

LAPORAN JOBSHEET 7
PEMROGRAMAN WEB LANJUT
Authentication dan Authorization di Laravel

Dosen pengajar: Dimas Wahyu Wibowo, ST., MT



Disusun Oleh:

Nama : M. Fatih Al Ghifary

NIM : 2341720194

Kelas : TI-2A

No : 16

PRODI TEKNIK INFORMATIKA D-IV
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\UserModel::class,  
    ],  
],
```

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

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
  
55 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'];  
    0 references  
    protected $hidden = ['password'];  
    0 references  
    protected $casts = ['password' => 'hashed'];  
  
    0 references | 0 overrides  
    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;

0 references | 0 implementations
class AuthController extends Controller
{
    0 references | 0 overrides
    public function login(): Factory|Redirector|RedirectResponse|View
    {
        if (auth()->check()) {
            return redirect(to: '/');
        }
        return view(view: 'auth.login');
    }
    0 references | 0 overrides
    public function postlogin(Request $request): JsonResponse|mixed|Redirector|RedirectRes...
    {
        if ($request->ajax() || $request->wantsJson()) {
            $credentials = $request->only('username', 'password');
            if (Auth::attempt($credentials)) {
                return response()->json([
                    'status' => true,
                    'message' => 'Login Berhasil',
                    'redirect' => url(path: '/')
                ]);
            }
            return response()->json([
                '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(path: 'adminlte/plugins/fontawesome-free/css/all.min.css') }}">
  <!-- icheck bootstrap -->
  <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
  <!-- SweetAlert2 -->
  <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
  <!-- Theme style -->
  <link rel="stylesheet" href="{{ asset(path: 'adminlte/dist/css/adminlte.min.css') }}">
</head>

<body class="hold-transition login-page">
  <div class="login-box">
    <!-- /.login-logo -->
    <div class="card card-outline card-primary">
      <div class="card-header text-center"><a href="{{ url(path: '/') }}" class="h1"><b>Admin</b></a></div>
      <div class="card-body">
        <p class="login-box-msg">Sign in to start your session</p>
        <form action="{{ url(path: 'auth/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="checkbox-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>
    </div>
  </div>

```

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

```
use App\Http\Controllers\AuthController;
```

```

// Auth
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');

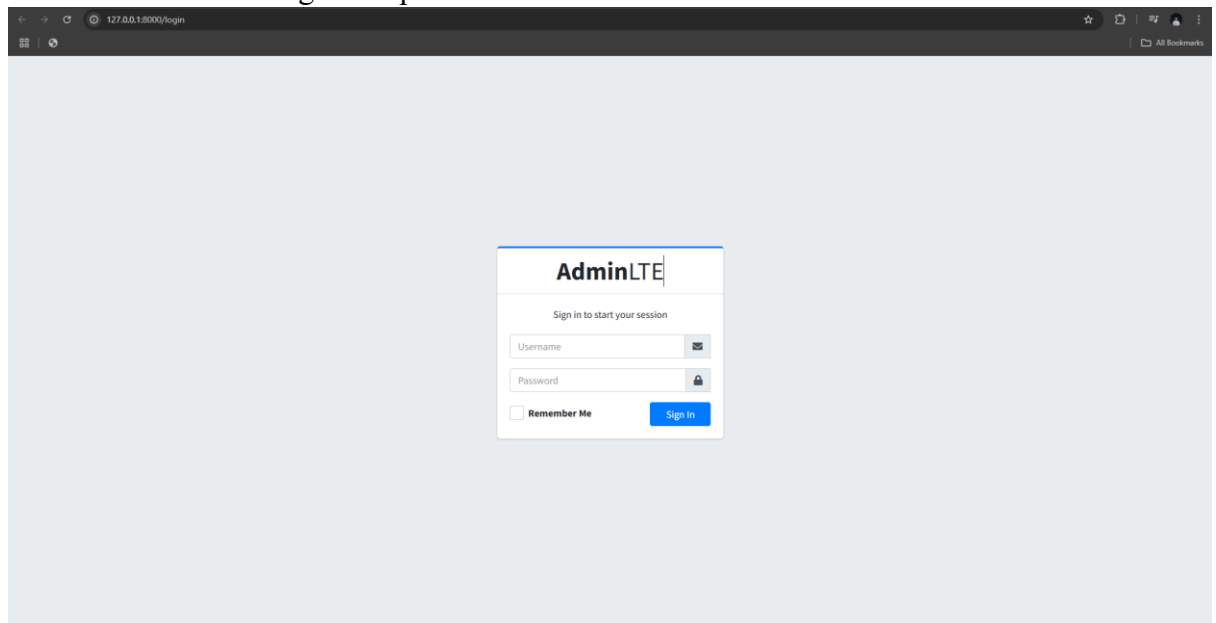
Route::middleware(['auth'])->group(function () { //artinya semua route di dalam group ini harus login dulu

    // masukkan semua route yang perlu autentikasi disini
    Route::get('/', [WelcomeController::class, 'index']);

});

