



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

Khuzaima Filla Januartha
SIB2A/17

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 meakukan 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         ],  
67
```

```
62     'providers' => [  
63         'users' => [  
64             'driver' => 'eloquent',  
65             'model' => App\Models\UserModel::class,  
66         ],  
67
```

2. Selanjutnya kita modifikasi sedikit pada **UserModel.php** untuk bisa melakukan proses 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');  
    }  
}
```



```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
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 ditampilkan 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');
    }
}
```

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');
    }
}
```




```
class AuthController extends Controller
{
    public function login()
    {
        // Jika sudah login, maka redirect ke halaman home
        if (Auth::check()) {
            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');
    }
}
```

- 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**)



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>

```
<!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=fallb
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>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({ header
  s: {
    '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
          $('<div>.error-text</div>').text('');
          $.each(response.msgField, function(prefix, val) {
            $('<div>#error-<div>'+prefix</div>').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('<div>.input-group</div>').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

```
<?php

use App\Http\Controllers\UserController;
use App\Http\Controllers\SupplierController;
use App\Http\Controllers\BarangController;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers>WelcomeController;
use Illuminate\Support\Facades\Route;

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

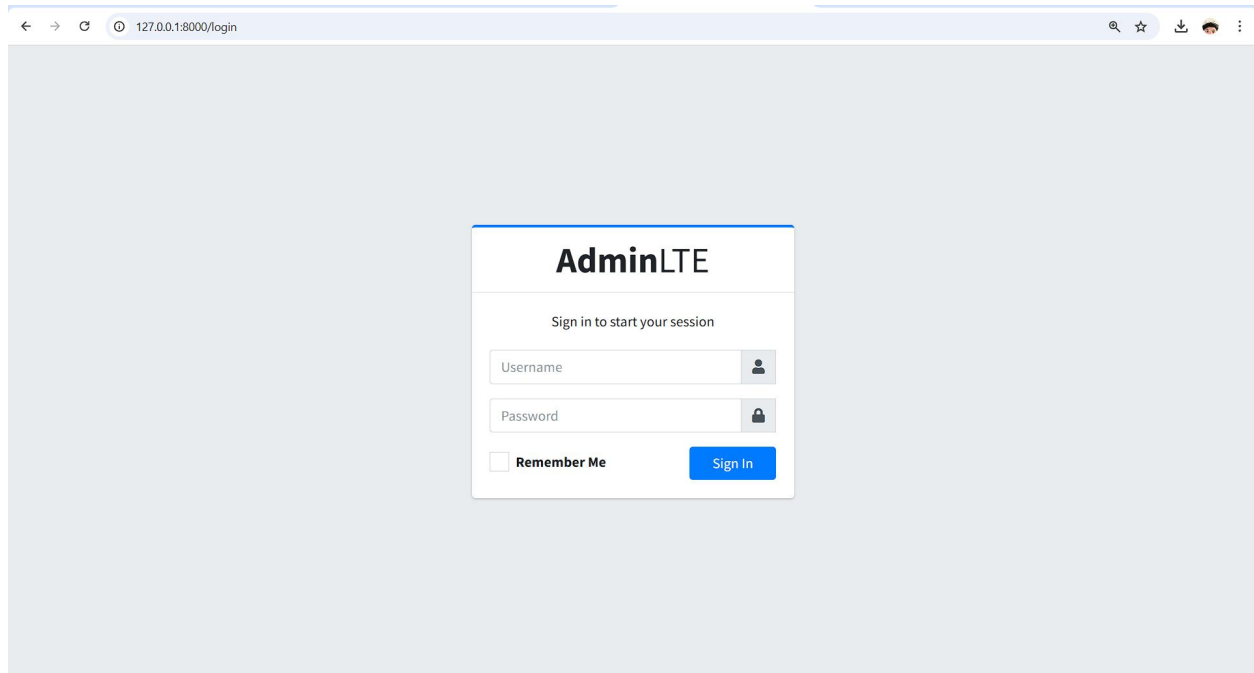
Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');

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

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



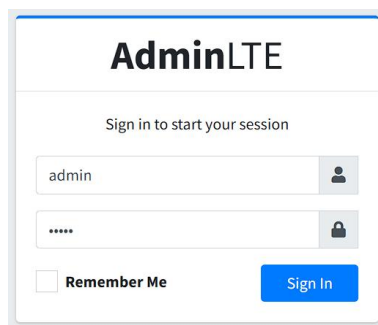
KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>



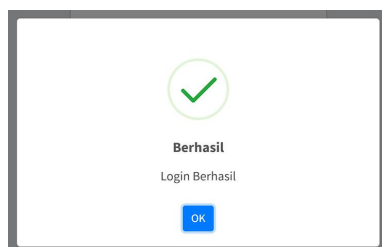
Tugas 1 – Implementasi Authentication :

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

Login



Disini saya login sebagai admin



Login berhasil



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>

Masuk ke laman dashboard

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

Logout

Berhasil logout dan kembali ke laman 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>

3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
 - a) Mengkonfigurasi dan memodifikasi auth dengan UserModel untuk tabel m_user
 - b) Membuat AuthController untuk mengapply function login/logout
 - c) Menambah route untuk login/logout
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;
}
```




