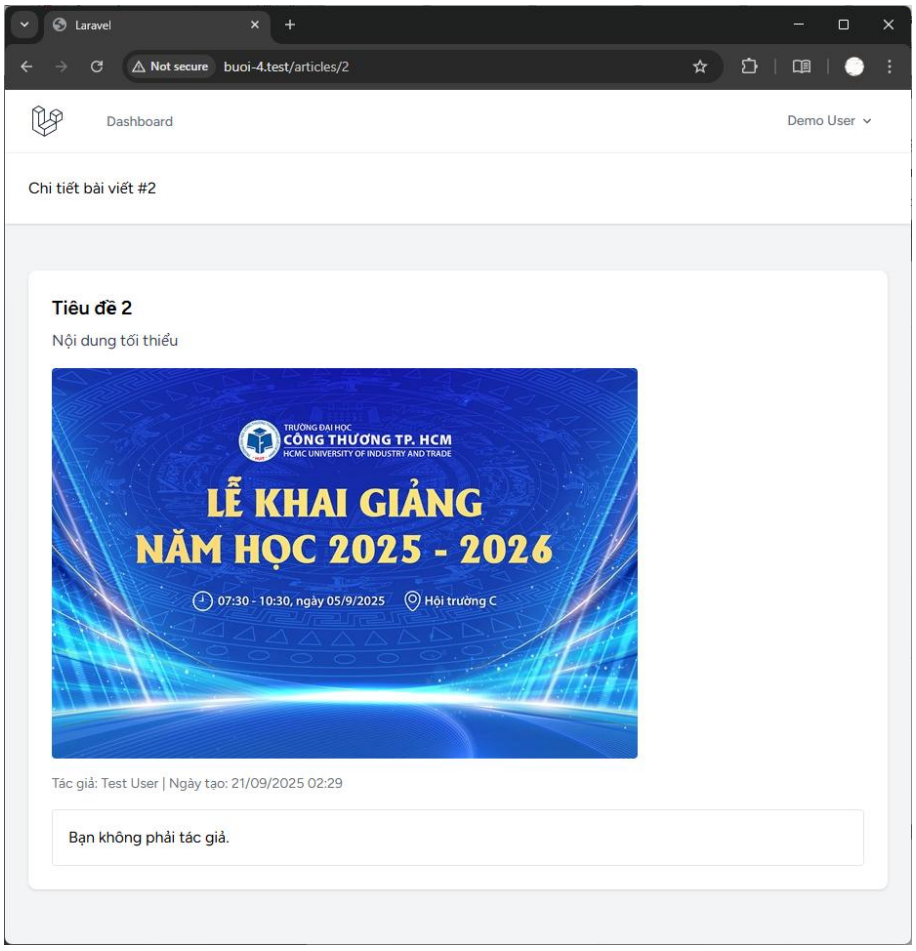
Có thể dùng thêm `@cannot('update', $article)` để hiển thị thông báo "Bạn không phải tác giả".

Kết quả mong đợi

- Controller từ chối thao tác update/delete nếu người dùng không phải tác giả → trả 403.
- Giao diện chỉ hiển thị nút Sửa/Xóa với tác giả; người khác không thấy nút này.
- Route middleware bảo vệ lớp ngoài cùng, tránh việc gọi trực tiếp endpoint.



Hình 4 Màn hình chi tiết có quyền



Hình 5 Màn hình chi tiết không quyền

Lỗi thường gặp & khắc phục

- Lúc nào cũng 403
  - `article.user_id` chưa gán khi tạo → bảo đảm `store()` gán `user_id = auth()->id()`.
  - Chưa đăng nhập mà đã truy cập tuyến yêu cầu auth.
  - Policy không được auto-discover → khai báo thủ công trong `AuthServiceProvider::$policies`.
- Nút Sửa/Xóa vẫn hiện với người không phải tác giả
  - View không dùng `@can/@cannot`; kiểm tra lại blade.
  - Truyền nhầm biến vào `@can('update', $article)`.
- Route middleware can: không nhận thực thể
  - Sai tham số: phải là can:update,article (tên biến binding) hoặc `can:create,App\Models\Article`.
  - Chưa bật Route Model Binding đúng (sai tên tham số `{article}`).
- Model không có quan hệ user (hiển thị tác giả lỗi)
  - Thêm quan hệ ở Article:

```php
public function user(){ return $this->belongsTo(User::class); }
```

3. Bài tập ôn luyện

Bài tập 06: Localization thông điệp lỗi

- **Yêu cầu:**
  - Việt hóa message/attributes Validation; tạo file ngôn ngữ `vi` cho lỗi phổ biến (`email`, `required`, `max`).

Bài tập 07: Xác minh email & đặt lại mật khẩu

- **Yêu cầu**:
  - Bật email verification và password reset (dùng mail driver log/Mailtrap); kiểm tra luồng end-to-end.

Bài tập 08: Middleware CSRF ngoại lệ (có chủ đích)

- **Yêu cầu**:
  - Cấu hình loại trừ một endpoint webhook khỏi CSRF trong *bootstrap/app.php* (chỉ demo); giải thích rủi ro bảo mật.

Bài tập 09: Gate tổng quát

- **Yêu cầu**:
  - Tạo Gate "quản trị viên" và dùng trong Blade `@can('admin')` để ẩn/hiện các mục menu quản trị.

Bài tập 10: Kiểm thử nhanh

- **Yêu cầu**:
  - Viết 1–2 Feature Test (đăng nhập thành công/thất bại; truy cập route bảo vệ trả 302→login).