

Step-by-Step Guide: Creating a User Roles and Permissions App with Laravel 12

When building modern applications, one of the most common requirements is managing **user roles** and **permissions**. In this post, we'll break down the concepts, then show you how to implement them in a Laravel 12 app step by step — including a simple admin UI to manage roles and permissions.

Definitions

Role: A role is a named bundle of permissions (e.g., *Admin*, *Editor*). Assigning a role grants all permissions that role contains.

Permission: A permission represents a specific action (e.g., *posts.create*, *users.delete*). Permissions are the atomic units of access.

RBAC (Role-Based Access Control): A strategy where users get roles, and roles contain permissions. This avoids assigning dozens of permissions directly to each user.

Step 1 - Install Laravel and Spatie Roles & Permissions

```
composer create-project laravel/laravel roles-permissions-app
cd roles-permissions-app
```

```
composer require spatie/laravel-permissionCode language: JavaScript  
(javascript)
```

Publish config and migrations:

```
php artisan vendor:publish --  
provider="Spatie\Permission\PermissionServiceProvider"Code language:  
JavaScript (javascript)
```

Step 2 - Run Migrations

```
php artisan migrate
```

This creates roles, permissions, and pivot tables to link them with users.

Step 3 - Add Trait to User Model

```
// app/Models/User.php  
namespace App\Models;  
  
use Illuminate\Foundation\Auth\User as Authenticatable;  
use Spatie\Permission\Traits\HasRoles;  
  
class User extends Authenticatable
```

```
{  
    use HasRoles;  
  
    // ...  
}
```

Code language: PHP (php)

Step 4 - Seed Roles & Permissions

Create a seeder to define a baseline policy:

```
// database/seeder/RolesAndPermissionsSeeder.php  
namespace Database\Seeders;
```

```
use Illuminate\Database\Seeder;  
use Spatie\Permission\Models\Role;  
use Spatie\Permission\Models\Permission;
```

```
class RolesAndPermissionsSeeder extends Seeder  
{  
    public function run(): void  
    {  
        $permissions = [  
            'posts.create',  
            'posts.edit',  
            'posts.delete',  
            'users.view',  
            'users.edit',  
        ];  
  
        foreach ($permissions as $name) {
```

```
        Permission::firstOrCreate(['name' => $name]);
    }

    $admin = Role::firstOrCreate(['name' => 'Admin']);
    $editor = Role::firstOrCreate(['name' => 'Editor']);

    $admin->givePermissionTo($permissions);
    $editor->givePermissionTo(['posts.create', 'posts.edit']);
}
}
```

Code language: PHP (php)

```
php artisan db:seed --class=RolesAndPermissionsSeederCode language:
JavaScript (javascript)
```

Step 5 - Gate Your Routes

```
// routes/web.php
use Illuminate\Support\Facades\Route;

Route::get('/admin', fn() => 'Admin panel')
    ->middleware('role:Admin');

Route::get('/posts/create', fn() => 'Create post form')
    ->middleware('permission:posts.create');
Code language: PHP (php)
```

Step 6 - Build a Simple Management UI (Blade)

What you'll build: Minimal CRUD screens to list, create, and edit Roles and Permissions, plus assign permissions to roles and roles to users.

Generate controllers:

```
php artisan make:controller Admin/RoleController --resource
php artisan make:controller Admin/PermissionController --resource
php artisan make:controller Admin/UserRoleController
```

Routes for the admin UI (protect with role:Admin):

```
// routes/web.php
use App\Http\Controllers\Admin\RoleController;
use App\Http\Controllers\Admin\PermissionController;
use App\Http\Controllers\Admin\UserRoleController;

Route::middleware(['auth',
'role:Admin'])->prefix('admin')->name('admin.')->group(function () {
    Route::resource('roles', RoleController::class);          //
    index, create, store, edit, update, destroy
    Route::resource('permissions', PermissionController::class);

    // Assign roles to users
    Route::get('users/{user}/roles', [UserRoleController::class,
'edit'])->name('users.roles.edit');
    Route::put('users/{user}/roles', [UserRoleController::class,
'update'])->name('users.roles.update');
});
```

Code language: PHP (php)

Controller examples:

```
// app/Http/Controllers/Admin/RoleController.php
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Spatie\Permission\Models\Role;
use Spatie\Permission\Models\Permission;

class RoleController extends Controller
{
    public function index()
    {
        $roles = Role::with('permissions')->paginate(10);
        return view('admin.roles.index', compact('roles'));
    }

    public function create()
    {
        $permissions = Permission::orderBy('name')->get();
        return view('admin.roles.create', compact('permissions'));
    }

    public function store(Request $request)
    {
        $data = $request->validate([
            'name' => 'required|string|max:255|unique:roles,name',
            'permissions' => 'array',
        ]);

        $role = Role::create(['name' => $data['name']]);
        $role->syncPermissions($data['permissions'] ?? []);

        return redirect()->route('admin.roles.index')->with('status',
        'Role created.');
```

```
public function edit(Role $role)
{
    $permissions = Permission::orderBy('name')->get();
    $role->load('permissions');
    return view('admin.roles.edit', compact('role',
'permissions'));
}

public function update(Request $request, Role $role)
{
    $data = $request->validate([
        'name' => 'required|string|max:255|unique:roles,name,' .
$role->id,
        'permissions' => 'array',
    ]);

    $role->update(['name' => $data['name']]);
    $role->syncPermissions($data['permissions'] ?? []);

    return redirect()->route('admin.roles.index')->with('status',
'Role updated.');
```

```
}
```

```
public function destroy(Role $role)
```

```
{
```

```
    $role->delete();
```

```
    return back()->with('status', 'Role deleted.');
```

```
}
```

```
}
```

