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
https://jti.polinema.ac.id

Mata Kuliah	:	Pemrograman Web Lanjut (PWL)
Program Studi	:	D4 – Teknik Informatika
Semester	:	4

Kelas	:	TI-2A
NIM	:	2341720124
Nama	:	Lelyta Meyda Ayu Budiyanti
Jobsheet Ke-	:	7

Laporan Jobsheet

	Praktikum 1 – Implementasi Authentication :

1	Saya buka kembali project laravel PWL_POS, dan saya modifikasi konfigurasi aplikasi saya di config/auth.php
	`'providers' => [`
	`  'users' => [`
	`  'driver' => 'eloquent',`
	`  'model' => App\Models\User::class,`
	`  ],`
	Pada bagian ini saya sesuaikan dengan Model untuk tabel m_user yang sudah saya buat
	`'providers' => [`
	`  'users' => [`
	`  'driver' => 'eloquent',`
	`  'model' => App\Models\UserModel::class,`
	`  ],`

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

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

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Foundation\Auth\User as Authenticatable; //Implementasi class Authenticatable


class UserModel extends Authenticatable
{
    use HasFactory;

    protected $table = 'm_user'; //mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primaryKey = 'user_id'; //mendefinisikan primary key tabel yang digunakan oleh model ini

    protected $fillable = [
        'level_id',
        'username',
        'nama',
        'password',
        'created_at',
        'updated_at'
    ];

    protected $hidden = ['password'];

    protected $casts = ['password' => 'hashed'];

    public function level():BelongsTo {

        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
    }

}
```

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

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()) {
            $validatedData = $request->validate([
                'username' => 'required|string',
                'password' => 'required|string',
            ]);

            $credentials = $request->only('username', 'password');
            if (Auth::attempt($credentials)) {
                $request->session()->regenerate();
                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 saya membuat AuthController.php, saya membuat view untuk menampilkan halaman login. View saya buat di auth/login.blade.php , tampilan login bisa saya ambil dari contoh login di template AdminLTE

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

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

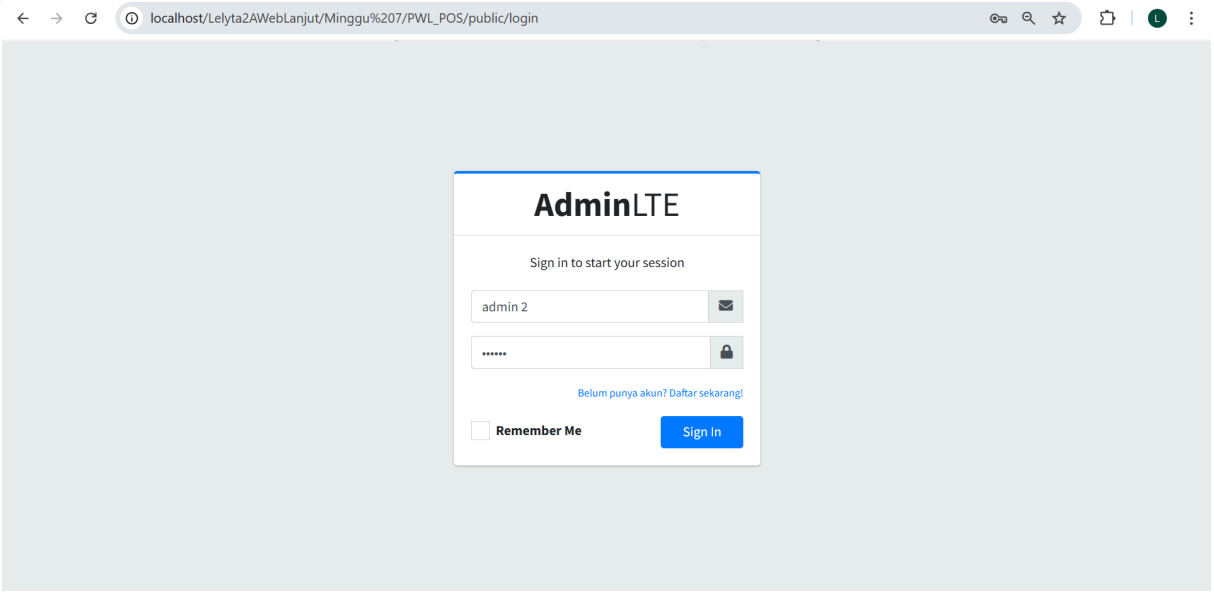
```blade
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Login Pengguna</title>
        <!-- Google Font: Source Sans Pro -->
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
        <!-- Font Awesome -->
        <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
        <!-- icheck bootstrap -->
        <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css')}}">
        <!-- SweetAlert2 -->
        <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
        <!-- Theme style -->
        <link rel="stylesheet" href="{{ asset('adminlte/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="mb-3 text-right">
                            <small>
                                <a href="{{url('register')}}" class="text-primary">Belum punya akun? Daftar sekarang!</a>
                            </small>
                        </div>
                        <div class="row">
                            <div class="col-8">
                                <div class="icheck-primary">
                                    <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
                                </div>
                            </div>
                            <!-- /.col -->
                            <div class="col-4">
                                <button type="submit" class="btn btn-primary btn-block">Sign In</button>
                            </div>
                            <!-- /.col -->
                        </div>
                    </form>
                </div>
                <!-- /.card-body -->
            </div>
            <!-- /.card -->
        </div>
        <!-- /.login-box -->
        <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
        <!-- Bootstrap 4 -->
        <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
        <!-- jquery-validation -->
        <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
        <script src="{{ asset('adminlte/plugins/jquery-validation/additional-methods.min.js') }}"></script>
        <!-- SweetAlert2 -->
        <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
        <!-- AdminLTE App -->
        <script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>

        <script>
            $.ajaxSetup({
            headers: {
                'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
            },
            error: function(xhr, status, error) {//sedikti tambahan debag gedebug
                console.error('AJAX Error: ' + status + error);
```

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

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

```php
// jobsheet 7
Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter (id), maka harus berupa angka

Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('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 saya coba mengakses halaman localhost/PWL_POS/public maka akan tampil halaman awal untuk login ke aplikasi, seperti berikut



Keterangan:

Halaman login pada sistem ini dirancang menggunakan template AdminLTE yang terintegrasi dengan framework Laravel. Halaman ini menyediakan antarmuka yang user-friendly dengan input untuk username, password, checkbox "Remember Me", dan link pendaftaran akun. Dari sisi backend, autentikasi pengguna dikendalikan oleh AuthController dengan metode login, postlogin, dan logout. Proses login menggunakan validasi dua arah, yaitu validasi sisi klien dengan jQuery Validation dan validasi sisi server menggunakan Laravel Request Validation. Jika kredensial valid, Laravel akan menjalankan Auth::attempt() untuk memverifikasi data pengguna berdasarkan tabel m_user melalui model UserModel, yang merupakan turunan dari Authenticatable agar dapat digunakan untuk autentikasi. Model ini juga memiliki relasi ke tabel level pengguna melalui method level() dengan relasi BelongsTo.

