**🎯 Các bảng cần thiết:**

1. **Users** – Laravel có sẵn
2. **Groups** – Nhóm làm việc
3. **Group Members** – Thành viên trong nhóm
4. **Tasks** – Nhiệm vụ
5. **Task Assignees** – Ai được giao nhiệm vụ
6. **Notifications** – Nhắc nhở deadline

**📦 Cách tạo bảng (migration)**

**1. Tạo bảng groups**

bash

CopyEdit

php artisan make:migration create\_groups\_table

php

CopyEdit

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

$table->id();

$table->string('name');

$table->text('description')->nullable();

$table->foreignId('created\_by')->constrained('users')->onDelete('cascade');

$table->timestamps();

});

**2. Tạo bảng group\_members**

bash

CopyEdit

php artisan make:migration create\_group\_members\_table

php

CopyEdit

Schema::create('group\_members', function (Blueprint $table) {

$table->id();

$table->foreignId('group\_id')->constrained()->onDelete('cascade');

$table->foreignId('user\_id')->constrained()->onDelete('cascade');

$table->enum('role', ['admin', 'member'])->default('member');

$table->timestamp('joined\_at')->useCurrent();

});

**3. Tạo bảng tasks**

bash

CopyEdit

php artisan make:migration create\_tasks\_table

php

CopyEdit

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

$table->id();

$table->foreignId('group\_id')->constrained()->onDelete('cascade');

$table->string('title');

$table->text('description')->nullable();

$table->enum('status', ['todo', 'in\_progress', 'done'])->default('todo');

$table->dateTime('due\_date')->nullable();

$table->foreignId('created\_by')->constrained('users')->onDelete('cascade');

$table->timestamps();

});

**4. Tạo bảng task\_assignees**

bash

CopyEdit

php artisan make:migration create\_task\_assignees\_table

php

CopyEdit

Schema::create('task\_assignees', function (Blueprint $table) {

$table->id();

$table->foreignId('task\_id')->constrained()->onDelete('cascade');

$table->foreignId('user\_id')->constrained()->onDelete('cascade');

});

**5. Tạo bảng notifications**

bash

CopyEdit

php artisan make:migration create\_notifications\_table

php

CopyEdit

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

$table->id();

$table->foreignId('user\_id')->constrained()->onDelete('cascade');

$table->foreignId('task\_id')->constrained()->onDelete('cascade');

$table->enum('type', ['email', 'web']);

$table->string('message');

$table->timestamp('sent\_at')->nullable();

});

Bạn đã cài Laravel trong XAMPP chưa, để mình hướng dẫn cấu hình kết nối database luôn?

Tuyệt! Dưới đây là hướng dẫn tạo **Model + các quan hệ Eloquent** tương ứng với từng bảng bạn đã tạo.

**🧩 1. Model: Group**

bash

CopyEdit

php artisan make:model Group

php

CopyEdit

// app/Models/Group.php

class Group extends Model

{

use HasFactory;

protected $fillable = ['name', 'description', 'created\_by'];

public function members()

{

return $this->hasMany(GroupMember::class);

}

public function creator()

{

return $this->belongsTo(User::class, 'created\_by');

}

public function tasks()

{

return $this->hasMany(Task::class);

}

}

**🧩 2. Model: GroupMember**

bash

CopyEdit

php artisan make:model GroupMember

php

CopyEdit

// app/Models/GroupMember.php

class GroupMember extends Model

{

use HasFactory;

public $timestamps = false;

protected $fillable = ['group\_id', 'user\_id', 'role', 'joined\_at'];

public function group()

{

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

}

public function user()

{

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

}

}

**🧩 3. Model: Task**

bash

CopyEdit

php artisan make:model Task

php

CopyEdit

// app/Models/Task.php

class Task extends Model

{

use HasFactory;

protected $fillable = ['group\_id', 'title', 'description', 'status', 'due\_date', 'created\_by'];

public function group()

{

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

}

public function creator()

{

return $this->belongsTo(User::class, 'created\_by');

}

public function assignees()

{

return $this->hasMany(TaskAssignee::class);

}

public function notifications()

{

return $this->hasMany(Notification::class);

}

}

**🧩 4. Model: TaskAssignee**

bash

CopyEdit

php artisan make:model TaskAssignee

php

CopyEdit

// app/Models/TaskAssignee.php

class TaskAssignee extends Model

{

use HasFactory;

public $timestamps = false;

protected $fillable = ['task\_id', 'user\_id'];

public function task()

