



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
```

```
'providers' => [  
    'users' => [  
        'driver' => 'eloquent',  
        'model' => App\Models\UserModel::class,  
    ],
```



2. Selanjutnya kita modifikasi sedikit pada `UserModel.php` untuk bisa melakukan proses autentikasi

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

```
<?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;

15 references | 0 implementations | ""
class UserModel extends Authenticatable
{
    use HasFactory;

    0 references
    protected $table = 'm_user';
    0 references
    protected $primaryKey = 'user_id';

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

    // Agar kolom password tidak ditampilkan saat select
    0 references
    protected $hidden = ['password'];

    // Otomatis hash password saat diset
    0 references
    protected $casts = [
        'password' => 'hashed',
    ];

    /**
     * Relasi ke tabel level
     */
    0 references | 0 overrides
    public function level(): BelongsTo
    {
        return $this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: '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');
    }
}
```



```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

0 references|0 implementations
class AuthController extends Controller
{
    0 references|0 overrides
    public function login(): Factory\View
    {
        if (Auth::check()) { // jika sudah login, maka redirect ke halaman home return redirect('/');
        }
        return view('auth.login');
    }

    0 references|0 overrides
    public function postlogin(Request $request): JsonResponse|mixed|Redirector|RedirectRes...
    {
        if ($request->ajax() || $request->wantsJson()) {
            $credentials = $request->only(keys: 'username', 'password');

            if (Auth::attempt(credentials: $credentials)) {
                return response()->json(data: [
                    'status' => true,
                    'message' => 'Login Berhasil',
                    'redirect' => url(path: '/')
                ]);
            }
            return response()->json(data: [
                'status' => false,
                'message' => 'Login Gagal'
            ]);
        }

        return redirect(to: 'login');
    }

    0 references|0 overrides
    public function logout(Request $request): Redirector|RedirectResponse
    {
        Auth::logout();

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

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

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



```
<!-- Google Font: Source Sans Pro -->
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
<!-- Font Awesome -->
<link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
<!-- icheck bootstrap -->
<link rel="stylesheet" href="{{ asset('plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
<!-- SweetAlert2 -->
<link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
<!-- Theme style -->
<link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
</head>
<body class="hold-transition login-page">
<div class="login-box">
  <!-- /.login-logo -->
  <div class="card card-outline card-primary">
    <div class="card-header text-center"><a href="{{ url('/') }}"
class="h1"><b>Admin</b>LTE</a></div>
    <div class="card-body">
      <p class="login-box-msg">Sign in to start your session</p>
      <form action="{{ url('login') }}" method="POST" id="form-login">
        @csrf
        <div class="input-group mb-3">
          <input type="text" id="username" name="username" class="form-control"
placeholder="Username">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-envelope"></span>
            </div>
          </div>
          <small id="error-username" class="error-text text-danger"></small>
        </div>
        <div class="input-group mb-3">
          <input type="password" id="password" name="password" class="form-control"
placeholder="Password">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-lock"></span>
            </div>
          </div>
          <small id="error-password" class="error-text text-danger"></small>
        </div>
        <div class="row">
          <div class="col-8">
            <div class="icheck-primary">
              <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
            </div>
          <!-- /.col -->
          <div class="col-4">
            <button type="submit" class="btn btn-primary btn-block">Sign In</button>
          </div>
          <!-- /.col -->
        </div>
      </form>
    </div>
    <!-- /.card-body -->
  </div>
  <!-- /.card -->
</div>
<!-- /.login-box -->

<!-- jQuery -->
```




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

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

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



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

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

```
<?php

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

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

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

Route::middleware(['auth'])->group(function() { // artinya semua route di dalam group ini harus login dulu
    // masukkan semua route yang perlu autentikasi di sini
});
```

```
Route::pattern(key: 'id', pattern: '[0-9]+');

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

// Group route yang hanya bisa diakses setelah login
Route::middleware(middleware: ['auth'])->group(callback: function () { void {

    // Masukkan semua route yang membutuhkan autentikasi di sini

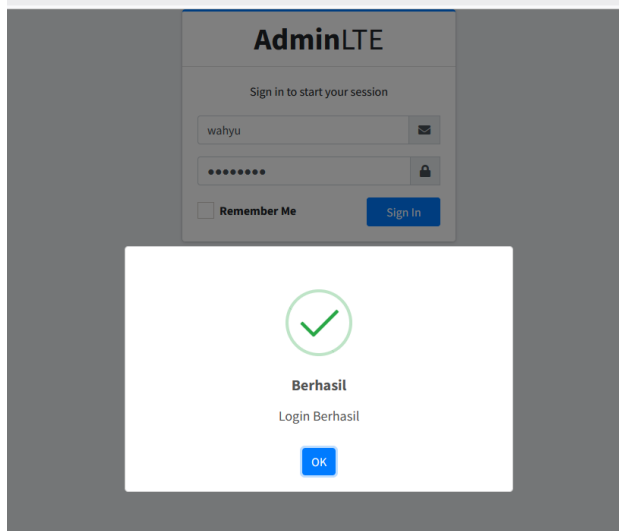
    // Contoh:
    // Route::get('/dashboard', [DashboardController::class, 'index'])->name('dashboard');
    // Route::resource('/user', UserController::class);

});
```

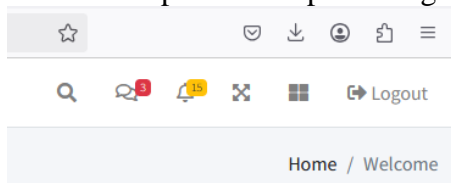
6. Ketika kita coba mengakses halaman `localhost/PWL_POS/public` maka akan tampil halaman awal untuk login ke aplikasi



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



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



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

Amati tahapan:

- Login akan mengarahkan user ke halaman dashboard jika berhasil.
 - Jika gagal, menampilkan pesan error melalui SweetAlert.
 -
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;
}
```



```
class UserModel extends Authenticatable
{
    use HasFactory;

    0 references
    protected $table = 'm_user';
    0 references
    protected $primaryKey = 'user_id';
    0 references
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];
    0 references
    protected $hidden = ['password'];
    0 references
    protected $casts = [
        'password' => 'hashed',
    ];

    /**
     * Relasi ke tabel level
     */
    0 references | 0 overrides
    public function level(): BelongsTo
    {
        return $this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id');
    }

    /**
     * Mendapatkan nama role dari relasi level
     */
    0 references | 0 overrides
    public function getRoleName(): string
    {
        return $this->level ? $this->level->level_nama : '-';
    }

    /**
     * Cek apakah user memiliki role tertentu
     */
    0 references | 0 overrides
    public function hasRole($role): bool
    {
        return $this->level && $this->level->level_kode == $role;
    }
}
```

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

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

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

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 }
```

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

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

0 references | 0 implementations
class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  \Closure $next
     * @param  string $role
     * @return \Symfony\Component\HttpFoundation\Response
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next, $role = ''): Response
    {
        $user = $request->user(); // Ambil user yang sedang login

        if ($user && $user->hasRole($role)) {
            return $next($request); // Jika punya role, lanjutkan request
        }

        // Jika tidak, tampilkan error 403
        abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

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,

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,
```



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>

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

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



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

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

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

    // route Kategori
```

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.

```
// Group route yang hanya bisa diakses setelah login
Route::middleware(middleware: ['auth', 'authorize:ADM'])->group(callback: function (): void {
    Route::prefix(prefix: 'level')->group(callback: function (): void {
        Route::get(uri: '/', action: [LevelController::class, 'index']);
        Route::post(uri: '/list', action: [LevelController::class, 'list']);
        Route::get(uri: '/create', action: [LevelController::class, 'create']);
        Route::post(uri: '/', action: [LevelController::class, 'store']);

        // Ajax Tambah
        Route::get(uri: '/create_ajax', action: [LevelController::class, 'create_ajax']);
        Route::post(uri: '/ajax', action: [LevelController::class, 'store_ajax']);

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

        // Ajax Update
        Route::get(uri: '/{id}/edit_ajax', action: [LevelController::class, 'edit_ajax']);
        Route::put(uri: '/{id}/update_ajax', action: [LevelController::class, 'update_ajax']);

        // Ajax Delete
        Route::get(uri: '/{id}/delete_ajax', action: [LevelController::class, 'confirm_ajax']);
        Route::delete(uri: '/{id}/delete_ajax', action: [LevelController::class, 'delete_ajax']);

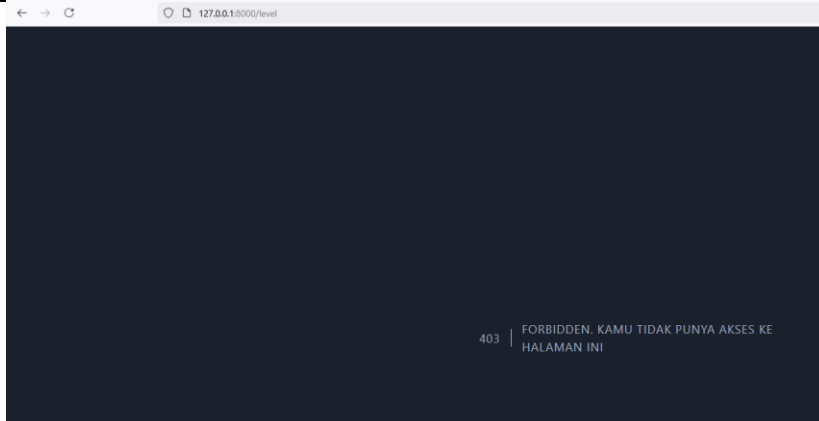
        // Non-Ajax Delete
        Route::delete(uri: '/{id}', action: [LevelController::class, 'destroy']);
    });
});
```

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

The image shows a screenshot of the AdminLTE login interface. It features a white login box on a light blue background. The box has the title "AdminLTE" at the top, followed by the text "Sign in to start your session". Below this, there are two input fields: one for the username with the value "manager" and another for the password represented by dots. To the right of the password field is a lock icon. Below the password field is a checkbox labeled "Remember Me". At the bottom right of the login box is a blue button labeled "Sign In".



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>



Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?
cara menerapkan authorization di Laravel menggunakan middleware untuk membatasi akses user berdasarkan role tertentu, seperti admin.
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;
}
```

```
public function getRoleName(): string
{
    return $this->level ? $this->level->level_nama : '-';
}

/**
 * Cek apakah user memiliki role tertentu
 */
0 references | 0 overrides
public function hasRole($role): bool
{
    return $this->level && $this->level->level_kode == $role;
}

/**
 * Mendapatkan kode role user
 */
0 references | 0 overrides
public function getRole(): string
{
    return $this->level ? $this->level->level_kode : '-';
}
```

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 }
```



```
<?php

namespace App\Http\Middleware;

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

1 reference | 0 implementations | You, 9 seconds ago | 1 author (You)
class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @param mixed ...$roles
     * @return \Symfony\Component\HttpFoundation\Response
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next, ...$roles): Response
    {
        $user = $request->user();

        // Jika tidak ada role yang diberikan, tolak akses
        if (!$user || empty($roles)) {
            abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini.');
        }

        // Ambil role user (level_code) dan periksa apakah termasuk role yang diizinkan
        if (in_array($user->getRole(), $roles)) {
            return $next($request);
        }

        // Jika tidak memiliki role yang sesuai, tampilkan error 403
        abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini.');
    }
}
```

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
});

Route::middleware(middleware: ['auth', 'authorize:ADM,MNG'])->group(callback: function (): void {
    Route::prefix(prefix: 'barang')->group(callback: function (): void {
        Route::get(uri: '/', action: [BarangController::class, 'index']);
        Route::post(uri: '/list', action: [BarangController::class, 'list']);
        Route::get(uri: '/create', action: [BarangController::class, 'create']);
        Route::post(uri: '/', action: [BarangController::class, 'store']);

        // Ajax Create
        Route::get(uri: '/create_ajax', action: [BarangController::class, 'create_ajax']);
        Route::post(uri: '/ajax', action: [BarangController::class, 'store_ajax']);

        // Show
        Route::get(uri: '/{id}', action: [BarangController::class, 'show']);

        // Edit & Update
        Route::get(uri: '/{id}/edit', action: [BarangController::class, 'edit']);
        Route::put(uri: '/{id}', action: [BarangController::class, 'update']);

        // Ajax Edit & Update
        Route::get(uri: '/{id}/edit_ajax', action: [BarangController::class, 'edit_ajax']);
        Route::put(uri: '/{id}/update_ajax', action: [BarangController::class, 'update_ajax']);

        // Ajax Delete
        Route::get(uri: '/{id}/delete_ajax', action: [BarangController::class, 'confirm_ajax']);
        Route::delete(uri: '/{id}/delete_ajax', action: [BarangController::class, 'delete_ajax']);

        // Delete
        Route::delete(uri: '/{id}', action: [BarangController::class, 'destroy']);
    });
});
```

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



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

AdminLTE

Sign in to start your session

☐ Remember Me

Sign In

✓

Berhasil

Login Berhasil

OK

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Barang

Data Supplier

Data Supplier

Data Transaksi

Stok Barang

Transaksi Penjualan

Daftar Barang

Daftar barang yang terdaftar dalam sistem

Filter: Semua

Kategori Barang

Show 10 entries

Search:

ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
1	BRG001	Barang 1	6935	1523	Kesehatan	Detail Edit Hapus
2	BRG002	Barang 2	2633	8220	Elektronik	Detail Edit Hapus
3	BRG003	Barang 3	8646	8958	Fashion	Detail Edit Hapus
4	BRG004	Barang 4	9699	9847	Fashion	Detail Edit Hapus
5	BRG005	Barang 5	9224	6106	Makanan	Detail Edit Hapus
6	BRG006	Barang 6	7318	1658	Kesehatan	Detail Edit Hapus
7	BRG007	Barang 7	4795	11953	Kesehatan	Detail Edit Hapus
8	BRG008	Barang 8	1301	2340	Makanan	Detail Edit Hapus
9	BRG0010	Barang 10	2982	8088	Fashion	Detail Edit Hapus

Showing 1 to 9 of 9 entries

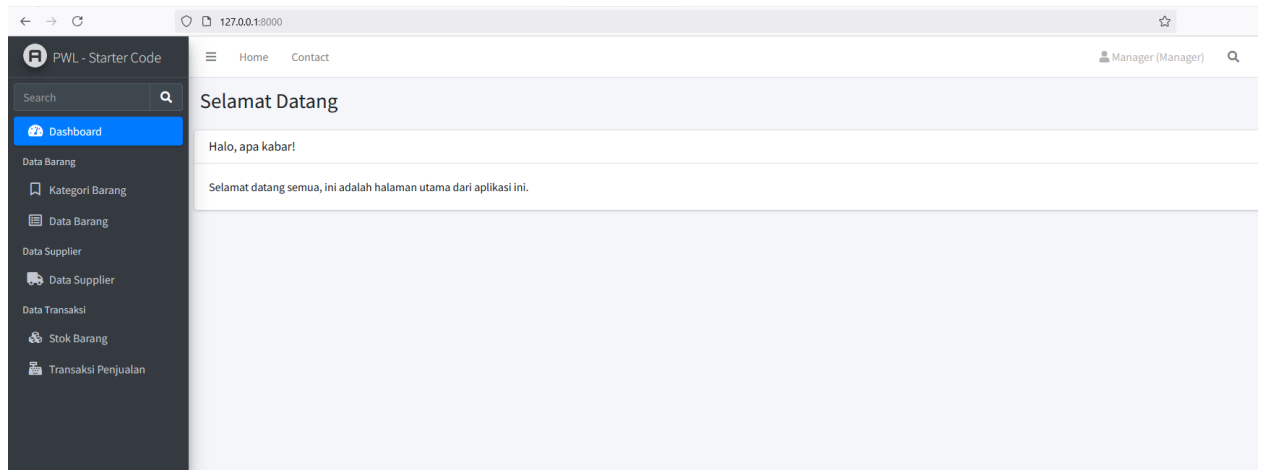
Page



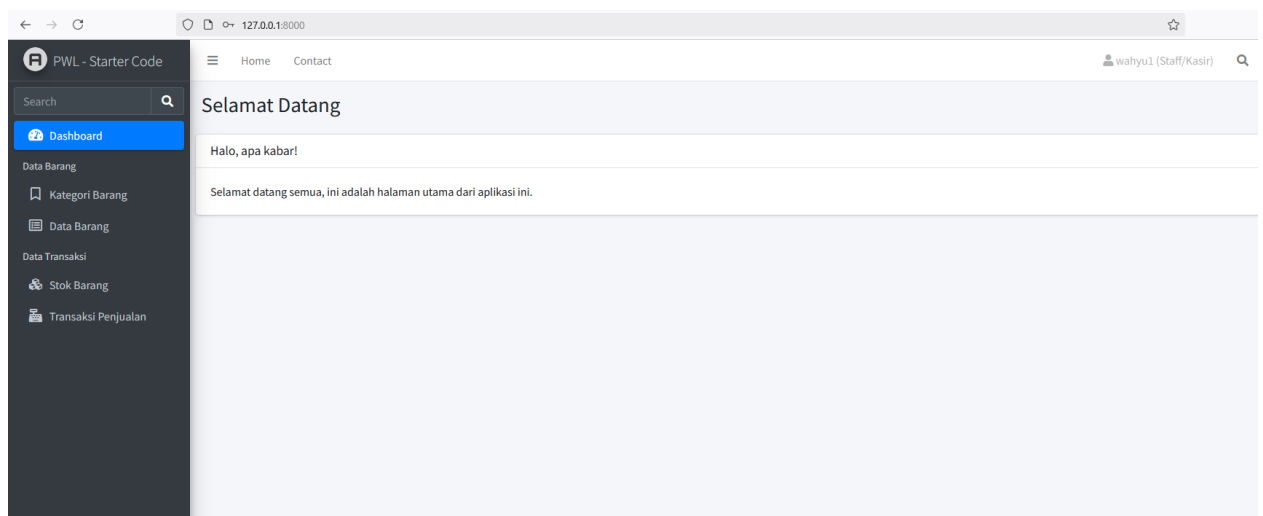
Tugas 3 – Implementasi Multi-Level Authorization :

