

LAPORAN JOBSHEET 7

AUTHENTICATION dan AUTHORIZATION di Laravel

MATA KULIAH PEMROGRAMAN WEB LANJUT

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.



Disusun oleh :

Dahniar Davina SIB-2A / 2341760023

JURUSAN TEKNOLOGI INFORMASI

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

POLITEKNIK NEGERI MALANG

2025

A. 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\UserModel::class,  
66         ],  
67     ],
```

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

```
app > Models > UserModel.php > ...  
1  <?php  
2  
3  namespace App\Models;  
4  
5  use Illuminate\Database\Eloquent\Model;  
6  use Illuminate\Database\Eloquent\Factories\HasFactory;  
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;  
8  use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class /  
9  
10 class UserModel extends Authenticatable  
11 {  
12     use HasFactory;  
13  
14     protected $table = 'm_user';  
15     protected $primaryKey = 'user_id';  
16     protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at'];  
17  
18     protected $hidden = ['password']; // jangan di tampilkan saat select  
19  
20     protected $casts = ['password' => 'hashed']; // casting password agar otomatis  
21  
22     /**  
23      * Relasi ke tabel level  
24      */  
25     public function level(): BelongsTo  
26     {  
27         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
28     }  
29 }  
30
```

3. Selanjutnya kita buat AuthController.php untuk memproses login yang akan kita lakukan

```
app > Http > Controllers > AuthController.php > AuthController  
1  <?php  
2  
3  namespace App\Http\Controllers;  
4  
5  use Illuminate\Http\Request;  
6  use Illuminate\Support\Facades\Auth;  
7  
8  class AuthController extends Controller  
9  {  
10     public function login()  
11     {  
12         if (Auth::check()) { // jika sudah login, maka redirect ke halaman  
13             return redirect('/');
```

```

14     }
15     return view('auth.login');
16 }
17
18 public function postlogin(Request $request)
19 {
20     if ($request->ajax() || $request->wantsJson()) {
21         $credentials = $request->only('username', 'password');
22
23         if (Auth::attempt($credentials)) {
24             return response()->json([
25                 'status' => true,
26                 'message' => 'Login Berhasil',
27                 'redirect' => url('/')
28             ]);
29         }
30     }

```

- 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)

```

resources > views > auth > login.blade.php > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Login Pengguna</title>
7
8     <!-- Google Font: Source Sans Pro -->
9     <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:400,700,900">
10
11     <!-- Font Awesome -->
12     <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
13
14     <!-- iCheck bootstrap -->
15     <link rel="stylesheet" href="{{ asset('adminlte/plugins/checkbox-bootstrap/checkbox-bootstrap.min.css') }}">
16
17     <!-- SweetAlert2 -->
18     <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap/sweetalert2-theme-bootstrap.min.css') }}">
19

```

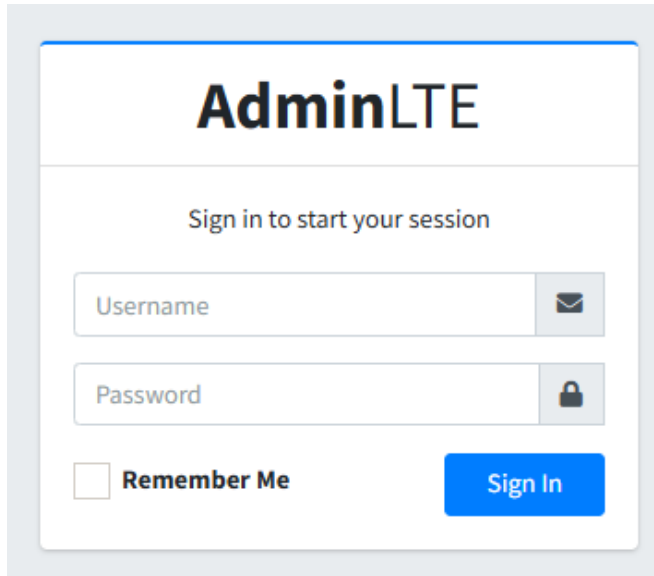
- Kemudian kita modifikasi route/web.php agar semua route masuk dalam auth

