



Mata Kuliah : Pemrograman Web Lanjut (PWL)  
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis  
Semester : 4 (empat) / 5 (lima)  
Pertemuan ke- : 7 (tujuh)

Muhammad Fairuz Daffa Athallah

SIB 2B / 15

2341760079

## JOBSHEET 07

### Authentication dan Authorization di Laravel

Laravel Authentication dipergunakan untuk memproteksi halaman atau fitur dari web yang hanya diakses oleh orang tertentu yang diberikan hak. Fitur seperti ini biasanya ditemui di sistem yang memiliki fitur administrator atau sistem yang memiliki pengguna yang boleh menambahkan datanya.

Laravel membuat penerapan otentikasi sangat sederhana dan telah menyediakan berbagai fitur yang dapat dimanfaatkan tanpa perlu melakukan penambahan instalasi modul tertentu. File konfigurasi otentikasi terletak di `config / auth.php`, yang berisi beberapa opsi yang terdokumentasi dengan baik untuk mengubah konfigurasi dari layanan otentikasi.

Pada intinya, fasilitas otentikasi Laravel terdiri dari “*guards*” dan “*providers*”. *Guards* menentukan bagaimana pengguna diautentikasi untuk setiap permintaan. Misalnya, Laravel mengirim dengan *guards* untuk sesi dengan menggunakan penyimpanan session dan cookie.

#### Middleware

**Middleware** adalah lapisan perantara antara permintaan *route HTTP* yang masuk dan *action* dari Controller yang akan dijalankan. **Middleware** memungkinkan kita untuk melakukan berbagai tugas baik itu sebelum ataupun sesudah tindakan dilakukan. Kita juga dapat menggunakan *tool CLI* untuk membuat sebuah **Middleware** dalam **Laravel**. Beberapa contoh penggunaan **Middleware** meliputi autentikasi, validasi, manipulasi permintaan, dan lainnya. Berikut di bawah ini adalah manfaat dari **Middleware** :

- **Keamanan** : dalam **Middleware** memungkinkan kita untuk memverifikasi apakah pengguna sudah diautentikasi sebelum mengakses halaman tertentu. Dengan demikian, kita dapat melindungi data sensitif dan mengontrol hak akses pengguna.
- **Pemfilteran Data** : **Middleware** dapat digunakan untuk memanipulasi data permintaan sebelum sebuah *action* dalam *controller* dilakukan. Misalnya, kita dapat memeriksa terlebih dahulu data yang dikirim oleh pengguna sebelum data tersebut diproses lebih



lanjut atau kita ingin memodifikasi data yang akan dikirim lalu kita dapat memeriksa ulang data yang akan dikirim oleh pengguna sebelum data tersebut diproses.

- **Logging dan Audit : Middleware** juga dapat digunakan untuk mencatat aktivitas pengguna atau melakukan audit terhadap permintaan yang masuk. Ini dapat membantu dalam pemantauan dan analisis aplikasi.

### INFO

Kita akan menggunakan Laravel Auth secara manual seperti  
<https://laravel.com/docs/10.x/authentication#authenticating-users>

Sesuai dengan **Studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL\_POS**.

*Project PWL\_POS* akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

## A. Implementasi Manual Authentication di Laravel

Autentikasi adalah proses untuk memverifikasi identitas pengguna yang mencoba mengakses sistem. Dalam konteks aplikasi web, autentikasi memastikan bahwa pengguna yang mencoba login memiliki hak akses yang sesuai berdasarkan kredensial seperti email dan password. Proses autentikasi berbeda dengan **otorisasi**, yang merupakan langkah lanjutan untuk menentukan hak akses apa yang dimiliki pengguna setelah mereka berhasil diautentikasi.

### Konsep Autentikasi di Laravel

Laravel menawarkan sistem autentikasi yang sangat fleksibel. Laravel menyediakan mekanisme autentikasi bawaan melalui layanan authentication scaffolding seperti Laravel *Jetstream* dan *Breeze*, yang dapat secara otomatis menghasilkan halaman dan logika autentikasi. Namun, terkadang pengembang memerlukan implementasi autentikasi yang lebih manual untuk memberikan kontrol penuh terhadap setiap aspek dari proses tersebut.

Beberapa komponen penting dalam sistem autentikasi Laravel meliputi:

- *Guard*: Komponen yang mengatur bagaimana pengguna diautentikasi untuk setiap permintaan. *Guard* default menggunakan sesi dan cookie.



- *Provider*: Mengatur bagaimana pengguna diambil dari database atau sumber data lainnya. *Provider* default mengambil data pengguna dari database dengan menggunakan Eloquent ORM.
- *Session*: Laravel menggunakan sesi untuk menyimpan status autentikasi pengguna. Sesi memungkinkan sistem untuk mengingat pengguna yang sudah login di antara permintaan HTTP yang berbeda.

Alur umum dari autentikasi meliputi:

1. *Login*: Pengguna mengirimkan kredensial (biasanya berupa email dan password).
2. *Verifikasi Kredensial*: Sistem memeriksa apakah kredensial yang diberikan sesuai dengan data di database.
3. *Pembuatan Sesi*: Jika kredensial benar, sistem akan membuat sesi untuk pengguna yang akan disimpan di server.
4. *Akses ke Halaman yang Dilindungi*: Pengguna yang terautentikasi dapat mengakses halaman-halaman yang dilindungi oleh *middleware* auth.
5. *Logout*: Pengguna bisa keluar dari sistem dan sesi mereka akan dihapus.

### Middleware Autentikasi

Middleware auth di Laravel digunakan untuk melindungi rute atau halaman agar hanya dapat diakses oleh pengguna yang sudah terautentikasi. Jika pengguna mencoba mengakses rute yang memerlukan autentikasi tanpa login, mereka akan diarahkan ke halaman login.

