



Nama	:	Nimas Septiandini
No/NIM	:	18/2341760087
Kelas	:	SIB-2B
Mata Kuliah	:	Pemrograman Web Lanjut (PWL)
Program Studi	:	D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester	:	4 (empat) / 5 (lima)
Pertemuan ke-	:	7 (tujuh)

## 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



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>

---

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         ],  
67     ],
```

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     ],
```

```
config > auth.php  
3     return [  
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 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');
    }
}
```

```
app > Models > UserModel.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable
9
10 class UserModel extends Model
11 {
12     use HasFactory;
13
14     protected $table = 'm_user'; //mendefinisikan nama tabel yang digunakan oleh model
15     protected $primaryKey = 'user_id'; //Mendefinisikan primary key dari tabel yang digunakan
16     protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];
17
18     protected $hidden = ['password']; //jangan ditampilkan saat select
19
20     protected $casts = ['password' => 'hashed']; //casting password agar otomatis di hash
21
22     public function level(): BelongsTo
23     {
24         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
25     }
26 }
```



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');
    }
}
```



```
app > Http > Controllers > AuthController.php
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Auth;
5
6  class AuthController extends Controller
7  {
8      public function login()
9      {
10         if(Auth::check()){ // jika sudah login, maka redirect ke halaman home
11             return redirect('/');
12         }
13         return view('auth.login');
14     }
15
16     public function postlogin(Request $request)
17     {
18         if($request->ajax() || $request->wantsJson()){
19             $credentials = $request->only('username', 'password');
20
21             if (Auth::attempt($credentials)) {
22                 return response()->json([
23                     'status' => true,
24                     'message' => 'Login Berhasil',
25                     'redirect' => url('/')
26                 ]);
27             }
28
29             return response()->json([
30                 'status' => false,
31                 'message' => 'Login Gagal'
32             ]);
33         }
34         return redirect('login');
35     }
36
37     public function logout(Request $request)
38     {
39         Auth::logout();
40
41         $request->session()->invalidate();
42         $request->session()->regenerateToken();
43         return redirect('login');
44     }
45 }
46 }
```

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/bootstrap-
4.min.css') }}">
<!-- Theme style -->
<link rel="stylesheet" href="{ asset('dist/css/adminlte.min.css') }}">
</head>
<body class="hold-transition login-page">
<div class="login-box">
  <!-- /.login-logo -->
  <div class="card card-outline card-primary">
    <div class="card-header text-center"><a href="{ url('/') }"
class="h1"><b>Admin</b>LTE</a></div>
    <div class="card-body">
      <p class="login-box-msg">Sign in to start your session</p>
      <form action="{ url('login') }" method="POST" id="form-login">
        @csrf
        <div class="input-group mb-3">
          <input type="text" id="username" name="username" class="form-control"
placeholder="Username">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-envelope"></span>
            </div>
          </div>
          <small id="error-username" class="error-text text-danger"></small>
        </div>
        <div class="input-group mb-3">
          <input type="password" id="password" name="password" class="form-control"
placeholder="Password">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-lock"></span>
            </div>
          </div>
          <small id="error-password" class="error-text text-danger"></small>
        </div>
        <div class="row">
          <div class="col-8">
            <div class="icheck-primary">
              <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
            </div>
          <!-- /.col -->
          <div class="col-4">
            <button type="submit" class="btn btn-primary btn-block">Sign In</button>
          </div>
          <!-- /.col -->
        </div>
      </form>
    </div>
    <!-- /.card-body -->
  </div>
  <!-- /.card -->
</div>
<!-- /.login-box -->

<!-- jQuery -->
```



```
<script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<!-- jquery-validation -->
<script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<script src="{{ asset('plugins/jquery-validation/additional-methods.min.js') }}"></script>
<!-- SweetAlert2 -->
<script src="{{ asset('plugins/sweetalert2/sweetalert2.min.js') }}"></script>
<!-- AdminLTE App -->
<script src="{{ asset('dist/js/adminlte.min.js') }}"></script>

<script>
$.ajaxSetup({
  headers: {
    'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
  }
});

$(document).ready(function() {
  $("#form-login").validate({
    rules: {
      username: {required: true, minlength: 4, maxlength: 20},
      password: {required: true, minlength: 6, maxlength: 20}
    },
    submitHandler: function(form) { // ketika valid, maka bagian yg akan dijalankan
      $.ajax({
        url: form.action,
        type: form.method,
        data: $(form).serialize(),
        success: function(response) {
          if(response.status){ // jika sukses
            Swal.fire({
              icon: 'success',
              title: 'Berhasil',
              text: response.message,
            }).then(function() {
              window.location = response.redirect;
            });
          }else{ // jika error
            $('.error-text').text('');
            $.each(response.msgField, function(prefix, val) {
              $('#error-'+prefix).text(val[0]);
            });
            Swal.fire({
              icon: 'error',
              title: 'Terjadi Kesalahan',
              text: response.message
            });
          }
        }
      });
    },
    errorElement: 'span',
    errorPlacement: function (error, element) {
      error.addClass('invalid-feedback');
      element.closest('.input-group').append(error);
    },
    highlight: function (element, errorClass, validClass) {
      $(element).addClass('is-invalid');
    },
    unhighlight: function (element, errorClass, validClass) {
      $(element).removeClass('is-invalid');
    }
  });
});
</script>
```



```
</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  
});
```

```
routes > web.php  
3 use App\Http\Controllers\LevelController;  
4 use Illuminate\Support\Facades\Route;  
5 use App\Http\Controllers\KategoriController;  
6 use App\Http\Controllers\UserController;  
7 use App\Http\Controllers>WelcomeController;  
8 use App\Http\Controllers\BarangController;  
9 use App\Http\Controllers\SupplierController;  
10 use App\Http\Controllers\AuthController;  
11  
12 //M7  
13 Route::pattern('id','[0-9]+'); // artinya ketika ada parameter {id}, maka harus berupa angka  
14  
15 Route::get('login', [AuthController::class,'login'])->name('login');  
16 Route::post('login', [AuthController::class,'postlogin']);  
17 Route::get('logout', [AuthController::class,'logout'])->middleware('auth');  
18  
19 Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu  
20  
21     // masukkan semua route yang perlu autentikasi di sini  
22  
23 });
```

