



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

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

Laporan Jobsheet

	Praktikum 1 – Implementasi Authentication :
1	<p>Saya buka kembali project laravel PWL_POS, dan saya modifikasi konfigurasi aplikasi saya di config/auth.php</p> <pre>'providers' => ['users' => ['driver' => 'eloquent', 'model' => App\Models\User::class,],],</pre> <p>Pada bagian ini saya sesuaikan dengan Model untuk tabel m_user yang sudah saya buat</p> <pre>'providers' => ['users' => ['driver' => 'eloquent', 'model' => App\Models\UserModel::class,],],</pre>
2	Selanjutnya saya modifikasi sedikit pada UserModel.php untuk bisa melakukan proses otentikasi



```
Minggu 7 > PWL_POS > app > Models > UserModel.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Illuminate\Foundation\Auth\User as Authenticatable; //Implementasi class Authenticatable
9
10
11 class UserModel extends Authenticatable
12 {
13     use HasFactory;
14
15     protected $table = 'm_user'; //mendefinisikan nama tabel yang digunakan oleh model ini
16     protected $primaryKey = 'user_id'; //mendefinisikan primary key tabel yang digunakan oleh model ini
17
18     protected $fillable = [
19         'level_id',
20         'username',
21         'nama',
22         'password',
23         'created_at',
24         'updated_at'
25     ];
26
27     protected $hidden = ['password'];
28
29     protected $casts = ['password' => 'hashed'];
30
31     public function level():BelongsTo {
32
33         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
34     }
35
36 }
```

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



```
Minggu 7 > PWL_POS > app > Http > Controllers > AuthController.php
1  <?php
2  namespace App\Http\Controllers;
3
4  use Illuminate\Http\Request;
5  use Illuminate\Support\Facades\Auth;
6
7  class AuthController extends Controller
8  {
9      public function login()
10     {
11         if (Auth::check()) { // jika sudah login, maka redirect ke halaman home
12             return redirect('/');
13         }
14         return view('auth.login');
15     }
16     public function postlogin(Request $request)
17     {
18         if ($request->ajax() || $request->wantsJson()) {
19             $validatedData = $request->validate([
20                 'username' => 'required|string',
21                 'password' => 'required|string',
22             ]);
23
24             $credentials = $request->only('username', 'password');
25             if (Auth::attempt($credentials)) {
26                 $request->session()->regenerate();
27                 return response()->json([
28                     'status' => true,
29                     'message' => 'Login Berhasil',
30                     'redirect' => url('/')
31                 ]);
32             }
33             return response()->json([
34                 'status' => false,
35                 'message' => 'Login Gagal'
36             ]);
37         }
38         return redirect('login');
39     }
40     public function logout(Request $request)
41     {
42         Auth::logout();
43         $request->session()->invalidate();
44         $request->session()->regenerateToken();
45         return redirect('login');
46     }
47 }
48
```

