



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

JOBSHEET 07

Authentication dan Authorization di Laravel

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

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

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

Middleware

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

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



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

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

INFO

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

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

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. Implementasi Manual Authentication di Laravel

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

Konsep Autentikasi di Laravel

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

Beberapa komponen penting dalam sistem autentikasi Laravel meliputi:

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



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

Alur umum dari autentikasi meliputi:

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

Middleware Autentikasi

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

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

Implementasi di Laravel 10

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



Praktikum 1 – Implementasi Authentication :

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

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

Pada bagian ini kita sesuaikan dengan Model untuk tabel m_user yang sudah kita buat

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

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');  
    }  
}
```



```
J57 > 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
10 16 references | 0 implementations
11 class UserModel extends Authenticatable
12 {
13     use HasFactory;
14
15     0 references
16     protected $table = 'm_user';
17     0 references
18     protected $primaryKey = 'user_id';
19     0 references
20     protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];
21     0 references
22     protected $hidden = ['password']; // jangan di tampilkan saat select
23     0 references
24     protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
25
26     /**
27      * Relasi ke tabel level
28      */
29     0 references | 0 overrides
30     public function level(): BelongsTo
31     {
32         return $this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id');
33     }
34 }
```

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');
}
}
```



```
JS7 > app > Http > Controllers > AuthController.php > PHP Intelephense > AuthController

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(): Factory|View
11     {
12         if (Auth::check()) { // jika sudah login, maka redirect ke halaman home return redirect('/');
13         }
14         return view(view: 'auth.login');
15     }
16
17     public function postlogin(Request $request): JsonResponse|mixed|Redirector|RedirectRes...
18     {
19         if ($request->ajax() || $request->wantsJson()) {
20             $credentials = $request->only(keys: 'username', 'password');
21
22             if (Auth::attempt(credentials: $credentials)) {
23                 return response()->json(data: [
24                     'status' => true,
25                     'message' => 'Login Berhasil',
26                     'redirect' => url(path: '/')
27                 ]);
28             }
29             return response()->json(data: [
30                 'status' => false,
31                 'message' => 'Login Gagal'
32             ]);
33         }
34         return redirect(to: 'login');
35     }
36
37     public function logout(Request $request): Redirector|RedirectResponse
38     {
39         Auth::logout();
40
41         $request->session()->invalidate();
42         $request->session()->regenerateToken();
43         return redirect(to: 'login');
44     }
45 }
```

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=fallba
ack">
    <!-- Font Awesome -->
    <link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
    <!-- icheck bootstrap -->
    <link rel="stylesheet" href="{{ asset('plugins/icheck-bootstrap/icheck-bootstrap.min.css')
}}">
    <!-- SweetAlert2 -->
    <link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-
4.min.css') }}">
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
</head>
```



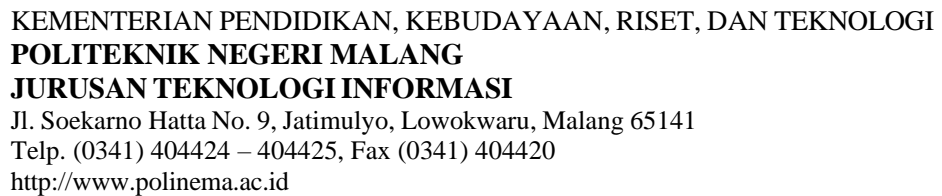
```
<body class="hold-transition login-page">
<div class="login-box">
  <!-- /.login-logo -->
  <div class="card card-outline card-primary">
    <div class="card-header text-center"><a href="{{ url('/') }}"
class="h1"><b>Admin</b><b>LTE</b></a></div>
    <div class="card-body">
      <p class="login-box-msg">Sign in to start your session</p>
      <form action="{{ url('login') }}" method="POST" id="form-login">
        @csrf
        <div class="input-group mb-3">
          <input type="text" id="username" name="username" class="form-control"
placeholder="Username">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-envelope"></span>
            </div>
          </div>
          <small id="error-username" class="error-text text-danger"></small>
        </div>
        <div class="input-group mb-3">
          <input type="password" id="password" name="password" class="form-control"
placeholder="Password">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-lock"></span>
            </div>
          </div>
          <small id="error-password" class="error-text text-danger"></small>
        </div>
        <div class="row">
          <div class="col-8">
            <div class="icheck-primary">
              <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
            </div>
          </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
            });
          }
        }
      });
      return false;
    },
    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>
</body>
</html>
```

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