- **Guard** bertanggung jawab untuk menangani proses autentikasi pengguna. Laravel secara default menggunakan *guard* berbasis sesi untuk autentikasi web, namun juga mendukung *guard* berbasis token (seperti API).
- **Provider** bertugas untuk mengambil pengguna dari database. Laravel menyediakan *provider* default yang menggunakan Eloquent, namun juga mendukung *provider* lain seperti Query Builder.

### Implementasi di Laravel 10

Kita akan menerapkan penggunaan authentication di Laravel. Dalam penerapan ini, kita akan mencoba membuat otentikasi secara di Laravel, agar kita paham langkah-langkah dalam membuat Authentication



## Praktikum 1 – Implementasi Authentication :

1. Kita buka project laravel **PWL\_POS** kita, dan kita modifikasi konfigurasi aplikasi kita di **config/auth.php**

```
62     'providers' => [  
63         'users' => [  
64             'driver' => 'eloquent',  
65             'model' => App\Models\User::class,  
66         ],
```

Pada bagian ini kita sesuaikan dengan Model untuk tabel m\_user yang sudah kita buat

```
62     'providers' => [  
63         'users' => [  
64             'driver' => 'eloquent',  
65             'model' => App\Models\UserModel::class,  
66         ],  
67
```

```
62     'providers' => [  
63         'users' => [  
64             'driver' => 'eloquent',  
65             'model' => App\Models\UserModel::class,  
66         ],
```

2. Selanjutnya kita modifikasi sedikit pada **UserModel.php** untuk bisa melakukan proses otentikasi

```
<?php  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable  
  
class UserModel extends Authenticatable  
{  
    use HasFactory;  
  
    protected $table = 'm_user';  
    protected $primaryKey = 'user_id';  
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];  
  
    protected $hidden = ['password']; // jangan di tampilkan saat select  
  
    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash  
  
    /**  
     * Relasi ke tabel level  
     */  
    public function level(): BelongsTo  
    {  
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
    }  
}
```



```
Minggu_7 > PWL_PCIS > app > Models > UserModel.php > UserModel > getRole
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;
8 use Illuminate\Foundation\Auth\User as Authenticatable;
9
10 class UserModel extends Authenticatable
11 {
12     use HasFactory;
13
14     protected $table = 'm_user';
15     protected $primaryKey = 'user_id'; // mendefinisikan primary key dari tabel
16
17     protected $fillable=['level_id','username','nama','password','created_at','updated_at'];
18
19     protected $hidden = ['password']; //jangan di tampilkan saat select
20     protected $casts = ['password' => 'hashed']; //casting password agar otomatis di hash
21
22     public function level(): BelongsTo
23     {
24         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
25     }
26 }
```

3. Selanjutnya kita buat [AuthController.php](#) untuk memproses login yang akan kita lakukan



```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    public function login()
    {
        if(Auth::check()){ // jika sudah login, maka redirect ke halaman home
            return redirect('/');
        }
        return view('auth.login');
    }

    public function postlogin(Request $request)
    {
        if($request->ajax() || $request->wantsJson()){
            $credentials = $request->only('username', 'password');

            if (Auth::attempt($credentials)) {
                return response()->json([
                    'status' => true,
                    'message' => 'Login Berhasil',
                    'redirect' => url('/')
                ]);
            }

            return response()->json([
                'status' => false,
                'message' => 'Login Gagal'
            ]);
        }

        return redirect('login');
    }

    public function logout(Request $request)
    {
        Auth::logout();

        $request->session()->invalidate();
        $request->session()->regenerateToken();
        return redirect('login');
    }
}
```



```
Minggu_7 > PWL_POS > app > Http > Controllers > AuthController.php > ...  
8  class AuthController extends Controller  
9  {  
10     public function login()  
11     {  
12         // Jika sudah login, redirect ke halaman home  
13         if (Auth::check()) {  
14             return redirect('/');  
15         }  
16         return view('auth.login');  
17     }  
18  
19     public function postlogin(Request $request)  
20     {  
21         if ($request->ajax() || $request->wantsJson()) {  
22             $credentials = $request->only('username', 'password');  
23  
24             if (Auth::attempt($credentials)) {  
25                 return response()->json([  
26                     'status' => true,  
27                     'message' => 'Login Berhasil',  
28                     'redirect' => url('/'),  
29                 ]);  
30             }  
31  
32             return response()->json([  
33                 'status' => false,  
34                 'message' => 'Login Gagal',  
35             ]);  
36         }  
37  
38         return redirect('login');  
39     }  
40  
41     public function logout(Request $request)  
42     {  
43         Auth::logout();  
44     }
```

4. Setelah kita membuat `AuthController.php`, kita buat view untuk menampilkan halaman login. View kita buat di `auth/login.blade.php`, tampilan login bisa kita ambil dari contoh login di template **AdminLTE** seperti berikut (pada contoh login ini, kita gunakan page `login-V2` di **AdminLTE**)

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <title>Login Pengguna</title>
```



