

Laporan Pemrograman Web Lanjut

Jobsheet 7 : Authentication dan Authorization di Laravel

Dosen Pengampu : Dimas Wahyu Wibowo, ST.,MT.



Disusun Oleh:

Queenadhynar Azarine Dwipa A.

2341760109

SIB 2B

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2023/2024

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,
    ],
    // 'users' => [
    //     'driver' => 'database',
    //     'table' => 'users',
    // ],
],
```

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

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

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

```
Minggu 7 > PWL_POS > app > Models > UserModel.php > UserModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\HasOne;
8  use Illuminate\Database\Eloquent\Relations\BelongsTo;
9  use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable
10
11  class UserModel extends Authenticatable
12  {
13      use HasFactory;
14
15      protected $table = 'm_user'; // mendefinisikan nama tabel yang digunakan oleh model ini
16      protected $primaryKey = 'user_id'; // mendefinisikan primary key dari tabel yang digunakan
17
18      protected $fillable = ['level_id', 'username', 'nama', 'password', 'created_at', 'updated_at', ];
19
20      protected $hidden = ['password']; // jangan ditampilkan saat select
21      protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
22
23
24      public function level(): BelongsTo
25      {
26          return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
27      }
28  }
```

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

```

Minggu 7 > PWL_POS > app > Http > Controllers > AuthController.php > AuthController
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     {
12         if (Auth::check()) { // jika sudah login, maka redirect ke halaman home
13             return redirect('/');
14         }
15         return view('auth.login');
16     }
17     public function postlogin(Request $request)
18     {
19         if ($request->ajax() || $request->wantsJson()) {
20             $credentials = $request->only('username', 'password');
21             if (Auth::attempt($credentials)) {
22                 return response()->json([
23                     'status' => true,
24                     'message' => 'Login Berhasil',
25                     'redirect' => url('/')
26                 ]);
27             }
28             return response()->json([
29                 'status' => false,
30                 'message' => 'Login Gagal'
31             ]);
32         }
33         return redirect('login');
34     }
35     public function logout(Request $request)
36     {
37         Auth::logout();
38         $request->session()->invalidate();
39         $request->session()->regenerateToken();
40         return redirect('login');
41     }
42 }

```

- 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)

```

Minggu 7 > PWL_POS > resources > views > auth > login.blade.php > html > body.hold-transition.login-page > script > <function> > submitHandler
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <title>Login Pengguna</title>
8      <!-- Google Font: Source Sans Pro -->
9      <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallb
10 ack">
11      <!-- Font Awesome -->
12      <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
13      <!-- iCheck bootstrap -->
14      <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
15  </head>
16  <!-- SweetAlert2 -->
17  <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
18  <!-- Theme style -->
19  <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
20 </head>
21
22 <body class="hold-transition login-page">
23     <div class="login-box">
24         <!-- /.login-logo -->
25         <div class="card card-outline card-primary">
26             <div class="card-header text-center"><a href="{{ url('/') }}" class="h1"><b>Admin</b></a></div>
27             <div class="card-body">
28                 <p class="login-box-msg">Sign in to start your session</p>
29                 <form action="{{ url('login') }}" method="POST" id="form-login">
30                     @csrf
31                     <div class="input-group mb-3">
32                         <input type="text" id="username" name="username" class="form-control" placeholder="Username">
33                         <div class="input-group-append">
34                             <div class="input-group-text">
35                                 <span class="fas fa-envelope"></span>
36                             </div>
37                         </div>
38                     <small id="error-username" class="error-text text-danger"></small>
39                 </div>
40                 <div class="input-group mb-3">
41                     <input type="password" id="password" name="password" class="form-control"
42                         placeholder="Password">
43                     <div class="input-group-append">

```

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

```

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

/*
|-----
| 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!
|
*/

// Jobsheet 7
Route::pattern('id', '[0-9]+'); //artinya ketika 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::get('/register', [AuthController::class, 'register'])->name('register');
Route::post('/register', [AuthController::class, 'store_user'])->name('store_user');

