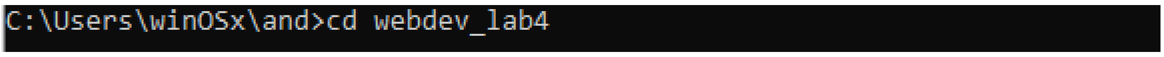
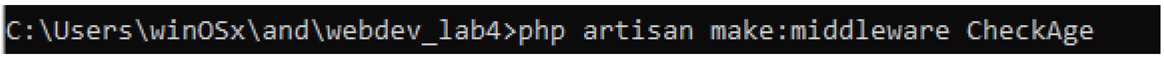
Sarah C. Abane   BSIT 3C   Webdev_Lab4_MIddleware

Part 1: Create and Register New Middleware:

Steps: Open command prompt or git bash and create CheckAge middle and LogRequests middleware

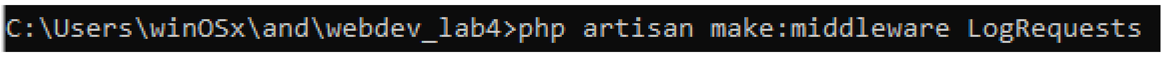


```
C:\Users\winOSx\and>cd webdev_lab4
```



```
C:\Users\winOSx\and\webdev_lab4>php artisan make:middleware CheckAge
```

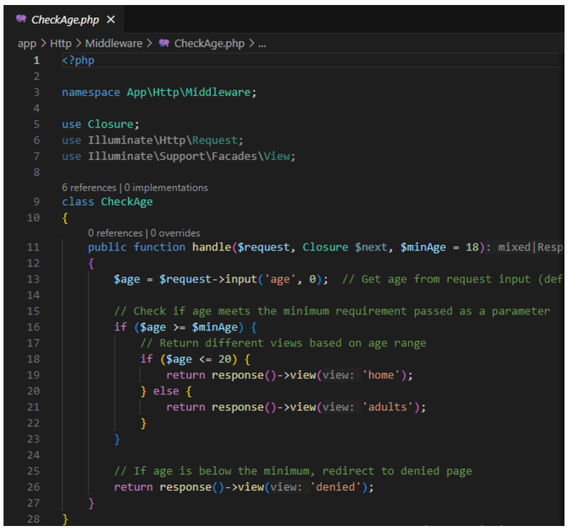
- ❖ This creates a new middleware file at app/Http/Middleware/CheckAge.php.



```
C:\Users\winOSx\and\webdev_lab4>php artisan make:middleware LogRequests
```

- ❖ This creates a new middleware file app/Http/Middleware/LogRequests.php.

Codes:



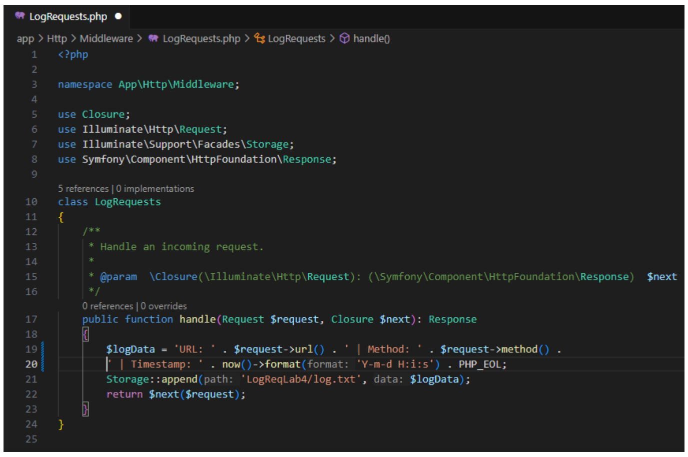
CheckAge.php ×

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

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\View;

6 references | 0 implementations
class CheckAge
{
    0 references | 0 overrides
    public function handle($request, Closure $next, $minAge = 18): mixed|Resp
    {
        $age = $request->input('age', 0);  // Get age from request input (def

        // Check if age meets the minimum requirement passed as a parameter
        if ($age >= $minAge) {
            // Return different views based on age range
            if ($age <= 20) {
                return response()->view(view: 'home');
            } else {
                return response()->view(view: 'adults');
            }
        }