```
<!-- Google Font: Source Sans Pro -->
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallb
ack">
<!-- Font Awesome -->
<link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
<!-- icheck bootstrap -->
<link rel="stylesheet" href="{{ asset('plugins/icheck-bootstrap/icheck-bootstrap.min.css')
}}">
<!-- SweetAlert2 -->
<link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-
4.min.css') }}">
<!-- Theme style -->
<link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
</head>
<body class="hold-transition login-page">
<div class="login-box">
<!-- /.login-logo -->
<div class="card card-outline card-primary">
<div class="card-header text-center"><a href="{{ url('/') }}"
class="h1"><b>Admin</b></a></div>
<div class="card-body">
<p class="login-box-msg">Sign in to start your session</p>
<form action="{{ url('login') }}" method="POST" id="form-login">
@csrf
<div class="input-group mb-3">
<input type="text" id="username" name="username" class="form-control"
placeholder="Username">
<div class="input-group-append">
<div class="input-group-text">
<span class="fas fa-envelope"></span>
</div>
</div>
<small id="error-username" class="error-text text-danger"></small>
</div>
<div class="input-group mb-3">
<input type="password" id="password" name="password" class="form-control"
placeholder="Password">
<div class="input-group-append">
<div class="input-group-text">
<span class="fas fa-lock"></span>
</div>
</div>
<small id="error-password" class="error-text text-danger"></small>
</div>
<div class="row">
<div class="col-8">
<div class="icheck-primary">
<input type="checkbox" id="remember"><label for="remember">Remember Me</label>
</div>
<!-- /.col -->
<div class="col-4">
<button type="submit" class="btn btn-primary btn-block">Sign In</button>
</div>
<!-- /.col -->
</div>
</form>
</div>
<!-- /.card-body -->
</div>
<!-- /.card -->
</div>
<!-- /.login-box -->

<!-- jQuery -->
```

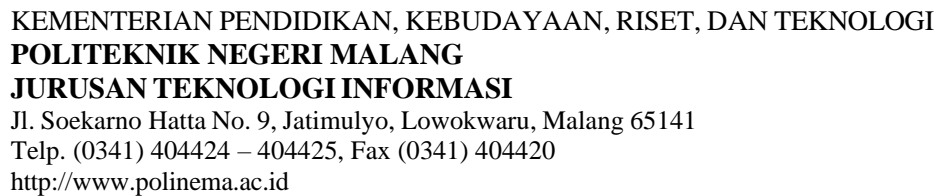




```
<script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<!-- jquery-validation -->
<script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<script src="{{ asset('plugins/jquery-validation/additional-methods.min.js') }}"></script>
<!-- SweetAlert2 -->
<script src="{{ asset('plugins/sweetalert2/sweetalert2.min.js') }}"></script>
<!-- AdminLTE App -->
<script src="{{ asset('dist/js/adminlte.min.js') }}"></script>

<script>
$.ajaxSetup({
  headers: {
    'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
  }
});

$(document).ready(function() {
  $("#form-login").validate({
    rules: {
      username: {required: true, minlength: 4, maxlength: 20},
      password: {required: true, minlength: 6, maxlength: 20}
    },
    submitHandler: function(form) { // ketika valid, maka bagian yg akan dijalankan
      $.ajax({
        url: form.action,
        type: form.method,
        data: $(form).serialize(),
        success: function(response) {
          if(response.status){ // jika sukses
            Swal.fire({
              icon: 'success',
              title: 'Berhasil',
              text: response.message,
            }).then(function() {
              window.location = response.redirect;
            });
          }else{ // jika error
            $('.error-text').text('');
            $.each(response.msgField, function(prefix, val) {
              $('#error-'+prefix).text(val[0]);
            });
            Swal.fire({
              icon: 'error',
              title: 'Terjadi Kesalahan',
              text: response.message
            });
          }
        }
      });
    },
    errorElement: 'span',
    errorPlacement: function (error, element) {
      error.addClass('invalid-feedback');
      element.closest('.input-group').append(error);
    },
    highlight: function (element, errorClass, validClass) {
      $(element).addClass('is-invalid');
    },
    unhighlight: function (element, errorClass, validClass) {
      $(element).removeClass('is-invalid');
    }
  });
});
</script>
```



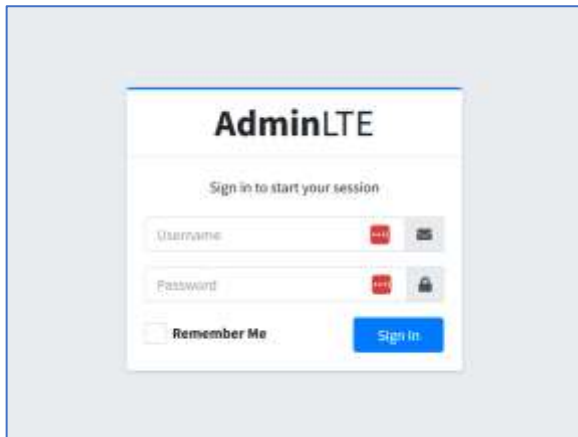
5. Kemudian kita modifikasi `route/web.php` agar semua route masuk dalam auth.

Jobsheet 07 – PWL 2023/2024

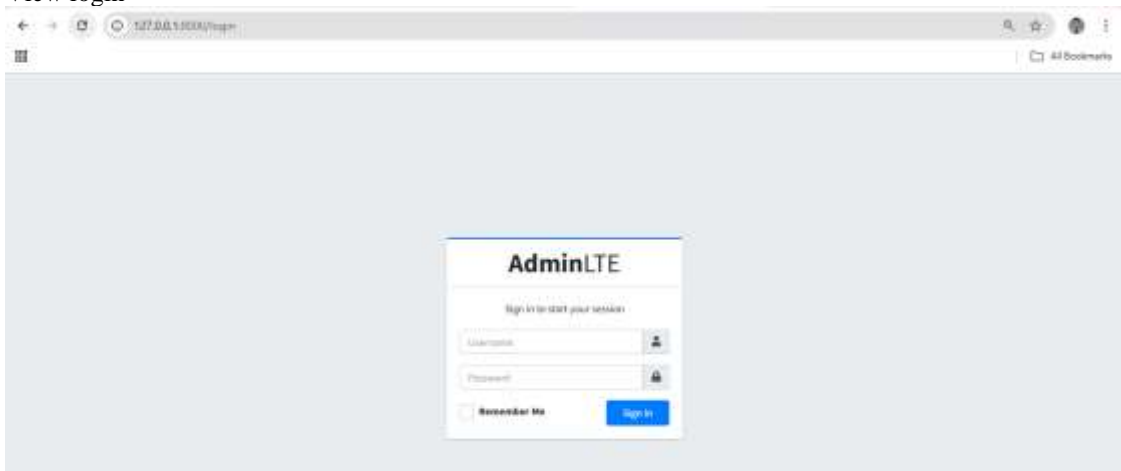


