



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	: My Babby Findia R.S
No. Absen	: 16
Kelas	: SIB-2B

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 menggunakan 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     ],
```

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

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



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



```
Minggu7 > PWL_POS > app > Models > UserModel.php > PHP Intelephense > UserModel
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 74 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(LevelModel::class, 'level_id', 'level_id');
33     }
34 }
```

- 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');
    }
}
```



```
Minggu7 > PWL_POS > app > Http > Controllers > AuthController.php > PHP > AuthController > postlogin()
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\Auth;
7
8  0 references | 0 implementations
9  class AuthController extends Controller
10 {
11     0 references | 0 overrides
12     public function login(): Factory|Redirector|RedirectResponse|View
13     {
14         if (Auth::check()) { // jika sudah login, maka redirect ke halaman home
15             return redirect(to: '/');
16         }
17         return view(view: 'auth.login');
18     }
19     0 references | 0 overrides
20     public function postlogin(Request $request): JsonResponse|mixed|Redirector|RedirectRes...
21     {
22         if ($request->ajax() || $request->wantsJson()) {
23             $credentials = $request->only('username', 'password');
24             if (Auth::attempt($credentials)) {
25                 return response()->json([
26                     'status' => true,
27                     'message' => 'Login Berhasil',
28                     'redirect' => url(path: '/')
29                 ]);
30             }
31             return response()->json([
32                 'status' => false,
33                 'message' => 'Login Gagal'
34             ]);
35             return redirect(to: 'login');
36         }
37         0 references | 0 overrides
38         public function logout(Request $request): Redirector|RedirectResponse
39         {
40             Auth::logout();
41         }
42     }
```

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=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/bootstrap4.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></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="checkbox-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>
</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');  
}  
});  
});
```

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  
  
});
```

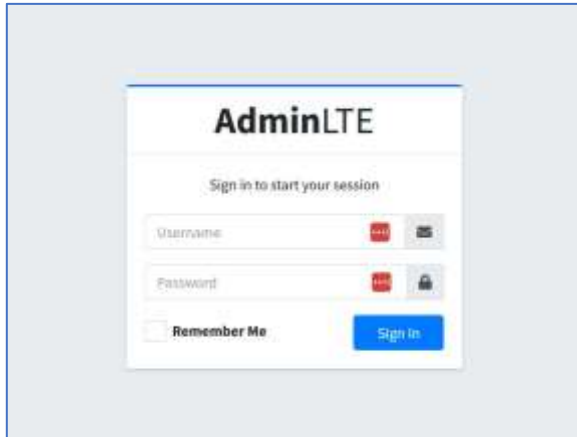
Minggu7 > PWL_POS > routes > web.php > ...

```
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\AuthController;  
7  use App\Http\Controllers\KategoriController;  
8  use App\Http\Controllers\LevelController;  
9  use App\Http\Controllers>WelcomeController;  
10 use Illuminate\Support\Facades\Route;  
11
```

```
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(): void { // artinya semua route di dalam group ini harus login dulu  
    // masukkan semua route yang perlu autentikasi di sini  
});
```



