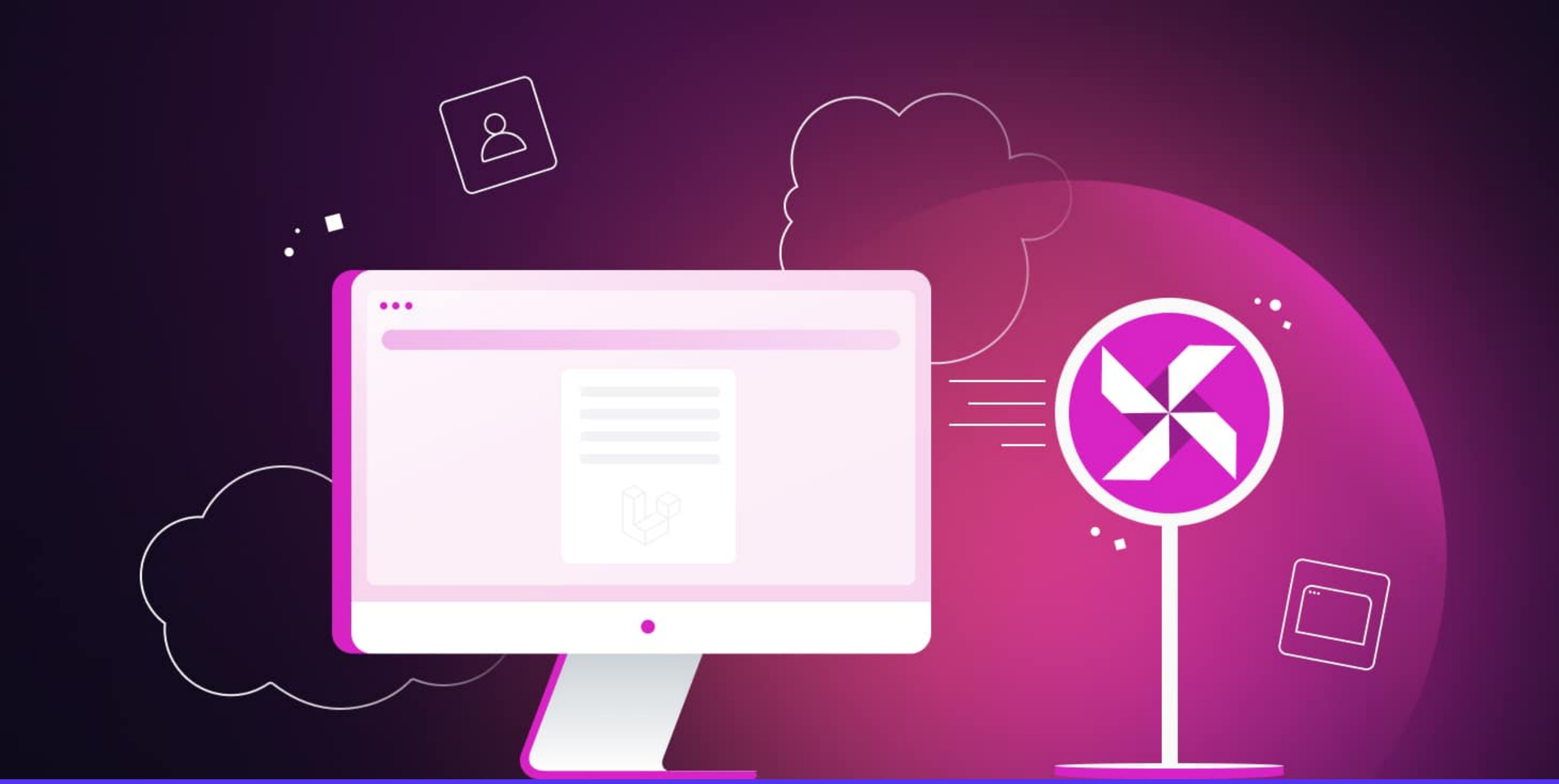
