



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Sistem Informasi Bisnis
Semester : 5

Kelas : SIB
NIM : 2241760007
Nama : Anaradi Octa Lavechia
Jobsheet Ke- : 7 (tujuh)

Laporan Jobsheet 07 - Authentication dan *Authorization* di Laravel

Praktikum Ke-1 Implementasi Authentication

Langkah	Jawaban/Deskripsi
1	<p>Kita buka project laravel PWL_POS kita, dan kita modifikasi konfigurasi aplikasi kita di config/auth.php</p> <pre>config > auth.php 3 return [61 62 'providers' => [63 'users' => [64 'driver' => 'eloquent', 65 'model' => App\Models\User::class, 66],</pre> <p>Pada bagian ini kita sesuaikan dengan Model untuk tabel m_user yang sudah kita buat</p> <pre>config > auth.php 3 return [61 62 'providers' => [63 'users' => [64 'driver' => 'eloquent', 65 'model' => App\Models\UserModel::class, 66],</pre>
2	Selanjutnya kita modifikasi sedikit pada UserModel.php untuk bisa melakukan



proses otentikasi

```
app > Models > UserModel.php > PHP Intelephense > UserModel
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;
8 use Illuminate\Foundation\Auth\User as Authenticatable; //implementasi class Authenticatable
9 use Illuminate\Foundation\Auth\User as Authenticatable;
10
11 16 references | 0 implementations
12 class UserModel extends Authenticatable
13 {
14     use HasFactory;
15
16     0 references
17     protected $table = 'm_user';
18     0 references
19     protected $primaryKey = 'user_id';
20     0 references
21     protected $fillable = ['username', 'password', 'name', 'level_id', 'created_at', 'updated_at'];
22     0 references
23     protected $hidden = ['password']; //jangan ditampilkan saat select
24     0 references
25     protected $casts = ['password' => 'hashed']; //casting password agar otomatis di hash
26
27     0 references | 0 overrides
28     public function level(): BelongsTo
29     {
30         return $this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id');
31     }
32
33     0 references | 0 overrides
34     public function getRoleName(): string
35     {
36         return $this->level->level_name;
37     }
38
39     0 references | 0 overrides
40     public function hasRole($role): bool
41     {
42         return $this->level->level_kode == $role;
43     }
44 }
```

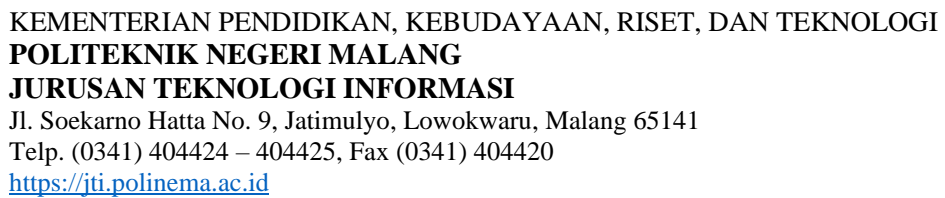
3

Selanjutnya kita buat **AuthController.php** untuk memproses login yang akan kitalakukan

```
app > Http > Controllers > AuthController.php > PHP Intelephense > AuthController > postlogin
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7
8 0 references | 0 implementations
9 class AuthController extends Controller
10 {
11     0 references | 0 overrides
12     public function login(): Factory|Redirector|RedirectResponse|View
13     {
14         if(Auth::check()) { //jika sudah login, maka redirect ke halaman home
15             return redirect(to: '/');
16         }
17         return view(view: 'auth.login');
18     }
19
20     0 references | 0 overrides
21     public function postlogin(Request $request): JsonResponse|mixed|Redirector|RedirectRes...
22     {
23         if ($request->ajax() || $request->wantsJson()) {
24             $credentials = $request->only(keys: 'username', 'password');
25             if (Auth::attempt(credentials: $credentials)) {
26                 return response()->json(data: [
27                     'status' => true,
28                     'message' => 'Login Berhasil',
29                     'redirect' => url(path: '/')
30                 ]);
31             }
32             return response()->json(data: [
33                 'status' => false,
34                 'message' => 'Login Gagal'
35             ]);
36         }
37         return redirect(to: 'login');
38     }
39
40     0 references | 0 overrides
41     public function logout(Request $request): Redirector|RedirectResponse
42     {
43         Auth::logout();
44         $request->session()->invalidate();
45         $request->session()->regenerateToken();
46         return redirect(to: 'login');
47     }
48 }
```

