



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



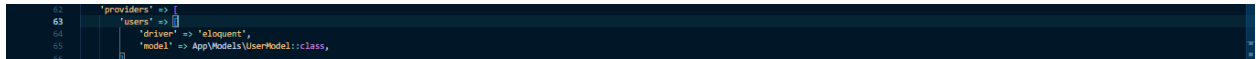
## Praktikum 1 – Implementasi Authentication :

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');  
    }  
}
```



```
Minggu_7 > Jobsheet7 > PWL_R05 > app > Models > UserModel.php > UserModel > level
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Relations\BelongsTo;
7 use Illuminate\Database\Eloquent\Model;
8
9 class UserModel extends Model
10 {
11     use HasFactory;
12
13     protected $table = 'u_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
14     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
15     protected $fillable = ['level_id', 'username', 'nama', 'password'];
16
17     protected $hidden = ['password']; // jangan di tampilkan saat select
18     protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
19
20     /**
21      * Relasi ke tabel Level
22      */
23     public function level(): BelongsTo
24     {
25         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
26     }
27
28
29 }
```

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');
    }
}
```



```
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         return redirect('login');
34     }
35
36     public function logout(Request $request)
37     {
38         Auth::logout();
39
40         $request->session()->invalidate();
41         $request->session()->regenerateToken();
42         return redirect('login');
43     }
44 }
```

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=fallback">
<!-- Font Awesome -->
<link rel="stylesheet" href="{ asset('plugins/fontawesome-free/css/all.min.css') }">
<!-- icheck bootstrap -->
<link rel="stylesheet" href="{ asset('plugins/icheck-bootstrap/icheck-bootstrap.min.css')
}"">
<!-- SweetAlert2 -->
<link rel="stylesheet" href="{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-
4.min.css') }"">
<!-- Theme style -->
<link rel="stylesheet" href="{ asset('dist/css/adminlte.min.css') }"">
</head>
<body class="hold-transition login-page">
<div class="login-box">
  <!-- /.login-logo -->
  <div class="card card-outline card-primary">
    <div class="card-header text-center"><a href="{ url('/') }"
class="h1"><b>Admin</b>LTE</a></div>
    <div class="card-body">
      <p class="login-box-msg">Sign in to start your session</p>
      <form action="{ url('login') }" method="POST" id="form-login">
        @csrf
        <div class="input-group mb-3">
          <input type="text" id="username" name="username" class="form-control"
placeholder="Username">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-envelope"></span>
            </div>
          </div>
          <small id="error-username" class="error-text text-danger"></small>
        </div>
        <div class="input-group mb-3">
          <input type="password" id="password" name="password" class="form-control"
placeholder="Password">
          <div class="input-group-append">
            <div class="input-group-text">
              <span class="fas fa-lock"></span>
            </div>
          </div>
          <small id="error-password" class="error-text text-danger"></small>
        </div>
        <div class="row">
          <div class="col-8">
            <div class="icheck-primary">
              <input type="checkbox" id="remember"><label for="remember">Remember Me</label>
            </div>
          <!-- /.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>
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>Login Pengguna</title>
7 <meta name="csrf-token" content="{{ csrf_token() }}">
8
9 <!-- Google Font: Source Sans Pro -->
10 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
11 <!-- Font Awesome -->
12 <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
13 <!-- iCheck Bootstrap -->
14 <link rel="stylesheet" href="{{ asset('adminlte/plugins/icheck-bootstrap/icheck-bootstrap.min.css') }}">
15 <!-- SweetAlert2 -->
16 <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
17 <!-- Theme style -->
18 <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
19 </head>
20 <body class="hold-transition login-page">
21 <div class="login-box">
22 <div class="card outline card-primary">
23 <div class="card-header text-center">
24 <a href="{{ url('/') }}" class="h1"><b>Admin</b></a>
25 </div>
26 <div class="card-body">
27 <p class="login-box-msg">Sign in to start your session</p>
28
29 <form action="{{ url('login') }}" method="post" id="form-login">
30 <csrf
31 <div class="input-group mb-3">
32 <input type="text" id="username" name="username" class="form-control" placeholder="Username" required>
33 <div class="input-group-append">
34 <div class="input-group-text">
35 <span class="fas fa-user"></span>
36 </div>
37 </div>
38 <small id="error-username" class="error-text text-danger"></small>
39 </div>
40 <div class="input-group mb-3">
41 <input type="password" id="password" name="password" class="form-control" placeholder="Password" required>
42 <div class="input-group-append">
43 <div class="input-group-text">
44 <span class="fas fa-lock"></span>
45 </div>
46 </div>
47 <small id="error-password" class="error-text text-danger"></small>
48 </div>
49 </form>
```

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

});
```

```
126 Route::pattern('id','[0-9]+'); // artinya ketika ada parameter {id}, maka harus berupa angka
127
128 Route::get('login', [AuthController::class,'login'])->name('login');
129 Route::post('login', [AuthController::class,'postlogin']);
130 Route::get('logout', [AuthController::class,'logout'])->middleware('auth');
131
132 Route::middleware(['auth'])->group(function(){ // artinya semua route di dalam group ini harus login dulu
133
134     // masukkan semua route yang perlu autentikasi di sini
135
136 });
```

