



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 5 (lima)
Pertemuan ke- : 7 (tujuh)

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 *guards* 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 melakukan berbagai tugas baik itu sebelum ataupun sesudah tindakan dilakukan. Kita juga dapat menggunakan **tool CLI** untuk membuat sebuah **Middleware** dalam **Laravel**. Beberapa contoh penggunaan **Middleware** meliputi autentikasi, validasi, manipulasi permintaan, dan lainnya. Berikut di bawah ini adalah manfaat dari **Middleware** :

- **Keamanan** : dalam **Middleware** memungkinkan kita untuk memverifikasi apakah pengguna sudah diautentikasi sebelum mengakses halaman tertentu. Dengan demikian, kita dapat melindungi data sensitif dan mengontrol hak akses pengguna.
- **Pemfilteran Data** : **Middleware** dapat digunakan untuk memanipulasi data permintaan sebelum sebuah **action** dalam **controller** dilakukan. Misalnya, kita dapat memeriksa terlebih dahulu data yang dikirim oleh pengguna sebelum data tersebut diproses lebih



lanjut atau kita ingin memodifikasi data yang akan dikirim lalu kita dapat memeriksa ulang data yang akan dikirim oleh pengguna sebelum data tersebut diproses.

- **Logging dan Audit : Middleware** juga dapat digunakan untuk mencatat aktivitas pengguna atau melakukan audit terhadap permintaan yang masuk. Ini dapat membantu dalam pemantauan dan analisis aplikasi.

INFO

Kita akan menggunakan Laravel Auth secara manual seperti
<https://laravel.com/docs/10.x/authentication#authenticating-users>

Sesuai dengan **Studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. Implementasi Manual Authentication di Laravel

Autentikasi adalah proses untuk memverifikasi identitas pengguna yang mencoba mengakses sistem. Dalam konteks aplikasi web, autentikasi memastikan bahwa pengguna yang mencoba login memiliki hak akses yang sesuai berdasarkan kredensial seperti email dan password. Proses autentikasi berbeda dengan **otorisasi**, yang merupakan langkah lanjutan untuk menentukan hak akses apa yang dimiliki pengguna setelah mereka berhasil diautentikasi.

Konsep Autentikasi di Laravel

Laravel menawarkan sistem autentikasi yang sangat fleksibel. Laravel menyediakan mekanisme autentikasi bawaan melalui layanan authentication scaffolding seperti Laravel *Jetstream* dan *Breeze*, yang dapat secara otomatis menghasilkan halaman dan logika autentikasi. Namun, terkadang pengembang memerlukan implementasi autentikasi yang lebih manual untuk memberikan kontrol penuh terhadap setiap aspek dari proses tersebut.

Beberapa komponen penting dalam sistem autentikasi Laravel meliputi:

- *Guard*: Komponen yang mengatur bagaimana pengguna diautentikasi untuk setiap permintaan. *Guard* default menggunakan sesi dan cookie.



- *Provider*: Mengatur bagaimana pengguna diambil dari database atau sumber data lainnya. *Provider* default mengambil data pengguna dari database dengan menggunakan Eloquent ORM.
- *Session*: Laravel menggunakan sesi untuk menyimpan status autentikasi pengguna. Sesi memungkinkan sistem untuk mengingat pengguna yang sudah login di antara permintaan HTTP yang berbeda.

Alur umum dari autentikasi meliputi:

1. *Login*: Pengguna mengirimkan kredensial (biasanya berupa email dan password).
2. *Verifikasi Kredensial*: Sistem memeriksa apakah kredensial yang diberikan sesuai dengan data di database.
3. *Pembuatan Sesi*: Jika kredensial benar, sistem akan membuat sesi untuk pengguna yang akan disimpan di server.
4. *Akses ke Halaman yang Dilindungi*: Pengguna yang terautentikasi dapat mengakses halaman-halaman yang dilindungi oleh *middleware* auth.
5. *Logout*: Pengguna bisa keluar dari sistem dan sesi mereka akan dihapus.

Middleware Autentikasi

Middleware auth di Laravel digunakan untuk melindungi rute atau halaman agar hanya dapat diakses oleh pengguna yang sudah terautentikasi. Jika pengguna mencoba mengakses rute yang memerlukan autentikasi tanpa login, mereka akan diarahkan ke halaman login.

- **Guard** bertanggung jawab untuk menangani proses autentikasi pengguna. Laravel secara default menggunakan *guard* berbasis sesi untuk autentikasi web, namun juga mendukung *guard* berbasis token (seperti API).
- **Provider** bertugas untuk mengambil pengguna dari database. Laravel menyediakan *provider* default yang menggunakan Eloquent, namun juga mendukung *provider* lain seperti Query Builder.

Implementasi di Laravel 10

Kita akan menerapkan penggunaan authentication di Laravel. Dalam penerapan ini, kita akan mencoba membuat otentikasi secara di Laravel, agar kita paham langkah-langkah dalam membuat Authentication



Praktikum 1 – Implementasi Authentication :

1. Kita buka project laravel **PWL_POS** kita, dan kita modifikasi konfigurasi aplikasi kita di **config/auth.php**

```
62     'providers' => [  
63         'users' => [  
64             'driver' => 'eloquent',  
65             'model' => App\Models\User::class,  
66         ],
```

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

```
62     'providers' => [  
63         'users' => [  
64             'driver' => 'eloquent',  
65             'model' => App\Models\UserModel::class,  
66         ],  
67     ],
```

