



Mata Kuliah : Pemrograman Web Lanjut (PWL)  
Program Studi : D4 – Sistem Informasi Bisnis  
Semester : 4 (empat)  
Pertemuan ke- : 7 (tujuh)  
Nama : Aqila Nur Azza

## 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

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

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

```
<?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');  
    }  
}
```



```
1  <?php
2  namespace App\Models;
3
4  use Illuminate\Database\Eloquent\Model;
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Relations\BelongsTo;
7  use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable
8
9  class UserModel extends Authenticatable
10 {
11     use HasFactory;
12
13     protected $table = 'm_user';
14     protected $primaryKey = 'user_id';
15     protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];
16
17     protected $hidden = ['password']; // jangan di tampilkan saat select
18
19     protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
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 }
```

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');
    }
}
```

```
PWL_F05 > app > http > controllers > @ AuthController.php > @ AuthController > postlogin
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 }
```



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=fallback">
<!-- 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>LTE</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 -->
```

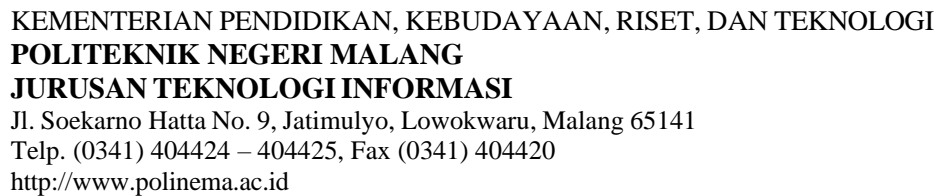




```
<script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<!-- jquery-validation -->
<script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<script src="{{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

[illegible]

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

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