- 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



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

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>Login Pengguna</title>
7 <!-- Google Font: Source Sans Pro -->
8 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
9 <!-- Font Awesome -->
10 <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
11 <!-- icheck bootstrap -->
12 <link rel="stylesheet" href="{{ asset('adminlte/plugins/ichack-bootstrap/ichack-bootstrap.min.css') }}">
13 <!-- SweetAlert2 -->
14 <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
15 <!-- Theme style -->
16 <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
17 </head>
18
19 <body class="hold-transition login-page">
20 <div class="login-box">
21 <!-- /.login-logo -->
22 <div class="card card-outline card-primary">
23 <div class="card-header text-center"><a href="{{ url('/') }}" class="h1"><b>Admin</b>LTE</a></div>
24 <div class="card-body">
25 <p class="login-box-msg">Sign in to start your session</p>
26 <form action="{{ url('login') }}" method="POST" id="form-login">
27 @csrf
28 <div class="input-group mb-3">
29 <input type="text" id="username" name="username" class="form-control" placeholder="Username">
30 <div class="input-group-append">
31 <div class="input-group-text">
32 <span class="fas fa-envelope"></span>
33 </div>
34 </div>
35 <small id="error-username" class="error-text text-danger"></small>
36 </div>
37 <div class="input-group mb-3">
38 <input type="password" id="password" name="password" class="form-control" placeholder="Password">
39 <div class="input-group-append">
40 <div class="input-group-text">
41 <span class="fas fa-lock"></span>
42 </div>
43 </div>
44 <small id="error-password" class="error-text text-danger"></small>
45 </div>
46 <div class="mb-3 text-right">
47 <small>
48 <a href="{{ url('register') }}" class="text-primary">Belum punya akun? Daftar sekarang!</a>
49 </small>
50 </div>
51 <div class="row">
52 <div class="col-8">
53 <div class="input-group">
54 <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
55 </div>
56 </div>
57 <!-- /.col -->
58 <div class="col-4">
59 <button type="submit" class="btn btn-primary btn-block">Sign In</button>
60 </div>
61 <!-- /.col -->
62 </div>
63 </form>
64 </div>
65 <!-- /.card-body -->
66 </div>
67 <!-- /.card -->
68 </div>
69 <!-- /.login-box -->
70 <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
71 <!-- Bootstrap 4 -->
72 <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
73 <!-- jquery-validation -->
74 <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
75 <script src="{{ asset('adminlte/plugins/jquery-validation/jquery-validation/additional-methods.min.js') }}"></script>
76 <!-- SweetAlert2 -->
77 <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
78 <!-- AdminLTE App -->
79 <script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
80
81 <script>
82 $.ajaxSetup({
83 headers: {
84 'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
85 },
86 error: function(xhr, status, error) {sedikti tambahan debag gedebag
87 console.error('AJAX Error: ' + status + error);
```



```
88     console.error(xhr.responseText);
89     Swal.fire({
90         icon: 'error',
91         title: 'Server Error',
92         text: 'An error occurred on the server. Please try again later.'
93     });
94 }
95 });
96
97 $(document).ready(function () {
98     $("#form-login").validate({
99         rules: {
100             username: { required: true, minlength: 4, maxlength: 20 },
101             password: { required: true, minlength: 6, maxlength: 20 }
102         },
103         submitHandler: function (form) { // ketika valid, maka bagian yg akan dijalankan
104             $.ajax({
105                 url: form.action,
106                 type: form.method,
107                 data: $(form).serialize(),
108                 success: function (response) {
109                     if (response.status) { // jika sukses
110                         Swal.fire({
111                             icon: 'success',
112                             title: 'Berhasil',
113                             text: response.message,
114                         }).then(function () {
115                             window.location = response.redirect;
116                         });
117                     } else { // jika error
118                         $('.error-text').text('');
119                         $.each(response.msgField, function (prefix, val) {
120                             $('#error-' + prefix).text(val[0]);
121                         });
122                         Swal.fire({
123                             icon: 'error',
124                             title: 'Terjadi Kesalahan',
125                             text: response.message
126                         });
127                     }
128                 }
129             });
130             return false;
131         },
132         errorPlacement: function (error, element) {
133             error.addClass('invalid-feedback');
134             element.closest('.input-group').append(error);
135         },
136         highlight: function (element, errorClass, validClass) {
137             $(element).addClass('is-invalid');
138         },
139         unhighlight: function (element, errorClass, validClass) {
140             $(element).removeClass('is-invalid');
141         }
142     });
143 });
144 </script>
145 </body>
146 </html>
```

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

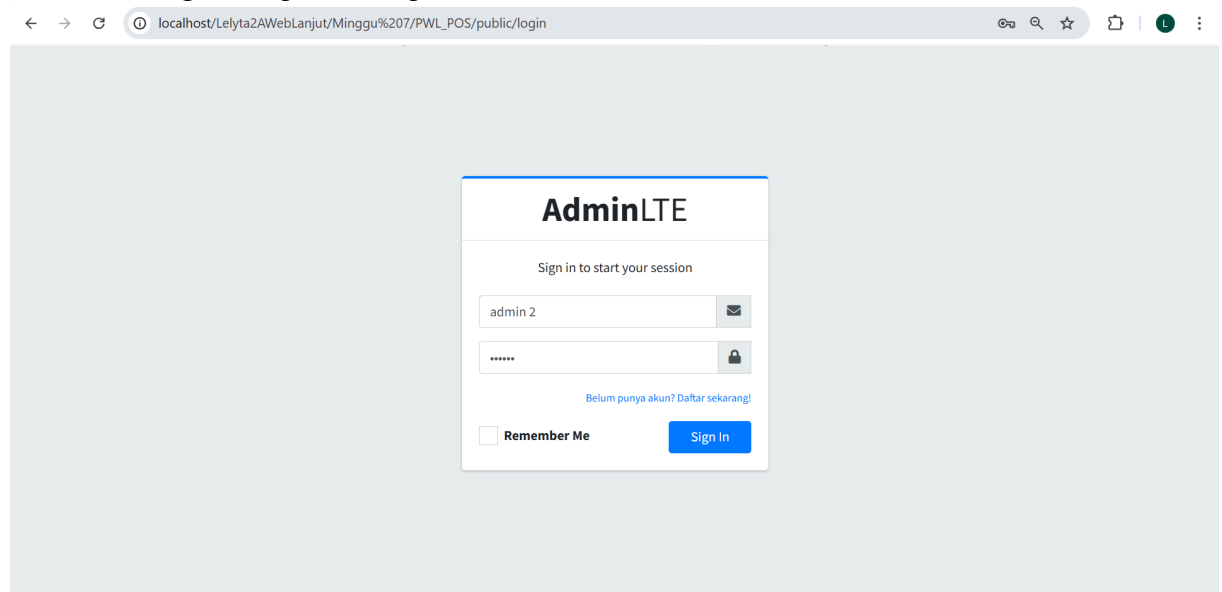