```

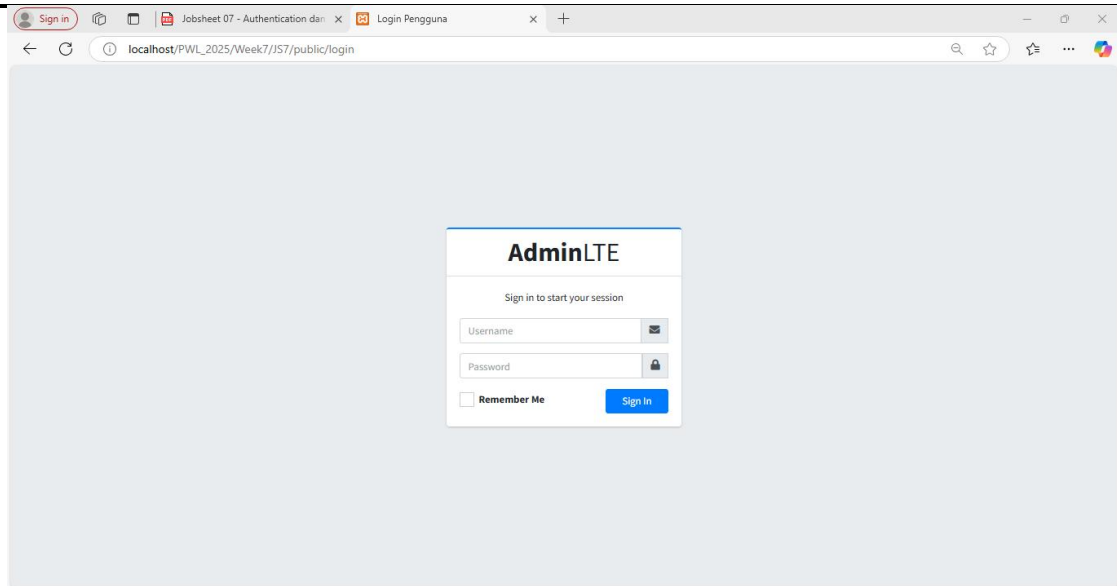
J57 > routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\UserController;
4  use App\Http\Controllers\SupplierController;
5  use App\Http\Controllers\BarangController;
6  use App\Http\Controllers\KategoriController;
7  use App\Http\Controllers\LevelController;
8  use App\Http\Controllers\WelcomeController;
9  use App\Http\Controllers\AuthController;
10 use Illuminate\Support\Facades\Route;
11
12 Route::pattern(key: 'id', pattern: '[0-9]+'); // Pastikan parameter {id} hanya berupa angka
13
14 // Rute otentikasi
15 Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login');
16 Route::post(uri: 'login', action: [AuthController::class, 'postlogin']);
17 Route::get(uri: 'logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth');
18
19 // Semua rute di bawah ini hanya bisa diakses jika sudah login
20 Route::middleware(middleware: 'auth')->group(callback: function (): void {
21     //masukkan semua route yang perlu autentikasi di sini
22 });

```

Hal. 9 / 22

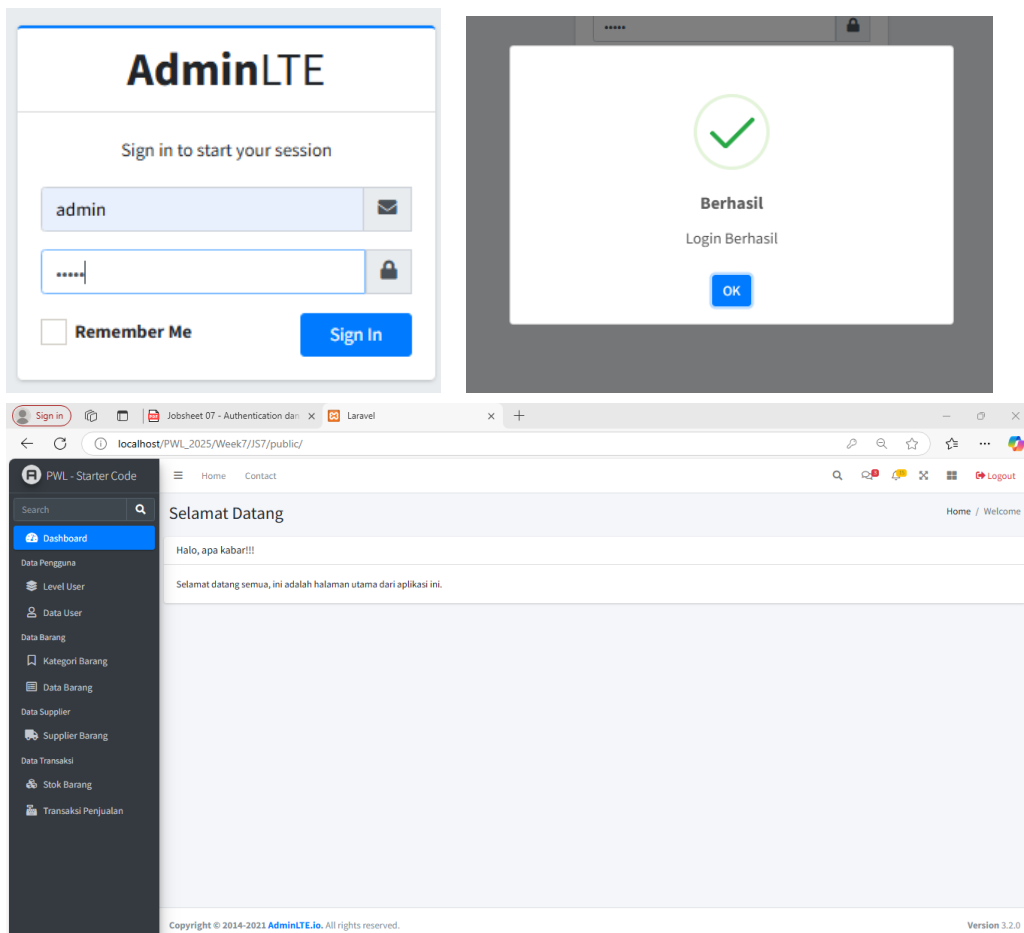


KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>



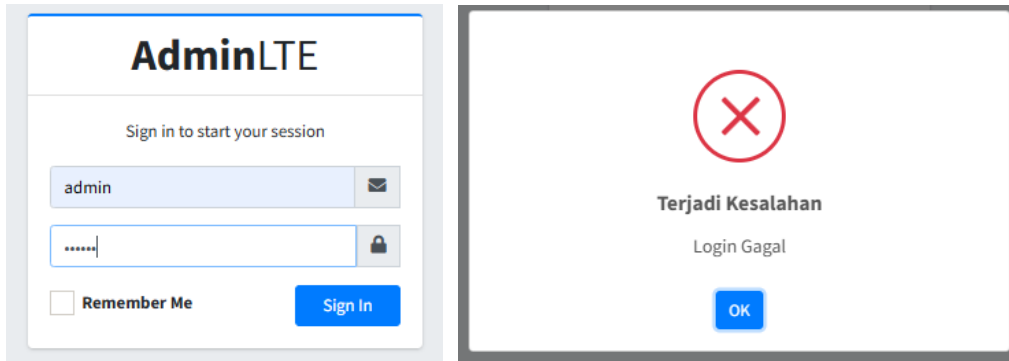
Tugas 1 – Implementasi Authentication :

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





➤ Gagal login



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

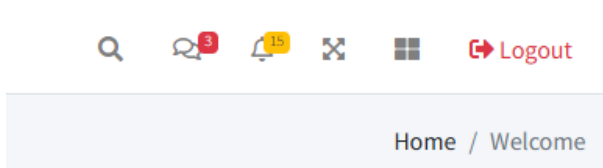
➤ layouts/header.blade.php