{

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

}

public function user()

{

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

}

}

**🧩 5. Model: Notification**

bash

CopyEdit

php artisan make:model Notification

php

CopyEdit

// app/Models/Notification.php

class Notification extends Model

{

use HasFactory;

public $timestamps = false;

protected $fillable = ['user\_id', 'task\_id', 'type', 'message', 'sent\_at'];

public function user()

{

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

}

public function task()

{

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

}

}

**🧩 6. Cập nhật User model**

Thêm các quan hệ liên quan trong User.php:

php

CopyEdit

public function createdGroups()

{

return $this->hasMany(Group::class, 'created\_by');

}

public function groupMemberships()

{

return $this->hasMany(GroupMember::class);

}

public function createdTasks()

{

return $this->hasMany(Task::class, 'created\_by');

}

public function assignedTasks()

{

return $this->hasMany(TaskAssignee::class);

}

public function notifications()

{

return $this->hasMany(Notification::class);

}

💡 **Gợi ý tiếp theo**:  
Bạn có muốn mình hướng dẫn tạo controller, route và logic cơ bản CRUD cho nhóm (Group) hoặc nhiệm vụ (Task) không?

**Tiếp theo: Tạo Controller + CRUD cho Group (quản lý nhóm)**

**📦 Bước 1: Tạo GroupController**

bash

CopyEdit

php artisan make:controller GroupController

**📁 app/Http/Controllers/GroupController.php**

php

CopyEdit

namespace App\Http\Controllers;

use App\Models\Group;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class GroupController extends Controller

{

public function index()

{

$groups = Group::with('members')->where('created\_by', Auth::id())->get();

return response()->json($groups);

}

public function store(Request $request)

{

$request->validate([

'name' => 'required|string|max:255',

'description' => 'nullable|string',

]);

$group = Group::create([

'name' => $request->name,

'description' => $request->description,

'created\_by' => Auth::id(),

]);

return response()->json($group, 201);

}

public function show($id)

{

$group = Group::with('members.user')->findOrFail($id);

return response()->json($group);

}

public function update(Request $request, $id)

{

$group = Group::findOrFail($id);

if ($group->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

$group->update($request->only(['name', 'description']));

return response()->json($group);

}

public function destroy($id)

{

$group = Group::findOrFail($id);

if ($group->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

$group->delete();

return response()->json(['message' => 'Group deleted']);

}

}

**🛣️ Bước 2: Thêm route vào routes/api.php (hoặc web.php nếu không dùng API)**

php

CopyEdit

use App\Http\Controllers\GroupController;

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

Route::apiResource('groups', GroupController::class);

});

🎯 Sau bước này, bạn đã có thể:

* GET /api/groups — xem danh sách nhóm
* POST /api/groups — tạo nhóm mới
* GET /api/groups/{id} — xem chi tiết nhóm
* PUT /api/groups/{id} — cập nhật nhóm
* DELETE /api/groups/{id} — xóa nhóm

Bạn muốn mình làm tiếp phần CRUD cho Task, hay phần quản lý thành viên nhóm (GroupMember) tiếp theo?

**✅ Tạo TaskController & thêm logic CRUD**

**📦 Bước 1: Tạo controller**

bash

CopyEdit

php artisan make:controller TaskController

**📁 app/Http/Controllers/TaskController.php**

php

CopyEdit

namespace App\Http\Controllers;

use App\Models\Task;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class TaskController extends Controller

{

public function index()

{

// Hiển thị tất cả task của user (có thể lọc theo group nếu muốn)

$tasks = Task::where('created\_by', Auth::id())->with(['group', 'assignees.user'])->get();

return response()->json($tasks);

}

public function store(Request $request)

{

$request->validate([

'group\_id' => 'required|exists:groups,id',

'title' => 'required|string|max:255',

'description' => 'nullable|string',

'due\_date' => 'nullable|date',

]);

$task = Task::create([

'group\_id' => $request->group\_id,

'title' => $request->title,

'description' => $request->description,

'due\_date' => $request->due\_date,

'created\_by' => Auth::id(),

]);

return response()->json($task, 201);

}

public function show($id)

{

$task = Task::with(['group', 'assignees.user'])->findOrFail($id);

return response()->json($task);

