

**LAPORAN PEMOGRAMAN WEB LANJUT**

**JOBSHEET 7**

**AUTHENTICATION dan AUTHORIZATION di LARAVEL**

**Oleh:**

**Reza Angelina Febriyanti**

**NIM 2341760015**



**PROGRAM STUDI D4 SISTEM INFORMASI BISNIS**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2025**

## Praktikum 1 – Implementasi Authentication

Kita buka project laravel **PWL\_POS** kita, dan kita modifikasi konfigurasi aplikasi kita di config/auth.php. Pada bagian ini kita sesuaikan dengan Model untuk tabel m\_user yang sudah kita buat

```
ui.php ^
g > auth.php
return [
    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\Models\UserModel::class,
        ],
    ],
];
```

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

```
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 Illuminate\Foundation\Auth\User as Authenticatable;
9
10 class UserModel extends Model
11 {
12     use HasFactory;
13
14     protected $table = 'm_user'; //mendefinisikan nama tabel yg digunakan oleh model ini
15     protected $primaryKey = 'user_id'; //mendefinisikan pk dari tabel yg digunakan
16     protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'update_at'];
17
18     protected $hidden = ['password'];
19
20     protected $casts = ['password' => 'hashed'];
21
22     public function level(): BelongsTo
23     {
24         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
25     }
26 }
```

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

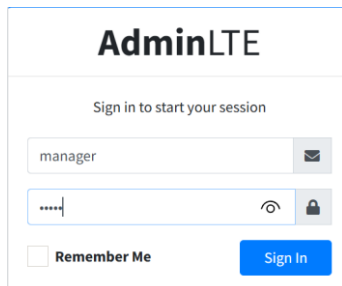
```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7
8 class AuthController extends Controller {
9     public function login() {
10         if(Auth::check()) {
11             return redirect('/');
12         }
13         return view('auth.login');
14     }
15
16     public function postlogin(Request $request) {
17         if($request->ajax() || $request->wantsJson()) {
18             $credentials = $request->only('username', 'password');
19
20             if (Auth::attempt($credentials)) {
21                 return response()->json([
22                     'status' => true,
23                     'message' => 'Login Berhasil',
24                     'redirect' => url('/')
25                 ]);
26
27                 return response()->json([
28                     'status' => false,
29                     'message' => 'Login Gagal'
30                 ]);
31             }
32             return redirect('login');
33         }
34     }
35
36     public function logout(Request $request) {
37         Auth::logout();
38
39         $request->session()->invalidate();
40         $request->session()->regenerateToken();
41         return redirect('login');
42     }
43 }
```

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**)

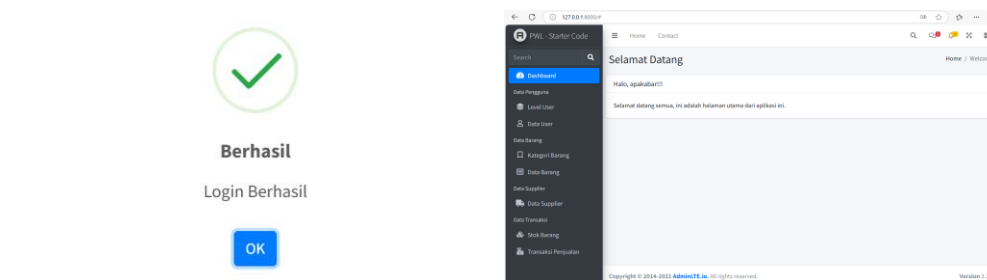
Kemudian kita modifikasi route/web.php agar semua route masuk dalam auth

```
use App\Http\Controllers\AuthController;  
Route::pattern(key: 'id', pattern: '[0-9]+' );  
  
Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login');  
Route::post(uri: 'login', action: [AuthController::class, 'postlogin']);  
Route::post(uri: '/logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth')->name(name: 'logout');  
  
Route::middleware(middleware: ['auth'])->group(callback: function(): void {  
  
});
```

Ketika kita coba mengakses halaman localhost/PWL\_POS/public makan akan tampil halaman awal untuk login ke aplikasi

A screenshot of the AdminLTE login page. It features a white background with a blue header. The main content area has a light gray border. At the top, it says "AdminLTE". Below that, it says "Sign in to start your session". There are two input fields: one for the username (containing "manager") and one for the password (containing "12345"). There are icons for showing/hiding the password and a lock icon. Below the password field, there is a "Remember Me" checkbox and a blue "Sign In" button.

Masukkan username = manager dan password = 12345 untuk melakukan login. Jika berhasil tampilan seperti ini dan diarahkan ke halaman dashboard



## Tugas 1 – Implementasi Authentication

Silahkan implementasikan proses login pada project kalian masing-masing

Silahkan implementasi proses logout pada halaman web yang kalian buat

Tambahkan tombol logout pada file header.blade.php folder view/layouts

```

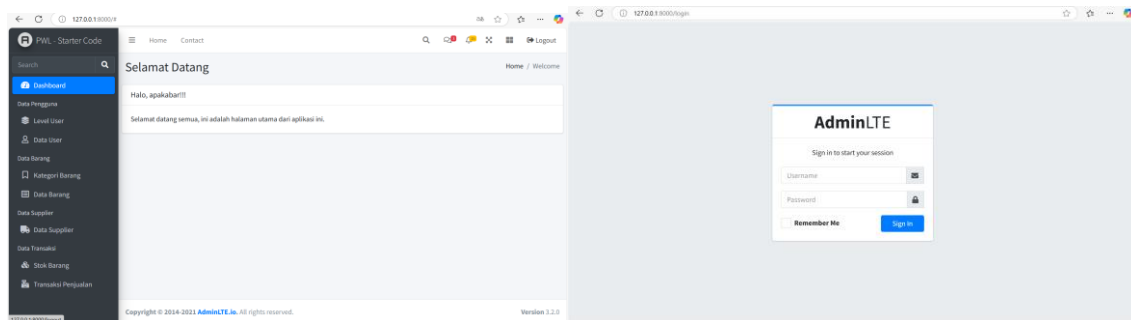
1 <!-- Tombol Logout -->
2 <li class="nav-item">
3   <a class="nav-link" href="{ route('logout') }}" onclick="event.preventDefault(); document.getElementById('logout-form').submit();" role="button">
4     <i class="fas fa-sign-out-alt"></i> Logout
5   </a>
6   <form id="logout-form" action="{ route('logout') }}" method="POST" style="display: none;">
7     @csrf
8   </form>
9 </li>

```

Tambahkan route logout pada web.php

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

Hasilnya muncul tombol logout pada tampilan dashboard



Jika diklik kita diarahkan ke halaman login

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

Route logout untuk membuat jalur logout dari file AuthController.php

Fungsi Logout pada file AuthController.php untuk mengakhiri sesi pengguna dan mengarahkan ke halaman login

Submit kode untuk implementasi Authentication pada repository github kalian.

## Praktikum 2 – Implementasi Authentication di Laravel dengan Middleware

Kita modifikasi UserModel.php dengan menambahkan kode berikut

```
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;
10
11 class UserModel extends Authenticatable
12 {
13     use HasFactory;
14
15     protected $table = 'm_user'; //mendefinisikan nama tabel yg digunakan oleh model ini
16     protected $primaryKey = 'user_id'; //mendefinisikan pk dari tabel yg digunakan
17     protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'update_at'];
18
19     protected $hidden = ['password'];
20
21     protected $casts = ['password' => 'hashed'];
22
23     public function level(): BelongsTo
24     {
25         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
26     }
27
28     public function getRoleName()
29     {
30         return $this->level->level_nama;
31     }
32
33     public function hasRole($role)
34     {
35         return $this->level->level_id == $role;
36     }
37 }
```

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

`php artisan make:middleware AuthorizeUser`

**INFO** Middleware [C:\laragon\www\PWL\_2025\Minggu7\Jobsheet7\app\Http\Middleware\AuthorizeUser.php] created successfully.

Kemudian kita edit *middleware* AuthorizeUser.php untuk bisa mengecek apakah pengguna yang mengakses memiliki Level/Role/Group yang sesuai

```
1 public function handle(Request $request, Closure $next, ...$roles): Response
2 {
3     $user_role = $request->user()->getRole(); // ambil data level_kode dari user yg login
4
5     if (in_array($user_role, $roles)) { // cek apakah level_kode user ada di dalam array roles
6         return $next($request); // jika ada, maka lanjutkan request
7     }
8
9     // jika tidak punya role, maka tampilkan error 403
10    abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
11 }
```

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

```
1 protected $middlewareAliases = [
2     'auth' => \App\Http\Middleware\Authenticate::class,
3     'authorize' => \App\Http\Middleware\AuthorizeUser::class, //mendaftarkan middleware AuthorizeUser
4     'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
5     'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
6     'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
7     'can' => \Illuminate\Auth\Middleware\Authorize::class,
8     'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
9     'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
10    'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
11    'signed' => \App\Http\Middleware\ValidateSignature::class,
12    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
13    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
14 ];
```

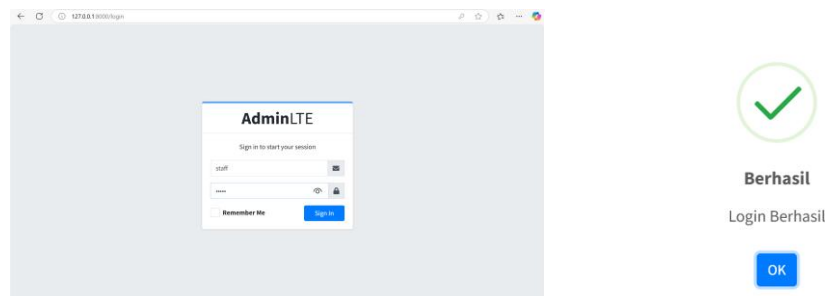
Sekarang kita perhatikan tabel m\_level yang menjadi tabel untuk menyimpan level/group/role dari user ada

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

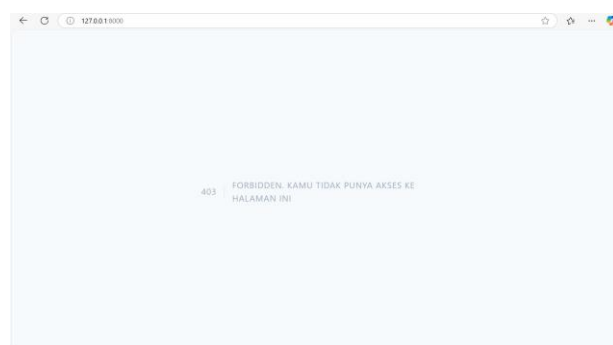
```
1 Route::middleware(['authorize:ADM'])->group(function (){
2     Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal user
3     Route::post('/list', [LevelController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
4     Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah user
5     Route::post('/', [LevelController::class, 'store']); // menyimpan data user baru
6     //Menambah data level
7     Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah level Ajax
8     Route::post('/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data level baru Ajax
9     //Edit data level
10    Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit level Ajax
11    Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan data level Ajax
12
13    Route::get('/{id}', [LevelController::class, 'show']); // menampilkan detail user
14    Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman form edit user
15    Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan data user
16
17    //AJAX
18    Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete level Ajax
19    Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Untuk hapus data level Ajax
20    Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus data user
21 });
```

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.

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



Loginlah menggunakan user selain admin. Saya menggunakan staff dan berhasil namun tidak memiliki akses



## **Tugas 2 – Implementasi Authoriization**

Apa yang kalian pahami pada praktikum 2 ini?

Authorization digunakan untuk mengatur batasan akses user untuk mengakses suatu fitur

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

Authorization digunakan untuk mengatur batasan akses user untuk mengakses suatu fitur

AuthorizeUser untuk otorisasi mengguna berdasarkan role

Modifikasi route level menjadi middleware digunakan untuk mengatur route agar diberi akses bila login menggunakan role sebagai admin

Submit kode untuk impementasi Authorization pada repository github kalian

## Praktikum 3 – Implementasi Multi-Level Authentication di Laravel dengan Middleware

Kita modifikasi UserModel.php untuk mendapatkan level\_kode dari user yang sudah login. Jadi kita buat fungsi dengan nama getRole()

Selanjutnya, Kita modifikasi middleware AuthorizeUser.php dengan kode berikut

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

```
1 Route::middleware(['authorize:ADM,MNG'])->group(function () {
2     Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal user
3     Route::post('/list', [LevelController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
4     Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah user
5     Route::post('/', [LevelController::class, 'store']); // menyimpan data user baru
6     //Menambah data level
7     Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah level Ajax
8     Route::post('/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data level baru Ajax
9     //Edit data level
10    Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit level Ajax
11    Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan data level Ajax
12
13    Route::get('/{id}', [LevelController::class, 'show']); // menampilkan detail user
14    Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman form edit user
15    Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan data user
16
17    //AJAX
18    Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete level Ajax
19    Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Untuk hapus data level Ajax
20    Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus data user
21 });
```

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

Jika login menggunakan role manager dan admin bisa mengakses fitur, jika login menggunakan staff tidak diberi akses

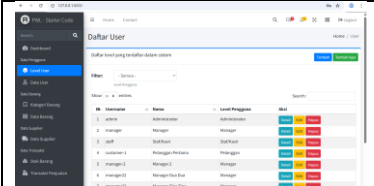
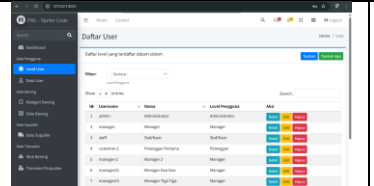
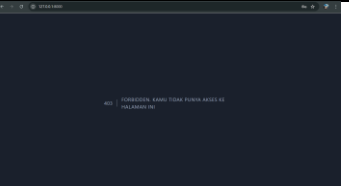
## Tugas 3 – Implementasi Multi-Level Authorization

Silahkan implementasikan multi-level authorization pada project kalian masing-masing

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

Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User

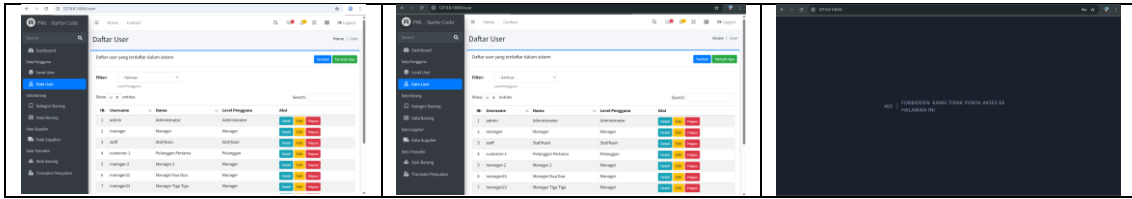
Level user

Admin	Manager	Staff
		

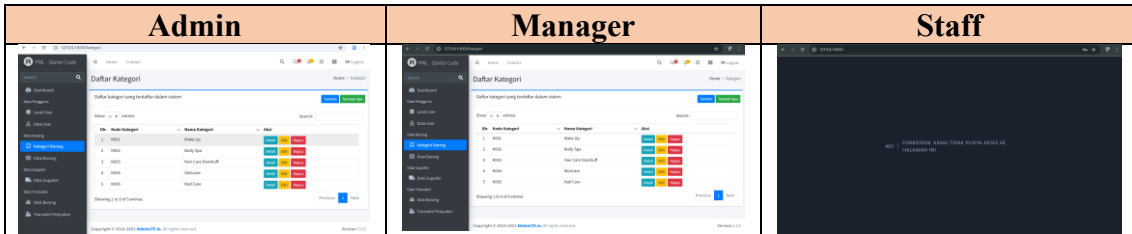
Data user

Admin	Manager	Staff
-------	---------	-------

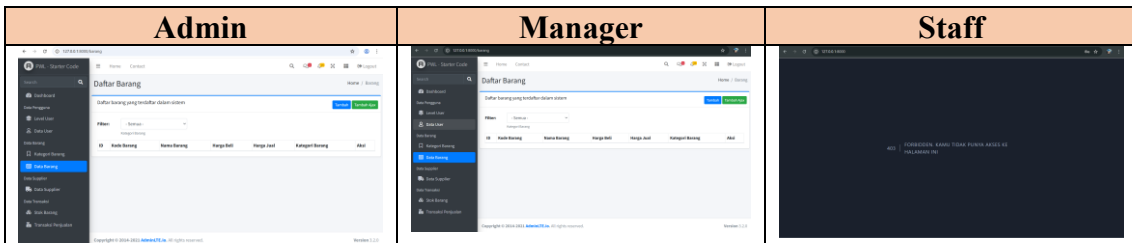




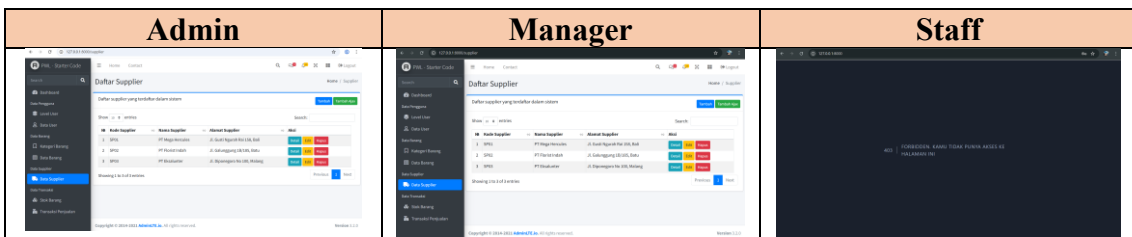
Data kategori



Data barang



Data supplier

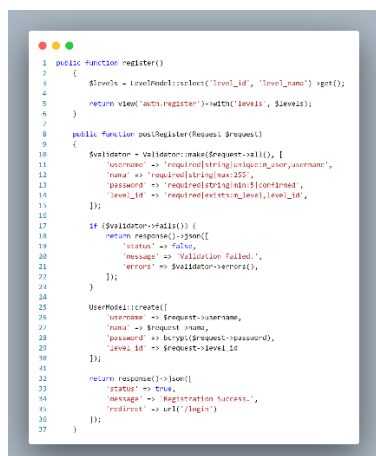


Submit kode untuk impementasi Authorization pada repository github kalian

## Tugas 4 – Implementasi Form Registrasi

Silahkan implementasikan form untuk registrasi user

Tambahkan fungsi register dan postRegister pada file AuthController.php



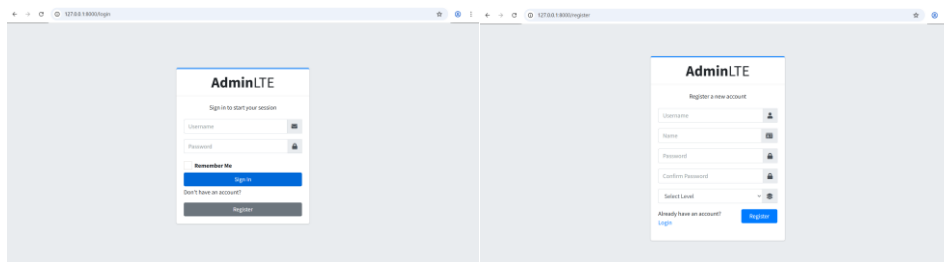
Tambahkan rute register pada file web.php

```
Route::get('/register', [AuthController::class, 'register'])-  
>name('register');  
Route::post('/register', [AuthController::class, 'postRegister']);  
Buat file register.blade.php pada folder views/Auth
```

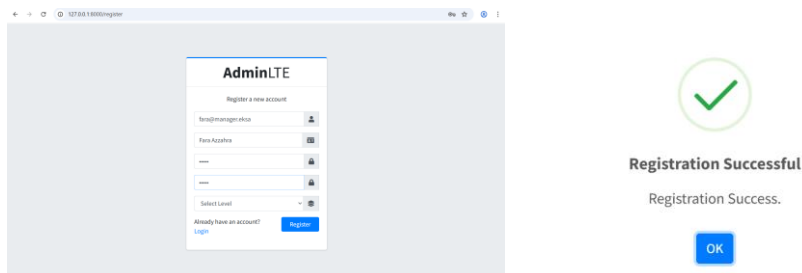
Modifikasi file views/Auth/login.blade.php

```
<!-- /.col -->  
  
        <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('/register') }}"  
class="btn btn-secondary btn-block">Register</a>  
        </div>  
<!-- /.col -->
```

Screenshot hasil yang kalian kerjakan



Saya melakukan register dengan data seperti ini



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