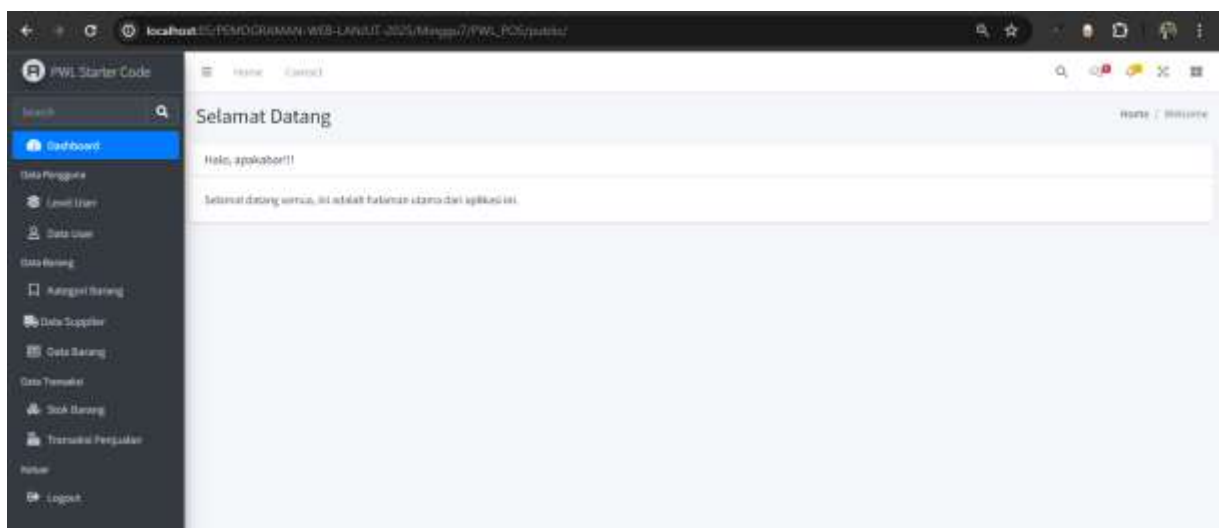
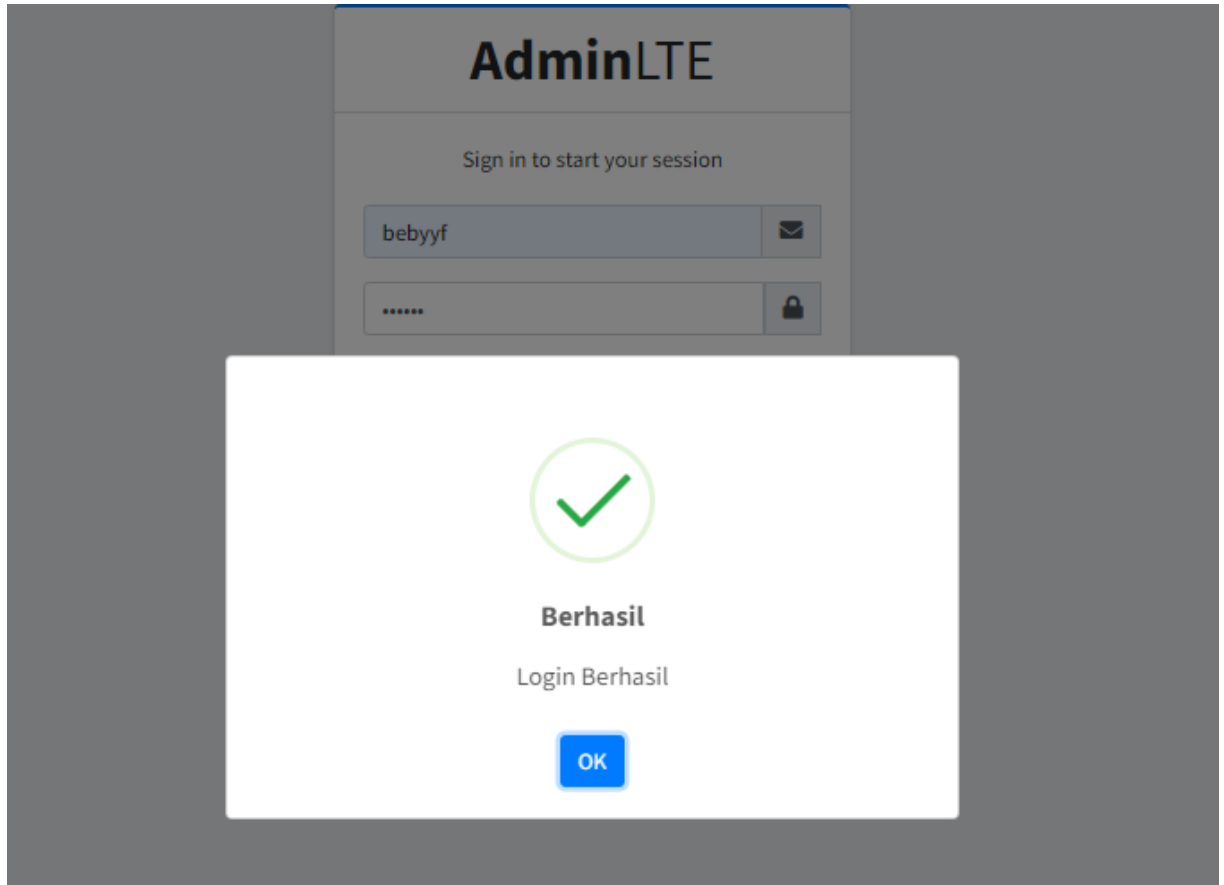
6. Ketika kita coba mengakses halaman localhost/PWL_POS/public maka akan tampil halaman awal untuk login ke aplikasi