```
41 //2x 3
42 Route::pattern('id', '{0-9}+');
43 Route::get('login', [AuthController::class, 'login'])->name('login');
44 Route::post('login', [AuthController::class, 'postlogin']);
45 Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
46
47 Route::middleware(['auth'])->group(function(){
48     //2x 5
49     //Praktikum 2
50 Route::get('/', [HomeController::class, 'index']);
51 //Praktikum 3
52 Route::group(['prefix' => 'user'], function(){
53     Route::get('/', [UserController::class, 'index'])->memanggilkan halaman awal
54     Route::post('/list', [UserController::class, 'list'])->memanggilkan data user bentuk json / datatables
55     Route::get('/create', [UserController::class, 'create'])->memanggilkan bentuk form untuk tambah user
56     Route::post('/', [UserController::class, 'store'])->memanggilkan user data baru
57     Route::get('/create_ajax', [UserController::class, 'create_ajax'])->memanggilkan bentuk form untuk tambah user ajax js 4
58     Route::post('/ajax', [UserController::class, 'store_ajax'])->memanggilkan user data baru ajax js 6
59     Route::get('/{id}', [UserController::class, 'show'])->memanggilkan detail user
60     Route::get('/{id}/edit', [UserController::class, 'edit'])->memanggilkan halaman form edit user
61     Route::put('/{id}', [UserController::class, 'update'])->memanggilkan perubahan data user
62     Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax'])->memanggilkan halaman form edit user ajax js 6
63     Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax'])->memanggilkan perubahan data user ajax js 6
64     Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax'])->untuk tampilan form confirm delete ajax js 6
65     Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax'])->untuk hapus data ajax js 6
66     Route::delete('/{id}', [UserController::class, 'destroy'])->menghapus data user
67 });
68
69 Route::group(['prefix' => 'level'], function(){
70     Route::get('/', [LevelController::class, 'index'])->memanggilkan halaman awal
71     Route::post('/list', [LevelController::class, 'list'])->memanggilkan data user bentuk json / datatables
72     Route::get('/create', [LevelController::class, 'create'])->memanggilkan bentuk form untuk tambah user
73     Route::post('/', [LevelController::class, 'store'])->memanggilkan user data baru
74     Route::get('/create_ajax', [LevelController::class, 'create_ajax'])->memanggilkan bentuk form untuk tambah user ajax js 6
75     Route::post('/ajax', [LevelController::class, 'store_ajax'])->memanggilkan user data baru ajax js 6
76     Route::get('/{id}', [LevelController::class, 'show'])->memanggilkan detail user
77 });
```

6. Ketika kita coba mengakses halaman [localhost/PWL\\_POS/public](localhost/PWL_POS/public) makan akan tampil halaman awal untuk login ke aplikasi



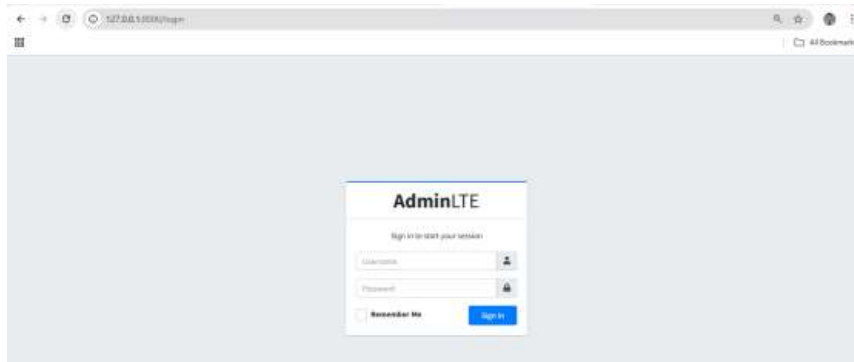
View login





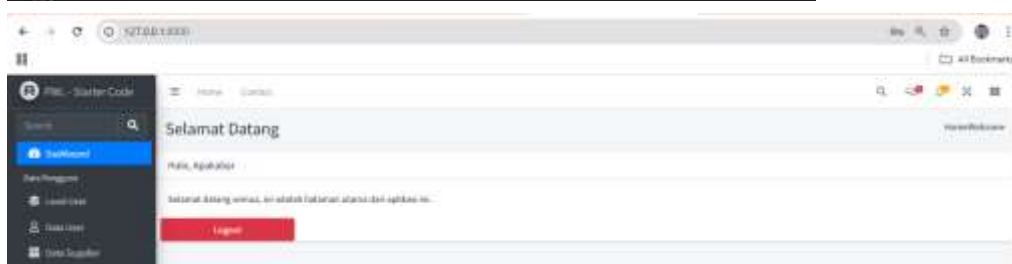
## Tugas 1 – Implementasi Authentication :

1. Silahkan implementasikan proses login pada project kalian masing-masing



2. Silahkan implementasi proses logout pada halaman web yang kalian buat

