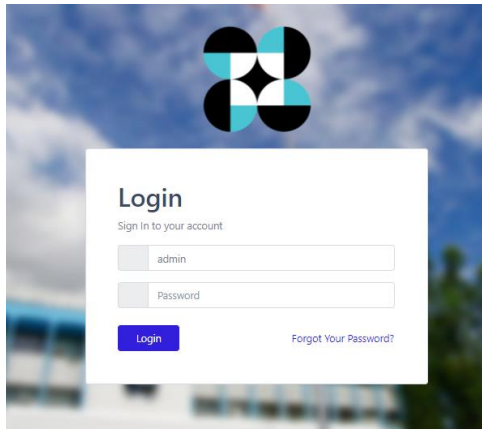


System Documentation Of HRMIS System For DOST XI with User manual, Installation guide and source code.

Note: All blades on this documented system and needed codes can be found at the Source code page.

Login Section



Login link is <http://127.0.0.1:8000>

Login section is composed of:

- username
- password

Login blade and other needed codes for the login section can be found at the Source code page.

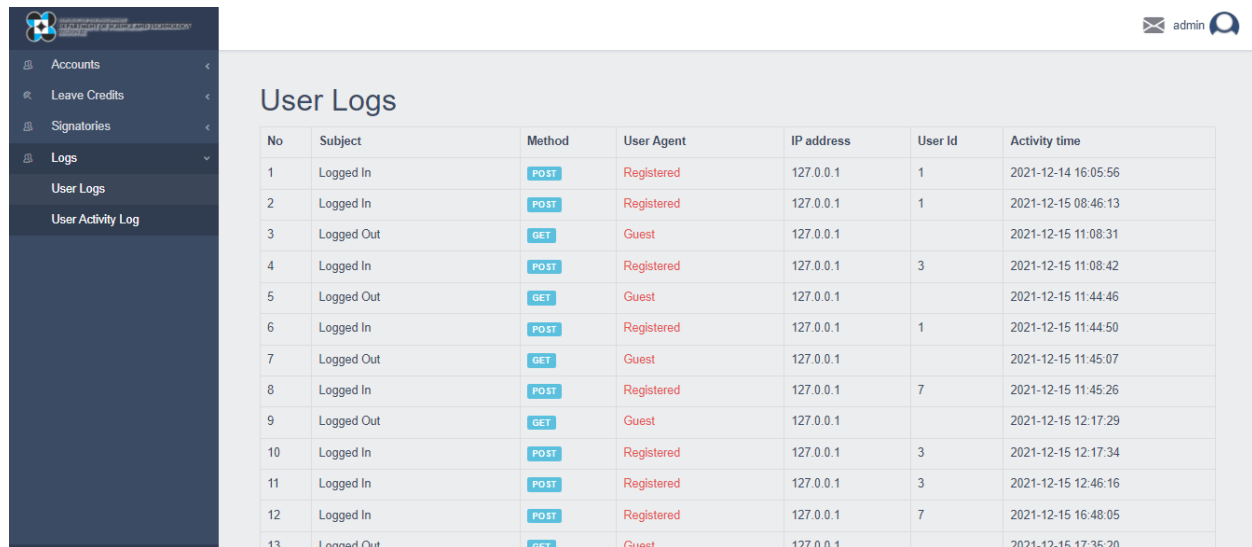
Add User Section

Adduser link is <http://127.0.0.1:8000/admin/users/add>

Add User section is composed of:

- First name , Middle name, Last name and email on Basic Information.
- Username, Password, Confirm Password, Role and Division on Username and Password table.

User Logs



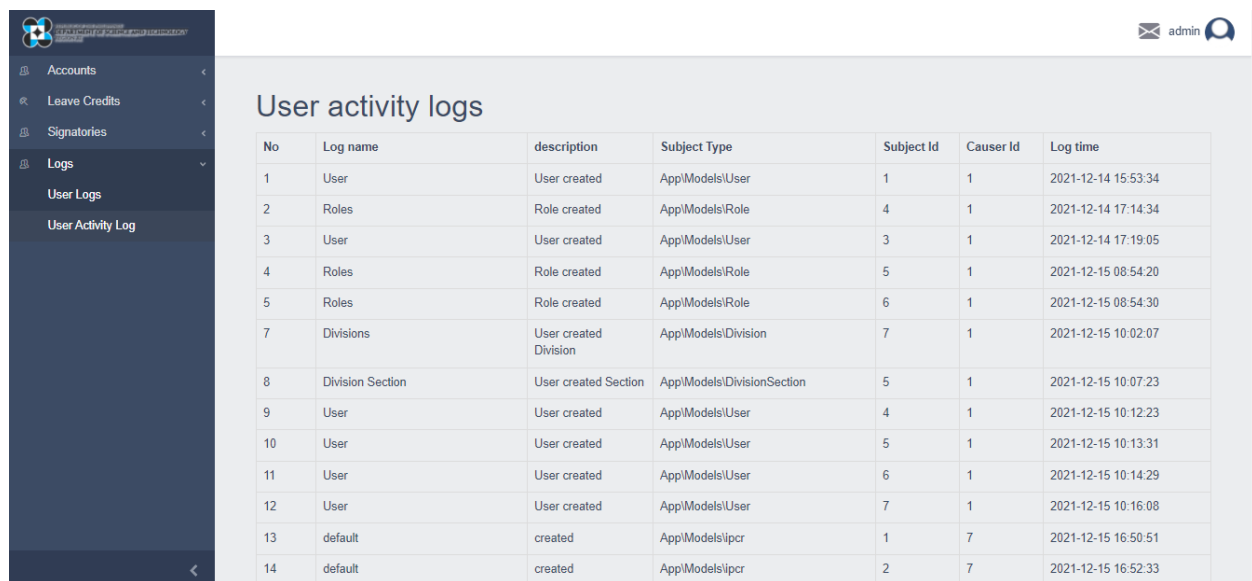
No	Subject	Method	User Agent	IP address	User Id	Activity time
1	Logged In	POST	Registered	127.0.0.1	1	2021-12-14 16:05:56
2	Logged In	POST	Registered	127.0.0.1	1	2021-12-15 08:46:13
3	Logged Out	GET	Guest	127.0.0.1		2021-12-15 11:08:31
4	Logged In	POST	Registered	127.0.0.1	3	2021-12-15 11:08:42
5	Logged Out	GET	Guest	127.0.0.1		2021-12-15 11:44:46
6	Logged In	POST	Registered	127.0.0.1	1	2021-12-15 11:44:50
7	Logged Out	GET	Guest	127.0.0.1		2021-12-15 11:45:07
8	Logged In	POST	Registered	127.0.0.1	7	2021-12-15 11:45:26
9	Logged Out	GET	Guest	127.0.0.1		2021-12-15 12:17:29
10	Logged In	POST	Registered	127.0.0.1	3	2021-12-15 12:17:34
11	Logged In	POST	Registered	127.0.0.1	3	2021-12-15 12:46:16
12	Logged In	POST	Registered	127.0.0.1	7	2021-12-15 16:48:05
13	Logged Out	GET	Guest	127.0.0.1		2021-12-15 17:35:20

