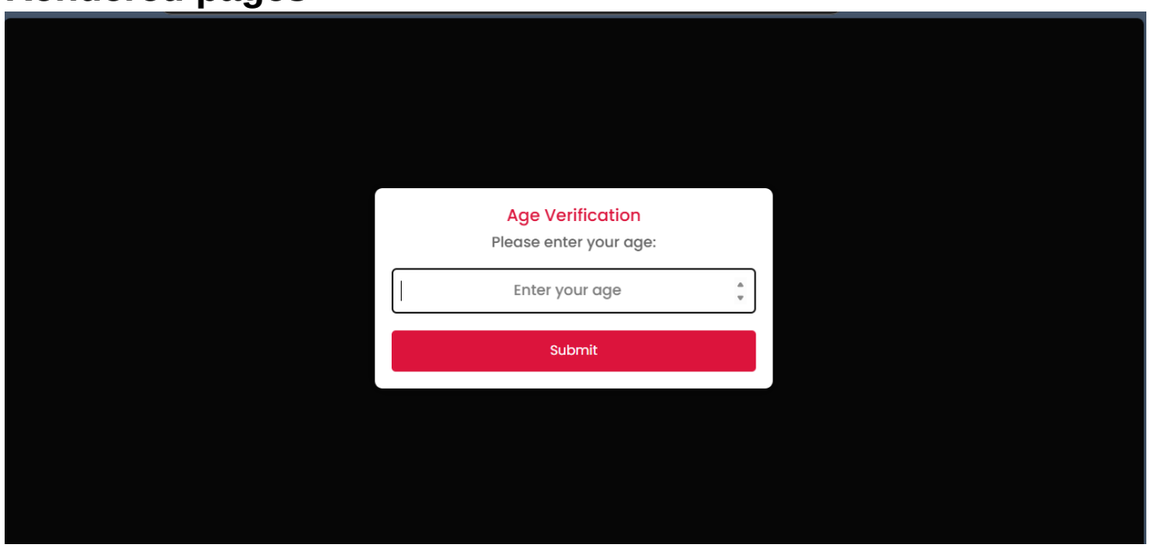
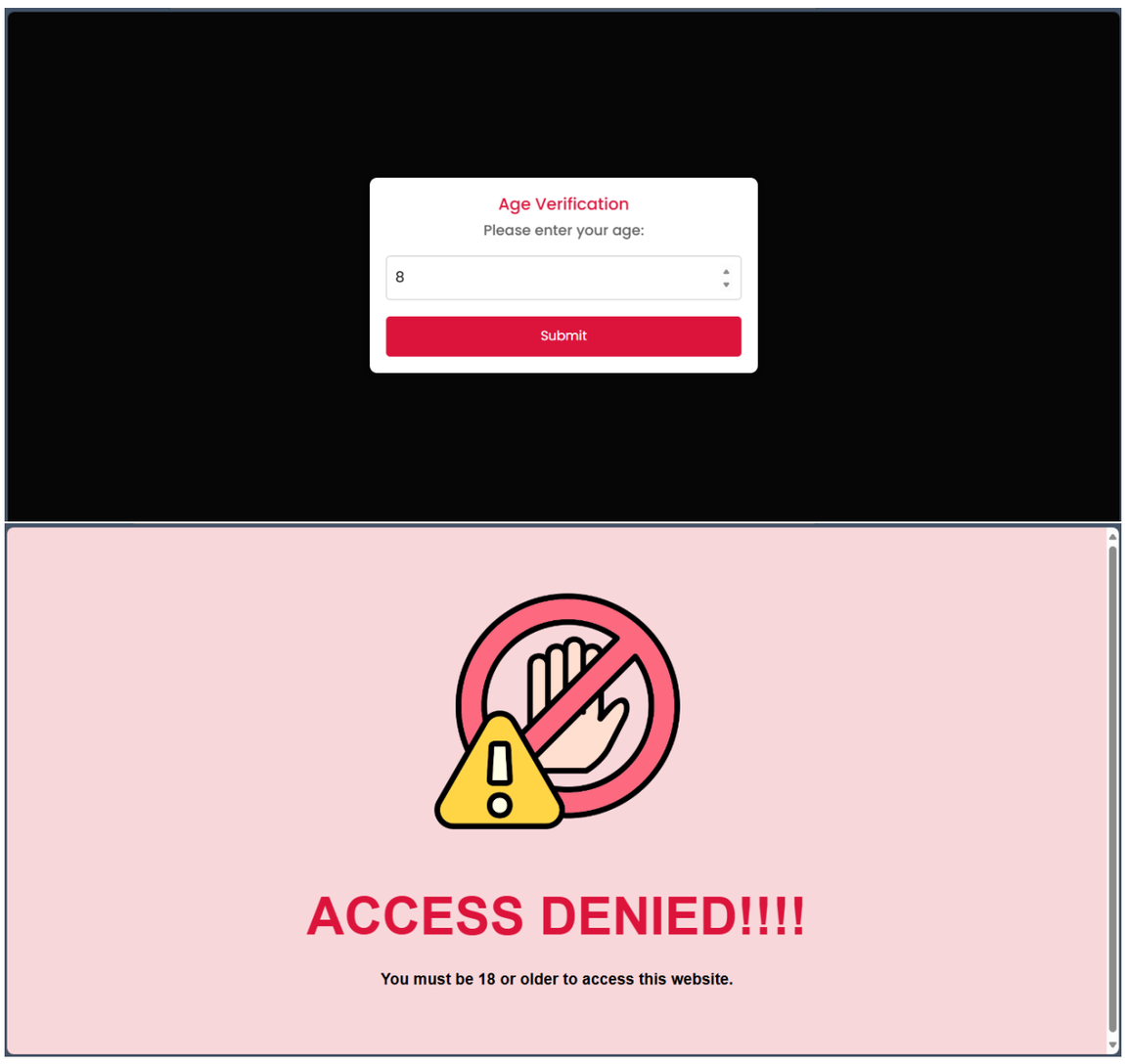
Individual Documentation
Middleware