```

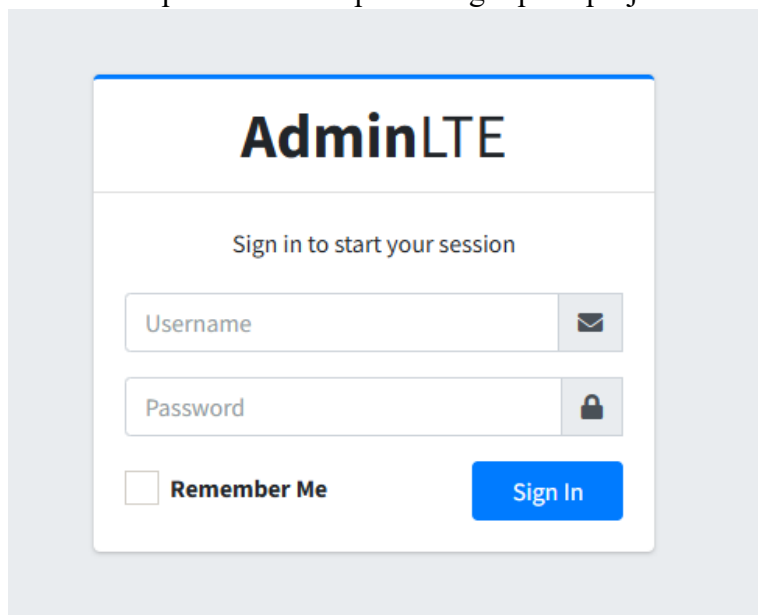
6. Ketika kita coba mengakses halaman localhost/PWL_POS/public makan akan tampil halaman awal untuk login ke aplikasi



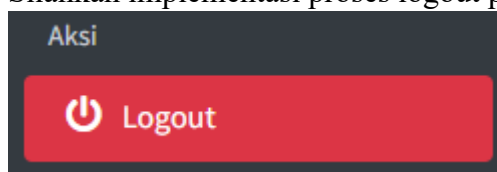
7. Commit
<https://github.com/fateehhh/PROWEBLNJT/commit/ae975d286d49af348b85b8e9b512d7700b9328ad>

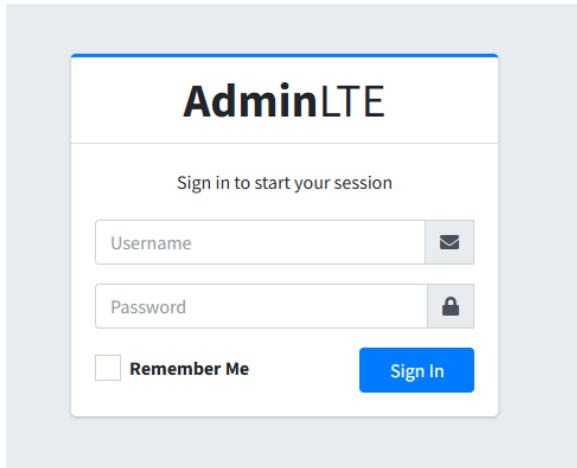
TUGAS 1 – Implementasi Authentication :

