membuat form persetujuan dengan 2 role yang berurutan ppk dahulu aprove, baru pptk bisa approve di laravel, jika keduanya sudah approve baru setuju

**ChatGPT said:**

ChatGPT

Untuk membuat form persetujuan dengan dua role yang berurutan (PPK dan PPTK) di Laravel, di mana PPK harus menyetujui terlebih dahulu, baru PPTK bisa memberikan persetujuan, dan setelah keduanya menyetujui baru status bisa dianggap "setuju", Anda bisa mengikuti langkah-langkah berikut.

**1. Struktur Database**

Anda perlu memiliki tabel yang bisa menyimpan status persetujuan dari PPK dan PPTK. Misalnya, Anda dapat menambahkan tabel approvals dengan struktur seperti berikut:

**Migrasi untuk Tabel approvals**

php

Copy code

php artisan make:migration create\_approvals\_table --create=approvals

Di dalam file migrasi (database/migrations/xxxx\_xx\_xx\_create\_approvals\_table.php), buat struktur tabel seperti berikut:

php

Copy code

public function up()

{

Schema::create('approvals', function (Blueprint $table) {

$table->id();

$table->foreignId('user\_id')->constrained()->onDelete('cascade'); // ID user yang menyetujui

$table->enum('role', ['ppk', 'pptk']); // Role (PPK atau PPTK)

$table->boolean('is\_approved')->default(false); // Status persetujuan (false = belum disetujui, true = disetujui)

$table->timestamps();

});

}

Setelah itu jalankan migrasi:

bash

Copy code

php artisan migrate

**2. Model Approval**

Buat model untuk tabel approvals.

php

Copy code

php artisan make:model Approval

Di dalam model (app/Models/Approval.php), tambahkan relasi dan logika yang dibutuhkan:

php

Copy code

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Approval extends Model

{

use HasFactory;

protected $fillable = ['user\_id', 'role', 'is\_approved'];

// Relasi ke User (PPK atau PPTK)

public function user()

{

return $this->belongsTo(User::class);

}

}

**3. Controller untuk Mengelola Persetujuan**

Buat controller untuk menangani logika persetujuan oleh PPK dan PPTK.

php

Copy code

php artisan make:controller ApprovalController

Di dalam controller (app/Http/Controllers/ApprovalController.php), tambahkan logika untuk persetujuan.

php

Copy code

namespace App\Http\Controllers;

use App\Models\Approval;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class ApprovalController extends Controller

{

public function index()

{

// Tampilkan form persetujuan berdasarkan role user

$approvals = Approval::where('user\_id', Auth::id())->get();

return view('approvals.index', compact('approvals'));

}

public function approve($id)

{

$approval = Approval::findOrFail($id);

// Pastikan hanya PPK yang bisa approve pertama kali

if (Auth::user()->role != 'ppk' && $approval->role == 'ppk') {

return redirect()->back()->with('error', 'Hanya PPK yang bisa menyetujui pertama.');

}

// Jika PPK sudah approve, maka PPTK bisa approve

if ($approval->role == 'ppk' && $approval->is\_approved == true) {

// Jika PPK sudah setujui, berikan kesempatan untuk PPTK approve

$approval->is\_approved = true;

$approval->save();

// Setelah keduanya approve, update status persetujuan akhir

$this->updateStatusSetuju($approval);

}

// Tandai approval sebagai approved

$approval->is\_approved = true;

$approval->save();

return redirect()->route('approvals.index')->with('success', 'Persetujuan berhasil.');

}

private function updateStatusSetuju(Approval $approval)

{

// Cek apakah keduanya sudah approve

$ppkApproval = Approval::where('role', 'ppk')

->where('user\_id', $approval->user\_id)

->first();

$pptkApproval = Approval::where('role', 'pptk')

->where('user\_id', $approval->user\_id)

->first();

if ($ppkApproval && $pptkApproval && $ppkApproval->is\_approved && $pptkApproval->is\_approved) {

// Status persetujuan telah disetujui kedua pihak

$approval->status = 'setuju';

$approval->save();

}

}

}

**4. Routing**

Tambahkan routing untuk controller di file routes/web.php.

php

Copy code

use App\Http\Controllers\ApprovalController;

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

Route::get('/approvals', [ApprovalController::class, 'index'])->name('approvals.index');

Route::post('/approvals/approve/{id}', [ApprovalController::class, 'approve'])->name('approvals.approve');

});

**5. View untuk Menampilkan Form Persetujuan**

Buat view untuk menampilkan form persetujuan sesuai role yang sedang login. Buat file resources/views/approvals/index.blade.php.