The screenshot shows the AdminLTE login interface. It has a white box on a light gray background. The title 'AdminLTE' is at the top. Below it is the text 'Sign in to start your session'. There are two input fields: 'Username' with an envelope icon and 'Password' with a lock icon. Below the password field is a 'Remember Me' checkbox and a blue 'Sign In' button.

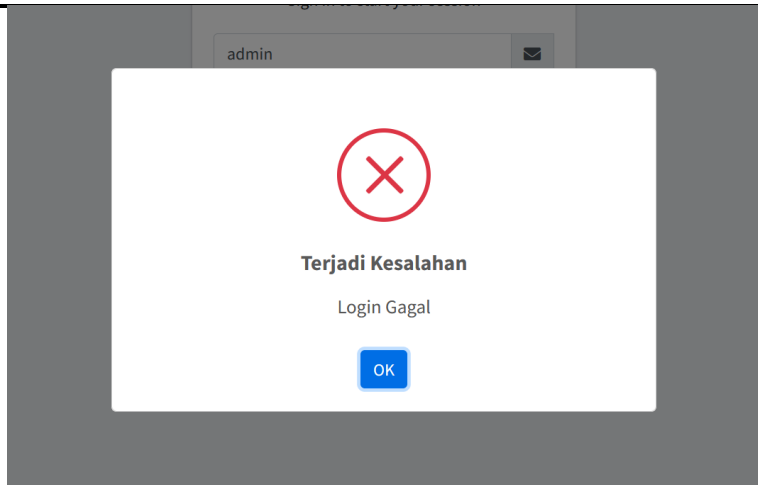
### Tugas 1 – Implementasi Authentication :

1. Silahkan implementasikan proses login pada project kalian masing-masing
  - Jika gagal Login

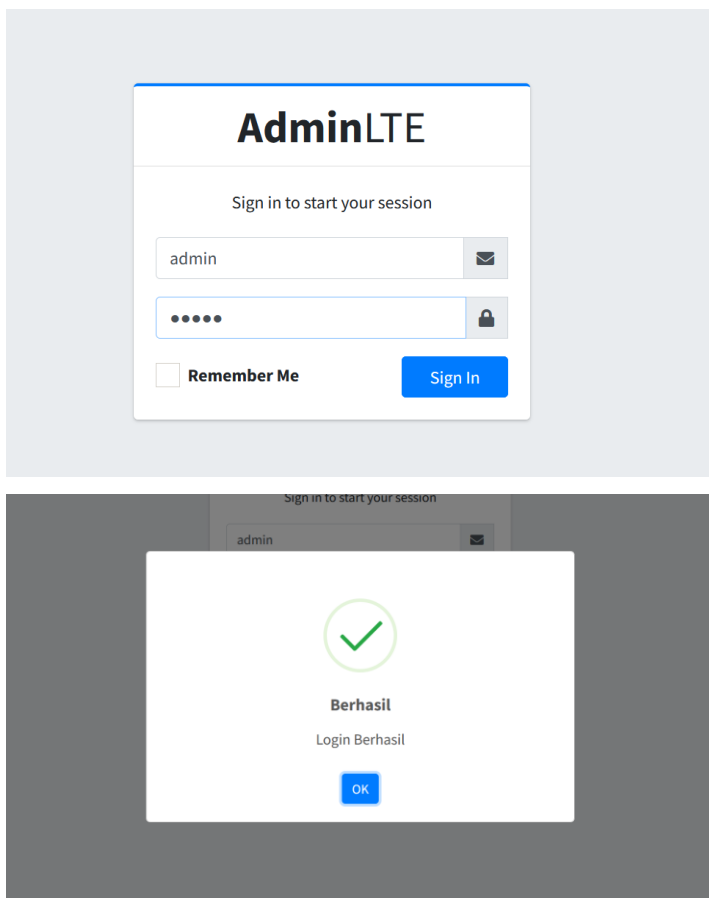
This screenshot shows the AdminLTE login form with the username 'admin' entered in the 'Username' field. The 'Password' field is masked with dots. The 'Remember Me' checkbox is unchecked, and the 'Sign In' button is visible.



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI  
**POLITEKNIK NEGERI MALANG**  
**JURUSAN TEKNOLOGI INFORMASI**  
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141  
Telp. (0341) 404424 – 404425, Fax (0341) 404420  
<http://www.polinema.ac.id>

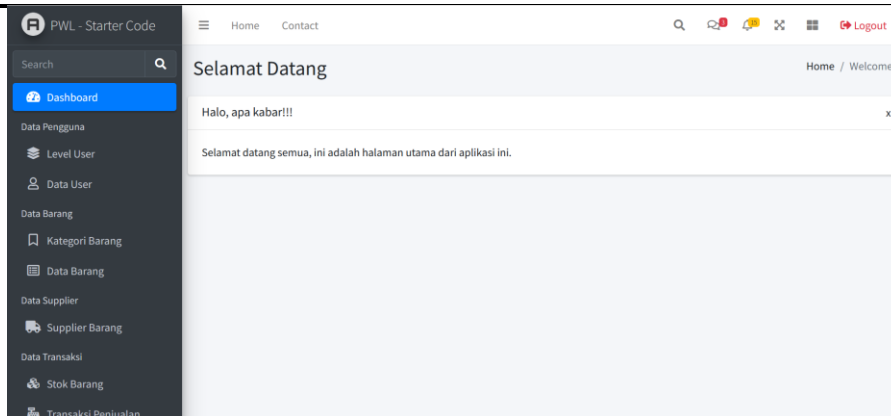


➤ Jika berhasil login



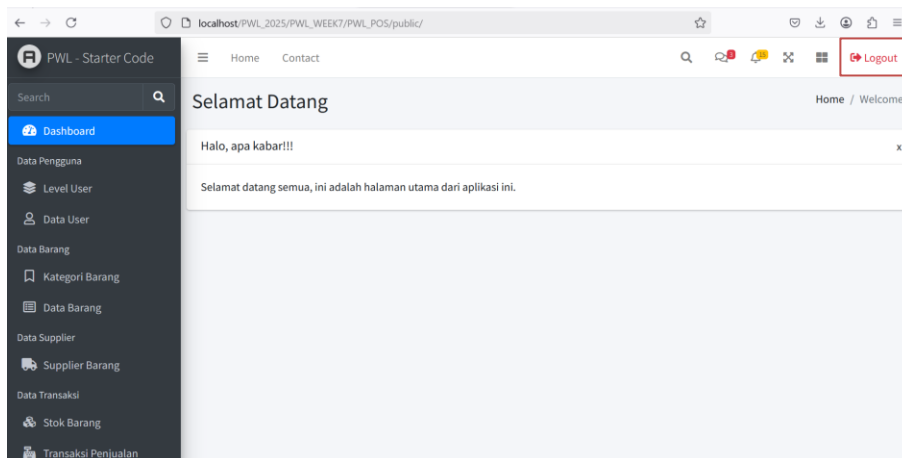


KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI  
**POLITEKNIK NEGERI MALANG**  
**JURUSAN TEKNOLOGI INFORMASI**  
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141  
Telp. (0341) 404424 – 404425, Fax (0341) 404420  
<http://www.polinema.ac.id>

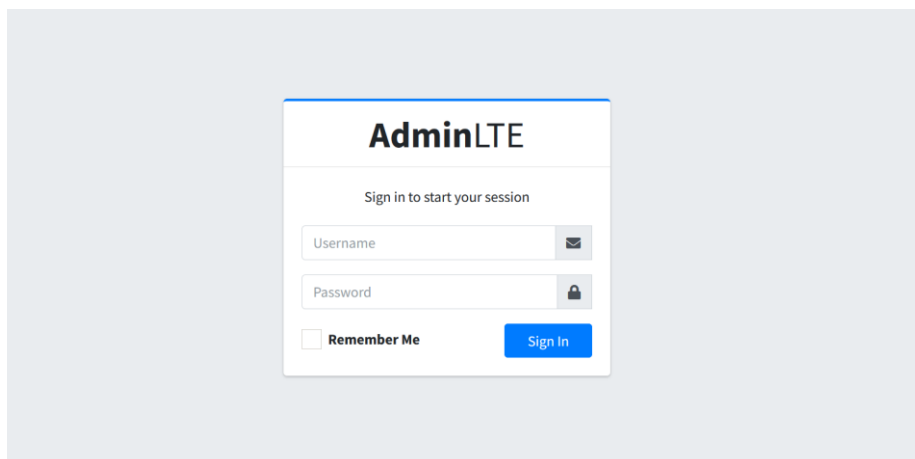


2. Silahkan implementasi proses logout pada halaman web yang kalian buat

➤ Tampilan Tombol Logout



➤ Tampilan setelah memencet tombol Logout



3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
4. Submit kode untuk implementasi 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;
}
```



```
/**
 * 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;
}
```