Apabila login berhasil, sistem akan menampilkan notifikasi keberhasilan menggunakan SweetAlert2 dan mengarahkan pengguna ke halaman utama. Jika gagal, sistem akan menampilkan pesan kesalahan yang spesifik pada input yang bermasalah. Semua request login dilakukan secara AJAX sehingga pengalaman pengguna menjadi lebih responsif. Routing login dan logout dikonfigurasi menggunakan Laravel Route dengan middleware auth untuk mengamankan halaman utama, sehingga hanya pengguna yang sudah login yang dapat

mengaksesnya. Secara keseluruhan, fitur login ini telah dibangun dengan pendekatan modular, aman, dan mengikuti best practice dari Laravel.

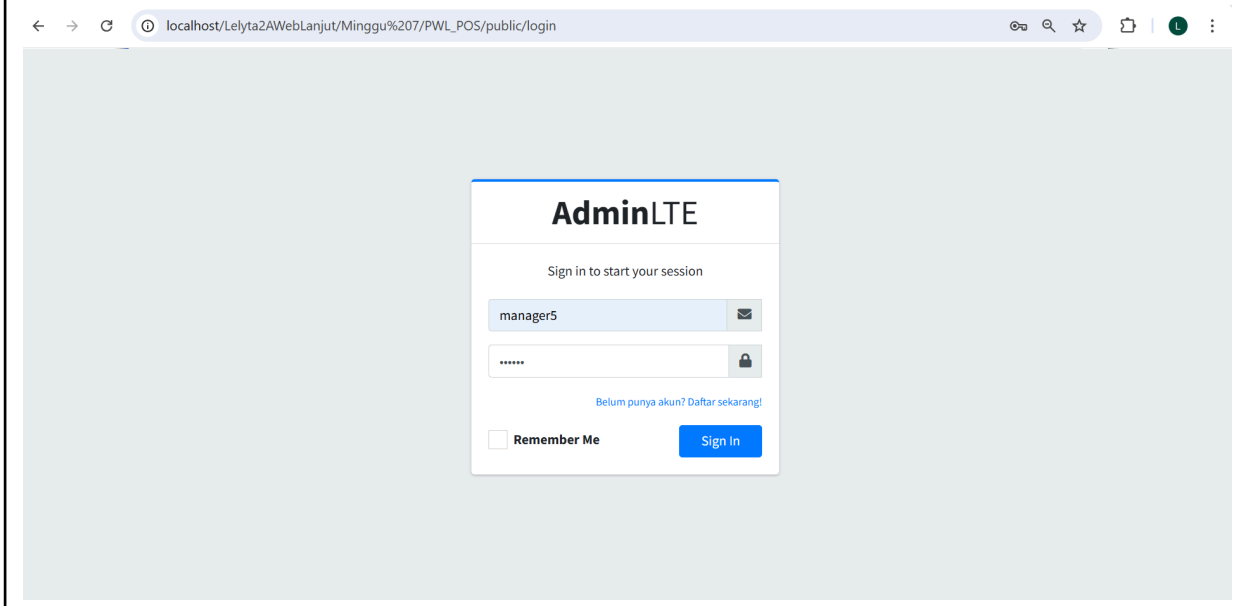
Tugas 1 – Implementasi Authentication

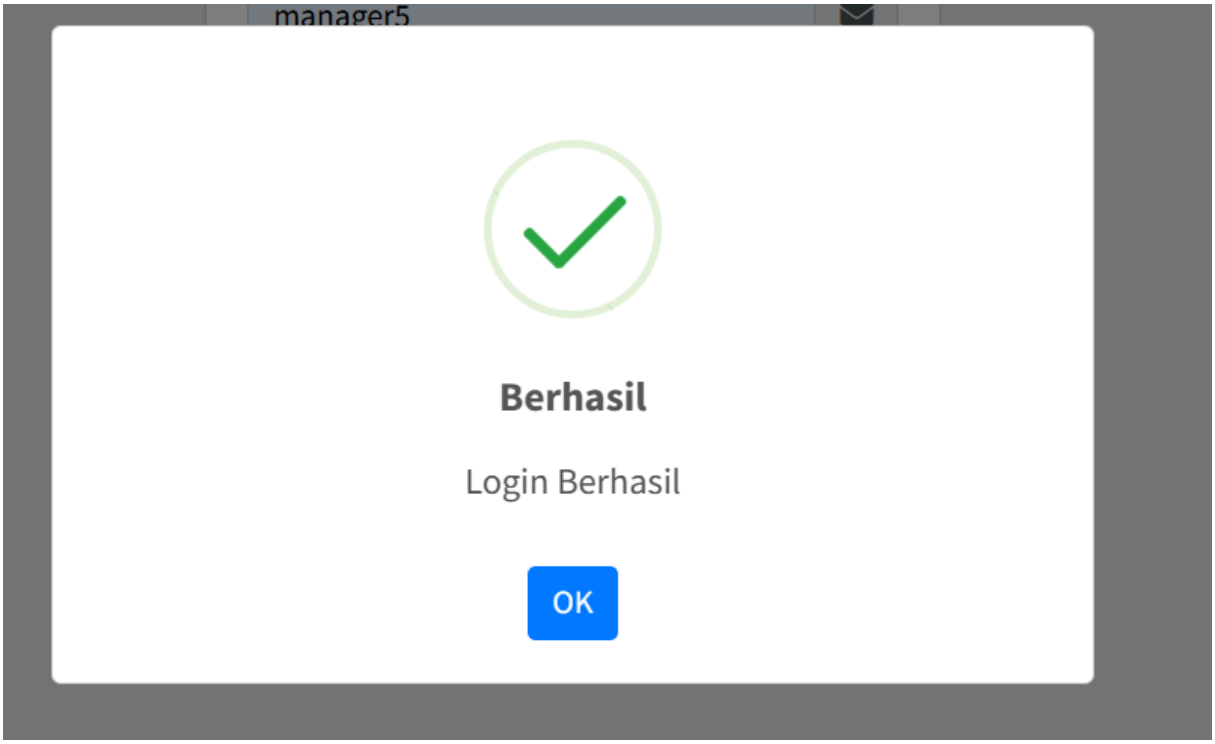
Berikut adalah tahap-tahap saya dalam mengimplementasikan proses login dan proses logout pada project PWL_POS saya:

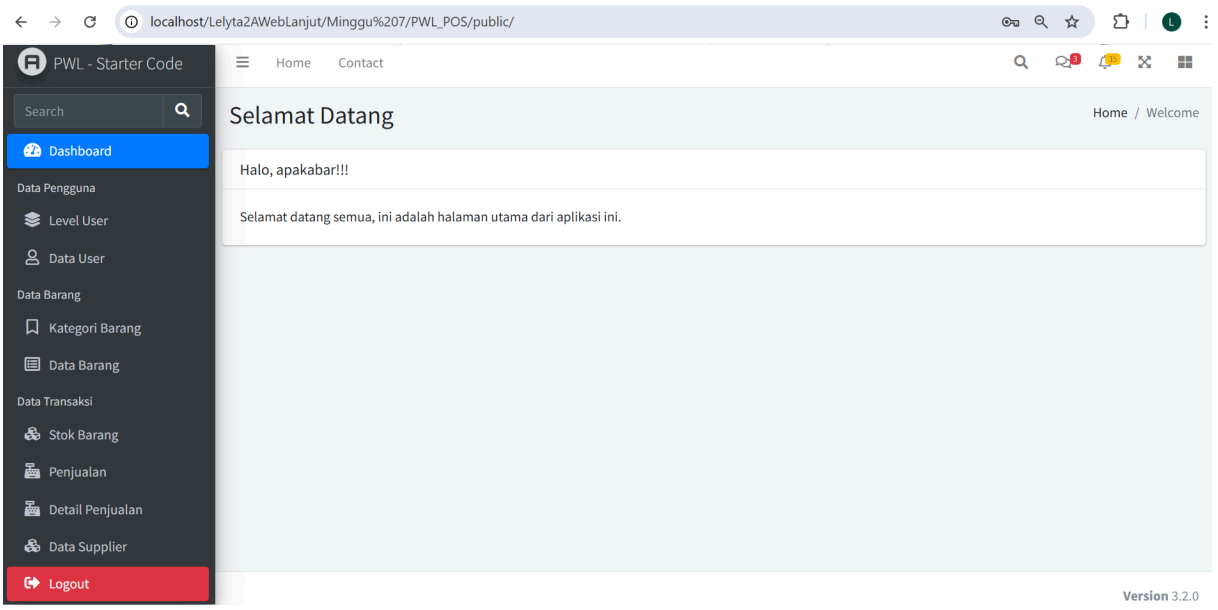
a. Saya menambahkan kode logout, saya menaruhnya di layout/sidebar.blade.php dan juga saya menambahkan pada route, berikut adalah kodenya untuk proses logout

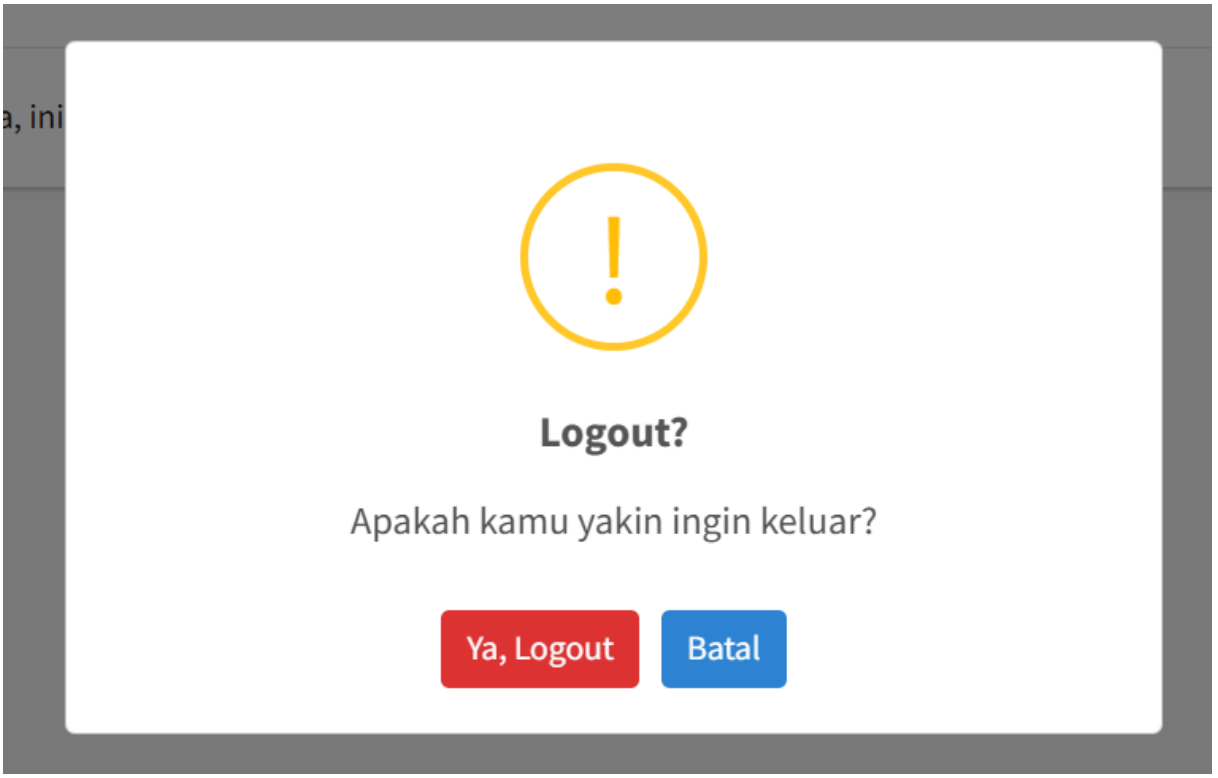
```
        </li>
        <!-- Logout -->
        <li class="nav-item">
            <a href="{{ url('/logout') }}" class="nav-link bg-danger text-white" onclick="confirmLogout(event)">
                <i class="nav-icon fas fa-sign-out-alt"></i>
                <p>Logout</p>
            </a>
        </li>
    </ul>
  </nav>
</div>

<!-- konfirmasi logout -->
<!-- SweetAlert2 -->
<script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
<script>
    function confirmLogout(event) {
        event.preventDefault();

        Swal.fire({
            title: 'Logout?',
            text: "Apakah kamu yakin ingin keluar?",
            icon: 'warning',
            showCancelButton: true,
            confirmButtonColor: '#d33',
            cancelButtonColor: '#3085d6',
            confirmButtonText: 'Ya, Logout',
            cancelButtonText: 'Batal'
        }).then((result) => {
            if (result.isConfirmed) {
                window.location.href = "{{ url('/logout') }}";
            }
        });
    }
</script>
```

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
https://jti.polinema.ac.id









Fitur logout pada sistem ini dilengkapi dengan konfirmasi menggunakan SweetAlert2 untuk memastikan bahwa pengguna benar-benar ingin keluar dari sesi login. Tombol logout ditampilkan di sidebar navigasi sebagai elemen <li> dengan ikon keluar dan label "Logout". Ketika tombol ini diklik, fungsi confirmLogout(event) akan dijalankan. Fungsi ini akan mencegah aksi default dari link (event.preventDefault()) dan menampilkan popup konfirmasi berupa alert bertipe warning. Popup ini memberikan dua opsi, yaitu "Ya, Logout" dan "Batal". Jika pengguna menekan tombol konfirmasi, maka browser akan diarahkan ke route /logout, yang sudah di-handle oleh controller untuk menghapus sesi dan mengarahkan kembali ke halaman login. Pendekatan ini menambah lapisan keamanan dan kenyamanan pengguna, sehingga tidak terjadi logout secara tidak sengaja. Integrasi ini juga konsisten dengan

	penggunaan SweetAlert2 pada fitur login sebelumnya, menjadikan tampilan alert lebih modern dan interaktif dibandingkan alert bawaan browser.
	Praktikum 2 – Implementasi Authorizaton di Laravel dengan Middleware :

1	Saya memodifikasi UserModel.php dengan menambahkan kode berikut
	Minggu 7 > PWL_POS > app > Models > UserModel.php
	`class UserModel extends Authenticatable`
	`{`
	`  use HasFactory;`
	
	`  protected $table = 'm_user'; //mendefinisikan nama tabel yang digunakan oleh model ini`
	`  protected $primaryKey = 'user_id'; //mendefinisikan primary key tabel yang digunakan oleh model ini`
	