}

public function update(Request $request, $id)

{

$task = Task::findOrFail($id);

if ($task->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

$task->update($request->only(['title', 'description', 'status', 'due\_date']));

return response()->json($task);

}

public function destroy($id)

{

$task = Task::findOrFail($id);

if ($task->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

$task->delete();

return response()->json(['message' => 'Task deleted']);

}

}

**🛣️ Bước 2: Thêm routes trong routes/api.php**

php

CopyEdit

use App\Http\Controllers\TaskController;

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

Route::apiResource('tasks', TaskController::class);

});

📌 Tính đến giờ, bạn đã có:

* CRUD cho nhóm (Group)
* CRUD cho nhiệm vụ (Task)
* Cấu trúc dữ liệu chặt chẽ với quan hệ User, Group, TaskAssignee, v.v.

🎯 Tiếp theo, bạn có muốn mình hướng dẫn:

1. Thêm/xóa người được giao nhiệm vụ (TaskAssignee), hay
2. Quản lý thành viên nhóm (GroupMember), hay
3. Tự động tạo Notification khi đến gần deadline?

Bạn chọn hướng nào để mình làm tiếp?

Tuyệt! Bây giờ mình sẽ hướng dẫn bạn xây dựng **chức năng thêm và xóa người được giao nhiệm vụ** (quản lý bảng task\_assignees).

**✅ Tạo TaskAssigneeController**

**📦 Bước 1: Tạo controller**

bash

CopyEdit

php artisan make:controller TaskAssigneeController

**📁 app/Http/Controllers/TaskAssigneeController.php**

php

CopyEdit

namespace App\Http\Controllers;

use App\Models\Task;

use App\Models\TaskAssignee;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class TaskAssigneeController extends Controller

{

public function store(Request $request)

{

$request->validate([

'task\_id' => 'required|exists:tasks,id',

'user\_id' => 'required|exists:users,id',

]);

$task = Task::findOrFail($request->task\_id);

if ($task->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

// Kiểm tra trùng

$exists = TaskAssignee::where('task\_id', $request->task\_id)

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

->exists();

if ($exists) {

return response()->json(['message' => 'User already assigned'], 409);

}

$assignee = TaskAssignee::create([

'task\_id' => $request->task\_id,

'user\_id' => $request->user\_id,

]);

return response()->json($assignee, 201);

}

public function destroy(Request $request)

{

$request->validate([

'task\_id' => 'required|exists:tasks,id',

'user\_id' => 'required|exists:users,id',

]);

$task = Task::findOrFail($request->task\_id);

if ($task->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

TaskAssignee::where('task\_id', $request->task\_id)

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

->delete();

return response()->json(['message' => 'User unassigned']);

}

}

**🛣️ Bước 2: Thêm routes trong routes/api.php**

php

CopyEdit

use App\Http\Controllers\TaskAssigneeController;

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

Route::post('task-assignees', [TaskAssigneeController::class, 'store']); // Giao nhiệm vụ

Route::delete('task-assignees', [TaskAssigneeController::class, 'destroy']); // Gỡ nhiệm vụ

});

**📌 Request mẫu gửi từ Frontend (POSTMAN hoặc JS frontend):**

* **Giao nhiệm vụ**:

http

CopyEdit

POST /api/task-assignees

Content-Type: application/json

Authorization: Bearer <token>

{

"task\_id": 3,

"user\_id": 7

}

* **Gỡ nhiệm vụ**:

http

CopyEdit

DELETE /api/task-assignees

Content-Type: application/json

Authorization: Bearer <token>

{

"task\_id": 3,

"user\_id": 7

}

Vậy là bạn đã có thể **quản lý danh sách người được giao cho mỗi task** một cách linh hoạt rồi.

Bạn có muốn mình hướng dẫn thêm cách **lấy danh sách ai đang được giao cho một task**, hoặc tiếp tục với **quản lý thành viên nhóm (GroupMember)**?

4o

**You said:**

oke

**ChatGPT said:**

Tốt lắm! Vì bạn đã xong phần quản lý người được giao nhiệm vụ, tiếp theo mình sẽ hướng dẫn bạn **quản lý thành viên nhóm (GroupMember)** – tức là thêm và xóa user khỏi một nhóm.

