

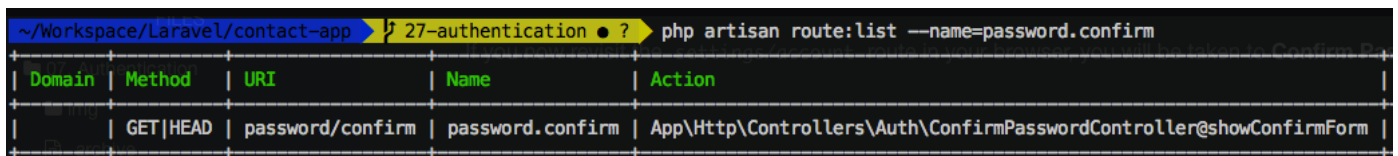
Password Confirmation

In version 6.x Laravel added new feature in the Authentication System to force user to confirm their password before accessing a certain page in your application. For example, you may require this before the user perform any payment or modifies their account settings within the application.

In you open the `app/Http/Kernel.php` you will find in the `routeMiddleware` array the `password.confirm` middleware has been defined.

```
protected $routeMiddleware = [  
    // ....  
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,  
    // ...  
];
```

If you apply this middleware in your resource, the authenticated user that try to access the resource will be redirected to **Confirm Password** page. If you take a look at your routes, you'll find a `password.confirm` route which point to `Controller/Auth/ConfirmPasswordController@showConfirmForm`.



Domain	Method	URI	Name	Action
	GET HEAD	password/confirm	password.confirm	App\Http\Controllers\Auth\ConfirmPasswordController@showConfirmForm

Now let's see how to implement the `password.confirm` middleware in our application.

1. Create a new Controller

Let's create a new controller by running this command in our terminal.

```
php artisan make:controller Settings\\AccountController
```

This command will generate a brand new controller called `AccountController` inside the `Settings` folder. I did this because I have a plan to add more settings in the future such as **Profile, Import & Export** that you can find in the Application template that I have provided in the course resource.

CONTACT APP Companies Contacts

LoginRegisterJohn Doe ▾

Settings

Profile

Account

Import & Export

Edit Profile

First Name

×

Please choose a username.

Last Name

Company

Bio

Profile picture

150 x 150

Select image

Now, let's open up the `AccountController`, then write some code like this.

```
class AccountController extends Controller
{
    public function index()
    {
        return "<h1>Account Settings</h1>";
    }
}
```

For now I only return a string in the `index` method. In other lesson of this course we will come back here and build the complete feature.

2. Define a route for the AccountController

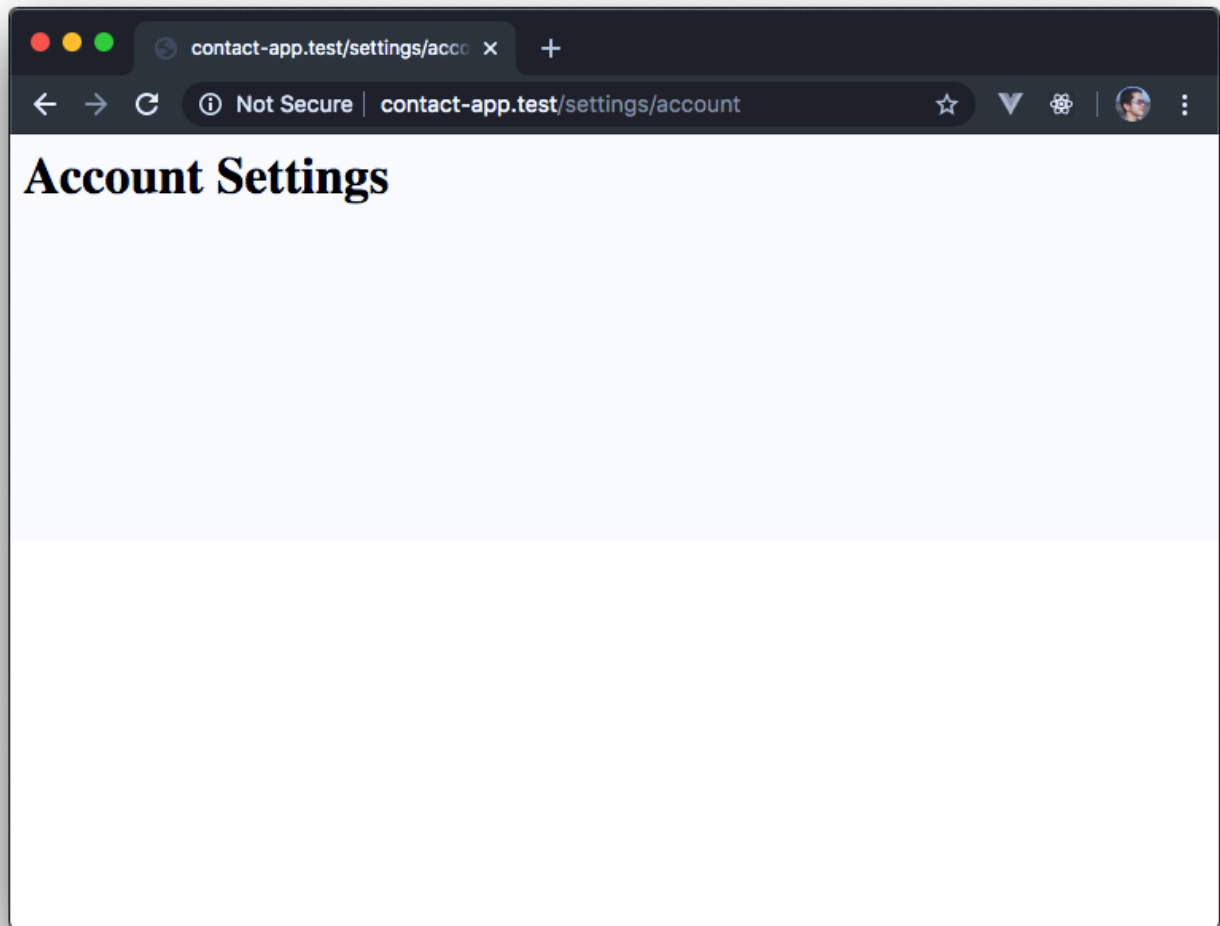
Let's open the `web.php` file then add a new route for the Account Controller.

```
Route::get('/settings/account', [AccountController::class, 'index']);
```