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>

---

### AdminLTE

Sign in to start your session





☐ Remember Me

### AdminLTE

Sign in to start your session



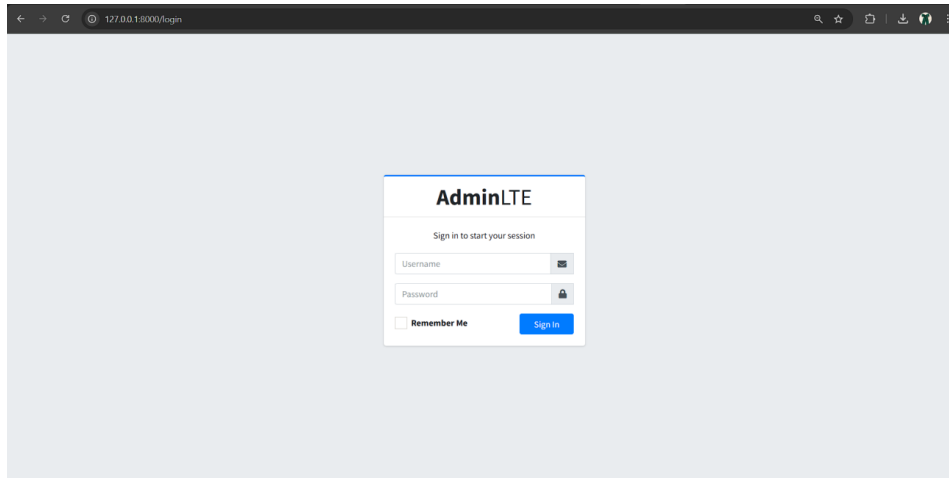


☐ Remember Me



## Tugas 1 – Implementasi Authentication :

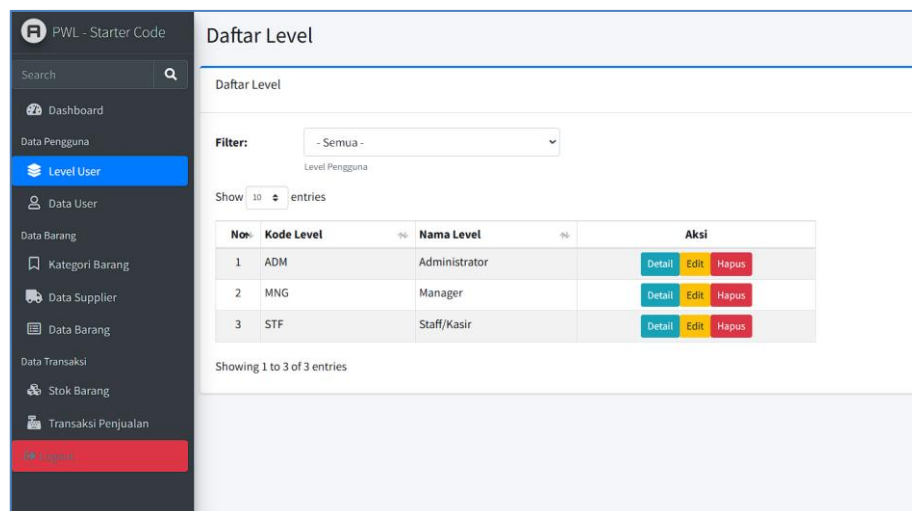
1. Silahkan implementasikan proses login pada project kalian masing-masing



2. Silahkan implementasi proses logout pada halaman web yang kalian buat  
= implementasi proses logout dengan menambahkan button logout pada file sidebar.blade.php

```
resources > views > layouts > sidebar.blade.php > div.sidebar > nav.mt-2 > ul.nav.nav-pills.nav-sidebar.flex-column > li.nav-item > a.nav-link
```

```
1 <div class="sidebar">
13 <nav class="mt-2">
14 <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-accordion="fa
73 <li class="nav-item">
74 <form id="logout-form-sidebar" action="{{ url('logout') }}" method="GET">
75 @csrf
76 <button type="submit" class="nav-link btn btn-danger text-left w-100">
77 <i class="fas fa-sign-out-alt"></i> Logout
78 </button>
79 </form>
80 </li>
81 </ul>
82 </nav>
83 </div>
```



3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan Tahapan:



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>

---

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



## B. Implementasi *Authorization* di Laravel

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

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

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

### Praktikum 2 – Implementasi *Authorization* di Laravel dengan Middleware

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

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

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

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

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



```
app > Models > UserModel.php
11 {
25 }
26
27 /**
28  * Mendapatkan nama role
29  */
30 public function getRoleName(): string
31 {
32     return $this->level->level_nama;
33 }
34
35 /**
36  * Cek apakah user memiliki role tertentu
37  */
38 public function hasRole($role): bool
39 {
40     return $this->level->level_kode == $role;
41 }
42
43 }
```

- 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 }
```





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,
```

app > Http > Kernel.php

```
8  
  
55 protected $middlewareAliases = [  
56     'auth' => \App\Http\Middleware\Authenticate::class,  
57     'authorize' => \App\Http\Middleware\AuthorizeUser::class,  
58     'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
59     'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
```