Tugas 1 – Implementasi Authentication :

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



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



```
Minggu7 > PWL_POS > resources > views > layouts > sidebar.blade.php > div.sidebar
1  <div class="sidebar">
14  <nav class="mt-2">
15  <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-accordion="false">
76      <li class="nav-header">Keluar</li>
77      <li class="nav-item">
78          <a href="#" class="nav-link"
79              onclick="event.preventDefault(); document.getElementById('logout-form').submit();"
80              <i class="nav-icon fas fa-sign-out-alt"></i>
81              <p>Logout</p>
82          </a>
83          <form id="logout-form" action="{{ url(path: 'logout') }}" method="POST" style="display: none;"
84              @csrf
85          </form>
86      </li>
87  </ul>
88  </nav>
89  </div>
```





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

B. Implementasi Authorization di Laravel

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

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

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



Praktikum 2 – Implementasi *Authorizaton* 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;
}
```

```
/**
 * Mendapatkan nama role
 */
0 references | 0 overrides
public function getRoleName(): string
{
    return $this->level->level_nama;
}

/**
 * Cek apakah user memiliki role tertentu
 */
0 references | 0 overrides
public function hasRole($role): bool
{
    return $this->level->level_kode == $role;
}
}
```



- 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`

- 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 }
```



```
Minggu7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > ...
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  0 references | 0 implementations
10 class AuthorizeUser
11 {
12     /**
13      * Handle an incoming request.
14      *
15      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
16      */
17     0 references | 0 overrides
18     public function handle(Request $request, Closure $next, $role = ''): Response
19     {
20         $user = $request->user();
21         if ($user->hasRole($role)) {
22             return $next($request);
23         }
24         abort(code: 403, message: 'Forbidden. kamu tidak punya akses ke halaman ini');
25     }
26 }
27
```

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,
```

```
0 references
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'authorize' => \App\Http\Middleware\AuthorizeUser::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
    'signed' => \App\Http\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::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

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	STF	Staff/Kasir	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
```