Don't forget to import the `AccountController` namespace at the top of the file.

```
use App\Http\Controllers\Settings\AccountController;
```

If we now access the `settings/account` route in our browser, we'll get the **Account Settings** page on the screen.



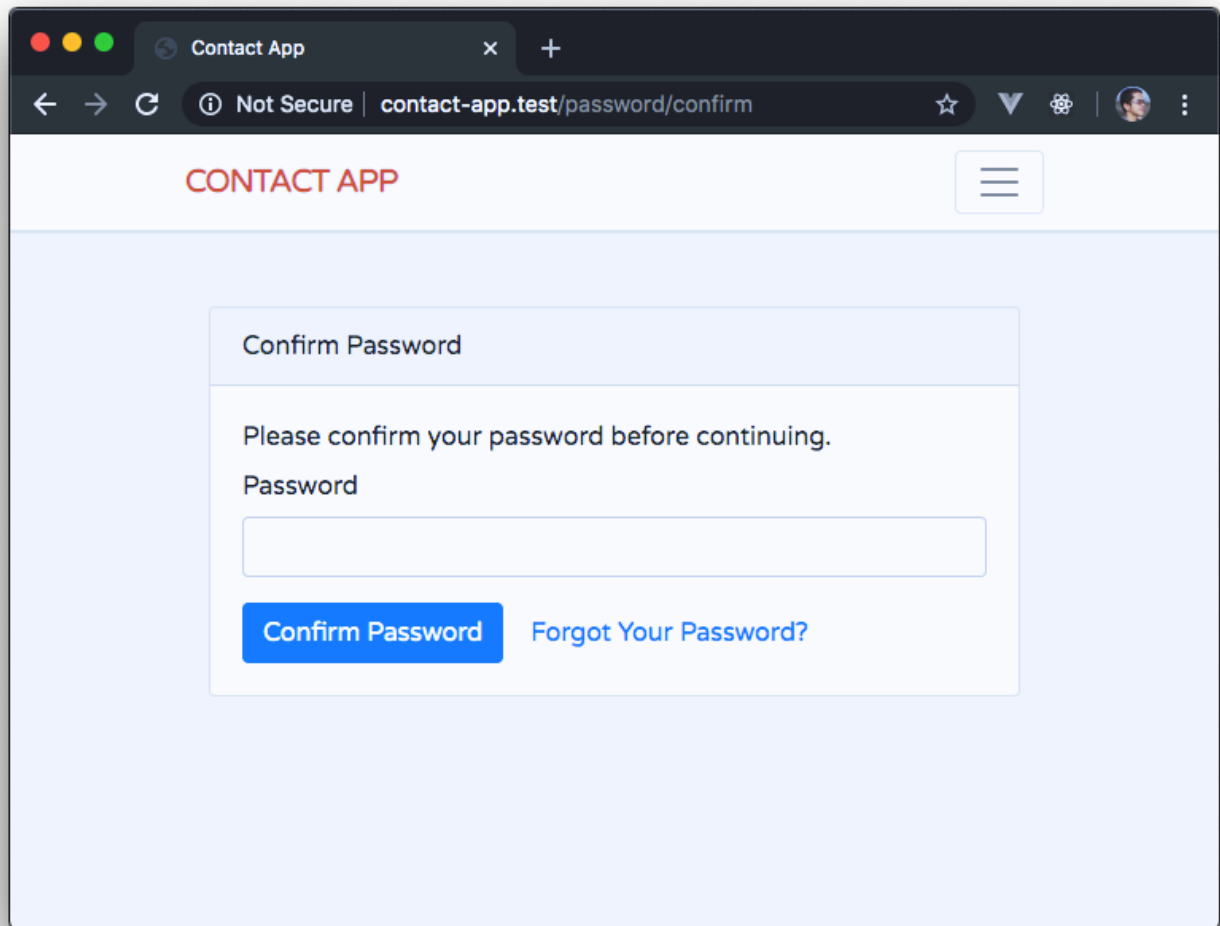
3. Implement the `password.confirm`

To implement the `password.confirm` you can do in the route definition or in controller's constructor. In this case I'll do that in the controller's constructor. In `AccountController` we call the `auth` middleware to ensure that this page can be accessed by authenticated user. We then also call `password.confirm` middleware to force the authenticated user to confirm their password.

```
class AccountController extends Controller
{
  public function __construct()
  {
    $this->middleware(['auth', 'password.confirm']);
  }

  public function index()
  {
    return "<h1>Account Settings</h1>";
  }
}
```

Now if you visit the `settings/account` route in your browser, you will be taken to **Confirm Password** page.



If you can still access the **Account Setting** page, probably you've just login to your app. You will not be asked to confirm their password again for three hours. You can configure the length of time by changing the value of the `password_timeout` in your `config/auth.php` file.

```
'password_timeout' => 10800,
```

Once you enter your password correctly, you will be redirected back to the **Account Settings** page.