User logs link is <http://127.0.0.1:8000/admin/logActivity>

User logs section is consist of log entry number , Subject, Method, User Agent, IP Address, User Id, Activity time.

This section shows all the logged In and Logged out time of all the person who access the system.

User Activity Logs



No	Log name	description	Subject Type	Subject Id	Causer Id	Log time
1	User	User created	App\Models\User	1	1	2021-12-14 15:53:34
2	Roles	Role created	App\Models\Role	4	1	2021-12-14 17:14:34
3	User	User created	App\Models\User	3	1	2021-12-14 17:19:05
4	Roles	Role created	App\Models\Role	5	1	2021-12-15 08:54:20
5	Roles	Role created	App\Models\Role	6	1	2021-12-15 08:54:30
7	Divisions	User created Division	App\Models\Division	7	1	2021-12-15 10:02:07
8	Division Section	User created Section	App\Models\DivisionSection	5	1	2021-12-15 10:07:23
9	User	User created	App\Models\User	4	1	2021-12-15 10:12:23
10	User	User created	App\Models\User	5	1	2021-12-15 10:13:31
11	User	User created	App\Models\User	6	1	2021-12-15 10:14:29
12	User	User created	App\Models\User	7	1	2021-12-15 10:16:08
13	default	created	App\Models\ipcr	1	7	2021-12-15 16:50:51
14	default	created	App\Models\ipcr	2	7	2021-12-15 16:52:33

User Activity Logs link is <http://127.0.0.1:8000/admin/activitylog>

User Activity section is consist of id, log_name, description, subject_type, subject_id, causer_id, logtime(created_at).

This section shows the user activity when accessing the system. It will show what task they made, who was the user and what time it was created.

Evaluation Section

-Create an Evaluation

Employee: ADMIN : ADMIN

Section Head: ADMIN : ADMIN

Division Head: ADMIN : ADMIN

HR: ADMIN : ADMIN

Year:

Submit

Create an evaluation section link is <http://127.0.0.1:8000/employee-eval/initial-form>

It consist of employee selection, Section Head selection, Division Selection, HR and Year.

In this section the hr user can choose the employee who can evaluate the person they want to be evaluated.

Once done the system will then saved the data to the database and will show on the next section which is the pending section.

Note: Only HR role can access and choose the list of user who are eligible for evaluation.

Evaluation Section(Rating section)

NAME	COST-EFFECTIVE AND EFFICIENCY	EXEMPLARY SERVICES	INNOVATIVENESS	ADHERENCE TO CODE OF ETHICS	WORK ETHICS	PERSONAL RELATIONS	TOTAL
	35%	10%	15%	10%	20%	10%	100%
CENA, JOHN	PENDING	PENDING	PENDING	PENDING	<input type="text"/> submit	PENDING	0%
ABUEVA, GEVY	PENDING	PENDING	PENDING	PENDING	<input type="text"/> submit	PENDING	0%

Pending rating of evaluation section link is <http://127.0.0.1:8000/employee-eval/view-pending>

This section is where the user who were chosen to rate and now can access this section.

The pending section can only be access by the person who was chosen by the hr user.

Once the rate is submitted the system will automatically saved the data and it will then show to other personnel who will rate other parts of the pending section.

After all the rates is submitted, it will then show to the completed section all the rates that was made by the person chosen by the hr user.

Note: Each rating table has its corresponding personnel who rate and only has the capable to input the score.

Here is the list of Users who can rate to the corresponding tables.


Employee: Cost- Effective and Efficiency & Exemplary Services tables.

Section Head: Innovativeness and Adherence to Code Ethics tables

Division Head: Personal Relations table

HR: Work Ethics table

Evaluation Section (Completed rate)

 OFFICE OF THE SECRETARY OF DEPARTMENT OF HUMAN RESOURCES MANAGEMENT

Leave Approvals <

Leave Requests <

IPCR <

IPCR Approvals <

Evaluation for HR >

Create

Pending

Completed

caluag123

Completed Evaluations

NAME	COST-EFFECTIVE AND EFFICIENCY	EXEMPLARY SERVICES	INNOVATIVENESS	ADHERENCE TO CODE OF ETHICS	WORK ETHICS	PERSONAL RELATIONS	TOTAL
	35%	10%	15%	10%	20%	10%	100%
CENA, JOHN	100%	100%	100%	100%	90%	100%	590%
ABUEVA, GEVY	100%	100%	100%	100%	0%	100%	500%
CALUAG, ED	90%	90%	90%	95%	90%	90%	545%

Link of the completed rating for evaluation section is <http://127.0.0.1:8000/employee-eval/view-completed>

This section it the completed ratings of the evaluation section.