php

Copy code

@extends('layouts.app')

@section('content')

<div class="container">

<h2>Persetujuan</h2>

@foreach ($approvals as $approval)

<div>

<h4>{{ $approval->role == 'ppk' ? 'PPK' : 'PPTK' }}: {{ $approval->is\_approved ? 'Disetujui' : 'Belum Disetujui' }}</h4>

@if (!$approval->is\_approved)

<form action="{{ route('approvals.approve', $approval->id) }}" method="POST">

@csrf

<button type="submit" class="btn btn-primary">Setujui</button>

</form>

@endif

</div>

@endforeach

</div>

@endsection

**6. Sistem Keamanan dan Middleware**

Pastikan bahwa setiap pengguna hanya bisa mengakses atau melakukan tindakan berdasarkan peran mereka. Misalnya, hanya PPK yang bisa menyetujui pertama, dan hanya PPTK yang bisa menyetujui setelah PPK.

Anda dapat menggunakan middleware untuk memverifikasi role pengguna. Sebagai contoh:

php

Copy code

// Middleware untuk memverifikasi peran

public function handle($request, Closure $next, $role)

{

if (Auth::check() && Auth::user()->role == $role) {

return $next($request);

}

return redirect()->route('home')->with('error', 'Access Denied');

}

**7. Testing dan Validasi**

Pastikan untuk menguji seluruh alur persetujuan:

* PPK harus memberikan persetujuan terlebih dahulu.
* Setelah PPK menyetujui, PPTK baru bisa memberikan persetujuan.
* Jika keduanya sudah menyetujui, status bisa diubah menjadi "setuju".

Dengan langkah-langkah ini, Anda akan memiliki sistem persetujuan dengan dua peran berurutan di Laravel.

4o

**You said:**

tambahkan polimorfis untuk table aprovable

**ChatGPT said:**

ChatGPT

Untuk menambahkan fitur polimorfisme ke tabel persetujuan (approval) di Laravel, kita dapat memanfaatkan hubungan polimorfis yang memungkinkan kita untuk menggunakan satu tabel untuk menyimpan hubungan ke berbagai model yang berbeda. Ini sangat berguna jika kita ingin agar persetujuan (approval) tidak hanya berlaku untuk satu entitas tertentu (misalnya, proyek), tetapi juga untuk berbagai entitas yang berbeda (misalnya, tugas, permintaan pengadaan, dll).

Berikut adalah langkah-langkah untuk menambahkan fungsionalitas polimorfis pada tabel approvals.

**1. Modifikasi Migrasi approvals untuk Mendukung Polimorfisme**

Tabel approvals harus dimodifikasi agar bisa menyimpan relasi polimorfis. Untuk itu, kita akan menambahkan dua kolom: approvable\_id dan approvable\_type yang digunakan untuk menyimpan referensi ke model yang berbeda.

Buat migrasi baru untuk menambahkan kolom polimorfis:

bash

Copy code

php artisan make:migration add\_approvable\_to\_approvals\_table --table=approvals

Kemudian, dalam file migrasi tersebut, tambahkan kolom approvable\_id dan approvable\_type:

php

Copy code

public function up()

{

Schema::table('approvals', function (Blueprint $table) {

$table->unsignedBigInteger('approvable\_id'); // ID dari model yang disetujui

$table->string('approvable\_type'); // Nama model yang disetujui

});

}

Jalankan migrasi:

bash

Copy code

php artisan migrate

**2. Modifikasi Model Approval**

Selanjutnya, kita akan mengubah model Approval untuk mendukung relasi polimorfis.

Di dalam model Approval (app/Models/Approval.php), tambahkan relasi morphTo() untuk menunjuk ke model yang berbeda-beda.

php

Copy code

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Approval extends Model

{

use HasFactory;

protected $fillable = ['user\_id', 'role', 'is\_approved', 'approvable\_id', 'approvable\_type'];

// Relasi ke User (PPK atau PPTK)

public function user()

{

return $this->belongsTo(User::class);

}

// Relasi polimorfis

public function approvable()

{

return $this->morphTo();

}

}

**3. Menambahkan Relasi Polimorfis pada Model yang Dapat Disetujui**

Misalkan Anda memiliki model Project, Task, atau model lain yang dapat disetujui, Anda perlu menambahkan relasi polimorfis di model-model tersebut. Sebagai contoh, untuk model Project:

php

Copy code

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Project extends Model

{

use HasFactory;

