Nama	**: Dini Elminingtyas**
Kelas/No.absen	**: SIB 2B / 07**
NIM	**: 2341760180**

PEMROGRAMAN WEB LANJUT

JOBSHEET 07

AUTHENTICATION and AUTHORIZATION

di LARAVEL

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
```

⇨ Source code

```
61
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
<?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');
    }
}
```

⇨ Source code

Minggu 7 (PWL_POS) > app > Models > UserModel.php > UserModel

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasOne;

class UserModel extends Model
{
    // public function level() : HasOne {
    //     return $this->hasOne(LevelModel::class);
    // }

    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');
    }

}
```

3. Selanjutnya kita buat `AuthController.php` untuk memproses login yang akan kita lakukan

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

⇨ Source code

Minggu 7 (PWL_POS) > app > Http > Controllers > AuthController.php > ...

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

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>
```

⇨ Source code

Minggu 7 (PWL_POS) > resources > views > auth > login.blade.php > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Login Pengguna</title>

    <!-- Google Font: Source Sans Pro -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">

    <!-- Font Awesome -->
    <link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">

    <!-- icheck bootstrap -->
    <link rel="stylesheet" href="{{ asset('plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">

    <!-- SweetAlert2 -->
    <link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">

    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
</head>
```

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

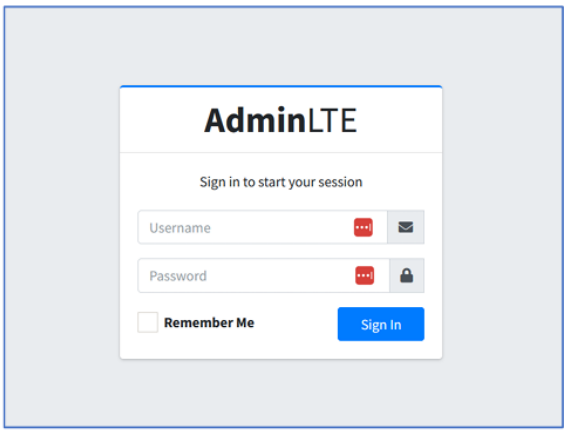
```
<?php

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

});
```

⇨ Source code

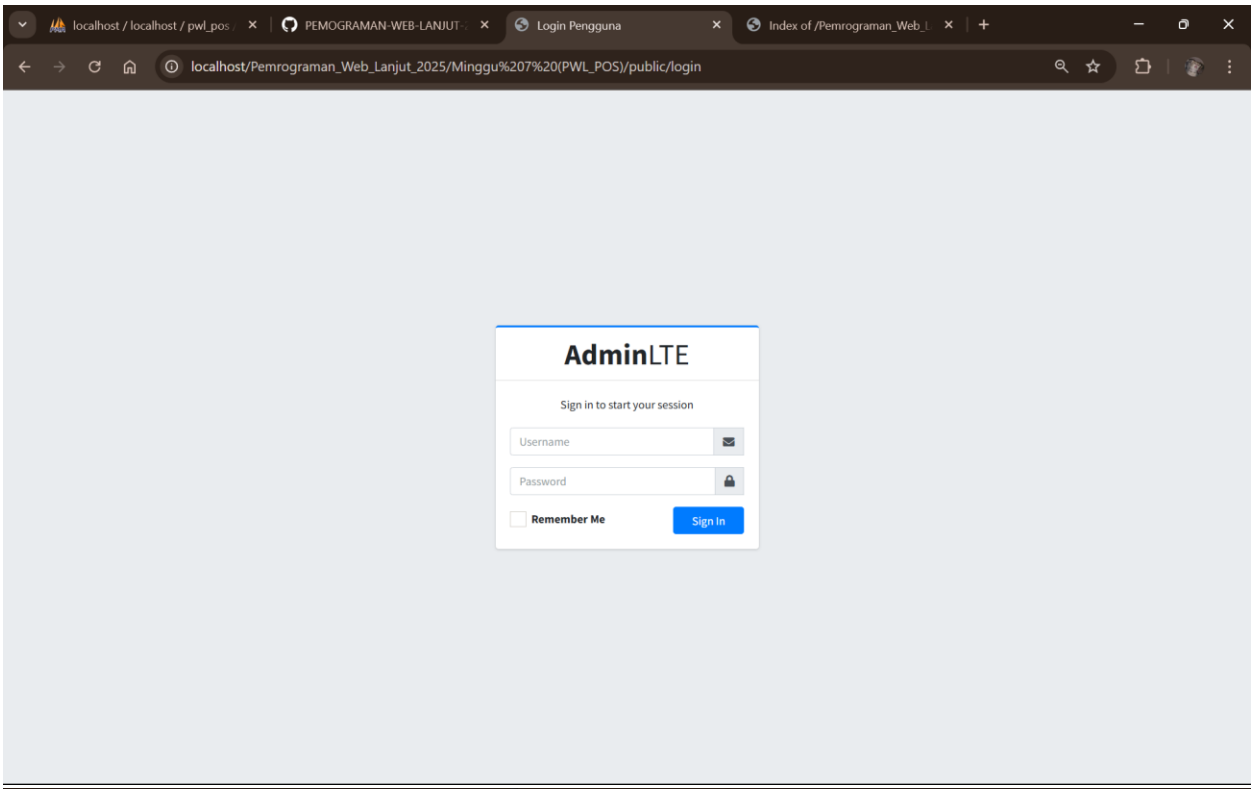
Minggu 7 (PWL_POS) > routes > web.php > ...