```
Minggu_7 > PWL_POS > resources > views > welcome.blade.php > @div.col-2 > @form#logout-form
1 @extends('layouts.template')
2
3 @section('content')
4
5 <div class="card">
6   <div class="card-header">
7     <h3 class="card-title">Halo, Apakabar</h3>
8     <div class="card-tools"></div>
9   </div>
10  <div class="card-body">
11    Selamat datang semua, ini adalah halaman utama dari aplikasi ini.
12  </div>
13  <div class="col-2">
14    <form id="logout-form" action="{{ url('logout') }}" method="GET">
15      @csrf
16      <button type="submit" class="btn btn-danger btn-block">Logout</button>
17    </form>
18  </div>
19 </div>
20 @endsection
21
```



3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan Pertama kita mengkonfigurasi UserModel dengan Authenticatable Kedua kita menambahkan route login dan logout dan middleware auth Ketiga membuat dan controllerAuth yang diisi dengan function function nya.
4. Submit kode untuk impementasi Authentication pada repository github kalian.



## B. Implementasi *Authorization* di Laravel

*Authorization* merupakan proses setelah *authentication* berhasil dilakukan (dalam kata lain, kita berhasil login ke sistem). *Authorization* berkenaan dengan hak akses pengguna dalam menggunakan sistem. *Authorization* memberikan/memastikan hak akses (ijin akses) kita, sesuai dengan aturan (role) yang ada di sistem. *Authorization* sangat penting untuk membatasi akses pengguna sesuai dengan peruntukannya.

**Contoh** ketika kita mengakses LMS dengan akun (*username* dan *password*) yang bertipe **Mahasiswa**. Saat berhasil melakukan *authentication*, maka hak akses kita juga akan diberikan selayaknya mahasiswa. Seperti melihat kursus (course), melihat materi, men-download file materi, mengerjakan/meng-upload tugas, mengikuti ujian, dll. Kita tidak akan diberikan hak akses oleh sistem untuk membuat materi, membuat soal ujian, membuat tugas, memberikan nilai tugas karena hak akses tersebut masuk ke ranah akun tipe **Dosen/Pengajar**.

Selain menyediakan layanan otentikasi bawaan, Laravel juga menyediakan cara sederhana untuk mengotorisasi tindakan pengguna terhadap sumber daya tertentu. Misalnya, meskipun pengguna diautentikasi, mereka mungkin tidak berwenang untuk memperbarui atau menghapus model Eloquent atau rekaman database tertentu yang dikelola oleh aplikasi Anda. Fitur otorisasi Laravel menyediakan cara yang mudah dan terorganisir untuk mengelola jenis pemeriksaan otorisasi ini.

### Praktikum 2 – Implementasi *Authorization* di Laravel dengan Middleware

Kita akan menerapkan *authorization* pada project Laravel dengan menggunakan Middleware sebagai pengecekan akses. Langkah-langkah yang kita kerjakan sebagai berikut:

1. Kita modifikasi **UserModel.php** dengan menambahkan kode berikut

```
/**
 * Relasi ke tabel level
 */
public function level(): BelongsTo
{
    return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
}

/**
 * Mendapatkan nama role
 */
public function getRoleName(): string
{
    return $this->level->level_nama;
}

/**
 * Cek apakah user memiliki role tertentu
 */
public function hasRole($role): bool
{
    return $this->level->level_kode == $role;
}
```





```
21  
22     public function level(): BelongsTo  
23     {  
24         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
25     }  
26  
27     public function getRoleName(): string{  
28         return $this->level->level_nama;  
29     }  
30  
31     public function hasRole($role): bool{  
32         return $this->level->level_kode == $role;  
33     }  
34 }
```

2. Kemudian kita buat *middleware* dengan nama `AuthorizeUser.php`. Kita bisa buat *middleware* dengan mengetikkan perintah pada terminal/CMD

```
php artisan make:middleware AuthorizeUser
```

File *middleware* akan dibuat di `app/Http/Middleware/AuthorizeUser.php`

3. Kemudian kita edit *middleware* `AuthorizeUser.php` untuk bisa mengecek apakah pengguna yang mengakses memiliki Level/Role/Group yang sesuai

```
1  <?php  
2  namespace App\Http\Middleware;  
3  
4  use Closure;  
5  use Illuminate\Http\Request;  
6  use Symfony\Component\HttpFoundation\Response;  
7  
8  class AuthorizeUser  
9  {  
10     /**  
11      * Handle an incoming request.  
12      *  
13      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  
14      */  
15     public function handle(Request $request, Closure $next, $role = ''): Response  
16     {  
17         $user = $request->user(); // ambil data user yg login  
18         // fungsi user() diambil dari UserModel.php  
19         if($user->hasRole($role)){ // cek apakah user punya role yg diinginkan  
20             return $next($request);  
21         }  
22         // jika tidak punya role, maka tampilkan error 403  
23         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');  
24     }  
25 }
```



### AuthorizeUser

```
Minggu_7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser > handle
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthorizeUser
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
15      */
16     public function handle(Request $request, Closure $next, $role = ''): Response
17     {
18         $user = $request->user();
19
20         if ($user->hasRole($role)) {
21             return $next($request);
22         }
23
24         abort(403, 'Forbidden kamu tidak punya akses ke halaman ini');
25     }
26 }
```

4. Kita daftarkan ke `app/Http/Kernel.php` untuk *middleware* yang kita buat barusan

```
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
```

```
Minggu_7 > PWL_POS > app > Http > Kernel.php > Kernel > $middlewareAliases
7  class Kernel extends HttpKernel
8
9  /**
10   * The application's middleware aliases.
11   *
12   * Aliases may be used instead of class names to conveniently assign middleware to routes and groups.
13   *
14   * @var array<string, class-string|string>
15   */
16     protected $middlewareAliases = [
17         'auth' => \App\Http\Middleware\Authenticate::class,
18         'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat
19         'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
20         'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
21         'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
22         'can' => \Illuminate\Auth\Middleware\Authorize::class,
23         'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
24         'password.confirmed' => \Illuminate\Auth\Middleware\RequirePassword::class,
25         'precognitive' => \Illuminate\Http\Middleware\HandlePrecognitiveRequests::class,
26         'signed' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
27         'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
28         'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
```



5. Sekarang kita perhatikan tabel `m_level` yang menjadi tabel untuk menyimpan level/group/role dari user ada

level_id	level_kode	level_nama	created_at	updated_at	deleted_at
1	ADM	Administrator	NULL	NULL	NULL
2	MNG	Manager	NULL	NULL	NULL
3	STF	Staf	NULL	2024-08-16 01:49:20	NULL
4	KSR	Kasir	NULL	NULL	NULL





6. Untuk mencoba *authorization* yang telah kita buat, maka perlu kita modifikasi `route/web.php` untuk menentukan route mana saja yang akan diberi hak akses sesuai dengan level user

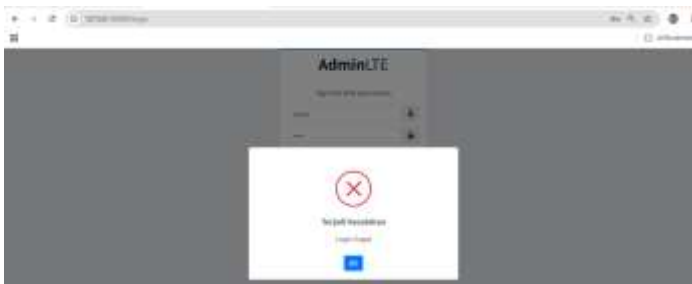
```
Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu
    Route::get('/', [WelcomeController::class,'index']);
    // route Level

