

**LAPORAN PRAKTIKUM**  
**PEMROGRAMAN WEB LANJUT**  
**AUTHENTICATION DAN AUTHORIZATION DI LARAVEL**



**Disusun Oleh :**

Qusnul Diah Mawanti

2341760035

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS**  
**JURUSAN TEKNOLOGI INFORMASI**  
**POLITEKNIK NEGERI MALANG**

**2025**



Jurusan Teknologi Informasi – Politeknik Negeri  
Malang  
Jobsheet 7 : Authentication dan Authorization di Laravel  
Mata Kuliah : Pemrograman Web Lanjut  
Kelas : 2A – SIB  
Nama : Qusnul Diah Mawanti  
No absen : 25  
NIM : 2341760035

April 2025

---

## PERSIAPAN

### 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

```
C:\laragon\www>composer create-project "laravel/laravel:^10.0" PWL_POS
Creating a "laravel/laravel:^10.0" project at "./PWL_POS"
Installing laravel/laravel (v10.3.3)
- Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\laragon\www\PWL_POS
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
- Locking brick/math (0.12.3)
```

## PRAKTIKUM

### 1. Praktikum 1: Implementasi Authentication

- a. 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         ],
```

- b. 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 Authenticatable  
9  use App\Models\LevelModel;  
10  
11  class UserModel extends Authenticatable  
12  {  
13      use HasFactory;  
14  
15      protected $table = 'm_user'; // Mendefinisikan nama tabel digunakan oleh model ini  
16      protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan  
17      protected $fillable = ['level_id', 'username', 'nama', 'password', 'created_at', 'update_at'];  
18  
19      protected $hidden = ['password']; // jangan di tampilkan di select  
20  
21      protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash  
22  
23      /**  
24       * Relasi ke tabel level  
25       */  
26      public function level(): BelongsTo  
27      {  
28          return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
29      }
```

- c. Selanjutnya kita buat **AuthController.php** untuk memproses login yang akan kita lakukan

```
C:\laragon\www\PWL_POS>php artisan make:controller AuthController
```

```
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\AuthController.php] created successfully.
```

```

app > Http > Controllers > AuthController.php > AuthController > logout
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 home
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
31             return response()->json([
32                 'status' => false,
33                 'message' => 'Login Gagal'
34             ]);
35         }
36
37         return redirect('login');
38     }
39
40     public function logout(Request $request)
41     {
42         Auth::logout();
43
44         $request->session()->invalidate();
45         $request->session()->regenerateToken();
46         return redirect('login');
47     }
48 }

```

- d. 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 > body.hold-transition.login-page > div.login-box > div.card.card-outline.card-primary >
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 <!-- Google Font: Source Sans Pro -->
8 <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">
9 <!-- Font Awesome -->
10 <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
11 <!-- icheck bootstrap -->
12 <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
13 <!-- SweetAlert2 -->
14 <link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap4.min.css') }}">
15 <!-- Theme style -->
16 <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
17 </head>
18 <body class="hold-transition login-page">
19 <div class="login-box">
20 <!-- /.login-logo -->
21 <div class="card card-outline card-primary">
22 <div class="card-header text-center">
23 <a href="{{ url('/') }}" class="h1"><b>Admin</b>LTE</a>
24 </div>
25 <div class="card-body">
26 <p class="login-box-msg">Sign in to start your session</p>
27 <form action="{{ url('login') }}" method="POST" id="form-login">
28 @csrf
29 <div class="input-group mb-3">
30 <input type="text" name="username" class="form-control" placeholder="Username">
31 <div class="input-group-append">
32 <div class="input-group-text">
33 <span class="fas fa-envelope"></span>
34 </div>
35 </div>
36 <small id="error-username" class="error-text text-danger"></small>
37 </div>
38 <div class="input-group mb-3">
39 <input type="password" name="password" class="form-control" placeholder="Password">
40 <div class="input-group-append">
41 <div class="input-group-text">
42 <span class="fas fa-lock"></span>
43 </div>
44 </div>
45 <small id="error-password" class="error-text text-danger"></small>
46 </div>
47 <div class="row">
48 <div class="col-8">
49 <div class="icheck-primary">
50 <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
51 </div>
52 </div>
53 <!-- /.col -->
54 <div class="col-4">
55 <button type="submit" class="btn btn-primary btn-block">Sign In</button>
56 </div>
57 <!-- /.col -->
58 </div>
59 </form>
60 </div>
61 <!-- /.card-body -->
62 </div>
63 <!-- /.card -->
64 </div>
65 <!-- /.login-box -->
66
67 <!-- jQuery -->
68 <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
69 <!-- Bootstrap 4 -->
70 <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
71 <!-- jquery-validation -->
72 <script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
73 <script src="{{ asset('plugins/jquery-validation/additional-methods.min.js') }}"></script>
74 <!-- SweetAlert2 -->
75 <script src="{{ asset('plugins/sweetalert2/sweetalert2.min.js') }}"></script>
76 <!-- AdminLTE App -->

```

