Nama : Adinda Ivanka Maysanda Putru

Kelas : SIB 2B

NIM : 2341760058

JOBSHEET 7

Authentication dan Authorization di Laravel

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\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');
    }
}
```

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

```
<?php

namespace App\Http\Controllers;

use App\Models\LevelModel;
use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

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');
    }
}
```

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=fallb
ack">
    <!-- 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 -->
                <!-- Tambahan: link ke halaman registrasi -->
                <div class="text-center mt-3">
                    <p>Belum punya akun? <a href="{{ url('register') }}">Daftar di sini</a></p>
                </div>
            </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>
```

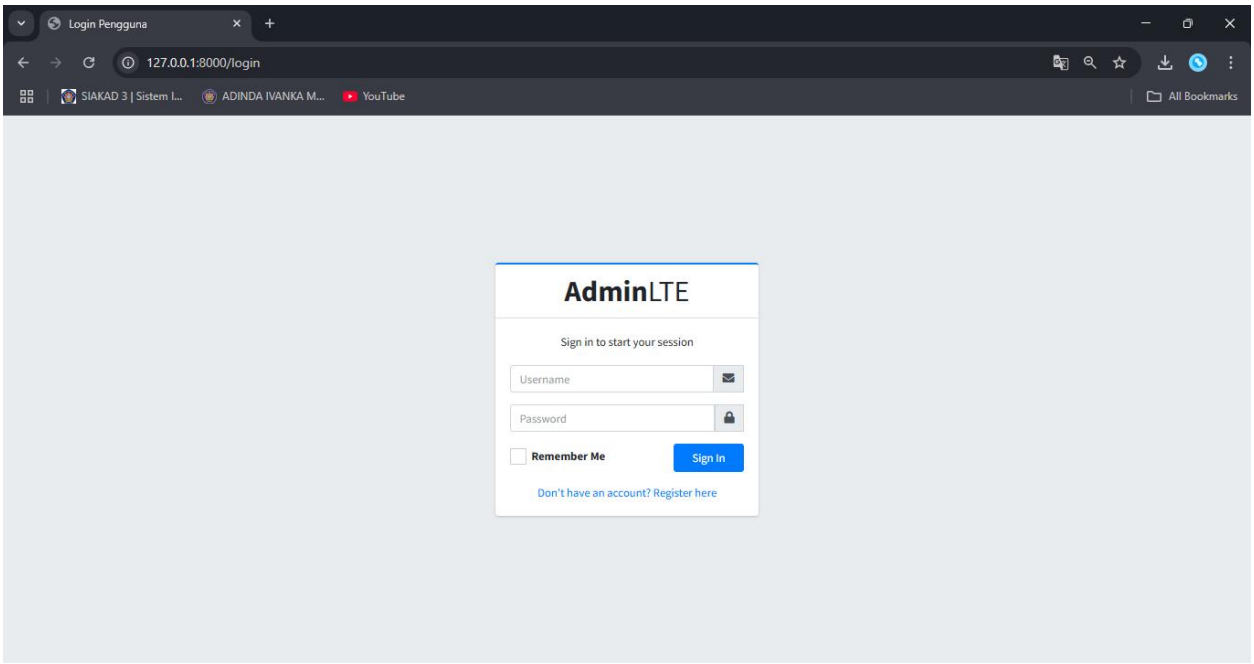
```
<!-- AdminLTE App -->
<script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
<script>
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    });
    $(document).ready(function () {
        $("#form-login").validate({
            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
                $.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

```
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

});
```

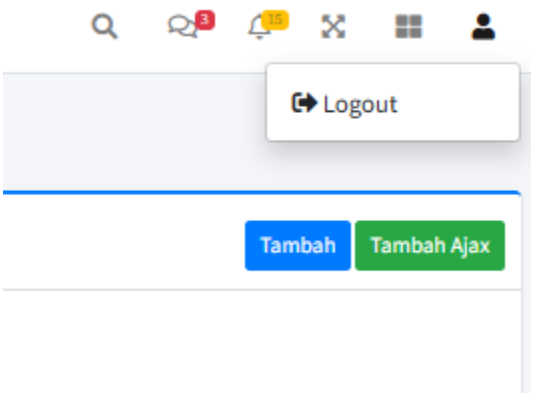
6. Ketika kita coba mengakses halaman localhost/PWL_POS/public makan akan tampil halaman awal untuk login ke aplikasi



Tugas 1 – Implementasi Authentication :

---

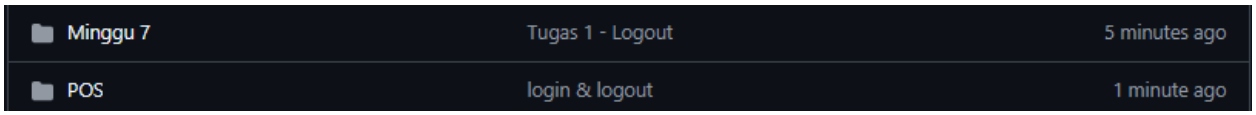
1. Silahkan implementasikan proses login pada project kalian masing-masing
2. Silahkan implementasi proses logout pada halaman web yang kalian buat




