

step1- Run the Spatie role permission comand

```
composer require spatie/laravel-permission
```

step2- after that run to publish spatie

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

step3- migrate the permission and roles table

```
php artisan migrate
```

step4- use inside the user model

```
use Illuminate\Foundation\Auth\User as Authenticatable;
```

```
use Spatie\Permission\Traits\HasRoles;
```

```
class User extends Authenticatable
```

```
{
```

```
    use HasRoles;
```

```
}
```

step5- create neccessary seeders in this sequence

```
run -> php artisan make:seeder UsersTableSeeder
```

```
<?php
```

```
namespace Database\Seeders;
```

```
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
```

```
use Illuminate\Database\Seeder;
```

```
use App\Models\User;
```

```
use Illuminate\Support\Facades\Hash;
```

```
class UsersTableSeeder extends Seeder
```

```

{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $users = [
            [
                'name' => 'Admin',
                'email' => 'admin@mail.com',
                'password' => 'Admin@123',
            ],
            [
                'name' => 'Super Admin',
                'email' => 'superadmin@gmail.com',
                'password' => 'superadmin@123',
            ]
        ];

        foreach($users as $user) {
            $created_user = User::create([
                'name' => $user['name'],
                'email' => $user['email'],
                'password' => Hash::make($user['password']),
            ]);
        }
    }
}

```

```
}  
}
```

run -> php artisan make:seeder RolesSeeder

```
<?php
```

```
namespace Database\Seeders;
```

```
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
```

```
use Illuminate\Database\Seeder;
```

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

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

```
class RolesSeeder extends Seeder
```

```
{
```

```
    /**
```

```
     * Run the database seeds.
```

```
     */
```

```
    public function run(): void
```

```
    {
```

```
        $role_admin    = Role::firstOrCreate(['name' => 'admin']);
```

```
        $role_superadmin= Role::firstOrCreate(['name' => 'superadmin']);
```

```
        $role_teacher  = Role::firstOrCreate(['name' => 'teacher']);
```

```
        $role_student  = Role::firstOrCreate(['name' => 'student']);
```

```
    }
```

```
}
```

```
run -> php artisan make:seeder PermissionSeeder
```

```
<?php
```

```
namespace Database\Seeders;
```

```
use App\Models\User;
```

```
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
```

```
use Illuminate\Database\Seeder;
```

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

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

```
class PermissionSeeder extends Seeder
```

```
{
```

```
    /**
```

```
     * Run the database seeds.
```

```
     */
```

```
    public function run(): void
```

```
    {
```

```
        $permissions = [
```

```
            'dashboard',
```

```
            'categories',
```

```
            'categories.category_level_0.create',
```

```
            'categories.category_level_0.view',
```

```
            'categories.category_level_0.edit',
```

```
            'categories.category_level_0.delete',
```

```

        'categories.category_level_1.create',
        'categories.category_level_1.view',
        'categories.category_level_1.edit',
        'categories.category_level_1.delete',
        'categories.category_level_2.create',
        'categories.category_level_2.view',
        'categories.category_level_2.edit',
        'categories.category_level_2.delete',
        'content',
    ];

    // Looping and Inserting Array's Permissions into Permission Table
    foreach ($permissions as $permission) {
        Permission::create(['name' => $permission]);
    }

    // Check if "admin", 'superadmin' role exists
    $role_admin = Role::firstOrCreate(['name' => 'admin']);
    $role_superadmin = Role::firstOrCreate(['name' => 'superadmin']);

    // Assign all permissions to the "admin", 'superadmin' role if not already assigned
    $role_admin->syncPermissions(Permission::all());
    $role_superadmin->syncPermissions(Permission::all());

    // Assign the "admin" role to all users who have this role
    $adminUsers = User::role('admin')->get();
    $superadminUsers = User::role('superadmin')->get();

    foreach ($adminUsers as $user1) {
        $user1->syncPermissions(Permission::all());
    }

```

```

        foreach ($superadminUsers as $user2) {

            $user2->syncPermissions(Permission::all());

        }

    }

}