	`  protected $fillable = [`
	`  'level_id',`
	`  'username',`
	`  'nama',`
	`  'password',`
	`  'created_at',`
	`  'updated_at'`
	`  ];`
	
	`  protected $hidden = ['password'];`
	
	`  protected $casts = ['password' => 'hashed'];`
	
	`  //public function level():BelongsTo {`
	
	`  //return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');`
	`  //}`
	
	`  // jobsheet 7`
	
	`  // Relasi ke tabel level`
	`  public function level(): BelongsTo`
	`  {`
	`  return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');`
	`  }`
	
	`  // Mendapatkan nama role`
	`  public function getRoleName(): string`
	`  {`
	`  return $this->level->level_nama;`
	`  }`
	
	`  // Cek apakah user memiliki role tertentu`
	`  public function hasRole($role): bool`
	`  {`
	`  return $this->level->level_kode == $role;`
	`  }`

2	Kemudian saya buat middleware dengan nama AuthorizeUser.php. Saya membuat middleware dengan mengetikkan perintah pada terminal/CMD, seperti berikut
	`C:\laragon\www\Lelyta2AWebLanjut\Minggu 7\PWL_POS(main -> origin)`
	`λ php artisan make:middleware AuthorizeUser`
	
	`INFO  Middleware [C:\laragon\www\Lelyta2AWebLanjut\Minggu 7\PWL_POS\app\Http\Middleware\AuthorizeUser.php] created successfully.`

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

Minggu 7 > PWL_POS > 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, string $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 Saya daftarkan ke app/Http/Kernel.php untuk middleware yang saya buat barusan

```php
    protected $middlewareAliases = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware js 7 praktikum 2
        '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 saya perhatikan tabel m_level yang menjadi tabel untuk menyimpan level/group/role dari user ada

pwl_pos.m_level: 7 rows total (approximately) Next Show all Sorting

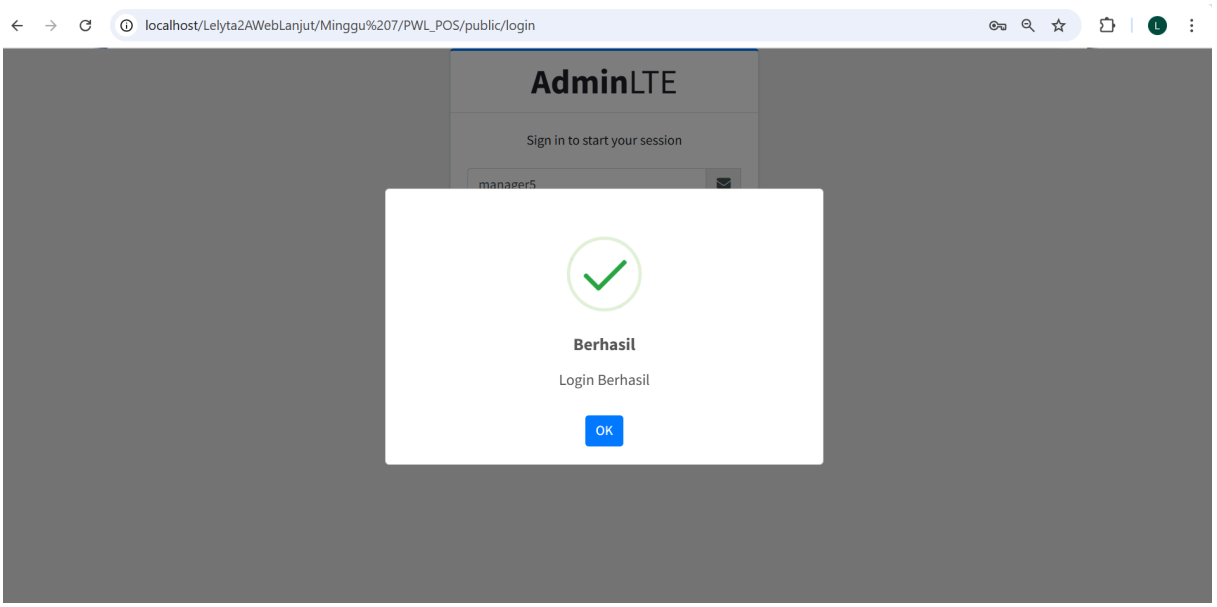
level_id	level_kode	level_nama	created_at	updated_at
1	ADM	Administrator	(NULL)	(NULL)
2	MNG	Manager	(NULL)	(NULL)
3	STF	Staff/Kasir	(NULL)	(NULL)
5	CUS	Pelanggan	2025-02-22 11:20:15	(NULL)
7	AMT	Administrator 2	2025-03-16 04:10:49	2025-03-16 04:10:49
8	STK	Staff Kantor	2025-03-22 12:40:13	2025-03-22 12:40:13
13	PGN	Pelanggan 3	2025-03-23 06:19:47	2025-03-23 06:19:47

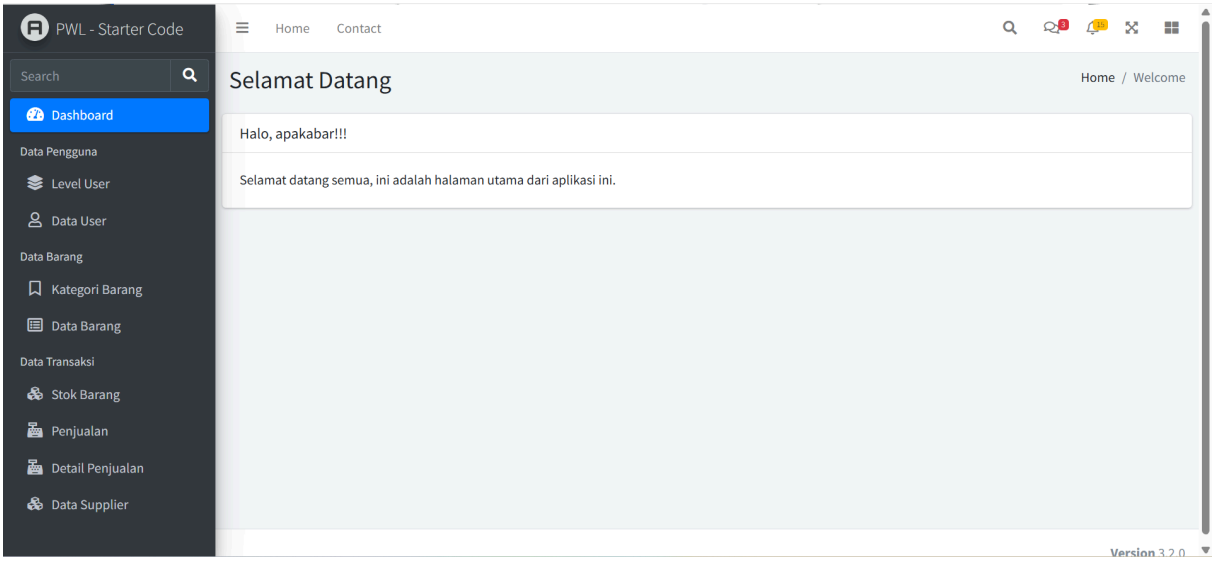
6 Untuk mencoba authorization yang telah dibuat, maka perlu memodiifikasi route/web.php

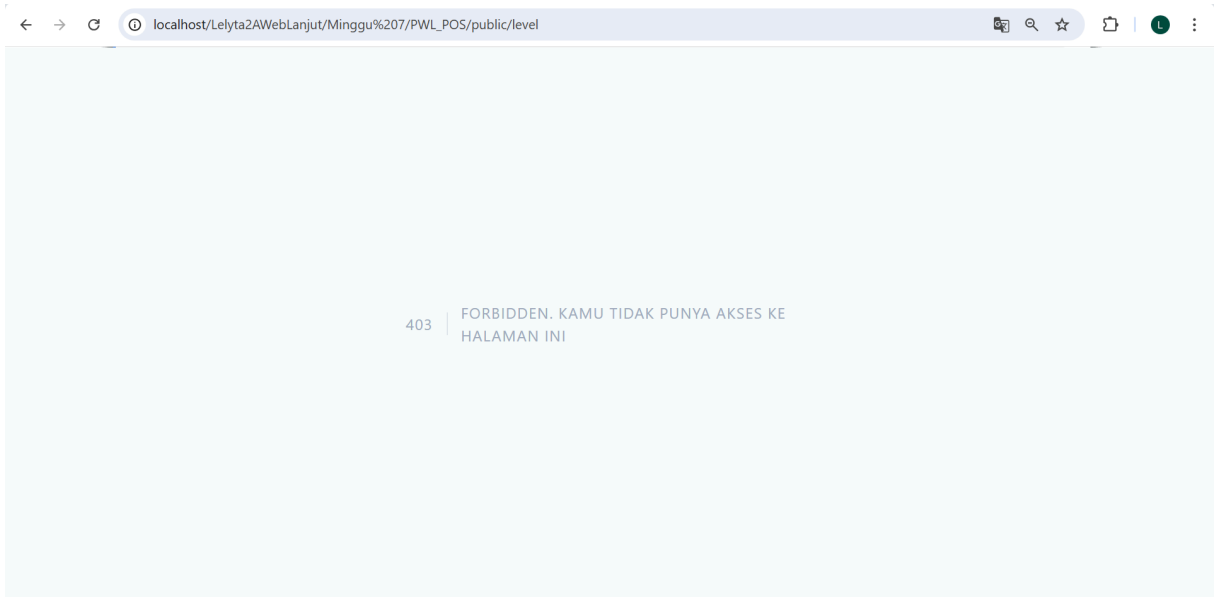
untuk menentukan route mana saja yang akan diberi hak akses sesuai dengan level user

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

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







Keterangan:

Pada langkah ini telah menerapkan kontrol akses berbasis role (role-based access control) menggunakan middleware kustom bernama AuthorizeUser. Middleware ini bertugas memfilter request yang masuk berdasarkan role pengguna yang sedang login. Dalam middleware tersebut, fungsi handle menerima objek request dan role yang diizinkan untuk mengakses route tertentu. Melalui $request->user(), data user yang sedang login diambil, lalu dicek apakah user tersebut memiliki role yang sesuai menggunakan method hasRole() dari UserModel. Jika sesuai, maka request dilanjutkan, jika tidak sistem akan memanggil fungsi abort(403) dan menampilkan pesan error “Forbidden. Kamu tidak punya akses ke halaman ini”.

Method hasRole() sendiri memanfaatkan relasi model UserModel dengan LevelModel melalui fungsi level() dan melakukan pembandingan kode level (level_kode) terhadap role yang diharapkan. Dengan ini, setiap user hanya bisa mengakses resource yang sesuai dengan perannya di sistem. Penerapan middleware ini dilakukan pada route group tertentu, contohnya Route::middleware(['authorize:ADM']), yang hanya dapat diakses oleh user dengan role ADM (Administrator). Ini memastikan bahwa hanya administrator yang dapat mengakses fitur manajemen level, seperti menambahkan, mengedit, atau menghapus data level. Pendekatan ini memperkuat keamanan dan kontrol akses aplikasi, serta memisahkan dengan jelas antara hak akses masing-masing pengguna.

Tugas 2 – Implementasi Authorization

Pada praktikum ini, telah diterapkan proses otorisasi menggunakan middleware yang berfungsi untuk membatasi akses ke route tertentu sesuai dengan level pengguna. Middleware ini memastikan bahwa hanya pengguna dengan level tertentu, dalam hal ini “ADM” atau Administrator, yang dapat mengakses menu level. Jika pengguna yang login bukan

	merupakan admin, maka sistem secara otomatis akan menolak akses tersebut dan menampilkan pesan error yang menandakan bahwa pengguna tidak memiliki hak akses ke halaman tersebut.
	Praktikum 3 – Implementasi Multi-Level Authorizaton di Laravel dengan Middleware
1	Saya modifikasi UserModel.php untuk mendapatkan level_kode dari user yang sudah login. Jadi saya buat fungsi dengan nama getRole()
2	Selanjutnya, saya modifikasi middleware AuthorizeUser.php dengan kode berikut

