

Pemrograman Web Lanjut

Jobsheet 7 - Authentication dan Authorization di Laravel

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.



Nama : Athallah Ayudya Paramesti
NIM : 2341760061
Program Studi : D-IV Sistem Informasi Bisnis

Jurusan Teknologi Informasi
Politeknik Negeri Malang
2025

Praktikum 1: Implementasi Authentication

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

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

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

```
'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 App\Models\LevelModel;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable  
  
class UserModel extends Authenticatable  
{  
    use HasFactory;  
  
    protected $table = 'm_user';  
    protected $primaryKey = 'user_id';  
    /**  
     * The attributes that are mass assignable.  
     *  
     * @var array  
     */  
    protected $fillable = ['level_id', 'username', 'nama', 'password'];  
    protected $hidden = ['password']; // jangan ditampilkan saat select  
    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash  
  
    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

```
λ php artisan make:controller AuthController  
  
Warning: PHP Startup: Unable to load dynamic library 'C:\Windows\ext\php_sqlsrv.dll' (The specified module could not be found)  
Warning: PHP Startup: Unable to load dynamic library 'C:\Windows\ext\pdo_sqlsrv.dll' (The specified module could not be found)  
  
INFO Controller [C:\laragon\www\laravel\PWL_POS\app\Http\Controllers\AuthController.php] created successfully.
```

```

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         if(Auth::check()){ // Jika sudah login, maka redirect ke halaman home}
12             return redirect('/');
13         }
14         return view('auth.login');
15     }
16
17     public function postLogin(Request $request){
18         if($request->ajax() || $request->wantsJson()){
19             $credentials = $request->only('username', 'password');
20
21             if(Auth::attempt($credentials)){
22                 return Response()->json([
23                     'status' => true,
24                     'message' => 'Login Berhasil',
25                     'redirect' => url('/')
26                 ]);
27             }
28
29             return response()->json([
30                 'status' => false,
31                 'message' => 'Login Gagal'
32             ]);
33         }
34         return redirect('login');
35     }
36
37     public function logout(Request $request){
38         Auth::logout();
39
40         $request->session()->invalidate();
41         $request->session()->regenerateToken();
42         return redirect('login');
43     }
44 }

```

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**)

```

k_ / Jobsheet / resources / views / auth / login.blade.php / html
<!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=fallb
ack">
  <!-- 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>

```

```

        </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>
<!-- /.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() {

```

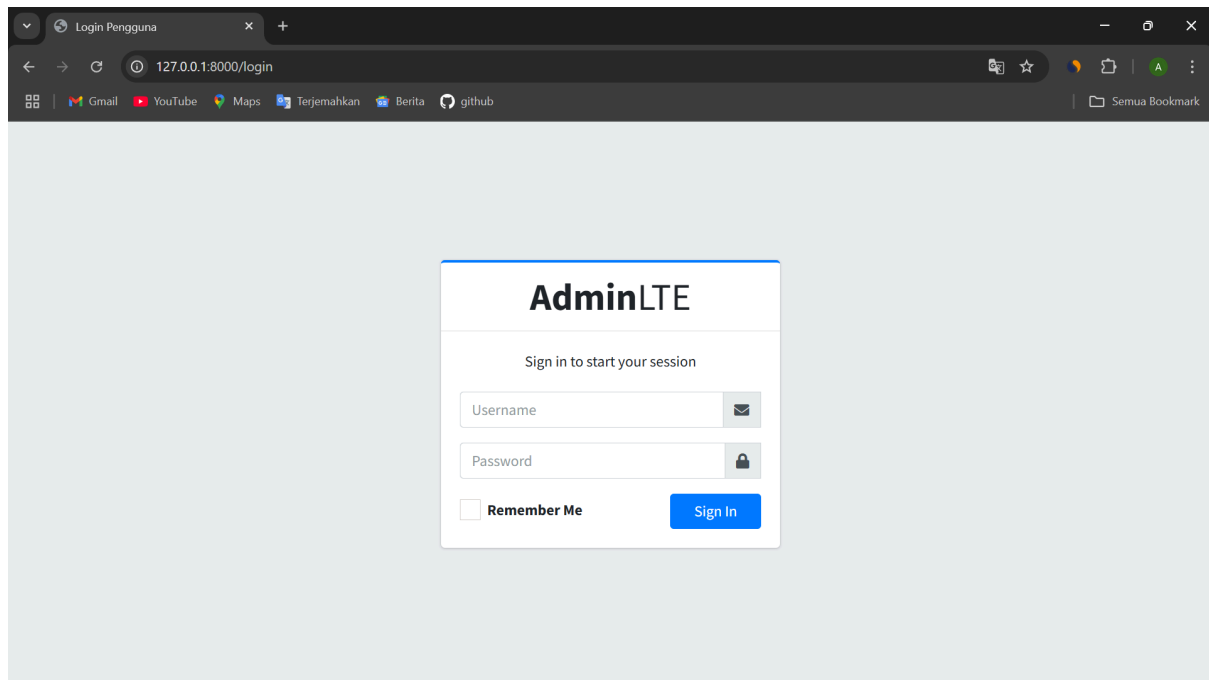
5. Kemudian kita modifikasi `route/web.php` agar semua route masuk dalam auth

