



Nama : Arimbi Putri Hapsari
Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 5 (lima)
Pertemuan ke- : 7 (tujuh)

JOBSHEET 07

Authentication dan *Authorization* di Laravel

Laravel Authentication dipergunakan untuk memproteksi halaman atau fitur dari web yang hanya diakses oleh orang tertentu yang diberikan hak. Fitur seperti ini biasanya ditemui di sistem yang memiliki fitur administrator atau sistem yang memiliki pengguna yang boleh menambahkan datanya.

Laravel membuat penerapan otentikasi sangat sederhana dan telah menyediakan berbagai fitur yang dapat dimanfaatkan tanpa perlu melakukan penambahan instalasi modul tertentu. File konfigurasi otentikasi terletak di config / auth.php, yang berisi beberapa opsi yang terdokumentasi dengan baik untuk mengubah konfigurasi dari layanan otentikasi.

Pada intinya, fasilitas otentikasi Laravel terdiri dari “guards” dan “providers”. Guards menentukan bagaimana pengguna diautentikasi untuk setiap permintaan. Misalnya, Laravel mengirim dengan guards untuk sesi dengan menggunakan penyimpanan session dan cookie.

Middleware

Middleware adalah lapisan perantara antara permintaan **route HTTP** yang masuk dan **action** dari Controller yang akan dijalankan. **Middleware** memungkinkan kita untuk melakukan berbagai tugas baik itu sebelum ataupun sesudah tindakan dilakukan. Kita juga dapat menggunakan **tool CLI** untuk membuat sebuah **Middleware** dalam **Laravel**. Beberapa contoh penggunaan **Middleware** meliputi autentikasi, validasi, manipulasi permintaan, dan lainnya. Berikut di bawah ini adalah manfaat dari **Middleware** :

- **Keamanan** : dalam **Middleware** memungkinkan kita untuk memverifikasi apakah pengguna sudah diautentikasi sebelum mengakses halaman tertentu. Dengan demikian, kita dapat melindungi data sensitif dan mengontrol hak akses pengguna.



- **Pemfilteran Data : Middleware** dapat digunakan untuk memanipulasi data permintaan sebelum sebuah *action* dalam *controller* dilakukan. Misalnya, kita dapat memeriksa terlebih dahulu data yang dikirim oleh pengguna sebelum data tersebut diproses lebih lanjut atau kita ingin memodifikasi data yang akan dikirim lalu kita dapat memeriksa ulang data yang akan dikirim oleh pengguna sebelum data tersebut diproses.
- **Logging dan Audit : Middleware** juga dapat digunakan untuk mencatat aktivitas pengguna atau melakukan audit terhadap permintaan yang masuk. Ini dapat membantu dalam pemantauan dan analisis aplikasi.

INFO

Kita akan menggunakan Laravel Auth secara manual seperti
<https://laravel.com/docs/10.x/authentication#authenticating-users>

Sesuai dengan **Studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. Implementasi Manual Authentication di Laravel

Autentikasi adalah proses untuk memverifikasi identitas pengguna yang mencoba mengakses sistem. Dalam konteks aplikasi web, autentikasi memastikan bahwa pengguna yang mencoba login memiliki hak akses yang sesuai berdasarkan kredensial seperti email dan password. Proses autentikasi berbeda dengan **otorisasi**, yang merupakan langkah lanjutan untuk menentukan hak akses apa yang dimiliki pengguna setelah mereka berhasil diautentikasi.

Konsep Autentikasi di Laravel

Laravel menawarkan sistem autentikasi yang sangat fleksibel. Laravel menyediakan mekanisme autentikasi bawaan melalui layanan authentication scaffolding seperti Laravel *Jetstream* dan *Breeze*, yang dapat secara otomatis menghasilkan halaman dan logika autentikasi. Namun, terkadang pengembang memerlukan implementasi autentikasi yang lebih manual untuk memberikan kontrol penuh terhadap setiap aspek dari proses tersebut.



Beberapa komponen penting dalam sistem autentikasi Laravel meliputi:

- *Guard*: Komponen yang mengatur bagaimana pengguna diautentikasi untuk setiap permintaan. *Guard* default menggunakan sesi dan cookie.
- *Provider*: Mengatur bagaimana pengguna diambil dari database atau sumber data lainnya. *Provider* default mengambil data pengguna dari database dengan menggunakan Eloquent ORM.
- *Session*: Laravel menggunakan sesi untuk menyimpan status autentikasi pengguna. Sesi memungkinkan sistem untuk mengingat pengguna yang sudah login di antara permintaan HTTP yang berbeda.

Alur umum dari autentikasi meliputi:

1. *Login*: Pengguna mengirimkan kredensial (biasanya berupa email dan password).
2. *Verifikasi Kredensial*: Sistem memeriksa apakah kredensial yang diberikan sesuai dengan data di database.
3. *Pembuatan Sesi*: Jika kredensial benar, sistem akan membuat sesi untuk pengguna yang akan disimpan di server.
4. *Akses ke Halaman yang Dilindungi*: Pengguna yang terautentikasi dapat mengakses halaman-halaman yang dilindungi oleh *middleware auth*.
5. *Logout*: Pengguna bisa keluar dari sistem dan sesi mereka akan dihapus.

Middleware Autentikasi

Middleware auth di Laravel digunakan untuk melindungi rute atau halaman agar hanya dapat diakses oleh pengguna yang sudah terautentikasi. Jika pengguna mencoba mengakses rute yang memerlukan autentikasi tanpa login, mereka akan diarahkan ke halaman login.

- **Guard** bertanggung jawab untuk menangani proses autentikasi pengguna. Laravel secara default menggunakan *guard* berbasis sesi untuk autentikasi web, namun juga mendukung *guard* berbasis token (seperti API).
- **Provider** bertugas untuk mengambil pengguna dari database. Laravel menyediakan *provider* default yang menggunakan Eloquent, namun juga mendukung *provider* lain seperti Query Builder.