```

5 use App\Http\Controllers\BarangController;
6 use App\Http\Controllers\AuthController;
7 use App\Http\Controllers\KategoriController;
8 use App\Http\Controllers\LevelController;
9 use App\Http\Controllers\WelcomeController;
10 use Illuminate\Support\Facades\Route;
11
12 Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter {id}, maka harus berupa angka
13
14 Route::get('login', [AuthController::class, 'login'])->name('login');
15 Route::post('login', [AuthController::class, 'postlogin']);
16 Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
17
18 Route::middleware(['auth'])->group(function() { // artinya semua route di dalam

```

6. Ketika kita coba mengakses halaman localhost/PWL_POS/public maka akan tampil halaman awal untuk login ke aplikasi

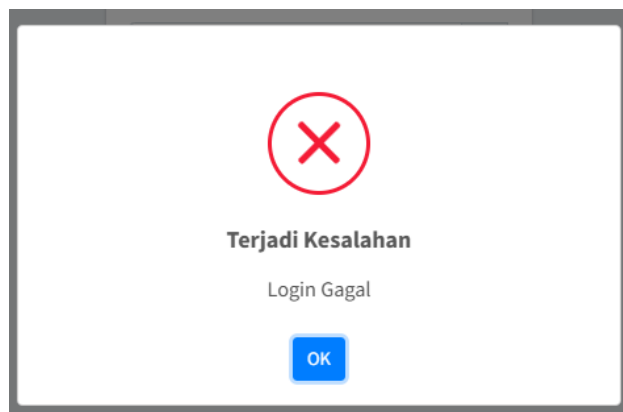


The image shows the AdminLTE login interface. It features a white login box with a blue border. At the top, the text "AdminLTE" is displayed in a large, bold, black font. Below this, the instruction "Sign in to start your session" is centered. The form contains two input fields: "Username" and "Password", each with a corresponding icon (an envelope for username and a padlock for password) to its right. Below the password field is a checkbox labeled "Remember Me". A blue "Sign In" button is positioned to the right of the "Remember Me" checkbox.

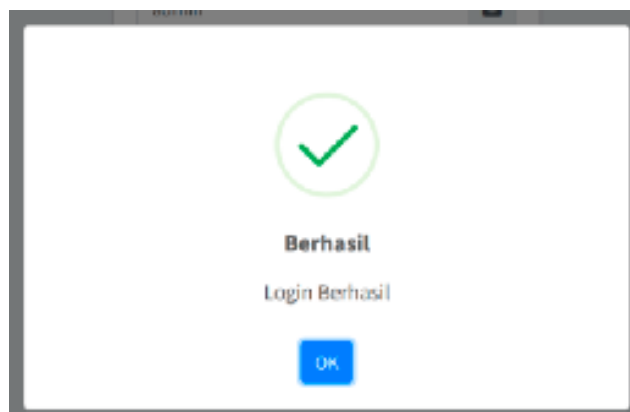
➤ **TUGAS 1 = Implementasi Authentication**

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

- Jika username dan password tidak sesuai



- Jika username dan password sesuai



- ```
134
135 <li class="nav-item">
136 <a href="#" class="nav-link text-danger"
137 onclick="event.preventDefault(); document.getElementById('logout-f
138 <i class="fas fa-sign-out-alt"></i> Logout
139
140 <form id="logout-form" action="{ { route('logout') } }" method="POST" s
141 @csrf
142 </form>
143
144
```

```
Route::post('/logout', [AuthController::class, 'logout'])->middleware('auth')->name('logout');
```



- ## B. PRAKTIKUM 2 : Implementasi Authorization di Laravel dengan Middleware

- ```

18 protected $casts = ['password' => 'hashed']; // casting password agar otomatis
19
20
21 /**
22  * Relasi ke tabel level
23  */
24 public function level(): BelongsTo
25 {
26     return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
27 }
28
29 /**
30  * Mendapatkan nama role
31  */
32 public function getRoleName(): string
33 {
34     return $this->level->level_nama;
35 }
36
37 /**
38  * Cek apakah user memiliki role tertentu
39  */
40 public function hasRole($role): bool
41 {
42     return $this->level->level_kode == $role;
43 }
44

```