5. Sekarang kita perhatikan tabel `m_level` yang menjadi tabel untuk menyimpan level/group/role dari user ada

level_id	level_kode	level_nama	created_at	updated_at	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
```

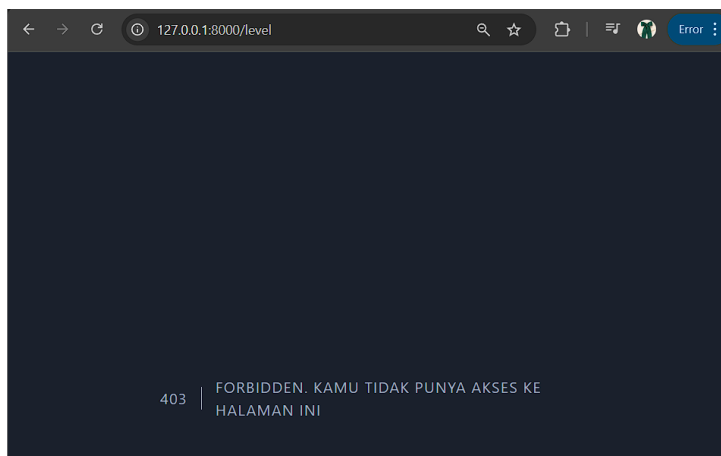
```
routes > web.php
158
159 Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu
160
161     // masukkan semua route yang perlu autentikasi di sini
162     Route::get('/', [WelcomeController::class, 'index']);
163
164     // route Level
165
166     // artinya semua route di dalam group ini harus punya role ADM ( Administrator )
167     Route::middleware(['authorize:ADM'])->group(function () {
168
169         Route::get('/level', [LevelController::class, 'index']);
170
171         Route::post('/level/list', [LevelController::class, 'list']); // untuk list json datatables
172
173         Route::get('/level/create', [LevelController::class, 'create']);
174
175         Route::post('/level', [LevelController::class, 'store']);
176
177         Route::get('/level/{id}/edit', [LevelController::class, 'edit']); // untuk tampilkan form edit
178         Route::put('/level/{id}', [LevelController::class, 'update']); // untuk proses update data
179
180         Route::delete('/level/{id}', [LevelController::class, 'destroy']); // untuk proses hapus data
181
182     });
183
184     // route Kategori
185
186 });
```

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

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



The image shows a login form titled "AdminLTE". Below the title, it says "Sign in to start your session". There are two input fields: the first is for the username, containing "managerbaru", and the second is for the password, containing "\*\*\*\*\*". To the right of each field is an icon (an envelope for email and a lock for password). Below the password field is a checkbox labeled "Remember Me". To the right of the checkbox is a blue button labeled "Sign In".



## Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?  
= Pada praktikum ini dibuat authorize untuk membatasi akses ke halaman tertentu. Pada praktikum ini, akses manager dibatasi, sehingga yang bisa melihat data level user hanya admin.
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
  - Menambahkan roleName dan hasRole pada UserModel
  - Kemudian membuat middleware AuthorizeUser untuk memeriksa peran user
  - Setelah itu mendaftarkan middleware untuk authorize pada kernel
3. Submit kode untuk impementasi Authorization pada repository github kalian.



## C. Multi-Level Authorization di Laravel

Bagaimana seandainya jika terdapat level/group/role satu dengan yang lain memiliki hak akses yang sama. Contoh sederhana, user level Admin dan Manager bisa sama-sama mengakses menu Barang pada aplikasi yang kita buat. Maka tidak mungkin kalau kita buat route untuk masing-masing level user. Hal ini akan memakan banyak waktu, dan proses yang lama.

```
// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});

// artinya semua route di dalam group ini harus punya role MNG (Manager)
Route::middleware(['authorize:MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

Hal ini jadi kendala ketika kita mau mengganti hak akses, maka kita akan mengganti sebagian besar route yang sudah kita tulis. Untuk itu, kita perlu mengelola middleware agar bisa mendukung penambahan hak akses secara dinamis.

### Praktikum 3 – Implementasi Multi-Level Authorizaton di Laravel dengan Middleware

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

1. Kita modifikasi `UserModel.php` untuk mendapatkan `level_kode` dari user yang sudah login. Jadi kita buat fungsi dengan nama `getRole()`



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

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

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

```
app > Models > UserModel.php
11 {
39 {
44     return $this->level->level_kode == $role;
45 }
46
47 // Mendapatkan kode role
48 public function getRole()
49 {
50     return $this->level->level_kode;
51 }
52
53 }
```

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 }
```



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>

```
app > Http > Middleware > AuthorizeUser.php
10
11 /**
12
13
14
15
16 public function handle(Request $request, Closure $next, ...$roles): Response
17 {
18     $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
19     if(in_array($user_role, $roles)){ // cek apakah level_kode user ada di dalam array roles
20         return $next($request); // jika ada, maka lanjutkan request
21     }
22
23     // jika tidak punya role, maka tampilkan error 403
24     abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
25 }
26
```

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





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

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

The image shows a login form titled "AdminLTE". Below the title is the text "Sign in to start your session". There are two input fields: one for the username "managerbaru" and one for the password, which is masked with dots. To the right of the password field is a lock icon. Below the password field is a checkbox labeled "Remember Me". To the right of the checkbox is a blue button labeled "Sign In".

Manager kini dapat melihat Level User

The image shows a screenshot of the "Daftar Level" page in AdminLTE. The page has a sidebar on the left with a search bar and a list of menu items: Dashboard, Data Pengguna, Level User (highlighted), Data User, Data Barang, Kategori Barang, Data Supplier, Data Barang, Data Transaksi, Stok Barang, Transaksi Penjualan, and Logout. The main content area is titled "Daftar Level" and contains a filter dropdown set to "- Semua -". Below the filter is a table with 3 entries. The table has columns: No., Kode Level, Nama Level, and Aksi. The rows are: 1. ADM, Administrator, with actions Detail, Edit, and Hapus; 2. MNG, Manager, with actions Detail, Edit, and Hapus; 3. STF, Staff/Kasir, with actions Detail, Edit, and Hapus. Below the table is the text "Showing 1 to 3 of 3 entries".

No.	Kode Level	Nama Level	Aksi
1	ADM	Administrator	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
2	MNG	Manager	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
3	STF	Staff/Kasir	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>



### Tugas 3 – Implementasi Multi-Level Authorization :

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