```

step6- load the new seeders inside the DatabaseSeeder

```

public function run(): void
{
    $this->call([

        UsersTableSeeder::class,

        RolesSeeder::class,

        PermissionSeeder::class,

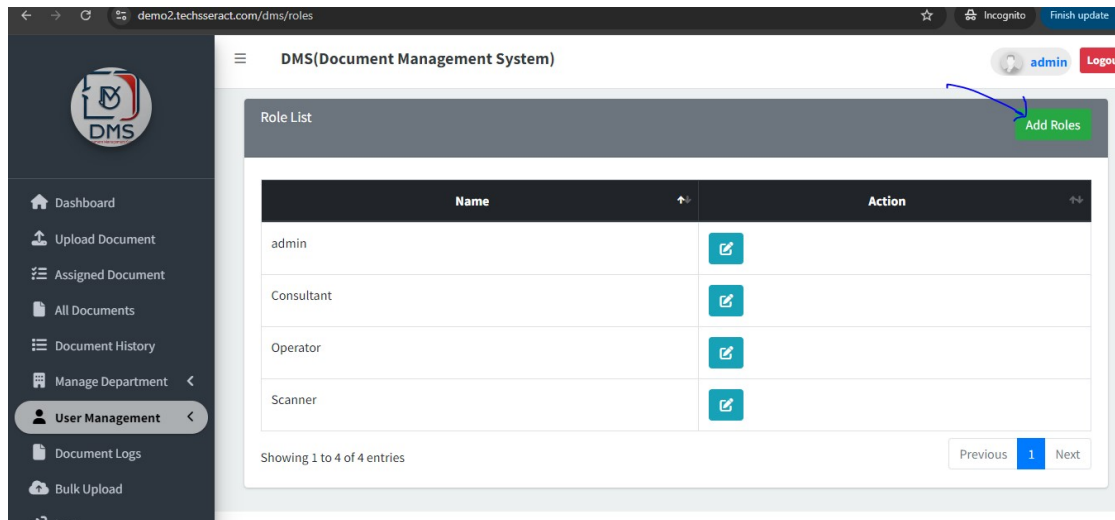
    ]);
}

```

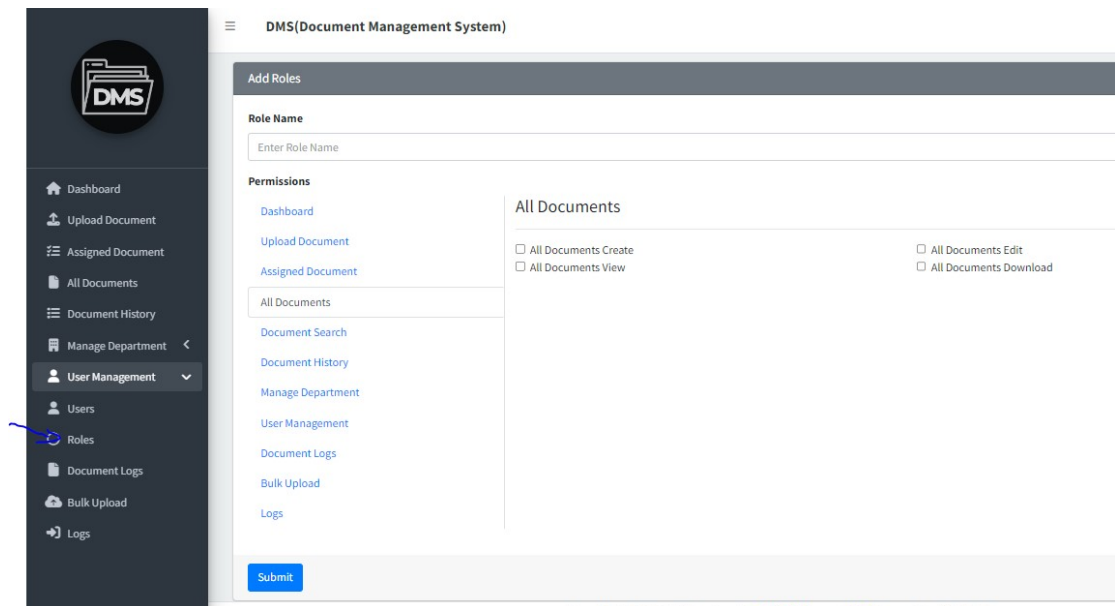
step7- Run this command to recreate tables and seeders

```
php artisan migrate:refresh --seed
```