```
134 <li class="nav-item">
135 <a href="#" class="nav-link text-danger"
136   onclick="event.preventDefault(); document.getElementById('logout-form').submit();"
137   <i class="fas fa-sign-out-alt"></i> Logout
138 </a>
139 <form id="logout-form" action="{{ route(name: 'logout') }}" method="POST" style="display: none;"
140   @csrf
141 </form>
142 </li>
143 </ul>
144 </nav>
```

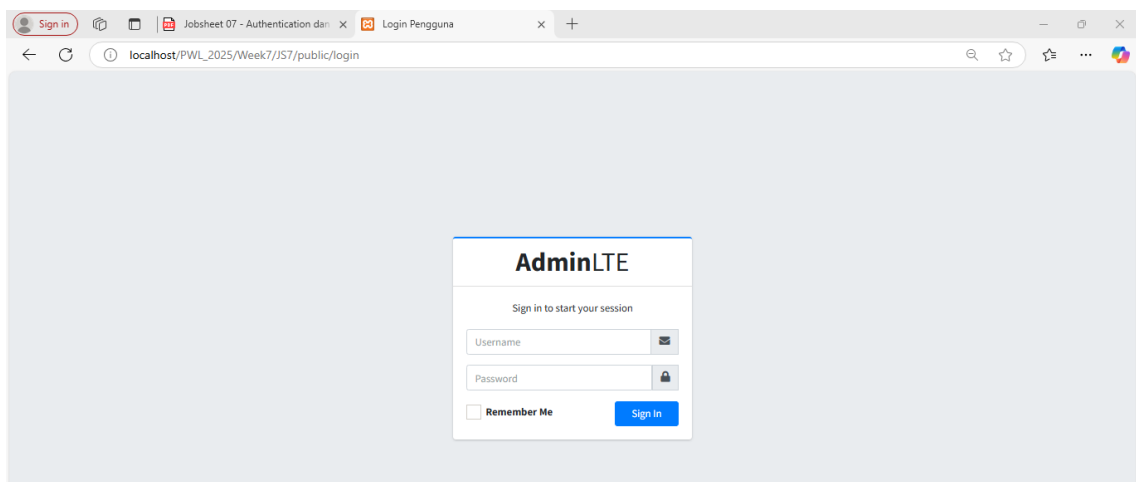
➤ routes/web.php

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

➤ tampilan



➤ ketika memencet tombol logout maka akan kembali pada halaman login



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



B. Implementasi *Authorization* di Laravel

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

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

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

Praktikum 2 – Implementasi *Authorization* di Laravel dengan Middleware

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

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

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

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

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