Source Codes:

Login Section

Login Blade:

```
@extends('layouts.auth_app')
@section('content')

<body style="background-image: url('/images/logos/dost1.jpg');
background-size: cover">
<div class="container" style="width:750px; margin-top:-80px;">
<div>

</div><br>
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card-group">
<div class="card p-4">
<div class="card-body">
<h1>Login</h1>
<p class="text-muted">Sign In to your account</p>
<form method="POST" action="{{ route('login') }}">
@csrf
<div class="input-group mb-3">
@if (count($errors) > 0)
<div class="error">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<div class="input-group-prepend">
<span class="input-group-text">
<svg class="c-icon">
<use xlink:href="assets/icons/coreui/free-symbol-
defs.svg#cui-user"></use>
</svg>
</span>
</div>
<input class="form-control" type="text" placeholder="{{
__('username') }}" name="username" value="{{ old('username') }}" required
autofocus>
</div>
<div class="input-group mb-4">
<div class="input-group-prepend">
<span class="input-group-text">
<svg class="c-icon">
<use xlink:href="assets/icons/coreui/free-symbol-
defs.svg#cui-lock-locked"></use>
</svg>
</span>
</div>
<input class="form-control" type="password" placeholder="{{
__('Password') }}" name="password" required>
</div>
<div class="row">
<div class="col-6">
<button class="btn btn-primary px-4" type="submit">{{
__('Login') }}</button>
</div>
</form>
<div class="col-6 text-right">
<a href="{{ route('password.request') }}" class="btn btn-link px-0">{{
__('Forgot Your Password?') }}</a>
</div>
</div>
</div>
</div>
</div>
```

```
</div>
</div>
</div>
</body>
@endsection
@section('javascript')
@endsection
```

Controllers:

Login Controller:

```
<?php
namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
//use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Auth;
use Session;
class LoginController extends Controller
{
    /*
    |-----
    | Login Controller
    |-----
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    //use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = '/dashboard';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }

    public function showLoginForm() {
        return view('auth.login');
    }

    public function handleLogin() {
        $user_login = ['username'=>request()->username,
            'password'=>request()->password
        ];
        //dd(Hash::make(request()->password));
        // $user = New User;
        // dd($user->getAuthPassword());
        //dd(Auth::attempt($user_login));
        if(Auth::attempt($user_login)) {
            return redirect()->route('dashboard');
            //dd(1);
            //return redirect('login')->with('err_msg', 'Test Password!');
        }
    }
}
```

```

        return redirect('login')->with('err_msg', 'Invalid username/password');
    }

    public function logout() {
        Session::flush();
        Auth::logout();
        return redirect('login');
    }
}

```

User Controller codes:

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use App\Models\Division;
use App\Models\User;
use App\Models\RoleUser;

class UserController extends Controller
{
    public function login(Request $request) {
        // GET
        if ($request->isMethod('get')) {
            return view('auth.login');
        }
        // POST
        if ($request->isMethod('post')) {
            $credentials = $request->validate([
                'username' => ['required'],
                'password' => ['required'],
            ]);
            if (Auth::attempt($credentials)) {
                $request->session()->regenerate();
                return redirect()->route('admin-dashboard');
            }
            return back()->withErrors([
                'username' => 'The provided credentials do not match our
records.',
            ]);
        }
    }

    public function register(Request $request) {
        // GET
        if ($request->isMethod('get')) {
            return view('auth.register');
        }
        // POST
        if ($request->isMethod('post')) {
            $role_selected = $request['role'];
            $div=$request['division'];

            $form_data = $request->all();

            $user = User::create([
                'username' => $form_data['username'],
                'first_name' => $form_data['first_name'],
                'middle_name' => $form_data['middle_name'],
                'last_name' => $form_data['last_name'],
                'division' => $div,
                'email' => $form_data['email'],
                'password' => Hash::make($form_data['password']),
            ]);

```

```

            $role = RoleUser::create([
                'user_id' => $user->id,
                'role_id' => $role_selected
            ]);

            return redirect()->route('admin-users', [])
                ->with('success_registration', 'New User Created');
        }
    }
}

```

AdminController

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Role;
use App\Models\Division;
use App\Models\Permission;
use App\Models\RolePermission;
use Pharlo\Manifest\CopyrightInformation;
use App\Models\Employee_Evaluation;
use App\Models\EvaluationDetail;

class AdminController extends Controller
{
    public function dashboard()
    {
        return view('admin.dashboard');
    }
    public function users()
    {
        $roles = array();
        foreach (Role::all() as $record) {
            $permissions = array();
            foreach ($record->permissions as $permission) {
                array_push($permissions, Permission::find($permission-
>permission_id)->name);
            }
            array_push($roles, [
                'id' => $record->id,
                'name' => $record->name,
                'permissions' => $permissions
            ]);
        }
        $roles_dict = Role::all();

        return view(
            'admin.users',
            [
                'users' => User::all(),
                'roles' => $roles,
                'roles_dict' => $roles_dict
            ],
        );
    }
    public function addUser()
    {
        return view('admin.addUser', ['roles' => Role::all()], ['div' =>
Division::all()]);
    }
    public function roles()
    {
        return view('admin.roles', [
            'roles' => Role::all(),
            'permissions' => Permission::all()
        ]);
    }
}