```

77 <script src="{ asset('adminlte/dist/js/adminlte.min.js') }"></script>
78
79 <script>
80   $.ajaxSetup({
81     headers: {
82       'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
83     }
84   });
85
86   $(document).ready(function() {
87     $("#form-login").validate({
88       rules: {
89         username: {required: true, minlength: 4, maxlength: 20},
90         password: {required: true, minlength: 6, maxlength: 20}
91       },
92       submitHandler: function(form) { // ketika valid, maka bagian yg akan dijalankan
93         $.ajax({
94           url: form.action,
95           type: form.method,
96           data: $(form).serialize(),
97           success: function(response) {
98             if(response.status){ // jika sukses
99               Swal.fire({
100                 icon: 'success',
101                 title: 'Berhasil',
102                 text: response.message,
103               }).then(function() {
104                 window.location = response.redirect;
105               });
106             }else{ // jika error
107               $('.error-text').text('');
108               $.each(response.msgField, function(prefix, val) {
109                 $('#error-'+prefix).text(val[0]);
110               });
111               Swal.fire({
112                 icon: 'error',
113                 title: 'Terjadi Kesalahan',
114                 text: response.message
115               });
116             }
117           }
118         });
119         return false;
120       },
121       errorElement: 'span',
122       errorPlacement: function (error, element) {
123         error.addClass('invalid-feedback');
124         element.closest('.input-group').append(error);
125       },
126       highlight: function (element, errorClass, validClass) {
127         $(element).addClass('is-invalid');
128       },
129       unhighlight: function (element, errorClass, validClass) {
130         $(element).removeClass('is-invalid');
131       }
132     });
133   });
134 </script>
135 </body>
136 </html>

```

e. Kemudian kita modifikasi [route/web.php](#) agar semua route masuk dalam auth