// Relasi ke Approval (polimorfis)

public function approvals()

{

return $this->morphMany(Approval::class, 'approvable');

}

}

Demikian pula, jika Anda memiliki model lain, misalnya Task, Anda dapat menambahkan relasi yang sama:

php

Copy code

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Task extends Model

{

use HasFactory;

// Relasi ke Approval (polimorfis)

public function approvals()

{

return $this->morphMany(Approval::class, 'approvable');

}

}

**4. Menggunakan Polimorfisme di Controller**

Sekarang kita dapat menggunakan hubungan polimorfis ini di controller untuk menangani approval pada entitas yang berbeda. Misalnya, jika Anda ingin memberikan approval pada sebuah Project atau Task, Anda cukup menetapkan approvable ke model yang sesuai.

Misalnya, di dalam ApprovalController, jika Anda ingin menambahkan approval untuk sebuah Project, Anda bisa menulis kode seperti berikut:

php

Copy code

namespace App\Http\Controllers;

use App\Models\Approval;

use App\Models\Project;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class ApprovalController extends Controller

{

public function approve(Request $request, $approvableId, $approvableType)

{

// Pastikan hanya PPK yang bisa menyetujui pertama kali

$approval = Approval::where('approvable\_id', $approvableId)

->where('approvable\_type', $approvableType)

->where('user\_id', Auth::id())

->first();

if ($approval) {

// Lakukan logika persetujuan

$approval->is\_approved = true;

$approval->save();

// Jika ini adalah persetujuan kedua, pastikan status selesai

$this->updateStatusSetuju($approval);

}

return redirect()->route('approvals.index')->with('success', 'Persetujuan berhasil.');

}

private function updateStatusSetuju(Approval $approval)

{

// Cek apakah keduanya sudah approve

$ppkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'ppk')

->first();

$pptkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'pptk')

->first();

if ($ppkApproval && $pptkApproval && $ppkApproval->is\_approved && $pptkApproval->is\_approved) {

// Status persetujuan telah disetujui kedua pihak

$approvable = $approval->approvable;

$approvable->status = 'setuju'; // Menandakan bahwa proses telah disetujui

$approvable->save();

}

}

}

**5. Menggunakan Form Persetujuan di View**

Di dalam view Anda, pastikan untuk mengirimkan data approvable\_type dan approvable\_id ketika membuat form persetujuan. Misalnya, jika kita ingin persetujuan untuk Project:

php

Copy code

@extends('layouts.app')

@section('content')

<div class="container">

<h2>Persetujuan untuk Proyek</h2>

@foreach ($approvals as $approval)

<div>

<h4>{{ $approval->role == 'ppk' ? 'PPK' : 'PPTK' }}: {{ $approval->is\_approved ? 'Disetujui' : 'Belum Disetujui' }}</h4>

@if (!$approval->is\_approved)

<form action="{{ route('approvals.approve', ['approvableId' => $approvable->id, 'approvableType' => get\_class($approvable)]) }}" method="POST">

@csrf

<button type="submit" class="btn btn-primary">Setujui</button>

</form>

@endif

</div>

@endforeach

</div>

@endsection

Di dalam kode tersebut, kita mengirimkan approvableId dan approvableType yang berfungsi untuk menyatakan model mana yang disetujui (misalnya Project atau Task).

**6. Menyelesaikan Proses Persetujuan**

Setelah keduanya (PPK dan PPTK) menyetujui, Anda dapat melakukan perubahan pada status model yang disetujui, seperti mengubah status proyek menjadi setuju atau status lainnya.

**7. Testing**

Pastikan Anda menguji:

* Penggunaan persetujuan pada model yang berbeda (misalnya Project, Task).
* Alur persetujuan berjalan sesuai dengan urutan yang benar (PPK harus approve dulu, baru PPTK).
* Setelah kedua pihak approve, status model yang disetujui diperbarui dengan benar.

Dengan menggunakan relasi polimorfis, Anda membuat sistem persetujuan yang lebih fleksibel dan dapat digunakan untuk berbagai jenis entitas di aplikasi Anda.

Urutannya pppk, pptk , penanggung jawab final -> selesaikan kontrak

Untuk menyempurnakan sistem persetujuan yang melibatkan dua peran berurutan (PPK dan PPTK) dengan tambahan fitur polimorfisme dan final approval oleh penanggung jawab, kita perlu membuat beberapa penyesuaian pada struktur database dan logika aplikasi. Berikut adalah cara untuk mengimplementasikannya di Laravel.