```

```

public function addRole(Request $request)
{
    $role = Role::create([
        'name' => $request->name
    ]);
    $role->save();
    $permissions = $request->permissions;
    foreach ($permissions as $permission) {
        RolePermission::create([
            'permission_id' => $permission,
            'role_id' => $role->id
        ]);
    }

    return redirect()->route('admin-roles')
        ->with('success_role_added', 'New Role Added');
}

public function permissions()
{
    return view('admin.permissions', [
        'permissions' => Permission::all()
    ]);
}

public function updateRole(Request $request)
{
    $role = Role::find($request->role_id);
    $role->name = $request->name;
    $role->save();

    // Delete all role_permissions
    $permission_record = RolePermission::where('role_id', $request->role_id)->delete();

    // Re-add all role_permissions
    $permissions = $request->permissions;
    foreach ($permissions as $permission) {
        RolePermission::create([
            'permission_id' => $permission,
            'role_id' => $role->id
        ]);
    }

    return redirect()->route('admin-roles');
}

public function addPermission(Request $request)
{
    $name = $request->name;

    Permission::create([
        'name' => $name
    ]);

    return redirect()
        ->route('admin-permissions')
        ->with('create_success', 'Permission Added');
}

public function updatePermission(Request $request)
{
    error_log($request);
    $permission_id = $request->permission_id;
    $permission = Permission::find($permission_id);
    $permission->name = $request->name;
    $permission->save();

    return redirect()
        ->route('admin-permissions')
        ->with('update_success', 'Permission Name Updated');
}

public function ipcrOperations(){
    return view('admin.ipcroperations');
}

```

```

public function __construct()
{
}
/**
 * Show the application dashboard.
 *
 * @return \Illuminate\Http\Response
 */
public function employee_evaluation()
{
    $users = User::where('id', '!=', auth()->id()->get());
    return view('employee_evaluation', compact('users'));
}
}
SignatoryController
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Division;
use App\Models\DivisionSection;
use App\Models\Signatory;
use App\Models\User;

class SignatoryController extends Controller
{
    public function divisions(Request $request) {
        if ($request->isMethod('get')) {
            return view('signatories.divisions', [
                'divisions' => Division::all()
            ]);
        }

        if ($request->isMethod('post')) {
            $division = Division::create([
                'name' => $request->name,
                'abbr' => $request->abbr,
                'active' => 1,
            ]);

            return redirect()->route('signatory-divisions');
        }
    }

    public function sections(Request $request) {
        if ($request->isMethod('get')) {
            return view('signatories.sections', [
                'divisions' => Division::all(),
                'sections' => DivisionSection::all(),
            ]);
        }

        if ($request->isMethod('post')) {
            // dd($request);
            $divisionSection = DivisionSection::create([
                'division_id' => $request->division,
                'name' => $request->name,
                'abbr' => $request->abbr,
                'office_site' => $request->office_site,
            ]);

            return redirect()->route('signatory-sections');
        }
    }

    public function assignments(Request $request) {
        if ($request->isMethod('get')) {
            return view('signatories.assign', [
                'users' => User::all(),
                'divisions' => Division::all(),
                'sections' => DivisionSection::all(),
            ]);
        }
    }
}

```

```

        if ($request->isMethod('post')) {
            if (Signatory::where('user_id', $request->user_id)->count() != 0) {
                // has record
                $signatory = Signatory::where('user_id', $request->user_id)-
>first();
                $signatory->division_section_id = $request->section;
                $signatory->save();
            } else {
                // no record
                Signatory::create([
                    'user_id' => $request->user_id,
                    'division_section_id' => $request->section,
                ]);
            }

            return redirect()->route('signatory-assignments');
        }
    }
}

```

EmployeeEvaluationController

```
<?php
```

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

```

use Illuminate\Http\Request;
use App\Models\EmployeeEvaluation;
use App\Models\EmployeeEvaluationDetails;
use App\Models\User;
use Illuminate\Support\Facades\Auth;

```

```
class EmployeeEvaluationController extends Controller
{
```

```

    public function initialFormView() {
        $users = User::All();
        return view('evaluation.create', [
            'users' => $users,
        ]);
    }

```

```

    public function createEvalForm(Request $request) {
        $eeval = EmployeeEvaluation::create([
            'createdby_user' => '1',
            'emp_status' => 'active',
            'employee' => $request->employee,
            'evaluator_section_head' => $request->evaluator_section_head,
            'evaluator_division_head' => $request->evaluator_division_head,
            'evaluator_hr' => $request->evaluator_hr,
            'year' => $request->year,
            'eval_status' => 'pending',
        ]);
    }

```

```

        EmployeeEvaluationDetails::create([
            'eeval_id' => $eeval->id,
            'userid' => $request->employee,
        ]);
    }

```

```

        return redirect()->route('eval-pending-forms');
    }

```

```

    public function viewPending(Request $request) {
        $evaluations = EmployeeEvaluation::where('eval_status', 'pending')-
>get();

        return view('evaluation.pending', [
            'evaluations' => $evaluations,
            'title' => 'Pending Evaluations',
        ]);
    }
}

```

```

    public function viewCompleted(Request $request) {
        $evaluations = EmployeeEvaluation::where('eval_status', 'completed')-
>get();
    }

```

```

        return view('evaluation.pending', [
            'evaluations' => $evaluations,
            'title' => 'Completed Evaluations',
        ]);
    }

```

```

    public function submitEmployee(Request $request) {
        $evaluation = EmployeeEvaluation::find($request->eval_id);
        $evaluation->details->pa_cost = $request->pa_cost;
        $evaluation->details->pa_exemplary = $request->pa_exemplary;
        $evaluation->details->save();
    }

```

```

        // quick update
        $status = $this->updateEvalStatus($evaluation);
    }

```

```

    if ($status == 'pending') {
        return redirect()->route('eval-pending-forms');
    }

```

```

        return redirect()->route('eval-completed-forms');
    }

```

```

    public function submitSectionHead(Request $request) {
        $evaluation = EmployeeEvaluation::find($request->eval_id);
        $evaluation->details->innovativeness = $request->innovativeness;
        $evaluation->details->ebcode_ethics = $request->ebcode_ethics;
        $evaluation->details->save();
    }

```

```

        // quick update
        $status = $this->updateEvalStatus($evaluation);
    }

```

```

    if ($status == 'pending') {
        return redirect()->route('eval-pending-forms');
    }

```

```

        return redirect()->route('eval-completed-forms');
    }

```

```

    public function submitHR(Request $request) {
        $evaluation = EmployeeEvaluation::find($request->eval_id);
        $evaluation->details->ebwork_ethics = $request->ebwork_ethics;
        $evaluation->details->save();
    }

