How to Add GitHub Login to Laravel 12 with Socialite

When building applications for developers, a top UX win is to let users sign in with their existing **GitHub** account. It reduces friction, avoids extra passwords, and signals trust. In this guide, you'll add **GitHub Login to a Laravel 12 app using Socialite** — with clear steps, fully explained code, and a simple UI button.

We'll install Socialite, create a GitHub OAuth app, wire up environment variables and services config, add routes and a controller, update the users table, and place a polished "Login with GitHub" button. Optional sections cover requesting private emails, linking/unlinking GitHub from a user profile, and common error fixes.

1 – Install Socialite

Socialite is Laravel's official OAuth client for providers such as GitHub, Google, and Facebook.

```
composer require laravel/socialiteCode language: Bash (bash)
```

Laravel 12 uses package auto-discovery, so you don't normally need to register anything. If auto-discovery is disabled, add the provider and alias manually:

```
// config/app.php (only if NOT auto-discovered)
'providers' => [
    // ...
    Laravel\Socialite\SocialiteServiceProvider::class,
],

'aliases' => [
    // ...
    'Socialite' => Laravel\Socialite\Facades\Socialite::class,
```

```
],Code language: PHP (php)
```

2 – Create GitHub OAuth App

Register your app with GitHub to obtain a Client ID and Client Secret:

1. Open GitHub → Settings → Developer settings → OAuth Apps.
2. Click **New OAuth App**.
3. Set:
    - **Homepage URL:** e.g. `https://your-domain.com`
    - **Authorization callback URL:** `https://your-domain.com/auth/github/callback`
4. Save and copy your **Client ID** and **Client Secret**.

For local development, use:

```
http://localhost:8000/auth/github/callbackCode language: JavaScript (javascript)
```

Callback URL must match exactly (protocol, domain, path) or GitHub will reject the redirect.

3 – Configure Laravel (.env & services.php)

Add credentials to your environment file so keys never live in source control:

```
# .env
```

```
GITHUB_CLIENT_ID=your_client_id_here
GITHUB_CLIENT_SECRET=your_client_secret_here
GITHUB_REDIRECT_URI=${APP_URL}/auth/github/callback

# Ensure these are correct for sessions/cookies:
APP_URL=http://localhost:8000
# SESSION_DOMAIN=.your-domain.com   # if you use subdomainsCode language: Bash (bash)
```

Tell Laravel where to read these values in `config/services.php`:

```
// config/services.php
'github' => [
    'client_id'     => env('GITHUB_CLIENT_ID'),
    'client_secret' => env('GITHUB_CLIENT_SECRET'),
    'redirect'      => env('GITHUB_REDIRECT_URI'),
],Code language: PHP (php)
```

Laravel will now use these secrets at runtime via `config('services.github.*')`.

4 – Routes

We'll add two guest-only routes: one to redirect the user to GitHub; one to handle the callback:

```
// routes/web.php
use App\Http\Controllers\Auth\GitHubController;

Route::middleware('guest')->group(function () {
    Route::get('/auth/github/redirect', [GitHubController::class, 'redirect'])
        ->name('github.redirect');
```

```
    Route::get('/auth/github/callback', [GitHubController::class,
'callback'])
        ->name('github.callback');
});Code language: PHP (php)
```

`/auth/github/redirect` sends users to GitHub's consent screen. After approval, GitHub redirects them to `/auth/github/callback` with a code Socialite exchanges for profile data.

5 – Users Table: store github_id (and avatar)

We'll store the GitHub user ID and optional avatar URL so returning users can log in seamlessly.

```
php artisan make:migration add_github_columns_to_users_table --
table=usersCode language: Bash (bash)
```

```
//
database/migrations/xxxx_xx_xx_xxxxxx_add_github_columns_to_users_tabl
e.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::table('users', function (Blueprint $table) {
            $table->string('github_id')->nullable()->index();
            $table->string('avatar')->nullable();
        });
    }
```

```
    public function down(): void {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn(['github_id','avatar']);
        });
    }
};Code language: PHP (php)
```

```
php artisan migrateCode language: Bash (bash)
```

This keeps a durable link between your local user and their GitHub identity.

