KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Mata Kuliah	:	Pemrograman Web Lanjut (PWL)
Program Studi	:	D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester	:	4 (empat) / 5 (lima)
Pertemuan ke-	:	7 (tujuh)
Nama	:	Aditya Yuhanda Putra

JOBSHEET 07

Authentication dan *Authorization* di Laravel

Laravel Authentication dipergunakan untuk memproteksi halaman atau fitur dari web yang hanya diakses oleh orang tertentu yang diberikan hak. Fitur seperti ini biasanya ditemui di sistem yang memiliki fitur administrator atau sistem yang memiliki pengguna yang boleh menambahkan datanya.

Laravel membuat penerapan otentikasi sangat sederhana dan telah menyediakan berbagai fitur yang dapat dimanfaatkan tanpa perlu melakukan penambahan instalasi modul tertentu. File konfigurasi otentikasi terletak di config / auth.php, yang berisi beberapa opsi yang terdokumentasi dengan baik untuk mengubah konfigurasi dari layanan otentikasi.

Pada intinya, fasilitas otentikasi Laravel terdiri dari "*guard*s" dan "*provider*s". *Guard*s menentukan bagaimana pengguna diautentikasi untuk setiap permintaan. Misalnya, Laravel mengirim dengan *guard*s untuk sesi dengan menggunakan penyimpanan session dan cookie.

Middleware

Middleware adalah lapisan perantara antara permintaan ***route HTTP*** yang masuk dan ***action*** dari Controller yang akan dijalankan. **Middleware** memungkinkan kita untuk meakukan berbagai tugas baik itu sebelum ataupun sesudah tindakan dilakukan. Kita juga dapat menggunakan *tool* **CLI** untuk membuat sebuah **Middleware** dalam **Laravel**. Beberapa contoh penggunaan **Middleware** meliputi autentikasi, validasi, manipulasi permintaan, dan lainnya. Berikut di bawah ini adalah manfaat dari **Middleware :**

- **Keamanan** : dalam **Middleware** memungkinkan kita untuk memverifikasi apakah pengguna sudah diautentikasi sebelum mengakses halaman tertentu. Dengan demikian, kita dapat melindungi data sensitif dan mengontrol hak akses pengguna.
- **Pemfilteran Data** : **Middleware** dapat digunakan untuk memanipulasi data permintaan sebelum sebuah ***action*** dalam *controller* dilakukan. Misalnya, kita dapat memeriksa terlebih dahulu data yang dikirim oleh pengguna sebelum data tersebut diproses lebih

lanjut atau kita ingin memodifikasi data yang akan dikirim lalu kita dapat memeriksa ulang data yang akan dikirim oleh pengguna sebelum data tersebut diproses.

- **Logging dan Audit** : **Middleware** juga dapat digunakan untuk mencatat aktivitas pengguna atau melakukan audit terhadap permintaan yang masuk. Ini dapat membantu dalam pemantauan dan analisis aplikasi.

INFO

Kita akan menggunakan Laravel Auth secara manual seperti https://laravel.com/docs/10.x/authentication#authenticating-users

Sesuai dengan `Studi Kasus PWL.pdf`.
Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS.**

Project **PWL_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. Implementasi Manual Authentication di Laravel

Autentikasi adalah proses untuk memverifikasi identitas pengguna yang mencoba mengakses sistem. Dalam konteks aplikasi web, autentikasi memastikan bahwa pengguna yang mencoba login memiliki hak akses yang sesuai berdasarkan kredensial seperti email dan password. Proses autentikasi berbeda dengan **otorisasi**, yang merupakan langkah lanjutan untuk menentukan hak akses apa yang dimiliki pengguna setelah mereka berhasil diautentikasi.

Konsep Autentikasi di Laravel

Laravel menawarkan sistem autentikasi yang sangat fleksibel. Laravel menyediakan mekanisme autentikasi bawaan melalui layanan authentication scaffolding seperti Laravel *Jetstream* dan *Breeze*, yang dapat secara otomatis menghasilkan halaman dan logika autentikasi. Namun, terkadang pengembang memerlukan implementasi autentikasi yang lebih manual untuk memberikan kontrol penuh terhadap setiap aspek dari proses tersebut.

Beberapa komponen penting dalam sistem autentikasi Laravel meliputi:

- *Guard*: Komponen yang mengatur bagaimana pengguna diautentikasi untuk setiap permintaan. *Guard* default menggunakan sesi dan cookie.

- *Provider*: Mengatur bagaimana pengguna diambil dari database atau sumber data lainnya. *Provider* default mengambil data pengguna dari database dengan menggunakan Eloquent ORM.
- *Session*: Laravel menggunakan sesi untuk menyimpan status autentikasi pengguna. Sesi memungkinkan sistem untuk mengingat pengguna yang sudah login di antara permintaan HTTP yang berbeda.

Alur umum dari autentikasi meliputi:

1. *Login*: Pengguna mengirimkan kredensial (biasanya berupa email dan password).
2. *Verifikasi Kredensial*: Sistem memeriksa apakah kredensial yang diberikan sesuai dengan data di database.
3. *Pembuatan Sesi*: Jika kredensial benar, sistem akan membuat sesi untuk pengguna yang akan disimpan di server.
4. *Akses ke Halaman yang Dilindungi*: Pengguna yang terautentikasi dapat mengakses halaman-halaman yang dilindungi oleh *middleware* auth.
5. *Logout*: Pengguna bisa keluar dari sistem dan sesi mereka akan dihapus.

Middleware Autentikasi

Middleware auth di Laravel digunakan untuk melindungi rute atau halaman agar hanya dapat diakses oleh pengguna yang sudah terautentikasi. Jika pengguna mencoba mengakses rute yang memerlukan autentikasi tanpa login, mereka akan diarahkan ke halaman login.

- ***Guard*** bertanggung jawab untuk menangani proses autentikasi pengguna. Laravel secara default menggunakan *guard* berbasis sesi untuk autentikasi web, namun juga mendukung *guard* berbasis token (seperti API).
- ***Provider*** bertugas untuk mengambil pengguna dari database. Laravel menyediakan *provider* default yang menggunakan Eloquent, namun juga mendukung *provider* lain seperti Query Builder.