```

web.php M X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\AuthController;
4  use App\Http\Controllers\UserController;
5  use App\Http\Controllers\KategoriController;
6  use App\Http\Controllers\BarangController;
7  use App\Http\Controllers\LevelController;
8  use App\Http\Controllers>WelcomeController;
9  use App\Http\Controllers\SupplierController;
10 use Illuminate\Support\Facades\Route;
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() { //artinya semua route di dalam group ini harus login dulu
19
20 });
21

```

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



Tugas 1

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

Praktikum 2: Implementasi Authorization di Laravel dengan Middleware

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

```
public function level(): BelongsTo{
    return $this->belongsTo(LevelModel::class, 'level_id', 'level_id')
}

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

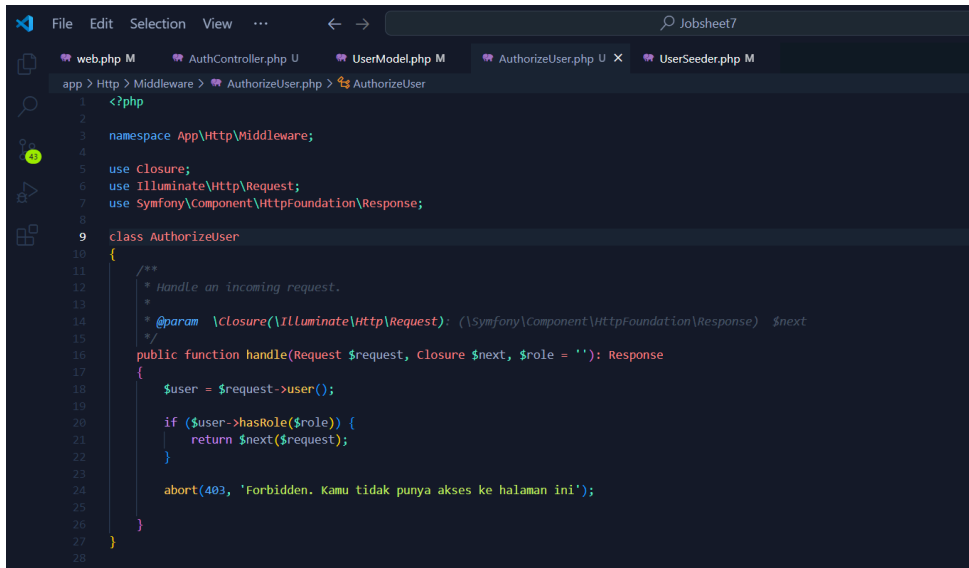
// Cek apakah user memiliki role tertentu
public function hasRole($role) : bool {
    return $this->level->level_kode == $role;
}
```

2. Kemudian kita buat middleware dengan nama `AuthorizeUser.php`. Kita bisa buat middleware dengan mengetikkan perintah pada terminal/CMD
File `middleware` akan dibuat di `app/Http/Middleware/AuthorizeUser.php`

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