- ```
PS C:\laragon\www\PWL_2025\Minggu7\Jobsheet7> php artisan make:middleware AuthorizeUser
```
- INFO** Middleware [C:\laragon\www\PWL\_2025\Minggu7\Jobsheet7\app\Http\Middleware\AuthorizeUser.php] created successfully.

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

```
app > Http > Middleware > AuthorizeUser.php > AuthorizeUser
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
15 */
16 public function handle(Request $request, Closure $next, $role = ''): Response
17 {
18 $user = $request->user(); // ambil data user yg login dari UserModel.php
19
20 if($user->hasRole($role)) { // cek apakah user punya role yg diinginkan
21 return $next($request);
22 }
23
24 // jika tidak punya role, maka tampilkan error 403
25 abort(403, 'Forbidden, kamu tidak punya akses ke halaman ini!');
26 }
27 }
```

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

```
'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg k
```

5. Sekarang kita perhatikan tabel m\_level yang menjadi tabel untuk menyimpan level/group/role dari user ada

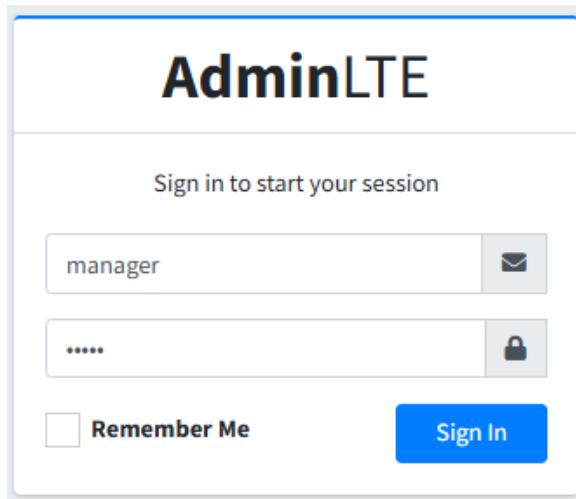
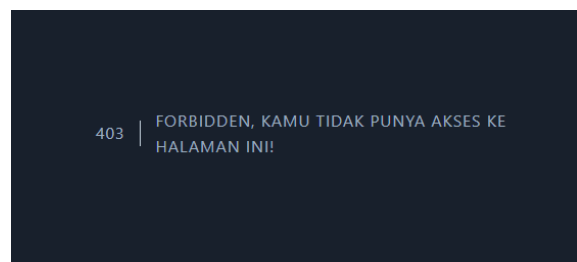
|                                           | level_id | level_kode | level_nama    | created_at          | updated_at |
|-------------------------------------------|----------|------------|---------------|---------------------|------------|
| <input type="checkbox"/> Edit Copy Delete | 1        | ADM        | Administrator | NULL                | NULL       |
| <input type="checkbox"/> Edit Copy Delete | 2        | MNG        | Manager       | NULL                | NULL       |
| <input type="checkbox"/> Edit Copy Delete | 3        | STF        | Staff/Kasir   | NULL                | NULL       |
| <input type="checkbox"/> Edit Copy Delete | 5        | CUS        | Pelanggan     | 2025-03-05 10:18:30 | 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

```
// 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
 Route::get('/level/create', [LevelController::class, 'create']);
 Route::post('/level', [LevelController::class, 'store']);
 Route::get('/level/{id}/edit', [LevelController::class, 'edit']); // u
 Route::put('/level/{id}', [LevelController::class, 'update']); // untu
 Route::delete('/level/{id}', [LevelController::class, 'destroy']); //
});
```

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

The image shows the login interface of AdminLTE. At the top, the text "AdminLTE" is displayed in a large, bold font. Below it, a subtitle reads "Sign in to start your session". There are two input fields: the first is for the username, containing the text "manager", and the second is for the password, represented by dots. To the right of each input field is an icon (an envelope for the username and a padlock for the password). Below the password field, there is a checkbox labeled "Remember Me" and a blue button labeled "Sign In".

## ➤ TUGAS 2 = Implementasi Authentication

1. Apa yang kalian pahami pada praktikum 2 ini?
  - Penerapan Authentication diatas membatasi user dengan kode "ADM" untuk megakses halaman "Level User" pada website.
  - Autentikasi memastikan hanya pengguna yang sah (dengan ketentuan yang benar, seperti username dan password tertentu) yang dapat mengakses sistem atau bagian tertentu dari aplikasi.
  - Setelah autentikasi berhasil, sistem dapat mengidentifikasi pengguna dan memberikan pengalaman yang dipersonalisasi, seperti menampilkan data spesifik pengguna (misalnya, dashboard khusus untuk user dengan User ID selain ADM).
  - Autentikasi mencegah akses tidak sah ke data sensitif.
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Submit kode untuk impementasi Authorization pada repository github kalian.

### C. PRAKTIKUM 3 : Implementasi Multi-Level Authorization dengan Middleware

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