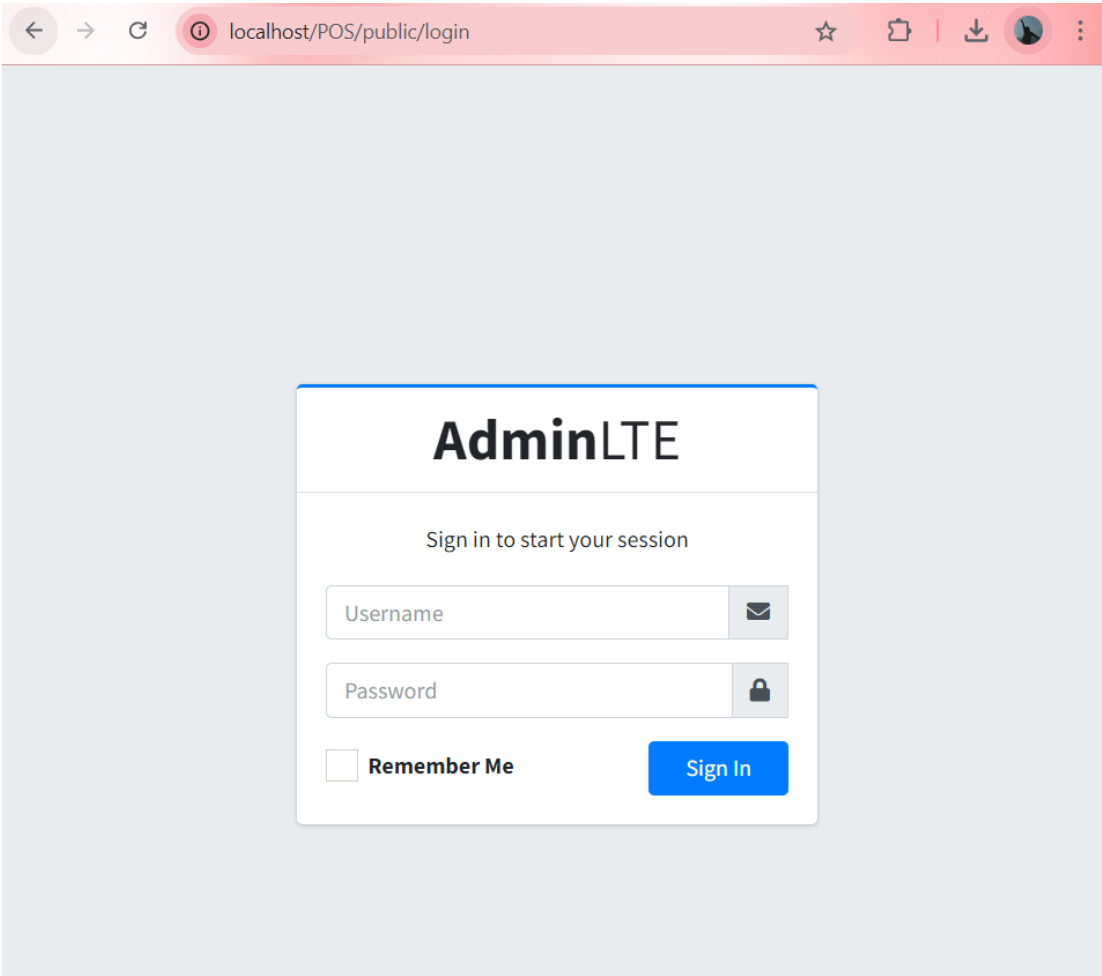
4

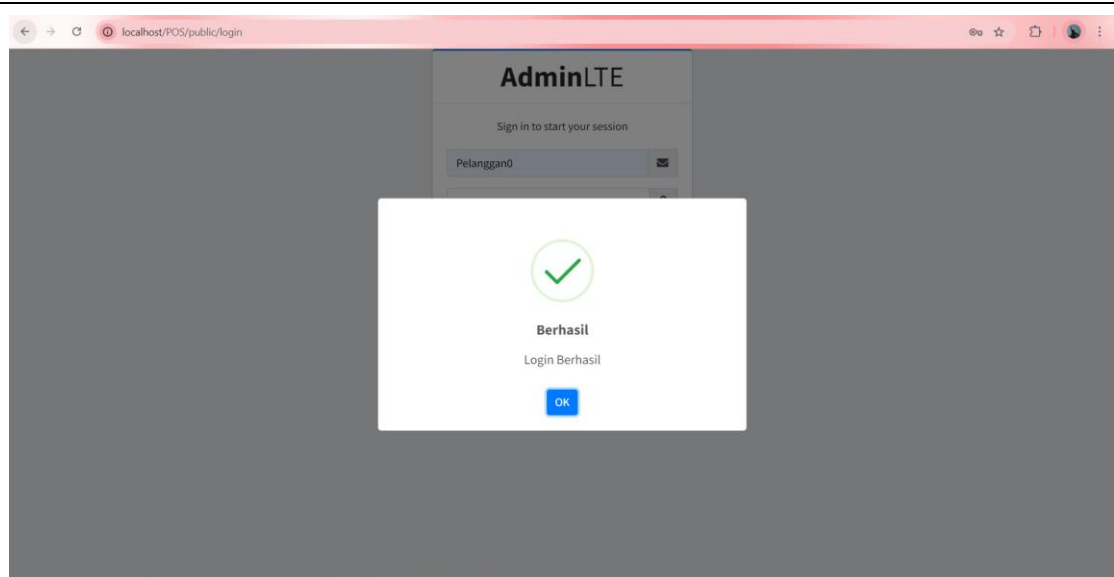
Setelah kita membuat **AuthController.php**, kita buat view untuk menampilkan

[illegible]

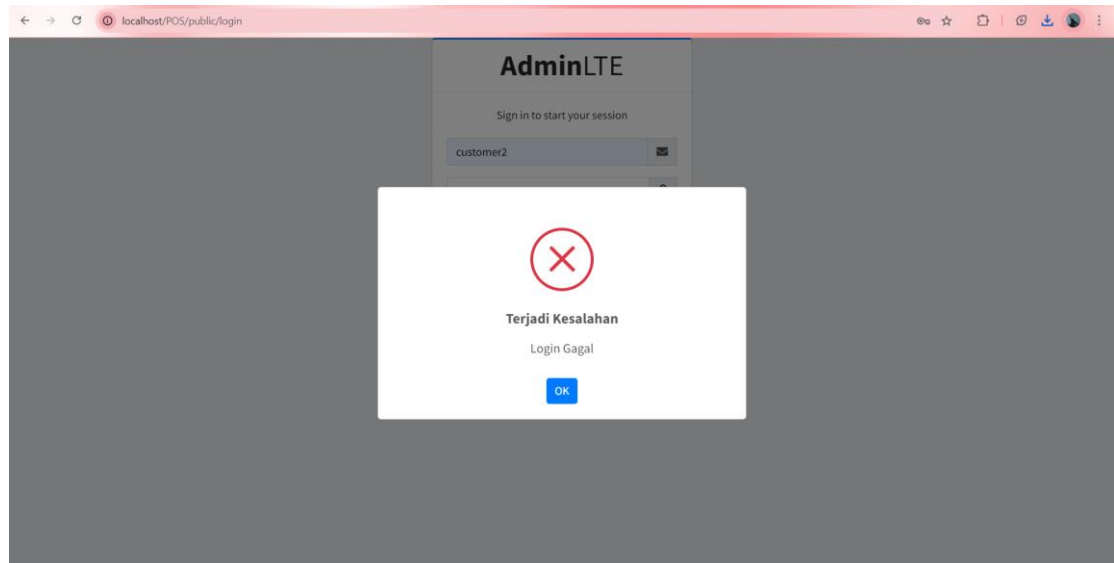
Kemudian kita modifikasi `route/web.php` agar semua route masuk dalam auth



	<pre>routes > web.php > ... You can unsubscribe at any time. (Press Enter to confirm or Escape to cancel) 1 <?php 2 3 4 use App\Http\Controllers\WelcomeController; 5 use Illuminate\Support\Facades\Route; 6 use App\Http\Controllers\UserController; 7 use App\Http\Controllers\LevelController; 8 use App\Http\Controllers\KategoriController; 9 use App\Http\Controllers\SupplierController; 10 use App\Http\Controllers\BarangController; 11 use App\Http\Controllers\AuthController; 12 13 Route::pattern(key: 'id', pattern: '[0-9]+'); //artinya ketika ada parameter (id), maka harus berupa angka 14 15 Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login'); 16 Route::post(uri: 'login', action: [AuthController::class, 'postlogin']); 17 Route::get(uri: 'logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth'); 18 19 Route::middleware(middleware: ['auth'])->group(callback: function (): void { //artinya semua route di dalam group ini harus login dulu 20 21 Route::get(uri: '/', action: [WelcomeController::class, 'index']);</pre>
6	<p>Ketika kita coba mengakses halaman localhost/PWL_POS/public maka akan tampil halaman awal untuk login ke aplikasi seperti berikut.</p>  <p>Berikut adalah tampilan halaman jika username dan password yang dimasukan sudah sesuai dengan pop up Login Berhasil.</p>



Namun, jika terdapat username atau password yang tidak cocok dengan data yang ada pada database maka tampilannya akan seperti berikut yang menampilkan pop up Terjadi Kesalahan yang menyatakan bahwa Login Gagal.



Tugas-1 Implementasi Authentication-Logout

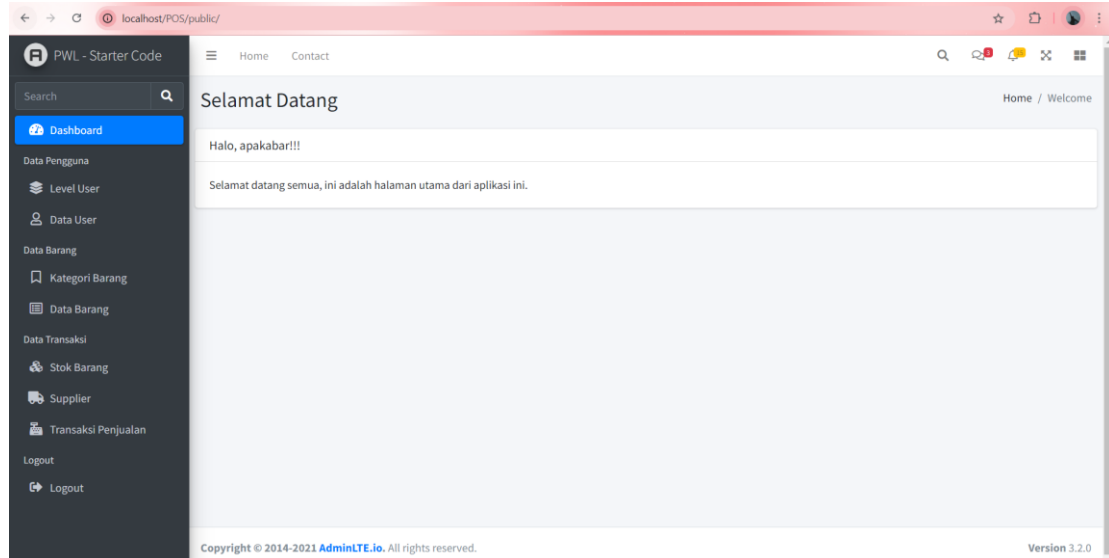
Langkah	Jawaban/Deskripsi
1	Lakukan modifikasi pada file sidebar.blade.php pada folder resources/views/layouts, dengan tambahan kode seperti berikut untuk menambahkan tampilan button menu logout.