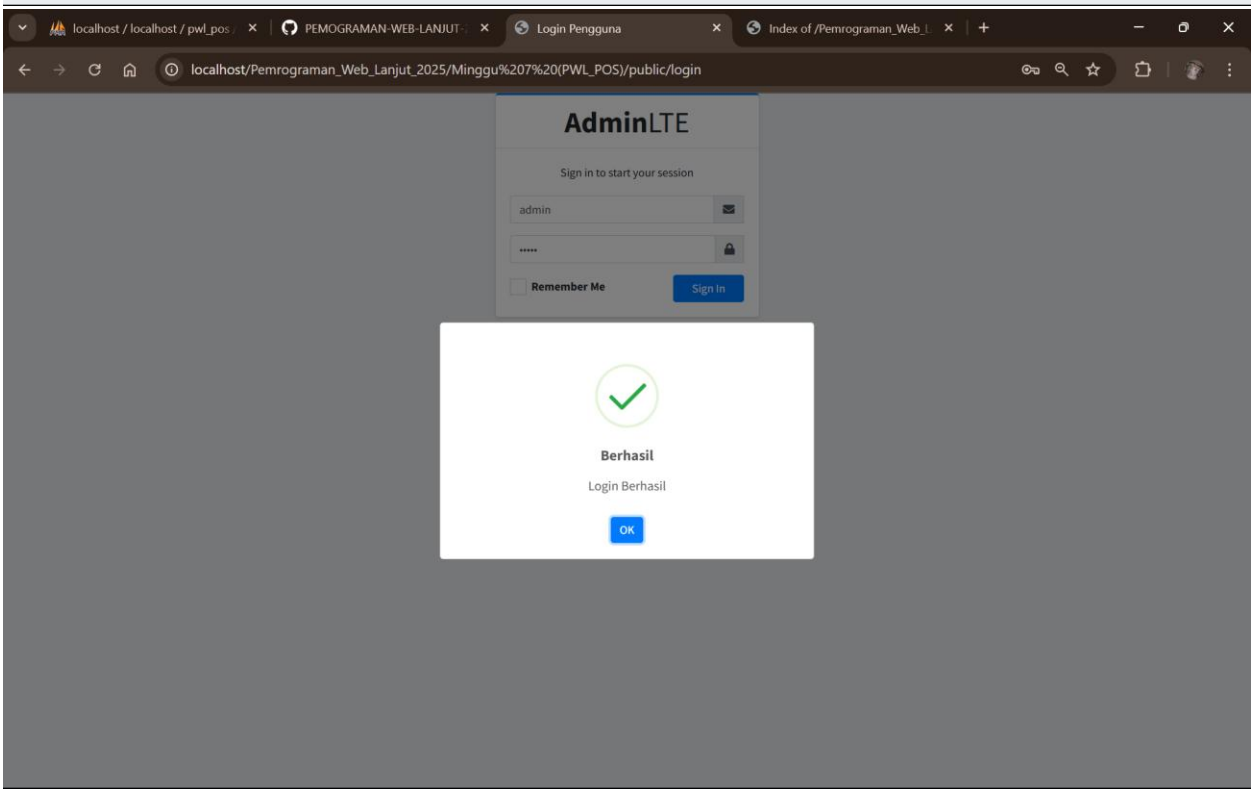
```
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {···
})->name('home');


Route::get('/level', [LevelController::class, 'index']);
// Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);

//praktikum 2.6
Route::get('/user/tambah', [UserController::class, 'tambah']);
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
Route::get('user/ubah/{id}', [UserController::class, 'ubah']);
Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);
Route::get('/user/hapus/{id}', [UserController::class, 'hapus']);

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

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



⇨ Halaman Browser

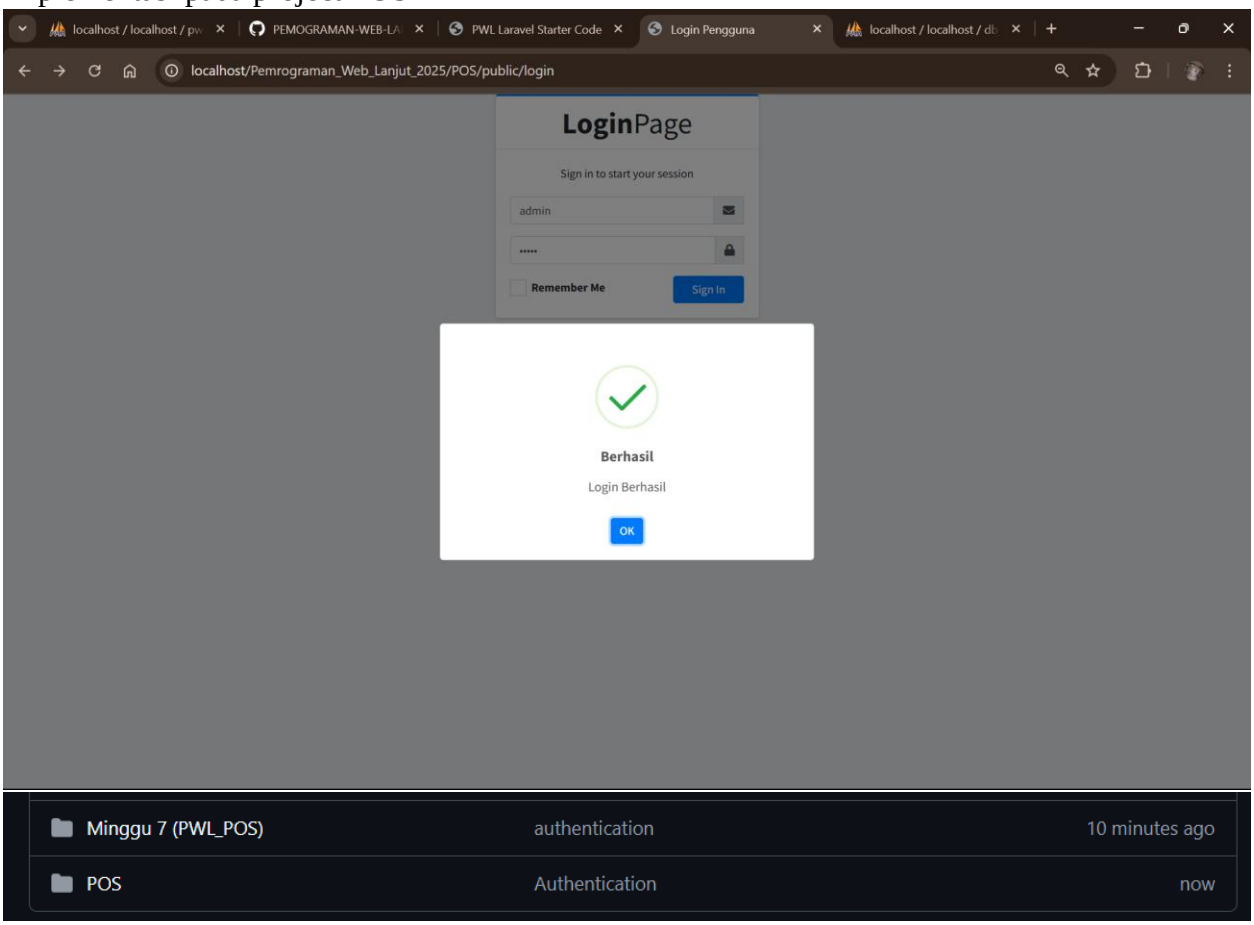

localhost / localhost / pwl_pos
PEMOGRAMAN-WEB-LANJUT
Login Pengguna
Index of /Pemrograman_Web_L
localhost/Pemrograman_Web_Lanjut_2025/Minggu%207%20(PWL_POS)/public/login
AdminLTE
Sign in to start your session
Username
Password
Remember Me
Sign In


localhost / localhost / pwl_pos
PEMOGRAMAN-WEB-LANJUT
Login Pengguna
Index of /Pemrograman_Web_L
localhost/Pemrograman_Web_Lanjut_2025/Minggu%207%20(PWL_POS)/public/login
AdminLTE
Sign in to start your session
admin
Remember Me
Sign In
Berhasil
Login Berhasil
OK

Tugas 1 – Implementasi Authentication :

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

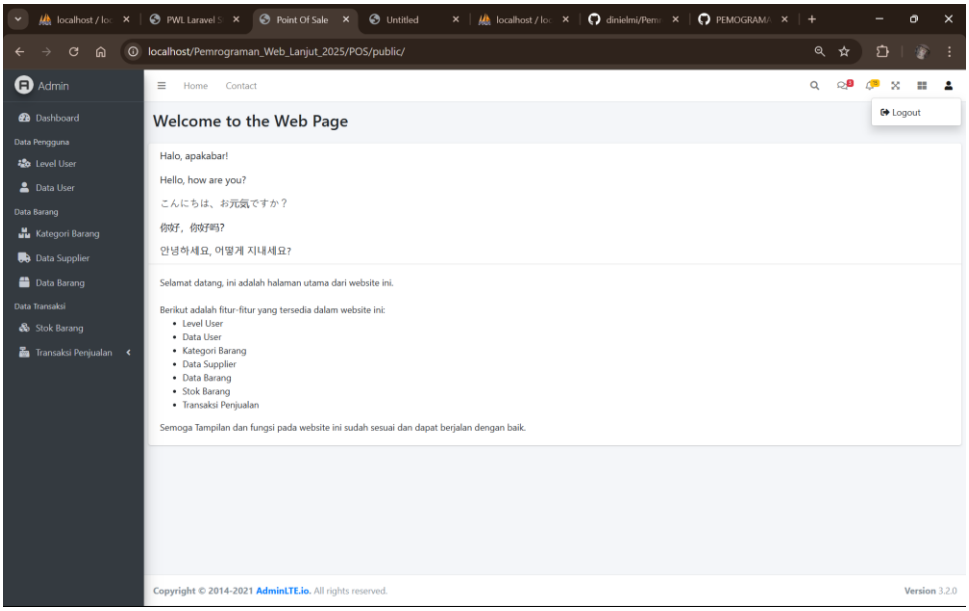
⇨ Implementasi pada project POS



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

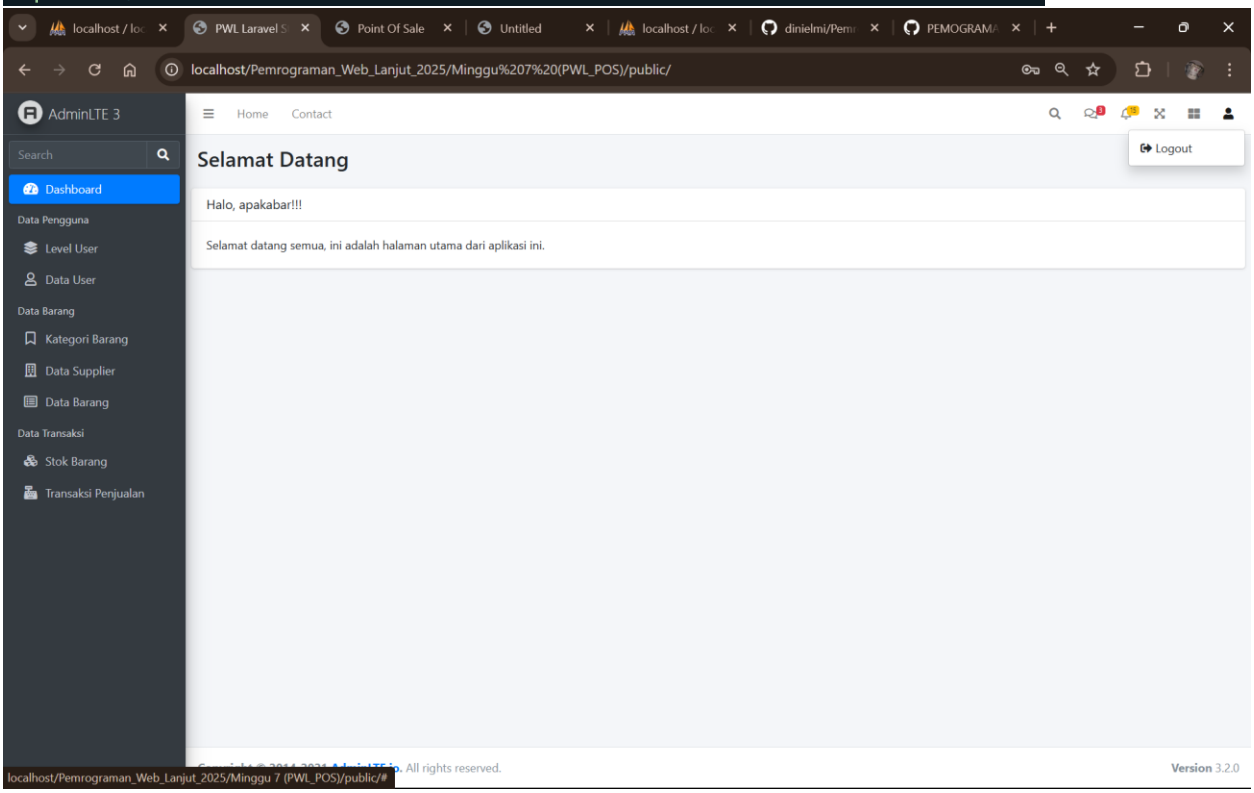
⇨ Logout project POS

