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: Yonanda Mayla Rusdiaty
Kelas: SIB 2A/30

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 "*guards*" dan "*providers*". *Guards* 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.

- **NOTES**:
  - Authentication: proses untuk **memverifikasi identitas pengguna** yang mencoba mengakses sistem (misalnya, dengan memeriksa kredensial seperti username dan password). Menentukan "siapa" pengguna tersebut
  - Authorization: proses yang menentukan **hak akses** atau izin yang dimiliki pengguna setelah mereka berhasil diautentikasi. Ini berkaitan dengan "apa" yang boleh dilakukan pengguna dalam sistem. Mengatur dan membatasi akses pengguna sesuai dengan peran (role) atau level mereka

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          ],
```

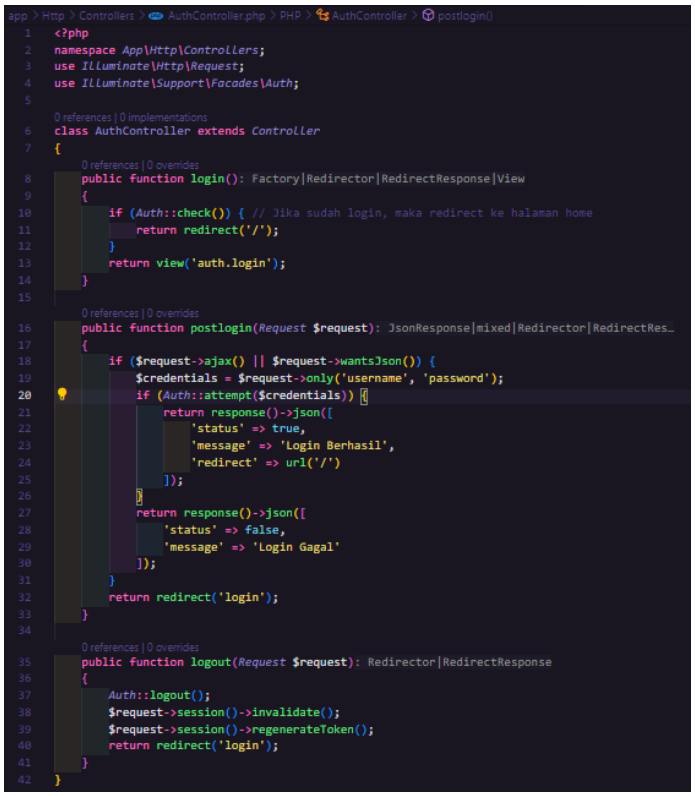
Pada bagian ini kita sesuaikan dengan Model untuk tabel m_user yang sudah kita buat

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

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

```
You, 6 seconds ago | 1 author (You)
<?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; //impelemntasi auth

19 references | 0 implementations | You, 6 seconds ago | 1 author (You)
class UserModel extends Authenticatable
{
    use HasFactory;
    0 references
    protected $table = 'm_user';
    0 references
    protected $primaryKey = 'user_id';
    0 references
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];
    0 references
    protected $hidden = ['password']; // Jangan ditampilkan saat select
    0 references
    protected $casts = ['password' => 'hashed']; // Casting password agar otomatis di-hash

    /**
     * Relasi ke tabel level
     */
    0 references | 0 overrides
    public function level(): BelongsTo      You, 7 hours ago • feat: initial commit
    {
        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(): Factory|Redirector|RedirectResponse|View
    {
        if (Auth::check()) { // Jika sudah login, maka redirect ke halaman home
            return redirect('/');
        }
        return view('auth.login');
    }

    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('/')
                ]);
            }
            return response()->json([
                'status' => false,
                'message' => 'Login Gagal'
            ]);
        }
        return redirect('login');
    }

    public function logout(Request $request): Redirector|RedirectResponse
    {
        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**)

```blade
<!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('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 -->
```

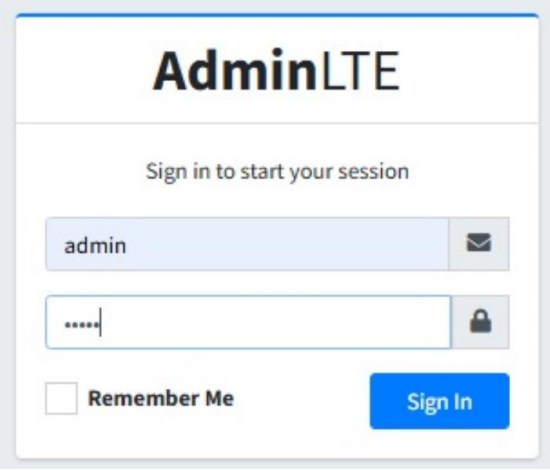
```
<!-- jQuery -->
<script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<!-- jquery-validation -->
<script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
<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>
```

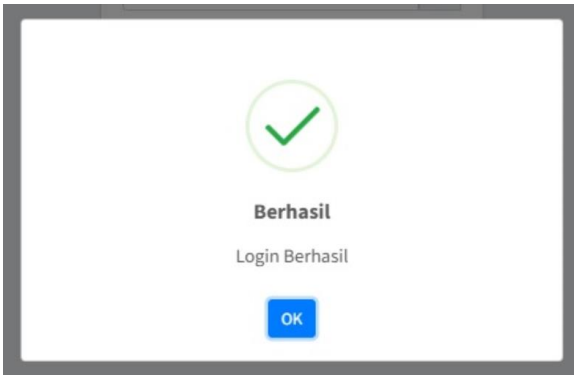
5. Kemudian kita modifikasi `route/web.php` agar semua route masuk dalam auth

```
// JB 7 Autentikasi
Route::pattern('id', '[0-9]+'); // Menambahkan pola untuk parameter id
Route::get('/login', [AuthController::class, 'login'])->name('login');
Route::post('/login', [AuthController::class, 'postlogin']);
Route::post('/logout', [AuthController::class, 'logout'])->name('logout');

Route::middleware(['auth'])->group(function (): void {
    Route::get('/', [WelcomeController::class, 'index']);

    // Tambahkan route grup untuk user yg memerlukan autentikasi disini
    Route::group(['prefix' => 'user'], function (): void {
        Route::get('/user', [UserController::class, 'index']); // menampilkan halaman awal user
        Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
        Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
        Route::get('/create_ajax', [UserController::class, 'create_ajax']); // menampilkan halaman form tambah user dengan ajax
        Route::post('/ajax', [UserController::class, 'store_ajax']); // menyimpan data user baru dengan ajax
        Route::post('/', [UserController::class, 'store']); // menyimpan data user baru
        Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user
        Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user
        Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user dengan ajax
        Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // menyimpan perubahan data user
        Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // menampilkan konfirmasi hapus user dengan ajax
        Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // menghapus data user AJAX
        Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
        Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
    });
});      You, 8 hours ago • feat: initial commit
```

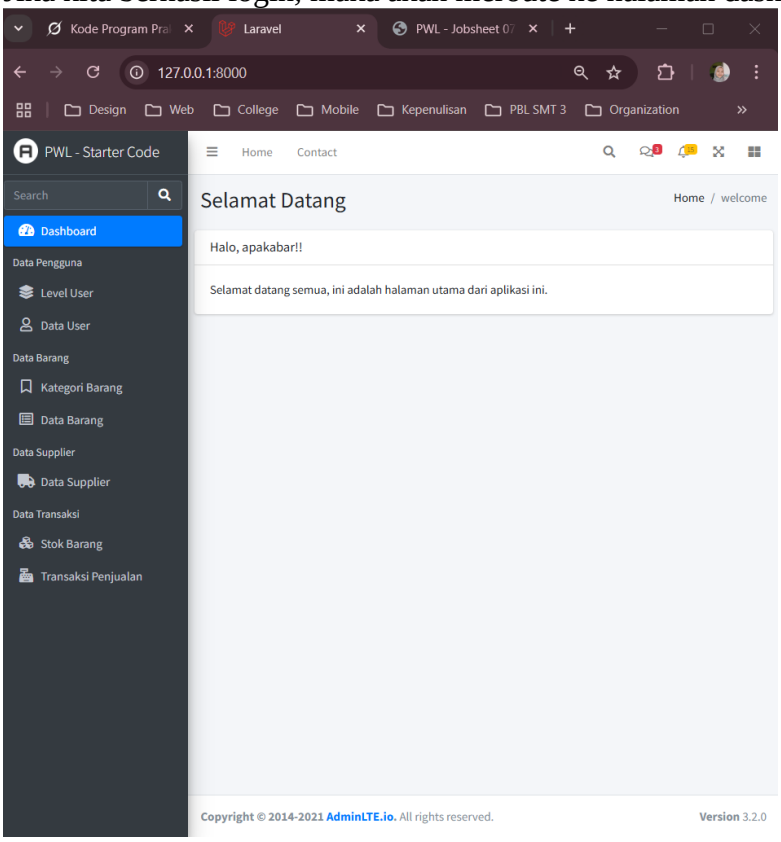
6. Ketika kita coba mengakses halaman `localhost/PWL_POS/public` makan akan tampil halaman awal untuk login ke aplikasi

Jawab:





Jika kita berhasil login, maka akan meroute ke halaman dashboard



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

   Jawab:

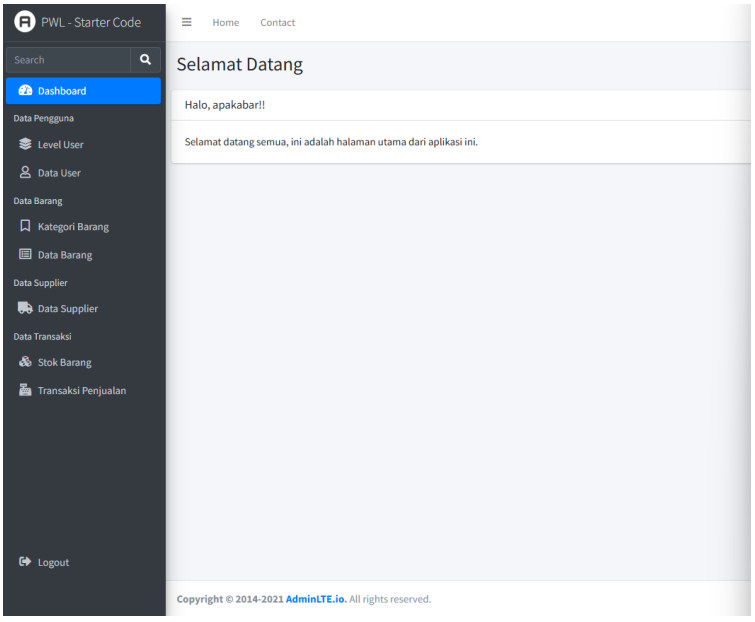
   a. Menambahkan button logout pada sidebar.blade

```
<br><br><br><br><br><br><br><br><br>
<!-- Logout Menu Item -->
<li class="nav-item">
    <a href="#" class="nav-link"
        onclick="event.preventDefault(); document.getElementById('logout-form').submit();">
        <i class="nav-icon fas fa-sign-out-alt"></i>
        <p>Logout</p>
    </a>

    <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">
        @csrf
    </form>
</li>
```

   b. Memodifikasi route pada web.php menjadi POST

```
Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
```

   c. Tampilan logout sidebar



   Jika diklik logout otomatis akan keluar dan kembali ke halaman login

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

   Jawab:

   a. Pengguna mengklik tombol "Logout" di sidebar

   b. Route::post('/logout', [AuthController::class, 'logout'])->name('logout'); pada web.php digunakan untuk proses logout dari sistem. Disini dia menggunakan method POST bukan GET karena logout adalah tindakan untuk mengakhiri sesi.

c. Route ini akan memanggil method /logout yg ada dalam AuthController (method yg menghapus data sesi pengguna dan mengakhiri status login)
d. AuthController menjalankan proses logout dan mengarahkan pengguna ke halaman login

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
 */
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() : string
{
    return $this->level->level_nama;
}

/**
 * Cek apakah user memiliki role tertentu
 */
0 references | 0 overrides
public function hasRole(string $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`

```
PS D:\Apps\laragon\www\PWL2025\week 7\jobsheet> php artis
an make:middleware AuthorizeUser

  INFO  Middleware [D:\Apps\laragon\www\PWL2025\week 7\j
obsheet\app\Http\Middleware\AuthorizeUser.php] created su
ccessfully.
```

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)  $next
     */
    public function handle(Request $request, Closure $next, $Role = ' '): Response
    {
        $user = $request->user(); //ambil user yang sedang login
        if ($user-> hasRole($Role)) { //cek role user
            return $next($request); //lanjutkan ke controller
        }
        // Jika tidak memiliki role yang sesuai, tampilkan pesan error
        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,
    '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,
    '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

level_id	level_kode	level_nama	created_at	updated_at
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staf/Kasir	NULL	NULL
4	CUS	Customer	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

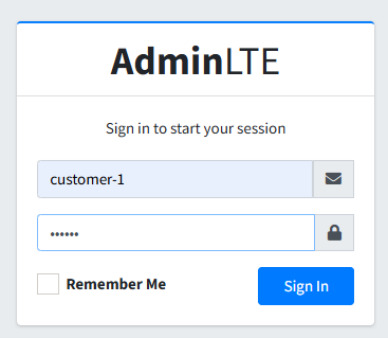
```php
Route::middleware(['auth'])->group(function (): void { //artinya semua route di dalam grup ini harus terautentikasi
    Route::get('/', [WelcomeController::class, 'index']);
    // route level

    // artinya semua route di dalam grup ini harus punya role ADM (admin)
    Route::middleware(['authorize:ADM'])->group(function(): void {
    // Tambahkan route grup untuk level
    Route::group(['prefix' => 'level'], function (): void {
        Route::get('/', [LevelController::class, 'index']); // Menampilkan halaman tabel level
        Route::post('/list', [LevelController::class, 'list']); // Mengambil data level untuk DataTables
        Route::get('/create', [LevelController::class, 'create']); // Menampilkan form tambah level
        Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan form tambah level ajax
        Route::post('/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data level baru ajax
        Route::post('/', [LevelController::class, 'store']); // Menyimpan data level baru
        Route::get('/{id}', [LevelController::class, 'show']); // Menampilkan detail level
        Route::get('/{id}/edit', [LevelController::class, 'edit']); // Menampilkan form edit level
        Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan form edit level ajax
        Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan level ajax
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Menampilkan konfirmasi hapus level
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Menghapus level ajax
        Route::put('/{id}', [LevelController::class, 'update']); // Menyimpan perubahan level
        Route::delete('/{id}', [LevelController::class, 'destroy']); // Menghapus level
    });
    });
});      You, 9 hours ago • feat: initial commit
```

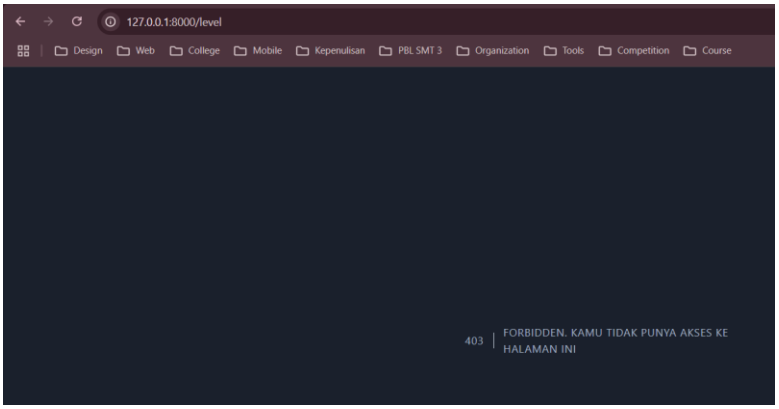
Pada kode yang ditandai merah, terdapat `authorize:ADM` . Kode `ADM` adalah nilai dari `level_kode` pada tabel `m_level`. Yang artinya, user yang bisa mengakses route untuk manage data level, adalah user yang memiliki level sebagai Administrator.

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

Jawab:

Disini saya mencoba login menggunakan akun customer, dan akses ditolak





Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?

   Jawab: Authorization adalah proses yang dilakukan setelah authentication berhasil, untuk menentukan hak akses pengguna dalam sistem sesuai dengan peran (role) atau level yang dimilikinya. alam konteks ini, kita membatasi akses ke rute atau fitur tertentu berdasarkan level pengguna, seperti Administrator, Manager, Staf, atau Cust

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

   Jawab:

   a. Memodifikasi UserModel dengan menambahkan function untuk relasi ke tabel m_level (mendapatkan role sesuai yg ada di tabel m_level dan memeriksa role hasRole()) → bertujuan untuk mengelola role pengguna agar bisa diakses oleh middleware

   b. Membuat Middleware AuthorizeUser → untuk memeriksa otorisasi berdasarkan role

   c. Edit Middleware AuthorizeUser.php → Menambahkan logika pengecekan role dan pengalihan jika tidak punya hak akses.

   d. Daftarkan Middleware di Kernel.php → Mendaftarkan middleware agar bisa digunakan di route.

   e. Tabel m_level → Memahami struktur data role apa saja yang akan digunakan untuk otorisasi

   f. Modifikasi Route → Mengatur route agar hanya pengguna dengan role tertentu yang bisa mengakses.

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
 */
0 references | 0 overrides
public function getRoleName() : string
{
    return $this->level->level_nama;
}

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

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

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

```php
You, 1 second ago | 1 author (You)
<?php

namespace App\Http\Middleware;

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

1 reference | 0 implementations | You, 1 second ago | 1 author (You)
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 role user yang login
        if (in_array(needle: $user_role, haystack: $roles)) { // jika role user ada di dalam array $roles
            return $next($request); // lanjutkan ke request selanjutnya
        }

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

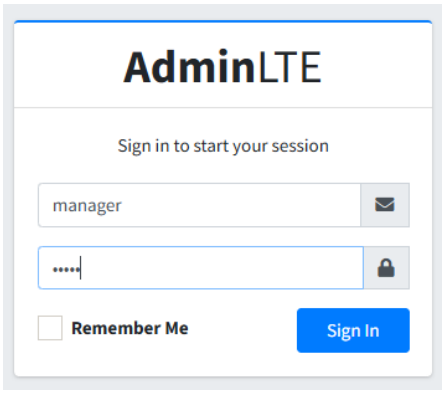
3. Setelah itu tinggal kita perbaiki `route/web.php` sesuaikan dengan role/level yang diinginkan. Contoh

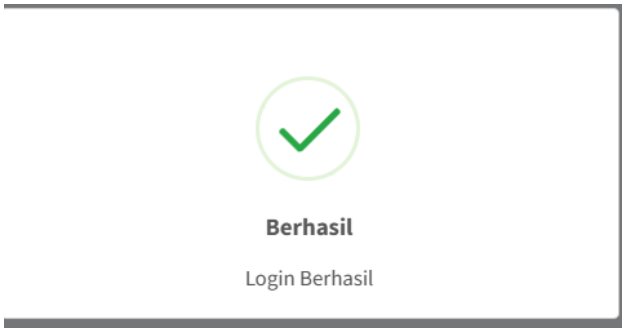
```php
Route::middleware(['auth'])->group(function (): void { //artinya semua route di dalam grup ini harus terautentikasi
    Route::get('/', [WelcomeController::class, 'index']);
    // route level

    // artinya semua route di dalam grup ini harus punya role ADM (admin) dan MNG (manager)
    Route::middleware(['authorize:ADM,MNG'])->group(function (): void {
        // Tambahkan route grup untuk barang
        Route::group(['prefix' => 'barang'], function (): void {
            Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal barang
            Route::post("/list", [BarangController::class, 'list']); // menampilkan data barang dalam bentuk json untuk
            Route::get('/create', [BarangController::class, 'create']); // menampilkan halaman form tambah barang
            Route::post('/', [BarangController::class, 'store']); // menyimpan data barang baru
            Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // Menampilkan halaman form tambah ba
            Route::post('/ajax', [BarangController::class, 'store_ajax']); // Menyimpan data barang baru Ajax
            Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // Menampilkan halaman form edit bara
            Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // Menyimpan perubahan data bara
            Route::get('/{id}', [BarangController::class, 'show']); // menampilkan detail barang
            Route::get('/{id}/edit', [BarangController::class, 'edit']); // menampilkan halaman form edit barang
            Route::put('/{id}', [BarangController::class, 'update']); // menyimpan perubahan data barang
            Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // Untuk tampilkan form confirm
            Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // Untuk hapus data barang Aja
            Route::delete('/{id}', [BarangController::class, 'destroy']); // menghapus data barang
        });
    });
});      You, 10 hours ago • feat: initial commit
```

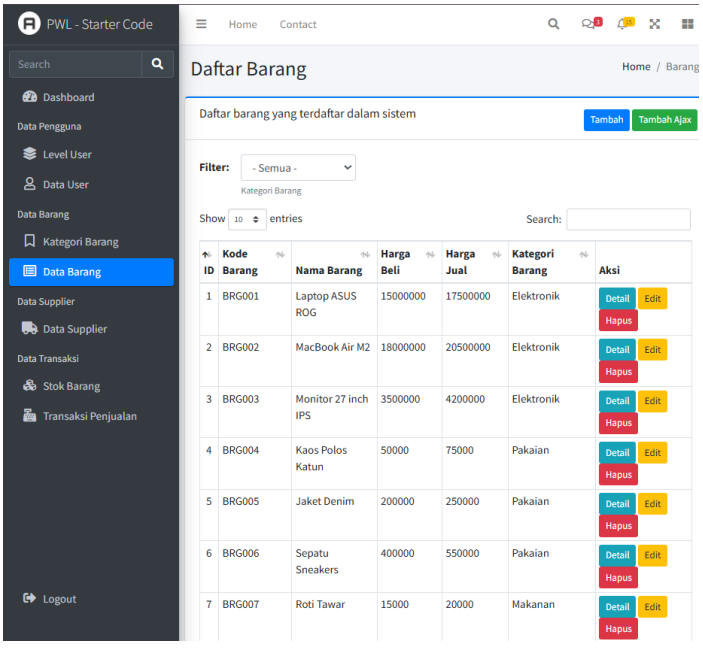
4. Sekarang kita sudah bisa memberikan hak akses menu/route ke beberapa level user

   Jawab:

   a. Disini saya mencoba login dengan akun manager untuk membuka halaman barang





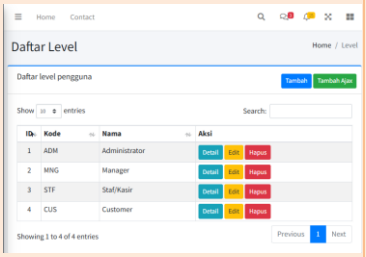
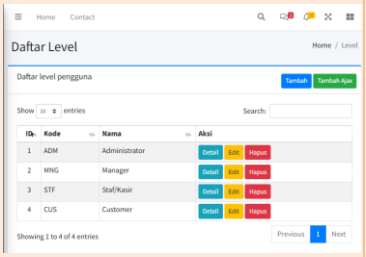
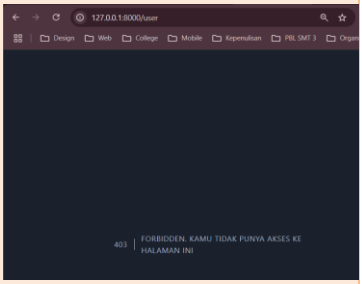
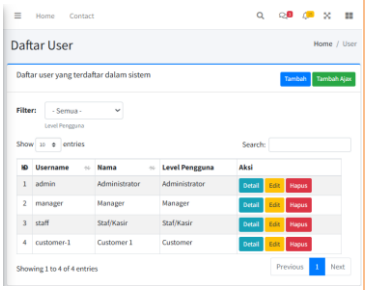
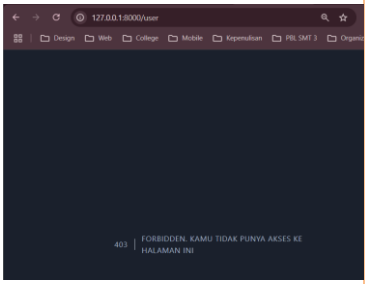
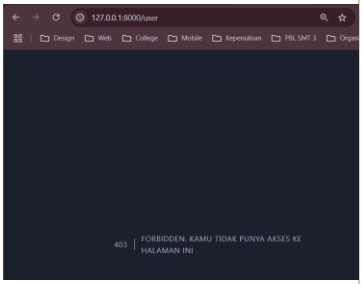
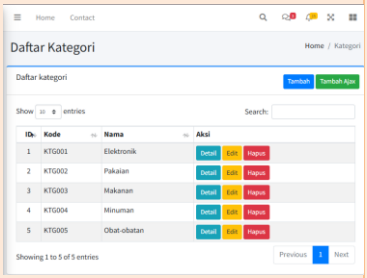
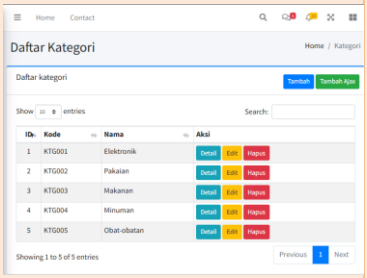
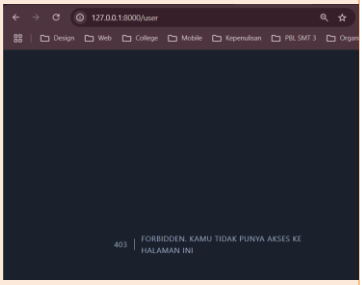


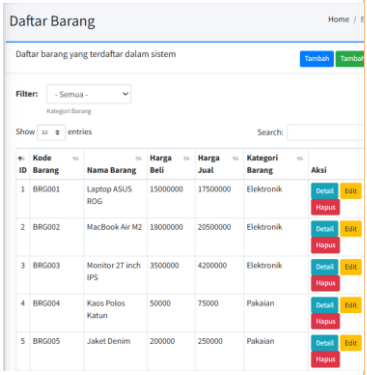
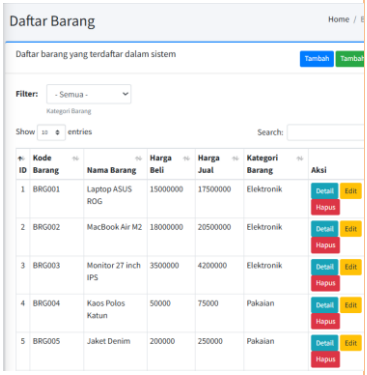
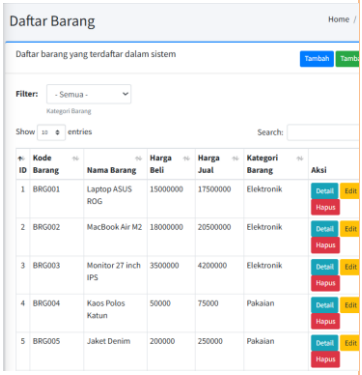
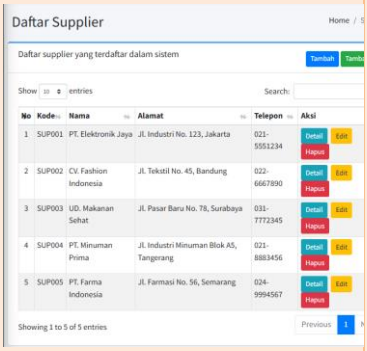
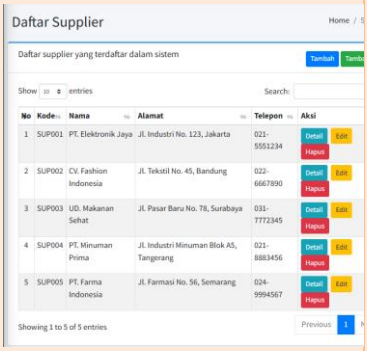
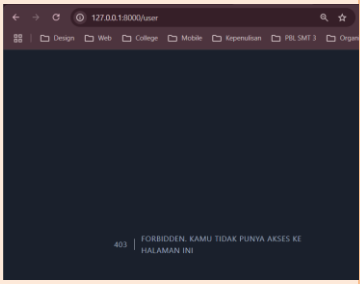
Dan hasilnya bisa dibuka, karena autorisasi/hak akses pengguna disini yaitu admin dan manager

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

Jawab:

Menu	Level Admin (ADM)	Level Manager (MNG)	Level Staff (STF)
Level User			
Data User			
Data Kategori			

Data Barang			
Data Supplier			

Notes: Staf hanya dapat read pada data barang. Tidak bisa create, update, dan delete

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

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

   Jawab:

   a. Membuat controller untuk menangani register

```
eek 7\jobsheet> php artisan make:controller Re
gisterController

  INFO  Controller [D:\Apps\laragon\www\PWL20
25\week 7\jobsheet\app\Http\Controllers\Regist
erController.php] created successfully.
```

```php
<?php

namespace App\Http\Controllers;

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

2 references | 0 implementations
class RegisterController extends Controller
{
    1 reference | 0 overrides
    public function showRegistrationForm(): Factory|View
    {
        // Ambil level untuk dropdown (level 4 = customer)
        $level = LevelModel::where('level_id', 4)->first();
        return view('auth.register', compact(var_name: 'level'));
    }

    1 reference | 0 overrides
    public function register(Request $request): RedirectResponse
    {
        // Validasi input
        $validator = Validator::make($request->all(), [
            'username' => 'required|string|min:3|max:100|unique:m_user',
            'nama' => 'required|string|max:100',
            'password' => 'required|string|min:5|confirmed',
        ], [
            'username.required' => 'Username harus diisi',
            'username.unique' => 'Username sudah digunakan',
            'nama.required' => 'Nama harus diisi',
            'password.required' => 'Password harus diisi',
            'password.min' => 'Password minimal 5 karakter',
            'password.confirmed' => 'Konfirmasi password tidak cocok',
        ]);

        if ($validator->fails()) {
            return redirect()
                ->route('register')
                ->withErrors($validator)
                ->withInput();
        }

        // Hitung user_id tertinggi
        $maxId = UserModel::max('user_id');
        $nextId = $maxId + 1;

        // Buat user baru (default level 4 = customer)
        $user = new UserModel();
        $user->user_id = $nextId;
        $user->username = $request->username;
        $user->nama = $request->nama;
        $user->password = Hash::make($request->password);
        $user->level_id = 4; // Customer
        $user->save();

        // Redirect ke halaman login dengan pesan sukses
        return redirect()
            ->route('login')
            ->with('success', 'Registrasi berhasil! Silahkan login.');
    }
}
```

b. View untuk registrasi

```
<!DOCTYPE html>      You, 1 second ago • Uncommitted changes
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>PWL 2025 | Registration Page</title>

    <!-- Google Font: Source Sans Pro -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
    <!-- Font Awesome -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
    <!-- icheck bootstrap -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
</head>
<body class="hold-transition register-page">
<div class="register-box">
    <div class="card card-outline card-primary">
        <div class="card-header text-center">
            <a href="{{ url('/') }}" class="h1"><b>PWL</b>2025</a>
        </div>
        <div class="card-body">
            <p class="login-box-msg">Daftar sebagai anggota baru</p>

            @if ($errors->any())
                <div class="alert alert-danger">
                    <ul>
                        @foreach ($errors->all() as $error)
                            <li>{{ $error }}</li>
                        @endforeach
                    </ul>
                </div>
            @endif

            <form action="{{ route('register') }}" method="post">
                @csrf
                <div class="input-group mb-3">
                    <input type="text" name="username" class="form-control" placeholder="Username" value="{{ old('username') }}">
                    <div class="input-group-append">
                        <div class="input-group-text">
                            <span class="fas fa-user"></span>
                        </div>
                    </div>
                </div>
                <div class="input-group mb-3">
                    <input type="text" name="nama" class="form-control" placeholder="Nama Lengkap" value="{{ old('nama') }}">
                    <div class="input-group-append">
                        <div class="input-group-text">
                            <span class="fas fa-user"></span>
                        </div>
                    </div>
                </div>
                <div class="input-group mb-3">
                    <input type="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>
                </div>
                <div class="input-group mb-3">
                    <input type="password" name="password_confirmation" class="form-control" placeholder="Konfirmasi password">
                    <div class="input-group-append">
                        <div class="input-group-text">
                            <span class="fas fa-lock"></span>
                        </div>
```

```
                <div class="row">
                    <div class="col-8">
                        <div class="icheck-primary">
                            <input type="checkbox" id="agreeTerms" name="terms" value="agree" required>
                            <label for="agreeTerms">
                                Saya setuju dengan <a href="#">syarat</a> dan ketentuan
                            </label>
                        </div>
                    </div>
                    <!-- /.col -->
                    <div class="col-4">
                        <button type="submit" class="btn btn-primary btn-block">Daftar</button>
                    </div>
                    <!-- /.col -->
                </div>
            </form>

            <a href="{{ route('login') }}" class="text-center">Saya sudah punya akun</a>
        </div>
        <!-- /.card-body -->
    </div>
    <!-- /.card -->
</div>
<!-- /.register-box -->

<!-- jQuery -->
<script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
<!-- Bootstrap 4 -->
<script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<!-- AdminLTE App -->
<script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
</body>
</html>
```

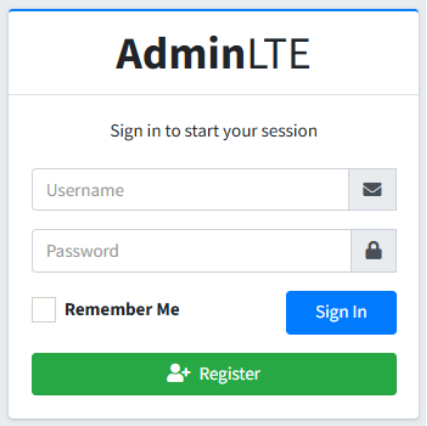
c. Route untuk mengakses halaman registrasi

```
// Routes for registration
Route::get('/register', [App\Http\Controllers\RegisterController::class, 'showRegistrationForm'])->name('register');
Route::post('/register', [App\Http\Controllers\RegisterController::class, 'register']);
```

2. Screenshot hasil yang kalian kerjakan

Jawab:

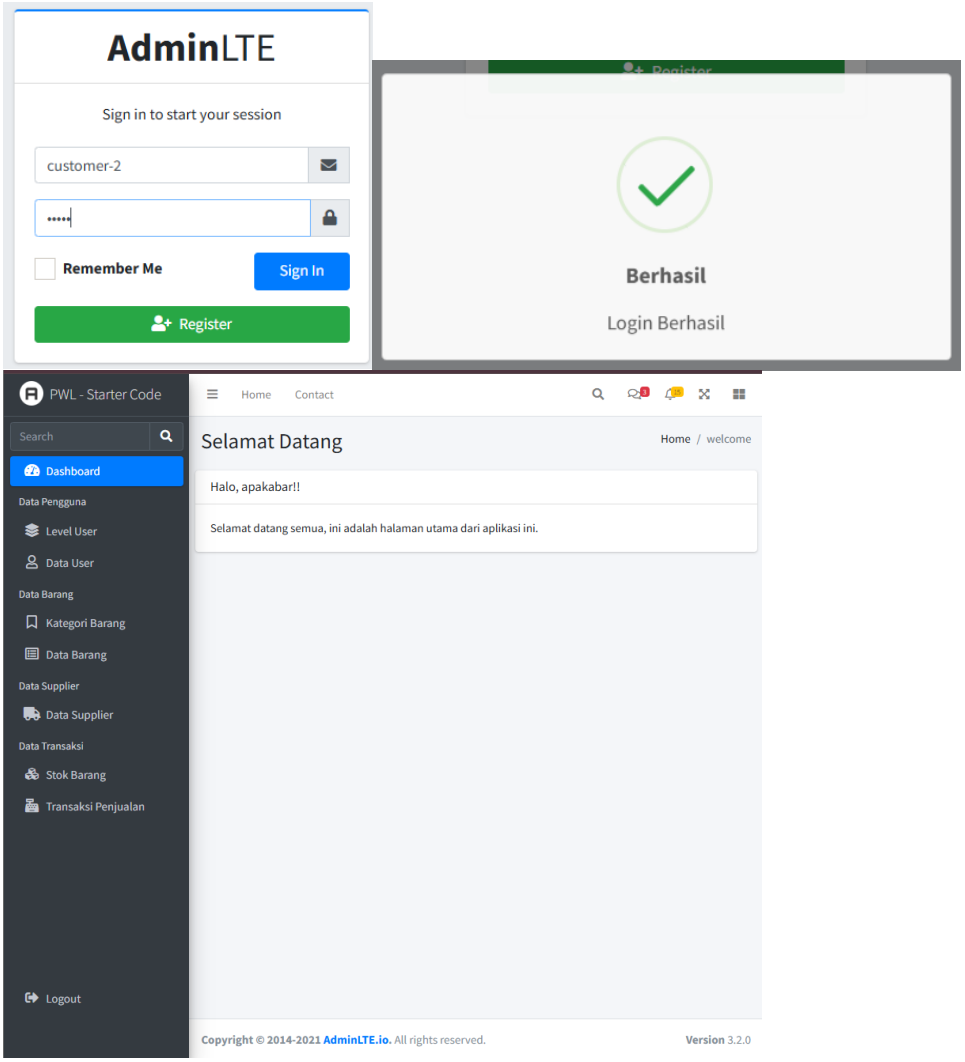
a. Berikut adalah tampilan form register



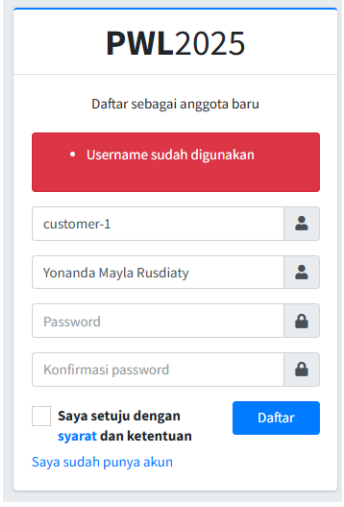
b. Saat klik register maka akan muncul halaman form register

c. Saya coba daftar dengan username customer-2

d. Saya coba login menggunakan username customer-2



e. Berikut adalah tampilan jika pengguna membuat akun dengan username yg sudah ada



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

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