

**TUGAS**  
**Analisis Desain**

**PERTEMUAN 7**

Authentication dan  
Authorization

**OLEH**

**MUHAMMAD NUR AZIZ NIM. 2341720237**



**PROGRAM STUDI D-IV TEKNIK INFORMATIKA**  
**JURUSAN TEKNOLOGI INFORMASI**  
**POLITEKNIK NEGERI MALANG**  
**31 Maret 2025**

## Praktikum 1 – Implementasi Authentication:

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

```
'providers' => [  
    'users' => [  
        'driver' => 'eloquent',  
        'model' => App\Models\UserModel::class,  
    ],  
],
```

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

```
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
  
class UserModel extends Authenticatable  
{  
    //praktikum week 3  
    use HasFactory;  
  
    protected $table = 'm_user'; // mendefinisikan tabel yang akan digunakan  
    protected $primaryKey = 'user_id'; // mendefinisikan primary key  
  
    //praktikum week 4  
    protected $fillable = ['level_id', 'username', 'nama', 'password', 'created_at', 'updated_at'];  
    //praktikum week 7b  
    protected $hidden = ['password']; // jangan di tampilkan saat select  
    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash  
  
    public function level(): BelongsTo  
    {  
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
    }  
}
```

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;

Windsurf: Refactor | Explain
class AuthController extends Controller
{
    Windsurf: Refactor | Explain | Generate Function Comment | X
    public function login()
    {
        if (Auth::check()) { // jika sudah Login, maka redirect ke halaman home
            return redirect('/');
        }
        return view('auth.login');
    }

    Windsurf: Refactor | Explain | Generate Function Comment | X
    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');
    }

    Windsurf: Refactor | Explain | Generate Function Comment | X
    public function logout(Request $request)

```

- 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&displa
    <!-- 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/sweetalert2-theme-bootstrap-4.css') }}">
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
</head>

<body class="hold-transition login-page">
    <div class="login-box">
        <!-- /.login-logo -->

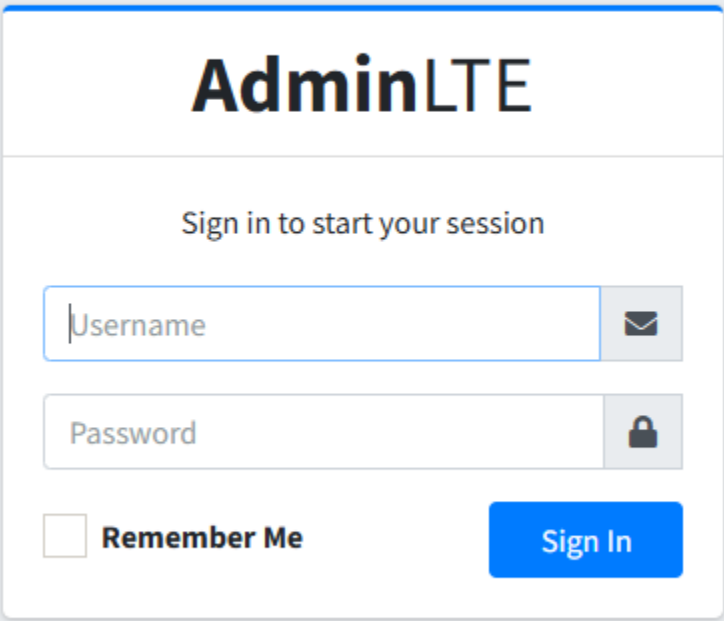
```

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

```
use App\Http\Controllers\WelcomeController;
use App\Http\Controllers\AuthController;

Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter {id}, maka harus berupa angka
Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini harus
    // masukkan semua route yang perlu autentikasi di sini
});
```

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



The image shows a login page for AdminLTE. It features a white card on a light gray background. The card has a header with the text "AdminLTE" in a large, bold, black font. Below the header, the text "Sign in to start your session" is centered. There are two input fields: "Username" with an envelope icon and "Password" with a lock icon. Below the password field is a checkbox labeled "Remember Me". A blue "Sign In" button is located at the bottom right of the card.