Code language: PHP (php)

```
// app/Http/Controllers/Admin/PermissionController.php
```

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\Controller;
```

```
use Illuminate\Http\Request;
```

```
use Spatie\Permission\Models\Permission;
```

```
class PermissionController extends Controller
```

```
{
    public function index()
    {
        $permissions = Permission::orderBy('name')->paginate(15);
        return view('admin.permissions.index',
compact('permissions'));
    }

    public function create()
    {
        return view('admin.permissions.create');
    }

    public function store(Request $request)
    {
        $data = $request->validate([
            'name' =>
'required|string|max:255|unique:permissions,name',
        ]);

        Permission::create(['name' => $data['name']]);
        return
redirect()->route('admin.permissions.index')->with('status',
'Permission created.');
```



```
        $permission->update(['name' => $data['name']]);
        return
    redirect()->route('admin.permissions.index')->with('status',
    'Permission updated.');
```

```
    }

    public function destroy(Permission $permission)
    {
        $permission->delete();
        return back()->with('status', 'Permission deleted.');
```

```
    }
}
```

Code language: PHP (php)

```
// app/Http/Controllers/Admin/UserRoleController.php
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Spatie\Permission\Models\Role;

class UserRoleController extends Controller
{
    public function edit(User $user)
    {
        $roles = Role::orderBy('name')->get();
        $user->load('roles');
        return view('admin.users.roles', compact('user', 'roles'));
    }

    public function update(Request $request, User $user)
    {
        $data = $request->validate([
            'roles' => 'array',
        ]);

        $user->syncRoles($data['roles'] ?? []);
        return redirect()->route('admin.users.roles.edit',
```

```
$user)->with('status', 'User roles updated.');
```

```
    }
```

```
}
```

Code language: PHP (php)

Example Blade views (Bootstrap 5 layout implied):

```
<!-- resources/views/admin/roles/index.blade.php -->
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="d-flex justify-content-between align-items-center mb-3">
        <h1 class="h3">Roles</h1>
        <a href="{{ route('admin.roles.create') }}" class="btn btn-
primary">New Role</a>
    </div>

    <table class="table table-striped">
        <thead><tr><th>Name</th><th>Permissions</th><th class="text-
end">Actions</th></tr></thead>
        <tbody>
            @foreach($roles as $role)
                <tr>
                    <td>{{ $role->name }}</td>
                    <td>{{ $role->permissions->pluck('name')->join(', ') ?: '-'
}}</td>
                    <td class="text-end">
                        <a href="{{ route('admin.roles.edit', $role) }}"
class="btn btn-sm btn-outline-secondary">Edit</a>
                        <form action="{{ route('admin.roles.destroy', $role) }}"
method="POST" class="d-inline">
                            @csrf @method('DELETE')
                            <button class="btn btn-sm btn-outline-danger"
onclick="return confirm('Delete this role?')">Delete</button>
                        </form>
                    </td>
                </tr>
            @endforeach
        </tbody>
    </table>
</div>
```

```
</tbody>
</table>

{{ $roles->links() }}
</div>
@endsection
Code language: PHP (php)

<!-- resources/views/admin/users/roles.blade.php -->
@extends('layouts.app')

@section('content')
<div class="container">
    <h1 class="h3 mb-3">Manage Roles for: {{ $user->name }}</h1>

    @if(session('status'))
        <div class="alert alert-success">{{ session('status') }}</div>
    @endif

    <form method="POST" action="{{ route('admin.users.roles.update',
$user) }}" class="card card-body">
        @csrf @method('PUT')

        <div class="row">
            @foreach($roles as $role)
                <div class="col-md-4 mb-2">
                    <div class="form-check">
                        <input class="form-check-input" type="checkbox"
name="roles[]" value="{{ $role->name }}"
                        id="role_{{ $role->id }}" {{ $user->hasRole($role->name)
? 'checked' : '' }}>
                        <label for="role_{{ $role->id }}" class="form-check-
label">{{ $role->name }}</label>
                    </div>
                </div>
            @endforeach
        </div>

        <div class="mt-3">
```

```
        <button class="btn btn-primary">Save</button>
    </div>
</form>
</div>
@endsection
Code language: PHP (php)
```

In Blade, you can also show/hide controls using directives:

```
@role('Admin')
    <a href="{{ route('admin.roles.index') }}" class="btn btn-sm btn-
outline-primary">Manage Roles</a>
@endrole