```
public function getRole()
{
 return $this->level->level_kode;
}
```

2. Selanjutnya, Kita modifikasi middleware AuthorizeUser.php dengan kode Berikut

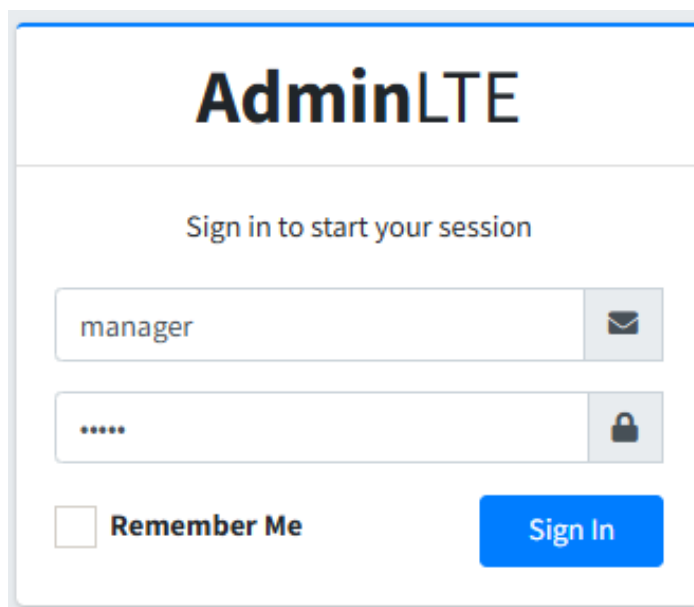
```
public function handle(Request $request, Closure $next, ...$roles): Response
{
 $user_role = $request->user()->getRole(); // Ambil data level_kode dari user yang login

 if (in_array($user_role, $roles)) { // Cek apakah level_kode user ada di dalam array roles
 return $next($request); // Jika ada, maka lanjutkan request
 }
}
```

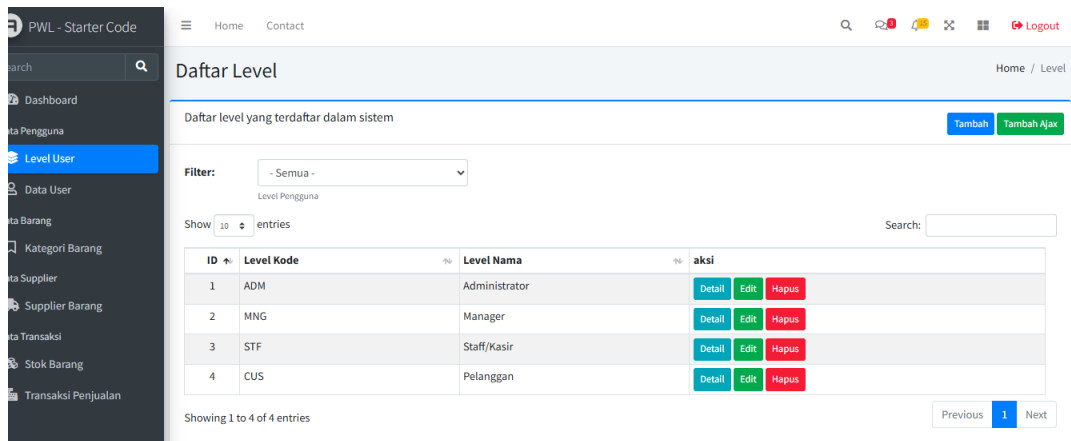
3. Setelah itu tinggal kita perbaiki route/web.php sesuaikan dengan role/level yang diinginkan. Contoh