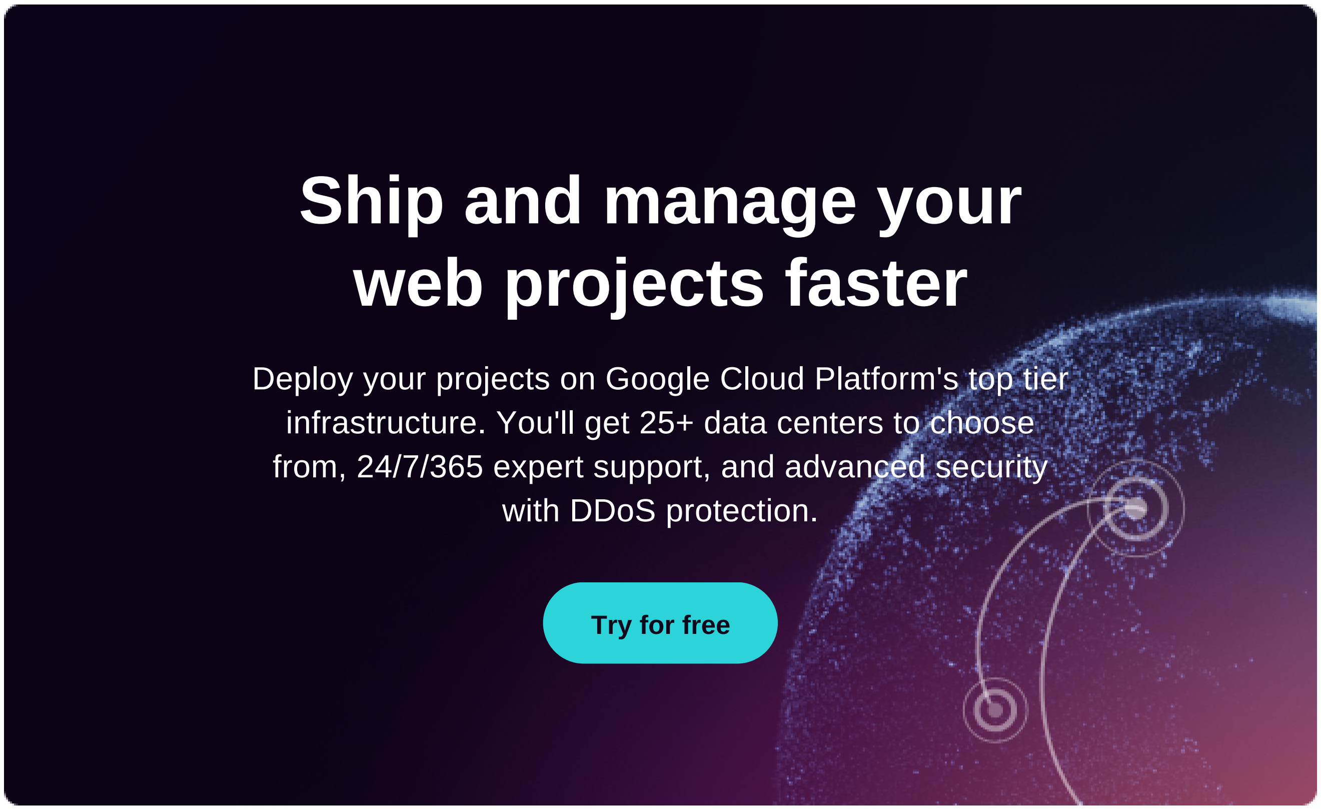