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

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`

```
PS C:\laragon\www\PWL_2025\Minggu7\Jobsheet7> php artisan make:middleware
AuthorizeUser

INFO Middleware [C:\laragon\www\PWL_2025\Minggu7\Jobsheet7\app\Http\M
iddleware\AuthorizeUser.php] created successfully.
```

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 }
```

```
<?php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
     */
    public function handle(Request $request, Closure $next, $role = ''): Response
    {
        $user = $request->user(); // ambil data user yg login

        // fungsi user() diambil dari UserModel.php
        if ($user->hasRole($role)) { // cek apakah user punya role yg diinginkan
            return $next($request);
        }

        // jika tidak punya role, maka tampilkan error 403
        abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

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

```
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
```



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>

```
protected $middlewareAliases = [  
    'auth' => \App\Http\Middleware\Authenticate::class,  
    'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat  
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
```

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	deleted_at
1	ADM	Administrator	NULL	NULL	NULL
2	MNG	Manager	NULL	NULL	NULL
3	STF	Staf	NULL	2024-08-16 01:49:20	NULL
4	KSR	Kasir	NULL	NULL	NULL



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

```
Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu
    Route::get('/', [WelcomeController::class, 'index']);
    // route Level

    // 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
    });

    // route Kategori
```

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.

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

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

Tugas 2 – Implementasi Authorization :


1. Apa yang kalian pahami pada praktikum 2 ini?
Implementasi Authorization dalam laravel yang digunakan untuk mengotorisasi level/jenis pengguna sebagai klasifikasi hak akses dalam sistem.
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
 - a) Membuat middleware dengan nama `AuthorizeUser` untuk mengotorisasi jenis pengguna yang nantinya akan diklasifikasikan hak aksesnya berdasarkan route/akses tertentu.
 - b) Mengaplikasi middleware yang telah dibuat ke kernel agar bisa diakses
 - c) Mengotorisasi route dengan `ADM/Administrator`, yang artinya semua route hanya bisa diakses oleh jenis pengguna “`ADM/Administrator`” saja.
3. Submit kode untuk implementasi Authorization pada repository github kalian.