Penjelasan :

➤ Fungsi getRoleName()

- Mengecek apakah user memiliki role tertentu.
- Membandingkan `level_kode` user dengan role yang diberikan sebagai parameter.

➤ Fungsi hasRole(\$role)

- Mengecek apakah user memiliki role tertentu.
- Membandingkan `level_kode` user dengan role yang diberikan sebagai parameter.

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`

```
laragon\www\PWL_2025\PWL_WEEK7\PWL_POS>php artisan make:middleware AuthorizeUser
```

```
INFO Middleware [C:\laragon\www\PWL_2025\PWL_WEEK7\PWL_POS\app\Http\Middleware\AuthorizeUser.php] created successful
```

Penjelasan :

- Middleware ini akan digunakan untuk mengecek apakah pengguna memiliki hak akses tertentu sebelum mengakses halaman atau fitur dalam aplikasi.
- Middleware `AuthorizeUser.php` dibuat untuk menangani **authorization** di Laravel. Dengan middleware ini, kita dapat mengontrol akses pengguna berdasarkan role mereka



sebelum mereka bisa mengakses halaman tertentu.

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

```
PWL_POS > app > Http > Middleware > AuthorizeUser.php > ...
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
23         // jika tidak punya role, maka tampilkan error 403
24         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
25     }
26 }
27
```

Penjelasan :

Middleware `AuthorizeUser.php` digunakan untuk mengecek role pengguna sebelum mengizinkan akses ke halaman tertentu dalam aplikasi Laravel. Jika user tidak memiliki peran yang sesuai, akses akan diblokir dengan error 403 (Forbidden).

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

```
48  /**
49   * The application's middleware aliases.
50   *
51   * Aliases may be used instead of class names to conveniently assign middleware to routes and groups.
52   *
53   * @var array<string, class-string|string>
54   */
55  protected $middlewareAliases = [
56      'auth' => \App\Http\Middleware\Authenticate::class,
57      'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat
58      'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
59      'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
60  ];
61
62
```

Penjelasan :

- 'auth' => \App\Http\Middleware\Authenticate::class,
  - Middleware bawaan Laravel untuk otentikasi pengguna.
  - Memastikan user sudah login sebelum mengakses halaman tertentu.
- 'authorize' => \App\Http\Middleware\AuthorizeUser::class,
  - Middleware custom yang dibuat sebelumnya (`AuthorizeUser.php`).
  - Digunakan untuk mengecek apakah user memiliki role tertentu sebelum diizinkan mengakses halaman.
- 'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
  - Middleware bawaan Laravel untuk Basic Authentication (username dan password via HTTP).
  - Biasanya digunakan untuk API sederhana yang membutuhkan login tanpa sesi.