```

```

        // quick update
        $status = $this->updateEvalStatus($evaluation);
    }

```

```

    if ($status == 'pending') {
        return redirect()->route('eval-pending-forms');
    }

```

```

        return redirect()->route('eval-completed-forms');
    }

```

```

    public function submitDivisionHead(Request $request) {
        $evaluation = EmployeeEvaluation::find($request->eval_id);
        $evaluation->details->ebp_relations = $request->ebp_relations;
        $evaluation->details->save();
    }

```

```

        // quick update
        $status = $this->updateEvalStatus($evaluation);
    }

```

```

    if ($status == 'pending') {
        return redirect()->route('eval-pending-forms');
    }

```

```

        return redirect()->route('eval-completed-forms');
    }

```

```

    public function updateEvalStatus($evaluation) {
        $status = 'pending';
    }

```



```

$fields = [
    $evaluation->details->pa_cost,
    $evaluation->details->pa_exemplary,
    $evaluation->details->innovativeness,
    $evaluation->details->ebcode_ethics,
    $evaluation->details->ebwork_ethics,
    $evaluation->details->ebp_relations,
];

// check if all fields are filled,
// then change status to 'completed'
if (min($fields) != -1) {
    error_log('evaluation has been completed');
    $evaluation->eval_status = 'completed';
    $evaluation->save();
    $status = 'completed';
}

$total = 0;

// calculate total
foreach ($fields as $field) {
    $total += ($field == -1) ? 0 : $field;
    error_log('total: ' . $total);
    $evaluation->details->total = $total;
    $evaluation->details->save();
}

return $status;
}
}

```

HomeController

```
<?php
```

```

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\dt_documents;
use App\dt_document_form;
use App\Models\laravel_logger_activity;
use App\Models\activity_log;
// use App\User;
use Auth;
use DB;

class HomeController extends Controller
{
    public function dashboard() {
        $routed = dt_documents::where('uid', Auth::user()->id)->count();
        //dd($routed);
        $open = dt_documents::where('status', 'open')->where('uid',
Auth::user()->id)->count();
        $close = dt_documents::where('status', 'closed')->where('uid',
Auth::user()->id)->count();

        //dd($documents[0]->dt_type->name);
        //dd($documents[0]->dt_type->name);
        return view('dashboard.index', compact('routed', 'open', 'close'));
    }

    public function documents() {
        // $documents = DB::table('dt_documents as dt')
        // ->select(DB::raw('SELECT name FROM dt_document_forms df
WHERE df.df_id=dt.d_type) as name'),
        //     DB::raw('COUNT(*) as total')
        // )
        // ->groupBy('d_type')
        // ->orderBy('total', 'DESC')
        // ->get();

        // $documents = DB::table('dt_document_forms as df')

```

```

//         ->join('dt_documents as dt', 'dt.d_type', 'df.df_id')
//         ->select('df.name', DB::raw('COUNT(dt.d_type)'))
//         ->groupBy('dt.d_type')
//         ->orderBy('total')
//         ->get();
// dd($documents);
// $docs = array();
// foreach ($documents as $d) {
//     $docs['forms'][] = $d->name;
//     $docs['form_num'][] = $d->total;
// }
// $docs = array();
// $sort_docs = array();

if(request()->input('ch')==='dhbar') {
    $user_document = dt_documents::select('d_type',
DB::raw('COUNT(*) as tcount'))->where('uid', Auth::user()->id)-
>groupBy('d_type')->get();

    foreach ($user_document as $ud) {
        $docs[] = array('form_name'=>$ud->dt_type->name,
'tcount'=>$ud->tcount);
    }

    usort($docs, function($a, $b) {
        return $b['tcount'] - $a['tcount'];
    });

    foreach ($docs as $d) {
        $sort_docs['forms'][] = $d['form_name'];
        $sort_docs['form_num'][] = $d['tcount'];
    }
} elseif(request()->input('ch')==='dpie') {
    $user_document = dt_documents::select('status',
DB::raw('COUNT(*) as tcount'))->where('uid', Auth::user()->id)-
>groupBy('status')->orderBy('status', 'DESC')->get();
    foreach ($user_document as $ud) {
        // $docs[] = array('form_name'=>$ud->dt_type->name,
'tcount'=>$ud->tcount);
        $sort_docs['status'][] = $ud->status;
        $sort_docs['tcount'][] = $ud->tcount;
    }
}

/**
$documents = dt_document_form::all();
// $documents = dt_document_form::orderBy('name', 'DESC')->get();

$docs = array();
$sort_docs = array();
foreach ($documents as $d) {
    // $docs['forms'][] = $d->name;
    // $docs['form_num'][] = $d->dt_documents->count();
    $docs[] = array('form_name'=>$d->name, 'tcount'=>$d-
>dt_documents->count());
    // $docs[] = array($d->dt_documents->count()=>$d->name);
    // $docs[] = (object) [$d->name=>$d->dt_documents->count()];
}
usort($docs, function($a, $b) {
    return $b['tcount'] - $a['tcount'];
});
// ($docs);

foreach ($docs as $d) {
    $sort_docs['forms'][] = $d['form_name'];
    $sort_docs['form_num'][] = $d['tcount'];
}
// dd($docs);
// $data = array('july');
*/
return response()->json($sort_docs);
}

public function __construct()
{

```

```

    }

    function show()
    {
        $data=laravel_logger_activity::all();
        return view('admin/logActivity',['laravel_logger_activity'=>$data]);
    }

    function show1()
    {
        $data=activity_log::all();
        return view('admin/activitylog',['activity_log'=>$data]);
    }
}

```

PostController

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Employee_Evaluation;

class PostController extends Controller
{
    public function postCreate()
    {
        return view('employee_evaluation');
    }

    public function postData(Request $request)
    {
        $input = $request->all();