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>

AdminLTE

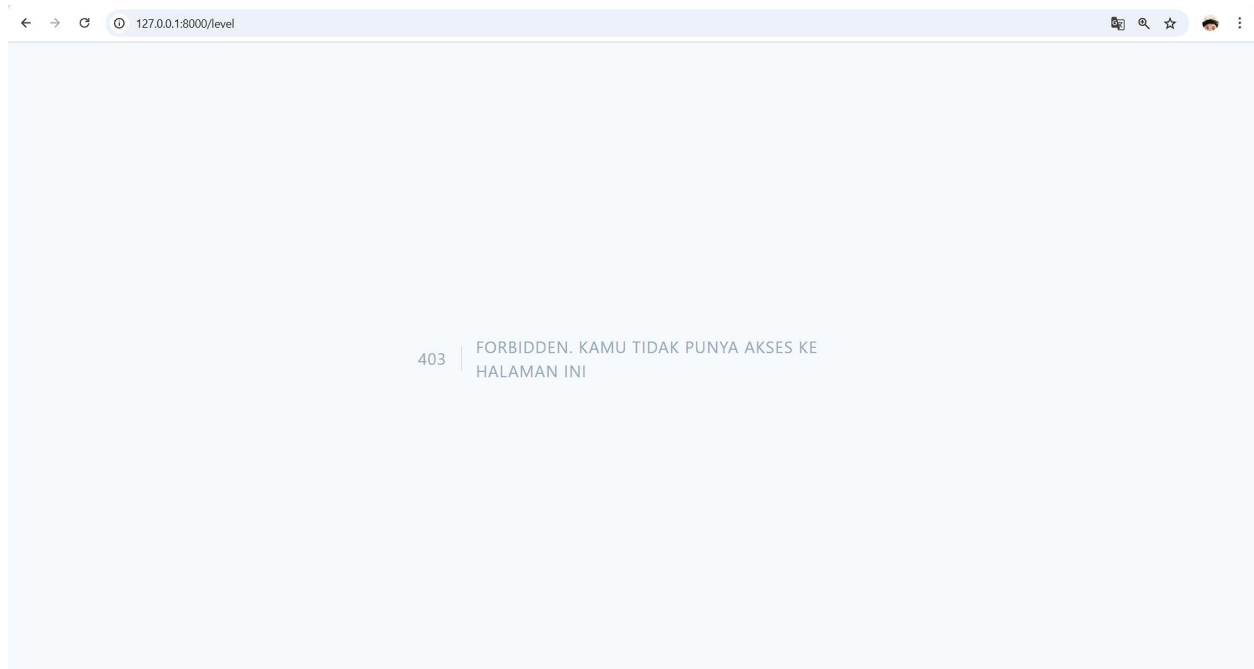
Sign in to start your session





☐ Remember Me

Disini saya login sebagai manager



Disaat ingin melihat Data Level, page mentrigger Error 404. karena kita login sebagai “Manager”, sedangkan yang memiliki hak akses ke page Data level adalah “Administrator”



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 Authorization 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;
}
```

```
/**
 * 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 }
```

```
class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
     */
    public function handle(Request $request, Closure $next, ...$roles): Response
    {
        $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
        if (in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
            return $next($request); // jika ada, maka lanjutkan request
        }
        // jika tidak punya role, maka tampilkan error 403
        abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

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



```
// 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
});
```

```
web.php > ...
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::get('/', [WelcomeController::class, 'index']);

    // route Level
    // 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
    });

    // 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

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




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

```
// 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
});
});
```

2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
 - a) Menambahkan function getRole() untuk mengambil kode level
 - b) Memodifikasi middleware Authorize user agar bisa digunakan untuk multilevel auth
 - c) Mengkonfigurasi route yang hanya dapat diakses oleh role ADM/Administrator dan MNG/Manager.
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User

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



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

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




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

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

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



Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

Controller

```
public function register()
{
    $levels = LevelModel::all();
    return view('auth.register', ['level' => $levels]);
}

public function postRegister(Request $request)
{
    $request->validate([
        'username' => 'required|string|min:3|unique:m_user,username',
        'nama' => 'required|string|max:50',
        'password' => 'required|min:4',
        'level_id' => 'required|integer',
    ]);

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

    return redirect('login')->with('success', 'Registrasi berhasil ditambahkan');
}
```

Views

```
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="csrf-token" content="{{ csrf_token() }}">
<title>Registrasi Pengguna</title>

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
<link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
<link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
<link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
<link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
</head>
```




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>