Implementasi di Laravel 10

Kita akan menerapkan penggunaan authentication di Laravel. Dalam penerapan ini, kita akan mencoba membuat otentikasi secara di Laravel, agar kita paham langkah-langkah dalam membuat Authentication

Praktikum 1 – Implementasi Authentication :

1. Kita buka project laravel **PWL_POS** kita, dan kita modifikasi konfigurasi aplikasi kita di **config/auth.php**

```
62     'providers' => [
63         'users' => [
64             'driver' => 'eloquent',
65             'model' => App\Models\User::class,
66         ],

```

Pada bagian ini kita sesuaikan dengan Model untuk tabel **m_user** yang sudah kita buat

```
62     'providers' => [
63         'users' => [
64             'driver' => 'eloquent',
65             'model' => App\Models\UserModel::class,
66         ],
67     ],

```

Hasil

```
jobsheet > PWL_POS > config > auth.php
return [
    //js7
    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\Models\UserModel::class,
        ],
    ],
]
```

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



```
<?php
namespace App\Models;

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

class UserModel extends Authenticatable
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];

    protected $hidden = ['password']; // jangan di tampilkan saat select

    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash

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

Hasil

```
> jobsheet > PWL_POS > app > Models > UserModel.php > UserModel > level
<?php

namespace App\Models;

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

class UserModel extends Authenticatable
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];

    protected $hidden = ['password']; // jangan ditampilkan saat select

    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash

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



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

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

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    public function login()
    {
        if(Auth::check()){ // jika sudah login, maka redirect ke halaman home
            return redirect('/');
        }
        return view('auth.login');
    }
    public function postlogin(Request $request)
    {
        if($request->ajax() || $request->wantsJson()){
            $credentials = $request->only('username', 'password');
            if (Auth::attempt($credentials))
            {
                return response()->json([
                    'status' => true,
                    'message' => 'Login Berhasil',
                    'redirect' => url('/')
                ]);
            }
            return response()->json([
                'status' => false,
                'message' => 'Login Gagal'
            ]);
        }
        return redirect('login');
    }
    public function logout(Request $request)
    {
        Auth::logout();

        $request->session()->invalidate();
        $request->session()->regenerateToken();
        return redirect('login');
    }
}
```

Hasil



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

4. Setelah kita membuat [AuthController.php](#), kita buat view untuk menampilkan halaman
5. login. View kita buat di [auth/login.blade.php](#) , tampilan login bisa kita ambil dari contoh login di template **AdminLTE** seperti berikut (pada contoh login ini, kita gunakan page [login-V2](#) di **AdminLTE**)

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



```
<!-- Google Font: Source Sans Pro -->
<link rel="stylesheet"
      href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
<!-- Font Awesome -->
<link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
<!-- iCheck bootstrap -->
<link rel="stylesheet" href="{{ asset('plugins/iCheck-bootstrap/iCheck-bootstrap.min.css') }}">
<!-- SweetAlert2 -->
<link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap4.min.css') }}">
<!-- Theme style -->
<link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}"> </head>
<body class="hold-transition login-page">
<div class="login-box">
    <!-- /.login-logo -->
    <div class="card card-outline card-primary">
        <div class="card-header text-center"><a href="{{ url('/') }}">
            <b>Admin</b>LTE</a></div>
        <div class="card-body">
            <p class="login-box-msg">Sign in to start your session</p>
            <form action="{{ url('login') }}" method="POST" id="form-login">
                @csrf
                <div class="input-group mb-3">
                    <input type="text" id="username" name="username" class="form-control"
                           placeholder="Username">
                    <div class="input-group-append">
                        <div class="input-group-text">
                            <span class="fas fa-envelope"></span>
                        </div>
                    </div>
                    <small id="error-username" class="error-text text-danger"></small>
                </div>
                <div class="input-group mb-3">
                    <input type="password" id="password" name="password" class="form-control"
                           placeholder="Password">
                    <div class="input-group-append">
                        <div class="input-group-text">
                            <span class="fas fa-lock"></span>
                        </div>
                    </div>
                    <small id="error-password" class="error-text text-danger"></small>
                </div>
                <div class="row">
                    <div class="col-8">
                        <div class="icheck-primary">
                            <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
                        </div>
                    </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>

```



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

```
<!-- /.card -->
</div>
<!-- /.login-box -->

<!-- jQuery -->
```



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



```
<script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script> <!--
jquery-validation -->
<script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<script src="{{ asset('plugins/jquery-validation/additional-methods.min.js') }}"></script>
<!-- SweetAlert2 -->
<script src="{{ asset('plugins/sweetalert2/sweetalert2.min.js') }}"></script> <!--
AdminLTE App -->
<script src="{{ asset('dist/js/adminlte.min.js') }}"></script>
<script>
$.ajaxSetup({
headers: {
    'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
}
});

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



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

</script>



```
</body>  
</html>
```

Hasil

```
1     
2  html lang="en"
3  head
4  meta charset="UTF-8"
5  meta name="viewport" content="width=device-width, initial-scale=1.0"
6  titleHigh Reseller Title
7
8  link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Source+Sans+Pro:300,400,600,700&display=fallback"
9
10 link rel="icon" href="https://www.google.com/favicon.ico" type="image/x-icon"/>
```



6. Kemudian kita modifikasi [route/web.php](#) agar semua route masuk dalam auth

```
<?php

use App\Http\Controllers\UserController;
use App\Http\Controllers\SupplierController;
use App\Http\Controllers\BarangController;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers>WelcomeController;
use Illuminate\Support\Facades\Route;

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

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

Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu

    // masukkan semua route yang perlu autentikasi di sini

});;
```

Hasil

```
EEK7 > jobsheet > PWL_POS > routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use App\Http\Controllers\UserController;
6  use Illuminate\Support\Facades\Route;
7  use App\Http\Controllers>WelcomeController;
8  use App\Http\Controllers\SupplierController;
9  use App\Http\Controllers\BarangController;
10 use App\Http\Controllers\AuthController;
11
12 Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter (id), maka harus berupa angka
13
14 Route::get('login', [AuthController::class, 'login'])->name('login');
15 Route::post('login', [AuthController::class, 'postlogin']);
16 Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
17
18 Route::middleware(['auth'])->group(function (): void { // artinya semua route di dalam group ini
19     Route::get('/', [WelcomeController::class, 'index']);
20 });
21
```

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



Hasil

Tugas 1 – Implementasi Authentication :

1. Silahkan implementasikan proses login pada project kalian masing-masing
⇒ Tampilan Awal saat melakukan login



127.0.0.1:8000/login

AdminLTE

Sign in to start your session

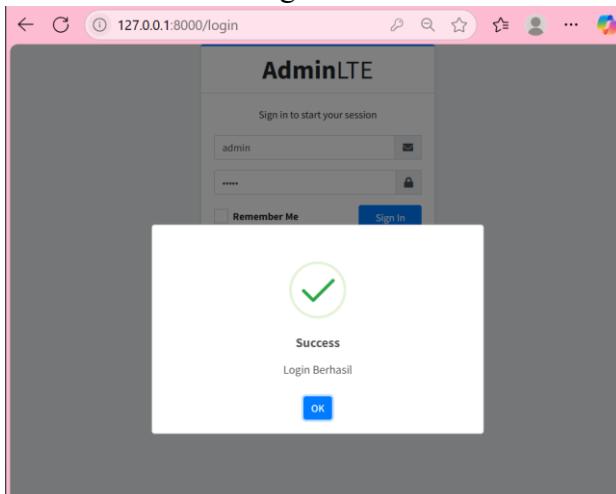
admin

.....