        Post::create($input);

        dd('Post created successfully.');
```

```

    public function users()
    {
        $roles = array();

        foreach (Role::all() as $record) {
            $permissions = array();
            foreach ($record->permissions as $permission) {
                array_push($permissions, Permission::find($permission->permission_id->name);
            }

            array_push($roles, [
                'id' => $record->id,
                'name' => $record->name,
                'permissions' => $permissions
            ]);
        }

        $roles_dict = Role::all();

        return view(
            'admin.users',
            [
                'users' => User::all(),
                'roles' => $roles,
                'roles_dict' => $roles_dict
            ],
        );
    }
}

```

Add User Section

Add user Blade:

```

@extends('layouts.app')
@section('after-styles')
<link
href="https://cdn.datatables.net/1.10.21/css/jquery.dataTables.min.css"
rel="stylesheet">
<link
href="https://cdn.datatables.net/1.10.21/css/dataTables.bootstrap4.min.cs
s" rel="stylesheet">
<script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/Chart.js/2.1.2/Chart.min.js"></script>
@endsection
@section('content')
<body>

    @if (session('success_registration'))
        <h3 style="margin-left: 30px; color: green;">{{
session('success_registration') }}</h3>
    @endif

    <form action="{{ route('register') }}" method="POST">
        @csrf

        <div style="margin-right:24px; width:45%; float:right">

            <div class="card">
                <div class="card-header"><strong>USER NAME AND
PASSWORD</strong></div>
                <div class="card-body">
                    <div class="form-group">
                        <label for="company">Username</label>
                        <input class="form-control" name="username"
id="username" type="text">
                    </div>
                    <div class="form-group">
                        <label for="vat">Password</label>
                        <input class="form-control" name="password" id="password"
type="password">
                    </div>
                    <div class="form-group">
                        <label for="street">Confirm Password</label>
                        <input class="form-control" name="confirm_password"
id="confirm_password" type="password">
                    </div>
                    <div class="form-group">
                        <label for="street">Role</label>
                        <select class="custom-select custom-select-lg mb-3"
name="role">
                            <option selected>Select A Role</option>
                            @foreach ($roles as $role)
                                <option value="{{ $role->id }}">{{ $role->name
}}</option>
                            @endforeach
                        </select>
                    </div>
                    <div class="form-group">
                        <label for="street">Division</label>
                        <select class="custom-select custom-select-lg mb-3"
name="division">

```

```

        <option selected>Select Division</option>
        @foreach ($div as $divs)
            <option value="{{ $divs->abbr }}">{{ strtoupper($divs-
>abbr) }}</option>
        @endforeach
    </select>
</div>

    <button type="submit" class="btn btn-primary mb-
2">Submit</button>
    <button type="reset" class="btn btn-primary mb-
2">Reset</button>
</div>
</div>

<div style="padding-left:30px; width:50%">

    <div class="card">
        <div class="card-header"><strong>BASIC
INFORMATION</strong></div>
        <div class="card-body">
            <div class="form-group">
                <label for="company">First Name</label>
                <input class="form-control" name="first_name"
id="first_name" type="text" >
            </div>
            <div class="form-group">
                <label for="vat">Middle Name</label>
                <input class="form-control" name="middle_name"
id="middle_name" type="text" >
            </div>
            <div class="form-group">
                <label for="street">Last Name</label>
                <input class="form-control" name="last_name"
id="last_name" type="text" >
            </div>
            <div class="form-group">
                <label for="street">Email</label>
                <input class="form-control" name="email" id="email"
type="text" >
            </div>
        </div>
    </div>
</div>
</div>

```

```

<!-- <div class="col-6 col-sm-4 col-md-2 col-xl mb-3 mb-xl-0"
style="width:30%; float:right; margin-top:-80px; margin-right:10%">
    <button class="btn btn-block btn-success" type="submit">
        <h5>SUBMIT</h5>
    </button>
</div>

    <div class="col-6 col-sm-4 col-md-2 col-xl mb-3 mb-xl-
0"style="width:30%; float:right; margin-top:-20px; margin-right:10%">
        <button class="btn btn-block btn-danger active" type="reset" aria-
pressed="true">
            <h5>RESET</h5>
        </button>
    </div -->
</form>
</body>

@endsection

```

Logs Activity Section

Log Activity Installation Instruction.

1. From your projects root folder in terminal run:
composer require jeremykenedy/laravel-logger
2. Register the package
Laravel 5.5 and up Uses package auto discovery feature, no need to edit the config/app.php file.

Laravel 5.4 and below Register the package with laravel in config/app.php under providers with the following:

```

'providers' => [
    jeremykenedy\LaravelLogger\LaravelLoggerServiceProvider::class,
];

```

3. Run the migration to add the table to record the activities to:
php artisan migrate
4. Done. Check you database and see the table `laravel_logger_activity`.

For more information regarding the installation please check:
<https://github.com/jeremykenedy/laravel-logger>

User activity logs steps.

<https://spatie.be/docs/laravel-activitylog/v4/installation-and-setup>

After the installation and setup please follow and add these codes to the system.

For **User logs** blade:

```
@extends('layouts.app')
@section('after-styles')
<link
href="https://cdn.datatables.net/1.10.21/css/jquery.dataTables.min.css"
rel="stylesheet">
<link
href="https://cdn.datatables.net/1.10.21/css/dataTables.bootstrap4.min.css"
rel="stylesheet">
<script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/Chart.js/2.1.2/Chart.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/3.3.7/css/bootstrap.min.css" />
@endsection

@section('content')
<body>
<div class="container">
    <h1>User Logs</h1>
    <table class="table table-bordered">
        <tr>
            <th>No</th>
            <th>Subject</th>
            <th>Method</th>
            <th width="200px">User Agent</th>
            <th>IP address</th>
            <th>User Id</th>
            <th>Activity time </th>

