Jurusan Teknologi Informasi Politeknik Negeri Malang
JOBSHEET 07 Authentication dan Authorization di Laravel
Mata Kuliah Pemrograman Web Lanjut (PWL)

Nama	:	Candra Ahmad Dani
Nim	:	2341720187
Kelas	:	TI-2A

Praktikum 1 – Implementasi Authentication

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

58 references | 0 implementations
class UserModel extends Model
{
    use HasFactory;

    0 references
    protected $table = 'm_user';
    0 references
    protected $primaryKey = 'user_id';
    // protected $fillable = ['level_id', 'username', 'nama'];
    0 references
    protected $fillable = ['level_id', 'username', 'nama', 'password'];
    0 references
    protected $hidden = ['password']; // jangan ditampilkan saat di select
    0 references
    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
    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
<?php

namespace App\Http\Controllers;

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

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

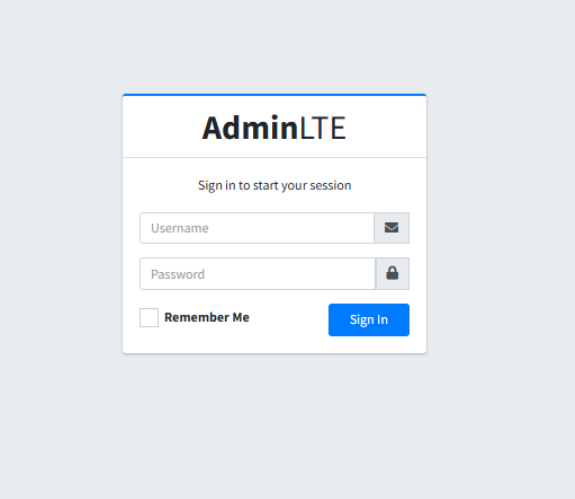
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <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('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
    <!-- icheck bootstrap -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
    <!-- SweetAlert2 -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('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('/') }}" 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>
                        </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('adminlte/plugins/jquery/jquery.min.js') }}"></script>
    <!-- Bootstrap 4 -->
    <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
    <!-- jquery-validation -->
    <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/jquery-validation/additional-methods.min.js') }}"></script>
    <!-- SweetAlert2 -->
    <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
    <!-- AdminLTE App -->
    <script src="{{ asset('adminlte/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: 5,
                        maxlength: 20
                    }
                },
                submitHandler: function(form) { // ketika valid, maka bagian yg akan dijalankan
                    let formData = $(form).serialize(); // Ambil data form

                    $.ajax({
                        url: form.action,
                        type: form.method,
                        data: $(form).serialize(),
                        success: function(response) {
                            if (response.status) { // jika sukses
                                Swal.fire({
                                    icon: 'success',
                                    title: 'Berhasil',
                                    text: response.message,
                                }).then(function() {
                                    window.location = response.redirect;
                                });
                            } else { // jika error
                                $('.error-text').text('');
                                $.each(response.msgField, function(prefix, val) {
                                    $('#error-' + prefix).text(val[0]);
                                });
                                Swal.fire({
                                    icon: 'error',
                                    title: 'Terjadi Kesalahan',
                                    text: response.message
                                });
                            }
                        }
                    });
                    return false;
                },
                errorElement: 'span',
                errorPlacement: function(error, element) {
                    error.addClass('invalid-feedback');
                    element.closest('.input-group').append(error);
                },
                highlight: function(element, errorClass, validClass) {
                    $(element).addClass('is-invalid');
                },
                unhighlight: function(element, errorClass, validClass) {
                    $(element).removeClass('is-invalid');
                }
            });
        });
    </script>
</body>
</html>
```

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

```php
<?php

use App\Http\Controllers\AuthController;
use App\Http\Controllers\KategoriController;
use App\Http\Controllers\BarangController;
use App\Http\Controllers\SupplierController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
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(): void{ //artinya semua route di dalam group ini harus login dulu