➤ 'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,

- Middleware yang menangani session authentication untuk memastikan pengguna yang masuk tetap terautentikasi di seluruh sesi browser.

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"/>	Check all	With selected:		Edit	Copy	Delete	Export	



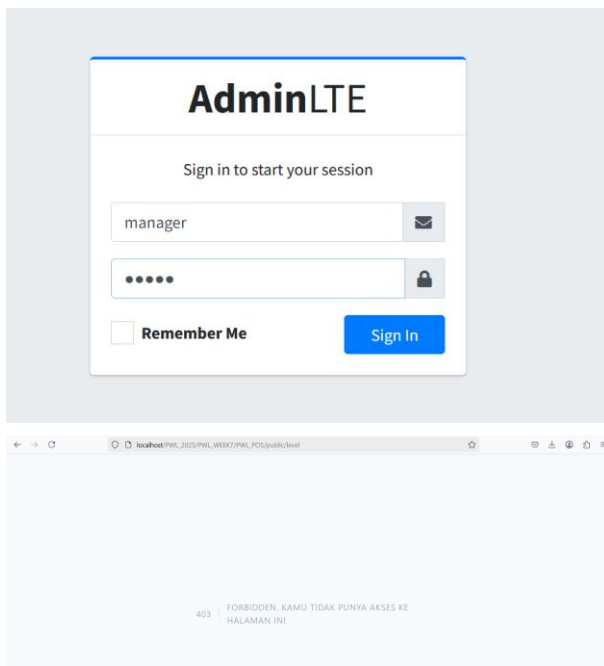
6. Untuk mencoba *authorization* yang telah kita buat, maka perlu kita modifikasi `route/web.php` untuk menentukan route mana saja yang akan diberi hak akses sesuai dengan level user

```
// 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 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
});
```

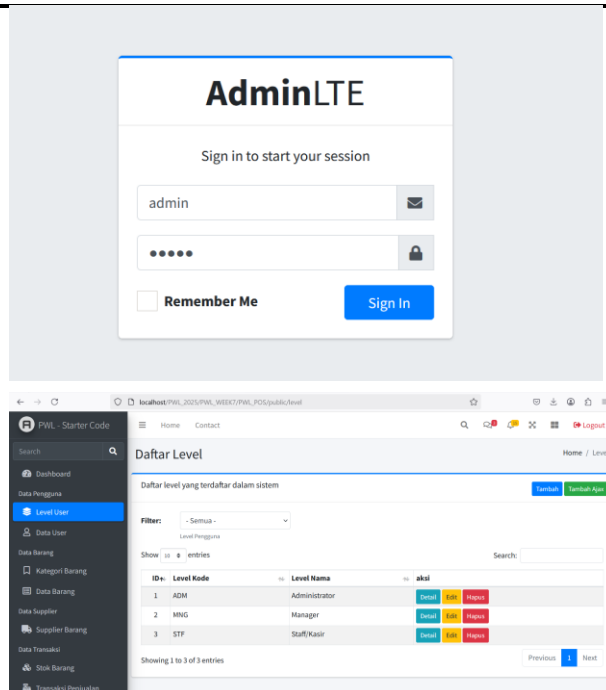
Pada kode yang ditandai merah, terdapat `authorize:ADM`. Kode `ADM` adalah nilai dari `level_kode` pada tabel `m_level`. Yang artinya, user yang bisa mengakses route untuk manage data level, adalah user yang memiliki level sebagai Administrator.

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

➤ Ketika login menggunakan user manager



➤ Ketika login menggunakan user admin



## Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?