        // If age is below the minimum, redirect to denied page
        return response()->view(view: 'denied');
    }
}
```

CheckAge middleware
The CheckAge middleware checks if a user's age is greater than or equal to 18.

- If the user's age is **greater than or equal to 18**: If the age is **20 or below**, it shows the home view. If the age is **above 20**, it shows the adults view. If the user's age is **less than 18**, it redirects them to a denied view (Access Denied page).
- This ensures that users see different pages based on their age.



LogRequests.php ●

app > Http > Middleware > LogRequests.php > LogRequests > handle()

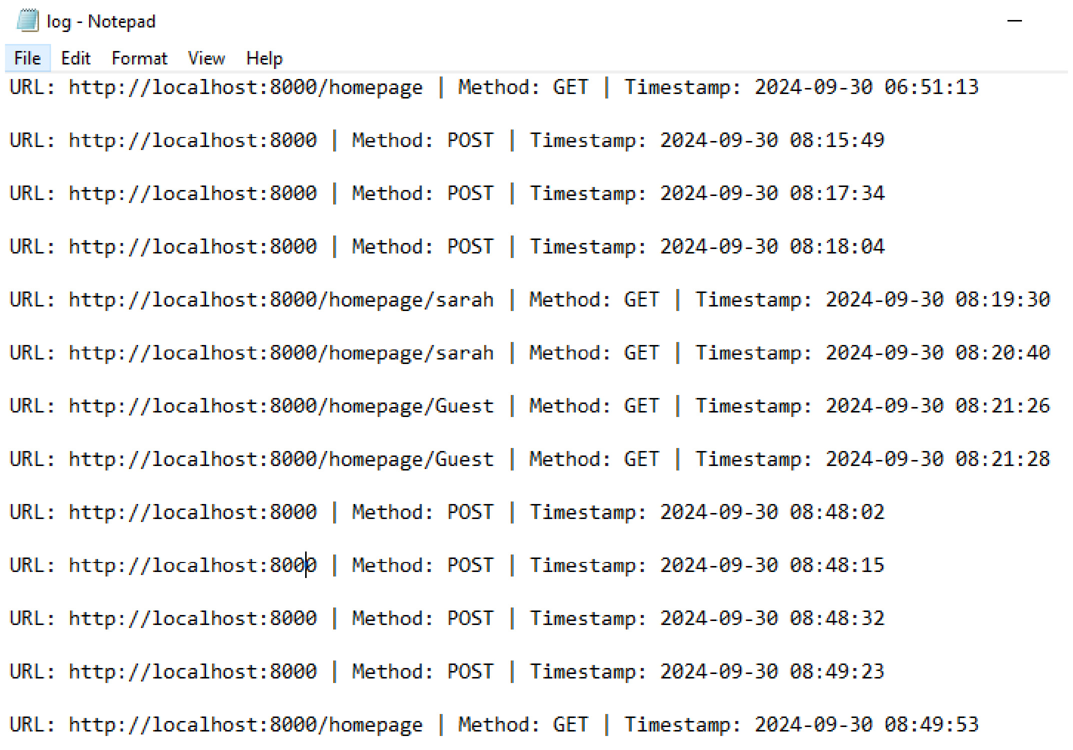
```php
<?php

namespace App\Http\Middleware;

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

5 references | 0 implementations
class LogRequests
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next): Response
    {
        $logData = 'URL: ' . $request->url() . ' | Method: ' . $request->method() .
        ' | Timestamp: ' . now()->format(format: 'Y-m-d H:i:s') . PHP_EOL;
        Storage::append(path: 'LogReqLab4/log.txt', data: $logData);
        return $next($request);
    }
}
```

LogRequests middleware

It captures the URL, HTTP method (e.g., GET, POST), and the current timestamp. The log message is then saved to a file called log.txt which adds new entries to the file. After logging, the middleware passes the request to the next process using $next($request).

- ❖ This middleware helps keep track of every incoming request to our page.

log.txt

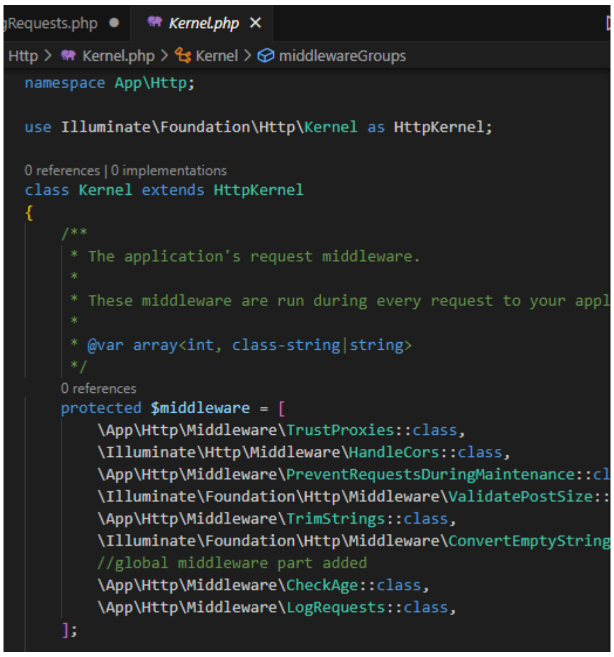