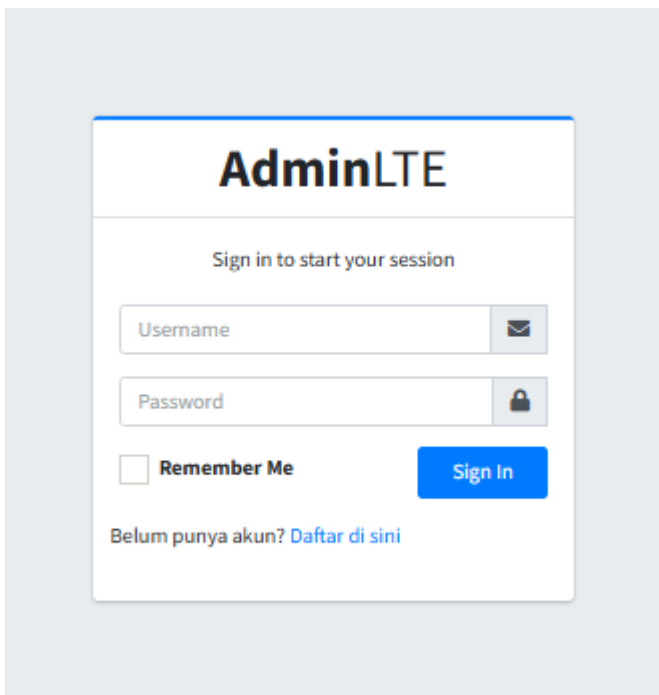
Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini harus login dulu

Route::get('/', [WelcomeController::class, 'index']);

Route::middleware(['authorize:ADM,MNG,STF'])->prefix('user')->group(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::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}', [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
});
});

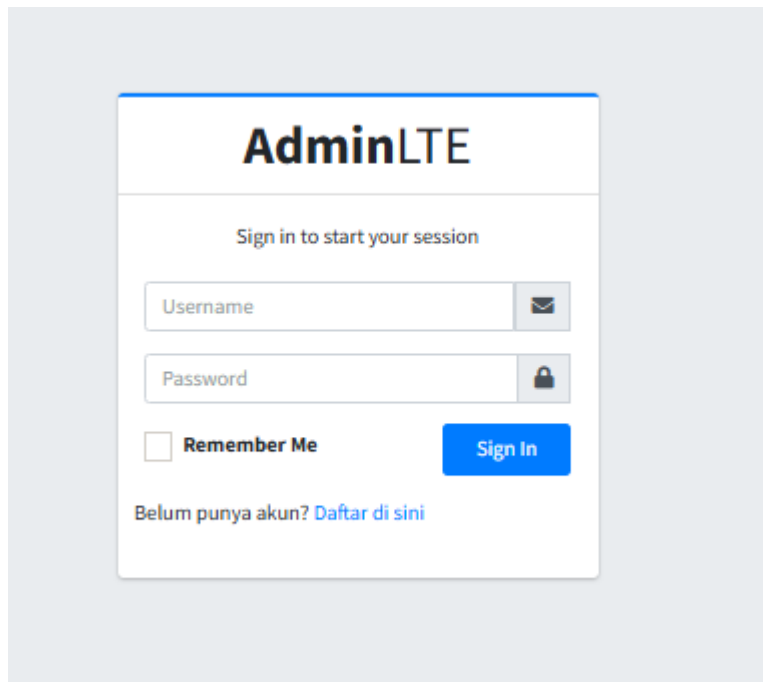
```

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

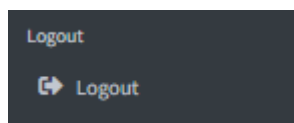


Tugas 1 Implementasi Authentication :

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

```
}

/**
 * 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

3. Kemudian kita edit middleware AuthorizeUser.php untuk bisa mengecek apakah pengguna yang mengakses memiliki Level/Role/Group yang sesuai
4. Kita daftarkan ke app/Http/Kernel.php untuk middleware yang kita buat barusan
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
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	CUS	Pelanggan	2025-03-03 18:40:03	2025-03-03 18:40:03

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. 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.

```
Route::middleware(['authorize:ADM'])->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::get('/create_ajax', [LevelController::class, 'create_ajax']);
        Route::post('/ajax', [LevelController::class, 'store_ajax']);
        Route::get('/{id}', [LevelController::class, 'show']);
        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::delete('/{id}', [LevelController::class, 'destroy']);
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']);
    });
});
```