**✅ Tạo GroupMemberController**

**📦 Bước 1: Tạo controller**

bash

CopyEdit

php artisan make:controller GroupMemberController

**📁 app/Http/Controllers/GroupMemberController.php**

php

CopyEdit

namespace App\Http\Controllers;

use App\Models\Group;

use App\Models\GroupMember;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class GroupMemberController extends Controller

{

public function store(Request $request)

{

$request->validate([

'group\_id' => 'required|exists:groups,id',

'user\_id' => 'required|exists:users,id',

'role' => 'in:admin,member',

]);

$group = Group::findOrFail($request->group\_id);

// Chỉ người tạo nhóm mới có quyền thêm thành viên

if ($group->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

$exists = GroupMember::where('group\_id', $request->group\_id)

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

->exists();

if ($exists) {

return response()->json(['message' => 'User already in group'], 409);

}

$member = GroupMember::create([

'group\_id' => $request->group\_id,

'user\_id' => $request->user\_id,

'role' => $request->role ?? 'member',

]);

return response()->json($member, 201);

}

public function destroy(Request $request)

{

$request->validate([

'group\_id' => 'required|exists:groups,id',

'user\_id' => 'required|exists:users,id',

]);

$group = Group::findOrFail($request->group\_id);

if ($group->created\_by !== Auth::id()) {

return response()->json(['error' => 'Unauthorized'], 403);

}

GroupMember::where('group\_id', $request->group\_id)

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

->delete();

return response()->json(['message' => 'User removed from group']);

}

}

**🛣️ Bước 2: Thêm routes vào routes/api.php**

php

CopyEdit

use App\Http\Controllers\GroupMemberController;

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

Route::post('group-members', [GroupMemberController::class, 'store']); // Thêm thành viên

Route::delete('group-members', [GroupMemberController::class, 'destroy']); // Xóa thành viên

});

**🧪 Gợi ý test:**

* **Thêm thành viên vào nhóm**:

http

CopyEdit

POST /api/group-members

Content-Type: application/json

Authorization: Bearer <token>

{

"group\_id": 5,

"user\_id": 8,

"role": "member"

}

* **Xóa thành viên khỏi nhóm**:

http

CopyEdit

DELETE /api/group-members

Content-Type: application/json

Authorization: Bearer <token>

{

"group\_id": 5,

"user\_id": 8

}

Bạn đã có hệ thống:

* Nhóm & thành viên
* Nhiệm vụ & người được giao
* Bảo mật dựa theo người tạo (Auth::id())

👉 Tiếp theo, bạn muốn mình hướng dẫn:

1. **Gửi thông báo (notifications) khi sắp tới deadline**, hay
2. **Tạo giao diện (Blade hoặc Vue/React)**, hay
3. **Lọc, tìm kiếm, phân trang danh sách nhiệm vụ/nhóm?**

Tuyệt vời! Mình sẽ hướng dẫn bạn **tự động gửi thông báo (notifications) khi nhiệm vụ sắp đến hạn**.

**🧠 Ý tưởng tổng thể:**

1. Một **Laravel Command (scheduler)** chạy mỗi giờ (hoặc tùy bạn).
2. Nó sẽ kiểm tra các task có due\_date sắp đến (VD: trong vòng 24 giờ tới).
3. Gửi thông báo cho tất cả TaskAssignee của task đó.

**✅ Bước 1: Tạo Laravel Command**

bash

CopyEdit

php artisan make:command NotifyUpcomingTasks

**📁 app/Console/Commands/NotifyUpcomingTasks.php**

php

CopyEdit

namespace App\Console\Commands;

use App\Models\Task;

use App\Models\Notification;

use Illuminate\Console\Command;

use Illuminate\Support\Carbon;

class NotifyUpcomingTasks extends Command

{

protected $signature = 'tasks:notify-upcoming';

protected $description = 'Gửi thông báo cho các nhiệm vụ sắp hết hạn trong 24h tới';

public function handle()

{

$now = Carbon::now();

$soon = $now->copy()->addHours(24);

$tasks = Task::with('assignees')

->whereNotNull('due\_date')

->whereBetween('due\_date', [$now, $soon])

->get();

foreach ($tasks as $task) {

foreach ($task->assignees as $assignee) {

// Kiểm tra xem đã gửi chưa

$alreadyNotified = Notification::where('task\_id', $task->id)

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

->whereNotNull('sent\_at')

->whereDate('sent\_at', $now->toDateString())