```php
    // jobsheet 7

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

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

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

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

}
```

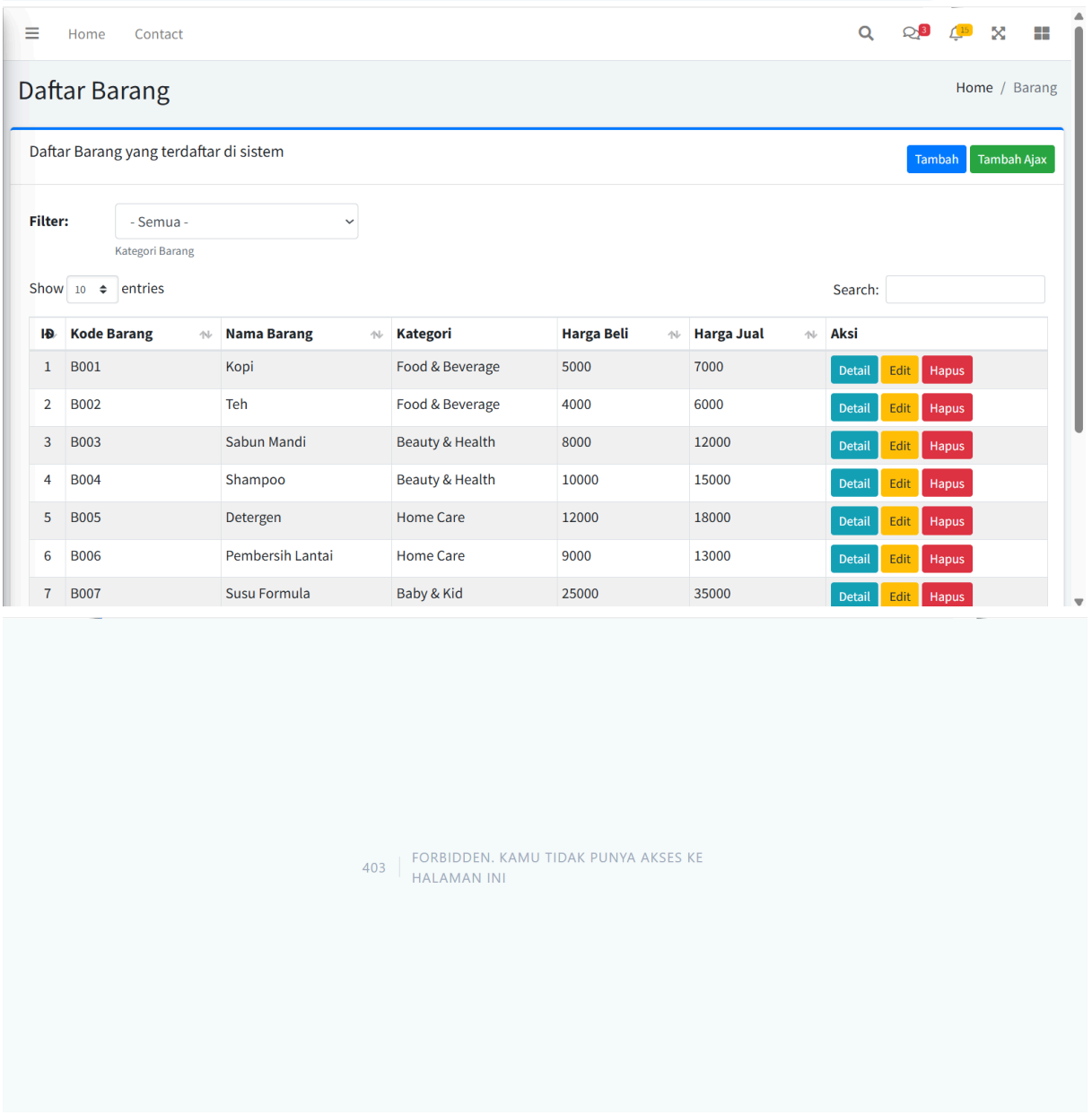
Minggu 7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php