```

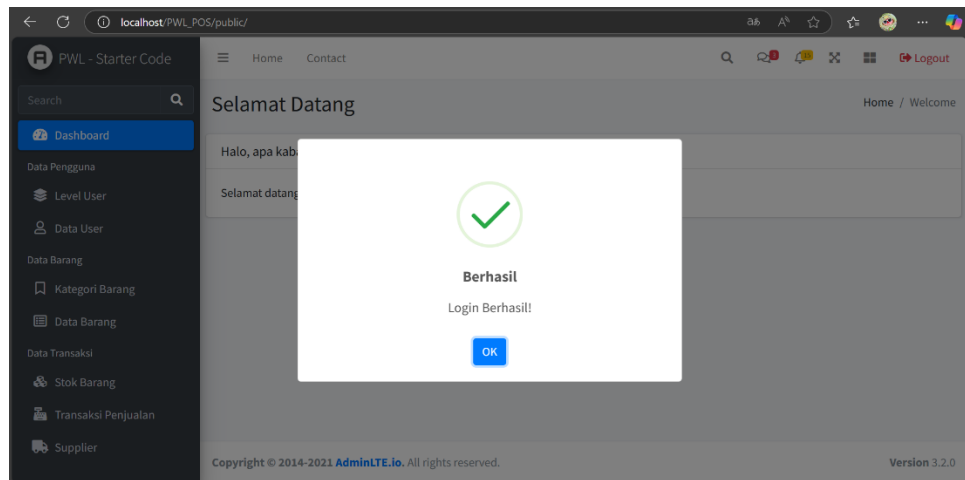
routes > web.php > Closure > Closure
1  <?php
2
3  use App\Http\Controllers\UserController;
4  use App\Http\Controllers\KategoriController;
5  use App\Http\Controllers\BarangController;
6  use App\Http\Controllers\LevelController;
7  use App\Http\Controllers>WelcomeController;
8  use App\Http\Controllers\SupplierController;
9  use App\Http\Controllers\AuthController;
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 group ini harus login dulu
19
20     // masukkan semua route yang perlu autentikasi di sini

```

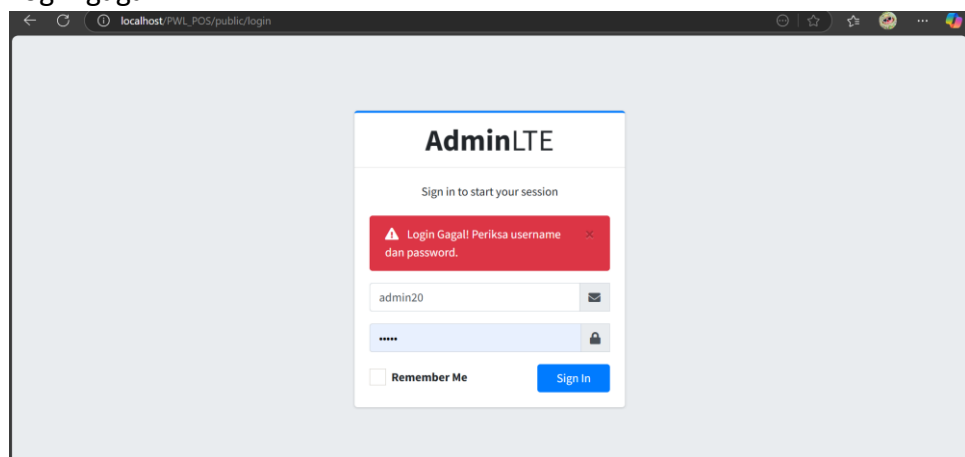