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

```
<?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; // implementasi class Authenticatable  
  
class UserModel extends Authenticatable  
{  
    use HasFactory;  
  
    protected $table = 'm_user';  
    protected $primaryKey = 'user_id';  
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at'];  
  
    protected $hidden = ['password']; // jangan di tampilkan saat select  
  
    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash  
  
    /**  
     * Relasi ke tabel level  
     */  
    public function level(): BelongsTo  
    {  
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
    }  
}
```



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

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    public function login()
    {
        if(Auth::check()){ // jika sudah login, maka redirect ke halaman home
            return redirect('/');
        }
        return view('auth.login');
    }

    public function postlogin(Request $request)
    {
        if($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)
        {
            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**)

```
<!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=fallb
ack">
<!-- 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></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="checkbox-primary">
<input type="checkbox" id="remember"><label for="remember">Remember Me</label>
</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
            });
          }
        }
      });
    },
    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>
```

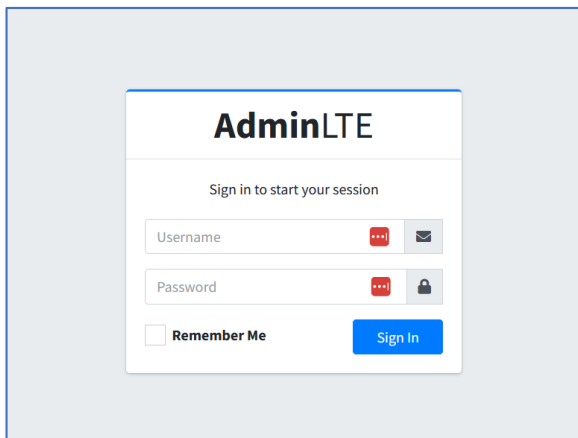



```
</body>  
</html>
```

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

```
<?php  
  
use App\Http\Controllers\UserController;  
use App\Http\Controllers\SupplierController;  
use App\Http\Controllers\BarangController;  
use App\Http\Controllers\AuthController;  
use App\Http\Controllers\KategoriController;  
use App\Http\Controllers\LevelController;  
use App\Http\Controllers>WelcomeController;  
use Illuminate\Support\Facades\Route;  
  
Route::pattern('id', '[0-9]+'); // artinya ketika ada parameter {id}, maka harus berupa angka  
  
Route::get('login', [AuthController::class, 'login'])->name('login');  
Route::post('login', [AuthController::class, 'postlogin']);  
Route::get('logout', [AuthController::class, 'logout'])->middleware('auth');  
  
Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu  
    // masukkan semua route yang perlu autentikasi di sini  
});
```

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



Tugas 1 – Implementasi Authentication :

1. Silahkan implementasikan proses login pada project kalian masing-masing



The image shows a login form for AdminLTE. It has a title 'AdminLTE' and a subtitle 'Sign in to start your session'. There are two input fields: 'Username' and 'Password'. Below the password field is a 'Remember Me' checkbox and a 'Sign In' button.

2. Silahkan implementasi proses logout pada halaman web yang kalian buat

The screenshot shows a web application interface. At the top, there is a navigation bar with several icons. A 'Logout' button is highlighted with a red box. Below the navigation bar, there is a 'HomeWelcome' section. Below the screenshot, the HTML code for the Logout button is shown, with line numbers 130 to 145.

```
130 <a class="nav-link" data-widget="control-sidebar" data-slide="true" href="#" role="button">
131 | <i class="fas fa-th-large"></i>
132 </a>
133 </li>
134 <li class="nav-item">
135 | <a href="{{ url('/logout') }}" class="nav-link"
136 | | onclick="event.preventDefault(); document.getElementById('logout-form').submit();"
137 | | <i class="fas fa-sign-out-alt"></i> Logout
138 | </a>
139 | <form id="logout-form" action="{{ url('/logout') }}" method="GET" style="display: none;"
140 | | @csrf
141 | </form>
142 | </li>
143 </ul>
144 </nav>
```

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

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

    // 'users' => [
    //     'driver' => 'database',
    //     'table' => 'users',
    // ],
],
```

Tahap pertama proses login yaitu mengganti menjadi
App\Models\UserModel::class



Minggu-7 > PWL_POS > app > Models > UserModel.php > UserModel

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Illuminate\Foundation\Auth\User as Authenticatable;
9
10 class UserModel extends Authenticatable
11 {
12     use HasFactory;
13
14     protected $table = 'm_user';
15     protected $primaryKey = 'user_id';
16
17     protected $fillable = [
18         'username', 'password', 'nama', 'level_id', 'created_at', 'updated_at',
19     ];
20
21     // Agar kolom password tidak ditampilkan saat select
22     protected $hidden = ['password'];
23
24     // Otomatis hash password saat diset
25     protected $casts = [
26         'password' => 'hashed',
27     ];
28
29     public function level(): BelongsTo
30     {
31         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
32     }
33 }
```

Memodifikasi UserModel

Minggu-7 > PWL_POS > resources > views > auth > login.blade.php > html > body,hold-transition.login-page > div.login-box > div.card.card-outline.card-primary > div.card-header.text-center

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <title>Login Pengguna</title>
8
9      <meta name="csrf-token" content="{{ csrf_token() }}">
10
11      <!-- Google Font -->
12      <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
13
14      <!-- Font Awesome -->
15      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@5.15.4/css/all.min.css">
16
17      <!-- icheck bootstrap -->
18      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/icheck-bootstrap@3.0.1/icheck-bootstrap.min.css">
19
20      <!-- SweetAlert2 -->
21      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/sweetalert2@10.15.4/dist/sweetalert2.min.css">
22
23      <!-- AdminLTE -->
24      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/admin-lte@3.2/dist/css/adminlte.min.css">
25
26  </head>
27
```