```
<!-- Profile Dropdown Menu -->
<li class="nav-item dropdown">
  <a class="nav-link" data-toggle="dropdown" href="#">
    <i class="fas fa-user"></i> <!-- Ikon profil -->
  </a>
  <div class="dropdown-menu dropdown-menu-right">
    <a class="dropdown-item" href="#" onclick="event.preventDefault(); document.getElementById('logout-form').submit();">
      <i class="fas fa-sign-out-alt"></i> Logout <!-- Ikon logout -->
    </a>
    <form id="logout-form" action="{{ url('logout') }}" method="POST" style="display: none;">
```

```
        @csrf
      </form>
    </div>
</li>
```

3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
    - Untuk mengimplementasikan proses autentikasi sebelum mengakses situs web, langkah pertama adalah konfigurasi autentikasi yang sesuai dengan struktur basis data yang ada, yang dilakukan melalui model UserModel. Selanjutnya, dibuat sebuah controller yang berfungsi untuk menangani logika autentikasi, yang kemudian dihubungkan dengan tampilan antarmuka pengguna, yaitu login.blade. Setelah itu, rute autentikasi perlu didefinisikan, dan rute-rute untuk menu yang memerlukan autentikasi ditempatkan dalam grup rute yang dilindungi. Untuk fungsionalitas logout, cukup menambahkan menu atau tombol logout pada antarmuka halaman web, lalu membuat fungsi logout di controller yang sama dengan fungsi login, serta mendefinisikan rute yang sesuai.
4. Submit kode untuk impementasi Authentication pada repository github kalian.



Praktikum 2 – Implementasi Authorizaton 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

```php
<?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');
    }

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

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

```
php artisan make:middleware AuthorizeUser

  INFO  Middleware [C:\laragon\www\Pemrograman-Web-Lanjut\Minggu7\app\Http\Middleware\AuthorizeUser.php] created
successfully.
```

4. 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;

class AuthorizeUser
```

```
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next, ...$roles): Response
    {
        $user_role = $request->user()->getRole();  // ambil data level_kode dari user yg login
        if (in_array($user_role, $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(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

5. 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, //yang kita buat
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
        'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
        'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
        'signed' => \App\Http\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
    ];
```

6. 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
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL

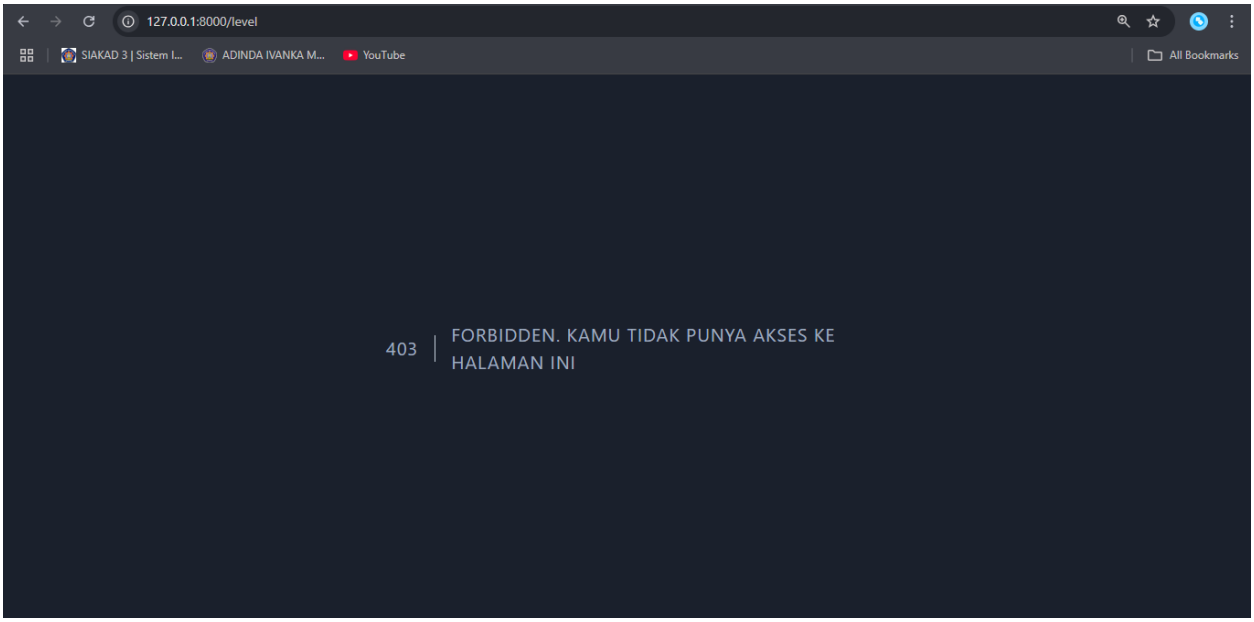
7. Untuk mencoba authorization yang telah kita buat, maka perlu kita modiifikasi route/web.php untuk menentukan route mana saja yang akan diberi hak akses sesuai dengan level user