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
 2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
- **Buat middleware AuthorizeUser**
 - Mengecek level_kode dari user yang login.
 - **Tambahkan method getRole() dan hasRole() di UserModel**
 - Untuk mengetahui dan membandingkan peran pengguna.
 - **Register middleware di Kernel.php**
 - Tambahkan alias 'authorize' => AuthorizeUser::class
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User

MANAGER



KASIR/STAFF





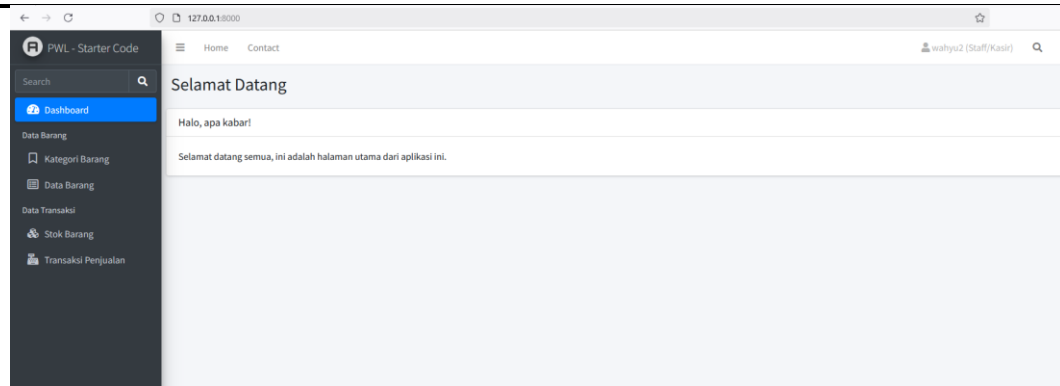
4. Submit kode untuk impementasi Authorization pada repository github kalian.

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.
2. Screenshot hasil yang kalian kerjakan
3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian



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>



- Menambahkan route register untuk menampilkan dan menyimpan data registrasi.
- Membuat method register() dan storeRegister() di AuthController.
- Mengatur agar level pengguna yang ditampilkan hanya STF (Staff) dan CUS (Customer).
- Membuat tampilan register.blade.php menggunakan template serupa halaman login.
- Validasi data input (username unik, password konfirmasi, dsb).
- Data berhasil disimpan menggunakan UserModel::create() dan diarahkan ke halaman login.

Pengguna bisa mendaftar dengan memilih level Staff atau Customer. Admin dan Manager disembunyikan agar hanya bisa dibuat oleh admin.

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