```
Minggu-7 > PWL_POS > app > Http > Controllers > AuthController.php > AuthController > postlogin
1  <?php
2  namespace App\Http\Controllers;
3
4  use Illuminate\Http\Request;
5  use Illuminate\Support\Facades\Auth;
6
7  class AuthController extends Controller
8  {
9      public function login()
10     {
11         if (Auth::check()) { // jika sudah login, maka redirect ke halaman home return redirect('/');
12         }
13         return view('auth.login');
14     }
15
16     public function postlogin(Request $request)
17     {
18         if ($request->ajax() || $request->wantsJson()) {
19             $credentials = $request->only('username', 'password');
20
21             if (Auth::attempt($credentials)) {
22                 return response()->json([
23                     'status' => true,
24                     'message' => 'Login Berhasil',
25                     'redirect' => url('/')
26                 ]);
27             }
28             return response()->json([
29                 'status' => false,
30                 'message' => 'Login Gagal'
31             ]);
32         }
33
34         return redirect('login');
35     }
36
37     public function logout(Request $request)
38     {
39         Auth::logout();
40
41         $request->session()->invalidate();
42         $request->session()->regenerateToken();
43         return redirect('login');
44     }
45 }
```

Kemudian pada gambar diatas membuat AuthController

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Login Pengguna</title>

    <meta name="csrf-token" content="{{ csrf_token() }}">

    <!-- Google Font -->
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallbac
k">

    <!-- Font Awesome -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-
free@5.15.4/css/all.min.css">

    <!-- icheck bootstrap -->
```



```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/icheck-bootstrap@3.0.1/icheck-bootstrap.min.css">

<!-- SweetAlert2 -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@sweetalert2/theme-bootstrap-4/bootstrap-4.min.css">

<!-- AdminLTE -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/adminlte@3.2/dist/css/adminlte.min.css">

</head>

<body class="hold-transition login-page">

  <div class="login-box">
    <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>
            <div class="col-4">
              <button type="submit" class="btn btn-primary btn-block">Sign
In</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>

  <!-- jQuery -->
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.4/dist/jquery.min.js"></script>
  <!-- Bootstrap -->
```



```
<script>
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>
<!-- jQuery Validation -->
<script src="https://cdn.jsdelivr.net/npm/jquery-
validation@1.19.5/dist/jquery.validate.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/jquery-validation@1.19.5/dist/additional-
methods.min.js"></script>
<!-- SweetAlert2 -->
<script
src="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.5/dist/sweetalert2.min.js"></script>
<!-- AdminLTE -->
<script src="https://cdn.jsdelivr.net/npm/admin-lte@3.2/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: 5,
        maxlength: 20
      }
    },
    submitHandler: function(form) {
      $.ajax({
        url: form.action,
        type: form.method,
        data: $(form).serialize(),
        success: function(response) {
          if (response.status) {
            Swal.fire({
              icon: 'success',
              title: 'Berhasil',
              text: response.message,
            }).then(function() {
              window.location = response.redirect;
            });
          } else {
            $('.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>
```

Kode diatas adalah login.blade

```
Minggu-7 > PWL_POS > routes > web.php > Closure > Closure  
1  <?php  
2  
3  use App\Http\Controllers\KategoriController;  
4  use App\Http\Controllers\LevelController;  
5  use Illuminate\Support\Facades\Route;  
6  use App\Http\Controllers\UserController;  
7  use App\Http\Controllers\WelcomeController;  
8  use App\Http\Controllers\BarangController;  
9  use App\Http\Controllers\SupplierController;  
10 use App\Http\Controllers\AuthController;  
11  
12 //Route Auth  
13 Route::pattern('id','[0-9]+'; // artinya ketika ada parameter {id}, maka harus berupa angka  
14  
15 Route::get('login', [AuthController::class,'login'])->name('login');  
16 Route::post('login', [AuthController::class,'postlogin']);  
17 Route::get('logout', [AuthController::class,'logout'])->middleware('auth');  
18  
19 Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu  
20     //masukkan semua route yang perlu autentikasi di sini  
21     Route::get('/level', [LevelController::class, 'index']);  
22     Route::get('/kategori', [KategoriController::class, 'index']);  
23     Route::get('/user', [UserController::class, 'index']);  
24  
25     // Route::get('/user/tambah', [UserController::class, 'tambah']);  
26     // Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);  
27  
28     // Route::get('/user/ubah/{id}', [UserController::class, 'ubah']);  
29     // Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);  
30  
31     // Route::get('/user/hapus/{id}', [UserController::class, 'hapus']);  
32  
33     Route::get('/', [WelcomeController::class, 'index']);  
34  
35     Route::group(['prefix' => 'user'], function () {  
36         Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user  
37         Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables  
38         Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user  
39         Route::post('/', [UserController::class, 'store']); // menyimpan data user baru  
40         Route::get('/create_ajax', [UserController::class, 'create_ajax']);  
41         Route::post('/ajax', [UserController::class, 'store_ajax']);  
42         Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user  
43         Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user  
44         Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // menyimpan perubahan data user  
45         Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Untuk tampilan form confirm delete user Aj
```