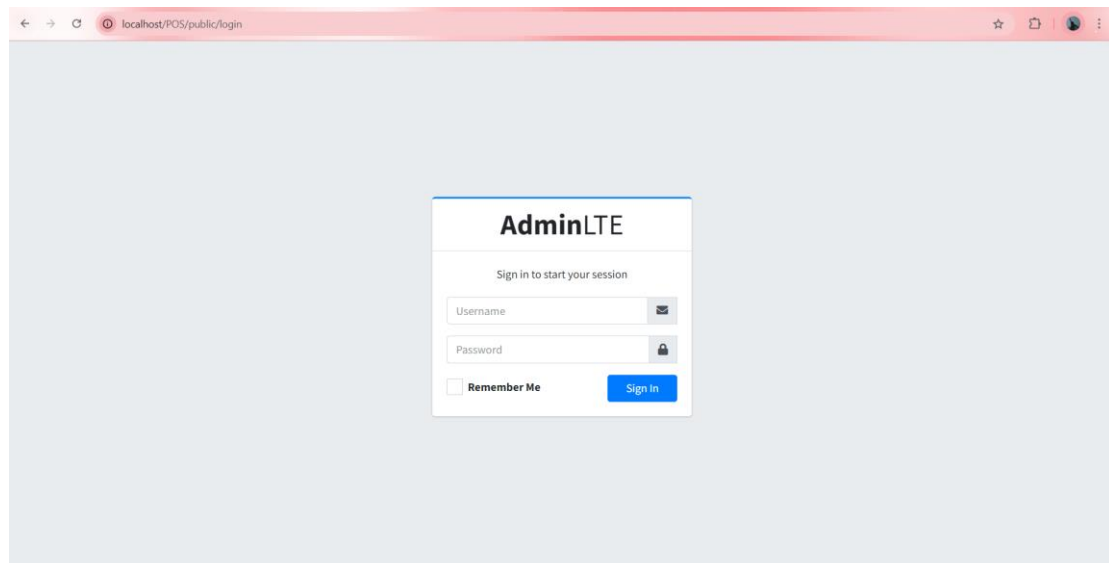
```
resources > views > layouts > sidebar.blade.php > div.sidebar > nav.mt-2 > ul.nav.nav-pills.nav-sidebar.flex-column > li.nav-item > form#logout-form
1 <div class="sidebar">
25 <nav class="mt-2">
26 <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-accordion="false">
80 <!-- Menambahkan Menu Logout -->
81 <li class="nav-header">Logout</li>
82 <li class="nav-item">
83 <a href="{{ url('/logout') }}" class="nav-link"
84 onclick="event.preventDefault(); document.getElementById('logout-form').submit();"
85 <i class="nav-icon fas fa-sign-out-alt"></i>
86 <p>Logout</p>
87 </a>
88 <form id="logout-form" action="{{ url('/logout') }}" method="GET" style="display: none;"
89 @csrf
90 </form>
91 </li>
92 </ul>
93 </nav>
94 <!-- /.sidebar-menu -->
95 </div>
```

2

Maka tampilan halaman akan seperti ini. Terdapat menu logout pada sidebar.



Dan ketika memilih logout, maka akan terarah kembali pada halaman login.





3	Submit kode untuk implementasi Authentication pada repository github kalian.
---	--

Praktikum Ke-2 Implementasi *Authorization* di Laravel dengan Middleware

Langkah	Jawaban/Deskripsi
1	<p>Kita modifikasi UserModel.php dengan menambahkan kode berikut</p> <pre>app > Models > UserModel.php > PHP Intelephense > UserModel > getRoleName 10 class UserModel extends Model 0 references 0 overrides 24 public function level(): BelongsTo 25 { 26 return \$this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id'); 27 } 28 0 references 0 overrides 29 public function getRoleName(): string 30 { 31 return \$this->level->level_nama; 32 } 33 0 references 0 overrides 34 public function hasRole(\$role): bool 35 { 36 return \$this->level->level_kode == \$role; 37 } 38 }</pre>
2	<p>Kemudian kita buat <i>middleware</i> dengan nama AuthorizeUser.php. Kita bisa buat <i>middleware</i> dengan mengetikkan perintah pada terminal/CMD</p> <pre>php artisan make:middleware AuthorizeUser</pre> <p>File <i>middleware</i> akan dibuat di app/Http/Middleware/AuthorizeUser.php</p> <pre>▼ TERMINAL JavaSE-23 + Active code page: 65001 ady loaded in Unknown on line 0 INFO Middleware [C:\laragon\www\POS\app\Http\Middleware\AuthorizeUser.php] created successfully. C:\laragon\www\POS></pre>
3	<p>Kemudian kita edit <i>middleware</i> AuthorizeUser.php untuk bisa mengecek apakah pengguna yang mengakses memiliki Level/Role/Group yang sesuai</p>



```

app > Http > Middleware > AuthorizeUser.php > PHP Intelephense > 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  0 references | 0 implementations
10 class AuthorizeUser
11 {
12     /**
13      * Handle an incoming request.
14      *
15      * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
16      */
17     0 references | 0 overrides
18     public function handle(Request $request, Closure $next, $role): Response
19     {
20         $user = $request->user(); //ambil data user yang login
21         // fungsi user() diambil dari UserModel.php
22         if($user->hasRole($role)){ //cek apakah user punya role yang diinginkan
23             return $next($request);
24         }
25         //jika tidak punya role, maka tampilkan error 403
26         abort(code: 403,message: 'Forbidden, kamu tidak punya akses ke halaman ini');
27     }
28 }

```

4

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

```

app > Http > Kernel.php > PHP Intelephense > Kernel > $middlewareAliases
7  class Kernel extends HttpKernel
8  {
9
10     /**
11      * The middleware aliases.
12      *
13      * @var array
14      */
15     protected $middlewareAliases = [
16
17         'auth' => \App\Http\Middleware\Authenticate::class,
18         'authorize' => \App\Http\Middleware\AuthorizeUser::class,
19         'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
20         'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
21         'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
22         'can' => \Illuminate\Auth\Middleware\Authorize::class,
23         'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
24         'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
25         'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
26         'signed' => \App\Http\Middleware\ValidateSignature::class,
27         'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
28         'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
29     ];
30 }

```

5

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

Server: localhost » Database: pwl_pos » Table: m_level

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Showing rows 0 - 4 (5 total, Query took 0.0016 seconds.)