**1. Struktur Database dengan Polimorfisme**

Tabel approvals sudah kita ubah sebelumnya untuk mendukung polimorfisme. Namun, kita perlu menambahkan peran untuk **penanggung jawab final** dan status akhir untuk kontrak atau entitas yang disetujui.

**Migrasi Tabel approvals**

Buat migrasi untuk menambahkan kolom baru yang mendukung penanggung jawab final (final\_approver) dan kolom status untuk menandai status akhir persetujuan.

bash

Copy code

php artisan make:migration add\_final\_approver\_and\_status\_to\_approvals\_table --table=approvals

Di dalam file migrasi (xxxx\_xx\_xx\_add\_final\_approver\_and\_status\_to\_approvals\_table.php):

php

Copy code

public function up()

{

Schema::table('approvals', function (Blueprint $table) {

$table->unsignedBigInteger('final\_approver\_id')->nullable(); // ID penanggung jawab akhir

$table->enum('status', ['pending', 'approved', 'rejected', 'final\_approved'])->default('pending'); // Status persetujuan

});

}

Jalankan migrasi:

bash

Copy code

php artisan migrate

**2. Model Approval dengan Relasi Polimorfis dan Final Approver**

Tambahkan logika di model Approval untuk mendukung relasi polimorfis dan penanggung jawab final.

php

Copy code

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Approval extends Model

{

use HasFactory;

protected $fillable = ['user\_id', 'role', 'is\_approved', 'approvable\_id', 'approvable\_type', 'final\_approver\_id', 'status'];

// Relasi ke User (PPK atau PPTK)

public function user()

{

return $this->belongsTo(User::class);

}

// Relasi polimorfis

public function approvable()

{

return $this->morphTo();

}

// Relasi ke final approver

public function finalApprover()

{

return $this->belongsTo(User::class, 'final\_approver\_id');

}

}

**3. Controller untuk Mengelola Persetujuan dengan Peran Berurutan**

Update ApprovalController untuk menangani urutan persetujuan oleh PPK, PPTK, dan penanggung jawab final.

php

Copy code

namespace App\Http\Controllers;

use App\Models\Approval;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class ApprovalController extends Controller

{

public function index()

{

// Ambil semua approvals yang terkait dengan model yang dapat disetujui

$approvals = Approval::where('approvable\_id', $approvableId)->get();

return view('approvals.index', compact('approvals'));

}

public function approve(Request $request, $approvableId, $approvableType)

{

// Temukan persetujuan yang sesuai dengan ID dan type

$approval = Approval::where('approvable\_id', $approvableId)

->where('approvable\_type', $approvableType)

->where('user\_id', Auth::id())

->first();

if (!$approval) {

return redirect()->back()->with('error', 'Persetujuan tidak ditemukan.');

}

// PPK menyetujui pertama

if ($approval->role == 'ppk') {

$approval->is\_approved = true;

$approval->save();

}

// PPTK hanya bisa menyetujui setelah PPK

if ($approval->role == 'pptk' && $approval->is\_approved) {

$approval->is\_approved = true;

$approval->save();

}

// Update status jika kedua PPK dan PPTK sudah menyetujui

if ($this->bothApproved($approval)) {

$this->updateFinalApprovalStatus($approval);

}

return redirect()->route('approvals.index')->with('success', 'Persetujuan berhasil.');

}

// Cek apakah PPK dan PPTK sudah menyetujui

private function bothApproved(Approval $approval)

{

$ppkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'ppk')

->first();

$pptkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'pptk')

->first();

return $ppkApproval && $pptkApproval && $ppkApproval->is\_approved && $pptkApproval->is\_approved;

}

// Jika keduanya sudah approve, status bisa diubah

private function updateFinalApprovalStatus(Approval $approval)

{

$approvable = $approval->approvable;

// Jika ada penanggung jawab akhir

$finalApprover = User::find($approval->final\_approver\_id);

if ($finalApprover) {

$approvable->status = 'final\_approved';

$approvable->save();

}

}

}

**4. Route untuk Persetujuan**

Tambahkan route baru untuk pengelolaan persetujuan di file routes/web.php.

php

Copy code

use App\Http\Controllers\ApprovalController;

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

Route::get('/approvals/{approvableId}/{approvableType}', [ApprovalController::class, 'index'])->name('approvals.index');

Route::post('/approvals/approve/{approvableId}/{approvableType}', [ApprovalController::class, 'approve'])->name('approvals.approve');

});

