**SOURCE CODE**

**NotifyDue.php**

**<?php**

**namespace App\Console\Commands;**

**use App\Events\DueTask;**

**use App\Http\Traits\InstrumentsTrait;**

**use App\Models\Notification;**

**use App\Models\TaskAssign;**

**use Carbon\Carbon;**

**use Illuminate\Console\Command;**

**class NotifyDue extends Command**

**{**

**use InstrumentsTrait;**

**/\*\***

**\* The name and signature of the console command.**

**\***

**\* @var string**

**\*/**

**protected $signature = 'notify:due';**

**/\*\***

**\* The console command description.**

**\***

**\* @var string**

**\*/**

**protected $description = 'Send Notification to user';**

**/\*\***

**\* Execute the console command.**

**\***

**\* @return int**

**\*/**

**public function handle()**

**{**

**$dateNow = Carbon::now()->format('Y-m-d');**

**$dueTask = TaskAssign::where('due', $dateNow)**

**->whereNull('notified')**

**->get(['id', 'instrumentId', 'userId', 'accredId'])**

**->map(function ($due) {**

**$parent = $this->getParents($due->instrumentId, true, null);**

**$accred = $due->accreditation;**

**if(count($parent) > 0){**

**$due->area = $parent[count($parent)-1]->only('title');**

**$due->parameter = $parent[count($parent)-2]->only('title');**

**}else{**

**$due->area = null;**

**$due->parameter = null;**

**}**

**$due->inst = $due->instrument;**

**$due->program = $accred->taggedPrograms->only('abbreviation');**

**$due->level = $accred->getLevelInstrument->only('title');**

**unset($due->instrument, $due->accreditation);**

**return $due;**

**});**

**//echo $dueTask;**

**foreach ($dueTask as $value) {**

**$notif = Notification::create([**

**'userId'=>$value->userId,**

**'action'=>'due',**

**'details' =>**

**'You have one due task today in '.$value->program['abbreviation'].' / '.$value->level['title'].**

**' / '.$value->area['title'].' / '.$value->parameter['title'].' / '.$value->inst['title']**

**]);**

**$task = TaskAssign::find($value->id);**

**$task->notified = true;**

**$task->save();**

**DueTask::dispatch($notif);**

**//Duedates::dispatch([$value->userId, $notif]);**

**}**

**}**

**}**

**ScheduledSurvey.php**

**<?php**

**namespace App\Console\Commands;**

**use App\Events\UpdateAccred;**

**use App\Models\Accreditation;**

**use App\Models\User;**

**use Carbon\Carbon;**

**use Illuminate\Console\Command;**

**class ScheduledSurvey extends Command**

**{**

**/\*\***

**\* The name and signature of the console command.**

**\***

**\* @var string**

**\*/**

**protected $signature = 'survey:open';**

**/\*\***

**\* The console command description.**

**\***

**\* @var string**

**\*/**

**protected $description = 'Open scheduled accreditation survey';**

**/\*\***

**\* Execute the console command.**

**\***

**\* @return int**

**\*/**

**public function handle()**

**{**

**$dateNow = Carbon::now()->timezone('Asia/Manila')->format('Y-m-d');**

**Accreditation::where('date\_self\_survey', $dateNow)**

**->whereNull('survey')**

**->whereNull('status')**

**->whereNull('selfsurvey')**

**->update(['survey' => 2]);**

**Accreditation::where('date\_actual\_survey', $dateNow)**

**->whereNull('survey')**

**->update(['survey' => 1, 'restrict' => true]);**

**foreach ([1, 6] as $value) {**

**UpdateAccred::dispatch(collect([**

**'auth' => $value,**

**'self' => Accreditation::where('date\_self\_survey', $dateNow)**

**->where('survey', 2)**

**->whereNull('status')**

**->whereNull('selfsurvey')**

**->get(['id', 'date\_actual\_survey', 'date\_self\_survey', 'instrumentId', 'programId', 'restrict', 'selfsurvey', 'survey'])**

**->map(function ($accred) {**

**return collect([**

**'id' => $accred->id,**

**'date\_actual\_survey' => $accred->date\_actual\_survey,**

**'date\_self\_survey' => $accred->date\_self\_survey,**

**'instrumentId' => $accred->instrumentId,**

**'programId' => $accred->programId,**

**'restrict' => $accred->restrict,**

**'selfsurvey' => $accred->selfsurvey,**

**'survey' => $accred->survey,**

**'program' => $accred->taggedPrograms->only('abbreviation')['abbreviation'],**

**'title' => $accred->getLevelInstrument->only('title')['title']**

**]);**

**}),**

**'actual' => Accreditation::where('date\_actual\_survey', $dateNow)**

**->where('survey', 1)**

**->whereNull('status')**

**->get(['id', 'date\_actual\_survey', 'date\_self\_survey', 'instrumentId', 'programId', 'restrict', 'selfsurvey', 'survey'])**