SELECT * FROM `m_level`

☐ Profiling
 [\[Edit inline \]](#)
[\[Edit \]](#)
[\[Explain SQL \]](#)
[\[Create PHP code \]](#)
[\[Refresh \]](#)

☐ Show all
 Number of rows: 25
 Filter rows:
Sort by key: None

Extra options

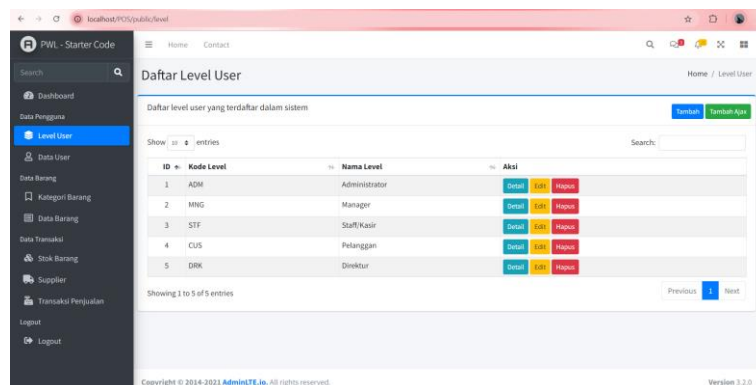
	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	CUS	Pelanggan	2024-09-14 06:00:31	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	DRK	Direktur	2024-10-04 06:24:07	2024-10-04 06:24:07

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

```
routes > web.php > Closure > Closure
19 Route::middleware('auth')->group(callback: function (): void { //artinya semua route di dalam group ini harus login dulu
41 //untuk M_LEVEL
42 Route::middleware('authorize:ADM')->group(callback: function (): void {
43 Route::get(uri: '/level', action: [LevelController::class, 'index']); // menampilkan halaman awal level
44 Route::post(uri: '/level/list', action: [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
45 Route::get(uri: '/level/create', action: [LevelController::class, 'create']); // menampilkan halaman form tambah level
46 Route::get(uri: '/level/create_ajax', action: [LevelController::class, 'create_ajax']);
47 Route::post(uri: '/level', action: [LevelController::class, 'store']); // menyimpan data level baru
48 Route::post(uri: '/level/ajax', action: [LevelController::class, 'store_ajax']);
49 Route::get(uri: '/level/{id}/edit_ajax', action: [LevelController::class, 'edit_ajax']);
50 Route::put(uri: '/level/{id}/update_ajax', action: [LevelController::class, 'update_ajax']);
51 Route::get(uri: '/level/{id}/delete_ajax', action: [LevelController::class, 'confirm_ajax']);
52 Route::delete(uri: '/level/{id}/delete_ajax', action: [LevelController::class, 'delete_ajax']);
53 Route::get(uri: '/level/{id}', action: [LevelController::class, 'show']); // menampilkan detail level
54 Route::get(uri: '/level/{id}/edit', action: [LevelController::class, 'edit']); // menampilkan halaman form edit level
55 Route::put(uri: '/level/{id}', action: [LevelController::class, 'update']); // menyimpan perubahan data level
56 Route::delete(uri: '/level/{id}', action: [LevelController::class, 'destroy']); // menghapus data level
57 });
```

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

Disini saya mencoba login menggunakan akun pelanggan, maka tampilannya 403





Tugas-2 Implementasi Authoriization

Langkah	Jawaban/Deskripsi
1	Apa yang kalian pahami pada praktikum 2 ini? Berdasarkan pemahaman saya dari praktikum di atas, praktikum 2 ini mengelola akses user yang bisa mengakses tiap menu. Seperti di atas itu kan ada menu data level user yang diatur agar hanya admin saja yang dapat mengelola menu tersebut.
2	Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3	Submit kode untuk impementasi Authorization pada repository github kalian.

Praktikum Ke-3 Implementasi Multi-Level Authorizaton di Laravel dengan Middleware

Langkah	Jawaban/Deskripsi
1	Kita modifikasi <code>UserModel.php</code> dengan menambahkan kode berikut <pre>app > Models > UserModel.php > PHP > UserModel > getRoleName() 11 class UserModel extends Authenticatable 22 30 0 references 0 overrides public function getRoleName(): string 31 { 32 return \$this->level->level_nama; 33 } 34 35 0 references 0 overrides public function hasRole(\$role): bool 36 { 37 return \$this->level->level_kode == \$role; 38 } 39 40 0 references 0 overrides public function getRole(): mixed 41 { 42 return \$this->level->level_kode; 43 } 44 }</pre>
2	Selanjutnya, Kita modifikasi middleware <code>AuthorizeUser.php</code> dengan kode berikut



	<pre>app > Http > Middleware > AuthorizeUser.php > ... 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 1 reference 0 implementations 10 class AuthorizeUser 11 { 12 /** 13 * Handle an incoming request. 14 * 15 * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) \$next 16 */ 17 public function handle(Request \$request, Closure \$next, ... \$role): Response 18 { 19 \$user_role = \$request->user()->getRole(); //ambil data user yang login 20 if(in_array (needle: \$user_role, haystack: \$role)){ //cek apakah user punya role yang diinginkan 21 return \$next(\$request); 22 } 23 //jika tidak punya role, maka tampilkan error 403 24 abort(code: 403,message: 'Forbidden, kamu tidak punya akses ke halaman ini'); 25 } 26 }</pre>
3	<p>Setelah itu tinggal kita perbaiki <code>route/web.php</code> sesuaikan dengan role/level yang diinginkan. Contoh</p> <pre>routes > web.php > Closure > Closure 19 Route::middleware(['auth'])->group(function () { //artinya semua route di dalam group ini harus login dulu 114 Route::group(attributes: ['prefix' => 'barang', 'middleware' => 'authorize:ADM,MNG'], routes: function(): void{ 115 Route::get(uri: '/', action: [BarangController::class, 'index']); 116 Route::post(uri: '/list', action: [BarangController::class, 'list']); 117 Route::get(uri: '/create', action: [BarangController::class, 'create']); 118 Route::post(uri: '/', action: [BarangController::class, 'store']); 119 Route::get(uri: '/create_ajax', action: [BarangController::class, 'create_ajax']); 120 Route::post(uri: '/ajax', action: [BarangController::class, 'store_ajax']); 121 Route::get(uri: '{id}', action: [BarangController::class, 'show']); 122 Route::get(uri: '{id}/edit', action: [BarangController::class, 'edit']); 123 Route::put(uri: '{id}', action: [BarangController::class, 'update']); 124 Route::delete(uri: '{id}', action: [BarangController::class, 'destroy']); 125 }); 126 }); 127 // }); 128 129 >> The closing >> tag should be omitted from files containing only PHP.</pre>
4	<p>Sekarang kita sudah bisa memberikan hak akses menu/route ke beberapa level user</p> 