```
Route::middleware(['authorize:ADM'])->group(function (): void {  
Route::group(['prefix' => 'level'], function (): void {  
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::delete('/{id}', [LevelController::class, 'destroy']);  
Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']);  
Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']);  
});  
});
```

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.

Show 10 entries

ID	Username	Nama	Level Pengguna	Aksi
1	admin	admin	Administrator	Detail Edit Hapus
2	manager	Manager	Manager	Detail Edit Hapus
3	staff	Staff/Kaur	Staff/Kaur	Detail Edit Hapus
4	manager_dua	Manager 2	Manager	Detail Edit Hapus
5	manager_tiga	Manager 3	Manager	Detail Edit Hapus
6	manager22	Manager Dua Dua	Manager	Detail Edit Hapus
7	manager11	Manager Tiga Tiga	Manager	Detail Edit Hapus
8	manager56	Manager55	Manager	Detail Edit Hapus
9	manager12	Manager11	Manager	Detail Edit Hapus
10	belbyy4	beli inda	Administrator	Detail Edit Hapus

Showing 1 to 10 of 11 entries

Previous 1 2 Next

AdminLTE

Sign in to start your session

belbyy4

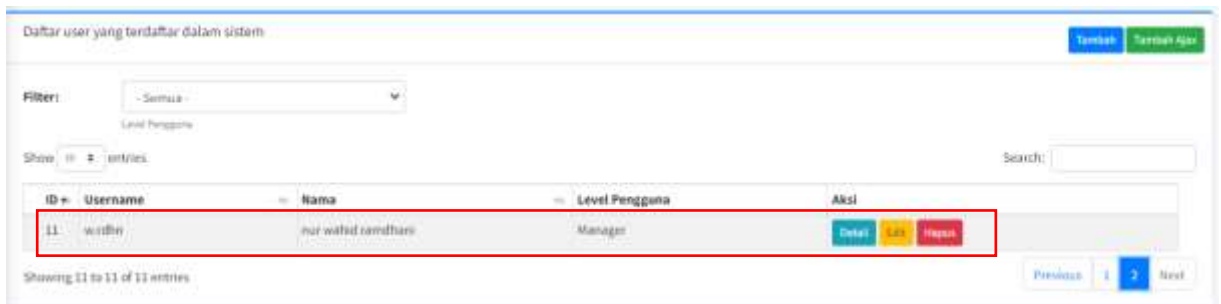
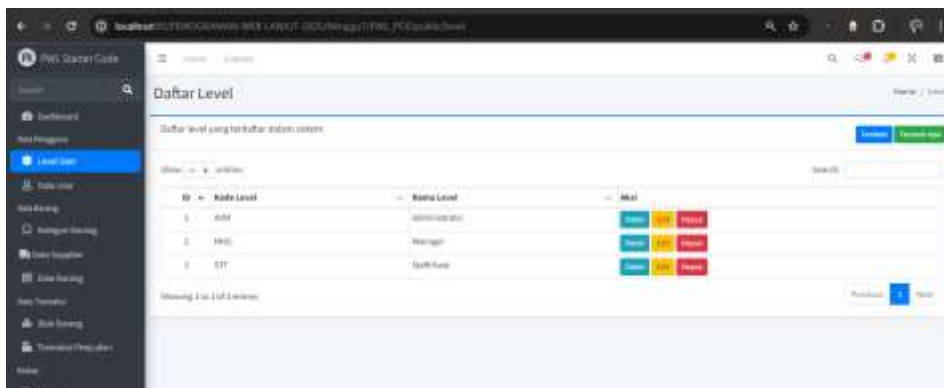
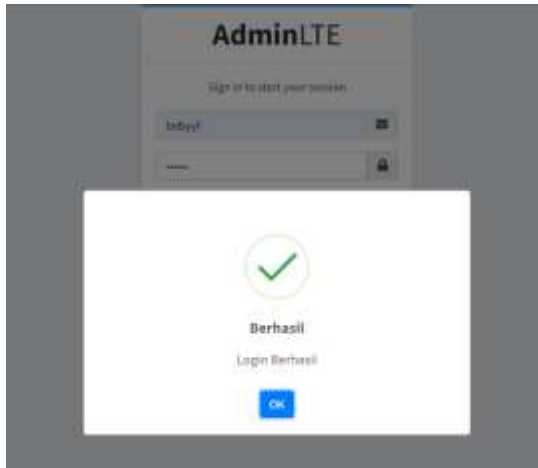
XXXXXX

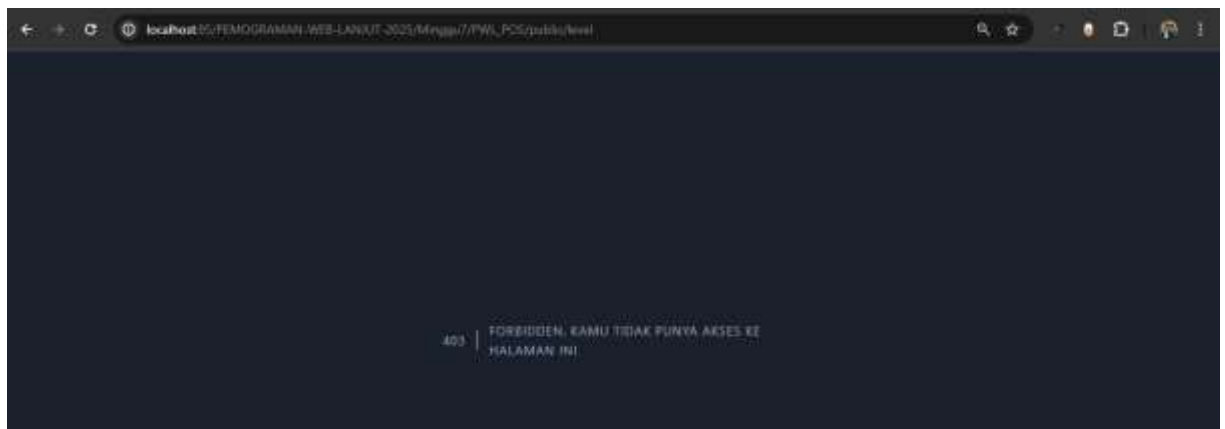
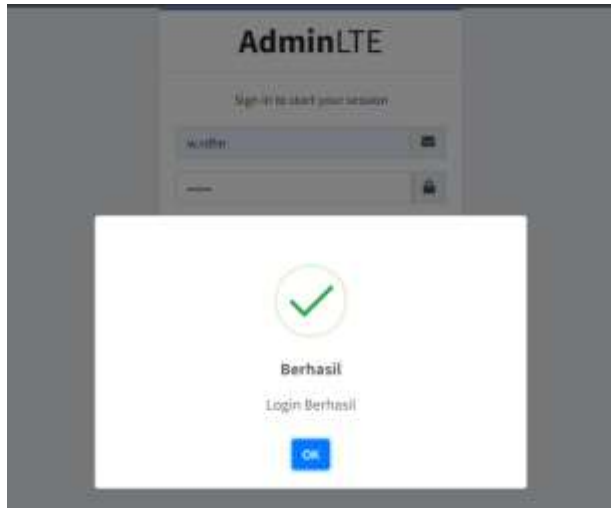
☐ Remember Me

Sign In



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>





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

Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?
 - Pada praktikum 2 ini, kita belajar cara membatasi akses halaman berdasarkan level user (authorization). Kita membuat middleware baru bernama `AuthorizeUser`, lalu mengaturnya agar hanya user dengan level tertentu yang bisa mengakses route tertentu. Middleware tersebut dicek melalui data user yang sedang login. Setelah itu, kita daftarkan middleware di `Kernel.php` dan uji akses menggunakan akun dengan level yang berbeda.
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()`



