PART 1: CREATE AND REGISTER NEW MIDDLWARE

- **Using the command line, create new middleware named CheckAge and LogRequests**

```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Herd\Lab4_Webdev>php artisan make:middleware CheckAge

  INFO  Middleware [C:\Users\user\Herd\Lab4_Webdev\app\Http\Middleware\CheckAge.php] created successfully.


C:\Users\user\Herd\Lab4_Webdev>php artisan make:middleware Logrequests

  INFO  Middleware [C:\Users\user\Herd\Lab4_Webdev\app\Http\Middleware\Logrequests.php] created successfully.
```

- **The CheckAge middleware should check if a user's age is greater than or equal to 18. If the age does not meet the condition, redirect the user to an "Access Denied" page.**

CODE:

CheckAge.php U

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

```php
<?php

namespace App\Http\Middleware;

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

class CheckAge
{
    public function handle($request, Closure $next, $minAge = 18)
    {
        $age = $request->input('age', 0);
        if ($age >= 18 && $age <= 20) {
            return response()->view('home');
        }
        if ($age >= 21) {
            return response()->view('adults');
        }
        return response()->view('denied');
    }
}
```

EXPLANATION:

The middleware checks the user's age from the request. If no age is provided, it assumes the age is 0.

- Conditions:
  - If the user is 18 or older, it shows them different pages:
    - 18 to 20 years old: They see the home page.
    - Older than 20: They see the adults page.
  - If the user is under 18, they are sent to an "Access Denied" page.

- **LogRequests should log the details of all HTTP requests to a file called log.txt, include the URL method, and timestamp**

CODE:

LogRequests.php U

app > Http > Middleware > LogRequests.php > ...

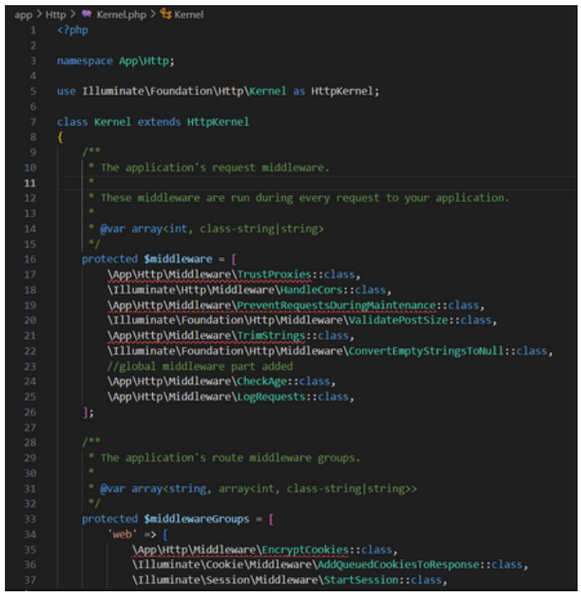
```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
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
    {
        $logData = 'URL: ' . $request->url() . ' | Method: ' . $request->method() . ' | Timestamp: ' . now()->format('Y-m-d H:i:s') . PHP_EOL;
        Storage::append('LogReqLab3/log.txt', $logData);
        return $next($request);
    }
}
```

EXPLANATION:

- Logs Request Details: The middleware records the URL, HTTP method (GET, POST, etc.), and the time the request was made.
- Creates a Log Entry: It formats this information into a line of text, like this:
  - "URL: [requested URL] | Method: [HTTP method] | Timestamp: [current time]"
- Saves to a File: This log entry is added to a file called **log.txt** in the **LogReqLab4** folder. Every request gets added to the file without deleting the previous logs.
- Continues Processing: After logging the request, it allows the application to handle the request.

- **Register the middleware in the app/Http/Kernel.php file under the appropriate section.**
  - **Register middleware both as global middleware and route-specific middleware**

CODE:



```php
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

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

    /**
     * The application's route middleware groups.
     *
     * @var array<string, array<int, class-string|string>>
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            \Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \App\Http\Middleware\VerifyCsrfToken::class,
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],

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

    /**
     * The application's route middleware.
     *
     * These middleware may be assigned to groups or used individually.
     *
     * @var array<string, class-string|string>
     */
    protected $routeMiddleware = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
        'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
        'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
        //routeMiddleware part added
        'check.age' => \App\Http\Middleware\CheckAge::class,
        'log.requests' => \App\Http\Middleware\LogRequests::class,
    ];
}
```

EXPLANATION:

1. Global Middleware:
   - In the protected $middleware section, we have registered middleware that runs on every request to our website. We added:
     - CheckAge::class: This checks if the user is 18 or older.
     - LogRequests::class: This logs request details like the URL, method, and timestamp.