**->map(function ($accred) {**

**return collect([**

**'id' => $accred->id,**

**'date\_actual\_survey' => $accred->date\_actual\_survey,**

**'date\_self\_survey' => $accred->date\_self\_survey,**

**'instrumentId' => $accred->instrumentId,**

**'programId' => $accred->programId,**

**'restrict' => $accred->restrict,**

**'selfsurvey' => $accred->selfsurvey,**

**'survey' => $accred->survey,**

**'program' => $accred->taggedPrograms->only('abbreviation')['abbreviation'],**

**'title' => $accred->getLevelInstrument->only('title')['title']**

**]);**

**}),**

**]));**

**}**

**}**

**}**

**Kernel.php**

**<?php**

**namespace App\Console;**

**use Illuminate\Console\Scheduling\Schedule;**

**use Illuminate\Foundation\Console\Kernel as ConsoleKernel;**

**class Kernel extends ConsoleKernel**

**{**

**/\*\***

**\* Define the application's command schedule.**

**\***

**\* @param \Illuminate\Console\Scheduling\Schedule $schedule**

**\* @return void**

**\*/**

**protected function schedule(Schedule $schedule)**

**{**

**$schedule->command('notify:due')->hourly();**

**$schedule->command('survey:open')->dailyAt('06:00');**

**}**

**/\*\***

**\* Register the commands for the application.**

**\***

**\* @return void**

**\*/**

**protected function commands()**

**{**

**$this->load(\_\_DIR\_\_.'/Commands');**

**require base\_path('routes/console.php');**

**}**

**}**

**Attachment.php**

**<?php**

**namespace App\Events;**

**use Illuminate\Broadcasting\Channel;**

**use Illuminate\Broadcasting\InteractsWithSockets;**

**use Illuminate\Broadcasting\PresenceChannel;**

**use Illuminate\Broadcasting\PrivateChannel;**

**use Illuminate\Contracts\Broadcasting\ShouldBroadcast;**

**use Illuminate\Foundation\Events\Dispatchable;**

**use Illuminate\Queue\SerializesModels;**

**use Illuminate\Support\Facades\Auth;**

**class Attachment implements ShouldBroadcast**

**{**

**use Dispatchable, InteractsWithSockets, SerializesModels;**

**public $attachment;**

**public function \_\_construct($attachment)**

**{**

**$this->attachment = $attachment;**

**}**

**/\*\***

**\* Get the channels the event should broadcast on.**

**\***

**\* @return \Illuminate\Broadcasting\Channel|array**

**\*/**

**public function broadcastOn()**

**{**

**return new PrivateChannel('attach.'.Auth::id());**

**}**

**}**

**DocumentComment.php**

**<?php**

**namespace App\Events;**

**use Illuminate\Broadcasting\Channel;**

**use Illuminate\Broadcasting\InteractsWithSockets;**

**use Illuminate\Broadcasting\PresenceChannel;**

**use Illuminate\Broadcasting\PrivateChannel;**

**use Illuminate\Contracts\Broadcasting\ShouldBroadcast;**

**use Illuminate\Foundation\Events\Dispatchable;**

**use Illuminate\Queue\SerializesModels;**

**class DocumentComment implements ShouldBroadcast**

**{**

**use Dispatchable, InteractsWithSockets, SerializesModels;**

**public $comment;**

**public function \_\_construct($comment)**

**{**

**$this->comment = $comment;**

**}**

**/\*\***

**\* Get the channels the event should broadcast on.**

**\***

**\* @return \Illuminate\Broadcasting\Channel|array**

**\*/**

**public function broadcastOn()**

**{**

**return new PrivateChannel('docuComment.'.$this->comment['userId']);**

**}**

**}**

**DocumentReview.php**

**<?php**

**namespace App\Events;**

**use Illuminate\Broadcasting\Channel;**

**use Illuminate\Broadcasting\InteractsWithSockets;**

**use Illuminate\Broadcasting\PresenceChannel;**

**use Illuminate\Broadcasting\PrivateChannel;**

**use Illuminate\Contracts\Broadcasting\ShouldBroadcast;**

**use Illuminate\Foundation\Events\Dispatchable;**

**use Illuminate\Queue\SerializesModels;**

**class DocumentReview implements ShouldBroadcast**

**{**

**use Dispatchable, InteractsWithSockets, SerializesModels;**

**public $review;**

**public function \_\_construct($review)**

**{**

**$this->review = $review;**

**}**

**/\*\***

**\* Get the channels the event should broadcast on.**

**\***

**\* @return \Illuminate\Broadcasting\Channel|array**

**\*/**

**public function broadcastOn()**

**{**

**return new PrivateChannel('review.'.$this->review['userId']);**

**}**

**}**

**DocumentUpload.php**

**<?php**

**namespace App\Events;**

**use Illuminate\Broadcasting\Channel;**

**use Illuminate\Broadcasting\InteractsWithSockets;**

**use Illuminate\Broadcasting\PresenceChannel;**

**use Illuminate\Broadcasting\PrivateChannel;**

**use Illuminate\Contracts\Broadcasting\ShouldBroadcast;**

**use Illuminate\Foundation\Events\Dispatchable;**

**use Illuminate\Queue\SerializesModels;**

**class DocumentUpload implements ShouldBroadcast**

**{**

**use Dispatchable, InteractsWithSockets, SerializesModels;**

**public $upload;**

**public function \_\_construct($upload)**

**{**

**$this->upload = $upload;**

**}**

**/\*\***

**\* Get the channels the event should broadcast on.**

**\***

**\* @return \Illuminate\Broadcasting\Channel|array**

**\*/**

**public function broadcastOn()**

**{**

**return new PrivateChannel('docupload.'.$this->upload['userId']);**

**}**

**}**