        </tr>

        @foreach($laravel_logger_activity as $laravel_logger_activities => $log)
            <tr>
                <td>{{ $log->id }}</td>
                <td>{{ $log->description }}</td>
                <td><label class="label label-info">{{ $log->methodType }}</label></td>
                <td class="text-danger">{{ $log->userType }}</td>
                <td>{{ $log->ipAddress }}</td>
                <td>{{ $log->userId }}</td>
                <td>{{ $log->created_at }}</td>
            </tr>
        @endforeach

    </table>
</div>

</body>
@endsection
```

For **User Activity Log** blade:

```
@extends('layouts.app')
@section('after-styles')
<link
href="https://cdn.datatables.net/1.10.21/css/jquery.dataTables.min.css"
rel="stylesheet">
<link
href="https://cdn.datatables.net/1.10.21/css/dataTables.bootstrap4.min.css"
rel="stylesheet">
<script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/Chart.js/2.1.2/Chart.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/3.3.7/css/bootstrap.min.css" />
@endsection

@section('content')
<body>

<div class="container">
    <h1>User activity logs</h1>

    <table class="table table-bordered">
        <tr>
            <th>No</th>
            <th>Log name</th>
            <th width="150px">description</th>
            <th width="200px">Subject Type</th>
            <th width="100px">Subject Id</th>
            <th width="100px">Causer Id</th>
            <!--<th>Properties </th>-->
            <th width="200px">Log time </th>

        </tr>

        @foreach($activity_log as $activity_logs => $log)

            <tr>
                <td>{{ $log->id }}</td>
                <td>{{ $log->log_name }}</td>
                <td>{{ $log->description }}</td>
                <td>{{ $log->subject_type }}</td>
                <td>{{ $log->subject_id }}</td>
                <td>{{ $log->causer_id }}</td>
                <!-- <td>{{ $log->properties }}</td>-->
                <td>{{ $log->created_at }}</td>

            </tr>

        @endforeach
    </table>
</div>

</body>
@endsection
```

Models

laravel_logger_activity

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class laravel_logger_activity extends Model
{

    public $table = "laravel_logger_activity";
}
```

activity_log

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class activity_log extends Model
{
    public $table = "activity_log";
}
```

Division

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;

class Division extends Model
{

    use HasFactory,LogsActivity;

    protected $guarded = [];
    protected static $recordEvents = ['created'];
    protected static $logName = 'Divisions';
    protected static $logAttributes = ['name','abbr'];

    public function getDescriptionForEvent(string $eventName):string
    {
        return "User {$eventName} Division";
    }
    public function sections(){
        return $this->hasMany(DivisionSection::class);
    }
}
```

divisions

```
<?php
namespace App;

use Illuminate\Database\Eloquent\Model;
class divisions extends Model
{

    protected $primaryKey = 'id';

    public function user() {
        return $this->belongsToMany('App\Models\Access\User\User.php',
        'division_id');
    }
}
```

DivisionSection

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;

class DivisionSection extends Model
{
    use HasFactory, LogsActivity;

    protected $guarded = [];

    protected static $recordEvents = ['created'];

    protected static $logName = 'Division Section';

    protected static $logAttributes = ['name','abbr','office_site'];

    public function getDescriptionForEvent(string $eventName):string
    {
        return "User {$eventName} Section";
    }

    public function division(){
        return $this->belongsTo(Division::class);
    }
}
```

EmployeeEvaluation

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;
use App\Models\User;
use App\Models\EmployeeEvaluationDetails;

class EmployeeEvaluation extends Model
{
    use HasFactory, LogsActivity;

    protected $guarded = [];

    protected $table = 'employee_evaluation';

    protected static $logName = 'Evaluation';

    protected static $logAttributes = ['createdby_user', 'employee', 'year'];
}
```

```

public function getDescriptionForEvent(string $eventName):string
{
    return "Evaluation {$eventName}";
}

public function user() {
    return $this->belongsTo(User::class, 'employee');
}

public function details() {
    return $this->hasOne(EmployeeEvaluationDetails::class, 'eval_id');
}
}

```

EmployeeEvaluationDetails

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;

class EmployeeEvaluationDetails extends Model
{
    use HasFactory, LogsActivity;

    protected $guarded = [];

    protected $table = 'evaluation_details';

    protected static $logName = 'Evaluation details';

    protected static $logAttributes = ['eval_id', 'userid',
    'pa_cost', 'pa_exemplary', 'innovativeness', 'ebcode_ethics', 'ebwork_ethics',
    'ebp_relations', 'total', 'created_at', 'updated_at'];

    public function getDescriptionForEvent(string $eventName):string
    {
        return "Evaluation Rate {$eventName}";
    }

    public function user() {
        return $this->belongsTo(User::class, 'employee');
    }
}

```

Permission

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;

class Permission extends Model
{
    use LogsActivity;

    protected $guarded = [];

    protected static $logName = 'Permission';

    protected static $logAttributes = ['name'];

    public $timestamps = false;

    public function logUnguarded(): LogOption
    {
        return LogOptions::defaults();
    }
}

```

```

public function getDescriptionForEvent(string $eventName):string
{
    return "Permission {$eventName}";
}
}

```

Role

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;

class Role extends Model
{
    use LogsActivity;

    protected $guarded = [];
    protected static $logName = 'Roles';
    protected static $logAttributes = ['name'];

    public $timestamps = false;
    public function permissions() {
        return $this->hasMany(RolePermission::class);
    }
    public function getPermissionRecord($permission_id) {
        return Permission::find($permission_id);
    }
    public function logUnguarded(): LogOption
    {
        return LogOptions::defaults();
    }
    public function getDescriptionForEvent(string $eventName):string
    {
        return "Role {$eventName}";
    }
}

```

RolePermission

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class RolePermission extends Model
{
    protected $guarded = [];
    public $timestamps = false;
    public function role() {
        return $this->belongsTo(Role::class);
    }
    public function permission() {
        return $this->belongsTo(Permission::class);
    }
}