```
public function getRoleName(): string
{
    return $this->level->level_nama;
}

/**
 * Cek apakah user memiliki role tertentu
 */
0 references | 0 overrides
public function hasRole($role): bool
{
    return $this->level->level_kode == $role;
}

/**
 * Mendapatkan kode role
 */
0 references | 0 overrides
public function getRole(): mixed
{
    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      /**
12       * Handle an incoming request.
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  }
```



```
Minggu7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > PHP > AuthorizeUser > handle()
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  1 reference | 0 implementations
10 class AuthorizeUser
11 {
12     /**
13      * Handle an incoming request.
14      *
15      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
16      */
17     0 references | 0 overrides
18     public function handle(Request $request, Closure $next, ...$roles): Response
19     {
20         $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
21         if (in_array($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(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
26     }
27 }
```

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
});
```



```
Route::middleware(['authorize:ADM,MNG'])->group(function(): void {
    Route::group(['prefix' => 'barang'], function(): void {
        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::delete('/{id}', [BarangController::class, 'destroy']);
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
        Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax']);
    });
});
```

```
Route::middleware(['authorize:ADM,MNG'])->group(function(): void{
    Route::group(['prefix' => 'barang'], function(): void {
        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::delete('/{id}', [BarangController::class, 'destroy']);
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
    });
});
```