->exists();

if (!$alreadyNotified) {

Notification::create([

'task\_id' => $task->id,

'user\_id' => $assignee->user\_id,

'type' => 'web', // hoặc 'email'

'message' => "Nhiệm vụ '{$task->title}' sắp đến hạn lúc {$task->due\_date->format('H:i d/m')}",

'sent\_at' => now(),

]);

}

}

}

$this->info("Đã gửi thông báo cho " . $tasks->count() . " nhiệm vụ.");

}

}

**🛠️ Bước 2: Đăng ký Command trong App\Console\Kernel.php**

php

CopyEdit

protected function schedule(Schedule $schedule): void

{

$schedule->command('tasks:notify-upcoming')->hourly();

}

**🗓️ Bước 3: Bật Scheduler trên máy cục bộ**

Thêm dòng sau vào app/Console/Kernel.php nếu chưa:

bash

CopyEdit

\* \* \* \* \* cd /path/to/project && php artisan schedule:run >> /dev/null 2>&1

(Với XAMPP, bạn dùng Task Scheduler của Windows hoặc cron nếu dùng WSL/Linux.)

**📦 Bước 4: Bổ sung Notification model nếu chưa có**

bash

CopyEdit

php artisan make:model Notification

Và đảm bảo trong model có:

php

CopyEdit

class Notification extends Model

{

protected $fillable = ['user\_id', 'task\_id', 'type', 'message', 'sent\_at'];

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

public function task() { return $this->belongsTo(Task::class); }

}

📬 Vậy là bạn đã hoàn tất hệ thống **gửi thông báo tự động** trước deadline!

Bạn có muốn mình hướng dẫn thêm:

* Gửi **email thực sự** thay vì chỉ lưu notification?
* Xây API để **frontend lấy danh sách thông báo chưa đọc**?

Bạn chọn nhé.

Đúng rồi! Đoạn code bạn viết trong Kernel.php như vậy là chính xác – nó sẽ chạy command tasks:notify-upcoming **mỗi giờ một lần**.

**💌 Giờ mình sẽ hướng dẫn bạn GỬI EMAIL thực sự thay vì chỉ lưu trong bảng notifications.**

**✅ Bước 1: Tạo Notification class (dùng Laravel Notification)**

bash

CopyEdit

php artisan make:notification TaskDeadlineReminder

**📁 app/Notifications/TaskDeadlineReminder.php**

php

CopyEdit

namespace App\Notifications;

use App\Models\Task;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Notifications\Notification;

use Illuminate\Notifications\Messages\MailMessage;

class TaskDeadlineReminder extends Notification implements ShouldQueue

{

use Queueable;

public $task;

public function \_\_construct(Task $task)

{

$this->task = $task;

}

public function via($notifiable)

{

return ['mail']; // Bạn có thể thêm 'database' nếu muốn lưu DB

}

public function toMail($notifiable)

{

return (new MailMessage)

->subject('⏰ Nhắc nhở deadline: ' . $this->task->title)

->line("Bạn được giao nhiệm vụ: {$this->task->title}")

->line("Hạn chót: " . $this->task->due\_date->format('H:i d/m/Y'))

->action('Xem chi tiết', url('/tasks/' . $this->task->id))

->line('Hãy hoàn thành đúng hạn nhé!');

}

}

**✅ Bước 2: Sửa NotifyUpcomingTasks để gửi email**

Trong vòng lặp foreach, thay đoạn:

php

CopyEdit

Notification::create([

...

]);

Thành:

php

CopyEdit

$assigneeUser = $assignee->user;

$assigneeUser->notify(new \App\Notifications\TaskDeadlineReminder($task));

// Nếu muốn vẫn lưu vào bảng notifications:

\App\Models\Notification::create([

'task\_id' => $task->id,

'user\_id' => $assignee->user\_id,

'type' => 'email',

'message' => "Đã gửi email nhắc deadline cho task '{$task->title}'",

'sent\_at' => now(),

]);

**✅ Bước 3: Cấu hình gửi email**

Mở file .env, cấu hình SMTP (ví dụ Gmail):

ini

CopyEdit

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.gmail.com

MAIL\_PORT=587

MAIL\_USERNAME=your\_email@gmail.com

MAIL\_PASSWORD=your\_app\_password

MAIL\_ENCRYPTION=tls

MAIL\_FROM\_ADDRESS=your\_email@gmail.com

MAIL\_FROM\_NAME="Task Manager"

🧠 **Lưu ý:** Nếu dùng Gmail, bạn cần tạo **App Password** trong Google Account (không dùng mật khẩu Gmail thường).