step8- Create layout to add role Like this



step9- Add role form have listing of permission to assign roles



permission to show in lsit tab(code)-

```
public function create()
{
    // $permissions = Permission::get();
    $roles = Role::get();
    $permissions = Permission::get();
    $data = [];
```

```

foreach ($permissions as $permission) {

    $temp = explode('.', $permission->name);

    if (!isset($data[$temp[0]])) {

        $data[$temp[0]] = [];

    }

    $name = "";

    for ($i = 0; $i < count($temp); $i++) {

        $namePart = str_replace('_', ' ', $temp[$i]);

        $name .= $namePart;

        if ($i < count($temp) - 1) {

            $name .= ' ';

        }

    }

    $data[$temp[0]][] = (object)['name' => $name, 'id' => $permission->id];

}

$dataArray = $data;

return view('role.create', compact('dataArray','roles'));

}

```

Role Blade Form to add permission-

```

@extends('layouts.app')

@section('title', 'DMS | Role Create')

@section('content')

<div class="container-fluid">

    <div class="row">

        <div class="col-md-12">

            <div class="card card-secondary">

                <div class="card-header">

```



```

        <h3 class="card-title">Add Roles</h3>
    </div>

    <form action="{{ route('roles.store') }}" method="POST">

        @csrf

        <div class="card-body">

            <div class="form-group">

                <label for="roleName">Role Name</label>

                <input type="text" name="name" class="form-control" id="roleName"
placeholder="Enter Role Name" required>

            </div>

            <div class="form-group">

                <label for="permissions">Permissions</label>

                <div class="row">

                    <div class="col-5 col-sm-3">

                        <div class="nav flex-column nav-tabs h-100" id="vert-tabs-tab"
role="tablist" aria-orientation="vertical">

                            @foreach($dataArray as $key=>$list)

                                <a class="nav-link" id="{{ $key }}-tab" data-toggle="pill"
href="#{{ $key }}" role="tab" aria-controls="{{ $key }}" aria-selected="true">

                                    @php

                                        $formattedKeyTitle = ucwords(str_replace('_', ' ', $key));

                                    @endphp

                                    {{ $formattedKeyTitle }}

                                </a>

                            @endforeach

                        </div>

                    </div>

                    <div class="col-7 col-sm-9">

```

```

<div class="tab-content" id="vert-tabs-tabContent">

    @foreach($dataArray as $key=>$item)

        <div class="tab-pane text-left fade {{ $loop->first ? 'show active' :
" }}" id="{{ $key }}" role="tabpanel" aria-labelledby="{{ $key }}-tab">

            @php
                $formattedKey = ucwords(str_replace('_', ' ', $key));
            @endphp

            <h4>{{ $formattedKey }}</h4>

            <hr>

            <div class="row">

                @foreach ($item as $keyval=>$list)

                    <div class="col-6">

                        <div class="form-check">

                            <input class="form-check-input" type="checkbox"

                                name="permission[]" id="permission_{{ $list->id }}"

                                value="{{ $list->id }}">

                            <label class="form-check-label"

                                for="permission_{{ $list->id }}">

                                {{ Str::title($list->name) }}

                            </label>

                        </div>

                    </div>

                </div>

            @endforeach

        </div>

    </div>