Manuel Andrei Lleva
BSIT – 3C

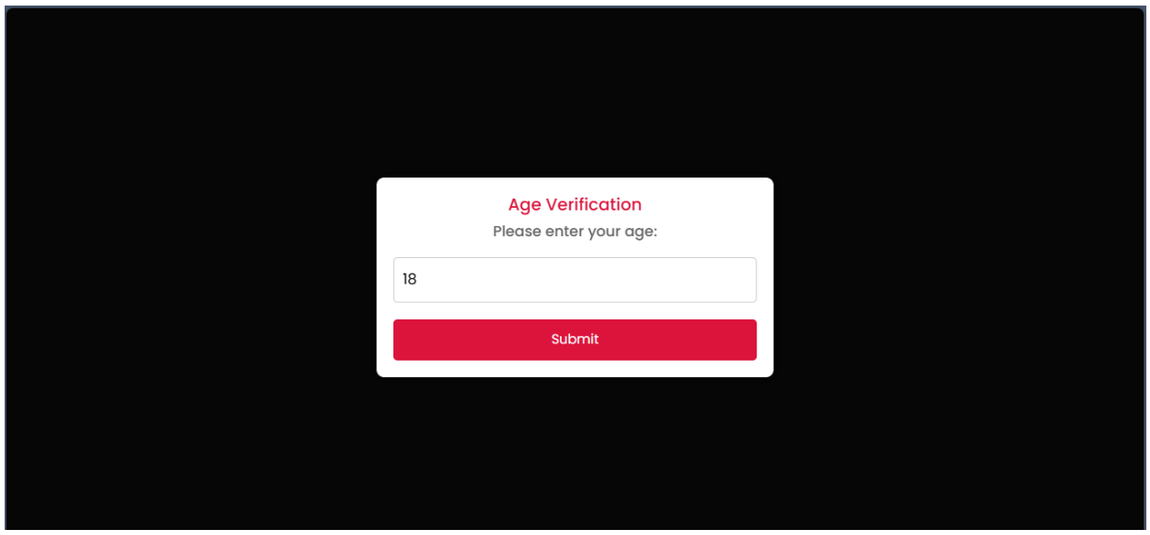
Rendered pages

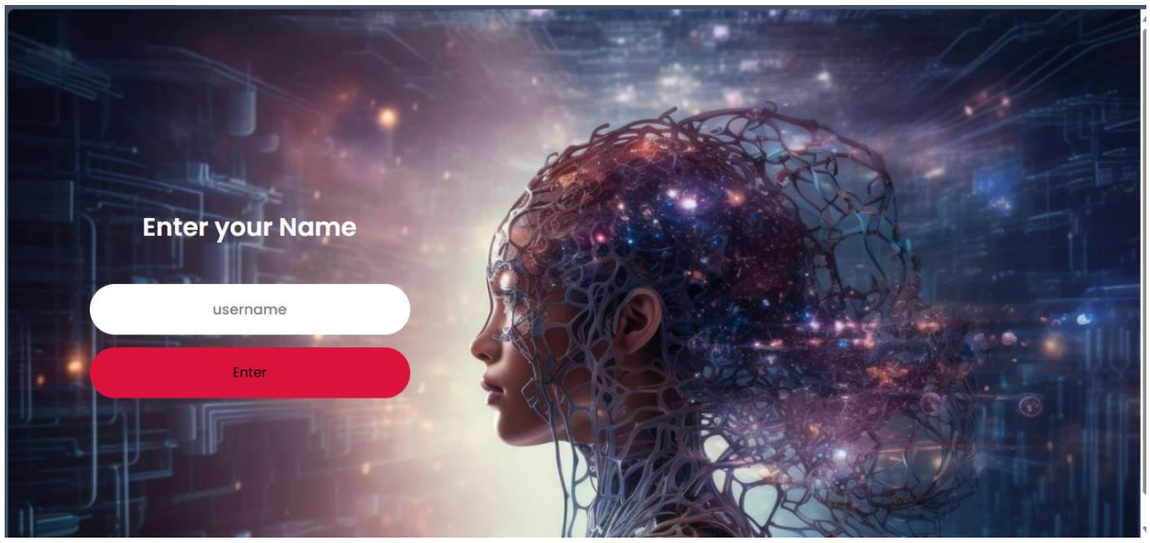


If below 18 yrs old

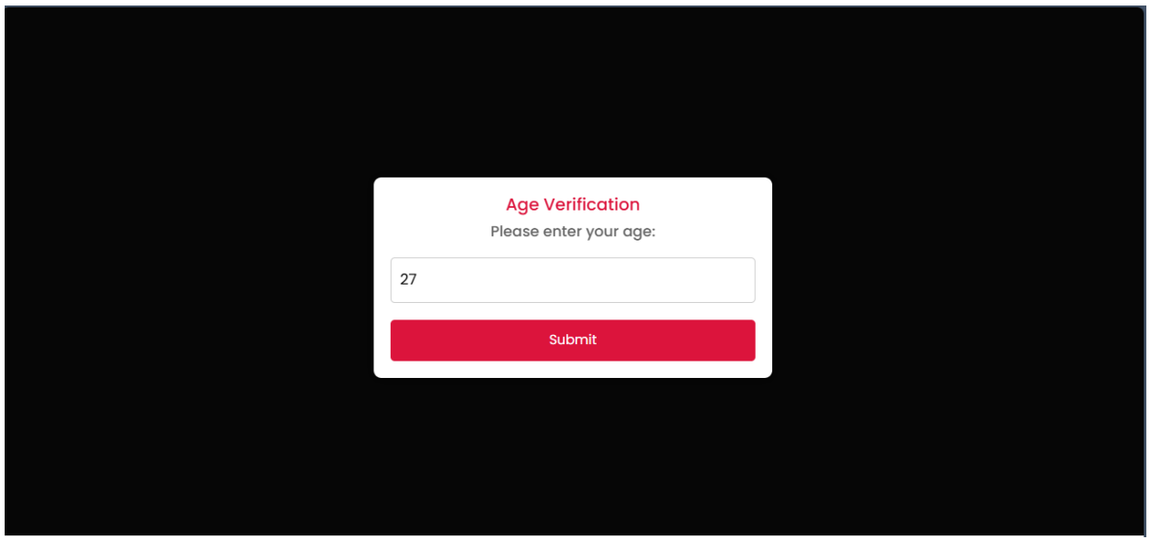


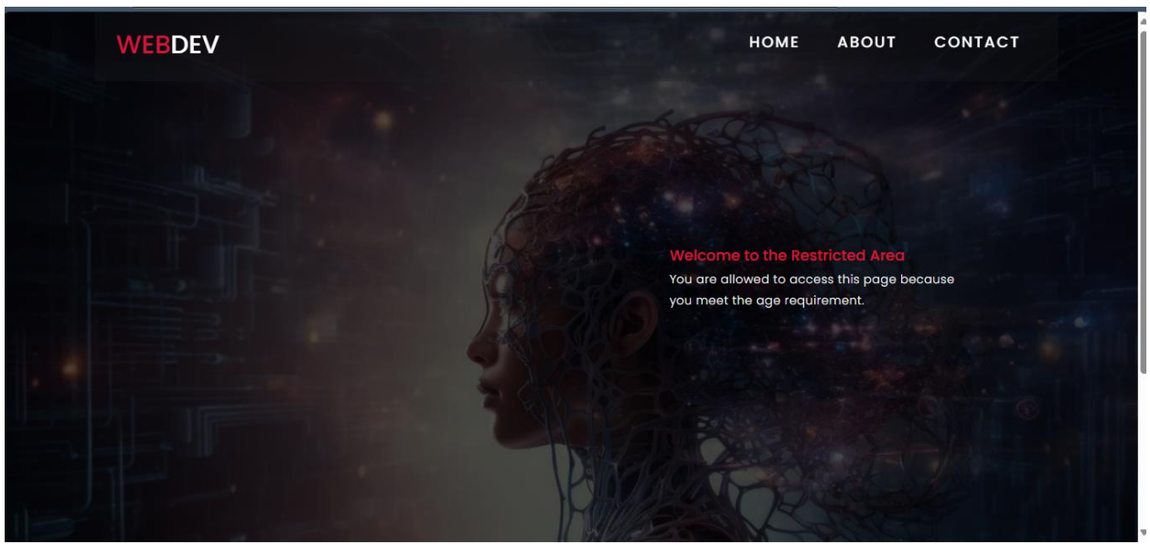
If 18-20 years old





If 21 years old and above





Middleware

app/Http/Middleware/CheckAge.php

app > Http > Middleware > CheckAge.php > CheckAge > handle

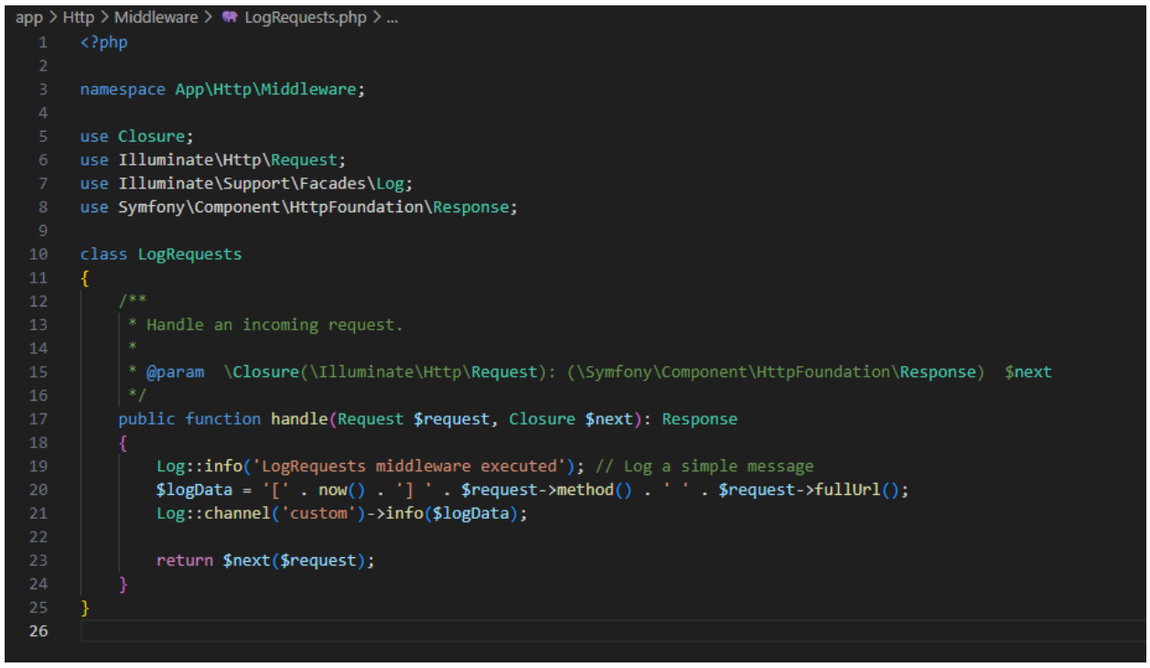
```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class CheckAge
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next, $minAge = 18) // Default to 18
    {

        \Log::info('CheckAge middleware executed.'); // Log to verify middleware execution
        $age = $request->query('age'); // Get the age from the query string

        // Check if the user is below the minimum age
        if (isset($age)) {
            if ($age < $minAge) {
                return redirect('/access-denied'); // Redirect to access denied page
            } elseif ($age >= 21) {
                return redirect('/restricted-area'); // Redirect to restricted area if age is 21 or above
            } elseif ($age == 18) {
                return redirect('/welcome'); // Redirect to home page if age is exactly 18
            }
        }
```

This CheckAge middleware checks the user's age from the query string, logs that the middleware was executed, and then either:

- Redirects users under 18 to an "Access Denied" page.
- Redirects users 21 or older to a "Restricted Area."
- Redirects users exactly 18 years old to a "Welcome" page.