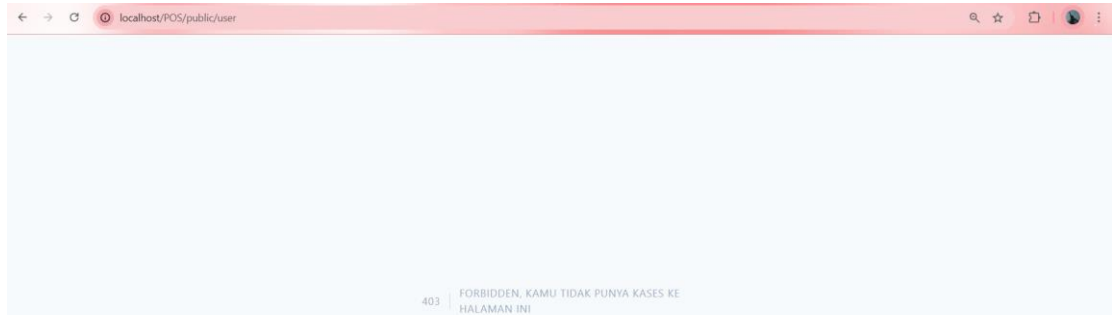


Tugas-3 Implementasi Multi-Level Authorization

Langkah	Jawaban/Deskripsi
1	Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2	Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3	Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User
4	<p>Untuk mengimplementasikan multi-level authorization untuk semua Level/Jenis User dan Menu-menu yang sesuai dengan Level/Jenis User saya melakukan modifikasi pada file web.php seperti berikut.</p> <pre>routes > web.php > Closure > Closure > Closure 14 Route::pattern('id', pattern: '[0-9]+'); //artinya ketika ada parameter (id), maka harus berupa angka 15 16 Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login'); 17 Route::post(uri: 'login', action: [AuthController::class, 'postlogin']); 18 Route::get(uri: 'logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth'); 19 20 Route::middleware(middleware: ['auth'])->group(callback: function () { //semua route di dalam group ini harus login dulu 21 22 Route::get(uri: '/', action: [WelcomeController::class, 'index']); 23 24 Route::middleware(middleware: ['auth'])->group(callback: function() { void{ 25 26 Route::get(uri: '/', action: [WelcomeController::class, 'index']); 27 28 Route::middleware(middleware: ['authorize:ADM'])->group(callback: function() { void{ 29 Route::get(uri: '/user', action: [UserController::class, 'index']); // menampilkan halaman user 30 Route::post(uri: '/user/list', action: [UserController::class, 'list']); // menampilkan data user dalam bentuk json datatables 31 Route::get(uri: '/user/create', action: [UserController::class, 'create']); // menampilkan halaman tambah user 32 Route::get(uri: '/user/create_ajax', action: [UserController::class, 'create_ajax']); //Menampilkan halaman form tambah user Ajax 33 Route::post(uri: '/user/ajax', action: [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax 34 Route::post(uri: '/user', action: [UserController::class, 'store']); // menyimpan data user baru 35 Route::get(uri: '/user/{id}', action: [UserController::class, 'show']); // menampilkan detail user 36 Route::get(uri: '/user/{id}/edit', action: [UserController::class, 'edit']); // menampilkan halaman form edit user 37 Route::put(uri: '/user/{id}', action: [UserController::class, 'update']); // menyimpan perubahan data user 38 Route::get(uri: '/user/{id}/edit_ajax', action: [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user ajax 39 Route::put(uri: '/user/{id}/update_ajax', action: [UserController::class, 'update_ajax']); // menyimpan halaman form edit user ajax 40 Route::get(uri: '/user/{id}/delete_ajax', action: [UserController::class, 'confirm_ajax']); //ambil form confirm delete user ajax 41 Route::delete(uri: '/user/{id}/delete_ajax', action: [UserController::class, 'delete_ajax']); //hapus data user 42 Route::delete(uri: '/user/{id}', action: [UserController::class, 'destroy']); //menghapus data user 43 44 }); 45 }); 46 });</pre> <p>Untuk akses penuh pada menu user saya berikan kepada admin. Sehingga hanya admin nantinya yang dapat mengakses semua data user.</p> <ul style="list-style-type: none">- Berikut tampilan jika saya mengakses data user menggunakan akun admin



- Berikut tampilan jika saya mengakses data user menggunakan akun selain admin(Pelanggan)



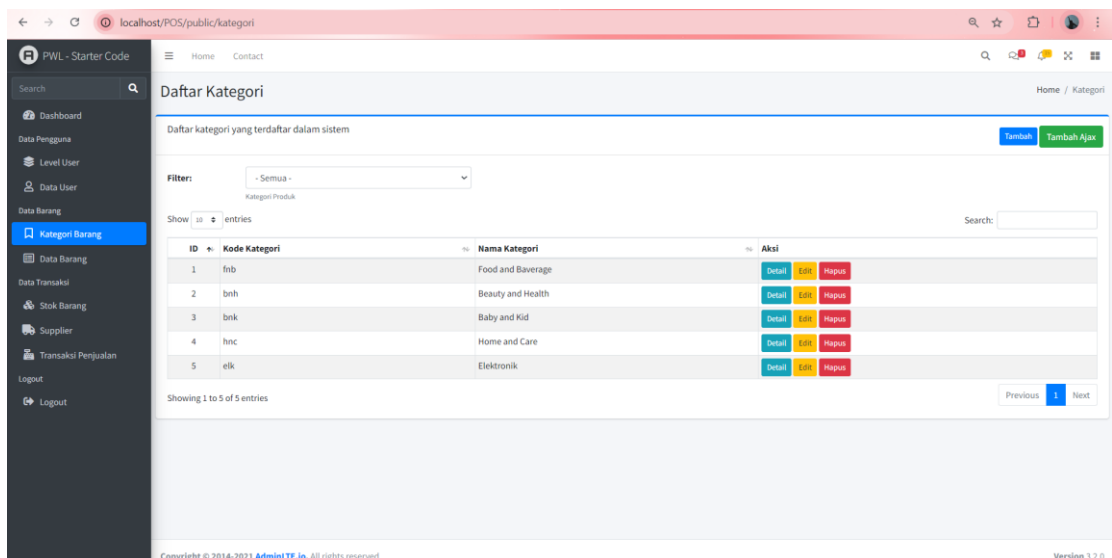
5

Hak akses Kategori

```
81 // LABEL KATEGORI
82 Route::middleware(middleware: ['authorize:ADM,MNG,STF,CUS'])->group(callback: function(): void{
83     Route::get(uri: '/kategori', action: [KategoriController::class, 'index']); //menampilkan halaman awal level
84     Route::post(uri: '/kategori/list', action: [KategoriController::class, 'list']); //menampilkan data Kategori dalam bentuk json
85     Route::get(uri: '/kategori/create', action: [KategoriController::class, 'create']); // menampilkan halaman form tambah Kategori
86     Route::get(uri: '/kategori/create_ajax', action: [KategoriController::class, 'create_ajax']); //Menampilkan halaman form tambah user Ajax
87     Route::post(uri: '/kategori/ajax', action: [KategoriController::class, 'store_ajax']); // Menyimpan data user baru Ajax
88     Route::post(uri: '/kategori', action: [KategoriController::class, 'store']); // menyimpan data Kategori baru
89     Route::get(uri: '/kategori/{id}/edit', action: [KategoriController::class, 'edit']); // menampilkan halaman form edit Kategori
90     Route::put(uri: '/kategori/{id}', action: [KategoriController::class, 'update']); // menyimpan perubahan data Kategori
91     Route::get(uri: '/kategori/{id}/edit_ajax', action: [KategoriController::class, 'edit_ajax']); //menampilkan halaman form edit user ajax
92     Route::put(uri: '/kategori/{id}/update_ajax', action: [KategoriController::class, 'update_ajax']); //menyimpan halaman form edit user ajax
93     Route::get(uri: '/kategori/{id}/delete_ajax', action: [KategoriController::class, 'confirm_ajax']); //tampil form confirm delete user ajax
94     Route::delete(uri: '/kategori/{id}/delete_ajax', action: [KategoriController::class, 'delete_ajax']); //hapus data user
95     Route::delete(uri: '/kategori/{id}', action: [KategoriController::class, 'destroy']); // menghapus data Kategori
96 });
```

Saya memberikan hak akses stok pada admin, staff, customer dan manager sehingga semua jenis user yang saya sebutkan tersebut dapat diakses semua.