```
routes > web.php
31 Route::middleware(['auth'])->group(function() {
52     // Level routes - hanya untuk ADMIN (ditambah multilevel ke manager dan staff)
53     Route::middleware(['auth', 'authorize:ADM,MNG,STF'])->group(function () {
```

The image shows a login form for AdminLTE. It has a title "AdminLTE" and a subtitle "Sign in to start your session". There are two input fields: one for the username "staffbaru" and one for the password, which is masked with dots. Below the password field is a "Remember Me" checkbox. To the right of the password field is a "Sign In" button.

The image shows a screenshot of the AdminLTE dashboard. The left sidebar contains a menu with items like "Dashboard", "Data Pengguna", "Level User", "Data Barang", "Kategori Barang", "Data Supplier", "Data Barang", "Data Transaksi", "Stok Barang", and "Transaksi Penjualan". The main content area is titled "Daftar Level" and shows a table of user levels.

No	Kode Level	Nama Level	Aksi
1	ADM	Administrator	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
2	MNG	Manager	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
3	STF	Staff/Kasir	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 3 of 3 entries

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

### Tugas 4 – Implementasi Form Registrasi :





1. Silahkan implementasikan form untuk registrasi user.

- Memodifikasi AuthController.php

```
app > Http > Controllers > AuthController.php
9   {
41   {
47   }
48   }
49   public function register(){
50       $level = LevelModel::select('level_id','level_name')->get();
51
52       return View('auth.register')
53       ->with('level', $level);
54   }
55
56   public function store_user(Request $request)
57   {
58       $request->validate([
59           'username' => 'required|string|min:3|unique:m_user,username',
60           'nama'     => 'required|string|max:100',
61           'password' => 'required|min:5',
62           'level_id' => 'required|integer',
63       ]);
64
65       // Menyimpan user baru
66       UserModel::create([
67           'username' => $request->username,
68           'nama'     => $request->nama,
69           'password' => bcrypt($request->password), // Enkripsi password
70           'level_id' => $request->level_id,
71       ]);
72
73       // Mengirim respons JSON jika permintaan AJAX
74       if ($request->ajax() || $request->wantsJson()) {
75           return response()->json([
76               'status' => true,
77               'message' => 'Registrasi Berhasil',
78               'redirect' => url('login')
79           ]);
80       }
81
82       // Jika bukan AJAX, redirect ke halaman utama dengan flash message
83       return redirect('/')->with('success', 'Registrasi berhasil');
84   }
85 }
```

- Menambahkan view form register



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta name="csrf-token" content="{{ csrf_token }}">
7   <title>Registrasi Pengguna</title>
8
9   <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans-Pro:100,400,400i,700&display=fallback">
10  <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
11  <link rel="stylesheet" href="{{ asset('adminlte/plugins/checkbox-bootstrap/checkbox-bootstrap.min.css') }}">
12  <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
13  <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
14 </head>
15
16 <body class="hold-transition login-page">
17   <div class="login-box">
18     <div class="card card-outline card-primary">
19       <div class="card-header text-center">
20         <a href="{{ url('/') }}" class="h1"><b>Admin</b></a>
21       </div>
22       <div class="card-body">
23         <p class="login-box-msg">Daftar Pengguna Baru</p>
24         <form action="{{ url('register') }}" method="POST" id="form-register">
25           @csrf
26           <div class="form-group">
27             <label>Username</label>
28             <input type="text" name="username" id="username" class="form-control" required minlength="4"
29               maxlength="20" placeholder="Masukkan Username">
30             <small id="error-username" class="error-text form-text text-danger"></small>
31           </div>
32           <div class="form-group">
33             <label>Nama</label>