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



13	Admin1	Dhynar	Administrator	Detail	Edit	Hapus
14	Kasir	azarine	Staff/Kasir	Detail	Edit	Hapus

saat yang akses admin

AdminLTE

Sign in to start your session





☐ Remember Me

Belum punya akun? [Daftar di sini](#)

PWL - Starter Code

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Supplier

Data Barang

Data Transaksi

Stok Barang

Transaksi Penjualan

Logout

Logout

Home

Contact

Daftar Level

HomeLevel

Daftar Level yang terdaftar dalam sistem

Showing 1 to 4 of 4 entries

Showing 1 to 4 of 4 entries


ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	<input type="button" value="Detail"/> <input type="button" value="Edit"/> <input type="button" value="Hapus"/>
2	MNG	Manager	<input type="button" value="Detail"/> <input type="button" value="Edit"/> <input type="button" value="Hapus"/>
3	STF	Staff/Kasir	<input type="button" value="Detail"/> <input type="button" value="Edit"/> <input type="button" value="Hapus"/>
4	CUS	Pelanggan	<input type="button" value="Detail"/> <input type="button" value="Edit"/> <input type="button" value="Hapus"/>



Previous 1 Next

saat yang akses selain admin

AdminLTE

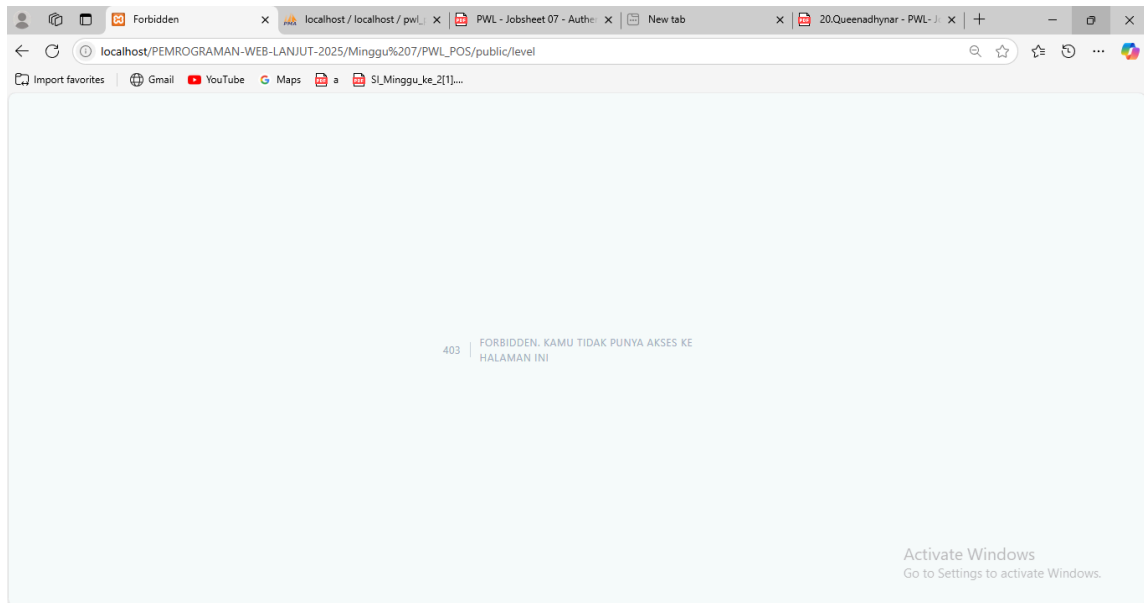
Sign in to start your session





☐ Remember Me

Belum punya akun? [Daftar di sini](#)



Tugas 2 - Implementasi Authentication :

1. Apa yang kalian pahami pada praktikum 2 ini?

- praktikum ini digunakan untuk membatasi akses halaman level user. dengan membuat middleware baru, agar hanya user dengan level cth ADM yang bisa akses data level. Middleware tersebut dicek melalui data user yang sedang login. Setelah itu, didaftarkan pada Kernel.php dan uji akses menggunakan akun dengan level yang berbeda. cth kasir dan admin

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

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


```

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');
    }
}

```

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

```

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::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::delete('/{id}', [BarangController::class, 'destroy']);
    Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']);
    Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']);
    Route::get('/{id}/show_ajax', [BarangController::class, 'show_ajax']);
});

```

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

Tugas 3 – Implementasi Multi-Level Authorization :

- Silahkan implementasikan multi-level authorization pada project kalian masing-masing
- 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
- Submit kode untuk implementasi Authorization pada repository github kalian.