- Berikut tampilan jika saya mengakses data kategori menggunakan akun admin





6

Hak Akses Supplier

```
98 // LABEL SUPPLIER
99 Route::middleware('authorize:ADM,MWG')->group(callback: function(): void{
100     Route::get(uri: '/supplier', action: [SupplierController::class, 'index']); //menampilkan halaman awal level
101     Route::post(uri: '/supplier/list', action: [SupplierController::class, 'list']); //menampilkan data Supplier dalam bentuk json
102     Route::get(uri: '/supplier/create', action: [SupplierController::class, 'create']); // menampilkan halaman form tambah Supplier
103     Route::post(uri: '/supplier/create_ajax', action: [SupplierController::class, 'create_ajax']); // menampilkan halaman form tambah supplier Ajax
104     Route::post(uri: '/supplier/ajax', action: [SupplierController::class, 'store_ajax']); // menyimpan data supplier baru Ajax
105     Route::put(uri: '/supplier', action: [SupplierController::class, 'store']); // menyimpan data Supplier baru
106     Route::get(uri: '/supplier/{id}/edit', action: [SupplierController::class, 'edit']); // menampilkan halaman form edit Supplier
107     Route::put(uri: '/supplier/{id}', action: [SupplierController::class, 'update']); // menyimpan perubahan data Supplier
108     Route::get(uri: '/supplier/{id}/edit_ajax', action: [SupplierController::class, 'edit_ajax']); //menampilkan halaman form edit user ajax
109     Route::put(uri: '/supplier/{id}/update_ajax', action: [SupplierController::class, 'update_ajax']); // menyimpan perubahan data supplier Ajax
110     Route::get(uri: '/supplier/{id}/delete_ajax', action: [SupplierController::class, 'confirm_ajax']); // untuk tampilan form confirm delete supplier Ajax
111     Route::delete(uri: '/supplier/{id}/delete_ajax', action: [SupplierController::class, 'delete_ajax']); // untuk hapus data supplier Ajax
112     Route::delete(uri: '/supplier/{id}', action: [SupplierController::class, 'destroy']); // menghapus data Supplier
113 });
```

Hanya admin dan manager yang dapat mengakses data Supplier

- Berikut tampilan jika saya mengakses data kategori menggunakan akun admin

ID	Kode Supplier	Nama Supplier	Alamat Supplier	Aksi
1	yol	Yolanda Supplier	Jln Hamidrusdi No.102	Detail Edit Hapus
2	snd	Sendi Supplier	Jln Buring No.54	Detail Edit Hapus
3	lly	Liya Supplier	Jln Tuban no 65	Detail Edit Hapus
4	nels	Nela Supplier	Jln yerusalem No.4	Detail Edit Hapus

- Berikut tampilan jika saya mengakses data user menggunakan akun selain admin(Pelanggan)

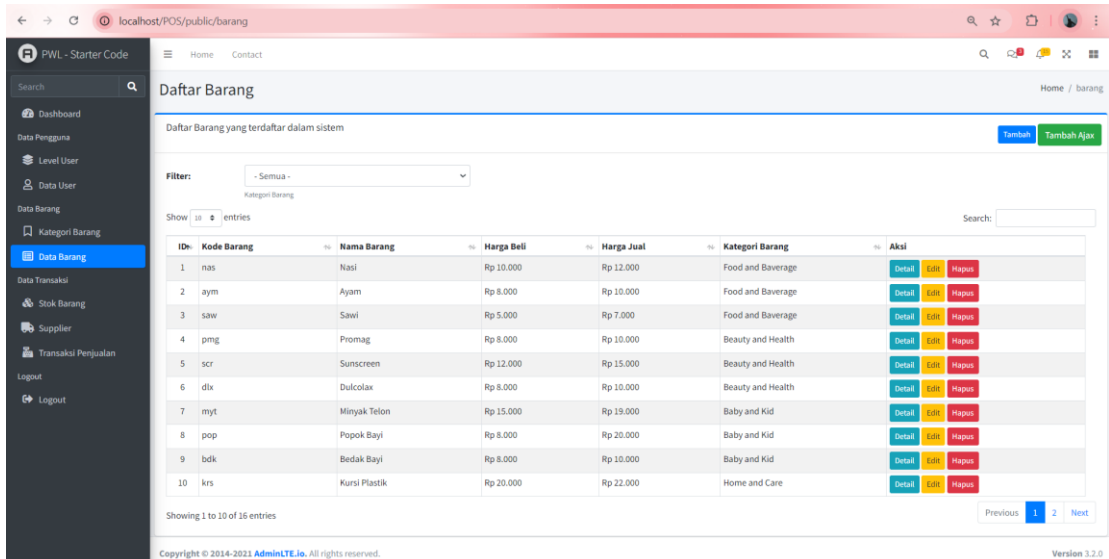


7

Hak Akses Stok Barang

```
115 // TABEL BARANG
116 Route::middleware(['authorize:ADM,MNG,STF,CUS'])->group(callback: function() { void{
117 Route::get(uri: '/barang', action: [BarangController::class, 'index']); //menampilkan halaman awal leevil
118 Route::post(uri: '/barang/list',action: [BarangController::class,'list']); //menampilkan data Barang dalam bentuk json
119 Route::get(uri: '/barang/create', action: [BarangController::class, 'create']); // menampilkan halaman form tambah Barang
120 Route::post(uri: '/barang/create_ajax', action: [BarangController::class, 'create_ajax']); // menampilkan halaman form tambah barang Ajax
121 Route::post(uri: '/barang/ajax', action: [BarangController::class, 'store_ajax']); // menyimpan data barang baru Ajax
122 Route::post(uri: '/barang', action: [BarangController::class, 'store']); // menyimpan data Barang baru
123 Route::get(uri: '/barang/id/edit_ajax', action: [BarangController::class, 'edit_ajax']); // menampilkan halaman form edit barang Ajax
124 Route::get(uri: '/barang/id/edit', action: [BarangController::class, 'edit']); // menampilkan halaman form edit Barang
125 Route::put(uri: '/barang/id', action: [BarangController::class, 'update']); // menyimpan perubahan data Barang
126 Route::put(uri: '/barang/id/update_ajax', action: [BarangController::class, 'update_ajax']); // menyimpan perubahan data barang Ajax
127 Route::get(uri: '/barang/id/delete_ajax', action: [BarangController::class, 'confirm_ajax']); // untuk tampilan form confirm delete barang Ajax
128 Route::delete(uri: '/barang/id/delete_ajax', action: [BarangController::class, 'delete_ajax']); // untuk hapus data barang Ajax
129 Route::delete(uri: '/barang/id', action: [BarangController::class, 'destroy']); // menghapus data Supplier
130 });
```

Saya memberikan hak akses stok pada admin, staff, customer dan manager sehingga semua jenis user yang saya sebutkan tersebut dapat diakses semua.



8

Submit kode untuk impementasi Authorization pada repository github kalian.

Tugas-4 Implementasi Form Registrasi User

Langkah	Jawaban/Deskripsi
1	Silahkan implementasikan form untuk registrasi user.
2	Screenshot hasil yang kalian kerjakan
3	Menambahkan fungsi Register dan postRegister pada file AuthController.php seperti di bawah ini



	<pre>app > Http > Controllers > AuthController.php > PHP Intelephense > AuthController > postRegister 11 class AuthController extends Controller 1 reference 0 overrides 49 public function register(): mixed View 50 { 51 \$level = LevelModel::select(columns: 'level_id', 'level_nama')->get(); 52 return view(view: 'auth.register')->with(key: 'level', value: \$level); 53 } 0 references 0 overrides 54 public function postRegister(Request \$request): JsonResponse mixed RedirectResponse 55 { 56 if (\$request->ajax() \$request->wantsJson()) { 57 \$rules = [58 'level_id' => 'required integer', 59 'username' => 'required string min:3 unique:m_user,username', 60 'nama' => 'required string max:100', 61 'password' => 'required min:5' 62]; 63 \$validator = Validator::make(data: \$request->all(), rules: \$rules); 64 if (\$validator->fails()) { 65 return response()->json(data: [66 'status' => false, 67 'message' => 'Validasi Gagal', 68 'msgField' => \$validator->errors(), 69]); 70 } 71 // Hash password sebelum disimpan 72 \$data = \$request->all(); 73 \$data['password'] = Hash::make(value: \$request->password); 74 // Simpan data user 75 userModel::create(attributes: \$data); 76 return response()->json(data: [77 'status' => true, 78 'message' => 'Data user berhasil disimpan', 79 'redirect' => url(path: 'login') // Redirect ke halaman login 80]); 81 } 82 // Jika bukan AJAX, arahkan ke halaman login 83 return redirect(to: 'login')->with(key: 'success', value: 'Registrasi berhasil!'); 84 } 85 }</pre>
4	<p>Kemudian tambahkan button Registrasi pada halaman login, dengan menambahkan inputan berikut pada file login.blade.php</p> <pre>resources > views > auth > login.blade.php > html > body.hold-transition.login-page > script > <function> > submitHandler 2 <html lang="en"> 19 <body class="hold-transition login-page"> 20 <div class="login-box"> 22 <div class="card card-outline card-primary"> 26 <div class="card-body"> 63 <div class=""> 64 Registrasi 65 </div> 66 </div> 67 <!-- /.card-body --> 68 </div> 69 <!-- /.card --> 70 </div> 71 <!-- /.login-box --></pre>
5	<p>Modifikasi file header.blade.php dengan tambahkan kode berikut</p>