```
Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'level'], function () {
        Route::get('/',[LevelController::class, 'index']);
        Route::post('/list',[LevelController::class, 'list']);
```

```
    Route::get('/create',[LevelController::class, 'create']);
    Route::post('/',[LevelController::class, 'store']);

    //route ajax
    Route::get('/create_ajax', [LevelController::class, 'create_ajax']);
    Route::post('/ajax', [LevelController::class, 'store_ajax']);
    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::get('/{id}',[LevelController::class, 'show']);
    Route::get('/{id}/edit',[LevelController::class, 'edit']);
    Route::put('/{id}',[LevelController::class, 'update']);
    Route::delete('/{id}',[LevelController::class, 'destroy']);
  });
});
```

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.

8. 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?
    - Pada praktikum 2, saya mempelajari implementasi otorisasi di Laravel yang bertujuan untuk membatasi akses pengguna berdasarkan peran atau tingkat akses mereka. Dengan memanfaatkan middleware seperti authorize, sistem secara otomatis menyaring akses sehingga hanya pengguna dengan hak yang sesuai yang dapat mengakses fitur-fitur tertentu. Hal ini sangat krusial untuk mengamankan aplikasi dan memastikan bahwa data sensitif hanya dapat diakses oleh pengguna yang memiliki wewenang.
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
    - Tahapan praktikum yang saya lakukan adalah sebagai berikut:
        - Modifikasi Middleware: Saya membuat middleware AuthorizeUser.php yang berfungsi untuk memeriksa apakah seorang pengguna memiliki tingkat akses tertentu. Middleware ini akan memeriksa peran pengguna melalui metode hasRole($role) dan meneruskan permintaan jika sesuai, atau menampilkan kesalahan 403 jika pengguna tidak memiliki hak akses yang diperlukan.
        - Registrasi Middleware: Middleware yang telah dibuat kemudian didaftarkan ke dalam app/Http/Kernel.php dengan memberikan alias authorize. Hal ini memungkinkan middleware tersebut untuk digunakan dalam penyaringan permintaan berdasarkan tingkat akses pengguna.
        - Pengaturan Rute: Rute-rute aplikasi diatur dengan menerapkan middleware authorize pada kelompok rute tertentu di routes/web.php. Langkah ini memastikan bahwa hanya pengguna dengan tingkat akses administrator yang dapat mengakses rute-rute sensitif, seperti manajemen pengguna dan pengaturan sistem.
3. Submit kode untuk impementasi Authorization pada repository github kalian.

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;
    }
```