- f. Ketika kita coba mengakses halaman [localhost/PWL\\_POS/public](localhost/PWL_POS/public) maka akan tampil halaman awal untuk login ke aplikasi

### Tugas 1 - Implementasi Authentication :

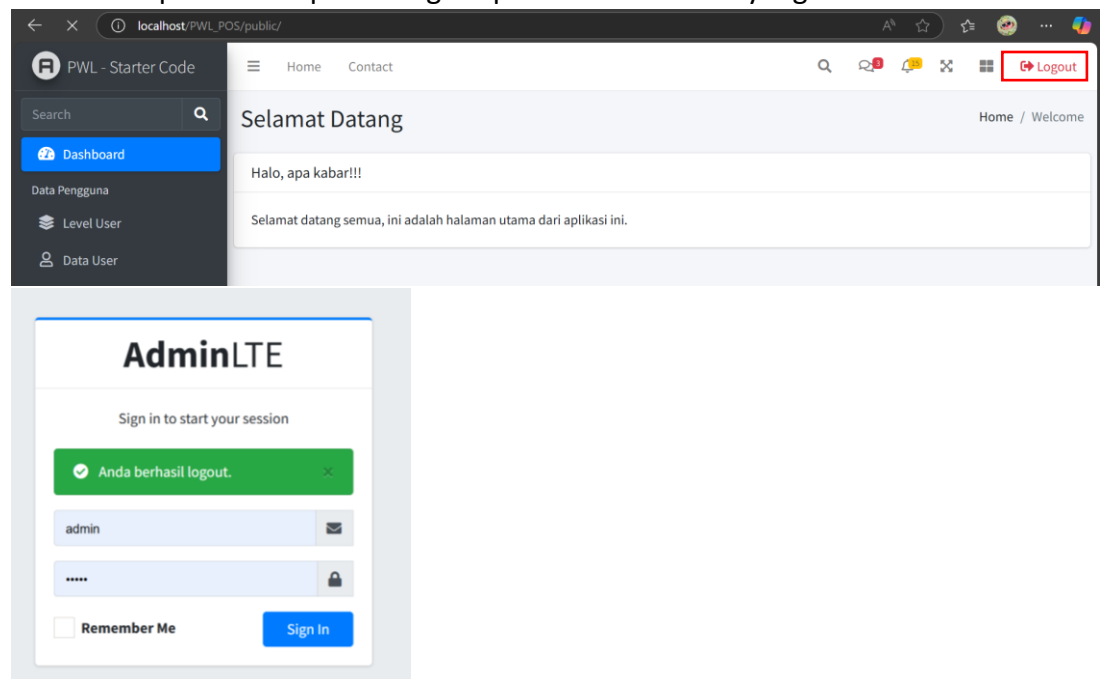
1. Silahkan implementasikan proses login pada project kalian masing-masing
  - a. Login berhasil



b. Login gagal



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



3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
4. Submit kode untuk implementasi Authentication pada repository github kalian.



## 2. Praktikum 2: Implementasi Authentication di Laravel dengan Middleware

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

- a. 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;
}
```