	<pre>resources > views > layouts > header.blade.php > script > <function> > <function> 1 <nav class="main-header navbar navbar-expand navbar-white navbar-light"> 16 <ul class="navbar-nav ml-auto"> 129 <li class="nav-item"> 130 131 <i class="fas fa-sign-out-alt" ></i> 132 133 134 135 </nav> 136 137 <script> 138 document.getElementById('logout-link').addEventListener('click', function (e) { 139 e.preventDefault(); // Prevent the default link behavior 140 141 // Trigger SweetAlert2 confirmation dialog 142 Swal.fire({ 143 title: 'Apakah yakin ingin keluar?', 144 text: "Session anda akan berakhir", 145 icon: 'warning', 146 showCancelButton: true, 147 confirmButtonColor: '#3085d6', 148 cancelButtonColor: '#d333', 149 confirmButtonText: 'Log Out', 150 cancelButtonText: 'Batal' 151 }).then((result) => { 152 if (result.isConfirmed) { 153 // If user confirms, redirect to the logout URL 154 window.location.href = "{{ url('logout') }}"; 155 } 156 }); 157 }); 158 </script></pre>
6	Kemudian buat file <code>register.blade.php</code> dalam folder <code>Auth</code> untuk menampilkan halaman registrasi user dengan isi seperti berikut.

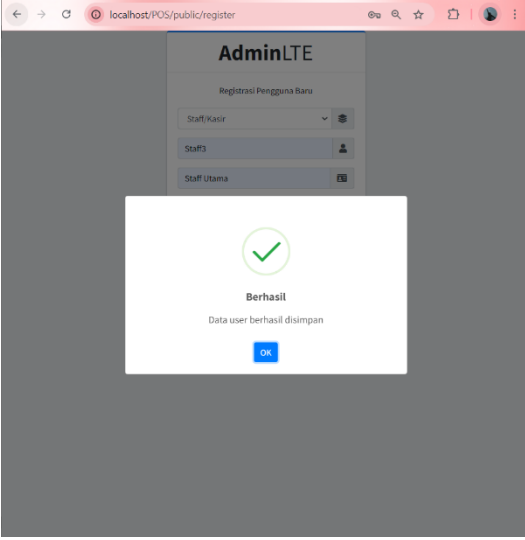
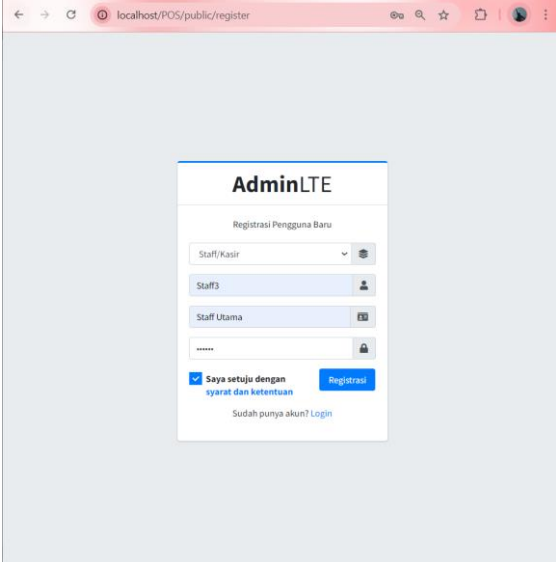


```
resources > views > register > register.blade.php > hml > body.hold-transition.register-page > div.register-box > div.card.card-outline.card-primary > div.card-body > form#form-register
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Register</title>
8
9   <!-- Google Font: Source Sans Pro -->
10  <link rel="stylesheet"
11    href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
12  <!-- Font Awesome -->
13  <link rel="stylesheet" href="adminlte/plugins/fontawesome-free/css/all.min.css">
14  <!-- iCheck bootstrap -->
15  <link rel="stylesheet" href="adminlte/plugins/ichex-bootstrap/ichex-bootstrap.min.css">
16
17  <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
18
19  <!-- Theme style -->
20  <link rel="stylesheet" href="adminlte/dist/css/adminlte.min.css">
21 </head>
22
23 <body class="hold-transition register-page">
24   <div class="register-box">
25     <div class="card card-outline card-primary">
26       <div class="card-header text-center">
27         <a href="../../index2.html" class="h1">Register</a>
28       </div>
29       <div class="card-body">
30         <p class="login-box-msg">Register akun baru</p>
31         <form action="{{ url('register') }}" method="POST" id="form-register">
32           @csrf
33           <div class="input-group mb-3">
34             <select class="form-control" id="level_id" name="level_id" required>
35               <option value="">- Pilih Level / Role -</option>
36               @foreach ($level as $item)
37                 <option value="{{ $item->level_id }}">{{ $item->level_nama }}</option>
38               @endforeach
39             </select>
40             <div class="input-group-append">
41               <div class="input-group-text">
42                 <span class="fas fa-level-down-alt"></span>
43               </div>
44             </div>
45             @error('level_id')
46               <small class="form-text text-danger">{{ $message }}</small>
47             @enderror
48           </div>
49           <div class="input-group mb-3">
50             <input id="nama" name="nama" type="text" class="form-control" placeholder="Full Name" required>
51             <div class="input-group-append">
52               <div class="input-group-text">
53                 <span class="fas fa-user-tag"></span>
54               </div>
55             </div>
56           </div>
57           <div class="input-group mb-3">
58             <input id="username" name="username" type="text" class="form-control" placeholder="Username" required>
59             <div class="input-group-append">
60               <div class="input-group-text">
61                 <span class="fas fa-envelope"></span>
62               </div>
63             </div>
64           </div>
65           <div class="input-group mb-3">
66             <input id="password" name="password" type="password" class="form-control" placeholder="Password" required>
67             <div class="input-group-append">
68               <div class="input-group-text">
69                 <span class="fas fa-lock"></span>
70               </div>
71             </div>
72           </div>
73           <div class="row">
74             <div class="col-8">
75
76             </div>
77             <div class="col-4">
78               <button type="submit" class="btn btn-primary btn-block">Register</button>
79             </div>
80           </div>
81           </div>
82         </div>
```



```
resources > views > register > register.blade.php > html > body.hold-transition.register-page > div.register-box > div.card.card-outline.card-primary > div.card-body > form#form-register
2  <html lang="en">
23 <body class="hold-transition register-page">
24   <div class="register-box">
25     <div class="card card-outline card-primary">
26       <div class="card-body">
27         <form>
28           <a href="{{url('login')}}" class="text-center">Sudah punya akun?</a>
29         </div>
30       </div>
31     </div>
32   </div>
33   <!-- jQuery -->
34   <script src="adminlte/plugins/jquery/jquery.min.js"></script>
35   <!-- Bootstrap 4 -->
36   <script src="adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
37   <!-- AdminLTE App -->
38   <script src="adminlte/dist/js/adminlte.min.js"></script>
39   <script src="{{ asset('adminlte/plugins/jquery-validation/jquery.validate.min.js') }}"></script>
40   <script src="{{ asset('adminlte/plugins/sweetalert2/sweetalert2.min.js') }}"></script>
41   <script>
42     $(document).ready(function() {
43       $("#form-register").validate({
44         rules: {
45           username: {
46             required: true,
47             minlength: 4,
48             maxlength: 20
49           },
50           password: {
51             required: true,
52             minlength: 8,
53             maxlength: 20
54           },
55           level_id: {
56             required: true,
57           },
58           nama: {
59             required: true,
60             minlength: 3,
61             maxlength: 100
62           }
63         },
64         submitHandler: function(form) {
65           $.ajax({
66             url: form.action,
67             type: form.method,
68             data: $(form).serialize(),
69             success: function(response) {
70               console.log(response)
71               if (response.status) { // jika sukses
72                 Swal.fire({
73                   icon: 'success',
74                   title: 'Register Berhasil',
75                   text: response.message,
76                 }).then(function() {
77                   window.location = response.redirect;
78                 });
79               } else { // jika error
80                 $(".error-text").text('');
81                 $.each(response.msgField, function(prefix, val) {
82                   $(".error-" + prefix).text(val[0]);
83                 });
84                 Swal.fire({
85                   icon: 'error',
86                   title: 'Terjadi Kesalahan',
87                   text: response.message
88                 });
89               }
90             }
91           });
92           return false;
93         },
94         errorElement: 'span',
95         errorPlacement: function(error, element) {
96           error.addClass('invalid-feedback');
97         }
98       });
99     }
100   </script>
101   </body>
102 </html>
```