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

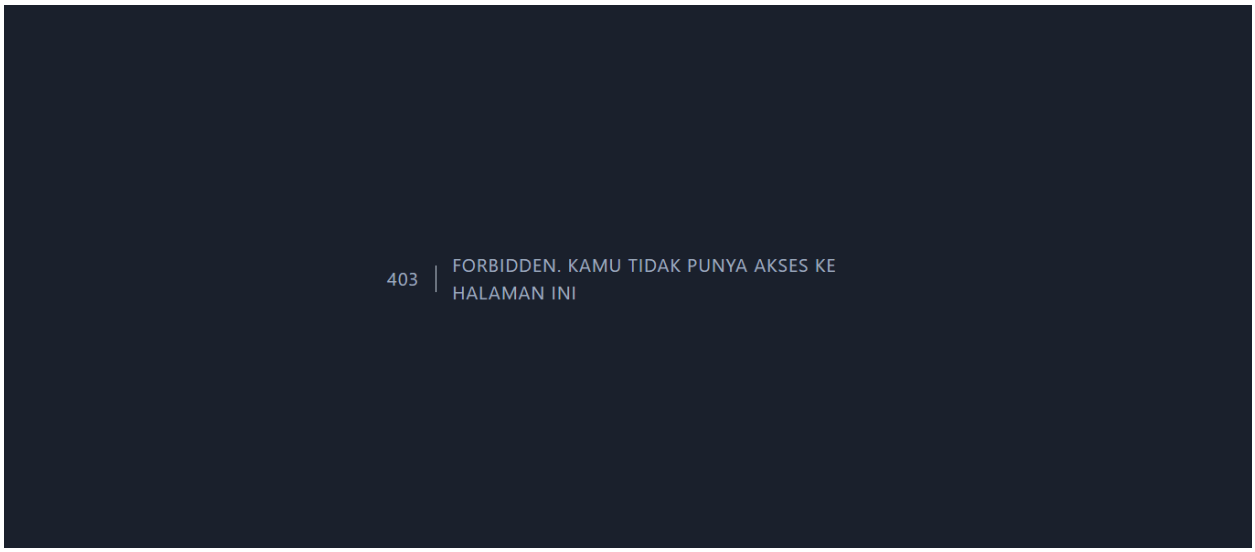
```
<?php

namespace App\Http\Middleware;

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

class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next, ...$roles): Response
    {
        $user_role = $request->user()->getRole();  // ambil data level_kode dari user yg login
        if (in_array($user_role, $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(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:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'level'], function () {
        Route::get('/',[LevelController::class, 'index']);
        Route::post('/list',[LevelController::class, 'list']);
        Route::get('/create',[LevelController::class, 'create']);
        Route::post('/',[LevelController::class, 'store']);

        //route ajax
```

```
    Route::get('/create_ajax', [LevelController::class, 'create_ajax']);
    Route::post('/ajax', [LevelController::class, 'store_ajax']);
    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::get('/{id}',[LevelController::class, 'show']);
    Route::get('/{id}/edit',[LevelController::class, 'edit']);
    Route::put('/{id}',[LevelController::class, 'update']);
    Route::delete('/{id}',[LevelController::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

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

```
    }
```

2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
    - Untuk mengimplementasikan otorisasi berbasis peran, pertama, tambahkan fungsi getRole di model UserModel untuk mengambil peran pengguna dari basis data. Kemudian, modifikasi middleware otorisasi untuk memeriksa peran pengguna menggunakan getRole dan menolak akses jika peran tidak sesuai. Terakhir, terapkan middleware otorisasi pada grup rute yang memerlukan otorisasi di routes/web.php atau routes/api.php, dan sesuaikan peran yang diizinkan untuk setiap grup rute.
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu menu yang sesuai dengan Level/Jenis User

```php
<?php

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

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {
    return view('welcome');
})->name('home');


Route::get('/level', [LevelController::class, 'index']);
// Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);

//praktikum 2.6
Route::get('/user/tambah', [UserController::class, 'tambah']);
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
```

```php
Route::get('user/ubah/{id}', [UserController::class, 'ubah']);
Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);
Route::get('/user/hapus/{id}', [UserController::class, 'hapus']);

// Route::get('/', [WelcomeController::class, 'index']);
//Jobsheet 7
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::post('logout', [AuthController::class, 'logout'])->middleware('auth');

Route::get('/register', [AuthController::class, 'register'])->name('register');
Route::post('/register', [AuthController::class, 'store_user'])->name('store_user');

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

Route::middleware(['authorize:ADMIN, MNG, STF'])->group(function () {
    Route::group(['prefix' => 'user'], function () {
        Route::get('/', [UserController::class, 'index']);          // menampilkan halaman awal user
        Route::post('/list', [UserController::class, 'list']);      // menampilkan data user dalam bentuk json untuk datatables
        Route::get('/create', [UserController::class, 'create']);    // menampilkan halaman form tambah user
        Route::post('/', [UserController::class, 'store']);         // menyimpan data user baru
        //route ajax
        Route::get('/create_ajax', [UserController::class, 'create_ajax']);  // Menampilkan halaman form tambah user Ajax
        Route::post('/ajax', [UserController::class, 'store_ajax']);        // Menyimpan data user baru ajax
        Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']);        // Menampilkan halaman form edit user Ajax
        Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']);    // Menyimpan perubahan data user Ajax
        Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // Untuk hapus data user Ajax