Remember Me

Don't have an account? [Register](#)

⇒ Setelah melakukan login



⇒ Lalu masuk ke halaman dashboard



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

Jawab

Modifikasi sidebar

```
week7 > jobsheet > PWL_POS > resources > views > layouts > sidebar.blade.php > ...
1   <div class="sidebar">
15    <nav class="mt-2">
16      <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview"
17        <li class="nav-item">
18          | <a>
19          | </a>
20          | <li class="nav-item">
21            |   <a href="{{ url('/barang') }}" class="nav-link {{ ($activeMenu ==
22 'penjualan')? 'active' : '' }} " >
23            |     <i class="nav-icon fas fa-cash-register"></i>
24            |     <p>Transaksi Penjualan</p>
25            |   </a>
26          | </li>
27          | <li class="nav-header">Lainnya</li>
28          | <li class="nav-item">
29            |   <a href="{{ url('/logout') }}" class="nav-link {{ ($activeMenu == 'logout')?
30 'active' : '' }} " >
31            |     <i class="nav-icon fas fa-sign-out-alt"></i>
32            |     <p>Logout</p>
33            |   </a>
34          | </li>
35        </ul>
36    </nav>
37  </div>
```

Setelah modifikasi



Setelah klik Logout, maka akan keluar dari halaman dashboard

Register'."/>

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



B. Implementasi *Authorization* di Laravel

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

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

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

Praktikum 2 – Implementasi *Authorization* di Laravel dengan Middleware

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

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



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

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

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

Hasil

```
week7 > jobsheet > PWL_POS > app > Models > UserModel.php > UserModel
10  class UserModel extends Authenticatable
11
12      protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
13
14      /**
15      * Relasi ke tabel level
16      */
17      public function level(): BelongsTo
18      {
19          return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
20      }
21
22      /**
23      * Mendapatkan nama role
24      */
25      public function getRoleName(): string
26      {
27          return $this->level->level_nama;
28      }
29
30      /**
31      * Cek apakah user memiliki role tertentu
32      */
33      public function hasRole($role): bool
34      {
35          return $this->level->level_kode == $role;
36      }
37
38
39
40
41
42
43
44
```



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

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

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

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

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

Hasil

```
week7 > jobsheet > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser
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 = $request->user()->getRole(); // ambil data level_kode dari user y
19     if (in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
20         return $next($request); // jika ada, maka lanjutkan request
21     }
22     // jika tidak punya role, maka tampilkan error 403
23     abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
24 }
25 }
```

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



```
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
```

Hasil

```
week7 > jobsheet > PWL_POS > app > Http > Kernel.php > Kernel
7   class Kernel extends HttpKernel
54   /*
55     protected $middlewareAliases = [
56       'auth' => \App\Http\Middleware\Authenticate::class,
57       'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yg kita buat
58       'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
59       'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
60       'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
```

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

| level_id | level_kode | level_nama | created_at | updated_at | deleted_at |
|----------|------------|---------------|------------|---------------------|------------|
| 1 | ADM | Administrator | NULL | NULL | NULL |
| 2 | MNG | Manager | NULL | NULL | NULL |
| 3 | STF | Staf | NULL | 2024-08-16 01:49:20 | NULL |
| 4 | KSR | Kasir | NULL | NULL | NULL |

Jawab

| ← T → | ▼ | level_id | level_kode | level_nama | created_at | updated_at |
|---------------------------------------------|---|----------|---------------|------------|------------|------------|
| <input type="checkbox"/> Edit Copy Delete | 1 | ADM | Administrator | NULL | NULL | NULL |
| <input type="checkbox"/> Edit Copy Delete | 2 | MNG | Manager | NULL | NULL | NULL |
| <input type="checkbox"/> Edit Copy Delete | 3 | STF | Staff/Kasir | NULL | NULL | NULL |