Kemudian membuat route untuk login dan logout



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

Kemudian menambahkan kode logout pada template.blade

Sehingga tampilan login dan logout seperti pada gambar di atas

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 Authorization 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

```
27
28
29      /**
30       * Relasi ke tabel level
31       */
32      public function level(): BelongsTo
33      {
34          return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
35      }
36
37      /**
38       * Mendapatkan nama role dari relasi level
39       */
40      public function getRoleName(): string
41      {
42          return $this->level ? $this->level->level_nama : '-';
43      }
44
45      /**
46       * Cek apakah user memiliki role tertentu
47       */
48      public function hasRole($role): bool
49      {
50          return $this->level && $this->level->level_kode == $role;
51      }
```



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`

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

```
Minggu-7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthorizeUser
10
11     /**
12      * Handle an incoming request.
13      *
14      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
15      */
16     public function handle(Request $request, Closure $next, $role = ''): Response
17     {
18         $user = $request->user(); // Ambil user yang sedang login
19
20         if ($user && $user->hasRole($role)) {
21             return $next($request); // Jika punya role, lanjutkan request
22         }
23
24         // Jika tidak, tampilkan error 403
25         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
26     }
27
28
```

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

```
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,
```

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

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



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

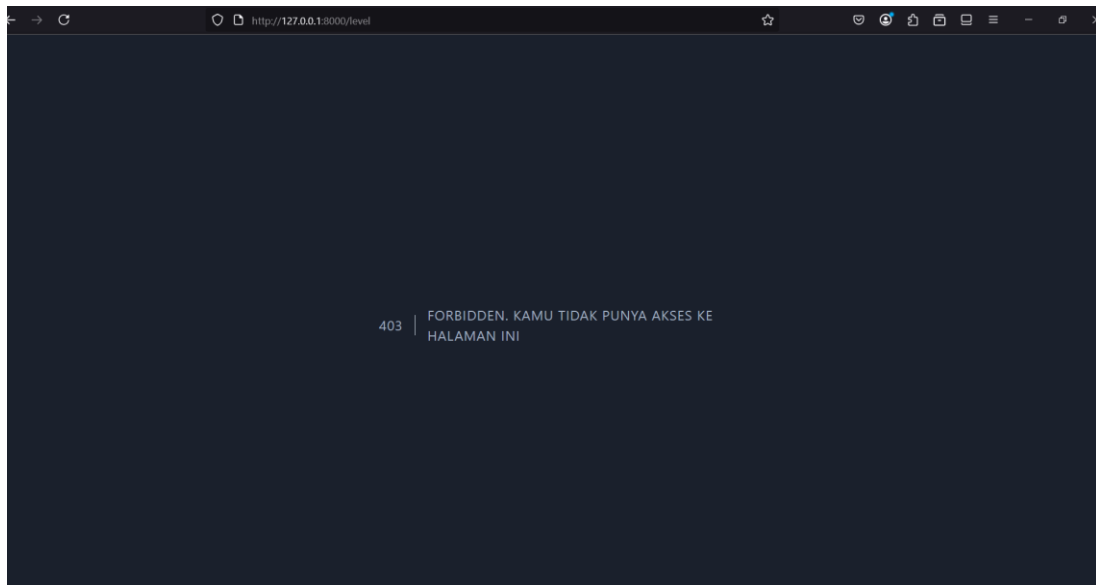
```
Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu
    Route::get('/', [WelcomeController::class, 'index']);
    // route Level

    // artinya semua route di dalam group ini harus punya role ADM (Administrator)
    Route::middleware(['authorize:ADM'])->group(function(){
        Route::get('/level', [LevelController::class, 'index']);
        Route::post('/level/list', [LevelController::class, 'list']); // untuk list json datatables
        Route::get('/level/create', [LevelController::class, 'create']);
        Route::post('/level', [LevelController::class, 'store']);
        Route::get('/level/{id}/edit', [LevelController::class, 'edit']); // untuk tampilkan form edit
        Route::put('/level/{id}', [LevelController::class, 'update']); // untuk proses update data
        Route::delete('/level/{id}', [LevelController::class, 'destroy']); // untuk proses hapus data
    });

    // route Kategori
```

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

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



Tugas 2 – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?
 - Praktikum ini menerapkan system authorization dengan menggunakan middleware untuk mengecek level akses user ke halaman tertentu berdasarkan role nya. Dan middleware akan memfilter berdasarkan role untuk akses route tertentu.



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

- Modifikasi UserModel.php

```
27
28
29      /**
30       * Relasi ke tabel level
31       */
32      public function level(): BelongsTo
33      {
34          return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
35      }
36
37      /**
38       * Mendapatkan nama role dari relasi level
39       */
40      public function getRoleName(): string
41      {
42          return $this->level ? $this->level->level_nama : '-';
43      }
44
45      /**
46       * Cek apakah user memiliki role tertentu
47       */
48      public function hasRole($role): bool
49      {
50          return $this->level && $this->level->level_kode == $role;
51      }
```

Pada gambar di atas menunjukkan bahwa telah menambahkan method `getRoleName()` untuk mendapatkan nama role dan menambahkan method `hasRole($role)` untuk mengecek apakah user memiliki role tertentu