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
Jawab: Pada proses login, pengguna terlebih dahulu membuka halaman /login untuk menampilkan form login. Setelah mengisi email dan password, data akan dikirim ke server menggunakan metode POST. Selanjutnya, Laravel memverifikasi kredensial tersebut dengan Auth::attempt(). Jika berhasil, pengguna diarahkan ke halaman utama. Sebaliknya, jika login gagal, akan ditampilkan pesan kesalahan. Untuk logout, saat pengguna memilih opsi keluar, Laravel akan menjalankan Auth::logout() yang bertugas menghapus sesi dan mengembalikan pengguna ke halaman login.
4. Submit kode untuk impementasi Authentication pada repository github kalian.
<https://github.com/fateehhh/PROWEBLNJT/commit/ebc86d7bf33a3261643bfe1edd07f9a46c108dfb>

Praktikum 2 – Implementasi Authorizaton di Laravel dengan Middleware

1. Modifikasi UserModel.php

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

// Cek apakah user memiliki role tertentu
0 references | 0 overrides
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

```
D:\KULIAH\SEMESTER-4\file\laragon\www\PROWEBLNJT\Week 7\PWL_POS>php artisan make:middleware AuthorizeUser

INFO Middleware [D:\KULIAH\SEMESTER-4\file\laragon\www\PROWEBLNJT\Week 7\PWL_POS\app\Http\Middleware\AuthorizeUser.p
hp] created successfully.
```

```

<?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 \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next): Response
    {
        return $next($request);
    }
}

```

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

```

<?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 \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next, $role = ''): Response
    {
        $user = $request->user();

        if ($user->hasRole($role)) {
            return $next($request);
        }
        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,