```
<body class="hold-transition login-page">
  <div class="login-box">
    <div class="card card-outline card-primary">
      <div class="card-header text-center">
        <a href="{{ url('/') }}" class="h1"><b>Admin</b></a>
      </div>
      <div class="card-body">
        <p class="login-box-msg">Daftar Pengguna Baru</p>
        <form action="{{ url('register') }}" method="POST" id="form-register">
          @csrf
          <div class="form-group">
            <label>Pilih Role</label>
            <select name="level_id" id="level_id" class="form-control" required>
              <option value="">- Pilih Level -</option>
              @foreach($level as $l)
                <option value="{{ $l->level_id }}">{{ $l->level_nama }}</option>
              @endforeach
            </select>
            <small id="error-level_id" class="error-text form-text text-danger"></small>
          </div>
          <div class="form-group">
            <label>Username</label>
            <input type="text" name="username" id="username" class="form-control" required minlength="4"
              maxlength="20" placeholder="Masukkan Username">
            <small id="error-username" class="error-text form-text text-danger"></small>
          </div>
          <div class="form-group">
            <label>Nama</label>
            <input type="text" name="nama" id="nama" class="form-control" required maxlength="100" placeholder="Masukkan Nama">
            <small id="error-nama" class="error-text form-text text-danger"></small>
          </div>
          <div class="form-group">
            <label>Password</label>
            <input type="password" name="password" id="password" class="form-control" required minlength="5"
              maxlength="20" placeholder="Masukkan Password">
            <small id="error-password" class="error-text form-text text-danger"></small>
          </div>
          <div class="row justify-content-end">
            <div class="col-auto">
              <button type="submit" class="btn btn-primary btn-block">Sign Up</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

<script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
<script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<script src="{{ asset('adminlte/plugins/jquery-validation/additional-methods.min.js') }}"></script>
<script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
<script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
```




```
<script>
    $(document).ready(function () {
        $("#form-register").validate({
            rules: {
                username: { required: true, minlength: 4, maxlength: 20 },
                nama: { required: true, maxlength: 50 },
                password: { required: true, minlength: 4, maxlength: 20 },
                password_confirmation: { equalTo: "[name='password']" },
                level_id: { required: true, number: true }
            },
            submitHandler: function (form) {
                $.ajax({
                    headers: {
                        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
                    },
                    url: form.action,
                    type: form.method,
                    data: $(form).serialize(),
                    success: function (response) {
                        if (response.status) {
                            Swal.fire({
                                icon: 'success',
                                title: 'Registrasi Berhasil',
                                text: response.message,
                            }).then(() => {
                                if (response.redirect) {
                                    window.location = response.redirect;
                                }
                            });
                        } else {
                            $('.text-danger').text('');
                            $.each(response.errors, function (key, val) {
                                $('#error-' + key).text(val[0]);
                            });
                            Swal.fire({
                                icon: 'error',
                                title: 'Terjadi Kesalahan',
                                text: response.message
                            });
                        }
                    }
                });
                return false;
            },
            errorElement: 'span',
            errorPlacement: function (error, element) {
                error.addClass('invalid-feedback');
                element.closest('.input-group').append(error);
            },
            highlight: function (element) {
                $(element).addClass('is-invalid');
            },
            unhighlight: function (element) {
                $(element).removeClass('is-invalid');
            }
        });
    });
</script>
</body>
</html>
```



Route

```
//Register
Route::get('register', [AuthController::class, 'register']);
Route::post('register', [AuthController::class, 'PostRegister']);
```

2. Screenshot hasil yang kalian kerjakan

The image displays two screenshots of the AdminLTE web application. The left screenshot shows the login page with the title 'AdminLTE' and the subtitle 'Sign in to start your session'. It features input fields for 'Username' and 'Password', a 'Remember Me' checkbox, and a 'Sign In' button. A link 'Don't Have an Account?' is at the bottom. The right screenshot shows the registration page with the title 'AdminLTE' and the subtitle 'Daftar Pengguna Baru'. It includes a 'Pilih Role' dropdown menu (currently showing '- Pilih Level -'), and input fields for 'Username', 'Nama', and 'Password'. There is a 'Sign Up' button and a 'Back to Login' link. Below these, a browser window shows the registration form filled out with the role 'Administrator', username 'cobaregist', name 'Coba regist user', and a masked password. The browser's address bar shows '127.0.0.1:8000/register'.



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>

<input type="checkbox"/>	Edit	Copy	Delete	25	1 cobaregist	Coba regist user	\$2y\$10\$yUaCET5HctsvcybWpSiQM.U.qqj/CkM317ql0IR6Q.av...	2025-03-29 07:42:39	2025-03-29 07:42:39
<input type="checkbox"/>	Check all	With selected:	Edit	Copy	Delete	Export			

Hasil regist tersimpan di Database

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

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