```

RoleUser

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

```

```
class RoleUser extends Model
{
```

```
    protected $guarded = [];
    public $timestamps = false;
```

```
    public function user() {
        return $this->belongsTo(User::class);
    }
    public function role() {
        return $this->belongsTo(Role::class);
    }
}
```

Signatory

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Spatie\Activitylog\LogOptions;
use Spatie\Activitylog\Traits\LogsActivity;
```

```
class Signatory extends Model
{
```

```
    use HasFactory, LogsActivity;
```

```
    protected $guarded = [];
```

```
    public $table = "signatories";
```

```
    protected static $recordEvents = ['created'];
```

```
    protected static $logName = 'Signatory';
```

```
    public function getDescriptionForEvent(string $eventName):string
    {
        return "User {$eventName}";
    }
}
```

```
    public function divisionSection() {
        return $this->belongsTo(DivisionSection::class);
    }
}
```

```
    public function user() {
        return $this->belongsTo(User::class);
    }
}
```

UserRole

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Model;
```

```
class UserRole extends Model
{
```

```
    protected $guarded = [];
```

```
    public $timestamps = false;
```

```
    public function user() {
        return $this->belongsTo(User::class);
    }
}
```

```
    public function role() {
        return $this->belongsTo(Role::class);
    }
}
```

```
}
```

User

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use App\Traits\RoleRelationship;
use Spatie\Activitylog\Traits\LogsActivity;
```

```
class User extends Authenticatable
```

```
{
    use Notifiable, RoleRelationship, LogsActivity;
```

```
    /**
     * The attributes that are mass assignable.
```

```
     *
     * @var array
     */
```

```
    protected $fillable = [
        'first_name', 'last_name', 'middle_name', 'username', 'email', 'division',
        'password',
    ];
    protected static $logAttributes = ['username', 'email', 'division'];
```

```
    protected static $recordEvents = ['created'];
```

```
    protected static $logName = 'User';
```

```
    public function getDescriptionForEvent(string $eventName):string
    {
        return "User {$eventName}";
    }
}
```

```
    /**
     * The attributes that should be hidden for arrays.
```

```
     *
     * @var array
     */
```

```
    protected $hidden = [
        'password', 'remember_token',
    ];
```

```
    public function role() {
        return $this->hasOne(RoleUser::class);
    }
}
```

```
    public function division(){
        return $this->hasOne(Division::class);
    }
}
```

```
    public function signatory() {
        return $this->hasOne(Signatory::class);
    }
}
```

```
    public function roleName() {
        $role_id = $this->role->role_id;
        return Role::find($role_id)->name;
    }
}
```

```
    public function roles() {
        return $this->hasMany(RoleUser::class);
    }
}
```

```
    public function getLeavePoints() {
        return LeavePointsLog::getLatestPoints($this->id);
    }
}
```

```
}
```

Migrations

create_users_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('username');
            $table->string('first_name');
            $table->string('middle_name');
            $table->string('last_name');
            $table->string('division');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

create_roles_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateRolesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('roles', function (Blueprint $table) {
            $table->id();
            $table->string('name');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('roles');
    }
}
```

create_permissions_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePermissionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('permissions', function (Blueprint $table) {
            $table->id();
            $table->string('name');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('permissions');
    }
}
```

create_role_user_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateRoleUserTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('role_users', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id');
            $table->foreignId('role_id');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('role_user');
    }
}
```


create_role_permission_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateRolePermissionTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('role_permissions', function (Blueprint $table) {
            $table->id();
            $table->foreignId('permission_id');
            $table->foreignId('role_id');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('role_permission');
    }
}
```

create_activity_log_table

<?php

```
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
```

```
class CreateActivityLogTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::connection(config('activitylog.database_connection'))->create(config('activitylog.table_name'), function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('log_name')->nullable();
            $table->text('description');
            $table->nullableMorphs('subject', 'subject');
            $table->nullableMorphs('causer', 'causer');
            $table->json('properties')->nullable();
            $table->timestamps();
            $table->index('log_name');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::connection(config('activitylog.database_connection'))->dropIfExists(config('activitylog.table_name'));
    }
}
```

create_divisions_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateDivisionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('divisions', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('abbr');
            $table->enum('active', [0, 1]);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('divisions');
    }
}
```

create_employees_table

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateEmployeesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('employees', function (Blueprint $table) {

            $table->increments('emp_id');
            $table->string('first_name');
            $table->string('middle_name');
            $table->string('last_name');
            $table->string('division');
            //$table->string('role');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('employees');
    }
}
```

create_division_sections_table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateDivisionSectionsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('division_sections', function (Blueprint $table) {
            $table->id();
            $table->integer('division_id');
            $table->string('name');
            $table->string('abbr');
            $table->string('office_site');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('division_sections');
    }
}
```

create_signatories_table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateSignatoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('signatories', function (Blueprint $table) {
            $table->id();
            $table->integer('user_id');
            $table->integer('division_section_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('signatories');
    }
}
```

create_employee_evaluation_table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateEmployeeEvaluationTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('employee_evaluation', function (Blueprint $table) {
            $table->id();
            $table->integer('createdby_user');
            $table->string('emp_status');
            $table->integer('employee');
            $table->string('evaluator_section_head');
            $table->string('evaluator_division_head');
            $table->string('evaluator_hr');
            $table->string('year');
            $table->string('eval_status');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('employee_evaluation');
    }
}
```

create_evaluation_details_table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateEvaluationDetailsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('evaluation_details', function (Blueprint $table) {
            $table->id();
            $table->integer('eeval_id');
            $table->integer('userid');
            $table->integer('pa_cost')->default('-1');
            $table->integer('pa_exemplary')->default('-1');
            $table->integer('innovativeness')->default('-1');
        });
    }
}
```

```
$table->integer('ebcode_ethics')->default('-1');
$table->integer('ebwork_ethics')->default('-1');
$table->integer('ebp_relations')->default('-1');
$table->integer('total')->default('0');
$table->timestamps();
});
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('evaluation_details');
}
}
```

End of Source Code.