   - This means that every time someone visits our website, their age will be checked, and the request will be logged automatically.
2. Route-Specific Middleware:
   - In the protected $routeMiddleware section, we registered middleware that we can assign to specific routes (individual pages or actions). We added:
     - 'check.age' => \App\Http\Middleware\CheckAge::class: This lets us use this middleware only on specific routes if I need to.
     - 'log.requests' => \App\Http\Middleware\LogRequests::class: This allows us to apply logging to certain routes only.

PART 2: ASSIGN MIDDLEWARE TO ROUTES:

- **Create a route group that assigns the CheckAge middleware to a specific route.**
  - **Ensure the middleware applies to the following routes:**

CODE:

```php
<?php

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

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

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

Route::get('/home', function () {
    return view('home');
})->name('home');

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

    Route::get('/about/{username?}', function ($username = 'Guest') {
        return view('about', ['username' => $username]);
    })->where('username', '[a-zA-Z]+')->name('about');

    Route::get('/content/{username?}', function ($username = 'Guest') {
        return view('content', ['username' => $username]);
    })->where('username', '[a-zA-Z]+')->name('content');

    Route::get('/contact/{username?}', function ($username = 'Guest') {
        return view('contactPage', ['username' => $username]);
    })->where('username', '[a-zA-Z]+')->name('contactPage');
```

```php
    Route::get('/contact-us', function () {
        return view('contactForm');
    })->name('contactForm');
});

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

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


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


Route::get('/logout', function (Request $request) {
    $request->session()->forget('age');
    return redirect('/');
})->name('logout');
```

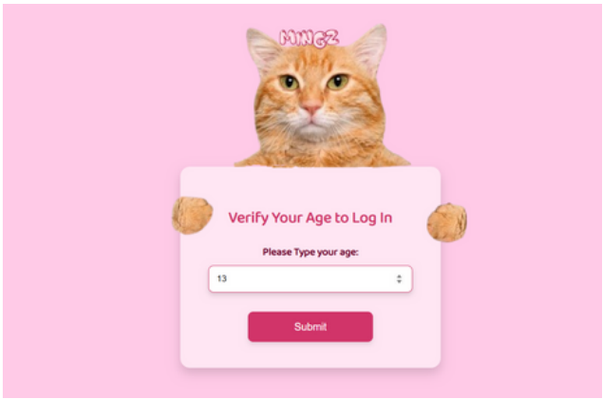
EXPLANATION:

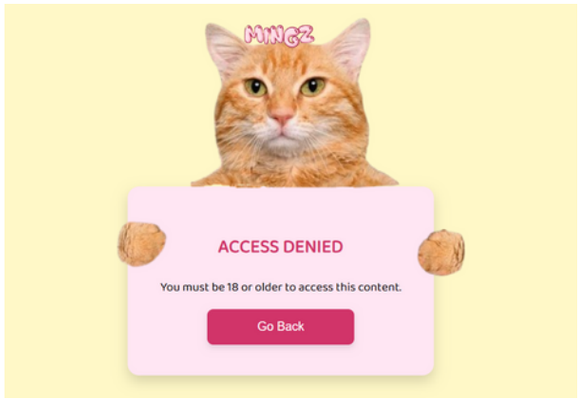
1. Grouping Routes:
   - The code uses a route group to apply the LogRequests middleware to several routes at once. This means that every time someone accesses any of these routes, their request will be logged automatically.
2. Defining Routes:
   - Inside the route group, there's a route for verifying a user's age when they submit the age verification form. This route also uses the CheckAge middleware:
     - The CheckAge middleware checks if the user meets the age requirement before allowing access to the /adults view.
3. Other Routes:
   - The group also includes routes for the homepage, about page, content, and contact page, which are accessible without additional age checks.