6 – GitHub Controller

Create a controller to handle the redirect and callback. It logs in existing users or creates a new account as needed.

```
php artisan make:controller Auth/GitHubControllerCode language: Bash (bash)
```

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

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Str;
use Laravel\Socialite\Facades\Socialite;

class GitHubController extends Controller
{
    // Step 1: send user to GitHub
    public function redirect()
    {
        // Request public profile + email (private emails require
```

```
'user:email' scope – see optional section)
        return Socialite::driver('github')->redirect();
    }

    // Step 2: handle callback
    public function callback()
    {
        $git = Socialite::driver('github')->user();

        $email   = $git->getEmail();            // may be null if
hidden
        $gitId   = $git->getId();
        $name    = $git->getName() ?: $git->getNickname() ?: 'GitHub
User';
        $avatar  = $git->getAvatar();

        // A) Already linked
        $user = User::where('github_id', $gitId)->first();

        // B) Not linked but email matches an existing account – link
it
        if (! $user && $email) {
            $user = User::where('email', $email)->first();
            if ($user) {
                $user->forceFill([
                    'github_id' => $gitId,
                    'avatar'    => $user->avatar ?: $avatar,
                ])->save();
            }
        }

        // C) No match – create a new local account
        if (! $user) {
            $user = User::create([
                'name'       => $name,
                'email'      => $email, // can be null if GitHub
doesn't provide it
                'password'   => bcrypt(Str::random(32)),
```

```
                'github_id'  => $gitId,
                'avatar'     => $avatar,
            ]);
        }

        Auth::login($user, remember: true);

        return redirect()->intended('/');
    }
}Code language: PHP (php)
```

This flow supports three scenarios: previously linked user, same-email user (auto-link), or brand-new user. You can swap “auto-link” with a confirmation screen if your policy requires explicit consent.

7 – UI: “Login with GitHub” Button

Add a clear call-to-action on your login page:

```
<a href="{{ route('github.redirect') }}" class="btn btn-dark w-100">
  <i class="bi bi-github me-2"></i> Login with GitHub
</a>Code language: HTML, XML (xml)
```

This simply hits the redirect route, takes the user to GitHub, and returns them to your app signed in.

8 – Optional: Request private emails (user:email scope)

Some users hide their email on GitHub. To retrieve private emails, request the `user:email` scope and use `stateless()` only if you understand the session tradeoffs.

```
// Redirect with scope
return Socialite::driver('github')
    ->scopes(['user:email'])
    ->redirect();

// Later in callback, iterate emails if primary is missing:
$git = Socialite::driver('github')->user();
$email = $git->getEmail();

if (! $email && isset($git->user['email'])) {
    $email = $git->user['email'];
}

// Some SDKs expose emails array under $git->user['emails']Code language: PHP (php)
```

Always handle the case where email remains `null` (e.g., prompt the user to add an email after first login).

9 – Optional: Link/Unlink GitHub from Account Settings

Let signed-in users connect or disconnect GitHub later. Protect routes with `auth` middleware and attach the `github_id` to the current user.

```
// routes/web.php (link & disconnect)
Route::middleware('auth')->group(function () {
    Route::get('/account/github/connect', [GitHubController::class,
```

```
'redirect'])
        ->name('github.connect');

    Route::post('/account/github/disconnect', function () {
        auth()->user()->forceFill(['github_id' => null])->save();
        return back()->with('status', 'GitHub disconnected.');
    })->name('github.disconnect');
});Code language: PHP (php)
```

In the callback, if `auth()->check()` is true, attach the `github_id` to the current user instead of logging in a different account. This prevents account takeovers.

10 – Common Issues & Fixes

- **Redirect URI mismatch:** Ensure the callback in GitHub settings equals `GITHUB_REDIRECT_URI` exactly (protocol/host/path).
- **Invalid state / CSRF:** Don't use `stateless()` unless building a pure API. Make sure sessions/cookies work (correct `APP_URL`, `SESSION_DOMAIN`).
- **Email is null:** Request `user:email` scope and still handle `null` gracefully by prompting for an email post-login.
- **Subdomain login issues:** Use a shared cookie domain, e.g., `SESSION_DOMAIN=.your-domain.com`.

1v0 Ship v1.0 Faster

Wrapping Up

You added a smooth **“Login with GitHub”** to Laravel 12 using Socialite: installed the package, created OAuth credentials, configured services, added routes and a controller, updated your users table, and shipped a clean UI button. Optional sections showed how to request private emails and how to let users link/unlink GitHub from their profile — production-friendly polish your users will appreciate.

What’s Next

- How to Add Google Login to a Laravel 12 App with Socialite — offer multiple social providers.
- Laravel Spatie Permissions: Step-by-Step Setup — assign roles/permissions after login.
- Implementing Passwordless Authentication in Laravel 12 — reduce password friction entirely.