```
URL: http://localhost:8000/homepage | Method: GET | Timestamp: 2024-09-30 06:51:13

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:15:49

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:17:34

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:18:04

URL: http://localhost:8000/homepage/sarah | Method: GET | Timestamp: 2024-09-30 08:19:30

URL: http://localhost:8000/homepage/sarah | Method: GET | Timestamp: 2024-09-30 08:20:40

URL: http://localhost:8000/homepage/Guest | Method: GET | Timestamp: 2024-09-30 08:21:26

URL: http://localhost:8000/homepage/Guest | Method: GET | Timestamp: 2024-09-30 08:21:28

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:48:02

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:48:15

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:48:32

URL: http://localhost:8000 | Method: POST | Timestamp: 2024-09-30 08:49:23

URL: http://localhost:8000/homepage | Method: GET | Timestamp: 2024-09-30 08:49:53
```

❖ The log.txt consists of URL, Method, and Timestamp

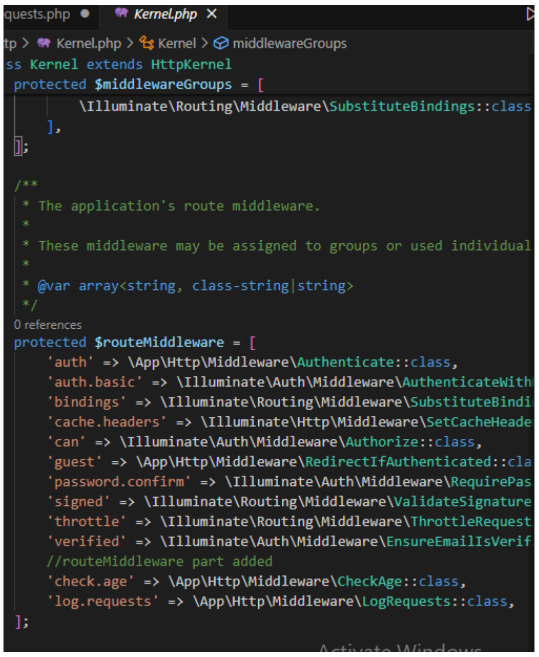
Registration of CheckAge and LogRequests middleware

Global Middleware:



```php
namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

0 references | 0 implementations
class Kernel extends HttpKernel
{
    /**
     * The application's request middleware.
     *
     * These middleware are run during every request to your appl
     *
     * @var array<int, class-string|string>
     */
    0 references
    protected $middleware = [
        \App\Http\Middleware\TrustProxies::class,
        \Illuminate\Http\Middleware\HandleCors::class,
        \App\Http\Middleware\PreventRequestsDuringMaintenance::cl
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyString
        //global middleware part added
        \App\Http\Middleware\CheckAge::class,
        \App\Http\Middleware\LogRequests::class,
    ];
```

Route-Specific Middleware:



```php
ss Kernel extends HttpKernel
 protected $middlewareGroups = [
            \Illuminate\Routing\Middleware\SubstituteBindings::class
        ],
 ];