Jawaban :

- Authorization digunakan untuk membatasi akses pengguna berdasarkan role atau level tertentu.
  - Middleware menjadi alat utama dalam Laravel untuk mengecek izin pengguna sebelum mengizinkan akses ke halaman atau fitur tertentu.
  - Pada UserModel.php, ditambahkan fungsi getRoleName() dan hasRole() untuk mendapatkan role user dan mengecek apakah user memiliki role tertentu.
  - Middleware AuthorizeUser.php dibuat untuk memvalidasi role pengguna sebelum mengizinkan akses ke halaman tertentu.
  - Middleware ini kemudian didaftarkan di Kernel.php agar bisa digunakan dalam route dengan alias authorize.
  - Middleware digunakan di dalam route grouping untuk membatasi akses hanya kepada user dengan role tertentu. Pada kasus ini hanya user dengan role "ADM" yang bisa mengakses halaman tersebut.
  - Dengan menggunakan authorization berbasis role, kita bisa memastikan bahwa hanya pengguna yang memiliki izin yang bisa mengakses fitur-fitur 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. 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 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:

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;
}
```

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)
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 }
```

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 Authorization :

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing

#### ➤ Http/Middleware/AuthorizeUser.php

```
PWL POS > app > Http > Middleware > AuthorizeUser.php > ...
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 $next, ...$roles): Response
16     {
17         $user_role = $request->user()->getRole(); // Ambil data level_kode dari user yang login
18
19         if (in_array($user_role, $roles)) { // Cek apakah level_kode user ada di dalam array roles
20             return $next($request); // Jika ada, maka lanjutkan request
21         }
22
23         // Jika tidak punya role, maka tampilkan error 403
24         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
25     }
26 }
```

Penjelasan :

Kode dalam file ini bertanggung jawab untuk menangani akses pengguna berdasarkan peran (peran) mereka .

- Middleware ini memeriksa role pengguna yang sedang login dan membandingkannya dengan role yang diperlukan untuk mengakses halaman tertentu.
- Jika pengguna memiliki peran yang sesuai, permintaan akan diteruskan ke halaman yang diminta.
- Jika tidak, pengguna akan menerima error 403 Forbidden , yang berarti mereka tidak diizinkan mengakses halaman tersebut.

#### ➤ Models/UserModel





```
/**
 * 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;
}
```

Penjelasan :

Model User mengelola informasi pengguna dalam sistem, termasuk peran atau level aksesnya.

- `getRoleName()` → Mengembalikan nama peran pengguna.
- `hasRole($role)` → Memeriksa apakah pengguna memiliki peran tertentu.
- `getRole()` → Mengambil kode peran pengguna.

#### ➤ Routes/web.php

```
// 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
});
```

Penjelasan :

Bagian ini menunjukkan bagaimana sistem routing Laravel mengatur endpoint yang hanya bisa diakses oleh pengguna dengan role tertentu.

- Middleware `authorize:ADM,MNG` digunakan untuk membatasi akses hanya untuk pengguna dengan peran "ADM" (Administrator) dan "MNG" (Manager) .
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
- Jika mengakses admin



## AdminLTE

Sign in to start your session

☐ Remember Me

Sign In

PIWL - Starter Code

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Barang

Data Supplier

Supplier Barang

Data Transaksi

Stok Barang

Transaksi Penjualan

Home

Contact

Logout

### Data Barang

Daftar barang yang terdaftar dalam sistem

Tambah Tambah Ajax

Filter: Semua

Kategori Barang

Search:

ID	Kode Barang	Nama Barang	Nama Kategori	Harga Beli	Harga Jual	Aksi
1	B001	Air Mineral	Food & Beverage	2000	3000	Detail Edit Hapus
2	B002	Roti Tawar	Food & Beverage	10000	12000	Detail Edit Hapus
3	B003	Sabun Mandi	Beauty & Health	5000	7000	Detail Edit Hapus
4	B004	Shampoo	Beauty & Health	15000	18000	Detail Edit Hapus
5	B005	Pembersih Lantai	Home Care	10000	13000	Detail Edit Hapus
6	B006	Tisu	Home Care	8000	10000	Detail Edit Hapus
7	B007	Popok Bayi	Baby & Kids	50000	55000	Detail Edit Hapus

➤ Jika mengakses Manager

## AdminLTE

Sign in to start your session

☐ Remember Me

Sign In





PWL - Starter Code

Home Contact

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Barang

Kategori Barang

Data Supplier

Supplier Barang

Data Transaksi

Stok Barang

Transaksi Penjualan

Data Barang

Home / Barang

Data Barang

Daftar barang yang terdaftar dalam sistem

Tambah Tambah Ajax

Filter: - Semua -

Kategori Barang

Show 10 entries

Search:

ID	Kode Barang	Nama Barang	Nama Kategori	Harga Beli	Harga Jual	Aksi
1	B001	Air Mineral	Food & Beverage	2000	3000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
2	B002	Roti Tawar	Food & Beverage	10000	12000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
3	B003	Sabun Mandi	Beauty & Health	5000	7000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
4	B004	Shampoo	Beauty & Health	15000	18000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
5	B005	Pembersih Lantai	Home Care	10000	13000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
6	B006	Tisu	Home Care	8000	10000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
7	B007	Popok Bayi	Baby & Kids	50000	55000	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

➤ Jika mengakses staff

AdminLTE

Sign in to start your session

staff

.....