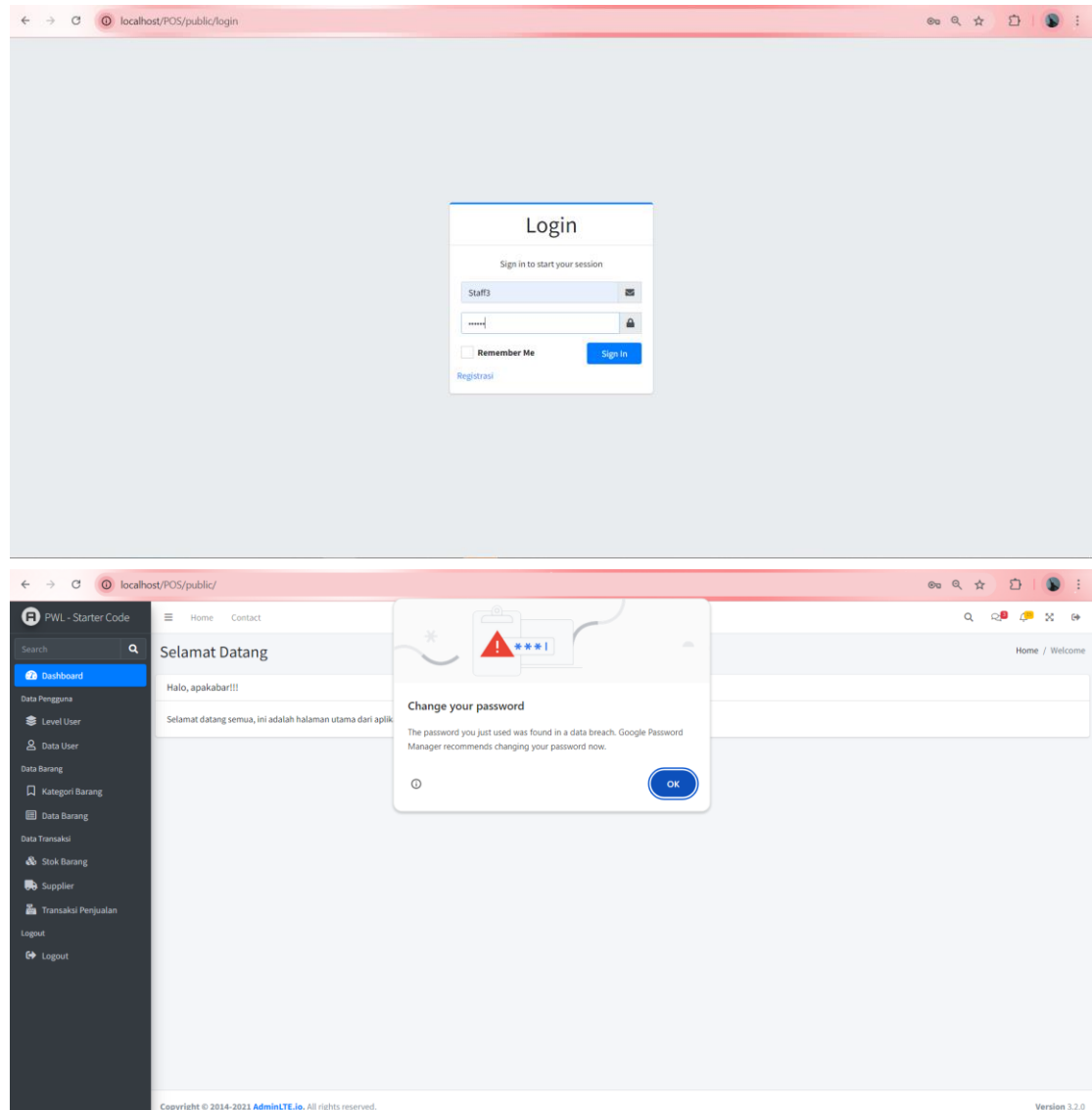


	<pre>resources > views > register > register.blade.php > html > body.hold-transition.register-page > script > <function> 2 <html lang="en"> 23 <body class="hold-transition register-page"> 103 <script> 104 \$(document).ready(function() { 156 errorElement: 'span', 157 errorPlacement: function(error, element) { 158 error.addClass('invalid-feedback'); 159 element.closest('.input-group').append(error); 160 }, 161 highlight: function(element, errorClass, validClass) { 162 \$(element).addClass('is-invalid'); 163 }, 164 unhighlight: function(element, errorClass, validClass) { 165 \$(element).removeClass('is-invalid'); 166 } 167 }); 168 }); 169 </script> 170 </body> 171 172 </html></pre>
7	<p>Maka tambahkan kode di bawah ini pada file web .php untuk dapat mengakses menu register</p> <pre>routes > web.php > ... 11 use App\Http\Controllers\AuthController; 12 use Illuminate\Foundation\Auth; 13 14 Route::pattern(key: 'id', pattern: '[0-9]+'); //artinya ketika ada parameter (id), maka harus berupa angka 15 16 Route::get(uri: 'login', action: [AuthController::class, 'login'])->name(name: 'login'); 17 Route::post(uri: 'login', action: [AuthController::class, 'postlogin']); 18 Route::get(uri: 'register', action: [AuthController::class, 'register']); 19 Route::post(uri: 'register', action: [AuthController::class, 'store']); 20 Route::get(uri: 'logout', action: [AuthController::class, 'logout'])->middleware(middleware: 'auth');</pre>
8	<p>Setelah itu, maka halaman register dapat diakses seperti ini. Jika data registrasi user yang dimasukan sudah sesuai ketentuan maka akan terdapat pop up yang menyatakan bahwa Registrasi Berhasil dan Data user berhasil di simpan</p> <div></div>



9

Kemudian user terdaftar akan terarah ke halaman login untuk melakukan login akun yang baru di daftarkan ke dalam sistem, jika berhasil login maka tampilannya akan seperti gambar di bawah ini.



10

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