```
// 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







### ➤ TUGAS 3 = Implementasi Multi-Level Authentication

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu menu yang sesuai dengan Level/Jenis User
  - Semua user dapat akses jika sudah login

```
// Semua rute di bawah ini hanya bisa diakses jika sudah login
Route::middleware(['authorize:ADM'])->group(function(){
 Route::group(['prefix' => 'user'], function () {
 Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user
 Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
 Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
 Route::post('/', [UserController::class, 'store']); // menyimpan data user baru
 //Create Menggunakan AJAX
 Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
 Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax
 Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user
 Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user
 Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
 //Edit Menggunakan AJAX
 Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax
 Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax
 //Delete Menggunakan AJAX
 Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax
 Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // Untuk hapus data user Ajax
 Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
 });
});
```

- Semua Admin memiliki semua pengaturan dalam sistem

```
// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function(){
 Route::group(['prefix' => 'level'], function () {
 Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal level
 Route::post('/list', [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
 Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level
 Route::post('/', [LevelController::class, 'store']); // menyimpan data level baru
 //Create Menggunakan AJAX
 Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah level Ajax
 Route::post('/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data level baru Ajax
 Route::get('/{id}', [LevelController::class, 'show']); // menampilkan detail level
 Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman form edit level
 Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan data level
 //Edit Menggunakan AJAX
 Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit level Ajax
 Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan data level Ajax
 //Delete Menggunakan AJAX
 Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete level Ajax
 Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Untuk hapus data level Ajax
 Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus data level
 });
});
```

- Semua Admin dan Manager memiliki semua pengaturan dalam sistem

```
// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function(){
 Route::group(['prefix' => 'kategori'], function () {
 Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal kategori
 Route::post('/list', [KategoriController::class, 'list']); // menampilkan data kategori dalam bentuk json untuk datatables
 Route::get('/create', [KategoriController::class, 'create']); // menampilkan halaman form tambah kategori
 Route::post('/', [KategoriController::class, 'store']); // menyimpan data kategori baru
 // Create menggunakan AJAX
 Route::get('/create_ajax', [KategoriController::class, 'create_ajax']); // menampilkan halaman form tambah kategori ajax
 Route::post('/ajax', [KategoriController::class, 'store_ajax']); // menyimpan data kategori baru ajax
 Route::get('/{id}', [KategoriController::class, 'show']); // menampilkan detail kategori
 Route::get('/{id}/edit', [KategoriController::class, 'edit']); // menampilkan halaman form edit kategori
 Route::put('/{id}', [KategoriController::class, 'update']); // menyimpan perubahan data kategori
 // Edit menggunakan AJAX
 Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']); // menampilkan halaman form edit kategori ajax
 Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']); // menyimpan perubahan data kategori ajax
 // Delete menggunakan AJAX
 Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']); //menampilkan form confirm delete kategori ajax
 Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']); // menghapus data kategori ajax
 Route::delete('/{id}', [KategoriController::class, 'destroy']); // menghapus data kategori
 });
});
```

- Staff hanya dapat melihat data barang

```
// Staff hanya bisa melihat data barang
Route::middleware(['authorize:ADM,MNG,STF'])->group(function(){
 Route::group(['prefix' => 'barang'], function () {
 Route::get('/', [BarangController::class, 'index']); // Menampilkan daftar barang
 Route::post('/list', [BarangController::class, 'list']); // Menampilkan data barang dalam bentuk JSON untuk datatables
 Route::get('/{id}', [BarangController::class, 'show']); // Menampilkan detail barang
 });
});
```

- Admin dan Manager dapat mengelola table barang

```
// ADM & MNG bisa menambah, mengedit, dan menghapus barang
Route::middleware(['authorize:ADM,MNG'])->group(function(){
 Route::group(['prefix' => 'barang'], function () {
 Route::get('/create', [BarangController::class, 'create']); // Form tambah barang
 Route::post('/', [BarangController::class, 'store']); // Simpan barang baru
 // Create menggunakan AJAX
 Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // Form tambah barang AJAX
 Route::post('/ajax', [BarangController::class, 'store_ajax']); // Simpan barang baru AJAX
 Route::get('/{id}/edit', [BarangController::class, 'edit']); // Form edit barang
 Route::put('/{id}', [BarangController::class, 'update']); // Simpan perubahan barang
 // Edit menggunakan AJAX
 Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // Form edit barang AJAX
 Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // Simpan perubahan barang AJAX
 // Delete menggunakan AJAX
 Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // Form konfirmasi hapus barang AJAX
 Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // Hapus barang AJAX
 Route::delete('/{id}', [BarangController::class, 'destroy']); // Hapus barang
 });
});
```

