

1. Global Middleware Registration

```
C:\Users> angel > webdev_lab4 > app > Http > Kernel.php
1  <?php
2
3  namespace App\Http;
4
5  use Illuminate\Foundation\Http\Kernel as HttpKernel;
6
7  class Kernel extends HttpKernel
8  {
9      /**
10       * The application's request middleware.
11       *
12       * These middleware are run during every request to your application.
13       *
14       * @var array<int, class-string|string>
15       */
16      protected $middleware = [
17          \App\Http\Middleware\TrustProxies::class,
18          \Illuminate\Http\Middleware\HandleCors::class,
19          \App\Http\Middleware\PreventRequestsDuringMaintenance::class,
20          \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
21          \App\Http\Middleware\TrimStrings::class,
22          \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
23          //global middleware part added
24          \App\Http\Middleware\CheckAge::class,
25          \App\Http\Middleware\LogRequests::class,
26      ];
```

In the **app/Http/Kernel.php** file, We registered the **LogRequests** and **CheckAge** middleware as global middleware by adding them to the **\$middleware** array. This means they will be applied to every incoming request throughout the entire application.

Global Middleware Purpose:

LogRequests: Logs every request made to the application. This helps track user activities across the app, regardless of the route.

CheckAge: Ensures that the user's age is validated globally, preventing access to restricted content for users who do not meet the age criteria.

2. Route-Specific Middleware

```
C:\Users> angel > webdev_lab4 > app > Http > Kernel.php
8  {
49      /**
50       * The application's route middleware.
51       *
52       * These middleware may be assigned to groups or used individually.
53       *
54       * @var array<string, class-string|string>
55       */
56      protected $routeMiddleware = [
57          'auth' => \App\Http\Middleware\Authenticate::class,
58          'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
59          'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
60          'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
61          'can' => \Illuminate\Auth\Middleware\Authorize::class,
62          'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
63          'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
64          'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
65          'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
66          'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
67          //routeMiddleware part added
68          'check.age' => \App\Http\Middleware\CheckAge::class,
69          'log.requests' => \App\Http\Middleware\LogRequests::class,
70      ];
71 }
```

In the same **Kernel.php** file, We also registered both middleware as route-specific by placing them in the **\$routeMiddleware** array. This allows them to be assigned to specific routes as needed.

We applied the **CheckAge** middleware to routes that require users to be of a certain age, and the **LogRequests** middleware was assigned to a group of routes to log the activity of users specifically for those routes.


3. Middleware with Parameters

To allow the **CheckAge** middleware to accept a parameter, I modified the handle method of the CheckAge middleware to accept a minimum age as a parameter. The middleware checks if the user's age meets the specified criteria.

```
C:\Users\angel> webdev_lab4 > app > Http > Middleware > CheckAge.php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\View;
8
9 class CheckAge
10 {
11     public function handle($request, Closure $next, $minAge = 18)
12     {
13         $age = $request->input('age', 0); // get age from request input (default 0 if not provided)
14
15         // Check if age meets the minimum requirement passed as a parameter
16         if ($age >= $minAge) {
17             // Return different views based on age range
18             if ($age <= 20) {
19                 return response()->view('home');
20             } else {
21                 return response()->view('adults');
22             }
23         }
24
25         // If age is below the minimum, redirect to denied page
26         return response()->view('denied');
27     }
28 }
```

- This conditional behavior allows you to serve different pages based on the user's age, offering a more dynamic response.
- If the user's age does not meet the minimum requirement, this line returns a custom "Access Denied" page.
- This informs the user that they are not allowed to access the content due to age restrictions.

- If the user's age is less than or equal to 20, it returns the 'home' view.




MINGZ

Verify Your Age to Log In

Please Type your age:

Submit



MINGZ

Mingkai Access

Please Access

Guest

← → ↻ 127.0.0.1:8000/homepage/Angela ☆ 📄 📄 📄 📄

MINGZ Home About Content Contact

MINGKAI'S LIFE AS A GINGER CATTO


Get To Know Me!

Meow! I'm Mingkai, and today I'm a cattol! Here's what my day looks like.

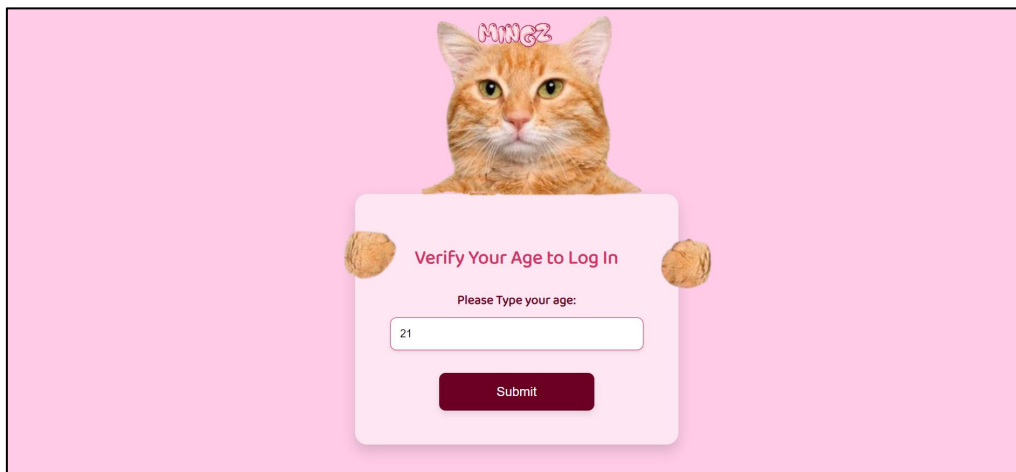
I start the day with a big stretch and soak up the morning sun. Then, I get breakfast—yum! After eating, I take a quick nap.