    // artinya semua route di dalam group ini harus punya role ADM (Administrator)
    Route::middleware(['authorize:ADM'])->group(function(){
        Route::get('/level',[LevelController::class,'index']);
        Route::post('/level/list',[LevelController::class,'list']); // untuk list json datatables
        Route::get('/level/create',[LevelController::class,'create']);
        Route::post('/level',[LevelController::class,'store']);
        Route::get('/level/{id}/edit',[LevelController::class,'edit']); // untuk tampilkan form edit
        Route::put('/level/{id}',[LevelController::class,'update']); // untuk proses update data
        Route::delete('/level/{id}',[LevelController::class,'destroy']); // untuk proses hapus data
    });
    // route Kategori
```

```
53 Route::middleware(['authorize:ADM'])->group(function() {
```

Pada kode yang ditandai merah, terdapat `authorize:ADM`. Kode `ADM` adalah nilai dari `level_kode` pada tabel `m_level`. Yang artinya, user yang bisa mengakses route untuk manage data level, adalah user yang memiliki level sebagai Administrator.

7. Untuk membuktikannya, sekarang kita coba login menggunakan akun selain level administrator, dan kita akses route menu level tersebut



## Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?  
Dalam praktikum ini hanya role yang terdaftar dalam database saja yang bisa melakukan login
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan  
Pertama dalam UserModel kita menambahkan fungsi `getroleName` dan `hasRole` untuk mengembalikan `level_nama` dan `level_kode`  
Kedua membuat file `authorizeUser` pada middleware dan melakukan cek apakah user memiliki role jika tidak maka tidak bisa melakukan login  
Ketiga menambahkan `authorize` pada kernel lalu menambahkan route middleware(`authorize adm`)
3. Submit kode untuk impementasi Authorization pada repository github kalian.



## C. Multi-Level Authorization di Laravel

Bagaimana seandainya jika terdapat level/group/role satu dengan yang lain memiliki hak akses yang sama. Contoh sederhana, user level Admin dan Manager bisa sama-sama mengakses menu Barang pada aplikasi yang kita buat. Maka tidak mungkin kalau kita buat route untuk masing-masing level user. Hal ini akan memakan banyak waktu, dan proses yang lama.

```
// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});

// artinya semua route di dalam group ini harus punya role MNG (Manager)
Route::middleware(['authorize:MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

Hal ini jadi kendala ketika kita mau mengganti hak akses, maka kita akan mengganti sebagian besar route yang sudah kita tulis. Untuk itu, kita perlu mengelola middleware agar bisa mendukung penambahan hak akses secara dinamis.

### Praktikum 3 – Implementasi Multi-Level Authorization di Laravel dengan Middleware

Kita akan menerapkan multi-level authorization pada project Laravel dengan menggunakan Middleware sebagai pengecekan akses. Langkah-langkah yang kita kerjakan sebagai berikut:

1. Kita modifikasi `UserModel.php` untuk mendapatkan `level_kode` dari user yang sudah login. Jadi kita buat fungsi dengan nama `getRole()`



```
/**
 * Mendapatkan nama role
 */
public function getRoleName(): string
{
    return $this->level->level_nama;
}

/**
 * Cek apakah user memiliki role tertentu
 */
public function hasRole($role): bool
{
    return $this->level->level_kode == $role;
}

/**
 * Mendapatkan kode role
 */
public function getRole()
{
    return $this->level->level_kode;
}
```

```
34
35     public function getRole() {
36         return $this->level->level_kode;
37     }
38
39 }
```

2. Selanjutnya, Kita modifikasi middleware `AuthorizeUser.php` dengan kode berikut

```
1 <?php
2 namespace App\Http\Middleware;
3
4 use Closure;
5 use Illuminate\Http\Request;
6 use Symfony\Component\HttpFoundation\Response;
7
8 class AuthorizeUser
9 {
10     /**
11      * Handle an incoming request.
12      *
13      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
14      */
15     public function handle(Request $request, Closure $next, ... $roles): Response
16     {
17         $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
18         if(in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
19             return $next($request); // jika ada, maka lanjutkan request
20         }
21         // jika tidak punya role, maka tampilkan error 403
22         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
23     }
24 }
```

```
9 class AuthorizeUser
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
15      */
16     public function handle(Request $request, Closure $next, ... $roles): Response
17     {
18         $user_role = $request->user()->getRole(); //ambil data level_kode dari user
19         if (in_array($user_role, $roles)) { // cek apakah level_kode user ada dalam array roles
20             return next($request); // jika ada lanjutkan req
21         }
22
23         abort(403, 'Forbidden kamu tidak punya akses ke halaman ini'); // error jika tidak ada
24     }
25 }
```