```
        <!-- Profile Dropdown Menu -->
        <li class="nav-item dropdown">
          <a class="nav-link" data-toggle="dropdown" href="#">
              <i class="fas fa-user"></i> <!-- Ikon profil -->
          </a>
          <div class="dropdown-menu dropdown-menu-right">
              <a class="dropdown-item" href="#" onclick="event.preventDefault(); document.getElementById('logout-form').submit();">
                  <i class="fas fa-sign-out-alt"></i> Logout <!-- Ikon logout -->
              </a>
              <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">
                  @csrf
              </form>
          </div>
      </li>
```



⇨ Logout praktikum 1 minggu 7

```
            <!-- Profile Dropdown Menu -->
            <li class="nav-item dropdown">
              <a class="nav-link" data-toggle="dropdown" href="#">
                  <i class="fas fa-user"></i> <!-- Ikon profil -->
              </a>
              <div class="dropdown-menu dropdown-menu-right">
                  <a class="dropdown-item" href="#" onclick="event.preventDefault(); document.getElementById('logout-form').submit();">
                      <i class="fas fa-sign-out-alt"></i> Logout <!-- Ikon Logout -->
                  </a>
                  <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">
                      @csrf
                  </form>
              </div>
            </li>
```

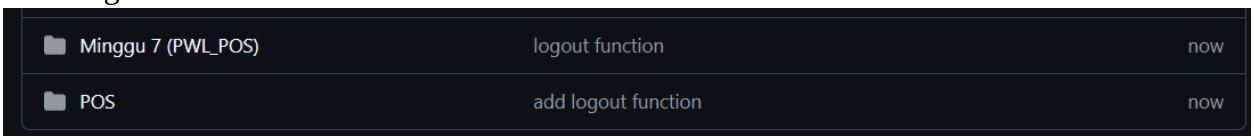


3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan

⇨ Tahap pertama yang dilakukan untuk menambahkan proses autentikasi sebelum ke websitanya adalah mengatur auth, supaya sesuai dengan database yang kita punya, dilakukan melalui UserModel, kemudian menambahkan Controller untuk melakukan fungsi autentikasi, dan menghubungkan dengan view yang dibuat yaitu login.blade. Terakhir menambahkan route autentikasi, dan menambahkan route ke menu yang perlu diautentikasi didalamnya. Untuk proses logout hanya perlu menambahkan menu/button untuk logout pada tampulan webpage kemudian membuat function logout di controller yang sama dengan login, dan menentukan routenya.

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

⇨ Submit github



Praktikum 2 - Implementasi Authorizaton di Laravel dengan Middleware

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

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

⇨ Source code

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

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

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

2. Kemudian kita buat *middleware* dengan nama `AuthorizeUser.php`. Kita bisa buat *middleware* dengan mengetikkan perintah pada terminal/CMD

```
php artisan make:middleware AuthorizeUser
```

File *middleware* akan dibuat di `app/Http/Middleware/AuthorizeUser.php`

⇨ Source code

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 7 (PWL_POS)> php artisan make:middleware AuthorizeUser

  INFO  Middleware [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 7 (PWL_POS)\app\Http\Middleware\AuthorizeUser.php] created successfully.
```

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

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

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

class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
     */
    public function handle(Request $request, Closure $next, $role = ''): Response
    {
        $user = $request->user();   // ambil data user yg login
                                    // fungsi user() diambil dari UserModel.php
        if($user->hasRole($role)){  // cek apakah user punya role yg diinginkan
            return $next($request);
        }
        // jika tidak punya role, maka tampilkan error 403
        abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