☐ Remember Me

Sign In

403 FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI

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

Route/Web.php



```
// Semua rute di bawah ini hanya bisa diakses jika sudah login
Route::middleware(['auth'])->group(function () {
    Route::get('/', [HomeController::class, 'index']);

    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 datatable
            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::post('/{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::post('/{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 tampilan 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
        });
    });
});
```

```
// artinya semua rute 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 datatable
        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::post('/{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::post('/{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 tampilan 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
    });
});

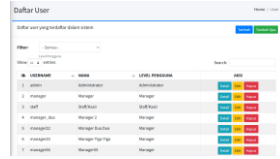


// artinya semua rute di dalam group ini harus punya role ADM (Administrator) dan PMG (Manager)
Route::middleware(['authorize:ADM,PMG'])->group(function () {
    Route::group(['prefix' => 'kategori'], function () {
        Route::get('/', [KategoriController::class, 'index']); // Menampilkan halaman awal kategori
        Route::post('/list', [KategoriController::class, 'list']); // Menampilkan data kategori dalam bentuk json untuk datatable
        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::post('/{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::post('/{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
    });
});
```

```
// User hanya bisa melihat data barang
Route::middleware(['authorize:ADM,PMG,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 datatable
        Route::get('/{id}', [BarangController::class, 'show']); // Menampilkan detail barang
    });
});

// ADM & PMG bisa menambah, mengedit, dan menghapus barang
Route::middleware(['authorize:ADM,PMG'])->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::post('/{id}', [BarangController::class, 'update']); // Simpan perubahan barang
        // Edit menggunakan AJAX
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // Form edit barang AJAX
        Route::post('/{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
    });
});

// artinya semua rute di dalam group ini harus punya role ADM (Administrator) dan PMG (Manager)
Route::middleware(['authorize:ADM,PMG'])->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::post('/{id}', [SupplierController::class, 'update']);
        // Edit menggunakan AJAX
        Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); // Menampilkan halaman form edit Supplier Ajax
        Route::post('/{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']);
    });
});
```

## Hak Akses User

Menu	Admin	Manager	Staff/Kasir
Data User			



Data Level			
Data Kategori			
Data Barang			 (hanya bisa melihat)
Data Supplier			

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

## Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

### ➤ AuthController

```
public function register()
{
    $levels = LevelModel::select('level_id', 'level_name')->get();
    return view('auth.register')->with('levels', $levels);
}

public function postRegister(Request $request)
{
    $validator = Validator::make($request->all(), [
        'username' => 'required|string|min:5|unique:users,username',
        'nama' => 'required|string|min:5',
        'password' => 'required|string|min:5|confirmed',
        'level_id' => 'required|exists:levels,level_id',
    ]);

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

    UserModel::create([
        'username' => $request->username,
        'nama' => $request->nama,
        'password' => bcrypt($request->password),
        'level_id' => $request->level_id,
    ]);

    return response()->json([
        'status' => true,
        'message' => 'Registration Success.',
        'redirect' => url('/login')
    ]);
}
```