@can('posts.create')
    <a href="{{ url('/posts/create') }}" class="btn btn-primary">New
Post</a>
@endcan
Code language: PHP (php)
```

Security notes: Keep the admin routes behind `auth + role:Admin`. Validate inputs. Prefer `syncRoles` / `syncPermissions` to avoid stale assignments.

Step 7 - Test Assignments Quickly

```
php artisan tinker
```

```
>>> $u = \App\Models\User::first();
>>> $u->assignRole('Admin');
```

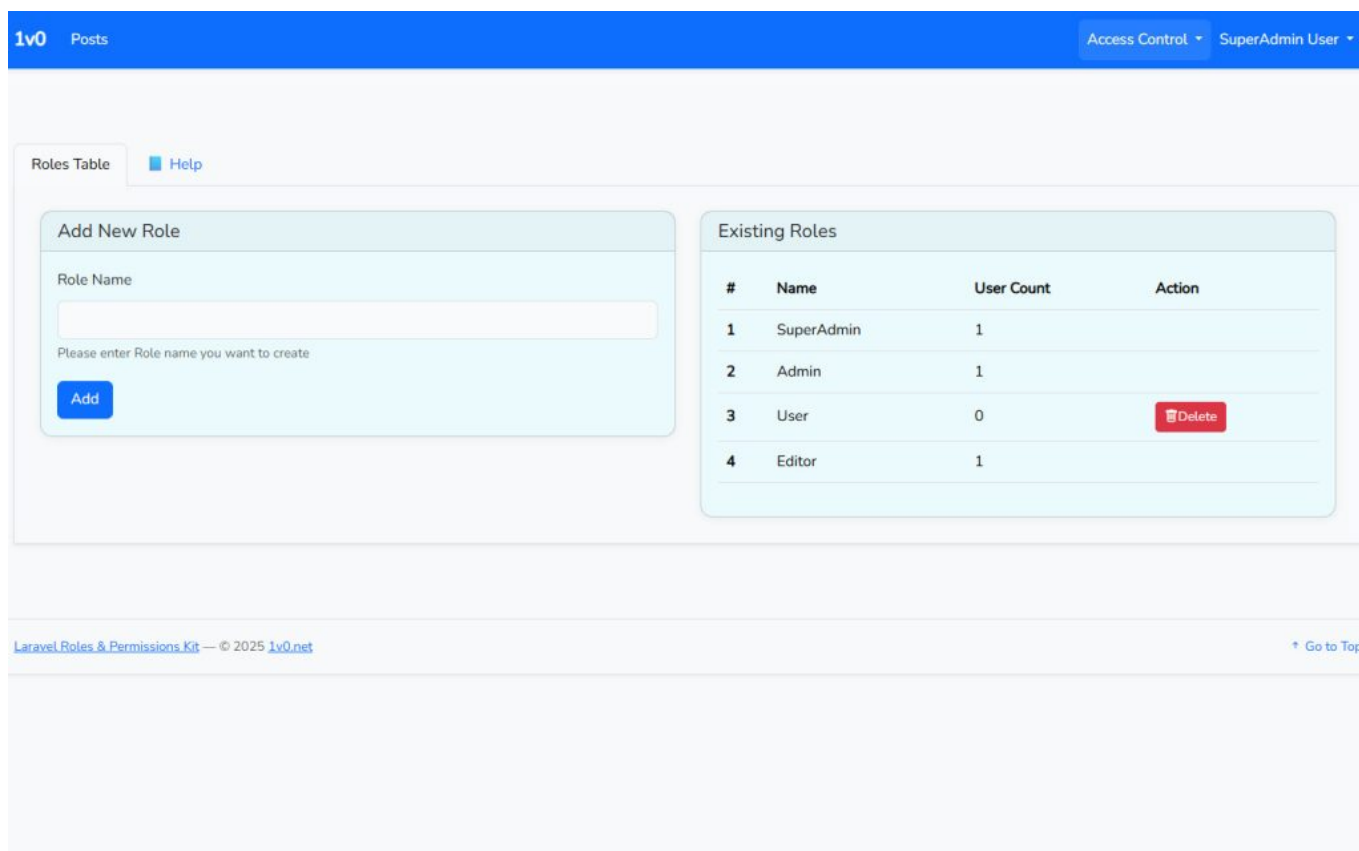
```
>>> $u->can('users.edit'); // true if Admin has this permissionCode  
language: PHP (php)
```

Conclusion

You now have a working RBAC foundation in Laravel 12: roles, permissions, protected routes, and a minimal UI to manage everything. From here, you can expand the interface (search, pagination, bulk assign), add activity logs, and expose APIs for admin operations.

Grab a production-ready implementation

If you don't want to wire up all the edge cases (UI, roles, permissions, settings toggles, demos), **Grab a production-ready implementation:**



Laravel Roles & Permissions UI

This lightweight Laravel starter kit helps you quickly integrate user roles and permissions using the popular spatie/laravel-permission package.

[Learn more](#)