⇨ Source code

app > Http > Middleware > AuthorizeUser.php > ...

```php
<?php

namespace App\Http\Middleware;

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

class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next, $role = ''): Response
    {
        $user = $request->user();    // ambil data user yg login
                                     // fungsi user() diambil dari UserModel.php
        if($user->hasRole($role)){  // cek apakah user punya role yg diinginkan
            return $next($request);
        }
        // jika tidak punya role, maka tampilkan error 403
        abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}

```

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

```php
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,
```

⇨ Source code

```php
    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

⇨ Database

	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADMIN	Administrator	NULL	2025-03-23 15:13:20
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
Edit Copy Delete	4	CUS	Customer	2025-03-02 08:21:06	2025-03-02 08:21:06
Edit Copy Delete	8	KURIR	kurir	2025-04-02 04:42:01	2025-04-02 05:08:51

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

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

    // artinya semua route di dalam group ini harus punya role ADM (Administrator)
    Route::middleware(['authorize:ADM'])->group(function(){
        Route::get('/level',[LevelController::class,'index']);
        Route::post('/level/list',[LevelController::class,'list']); // untuk list json datatables
        Route::get('/level/create',[LevelController::class,'create']);
        Route::post('/level',[LevelController::class,'store']);
        Route::get('/level/{id}/edit',[LevelController::class,'edit']); // untuk tampilkan form edit
        Route::put('/level/{id}',[LevelController::class,'update']);    // untuk proses update data
        Route::delete('/level/{id}',[LevelController::class,'destroy']); // untuk proses hapus data
    });

    // route Kategori
```

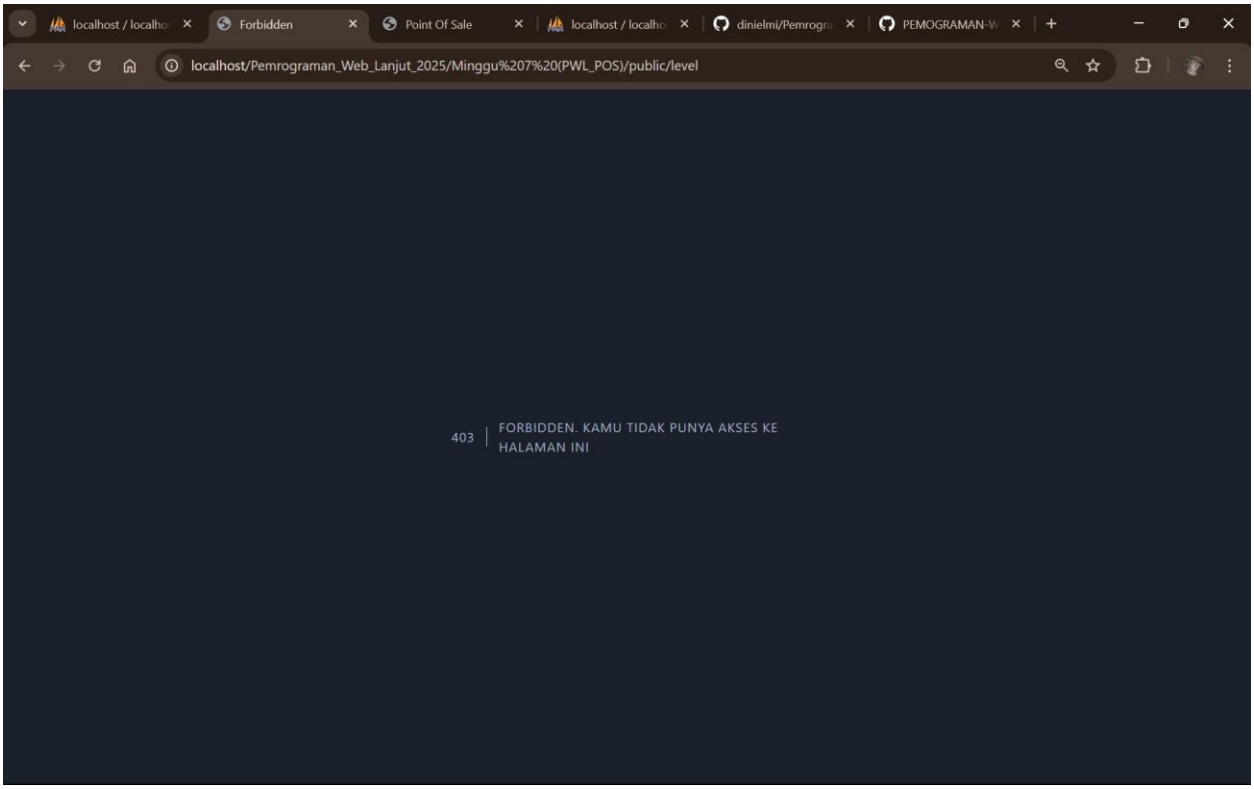
Pada kode yang ditandai merah, terdapat `authorize:ADM` . Kode `ADM` adalah nilai dari `level_kode` pada tabel `m_level`. Yang artinya, user yang bisa mengakses route untuk manage data level, adalah user yang memiliki level sebagai Administrator.

⇨ Source code

```php
Route::middleware(['authorize:ADMIN'])->group(function () {
    Route::group(['prefix' => 'level'], function () {
        Route::get('/',[LevelController::class, 'index']);
        Route::post('/list',[LevelController::class, 'list']);
        Route::get('/create',[LevelController::class, 'create']);
        Route::post('/',[LevelController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [LevelController::class, 'create_ajax']);
        Route::post('/ajax', [LevelController::class, 'store_ajax']);
        Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']);
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']);

        Route::get('/{id}',[LevelController::class, 'show']);
        Route::get('/{id}/edit',[LevelController::class, 'edit']);
        Route::put('/{id}',[LevelController::class, 'update']);
        Route::delete('/{id}',[LevelController::class, 'destroy']);
    });
});
```

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

⇨ Halaman Browser



Tugas 2 - Implementasi Authorization :

1. Apa yang kalian pahami pada praktikum 2 ini?

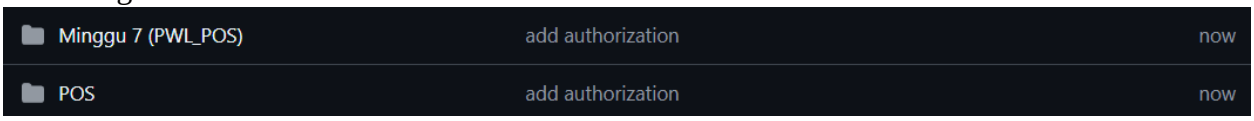
⇨ Pada praktikum 2, saya memahami implementasi Authorization di Laravel yang berfungsi membatasi akses pengguna berdasarkan role atau levelnya. Dengan menggunakan middleware seperti authorize, sistem secara otomatis memfilter akses sehingga hanya pengguna dengan hak yang sesuai yang dapat mengakses fitur tertentu. Hal ini sangat penting untuk mengamankan aplikasi dan memastikan bahwa data sensitif hanya dapat diakses oleh pengguna yang berwenang.