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



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>

## AdminLTE

Sign in to start your session

Username



Password



☐ Remember Me

Sign In

## AdminLTE

Sign in to start your session

Username



Password



☐ Remember Me

Sign In

Don't have an account? [Register here](#)

### Tugas 1 – Implementasi Authentication :

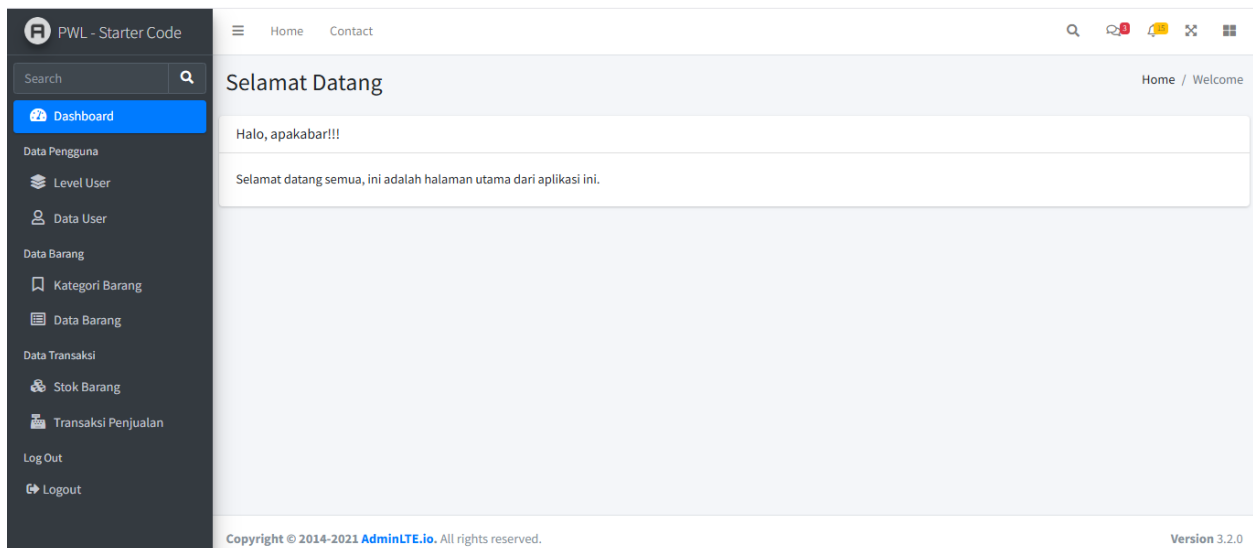
1. Silahkan implementasikan proses login pada project kalian masing-masing



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>

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 checkbox labeled 'Remember Me' and a blue 'Sign In' button.