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

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

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

    // route Kategori
```



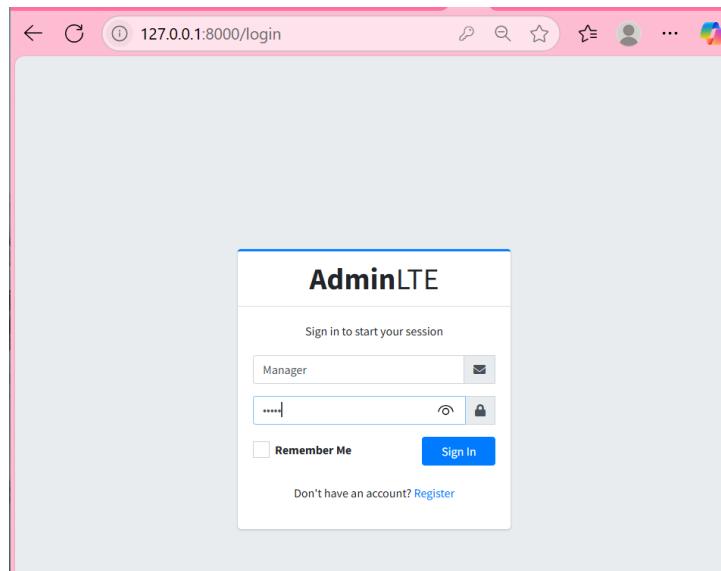
Hasil

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

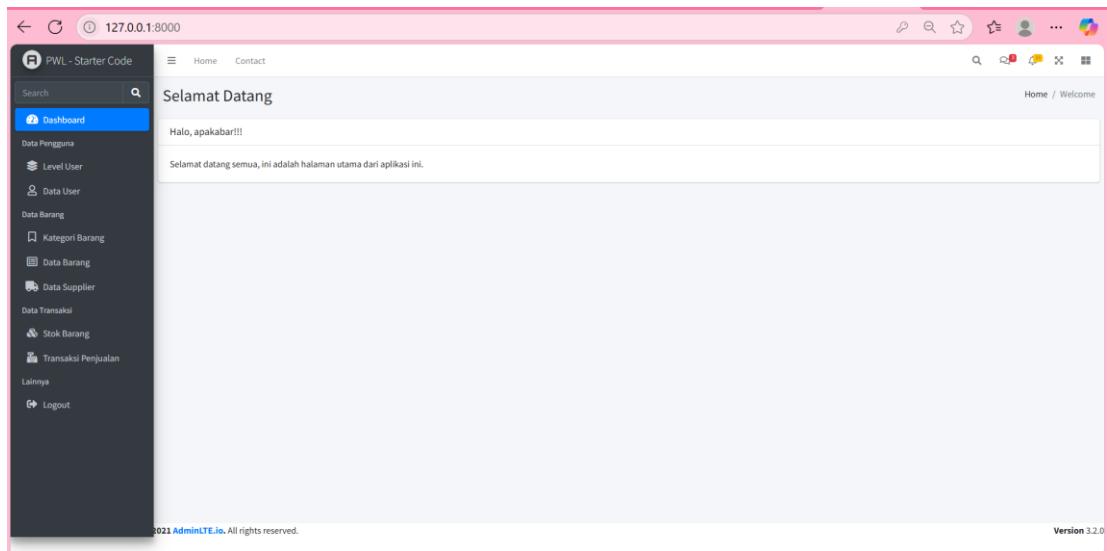
    // artinya semua route di dalam group ini harus punya role ADM (Administrator)
    Route::middleware(['authorize:ADM'])->group(function () : void {
        Route::get('/level', [LevelController::class, 'index']);
        Route::post('/level/list', [LevelController::class, 'list']); // untuk list json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

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

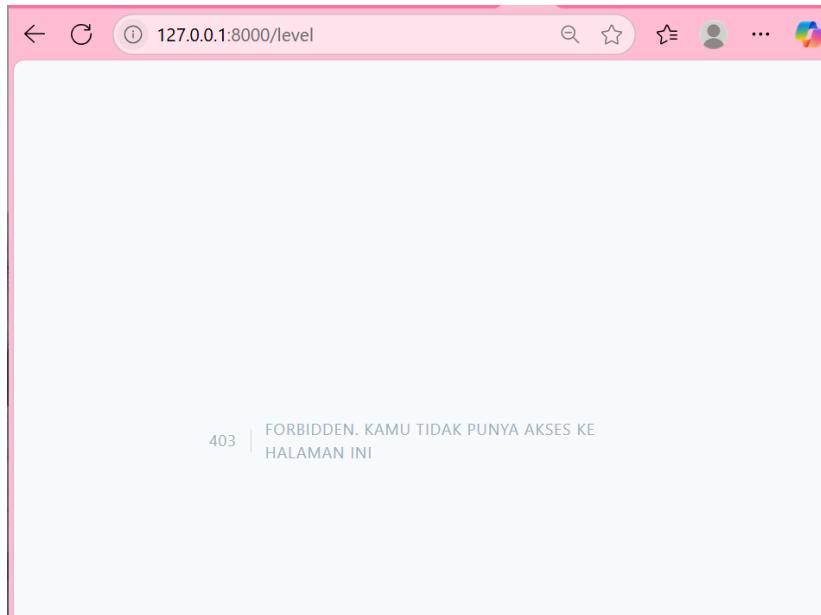
7. Untuk membuktikannya, sekarang kita coba login menggunakan akun selain level administrator, dan kita akses route menu level tersebut
⇒ Melakukan login selain role admin



⇒ masuk ke halaman dashboard



- ⇒ Hasil jika login dengan role selain admin, ketika ingin mengakses level user akan menghasilkan seperti gambar dibawah ini , itu dikarenakan izin akses hanya untuk role admin



- ⇒ Login lagi menggunakan role admin



127.0.0.1:8000/login

AdminLTE

Sign in to start your session

admin

.....

Remember Me **Sign In**

Don't have an account? [Register](#)

⇒ Lalu masuk ke halaman dashboard lagi

127.0.0.1:8000

PWL - Starter Code

Home Contact

Selamat Datang

Halo, apakah!!!

Selamat datang semua, ini adalah halaman utama dari aplikasi ini.

Home / Welcome

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Barang

Data Supplier

Data Transaksi

Stok Barang

Transaksi Penjualan

Lainnya

Logout

2021 AdminLTE.io. All rights reserved.

Version 3.2.1

⇒ Hasil jika login dengan role admin, ketika mengakses level user akan menampilkan gambar seperti dibawah ini



The screenshot shows a web application interface for managing user levels. On the left is a sidebar with a search bar and a list of menu items: Dashboard, Data Pengguna, Level User (which is selected and highlighted in blue), Data User, Data Barang, Kategori Barang, Data Barang, Data Supplier, Data Transaksi, Stok Barang, Transaksi Penjualan, Lainnya, and Logout. The main content area has a header "Daftar Level" and a sub-header "Daftar level yang terdaftar dalam sistem". It displays a table with three entries:

| ID | Kode Level | Nama Level | Aksi |
|----|------------|---------------|-------------------------|
| 1 | ADM | Administrator | [Detail] [Edit] [Hapus] |
| 2 | MNG | Manager | [Detail] [Edit] [Hapus] |
| 3 | STF | Staff/Kasir | [Detail] [Edit] [Hapus] |

At the bottom of the page, there is a footer with the text "2021 AdminLTE.io. All rights reserved." and "Version 3.2.0".

Tugas 2 – Implementasi Authorization :

1. Apa yang kalian pahami pada praktikum 2 ini?

Jawab: Dengan menambahkan authorize:ADM pada suatu route, akses ke route tersebut terbatas hanya untuk pengguna dengan role ADM. Pengguna dengan role lain tidak dapat mengaksesnya dan akan diarahkan ke halaman error 404.

2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Submit kode untuk impementasi Authorization pada repository github kalian.

C. Multi-Level Authorization di Laravel

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



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

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

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

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

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

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

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



Hasil

```
week7 > jobsheet > PWL_POS > app > Models > UserModel.php > UserModel
10 class UserModel extends Authenticatable
11 {
12     /**
13      * Cek apakah user memiliki role tertentu
14      */
15     public function hasRole($role): bool
16     {
17         return $this->level->level_kode == $role;
18     }
19
20     /**
21      * Mendapatkan kode role
22      */
23     public function getRole()
24     {
25         return $this->level->level_kode;
26     }
27 }
```



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

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

Hasil

```
week7 > jobsheet > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser > handle
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Symfony\Component\HttpFoundation\Response;
8
9 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 = $request->user()->getRole(); // ambil data level_kode dari user yg login
19         if (in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
20             return $next($request); // jika ada, maka lanjutkan request
21         }
22         // jika tidak punya role, maka tampilkan error 403
23         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
24     }
25 }
```

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