3. Setelah itu tinggal kita perbaiki `route/web.php` sesuaikan dengan role/level yang diinginkan. Contoh





```
// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

4. Sekarang kita sudah bisa memberikan hak akses menu/route ke beberapa level user

```
70 Route::middleware(['authorize:ADM'])->prefix('level')->group(function () {
71     Route::get('/',[LevelController::class,'index']); // menampilkan halaman awal
72     Route::post('/list',[LevelController::class,'list']); // menampilkan data user bentuk json / datatable
73     Route::get('/create',[LevelController::class,'create']); // menampilkan bentuk form untuk tambah user
74     Route::post('/',[LevelController::class,'store']); // menyimpan user data baru
75     Route::get('/create_ajax',[LevelController::class,'create_ajax']); // menampilkan bentuk form untuk tambah user ajax js 6
76     Route::post('/ajax',[LevelController::class,'store_ajax']); // menyimpan user data baru ajax js 6
77     Route::get('/{id}',[LevelController::class,'show']); // menampilkan detail user
78     Route::get('/{id}/edit',[LevelController::class,'edit']); // menampilkan halaman form edit user
79     Route::put('/{id}',[LevelController::class,'update']); // menyimpan perubahan data user
80     Route::get('/{id}/edit_ajax',[LevelController::class,'edit_ajax']); // menampilkan halaman form edit user ajax js 6
81     Route::put('/{id}/update_ajax',[LevelController::class,'update_ajax']); // menyimpan perubahan data user ajax js 6
82     Route::get('/{id}/delete_ajax',[LevelController::class,'confirm_ajax']); // menghapus data user ajax js 6
83     Route::delete('/{id}/delete_ajax',[LevelController::class,'delete_ajax']); // menghapus data user ajax js 6
84     Route::delete('/{id}',[LevelController::class,'destroy']); // menghapus data user
85 });
```

### Tugas 3 – Implementasi Multi-Level Authorization :

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan  
Pertama modifikasi authorizeUser agar bisa mengambil nilai role lalu cek role yang tersedia menggunakan array  
Kedua modifikasi untuk akses pada route web.php
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User

```
70 Route::middleware(['authorize:ADM'])->prefix('level')->group(function () {
71     Route::get('/',[LevelController::class,'index']); // menampilkan halaman awal
72     Route::post('/list',[LevelController::class,'list']); // menampilkan data user bentuk json / datatable
73     Route::get('/create',[LevelController::class,'create']); // menampilkan bentuk form untuk tambah user
74     Route::post('/',[LevelController::class,'store']); // menyimpan user data baru
75     Route::get('/create_ajax',[LevelController::class,'create_ajax']); // menampilkan bentuk form untuk tambah user ajax js 6
76     Route::post('/ajax',[LevelController::class,'store_ajax']); // menyimpan user data baru ajax js 6
77     Route::get('/{id}',[LevelController::class,'show']); // menampilkan detail user
78     Route::get('/{id}/edit',[LevelController::class,'edit']); // menampilkan halaman form edit user
79     Route::put('/{id}',[LevelController::class,'update']); // menyimpan perubahan data user
80     Route::get('/{id}/edit_ajax',[LevelController::class,'edit_ajax']); // menampilkan halaman form edit user ajax js 6
81     Route::put('/{id}/update_ajax',[LevelController::class,'update_ajax']); // menyimpan perubahan data user ajax js 6
82     Route::get('/{id}/delete_ajax',[LevelController::class,'confirm_ajax']); // menghapus data user ajax js 6
83     Route::delete('/{id}/delete_ajax',[LevelController::class,'delete_ajax']); // menghapus data user ajax js 6
84     Route::delete('/{id}',[LevelController::class,'destroy']); // menghapus data user
85 });
86
87 Route::middleware(['authorize:ADM,MNG'])->prefix('kategori')->group(function () {
88     Route::get('/',[KategoriController::class,'index']); // menampilkan halaman awal
89     Route::post('/list',[KategoriController::class,'list']); // menampilkan data user bentuk json / datatable
90     Route::get('/create',[KategoriController::class,'create']); // menampilkan bentuk form untuk tambah user
91     Route::post('/',[KategoriController::class,'store']); // menyimpan user data baru
92     Route::get('/create_ajax',[KategoriController::class,'create_ajax']); // menampilkan bentuk form untuk tambah user ajax js 6
93     Route::post('/ajax',[KategoriController::class,'store_ajax']); // menyimpan user data baru ajax js 6
94     Route::get('/{id}',[KategoriController::class,'show']); // menampilkan detail user
95     Route::get('/{id}/edit',[KategoriController::class,'edit']); // menampilkan halaman form edit user
96     Route::put('/{id}',[KategoriController::class,'update']); // menyimpan perubahan data user
97     Route::get('/{id}/edit_ajax',[KategoriController::class,'edit_ajax']); // menampilkan halaman form edit user ajax js 6
98     Route::put('/{id}/update_ajax',[KategoriController::class,'update_ajax']); // menyimpan perubahan data user ajax js 6
99     Route::get('/{id}/delete_ajax',[KategoriController::class,'confirm_ajax']); // menghapus data user ajax js 6
100     Route::delete('/{id}/delete_ajax',[KategoriController::class,'delete_ajax']); // menghapus data user ajax js 6
101     Route::delete('/{id}',[KategoriController::class,'destroy']); // menghapus data user
102 });
```