```
// jobsheet 7
Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter (id), maka harus berupa angka

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:

- 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://iti.polinema.ac.id>

← → ↻ ⓘ localhost/Lelyta2AWebLanjut/Minggu%207/PWL_POS/public/login 🔍 ☆ 📄 🌐 ⋮

AdminLTE

Sign in to start your session

[Belum punya akun? Daftar sekarang!](#)

☐ Remember Me

Berhasil

Login Berhasil



The screenshot displays a web application interface. The browser's address bar shows the URL: `localhost/Lelyta2AWebLanjut/Minggu%207/PWL_POS/public/`. The application has a sidebar on the left with a search bar and a list of menu items: Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Barang, Data Transaksi, Stok Barang, Penjualan, Detail Penjualan, Data Supplier, and Logout. The main content area shows a 'Selamat Datang' (Welcome) message with a greeting and a welcome message. A modal dialog is open in the center of the screen, featuring a yellow exclamation mark icon and the text 'Logout? Apakah kamu yakin ingin keluar?' (Logout? Are you sure you want to leave?). Below the text are two buttons: 'Ya, Logout' (Yes, Logout) and 'Batal' (Cancel). The version number 'Version 3.2.0' is visible in the bottom right corner of the application.

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 `` 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
11 class UserModel extends Authenticatable
12 {
13     use HasFactory;
14
15     protected $table = 'm_user'; //mendefinisikan nama tabel yang digunakan oleh model ini
16     protected $primaryKey = 'user_id'; //mendefinisikan primary key tabel yang digunakan oleh model ini
17
18     protected $fillable = [
19         'level_id',
20         'username',
21         'nama',
22         'password',
23         'created_at',
24         'updated_at'
25     ];
26
27     protected $hidden = ['password'];
28
29     protected $casts = ['password' => 'hashed'];
30
31     //public function level():BelongsTo {
32     //    //return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
33     //}
34
35     // jobsheet 7
36
37     // Relasi ke tabel level
38     public function level(): BelongsTo
39     {
40         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
41     }
42
43     // Mendapatkan nama role
44     public function getRoleName(): string
45     {
46         return $this->level->level_nama;
47     }
48
49     // Cek apakah user memiliki role tertentu
50     public function hasRole($role): bool
51     {
52         return $this->level->level_kode == $role;
53     }
54 }
```

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 yang sesuai
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
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthorizeUser
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
15      */
16     public function handle(Request $request, Closure $next, string $role = ''): Response
17     {
18         $user = $request->user(); // ambil data user yg login
19         // fungsi user() diambil dari UserModel.php
20         if ($user->hasRole($role)) { // cek apakah user punya role yg diinginkan
21             return $next($request);
22         }
23
24         // jika tidak punya role, maka tampilkan error 403
25         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
26     }
27 }
```