- b. Kemudian kita buat *middleware* dengan nama [AuthorizeUser.php](#). Kita bisa buat *middleware* dengan mengetikkan perintah pada terminal/CMD  
[php artisan make:middleware AuthorizeUser](#)

```
C:\laragon\www\PWL_POS>php artisan make:middleware AuthorizeUser

INFO Middleware [C:\laragon\www\PWL_POS\app\Http\Middleware\
AuthorizeUser.php] created successfully.
```

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

- c. Kemudian kita edit *middleware* [AuthorizeUser.php](#) untuk bisa mengecek apakah pengguna yang mengakses memiliki Level/Role/Group yang sesuai

```

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

```













- d. 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,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,

```

- e. Sekarang kita perhatikan tabel [m\\_level](#) yang menjadi tabel untuk menyimpan level/group/role dari user ada

<div><div>←</div><div>T</div><div>→</div></div>				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	4	PLG	Pelanggan	NULL	NULL

- f. 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

```

// group route untuk admin (level ADM)
Route::middleware(['authorize:ADM'])->group(function () {
    Route::get('/level', [LevelController::class, 'index']); // menampilkan halaman level
    Route::post('/level/list', [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
    Route::get('/level/create', [LevelController::class, 'create']); // menampilkan form tambah level
    Route::post('/level', [LevelController::class, 'store']); // menyimpan data level
    Route::get('/level/{id}/edit', [LevelController::class, 'edit']); // menampilkan form edit level
    Route::put('/level/{id}', [LevelController::class, 'update']); // mengupdate data level
    Route::delete('/level/{id}', [LevelController::class, 'destroy']); // menghapus data level
});

```

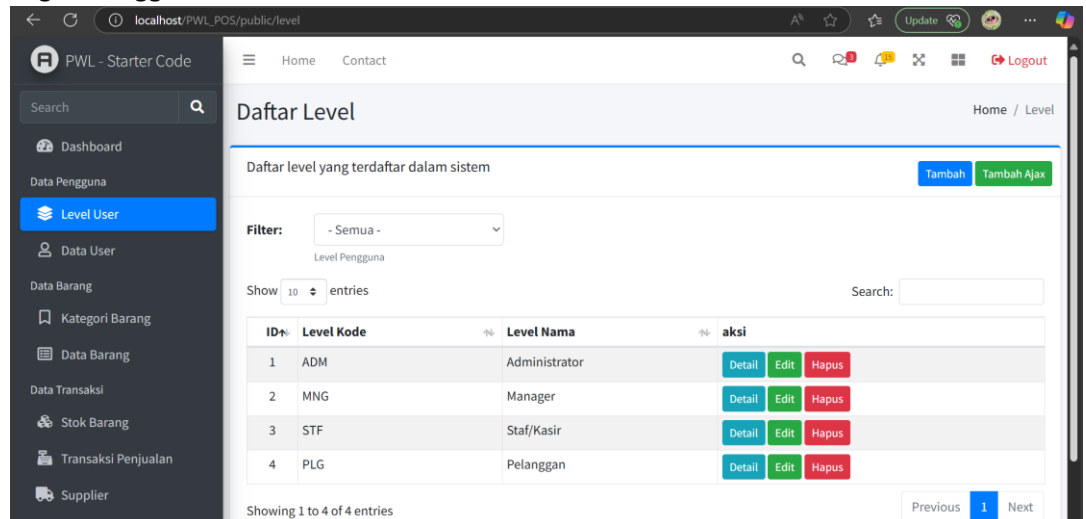
Pada kode yang ditandai merah, terdapat `authorize:ADM`. Kode `ADM` adalah nilai dari `level_kode` pada tabel `m_level`. Yang artinya, user yang bisa mengakses route untuk manage data level, adalah user yang memiliki level sebagai Administrator.

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

- Login menggunakan level staf



- Login menggunakan level administrator



## Tugas 2 - Implementasi Authentication :

1. Apa yang kalian pahami pada praktikum 2 ini?
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Submit kode untuk impementasi Authorization pada repository github kalian.

### 3. Praktikum 3: Implementasi Multi-Level Authorizaton 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:

- a. 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;
}
```

- b. Selanjutnya, Kita modifikasi middleware [AuthorizeUser.php](#) dengan kode berikut