2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan

⇨ Pertama, saya memodifikasi middleware dengan membuat AuthorizeUser.php yang berfungsi mengecek apakah pengguna memiliki level tertentu. Middleware ini akan memeriksa role user dengan method hasRole($role) dan meneruskan request jika sesuai atau menampilkan error 403 jika tidak memiliki hak akses.
Kemudian, mendaftarkan middleware ini ke dalam app/Http/Kernel.php dengan memberi alias 'authorize' sehingga dapat digunakan untuk memfilter request berdasarkan level pengguna.
Terakhir, mengatur route dengan menerapkan middleware 'authorize' pada group route tertentu di routes/web.php. Ini memastikan bahwa hanya pengguna dengan level administrator yang dapat mengakses route-route sensitif seperti manajemen user dan pengaturan sistem.

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

⇨ Commit github



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

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

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

⇨ Source code

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

2. Selanjutnya, Kita modifikasi middleware `AuthorizeUser.php` dengan kode berikut

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

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

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

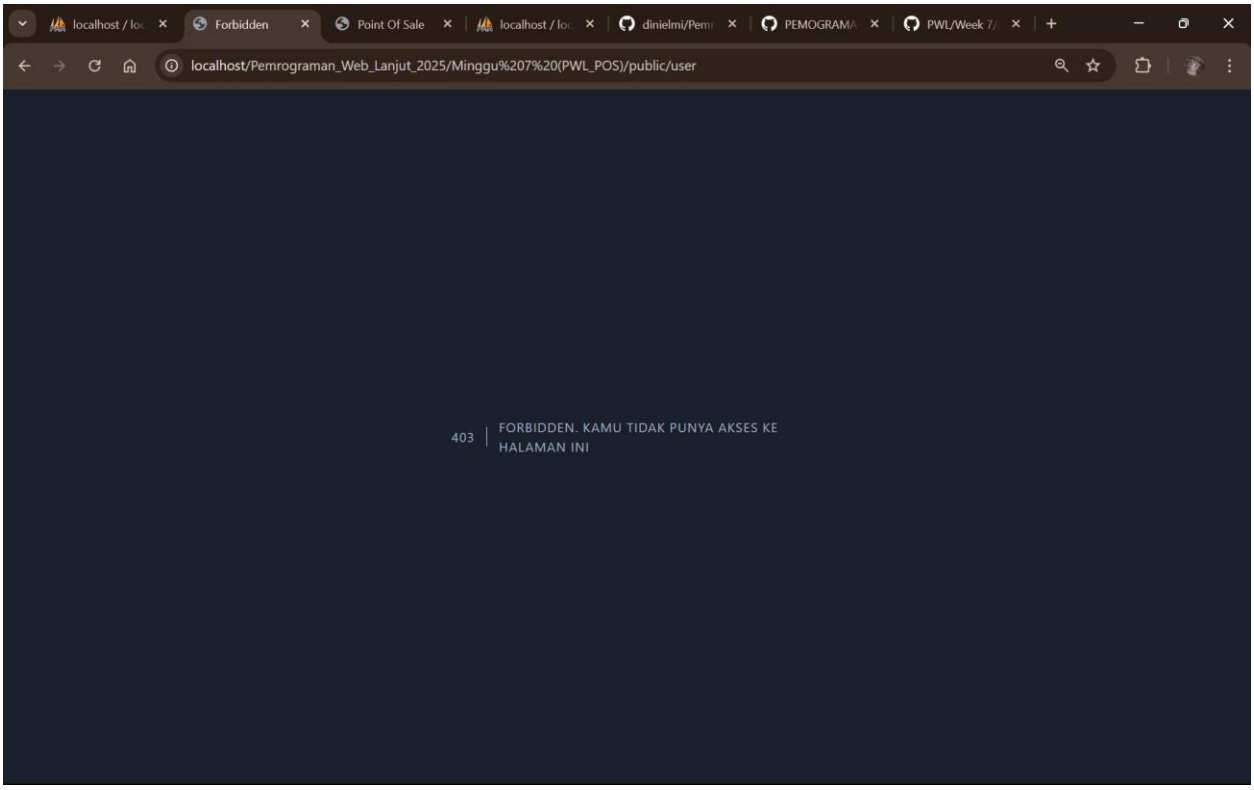
⇨ Source code

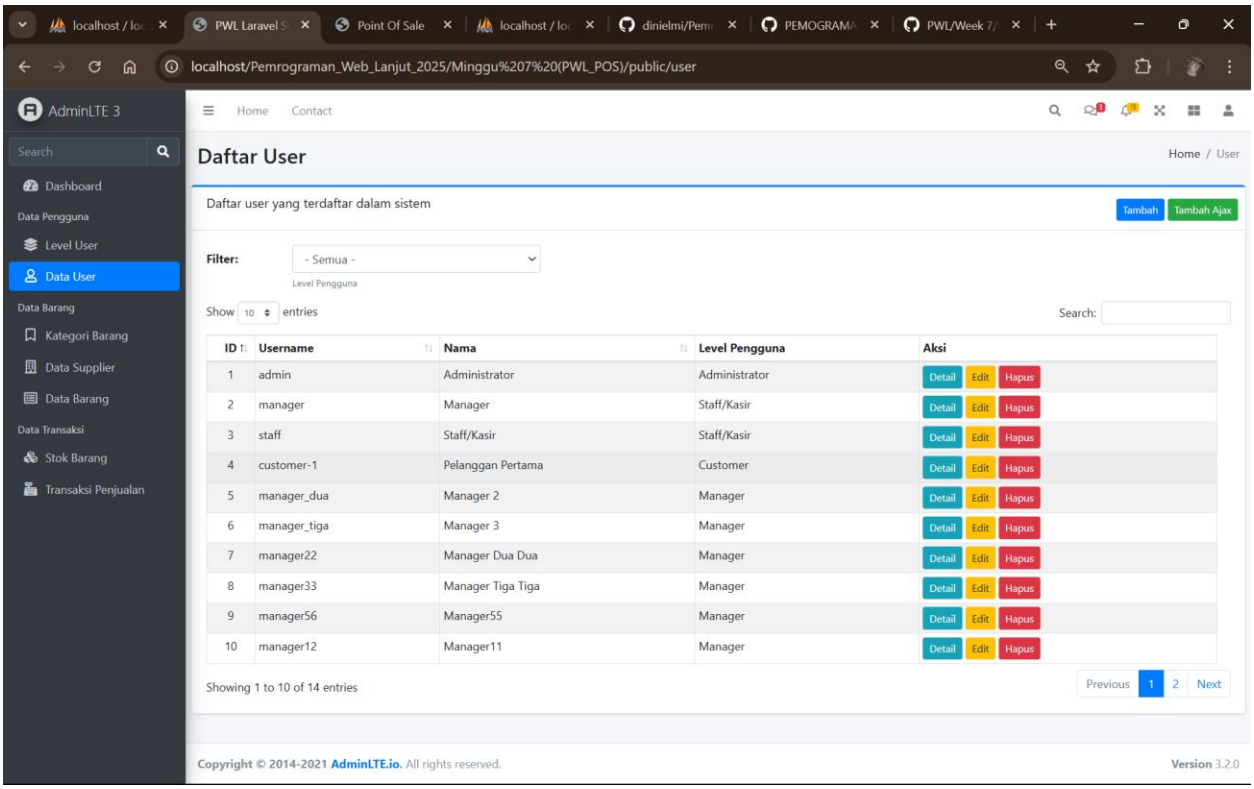
```php
    public function handle(Request $request, Closure $next, ...$roles): Response
    {
        $user_role = $request->user()->getRole();  // ambil data level_kode dari user yg login
        if (in_array($user_role, $roles)) {  // cek apakah level_kode user ada di dalam array roles
           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

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