 /**
  * The application's route middleware.
  *
  * These middleware may be assigned to groups or used individual
  *
  * @var array<string, class-string|string>
  */
 0 references
 protected $routeMiddleware = [
     'auth' => \App\Http\Middleware\Authenticate::class,
     'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWith
     'bindings' => \Illuminate\Routing\Middleware\SubstituteBindi
     'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeade
     'can' => \Illuminate\Auth\Middleware\Authorize::class,
     'guest' => \App\Http\Middleware\RedirectIfAuthenticated::cla
     'password.confirm' => \Illuminate\Auth\Middleware\RequirePas
     'signed' => \Illuminate\Routing\Middleware\ValidateSignature
     'throttle' => \Illuminate\Routing\Middleware\ThrottleRequest
     'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerif
     //routeMiddleware part added
     'check.age' => \App\Http\Middleware\CheckAge::class,
     'log.requests' => \App\Http\Middleware\LogRequests::class,
 ];
```

❖ **Global Middleware**: These middleware will run **on every request** to the application. By adding CheckAge and LogRequests here, it ensures that every request will have its age checked and the request logged. Global middleware applies to all requests.

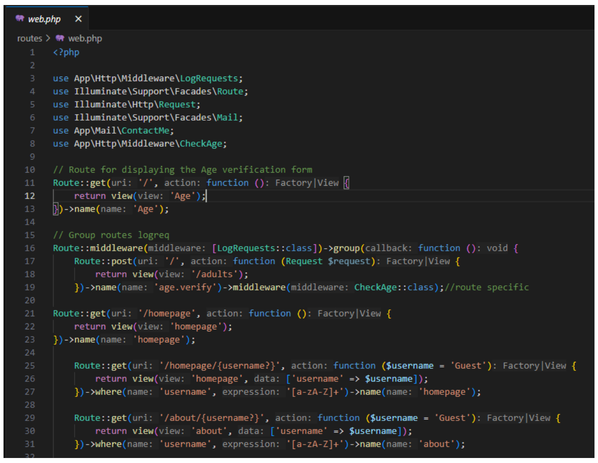
- ❖ **Route Middleware**: These middleware are applied **only to specific routes** where we explicitly include them. This allows us to control which routes should have age checking and logging functionality.

Explanation/Summary:

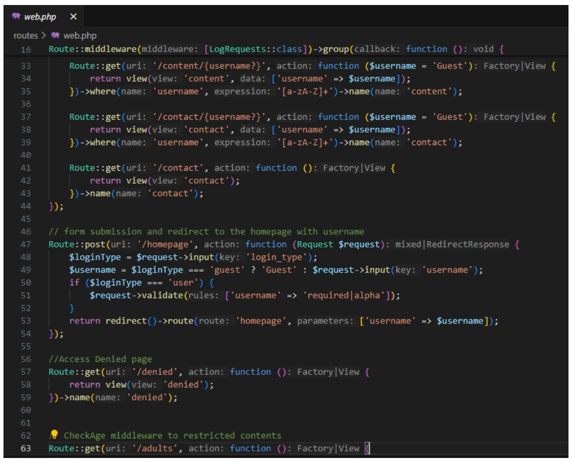
1. Middlewares are used to filter HTTP requests and responses. CheckAge can validate the age of a user, and LogRequests can log incoming requests.
   a. CheckAge: This middleware helps control access to specific content based on the user's age, ensuring that the application responds appropriately to different age groups.
   b. LogRequests: This middleware is essential for tracking and auditing user activity, helping developers understand how the application is being used and troubleshooting any issues that may arise.
2. By registering middleware as both global and route-specific, we can decide whether to apply the middleware universally or only to certain parts of the application, offering flexibility and control over how requests are processed

Part 2: Assign Middleware to Routes:

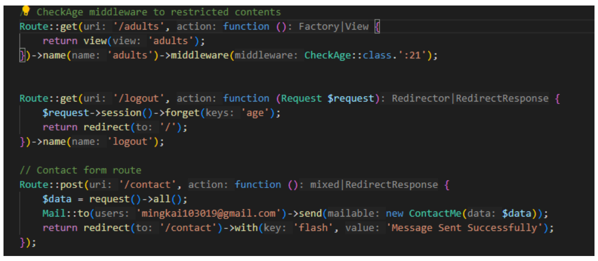
Code: web.php



```
web.php ×
routes > web.php
<?php

use App\Http\Middleware\LogRequests;
use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;
use App\Mail\ContactMe;
use App\Http\Middleware\CheckAge;

// Route for displaying the Age verification form
Route::get(uri: '/', action: function (): Factory|View {
    return view(view: 'Age');
})->name(name: 'Age');