```

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

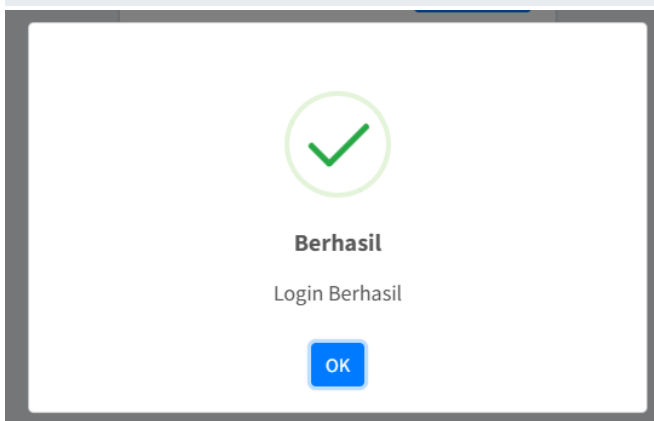
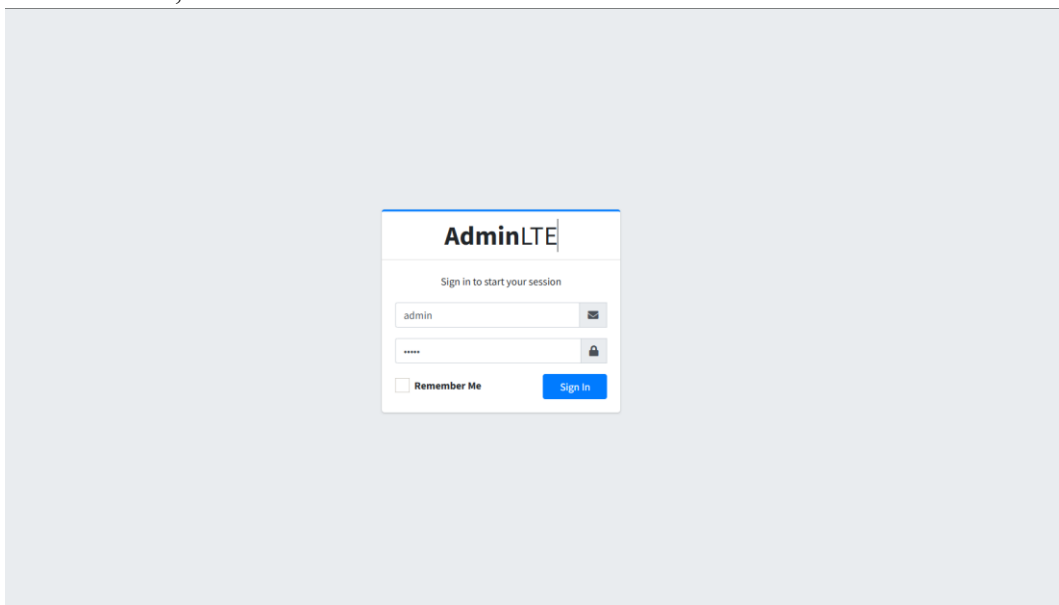
```
Route::middleware(['auth'])->group(function (): void { //artinya semua route di dalam group ini harus login dulu

    // masukkan semua route yang perlu autentikasi disini
    Route::get('/', [WelcomeController::class, 'index']);

    //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']); //untuk form tampilan edit
        Route::put('/level/{id}', [LevelController::class, 'update']); //untuk proses update data
        Route::delete('/level/{id}', [LevelController::class, 'destroy']); //untuk proses delete 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



Tugas 2 – Implementasi Authorization

1. Apa yang kalian pahami pada praktikum 2 ini?

Jawab: saya memahami bagaimana proses *authorization* bekerja di Laravel, yaitu untuk membatasi akses pengguna berdasarkan peran atau role tertentu. Authorization dilakukan dengan cara membuat *middleware* khusus (AuthorizeUser) yang akan memeriksa apakah pengguna memiliki role tertentu sebelum mengakses halaman. Selain itu, model UserModel dimodifikasi agar dapat membaca relasi ke tabel level dan memeriksa peran pengguna secara langsung. Dengan mekanisme ini, hanya pengguna dengan role yang sesuai yang bisa mengakses halaman tertentu, sedangkan lainnya akan ditolak dengan kode error 403 (Forbidden).

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

Jawab:

- Pertama, memodifikasi file UserModel.php untuk menambahkan relasi level() agar bisa mengambil data dari tabel level, lalu membuat fungsi getRoleName() untuk mengambil nama role, dan fungsi hasRole(\$role) untuk memeriksa apakah pengguna memiliki role yang sesuai.
- Kedua, membuat file middleware baru bernama AuthorizeUser.php. Di dalam middleware ini, kami mengambil user dari request, lalu mengecek apakah role-nya sesuai dengan yang diminta. Jika cocok, request dilanjutkan (\$next(\$request)), jika tidak, proses dihentikan dan Laravel mengembalikan error 403 dengan pesan “Kamu tidak punya akses ke halaman ini.”
- Middleware ini nantinya bisa digunakan di route atau controller yang ingin dibatasi aksesnya berdasarkan role, misalnya hanya admin atau petugas tertentu saja yang bisa mengaksesnya.

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

<https://github.com/fatechhh/PROWEBLNJT/commit/94740a01377644253e2f5ab784a441f9e3a44255>

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()

```
0 references | 0 overrides
public function getRoleName(): string
{
    return $this->level->level_nama;
}

0 references | 0 overrides
public function hasRole($role): bool
{
    return $this->level->level_kode == $role;
}

0 references | 0 overrides
public function getRole(): mixed
{
    return $this->level->level_kode;
}
```

2. Modifikasi middleware AuthorizeUser.php

```
public function handle(Request $request, Closure $next, ... $roles): Response
{
    $user_role = $request->user()->getRole(); //digunakan untuk mengambil data level_kode dari u
    if (in_array(needle: $user_role, haystack: $roles)) //memeriksa apakah level_kode user saat login ada
    {
        return $next($request); //jika ada, maka jalankan $next
    }
    abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