```
28      /**
29       * Mendapatkan nama role
30       */
31      0 references | 0 overrides
32      public function getRoleName(): string
33      {
34          return $this->level->level_nama;
35      }
36      /**
37       * Cek apakah user memiliki role tertentu
38       */
39      0 references | 0 overrides
40      public function hasRole($role): bool
41      {
42          return $this->level->level_kode == $role;
43      }
44  }
```

- `getRoleName()` → Mengembalikan nama role pengguna berdasarkan properti `level_nama` dari objek `level`.
- `hasRole($role)` → Memeriksa apakah role pengguna sesuai dengan kode yang diberikan (`level_kode`). Mengembalikan `true` jika cocok, `false` jika tidak.

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`

```
C:\laragon\www\PWL_2025\Week7\JS7>php artisan make:middleware AuthorizeUser
INFO  Middleware [C:\laragon\www\PWL_2025\Week7\JS7\app\Http\Middleware\AuthorizeUser.php] created successfully.
```

Setelah dibuat, *middleware* ini bisa digunakan untuk membatasi akses berdasarkan role pengguna atau aturan tertentu.

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

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



```
Week7 > JS7 > app > Http > Middleware > AuthorizeUser.php > PHP > AuthorizeUser
1  <?php
2  namespace App\Http\Middleware;
3
4  use Closure;
5  use Illuminate\Http\Request;
6  use Symfony\Component\HttpFoundation\Response;
7
8  0 references | 0 implementations
9  class AuthorizeUser
10
11  /**
12   * Handle an incoming request.
13   *
14   * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
15   */
16  0 references | 0 overrides
17  public function handle(Request $request, Closure $next, $role = ''): Response
18  {
19      $user = $request->user(); // ambil data user yg login
20      // fungsi user() diambil dari UserModel.php
21      if ($user->hasRole($role)) { // cek apakah user punya role yg diinginkan
22          return $next($request);
23      }
24
25      // jika tidak punya role, maka tampilkan error 403
26      abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
```

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