```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, ... $roles): Response
    {
        $user_role = $request->user()->getRole();  // ambil data user yg login
        if (in_array($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 saya perbaiki route/web.php sesuaikan dengan role/level yang diinginkan.

```php
// route Barang
// 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 saya sudah bisa memberikan hak akses menu/route ke beberapa level user



Keterangan:

Dalam praktikum ini, proses otorisasi dilakukan dengan menggunakan middleware yang dirancang untuk memeriksa level akses pengguna sebelum mengizinkan akses ke route tertentu. Middleware AuthorizeUser akan mengambil kode role dari pengguna yang sedang login, kemudian memeriksa apakah role tersebut termasuk dalam daftar role yang diizinkan, seperti “ADM” (Admin) atau “MNG” (Manager). Jika role pengguna sesuai, maka akses akan diberikan. Namun, apabila role pengguna tidak terdaftar dalam daftar tersebut, maka sistem akan menolak akses dengan menampilkan error 403 dan pesan yang menyatakan bahwa pengguna tidak memiliki izin untuk membuka halaman tersebut. Contoh penerapan middleware ini ditunjukkan pada route menu *barang*, yang hanya dapat diakses oleh pengguna dengan level role ADM atau MNG.

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
https://jti.polinema.ac.id

Tugas 3 – Implementasi Multi-Level Authorization

Berikut adalah kode saat saya menambahkan dan memodifikasi route untuk implementasikan praktikum 3 ke menu-menu yang saya punya pada project:

```php
// jobsheet 7
Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter (id), maka harus berupa angka

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

    // route Level
    Route::group(['prefix' => 'level', 'middleware' => ['authorize:ADM']], function () {
        Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal level
        Route::post('/list', [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
        Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level
        Route::get('/create_ajax', [LevelController::class,'create_ajax']); // menampilkan halaman form tambah level ajax
        Route::post('/ajax', [LevelController::class,'store_ajax']); // menyimpan data level ajax
        Route::get('/{id}/show_ajax', [LevelController::class,'show_ajax']); // menampilkan detail level ajax
        Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // menampilkan halaman form edit level ajax
        Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // menyimpan perubahan data level ajax
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete level ajax
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // menghapus data level ajax
    });

    // route User
    Route::group(['prefix' => 'user', 'middleware' => ['authorize:ADM,MNG']], 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::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}/show_ajax', [UserController::class,'show_ajax']); // menampilkan detail user 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']); // menampilkan form konfirmasi delete user ajax
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // menghapus data user ajax
    });

    // route Kategori
    Route::group(['prefix' => 'kategori', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
        Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal kategori
        Route::post('/list', [KategoriController::class, 'list']); // menampilkan data kategori dalam bentuk json untuk datatables
        Route::get('/create', [KategoriController::class, 'create']); // menampilkan halaman form tambah kategori
        Route::get('/create_ajax', [KategoriController::class, 'create_ajax']); // menampilkan halaman form tambah kategori ajax
        Route::post('/ajax', [KategoriController::class, 'store_ajax']); // menyimpan data kategori baru ajax
        Route::get('/{id}/show_ajax', [KategoriController::class,'show_ajax']); // menampilkan detail kategori ajax
        Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']); // menampilkan halaman form edit kategori ajax
        Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']); // menyimpan perubahan data kategori ajax
        Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete kategori ajax
        Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']); // menghapus data kategori ajax
    });
```

```php
    // route Barang
    Route::group(['prefix' => 'barang', 'middleware' => ['authorize:ADM,MNG,CUS,STF']], function () {
        Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal barang
        Route::post('/list', [BarangController::class, 'list']); // menampilkan data barang dalam bentuk json untuk datatables
        Route::get('/create', [BarangController::class, 'create']); // menampilkan halaman form tambah barang
        Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // menampilkan halaman form tambah barang ajax
        Route::post('/ajax', [BarangController::class, 'store_ajax']); // menyimpan data barang baru ajax
        Route::get('/{id}/show_ajax', [BarangController::class,'show_ajax']); // menampilkan detail barang ajax
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // menampilkan halaman form edit barang ajax
        Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // menyimpan perubahan data barang ajax
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete barang ajax
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // menghapus data barang ajax
    });

    // route Supplier
    Route::group(['prefix' => 'supplier', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
        Route::get('/', [SupplierController::class, 'index']); // menampilkan halaman awal supplier
        Route::post('/list', [SupplierController::class, 'list']); // menampilkan data supplier dalam bentuk json untuk datatables
        Route::get('/create', [SupplierController::class, 'create']); // menampilkan halaman form tambah supplier
        Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); // menampilkan halaman form tambah supplier ajax
        Route::post('/ajax', [SupplierController::class, 'store_ajax']); // menyimpan data supplier baru ajax
        Route::get('/{id}/show_ajax', [SupplierController::class,'show_ajax']); // menampilkan detail supplier ajax
        Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); // menampilkan halaman form edit supplier ajax
        Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']); // menyimpan perubahan data supplier ajax
        Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete supplier ajax
        Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']); // menghapus data supplier ajax
    });