// Group routes logreq
Route::middleware(middleware: [LogRequests::class])->group(callback: function (): void {
    Route::post(uri: '/', action: function (Request $request): Factory|View {
        return view(view: '/adults');
    })->name(name: 'age.verify')->middleware(middleware: CheckAge::class);//route specific

Route::get(uri: '/homepage', action: function (): Factory|View {
    return view(view: 'homepage');
})->name(name: 'homepage');

    Route::get(uri: '/homepage/{username?}', action: function ($username = 'Guest'): Factory|View {
        return view(view: 'homepage', data: ['username' => $username]);
    })->where(name: 'username', expression: '[a-zA-Z]+')->name(name: 'homepage');

    Route::get(uri: '/about/{username?}', action: function ($username = 'Guest'): Factory|View {
        return view(view: 'about', data: ['username' => $username]);
    })->where(name: 'username', expression: '[a-zA-Z]+')->name(name: 'about');
```



```
web.php ×
routes > web.php
Route::middleware(middleware: [LogRequests::class])->group(callback: function (): void {
    Route::get(uri: '/content/{username?}', action: function ($username = 'Guest'): Factory|View {
        return view(view: 'content', data: ['username' => $username]);
    })->where(name: 'username', expression: '[a-zA-Z]+')->name(name: 'content');

    Route::get(uri: '/contact/{username?}', action: function ($username = 'Guest'): Factory|View {
        return view(view: 'contact', data: ['username' => $username]);
    })->where(name: 'username', expression: '[a-zA-Z]+')->name(name: 'contact');

    Route::get(uri: '/contact', action: function (): Factory|View {
        return view(view: 'contact');
    })->name(name: 'contact');
});

// form submission and redirect to the homepage with username
Route::post(uri: '/homepage', action: function (Request $request): mixed|RedirectResponse {
    $loginType = $request->input(key: 'login_type');
    $username = $loginType === 'guest' ? 'Guest' : $request->input(key: 'username');
    if ($loginType === 'user') {
        $request->validate(rules: ['username' => 'required|alpha']);
    }
    return redirect()->route(route: 'homepage', parameters: ['username' => $username]);
});

//Access Denied page
Route::get(uri: '/denied', action: function (): Factory|View {
    return view(view: 'denied');
})->name(name: 'denied');


// CheckAge middleware to restricted contents
Route::get(uri: '/adults', action: function (): Factory|View {
```



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


Route::get(uri: '/logout', action: function (Request $request): Redirector|RedirectResponse {
    $request->session()->forget(keys: 'age');
    return redirect(to: '/');
})->name(name: 'logout');

// Contact form route
Route::post(uri: '/contact', action: function (): mixed|RedirectResponse {
    $data = request()->all();
    Mail::to(users: 'mingkai103019@gmail.com')->send(mailable: new ContactMe(data: $data));
    return redirect(to: '/contact')->with(key: 'flash', value: 'Message Sent Successfully');
});
```

1. **Welcome Page Route:**
   a. Route: /
   b. Purpose: Displays the age verification form (view called Age).
2. **Route Group (with LogRequests Middleware):**

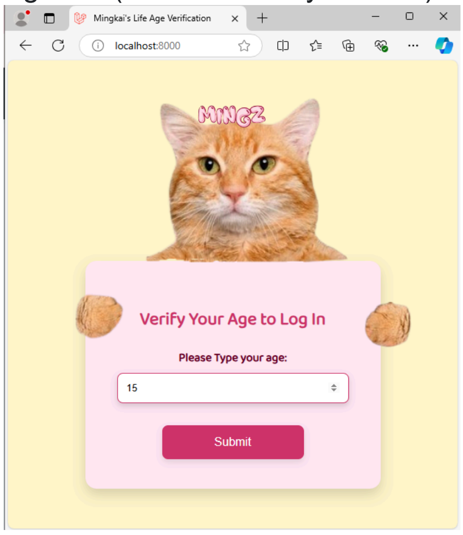
    a. Inside this group, all routes will log the request details (like URL and timestamp) via the LogRequests middleware.
    b. Age Verification Route:
        i. Route: /
        ii. Purpose: After submitting the age form, the CheckAge middleware is applied to ensure the user meets the age requirement. If the user is old enough, they are shown the adults view.
    c. Other Routes (/homepage, /about, /content, etc.):
        i. These pages display content specific to the route, but also benefit from logging due to the group middleware.