34             <input type="text" name="nama" id="nama" class="form-control" required maxlength="100" placeholder="Masukkan Nama">
35             <small id="error-nama" class="error-text form-text text-danger"></small>
36           </div>
37           <div class="form-group">
38             <label>Password</label>
39             <input type="password" name="password" id="password" class="form-control" required minlength="5"
40               maxlength="20" placeholder="Masukkan Password">
41             <small id="error-password" class="error-text form-text text-danger"></small>
42           </div>
43           <div class="form-group">
44             <label>Pilih Role</label>
45             <select name="level_id" id="level_id" class="form-control" required>
46               <option value="">Pilih level </option>
47               @foreach($level as $l)
48                 <option value="{{ $l->level_id }}">{{ $l->level_name }}</option>
49             @endforeach
49             </select>
50             <small id="error-level_id" class="error-text form-text text-danger"></small>
51           </div>
52           <div class="row justify-content-center">
53             <div class="col-auto">
54               <button type="submit" class="btn btn-primary btn-block">Sign Up</button>
55             </div>
56           </div>
57           <div class="text-center mt-3">
58             <p>Sudah punya akun? <a href="{{ url('login') }}" class="text-primary" style="text-decoration: underline;">Silakan login</a></p>
59           </div>
60         </form>
61       </div>
62     </div>
63   </div>
64
65   <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
66   <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
67   <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
68   <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.additional-methods.min.js') }}"></script>
69   <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
70   <script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
71
72   <script>
73     (document).ready(function () {
74       $("#form-register").validate({
75         rules: {
76           username: { required: true, minlength: 4, maxlength: 20 },
77           nama: { required: true, minlength: 50 },
78           password: { required: true, minlength: 5, maxlength: 20 },
79           password_confirmation: { equalTo: "password" },
80           level_id: { required: true, number: true }
81         },
82         submitHandler: function (form) {
83           $.ajax({
84             headers: {
85               'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
86             },
87             url: form.action,
88             type: form.method,
89             data: $(form).serialize(),
90             success: function (response) {
91               console.log(response); // log respons untuk melihat data yang diterima
92               if (response.status) {
93                 Swal.fire({
94                   icon: 'success',
95                   title: 'Registrasi Berhasil',
96                   text: response.message,
97                 }).then(() => {
98                   if (response.redirect) {
99                     window.location = response.redirect;
100                   }
101                 });
102               } else {
103                 $('#text-danger').text('');
104                 $.each(response.errors, function (key, val) {
105                   $('#error-' + key).text(val[0]);
106                 });
107                 Swal.fire({
108                   icon: 'error',
109                   title: 'Terjadi Kesalahan',
110                   text: response.message
111                 });
112               }
113             }
114           });
115           return false;
116         },
117         errorElement: 'span',
118         errorPlacement: function (error, element) {
119           error.addClass('invalid-feedback');
120           element.closest('.input-group').append(error);
121         },
122         highlight: function (element) {
123           $(element).addClass('is-invalid');
124         },
125         unhighlight: function (element) {
126           $(element).removeClass('is-invalid');
127         }
128       });
129     });
130   </script>
131 </body>
132 </html>
```