    // route Stok
    Route::group(['prefix' => 'stok', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
        Route::get('/', [StokController::class, 'index']); // menampilkan halaman awal stok
        Route::post('/list', [StokController::class, 'list']); // menampilkan data stok dalam bentuk json untuk datatables
        Route::get('/create', [StokController::class, 'create']); // menampilkan halaman form tambah stok
        Route::get('/create_ajax', [StokController::class, 'create_ajax']); // menampilkan halaman form tambah stok ajax
        Route::post('/ajax', [StokController::class, 'store_ajax']); // menyimpan data stok baru ajax
        Route::get('/{id}/show_ajax', [StokController::class,'show_ajax']); // menampilkan detail stok ajax
        Route::get('/{id}/edit_ajax', [StokController::class, 'edit_ajax']); // menampilkan halaman form edit stok ajax
        Route::put('/{id}/update_ajax', [StokController::class, 'update_ajax']); // menyimpan perubahan data stok ajax
        Route::get('/{id}/delete_ajax', [StokController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete stok ajax
        Route::delete('/{id}/delete_ajax', [StokController::class, 'delete_ajax']); // menghapus data stok ajax
    });
```

```php
    // route Penjualan
    Route::group(['prefix' => 'penjualan', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
        Route::get('/', [PenjualanController::class, 'index']); // menampilkan halaman awal penjualan
        Route::post('/list', [PenjualanController::class, 'list']); // menampilkan data penjualan dalam bentuk json untuk datatables
        Route::get('/create', [PenjualanController::class, 'create']); // menampilkan halaman form tambah penjualan
        Route::get('/create_ajax', [PenjualanController::class, 'create_ajax']); // menampilkan halaman form tambah penjualan ajax
        Route::post('/ajax', [PenjualanController::class, 'store_ajax']); // menyimpan data penjualan baru ajax
        Route::get('/{id}/show_ajax', [PenjualanController::class,'show_ajax']); // menampilkan detail penjualan ajax
        Route::get('/{id}/edit_ajax', [PenjualanController::class, 'edit_ajax']); // menampilkan halaman form edit penjualan ajax
        Route::put('/{id}/update_ajax', [PenjualanController::class, 'update_ajax']); // menyimpan perubahan data penjualan ajax
        Route::get('/{id}/delete_ajax', [PenjualanController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete penjualan ajax
        Route::delete('/{id}/delete_ajax', [PenjualanController::class, 'delete_ajax']); // menghapus data penjualan ajax
    });

    // route Detail Penjualan
    Route::group(['prefix' => 'penjualan_detail', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
        Route::get('/', [DetailPenjualanController::class, 'index']); // menampilkan halaman awal detail penjualan
        Route::post('/list', [DetailPenjualanController::class, 'list']); // menampilkan data detail penjualan dalam bentuk json untuk datatables
        Route::get('/create', [DetailPenjualanController::class, 'create']); // menampilkan halaman form tambah detail penjualan
        Route::get('/create_ajax', [DetailPenjualanController::class, 'create_ajax']); // menampilkan halaman form tambah detail penjualan ajax
        Route::post('/ajax', [DetailPenjualanController::class, 'store_ajax']); // menyimpan data detail penjualan baru ajax
        Route::get('/{id}/show_ajax', [DetailPenjualanController::class,'show_ajax']); // menampilkan detail detail penjualan ajax
        Route::get('/{id}/edit_ajax', [DetailPenjualanController::class, 'edit_ajax']); // menampilkan halaman form edit detail penjualan ajax
        Route::put('/{id}/update_ajax', [DetailPenjualanController::class, 'update_ajax']); // menyimpan perubahan data detail penjualan ajax
        Route::get('/{id}/delete_ajax', [DetailPenjualanController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete detail penjualan ajax
        Route::delete('/{id}/delete_ajax', [DetailPenjualanController::class, 'delete_ajax']); // menghapus data detail penjualan ajax
    });
});
```

Dalam sistem ini, otorisasi akses ke setiap route diatur menggunakan middleware AuthorizeUser yang memverifikasi role (level) pengguna sebelum mengizinkan akses. Setiap menu seperti user, level, kategori, supplier, barang, stok, penjualan, dan detail penjualan dibatasi hak aksesnya berdasarkan role tertentu yang telah ditentukan dalam middleware route. Sebagai contoh, route penjualan dan detail penjualan hanya dapat diakses oleh user dengan level role “ADM”, “MNG”, atau “STF”. Hal ini dilakukan dengan cara menetapkan middleware authorize:ADM,MNG,STF pada setiap group route. Ketika pengguna dengan role yang tidak sesuai mencoba mengakses route tersebut, sistem akan menolak permintaan tersebut dan menampilkan pesan error 403 “Forbidden. Kamu tidak punya akses ke halaman ini”. Dengan penerapan ini, sistem menjadi lebih aman dan setiap pengguna hanya dapat mengakses menu sesuai hak akses yang telah diberikan sesuai perannya.

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
https://jti.polinema.ac.id

Tugas 4 – Implementasi Form Registrasi :

Berikut adalah langkah-langkah saat saya mengimplementasikan register:

a. Saya menambahkan kode di AuthController

```php
    public function register() {
        return view('auth.register');
    }

    public function postregister(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'username' => 'required|string|min:3|unique:m_user,username',
            'nama' => 'required|string|min:3|max:100',
            'password' => 'required|string|min:6'
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => false,
                'message' => 'Validasi gagal!',
                'msgField' => $validator->errors()
            ]);
        }

        $user = new UserModel();
        $user->username = $request->username;
        $user->nama = $request->nama;
        $user->password = Hash::make($request->password);
        $user->level_id = 3; // Level default (staff)

        $user->save();