```
// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
4.});
```

Hasil

```
36 Route::group(['prefix' => 'user'], function () {
37     Route::middleware(['authorize:ADM'])->group(function() {
38         Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user
56     Route::group(['prefix' => 'level'], function () {
57         Route::middleware(['authorize:ADM'])->group(function () {
58             Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal level
76     Route::group(['prefix' => 'kategori'], function () {
77         Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
78             Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal kategori
96     Route::group(['prefix' => 'supplier'], function () {
97         Route::middleware(['authorize:ADM,MNG'])->group(function () {
98             Route::get('/', [SupplierController::class, 'index']); // menampilkan halaman awal supplier
116     Route::group(['prefix' => 'barang'], function () {
117         Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
118             Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal barang

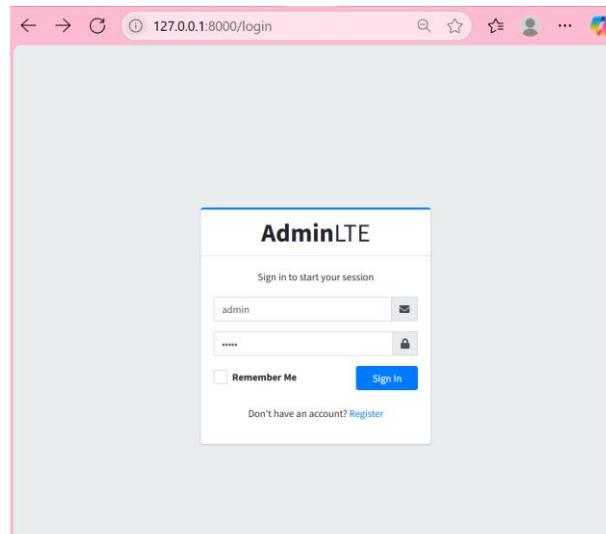
```

Tugas 3 – Implementasi Multi-Level Authorization :

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

Jawab

⇒ Admin





The screenshot shows a web-based administration interface for the PWL-Starter Code application. On the left, a sidebar menu lists various data management modules: Dashboard, Data Pengguna, Level User (selected), Data Barang, Kategori Barang, Data Barang, Data Supplier, Data Transaksi, Stok Barang, Transaksi Penjualan, Lainnya, and Logout. The main content area is titled 'Daftar Level' and displays a table of user levels:

| ID | Kode Level | Nama Level | Aksi |
|----|------------|---------------|-------------------------|
| 1 | ADM | Administrator | [Detail] [Edit] [Hapus] |
| 2 | MNG | Manager | [Detail] [Edit] [Hapus] |
| 3 | STF | Staff/Kasir | [Detail] [Edit] [Hapus] |

Below the table, a message indicates 'Showing 1 to 3 of 3 entries'. The top right of the page includes links for 'Home / Level', 'Tambah', and 'Tambah Ajax'. The bottom right shows 'Version 3.2.0'.

The second screenshot shows the 'Daftar User' (User List) page. The sidebar and overall layout are identical to the first screenshot. The main content area is titled 'Daftar User' and displays a table of users:

| ID | Username | Nama | Level Pengguna | Aksi |
|----|----------|---------------|----------------|-------------------------|
| 1 | admin | Administrator | Administrator | [Detail] [Edit] [Hapus] |
| 2 | Manager | Manager | Manager | [Detail] [Edit] [Hapus] |
| 3 | Staff | Staff/Kasir | Staff/Kasir | [Detail] [Edit] [Hapus] |

Below the table, a message indicates 'Showing 1 to 3 of 3 entries'. The top right of the page includes links for 'Home / User', 'Tambah', and 'Tambah Ajax'. The bottom right shows 'Version 3.2.0'.



127.0.0.1:8000/kategori

Daftar Kategori

| ID | Kode Kategori | Nama Kategori | Aksi |
|----|---------------|---------------|-------------------------|
| 1 | KAT001 | Pakaian | [Detail] [Edit] [Hapus] |
| 2 | KAT002 | Elektronik | [Detail] [Edit] [Hapus] |
| 3 | KAT003 | Makanan | [Detail] [Edit] [Hapus] |
| 4 | KAT004 | Minuman | [Detail] [Edit] [Hapus] |
| 5 | KAT005 | Alat Tulis | [Detail] [Edit] [Hapus] |

Showing 1 to 5 of 5 entries

127.0.0.1:8000/barang

Daftar Barang

| ID | Kode Barang | Nama Barang | Harga Beli | Harga Jual | Kategori Barang | Aksi |
|----|-------------|---------------|------------|------------|-----------------|-------------------------|
| 1 | BRG001 | Facial Wash A | 20000 | 30000 | Pakaian | [Detail] [Edit] [Hapus] |
| 2 | BRG002 | Facial Wash B | 25000 | 35000 | Pakaian | [Detail] [Edit] [Hapus] |
| 3 | BRG003 | Toner A | 22000 | 32000 | Elektronik | [Detail] [Edit] [Hapus] |
| 4 | BRG004 | Toner B | 27000 | 37000 | Elektronik | [Detail] [Edit] [Hapus] |
| 5 | BRG005 | Serum A | 30000 | 40000 | Makanan | [Detail] [Edit] [Hapus] |
| 6 | BRG006 | Serum B | 35000 | 45000 | Makanan | [Detail] [Edit] [Hapus] |
| 7 | BRG007 | Moisturizer A | 28000 | 38000 | Minuman | [Detail] [Edit] [Hapus] |
| 8 | BRG008 | Moisturizer B | 32000 | 42000 | Minuman | [Detail] [Edit] [Hapus] |
| 9 | BRG009 | Sunscreen A | 25000 | 35000 | Alat Tulis | [Detail] [Edit] [Hapus] |
| 10 | BRG010 | Sunscreen B | 27000 | 37000 | Alat Tulis | [Detail] [Edit] [Hapus] |

Showing 1 to 10 of 15 entries



| ID | Kode Supplier | Nama Supplier | Alamat Supplier | Aksi |
|----|---------------|--------------------|-----------------|-----------------------|
| 1 | SKT001 | Skintific Official | Jakarta | [Detail, Edit, Hapus] |
| 2 | SKT002 | Distributor A | Surabaya | [Detail, Edit, Hapus] |
| 3 | SKT003 | Distributor B | Bandung | [Detail, Edit, Hapus] |

⇒ Manager

AdminLTE

Sign in to start your session

Manager

Remember Me

[Don't have an account? Register](#)

| ID | Kode Kategori | Nama Kategori | Aksi |
|----|---------------|---------------|-----------------------|
| 1 | KAT001 | Pakalan | [Detail, Edit, Hapus] |
| 2 | KAT002 | Elektronik | [Detail, Edit, Hapus] |
| 3 | KAT003 | Makanan | [Detail, Edit, Hapus] |
| 4 | KAT004 | Minuman | [Detail, Edit, Hapus] |
| 5 | KAT005 | Alat Tulis | [Detail, Edit, Hapus] |



127.0.0.1:8000/barang

PWL - Starter Code

Daftar Barang

Daftar barang yang terdaftar dalam sistem

Filter: Kategori Barang

Show 10 entries

| ID | Kode Barang | Nama Barang | Harga Beli | Harga Jual | Kategori Barang | Aksi |
|----|-------------|---------------|------------|------------|-----------------|-------------------------------------------------------------------|
| 1 | BRG001 | Facial Wash A | 20000 | 30000 | Pakaian | Detail Edit Hapus |
| 2 | BRG002 | Facial Wash B | 25000 | 35000 | Pakaian | Detail Edit Hapus |
| 3 | BRG003 | Toner A | 22000 | 32000 | Elektronik | Detail Edit Hapus |
| 4 | BRG004 | Toner B | 27000 | 37000 | Elektronik | Detail Edit Hapus |
| 5 | BRG005 | Serum A | 30000 | 40000 | Makanan | Detail Edit Hapus |
| 6 | BRG006 | Serum B | 35000 | 45000 | Makanan | Detail Edit Hapus |
| 7 | BRG007 | Moisturizer A | 28000 | 38000 | Minuman | Detail Edit Hapus |
| 8 | BRG008 | Moisturizer B | 32000 | 42000 | Minuman | Detail Edit Hapus |
| 9 | BRG009 | Sunscreen A | 25000 | 35000 | Alat Tulis | Detail Edit Hapus |
| 10 | BRG010 | Sunscreen B | 27000 | 37000 | Alat Tulis | Detail Edit Hapus |

Showing 1 to 10 of 15 entries

Previous [1](#) [2](#) Next

127.0.0.1:8000/supplier

PWL - Starter Code

Daftar Supplier

Daftar supplier yang terdaftar dalam sistem

Filter: Show 10 entries

| ID | Kode Supplier | Nama Supplier | Alamat Supplier | Aksi |
|----|---------------|--------------------|-----------------|-------------------------------------------------------------------|
| 1 | SKT001 | Skintific Official | Jakarta | Detail Edit Hapus |
| 2 | SKT002 | Distributor A | Surabaya | Detail Edit Hapus |
| 3 | SKT003 | Distributor B | Bandung | Detail Edit Hapus |

Showing 1 to 3 of 3 entries

Previous [1](#) Next

2021 AdminLTE.io. All rights reserved.

Version 3.2.0

⇒ Staff



AdminLTE

Sign in to start your session

Staff

.....

Remember Me

Don't have an account? [Register](#)

Daftar Kategori

Daftar kategori yang terdaftar dalam sistem

| ID | Kode Kategori | Nama Kategori | Aksi |
|----|---------------|---------------|-------------------------------------------------------------------|
| 1 | KAT001 | Pakaian | Detail Edit Hapus |
| 2 | KAT002 | Elektronik | Detail Edit Hapus |
| 3 | KAT003 | Makanan | Detail Edit Hapus |
| 4 | KAT004 | Minuman | Detail Edit Hapus |
| 5 | KAT005 | Alat Tulis | Detail Edit Hapus |

Showing 1 to 5 of 5 entries

2021 AdminLTE.io. All rights reserved. Version 3.2.1



- Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
- Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menumenu yang sesuai dengan Level/Jenis User

Hasil

⇒ User