**5. View untuk Menampilkan Form Persetujuan**

Tampilkan form persetujuan sesuai dengan status dan peran pengguna. Misalnya di resources/views/approvals/index.blade.php:

php

Copy code

@extends('layouts.app')

@section('content')

<div class="container">

<h2>Persetujuan untuk {{ $approvable->name }}</h2>

@foreach ($approvals as $approval)

<div>

<h4>{{ $approval->role == 'ppk' ? 'PPK' : ($approval->role == 'pptk' ? 'PPTK' : 'Penanggung Jawab Final') }}: {{ $approval->is\_approved ? 'Disetujui' : 'Belum Disetujui' }}</h4>

@if (!$approval->is\_approved)

<form action="{{ route('approvals.approve', ['approvableId' => $approvable->id, 'approvableType' => get\_class($approvable)]) }}" method="POST">

@csrf

<button type="submit" class="btn btn-primary">Setujui</button>

</form>

@endif

</div>

@endforeach

</div>

@endsection

**6. Penanggung Jawab Final**

Pastikan penanggung jawab final bisa memberikan approval terakhir pada entitas yang telah disetujui oleh PPK dan PPTK.

**7. Testing**

Lakukan pengujian untuk memastikan alur persetujuan bekerja sesuai urutan:

1. PPK memberi persetujuan terlebih dahulu.
2. Setelah PPK menyetujui, PPTK baru bisa memberikan persetujuan.
3. Jika keduanya menyetujui, status entitas akan diubah menjadi final\_approved oleh penanggung jawab final.

Dengan demikian, Anda telah berhasil menambahkan logika persetujuan dengan polimorfisme dan memperkenalkan peran penanggung jawab final di Laravel.

4o

**You said:**

buatkan component livewirenya agar dapat digunakan kembali disemua blade

**ChatGPT said:**

ChatGPT

Untuk membuat komponen Livewire yang dapat digunakan kembali di semua blade untuk menangani persetujuan dengan dua role berurutan (PPK dan PPTK), kita akan membuat komponen Livewire terpisah dan menyertakan logika persetujuan di dalamnya.

Langkah-langkah berikut menjelaskan cara membuat dan menggunakan komponen Livewire untuk sistem persetujuan.

**1. Buat Komponen Livewire**

Pertama, kita buat komponen Livewire yang akan menangani logika persetujuan.

Jalankan perintah artisan untuk membuat komponen Livewire:

bash

Copy code

php artisan make:livewire approval-component

Ini akan menghasilkan dua file:

* app/Http/Livewire/ApprovalComponent.php (class Livewire)
* resources/views/livewire/approval-component.blade.php (view Livewire)

**2. Tulis Logika Komponen Livewire**

Edit file ApprovalComponent.php untuk menambahkan logika persetujuan PPK, PPTK, dan penanggung jawab akhir (final approver).

php

Copy code

namespace App\Http\Livewire;

use App\Models\Approval;

use App\Models\User;

use Livewire\Component;

use Illuminate\Support\Facades\Auth;

class ApprovalComponent extends Component

{

public $approvableId;

public $approvableType;

public $approvals = [];

public $user;

// Konstruktor untuk inisialisasi ID dan Type Approvable

public function mount($approvableId, $approvableType)

{

$this->approvableId = $approvableId;

$this->approvableType = $approvableType;

$this->user = Auth::user();

// Ambil data approvals berdasarkan approvable ID dan Type

$this->loadApprovals();

}

// Memuat data persetujuan

public function loadApprovals()

{

$this->approvals = Approval::where('approvable\_id', $this->approvableId)

->where('approvable\_type', $this->approvableType)

->get();

}

// Fungsi untuk menghandle persetujuan

public function approve($approvalId)

{

$approval = Approval::find($approvalId);

// Validasi peran pengguna (PPK atau PPTK)

if ($approval->role == 'ppk' && $this->user->role != 'ppk') {

session()->flash('error', 'Hanya PPK yang bisa memberikan persetujuan pertama.');

return;

}

if ($approval->role == 'pptk' && $this->user->role != 'pptk') {

session()->flash('error', 'Hanya PPTK yang bisa memberikan persetujuan kedua.');

return;

}

// Update status approval

$approval->is\_approved = true;

$approval->save();

// Cek apakah keduanya sudah approve (PPK dan PPTK)

if ($this->bothApproved($approval)) {

$this->updateFinalApprovalStatus($approval);

}

// Reload data approvals

$this->loadApprovals();

session()->flash('success', 'Persetujuan berhasil.');

}

// Cek apakah PPK dan PPTK sudah memberikan persetujuan

private function bothApproved(Approval $approval)