### ➤ Web.php

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



➤ View/Auth/Register.blade.php

```
PWL: POL > resources > views > auth > register.blade.php > 60.html
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 - Polineka >title>
8 <!-- Google Font: Source Sans Pro -->
9 <link rel="stylesheet"
10 | href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=flex"
11 |>
12 <!-- Font Awesome -->
13 <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
14 <!-- iCheck Bootstrap -->
15 <link rel="stylesheet" href="{{ asset('adminlte/plugins/iCheck-bootstrap/iCheck-bootstrap.min.css') }}">
16 <!-- SweetAlert2 -->
17 <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
18 <!-- Theme style -->
19 <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
20 </head>
21
22 <body class="hold-transition login-page">
23 <div class="login-box">
24 <!-- .login-box -->
25 <div class="card card-outline card-primary">
26 <div class="card-header text-center"><a href="{{ url('/') }}" class="h1">AdminLTE</a></div>
27 <div class="card-body">
28 <div class="login-box-msg">Register a new account</p>
29 <form action="{{ url('/register') }}" method="POST" id="form-register">
30 <div class="input-group mb-3">
31 <input type="text" id="username" name="username" class="form-control" placeholder="Username"
32 | required:
33 <div class="input-group-append">
34 <div class="input-group-text">
35 <span class="fas fa-user"></span>
36 </div>
37 </div>
38 <small id="error-username" class="error-text text-danger"></small>
39 </div>
40 <div class="input-group mb-3">
41 <input type="text" id="name" name="name" class="form-control" placeholder="Name"
42 | required:
43 <div class="input-group-append">
44 <div class="input-group-text">
45 <span class="fas fa-id-card"></span>
46 </div>
47 </div>
48 <small id="error-name" class="error-text text-danger"></small>
49 </div>
50 <div class="input-group mb-3">
51 <input type="password" id="password" name="password" class="form-control"
52 | placeholder="Password" required:
53 <div class="input-group-append">
54 <div class="input-group-text">
55 <span class="fas fa-lock"></span>
56 </div>
57 </div>
58 </div>
59 </div>
60
```

➤ View/Auth/Login.blade.php

```
</div>
<div class="col-12 mt-1">
  <p>Don't have an account?</p>
  <a href="{{ url('/register') }}" class="btn btn-secondary btn-block">Register</a>
</div>
```

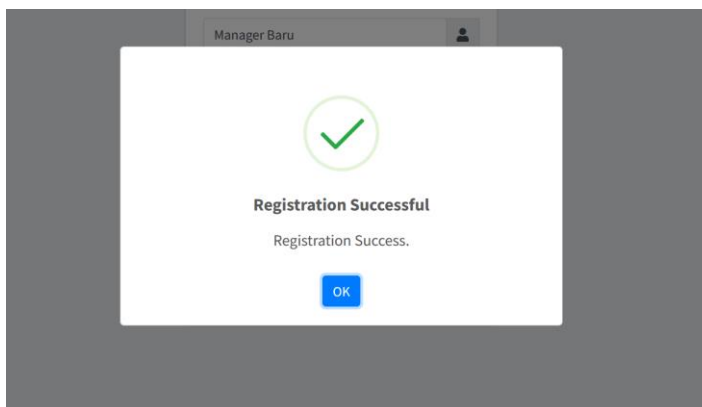
2. Screenshot hasil yang kalian kerjakan

➤ Tampilan Halaman Login

➤ Masuk Halaman Registrasi



➤ Berhasil Melakukan Registrasi



➤ Cek pada m\_user

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$uxsRMuP/zBC3QuuduxLeyKYK2DP1WyYqVv7w5kNS7...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$ya.EH3ZHPrGD95Vuo6leEtVsWIVBuByAJpy17eOym...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$6ktJK0Hc0GxuZL2je4HesyhP/p6eubx/NOAR6UgJ2...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	2	manager_dua	Manager 2	\$2y\$12\$nwFhR3lg07Jm2Us6Mgz3.i/CyuUJ/YndUjM6u.Rt6F...	2025-03-07 13:29:25	2025-03-07 13:29:25
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	2	manager22	Manager Dua Dua	\$2y\$12\$uaySea1JSL2j/e96xFrpF.qkh4zW9IRoMnGXQI.AQ1l...	2025-03-07 17:51:05	2025-03-07 17:51:05
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	2	manager33	Manager Tiga Tiga	\$2y\$12\$FW3z0d5KintF7zHMMqvFwJ66hWHuz/g5/KCQialRzp...	2025-03-07 18:07:39	2025-03-07 18:07:39
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	2	manager56	Manager55	\$2y\$12\$5hXqv/k.KIEZwPWDcipsx.nM4JFgaLLn.iPuOZ88cuc...	2025-03-07 19:58:17	2025-03-07 19:58:17
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	15	2	Manager Baru	Aqila	\$2y\$12\$PunHel3pM3YNRCFv24dJOq7nl.4uSFRwE.JtESnAWF0...	2025-03-27 17:46:30	2025-03-27 17:46:30

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

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