3. Perbaiki route/web.php sesuaikan dengan role/level yang diinginkan

```
//artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manajer)
Route::middleware(['authorize:ADM,MNG'])->group(function (): void {
    Route::get('/barang', [BarangController::class, 'index']);
    Route::post('/barang/list', [BarangController::class, 'list']);
    Route::get('/barang/create', [BarangController::class, 'create']);
    Route::post('/barang/', [BarangController::class, 'store']);
    Route::get('/barang/create_ajax', [BarangController::class, 'create_ajax']);
    Route::post('/barang/ajax', [BarangController::class, 'store_ajax']);
    Route::get('/barang/{id}', [BarangController::class, 'show']);
    Route::get('/barang/{id}/show_ajax', [BarangController::class, 'show_ajax']);
    Route::get('/barang/{id}/edit', [BarangController::class, 'edit']);
    Route::put('/barang/{id}', [BarangController::class, 'update']);
    Route::get('/barang/{id}/edit_ajax', [BarangController::class, 'edit_ajax']);
    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']);
    Route::delete('/barang/{id}', [BarangController::class, 'destroy']);
});
```

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

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 jawab:

- Pertama, menambahkan kode pada UserModel.php

```
public function getRole (): mixed
{
    return $this->level->level_kode;
}
```

Method getRole() akan mengembalikan kode dari user berdasarkan relasi ke tabel level. Misalnya, kalau user punya level_kode = 'admin', maka saat memanggil \$user->getRole(), hasilnya adalah 'admin'.

- Kedua, memodifikasi AuthorizeUser.php

```
public function handle(Request $request, Closure $next, ... $roles): Response
{
    $user_role = $request->user()->getRole(); //digunakan untuk mengambil data level_kode dari u
    if (in_array(needle: $user_role, haystack: $roles)) //memeriksa apakah level_kode user saat login ada
    {
        return $next($request); //jika ada, maka jalankan $next
    }
    abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

dengan perubahan/modifikasi AuthorizeUser dengan menggunakan parameter variadic ...\$roles membuatnya bisa mengecek lebih dari satu role (multi-role)

- Ketiga, memindahkan route barang agar bisa diakses oleh ADM (admin) dan MNG (Manajer)
- Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu yang sesuai dengan Level/Jenis User
 - User

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

- Level (sudah ada sebelumnya)

```
//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']); //untuk form tampilan edit
    Route::put('/level/{id}', [LevelController::class, 'update']); //untuk proses update data
    Route::delete('/level/{id}', [LevelController::class, 'destroy']); //untuk proses delete data
});
```

- Barang (sudah ada sebelumnya)

```
//barang
//artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manajer)
Route::middleware(['authorize:ADM,MNG'])->group(function (): void {
    Route::get('/barang', [BarangController::class, 'index']);
    Route::post('/barang/list', [BarangController::class, 'list']);
    Route::get('/barang/create', [BarangController::class, 'create']);
    Route::post('/barang/', [BarangController::class, 'store']);
    Route::get('/barang/create_ajax', [BarangController::class, 'create_ajax']);
    Route::post('/barang/ajax', [BarangController::class, 'store_ajax']);
    Route::get('/barang/{id}', [BarangController::class, 'show']);
    Route::get('/barang/{id}/show_ajax', [BarangController::class, 'show_ajax']);
    Route::get('/barang/{id}/edit', [BarangController::class, 'edit']);
    Route::put('/barang/{id}', [BarangController::class, 'update']);
    Route::get('/barang/{id}/edit_ajax', [BarangController::class, 'edit_ajax']);
    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']);
    Route::delete('/barang/{id}', [BarangController::class, 'destroy']);
});
```

- Kategori