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



3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan Menampilkan tampilan login dimana login pengguna nanti menggunakan username dan password yang sudah dibuat dan diambil melalui database myphpadmin.

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

```
/**
 * Relasi ke tabel level
 */
public function level(): BelongsTo
{
    return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
}

/**
 * Mendapatkan nama role
 */
public function getRoleName(): string
{
    return $this->level->level_nama;
}

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



```
23  /**  
24   * Relasi ke tabel Level  
25   */  
26  public function level(): BelongsTo  
27  {  
28      return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');  
29  }  
30  
31  /**  
32   * Mendapatkan nama role  
33   */  
34  public function getRoleName(): string  
35  {  
36      return $this->level->level_nama;  
37  }  
38  
39  /**  
40   * Cek apakah user memiliki role tertentu  
41   */  
42  public function hasRole($role): bool  
43  {  
44      return $this->level->level_kode == $role;  
45  }  
46  }
```

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

```
1  <?php  
2  namespace App\Http\Middleware;  
3  
4  use Closure;  
5  use Illuminate\Http\Request;  
6  use Symfony\Component\HttpFoundation\Response;  
7  
8  class AuthorizeUser  
9  {  
10     /**  
11      * Handle an incoming request.  
12      *  
13      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  
14      */  
15     public function handle(Request $request, Closure $next, $role = ''): Response  
16     {  
17         $user = $request->user(); // ambil data user yg login  
18         // fungsi user() diambil dari UserModel.php  
19         if($user->hasRole($role)){ // cek apakah user punya role yg diinginkan  
20             return $next($request);  
21         }  
22         // jika tidak punya role, maka tampilkan error 403  
23         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');  
24     }  
25 }
```



```
Minggu_7 > Jobsheet7 > PWL_POS > app > Http > Middleware > AuthorizeUser.php > AuthorizeUser > handle
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 = $request->user(); // ambil data user yang login
19         // fungsi user() diambil dari UserModel.php
20         if($user->hasRole($role)){
21             return $next($request); // cek apakah user punya role yang diinginkan
22         }
23         // jika tidak punya role, maka tampilan error 403
24         abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');
25     }
26 }
```

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,
```

```
57 | 'authorize' => \App\Http\Middleware\AuthorizeUser::class, // middleware yang kita buat
```

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
```

```
168 Route::middleware(['authorize:ADM'])->group(function() {
169     Route::get('/', [LevelController::class, 'index']); // Menampilkan halaman awal level user
170     Route::post('/list', [LevelController::class, 'list']); // menampilkan level user dalam bentuk json untuk datatables
171     Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level user
172     Route::post('/', [LevelController::class, 'store']); // menyimpan level user baru
173     Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman form edit level user
174     Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan level user
175     Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus level user
176 });
```

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

403 | FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI





## **Tugas 2 – Implementasi Authoriization :**

1. Apa yang kalian pahami pada praktikum 2 ini?

Mempelajari mengenai authorization dimana authorization merupakan cara untuk menentukan level user untuk bisa mengakses fitur fitur tertentu.

2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
  - a. Melakukan modif di dalam user model untuk menambahkan fungsi getrolename dan hasname untuk melakukan pemanggilan dari database
  - b. Membuat auth didalam middleware dengan perintah di dalam terminal
  - c. Melakukan edit didalam authuser untuk melakukan cek kepada role
  - d. Mendaftarkan authorizeruser ke kernel didalam http
  - e. Melakukan cek didatabase phpmyadmin untuk tabel m\_user apakah sudah sesuai
  - f. Melakukan edit di route untuk menambahkan authorize:ADM sesuai dengan role tersebut ingin mengakses fitur apa saja
  - g. Kemudian melakukan ujicoba jika user admin melakukan akses ke user bisa namun jika ke fitur lain tidak bisa karena pembatasan hak akses dan akan muncul error 403
3. Submit kode untuk impementasi Authorization pada repository github kalian.



## C. Multi-Level Authorization di Laravel

Bagaimana seandainya jika terdapat level/group/role satu dengan yang lain memiliki hak akses yang sama. Contoh sederhana, user level Admin dan Manager bisa sama-sama mengakses menu Barang pada aplikasi yang kita buat. Maka tidak mungkin kalau kita buat route untuk masing-masing level user. Hal ini akan memakan banyak waktu, dan proses yang lama.