Authentication in Laravel Using Breeze

Kinsta

Downloaded on: 10 November 2023


Ship and manage your web projects faster

Deploy your projects on Google Cloud Platform's top tier infrastructure. You'll get 25+ data centers to choose from, 24/7/365 expert support, and advanced security with DDoS protection.

Try for free

This article will dive into the features of Laravel Breeze, compare it to other Laravel starter kits, and walk you through the installation process. Additionally, we'll explore generated files, customize the registration flow, and tweak the UI (user interface) to suit your project's needs.

What Is Laravel Breeze

Laravel Breeze is an authentication scaffolding package for Laravel. Using it you can have a fully working login and registration system in minutes. It supports Blade, Vue, and React and also has an API version.

The main features of Laravel Breeze are:

- Login
- Registration
- Password reset
- Email verification
- Profile page, with editing

A commonly asked question can be when to choose Breeze and when to use other Laravel authentication packages.

There are two similar packages in the Laravel ecosystem which can be confusing if you are new to this space.

The first one is Laravel Fortify which is a headless authentication backend, making it ideal for building custom authentication systems without a pre-built UI.

Choose Fortify if you have very custom UI needs or if you are only responsible for the backend of the authentication.

The other package is Laravel Jetstream which offers a more advanced starting point for Laravel applications, including features like two-factor authentication and team management.

In contrast, Laravel Breeze is best suited for developers looking for a simple yet customizable authentication scaffold with support for various frontend frameworks and minimal overhead.

Installing Laravel Breeze to a Fresh Laravel Project

To keep it simple, assume we already created a new Laravel project, if you need help with it you can follow our guide to setup a new Laravel application at Kinsta.

After that, we need to install Laravel Breeze with the following command:

```
composer require laravel/breeze --dev
```

In this tutorial, we will use Blade which is the default templating engine for Laravel. To start the scaffolding run these commands:

```
php artisan breeze:install blade

php artisan migrate
npm install
npm run dev
```

Laravel Breeze also has Vue / React / custom API versions, to use them you just need to put a flag in the command.

For Vue run:

```
php artisan breeze:install vue
```

For React run

```
php artisan breeze:install react
```

For custom API run

```
php artisan breeze:install api
```

After installing Laravel Breeze, you’ll notice that several files have been generated in your project directory. These files include routes, controllers, and views that handle authentication, password reset, and email verification. You can explore these files and customize them to fit your application requirements.

How To Customize the UI

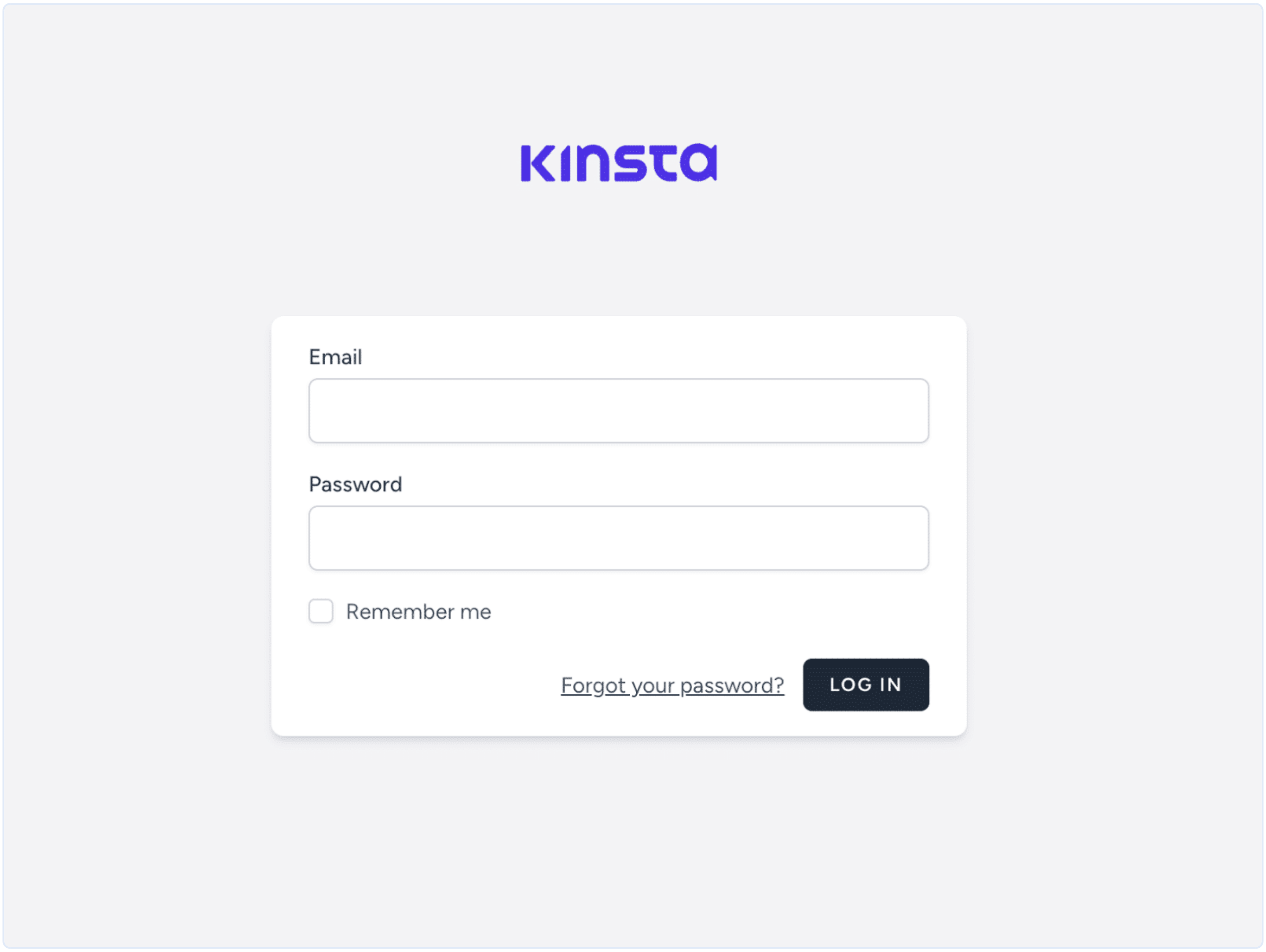
Laravel Breeze uses TailwindCSS under the hood, to customize the UI we can use any Tailwind utility class.