- Menambahkan route pada web.php

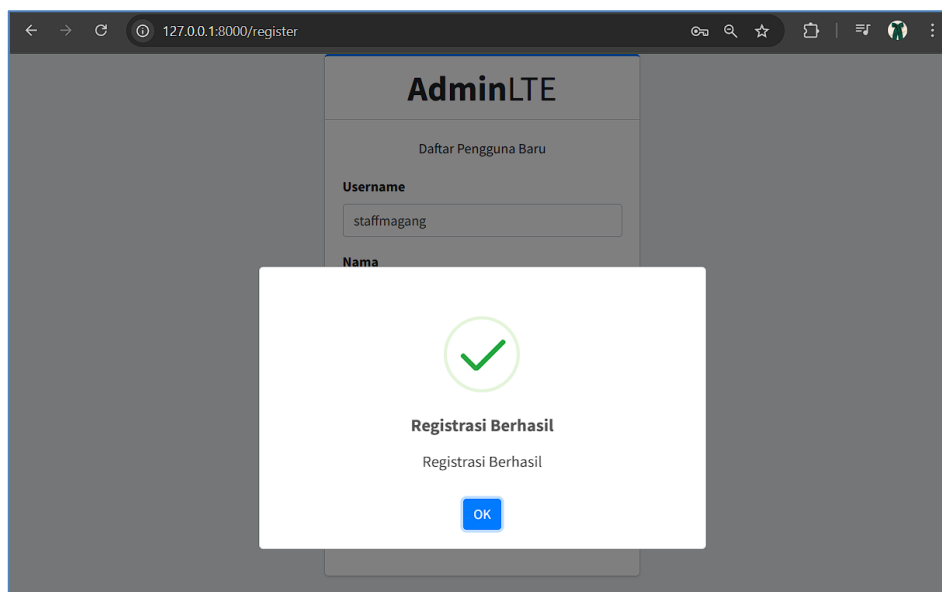
```
routes > web.php
23 // Jobsheet 7
24 Route::pattern('id', '[0-9]+'); //artinya ketika parameter {id}, maka harus berupa angka
25
26 Route::get('login', [AuthController::class, 'login'])->name('login');
27 Route::post('login', [AuthController::class, 'postLogin']);
28 Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
29
30 //route W7Prak4 form register
31 Route::get('/register', [AuthController::class, 'register'])->name('register');
32 Route::post('/register', [AuthController::class, 'store_user'])->name('store_user');
33
```