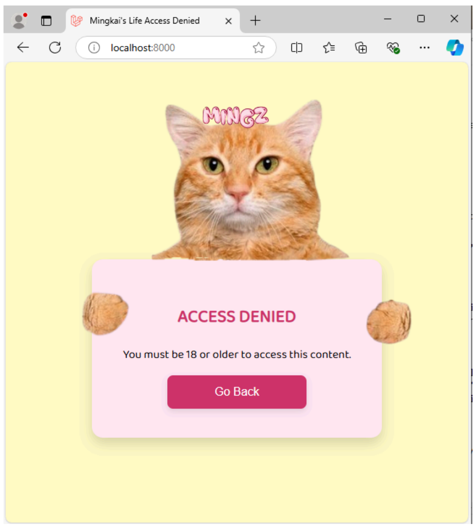
3. **Access Denied Page:**
    a. Route: /denied
    b. Purpose: This page is shown if the user fails the age check.
4. **Restricted Content Route (Adults Only):**
    a. Route: /adults
    b. Purpose: Only users 21 years or older can access this route. The CheckAge middleware checks the age and restricts access if the condition isn't met.

Testing the middleware by simulating different age values in the request

Instances:

Age: 15 (and below 18 years old)

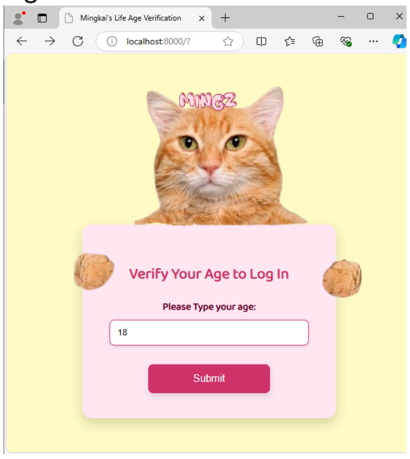


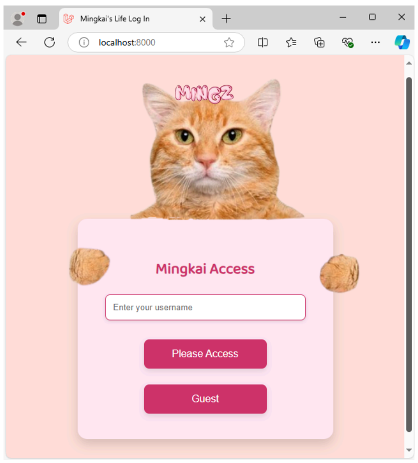


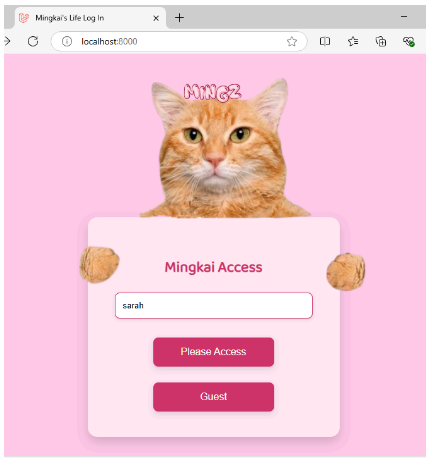
You must be 18 or older to access this content.

❖ If below 18: It goes to access denied page

Age: 18

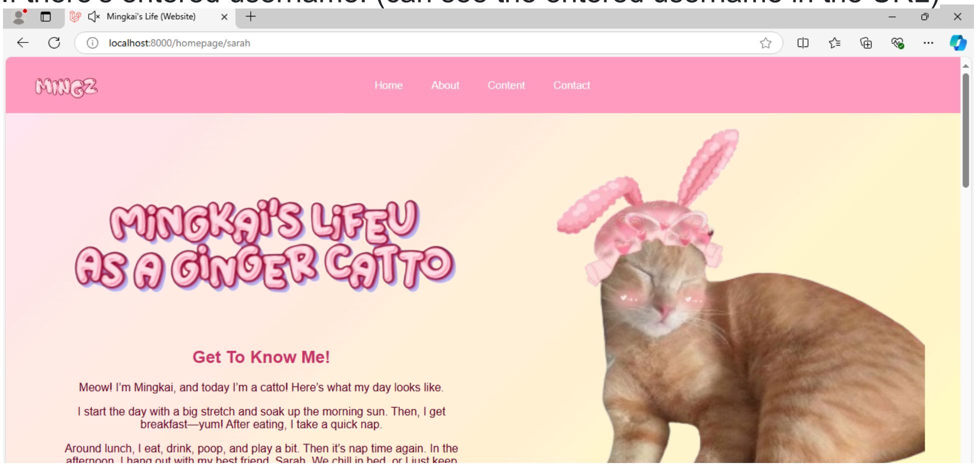






❖ If 18-20 yrs old: Can access the page

If there's entered username: (can see the entered username in the URL)


