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

Courses	:	Advanced Web Programming (PWL)
Program Studi	:	D4 – Informatics Engineering / D4 – Business Information Systems
Semester	:	4 (four) / 6 (six)
Meeting to-	:	9 (nine)

JOBSHEET 09
AUTHENTICATION, MIDDLEWARE

Previously we discussed *about Migration, Seeder*, *DB Façade, Query Builder,* and a little about *Eloquent ORM* in Laravel. Before we get into making website-based applications, it would be nice for us to prepare a database as a place to store data in our application later. In addition, generally we need to prepare also the initial data that we use before creating an application, such as administrator user data, system settings data, etc.

In this meeting, we will understand how to display data, change data, and delete data using the Eloquent technique.

In accordance with `Case Study PWL.pdf`. So our Laravel 10 project is still the same as using **the PWL_POS repository.**

We will use PWL_POS project until the 12th meeting later, as a project that we will study

Laravel Authentication

Laravel Authentication is used to protect pages or features of the web that are only accessed by certain people who are granted rights. Features like this are usually found on systems that have administrator features or systems that have users who can add data.

Laravel makes implementing authentication very simple and has provided various features that can be utilized without the need to install specific modules. The authentication configuration file is located in config/auth.php, which contains several well-documented options for changing the configuration of the authentication service.

At its core, Laravel's authentication facility consists of "guards" and "providers". Guards define how users are authenticated for each request. For example, Laravel sends with guards to sessions by using session storage and cookies.

Middleware

Middleware is the intermediate layer between incoming ***HTTP route*** requests and controller ***actions*** . **Middleware** allows us to perform various tasks either before or after the action is performed. We can also use *CLI* **tools** to create a **Middleware** in **Laravel**. Some examples of **using middleware** include authentication, validation, request manipulation, and more. Below are the benefits of **Middleware:**

- **Security** : in **Middleware** allows us to verify if the user is authenticated before accessing a particular page. Thus, we can protect sensitive data and control user access rights.
- **Data Filtering**: **Middleware** can be used to manipulate request data before an ***action*** in the *controller* is performed. For example, we can first check the data sent by the user before the data is further processed or we want to modify the data to be sent and then we can double-check the data to be sent by the user before the data is processed.
- **Logging and** Auditing: **Middleware** can also be used to log user activity or audit incoming requests. It can help in monitoring and analysis of applications.

Practicum – Auth and Middleware

This practicum is a continuation of the 4th meeting. The table used for this lab uses m_user. Before doing practicum, run the command below:

a. Start Laragon or Xampp
b. Run "php artisan serve"
c. Jalankan “npm run dev”

If the 3 stages above have been done then do the practicum steps below

1. Before going any further, setting the file in **config/auth.php** change the model section to the model that we created earlier to store usernames and passwords

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

2. Create middleware with commands on cmd or terminal

```
php artisan make:middleware Cek_login
```

After that, check in the app/Http/Middleware folder whether there is already a file named **Cek_login.php**

3. Then add the code below, place it in the handle() function

```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');
    }
}
```

4. When you have created middleware, the next step must be registered on kernel.php. So that the middleware can be read by the system. Open the **App/Http/Kernel.php** folder and add code to the $middlewareAliases variable as below

```php
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
    'signed' => \App\Http\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
    // tambahkan middleware alias dengan nama cek_login
    'cek_login' => \App\Http\Middleware\Cek_login::class,
];
```

5. Create an auth controller. In this step we will create an auth controller, in which there is the authentication process of the login request. Run the make:controller command

```
php artisan make:controller AuthController
```

Open the **App/Http/Controllers/AuthController.php file** that we created. Then we will fill in the code command to create the authentication, verification, and logout processes