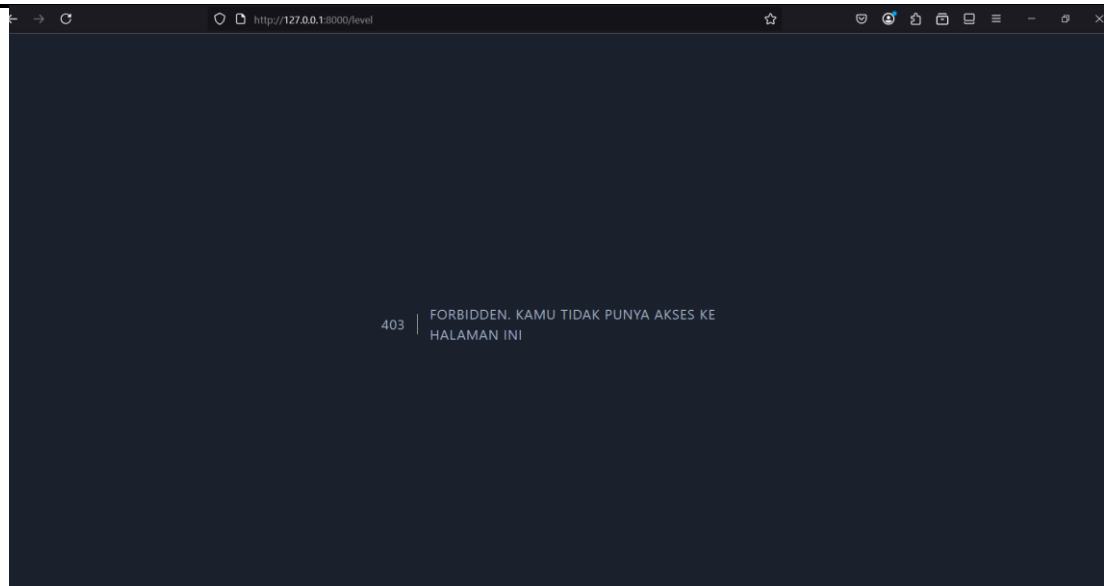
- Membuat middleware `AuthorizeUser.php`

```
Minggu-7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthorizeUser
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
15      */
16     public function handle(Request $request, Closure $next, $role = ''): Response
17     {
18         $user = $request->user(); // Ambil user yang sedang login
19
20         if ($user && $user->hasRole($role)) {
21             return $next($request); // Jika punya role, lanjutkan request
22         }
23
24         // Jika tidak, tampilkan error 403
25         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
26     }
27 }
28
```

Middleware ini mengecek apakah user yang sedang login memiliki role yang sesuai. Jika iya, dapat mengakses, jika tidak maka akan menampilkan seperti gambar di bawah ini



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>



3. Submit kode untuk implementasi 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 Authorization 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()`



```
/**
 * Mendapatkan kode role
 */
public function getRole()
{
    return $this->level->level_kode;
}
```

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

```
Minggu-7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser > handle > $roles
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthorizeUser
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
15      */
16     public function handle(Request $request, Closure $next, ... $roles): Response
17     {
18         $user_role = $request->user()->getRole(); // Ambil user yang sedang login
19         if (in_array($user_role, $roles)) {
20             return $next($request); // Jika punya role, lanjutkan request
21         }
22
23         // Jika tidak, tampilkan error 403
24         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
25     }
26 }
```

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



```
// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG (Manager)
Route::middleware(['authorize:ADM,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
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

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

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
 - Admin mengakses data barang

The screenshot shows the AdminLTE login interface on the left, where the username 'Admin' is entered. On the right, a sample data table titled 'Daftar Barang' is displayed, showing a list of items with their details.

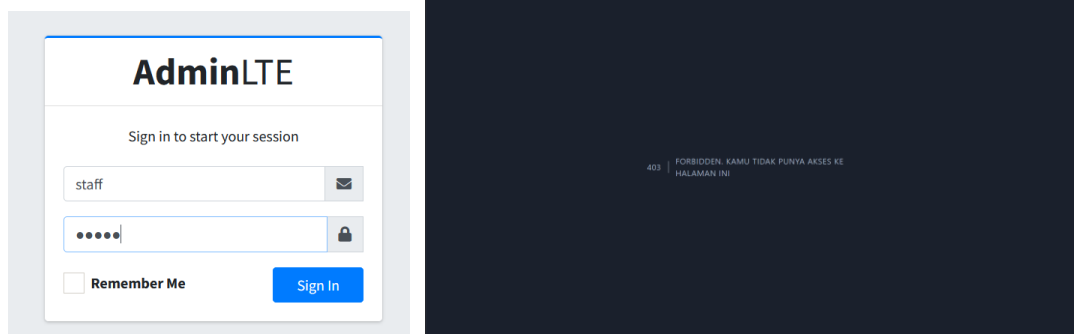
ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
1	B001	Laptop	500000	600000	Makanan	Detail Edit Hapus
2	B002	Handphone	300000	350000	Makanan	Detail Edit Hapus
3	B003	Polo	50000	75000	Elektronik	Detail Edit Hapus
4	B004	Pants	150000	200000	Elektronik	Detail Edit Hapus
5	B005	Nasi Bungkus	25000	35000	Meubel	Detail Edit Hapus
6	B006	Air AQUA	5000	10000	Minuman	Detail Edit Hapus
7	B007	Kipas Angin	200000	300000	Peralatan Perahu Belah	Detail Edit Hapus
8	B008	Rice Cooker	400000	500000	Peralatan Perahu Belah	Detail Edit Hapus
9	B009	Speaker Bluetooth	100000	150000	Makanan	Detail Edit Hapus

- Manager mengakses data barang

The screenshot shows the AdminLTE login interface on the left, where the username 'manager' is entered. On the right, a sample data table titled 'Daftar Barang' is displayed, showing a list of items with their details.

ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
1	B001	Laptop	500000	600000	Makanan	Detail Edit Hapus
2	B002	Handphone	300000	350000	Makanan	Detail Edit Hapus
3	B003	Polo	50000	75000	Elektronik	Detail Edit Hapus
4	B004	Pants	150000	200000	Elektronik	Detail Edit Hapus
5	B005	Nasi Bungkus	25000	35000	Meubel	Detail Edit Hapus
6	B006	Air AQUA	5000	10000	Minuman	Detail Edit Hapus
7	B007	Kipas Angin	200000	300000	Peralatan Perahu Belah	Detail Edit Hapus
8	B008	Rice Cooker	400000	500000	Peralatan Perahu Belah	Detail Edit Hapus
9	B009	Speaker Bluetooth	100000	150000	Makanan	Detail Edit Hapus
10	B010	Tek. Barok	7000	13000	Minuman	Detail Edit Hapus

- Staff mengakses data barang



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

<?php