If no redirection happens, the request continues processing as usual.

app/Http/Middleware/LogRequest.php



```php
app > Http > Middleware > LogRequests.php > ...
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Symfony\Component\HttpFoundation\Response;

class LogRequests
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        Log::info('LogRequests middleware executed'); // Log a simple message
        $logData = '[' . now() . '] ' . $request->method() . ' ' . $request->fullUrl();
        Log::channel('custom')->info($logData);

        return $next($request);
    }
}
```

This LogRequests middleware is designed to:

- Log a simple message when the middleware is executed.
- Log a more detailed message containing the request method and full URL, along with a timestamp, to a custom log channel.
- Allow the request to continue being processed by the next middleware or controller in the request pipeline.

This is useful for tracking requests, auditing, or debugging purposes.

Config/Logging.php

```php
config > logging.php
return [
    'channels' => [
        'emergency' => [
        ],

        // Add this part for the custom log channel
        'custom' => [
            'driver' => 'single',
            'path' => storage_path('logs/log.txt'),
            'level' => 'info',
        ],

    ],

];
```

We use this setup for a dedicated log file (log.txt) where we can send specific logs, rather than using the default Laravel log file. This helps me separate certain logs (like request logs, in our case) from the rest of the application's logging.

Sample output of log.txt

storage > logs > log.txt

```
[2024-10-05 13:12:25] local.INFO: [2024-10-05 13:12:25] GET http://127.0.0.1:8000
[2024-10-05 13:12:29] local.INFO: [2024-10-05 13:12:29] GET http://127.0.0.1:8000/access-denied
[2024-10-05 13:12:34] local.INFO: [2024-10-05 13:12:34] GET http://127.0.0.1:8000/restricted-area
[2024-10-05 13:12:37] local.INFO: [2024-10-05 13:12:37] GET http://127.0.0.1:8000/welcome
[2024-10-05 13:12:38] local.INFO: [2024-10-05 13:12:38] GET http://127.0.0.1:8000/about
[2024-10-05 13:12:40] local.INFO: [2024-10-05 13:12:40] GET http://127.0.0.1:8000/contact
[2024-10-05 13:12:42] local.INFO: [2024-10-05 13:12:42] GET http://127.0.0.1:8000/about
[2024-10-05 13:12:43] local.INFO: [2024-10-05 13:12:43] GET http://127.0.0.1:8000/welcome
[2024-10-05 13:12:45] local.INFO: [2024-10-05 13:12:45] GET http://127.0.0.1:8000/about
[2024-10-05 13:12:50] local.INFO: [2024-10-05 13:12:50] GET http://127.0.0.1:8000
[2024-10-05 13:14:51] local.INFO: [2024-10-05 13:14:51] GET http://127.0.0.1:8000
[2024-10-05 13:14:54] local.INFO: [2024-10-05 13:14:54] GET http://127.0.0.1:8000/restricted-area
[2024-10-05 13:14:55] local.INFO: [2024-10-05 13:14:55] GET http://127.0.0.1:8000/welcome
[2024-10-05 13:14:58] local.INFO: [2024-10-05 13:14:58] GET http://127.0.0.1:8000/about
[2024-10-05 13:14:59] local.INFO: [2024-10-05 13:14:59] GET http://127.0.0.1:8000/about
[2024-10-05 13:15:03] local.INFO: [2024-10-05 13:15:03] GET http://127.0.0.1:8000/contact
[2024-10-07 08:17:23] local.INFO: [2024-10-07 08:17:23] GET http://127.0.0.1:8000
[2024-10-07 08:19:23] local.INFO: [2024-10-07 08:19:23] GET http://127.0.0.1:8000
[2024-10-07 08:19:25] local.INFO: [2024-10-07 08:19:25] GET http://127.0.0.1:8000
[2024-10-07 08:19:26] local.INFO: [2024-10-07 08:19:26] GET http://127.0.0.1:8000
```

app/Http/Kernel.php

app > Http > Kernel.php > Kernel

```php
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    protected $middleware = [
        // Global HTTP middleware
        \App\Http\Middleware\LogRequests::class, // Register LogRequests as global middleware
    ];

    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            \App\Http\Middleware\CheckAge::class, // Add CheckAge to web middleware group

        ],

        'api' => [
            'throttle:api',
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],
    ];

    protected $routeMiddleware = [
        'checkAge' => \App\Http\Middleware\CheckAge::class, // Register CheckAge middleware
    ];
```

- **Global Middleware**: Automatically applies to every request made to the application.
- **Middleware Groups**: Organizes middleware for specific types of requests (web, api), allowing different sets of middleware to be applied depending on the type of request.
- **Route-Specific Middleware**: Allows you to apply middleware to individual routes, providing flexibility in request handling.

In this example, LogRequests is applied globally, CheckAge is applied to web routes via the 'web' group, and CheckAge can also be applied to specific routes via the 'checkAge' key in the route middleware.