## 2. Screenshot hasil yang kalian kerjakan

- Database sebelum dilakukan registrasi

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	admin_admin	\$2y\$12\$w.9prZC12ONyA7OtLS/EJOy1Dbrbe1uD7IKCSLX6SOy...	NULL	2025-03-06 15:18:14
2	2	manager	Manager	\$2y\$12\$CRx4iaJuKRz.HTbRHDtNHepflx/Pb9cemJOYEzLeKY9...	NULL	NULL
3	3	staff	Staff/Kasir	\$2y\$12\$LUd0dWnkapTR5IPJOJrQuEASqCSseje2OxBXURdQm0...	NULL	NULL
5	2	manager_dua	Manager 2	\$2y\$12\$HpCLtKO85tMG8tbtWK4l8eI5R9TpjYqyAxuiCmYe4or...	2025-03-06 04:49:11	2025-03-06 04:49:11
6	2	manager22	Manager Dua Dua	\$2y\$12\$9nAL7bHt4JqTabMTuQEPgOjicrPGOmZtqiYAqdTaMBP...	2025-03-06 12:58:23	2025-03-06 12:58:23
7	2	manager33	Manager Tiga Tiga	\$2y\$12\$8vdg4GsDgnQ1lowpai34vOafe2pumR4C/AjKxDRaK6...	2025-03-06 13:44:08	2025-03-06 13:44:08
10	2	manager56	Manager55	\$2y\$12\$I5utX0J4x1GBuSmUIAYXxuwmvnhdjjHtI./4MwDOQH...	2025-03-06 13:59:31	2025-03-06 13:59:31
11	2	manager55	Manager55	\$2y\$12\$kbJe85gvQF/ImuShjk.yuqxLfkWVp7vyUPLIMjnbcn...	2025-03-06 14:01:01	2025-03-06 14:01:01
12	2	manager12	Manager11	\$2y\$12\$hyPI6sEjalXoD641aKn8OkasHw2uC8XoFE3WD91kRh...	2025-03-06 14:10:57	2025-03-06 14:10:57
19	1	adminbaru	AdminBaru	\$2y\$12\$BauMSxtiXptRFkLsl5glu.PkR9Vn6XXdTauOYqBE1d/...	2025-04-08 19:08:39	2025-04-08 19:08:39
20	2	managerbaru	ManagerBaru	\$2y\$12\$gP5bK5gVqySLZSmcr7b9Hu7JhT5U2FZznZPIHn/onTu...	2025-04-08 19:47:57	2025-04-08 19:47:57
22	1	staffbaru	StaffBaru	\$2y\$12\$7VD8k.dJGU1pSMpCyAhtUe7T0jg7KcWn6KcxSy0bFss...	2025-04-09 23:44:25	2025-04-09 23:44:25

