Lets first understand how the authentication process works in laravel.

Laravel uses guards for authentication.

'guards' **=>** [

'web' **=>** [

'driver' **=>** 'session',

'provider' **=>** 'users',

],

'admin' **=>** [

'driver' **=>** 'session',

'provider' **=>** 'admins',

],

'api' **=>** [

'driver' **=>** 'token',

'provider' **=>** 'users',

],

],

Guards are defined in the auth config file.

Guards use session driver to keep track of the authentication data such as id and password.

Guards are mapped to a model via the service provider in the service provider section of the auth config file.

'providers' **=>** [

'users' **=>** [

'driver' **=>** 'eloquent',

'model' **=>** **App\User::class**,

],

'admins' **=>** [

'driver' **=>** 'eloquent',

'model' **=>** **App\Models\Admin::class**,

],

The model referenced must extend the Auth\User

**use** **Illuminate\Foundation\Auth\User** **as** Authenticatable;

class Admin extends *Authenticatable*

Instead of the Eloquent model

**Illuminate\Database\Eloquent\Model**

Notifiable class

**use** **Illuminate\Notifications\Notifiable**;

can be optionally used. In such case

**use** **Notifiable**;

should be defined inside the class.

If you want change the primary key name from id to something else then you have to define that in your model. You can also define timestamps and and primary key incrementing to false.

public $timestamps**=** false;

protected $primaryKey**=**'admin\_id';

public $incrementing **=**false;

and also you will have to add corresponding primary fluent method along with remember token on your migration file.

$table**->**integer('admin\_id')**->**primary();

$table**->**rememberToken();

Once the model has been adjusted for being used by the auth middleware, the next job is to add the middleware to the respective controller.

public function \_\_construct(){

$this**->**middleware('auth:admin');

}

Now, in the Logincontroller login method check if the credentials provided for login matches the database record. If so, call the auth helper’s attempt method to login with the guards provided as the parameter.

**if** (auth()**->**guard('admin')**->**attempt($userdata)) {

*// validation successful!*

*// redirect them to the secure section or whatever*

**return** redirect()**->**route('dashboard');

}

However note that since controller uses the authenticate middleware you will have to comment out the default if condition

*// if (! $request->expectsJson()) {*

*// return route('login');*

*//}*

and do the redirection in that middleware with your own logic.

**if**(auth('admin')**->**check()){

**return** route('dashboard');

}

**else**{

**return** route('login');

}

Now the user should be successfully logged in.

**Logout**

Simply call the auth($guard)->logout() method and the user will be logged out. Don’t forget to redirect the user to login page.

public function logout(){

auth('admin')**->**logout();

*//dd(auth('admin')->check());*

**return** redirect()**->**route('login');

}

Another Note: what happens when a logged in user requests the login route?

In that case laravel will not automatically logout the user and as such they will be able to view both the dashboard and login page which would cause an awkward user experience.

To avoid such scenario first call the logout method inside the showLoginform method and then return the login view. Doing so will flush all session and auth data and if the user now tries to access the dashboard he will be redirected to login page.

public function loginForm(){

*//flush auth data*

$this**->**logout();

**return** view('backend.login');

}

That’s all about custom authentication.