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

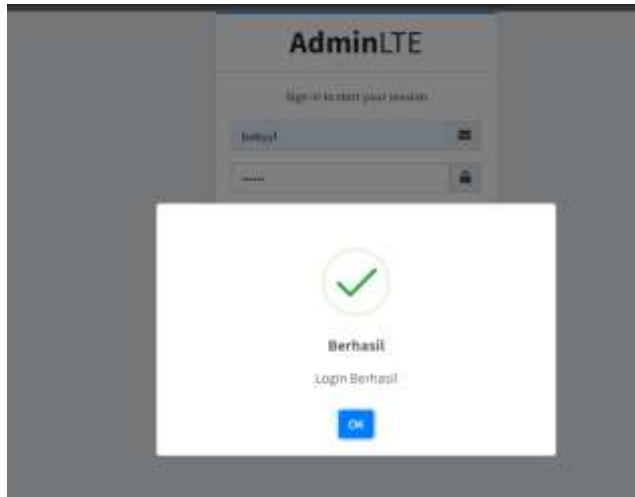


The image shows the AdminLTE login interface. It features a white login box on a light gray background. The box has the title "AdminLTE" and the instruction "Sign in to start your session". There are two input fields: one for the username "w.rdh" and another for the password, which is masked with dots. Below the password field is a "Remember Me" checkbox. A blue "Sign In" button is located at the bottom right of the login box.

The image shows the AdminLTE login success screen. It features a white box on a dark gray background. The box has a green checkmark icon and the text "Berhasil" (Successful) and "Login Berhasil" (Login Successful). A blue "OK" button is located at the bottom of the box.

The image shows the AdminLTE inventory management dashboard. It features a dark sidebar on the left with a search bar and a list of menu items. The main content area has a title "Daftar Barang" (Inventory List) and a subtitle "Daftar barang yang terdaftar dalam sistem" (List of goods registered in the system). There is a filter dropdown set to "Semua" (All) and a "Show" dropdown set to "10 per page". A table lists the inventory items with columns for ID, Kode Barang, Nama Barang, Harga Beli, Harga Jual, Kategori Barang, and Aksi. The table contains 10 rows of data. At the bottom right, there are buttons for "Tambah" (Add) and "Tambah Baru" (Add New).

ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
1	BRG001	Laptop ASUS ROG	1500000	1800000	Elektronik	Detail Edit Hapus
2	BRG002	Smartphone Samsung S23	1200000	1400000	Elektronik	Detail Edit Hapus
3	BRG003	Headset Gaming Logitech	150000	200000	Peripherals & Accessories	Detail Edit Hapus
4	BRG004	Kursi Gaming	1200000	1500000	Furniture & Home Decor	Detail Edit Hapus
5	BRG005	Kopi Bubuk Arabika	50000	75000	Makanan & Minuman	Detail Edit Hapus
6	BRG006	Mesin Cuci Piring	250000	300000	Makanan & Minuman	Detail Edit Hapus
7	BRG007	Botol Air Plastik 1.5L	10000	20000	Kebutuhan Rumah Tangga	Detail Edit Hapus
8	BRG008	Sepatu Sneakers	200000	250000	Kebutuhan Rumah Tangga	Detail Edit Hapus
9	BRG009	Spesifikasi Komputer	3000000	4000000	Spesifikasi Komputer	Detail Edit Hapus
10	BRG010	Mouse Gaming Logitech	100000	150000	Spesifikasi Komputer	Detail Edit Hapus



ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
1	BRG001	Laptop ASUS ROG	1000000	1200000	Elektronik	Detail Edit Hapus
2	BRG002	Samsung Galaxy S21	1200000	1400000	Elektronik	Detail Edit Hapus
3	BRG003	Jaket Hoodie Pria	100000	200000	Pakaian & Mode	Detail Edit Hapus
4	BRG004	Ayam Sate	150000	180000	Makanan & Minuman	Detail Edit Hapus
5	BRG005	Kopi Bubuk Arabika	90000	120000	Makanan & Minuman	Detail Edit Hapus
6	BRG006	Mie Instan Goreng	2000	3000	Makanan & Minuman	Detail Edit Hapus
7	BRG007	Susu Cair Puring SL	10000	20000	Kebutuhan Rumah Tangga	Detail Edit Hapus
8	BRG008	Sapu Lantai Kayu	20000	30000	Kebutuhan Rumah Tangga	Detail Edit Hapus
9	BRG009	Sepeda Gunung Polygon	1000000	1200000	Sport & Outdoor	Detail Edit Hapus
10	BRG010	Matras Yoga Anti Slip	100000	150000	Sport & Outdoor	Detail Edit Hapus