Implementasi di Laravel 10

Kita akan menerapkan penggunakan authentication di Laravel. Dalam penerapan ini, kita akan mencoba membuat otentikasi secara di Laravel, agar kita paham langkah-langkah dalam membuat Authentication

Praktikum 1 – Implementasi Authentication :

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

```
62      'providers' => [
63          'users' => [
64              'driver' => 'eloquent',
65              'model' => App\Models\User::class,
66          ],
```

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

```
62      'providers' => [
63          'users' => [
64              'driver' => 'eloquent',
65              'model' => App\Models\UserModel::class,
66          ],
```

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

```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');
    }
}
```

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;

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**)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Login Pengguna</title>
```

```
  <!-- Google Font: Source Sans Pro -->
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallb
ack">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
  <!-- icheck bootstrap -->
  <link rel="stylesheet" href="{{ asset('plugins/icheck-bootstrap/icheck-bootstrap.min.css')
}}">
  <!-- SweetAlert2 -->
  <link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-
4.min.css') }}">
  <!-- Theme style -->
  <link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
</head>
<body class="hold-transition login-page">
<div class="login-box">
  <!-- /.login-logo -->
  <div class="card card-outline card-primary">
    <div class="card-header text-center"><a href="{{ url('/') }}"
class="h1"><b>Admin</b>LTE</a></div>
    <div class="card-body">
      <p class="login-box-msg">Sign in to start your session</p>
      <form action="{{ url('login') }}" method="POST" id="form-login">
        @csrf
        <div class="input-group mb-3">
          <input type="text" id="username" name="username" class="form-control"
placeholder="Username">
           <div class="input-group-append">
             <div class="input-group-text">
               <span class="fas fa-envelope"></span>
             </div>
           </div>
          <small id="error-username" class="error-text text-danger"></small>
        </div>
        <div class="input-group mb-3">
          <input type="password" id="password" name="password" class="form-control"
placeholder="Password">
           <div class="input-group-append">
             <div class="input-group-text">
               <span class="fas fa-lock"></span>
             </div>
           </div>
          <small id="error-password" class="error-text text-danger"></small>
        </div>
        <div class="row">
          <div class="col-8">
            <div class="icheck-primary">
              <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
            </div>
          </div>
          <!-- /.col -->
          <div class="col-4">
            <button type="submit" class="btn btn-primary btn-block">Sign In</button>
          </div>
          <!-- /.col -->
        </div>
      </form>
    </div>
    <!-- /.card-body -->
  </div>
  <!-- /.card -->
</div>
<!-- /.login-box -->

<!-- jQuery -->
```

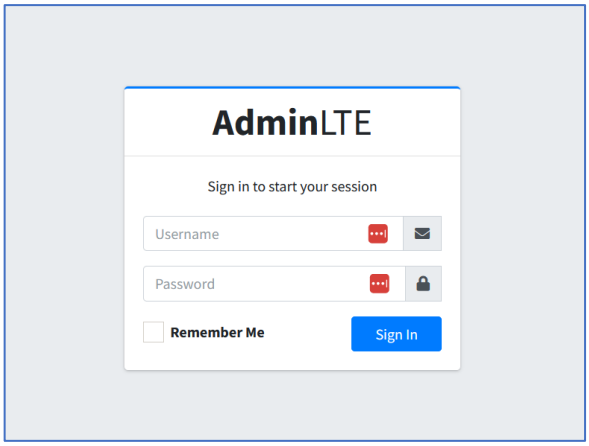
```
<script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<!-- jquery-validation -->
<script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<script src="{{ asset('plugins/jquery-validation/additional-methods.min.js') }}"></script>
<!-- SweetAlert2 -->
<script src="{{ asset('plugins/sweetalert2/sweetalert2.min.js') }}"></script>
<!-- AdminLTE App -->
<script src="{{ asset('dist/js/adminlte.min.js') }}"></script>

<script>
  $.ajaxSetup({
    headers: {
      'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
    }
  });

  $(document).ready(function() {
    $("#form-login").validate({
      rules: {
        username: {required: true, minlength: 4, maxlength: 20},
        password: {required: true, minlength: 6, 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

```php
<?php

use App\Http\Controllers\UserController;
use App\Http\Controllers\SupplierController;
use App\Http\Controllers\BarangController;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
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(){ // 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
   - Implementasi proses login sudah mengikuti langkah praktikum 1 Jobsheet diatas.
2. Silahkan implementasi proses logout pada halaman web yang kalian buat
   - Baik saya akan buat.

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

   - Mengupdate isi dari function logout pada **AuthController.php**.

```php
public function logout(Request $request): R
{
    Auth::logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();

    return redirect(to: '/login');
}
```

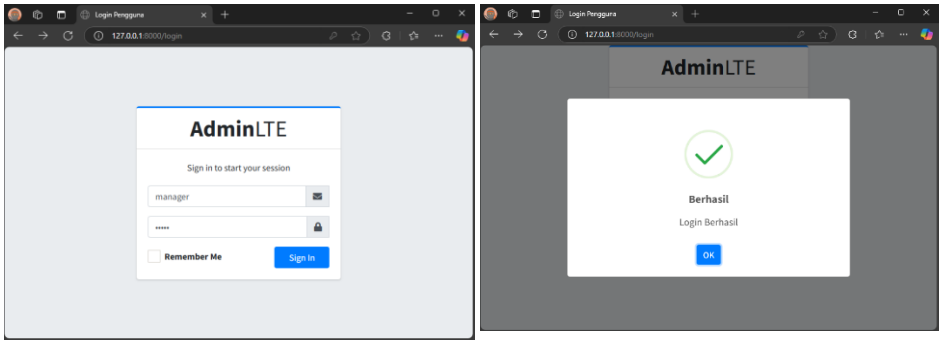
   - Mengupdate Route logout di **web.php**.