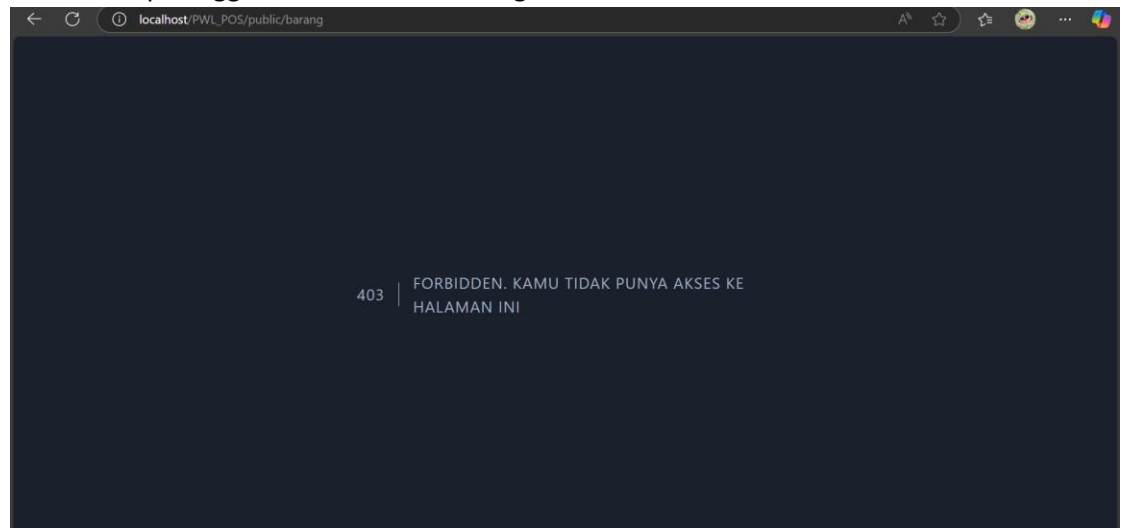
```
app > Http > Middleware > AuthorizeUser.php > ...
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthorizeUser
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param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

- c. Setelah itu tinggal kita perbaiki [route/web.php](#) sesuai dengan role/level yang diinginkan. Contoh

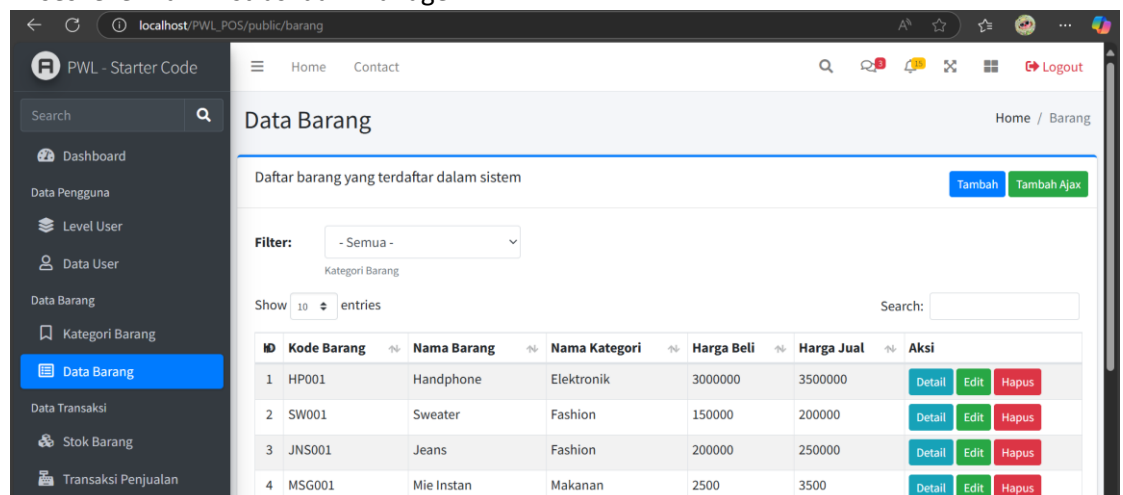
```
// artinya semua route di dalam group ini harus punya role ADM (Administrasi) dan MNG (Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function() {
    Route::get('/barang', [BarangController::class, 'index']); // halaman barang
    Route::post('/barang/list', [BarangController::class, 'list']); // data json
    Route::get('/barang/create', [BarangController::class, 'create']); // form tambah
    Route::post('/barang', [BarangController::class, 'store']); // simpan
    Route::get('/barang/create_ajax', [BarangController::class, 'create_ajax']); // form ajax
    Route::post('/barang/ajax', [BarangController::class, 'store_ajax']); // simpan ajax
    Route::get('/barang/{id}/edit', [BarangController::class, 'edit']); // form edit
    Route::put('/barang/{id}', [BarangController::class, 'update']); // update
    Route::get('/barang/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // form edit ajax
    Route::put('/barang/{id}/update_ajax', [BarangController::class, 'update_ajax']); // update ajax
    Route::get('/barang/{id}/show_ajax', [BarangController::class, 'show_ajax']); // menampilkan detail barang
    Route::get('/barang/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // konfirmasi hapus ajax
    Route::delete('/barang/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // hapus ajax
    Route::delete('/barang/{id}', [BarangController::class, 'destroy']); // hapus biasa
});
```

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

- Akses level pelanggan ke menu data barang



- Akses level Administrasi dan Manager



### 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
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu menu yang sesuai dengan Level/Jenis User

```
// route level hanya untuk MNG
Route::middleware(['authorize:MNG'])->group(function () {
    Route::prefix('level')->group(function () {
        Route::get('/', [LevelController::class, 'index']);
        Route::post('/list', [LevelController::class, 'list']);
        Route::get('/create', [LevelController::class, 'create']);
        Route::post('/', [LevelController::class, 'store']);
        Route::get('/create_ajax', [LevelController::class, 'create_ajax']);
        Route::post('/ajax', [LevelController::class, 'store_ajax']);
        Route::get('/{id}/edit', [LevelController::class, 'edit']);
        Route::put('/{id}', [LevelController::class, 'update']);
        Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']);
        Route::get('/{id}/show_ajax', [LevelController::class, 'show_ajax']);
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']);
        Route::delete('/{id}', [LevelController::class, 'destroy']);
    });
});
```