```
36 Route::group(['prefix' => 'user'], function () {
37     Route::middleware(['authorize:ADM'])->group(function() {
38         Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user
39         Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
40         Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
41         Route::post('/{id}', [UserController::class, 'store']); // menyimpan data user baru
42         Route::get('/create_ajax', [UserController::class, 'create_ajax']); // menampilkan halaman form tambah user ajax
43         Route::post('/ajax', [UserController::class, 'store_ajax']); // menyimpan data user baru ajax
44         Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user
45         Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user
46         Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
47         Route::get('/{id}/show_ajax', [UserController::class, 'show_ajax']); // menampilkan halaman detail user ajax
48         Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user ajax
49         Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // menyimpan perubahan data user ajax
50         Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // menampilkan halaman konfirmasi delete user ajax
51         Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // menghapus data user ajax
52         Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
53     });
54 });


```

⇒ Level

```
56 Route::group(['prefix' => 'level'], function () {
57     Route::middleware(['authorize:ADM'])->group(function () {
58         Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal level
59         Route::post('/list', [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
60         Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level
61         Route::post('/{id}', [LevelController::class, 'store']); // menyimpan data level baru
62         Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // menampilkan halaman form tambah level ajax
63         Route::post('/ajax', [LevelController::class, 'store_ajax']); // menyimpan data level baru ajax
64         Route::get('/{id}', [LevelController::class, 'show']); // menampilkan detail level
65         Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman form edit level
66         Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan data level
67         Route::get('/{id}/show_ajax', [LevelController::class, 'show_ajax']); // menampilkan halaman detail level ajax
68         Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // menampilkan halaman form edit level ajax
69         Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // menyimpan perubahan data level ajax
70         Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // menampilkan halaman konfirmasi delete level ajax
71         Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // menghapus data level ajax
72         Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus data level
73     });
74 });


```

⇒ Kategori



```
76     Route::group(['prefix' => 'kategori'], function () {
77         Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
78             Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal kategori
79             Route::post('/list', [KategoriController::class, 'list']); // menampilkan data kategori dalam bentuk json untuk datatables
80             Route::get('/create', [KategoriController::class, 'create']); // menampilkan halaman form tambah kategori
81             Route::post('/store', [KategoriController::class, 'store']); // menyimpan data kategori baru
82             Route::get('/create_ajax', [KategoriController::class, 'create_ajax']); // menampilkan halaman form tambah kategori ajax
83             Route::post('/ajax', [KategoriController::class, 'store_ajax']); // menyimpan data kategori baru ajax
84             Route::get('/id', [KategoriController::class, 'show']); // menampilkan detail kategori
85             Route::get('/id/edit', [KategoriController::class, 'edit']); // menampilkan halaman form edit kategori
86             Route::put('/id', [KategoriController::class, 'update']); // menyimpan perubahan data kategori
87             Route::get('/id/show_ajax', [KategoriController::class, 'show_ajax']); // menampilkan halaman detail kategori ajax
88             Route::get('/id/edit_ajax', [KategoriController::class, 'edit_ajax']); // menampilkan halaman form edit kategori ajax
89             Route::put('/id/update_ajax', [KategoriController::class, 'update_ajax']); // menyimpan perubahan data kategori ajax
90             Route::get('/id/delete_ajax', [KategoriController::class, 'confirm_ajax']); // menampilkan halaman konfirmasi delete kategori
91             Route::delete('/id/delete_ajax', [KategoriController::class, 'delete_ajax']); // menghapus data kategori ajax
92             Route::delete('/id', [KategoriController::class, 'destroy']); // menghapus data kategori
93         });
94     });
95 }
```

⇒ Supplier

```
96     Route::group(['prefix' => 'supplier'], function () {
97         Route::middleware(['authorize:ADM,MNG'])->group(function () {
98             Route::get('/', [SupplierController::class, 'index']); // menampilkan halaman awal supplier
99             Route::post('/list', [SupplierController::class, 'list']); // menampilkan data supplier dalam bentuk json untuk datatables
100            Route::get('/create', [SupplierController::class, 'create']); // menampilkan halaman form tambah supplier
101            Route::post('/store', [SupplierController::class, 'store']); // menyimpan data supplier baru
102            Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); // menampilkan halaman form tambah supplier ajax
103            Route::post('/ajax', [SupplierController::class, 'store_ajax']); // menyimpan data supplier baru ajax
104            Route::get('/id', [SupplierController::class, 'show']); // menampilkan detail supplier
105            Route::get('/id/edit', [SupplierController::class, 'edit']); // menampilkan halaman form edit supplier
106            Route::put('/id', [SupplierController::class, 'update']); // menyimpan perubahan data supplier
107            Route::get('/id/show_ajax', [SupplierController::class, 'show_ajax']); // menampilkan halaman detail supplier ajax
108            Route::get('/id/edit_ajax', [SupplierController::class, 'edit_ajax']); // menampilkan halaman form edit supplier ajax
109            Route::put('/id/update_ajax', [SupplierController::class, 'update_ajax']); // menyimpan perubahan data supplier ajax
110            Route::get('/id/delete_ajax', [SupplierController::class, 'confirm_ajax']); // menampilkan halaman konfirmasi delete supplier
111            Route::delete('/id/delete_ajax', [SupplierController::class, 'delete_ajax']); // menghapus data supplier ajax
112            Route::delete('/id', [SupplierController::class, 'destroy']); // menghapus data supplier
113        });
114    });
115 }
```

⇒ Barang

```
116     Route::group(['prefix' => 'barang'], function () {
117         Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
118             Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal barang
119             Route::post('/list', [BarangController::class, 'list']); // menampilkan data barang dalam bentuk json untuk datatables
120             Route::get('/create', [BarangController::class, 'create']); // menampilkan halaman form tambah barang
121             Route::post('/store', [BarangController::class, 'store']); // menyimpan data barang baru
122             Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // menampilkan halaman form tambah barang ajax
123             Route::post('/ajax', [BarangController::class, 'store_ajax']); // menyimpan data barang baru ajax
124             Route::get('/id', [BarangController::class, 'show']); // menampilkan detail barang
125             Route::get('/id/edit', [BarangController::class, 'edit']); // menampilkan halaman form edit barang
126             Route::put('/id', [BarangController::class, 'update']); // menyimpan perubahan data barang
127             Route::get('/id/show_ajax', [BarangController::class, 'show_ajax']); // menampilkan halaman detail barang ajax
128             Route::get('/id/edit_ajax', [BarangController::class, 'edit_ajax']); // menampilkan halaman form edit barang ajax
129             Route::put('/id/update_ajax', [BarangController::class, 'update_ajax']); // menyimpan perubahan data barang ajax
130             Route::get('/id/delete_ajax', [BarangController::class, 'confirm_ajax']); // menampilkan halaman konfirmasi delete barang ajax
131             Route::delete('/id/delete_ajax', [BarangController::class, 'delete_ajax']); // menghapus data barang ajax
132             Route::delete('/id', [BarangController::class, 'destroy']); // menghapus data barang
133         });
134     });
135 }
```

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

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

Hasil

⇒ AuthController.php



```
week7 > jobsheet > PWL_POS > app > Http > Controllers > AuthController.php > postlogin
10  class AuthController extends Controller
49    public function register()
50    {
51      $level = LevelModel::all(); // Ambil semua level dari database
52      return view('auth.register', compact('level'));
53    }
54
55    public function postregister(Request $request)
56    {
57      $request->validate([
58        'username' => 'required',
59        'nama' => 'required',
60        'password' => 'required|min:5',
61        'level_id' => 'required|exists:m_level,level_id' // Validasi level harus ada di tabel m_level
62      ]);
63
64      $user = new UserModel();
65      $user->username = $request->username;
66      $user->nama = $request->nama;
67      $user->password = bcrypt($request->password);
68      $user->level_id = $request->level_id;
69      $user->save();
70
71      return redirect('login')->with('success', 'Registrasi berhasil! Silakan login.');
72    }
73  }
```

⇒ UserController.php

```
week7 > jobsheet > PWL_POS > app > Http > Controllers > UserController.php > delete_ajax
13  class UserController extends Controller
14
15    public function delete_ajax(Request $request, $id)
16    {
17      // cek apakah request dari ajax
18      if ($request->ajax() || $request->wantsJson()) {
19        $user = UserModel::find($id);
20        if ($user) {
21          try {
22            // Cek apakah user yang dihapus adalah user yang sedang login
23            if ($user->username == Auth::user()->username) {
24              $logout = true;
25            }
26
27            // Menghapus data user berdasarkan ID
28            UserModel::destroy($id);
29            if($logout) {
30              return response()->json([
31                'status' => true,
32                'logout' => true,
33                'message' => 'Data user berhasil dihapus'
34              ]);
35            }
36
37            return response()->json([
38              'status' => true,
39              'message' => 'Data user berhasil dihapus'
40            ]);
41          } catch (\Illuminate\Database\QueryException $e) {
42            // jika terjadi error ketika menghapus data
43          }
44        }
45      }
46    }
47  }
```

⇒ login.blade.php

```
jobsheet > PWL_POS > resources > views > auth > login.blade.php > html
<html lang="en">
<body class="hold-transition login-page">
  <div class="login-box">
    <div class="card card-outline card-primary">
      <div class="card-body">
        @if (session('error'))
          <div class="alert alert-danger text-center">{{ session('error') }}</div>
        @endif
        <p class="login-box-msg">Sign in to start your session</p>
        <form action="{{ url('login') }}" method="post" id="form-login">
          <input type="text" name="email" value="{{ old('email') }}"
          <input type="password" name="password" value="{{ old('password') }}"/>
        </div>
        <div class="text-center mt-4">
          <p>Don't have an account? <a href="{{ url('register') }}>Register</a></p>
        </div>
```

⇒ register.blade.php



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta name="csrf-token" content="{{ csrf_token() }}>
8   <title>Registrasi Pengguna</title>
9
10  <link rel="stylesheet"
11    href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
12  <link rel="stylesheet" href="{{ asset('adminLTE/plugins/fontawesome-free/css/all.min.css') }}>
13  <link rel="stylesheet" href="{{ asset('adminLTE/plugins/iCheck/hootstrap/iCheck.hootstrap.min.css') }}>
14  <link rel="stylesheet" href="{{ asset('adminLTE/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}>
15  <link rel="stylesheet" href="{{ asset('adminLTE/dist/css/adminlte.min.css') }}>
16 </head>
17
18 <body class="hold-transition login-page">
19   <div class="login-box">
20     <div class="card card-outline card-primary">
21       <div class="card-header text-center">
22         <a href="{{ url('/') }}" class="h1"><b>Admin</b></a>
23       </div>
24       <div class="card-body">
25         <p class="login-box-msg">Daftar Pengguna Baru</p>
26         <form action="{{ url('register') }}" method="POST" id="form-register">
27           <div class="form-group">
28             <label>Pilih Role</label>
29             <select name="level_id" id="level_id" class="form-control" required>
30               <option value="">> Pilih Level ...</option>
31               @foreach($level as $s)
32                 <option value="{{ $s->level_id }}>{{ $s->level_name }}</option>
33               @endforeach
34             </select>
35             <small id="error-level_id" class="error-text form-text text-danger"></small>
36           </div>
37           <div class="form-group">
38             <label>Username</label>
39             <input type="text" name="username" id="username" class="form-control" required minlength="4" maxlength="20" placeholder="Masukkan Username">
40             <small id="error-username" class="error-text form-text text-danger"></small>
41           </div>
42           <div class="form-group">
43             <label>Nama</label>
44             <input type="text" name="name" id="name" class="form-control" required minlength="100" placeholder="Masukkan Nama">
45             <small id="error-name" class="error-text form-text text-danger"></small>
46           </div>
47           <div class="form-group">
48             <label>Password</label>
49             <input type="password" name="password" id="password" class="form-control" required minlength="5" maxlength="20" placeholder="Masukkan Password">
50             <small id="error-password" class="error-text form-text text-danger"></small>
51           </div>
52           <div class="row justify-content-end">
53             <div class="col auto">
54               <button type="submit" class="btn btn-primary btn-block">Sign Up</button>
55             </div>
56           </div>
57           <div class="text-center mt-3">
58             <small>Sudah punya akun? <a href="{{ url('login') }}>Login</a></small>
59           </div>
60         </div>
61       </form>
62     </div>
63   </div>
64
65   <script src="{{ asset('adminLTE/plugins/jquery/jquery.min.js') }}></script>
66   <script src="{{ asset('adminLTE/plugins/bootstrap/js/bootstrap.bundle.min.js') }}></script>
67   <script src="{{ asset('adminLTE/plugins/jquery-validation/jquery.validate.min.js') }}></script>
68   <script src="{{ asset('adminLTE/plugins/jquery-validation/additional-methods.min.js') }}></script>
69   <script src="{{ asset('adminLTE/plugins/sweetalert2/sweetalert2.min.js') }}></script>
70   <script src="{{ asset('adminLTE/dist/js/adminlte.min.js') }}></script>
71
72   <script>
73     $(document).ready(function () {
74       $('#form-register').validate({
75         rules: {
76           username: { required: true, minlength: 4, maxlength: 20 },
77           name: { required: true, maxlength: 100 },
78           password: { required: true, minlength: 5, maxlength: 20 },
79           password_confirmation: { equalTo: '#name#password' },
80           level_id: { required: true, number: true }
81         },
82         submitHandler: function (form) {
83           $.signIn({
84             headers: {
85               'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
86             },
87             url: form.action,
88             type: form.method,
89             data: $(form).serialize(),
90             success: function(response) {
91               if (response.status) {
92                 Swal.fire({
93                   icon: 'success',
94                   title: 'Registrasi Berhasil',
95                   text: response.message,
96                 }).then((result) => {
97                   if (result.value) {
98                     window.location = response.redirect;
99                   }
100                 });
101               } else {
102                 $('#text-danger').text('');
103                 $.each(response.errors, function (key, val) {
104                   $('#error-' + key).text(val[0]);
105                 });
106                 Swal.fire({
107                   icon: 'error',
108                   title: 'Terjadi Kesalahan',
109                   text: response.message
110                 });
111               }
112             }
113           });
114         },
115         highlight: function (element) {
116           $(element).addClass('is-invalid');
117         },
118         errorElement: 'span',
119         errorPlacement: function (error, element) {
120           error.addClass('invalid-feedback');
121           element.closest('.input-group').append(error);
122         },
123         invalidInput: function (element) {
124           $(element).addClass('is-invalid');
125         },
126         unhighlight: function (element) {
127           $(element).removeClass('is-invalid');
128         }
129       });
130     });
131   </script>
132 </body>
133 </html>
```



⇒ confirm_ajax.blade.php

```
jobsheet > PWL_POS > resources > views > user > confirm_ajax.blade.php > ...
<script>
$(document).ready(function() {
    submitHandler: function(form) {
        success: function(response) {
            text: response.message
        };
        // Cek apakah user yang dihapus adalah user yang sedang login
        if (response.logout) {
            window.location.href = "/logout"; // Redirect ke halaman logout
        } else {
            dataUser.ajax.reload(); // Reload tabel jika bukan user yang sedang login
        }
    }
});
```

⇒ Web.php

```
Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');
Route::get('register', [AuthController::class, 'register']);
Route::post('register', [AuthController::class, 'PostRegister']);
```

2. Screenshot hasil yang kalian kerjakan

Hasil

The screenshot shows a web browser window with the URL 127.0.0.1:8000/register. The page title is "AdminLTE". The main content is a form titled "Daftar Pengguna Baru". It has a dropdown menu "Pilih Role" set to "Staff/Kasir", a "Username" field containing "Staffarimbi", a "Nama" field containing "Arimbi", and a "Password" field (redacted). At the bottom right is a blue "Sign Up" button, and at the bottom left is a link "Sudah punya akun? [Login](#)".



127.0.0.1:8000

PWL - Starter Code

Home Contact

Selamat Datang

Halo, apakah!!!

Selamat datang semua, ini adalah halaman utama dari aplikasi ini.

Home / Welcome

Version 3.2.0

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

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