

[How to Add Google Login to a Laravel 12 App with Socialite](#)

Every time I built a new Laravel project, the request came up:

“Can we let users sign in with Google?”

At first, I’d copy snippets from old projects, tweak migrations, and hope nothing broke. Eventually, I streamlined the process — and in this post I’ll walk you through the clean, Laravel 12-friendly way to add Google login with Socialite.

Google login removes password friction and boosts conversions. In Laravel, the easiest, most secure way to add it is with **Socialite**.

What you’ll build: a “Continue with Google” button that redirects users to Google, returns with their profile, and logs them in (or signs them up) safely.

1 - Install Socialite

```
composer require laravel/socialiteCode language: JavaScript (javascript)
```

Register the facade if you don’t use package auto-discovery (most apps don’t need this):

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

Code language: PHP (php)

2 - Create Google OAuth credentials

1. Go to **Google Cloud Console** → **APIs & Services** →
2. **Credentials**. Create **OAuth 2.0 Client ID** (type: *Web application*).
3. Add an **Authorized redirect URI** (must match exactly):

`https://your-domain.com/auth/google/callback` language: JavaScript
(javascript)

for local dev

`http://localhost:8000/auth/google/callback` language: JavaScript
(javascript)

Copy your **Client ID** and **Client Secret**.

3 - Configure Laravel

.env

```
GOOGLE_CLIENT_ID=your_id_here
GOOGLE_CLIENT_SECRET=your_secret_here
GOOGLE_REDIRECT_URI=${APP_URL}/auth/google/callback
```

```
# Make sure APP_URL and session domain are correct for your env:
APP_URL=http://localhost:8000
# SESSION_DOMAIN=.your-domain.com      # if using subdomains like
demo.your-domain.com
Code language: PHP (php)
```

config/services.php

```
'google' => [
    'client_id'      => env('GOOGLE_CLIENT_ID'),
    'client_secret' => env('GOOGLE_CLIENT_SECRET'),
    'redirect'       => env('GOOGLE_REDIRECT_URI'),
],
Code language: PHP (php)
```

4 - Add routes

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

Route::middleware('guest')->group(function () {
    Route::get('/auth/google/redirect', [GoogleController::class,
    'redirect'])
        ->name('google.redirect');
    Route::get('/auth/google/callback', [GoogleController::class,
    'callback'])
        ->name('google.callback');
});

// (Optional) If you'll allow linking a Google account for logged-in
```

```
users
Route::middleware('auth')->group(function () {
    Route::post('/account/google/disconnect',
[GoogleController::class, 'disconnect'])
        ->name('google.disconnect');
});
Code language: PHP (php)
```

5 - Prepare your users table (store google_id, avatar, etc.)

```
php artisan make:migration add_google_columns_to_users_tableCode
language: CSS (css)

//
database/migrations/xxxx_xx_xx_xxxxxx_add_google_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('google_id')->nullable()->index();
            $table->string('avatar')->nullable();
            $table->json('social_meta')->nullable(); // optional for
raw payload
        });
    }
}
```

```
public function down(): void {
    Schema::table('users', function (Blueprint $table) {
        $table->dropColumn(['google_id', 'avatar',
'social_meta']);
    });
}
};
```

Code language: PHP (php)

php artisan migrate

6 - Build the controller

```
// app/Http/Controllers/Auth/GoogleController.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 GoogleController extends Controller
{
    // Step 1: send user to Google
    public function redirect()
    {
        // Request minimal scopes; OpenID gives you a stable sub (id)
        return Socialite::driver('google')
            ->scopes(['openid', 'profile', 'email'])
```

```
        ->redirect();
    }

    // Step 2: handle Google callback
    public function callback()
    {
        // Use stateless() only if you know session/state won't
        persist (e.g., API)
        $googleUser = Socialite::driver('google')->user();

        // Defensive: Google should always provide email when using
        'email' scope
        $email = $googleUser->getEmail();
        $googleId = $googleUser->getId();
        $name = $googleUser->getName() ?: Str::before($email, '@');
        $avatar = $googleUser->getAvatar();

        // 1) If a user has already linked Google, log them in
        $user = User::where('google_id', $googleId)->first();

        if (! $user && $email) {
            // 2) If email exists but not linked yet, decide your
            policy:
            // Option A: link automatically (safer if email is
            verified by Google)
            // Option B: ask user to confirm linking via email step
            $user = User::where('email', $email)->first();
            if ($user) {
                // Link existing account
                $user->forceFill([
                    'google_id' => $googleId,
                    'avatar' => $user->avatar ?: $avatar,
                    'social_meta' => json_encode(['google' =>
                    $googleUser->user]),
                ])->save();
            }
        }
    }
}
```

```
        if (! $user) {
            // 3) Otherwise create a new user account
            $user = User::create([
                'name'          => $name,
                'email'         => $email,                // may be null if
Google doesn't return it
                'password'     => bcrypt(Str::random(32)), // random
placeholder
                'google_id'    => $googleId,
                'avatar'       => $avatar,
                'social_meta' => json_encode(['google' =>
$googleUser->user]),
                ]);

            // (Optional) assign default role, seed profile, etc.
            // $user->assignRole('User');
        }

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

        return redirect()->intended('/'); // or your dashboard route
    }

    // Optional disconnect (keep local password login)
    public function disconnect()
    {
        $user = auth()->user();
        $user->forceFill(['google_id' => null])->save();

        return back()->with('status', 'Google account disconnected.');
```

Code language: PHP (php)

Security note: If you *don't* want automatic linking when emails match, replace that part with a “confirm link” screen and send a verification email to the address first.

7 - Add the “Continue with Google” button