```
// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});

// artinya semua route di dalam group ini harus punya role MNG (Manager)
Route::middleware(['authorize:MNG'])->group(function(){
    Route::get('/barang',[BarangController::class,'index']);
    Route::post('/barang/list',[BarangController::class,'list']);
    Route::get('/barang/create_ajax',[BarangController::class,'create_ajax']); // ajax form create
    Route::post('/barang_ajax',[BarangController::class,'store_ajax']); // ajax store
    Route::get('/barang/{id}/edit_ajax',[BarangController::class,'edit_ajax']); // ajax form edit
    Route::put('/barang/{id}/update_ajax',[BarangController::class,'update_ajax']); // ajax update
    Route::get('/barang/{id}/delete_ajax',[BarangController::class,'confirm_ajax']); // ajax form confirm delete
    Route::delete('/barang/{id}/delete_ajax',[BarangController::class,'delete_ajax']); // ajax delete
});
```

Hal ini jadi kendala ketika kita mau mengganti hak akses, maka kita akan mengganti sebagian besar route yang sudah kita tulis. Untuk itu, kita perlu mengelola middleware agar bisa mendukung penambahan hak akses secara dinamis.

### Praktikum 3 – Implementasi Multi-Level Authorizat on 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 nama role
 */
public function getRoleName(): string
{
    return $this->level->level_nama;
}

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

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

```
32  * Mendapatkan nama role
33  */
34  public function getRoleName(): string
35  {
36      return $this->level->level_nama;
37  }
38
39  /**
40   * Cek apakah user memiliki role tertentu
41   */
42  public function hasRole($role): bool
43  {
44      return $this->level->level_kode == $role;
45  }
46
47  /**
48   * Mendapatkan kode role
49   */
50  public function getRole()
51  {
52      return $this->level->level_kode;
53  }
54 }
```

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

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



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>

```
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 = $request->user()->getRole(); // ambil data level_hode dari user yang Login  
19      if(in_array($user_role, $roles)) {  
20          return $next($request); // cek apakah user punya role yang diinginkan  
21      }  
22      // jika tidak punya role, maka tampilan error 403  
23      abort(403, 'Forbidden. Kamu tidak punya akses ke halaman ini');  
24  }  
25  }
```

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

403 | FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI

### Tugas 3 – Implementasi Multi-Level Authorization :

#### 1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing

```
30 Route::pattern('id', '[0-9]+');
31
32 Route::get('login',[AuthController::class,'login'])->name('login');
33 Route::post('login',[AuthController::class,'postlogin']);
34 Route::get('logout',[AuthController::class,'logout'])->middleware('auth');
35
36 Route::group(['prefix' => 'user'], function () {
37     Route::middleware(['authorize:ADM,MNG,STF,CUS'])->group(function () {
38         Route::get('/',[UserController::class,'index']); // Menampilkan halaman awal level user
39         Route::post('/list',[UserController::class,'list']); // menampilkan level user dalam bentuk json untuk datatables
40         Route::get('/create',[UserController::class,'create']); // menampilkan halaman form tambah level user
41         Route::post('/',[UserController::class,'store']); // menyimpan level user baru
42         Route::get('/create_ajax',[UserController::class,'create_ajax']); // menampilkan halaman form tambah user Ajax
43         Route::post('/ajax',[UserController::class,'store_ajax']); // menyimpan data user baru Ajax
44         Route::get('/{id}',[UserController::class,'show']); // Menampilkan detail level user
45         Route::get('/{id}/edit',[UserController::class,'edit']); // menampilkan halaman form edit level user
46         Route::put('/{id}',[UserController::class,'update']); // menyimpan perubahan level user
47         Route::get('/{id}/edit_ajax',[UserController::class,'edit_ajax']); // Menampilkan halaman form edit user Ajax
48         Route::put('/{id}/update_ajax',[UserController::class,'update_ajax']); // Menyimpan perubahan data user Ajax
49         Route::get('/{id}/delete_ajax',[UserController::class,'confirm_ajax']); // Untuk tampilan form confirm delete user Ajax
50         Route::delete('/{id}/delete_ajax',[UserController::class,'delete_ajax']); // Untuk hapus data user Ajax
51         Route::delete('/{id}',[UserController::class,'destroy']); // menghapus level user
52     });
53 });
54
55 Route::group(['prefix' => 'product'], function () {
56     Route::middleware(['authorize:MNG,ADM'])->group(function () {
57         Route::get('/',[ProductController::class,'index']); // Menampilkan halaman awal level user
58         Route::post('/list',[ProductController::class,'list']); // menampilkan level user dalam bentuk json untuk datatables
59         Route::get('/create',[ProductController::class,'create']); // menampilkan halaman form tambah level user
60         Route::post('/',[ProductController::class,'store']); // menyimpan level user baru
61         Route::get('/create_ajax',[ProductController::class,'create_ajax']); // menampilkan halaman form tambah user Ajax
62         Route::post('/ajax',[ProductController::class,'store_ajax']); // menyimpan data user baru Ajax
63         Route::get('/{id}',[ProductController::class,'show']); // Menampilkan detail level user
64         Route::get('/{id}/edit',[ProductController::class,'edit']); // menampilkan halaman form edit level user
65         Route::put('/{id}',[ProductController::class,'update']); // menyimpan perubahan level user
66         Route::get('/{id}/edit_ajax',[ProductController::class,'edit_ajax']); // Menampilkan halaman form edit user Ajax
67         Route::put('/{id}/update_ajax',[ProductController::class,'update_ajax']); // Menyimpan perubahan data user Ajax
68     });
69 });
```



2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
  - a. Melakukan modif di usermodel untuk menambahkan fungsi getrole untuk memanggil semua role dari database
  - b. Melakukan modif di authuser untuk melakukan cek menggunakan in\_array apakah role tersebut berada didalam database
  - c. Melakukan modif di route untuk menambahkan authorize:ADM untuk melakukan halaman tersebut hanya bisa diakses oleh admin
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User

