LAPORAN PRAKTIKUM

JOBSHEET VII: AUTHENTICATION DAN AUTHORIZATION

MATA KULIAH PEMROGRAMAN WEB LANJUT

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.

Disusun oleh :

Nama : Vita Eka Saraswati

NIM : 2341760082

No. Absen : 29

JURUSAN TEKNOLOGI INFORMASI

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

POLITEKNIK NEGERI MALANG

2025

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

Nama	:	Vita Eka Saraswati
NIM / Kelas	:	2341760082/SIB 2A
No. Absen	:	29
Mata Kuliah	:	Pemrograman Web Lanjut (PWL)

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          ],
67
68          // 'users' => [
69          //     'driver' => 'database',
70          //     'table' => 'users',
71          // ],
72      ],
```

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

```
UserModel.php M
Minggu7 > Jobsheet7 > app > Models > UserModel.php > PHP Intelephense > 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\BelongsTo;
8   use App\Models\LevelModel;
9   use Illuminate\Foundation\Auth\User as Authenticatable; //implementasi class Authenticatable
10
    49 references | 0 implementations
11  class UserModel extends Authenticatable
12  {
13      use HasFactory;
14
        0 references
15      protected $table = 'm_user';       // Mendefinisikan nama tabel yang digunakan oleh model ini
        0 references
16      protected $primaryKey = 'user_id';  // Mendefinisikan primary key dari tabel yang digunakan
        0 references
17      protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_ad', 'updated_ad']; // Mendefinisikan field yang dapat diisi saat melakukan insert atau update data
        0 references
18      protected $hidden = ['password']; // Mendefinisikan field yang tidak akan ditampilkan saat
        0 references
19      protected $casts = ['password' => 'hashed']; //casting password agar otomatis di hash
20
21      //relasi ke tabel level
        0 references | 0 overrides
22      public function level(): BelongsTo
23      {
24          return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
25      }
26  }
```

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

```
PS F:\= KULIAH\SEMESTER 4\3. PRAK JOBSHEET\PWL25\Minggu7\Jobsheet7> php artisan make:controller AuthController

  INFO  Controller [F:\= KULIAH\SEMESTER 4\3. PRAK JOBSHEET\PWL25\Minggu7\Jobsheet7\app\Http\Controllers\AuthController.php] created successfully.
```

Minggu7 > Jobsheet7 > app > Http > Controllers > AuthController.php > PHP Intelephense > AuthController

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

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

    0 references | 0 overrides
    public function postlogin(Request $request): JsonResponse|mixed|Redirector|RedirectRes…
    {
        if($request->ajax() || $request->wantsJson()) {
            $credentials = $request->only('username', 'password');

            if (Auth::attempt($credentials)) {
                return response()->json([
                    'status' => true,
                    'message' => 'Login Berhasil',
                    'redirect' => url(path: '/')
                ]);
            }
            return response()->json([
                'status' => false,
                'message' => 'Login Gagal'
            ]);
        }
        return redirect(to: 'login');
    }

    0 references | 0 overrides
    public function logout(Request $request): Redirector|RedirectResponse
    {
        Auth::logout();

        $request->session()->invalidate();
        $request->session()->regenerateToken();
        return redirect(to: 'login');
    }
}
```

4. Setelah kita membuat `AuthController.php`, kita buat view untuk menampilkan halaman login. View kita buat di `auth/login.blade.php` , tampilan login bisa kita ambil dari contoh login di template **AdminLTE** seperti berikut (pada contoh login ini, kita gunakan page `login-V2` di **AdminLTE**)