```
INFO Middleware [C:\laragon\www\la
thorizeUser.php] created successfully.
```

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

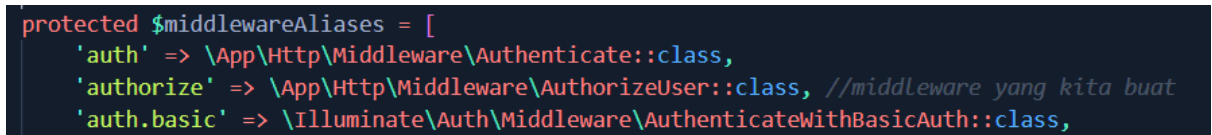


```

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, $role = ''): Response
17     {
18         $user = $request->user();
19
20         if ($user->hasRole($role)) {
21             return $next($request);
22         }
23
24         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
25     }
26 }
27
28

```

- 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 yang kita buat
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

```

- 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
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staff/Kasir	NULL	NULL
4	CUS	Customer	NULL	NULL

- 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



```

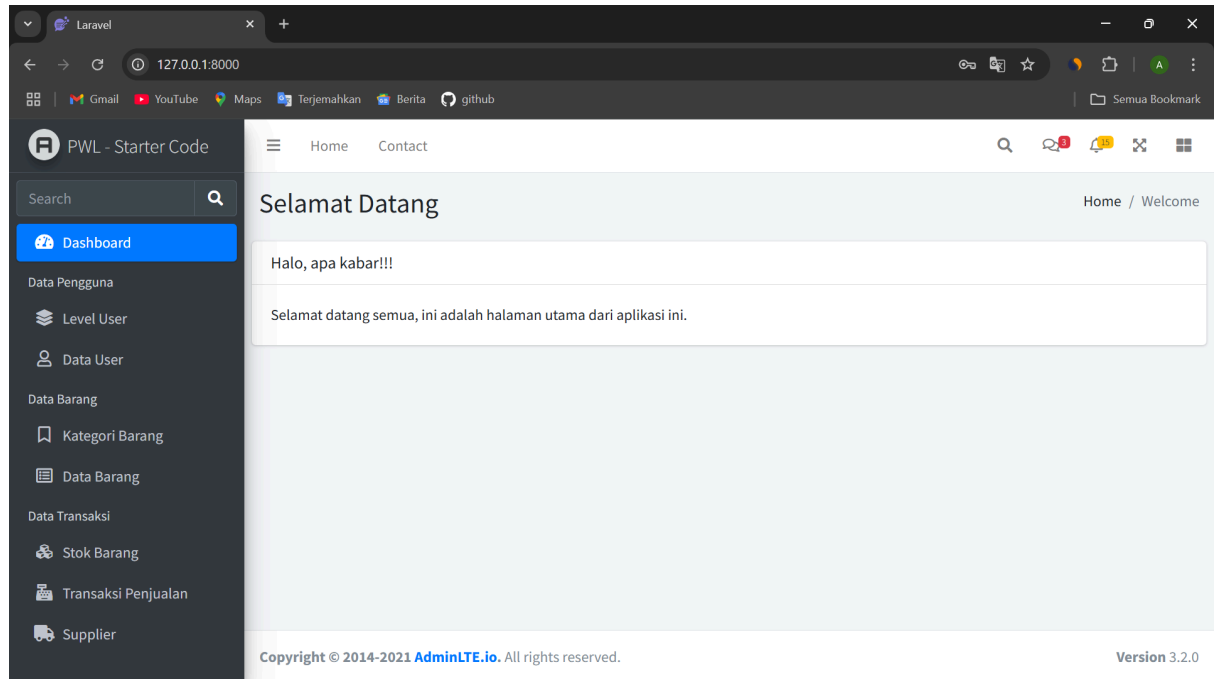