**✅ Bước 4: Test gửi**

Chạy lệnh thủ công:

bash

CopyEdit

php artisan tasks:notify-upcoming

Kiểm tra email của user trong bảng users xem có thư đến không.

Bạn muốn mình hỗ trợ thêm về:

* **Giao diện frontend thông báo (web)?**
* **Hiển thị thông báo chưa đọc?**
* **Bật gửi email khi tạo task mới?**

Bạn cần phần nào tiếp theo?

Updated saved memory

Cảm ơn bạn đã cung cấp thông tin! Dưới đây là hướng dẫn tích hợp **Laravel Notifications** với **React** (frontend).

**💻 Tích hợp Laravel Notifications với React**

Để hiển thị thông báo trên frontend React, chúng ta sẽ sử dụng một API từ Laravel để gửi và nhận thông báo.

**✅ Bước 1: Tạo API để lấy thông báo từ Laravel**

1. **Tạo route API trong Laravel** để lấy các thông báo chưa đọc.

Mở file routes/api.php và thêm route:

php

CopyEdit

use App\Http\Controllers\Api\NotificationController;

Route::middleware('auth:sanctum')->get('/notifications', [NotificationController::class, 'index']);

Route::middleware('auth:sanctum')->post('/notifications/{id}/read', [NotificationController::class, 'markAsRead']);

1. **Tạo Controller** để xử lý các yêu cầu từ API.

bash

CopyEdit

php artisan make:controller Api/NotificationController

**📁 app/Http/Controllers/Api/NotificationController.php**

php

CopyEdit

namespace App\Http\Controllers\Api;

use App\Models\Notification;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class NotificationController extends Controller

{

// Lấy tất cả thông báo chưa đọc

public function index(Request $request)

{

$notifications = $request->user()->unreadNotifications;

return response()->json($notifications);

}

// Đánh dấu thông báo là đã đọc

public function markAsRead(Request $request, $id)

{

$notification = $request->user()->notifications()->findOrFail($id);

$notification->markAsRead();

return response()->json(['message' => 'Notification marked as read']);

}

}

**✅ Bước 2: Sử dụng Sanctum cho API Authentication**

Để React có thể giao tiếp với API, bạn cần sử dụng **Laravel Sanctum** cho xác thực API.

1. **Cài đặt Sanctum**:

bash

CopyEdit

composer require laravel/sanctum

1. **Publish Sanctum config**:

bash

CopyEdit

php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

1. **Migrate Sanctum tables**:

bash

CopyEdit

php artisan migrate

1. **Thêm middleware vào api middleware group** trong app/Http/Kernel.php:

php

CopyEdit

'api' => [

\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,

'throttle:api',

\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,

],

1. **Cấu hình Sanctum** trong file .env:

env

CopyEdit

SANCTUM\_STATEFUL\_DOMAINS=your-frontend-domain.com

SESSION\_DOMAIN=your-frontend-domain.com

1. **Đảm bảo User model sử dụng HasApiTokens**:

php

CopyEdit

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

use HasApiTokens;

}

**✅ Bước 3: Gửi thông báo từ backend (Laravel)**

Laravel sẽ gửi thông báo khi deadline sắp đến, như đã hướng dẫn trong các bước trước.

**✅ Bước 4: Kết nối với React Frontend**

1. **Cài Axios để gọi API**:

bash

CopyEdit

npm install axios

1. **Cấu hình Axios và gửi yêu cầu lấy thông báo**:

js

CopyEdit

import axios from 'axios';

const API\_URL = 'http://your-laravel-backend.com/api';

axios.defaults.withCredentials = true; // Quan trọng để gửi cookies cho Sanctum

// Hàm lấy thông báo

const getNotifications = async () => {

try {

const response = await axios.get(`${API\_URL}/notifications`);

return response.data;

} catch (error) {

console.error('Error fetching notifications:', error);

return [];

}

};

// Hàm đánh dấu thông báo đã đọc

const markAsRead = async (id) => {

try {

await axios.post(`${API\_URL}/notifications/${id}/read`);

} catch (error) {

console.error('Error marking notification as read:', error);

}

};