```
//kategori
//artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manajer)
Route::middleware(['authorize:ADM,MNG'])->group(function (): void {
    Route::get('/kategori/', [KategoriController::class, 'index']);
    Route::post('/kategori/list', [KategoriController::class, 'list']);
    Route::get('/kategori/create', [KategoriController::class, 'create']);
    Route::post('/kategori/', [KategoriController::class, 'store']);
    Route::get('/kategori/create_ajax', [KategoriController::class, 'create_ajax']);
    Route::post('/kategori/ajax', [KategoriController::class, 'store_ajax']);
    Route::get('/kategori/{id}', [KategoriController::class, 'show']);
    Route::get('/kategori/{id}/show_ajax', [KategoriController::class, 'show_ajax']);
    Route::get('/kategori/{id}/edit', [KategoriController::class, 'edit']);
    Route::put('/kategori/{id}', [KategoriController::class, 'update']);
    Route::get('/kategori/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']);
    Route::put('/kategori/{id}/update_ajax', [KategoriController::class, 'update_ajax']);
    Route::get('/kategori/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']);
    Route::delete('/kategori/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']);
    Route::delete('/kategori/{id}', [KategoriController::class, 'destroy']);
});
```

- Supplier

```
//supplier
//artinya semua route di dalam group ini harus punya role ADM (Administrator), MNG (Manajer), dan SUP (Supplier)
Route::middleware(['authorize:ADM,MNG,SUP'])->group(function (): void {
    Route::get('/supplier/', [SupplierController::class, 'index']);
    Route::post('/supplier/list', [SupplierController::class, 'list']);
    Route::get('/supplier/create', [SupplierController::class, 'create']);
    Route::post('/supplier/', [SupplierController::class, 'store']);
    Route::get('/supplier/create_ajax', [SupplierController::class, 'create_ajax']);
    Route::post('/supplier/ajax', [SupplierController::class, 'store_ajax']);
    Route::get('/supplier/{id}', [SupplierController::class, 'show']);
    Route::get('/supplier/{id}/show_ajax', [SupplierController::class, 'show_ajax']);
    Route::get('/supplier/{id}/edit', [SupplierController::class, 'edit']);
    Route::put('/supplier/{id}', [SupplierController::class, 'update']);
    Route::get('/supplier/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']);
    Route::put('/supplier/{id}/update_ajax', [SupplierController::class, 'update_ajax']);
    Route::get('/supplier/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']);
    Route::delete('/supplier/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']);
    Route::delete('/supplier/{id}', [SupplierController::class, 'destroy']);
});
```

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

<https://github.com/fatechhh/PROWEBLNJT/commit/ea66722ec6431020c726e6e889c50b6bec33aa39>

Tugas 4 – Implementasi Form Registrasi

1. Silahkan implementasikan form untuk registrasi user.
2. Screenshot hasil yang kalian kerjakan
 - Menambahkan function register() dan postregister() di AuthControll

```
public function register(): Factory|View
{
    $level = LevelModel::select('level_id', 'level_nama')->get();
    return view(view: 'auth.register', data: ['level' => $level]);
}

1 reference | 0 overrides
public function postregister(Request $request): JsonResponse|mixed
{
    if ($request->ajax() || $request->wantsJson()) {
        $rules = [
            'level_id' => 'required|integer',
            'username' => 'required|string|min:3|unique:m_user,username',
            'nama' => 'required|string|max:100',
            'password' => 'required|min:5|confirmed'
        ];

        $validator = Validator::make($request->all(), $rules);

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

        // Enkripsi password pakai Hash
        $request['password'] = Hash::make($request->password);

        UserModel::create($request->all());

        return response()->json([
            'status' => true,
            'message' => 'Data user berhasil disimpan',
            'redirect' => url(path: '/login'),
        ]);
    }
}
```

- Menambahkan shortcut di halaman login

