\*\*Must follow the number 1 documentation to setup the project

**## Private Channel**

Private channel works with the authenticated users.

From **channels.php**,

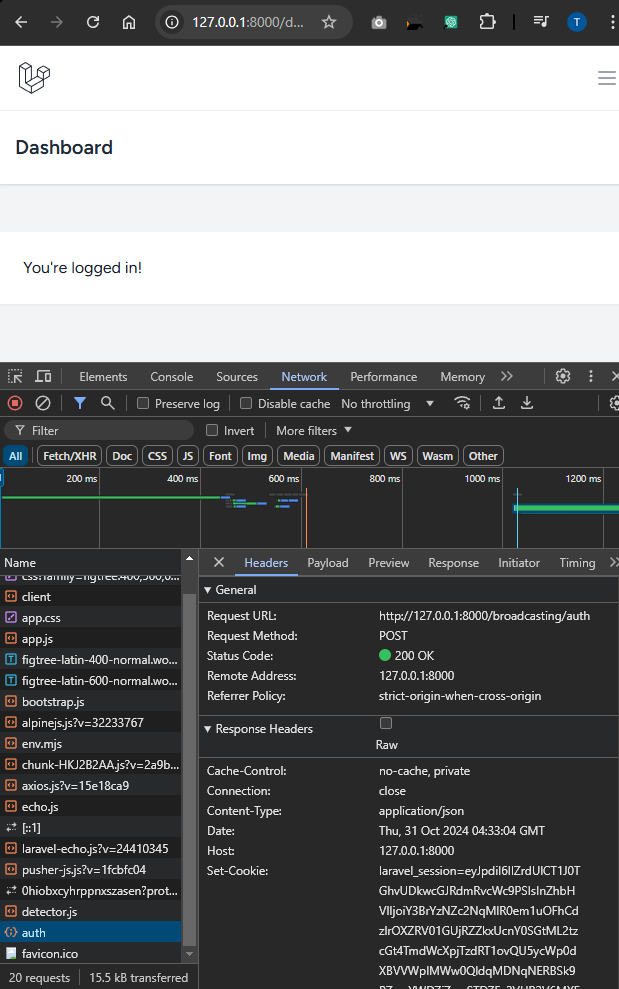
Broadcast::channel('users.{id}', function (User $user, $id) {  
 return (int) $user->id === (int) $id;  
});

Now checking if its working, from **dashboard.php**,

<div class="py-12">  
 <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">  
 <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">  
 <div class="p-6 text-gray-900" x-init="privateClass()" x-data="{dispatched: false}">  
 {{ \_\_("You're logged in!") }}  
 </div>  
 </div>  
 </div>  
</div>  
  
<script>  
 function privateClass(){  
 Echo.private('users.{{ auth()->id() }}')  
 }  
</script>

* Now **x-init** e function add kora hoise jate niche <script> block e easily code lekha jay.
* auth()->id() diye authenticated specific user ke bojhano hoise
* x-data="{dispatched: false}" is a part of alpine.js

## Testing if its working, just refresh the web page and view the network tab



* Ekhane **auth** er request tar Status Code : 200 OK, cause authenticated user is connected

Step 2: Creating an event:

php artisan make:event OrderDispatched

Full **OrderDispatched.php** code,

<?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\Contracts\Broadcasting\ShouldBroadcastNow;  
use Illuminate\Foundation\Events\Dispatchable;  
use Illuminate\Queue\SerializesModels;  
use App\Models\User;  
  
class OrderDispatched implements ShouldBroadcastNow  
{  
 use Dispatchable, InteractsWithSockets, SerializesModels;  
   
 public function \_\_construct(public User $user)  
 {  
 //  
 }  
   
 public function broadcastOn(): array  
 {  
 return [  
 new PrivateChannel('users.'.$this->user->id),  
 ];  
 }  
}

Add routes in **web.php**,

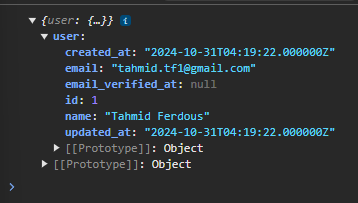
Route::get('/broadcast', function () {  
 broadcast(new \App\Events\OrderDispatched(User::*find*(1)));  
});

Now from **dashboard.php**,

<x-app-layout>  
 <x-slot name="header">  
 <h2 class="font-semibold text-xl text-gray-800 leading-tight">  
 {{ \_\_('Dashboard') }}  
 </h2>  
 </x-slot>  
  
 <div class="py-12">  
 <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">  
 <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">  
 <div class="p-6 text-gray-900" x-init="privateClass()"

>  
 {{ \_\_("You're logged in!") }}  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <script>  
 function privateClass(){  
 Echo.private('users.{{ auth()->id() }}').listen('OrderDispatched', (event)=>{  
 *console*.log(event)  
 })  
 }  
 </script>  
  
</x-app-layout>

* After hitting the route **/broadcast** from route, this output will occur



**## Showing Data based on dispatch**

Step 1: Creating an alpine store,

{{-- Alpine js store script --}}  
  
<script>  
 *document*.addEventListener('alpine:init', () => {  
 Alpine.store('dispatchedStore', {dispatched: false});  
 });  
</script>

Step 2: triggering alpine.store variables,

Alpine.store('dispatchedStore').dispatched = true;

Here is a script code of the function,

<script>  
 function privateClass() {  
 Echo.private('users.{{ auth()->id() }}').listen('OrderDispatched', (event) => {  
  
 Alpine.store('dispatchedStore').dispatched = true;  
 })  
 }  
</script>

Step 3: Using it as a property, **x-if** is also a part of alpine js,

<template x-if="$store.dispatchedStore.dispatched">  
 <div>  
 Order has been dispatched  
 </div>  
</template>

Now the full updated code is here,

<div class="py-12">  
 <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">  
 <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">  
 <div class="p-6 text-gray-900" x-init="privateClass()">  
  
 <template x-if="$store.dispatchedStore.dispatched">  
 <div>  
 Order has been dispatched  
 </div>  
 </template>  
 </div>  
 </div>  
 </div>  
</div>

<script>  
 function privateClass() {  
 Echo.private('users.{{ auth()->id() }}').listen('OrderDispatched', (event) => {  
  
 Alpine.store('dispatchedStore').dispatched = true;  
 })  
 }  
</script>  
  
{{-- Alpine js store script --}}  
  
<script>  
 *document*.addEventListener('alpine:init', () => {  
 Alpine.store('dispatchedStore', {dispatched: false});  
 });  
</script>