```

```
@endforeach

</div>

</div>

</div>

</div>

</div>

<div class="card-footer">

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

</div>

</form>

</div>

</div>

</div>

</div>

@endsection
```

```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|unique:roles,name',
        'permission' => 'required|array',
    ]);

    $role = Role::create(['name' => $request->input('name')]);
}
```

```

$permissionNames = $request->input('permission');

$permissions = Permission::whereIn('id', $permissionNames)->pluck('id');

$role->syncPermissions($permissions);

return redirect()->route('roles.index')

->with('success', 'Role created successfully');
}

```

Edit And Update code of roled assign permissions-

Edit Role

Role Name
admin

Permissions

<input checked="" type="checkbox"/> Dashboard	<input checked="" type="checkbox"/> Upload Document	<input checked="" type="checkbox"/> Assigned Document
<input checked="" type="checkbox"/> Assigned Document Document Create	<input checked="" type="checkbox"/> Assigned Document Document View	<input checked="" type="checkbox"/> All Documents Create
<input checked="" type="checkbox"/> All Documents Edit	<input checked="" type="checkbox"/> All Documents View	<input checked="" type="checkbox"/> All Documents Download
<input checked="" type="checkbox"/> Document Search	<input checked="" type="checkbox"/> Document History	<input checked="" type="checkbox"/> Manage Department Add Department Create
<input checked="" type="checkbox"/> Manage Department Department Show	<input checked="" type="checkbox"/> Manage Department Department Edit	<input checked="" type="checkbox"/> Manage Department Department Delete
<input checked="" type="checkbox"/> Manage Department Department Section Create	<input checked="" type="checkbox"/> Manage Department Department Section Show	<input checked="" type="checkbox"/> Manage Department Department Section Edit
<input checked="" type="checkbox"/> Manage Department Department Section Delete	<input checked="" type="checkbox"/> Manage Department File Master Create	<input checked="" type="checkbox"/> Manage Department File Master Show
<input checked="" type="checkbox"/> Manage Department File Master Edit	<input checked="" type="checkbox"/> Manage Department File Master Delete	<input checked="" type="checkbox"/> Manage Department File Master Upload
<input checked="" type="checkbox"/> User Management Users Create	<input checked="" type="checkbox"/> User Management Users View	<input checked="" type="checkbox"/> User Management Users Edit
<input checked="" type="checkbox"/> User Management Users Delete	<input checked="" type="checkbox"/> User Management Users Assign	<input checked="" type="checkbox"/> User Management Roles Create
<input checked="" type="checkbox"/> User Management Roles View	<input checked="" type="checkbox"/> User Management Roles Edit	<input checked="" type="checkbox"/> User Management Roles Delete
<input checked="" type="checkbox"/> User Management Category Create	<input checked="" type="checkbox"/> User Management Category View	<input checked="" type="checkbox"/> User Management Category Edit
<input checked="" type="checkbox"/> User Management Category Delete	<input checked="" type="checkbox"/> Document Logs	<input checked="" type="checkbox"/> Bulk Upload Create
<input checked="" type="checkbox"/> Bulk Upload Show	<input checked="" type="checkbox"/> Bulk Upload Edit	<input checked="" type="checkbox"/> Bulk Upload Delete
<input checked="" type="checkbox"/> Logs		

Update

Controller code-

```

public function edit($id)
{
    $role = Role::find($id);

    $permissions = Permission::get();

    $rolePermissions = DB::table("role_has_permissions")->

```

```

where("role_has_permissions.role_id", $id)

    ->pluck('role_has_permissions.permission_id', 'role_has_permissions.permission_id')

    ->all();

return view('role.edit', compact('role', 'permissions', 'rolePermissions'));
}

public function update(Request $request, $id)
{

    $this->validate($request, [

        'name' => 'required',

        'permissions.*' => 'exists:permissions,id',

    ]);

    $role = Role::find($id);

    $role->name = $request->input('name');

    $role->save();

    $permissionNames = $request->input('permissions');

    if (!empty($permissionNames)) {

        $validPermissions = Permission::whereIn('id', $permissionNames)->pluck('id');

        $role->syncPermissions($validPermissions);

    } else {

        $role->syncPermissions([]);

    }

    return redirect()->route('roles.index')

        ->with('success', 'Role updated successfully');
}