```
0 references
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,
```

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"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	4	CUS	Customer	NULL	NULL
<input type="checkbox"/>	6	CEO	CEO	2025-03-25 12:52:13	2025-03-25 12:52:13

☐ Check all With selected:

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

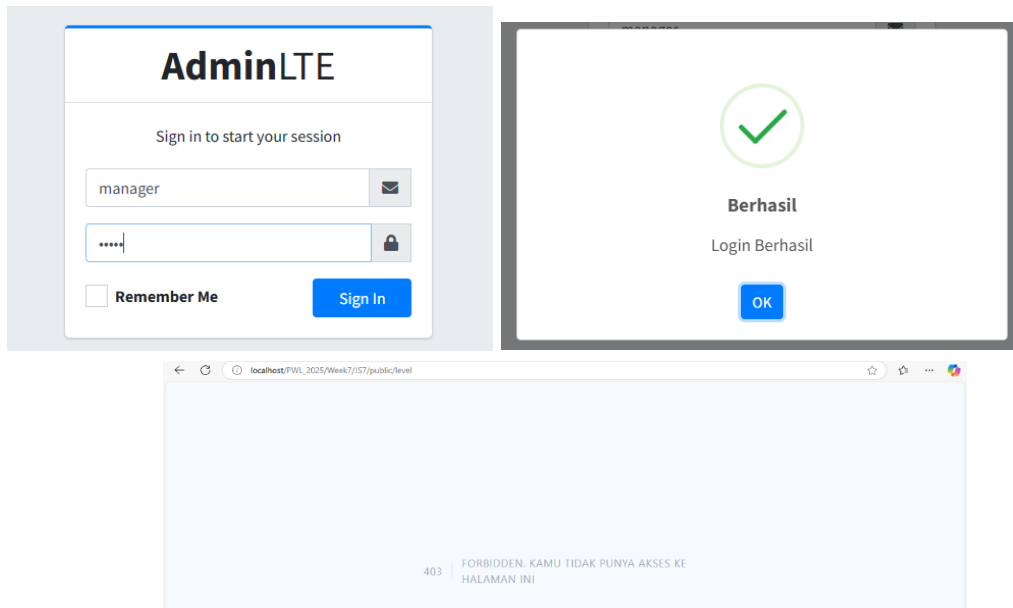
```
123 // artinya semua route di dalam group ini harus punya role ADM (Administrator)
124 Route::middleware(['authorize:ADM'])->group(function(): void{
125     Route::get('/level', [LevelController::class, 'index']);
126     Route::post('/level/list', [LevelController::class, 'list']); // untuk list json datatables
127     Route::get('/level/create', [LevelController::class, 'create']);
128     Route::post('/level', [LevelController::class, 'store']);
129     Route::get('/level/{id}/edit', [LevelController::class, 'edit']); // untuk tampilkan form edit
130     Route::put('/level/{id}', [LevelController::class, 'update']); // untuk proses update data
131     Route::delete('/level/{id}', [LevelController::class, 'destroy']); // untuk proses hapus data
132 });
133 });
134 });
```

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

➤ Login selain level administrator



Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?

Pemahaman dari Praktikum 2:

- Authorization digunakan untuk mengatur akses pengguna berdasarkan role atau level tertentu.
- Middleware berperan sebagai alat utama dalam Laravel untuk memverifikasi izin pengguna sebelum memberikan akses ke halaman atau fitur tertentu.
- Pada UserModel.php, ditambahkan fungsi getRoleName() untuk mendapatkan nama role pengguna serta hasRole() untuk mengecek apakah pengguna memiliki role tertentu.
- Middleware AuthorizeUser.php dibuat untuk mengecek role pengguna sebelum mengizinkan akses ke halaman tertentu.
- Middleware ini kemudian didaftarkan di Kernel.php, sehingga dapat digunakan dalam route dengan alias authorize.
- Middleware diterapkan dalam route grouping untuk membatasi akses hanya kepada pengguna dengan role tertentu. Dalam kasus ini, hanya pengguna dengan role "ADM" yang dapat mengakses halaman tersebut.
- Dengan menerapkan authorization berbasis role, akses ke fitur-fitur penting dapat dikendalikan sehingga hanya pengguna yang ditentukan yang bisa menggunakannya.

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;
}
```

```
28 /**
29  * Mendapatkan nama role
30  */
31 0 references | 0 overrides
32 public function getRoleName(): string
33 {
34     return $this->level->level_nama;
35 }
36
37 /**
38  * Cek apakah user memiliki role tertentu
39  */
40 0 references | 0 overrides
41 public function hasRole($role): bool
42 {
43     return $this->level->level_kode == $role;
44 }
45
46 0 references | 0 overrides
47 public function getRole(): mixed
48 {
49     return $this->level->level_kode;
50 }
```



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 }
```

```
Week7 > JS7 > app > Http > Middleware > AuthorizeUser.php > PHP > AuthorizeUser
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 = $request->user()->getRole(); // Ambil data level_kode dari user yang login
18         if (in_array($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(code: 403, message: '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
});
```