## Tugas 1 – Implementasi Authentication:

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

Proses login:

route

```
Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter {id}, maka harus  
  
Route::get('login', [AuthController::class, 'login'])->name('login');  
Route::post('login', [AuthController::class, 'postlogin']);
```

Authcontroller

```
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');  
}
```

Lanjut ke view login

```

});
$(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('');
                    }
                }
            });
        }
    });
});

```

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

Route

```

Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');

Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini

```

Authcontroller

```

Windsurf: Refactor | Explain | Generate Function Comment | X
public function logout(Request $request)
{
    Auth::logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();
    return redirect('login');
}

```

Tambah button logout navbar header

```

</li>
<li class="nav-item">
    <a class="nav-link" href="{{url('logout')}}" role="button">
        <i class="fas fa-sign-out-alt"></i>
    </a>

```

3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
  - Mengatur config/auth.php dengan mengganti user ke UserModel

```
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model' => App\Models\UserModel::class,
    ],
],
```

- Mengatur UserModel dengan menambahkan Auth dengan use Illuminate\Foundation\Auth\User as Authenticatable; dan menambahkan beberapa penyesuaian parameter

```
//praktikum week 4
protected $fillable = ['level_id', 'username', 'nama', 'password', 'created_at', 'updated_at'];
//praktikum week 7b
protected $hidden = ['password']; // jangan di tampilkan saat select
protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
```

- Membuat AuthController, pada AuthController terdapat fungsi untuk login dan logout

```
public function login()
{
    if (Auth::check()) { // jika su

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

- Selanjutnya membuat view dengan membuat folder baru auth dan file login.blade.php

```
views
└── auth
    └── login.blade.php
```

```
<!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=fallback">
    <!-- Font Awesome -->
```

- Membuat route untuk login dan logout, yang nantinya digunakan untuk user login terlebih dahulu kemudian baru memasuki konten utama web dan bisa logout dengan tombol logout disebelah kanan navbar

```
Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter {id}, maka harus berupa angka

Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');

Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini harus login dulu
    // masukkan semua route yang perlu autentikasi di sini
});
```

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

## Praktikum 2 – Implementasi Authorizaton di Laravel dengan Middleware

1. Kita modifikasi UserModel.php dengan menambahkan kode berikut

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

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

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

```
2025-04-10 18:27:12 /favicon.ico ..... ~ 0s
PS C:\laragon\www\smt4\WebLanjut\Week7b\JS7New> php artisan make:middleware AuthorizeUser

INFO Middleware [C:\laragon\www\smt4\WebLanjut\Week7b\JS7New\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

```
Windsurf: Refactor | Explain | X
public function handle(Request $request, Closure $next, $role = ''): Response
{
    $user = $request->user(); // ambil data user yg login
    // fungsi user() diambil dari UserModel.php
    if ($user->hasRole($role)) { // cek apakah user punya role yg diinginkan
        return $next($request);
    }
    // jika tidak punya role, maka tampilkan error 403
    abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

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,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,

```

- 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
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staff/Kasir	NULL	NULL
4	CUS	pelanggan	NULL	NULL
5	TST	TESTT	2025-03-22 08:10:34	2025-03-22 08:11:05

- 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']);
    Route::get('/level/create', [LevelController::class, 'create']);
    Route::post('/level/', [LevelController::class, 'store']);
    Route::get('/level/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
    Route::post('/level/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data user baru Ajax
    Route::get('/level/{id}', [LevelController::class, 'show']);
    Route::get('/level/{id}/show_ajax', [LevelController::class, 'show_ajax']);
    Route::get('/level/{id}/edit', [LevelController::class, 'edit']);
    Route::put('/level/{id}', [LevelController::class, 'update']);
    Route::get('/level/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax
    Route::put('/level/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax
    Route::get('/level/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax
    Route::delete('/level/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Untuk hapus data user Ajax
    Route::delete('/level/{id}', [LevelController::class, 'destroy']);
});

```

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

403 | FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI

## Tugas 2 – Implementasi Authoriization:

1. Apa yang kalian pahami pada praktikum 2 ini?
  - Memodifikasi UserModel untuk menambahkan role masing-masing level
  - Menambahkan middleware authorization
  - Dan memberikan hak access tiap masing-masing level user
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
  - Memodifikasi UserModel untuk menambahkan role masing-masing level

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

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

- Membuat dan Menambahkan middleware authorization pada kernel

```
2025-04-10 18:27:12 /favicon.ico ..... ~ 0s
PS C:\laragon\www\smt4\WebLanjut\Week7b\JS7New> php artisan make:middleware AuthorizeUser

INFO Middleware [C:\laragon\www\smt4\WebLanjut\Week7b\JS7New\app\Http\Middleware\AuthorizeUser.php]
created successfully.
```