You can customize every part of the UI by editing the view files in the `resources/views/auth`; folder, some part of the UI is organized into Blade components, you can find these in the `resources/views/components` folder.

Changing the Laravel Logo to Our Organization Logo

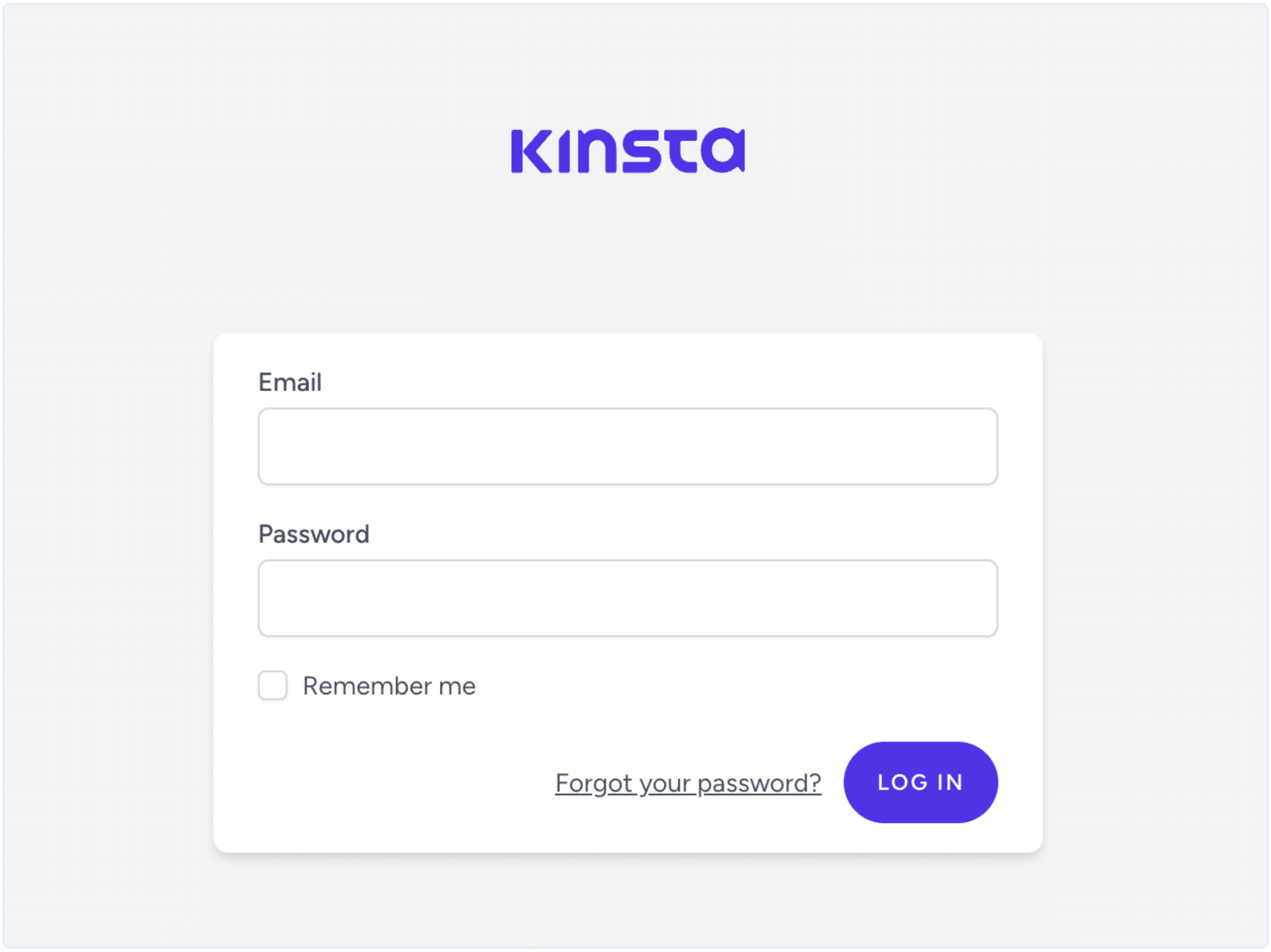
Laravel Breeze uses Blade components to organize codes used multiple times. So, for example, here’s how you can change the logo in the `resources/views/components/application-blade.php` file.