```
134 // artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
135 Route::middleware(['authorize:ADM,MNG'])->group(function(): void{
136     Route::get('/barang', [BarangController::class, 'index']);
137     Route::post('/barang/list', [BarangController::class, 'list']);
138     Route::get('/barang/create_ajax', [BarangController::class, 'create_ajax']); // ajax form create
139     Route::post('/barang_ajax', [BarangController::class, 'store_ajax']); // ajax store
140     Route::get('/barang/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // ajax form edit
141     Route::put('/barang/{id}/update_ajax', [BarangController::class, 'update_ajax']); // ajax update
142     Route::get('/barang/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // ajax form confirm
143     Route::delete('/barang/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // ajax delete
144 });
145
146 };
```

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

```
Week7 > JS7 > app > Http > Middleware > AuthorizeUser.php > PHP > AuthorizeUser
2 namespace App\Http\Middleware;
3
4 use Closure;
5 use Illuminate\Http\Request;
6 use Symfony\Component\HttpFoundation\Response;
7
8 /**
9  * reference [0 implementations]
10  */
11 class AuthorizeUser
12 {
13     /**
14      * Handle an incoming request.
15      *
16      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
17      */
18     public function handle(Request $request, Closure $next, ...$roles): Response
19     {
20         $user_role = $request->user()->getRole(); // Ambil data level_kode dari user yang login
21         if (in_array($user_role, $roles)) { // Cek apakah level_kode user ada di dalam array roles
22             return $next($request); // Jika ada, maka lanjutkan request
23         }
24         // jika tidak punya role, maka tampilkan error 403
25         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
26     }
27 }
```

Kode di atas adalah middleware Laravel bernama AuthorizeUser, yang digunakan untuk membatasi akses pengguna berdasarkan peran (role).

Penjelasan:

- Mengambil role pengguna yang sedang login dengan `$request->user()->getRole()`.
- Memeriksa apakah role pengguna termasuk dalam daftar yang diizinkan (`$roles`).
- Jika role cocok, request dilanjutkan (`$next($request)`).
- Jika tidak cocok, akses ditolak dengan error 403 (Forbidden).

➤ Models/UserModel.php

```
43
44 0 references [0 overrides]
45 public function getRole(): mixed
46 {
47     return $this->level->level_kode;
48 }
```

Fungsi `getRole()` untuk mengambil kode role pengguna dari properti `level_kode` dalam objek `level`.

Penjelasan:

- Mengakses `$this->level->level_kode`.
- Mengembalikan kode role pengguna, misalnya "ADM", "USR", atau kode lain sesuai dengan sistem role yang digunakan.
- Digunakan dalam middleware untuk pengecekan role sebelum mengizinkan akses ke suatu fitur atau halaman.

➤ routes/web.php