```
use App\Http\Controllers\UserController;
use App\Http\Controllers\SupplierController;
use App\Http\Controllers\BarangController;
use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers>WelcomeController;
use App\Http\Controllers\AuthController;
use Illuminate\Support\Facades\Route;

Route::pattern('id', '[0-9]+'); // Pastikan parameter {id} hanya berupa angka

// Rute otentikasi
Route::get('login', [AuthController::class, 'login'])->name('login');
Route::post('login', [AuthController::class, 'postlogin']);
Route::get('/register', [AuthController::class, 'register']);
Route::post('/register', [AuthController::class, 'postRegister']);
Route::post('/logout', [AuthController::class, 'logout'])->middleware('auth')->name('logout');

// Semua rute di bawah ini hanya bisa diakses jika sudah login
Route::middleware(['auth'])->group(function () {
    Route::get('/', [WelcomeController::class, 'index']);

    Route::middleware(['authorize:ADM'])->group(function () {
        Route::group(['prefix' => 'user'], function () {
            Route::get('/', [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::post('/', [UserController::class, 'store']); // menyimpan data user baru //Create Menggunakan AJAX
            Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
            Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax
            Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user
            Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user
            Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
            //Edit Menggunakan AJAX
            Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax
        });
    });
});
```



```
Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); //
Menyimpan perubahan data user Ajax
//Delete Menggunakan AJAX
Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Untuk
tampilkan form confirm delete user Ajax
Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); //
Untuk hapus data user Ajax
Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
});
});

// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function(){
    Route::group(['prefix' => 'level'], function () {
        Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal
level
        Route::post("/list", [LevelController::class, 'list']); // menampilkan data level
dalam bentuk json untuk datatables
        Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman
form tambah level
        Route::post('/', [LevelController::class, 'store']); // menyimpan data level baru
//Create Menggunakan AJAX
        Route::get('/create_ajax', [LevelController::class, 'create_ajax']); //
Menampilkan halaman form tambah level Ajax
        Route::post('/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data
level baru Ajax
        Route::get('/{id}', [LevelController::class, 'show']); // menampilkan detail level
        Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman
form edit level
        Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan
data level
//Edit Menggunakan AJAX
        Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); //
Menampilkan halaman form edit level Ajax
        Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); //
Menyimpan perubahan data level Ajax
//Delete Menggunakan AJAX
        Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); //
Untuk tampilkan form confirm delete level Ajax
        Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); //
Untuk hapus data level Ajax
        Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus data
level
    });
});

// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG
(Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::group(['prefix' => 'kategori'], function () {
        Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal
kategori
        Route::post("/list", [KategoriController::class, 'list']); // menampilkan data
kategori dalam bentuk json untuk datatables
        Route::get('/create', [KategoriController::class, 'create']); // menampilkan
halaman form tambah kategori
        Route::post('/', [KategoriController::class, 'store']); // menyimpan data kategori
baru
// Create menggunakan AJAX
        Route::get('/create_ajax', [KategoriController::class, 'create_ajax']); //
menampilkan halaman form tambah kategori ajax
        Route::post('/ajax', [KategoriController::class, 'store_ajax']); // menyimpan data
kategori baru ajax
```



```
Route::get('/{id}', [KategoriController::class, 'show']); // menampilkan detail
kategori
Route::get('/{id}/edit', [KategoriController::class, 'edit']); // menampilkan
halaman form edit kategori
Route::put('/{id}', [KategoriController::class, 'update']); // menyimpan perubahan
data kategori
// Edit menggunakan AJAX
Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']); //
menampilkan halaman form edit kategori ajax
Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']); //
menyimpan perubahan data kategori ajax
// Delete menggunakan AJAX
Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']);
//menampilkan form confirm delete kategori ajax
Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']); //
menghapus data kategori ajax
Route::delete('/{id}', [KategoriController::class, 'destroy']); // menghapus data
kategori
});
});

// Staff hanya bisa melihat data barang
Route::middleware(['authorize:ADM,MNG,STF'])->group(function(){
    Route::group(['prefix' => 'barang'], function () {
        Route::get('/', [BarangController::class, 'index']); // Menampilkan daftar barang
        Route::post('/list', [BarangController::class, 'list']); // Menampilkan data
barang dalam bentuk JSON untuk datatables
        Route::get('/{id}', [BarangController::class, 'show']); // Menampilkan detail
barang
    });
});

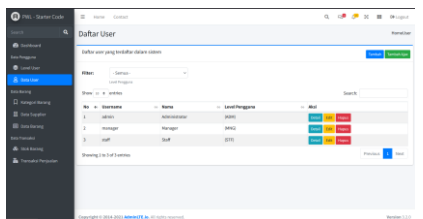
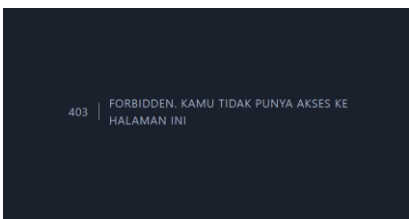
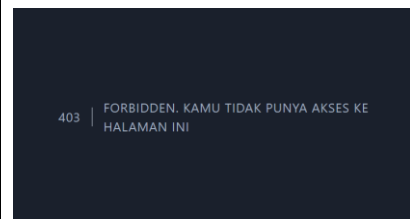