1. **React Component để hiển thị thông báo**:

js

CopyEdit

import React, { useEffect, useState } from 'react';

const Notifications = () => {

const [notifications, setNotifications] = useState([]);

useEffect(() => {

const fetchNotifications = async () => {

const data = await getNotifications();

setNotifications(data);

};

fetchNotifications();

}, []);

return (

<div>

<h1>Thông báo của bạn</h1>

<ul>

{notifications.map((notification) => (

<li key={notification.id}>

<strong>{notification.data.message}</strong><br />

<span>{new Date(notification.created\_at).toLocaleString()}</span>

<button onClick={() => markAsRead(notification.id)}>Đánh dấu đã đọc</button>

</li>

))}

</ul>

</div>

);

};

export default Notifications;

**✅ Bước 5: Cấu hình CORS**

Để React frontend có thể gửi yêu cầu tới API Laravel, bạn cần cấu hình CORS.

1. **Cài đặt Laravel CORS**:

bash

CopyEdit

composer require fruitcake/laravel-cors

1. **Cấu hình CORS** trong file config/cors.php:

php

CopyEdit

'supports\_credentials' => true,

'allowed\_origins' => ['http://your-frontend-domain.com'],

**✅ Bước 6: Chạy và kiểm tra**

1. **Chạy Laravel**:

bash

CopyEdit

php artisan serve

1. **Chạy React**:

bash

CopyEdit

npm start

Giờ bạn có thể **hiển thị thông báo** trên frontend React, và **đánh dấu đã đọc** thông qua API Laravel.

Nếu bạn muốn thêm tính năng **real-time notifications** (sử dụng **Pusher** hoặc **Laravel Echo**), mình có thể giúp bạn tích hợp tiếp.

Bạn cần hỗ trợ thêm phần nào?

**🧩 Cấu trúc thư mục frontend (React)**

plaintext

CopyEdit

src/

│

├── components/

│ ├── GroupList.js

│ ├── GroupDetail.js

│ ├── TaskList.js

│ ├── TaskDetail.js

│ ├── NotificationList.js

│ └── Navbar.js

│

├── services/

│ └── api.js

│

├── App.js

└── index.js

**🧩 Cài đặt các thư viện cần thiết**

Trước hết, bạn cần cài đặt Axios để thực hiện các API calls từ frontend React đến backend Laravel:

bash

CopyEdit

npm install axios react-router-dom

**🧩 API Service (Kết nối với Laravel API)**

Tạo một file services/api.js để quản lý các API call:

javascript

CopyEdit

import axios from 'axios';

const api = axios.create({

baseURL: 'http://localhost:8000/api', // Đảm bảo rằng đây là URL đúng của API Laravel

headers: {

'Content-Type': 'application/json',

},

});

// Lấy tất cả các nhóm của người dùng

export const getGroups = async () => {

try {

const response = await api.get('/groups');

return response.data;

} catch (error) {

console.error('Lỗi khi lấy danh sách nhóm:', error);

throw error;

}

};

// Lấy chi tiết một nhóm

export const getGroupDetails = async (id) => {

try {

const response = await api.get(`/groups/${id}`);

return response.data;

} catch (error) {

console.error('Lỗi khi lấy thông tin nhóm:', error);

throw error;

}

};

// Lấy tất cả các nhiệm vụ của một nhóm

export const getTasksByGroup = async (groupId) => {

try {

const response = await api.get(`/groups/${groupId}/tasks`);

return response.data;

} catch (error) {

console.error('Lỗi khi lấy danh sách nhiệm vụ:', error);

throw error;

}

};

// Tạo nhiệm vụ mới

export const createTask = async (taskData) => {

try {

const response = await api.post('/tasks', taskData);

return response.data;

} catch (error) {

console.error('Lỗi khi tạo nhiệm vụ:', error);

throw error;

}

};

**🧩 Component: GroupList.js (Hiển thị danh sách nhóm)**

javascript

CopyEdit

import React, { useEffect, useState } from 'react';

import { getGroups } from '../services/api';

import { Link } from 'react-router-dom';

const GroupList = () => {

const [groups, setGroups] = useState([]);

useEffect(() => {

const fetchGroups = async () => {

try {

const data = await getGroups();

setGroups(data);

} catch (error) {

console.error('Lỗi khi lấy danh sách nhóm:', error);

}

};

fetchGroups();