- 4 Saya daftarkan ke app/Http/Kernel.php untuk middleware yang saya buat barusan

```
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 memodifikasi route/web.php



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

```
// 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

The screenshot shows a web browser window with the URL `localhost/Lelyta2AWebLanjut/Minggu%207/PWL_POS/public/login`. The page displays a login form for 'AdminLTE' with the text 'Sign in to start your session'. Below the form, a modal box shows a green checkmark and the text 'Berhasil Login Berhasil' with an 'OK' button. Below the login form, the dashboard is visible with a sidebar menu containing items like 'Dashboard', 'Data Pengguna', 'Level User', 'Data User', 'Data Barang', 'Kategori Barang', 'Data Transaksi', 'Stok Barang', 'Penjualan', 'Detail Penjualan', and 'Data Supplier'. The main content area shows a welcome message 'Selamat Datang' and 'Halo, apakabar!!!'.



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 perbandingan 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()

```
// 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;
}
}
```

- 2 Selanjutnya, saya modifikasi middleware AuthorizeUser.php dengan kode berikut



	<pre>Minggu 7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php 1 <?php 2 3 namespace App\Http\Middleware; 4 5 use Closure; 6 use Illuminate\Http\Request; 7 use Symfony\Component\HttpFoundation\Response; 8 9 class AuthorizeUser 10 { 11 /** 12 * Handle an incoming request. 13 * 14 * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) \$next 15 */ 16 public function handle(Request \$request, Closure \$next, ... \$roles): Response 17 { 18 \$user_role = \$request->user()->getRole(); // ambil data user yg login 19 if (in_array(\$user_role, \$roles)) { // cek apakah level_kode user ada di dalam array roles 20 return \$next(\$request); // jika ada, maka lanjutkan request 21 } 22 23 // jika tidak punya role, maka tampilkan error 403 24 abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini'); 25 } 26 }</pre>
3	<p>Setelah itu tinggal saya perbaiki route/web.php sesuaikan dengan role/level yang diinginkan.</p> <pre>// 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 });</pre>
4	<p>Sekarang saya sudah bisa memberikan hak akses menu/route ke beberapa level user</p>



HomeContact

Daftar Barang

Home / Barang

Daftar Barang yang terdaftar di sistem

TambahTambah Ajax

Filter: - Semua -

Kategori Barang

Show 10 entries

Search:

ID	Kode Barang	Nama Barang	Kategori	Harga Beli	Harga Jual	Aksi
1	B001	Kopi	Food & Beverage	5000	7000	Detail Edit Hapus
2	B002	Teh	Food & Beverage	4000	6000	Detail Edit Hapus
3	B003	Sabun Mandi	Beauty & Health	8000	12000	Detail Edit Hapus
4	B004	Shampoo	Beauty & Health	10000	15000	Detail Edit Hapus
5	B005	Detergen	Home Care	12000	18000	Detail Edit Hapus
6	B006	Pembersih Lantai	Home Care	9000	13000	Detail Edit Hapus
7	B007	Susu Formula	Baby & Kid	25000	35000	Detail Edit Hapus

403 | FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI

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.