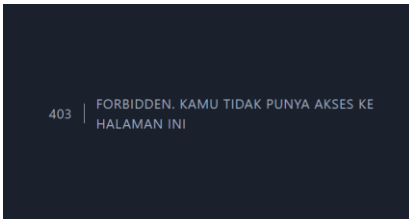
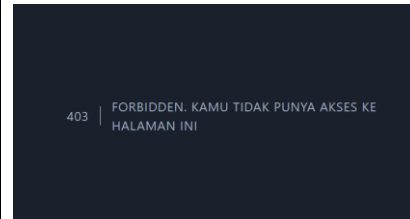
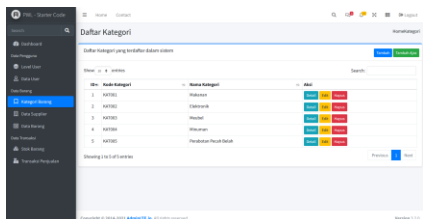


// ADM & MNG bisa menambah, mengedit, dan menghapus barang
Route::middleware(['authorize:ADM,MNG'])->group(function(){
    Route::group(['prefix' => 'barang'], function () {
        Route::get('/create', [BarangController::class, 'create']); // Form tambah barang
        Route::post('/', [BarangController::class, 'store']); // Simpan barang baru
// Create menggunakan AJAX
        Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // Form
tambah barang AJAX
        Route::post('/ajax', [BarangController::class, 'store_ajax']); // Simpan barang
baru AJAX
        Route::get('/{id}/edit', [BarangController::class, 'edit']); // Form edit barang
        Route::put('/{id}', [BarangController::class, 'update']); // Simpan perubahan
barang
        // Edit menggunakan AJAX
        Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // Form
edit barang AJAX
        Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); //
Simpan perubahan barang AJAX
        // Delete menggunakan AJAX
        Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); //
Form konfirmasi hapus barang AJAX
        Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); //
Hapus barang AJAX
        Route::delete('/{id}', [BarangController::class, 'destroy']); // Hapus barang
    });
});

// artinya semua route di dalam group ini harus punya role ADM (Administrator) dan MNG
(Manager)
Route::middleware(['authorize:ADM,MNG'])->group(function(){
```



```
Route::group(['prefix' => 'supplier'], function () {
    Route::get('/', [SupplierController::class, 'index']);
    Route::post('/list', [SupplierController::class, 'list']);
    Route::get('/create', [SupplierController::class, 'create']);
    Route::post('/', [SupplierController::class, 'store']);
    // Create menggunakan AJAX
    Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); //
menampilkan halaman form tambah Supplier ajax
    Route::post('/ajax', [SupplierController::class, 'store_ajax']); // menyimpan data
Supplier baru ajax
    Route::get('/{id}', [SupplierController::class, 'show']);
    Route::get('/{id}/edit', [SupplierController::class, 'edit']);
    Route::put('/{id}', [SupplierController::class, 'update']);
    // Edit menggunakan AJAX
    Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); //
menampilkan halaman form edit Supplier ajax
    Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']); //
menyimpan perubahan data Supplier ajax
    // Delete menggunakan AJAX
    Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']);
//menampilkan form confirm delete Supplier ajax
    Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']);
// menghapus data Supplier ajax
    Route::delete('/{id}', [SupplierController::class, 'destroy']);
});
});
});
```

HASIL:

| | Admin | Manager | Staff/kasir |
|---------------|---|--|---|
| Data User |  |  |  |
| Data Level |  |  |  |
| Data Kategori |  |  |  |



| | | | |
|----------------|--|--|--|
| Data
Barang | | | |
|----------------|--|--|--|

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

Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

- Menambah function register pada AuthController

```
public function register()  
{  
    $levels = LevelModel::select('level_id', 'level_nama')->get();  
    return view('auth.register')->with('levels', $levels);  
}
```

```
public function postRegister(Request $request)  
{  
    $validator = Validator::make($request->all(), [  
        'username' => 'required|string|unique:m_user,username',  
        'nama' => 'required|string|max:255',  
        'password' => 'required|string|min:5|confirmed',  
        'level_id' => 'required|exists:m_level,level_id',  
    ]);  
    if ($validator->fails()) {  
        return response()->json([  
            'status' => false,  
            'message' => 'Validation Failed.',  
            'errors' => $validator->errors(),  
        ]);  
    }  
    UserModel::create([  
        'username' => $request->username,  
        'nama' => $request->nama,  
        'password' => bcrypt($request->password),  
        'level_id' => $request->level_id  
    ]);  
    return response()->json([  
        'status' => true,  
        'message' => 'Registration Success.',  
        'redirect' => url('/login')  
    ]);  
}
```




- Menambah route register dan postregister

```
Route::get('/register', [AuthController::class, 'register']);  
Route::post('/register', [AuthController::class, 'postRegister']);
```

- Membuat register.blade.php pada folder auth

```
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <title>Register Pengguna</title>  
    <!-- Google Font: Source Sans Pro -->  
    <link rel="stylesheet"  
        href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">  
    <!-- Font Awesome -->  
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">  
    <!-- icheck bootstrap -->  
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">  
    <!-- SweetAlert2 -->  
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">  
    <!-- Theme style -->  
    <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">  
</head>  
  