Minggu7 > Jobsheet7 > resources > views > auth > login.blade.php

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

```
        <!-- Google Font: Source Sans Pro -->
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
        <!-- Font Awesome -->
        <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/fontawesome-free/css/all.min.css') }}">
        <!-- icheck bootstrap -->
        <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
        <!-- SweetAlert2 -->
        <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
        <!-- Theme style -->
        <link rel="stylesheet" href="{{ asset(path: 'adminlte/dist/css/adminlte.min.css') }}">
    </head>
    <body class="hold-transition login-page">
        <div class="login-box">
            <!-- /.login-logo -->
            <div class="card card-outline card-primary">
                <div class="card-header text-center">
                    <a href="{{ url(path: '/') }}" class="h1"><b>Admin</b>LTE</a>
                </div>
                <div class="card-body">
                    <p class="login-box-msg">Sign in to start your session</p>
                    <form action="{{ url(path: '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(path: 'adminlte/plugins/jquery/jquery.min.js') }}"></script>
        <!-- Bootstrap 4 -->
        <script src="{{ asset(path: 'adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
        <!-- jquery-validation -->
        <script src="{{ asset(path: 'adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
        <script src="{{ asset(path: 'adminlte/plugins/jquery-validation/additional-methods.min.js') }}"></script>
        <!-- SweetAlert2 -->
        <script src="{{ asset(path: 'adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
        <!-- AdminLTE App -->
        <script src="{{ asset(path: '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: {
```

**Selengkapnya:*
https://github.com/vitasaraswati/PWL25/blob/master/Minggu7/Jobsheet7/resources/views/auth/login.blade.php

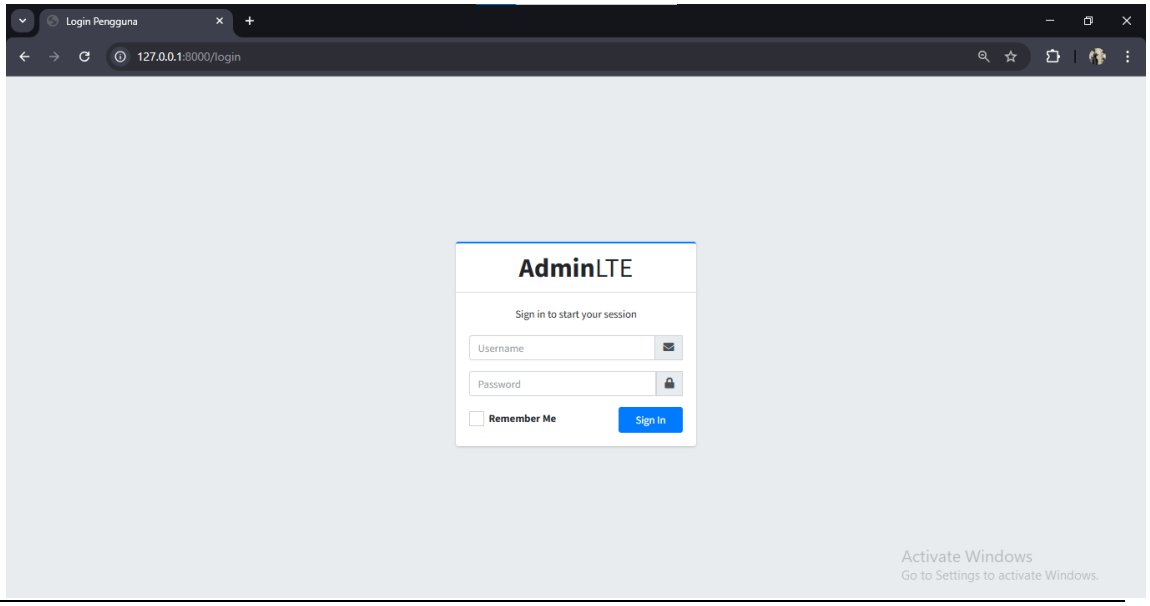
5. Kemudian kita modifikasi `route/web.php` agar semua route masuk dalam auth

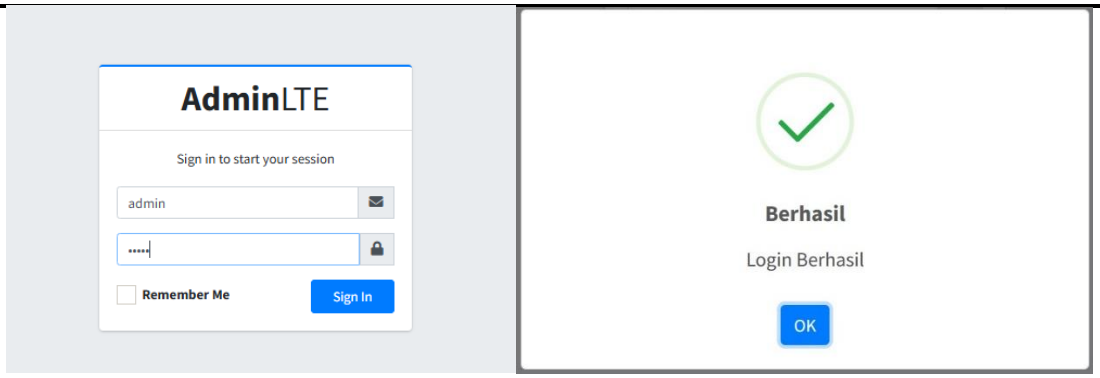
```
use App\Http\Controllers\AuthController; //jobsheet 7 prak1 no 5
```

```
//JOBSHEET 7
//praktikum 1 - no 5
Route::pattern('id', '[0-9]+');

Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('logout', [AuthController::class, 'logout'])->name('logout');

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

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

    Route::group(['prefix' => 'user'], function (): void {
        Route::get('/', [UserController::class, 'index']); // Halaman awal user
        Route::post('/list', [UserController::class, 'list']); // Data JSON untuk datatables
        Route::get('/create', [UserController::class, 'create']); // Form tambah user
        Route::post('/', [UserController::class, 'store']); // Simpan user baru
        Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Form tambah user Ajax
        Route::post('/ajax', [UserController::class, 'store_ajax']); // Simpan user Ajax
        Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Form edit user Ajax
        Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // Update user Ajax
        Route::get('/{id}', [UserController::class, 'show']); // Detail user
        Route::get('/{id}/edit', [UserController::class, 'edit']); // Form edit user
        Route::put('/{id}', [UserController::class, 'update']); // Update user
        Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Confirm delete Ajax
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // Hapus user Ajax
        Route::delete('/{id}', [UserController::class, 'destroy']); // Hapus user
    });

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

```

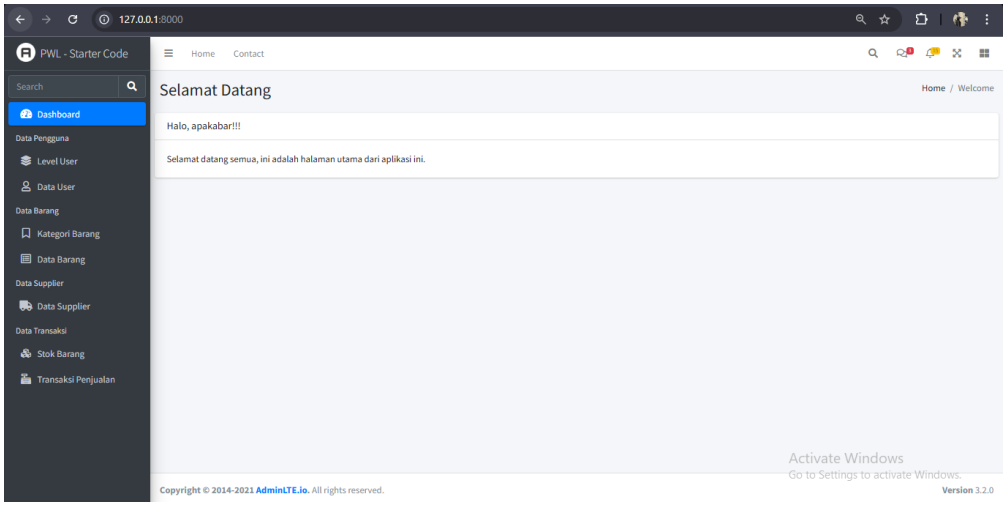
**Selengkapnya* :
https://github.com/vitasaraswati/PWL25/commit/e97ccb7dbe9db7e493c9a7a3cddef743a9a636de#diff-c67e59410fb8e105e1a209c7520ebd7eb5880f9d8d9b55d334f1083711fe6673

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





Menampilkan home setelah user berhasil login ke sistem



Tugas 1 – Implementasi Authentication :

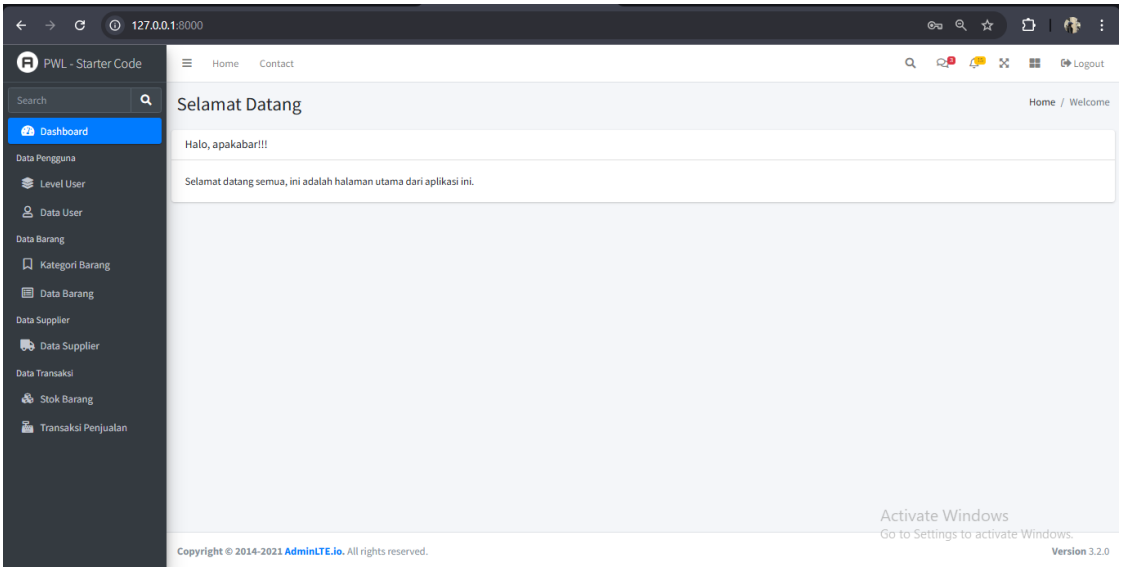
1. Silahkan implementasikan proses login pada project kalian masing-masing
2. Silahkan implementasi proses logout pada halaman web yang kalian buat
   - Modifikasi right navbar dengan menambahkan button logout pada file layouts/header.blade.php

```
135      <!-- Logout Button -->
136      <li class="nav-item">
137        <a class="nav-link" href="{{ route(name: 'logout') }}" onclick="event.preventDefault(); document.getElementById('logout-form').submit();" role="button">
138          <i class="fas fa-sign-out-alt"></i> Logout
139        </a>
140          <form id="logout-form" action="{{ route(name: 'logout') }}" method="POST" style="display: none;">
141        @csrf
142          </form>
143      </li>
144    </ul>
145  </nav>
146  <!-- /.navbar -->
```

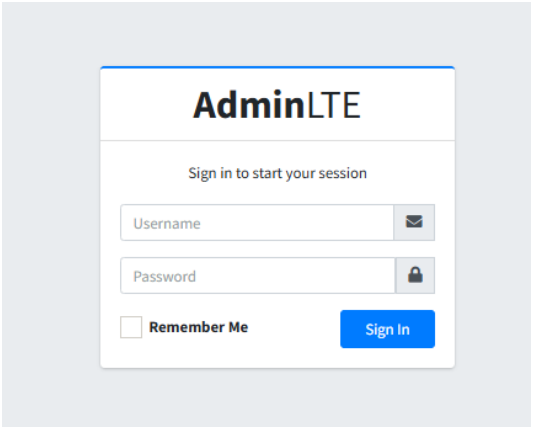
   - Modifikasi route pada file web.php

```
33  Route::POST('logout', [AuthController::class, 'logout'])->name('logout'); //ubah methodnya jadi POST
34
```

- Tampilan web browser dengan button logout



Saat user mengklik logout maka akan muncul seperti gambar di bawah ini (kembali ke halaman login)



3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
   - **Route method post /logout pada file web.php digunakan untuk membuat jalur atau pengalamatan ke function logout di file AuthController.php**
   - **Function logout pada file AuthController.php**
     Fungsi ini menggunakan Auth::logout() untuk mengakhiri sesi pengguna, menghapus sesi, dan mengarahkan pengguna ke halaman login.
   - **Menambahkan button "logout" pada file header.blade.php**
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
class UserModel extends Authenticatable
{
    use HasFactory;

    protected $table = 'm_user';      // Mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primaryKey = 'user_id';  // Mendefinisikan primary key dari tabel yang digunakan
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_ad', 'updated_ad']; // Mendefinisikan field yang dapat diisi saat melakukan insert atau update data
    protected $hidden = ['password']; // Mendefinisikan field yang tidak akan ditampilkan saat
    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');
    }

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

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

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

```
PS F:\= KULIAH\SEMESTER 4\3. PRAK JOBSHEET\PWL25\Minggu7\Jobsheet7> php artisan make:middleware AuthorizeUser

  INFO  Middleware [F:\= KULIAH\SEMESTER 4\3. PRAK JOBSHEET\PWL25\Minggu7\Jobsheet7\app\Http\Middleware\AuthorizeUser.php] created successfully.
```

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(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

4. Kita daftarkan ke `app/Http/Kernel.php` untuk *middleware* yang kita buat barusan

```php
    protected $middlewareAliases = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'authorize' => \App\Http\Middleware\AuthorizeUser::class, //middleware yg kiat 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,
    ];
}
```

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

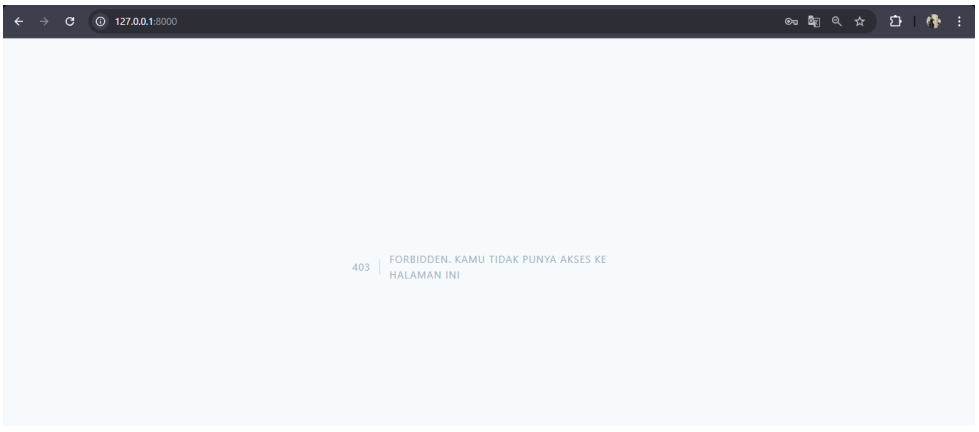
level_id	level_kode	level_nama	created_at	updated_at	deleted_at
1	ADM	Administrator	NULL	NULL	NULL
2	MNG	Manager	NULL	NULL	NULL
3	STF	Staf	NULL	2024-08-16 01:49:20	NULL
4	KSR	Kasir	NULL	NULL	NULL

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

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

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

    Route::group(['prefix' => 'user'], function (): void {
        Route::get('/', [UserController::class, 'index']); // Halaman awal user
        Route::post('/list', [UserController::class, 'list']); // Data JSON untuk datatables
        Route::get('/create', [UserController::class, 'create']); // Form tambah user
        Route::post('/', [UserController::class, 'store']); // Simpan user baru
        Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Form tambah user Ajax
        Route::post('/ajax', [UserController::class, 'store_ajax']); // Simpan user Ajax
        Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Form edit user Ajax
        Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // Updat   e user Ajax
        Route::get('/{id}', [UserController::class, 'show']); // Detail user
        Route::get('/{id}/edit', [UserController::class, 'edit']); // Form edit user
        Route::put('/{id}', [UserController::class, 'update']); // Update user
        Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Confirm delete Ajax
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // Hapus user Ajax
        Route::delete('/{id}', [UserController::class, 'destroy']); // Hapus user
    });

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

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



Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?
    - Authorization berfungsi untuk membatasi hak akses pengguna dalam mengakses halaman/fitur web sesuai role tertentu
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
    - Authorization berfungsi untuk membatasi hak akses pengguna dalam mengakses halaman/fitur web
    - Middleware AuthorizeUser dan digunakan untuk otorisasi pengguna berdasarkan role (level_kode).
    - Karena telah dilakukan modifikasi pada route level, sehingga halaman web hanya dapat ditampilkan ke role yang memiliki level_kode : ADM (admin).
    - Apa yang terjadi jika kita mencoba mengakses halaman web tetapi login sebagai manager atau selain admin? Web akan memunculkan pesan 403 forbidden tidak memiliki akses ke halaman web
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
    //relasi ke tabel level
    0 references | 0 overrides
    public function level(): BelongsTo
    {
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
    }

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

    //Cek apakah user memeliki role tertentu
    0 references | 0 overrides
    public function hasRole($role): bool{
        return $this->level->level_id == $role;
    }

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

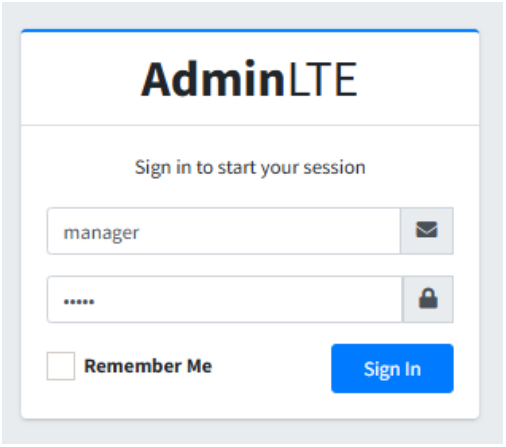
2. Selanjutnya, Kita modifikasi middleware `AuthorizeUser.php` dengan kode berikut

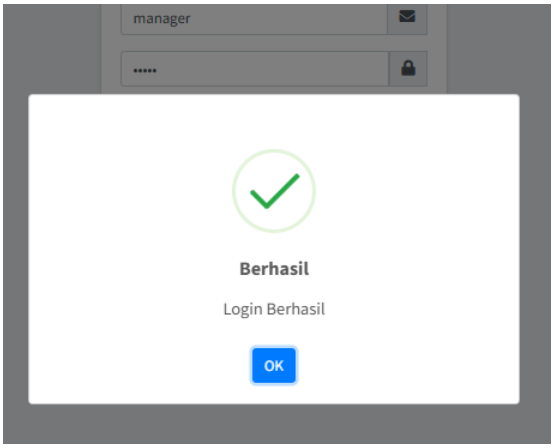
```php
Minggu7 > Jobsheet7 > app > Http > Middleware > AuthorizeUser.php > ...
<?php

namespace App\Http\Middleware;

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

1 reference | 0 implementations
class AuthorizeUser
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next, ...$roles): Response
    {
        $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login

        if (in_array(needle: $user_role, haystack: $roles)) { // cek apakah level_kode user ada di dalam array roles
            return $next($request); // jika ada, maka lanjutkan request
        }

        // jika tidak punya role, maka tampilkan error 403
        abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
    }
}
```

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

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

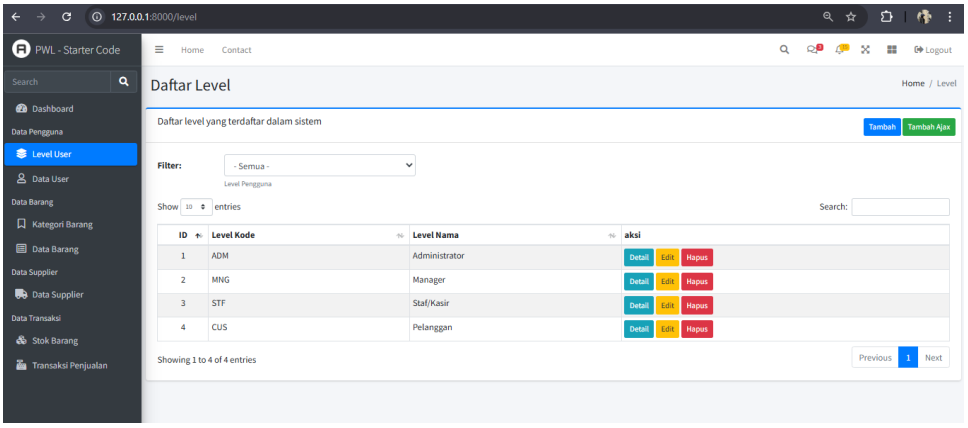
4. Sekarang kita sudah bisa memberikan hak akses menu/route ke beberapa level user
   - Login dengan data manager





   - Akses menu Data Level

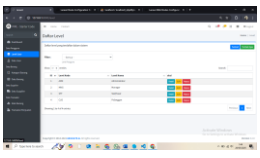
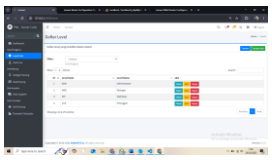
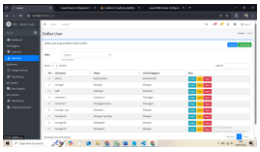
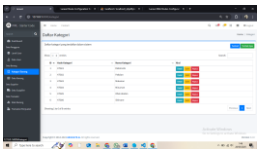
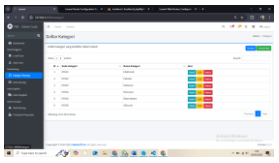
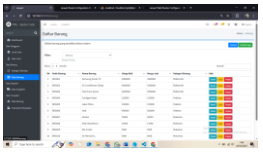
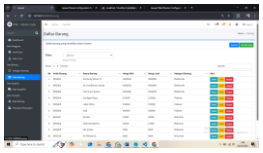
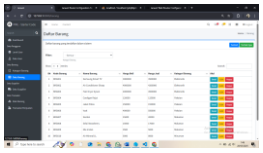
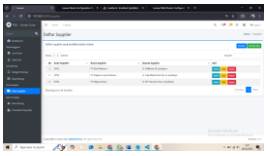
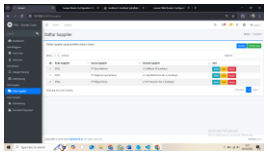
     Halaman Data Level juga dapat digunakan oleh user dengan level_kode MNG karena telah ditambahkan multi level authorization



     Namun jika login selain menggunakan data manager dan admin maka halaman web menampilkan 403 forbidden.

Tugas 3 – Implementasi Multi-Level Authorization :

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

Menu	Level admin (ADM)	Level Manager (MNG)	Level Staff (STF)
Level user			
Data user			
Data kategori			
Data barang			 **Hanya lihat data tanpa create, edit, delete*
Data supplier			

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

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.
   - Modifikasi file controller AuthController.php

```
1 reference | 0 overrides
public function register(): mixed|View
{
    $levels = LevelModel::select('level_id', 'level_nama')->get();

    return view(view: 'auth.register')->with('levels', $levels);
}
```

```php
public function postRegister(Request $request): JsonResponse|mixed
{
    $validator = Validator::make($request->all(), [
        'username' => 'required|string|unique:m_user,username',
        'nama' => 'required|string|max:255',
        'password' => 'required|string|min:5|confirmed',
        '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(value: $request->password),
        'level_id' => $request->level_id
    ]);

    return response()->json([
        'status' => true,
        'message' => 'Registration Success.',
        'redirect' => url(path: '/login')
    ]);
}
}
```

- Modifikasi file web.php dengan menambahkan route register

```php
//Route Register akun
Route::get('/register', [AuthController::class, 'register'])->name('register');
Route::post('/register', [AuthController::class, 'postRegister']);
```

- Buat file views/Auth/register.blade.php untuk form register

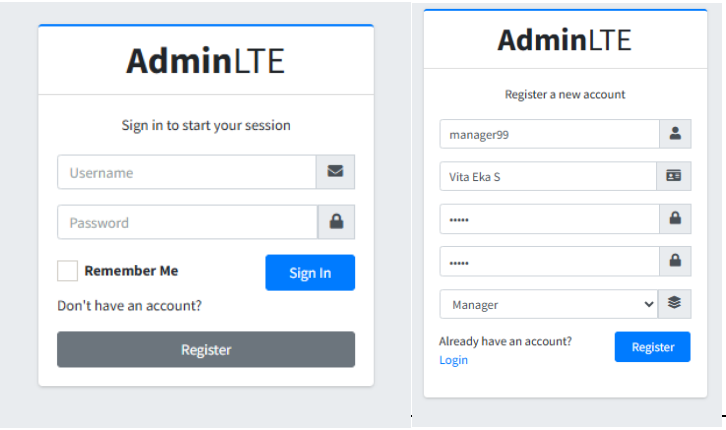
```
Minggu7 > Jobsheet7 > resources > views > auth > register.blade.php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Register Pengguna</title>
    <!-- Google Font: Source Sans Pro -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
    <!-- Font Awesome -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/fontawesome-free/css/all.min.css') }}">
    <!-- icheck bootstrap -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
    <!-- SweetAlert2 -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset(path: 'adminlte/dist/css/adminlte.min.css') }}">
</head>
```

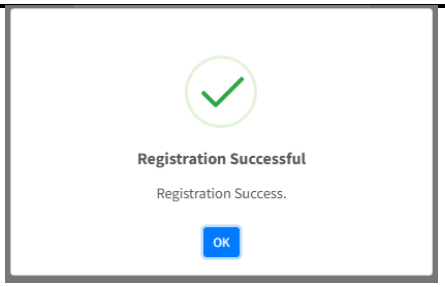
https://github.com/vitasaraswati/PWL25/blob/master/Minggu7/Jobsheet7/resources/views/auth/register.blade.php

- Modifikasi file views/Auth/login.blade.php

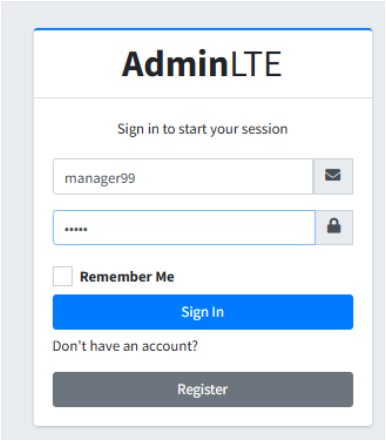
```
<div class="col-12">
    <button type="submit" class="btn btn-primary btn-block">Sign In</button>
</div>
<div class="col-12 mt-1">
    <p>Don't have an account?</p>
    <a href="{{ url(path: '/register') }}" class="btn btn-secondary btn-block">Register</a>
</div>
```

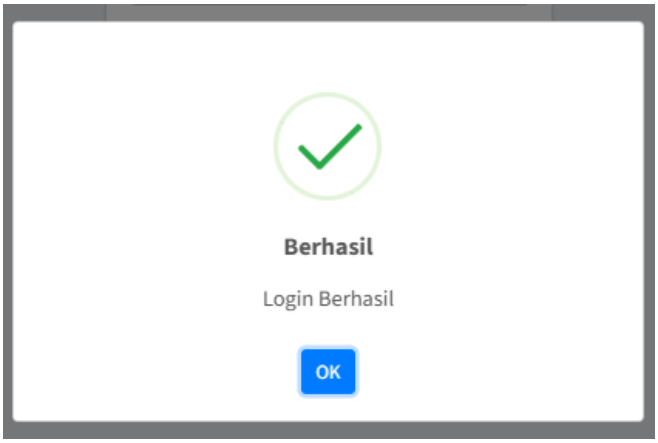
2. Screenshot hasil yang kalian kerjakan
   - Tampilan login dan form register



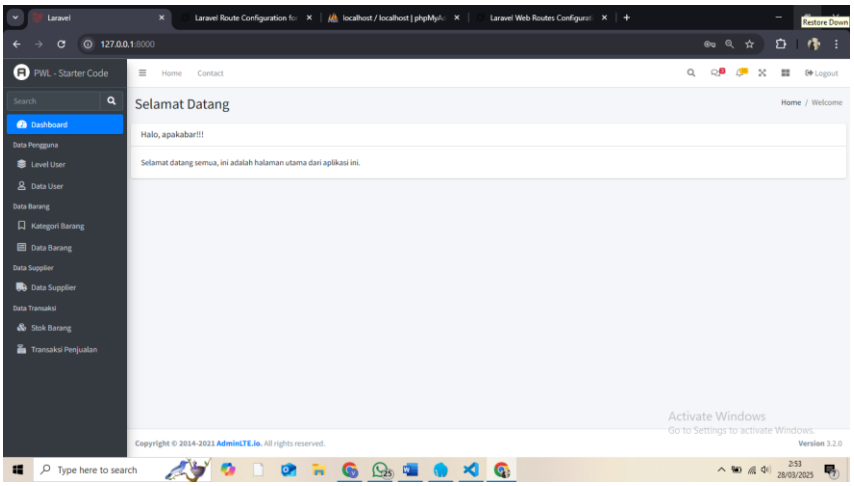


- Login ke sistem dengan akun yang baru saja dibuat





- Dashboard setelah berhasil login



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

**** Sekian, dan selamat belajar ****