Route::middleware(['auth'])->group(function(){ //artinya semua route di dalam group ini harus login dulu
    Route::get('/', [WelcomeController::class, 'index']);
});
//route level

//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']); //List dattables
    Route::get('/level/create', [LevelController::class, 'create']);
    Route::post('/level', [LevelController::class, 'store']);
    Route::get('/level/{id}/edit', [LevelController::class, 'edit']); // tampilkan form edit
    Route::put('/level/{id}', [LevelController::class, 'update']); // tampilkan proses update
    Route::delete('/level/{id}', [LevelController::class, 'destroy']); // tampilkan proses hapus data
});

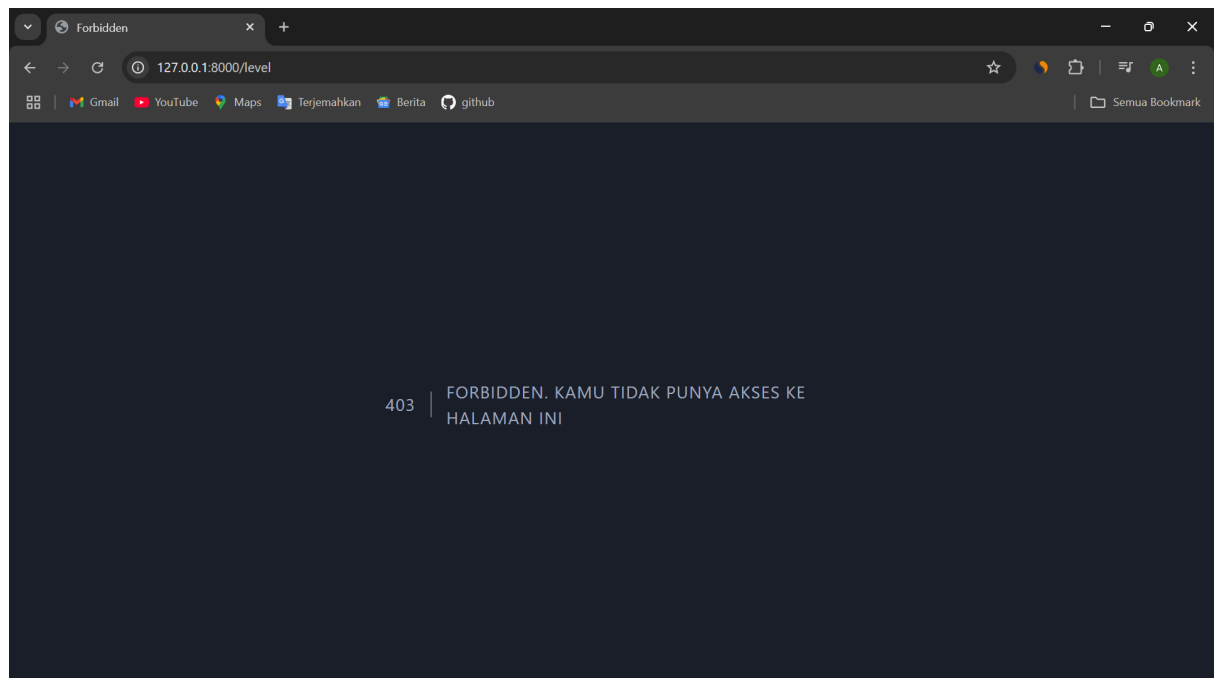
```

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



Ketika login menjadi staff, ketika memilih menu level user akan muncul seperti ini