Open the file and replace the current content with your svg file.



— Changing the Color of the Primary Button

Open the `resources/views/components/primary-button.blade.php` file. You can make any modification here, like customizing the button of your login page to your brand's color.

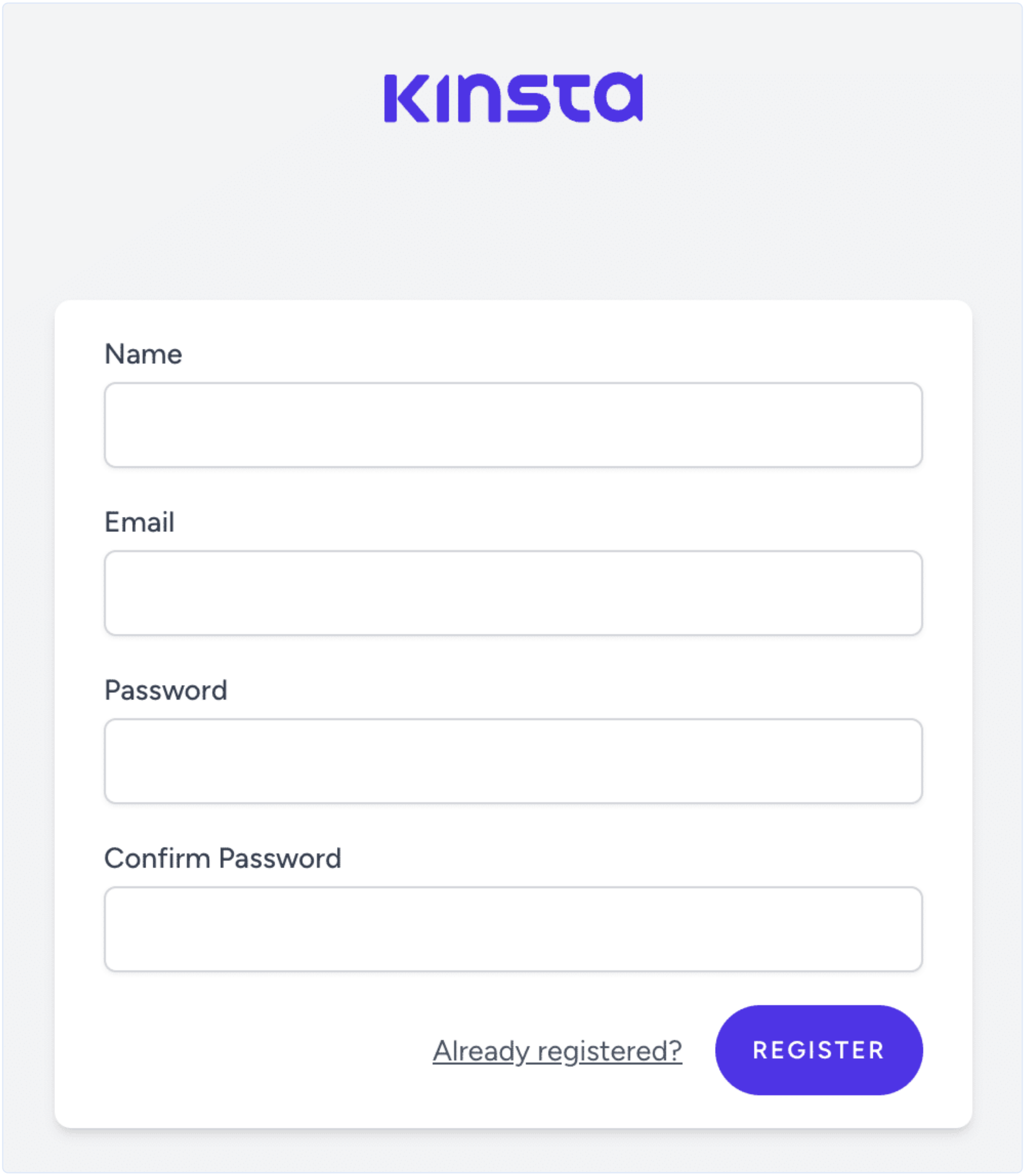


— The primary button changed to the brand color

How To Customize the Registration Flow

The Laravel Breeze registration page comes with 4 predefined fields:

1. Name
2. Email
3. Password
4. Password confirmation



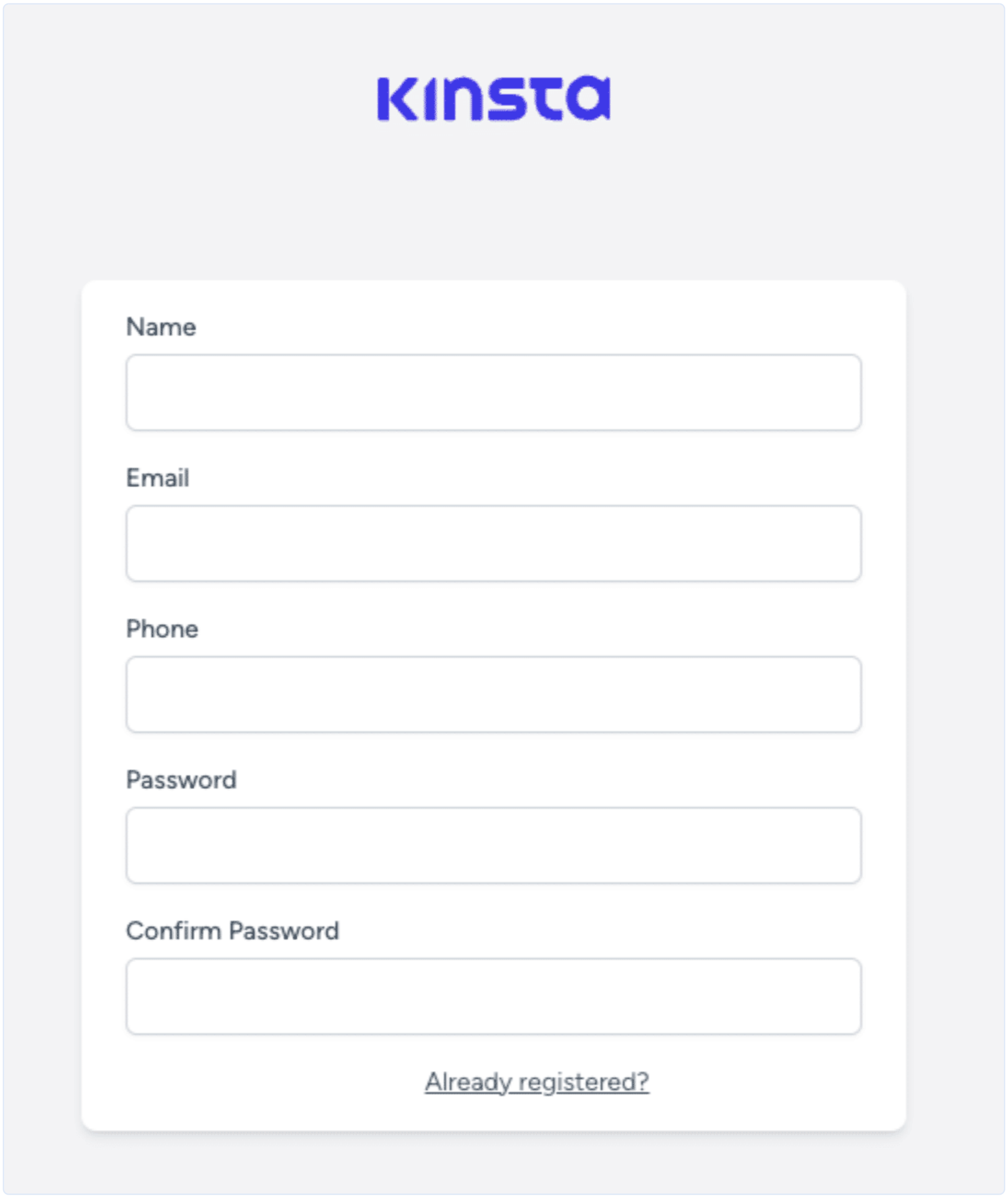
— Registration page predefined fields

To extend the fields we’d like our registration form to feature, we need to open the `resources/views/auth/register.blade.php` file.

To continue with our example, we will make a phone field after the email field. To make this happen, add the following code after the email field:

```
<div class="mt-4">
   <x-input-label for="phone" :value="__('Phone')" />
   <x-text-input id="phone" class="block mt-1 w-full" type="text" name="pl
   <x-input-error :messages="$errors->get('phone')" class="mt-2" />
</div>
```

The phone field is now visible in the registration form.


kinsta
Name
Email
Phone
Password
Confirm Password
Already registered?

— Phone field added

Modifying the Backend to Store the New Phone Field

We now need to handle the new data in the backend. These require three steps: first, create and run a new migration, then add logic to the controller to store the data, and finally, add `phone` to the fillable properties in the `User` model.

Create a new migration that will add a phone field to our `users` table.

```
php artisan make:migration add_phone_field_to_users_table
```

Open the created file and add a string field called ‘phone’:

```
Schema::table('users', function (Blueprint $table) {
   $table->string('phone')->nullable();
});
```

After that run the migration:

```
php artisan migrate
```

To store the phone field we need to modify the `RegisteredUserController.php`, in the `store` method make these modifications:

```
$request->validate([
   'name' => ['required', 'string', 'max:255'],
   'email' => ['required', 'string', 'email', 'max:255', 'unique:'.User::c
   ‘phone’ => [‘required’, ‘string’, ‘max:255’],
   'password' => ['required', 'confirmed', Rules\Password::defaults()],