- Admin dan Manager dapat mengelola table supplier

```
// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager) supplier
Route::middleware(['authorize:ADM,MNG'])->group(function(){
 Route::group(['prefix' => 'supplier'], function () {
 Route::get('/', [SupplierController::class, 'index']);
 Route::post('/list', [SupplierController::class, 'list']);
 Route::get('/create', [SupplierController::class, 'create']);
 Route::post('/', [SupplierController::class, 'store']);
 // Create menggunakan AJAX
 Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); // menampilkan halaman form tambah Supplier ajax
 Route::post('/ajax', [SupplierController::class, 'store_ajax']); // menyimpan data Supplier baru ajax
 Route::get('/{id}', [SupplierController::class, 'show']);
 Route::get('/{id}/edit', [SupplierController::class, 'edit']);
 Route::put('/{id}', [SupplierController::class, 'update']);
 // Edit menggunakan AJAX
 Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); // menampilkan halaman form edit Supplier ajax
 Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']); // menyimpan perubahan data Supplier ajax
 // Delete menggunakan AJAX
 Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']); //menampilkan form confirm delete Supplier ajax
 Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']); // menghapus data Supplier ajax
 Route::delete('/{id}', [SupplierController::class, 'destroy']);
 });
});
```

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

## D. TUGAS 4 = Implementasi Form Registrasi

1. Silahkan implementasikan form untuk registrasi user.

a) Modifikasi AuthController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\UserModel; // Gunakan UserModel

public function register()
{
 return view('auth.register');
}

public function postregister(Request $request)
{
 try {
 if ($request->ajax() || $request->wantsJson()) {
 $validator = Validator::make($request->all(), [
 'username' => 'required|min:4|max:20|unique:m_user,username', // Sesuaikan dengan tabel m_user
 'nama' => 'required|min:3|max:50',
 'password' => 'required|min:5|max:20|confirmed'
]);

 if ($validator->fails()) {
 return response()->json([
 'status' => false,
 'message' => 'Validasi gagal',
 'msgField' => $validator->errors()
]);
 }

 // Simpan data pengguna baru
 UserModel::create([
 'username' => $request->username,
 'nama' => $request->nama,
 'password' => $request->password, // Otomatis di-hash karena casting di model
 'level_id' => 2, // Misalnya, level default untuk pengguna baru (sesuaikan dengan ID level)
]);

 return response()->json([
 'status' => true,
 'message' => 'Registrasi Berhasil! Silakan login.',
 'redirect' => url('/login')
]);
 }
 } catch (\Throwable $th) {
 return response()->json([
 'status' => false,
 'message' => 'Terjadi kesalahan server: ' . $th->getMessage()
]);
 }

 return redirect('/register');
}
```

b) Modifikasi route/web.php

```
// Route untuk registrasi (di luar middleware auth, agar bisa diakses tanpa login)
Route::get('/register', [AuthController::class, 'register'])->name('register');
Route::post('/register', [AuthController::class, 'postregister']);
```

c) Modifikasi login.blade.php