{

$ppkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'ppk')

->first();

$pptkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'pptk')

->first();

return $ppkApproval && $pptkApproval && $ppkApproval->is\_approved && $pptkApproval->is\_approved;

}

// Update status final approval jika keduanya sudah approve

private function updateFinalApprovalStatus(Approval $approval)

{

// Status persetujuan final

$approvable = $approval->approvable;

$approvable->status = 'final\_approved';

$approvable->save();

}

public function render()

{

return view('livewire.approval-component');

}

}

Penjelasan:

* **mount()**: Digunakan untuk inisialisasi data seperti approvableId, approvableType, dan memuat data persetujuan.
* **loadApprovals()**: Memuat data persetujuan untuk entitas yang sedang diproses.
* **approve()**: Menghandle logika persetujuan berdasarkan peran pengguna (PPK atau PPTK).
* **bothApproved()**: Mengecek apakah PPK dan PPTK sudah memberikan persetujuan.
* **updateFinalApprovalStatus()**: Mengubah status entitas jika sudah disetujui oleh PPK dan PPTK.

**3. Buat View untuk Komponen Livewire**

Sekarang, buka file resources/views/livewire/approval-component.blade.php dan sesuaikan dengan HTML dan Livewire directive.

blade

Copy code

<div>

<h2>Persetujuan untuk {{ $approvableType }}</h2>

@foreach ($approvals as $approval)

<div>

<h4>{{ $approval->role == 'ppk' ? 'PPK' : ($approval->role == 'pptk' ? 'PPTK' : 'Penanggung Jawab Final') }}:

{{ $approval->is\_approved ? 'Disetujui' : 'Belum Disetujui' }}

</h4>

@if (!$approval->is\_approved)

<button wire:click="approve({{ $approval->id }})" class="btn btn-primary">

Setujui

</button>

@endif

</div>

@endforeach

<!-- Flash message -->

@if (session()->has('success'))

<div class="alert alert-success mt-3">

{{ session('success') }}

</div>

@elseif (session()->has('error'))

<div class="alert alert-danger mt-3">

{{ session('error') }}

</div>

@endif

</div>

Penjelasan:

* Menggunakan directive @foreach untuk menampilkan daftar approvals.
* Menggunakan wire:click untuk memanggil fungsi approve() ketika tombol setujui ditekan.
* Menampilkan pesan flash jika persetujuan berhasil atau terjadi kesalahan.

**4. Penggunaan Komponen di Blade**

Sekarang kita dapat menggunakan komponen Livewire ini di blade mana pun. Misalnya, di file resources/views/approvals/index.blade.php:

blade

Copy code

@extends('layouts.app')

@section('content')

<div class="container">

<h2>Persetujuan Proyek</h2>

<!-- Gunakan komponen Livewire -->

@livewire('approval-component', ['approvableId' => $project->id, 'approvableType' => 'App\Models\Project'])

</div>

@endsection

Penjelasan:

* Kita menggunakan komponen Livewire dengan direktif @livewire.
* Kita mengirimkan approvableId dan approvableType ke dalam komponen sebagai parameter.
* approvableType bisa berupa model lain, seperti App\Models\Task, jika diperlukan untuk entitas lain.

**5. Menggunakan Komponen di Semua Blade**

Dengan cara ini, kita bisa menggunakan komponen approval-component di semua blade dengan mengganti approvableId dan approvableType sesuai dengan entitas yang ingin disetujui. Anda bisa menambahkannya di berbagai halaman, seperti detail proyek, tugas, atau entitas lain yang membutuhkan persetujuan.

**6. Tambahkan Livewire ke Layout**

Jika belum, pastikan Anda menambahkan script dan style Livewire di file resources/views/layouts/app.blade.php:

blade

Copy code

<head>

@livewireStyles

</head>

<body>

@yield('content')

@livewireScripts

</body>

**7. Testing**

Sekarang Anda sudah dapat menggunakan komponen persetujuan ini di berbagai blade. Cobalah untuk:

* Memastikan komponen dapat menampilkan data persetujuan berdasarkan entitas yang diberikan.
* Memastikan tombol persetujuan berfungsi sesuai dengan peran pengguna (PPK, PPTK, dan penanggung jawab akhir).
* Memastikan bahwa status entitas berubah setelah persetujuan final.

Dengan menggunakan Livewire, komponen persetujuan ini dapat dipakai kembali di berbagai tempat dalam aplikasi Anda dengan cara yang lebih terstruktur dan reaktif.