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

⇨ Source code

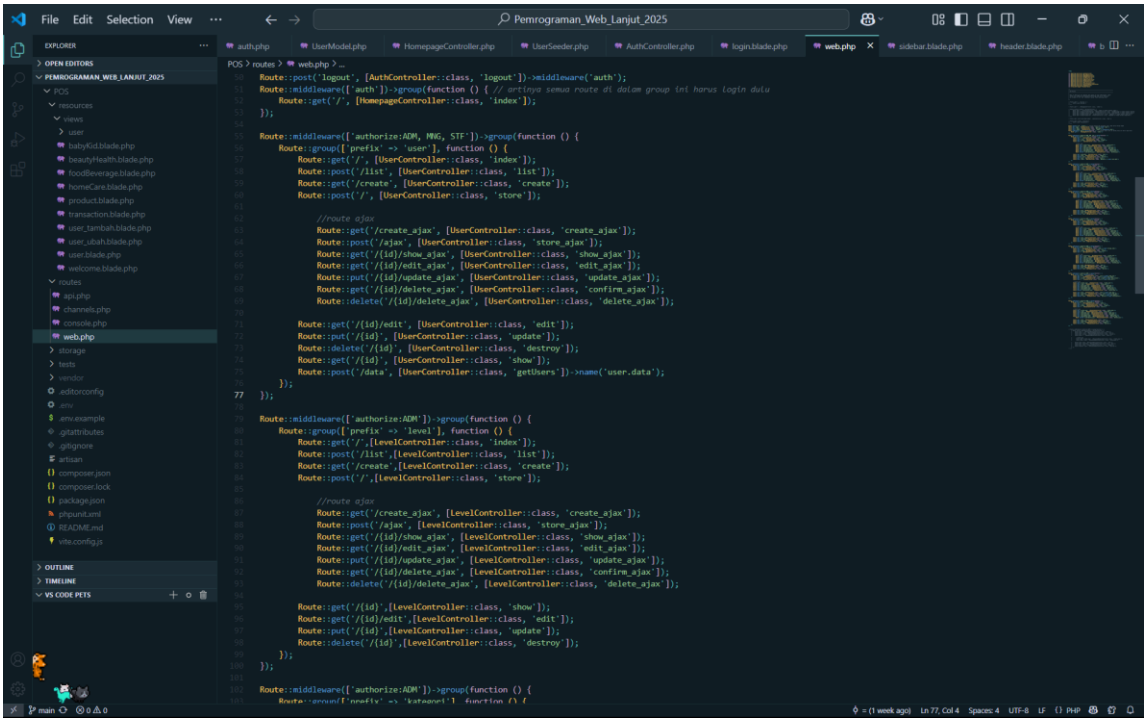




Tugas 3 – Implementasi Multi-Level Authorization :

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

⇨ Source Code

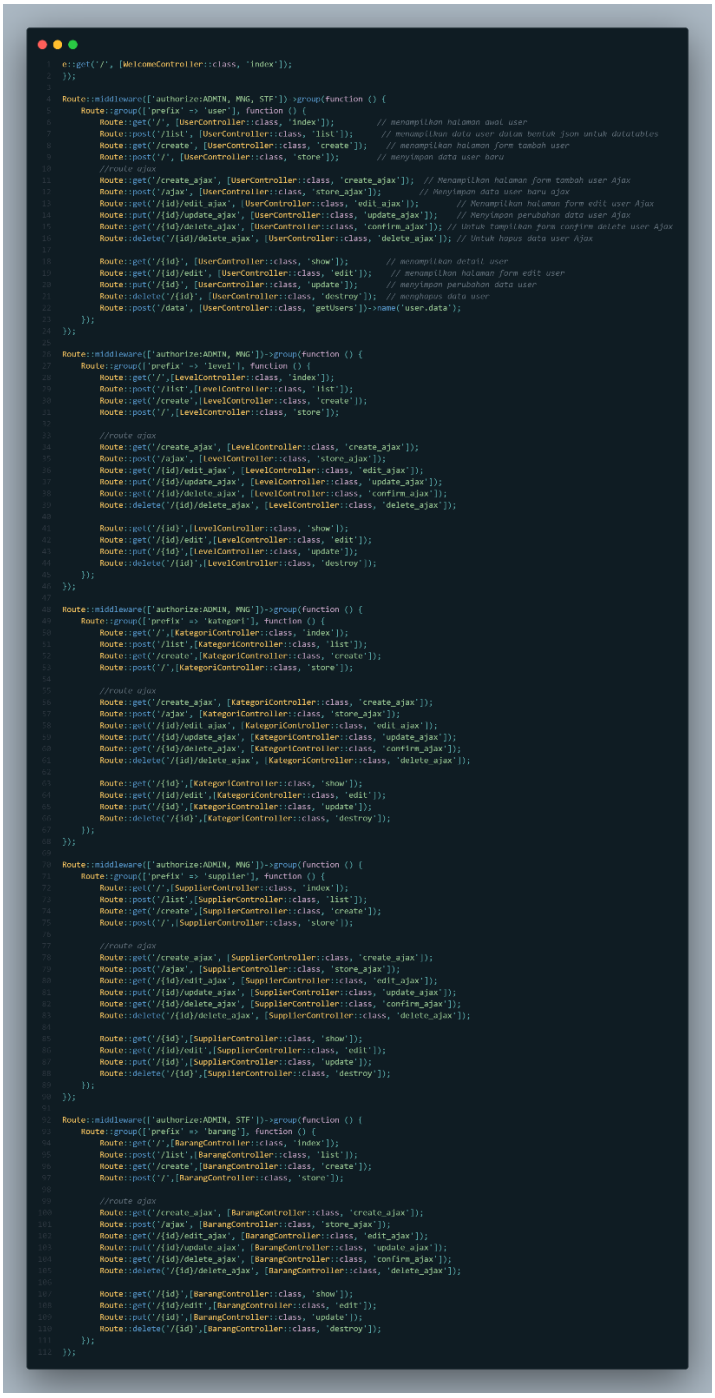


2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan

⇨ Pertama, buat sistem getrole pada UserModel. Kedua, modifikasi middleware untuk memeriksa berbagai level akses dengan implementasi fungsi getRole yang dapat memvalidasi apakah pengguna memiliki hak akses yang diperlukan. Terakhir, atur berbagai grup route dengan middleware authorization sesuai level

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

⇨ Source Code



```
e::get('/', [WelcomeController::class, 'index']);
});

Route::middleware(['authorize:ADMIN, MNG, STF'])->group(function () {
    Route::group(['prefix' => 'user'], function () {
        Route::get('/', [UserController::class, 'index']);          // menampilkan halaman awal user
        Route::post('/list', [UserController::class, 'list']);      // menampilkan data user dalam bentuk json untuk datatables
        Route::get('/create', [UserController::class, 'create']);   // menampilkan halaman form tambah user
        Route::post('/', [UserController::class, 'store']);         // menyimpan data user baru
        //route ajax
        Route::get('/create_ajax', [UserController::class, 'create_ajax']);  // Menampilkan halaman form tambah user Ajax
        Route::post('/ajax', [UserController::class, 'store_ajax']);         // Menyimpan data user baru ajax
        Route::get('/{id}/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::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::delete('/{id}', [UserController::class, 'destroy']);  // menghapus data user
        Route::post('/data', [UserController::class, 'getUsers'])->name('user.data');
    });
});

Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'level'], function () {
        Route::get('/',[LevelController::class, 'index']);
        Route::post('/list',[LevelController::class, 'list']);
        Route::get('/create',[LevelController::class, 'create']);
        Route::post('/',[LevelController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [LevelController::class, 'create_ajax']);
        Route::post('/ajax', [LevelController::class, 'store_ajax']);
        Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']);
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']);

        Route::get('/{id}',[LevelController::class, 'show']);
        Route::get('/{id}/edit',[LevelController::class, 'edit']);
        Route::put('/{id}',[LevelController::class, 'update']);
        Route::delete('/{id}',[LevelController::class, 'destroy']);
    });
});

Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'kategori'], function () {
        Route::get('/',[KategoriController::class, 'index']);
        Route::post('/list',[KategoriController::class, 'list']);
        Route::get('/create',[KategoriController::class, 'create']);
        Route::post('/',[KategoriController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [KategoriController::class, 'create_ajax']);
        Route::post('/ajax', [KategoriController::class, 'store_ajax']);
        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}',[KategoriController::class, 'show']);
        Route::get('/{id}/edit',[KategoriController::class, 'edit']);
        Route::put('/{id}',[KategoriController::class, 'update']);
        Route::delete('/{id}',[KategoriController::class, 'destroy']);
    });
});

Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'supplier'], function () {
        Route::get('/',[SupplierController::class, 'index']);
        Route::post('/list',[SupplierController::class, 'list']);
        Route::get('/create',[SupplierController::class, 'create']);
        Route::post('/',[SupplierController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [SupplierController::class, 'create_ajax']);
        Route::post('/ajax', [SupplierController::class, 'store_ajax']);
        Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']);
        Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']);

        Route::get('/{id}',[SupplierController::class, 'show']);
        Route::get('/{id}/edit',[SupplierController::class, 'edit']);
        Route::put('/{id}',[SupplierController::class, 'update']);
        Route::delete('/{id}',[SupplierController::class, 'destroy']);
    });
});

Route::middleware(['authorize:ADMIN, STF'])->group(function () {
    Route::group(['prefix' => 'barang'], function () {
        Route::get('/',[BarangController::class, 'index']);
        Route::post('/list',[BarangController::class, 'list']);
        Route::get('/create',[BarangController::class, 'create']);
        Route::post('/',[BarangController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [BarangController::class, 'create_ajax']);
        Route::post('/ajax', [BarangController::class, 'store_ajax']);
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']);
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);

        Route::get('/{id}',[BarangController::class, 'show']);
        Route::get('/{id}/edit',[BarangController::class, 'edit']);
        Route::put('/{id}',[BarangController::class, 'update']);
        Route::delete('/{id}',[BarangController::class, 'destroy']);
    });
});
```

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

⇨ Github

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

⇨ Source code

```
                    </div>
                    <div class="col-12 text-center mt-3">
                        <a href="{{ url('/register') }}">Don't have an account? Register here</a>
                    </div>
```

```
    public function register(){
        $level = LevelModel::select('level_id','level_nama')->get();

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

    public function store_user(Request $request)
    {
        $request->validate([
            'username' => 'required|string|min:3|unique:m_user,username',
            'nama'     => 'required|string|max:100',
            'password' => 'required|min:5',
            'level_id' => 'required|integer',
        ]);

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

        // Mengirim respons JSON jika permintaan AJAX
        if ($request->ajax() || $request->wantsJson()) {
            return response()->json([
                'status' => true,
                'message' => 'Registrasi Berhasil',
                'redirect' => url('login')
            ]);
        }