Tugas 3 – Implementasi Multi-Level Authorization :

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu yang sesuai dengan Level/Jenis User



Tabel m_user

```
Route::middleware(['authorize:ADM,MNG,STF'])->group(function (): void {  
    Route::group(['prefix' => 'user'], function (): void {  
        Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user  
        Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables  
        Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user  
        Route::post('/', [UserController::class, 'store']); // menyimpan data user baru  
        Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax  
        Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax  
        Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user  
        Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user  
        Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user  
        Route::get('/{id}/show_ajax', [UserController::class, 'show_ajax']); // Menampilkan detail user menggunakan AJAX  
        Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax  
        Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax  
        Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax  
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // Untuk hapus data user Ajax  
        Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user  
    });  
});
```

Tabel m_level

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

Tabel m_kategori



```
Route::middleware(['authorize:ADM,MNG,STF'])->group(function (): void {
    Route::group(['prefix' => 'kategori'], function (): void {
        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::get('/{id}/show_ajax', [KategoriController::class, 'show_ajax']);
        Route::delete('/{id}', [KategoriController::class, 'destroy']);
    });
});
```

Tabel m_supplier

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




Tabel m_barang

```
Route::middleware(['authorize:ADM,MNG,STF'])->group(function (): void {
    Route::group(['prefix' => 'barang'], function (): void {
        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::delete('/{id}', [BarangController::class, 'destroy']);
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
        Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax']);
    });
});
```

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

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.
2. Screenshot hasil yang kalian kerjakan
3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian



AuthController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\LevelModel;
use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

6 references | 0 implementations
class AuthController extends Controller
{
    1 reference | 0 overrides
    public function register(): mixed|View
    {
        $level = LevelModel::select('level_id', 'level_nama')->get();

        return view('auth.register')->with('level', $level);
    }

    1 reference | 0 overrides
    public function postRegister(Request $request): JsonResponse|mixed
    {
        $validator = Validator::make($request->all(), [
            'username' => 'required|string|unique:m_user,username',
            'nama' => 'required|string|max:255',
            'password' => 'required|string|min:5', // <--- tanpa konfirmasi
            'level_id' => 'required|exists:m_level,level_id',
        ]);

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



RegisterController.php

```
Minggu / > PWL_FOS > resources > views > auth > register.blade.php > ...
1 <!(DOCTYPE html)
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>Register Pengguna</title>
8     <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
9     <link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
10    <link rel="stylesheet" href="{{ asset('plugins/icheck-bootstrap/ichack-bootstrap.min.css') }}">
11    <link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
12    <link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
13 </head>
14
15 <body class="hold-transition login-page">
16     <div class="login-box">
17         <div class="card card-outline card-primary">
18             <div class="card-header text-center"><a href="{{ url('login') }}" class="text-decoration: none;">Admin</a></div>
19             <div class="card-body">
20                 <p class="login-box-msg">Register a new account</p>
21                 <form action="{{ url('register') }}" method="POST" id="form-register">
22                     @csrf
23                     <div class="input-group mb-3">
24                         <select id="level_id" name="level_id" class="form-control" required>
25                             <option value="">Select Level</option>
26                             @foreach ($level as $l)
27                                 <option value="{{ $l->level_id }}">{{ $l->level_nama }}</option>
28                             @endforeach
29                         </select>
30                         <div class="input-group-append">
31                             <div class="input-group-text">
32                                 <span class="fas fa-layer-group"></span>
33                             </div>
34                         </div>
35                         <small id="error-level_id" class="error-text text-danger"></small>
36                     </div>
37                     <div class="input-group mb-3">
38                         <input type="text" id="username" name="username" class="form-control" placeholder="Username" required>
39                         <div class="input-group-append">
```