```
105 Route::middleware(['authorize:ADM,PMO'])->prefix('barang')->group(function () {
106     Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal
107     Route::post('/list', [BarangController::class, 'list']); // menampilkan data user bentuk json / datatable
108     Route::get('/create', [BarangController::class, 'create']); // menampilkan bentuk form untuk tambah user
109     Route::post('/', [BarangController::class, 'store']); // menyimpan user data baru
110     Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // menampilkan bentuk form untuk tambah user ajax js &
111     Route::post('/ajax', [BarangController::class, 'store_ajax']); // menyimpan user data baru ajax js &
112     Route::get('/{id}', [BarangController::class, 'show']); // menampilkan detail user
113     Route::get('/{id}/edit', [BarangController::class, 'edit']); // menampilkan halaman form edit user
114     Route::put('/{id}', [BarangController::class, 'update']); // menyimpan perubahan data user
115     Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // menampilkan halaman form edit user ajax js &
116     Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // menyimpan perubahan data user ajax js &
117     Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // menghapus data user ajax js &
118     Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // menghapus data user ajax js &
119     Route::delete('/{id}', [BarangController::class, 'destroy']); // menghapus data user
120 });
121
122 Route::middleware(['authorize:PMO'])->prefix('supplier')->group(function () {
123     Route::get('/', [SupplierController::class, 'index']); // menampilkan halaman awal
124     Route::post('/list', [SupplierController::class, 'list']); // menampilkan data user bentuk json / datatable
125     Route::get('/create', [SupplierController::class, 'create']); // menampilkan bentuk form untuk tambah user
126     Route::post('/', [SupplierController::class, 'store']); // menyimpan user data baru
127     Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); // menampilkan bentuk form untuk tambah user
128     Route::post('/ajax', [SupplierController::class, 'store_ajax']); // menyimpan user data baru
129     Route::get('/{id}', [SupplierController::class, 'show']); // menampilkan detail user
130     Route::get('/{id}/edit', [SupplierController::class, 'edit']); // menampilkan halaman form edit user
131     Route::put('/{id}', [SupplierController::class, 'update']); // menyimpan perubahan data user
132     Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); // menampilkan halaman form edit user
133     Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']); // menyimpan perubahan data user
134     Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']); // menghapus data user
135     Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']); // menghapus data user
136     Route::delete('/{id}', [SupplierController::class, 'destroy']); // menghapus data user
137 });
```

4. Submit kode untuk implementasi Authorization pada repository github kalian.

#### Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

Web.php

```
45 Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
46 //regis
47 Route::get('register', [AuthController::class, 'register'])->name('register');
48 Route::post('register', [AuthController::class, 'store_user']);
49
```

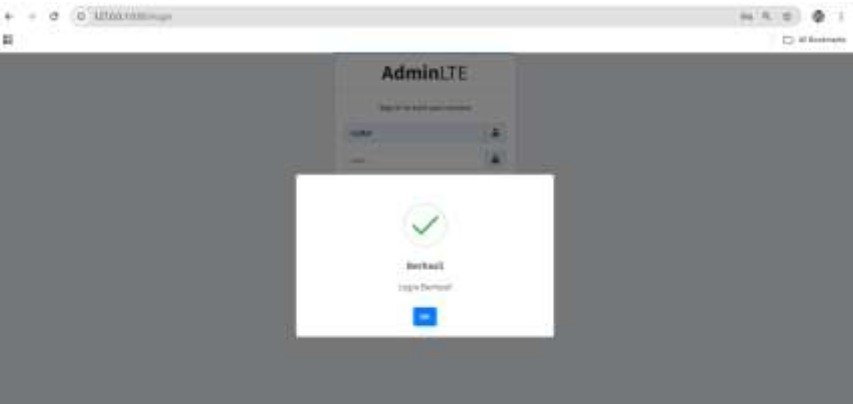

AuthController

```
53
54 public function register(){
55     $level = LevelModel::select('level_id', 'level_nama')->get();
56
57     return View('auth.register')
58     ->with('level', $level);
59 }
60
61 public function store_user(Request $request){
62     $request->validate([
63         'username' => 'required|string|min:3|unique:m_user,username',
64         'nama' => 'required|string|max:100',
65         'password' => 'required|min:5',
66         'level_id' => 'required|integer',
67     ]);
68
69     UserModel::create([
70         'username' => $request->username,
71         'nama' => $request->nama,
72         'password' => bcrypt($request->password), // enkripsi pass
73         'level_id' => $request->level_id,
74     ]);
75
76     return redirect('/')->with('success ', ' Registrasi berhasil');
77 }
78 }
```



## Register.blade

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <meta name="csrf-token" content="{{ csrf_token() }}">
8     <title>Registrasi Pengguna</title>
9
10    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
11    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
12    <link rel="stylesheet" href="{{ asset('adminlte/plugins/icomoon-bootstrap/icomoon-bootstrap.min.css') }}">
13    <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
14    <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
15 </head>
16
17 <body class="hold-transition login-page">
18     <div class="login-box">
19         <div class="card card-outline card-primary">
20             <div class="card-header text-center">
21                 <a href="{{ url('/') }}"> <h3>AdminLTE</h3>
22             </div>
23             <div class="card-body">
24                 <p class="login-box-msg">Daftar Pengguna Baru</p>
25                 <form action="{{ url('register') }}" method="POST" id="form-register">
26                     @csrf
27                     <div class="form-group">
28                         <label>Pilih Role</label>
29                         <select name="level_id" id="level_id" class="form-control" required>
30                             <option value="">Pilih Level </option>
31                             @foreach($levels as $l)
32                                 <option value="{{ $l->level_id }}">{{ $l->level_name }}</option>
33                             @endforeach
34                         </select>
35                     </div>
36                     <div class="form-group">
37                         <input type="text" class="form-control" value="Nama" />
38                     </div>
39                     <div class="form-group">
40                         <input type="password" class="form-control" value="Password" />
41                     </div>
42                     <div class="form-group">
43                         <input type="password" class="form-control" value="Konfirmasi Password" />
44                     </div>
45                     <div class="form-group">
46                         <button type="submit" class="btn btn-primary">Daftar</button>
47                     </div>
48                 </form>
49             </div>
50         </div>
51     </div>
52 </body>
53 </html>
```



2. Screenshot hasil yang kalian kerjakan
3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian

\*\*\* Sekian, dan selamat belajar \*\*\*