@props(['post' => $post])

<div class="mb-4 pl-3">

<a href="{{ route('users.posts', $post->user) }}" class="font-bold">{{ $post->user->name }}</a> <span class="text-gray text-sm">{{ $post->created\_at->diffForHumans() }}</span>

<p class="mb-2">{{ $post->body }}</p>

<div class="flex items-center">

    @auth

    <div class="btn-toolbar" role="toolbar" aria-label="Toolbar with button groups">

    @if (!$post->likedBy(Auth::user()))

    <form action="{{ route('posts.likes', $post) }}" method="POST" class="mr-1">

        @csrf

        <div class="btn-group me-2" role="group" aria-label="First group">

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

  </div>

    </form>

    @else <form action="{{ route('posts.likes', $post) }}" method="POST" class="mr-1">

        @csrf

        @method('DELETE')

        <div class="btn-group me-2" role="group" aria-label="Second group">

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

  </div>

    </form>

    @endif

  <div class="btn-group" role="group" aria-label="Second group">

  @can('delete', $post)

    <form action="{{ route('posts.destroy', $post) }}" method="POST" class="mr-1">

        @csrf

        @method('DELETE')

        <button type="submit" class="btn btn-warning pl-3">Delete</button>

    </form>

   @endcan

  </div>

</div>

    @endauth

    <div class="pt-2">

    <span>{{ $post->likes->count() }} {{ Str::plural('like', $post->likes->count()) }}</span>

    </div>

</div>

</div>

class Like extends Model

{

    use HasFactory, SoftDeletes;

    protected $fillable = [

        'user\_id',

    ];

}

class Post extends Model

{

    use HasFactory;

    protected $fillable = [

        'body'

    ];

    public function likedBy(User $user)

{

    return $this->likes->contains('user\_id', $user->id); // Collection method

}

public function user()

{

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

}

public function likes()

{

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

}

}

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    protected $casts = [

        'email\_verified\_at' => 'datetime',

    ];

public function posts()

{

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

}

public function likes()

{

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

}

public function receivedLikes()

{

    return $this->hasManyThrough(Like::class, Post::class);

}

}

Follows controller

   public function store(Request $request, Profile $profile){

   $profile->follows()->create([

    'user\_id' => $request->user()->id

    ]);

    $user = auth()->user(); //May need to be removed

    if ($post->likes()->onlyTrashed()->where('user\_id', $request->user()->id)->count()) {

        Mail::to($post->user)->send(new PostLiked(auth()->user(), $post));

    }

    return back();

   }

   public function destroy(Post $post, Request $request)

   {

        $request->user()->follows()

        ->where('profile\_id', $profile->id)

        ->delete();

        return back();

   }

Routes

Route::post('/profiles/{profile}/follows', [FollowsController::class, 'store'])->name(profiles.follows);

Route::delete('/ profiles/{profile}/follows', [FollowsController::class, 'destroy'])->name(profiles.follows);