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:

```
187 // jobsheet 7
188 Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter (id), maka harus berupa angka
189
190 Route::get('login', [AuthController::class, 'login'])->name('login');
191 Route::post('login', [AuthController::class, 'postlogin']);
192 Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
193
194 Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini harus login dulu
195
196     Route::get('/', [WelcomeController::class, 'index']);
197
198     // route Level
199     Route::group(['prefix' => 'level', 'middleware' => ['authorize:ADM']], function () {
200         Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal level
201         Route::post('/list', [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
202         Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level
203         Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // menampilkan halaman form tambah level ajax
204         Route::post('/ajax', [LevelController::class, 'store_ajax']); // menyimpan data level ajax
205         Route::get('/{id}/show_ajax', [LevelController::class, 'show_ajax']); // menampilkan detail level ajax
206         Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // menampilkan halaman form edit level ajax
207         Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // menyimpan perubahan data level ajax
208         Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete level ajax
209         Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // menghapus data level ajax
210     });
211
212     // route User
213     Route::group(['prefix' => 'user', 'middleware' => ['authorize:ADM,MNG']], function () {
214         Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user
215         Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
216         Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
217         Route::get('/create_ajax', [UserController::class, 'create_ajax']); // menampilkan halaman form tambah user ajax
218         Route::post('/ajax', [UserController::class, 'store_ajax']); // menyimpan data user baru ajax
219         Route::get('/{id}/show_ajax', [UserController::class, 'show_ajax']); // menampilkan detail user ajax
220         Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user ajax
221         Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // menyimpan perubahan data user ajax
222         Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete user ajax
223         Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // menghapus data user ajax
224     });
225
226     // route Kategori
227     Route::group(['prefix' => 'kategori', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
228         Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal kategori
229         Route::post('/list', [KategoriController::class, 'list']); // menampilkan data kategori dalam bentuk json untuk datatables
230         Route::get('/create', [KategoriController::class, 'create']); // menampilkan halaman form tambah kategori
231         Route::get('/create_ajax', [KategoriController::class, 'create_ajax']); // menampilkan halaman form tambah kategori ajax
232         Route::post('/ajax', [KategoriController::class, 'store_ajax']); // menyimpan data kategori baru ajax
233         Route::get('/{id}/show_ajax', [KategoriController::class, 'show_ajax']); // menampilkan detail kategori ajax
234         Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']); // menampilkan halaman form edit kategori ajax
235         Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']); // menyimpan perubahan data kategori ajax
236         Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete kategori ajax
237         Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']); // menghapus data kategori ajax
238     });
239 }
```



```
240 // route Barang
241 Route:group(['prefix' => 'barang', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
242     Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal barang
243     Route::post('/list', [BarangController::class, 'list']); // menampilkan data barang dalam bentuk json untuk datatables
244     Route::get('/create', [BarangController::class, 'create']); // menampilkan halaman form tambah barang
245     Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // menampilkan halaman form tambah barang ajax
246     Route::post('/ajax', [BarangController::class, 'store_ajax']); // menyimpan data barang baru ajax
247     Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax']); // menampilkan detail barang ajax
248     Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // menampilkan halaman form edit barang ajax
249     Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // menyimpan perubahan data barang ajax
250     Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete barang ajax
251     Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // menghapus data barang ajax
252 });
253
254 // route Supplier
255 Route:group(['prefix' => 'supplier', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