4o

**You said:**

kombinasikan dengan permission roles dari spatie

**ChatGPT said:**

ChatGPT

Untuk mengkombinasikan sistem persetujuan menggunakan Livewire dengan **Permission & Role Management** dari **Spatie** di Laravel, kita akan menambahkan pemeriksaan izin dan peran (role) pada komponen persetujuan untuk memastikan bahwa hanya pengguna yang memiliki hak akses yang sesuai (PPK, PPTK, atau penanggung jawab akhir) yang dapat melakukan tindakan persetujuan.

Berikut adalah langkah-langkah untuk mengintegrasikan Spatie dengan komponen persetujuan Livewire yang telah kita buat sebelumnya.

**1. Instalasi dan Konfigurasi Spatie Permission**

Jika belum menginstal paket **Spatie Laravel Permission**, Anda bisa menginstalnya dengan langkah-langkah berikut:

**a. Install Paket Spatie Permission**

Jalankan perintah berikut untuk menginstal paket Spatie Permission:

bash

Copy code

composer require spatie/laravel-permission

**b. Publikasikan File Konfigurasi**

Setelah itu, publikasikan file konfigurasi dan migrasi tabel yang diperlukan:

bash

Copy code

php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"

**c. Jalankan Migrasi**

Jalankan migrasi untuk membuat tabel roles dan permissions:

bash

Copy code

php artisan migrate

**2. Menambahkan Role dan Permission ke User**

Pada model User, pastikan bahwa model User menggunakan trait Spatie\Permission\Traits\HasRoles.

php

Copy code

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable

{

use HasFactory, Notifiable, HasRoles;

// other code...

}

**d. Menambahkan Role dan Permission**

Di file DatabaseSeeder.php, Anda bisa menambahkan peran dan izin untuk setiap pengguna, misalnya PPK, PPTK, dan Penanggung Jawab Akhir:

php

Copy code

use Spatie\Permission\Models\Role;

use Spatie\Permission\Models\Permission;

public function run()

{

// Membuat Role

Role::create(['name' => 'ppk']);

Role::create(['name' => 'pptk']);

Role::create(['name' => 'penanggung-jawab-akhir']);

// Menambahkan Permission

Permission::create(['name' => 'approve.project']);

Permission::create(['name' => 'approve.task']);

// Assign permission ke role

$ppk = Role::findByName('ppk');

$ppk->givePermissionTo('approve.project');

$pptk = Role::findByName('pptk');

$pptk->givePermissionTo('approve.task');

// Assign role ke user

$user = User::find(1);

$user->assignRole('ppk'); // Assign role PPK ke user

}

Jalankan perintah php artisan db:seed untuk menambahkan role dan permission.

**3. Integrasi Spatie dengan Komponen Livewire**

Sekarang kita akan menyesuaikan komponen Livewire yang sudah dibuat sebelumnya agar hanya pengguna dengan **role** dan **permission** yang sesuai yang bisa memberikan persetujuan. Kita akan menggunakan middleware atau pengecekan permission dari Spatie untuk mengontrol akses.

**a. Update Komponen Livewire (ApprovalComponent.php)**

Di dalam komponen Livewire, kita akan memeriksa apakah pengguna memiliki role dan permission yang sesuai.

php

Copy code

namespace App\Http\Livewire;

use App\Models\Approval;

use Livewire\Component;

use Illuminate\Support\Facades\Auth;

use Spatie\Permission\Models\Role;

class ApprovalComponent extends Component

{

public $approvableId;

public $approvableType;

public $approvals = [];

public $user;

public function mount($approvableId, $approvableType)

{

$this->approvableId = $approvableId;

$this->approvableType = $approvableType;

$this->user = Auth::user();

// Pastikan pengguna memiliki role yang sesuai untuk proses persetujuan

$this->authorizeApproval();

// Muat data persetujuan

$this->loadApprovals();

}

// Periksa apakah pengguna memiliki role yang tepat

private function authorizeApproval()

{

if (!$this->user->hasRole('ppk') && !$this->user->hasRole('pptk')) {

abort(403, 'Akses ditolak. Anda tidak memiliki hak untuk memberikan persetujuan.');

}

// Jika role pengguna adalah ppk, mereka hanya bisa menyetujui PPK

if ($this->user->hasRole('ppk') && !$this->user->can('approve.project')) {

abort(403, 'Akses ditolak. Anda tidak memiliki izin untuk menyetujui proyek.');

