Step 1 : php artisan make:event UserSessionChanged

Ekta event khola holo jekhane class er shathe **ShouldBroadcast** implement kora holo and duita variable newa holo class er under and construction moddhe set kora holo, **Log** kora hoise just data gula ke lo kore paoar jonno,

From Events-> UserSessionChanged.php,

```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 UserSessionChanged implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $message;
    public $type;

    /**
     * Create a new event instance.
     *
     * @return void
     */
    public function __construct($message, $type)
    {
        $this->message = $message;
        $this->type = $type;
    }

    /**
     * Get the channels the event should broadcast on.
     *
     * @return \Illuminate\Broadcasting\Channel|array
     */
    public function broadcastOn()
    {
        \Log::debug($this->message);
        \Log::debug($this->type);
        return new Channel('notifications');
    }
}
```

Step 2 : php artisan make:listener BroadcastUserLoginNotification

Then ekta listener banano holo jekhane Login event banano holo.

From Listeners-> BroadcastUserLoginNotification.php, jekhane **Login $event** hocche use hoy for login purposes, Logout er jonno hole **Logout $event.**

```php
<?php

namespace App\Providers;

use App\Listeners\BroadcastUserLoginNotification;
use App\Listeners\BroadcastUserLogoutNotification;
use Illuminate\Auth\Events\Login;
use Illuminate\Auth\Events\Logout;
use Illuminate\Auth\Events\Registered;
use Illuminate\Auth\Listeners\SendEmailVerificationNotification;
use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;
use Illuminate\Support\Facades\Event;

class EventServiceProvider extends ServiceProvider
{
    /**
     * The event listener mappings for the application.
     *
     * @var array
     */
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],

        Login::class => [
            BroadcastUserLoginNotification::class,
        ],
//
//        Logout::class => [
//            BroadcastUserLogoutNotification::class,
//        ],
    ];

    /**
     * Register any events for your application.
     *
     * @return void
     */
    public function boot()
    {
        parent::boot();

        //
    }
}
```

Step 3 : From Providers->EventServiceProvider e listener gula ke register korte hobe,

```php
<?php

namespace App\Providers;

use App\Listeners\BroadcastUserLoginNotification;
use App\Listeners\BroadcastUserLogoutNotification;
use Illuminate\Auth\Events\Login;
use Illuminate\Auth\Events\Logout;
use Illuminate\Auth\Events\Registered;
use Illuminate\Auth\Listeners\SendEmailVerificationNotification;
use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;
use Illuminate\Support\Facades\Event;

class EventServiceProvider extends ServiceProvider
{
    /**
     * The event listener mappings for the application.
     *
     * @var array
     */
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],

//         Login::class => [
//             BroadcastUserLoginNotification::class,
//         ],
//
//         Logout::class => [
//             BroadcastUserLogoutNotification::class,
//         ],
    ];

    /**
     * Register any events for your application.
     *
     * @return void
     */
    public function boot()
    {
        parent::boot();

        //
    }
}
```

********** Performing the result from **resources->js->app.js**,

```
require('./bootstrap');

Echo.channel("notifications").listen('UserSessionChanged', function (e) {
    let notificationElement = document.getElementById("notification");
    notificationElement.innerText = e.message;

    notificationElement.classList.remove('invisible');
    notificationElement.classList.remove('alert-success');
    notificationElement.classList.remove('alert-danger');
    notificationElement.classList.add('alert-' + e.type);
});
```

Ekhane,

1. Notifications hocche channel name from **UserSessionChanged.php**
2. 'UserSessionChanged' hocche event ta
3. Erpor hocche ekta callback action ar vitore hocche function output

********** **Broadcasting the Event Only to Authenticated Users**

Concept ta jevabe kaj korse, private channel e jara jara logged in thakbe tarai update pabe

From **Events->UserSessionChanged.php**, just private channel kore dewa hoise

```
public function broadcastOn()
{
    \Log::debug($this->message);
    \Log::debug($this->type);
    return new PrivateChannel('notifications');
}
```

then from **resources->js->app.js,** channel type provate kore dewa hoise,

```
Echo.private("notifications").listen('UserSessionChanged', function (e) {
    let notificationElement = document.getElementById("notification");
    notificationElement.innerText = e.message;

    notificationElement.classList.remove('invisible');
    notificationElement.classList.remove('alert-success');
    notificationElement.classList.remove('alert-danger');
    notificationElement.classList.add('alert-' + e.type);
});
```

Then lastly from **routes->channels,php,** normally channel ke declare kore dewa hoise

```
Broadcast::channel('notifications', function ($user) {
    return $user != null;
});
```