Implementing Password Reset in Laravel 12 Without Packages

A great user experience includes a safe way to recover accounts. In this guide, you'll build a complete **Password Reset flow in Laravel 12 — without using packages**. We'll cover the database, routes, controller, mailer, Blade UI, and important security measures like token hashing, throttling, expirations, and single-use links. Every block includes explanations so beginners won't get lost.

By the end, users will be able to request a reset link, receive an email with a secure one-time URL, and set a new password — all implemented with clean Laravel conventions.

1 – What is a Password Reset flow?

A password reset flow lets a user prove ownership of their account (via email) and set a new password even if they forgot the old one. The core idea: the app generates a **short-lived, single-use token**, sends it to the user's email, and only someone with access to that mailbox can redeem it to change the password. We'll build this end to end with secure defaults.

2 – Prerequisites

- Laravel 12 project (PHP 8.2+) with a working database connection in `.env`
- Users table with `email` and `password` columns
- Mail configured in `.env` (SMTP, Mailgun, etc.) so the app can send emails

3 – Create the password reset table

```
php artisan make:migration create_password_resets_table
```
Code language: Bash (bash)

Edit the migration to store the email, a *hashed* token (safer than plain text), and a timestamp to expire tokens.

```
//
database/migrations/xxxx_xx_xx_xxxxxx_create_password_resets_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    public function up(): void {
        Schema::create('password_resets', function (Blueprint $table)
{
            $table->string('email')->index();
            $table->string('token');             // store hashed token
here
            $table->timestamp('created_at')->nullable();
        });
    }
```

```
    public function down(): void {
        Schema::dropIfExists('password_resets');
    }
};
```
Code language: PHP (php)

We'll store a **hashed** version of the token (HMAC) so if the DB were leaked, attackers cannot use reset links. The raw token only lives in the user's email link.

```
php artisan migrate
```
Code language: Bash (bash)

4 – Configure Mail (so emails actually send)

```
# .env (example for SMTP)
MAIL_MAILER=smtp
MAIL_HOST=smtp.yourprovider.com
MAIL_PORT=587
MAIL_USERNAME=your_user
MAIL_PASSWORD=your_password
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=no-reply@your-domain.com
MAIL_FROM_NAME="Your App"Code language: PHP (php)
```

Make sure these values are correct for your provider. Incorrect mail settings are the #1 reason reset flows "don't work".

5 – Routes (guest-only flow with throttling)

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

// Show "Forgot Password" form & send reset link
Route::middleware(['guest', 'throttle:6,1'])->group(function () {
    Route::get('/forgot-password', [PasswordResetController::class,
'requestForm'])
        ->name('password.request');

    Route::post('/forgot-password', [PasswordResetController::class,
'sendLink'])
        ->name('password.email');

    // User lands here from email link (contains token)
    Route::get('/reset-password/{token}',
[PasswordResetController::class, 'resetForm'])
        ->name('password.reset');

    // Submit new password
    Route::post('/reset-password', [PasswordResetController::class,
'reset'])
        ->name('password.update');
});
```

Code language: PHP (php)

What this does: all four routes are guest-only and rate-limited (6 attempts/minute) to reduce abuse. The GET routes show forms; the POST routes perform actions.

6 – Mailable for the reset link

```
php artisan make:mail ResetPasswordMail --markdown=mail.reset-password
```

This creates a mailable and a Markdown email template. We'll pass a signed, single-use style URL (with raw token) to the email template.

```
// app/Mail/ResetPasswordMail.php
namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class ResetPasswordMail extends Mailable
{
    use Queueable, SerializesModels;

    public string $url;

    public function __construct(string $url)
    {
        $this->url = $url;
    }

    public function build()
    {
        return $this->subject('Reset your password')
            ->markdown('mail.reset-password', [
                'url' => $this->url
            ]);
    }
```

```
}
```
Code language: PHP (php)

```
<!-- resources/views/mail/reset-password.blade.php -->
@component('mail::message')
# Reset your password

Click the button below to choose a new password. If you didn’t request
this, you can ignore this email.

@component('mail::button', ['url' => $url])
Reset Password
@endcomponent

This link will expire shortly for your security.

Thanks,

{{ config('app.name') }}
@endcomponent
```
Code language: PHP (php)

How it works: the controller will generate a raw token (random string), store a hashed version in DB, and include the raw token in the URL emailed to the user. The form that receives the token will recompute the hash and compare to DB.

7 – Controller: generate, email, verify, and reset

```
// app/Http/Controllers/Auth/PasswordResetController.php
namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
```

```
use App\Mail\ResetPasswordMail;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Str;

class PasswordResetController extends Controller
{
    // 1) Show "Forgot Password" form
    public function requestForm()
    {
        return view('auth.forgot-password');
    }

    // 2) Accept email, generate token, store hashed token, email link
    public function sendLink(Request $request)
    {
        $request->validate([
            'email' => 'required|email',
        ]);

        // Always respond the same to avoid leaking if email exists
        $email = (string) $request->input('email');

        // Create raw token and hashed token
        $rawToken   = Str::random(64);
        $hashedToken = hash_hmac('sha256', $rawToken,
config('app.key'));

        // Upsert the record
        DB::table('password_resets')->updateOrInsert(
            ['email' => $email],
            ['token' => $hashedToken, 'created_at' => now()]
        );

        // Build the URL user will click (token in path, email as
```

```
query)
        $url = route('password.reset', $rawToken) . '?email=' .
urlencode($email);

        // Send email (do not reveal whether email exists)
        Mail::to($email)->send(new ResetPasswordMail($url));

        return back()->with('status', 'If we found an account with
that email, a reset link has been sent.');
    }