        // Jika bukan AJAX, redirect ke halaman utama dengan flash message
        return redirect('/')->with('success', 'Registrasi berhasil');
    }
```

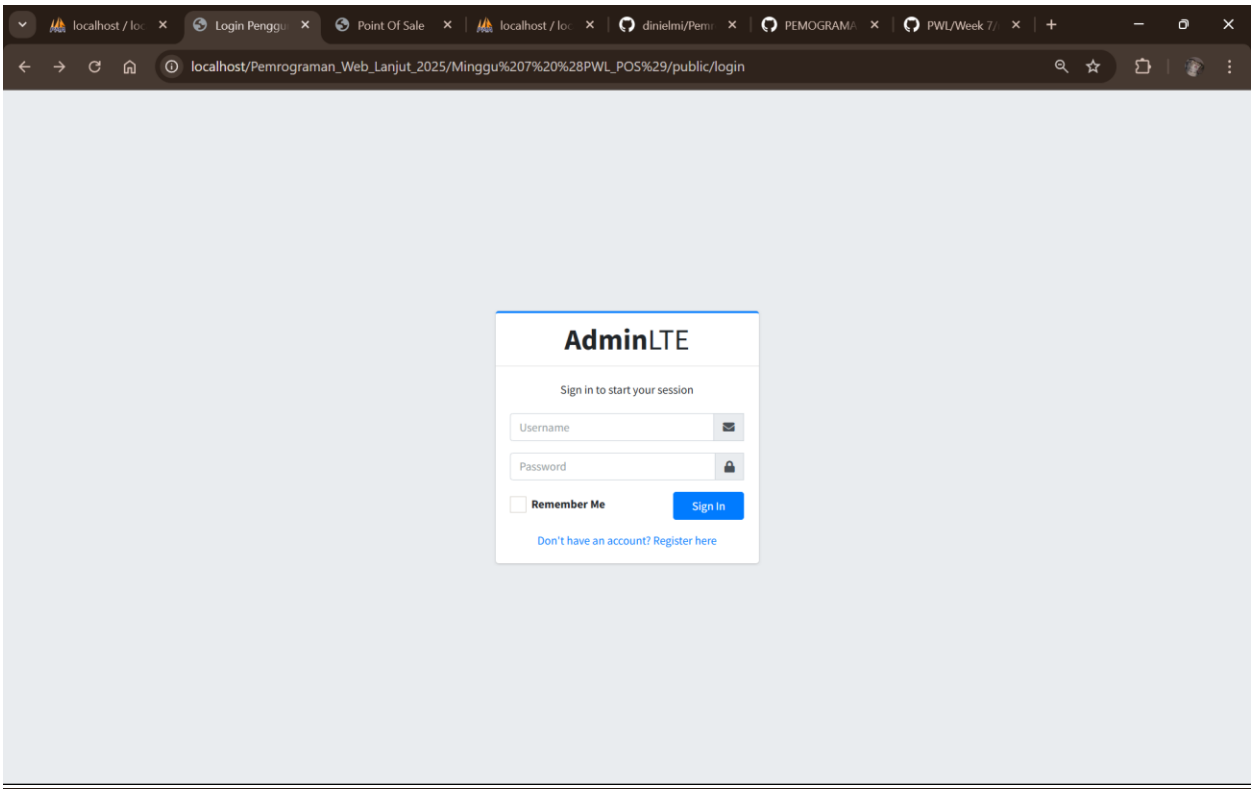
```php
Route::get('/register', [AuthController::class, 'register'])->name('register');
Route::post('/register', [AuthController::class, 'store_user'])->name('store_user');
```

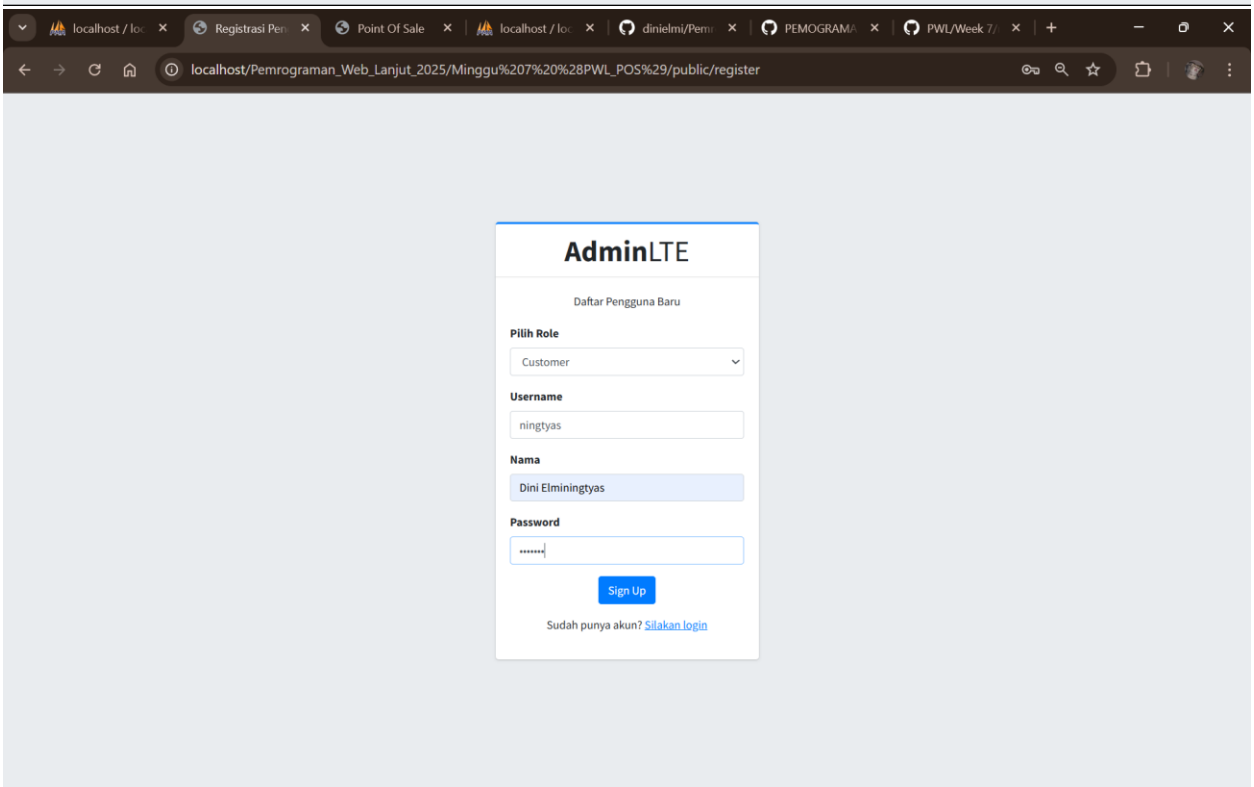
resources > views > auth > register.blade.php > html > body.hold-transition.login-page > div.login-box > div.card.card-outline.card-primary > div.card-body > form#form-register > div.form-group

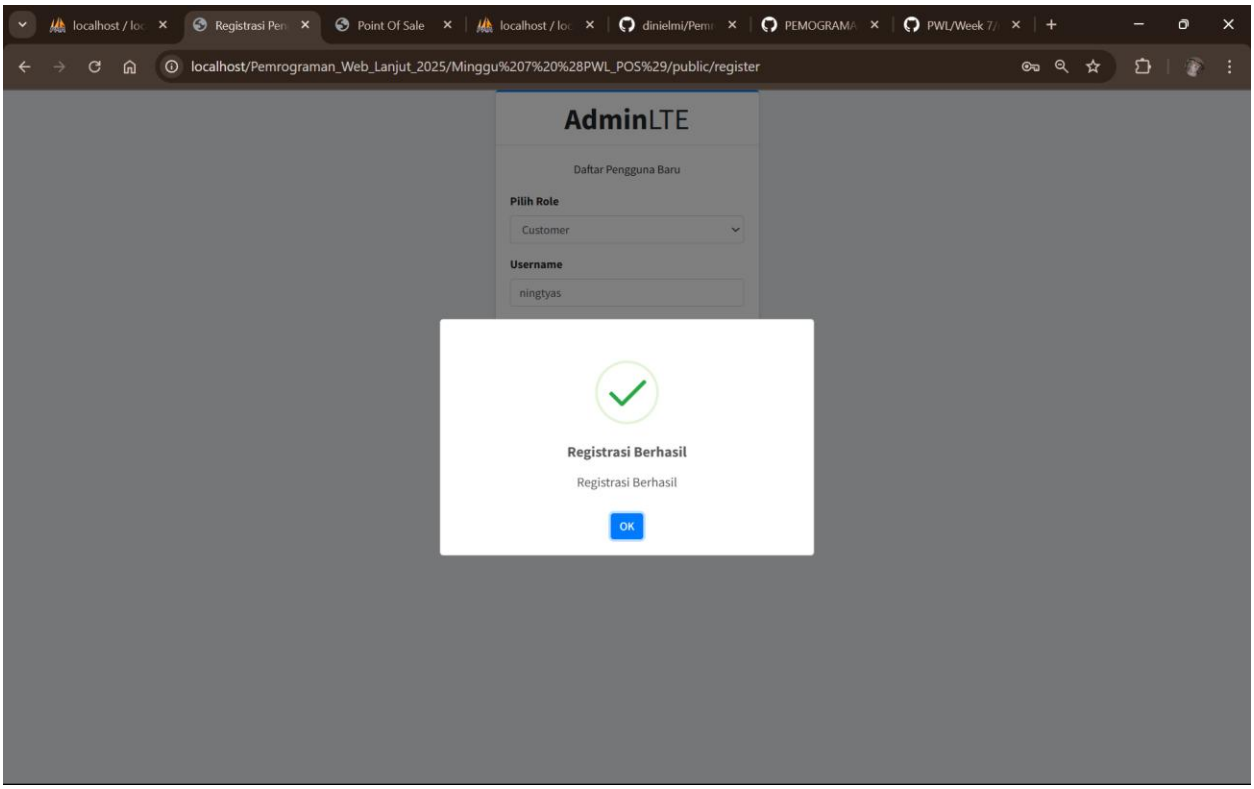
```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>Registrasi Pengguna</title>

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

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

2. Screenshot hasil yang kalian kerjakan

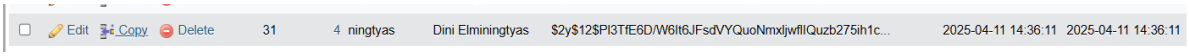
⇨ Halaman browser


localhost / loc
Login Penggu
Point Of Sale
localhost / loc
dinielmi/Pemr
PEMOGRAMA
PWL/Week 7/
localhost/Pemrograman_Web_Lanjut_2025/Minggu%207%20%28PWL_POS%29/public/login
AdminLTE
Sign in to start your session
Username
Password
Remember Me
Sign In
Don't have an account? Register here


localhost / loc
Registrasi Pen
Point Of Sale
localhost / loc
dinielmi/Pemr
PEMOGRAMA
PWL/Week 7/
localhost/Pemrograman_Web_Lanjut_2025/Minggu%207%20%28PWL_POS%29/public/register
AdminLTE
Daftar Pengguna Baru
Pilih Role
Customer
Username
ningtyas
Nama
Dini Elmiringtyas
Password
Sign Up
Sudah punya akun? Silakan login



database



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

⇨ Github