```
<div class="mt-3 text-center">
    <a href="{{ url(path: 'register') }}">Belum punya akun? Daftar di sini</a>
</div>
```

- Membuat register.blade.php untuk memuat tampilan form registrasi

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Registrasi 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(path: 'adminlte/plugins/fontawesome-free/css/all.min.css') }}">

    <!-- iCheck bootstrap -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/ichack-bootstrap/ichack-bootstrap.min.css') }}">

    <!-- SweetAlert2 -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">

    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/dist/css/adminlte.min.css') }}">
</head>

<body class="hold-transition login-page">
    <div class="login-box">
        <div class="card card-outline card-primary">
            <div class="card-header text-center">
                <a href="{{ url(path: '/') }}" class="h1"><b>Admin</b>LTE</a>
            </div>
            <div class="card-body">
                <p class="login-box-msg">Register akun baru</p>

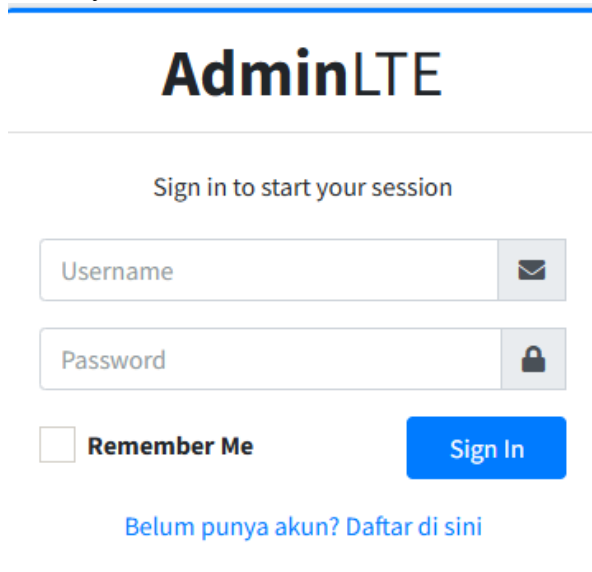
                <form action="{{ url(path: '/register') }}" method="POST" id="form-register">
                    @csrf
                    <div class="form-group">
                        <label>Level Pengguna</label>
                        <select name="level_id" class="form-control">
                            <option value="">- Pilih Level -</option>
                            @foreach($level as $l)
                                <option value="{{ $l->level_id }}">{{ $l->level_nama }}</option>
                            @endforeach
                        </select>
                        <small id="error-level_id" class="text-danger error-text"></small>
                    </div>

                    <div class="form-group">
                        <label>Username</label>
                        <input name="username" class="form-control">
                        <small id="error-username" class="text-danger error-text"></small>
                    </div>

                    <div class="form-group">
                        <label>Nama</label>
                        <input name="nama" class="form-control">
                        <small id="error-nama" class="text-danger error-text"></small>
                    </div>

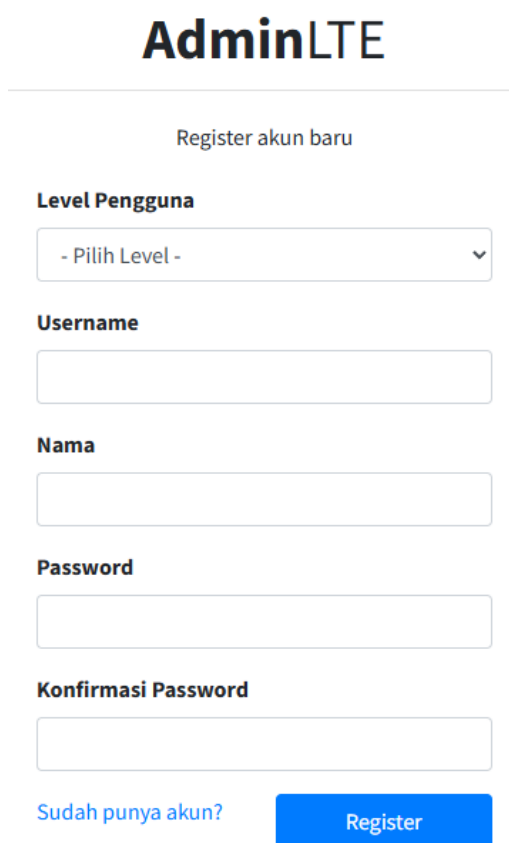
                    <div class="form-group">
                        <label>Password</label>
                        <input name="password" type="password" class="form-control">
                    </div>
                </form>
            </div>
        </div>
    </div>
```

- Hasilnya



The image shows the 'AdminLTE' sign-in interface. It features a header with the 'AdminLTE' logo. Below the header, the text 'Sign in to start your session' is displayed. There are two input fields: 'Username' with an envelope icon and 'Password' with a lock icon. Below these fields is a 'Remember Me' checkbox and a blue 'Sign In' button. At the bottom, there is a link that says 'Belum punya akun? Daftar di sini'.