        return response()->json([
            'status' => true,
            'message' => 'Registrasi berhasil! Silakan login.',
            'redirect' => url('login')
        ]);
    }
}
```

b. Kemudian saya membuat kode untuk file baru, yaitu register.blade.php

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
https://jti.polinema.ac.id

Minggu 7 > PWL_POS > resources > views > auth > register.blade.php

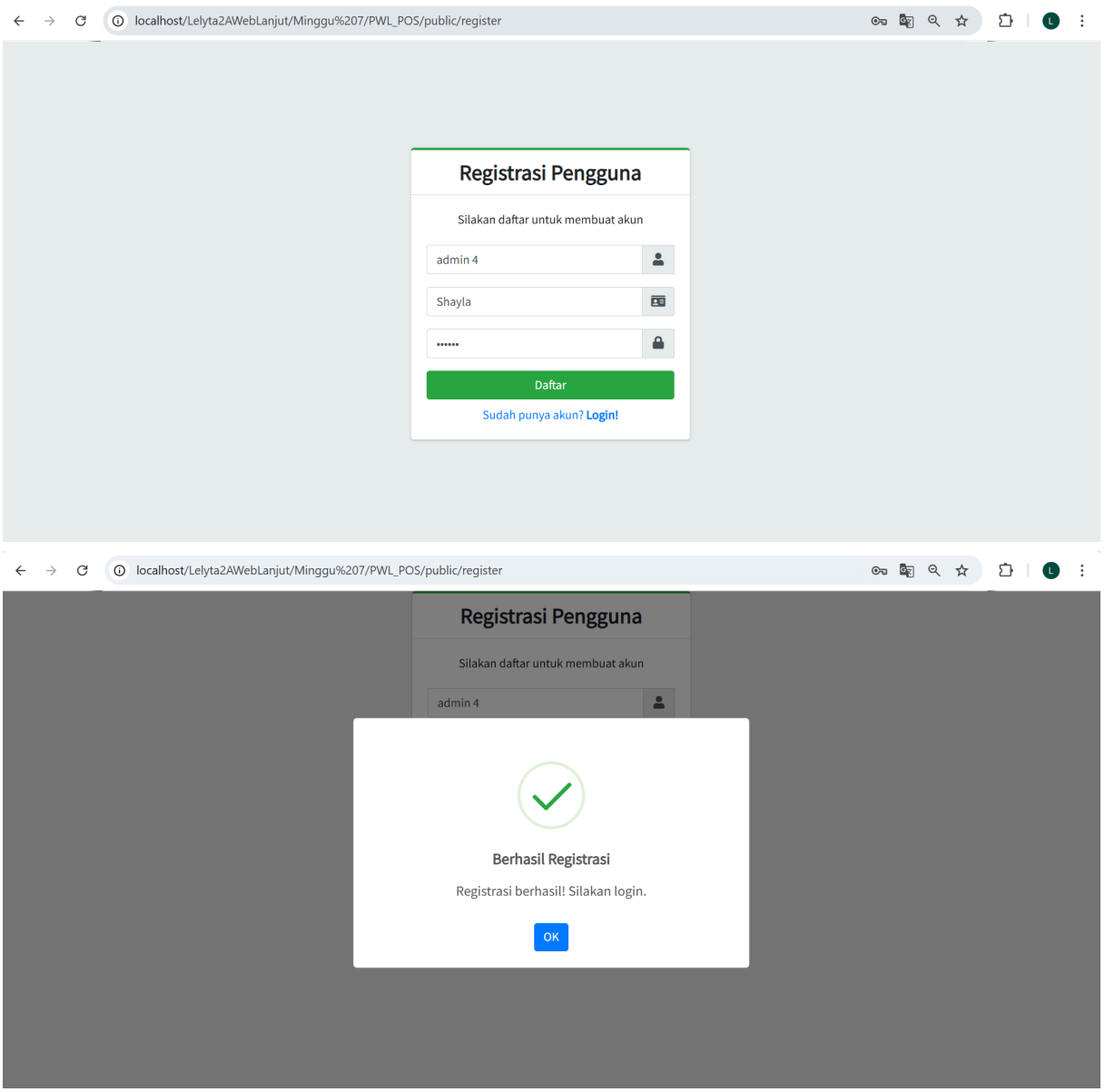
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Registrasi Pengguna</title>
    <!-- Google Font -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro">
    <!-- Font Awesome -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
    <!-- icheck bootstrap -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
    <!-- SweetAlert2 -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
    <!-- AdminLTE -->
    <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-success">
            <div class="card-header text-center">
                <a href="{{ url('/') }}" class="h3"><b>Registrasi Pengguna</b></a>
            </div>
            <div class="card-body">
                <p class="login-box-msg">Silakan daftar untuk membuat akun</p>
                <form action="{{ url('register') }}" method="POST" id="form-register">
                    @csrf
                    <div class="input-group mb-3">
                        <input id="username" type="text" name="username" class="form-control" placeholder="Username">
                        <div class="input-group-append">
                            <div class="input-group-text d-flex align-items-center justify-content-center" style="width: 42px;">
                                <i class="fas fa-user"></i>
                            </div>
                        </div>
                    </div>

                    <div class="input-group mb-3">
                        <input id="nama" type="text" name="nama" class="form-control" placeholder="Nama">
                        <div class="input-group-append">
                            <div class="input-group-text d-flex align-items-center justify-content-center" style="width: 42px;">
                                <i class="fas fa-id-card"></i>
                            </div>
                        </div>
                    </div>
```

```
                    <div class="input-group mb-3">
                        <input id="password" type="password" name="password" class="form-control" placeholder="Password">
                        <div class="input-group-append">
                            <div class="input-group-text d-flex align-items-center justify-content-center" style="width: 42px;">
                                <i class="fas fa-lock"></i>
                            </div>
                        </div>
                    </div>

                    <div class="row">
                        <div class="col-12 mb-2">
                            <button type="submit" class="btn btn-success btn-block">Daftar</button>
                        </div>
                        <div class="col-12 text-center">
                            <a href="{{ url('login') }}">Sudah punya akun? <strong>Login!</strong></a>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <!-- Scripts -->
    <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
    <script>
        $.ajaxSetup({
            headers: {
                'X-CSRF-TOKEN': $('input[name="_token"]').val()
            }
        });

        $(document).ready(function () {
            $("#form-register").validate({
                rules: {
                    username: { required: true, minlength: 4, maxlength: 20 },
                    nama: { required: true, minlength: 3, maxlength: 100 },
                    password: { required: true, minlength: 6, maxlength: 20 },
                },
                submitHandler: function (form) {
```

```
                $.ajax({
                    url: form.action,
                    method: form.method,
                    data: $(form).serialize(),
                    success: function (response) {
                        $('.error-text').text('');
                        if (response.status) {
                            Swal.fire({
                                icon: 'success',
                                title: 'Berhasil Registrasi',
                                text: response.message
                            }).then(() => {
                                window.location.href = "{{ url('login') }}";
                            });
                        } else {
                            if (response.msgField) {
                                $.each(response.msgField, function (prefix, val) {
                                    $('#error-' + prefix).text(val[0]);
                                });
                            }
                            Swal.fire({
                                icon: 'error',
                                title: 'Gagal Registrasi',
                                text: response.message || 'Silakan periksa kembali data yang diisi.'
                            });
                        }
                    },
                    error: function (xhr) {
                        Swal.fire({
                            icon: 'error',
                            title: 'Error',
                            text: 'Terjadi kesalahan di server.'
                        });
                    }
                });
                return false;
            },
            errorElement: 'span',
            errorPlacement: function (error, element) {
                error.addClass('invalid-feedback');
                element.closest('.input-group').append(error);
            },
            highlight: function (element) {
                $(element).addClass('is-invalid');
            },
            unhighlight: function (element) {
                $(element).removeClass('is-invalid');
            }
        });
    });
</script>
</body>
</html>
```

c. Kemudian saya menambahkan di routenya:

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



Fitur registrasi pada sistem ini dirancang agar pengguna baru dapat mendaftar akun secara mandiri. Saat pengguna mengakses route register, sistem akan menampilkan halaman formulir registrasi. Formulir ini meminta data berupa username, nama lengkap, dan password. Seluruh input akan divalidasi di sisi klien menggunakan plugin jQuery Validation, dan juga di sisi server menggunakan validator Laravel.

Jika validasi berhasil, data akan dikirim melalui AJAX ke route POST /register dan disimpan ke dalam database. Password pengguna akan dienkripsi menggunakan fungsi Hash::make() untuk menjaga keamanan data. Selain itu, setiap pengguna baru secara otomatis akan diberikan level default yaitu 3, yang merepresentasikan role Staff (STF). Hal ini dilakukan dengan men-set $user->level_id = 3 sebelum disimpan ke database.

Apabila proses registrasi berhasil, sistem akan menampilkan notifikasi menggunakan SweetAlert2 dan langsung mengarahkan pengguna ke halaman login. Namun, jika terjadi kesalahan validasi atau kesalahan server, pesan error yang sesuai akan ditampilkan kepada

	pengguna. Dengan pendekatan ini, sistem mampu menangani registrasi pengguna secara interaktif dan aman, serta memastikan hanya data valid yang tersimpan ke dalam sistem.