        Route::get('/{id}', [UserController::class, 'show']);         // menampilkan detail user
        Route::get('/{id}/edit', [UserController::class, 'edit']);    // menampilkan halaman form edit user
        Route::put('/{id}', [UserController::class, 'update']);      // menyimpan perubahan data user
        Route::delete('/{id}', [UserController::class, 'destroy']);  // menghapus data user
        Route::post('/data', [UserController::class, 'getUsers'])->name('user.data');
    });
});

Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'level'], function () {
        Route::get('/',[LevelController::class, 'index']);
        Route::post('/list',[LevelController::class, 'list']);
        Route::get('/create',[LevelController::class, 'create']);
        Route::post('/',[LevelController::class, 'store']);
```

```
        //route ajax
        Route::get('/create_ajax', [LevelController::class, 'create_ajax']);
        Route::post('/ajax', [LevelController::class, 'store_ajax']);
        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::get('/{id}',[LevelController::class, 'show']);
        Route::get('/{id}/edit',[LevelController::class, 'edit']);
        Route::put('/{id}',[LevelController::class, 'update']);
        Route::delete('/{id}',[LevelController::class, 'destroy']);
    });
});

Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'kategori'], function () {
        Route::get('/',[KategoriController::class, 'index']);
        Route::post('/list',[KategoriController::class, 'list']);
        Route::get('/create',[KategoriController::class, 'create']);
        Route::post('/',[KategoriController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [KategoriController::class, 'create_ajax']);
        Route::post('/ajax', [KategoriController::class, 'store_ajax']);
        Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']);
        Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']);

        Route::get('/{id}',[KategoriController::class, 'show']);
        Route::get('/{id}/edit',[KategoriController::class, 'edit']);
        Route::put('/{id}',[KategoriController::class, 'update']);
        Route::delete('/{id}',[KategoriController::class, 'destroy']);
    });
});

Route::middleware(['authorize:ADMIN, MNG'])->group(function () {
    Route::group(['prefix' => 'supplier'], function () {
        Route::get('/',[SupplierController::class, 'index']);
        Route::post('/list',[SupplierController::class, 'list']);
        Route::get('/create',[SupplierController::class, 'create']);
        Route::post('/',[SupplierController::class, 'store']);

        //route ajax
        Route::get('/create_ajax', [SupplierController::class, 'create_ajax']);
        Route::post('/ajax', [SupplierController::class, 'store_ajax']);
```

```
    Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']);
    Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']);
    Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']);
    Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']);

    Route::get('/{id}',[SupplierController::class, 'show']);
    Route::get('/{id}/edit',[SupplierController::class, 'edit']);
    Route::put('/{id}',[SupplierController::class, 'update']);
    Route::delete('/{id}',[SupplierController::class, 'destroy']);
  });
});

Route::middleware(['authorize:ADMIN, STF'])->group(function () {
  Route::group(['prefix' => 'barang'], function () {
    Route::get('/',[BarangController::class, 'index']);
    Route::post('/list',[BarangController::class, 'list']);
    Route::get('/create',[BarangController::class, 'create']);
    Route::post('/',[BarangController::class, 'store']);

    //route ajax
    Route::get('/create_ajax', [BarangController::class, 'create_ajax']);
    Route::post('/ajax', [BarangController::class, 'store_ajax']);
    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::get('/{id}',[BarangController::class, 'show']);
    Route::get('/{id}/edit',[BarangController::class, 'edit']);
    Route::put('/{id}',[BarangController::class, 'update']);
    Route::delete('/{id}',[BarangController::class, 'destroy']);
  });
});
```

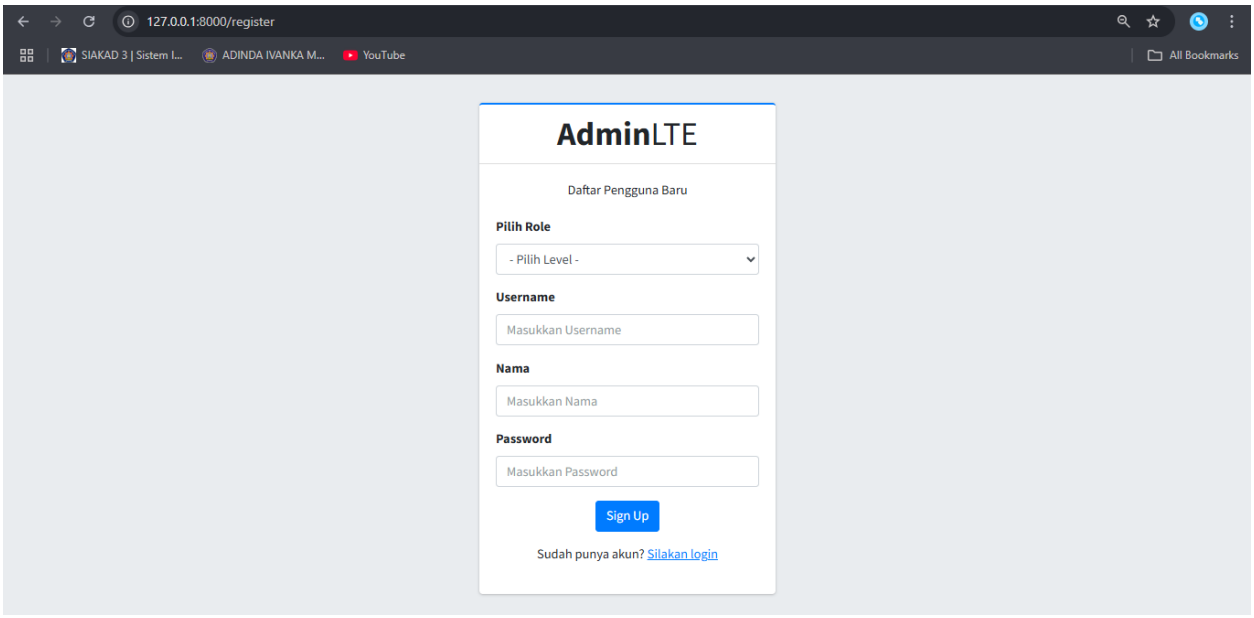
4.

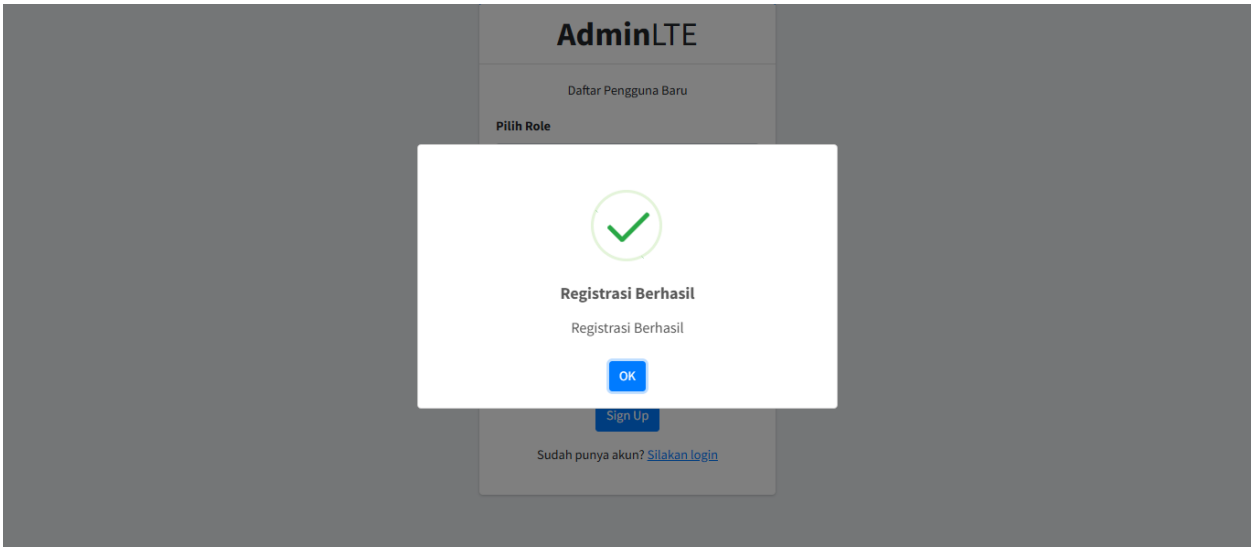
5. Submit kode untuk impementasi Authorization pada repository github kalian.

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