<body class="hold-transition login-page">  
    <div class="login-box">  
        <!-- /.login-logo -->  
        <div class="card card-outline card-primary">  
            <div class="card-header text-center"><a href="{{ url('/') }}" class="h1"><b>Admin</b></a></div>  
            <div class="card-body">  
                <p class="login-box-msg">Register a new account</p>  
                <form action="{{ url('register') }}" method="POST" id="form-register">  
                    @csrf  
                    <div class="input-group mb-3">  
                        <input type="text" id="username" name="username" class="form-control" placeholder="Username" required>  
                        <div class="input-group-append">  
                            <div class="input-group-text">  
                                <span class="fas fa-user"></span>  
                            </div>  
                        <small id="error-username" class="error-text text-danger"></small>  
                    </div>  
  
                    <div class="input-group mb-3">  
                        <input type="text" id="name" name="nama" class="form-control" placeholder="Name" required>
```




```
<div class="input-group-append">
  <div class="input-group-text">
    <span class="fas fa-id-card"></span>
  </div>
</div>
<small id="error-name" 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" required>
  <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="input-group mb-3">
  <input type="password" id="password_confirmation"
name="password_confirmation"
class="form-control" placeholder="Confirm
Password" required>
  <div class="input-group-append">
    <div class="input-group-text">
      <span class="fas fa-lock"></span>
    </div>
  </div>
  <small id="error-password_confirmation"
class="error-text text-danger"></small>
</div>

<div class="input-group mb-3">
  <select id="level_id" name="level_id" class="form-
control" required>
    <option value="">Select Level</option>
    @foreach ($levels as $level)
      <option value="{{ $level->level_id }}">{{
$level->level_nama }}</option>
    @endforeach
  </select>
  <div class="input-group-append">
    <div class="input-group-text">
      <span class="fas fa-layer-group"></span>
    </div>
  </div>
  <small id="error-level_id" class="error-text text-
danger"></small>
</div>

<div class="row">
  <div class="col-8">
    <p>Already have an account? <a href="{{
url('/login') }}">Login</a></p>
  </div>
</div>
<!-- /.col -->
```



```

                <div class="col-4">
                    <button type="submit" class="btn btn-primary
btn-block">Register</button>
                </div>
            <!-- /.col -->
        </div>
    </form>
</div>
<!-- /.card-body -->
</div>
<!-- /.card -->
</div>
<!-- /.login-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>
<!-- jquery-validation -->
<script src="{ asset('adminlte/plugins/jquery-
validation/jquery.validate.min.js') }}"></script>
<script src="{ asset('adminlte/plugins/jquery-validation/additional-
methods.min.js') }}"></script>
<!-- SweetAlert2 -->
<script src="{
asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
<!-- AdminLTE App -->
<script src="{ asset('adminlte/dist/js/adminlte.min.js')
}}"></script>
<script>
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')
        }
    });
    $(document).ready(function() {
        $("#form-register").validate({
            rules: {
                username: {
                    required: true,
                    minlength: 4,
                    maxlength: 20
                },
                nama: {
                    required: true,
                    minlength: 2,
                    maxlength: 50
                },
                password: {
                    required: true,
                    minlength: 5
                },
                password_confirmation: {
                    required: true,
                    equalTo: '#password'
                },
                level_id: {

```



```
        required: true
    },
    submitHandler: function(form) {
        $.ajax({
            url: form.action,
            type: form.method,
            data: $(form).serialize(),
            success: function(response) {
                console.log(response);
                if (response.status) {
                    Swal.fire({
                        icon: 'success',
                        title: 'Registration Successful',
                        text: response.message,
                    }).then(function() {
                        window.location = response
                            .redirect;
                    });
                } else {
                    $('#error-text').text('');
                    $.each(response.errors, function(key, val) {
                        $('#error-' + key).text(val[0]);
                    });
                    Swal.fire({
                        icon: 'error',
                        title: 'Error Occurred',
                        text: response
                            .message
                    });
                }
            },
            error: function(xhr, status, error) {
                console.error(xhr
                    .responseText);
                Swal.fire({
                    icon: 'error',
                    title: 'Unexpected Error',
                    text: 'Please try again later.'
                });
            }
        });
        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>
```



- Menambah tombol sign up/register pada login.blade

```
</div>
<div class="text-center mt-3">
  <p>Don't have an account? <a href="{{ url('register') }}"
  <div class="text-primary">Sign up/Register</div>
</div>
'form'
```

2. Screenshot hasil yang kalian kerjakan

- Tombol register pada login

- Halaman Sign up/Register



- Proses sign up/register

The image shows two screenshots of the AdminLTE registration process. The left screenshot displays the 'AdminLTE' registration form with fields for username (sayveyy), password (veve), confirm password, and role (Staff). It includes a 'Register' button and a 'Login' link for existing users. The right screenshot shows a success message: 'Registration Successful' with a green checkmark and an 'OK' button.

- Data User bertambah

Show 10 entries

| No | Username | Nama | Level Pengguna | Aksi |
|----|----------|---------------|----------------|---|
| 1 | admin | Administrator | (ADM) | Detail Edit Hapus |
| 2 | manager | Manager | (MNG) | Detail Edit Hapus |
| 3 | staff | Staff | (STF) | Detail Edit Hapus |
| 4 | sayveyy | veve | (STF) | Detail Edit Hapus |

Showing 1 to 4 of 4 entries

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

*** Sekian, dan selamat belajar ***