```php
Route::post(uri: 'logout', action: [AuthController::class, 'logout'])->name(name: 'logout')->middleware(middleware: 'auth');
```

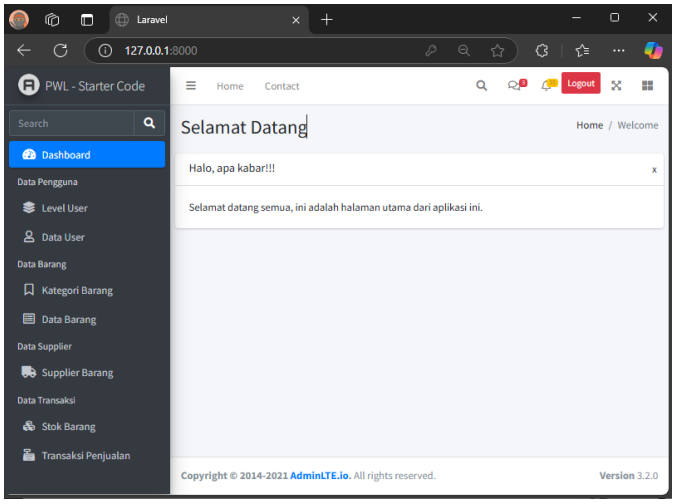
   - Menambahkan tombol Logout pada tampilan di **header.blade.php.**

```blade
<li class="nav-item">
    <form method="POST" action="{{ route(name: 'logout') }}">
        @csrf
        <button type="submit" class="btn btn-danger btn-sm">Logout</button>
    </form>
</li>
```

   - Maka akan muncul tombol logout pada header, namun login dulu:



   - Halaman tampilan ketika sudah login, dengan tombol Logout pada bagian header



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

B. Implementasi *Authorization* di Laravel

Authorization merupakan proses setelah authentication berhasil dilakukan (dalam kata lain,kita berhasil login ke sistem). *Authorization* berkenaan dengan hak akses pengguna dalam menggunakan sistem. *Authorization* memberikan/memastikan hak akses (ijin akses) kita, sesuai dengan aturan (role) yang ada di sistem. *Authorization* sangat penting untuk membatasi akses pengguna sesuai dengan peruntukannya.

Contoh ketika kita mengakses LMS dengan akun (*username* dan *password*) yang bertipe `Mahasiswa`. Saat berhasil melakukan authentication, maka hak akses kita juga akan diberikan selayaknya mahasiswa. Seperti melihat kursus (course), melihat materi, men-download file materi, mengerjakan/meng-upload tugas, mengikuti ujian, dll. Kita tidak akan diberikan hak akses oleh sistem untuk membuat materi, membuat soal ujian, membuat tugas, memberikan nilai tugas karena hak akses tersebut masuk ke ranah akun tipe `Dosen/Pengajar`.

Selain menyediakan layanan otentikasi bawaan, Laravel juga menyediakan cara sederhana untuk mengotorisasi tindakan pengguna terhadap sumber daya tertentu. Misalnya, meskipun pengguna diautentikasi, mereka mungkin tidak berwenang untuk memperbarui atau menghapus model Eloquent atau rekaman database tertentu yang dikelola oleh aplikasi Anda. Fitur otorisasi Laravel menyediakan cara yang mudah dan terorganisir untuk mengelola jenis pemeriksaan otorisasi ini.

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
/**
 * Relasi ke tabel level
 */
public function level(): BelongsTo
{
    return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
}

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

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

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

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

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

```php
<?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)
     */
    public function handle(Request $request, Closure $next, $role = ''): Response
    {
        $user = $request->user();   // ambil data user yg login
                                    // fungsi user() diambil dari UserModel.php
        if($user->hasRole($role)){  // cek apakah user punya role yg diinginkan
            return $next($request);
        }
        // jika tidak punya role, maka tampilkan error 403
        abort(403, '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,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::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	deleted_at
1	ADM	Administrator	NULL	NULL	NULL
2	MNG	Manager	NULL	NULL	NULL
3	STF	Staf	NULL	2024-08-16 01:49:20	NULL
4	KSR	Kasir	NULL	NULL	NULL

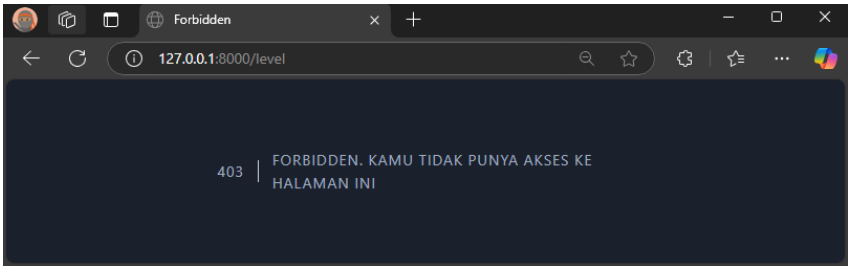
6. Untuk mencoba *authorization* yang telah kita buat, maka perlu kita modiifikasi `route/web.php` untuk menentukan route mana saja yang akan diberi hak akses sesuai dengan level user

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

    // artinya semua route di dalam group ini harus punya role ADM (Administrator)
    Route::middleware(['authorize:ADM'])->group(function(){
        Route::get('/level',[LevelController::class,'index']);
        Route::post('/level/list',[LevelController::class,'list']); // 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 tampilkan form edit
        Route::put('/level/{id}',[LevelController::class,'update']);    // untuk proses update data
        Route::delete('/level/{id}',[LevelController::class,'destroy']); // untuk proses hapus data
    });