Mingkai's Life (Website)

localhost:8000/homepage/sarah

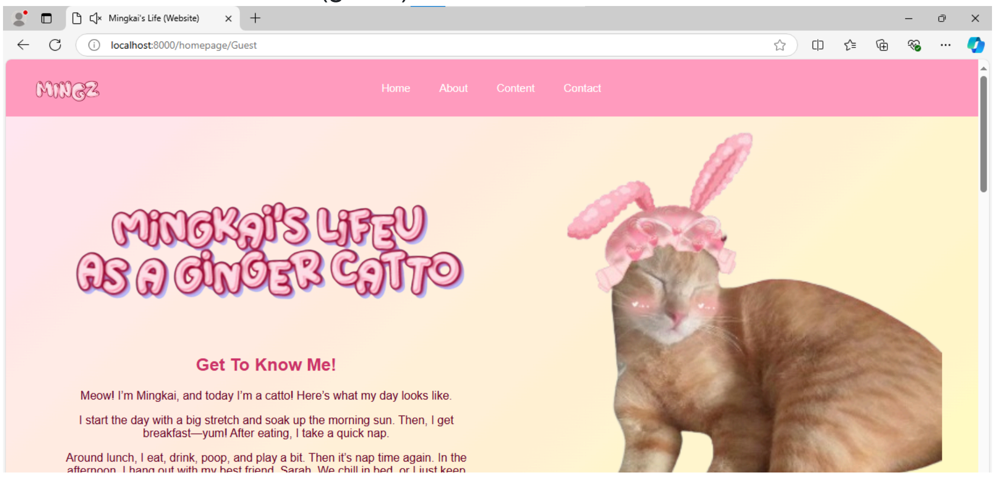
Get To Know Me!

Meow! I'm Mingkai, and today I'm a catto! 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 I just keep

If no entered username: (guest)


Mingkai's Life (Website)

localhost:8000/homepage/Guest

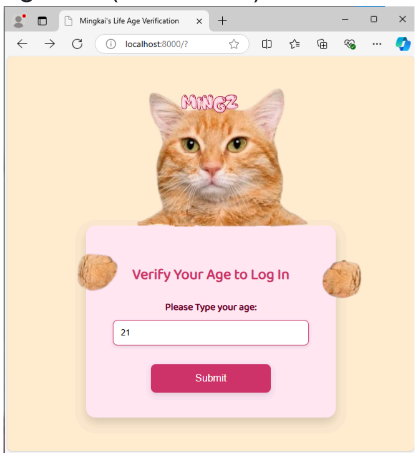
Get To Know Me!

Meow! I'm Mingkai, and today I'm a catto! 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 I just keep

Age: 21 (For Adults)


Mingkai's Life Age Verification

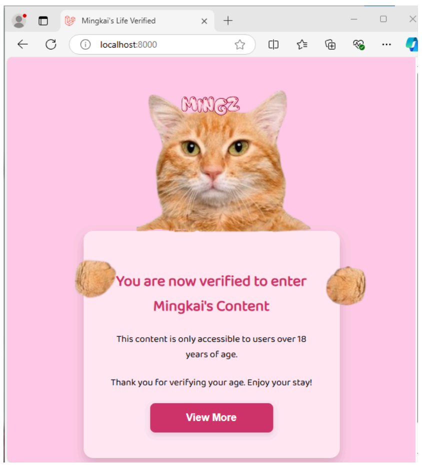
localhost:8000/?

Verify Your Age to Log In

Please Type your age:

21

Submit


Mingkai's Life Verified

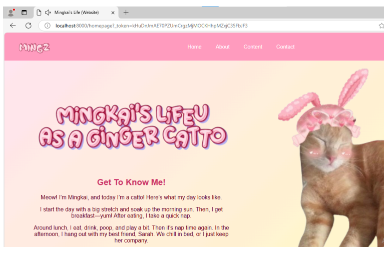
localhost:8000

You are now verified to enter Mingkai's Content

This content is only accessible to users over 18 years of age.