Ketika klik tulisan biru maka akan di direct ke form registrasi



The image shows the 'AdminLTE' registration interface. It features a header with the 'AdminLTE' logo. Below the header, the text 'Register akun baru' is displayed. There are several input fields: a dropdown menu for 'Level Pengguna' with the text '- Pilih Level -', a 'Username' field, a 'Nama' field, a 'Password' field, and a 'Konfirmasi Password' field. At the bottom, there is a link that says 'Sudah punya akun?' and a blue 'Register' button.

Mencoba menambahkan user baru

AdminLTE

Register akun baru

Level Pengguna

Manager

Username

kaka

Nama

kaka

Password

.....

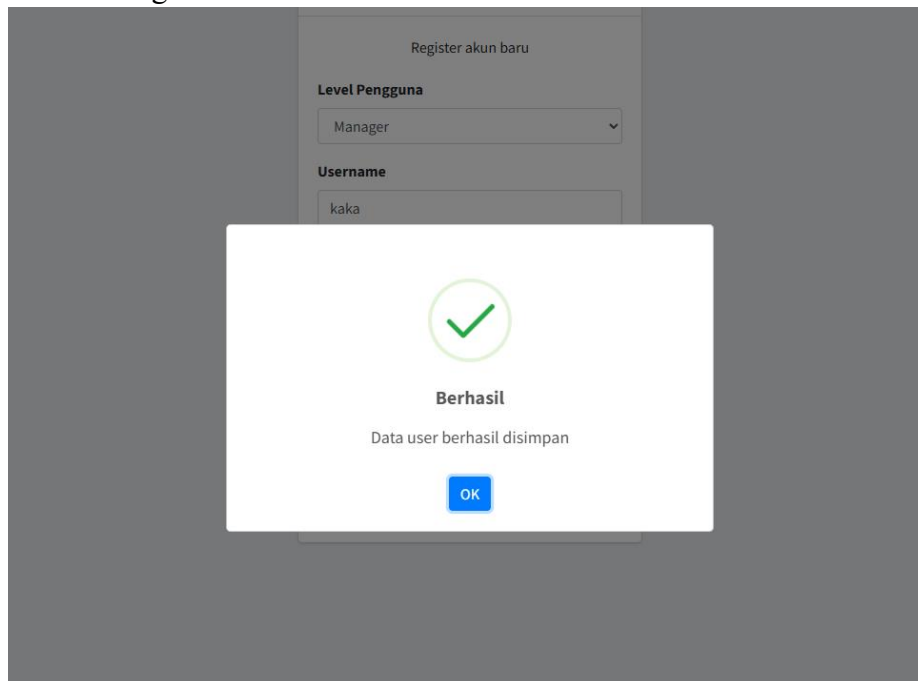
Konfirmasi Password

.....

[Sudah punya akun?](#)

Registrasi

ketika di registrasi maka muncul notifikasi berhasil



kita cek di dalam tabel user untuk memastikan berhasil menambah usernya

Daftar User Home / User

Daftar user yang terdaftar dalam sistem Tambah Tambah Aja

Filter:
 Level Pengguna

Show entries Search:

ID	Username	Nama	Level Pengguna	Aksi
11	joko	joko	Manager	Detail Edit Hapus
12	kaka	kaka	Manager	Detail Edit Hapus

Showing 11 to 12 of 12 entries Previous 1 2 Next

3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian
<https://github.com/fatechhh/PROWEBLNJT/commit/6630efb70078af8c89864ccc5ba6ae3dcf5fe07d>