    // route Kategori
```

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

   Maka tampilan akan seperti ini kalau dari level manager :



Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?
   - Pada praktikum ini, kita menerapkan **authorization** di Laravel dengan menggunakan **Middleware** untuk mengecek apakah pengguna memiliki akses ke halaman tertentu berdasarkan **role**-nya. Middleware ini akan memfilter request dan memastikan hanya pengguna dengan level yang sesuai yang bisa mengakses route tertentu.
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
   - Modifikasi UserModel.php
     - Menambahkan method getRoleName() untuk mendapatkan nama role.
     - Menambahkan method hasRole($role) untuk mengecek apakah user memiliki role tertentu.

- Membuat Middleware AuthorizeUser.php
  - Menjalankan perintah:
    `php artisan make:middleware AuthorizeUser`
  - Middleware ini mengecek apakah user yang sedang login memiliki role yang sesuai. Jika iya, request diteruskan. Jika tidak, ditampilkan error **403 (Forbidden).**
- Mendaftarkan Middleware di Kernel.php
  - Middleware AuthorizeUser didaftarkan dengan alias 'authorize', sehingga bisa digunakan di routes dengan authorize:ROLE_KODE
- Menyesuaikan Data di Database (m_level)
  - Tabel m_level berisi daftar role seperti **Administrator (ADM), Manager (MNG), Staff (STF), dan Pelanggan (CUS).**
- Menggunakan Middleware di web.php
  - Semua route dalam group ini hanya bisa diakses oleh user dengan **level_kode = ADM** (Administrator).

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

C. Multi-Level Authorization di Laravel

Bagaimana seandainya jika terdapat level/group/role satu dengan yang lain memiliki hak akses yang sama. Contoh sederhana, user level Admin dan Manager bisa sama-sama mengakses menu Barang pada aplikasi yang kita buat. Maka tidak mungkin kalau kita buat route untuk masing-masing level user. Hal ini akan memakan banyak waktu, dan proses yang lama.

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

// artinya semua route di dalam group ini harus punya role MNG (Manager)
Route::middleware(['authorize:MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

Hal ini jadi kendala ketika kita mau mengganti hak akses, maka kita akan mengganti sebagian besar route yang sudah kita tulis. Untuk itu, kita perlu mengelola middleware agar bisa mendukung penambahan hak akses secara dinamis.

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()`

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

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

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

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

```php
<?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)
     */
    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');
    }
}
```

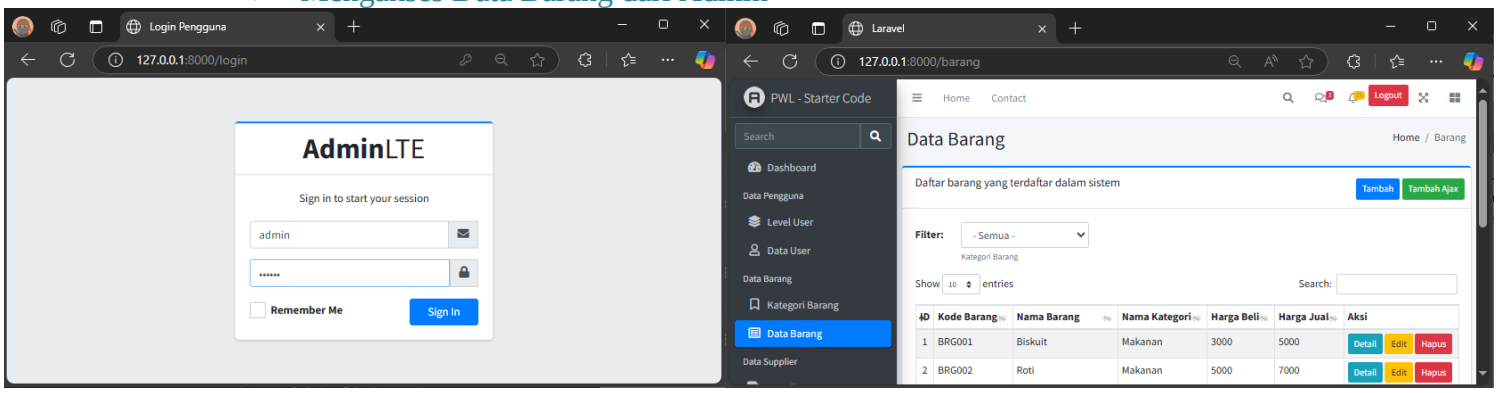
3. Setelah itu tinggal kita perbaiki `route/web.php` sesuaikan dengan role/level yang diinginkan. Contoh

```php
// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

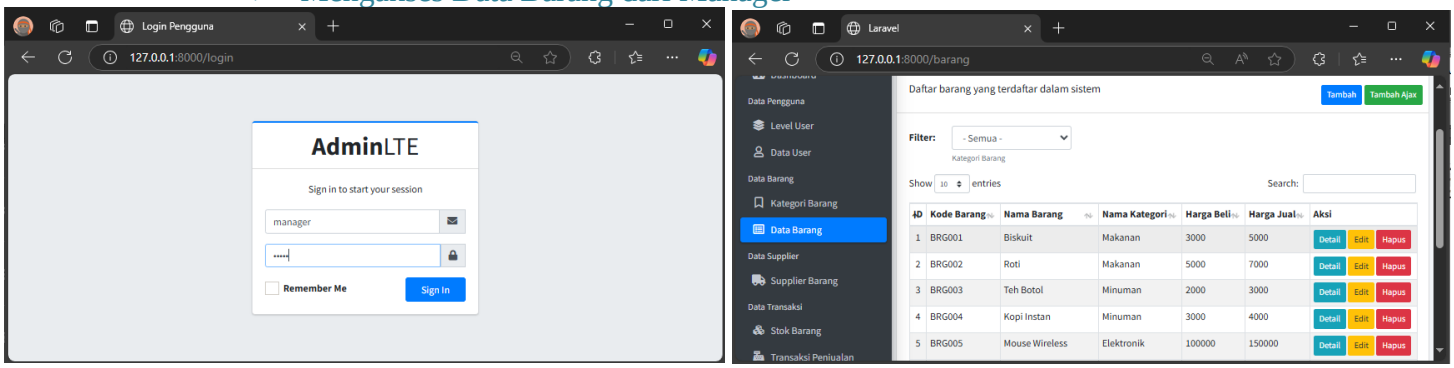
4. Sekarang kita sudah bisa memberikan hak akses menu/route ke beberapa level user

Tugas 3 – Implementasi Multi-Level Authorization :

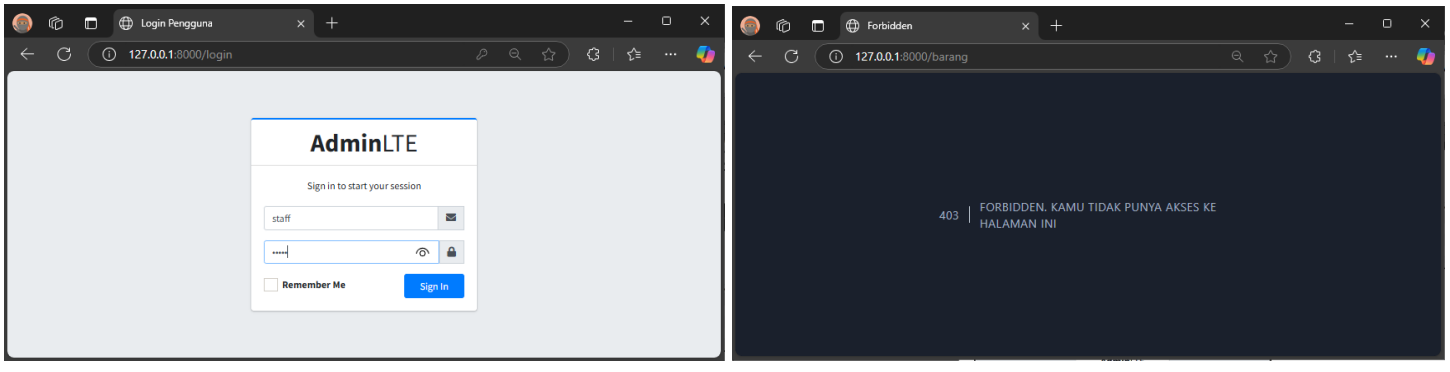
1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
   - Mengakses Data Barang dari Admin



   - Mengakses Data Barang dari Manager



   - Mengakses Data Barang dari Staff

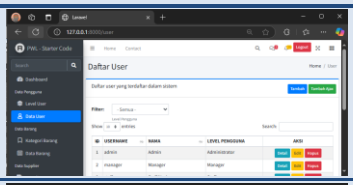
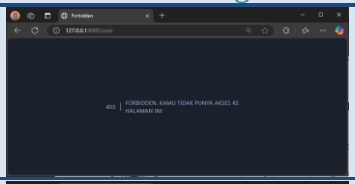
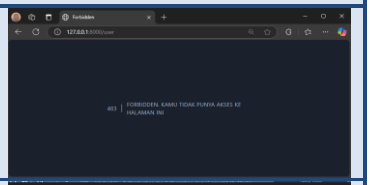
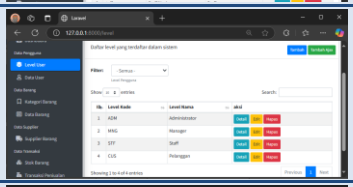
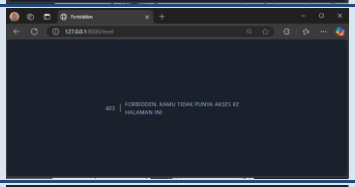
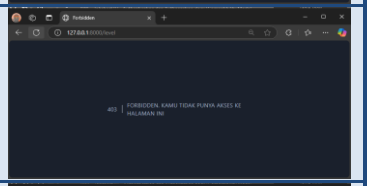
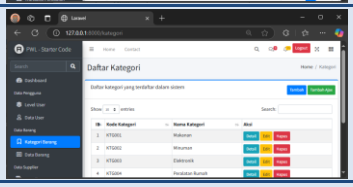
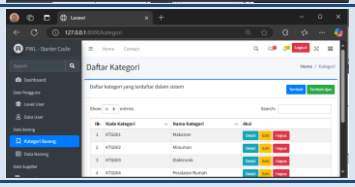
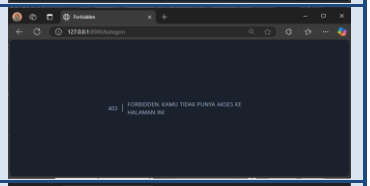
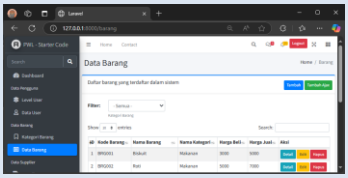
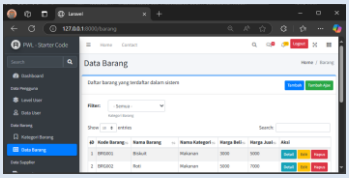
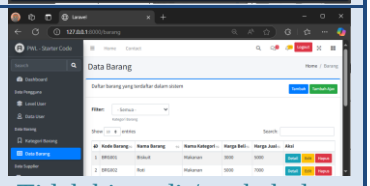
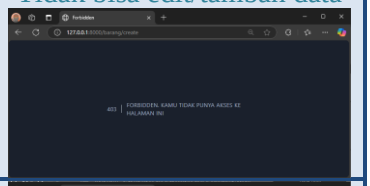
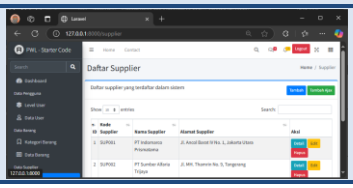
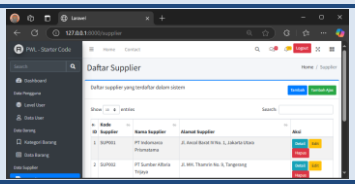
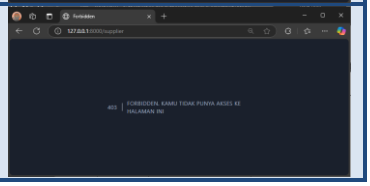


3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User
   - Berikut Kode Program routes/web.php

routes > web.php > ...

```php
<?php

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

Route::pattern(key: 'id', pattern: '[0-9]+'); // Pastikan parameter {id} hanya berupa angka

// Rute otentikasi
Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login');
Route::post(uri: 'login', action: [AuthController::class, 'postlogin']);
Route::post(uri: '/logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth')->name(name: 'logout');


// Semua rute di bawah ini hanya bisa diakses jika sudah login
Route::middleware(middleware: ['auth'])->group(callback: function (): void {
    Route::get(uri: '/', action: [WelcomeController::class, 'index']);

    Route::middleware(middleware: ['authorize:ADM'])->group(callback: function(): void{
        Route::group(attributes: ['prefix' => 'user'], routes: function (): void {
            Route::get(uri: '/', action: [UserController::class, 'index']); // menampilkan halaman awal user
            Route::post(uri: '/list', action: [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
            Route::get(uri: '/create', action: [UserController::class, 'create']); // menampilkan halaman form tambah user
            Route::post(uri: '/', action: [UserController::class, 'store']); // menyimpan data user baru
            //Create Menggunakan AJAX
            Route::get(uri: '/create_ajax', action: [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
            Route::post(uri: '/ajax', action: [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax
            Route::get(uri: '/{id}', action: [UserController::class, 'show']); // menampilkan detail user
            Route::get(uri: '/{id}/edit', action: [UserController::class, 'edit']); // menampilkan halaman form edit user
            Route::put(uri: '/{id}', action: [UserController::class, 'update']); // menyimpan perubahan data user
            //Edit Menggunakan AJAX
            Route::get(uri: '/{id}/edit_ajax', action: [UserController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax
            Route::put(uri: '/{id}/update_ajax', action: [UserController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax
            //Delete Menggunakan AJAX
            Route::get(uri: '/{id}/delete_ajax', action: [UserController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete user Ajax
            Route::delete(uri: '/{id}/delete_ajax', action: [UserController::class, 'delete_ajax']); // Untuk hapus data user Ajax
            Route::delete(uri: '/{id}', action: [UserController::class, 'destroy']); // menghapus data user
        });
    });

    // artinya semua route di dalam group ini harus punya role ADM (Administrator)
    Route::middleware(middleware: ['authorize:ADM'])->group(callback: function(): void{
        Route::group(attributes: ['prefix' => 'level'], routes: function (): void {
            Route::get(uri: '/', action: [LevelController::class, 'index']); // menampilkan halaman awal level
            Route::post(uri: '/list', action: [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
            Route::get(uri: '/create', action: [LevelController::class, 'create']); // menampilkan halaman form tambah level
            Route::post(uri: '/', action: [LevelController::class, 'store']); // menyimpan data level baru
            //Create Menggunakan AJAX
            Route::get(uri: '/create_ajax', action: [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah level Ajax
            Route::post(uri: '/ajax', action: [LevelController::class, 'store_ajax']); // Menyimpan data level baru Ajax
            Route::get(uri: '/{id}', action: [LevelController::class, 'show']); // menampilkan detail level
            Route::get(uri: '/{id}/edit', action: [LevelController::class, 'edit']); // menampilkan halaman form edit level
            Route::put(uri: '/{id}', action: [LevelController::class, 'update']); // menyimpan perubahan data level
            //Edit Menggunakan AJAX
            Route::get(uri: '/{id}/edit_ajax', action: [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit level Ajax
            Route::put(uri: '/{id}/update_ajax', action: [LevelController::class, 'update_ajax']); // Menyimpan perubahan data level Ajax
            //Delete Menggunakan AJAX
            Route::get(uri: '/{id}/delete_ajax', action: [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete level Ajax
            Route::delete(uri: '/{id}/delete_ajax', action: [LevelController::class, 'delete_ajax']); // Untuk hapus data level Ajax
            Route::delete(uri: '/{id}', action: [LevelController::class, 'destroy']); // menghapus data level
        });
    });

    // artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
    Route::middleware(middleware: ['authorize:ADM,MNG'])->group(callback: function(): void{
        Route::group(attributes: ['prefix' => 'kategori'], routes: function (): void {
            Route::get(uri: '/', action: [KategoriController::class, 'index']); // menampilkan halaman awal kategori
            Route::post(uri: "/list", action: [KategoriController::class, 'list']); // menampilkan data kategori dalam bentuk json untuk datatables
            Route::get(uri: '/create', action: [KategoriController::class, 'create']); // menampilkan halaman form tambah kategori
            Route::post(uri: '/', action: [KategoriController::class, 'store']); // menyimpan data kategori baru
            // Create menggunakan AJAX
            Route::get(uri: '/create_ajax', action: [KategoriController::class, 'create_ajax']); // menampilkan halaman form tambah kategori ajax
            Route::post(uri: '/ajax', action: [KategoriController::class, 'store_ajax']); // menyimpan data kategori baru ajax
            Route::get(uri: '/{id}', action: [KategoriController::class, 'show']); // menampilkan detail kategori
            Route::get(uri: '/{id}/edit', action: [KategoriController::class, 'edit']); // menampilkan halaman form edit kategori
            Route::put(uri: '/{id}', action: [KategoriController::class, 'update']); // menyimpan perubahan data kategori
            // Edit menggunakan AJAX
            Route::get(uri: '/{id}/edit_ajax', action: [KategoriController::class, 'edit_ajax']); // menampilkan halaman form edit kategori ajax
            Route::put(uri: '/{id}/update_ajax', action: [KategoriController::class, 'update_ajax']); // menyimpan perubahan data kategori ajax
            // Delete menggunakan AJAX
            Route::get(uri: '/{id}/delete_ajax', action: [KategoriController::class, 'confirm_ajax']); //menampilkan form confirm delete kategori ajax
            Route::delete(uri: '/{id}/delete_ajax', action: [KategoriController::class, 'delete_ajax']); // menghapus data kategori ajax
            Route::delete(uri: '/{id}', action: [KategoriController::class, 'destroy']); // menghapus data kategori
        });
    });


    // Staff hanya bisa melihat data barang
    Route::middleware(middleware: ['authorize:ADM,MNG,STF'])->group(callback: function(): void{
        Route::group(attributes: ['prefix' => 'barang'], routes: function (): void {
            Route::get(uri: '/', action: [BarangController::class, 'index']); // Menampilkan daftar barang
            Route::post(uri: '/list', action: [BarangController::class, 'list']); // Menampilkan data barang dalam bentuk JSON untuk datatables
            Route::get(uri: '/{id}', action: [BarangController::class, 'show']); // Menampilkan detail barang
        });
    });

    // ADM & MNG bisa menambah, mengedit, dan menghapus barang
    Route::middleware(middleware: ['authorize:ADM,MNG'])->group(callback: function(): void{
        Route::group(attributes: ['prefix' => 'barang'], routes: function (): void {
            Route::get(uri: '/create', action: [BarangController::class, 'create']); // Form tambah barang
            Route::post(uri: '/', action: [BarangController::class, 'store']); // Simpan barang baru
            // Create menggunakan AJAX
            Route::get(uri: '/create_ajax', action: [BarangController::class, 'create_ajax']); // Form tambah barang AJAX
            Route::post(uri: '/ajax', action: [BarangController::class, 'store_ajax']); // Simpan barang baru AJAX
            Route::get(uri: '/{id}/edit', action: [BarangController::class, 'edit']); // Form edit barang
            Route::put(uri: '/{id}', action: [BarangController::class, 'update']); // Simpan perubahan barang
            // Edit menggunakan AJAX
            Route::get(uri: '/{id}/edit_ajax', action: [BarangController::class, 'edit_ajax']); // Form edit barang AJAX
            Route::put(uri: '/{id}/update_ajax', action: [BarangController::class, 'update_ajax']); // Simpan perubahan barang AJAX
            // Delete menggunakan AJAX
            Route::get(uri: '/{id}/delete_ajax', action: [BarangController::class, 'confirm_ajax']); // Form konfirmasi hapus barang AJAX
            Route::delete(uri: '/{id}/delete_ajax', action: [BarangController::class, 'delete_ajax']); // Hapus barang AJAX
            Route::delete(uri: '/{id}', action: [BarangController::class, 'destroy']); // Hapus barang
        });
    });

    // artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
    Route::middleware(middleware: ['authorize:ADM,MNG'])->group(callback: function(): void{
        Route::group(attributes: ['prefix' => 'supplier'], routes: function (): void {
            Route::get(uri: '/', action: [SupplierController::class, 'index']);
            Route::post(uri: '/list', action: [SupplierController::class, 'list']);
            Route::get(uri: '/create', action: [SupplierController::class, 'create']);
            Route::post(uri: '/', action: [SupplierController::class, 'store']);
            // Create menggunakan AJAX
            Route::get(uri: '/create_ajax', action: [SupplierController::class, 'create_ajax']); // menampilkan halaman form tambah Supplier ajax
            Route::post(uri: '/ajax', action: [SupplierController::class, 'store_ajax']); // menyimpan data Supplier baru ajax
            Route::get(uri: '/{id}', action: [SupplierController::class, 'show']);
            Route::get(uri: '/{id}/edit', action: [SupplierController::class, 'edit']);
            Route::put(uri: '/{id}', action: [SupplierController::class, 'update']);
            // Edit menggunakan AJAX
            Route::get(uri: '/{id}/edit_ajax', action: [SupplierController::class, 'edit_ajax']); // menampilkan halaman form edit Supplier ajax
            Route::put(uri: '/{id}/update_ajax', action: [SupplierController::class, 'update_ajax']); // menyimpan perubahan data Supplier ajax
            // Delete menggunakan AJAX
            Route::get(uri: '/{id}/delete_ajax', action: [SupplierController::class, 'confirm_ajax']); //menampilkan form confirm delete Supplier ajax
            Route::delete(uri: '/{id}/delete_ajax', action: [SupplierController::class, 'delete_ajax']); // menghapus data Supplier ajax
            Route::delete(uri: '/{id}', action: [SupplierController::class, 'destroy']);
        });
    });
});
```

➢ Berikut Hak Akses tiap level nya :

Menu	Level Admin	Level Manager	Staff
Data User			
Data Level			
Data Kategori			
Data Barang			 Tidak bisa edit/tambah data 
Data Supplier			

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

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.
   ➢ AuthController.php :

```
42    public function register(): Factory|View
43    {
44        return view(view: 'auth.register');
45    }
46
```

```php
public function postRegister(Request $request): Redirector|RedirectRe
{
    $validatedData = $request->validate(rules: [
        'username' => 'required|string|unique:m_user,username',
        'nama' => 'required|string|max:255',
        'password' => 'required|string|min:6|confirmed',
        'level_id' => 'required|exists:m_level,level_id',
    ]);

    $user = UserModel::create(attributes: [
        'username' => $validatedData['username'],
        'nama' => $validatedData['nama'],
        'password' => Hash::make(value: $validatedData['password']),
        'level_id' => $validatedData['level_id'],
        'created_at' => now(),
        'updated_at' => now(),
    ]);

    Auth::login(user: $user);

    return redirect(to: '/');
}
```

- Routes/web.php :

```php
// Rute otentikasi
Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login');
Route::post(uri: 'login', action: [AuthController::class, 'postlogin']);
Route::get(uri: '/register', action: [AuthController::class, 'register']);
Route::post(uri: '/register', action: [AuthController::class, 'postRegister']);
Route::post(uri: '/logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth')->name(name: 'logout');
```

- Views/auth/register.blade.php

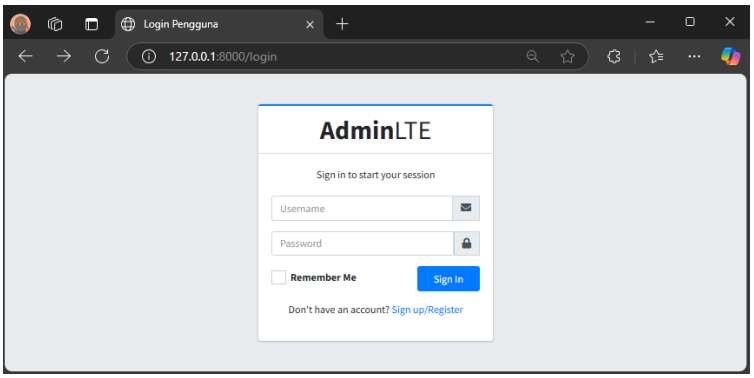
```
resources > views > auth > register.blade.php > ...
@extends(view: 'layouts.app')

@section(section: 'content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">Registrasi</div>

                <div class="card-body">
                    <form method="POST" action="{{ url(path: '/register') }}">
                        @csrf

                        <div class="mb-3">
                            <label for="username" class="form-label">Username</label>
                            <input type="text" class="form-control" id="username" name="username" required>
                        </div>

                        <div class="mb-3">
                            <label for="nama" class="form-label">Nama Lengkap</label>
                            <input type="text" class="form-control" id="nama" name="nama" required>
                        </div>

                        <div class="mb-3">
                            <label for="password" class="form-label">Password</label>
                            <input type="password" class="form-control" id="password" name="password" required>
                        </div>

                        <div class="mb-3">
                            <label for="password_confirmation" class="form-label">Konfirmasi Password</label>
                            <input type="password" class="form-control" id="password_confirmation" name="password_confirmation" required>
                        </div>

                        <div class="mb-3">
                            <label for="level_id" class="form-label">Level</label>
                            <select class="form-control" id="level_id" name="level_id">
                                <option value="1">Admin</option>
                                <option value="2">Manager</option>
                                <option value="3">Staff/Kasir</option>
                                <option value="4">Pelanggan</option>
                            </select>
                        </div>

                        <button type="submit" class="btn btn-primary">Daftar</button>
                        <a href="{{ url(path: '/login') }}" class="btn btn-link">Sudah punya akun?</a>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

- Views/auth/login.blade.php

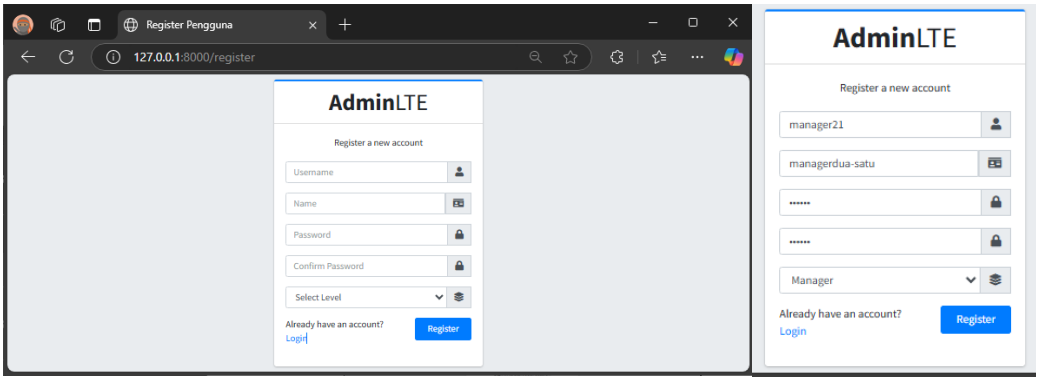
```
                    <div class="row">
                        <div class="col-8">
                            <div class="icheck-primary">
                                <input type="checkbox" id="remember">
                                <label for="remember">Remember Me</label>
                            </div>
                        </div>
                        <div class="col-4">
                            <button type="submit" class="btn btn-primary btn-block">Sign In</button>
                        </div>
                    </div>

                    <div class="text-center mt-3">
                        <p>Don't have an account? <a href="{{ url(path: 'register') }}" class="text-primary">Sign up</a></p>
                    </div>
                </form>
            </div>
```

2. Screenshot hasil yang kalian kerjakan

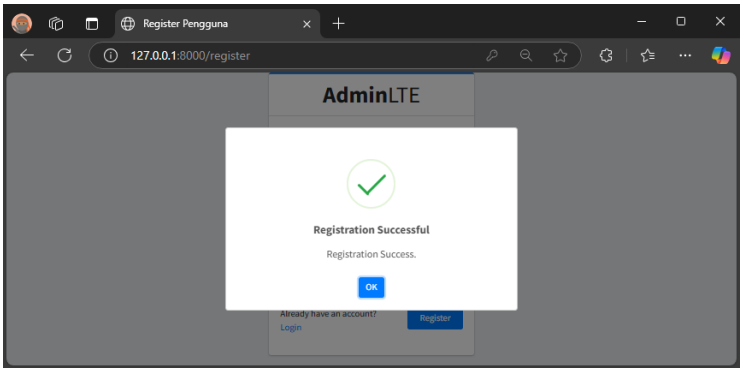
➢ Menambahkan tombol Sign Up/Register di view login :



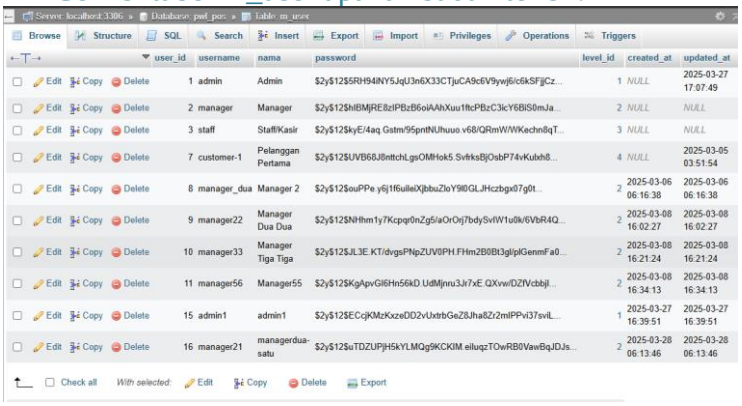
➢ Halaman Register :



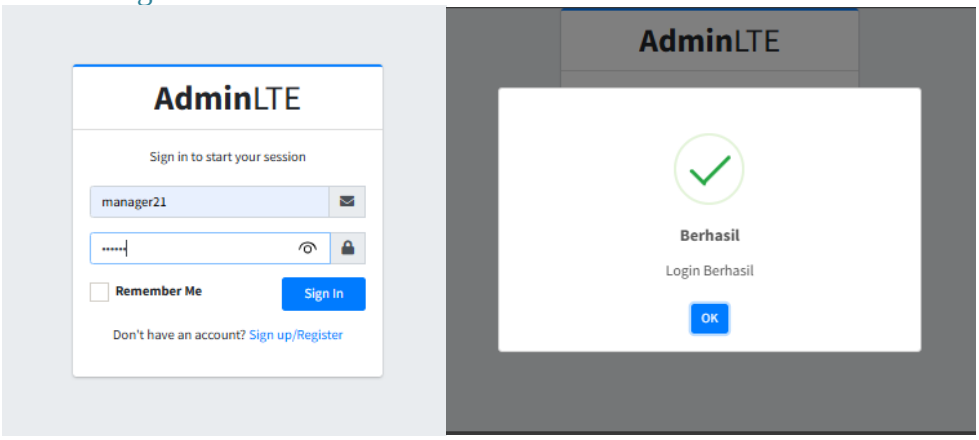
➢ Register berhasil :



➢ Cek isi tabel m_user apakah sudah terisi :



- Cek login user baru:



3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian