ScopeTreasure Courses

LARAVEL COURSES · LESSON LV.7 · CODE SCOPE SERIES

Laravel Authentication — Login System

A lock on the app: Breeze installs ready login and register pages, passwords sleep hashed, and the auth middleware guards every student route.

Introduction

Students Manager works — but right now *anyone* who finds the address can delete a student. Real apps ask two questions at the gate: **who are you?** (authentication) and only then let you in. Building login by hand is a famous trap: passwords stored wrong, sessions handled loosely, reset flows forgotten. Laravel's answer is **Breeze** — a starter kit that installs honest, ready-made login, register and forgot-password pages in minutes, using the framework's own tested machinery.

In this lesson you will install Breeze, register a real user, watch the password sleep **hashed** in the database (never readable, not even by you), log in wrong and right, and put the Students routes behind the `auth` guard so the whole app locks when you log out.

(Roman Urdu: Login system ghar ka taala hai — khud banana risky hai, is liye Laravel ki tayyar, aazmai hui chaabi istemal karte hain.)

Key points

- **Authentication = who are you; it is not permissions.** Login proves identity. Deciding what a logged-in user may *do* (admin vs teacher) is a second question — middleware guards return to it at LV.18.
- **Breeze installs the whole gate in minutes:** `composer require laravel/breeze --dev`, then `php artisan breeze:install blade`, then build the front-end assets and migrate. Register, login, logout, forgot-password and a dashboard arrive as ordinary Blade pages you can restyle.
- **Passwords are hashed, never stored:** the users table holds a bcrypt hash like `$2y$10$...` — a one-way scramble. Even with the database in hand, nobody can read the original password; login works by scrambling the typed password the same way and comparing.
- **The navbar learns your name:** after login, Breeze's layout greets the user (`Auth::user()->name`) and offers logout. Log in with wrong details and you bounce back with one honest red line: *These credentials do not match our records.*

  (Roman Urdu: Password kabhi seedha save nahi hota — hamesha ulti seedhi scramble shakal mein, taake database chori bhi ho jaye to password na mile.)
- **The `auth` middleware is the lock:** routes inside `Route::middleware('auth')->group(...)` demand a logged-in user. Guests are redirected to `/login` automatically — no checks typed inside controllers.
- **Remember me is a cookie promise:** ticking it plants a long-life cookie so the browser re-enters without typing for weeks. Convenient on a personal laptop, dangerous on a shared one — know the trade.
- **Never hand-roll login:** not the hashing, not the session handling, not the reset tokens. The hand-built version always forgets one guard; Breeze forgets none.

Code examples

Example 1 — Installing the gate: Breeze in four commands

```
composer require laravel/breeze --dev
php artisan breeze:install blade
npm install && npm run build
php artisan migrate
```

What it does: Composer fetches Breeze. The installer publishes its controllers, routes (`routes/auth.php`) and Blade views into YOUR app — they are your files now, readable and editable. The npm step builds the CSS/JS the pages use, and migrate adds the tables auth needs (including password reset tokens).

What you will see: new pages alive at once: `/login`, `/register`, `/forgot-password` and a `/dashboard` after login — styled, working, and waiting for your app's colours.

Example 2 — Register, then peek at the password that is not there

What it does: Open `/register`, create the teacher account — name `Ali Teacher`, email `ali@school.test`, password of your choice — and submit. Then look at the users table in phpMyAdmin.

What you will see: you are logged in immediately (the navbar shows your name), and the users table holds one row whose password field reads like:

```
$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi
```

That scramble IS the password, stored the only safe way. Log out and log in with a wrong password: you bounce back to the form with the red line `These credentials do not match our records.` Log in right: the dashboard greets you by name.

Example 3 — Locking Students Manager: the auth group

```
// routes/web.php — the public door stays open...
Route::get('/', function () {
    return view('welcome');
});

// ...and everything private moves behind the lock
Route::middleware('auth')->group(function () {
    Route::resource('students', StudentController::class);
    Route::get('/dashboard', function () {
        return view('dashboard');
    })->name('dashboard');
});
```

What it does: One wrapper changes the app's security completely. Every route inside the group first passes the `auth` middleware — Laravel's built-in guard (middleware gets its own deep lesson at LV.18). Guests never reach the controller at all.

What you will see: log out, then visit `/students` directly — you land on `/login` instead, with the app remembering where you were headed. Log in, and `/students` opens at once: the lock recognised its owner.

Quick recap

- Breeze installs honest login/register/reset pages in minutes — never hand-roll authentication.
- Passwords sleep as bcrypt hashes; the original text exists nowhere in your database.
- `Route::middleware('auth')` is the lock: guests redirect to /login, members walk in.
- Login answers "who are you" — what each user may do is a later, separate guard.

Common mistakes

1. **Building login by hand "to learn"** — and shipping plain-text passwords or a forgotten reset flow. Fix: learn by READING Breeze's installed files (they are in your app, in plain sight) — build custom flows only on top of Laravel's auth, never instead of it.
2. **Protecting pages but not actions** — hiding the students link while `Route::resource` stays public means anyone can still POST a delete directly. Fix: the ROUTES go inside the auth group — the lock belongs on the door, not on the signboard.
3. **Storing any password hint in plain text** — a "password_text" column or the password inside an email ruins the hashing completely. Fix: the hash is the only copy; forgot-password sends a reset LINK, never the password itself.
4. **Testing only the happy login** — the wrong-password and logged-out paths are where apps embarrass themselves. Fix: test all three every time: wrong details, right details, and a direct visit while logged out.

Practice

1. Install Breeze on the school app, register two users, and lock the Students routes. Prove the lock three ways: logged-out visit, wrong password, right password — and screenshot the red error line for your notes.
2. Restyle one auth page (login) to match your app: change the heading and button colour in its Blade file and rebuild assets. Write one line: which file did you edit, and why was it safe to edit?

Practice quiz

1. What is the difference between authentication and permissions?

- They are the same
- Authentication answers who are you (login proves identity); what a logged-in user may do, admin vs teacher, is a second question answered by guards

- Permissions come first, authentication second
- Authentication is for APIs only

Answer: Authentication answers who are you (login proves identity); what a logged-in user may do, admin vs teacher, is a second question answered by guards

2. What does the users table hold for a password, and how does login check it?

- The plain password, compared directly
- A bcrypt hash like $2y$10$..., a one-way scramble nobody can read back; login scrambles the typed password the same way and compares
- An encrypted copy that can be decrypted
- Nothing, passwords are not stored at all

Answer: A bcrypt hash like $2y$10$..., a one-way scramble nobody can read back; login scrambles the typed password the same way and compares

3. What does Route::middleware('auth')->group(...) do to a guest?

- It shows them a blank page
- It redirects them to /login automatically; the lock belongs on the routes, not on the signboard
- It lets them in read-only
- It asks for their password on every page

Answer: It redirects them to /login automatically; the lock belongs on the routes, not on the signboard

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/laravel-authentication · More free tools and notes at https://scopetreasure.com