```
130 // artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
131 Route::middleware(['authorize:ADM,MNG'])->group(function() {
132     Route::get('/barang', [BarangController::class, 'index']);
133     Route::post('/barang/list', [BarangController::class, 'list']);
134     Route::get('/barang/create_ajax', [BarangController::class, 'create_ajax']); // ajax form create
135     Route::post('/barang_ajax', [BarangController::class, 'store_ajax']); // ajax store
136     Route::get('/barang/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // ajax form edit
137     Route::put('/barang/{id}/update_ajax', [BarangController::class, 'update_ajax']); // ajax update
138     Route::get('/barang/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // ajax form confirm
139     Route::delete('/barang/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // ajax delete
140 });
```



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
 - Ketika login administrator dan manager

The first set of screenshots shows the AdminLTE login interface. The left panel shows the login form for 'admin' with a password field. The right panel shows the login form for 'manager' with a password field. The third panel shows a successful login message: 'Berhasil Login Berhasil' with an 'OK' button.

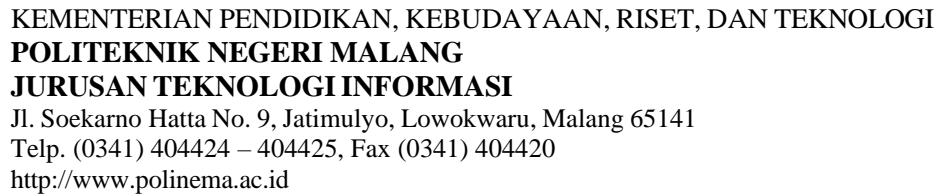
The second set of screenshots shows the AdminLTE dashboard for a user with the role of 'Data Barang'. The dashboard displays a table of goods (Data Barang) with columns: ID, Kode Barang, Nama Barang, Nama Kategori, Harga Beli, Harga Jual, and Aksi. The table contains 10 rows of data. The dashboard also includes a sidebar with navigation links and a top navigation bar.

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	Pegok Bayi	Baby & Kids	50000	55000	Detail Edit Hapus
8	B008	Susu Formula	Baby & Kids	60000	65000	Detail Edit Hapus
9	B009	Lampu LED	Electronics	25000	30000	Detail Edit Hapus
10	B010	Charger HP	Electronics	35000	40000	Detail Edit Hapus

- Ketika login selain level administrator dan manager.

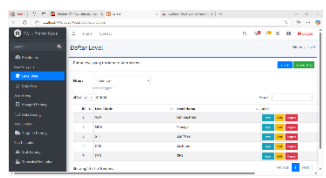
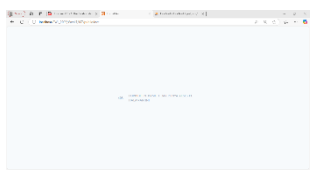
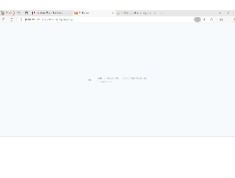
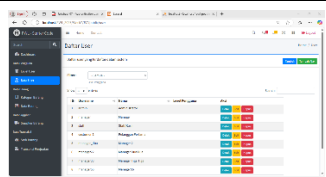
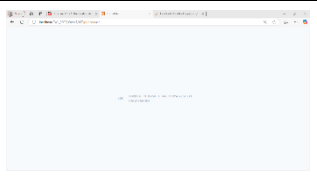
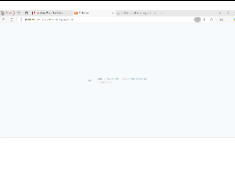
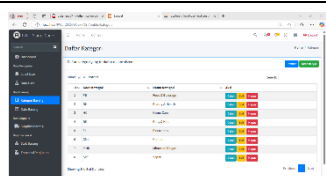
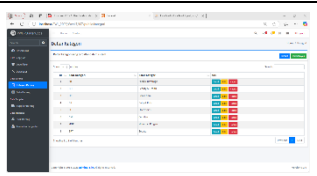
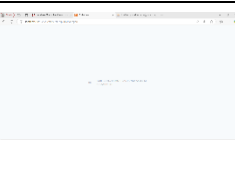
The first set of screenshots shows the AdminLTE login interface for a user with the role of 'staff'. The login form shows the username 'staff' and a password field. The second panel shows a successful login message: 'Berhasil Login Berhasil' with an 'OK' button.

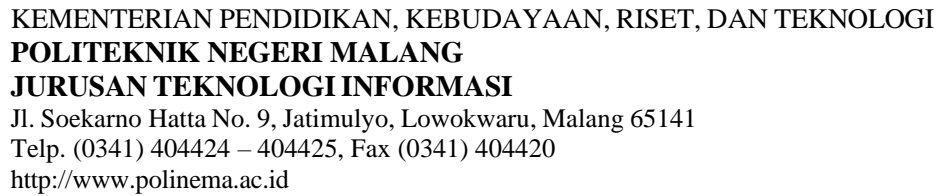
The second set of screenshots shows a 403 Forbidden error page. The message reads: '403 FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI'.



➤ routes/web.php

[illegible]

Menu	admin	manager	staff
Data user			
Data level			
Data kategori			