}

// Jika role pengguna adalah pptk, mereka hanya bisa menyetujui PPTK

if ($this->user->hasRole('pptk') && !$this->user->can('approve.task')) {

abort(403, 'Akses ditolak. Anda tidak memiliki izin untuk menyetujui tugas.');

}

}

// Memuat data persetujuan

public function loadApprovals()

{

$this->approvals = Approval::where('approvable\_id', $this->approvableId)

->where('approvable\_type', $this->approvableType)

->get();

}

// Fungsi untuk menghandle persetujuan

public function approve($approvalId)

{

$approval = Approval::find($approvalId);

// Validasi peran pengguna (PPK atau PPTK)

if ($approval->role == 'ppk' && !$this->user->hasRole('ppk')) {

session()->flash('error', 'Hanya PPK yang bisa memberikan persetujuan pertama.');

return;

}

if ($approval->role == 'pptk' && !$this->user->hasRole('pptk')) {

session()->flash('error', 'Hanya PPTK yang bisa memberikan persetujuan kedua.');

return;

}

// Update status persetujuan

$approval->is\_approved = true;

$approval->save();

// Cek apakah keduanya sudah approve (PPK dan PPTK)

if ($this->bothApproved($approval)) {

$this->updateFinalApprovalStatus($approval);

}

// Reload data approvals

$this->loadApprovals();

session()->flash('success', 'Persetujuan berhasil.');

}

// Cek apakah PPK dan PPTK sudah memberikan persetujuan

private function bothApproved(Approval $approval)

{

$ppkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'ppk')

->first();

$pptkApproval = Approval::where('approvable\_id', $approval->approvable\_id)

->where('approvable\_type', $approval->approvable\_type)

->where('role', 'pptk')

->first();

return $ppkApproval && $pptkApproval && $ppkApproval->is\_approved && $pptkApproval->is\_approved;

}

// Update status final approval jika keduanya sudah approve

private function updateFinalApprovalStatus(Approval $approval)

{

// Status persetujuan final

$approvable = $approval->approvable;

$approvable->status = 'final\_approved';

$approvable->save();

}

public function render()

{

return view('livewire.approval-component');

}

}

Penjelasan:

* **authorizeApproval()**: Fungsi ini memeriksa apakah pengguna memiliki role yang sesuai (PPK atau PPTK) dan apakah mereka memiliki izin untuk menyetujui entitas tertentu (approve.project atau approve.task).
* Jika pengguna tidak memiliki role atau izin yang sesuai, maka akses akan diblokir dengan menggunakan abort(403, 'Akses ditolak...').

**b. Update View Livewire**

Pastikan view approval-component.blade.php dapat menampilkan pesan error jika pengguna tidak memiliki akses untuk menyetujui:

blade

Copy code

<div>

<h2>Persetujuan untuk {{ $approvableType }}</h2>

@foreach ($approvals as $approval)

<div>

<h4>{{ $approval->role == 'ppk' ? 'PPK' : ($approval->role == 'pptk' ? 'PPTK' : 'Penanggung Jawab Final') }}:

{{ $approval->is\_approved ? 'Disetujui' : 'Belum Disetujui' }}

</h4>

@if (!$approval->is\_approved)

<button wire:click="approve({{ $approval->id }})" class="btn btn-primary">

Setujui

</button>

@endif

</div>

@endforeach

<!-- Flash message -->

@if (session()->has('success'))

<div class="alert alert-success mt-3">

{{ session('success') }}

</div>

@elseif (session()->has('error'))

<div class="alert alert-danger mt-3">

{{ session('error') }}

</div>

@endif

</div>

**4. Menggunakan Komponen di Blade**

Sekarang Anda bisa menggunakan komponen persetujuan ini di blade mana pun seperti biasa. Pastikan hanya pengguna dengan **role** dan **permission** yang sesuai yang dapat mengakses dan menyetujui.

blade

Copy code

@extends('layouts.app')

@section('content')

<div class="container">

<h2>Persetujuan Proyek</h2>

<!-- Gunakan komponen Livewire -->

@livewire('approval-component', ['approvableId' => $project->id, 'approvableType' => 'App\Models\Project'])

</div>

@endsection

**5. Testing**

Cobalah untuk menguji fungsionalitas ini dengan berbagai pengguna yang memiliki role yang berbeda (PPK, PPTK, Penanggung Jawab Akhir) untuk memastikan bahwa mereka dapat melakukan persetujuan sesuai dengan izin yang diberikan.

Dengan cara ini, Anda telah mengintegrasikan sistem persetujuan dengan **Spatie Permission & Role** dalam aplikasi Livewire Anda.