]);

$user = User::create([
   'name' => $request->name,
   'email' => $request->email,
   ‘phone’ => $request->phone,
   'password' => Hash::make($request->password),
]);
```

Don’t forget to add the `phone` field to the fillable properties in the User model.

```
protected $fillable = [
   'name',
   'email',
   'phone',
   'password',
];
```

That’s it, now we have the modified registration form!

How To Enable Email Verification

Email verification is the process of checking and authenticating emails that users have been provided in the registration form.

To enable this feature we need to implement `MustVerifyEmail` interface in our User model.

```
use Illuminate\Contracts\Auth\MustVerifyEmail;
…

class User extends Authenticatable implements MustVerifyEmail
{
…
}
```

After that, an email will be sent out when a user registers with a link to verify their email.

However, we still need to add a middleware to our routes where we want to restrict access to unverified users.

We will create a new route called ‘only-verified’ and we will add ‘auth’ and ‘verified’ middleware. The auth middleware prevents access to guests and the verified middleware checks whether the user has verified their email.

Here is an example:

```
Route::get('/only-verified', function () {
   return view('only-verified');
```

```
    })->middleware(['auth', 'verified']);
```

Summary

Laravel Breeze is a great tool for quickly setting up an authentication system for your Laravel project.

With its simple yet customizable scaffolding, you can focus on building your app without worrying about the authentication process.

If you are looking for a place to host your new Laravel application, check out our Laravel hosting solution with its powerful features that make app deployment and management quick and easy.