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


});
```

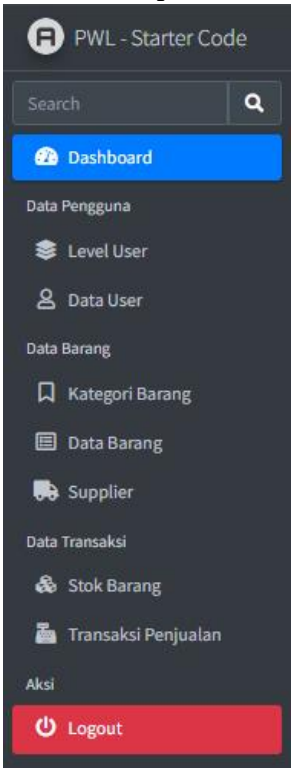
6. Ketika kita coba mengakses halaman localhost/PWL_POS/public makan akan tampil halaman awal untuk login ke aplikasi



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
4. Submit kode untuk impementasi Authentication pada repository github kalian

Praktikum 2 – Implementasi Authorizaton di Laravel dengan Middleware

1. Kita modifikasi UserModel.php dengan menambahkan kode berikut

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

// 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
php artisan make:middleware AuthorizeUser

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

```php
<?php

namespace App\Http\Middleware;

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

1 reference | 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 = $request->user(); //ambil data dari user yang login
        //fungsi user diambil dari userModel.php
        if ($user->hasRole($role)) { // cek apakah user punya role yang diinginkan
            return $next($request);
        }
        // jika tidak punya role, maka 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

```php
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,
```

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
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staff/Kasir	NULL	NULL
4	PEL	Pelanggan	2025-03-06 17:26:58	2025-03-15 16:12:22

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

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

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

Tugas 2 – Implementasi Authoriization

1. Apa yang kalian pahami pada praktikum 2 ini?
   membuat login dan authorization untuk admin
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Submit kode untuk impementasi 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()

```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;
    }

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

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

```php
<?php

namespace App\Http\Middleware;

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

1 reference | 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, ... $roles): Response
    {
        // $user = $request->user(); //ambil data dari user yang login
        //                           //fungsi user diambil dari userModel.php
        // if ($user->hasRole($role)) { // cek apakah user punya role yang diinginkan
        //     return $next($request);
        // }
        $user_role =$request->user()->getRole(); // ambil data level_kode dari user yang login
        if (in_array(needle: $user_role, haystack: $roles)) { // cek apakah level_kode user ada di dalam array roles
            return $next($request); // jika ada, maka lanjutkan request
        }
        // jika tidak punya role, maka 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

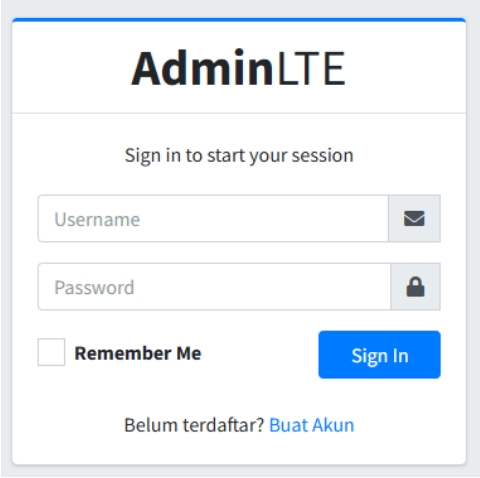
```php
Route::middleware(['authorize:ADM,MNG'])->group(function (): void {
    Route::group(['prefix' => 'barang'], function (): void {
        Route::get('/', [BarangController::class, 'index']);
        Route::post('/list', [BarangController::class, 'list']);
        Route::get('/create', [BarangController::class, '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}/show_ajax', [BarangController::class, 'show_ajax']);
        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}/delete_ajax', [BarangController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
        Route::delete('/{id}', [BarangController::class, 'destroy']);
    });
});
```

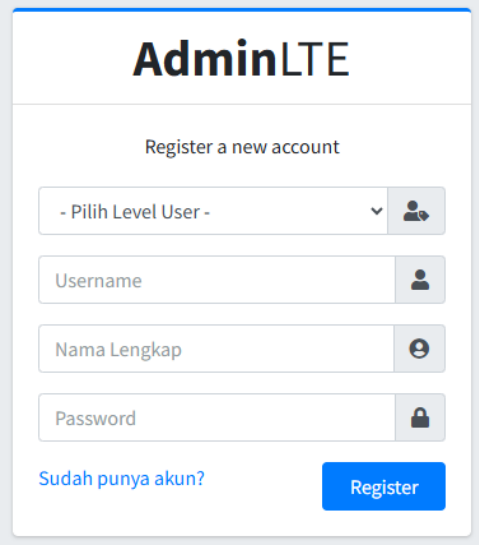
Tugas 3 – Implementasi Multi-Level Authorization

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User
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