256     Route::get('/', [SupplierController::class, 'index']); // menampilkan halaman awal supplier
257     Route::post('/list', [SupplierController::class, 'list']); // menampilkan data supplier dalam bentuk json untuk datatables
258     Route::get('/create', [SupplierController::class, 'create']); // menampilkan halaman form tambah supplier
259     Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); // menampilkan halaman form tambah supplier ajax
260     Route::post('/ajax', [SupplierController::class, 'store_ajax']); // menyimpan data supplier baru ajax
261     Route::get('/{id}/show_ajax', [SupplierController::class, 'show_ajax']); // menampilkan detail supplier ajax
262     Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); // menampilkan halaman form edit supplier ajax
263     Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']); // menyimpan perubahan data supplier ajax
264     Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete supplier ajax
265     Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']); // menghapus data supplier ajax
266 });
267
268 // route Stok
269 Route:group(['prefix' => 'stok', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
270     Route::get('/', [StokController::class, 'index']); // menampilkan halaman awal stok
271     Route::post('/list', [StokController::class, 'list']); // menampilkan data stok dalam bentuk json untuk datatables
272     Route::get('/create', [StokController::class, 'create']); // menampilkan halaman form tambah stok
273     Route::get('/create_ajax', [StokController::class, 'create_ajax']); // menampilkan halaman form tambah stok ajax
274     Route::post('/ajax', [StokController::class, 'store_ajax']); // menyimpan data stok baru ajax
275     Route::get('/{id}/show_ajax', [StokController::class, 'show_ajax']); // menampilkan detail stok ajax
276     Route::get('/{id}/edit_ajax', [StokController::class, 'edit_ajax']); // menampilkan halaman form edit stok ajax
277     Route::put('/{id}/update_ajax', [StokController::class, 'update_ajax']); // menyimpan perubahan data stok ajax
278     Route::get('/{id}/delete_ajax', [StokController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete stok ajax
279     Route::delete('/{id}/delete_ajax', [StokController::class, 'delete_ajax']); // menghapus data stok ajax
280 });
281
282 // route Penjualan
283 Route:group(['prefix' => 'penjualan', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
284     Route::get('/', [PenjualanController::class, 'index']); // menampilkan halaman awal penjualan
285     Route::post('/list', [PenjualanController::class, 'list']); // menampilkan data penjualan dalam bentuk json untuk datatables
286     Route::get('/create', [PenjualanController::class, 'create']); // menampilkan halaman form tambah penjualan
287     Route::get('/create_ajax', [PenjualanController::class, 'create_ajax']); // menampilkan halaman form tambah penjualan ajax
288     Route::post('/ajax', [PenjualanController::class, 'store_ajax']); // menyimpan data penjualan baru ajax
289     Route::get('/{id}/show_ajax', [PenjualanController::class, 'show_ajax']); // menampilkan detail penjualan ajax
290     Route::get('/{id}/edit_ajax', [PenjualanController::class, 'edit_ajax']); // menampilkan halaman form edit penjualan ajax
291     Route::put('/{id}/update_ajax', [PenjualanController::class, 'update_ajax']); // menyimpan perubahan data penjualan ajax
292     Route::get('/{id}/delete_ajax', [PenjualanController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete penjualan ajax
293     Route::delete('/{id}/delete_ajax', [PenjualanController::class, 'delete_ajax']); // menghapus data penjualan ajax
294 });
295
296 // route Detail Penjualan
297 Route:group(['prefix' => 'penjualan_detail', 'middleware' => ['authorize:ADM,MNG,STF']], function () {
298     Route::get('/', [DetailPenjualanController::class, 'index']); // menampilkan halaman awal detail penjualan
299     Route::post('/list', [DetailPenjualanController::class, 'list']); // menampilkan data detail penjualan dalam bentuk json untuk datatables
300     Route::get('/create', [DetailPenjualanController::class, 'create']); // menampilkan halaman form tambah detail penjualan
301     Route::get('/create_ajax', [DetailPenjualanController::class, 'create_ajax']); // menampilkan halaman form tambah detail penjualan ajax
302     Route::post('/ajax', [DetailPenjualanController::class, 'store_ajax']); // menyimpan data detail penjualan baru ajax
303     Route::get('/{id}/show_ajax', [DetailPenjualanController::class, 'show_ajax']); // menampilkan detail detail penjualan ajax
304     Route::get('/{id}/edit_ajax', [DetailPenjualanController::class, 'edit_ajax']); // menampilkan halaman form edit detail penjualan ajax
305     Route::put('/{id}/update_ajax', [DetailPenjualanController::class, 'update_ajax']); // menyimpan perubahan data detail penjualan ajax
306     Route::get('/{id}/delete_ajax', [DetailPenjualanController::class, 'confirm_ajax']); // menampilkan form konfirmasi delete detail penjualan ajax
307     Route::delete('/{id}/delete_ajax', [DetailPenjualanController::class, 'delete_ajax']); // menghapus data detail penjualan ajax
308 });
309 });
```