```
<a href="{{ route('google.redirect') }}" class="btn btn-outline-dark w-100">  
    <i class="bi bi-google me-2"></i> Continue with Google  
</a>
```

Code language: HTML, XML (xml)

8 - Common Errors

Redirect URI mismatch

Ensure the URI in Google Console matches your `.env` `GOOGLE_REDIRECT_URI` **exactly** (protocol, domain, path).

Invalid state / CSRF

Make sure sessions work (correct `APP_URL`, `SESSION_DOMAIN`, cookies not blocked). Avoid `stateless()` unless you know why.

Email is null

Request email scope and ensure the Google account actually has an email. If missing, consider prompting the user to add one after login.

Subdomains (demo.your-domain.com)

Use a shared cookie domain (e.g., `SESSION_DOMAIN=.your-domain.com`) if you log in on one subdomain and return to another.

9 - “Link Google” from account settings

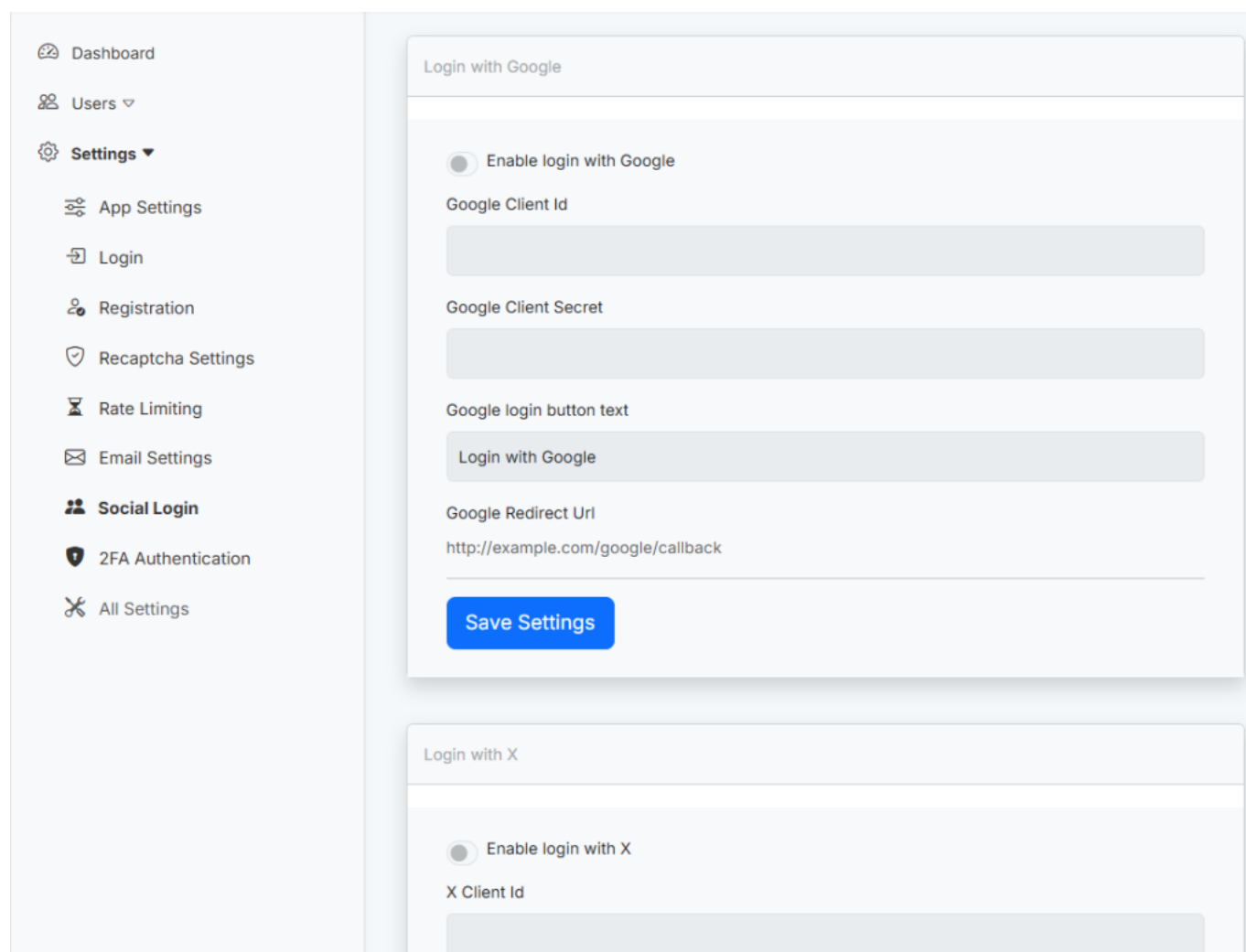
Allow logged-in users to connect their Google account later:

```
Route::middleware('auth')->get('/account/google/connect', function() {  
    return Socialite::driver('google')->redirect();  
})->name('google.connect');  
Code language: PHP (php)
```

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

Grab a production-ready implementation

If you don't want to wire up all the edge cases (UI, roles, permissions, settings toggles, demos), **Grab a production-ready implementation:**



Laravel Authentication & User Management Kit

Production-ready auth (Fortify + Socialite), roles/permissions (Spatie), settings UI, polished Bootstrap frontend, and full docs — so you can launch in minutes, not weeks.

[Learn more](#)