    // 3) Show "Reset Password" form (user clicked link)
    public function resetForm(string $token, Request $request)
    {
        $email = (string) $request->query('email', '');
        return view('auth.reset-password', compact('token', 'email'));
    }

    // 4) Validate token, update password, delete token
    public function reset(Request $request)
    {
        $data = $request->validate([
            'email' => 'required|email',
            'token' => 'required',
            'password' => 'required|confirmed|min:8',
        ]);

        // Recreate hashed token from raw token
        $hashedToken = hash_hmac('sha256', $data['token'],
config('app.key'));

        // Look up the reset row
        $row = DB::table('password_resets')
            ->where('email', $data['email'])
            ->where('token', $hashedToken)
            ->first();

        // Fail if token not found or expired (e.g., > 60 minutes old)
```

```
        if (! $row || now()->diffInMinutes($row->created_at) > 60) {
            return back()->withErrors(['email' => 'This reset link is
invalid or has expired.']);
        }

        // Update user password
        $user = User::where('email', $data['email'])->first();
        if (! $user) {
            // Generic failure message (don't reveal existence)
            return back()->withErrors(['email' => 'This reset link is
invalid or has expired.']);
        }

        $user->forceFill([
            'password' => Hash::make($data['password']),
        ])->save();

        // Invalidate token (single-use)
        DB::table('password_resets')->where('email',
$data['email'])->delete();

        // Optionally logout all sessions by rotating remember_token
        $user->setRememberToken(Str::random(60));
        $user->save();

        return redirect()->route('login')->with('status', 'Password
updated successfully. You can log in now.');
    }
}
```

Code language: PHP (php)

Highlights: we *never* reveal whether an email exists; tokens are **HMAC-hashed** in DB; links expire (60 minutes example); tokens are **single-use**; we rotate the user's remember token to invalidate existing sessions.

8 – UI: the two forms (Forgot & Reset)

```
<!-- resources/views/auth/forgot-password.blade.php -->
@extends('layouts.app')

@section('content')
<div class="container py-5">
  <h1 class="h4 mb-3">Forgot your password?</h1>

  @if (session('status'))
    <div class="alert alert-success">{{ session('status') }}</div>
  @endif

  <form method="POST" action="{{ route('password.email') }}"
class="card card-body">
    @csrf
    <label class="form-label">Email</label>
    <input class="form-control mb-2" type="email" name="email"
value="{{ old('email') }}" required autocomplete="email">
    @error('email') <div class="text-danger small">{{ $message
}}</div> @enderror

    <button class="btn btn-primary mt-2">Email me a reset
link</button>
  </form>
</div>
@endsection
Code language: PHP (php)
```

This is a simple, accessible form that asks for the email and shows success or validation messages. The status message is always neutral (doesn't leak if the email exists).

```
<!-- resources/views/auth/reset-password.blade.php -->
@extends('layouts.app')

@section('content')
<div class="container py-5">
  <h1 class="h4 mb-3">Choose a new password</h1>

  @if (session('status'))
    <div class="alert alert-success">{{ session('status') }}</div>
  @endif

  <form method="POST" action="{{ route('password.update') }}"
class="card card-body">
    @csrf
    <input type="hidden" name="token" value="{{ $token }}">

    <label class="form-label">Email</label>
    <input class="form-control mb-2" type="email" name="email"
value="{{ old('email', $email ?? '') }}" required
autocomplete="email">
    @error('email') <div class="text-danger small">{{ $message
}}</div> @enderror

    <label class="form-label">New password</label>
    <input class="form-control mb-2" type="password" name="password"
required autocomplete="new-password">
    @error('password') <div class="text-danger small">{{ $message
}}</div> @enderror

    <label class="form-label">Confirm new password</label>
    <input class="form-control mb-2" type="password"
name="password_confirmation" required autocomplete="new-password">

    <button class="btn btn-success mt-2">Update Password</button>
  </form>
</div>
@endsection
Code language: PHP (php)
```

The reset form includes the hidden `token`, email, and both password fields with confirmation. Errors appear inline so users immediately know what to fix.

9 – Extra hardening (recommended)

- **Token hashing:** we used `hash_hmac('sha256', $rawToken, app.key)` so DB never stores the raw token.
- **Short expiry:** 60 minutes is common. Tighten to 15–30 minutes for extra security.
- **Single use:** we delete the row once used.
- **Neutral messages:** always show a generic success after requesting a link to avoid email enumeration.
- **Session rotation:** after reset, rotate `remember_token` to log out other sessions.
- **Queue mails:** mark the mailable as `ShouldQueue` for reliability on busy apps.

10 – Test the whole flow (step-by-step)

1. Visit `/forgot-password`, enter your email, submit the form.
2. Check your inbox and click the “Reset Password” button.
3. You’ll land at `/reset-password/{token}?email=you@example.com`.
4. Choose a new password and submit.
5. Try logging in with the new password — the old one no longer works.

11 – Common errors & quick fixes

- **Emails aren't arriving:** verify `.env` mail settings; try a known-good SMTP; check logs for auth errors.
- **"Invalid or expired link":** token expired or doesn't match; resend and ensure the exact emailed URL is used.
- **Token works twice:** confirm you delete the row after a successful reset.
- **Users stuck logged in elsewhere:** ensure `remember_token` is rotated after password change.

Wrapping Up

You built a production-ready **Password Reset in Laravel 12** without packages: secure token storage, rate-limited endpoints, full Blade UI, and strong hygiene like single-use tokens and session rotation. This improves both security and user trust while keeping the code understandable and lightweight.

1v0 Ship v1.0 Faster

What’s Next

- How to Build Email Verification in Laravel 12 (Step by Step) — confirm real users before granting access.
- Implementing Two-Factor Authentication in Laravel — add an extra layer of login security.
- How to Restrict Page Access by Role in Laravel 12 — protect admin-only areas.