- Proses registrasi berhasil

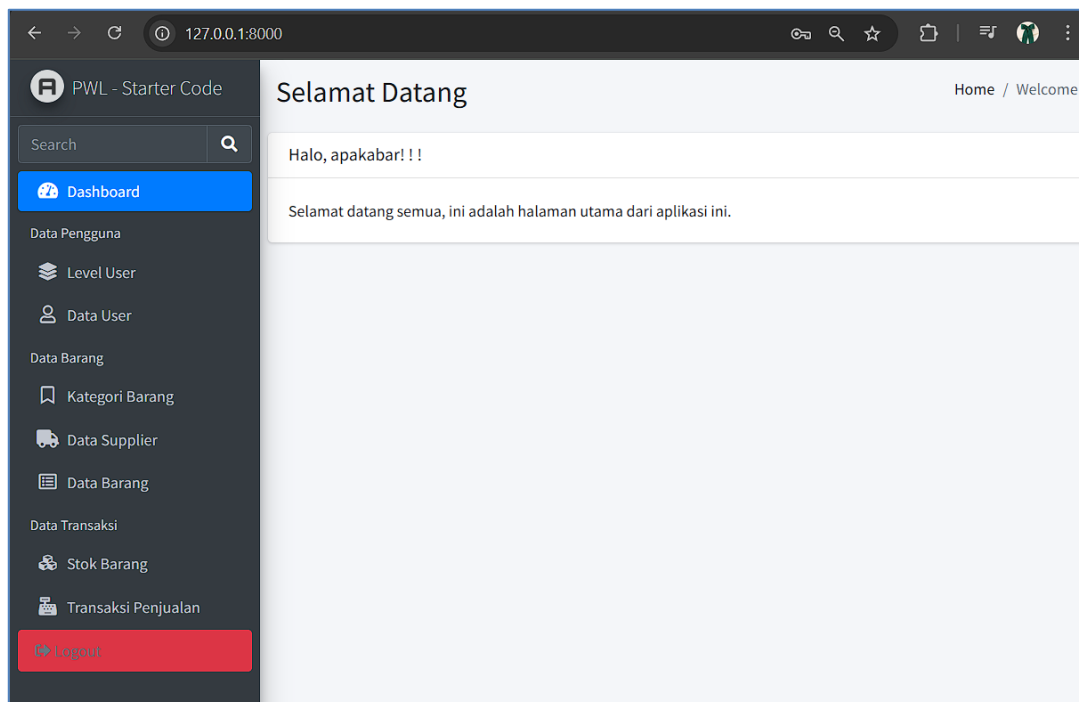




- Registrasi user baru sudah masuk ke database

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	admin_admin	\$2y\$12\$w.9prZC12ONyA70tLS/EJOy1Dbrbe1uD7IKCSLX6SOy...	NULL	2025-03-06 15:18:14
2	2	manager	Manager	\$2y\$12\$CRx4iaJuKRz HTbRHDtNHepfIx/Pb9cemJOYEzLeKY9...	NULL	NULL
3	3	staff	Staff/Kasir	\$2y\$12\$LUd0dWnkapTR5fPOJ0rQruEASqCSseje2OxBXURdQm0...	NULL	NULL
5	2	manager_dua	Manager 2	\$2y\$12\$HpCLtKO85tMG8tbtWK4l8eL5R9TjYqyAxuiCmYe4or...	2025-03-06 04:49:11	2025-03-06 04:49:11
6	2	manager22	Manager Dua Dua	\$2y\$12\$9nAL7bHt4JqTabMTuQEPgOtjcrPGOmZtqiYAqdTaMBP...	2025-03-06 12:58:23	2025-03-06 12:58:23
7	2	manager33	Manager Tiga Tiga	\$2y\$12\$8vdg4GsDgnQ1lowpai34vOafe2pumR4C/AjKxDhRaK6...	2025-03-06 13:44:08	2025-03-06 13:44:08
10	2	manager56	Manager55	\$2y\$12\$I5utX0J4x1GBuSmUiAYXxuwvmvnhdjjHzti./4MwDOQH...	2025-03-06 13:59:31	2025-03-06 13:59:31
11	2	manager55	Manager55	\$2y\$12\$kbJe85gvQF/lmuShljk.yuqxLfkWVp7vyUPLfMjnbcn...	2025-03-06 14:01:01	2025-03-06 14:01:01
12	2	manager12	Manager11	\$2y\$12\$hcYPl6sEjalXoD641aKn8OkasHw2uC8XoFE3WD91kRh...	2025-03-06 14:10:57	2025-03-06 14:10:57
19	1	adminbaru	AdminBaru	\$2y\$12\$BauMSxtiXptRFKlSl5glu.PkR9Vn6XXdTauOYqBE1d/...	2025-04-08 19:08:39	2025-04-08 19:08:39
20	2	managerbaru	ManagerBaru	\$2y\$12\$gP5bK5gVqySLZSmcr7b9Hu7JhT5U2FZznZPIHn/onTu...	2025-04-08 19:47:57	2025-04-08 19:47:57
22	1	staffbaru	StaffBaru	\$2y\$12\$7VD8k.dJGU1pSMpCyAhtUe7T0Jg7KcWn6KxxSy0bFss...	2025-04-09 23:44:25	2025-04-09 23:44:25
23	3	staffmagang	Staff Magang	\$2y\$12\$FYHj52wz5Ka0jOmkDLRs7.1ryWWAtPuOeMYGpXBTCBg...	2025-04-12 10:00:27	2025-04-12 10:00:27

- User baru sudah berhasil masuk



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

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