```

```
}
```

blade code Role edit-

```
@extends('layouts.app')

@section('title', 'DMS | Role Edit')

@section('content')

<div class="container-fluid">

    <div class="row">

        <div class="col-md-12">

            <div class="card card-secondary">

                <div class="card-header">

                    <h3 class="card-title">Edit Role</h3>

                </div>

                <form action="{{ route('roles.update', $role->id) }}" method="POST">

                    @csrf

                    @method('PUT')

                    <div class="card-body">

                        <div class="form-group">

                            <label for="roleName">Role Name</label>

                            <select name="name" id="name" class="form-control">

                                <option value="{{ $role->name }}">{{ $role->name }}</option>

                            </select>

                        </div>

                    </div>

                </form>

            </div>

        </div>

    </div>

</div>
```

```

<div class="form-group">

    <label for="permissions">Permissions</label>

    <div class="row">

        @foreach($permissions as $permission)

            <div class="col-md-4">

                <div class="form-check">

                    <input type="checkbox" name="permissions[]"
value="{{ $permission->id }}" class="form-check-input" id="permission{{ $permission->id }}"

                    @if($role->permissions->contains($permission->id)) checked

                @endif>

                    <label class="form-check-label" for="permission{{ $permission->
id }}">{{ $formattedKeyTitle = ucwords(str_replace(['.', '_'], ' ', $permission->name)); }}
                </label>

            </div>

        </div>

    @endforeach

</div>

<div class="card-footer">

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

</div>

</form>

</div>

</div>

</div>

@endsection

```

step-10 - Create a middleware 'CheckPermission'

```
php artisan make:middleware CheckPermission
```

assign in kernel.php (inside middleware aliases)-

```
'permission' => \App\Http\Middleware\CheckPermission::class,
```

CheckPermission.php-

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use Closure;
```

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

```
use Illuminate\Support\Facades\Auth;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
class CheckPermission
```

```
{
```

```
    /**
```

```
     * Handle an incoming request.
```

```
     *
```

```
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation
```



```
\Response) $next

*/

public function handle(Request $request, Closure $next, $permission)
{
    // Check if the user is authenticated and has the specified permission
    if (Auth::check() && Auth::user()->can($permission)) {
        return $next($request);
    }

    // Redirect or abort if permission check fails
    return redirect('home')->with('error', 'You do not have permission to access this page.');
```

step-10 - Apply this Permission middleware on Web

```
Subject: Resignation Notice Untitled-1 • web.php 9+ X RoleController.php Kernel.php
routes > web.php > Closure > Closure
40 route::post('password/reset', [PasswordResetController::class, 'reset'])->name('password.update');
41
42
43 Route::middleware('auth')->group(function () {
44     Route::middleware('log.user.access')->group(function () {
45         Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
46
47         // Route::get('/home', 'HomeController@index')->name('home');
48         Route::resource('roles', RoleController::class)->middleware('permission:user_management.roles.view');
49         Route::resource('users', UserController::class)->middleware('permission:user_management.users.view');
50         Route::get('change_profile', [UserController::class, 'changeProfile'])->name('change_profile');
51         Route::post('change_profile_store', [UserController::class, 'changeProfileStore'])->name('change_prof
52
```

like this-

```
Route::resource('roles', RoleController::class)->
middleware('permission:user_management.roles.view');
```

on blade file apply permission-

```
@can('assigned_document.document.view')
<li class="nav-item">
    <a href="{{ route('assigned_document') }}" class="nav-link">
        <i class="fa fa-tasks nav-icon" aria-hidden="true"></i>
        <p>Assigned Document</p>
    </a>
</li>
@endcan
@can('all_documents.view')
<li class="nav-item">
    <a href="{{ route('documents.index') }}" class="nav-link">
        <i class="fa fa-file nav-icon" aria-hidden="true"></i>
        <p>All Documents</p>
    </a>
</li>
@endcan
```