```
124 Route::pattern('id', '[0-9]*'); // artinya ketika ada parameter (id), maka harus berupa angka
125
126 Route::get('login', [AuthController::class, 'login'])->name('login');
127 Route::post('login', [AuthController::class, 'postlogin']);
128 Route::get('logout', [AuthController::class, 'logout'])->middleware('logout');
129
130 Route::middleware(['auth'])->group(function () { // artinya semua route di dalam group ini harus login dulu
131     Route::get('/', [WelcomeController::class, 'index']);
132
133     // masukkan semua route yang perlu autentikasi di sini
134     Route::group(['prefix' => 'user'], function () {
135         Route::middleware(['authorize:ADM,MMG,SIF,CUS'])->group(function () {
136             Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user
137             Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
138             Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
139             Route::post('/', [UserController::class, 'store']); // menyimpan data user baru
140             Route::get('/create_ajax', [UserController::class, 'create_ajax']); // menampilkan halaman form tambah user Ajax
141             Route::post('/ajax', [UserController::class, 'store_ajax']); // menyimpan data user baru Ajax
142             Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user
143             Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user
144             Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
145             Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user Ajax
146             Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // menyimpan perubahan data user Ajax
147             Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // untuk tampilan form confirm delete user Ajax
148             Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // untuk hapus data user Ajax
149             Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
150         });
151     });
152
153     Route::group(['prefix' => 'level'], function () {
154         Route::middleware(['authorize:ADM,MMG'])->group(function () {
155             Route::get('/', [LevelController::class, 'index']); // Menampilkan halaman awal level user
156             Route::post('/list', [LevelController::class, 'list']); // menampilkan level user dalam bentuk json untuk datatables
157             Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level user
158             Route::post('/', [LevelController::class, 'store']); // menampilkan level user baru
159             Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // menampilkan halaman form tambah user Ajax
160             Route::post('/ajax', [LevelController::class, 'store_ajax']); // menyimpan data user baru Ajax
161             Route::get('/{id}', [LevelController::class, 'show']); // Menampilkan detail level user
```

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

#### Tugas 4 – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.





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>

**PWL - Starter Code**

New User Registration

Pelanggan

devin

Devin Izaz Radin Dewantoro

.....

**Register**

Already have an account? [Sign In](#)

## 2. Screenshot hasil yang kalian kerjakan

**PWL - Starter Code**

Home Contact

Selamat Datang

Halo, apakabar!!!

Selamat datang semua, ini adalah halaman utama dari aplikasi ini.

Copyright © 2014-2021 AdminLTE.io. All rights reserved. Version 3.2.0

phpMyAdmin

Server: localhost:3306 Database: pwl\_pos Table: m\_user

	user_id	username	nama	password	level_id	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	admin	Administrator	\$2y\$12\$GfVhOuWaTiduolMAfomWou6UIzhBufnztDbTRfWx...	1	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	manager	Manager	\$2y\$12\$JDS23iO7zLnIMKRm0f05Qe8yRCYuwHuzrN1bqx4J82...	2	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	staff	Staff/Kasir	\$2y\$12\$2eFB/Wmvtc7J02syRkp2wuS42DGoddv0pgBEoYAK...	3	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	16	customer-1	Pelanggan Pertama	\$2y\$12\$T.sBvA3ZYxDR2WuEJlhhHONIBcB.206mLuaEZgv5gSH...	5	NULL	2025-03-05 02:42:14
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	17	manager2	Manager 2	\$2y\$12\$5aGYrA.F4v2C5woEBOGru1su0G67thDLPEdEITJ7e...	2	2025-03-06 06:51:27	2025-03-06 06:51:27
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	18	manager22	Manager Dua Dua	\$2y\$12\$WjWtdJzeNhy0Ygz78iOEgmerHATZQe4iWTRt7go...	2	2025-03-06 11:45:08	2025-03-06 11:45:08
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	19	manager33	Manager Tiga Tiga	\$2y\$12\$FqrPPTnGWUEhizgF2kcGuF501OXip2U3luG9NqCx1F...	2	2025-03-06 12:02:23	2025-03-06 12:02:23
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	20	manager56	Manager55	\$2y\$12\$2nli5rDKL40roYnKGilhedC7FKyxewFUT5w340FOjn...	2	2025-03-06 12:09:26	2025-03-06 12:09:26
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	21	manager12	Manager11	\$2y\$12\$mEBTSKZ9NQob.KmfOo9Tku0ux5A7vk3cns8AI5acZ7...	2	2025-03-06 12:17:55	2025-03-06 12:17:55
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	26	Ahmad	Ahmad Bejo	\$2y\$12\$Shkm.FFmFvFUzV7qhw2GeChl85zlwKq9E/AzzB4D...	1	2025-03-26 07:15:52	2025-03-27 08:50:54
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	28	devin	Devin Radin	\$2y\$12\$GkxRU692D87/mJmYQmoNODO9Qb9AXLjwhwtkvcH9...	1	2025-03-30 06:48:37	2025-03-30 06:48:37



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

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