```
<!-- Tambahkan tautan ke halaman registrasi -->
<p class="mb-0 mt-3">
 Don't have an account? Register
</p>
```

d) Menambahkan view/register.blade.php

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

 <!-- Google Font: Source Sans Pro -->
 <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,4
00i,700&display=fallback">

 <!-- Font Awesome -->
 <link rel="stylesheet" href="{{
asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">

 <!-- icheck bootstrap -->
 <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-
bootstrap/icheck-bootstrap.min.css') }}">

 <!-- SweetAlert2 -->
 <link rel="stylesheet" href="{{
asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-
4.min.css') }}">

 <!-- Theme style -->
 <link rel="stylesheet" href="{{
asset('adminlte/dist/css/adminlte.min.css') }}">
 <meta name="csrf-token" content="{{ csrf_token() }}">
</head>
<body class="hold-transition register-page">
 <div class="register-box">
 <div class="card card-outline card-primary">
 <div class="card-header text-center">
 AdminLTE
 </div>
 <div class="card-body">
 <p class="login-box-msg">Register a new account</p>

 <form action="{{ url('/register') }}" method="POST"
id="form-register">
 @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">

 </div>
 </div>
 <small id="error-username" class="error-text
text-danger"></small>
 </div>

 <div class="input-group mb-3">
 <input type="text" id="nama" name="nama"
class="form-control" placeholder="Full Name">
 <div class="input-group-append">
 <div class="input-group-text">

 </div>
 </div>
 <small id="error-nama" 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">

 </div>
 </div>
 <small id="error-password" class="error-text
text-danger"></small>
 </div>

 <div class="input-group mb-3">
 <input type="password"
id="password_confirmation" name="password_confirmation" class="form-
control" placeholder="Confirm Password">
 <div class="input-group-append">
 <div class="input-group-text">

 </div>
 </div>
 <small id="error-password_confirmation"
class="error-text text-danger"></small>
 </div>

 <div class="row">
 <div class="col-12">

```

```

 <button type="submit" class="btn btn-
primarysoft.com/images/2024/10/14/672f04ae6de4e_image.png

 <button type="submit" class="btn btn-
primary btn-block">Register</button>
 </div>
</div>
</form>

 <p class="mb-0 mt-3">
 <a href="{{ url('/login') }}" class="text-
center">Already have an account? Login
 </p>
</div>
</div>
</div>

<!-- jQuery -->
<script src="{{ asset('adminlte/plugins/jquery/jquery.min.js')
}}"></script>

<!-- Bootstrap 4 -->
<script src="{{
asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js')
}}"></script>

<!-- jquery-validation -->
<script src="{{ asset('adminlte/plugins/jquery-
validation/jquery.validate.min.js') }}"></script>
<script src="{{ asset('adminlte/plugins/jquery-
validation/additional-methods.min.js') }}"></script>

<!-- SweetAlert2 -->
<script src="{{
asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>

<!-- AdminLTE App -->
<script src="{{ asset('adminlte/dist/js/adminlte.min.js')
}}"></script>

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

```

```

$(document).ready(function() {
 $("#form-register").validate({
 rules: {
 username: { required: true, minlength: 4,
maxlength: 20 },
 nama: { required: true, minlength: 3, maxlength: 50
},
 password: { required: true, minlength: 5,
maxlength: 20 },
 password_confirmation: { required: true, equalTo:
"#password" }
 },
 submitHandler: function(form) {
 $.ajax({
 url: form.action,
 type: form.method,
 data: $(form).serialize(),
 success: function(response) {
 if (response.status) {
 Swal.fire({
 icon: 'success',
 title: 'Berhasil',
 text: response.message,
 }).then(function() {
 window.location =
response.redirect;
 });
 } else {
 $('#error-text').text('');
 $.each(response.msgField,
function(prefix, val) {
 $('#error-' + prefix).text(val[0]);
 });
 Swal.fire({
 icon: 'error',
 title: 'Terjadi Kesalahan',
 text: response.message
 });
 }
 }
 });
 return false;
 },
 errorElement: 'span',
 errorPlacement: function(error, element) {
 error.addClass('invalid-feedback');
 element.closest('.input-group').append(error);
 },
 });

```

```

highlight: function(element) {
 $(element).addClass('is-invalid');
},
unhighlight: function(element) {
 $(element).removeClass('is-invalid');
}
});
});
</script>
</body>
</html>

```

e) Tampilan Login

**AdminLTE**

Sign in to start your session

Username

Password

☐ **Remember Me**

[Don't have an account? Register](#)

**AdminLTE**

Register a new account

staff2

cio

\*\*\*\*\*

\*\*\*\*\*

[Already have an account? Login](#)

**Berhasil**

Registrasi Berhasil! Silakan login.

33 2 staff2 cio \$2y\$12\$0J3ly/Y4who41yJgje1fcOSmb93w/YtsqoKcDkrY7... 2025-04-15 02:57:40 2025-04-15 02:57:40

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