}, []);

return (

<div>

<h1>Danh sách Nhóm</h1>

<ul>

{groups.map(group => (

<li key={group.id}>

<Link to={`/groups/${group.id}`}>{group.name}</Link>

</li>

))}

</ul>

</div>

);

};

export default GroupList;

**🧩 Component: GroupDetail.js (Hiển thị chi tiết nhóm)**

javascript

CopyEdit

import React, { useEffect, useState } from 'react';

import { useParams } from 'react-router-dom';

import { getGroupDetails, getTasksByGroup } from '../services/api';

const GroupDetail = () => {

const { id } = useParams();

const [group, setGroup] = useState(null);

const [tasks, setTasks] = useState([]);

useEffect(() => {

const fetchGroupDetails = async () => {

try {

const groupData = await getGroupDetails(id);

setGroup(groupData);

const tasksData = await getTasksByGroup(id);

setTasks(tasksData);

} catch (error) {

console.error('Lỗi khi lấy thông tin nhóm:', error);

}

};

fetchGroupDetails();

}, [id]);

if (!group) return <div>Đang tải...</div>;

return (

<div>

<h1>{group.name}</h1>

<p>{group.description}</p>

<h2>Nhiệm vụ</h2>

<ul>

{tasks.map(task => (

<li key={task.id}>

<strong>{task.title}</strong> - {task.status}

</li>

))}

</ul>

</div>

);

};

export default GroupDetail;

**🧩 Component: TaskList.js (Hiển thị danh sách nhiệm vụ)**

javascript

CopyEdit

import React, { useEffect, useState } from 'react';

import { getTasksByGroup } from '../services/api';

const TaskList = ({ groupId }) => {

const [tasks, setTasks] = useState([]);

useEffect(() => {

const fetchTasks = async () => {

try {

const data = await getTasksByGroup(groupId);

setTasks(data);

} catch (error) {

console.error('Lỗi khi lấy danh sách nhiệm vụ:', error);

}

};

fetchTasks();

}, [groupId]);

return (

<div>

<h2>Nhiệm vụ</h2>

<ul>

{tasks.map(task => (

<li key={task.id}>

<strong>{task.title}</strong> - {task.status}

</li>

))}

</ul>

</div>

);

};

export default TaskList;

**🧩 Component: NotificationList.js (Hiển thị thông báo)**

javascript

CopyEdit

import React, { useEffect, useState } from 'react';

import { getNotifications } from '../services/api';

const NotificationList = () => {

const [notifications, setNotifications] = useState([]);

useEffect(() => {

const fetchNotifications = async () => {

try {

const data = await getNotifications(); // API tương ứng trong `api.js`

setNotifications(data);

} catch (error) {

console.error('Lỗi khi lấy thông báo:', error);

}

};

fetchNotifications();

}, []);

return (

<div>

<h1>Thông báo</h1>

<ul>

{notifications.map(notification => (

<li key={notification.id}>

<strong>{notification.message}</strong> - {notification.sent\_at}

</li>

))}

</ul>

</div>

);

};

export default NotificationList;

**🧩 Cập nhật App.js để cấu hình routing**

javascript

CopyEdit

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import GroupList from './components/GroupList';

import GroupDetail from './components/GroupDetail';

import Navbar from './components/Navbar';

import TaskList from './components/TaskList';

function App() {

return (

<Router>

<Navbar />

<div className="container">

<Routes>

<Route path="/" element={<GroupList />} />

<Route path="/groups/:id" element={<GroupDetail />} />

</Routes>

</div>

</Router>

);

}

export default App;

**🧩 Component Navbar (Navbar.js)**

javascript

CopyEdit

import React from 'react';

import { Link } from 'react-router-dom';

const Navbar = () => {

return (

<nav>

<ul>

<li><Link to="/">Trang chủ</Link></li>

<li><Link to="/groups">Danh sách nhóm</Link></li>

</ul>

</nav>

);

};

export default Navbar;

**🧩 Cấu hình Routes trong index.js**

javascript

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

**🧩 Kiểm tra và chạy ứng dụng**

* Đảm bảo rằng bạn đã chạy ứng dụng Laravel API trên http://localhost:8000.
* Chạy ứng dụng React:

bash

CopyEdit

npm start

Khi hoàn tất các bước trên, bạn sẽ có một ứng dụng React có khả năng tạo, quản lý nhóm, nhiệm vụ và thông báo, và giao tiếp với API Laravel backend.