```
Windsurf: Refactor | Explain | X
public function handle(Request $request, Closure $next, $role = ''): Response
{
    $user = $request->user(); // ambil data user yg login
    // fungsi user() diambil dari UserModel.php
    if ($user->hasRole($role)) { // cek apakah user punya role yg diinginkan
        return $next($request);
    }
    // jika tidak punya role, maka tampilkan error 403
    abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

```

*/
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,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
];

```

- Membuat hak access tiap role

```

// 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']);
    Route::get('/level/create', [LevelController::class, 'create']);
    Route::post('/level/', [LevelController::class, 'store']);
    Route::get('/level/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
    Route::post('/level/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data user baru Ajax
    Route::get('/level/{id}', [LevelController::class, 'show']);
    Route::get('/level/{id}/show_ajax', [LevelController::class, 'show_ajax']);
    Route::get('/level/{id}/edit', [LevelController::class, 'edit']);
    Route::put('/level/{id}', [LevelController::class, 'update']);
    Route::get('/level/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax
    Route::put('/level/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax
    Route::get('/level/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax
    Route::delete('/level/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Untuk hapus data user Ajax
    Route::delete('/level/{id}', [LevelController::class, 'destroy']);
});

```

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

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

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

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

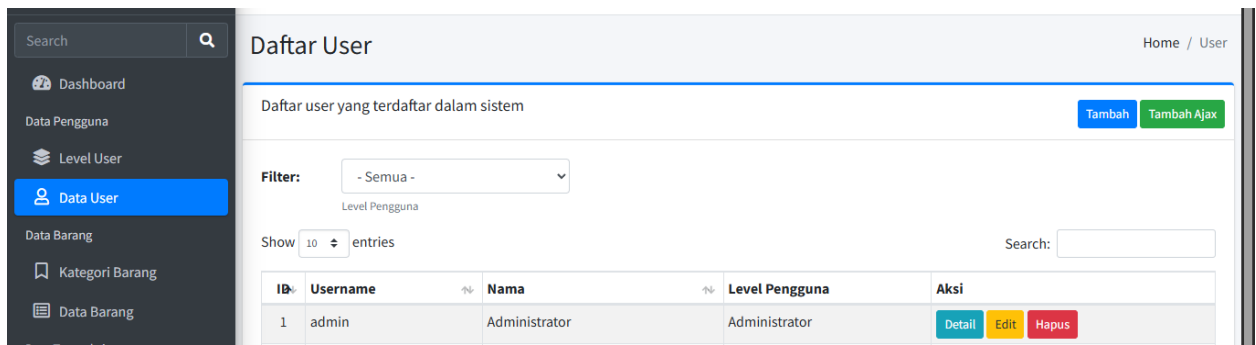
2. Selanjutnya, Kita modifikasi middleware AuthorizeUser.php dengan kode berikut

```
Windsurf Refactor | Explain | X
public function handle(Request $request, Closure $next, ...$roles): Response
{
    $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
    if (in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
        return $next($request); // jika ada, maka lanjutkan request
    }
    // jika tidak punya role, maka tampilkan error 403
    abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

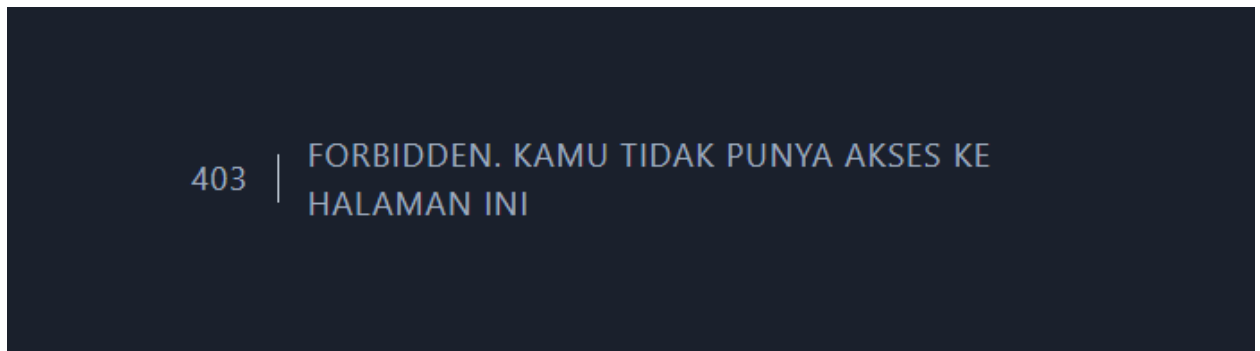
3. Setelah itu tinggal kita perbaiki route/web.php sesuaikan dengan role/level yang diinginkan. Contoh