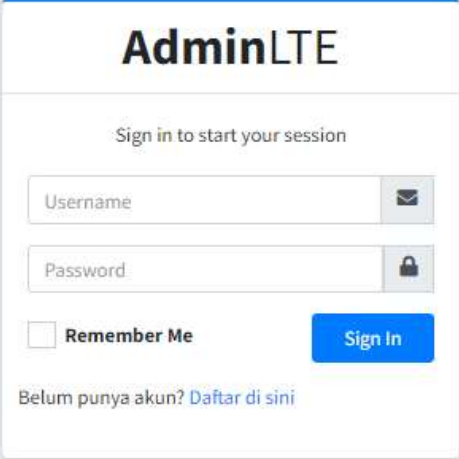
Route

```
//JOBSHEET 7
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::post('logout', [AuthController::class, 'logout'])->middleware('auth');

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

Login.blade.php

```
<!-- Tambahan: link ke halaman registrasi -->
<div class="text-center mt-3">
    <p>Belum punya akun? <a href="{{ url('register') }}">Daftar di sini</a></p>
</div>
</div>
```



AdminLTE

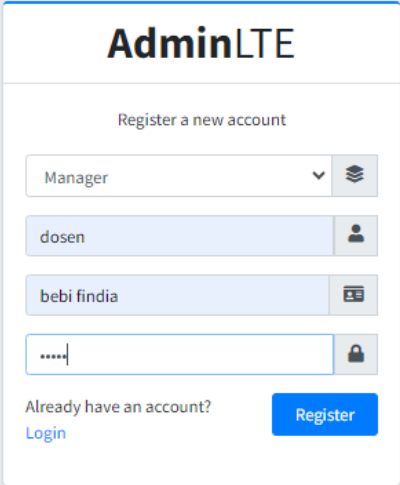
Sign in to start your session

Username

Password

☐ Remember Me

Belum punya akun? [Daftar di sini](#)



AdminLTE

Register a new account

Manager

dosen

bebi findia

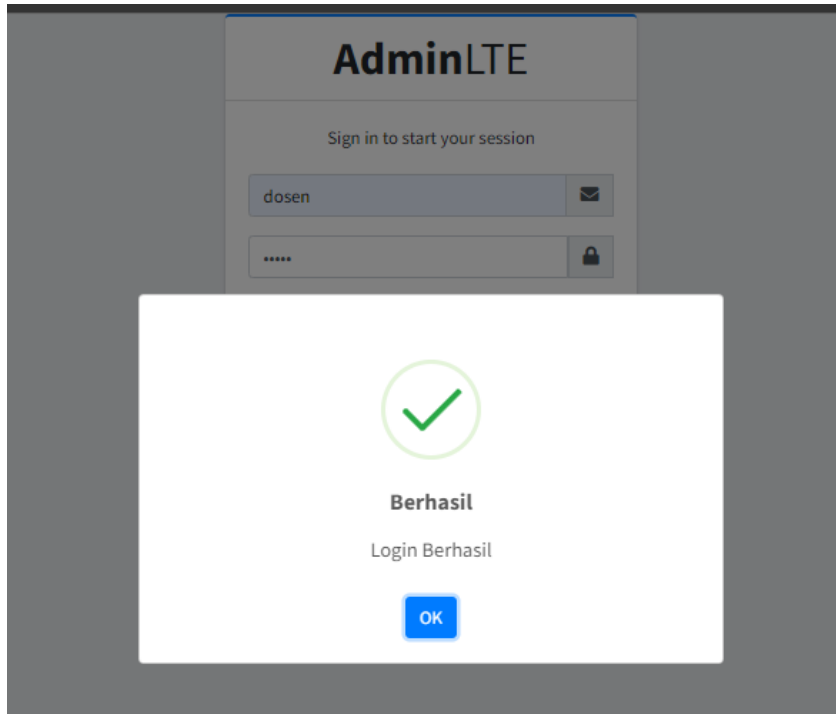
.....

Already have an account?

[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>



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