```
// route untuk user - hanya ADM & MNG
Route::middleware(['authorize:ADM,MNG'])->group(function () {
    Route::prefix('user')->group(function () {
        Route::get('/', [UserController::class, 'index']);
        Route::post('/list', [UserController::class, 'list']);
        Route::get('/create', [UserController::class, 'create']);
        Route::post('/', [UserController::class, 'store']);
        Route::get('/create_ajax', [UserController::class, 'create_ajax']);
        Route::post('/ajax', [UserController::class, 'store_ajax']);
        Route::get('/{id}/edit', [UserController::class, 'edit']);
        Route::put('/{id}', [UserController::class, 'update']);
        Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']);
        Route::get('/{id}/show_ajax', [UserController::class, 'show_ajax']);
        Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']);
        Route::delete('/{id}', [UserController::class, 'destroy']);
    });
});
```

```
// route untuk kategori - ADM, MNG, STF
Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
    Route::prefix('kategori')->group(function () {
        Route::get('/', [KategoriController::class, 'index']);
        Route::post('/list', [KategoriController::class, 'list']);
        Route::get('/create', [KategoriController::class, 'create']);
        Route::post('/', [KategoriController::class, 'store']);
        Route::get('/create_ajax', [KategoriController::class, 'create_ajax']);
        Route::post('/ajax', [KategoriController::class, 'store_ajax']);
        Route::get('/{id}/edit', [KategoriController::class, 'edit']);
        Route::put('/{id}', [KategoriController::class, 'update']);
        Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']);
        Route::get('/{id}/show_ajax', [KategoriController::class, 'show_ajax']);
        Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']);
        Route::delete('/{id}', [KategoriController::class, 'destroy']);
    });
});

// route untuk supplier - ADM, MNG, STF
Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
    Route::prefix('supplier')->group(function () {
        Route::get('/', [SupplierController::class, 'index']);
        Route::post('/list', [SupplierController::class, 'list']);
        Route::get('/create', [SupplierController::class, 'create']);
        Route::post('/', [SupplierController::class, 'store']);
        Route::get('/create_ajax', [SupplierController::class, 'create_ajax']);
        Route::post('/ajax', [SupplierController::class, 'store_ajax']);
        Route::get('/{id}/edit', [SupplierController::class, 'edit']);
        Route::put('/{id}', [SupplierController::class, 'update']);
        Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']);
        Route::get('/{id}/show_ajax', [SupplierController::class, 'show_ajax']);
        Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']);
        Route::delete('/{id}', [SupplierController::class, 'destroy']);
    });
});

// route untuk barang - ADM, MNG, STF, PLG full akses
Route::prefix('barang')->group(function () {
    Route::get('/', [BarangController::class, 'index'])->middleware('authorize:ADM,MNG,STF,PLG');
    Route::post('/list', [BarangController::class, 'list'])->middleware('authorize:ADM,MNG,STF,PLG');
    Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax'])->middleware('authorize:ADM,MNG,STF,PLG');

    // Akses hanya untuk ADM, MNG, STF
    Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
        Route::get('/create', [BarangController::class, 'create']);
        Route::post('/', [BarangController::class, 'store']);
        Route::get('/create_ajax', [BarangController::class, 'create_ajax']);
        Route::post('/ajax', [BarangController::class, 'store_ajax']);
        Route::get('/{id}/edit', [BarangController::class, 'edit']);
        Route::put('/{id}', [BarangController::class, 'update']);
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']);
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
        Route::delete('/{id}', [BarangController::class, 'destroy']);
    });
});
```

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

#### Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.
2. Screenshot hasil yang kalian kerjakan

- Registrasi

- Registrasi Berhasil

- Berhasil tersimpan secara default dengan level PLG

ID	Username	Nama	Level Pengguna	Aksi
1	customer-1	Pelanggan Pertama	Pelanggan	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
2	customer-2	pelanggan2	Pelanggan	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

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