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

1. Silahkan implementasikan form untuk registrasi user.

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

- [illegible]

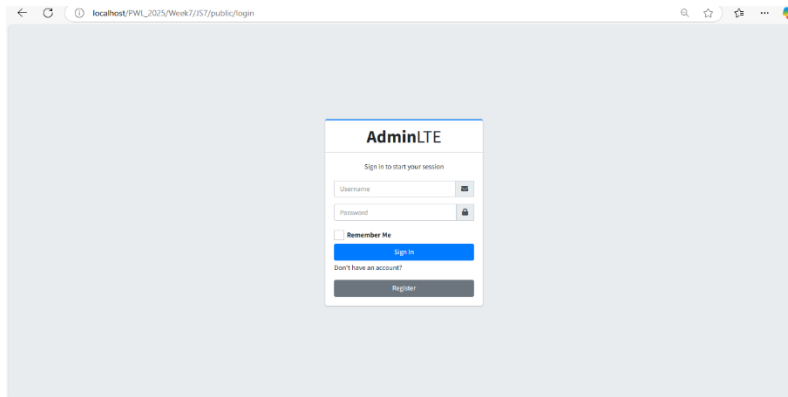


➤ views/auth/login.blade.php

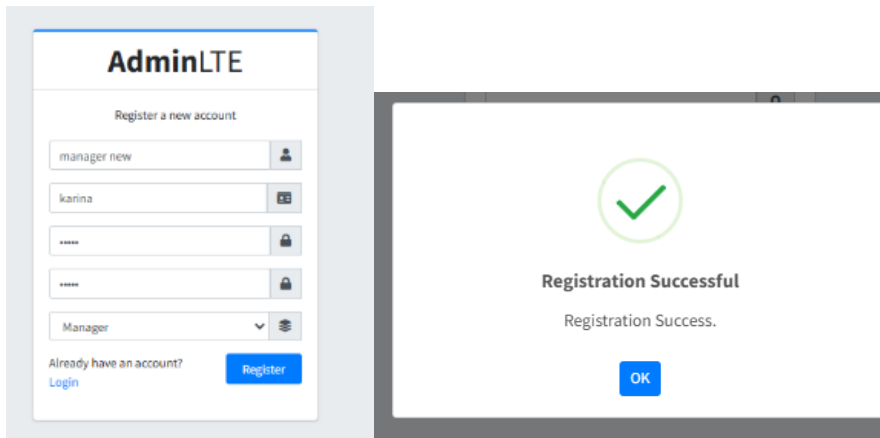
```
61 </div>
62 <div class="col-12 mt-1">
63 <p>Don't have an account?</p>
64 Register
65 </div>
66 <!-- /.col -->
```

2. Screenshot hasil yang kalian kerjakan

➤ Hasil



➤ Coba register



➤ Cek database

|                                                                           | user_id | level_id | username    | nama              | password                                                 | created_at          | updated_at          |
|---------------------------------------------------------------------------|---------|----------|-------------|-------------------|----------------------------------------------------------|---------------------|---------------------|
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 1       | 1        | admin       | Administrator     | \$2y\$12\$Tbat8fyofozv0hgAFeRH45K47QNS6mqEoluzX...       | NULL                | NULL                |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 2       | 2        | manager     | Manager           | \$2y\$12\$gQBSTAIxlooEuHvLnafseOjCBH4Mx9b6BpinpuDaW...   | NULL                | NULL                |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 3       | 3        | staff       | StaffKasir        | \$2y\$12\$2Fwun3kRudRy5PR.mdYfCp6lqR2zbox0.OGV5FP2...    | NULL                | NULL                |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 5       | 4        | customer-1  | Pelanggan Pertama | \$2y\$12\$vmid3ooXUTDf.bzr6lqgFwe9ZdvZEfetzpkeYGeMvZC... | NULL                | 2025-03-04 05:03:44 |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 6       | 2        | manager_dua | Manager 2         | \$2y\$12\$1TW5LH6TTgza6S.wBMm1DuOYKJGMWLeuKEglLjZ2...    | 2025-03-08 07:20:17 | 2025-03-08 07:20:17 |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 8       | 2        | manager22   | Manager Dua Dua   | \$2y\$12\$F8VWXnqzU5ys44Ms2Ofsu8m4DetS7KgmCt5SwQmH...    | 2025-03-08 10:03:45 | 2025-03-08 10:03:45 |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 11      | 2        | manager33   | Manager Tiga Tiga | \$2y\$12\$zltbz2OUGE8JD4HQZuRunYalc.a37ToASXE08g77...    | 2025-03-08 10:37:57 | 2025-03-08 10:37:57 |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 12      | 2        | manager55   | Manager55         | \$2y\$12\$UPF6ASqRCIEBMCETTK4bsuFuUq7aFWqjooHhUzpw09...  | 2025-03-08 10:56:31 | 2025-03-08 10:56:32 |
| <input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/> | 16      | 2        | manager new | karina            | \$2y\$12\$8smsW3oinkdthOP9Gbku4bYy1FV3g1rEpx.0S2xW...    | 2025-03-29 16:54:38 | 2025-03-29 16:54:38 |

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

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