```
// BARANG
Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
    Route::group(['prefix' => 'barang'], function () {
        Route::get('/', [BarangController::class, 'index']);
        Route::post('/list', [BarangController::class, 'list']);
        Route::get('/create', [BarangController::class, 'create']);
        Route::post('/', [BarangController::class, 'store']);
        Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
        Route::post('/ajax', [BarangController::class, 'store_ajax']); // Menyimpan data user baru Ajax
        Route::get('/{id}', [BarangController::class, 'show']);
        Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax']);
        Route::get('/{id}/edit', [BarangController::class, 'edit']);
        Route::put('/{id}', [BarangController::class, 'update']);
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax
        Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // Untuk hapus data user Ajax
        Route::delete('/{id}', [BarangController::class, 'destroy']);
    });
});
```

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



Role manager akses data user



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

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
  - Multilevel pada kategori dengan role admin dan manager

```

5
6 // KATEGORI
7 Route::middleware(['authorize:ADM,MNG'])->group(function () {
8     Route::group(['prefix' => 'kategori'], function () {
9         Route::get('/', [KategoriController::class, 'index']);

```

2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
  - Modifikasi user model dengan menambahkan getrole

```

/**
 * Mendapatkan kode role
 */
Windsurf: Refactor | Explain | X
public function getRole()
{
    return $this->level->level_kode;
}

```

- Modifikasi authorization untuk multilevel

```

public function handle(Request $request, Closure $next, ...$roles): Response
{
    $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
    if (in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
        return $next($request); // jika ada, maka lanjutkan request
    }
    // jika tidak punya role, maka tampilkan error 403
    abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
}

```

- Modifikasi route untuk menyesuaikan tiap hak access tiap role
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu yang sesuai dengan Level/Jenis User
    - Level, user untuk role admin

```

// LEVEL
// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function () {
    Route::get('/level/{level_id}', [LevelController::class, 'index']);
});

// USER
Route::middleware(['authorize:ADM'])->group(function () {
    Route::get('/user/{user_id}', [UserController::class, 'index']);
});

```

- Kategori, supplier untuk role admin, manager

```

// KATEGORI
Route::middleware(['authorize:ADM,MNG'])->group(function () {
    Route::get('/kategori/{kategori_id}', [KategoriController::class, 'index']);
});

// SUPPLIER
Route::middleware(['authorize:ADM,MNG'])->group(function () {
    Route::get('/supplier/{supplier_id}', [SupplierController::class, 'index']);
});

```

- Barang untuk role admin, manager, staf

```

// BARANG
Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
    Route::get('/barang/{barang_id}', [BarangController::class, 'index']);
});

```

4. Submit kode untuk implementasi Authorization pada repository github kalian

#### Tugas 4 – Implementasi Form Registrasi:

1. Silahkan implementasikan form untuk registrasi user.

2. Screenshot hasil yang kalian kerjakan

AdminLTE

Register a new account

Full Name

Username

Password

Re-type Password

- Pilih Level -

Register

Sudah terdaftar? [Login disini](#)

Modifikasi authcontroller

```

Windsurf: Refactor | Explain | Generate Function Comment | ✕
public function postregister(Request $request)
{
    if ($request->ajax() || $request->wantsJson()) {
        $rules = [
            'level_id' => 'required|integer',
            'username' => 'required|string|min:3|unique:m_user,username',
            'nama' => 'required|string|max:100',
            'password' => 'required|min:5'
        ];

        $validator = Validator::make($request->all(), $rules);

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

        UserModel::create($request->all());

        return response()->json([
            'status' => true,
            'message' => 'Data user berhasil disimpan',
            'redirect' => url('/login'),
        ]);
    }
}

```

Modifikasi login.blade

```

        </div>
    </form>
    <p class="mb-0 mt-4">
        New User? <a href="{{ url('register') }}" class="text-center">Register Here</a>
    </p>
</div>

```

Buat view register

```

resources > views > auth > register.blade.php > html > body.hold-transition.register-page > div.register-box > div.card.card-outline.card-primary >
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">
7     <title>Register Pengguna</title>
8
9     <!-- Google Font: Source Sans Pro -->
10    <link rel="stylesheet"
11        href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
12    <!-- Font Awesome -->
13    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">

```

Tambahkan route register

```

Route::get('register', [AuthController::class, 'register'])->name('register');
Route::post('register', [AuthController::class, 'postregister']);

```

3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian