PEMROGRAMAN WEB LANJUT

"Authentication, Middleware"

Kelas : TI-2H

Disusun Oleh :

Fanesabhirawaning Sulistyo

2241720027 (15)

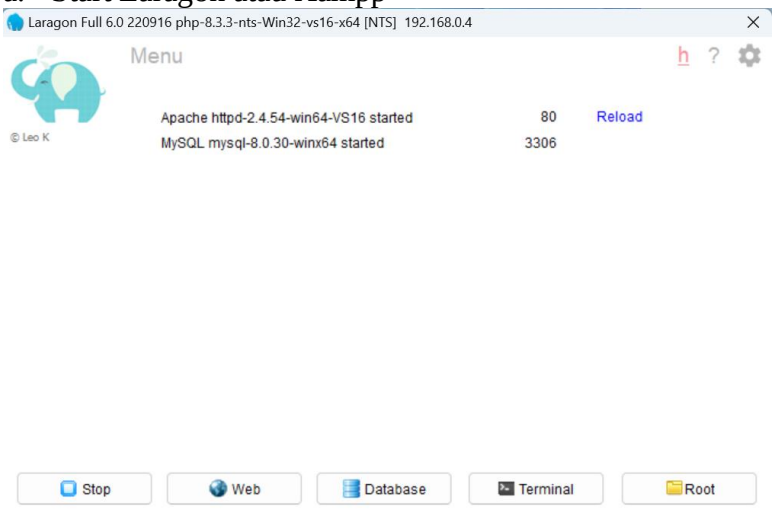
PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141

a. Start Laragon atau Xampp



b. Jalankan "php artisan serve"

```
PS C:\laragon\www\PWL_POS> php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server
```

c. Jalankan "npm run dev"

```
PS C:\laragon\www\PWL_POS> npm run dev

> dev
> vite


  VITE v5.1.6  ready in 1032 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help

  LARAVEL v10.47.0  plugin v1.0.2

  ➜  APP_URL: http://localhost
```

Jika 3 tahap diatas sudah dilakukan kemudian lakukan langkah-langkah praktikum di bawah ini

1. Sebelum melangkah lebih lanjut setting file pada **config/auth.php** ubah pada bagian model menjadi model yang telah kita buat sebelumnya untuk menyimpan username dan password

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

Penjelasan:

konfigurasi tersebut memberitahukan Laravel untuk menggunakan model **UserModel** sebagai basis untuk mengelola autentikasi pengguna, dan menggunakan driver eloquent untuk berinteraksi dengan basis data.

2. Membuat middleware dengan perintah pada cmd atau terminal

```
PS C:\laragon\www\PWL_POS> php artisan make:middleware Cek_login

  INFO  Middleware [C:\laragon\www\PWL_POS\app\Http\Middleware\Cek_login.php] created successfully.
```

Setelah itu cek didalam folder app/Http/Middleware apakah sudah ada file bernama **Cek_login.php**

Penjelasan:

Perintah tersebut akan membuat file Middleware cek_login.php

3. Kemudian tambahkan kode dibawah ini, letakkan di function handle()

app > Http > Middleware > Cek_login.php > ...

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Symfony\Component\HttpFoundation\Response;

class Cek_login
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next, $roles): Response
    {
        // cek sudah login atau belum, jika belum kembali ke halaman login
        if (!Auth::check()) {
            return redirect('login');
        }

        // simpan data user pada variabel user
        $user = Auth::user();

        // jika user memiliki level sesuai pada kolom pada lanjutkan request
        if ($user->level_id == $roles) {
            return $next($request);
        }

        // jika tidak memiliki akses maka kembalikan ke halaman login
        return redirect('login')->with('error', 'Maaf anda tidak memiliki akses');

        // return $next($request);
    }
}
```

Penjelasan:

middleware ini bertugas untuk memeriksa apakah pengguna sudah login, serta memverifikasi apakah level pengguna sesuai dengan level yang diizinkan untuk mengakses suatu halaman atau fungsi dalam aplikasi. Jika tidak sesuai, pengguna akan diarahkan kembali ke halaman login dengan pesan kesalahan.

4. Ketika sudah membuat middleware, langkah berikutnya harus registrasikan pada **kernel.php**. Agar middleware dapat dibaca oleh sistem. Buka folder **App/Http/Kernel.php** dan tambahkan code pada variabel $middlewareAliases seperti dibawah

```
55      protected $middlewareAliases = [
56          'auth' => \App\Http\Middleware\Authenticate::class,
57          'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
58          'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
59          'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
60          'can' => \Illuminate\Auth\Middleware\Authorize::class,
61          'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
62          'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
63          'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
64          'signed' => \App\Http\Middleware\ValidateSignature::class,
65          'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
66          'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
67          // tambahkan middleware alias dengan nama cek_login
68          'cek_login' => \App\Http\Middleware\Cek_login::class,
```

Penjelasan:

Dengan menambahkan kode **cek_login** seperti di atas, Kita telah mendaftarkan middleware **Cek_login** sehingga dapat digunakan dalam definisi rute atau grup rute pada aplikasi Laravel Kita. Setelah mendaftarkan middleware, Kita dapat menggunakannya dengan cara yang diinginkan, misalnya dengan menambahkannya pada grup rute di file **routes/web.php**

5. Membuat Controller Auth. Pada langkah ini kita akan membuat controller auth, yang dimana di dalamnya terdapat proses authentikasi dari request login. Jalankan perintah make:controller

```
PS C:\laragon\www\PWL_POS> php artisan make:controller AuthController

  INFO  Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\AuthController.php] created successfully.
```

Buka file **App/Http/Controllers/AuthController.php** yang sudah kita buat. Lalu kita akan isi perintah code untuk membuat proses authentikasi , verifikasi, dan logout

Penjelasan:

Perintah tersebut akan membuat file AuthController

6. Kemudian buat kode AuthController.php seperti dibawah ini

app > Http > Controllers > AuthController.php > AuthController

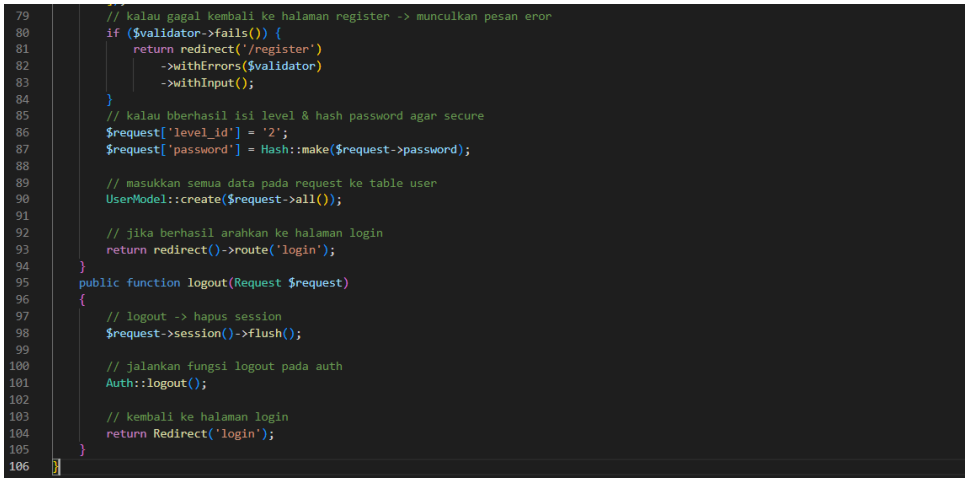
```php
<?php

namespace App\Http\Controllers;

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

class AuthController extends Controller
{
    public function index()
    {
        // kita ambil data user lalu simpan pada variable $user
        $user = Auth::user();

        // kondisi jika usernya ada
        if ($user) {
            // jika level admin
            if ($user->level_id == '1') {
                return redirect()->intended('admin');
            }
            //jika level manager
            else if ($user->level_id == '2') {
                return redirect()->intended('manager');
            }
        }
        return view('login');
    }
    public function proses_login(Request $request)
    {
        // kita buat validasi pd saat tombol login diklik
        // validasi username & password wajib diisi
        $request->validate([
            'username' => 'required',
            'password' => 'required'
        ]);
        // ambil data request username & password saja
        $credential = $request->only('username', 'password');
        // cek jika username & password valid (sesuai) dengan data
        if (Auth::attempt($credential)) {
            // jika berhasil, simpan data user di variabel $user
            $user = Auth::user();
            // cek lg jika level user = admin maka arahkan ke halaman admin
            if ($user->level_id == '1') {
                // dd($user->level_id);
                return redirect()->intended('admin');
            }
            // jika level user biasa maka arahkan ke halaman user (manager)
            else if ($user->level_id == '2') {
                // dd($user->level_id);
                return redirect()->intended('manager');
            }
            // jika belum ada role maka ke halaman /
            return redirect()->intended('/');
        }
        // jika tidak ada user yang valid maka kembalikan ke halaman login
        // kirim pesan error juga kalau login gagal
        return redirect('login')->withInput()
            ->withErrors(['login_gagal' => 'Pastikan kembali username dan password yang dimasukkan sudah benar']);
    }
    public function register()
    {
        return view('register');
    }
    // aksi form register
    public function proses_register(Request $request)
    {
        // buat validasi buat program register
        // validasinya -> semua field wajib diisi
        // validasi username unique atau tidak boleh duplicate usernamenya
        $validator = Validator::make($request->all(), [
            'nama' => 'required',
            'username' => 'required|unique:m_user',
            'password' => 'required'
        ]);
```



```
        // kalau gagal kembali ke halaman register -> munculkan pesan eror
        if ($validator->fails()) {
            return redirect('/register')
                ->withErrors($validator)
                ->withInput();
        }
        // kalau bberhasil isi level & hash password agar secure
        $request['level_id'] = '2';
        $request['password'] = Hash::make($request->password);

        // masukkan semua data pada request ke table user
        UserModel::create($request->all());

        // jika berhasil arahkan ke halaman login
        return redirect()->route('login');
    }
    public function logout(Request $request)
    {
        // logout -> hapus session
        $request->session()->flush();

        // jalankan fungsi logout pada auth
        Auth::logout();

        // kembali ke halaman login
        return Redirect('login');
    }
}
```

Penjelasan:

1. index()
    - Metode ini mengembalikan tampilan halaman login jika pengguna belum login.
    - Jika pengguna sudah login, metode ini memeriksa level pengguna dan mengarahkannya ke halaman admin atau halaman manager sesuai dengan levelnya.
2. proses_login(Request $request)
    - Metode ini menangani proses autentikasi pengguna saat tombol login ditekan.
    - Melakukan validasi bahwa username dan password telah diisi.
    - Menggunakan Auth::attempt($credential) untuk memeriksa apakah kredensial pengguna valid.
    - Jika autentikasi berhasil, mengarahkan pengguna ke halaman admin atau manager sesuai dengan levelnya.
    - Jika autentikasi gagal, mengembalikan pengguna ke halaman login dengan pesan kesalahan.
3. register()
    - Metode ini mengembalikan tampilan halaman registrasi pengguna.
4. proses_register(Request $request)
    - Metode ini menangani proses registrasi pengguna.
    - Melakukan validasi bahwa semua field telah diisi dan username harus unik.
    - Jika validasi gagal, mengembalikan pengguna ke halaman registrasi dengan pesan kesalahan.
    - Jika berhasil, menambahkan level_id (diasumsikan level user) dan melakukan hashing password sebelum menyimpan pengguna ke dalam database.
    - Mengarahkan pengguna ke halaman login setelah registrasi selesai.
5. logout(Request $request)
    - Metode ini menangani proses logout pengguna.
    - Menghapus session pengguna, menjalankan fungsi logout dari Auth, dan mengarahkan pengguna kembali ke halaman login.

7. Kemudian tambahkan kode pada route pada file **routes/web.php** seperti kode program dibawah

```php
Route::get('login', [AuthController::class, 'index'])->name('login');
Route::get('register', [AuthController::class, 'register'])->name('register');
Route::post('proses_login', [AuthController::class, 'proses_login'])->name('proses_login');
Route::get('logout', [AuthController::class, 'logout'])->name('logout');
Route::post('proses_register', [AuthController::class, 'proses_register'])->name('proses_register');

//kita atur juga untuk middleware menggunakan group pada routing
//didalamnya terdapat group untuk mengecek kondisi login
//jika user yang login merupakan admin maka akan diarahkan ke AdminController
//jika user yang login merupakan manager maka akan diarahkan ke ManagerController

Route::group(['middleware' => 'auth'], function () {

    Route::group(['middleware' => ['cek_login:1']], function () {
        Route::resource('admin', AdminController::class);
    });
    Route::group(['middleware' => ['cek_login:2']], function () {
        Route::resource('manager', ManagerController::class);
    });
});
```

Penjelasan:

1. Rute Login, Registrasi, dan Proses Login:
    - login mengarahkan ke halaman login.
    - register mengarahkan ke halaman registrasi.
    - proses_login menangani proses login pengguna.
    - proses_register menangani proses registrasi pengguna.
2. Rute Logout:
    - logout menangani proses logout pengguna.
3. Grup Middleware auth:
    - Memastikan pengguna harus login sebelum mengakses rute-rute di dalamnya.
4. Grup Middleware cek_login:1 dan cek_login:2:
    - cek_login:1 untuk admin, cek_login:2 untuk manager.
    - Memastikan pengguna yang login sesuai dengan level yang diizinkan.
    - Mengarahkan admin ke AdminController.
    - Mengarahkan manager ke ManagerController.

8. Kemudian buat file controller dengan nama AdminController sebagai halaman yang boleh diakses oleh admin saja. Jalankan perintah dibawah

```
PS C:\laragon\www\PWL_POS> php artisan make:controller AdminController

  INFO  Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\AdminController.php] created successfully.
```

Penjelasan:

Perintah tersebut akan membuat file controller AdminController

9. Buka folder **App/Http/Controllers** dan isi file AdminController seperti dibawah

app > Http > Controllers > AdminController.php > AdminController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AdminController extends Controller
{
    public function index()
    {
        return view('admin');
    }
}
```

Penjelasan:

AdminController adalah controller yang mengelola halaman admin. Metode index() digunakan untuk menampilkan tampilan admin.

10. Kemudian buat file controller dengan nama ManagerController sebagai halaman yang boleh diakses oleh admin saja. Jalankan perintah dibawah

```
PS C:\laragon\www\PWL_POS> php artisan make:controller ManagerController

  INFO  Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\ManagerController.php] created successfully.
```

Penjelasan:

Perintah tersebut akan membuat file controller ManagerController

11. Buka folder **App/Http/Controllers** dan isi file **ManagerController** seperti dibawah

app > Http > Controllers > ManagerController.php > ManagerController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ManagerController extends Controller
{
    public function index()
    {
        return view('manager');
    }
}
```

Penjelasan:

ManagerController adalah controller yang mengelola halaman Manager. Metode index() digunakan untuk menampilkan tampilan Manager.

12. Pada langkah terakhir yaitu membuat layout sederhana

13. Buatlah halaman login dengan membuka folder resources/views dengan nama file l**ogin.blade.php.** Lalu kita isi filenya seperti contoh dibawah

resources > views > login.blade.php > ...

```
@extends('adminlte::auth.auth-page', ['auth_type' => 'login'])

@section('adminlte_css_pre')
    {{-- <link rel="stylesheet" href="{{asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css')}}"> --}}
    <link rel="stylesheet" href="{{ asset('vendor/icheck-bootstrap/icheck-bootstrap.min.css') }}">
@stop

@php($login_url = View::getSection('login_url') ?? config('adminlte.login_url', 'login'))
@php($register_url = View::getSection('register_url') ?? config('adminlte.register_url', 'register'))
@php($password_reset_url = View::getSection('password_reset_url') ?? config('adminlte.password_reset_url', 'password/reset'))

@if (config('adminlte.use_route_url', false))
    @php($login_url = $login_url ? route($login_url) : '')
    @php($register_url = $register_url ? route($register_url) : '')
    @php($password_reset_url = $password_reset_url ? route($password_reset_url) : '')
@else
    @php($login_url = $login_url ? url($login_url) : '')
    @php($register_url = $register_url ? url($register_url) : '')
    @php($password_reset_url = $password_reset_url ? url($password_reset_url) : '')
@endif

@section('auth_header', __('adminlte::adminlte.login_message'))
@section('auth_body')
    @error('login_gagal')
        <div class="alert alert-warning alert-dismissible fade show" role="alert">
            <span class="alert-inner--text">
                <strong>Warning!</strong>
                {{ $message }}
            </span>
            <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
        </div>
    @enderror
    <form action="{{ url('proses_login') }}" method="POST">
        @csrf
        {{-- Email field --}}
        <div class="input-group mb-3">
            <input type="text" name="username" class="form-control" @error('username') is-invalid @enderror"
                value="{{ old('username') }}" placeholder="Username" autofocus>
            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-envelope {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>
            @error('username')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>
        {{-- Password field --}}
        <div class="input-group mb-3">
            <input type="password" name="password" class="form-control" @error('password') is-invalid @enderror"
                placeholder="{{ __('adminlte::adminlte.password') }}">
            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-lock {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>
            @error('password')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>
        {{-- Login field --}}
        <div class="row">
            <div class="col-7">
                <div class="icheck-primary" title="{{ __('adminlte::adminlte.remember_me_hint') }}">
                    <input type="checkbox" name="remember" id="remember" {{ old('remember') ? 'checked' : '' }}>
                    <label for="remember">
                        {{ __('adminlte::adminlte.remember_me') }}
                    </label>
                </div>
            </div>

            <div class="col-5">
                <button type="submit"
                    class="btn btn-block{{ config('adminlte.classes_auth_btn', 'btn-flat btn-primary') }}">
                    <span class="fas fa-sign-in-alt"></span>
                    {{ __('adminlte::adminlte.sign_in') }}
                </button>
            </div>
        </div>
    </form>
@stop
@section('auth_footer')
    @if ($register_url)
        <p class="my-0">
            <a href="{{ route('register') }}">
                {{ __('adminlte::adminlte.register_a_new_membership') }}
            </a>
        </p>
    @endif
@stop
```

Penjelasan:

1. Ekstensi dan Section:
    - Meng-extend layout dari AdminLTE untuk halaman login (@extends(adminlte::auth.auth-page, [auth_type => login])).
    - Mengisi konten spesifik untuk halaman login menggunakan @section(auth_body) dan @section(auth_footer).
2. CSS dan URL:
    - Menambahkan CSS untuk tampilan halaman menggunakan @section(adminlte_css_pre).
    - Mengatur URL untuk login, registrasi, dan reset password berdasarkan konfigurasi AdminLTE atau route yang ditentukan.
3. Form Login:
    - Membuat form login dengan input untuk username, password, dan tombol login.
    - Menampilkan kembali input sebelumnya jika terjadi error validasi.
4. Pesan Kesalahan:
    - Menampilkan pesan error jika terjadi kesalahan validasi pada input username dan password.
5. Remember Me:
    - Menampilkan opsi "Remember Me" dengan checkbox

14. Buatlah halaman register dengan membuka folder resources/views dengan nama file **register.blade.php.** Lalu kita isi filenya seperti contoh dibawah

```blade
        {{-- Username field --}}
        <div class="input-group mb-3">
            <input type="text" name="username" class="form-control" @error('username') is-invalid @enderror"
                value="{{ old('username') }}" placeholder="Username" autofocus>
            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-user {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>
            @error('username')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>
        {{-- Password field --}}
        <div class="input-group mb-3">
            <input type="password" name="password" class="form-control" @error('password') is-invalid @enderror"
                placeholder="{{ __('adminlte::adminlte.password') }}">
            <div class="input-group-append">
                <div class="input-group-text">
                    <span class="fas fa-lock {{ config('adminlte.classes_auth_icon', '') }}"></span>
                </div>
            </div>
            @error('password')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>
        {{-- Register button --}}
        <button type="submit" class="btn btn-block{{ config('adminlte.classes_auth_btn', 'btn-flat btn-primary') }}">
            <span class="fas fa-user-plus"></span>
            {{ __('adminlte::adminlte.register') }}
        </button>
    </form>
@stop
@section('auth_footer')
    <p class="my-0">
        <a href="{{ route('login') }}">
            {{ __('adminlte::adminlte.i_already_have_a_membership') }}
        </a>
    </p>
@stop
```

Penjelasan:

1. Ekstensi dan Section:
    - Meng-extend layout dari AdminLTE untuk halaman registrasi.
    - Mengisi konten spesifik untuk halaman registrasi.
2. Pengaturan URL:
    - Mengambil URL login dan registrasi dari konfigurasi AdminLTE atau route yang ditentukan.
    - Menyesuaikan URL berdasarkan konfigurasi.
3. Form Registrasi:
    - Membuat form registrasi dengan input untuk nama, username, password, dan tombol registrasi.
    - Menampilkan kembali input sebelumnya jika terjadi error validasi.
4. Pesan Kesalahan:
    - Menampilkan pesan error jika terjadi kesalahan validasi pada input nama, username, dan password.

15. Buatlah halaman register dengan membuka folder resources/views dengan nama file **admin.blade.php.** Lalu kita isi filenya seperti contoh dibawah

resources > views > admin.blade.php > ...

```blade
@extends('layout.app')

{{-- Customize layout section --}}

@section('subtitle', 'Admin')
@section('content_header_title', 'Home')
@section('content_header_subtitle', 'Admin')

@section('content')
    <div class="container">
        <div class="card">
            <div class="card-header">
                Tampilan <?php echo Auth::user()->level_id == 1 ? 'Admin' : 'Manager'; ?>
            </div>
            <div class="card-body">
                <h1>
                    Login sebagai : <?php echo Auth::user()->level_id == 1 ? 'Admin' : 'Manager'; ?>
                </h1>
                <a href="{{ route('logout') }}">Logout</a>
            </div>
        </div>
    </div>
@endsection
@push('js')
@endpush
```

Penjelasan:

1. Ekstensi Layout:
    - Meng-extend layout app yang digunakan sebagai template halaman.
2. Customize Layout Section:
    - Mengisi bagian-bagian spesifik dari layout seperti judul, subjudul, dan konten header.
3. Konten Halaman:
    - Menampilkan konten halaman yang berisi informasi tentang jenis pengguna yang sedang login (admin atau manager) berdasarkan level_id dari user yang sedang login.
    - Menampilkan tombol logout yang mengarah ke route logout.

16. Buatlah halaman register dengan membuka folder resources/views dengan nama file **manager.blade.php.** Lalu kita isi filenya seperti contoh dibawah

resources > views > manager.blade.php

```blade
@extends('layout.app')

{{-- Customize layout section --}}

@section('subtitle', 'Manager')
@section('content_header_title', 'Home')
@section('content_header_subtitle', 'Manager')

@section('content')
    <div class="container">
        <div class="card">
            <div class="card-header">
                Tampilan <?php echo Auth::user()->level_id == 1 ? 'Admin' : 'Manager'; ?>
            </div>
            <div class="card-body">
                <h1>
                    Login sebagai : <?php echo Auth::user()->level_id == 1 ? 'Admin' : 'Manager'; ?>
                </h1>
                <a href="{{ route('logout') }}">Logout</a>
            </div>
        </div>
    </div>
@endsection
@push('js')
@endpush
```

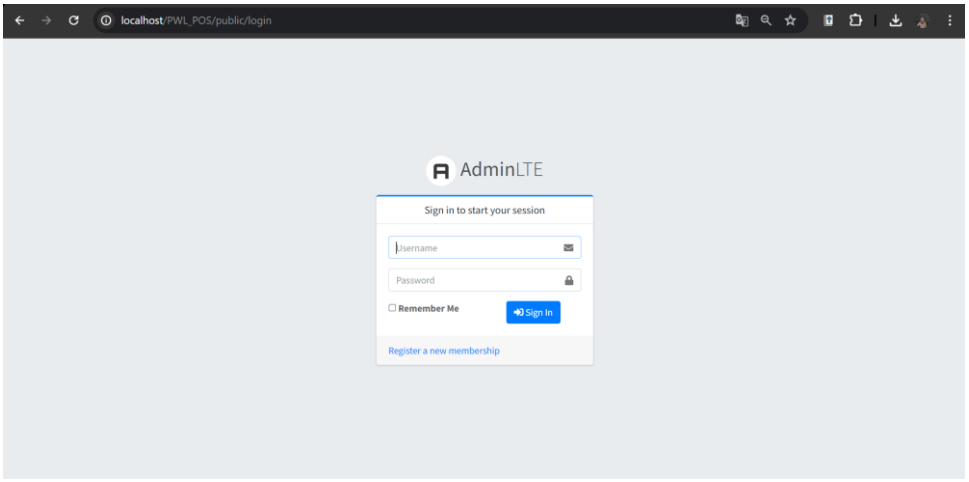
Penjelasan:

1. Ekstensi Layout:
    - Meng-extend layout app yang digunakan sebagai template halaman.
2. Customize Layout Section:
    - Mengisi bagian-bagian spesifik dari layout seperti judul, subjudul, dan konten header.
3. Konten Halaman:
    - Menampilkan konten halaman yang berisi informasi tentang jenis pengguna yang sedang login (admin atau manager) berdasarkan level_id dari user yang sedang login.
    - Menampilkan tombol logout yang mengarah ke route logout.

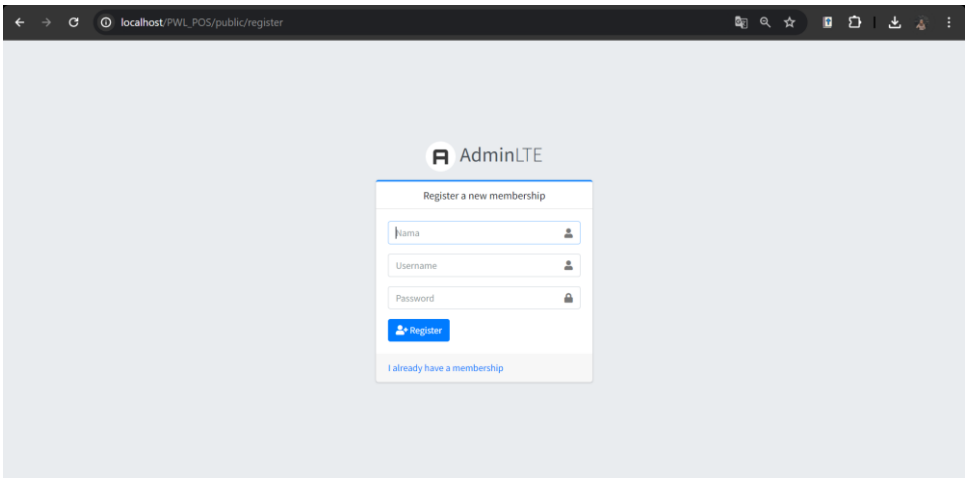
17. Jalankan kode diatas dan buat laporan dengan isi screenshoot hasil dan beri penjelasan pada setiap screenshoot hasil dan *commit* perubahan pada *git*.

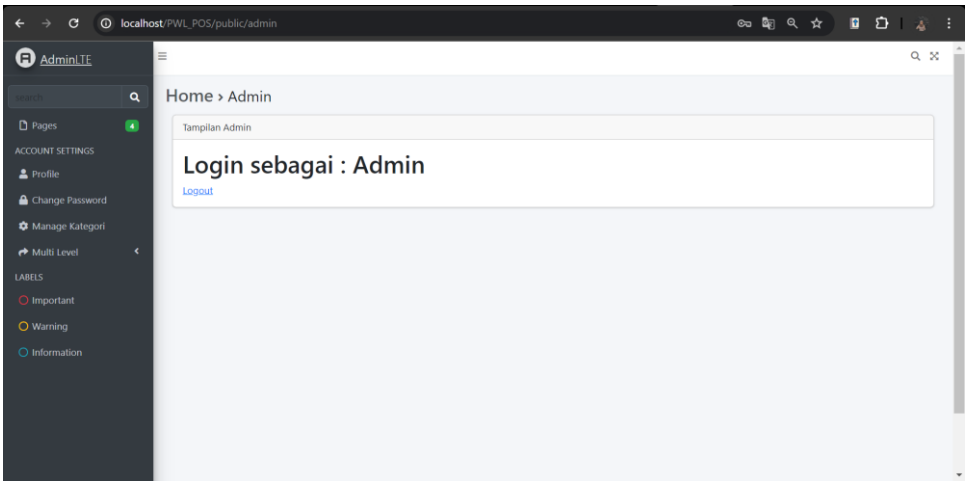
Jawab:

- Halaman Login



- Halaman Register



- 
localhost/PWL_POS/public/admin
AdminLTE
Pages
ACCOUNT SETTINGS
Profile
Change Password
Manage Kategori
Multi Level
LABELS
Important
Warning
Information
Home › Admin
Tampilan Admin
Login sebagai : Admin
Logout