Tabel m_user

```

Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
    Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini harus login dulu
        Route::get('/', [WelcomeController::class, 'index']);
        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::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}', [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::get('/{id}/show_ajax', [UserController::class, 'show_ajax']); // Menampilkan detail user menggunakan 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::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
        });
    });
});

```

Tabel m_kategori

```

Route::middleware(['authorize:ADM,MNG,STF'])->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::get('/create_ajax', [KategoriController::class, 'create_ajax']);
        Route::post('/ajax', [KategoriController::class, 'store_ajax']);
        Route::get('/{id}', [KategoriController::class, 'show']);
        Route::get('/{id}/edit', [KategoriController::class, 'edit']);
        Route::put('/{id}', [KategoriController::class, 'update']);
        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}/show_ajax', [KategoriController::class, 'show_ajax']);
        Route::delete('/{id}', [KategoriController::class, 'destroy']);
    });
});

```

Tabel m_supplier

```

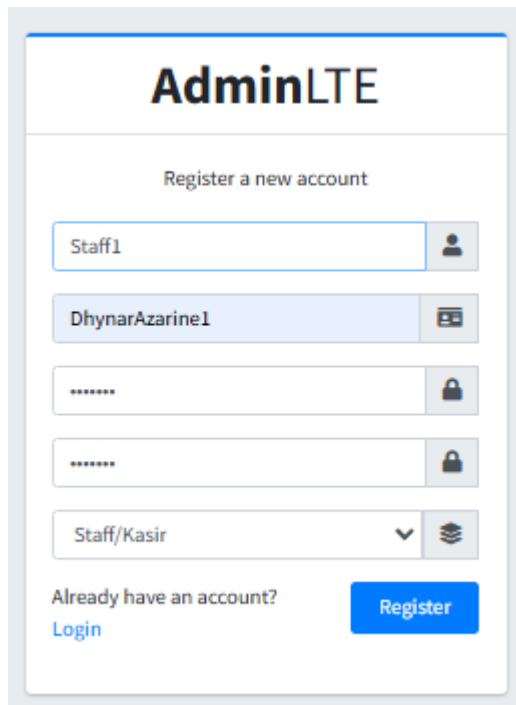
Route::middleware(['authorize:ADM,MNG,STF'])->group(function () {
    Route::group(['prefix' => 'supplier'], function () {
        Route::get('/', [SupplierController::class, 'index']);
        Route::post('/list', [SupplierController::class, 'list']);
        Route::get('/create', [SupplierController::class, 'create']);
        Route::post('/', [SupplierController::class, 'store']);
        Route::get('/create_ajax', [SupplierController::class, 'create_ajax']);
        Route::post('/ajax', [SupplierController::class, 'store_ajax']);
        Route::get('/{id}', [SupplierController::class, 'show']);
        Route::get('/{id}/edit', [SupplierController::class, 'edit']);
        Route::put('/{id}', [SupplierController::class, 'update']);
        Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']);
        Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']);
        Route::delete('/{id}', [SupplierController::class, 'destroy']);
        Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']);
        Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']);
        Route::get('/{id}/show_ajax', [SupplierController::class, 'show_ajax'])->name('supplier.show_ajax');
    });
});

```

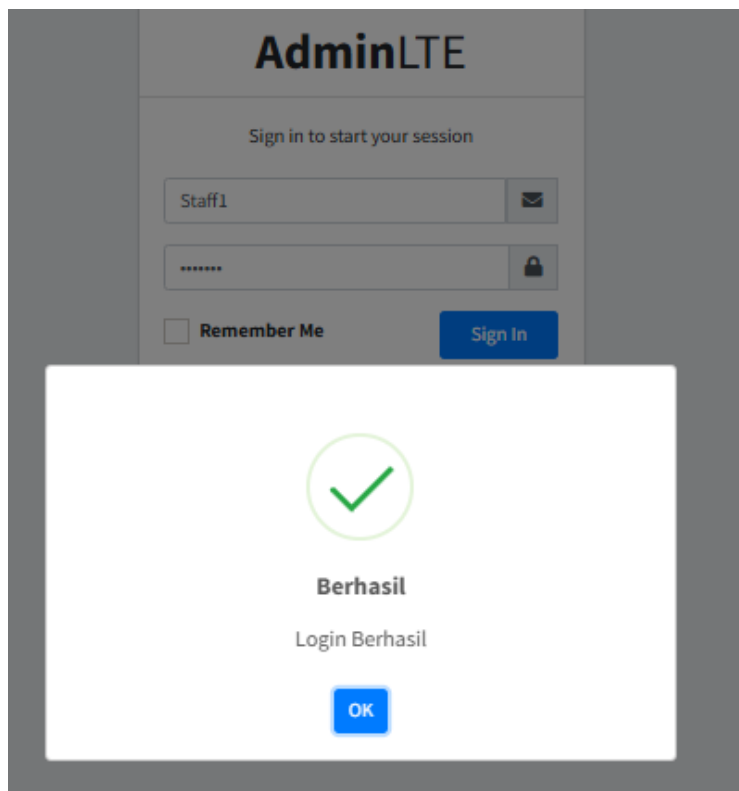
Tugas 4 – Implementasi Form Registrasi :

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



The image shows the 'AdminLTE' registration form. At the top, it says 'AdminLTE' in a large, bold font. Below that, it says 'Register a new account'. The form contains several input fields: a username field with 'Staff1', an email field with 'DhynarAzarine1', two password fields (both masked with '*****'), and a role dropdown menu currently showing 'Staff/Kasir'. There are icons for each field: a person for username, an envelope for email, a lock for passwords, and a server for the role. At the bottom, there is a link 'Already have an account? Login' and a blue 'Register' button.



The image shows the 'AdminLTE' login page with a success modal. The background is a dark gray. The login form is in the center, with fields for 'Staff1' and a masked password '*****'. There is a 'Remember Me' checkbox and a blue 'Sign In' button. Overlaid on top of the login form is a white modal box. Inside the modal, there is a green checkmark icon, the text 'Berhasil' (Success), 'Login Berhasil' (Login Successful), and a blue 'OK' button.

web.php

```
Route::get('/register', [AuthController::class, 'register']);
Route::post('/register', [AuthController::class, 'postRegister']);
```

login.blade

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

authcontroller

```
public function register()
{
    $level = LevelModel::select('level_id', 'level_nama')->get();

    return view('auth.register')->with('level', $level);
}

public function postRegister(Request $request)

$validator = Validator::make($request->all(), [
    'username' => 'required|string|unique:m_user,username',
    'nama' => 'required|string|max:255',
    'password' => 'required|string|min:5',
    'level_id' => 'required|exists:m_level,level_id',
]);

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

UserModel::create([
    'username' => $request->username,
    'nama' => $request->nama,
    'password' => bcrypt($request->password),
    'level_id' => $request->level_id
]);
```

registerblade

```
Minggu 7 > PWL_POS > resources > views > auth > register.blade.php > html > body.hold-transition.login-page > div.login-box > div.card.card-outline.card-primary > div.ca
1 <?php</?php>
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>Register Pengguna</title>
8     <!-- Google Font: Source Sans Pro -->
9     <link rel="stylesheet"
10         href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallb
11 >
12     <!-- Font Awesome -->
13     <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
14     <!-- iCheck bootstrap -->
15     <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
16     <!-- SweetAlert2 -->
17     <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
18     <!-- Theme style -->
19     <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
20 </head>
21
22 <body class="hold-transition login-page">
23     <div class="login-box">
24         <!-- /.login-logo -->
25         <div class="card card-outline card-primary">
26             <div class="card-header text-center"><a href="{{ url('/') }}" class="h1"><b>Admin</b><b>LTE</b></a></div>
27             <div class="card-body">
28                 <p class="login-box-msg">Register a new account</p>
```