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.



Tugas 4 – Implementasi Form Registrasi :

Berikut adalah langkah-langkah saat saya mengimplementasikan register:

a. Saya menambahkan kode di AuthController

```
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



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

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Registrasi Pengguna</title>
7     <!-- Google Font -->
8     <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro">
9     <!-- Font Awesome -->
10    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
11    <!-- iCheck bootstrap -->
12    <link rel="stylesheet" href="{{ asset('adminlte/plugins/iCheck-bootstrap/iCheck-bootstrap.min.css') }}">
13    <!-- SweetAlert2 -->
14    <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
15    <!-- AdminLTE -->
16    <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
17 </head>
18 <body class="hold-transition login-page">
19     <div class="login-box">
20         <div class="card card-outline card-success">
21             <div class="card-header text-center">
22                 <a href="{{ url('/') }}" class="h3"><b>Registrasi Pengguna</b></a>
23             </div>
24             <div class="card-body">
25                 <p class="login-box-msg">Silakan daftar untuk membuat akun</p>
26                 <form action="{{ url('register') }}" method="POST" id="form-register">
27                     @csrf
28                     <div class="input-group mb-3">
29                         <input id="username" type="text" name="username" class="form-control" placeholder="Username">
30                         <div class="input-group-append">
31                             <div class="input-group-text d-flex align-items-center justify-content-center" style="width: 42px;">
32                                 <i class="fas fa-user"></i>
33                             </div>
34                         </div>
35                     </div>
36                     <div class="input-group mb-3">
37                         <input id="nama" type="text" name="nama" class="form-control" placeholder="Nama">
38                         <div class="input-group-append">
39                             <div class="input-group-text d-flex align-items-center justify-content-center" style="width: 42px;">
40                                 <i class="fas fa-id-card"></i>
41                             </div>
42                         </div>
43                     </div>
44                 </form>
45             </div>
46         </div>
47     </div>
```



```
45
46
47     <div class="input-group mb-3">
48         <input id="password" type="password" name="password" class="form-control" placeholder="Password">
49         <div class="input-group-append">
50             <div class="input-group-text d-flex align-items-center justify-content-center" style="width: 42px;">
51                 <i class="fas fa-lock"></i>
52             </div>
53         </div>
54     </div>
55
56     <div class="row">
57         <div class="col-12 mb-2">
58             <button type="submit" class="btn btn-success btn-block">Daftar</button>
59         </div>
60         <div class="col-12 text-center">
61             <a href="{{ url('login') }}">Sudah punya akun? <strong>Login!</strong></a>
62         </div>
63     </div>
64 </form>
65 </div>
66 </div>
67
68 <!-- Scripts -->
69 <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
70 <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
71 <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
72 <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
73 <script>
74     $.ajaxSetup({
75         headers: {
76             'X-CSRF-TOKEN': $('input[name="_token"]').val()
77         }
78     });
79
80     $(document).ready(function () {
81         $("#form-register").validate({
82             rules: {
83                 username: { required: true, minlength: 4, maxlength: 20 },
84                 nama: { required: true, minlength: 3, maxlength: 100 },
85                 password: { required: true, minlength: 6, maxlength: 20 },
86             },
87             submitHandler: function (form) {
```



```
88 $.ajax({
89   url: form.action,
90   method: form.method,
91   data: $(form).serialize(),
92   success: function (response) {
93     $('.error-text').text('');
94     if (response.status) {
95       Swal.fire({
96         icon: 'success',
97         title: 'Berhasil Registrasi',
98         text: response.message
99       }).then(() => {
100         window.location.href = "{{ url('login') }}";
101       });
102     } else {
103       if (response.msgField) {
104         $.each(response.msgField, function (prefix, val) {
105           $('#error-' + prefix).text(val[0]);
106         });
107       }
108       Swal.fire({
109         icon: 'error',
110         title: 'Gagal Registrasi',
111         text: response.message || 'Silakan periksa kembali data yang diisi.'
112       });
113     }
114   },
115   error: function (xhr) {
116     Swal.fire({
117       icon: 'error',
118       title: 'Error',
119       text: 'Terjadi kesalahan di server.'
120     });
121   }
122 });
123 return false;
124 },
125 errorElement: 'span',
126 errorPlacement: function (error, element) {
127   error.addClass('invalid-feedback');
128   element.closest('.input-group').append(error);
129 },
130 highlight: function (element) {
131   $(element).addClass('is-invalid');
132 },
133 unhighlight: function (element) {
134   $(element).removeClass('is-invalid');
135 }
136 });
137 });
138 </script>
139 </body>
140 </html>
```

c. Kemudian saya menambahkan di routenya:

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




The top screenshot shows a web browser window with the URL `localhost/Lelyta2AWebLanjut/Minggu%207/PWL_POS/public/register`. The page displays a registration form titled "Registrasi Pengguna". The form includes the instruction "Silakan daftar untuk membuat akun" and three input fields: "admin 4" (with a user icon), "Shayla" (with a name icon), and a password field (with a lock icon). A green "Daftar" button is at the bottom, followed by a link "Sudah punya akun? Login!".

The bottom screenshot shows the same browser window, but the registration form is now in the background and slightly dimmed. A white modal box is centered on the screen, displaying a green checkmark icon and the text "Berhasil Registrasi". Below the text, it says "Registrasi berhasil! Silakan login." and there is a blue "OK" button.

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



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://iti.polinema.ac.id>

pengguna.

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