6. Then create the AuthController.php code as below

```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 user nya ada
        if ($user) {
            // jika user nya memiliki level admin
            if ($user->level_id == '1') {
                return redirect()->intended('admin');
            }
            // jika user nya memiliki level manager
            else if ($user->level == '2') {
                return redirect()->intended('manager');
            }
        }
        return view('login');
    }
    //
    public function proses_login(Request $request)
    {

        // kita buat validasi pada saat tombol login di klik
        // validas nya username & password wajib di isi
        $request->validate([
            'username' => 'required',
            'password' => 'required'
        ]);


        // ambil data request username & password saja
        $credential = $request->only('username', 'password');
        // cek jika data username dan password valid (sesuai) dengan data
        if (Auth::attempt($credential)) {

            // kalau berhasil simpan data user ya di variabel $user
            $user =  Auth::user();

            // cek lagi jika level user admin maka arahkan ke halaman admin
            if ($user->level_id == '1') {
                //dd($user->level_id);
                return redirect()->intended('admin');
            }
            // tapi jika level user nya user biasa maka arahkan ke halaman user
            else if ($user->level_id == '2') {
                return redirect()->intended('manager');
            }
            // jika belum ada role maka ke halaman /
            return redirect()->intended('/');
        }
        // jika ga ada data user yang valid maka kembalikan lagi ke halaman login
        // pastikan kirim pesan error juga kalau login gagal ya
        return redirect('login')
            ->withInput()
            ->withErrors(['login_gagal' => 'Pastikan kembali username dan password yang dimasukan sudah benar']);
    }

    public function register()
    {
        // tampilkan view register
        return view('register');
    }

    // aksi form register
    public function proses_register(Request $request)
    {
        //. kita buat validasi nih buat proses register
        // validasinya yaitu semua field wajib di isi
        // validasi username itu harus unique atau tidak boleh duplicate username ya
        $validator =  Validator::make($request->all(), [
            'nama' => 'required',
            'username' => 'required|unique:m_user',
            'password' => 'required'
        ]);

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

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

        // kalo berhasil arahkan ke halaman login
        return redirect()->route('login');
    }

    public function logout(Request $request)
    {
        // logout itu harus menghapus session nya
        $request->session()->flush();

        // jalan kan juga fungsi logout pada auth
        Auth::logout();
        // kembali kan ke halaman login
        return Redirect('login');
    }
}
```

7. Then add code to the route in the **routes / web.php** file like the program code below

```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 UserController
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);
    });
});
```

8. Then create a controller file named AdminController as a page that can be accessed by admins only. Run the command below

```
php artisan make:controller AdminController
```

9. Open the **App/Http/Controllers** folder and fill the AdminController file as below

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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

10. Then create a controller file named ManagerController as a page that can be accessed by admins only. Run the command below

```
php artisan make:controller ManagerController
```

11. Open the **App/Http/Controllers** folder and fill the **ManagerController** file as below

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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

12. The last step is to create a simple layout
13. Create a login page by opening the resources/views folder with the file name login.blade.php. Then we fill in the file as shown below

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

@section('adminlte_css_pre')
    <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
```

14. Create a register page by opening the resources/views folder with the file name **register.blade.php.** Then we fill in the file as shown below

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

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

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

@section('auth_header', __('adminlte::adminlte.register_message'))

@section('auth_body')
    <form action="{{route('proses_register')}}" method="post">
        @csrf

        {{-- Nama field --}}
        <div class="input-group mb-3">
            <input type="text" name="nama" class="form-control @error('nama') is-invalid @enderror"
                   value="{{ old('name') }}" placeholder="Nama" 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('nama')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>

        {{-- Username field --}}
        <div class="input-group mb-3">
            <input type="text" name="username" class="form-control @error('username') is-invalid @enderror"
                   value="{{ old('name') }}" 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
```

15. Create a register page by opening the resources/views folder with the file name **admin.blade.php.** Then we fill in the file as shown below

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

{{-- Customize layout sections --}}

@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 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
```

16. Create a register page by opening the resources/views folder with the name file **manager.blade.php.** Then we fill in the file as shown below

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

{{-- Customize layout sections --}}

@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 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
```

17. Run the code above and create a report with the contents of the result screenshot and give an explanation to each result screenshot and *commit* changes to *git*.

That's it, and happy learning ***