Tugas 2

1. Apa yang kalian pahami pada praktikum 2 ini?

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

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

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

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

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

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

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

```
public function handle(Request $request, Closure $next, ... $role): Response
{
    $user_role = $request->user()->getRole();
    if (in_array($user_role, $role)) {
        return $next($request);
    }

    abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

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

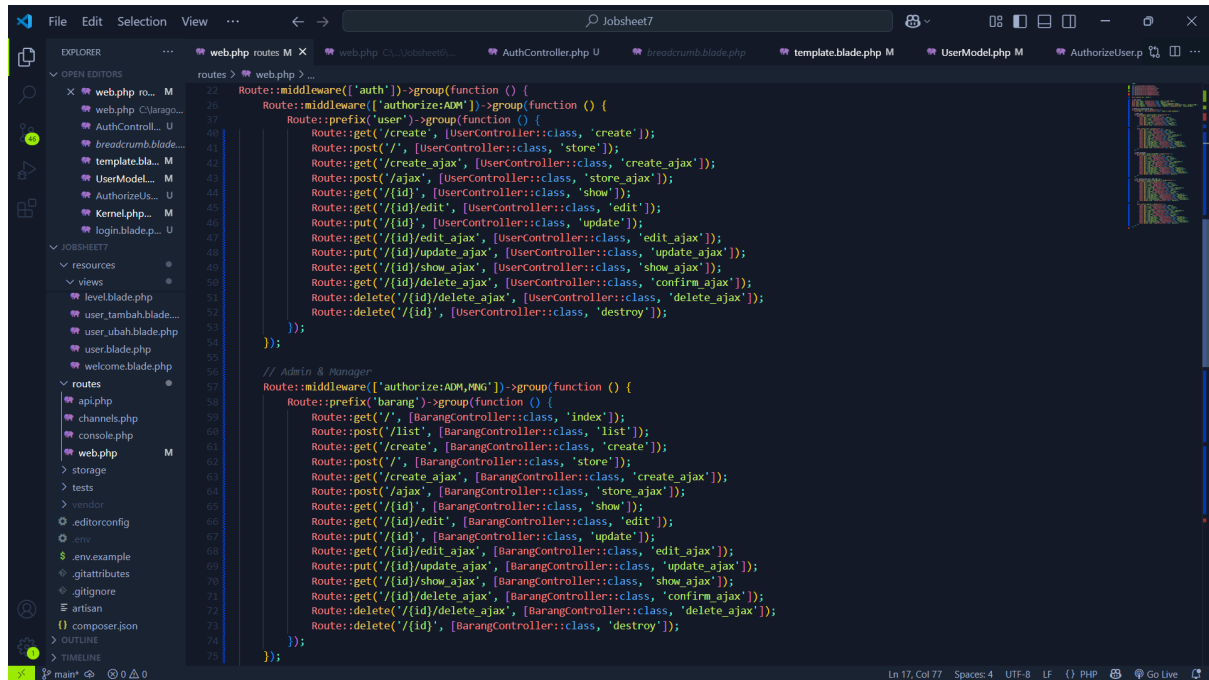
```
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']); //List dattables
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']);
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']);
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // tamp
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); //
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']);
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']);
});
```

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

Tugas 3

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan

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



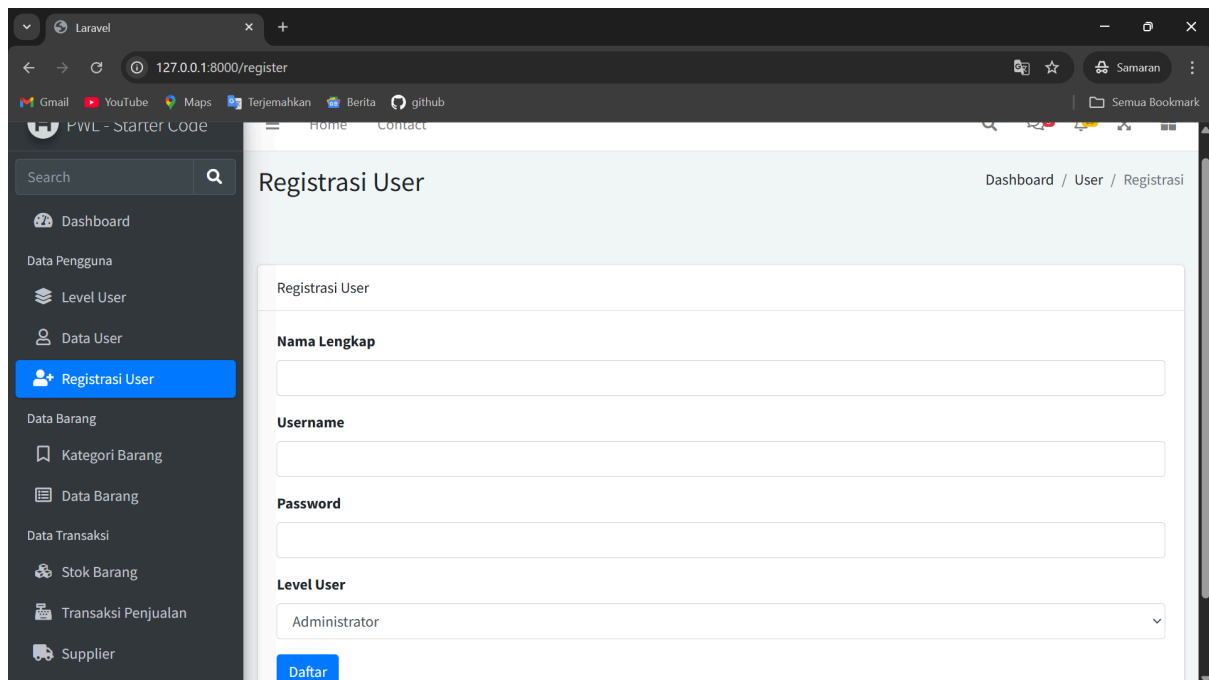
```
Route::middleware(['auth'])->group(function () {
    Route::middleware(['authorize:ADM'])->group(function () {
        Route::prefix('user')->group(function () {
            Route::get('/create', [UserController::class, 'create']);
            Route::post('/', [UserController::class, 'store']);
            Route::get('/create_ajax', [UserController::class, 'create_ajax']);
            Route::post('/ajax', [UserController::class, 'store_ajax']);