Thank you for verifying your age. Enjoy your stay!

View More



Get To Know Me!

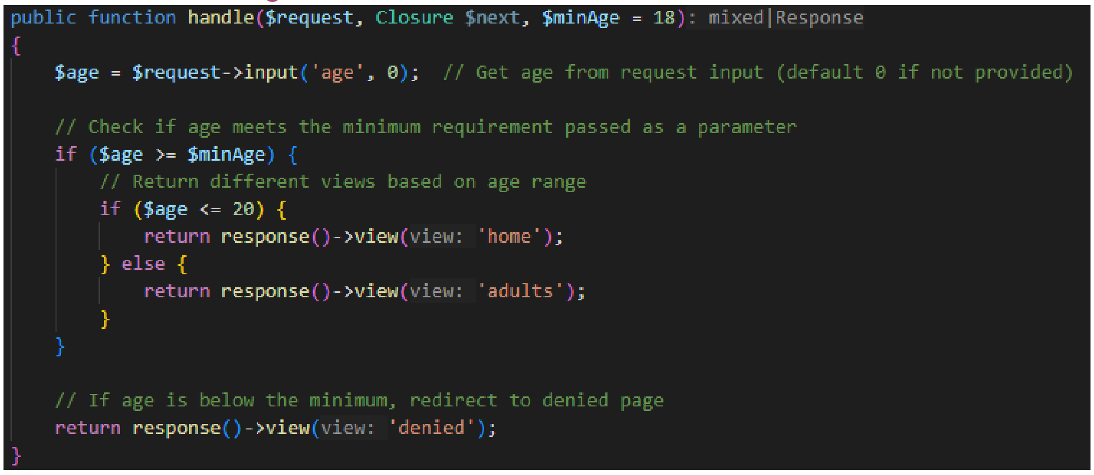
Meow! I'm Mingkai, and today I'm a catto! Here's what my day looks like.

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 I just keep her company.

Part 3: Create Middleware with Parameters:

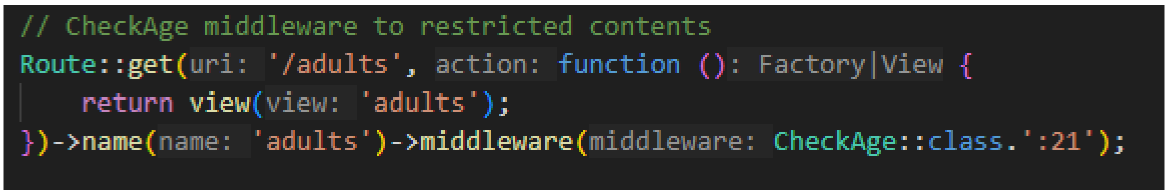
1. **Modified CheckAge Middleware**:



```php
public function handle($request, Closure $next, $minAge = 18): mixed|Response
{
    $age = $request->input('age', 0);  // Get age from request input (default 0 if not provided)

    // Check if age meets the minimum requirement passed as a parameter
    if ($age >= $minAge) {
        // Return different views based on age range
        if ($age <= 20) {
            return response()->view(view: 'home');
        } else {
            return response()->view(view: 'adults');
        }
    }

    // If age is below the minimum, redirect to denied page
    return response()->view(view: 'denied');
}
```

   a. The `CheckAge` middleware now accepts a **parameter** (`$minAge`), which specifies the minimum required age for access.
   b. The middleware checks the user's age (from request input) against the provided minimum age.
   c. If the user meets the age requirement:
      - If the age is between 18 and 20, the user is shown the "home" view.
      - If the age is above 20, the user is shown the "adults" view.
   d. If the user does **not** meet the age requirement, they are redirected to an "Access Denied" page (`denied` view)

2. **New Route with Parameterized Middleware**:



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

   a. The route `/adults` is protected by the `CheckAge` middleware, with a **minimum age requirement of 21** (`middleware(CheckAge::class . ':21')`).
   b. This ensures only users who are 21 or older can access the `/adults` page. Users younger than 21 will be redirected to the "Access Denied" page.

❖ This allows enforcing different age restrictions on various routes based on the parameter provided to the middleware.