Around lunch, I eat, drink, poop, and play a bit. Then it's nap time again.

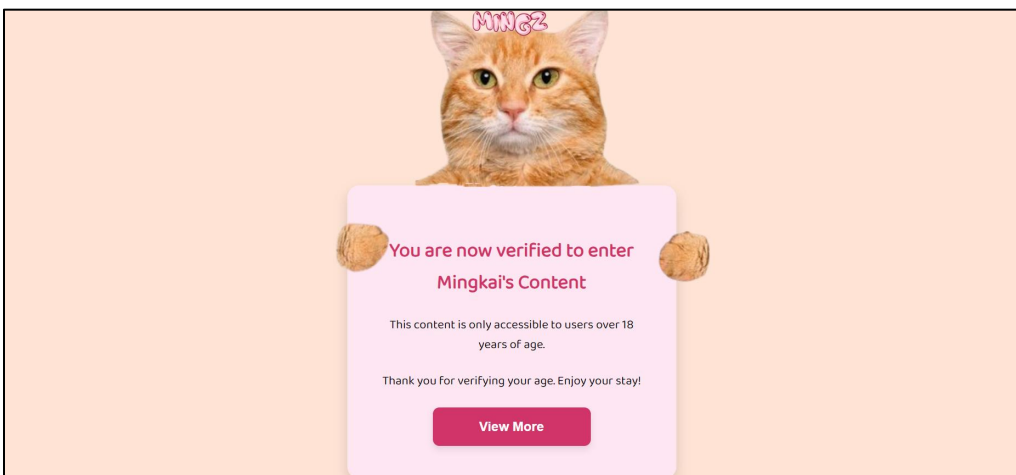
In the afternoon, I hang out with my best friend, Sarah. We chill in bed, or



- If the user's age is greater than 20, it returns the 'adults' view.



A screenshot of a web form titled "Verify Your Age to Log In" held by an orange cat named Mingz. The form is set against a pink background. It includes a text input field with the placeholder "Please Type your age:" and the number "21" entered. Below the input is a dark red "Submit" button.



This ensures that users must be 21 or older to access this particular route, redirecting younger users to an "Access Denied" page.

```
// CheckAge middleware to restricted contents
Route::get('/adults', function () {
    return view('adults');
})->name('adults')->middleware(CheckAge::class.':21');
```

4. Route Groups

We grouped several routes together and applied the LogRequests middleware to log requests to these routes. Additionally, some routes have their specific middleware for age verification (CheckAge), ensuring that certain routes are protected and users' requests are tracked.

```
C: > Users > angel > webdev_lab4 > routes > web.php
1  <?php
2
3  use App\Http\Middleware\LogRequests;
4  use Illuminate\Support\Facades\Route;
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\Mail;
7  use App\Mail\ContactMe;
8  use App\Http\Middleware\CheckAge;
9
10 // Route for displaying the Age verification form
11 Route::get('/', function () {
12     return view('Age');
13 }->name('Age'));
14
15 // Group routes logreq
16 Route::middleware([LogRequests::class])>group(function () {
17     Route::post('/', function (Request $request) {
18         return view('/adults');
19     }->name('age.verify')->middleware(CheckAge::class); //route specific
20
21     Route::get('/homepage', function () {
22         return view('homepage');
23     }->name('homepage');
24
25     Route::get('/homepage/{username?}', function ($username = 'Guest') {
26         return view('homepage', ['username' => $username]);
27     }->where('username', '[a-zA-Z]+')->name('homepage');
```

This structure ensures that all routes within the group are logged by the LogRequests middleware, and specific routes are further protected by CheckAge.

4. LogRequests Middleware

The LogRequests middleware logs each incoming request, including the HTTP method and URL, to a file. This allows monitoring of requests to the application, which is particularly useful for debugging and tracking activity.

```
C: > Users > angel > webdev_lab4 > app > Http > Middleware > LogRequests.php
4
5 Closure;
6 Illuminate\Http\Request;
7 Illuminate\Support\Facades\Storage;
8 Symfony\Component\HttpFoundation\Response;
9
10 ss LogRequests
11
12 /**
13  * Handle an incoming request.
14  *
15  * @param Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
16  */
17 public function handle(Request $request, Closure $next): Response
18 {
19     $logData = 'URL: ' . $request->url() . ' | Method: ' . $request->method() . ' | Timestamp: ' . now()->format('Y-m-d H:i:s') . PHP_EOL;
20     Storage::append('LogReqLab4/log.txt', $logData);
21     return $next($request);
22 }
23
24
```

This logs each request in **log.txt** inside the **storage/app/private/LogReqLab4** directory, capturing the method, URL, and timestamp for every request made to the application.