            Route::get('/{id}', [UserController::class, 'show']);
            Route::put('/{id}/edit', [UserController::class, 'edit']);
            Route::get('/{id}', [UserController::class, 'update']);
            Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']);
            Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']);
            Route::get('/{id}/show_ajax', [UserController::class, 'show_ajax']);
            Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']);
            Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']);
            Route::delete('/{id}', [UserController::class, 'destroy']);
        });
    });
});

// Admin & Manager
Route::middleware(['authorize:ADM,MNG'])->group(function () {
    Route::prefix('barang')->group(function () {
        Route::get('/', [BarangController::class, 'index']);
        Route::post('/list', [BarangController::class, 'list']);
        Route::get('/create', [BarangController::class, 'create']);
        Route::post('/', [BarangController::class, 'store']);
        Route::get('/create_ajax', [BarangController::class, 'create_ajax']);
        Route::post('/ajax', [BarangController::class, 'store_ajax']);
        Route::get('/{id}', [BarangController::class, 'show']);
        Route::get('/{id}/edit', [BarangController::class, 'edit']);
        Route::put('/{id}', [BarangController::class, 'update']);
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']);
        Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax']);
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
        Route::delete('/{id}', [BarangController::class, 'destroy']);
    });
});
```

- Submit kode untuk implementasi Authorization pada repository github kalian.

Tugas 4

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



Registrasi User

Nama Lengkap

Username

Password

Level User

Administrator

Daftar