- **Test the middleware by simulating different age values in the request (Test various scenarios where the middleware passes or fails the request).**

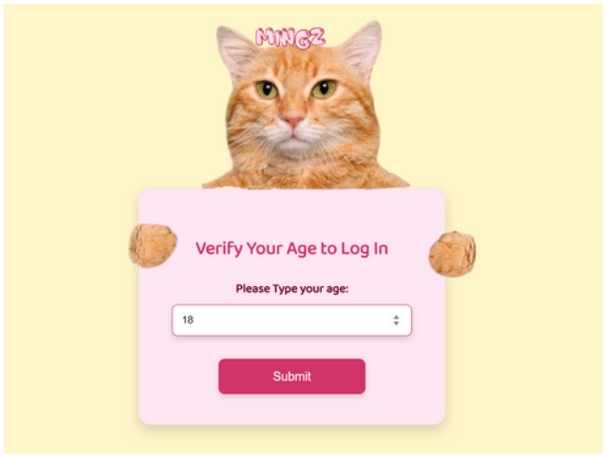
OUTPUT:

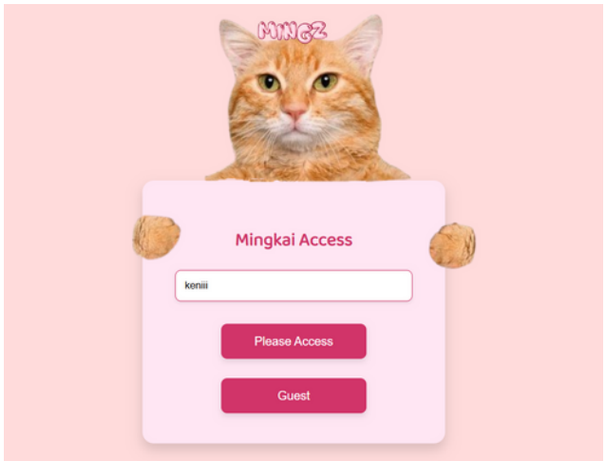
Below 18 years old

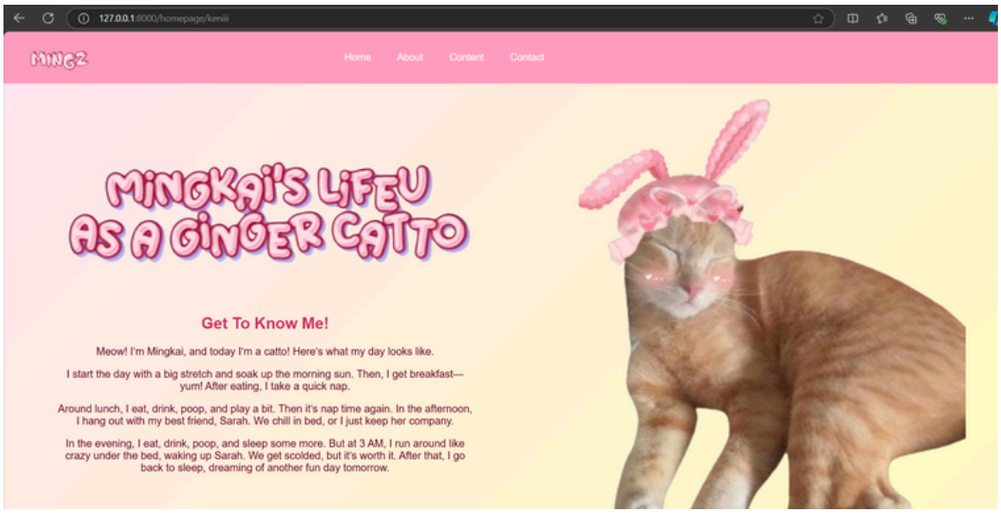




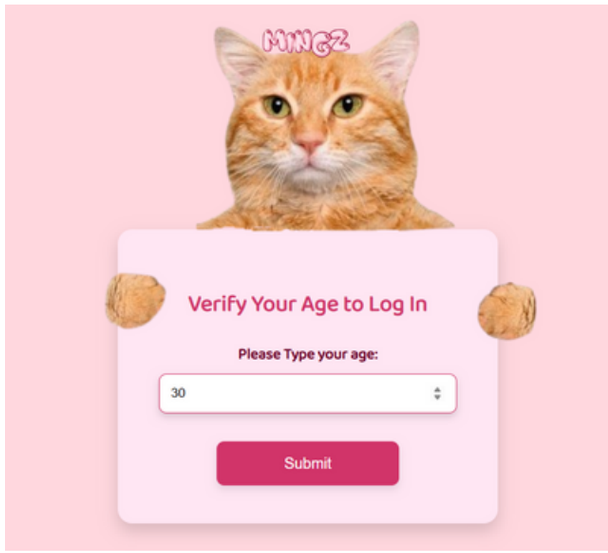
18-20 years old

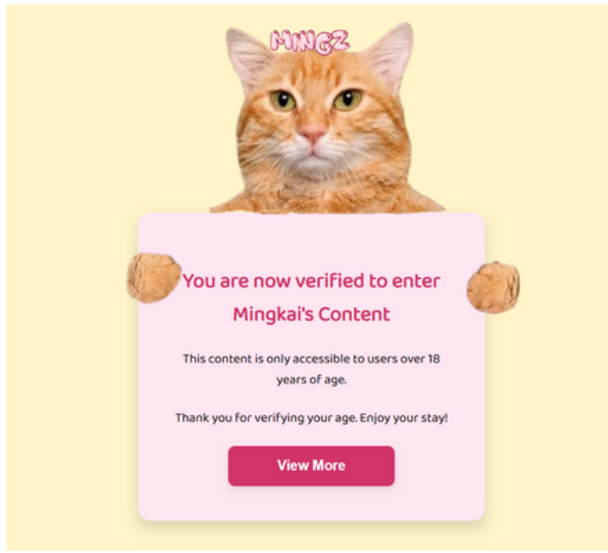


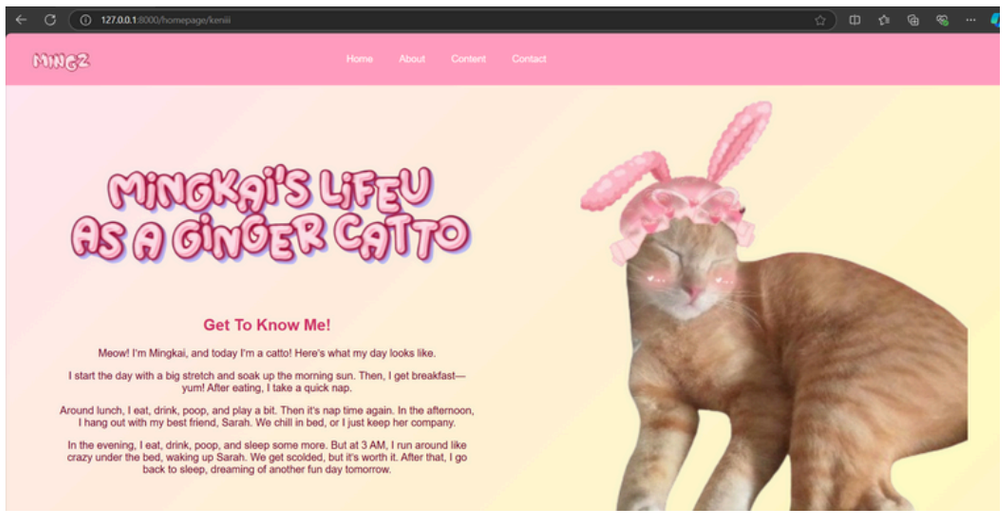




Adults or 20 years old above







PART 3: CREATE MIDDLEWARE WITH PARAMETERS:

- **Modify the CheckAge middleware to accept a parameter (e.g., the minimum age requirement).**

CODE:

CheckAge.php M

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

```php
<?php

namespace App\Http\Middleware;

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

class CheckAge
{
    public function handle($request, Closure $next, $minAge = 18)
    {
        $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('home');
            } else {
                return response()->view('adults');
            }
        }

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

EXPLANATION:

1. Accepting a Minimum Age:
   - The middleware can now accept a parameter called $minAge. This allows you to set different minimum age requirements for different routes.
2. Getting the User's Age:
   - It gets the user's age from the request using $request->input('age', 0). If the age isn't provided, it defaults to 0.
3. Age Check:
   - If the user's age is equal to or greater than $minAge:
     - If they are 20 or younger, it shows the home view.
     - If they are older than 20, it shows the adults view.
     - If the user is younger than the minimum age, it shows the denied page.

- **Create a new route that assigns the middleware with a parameter to enforce a different age restriction (e.g., 21 years old).**

CODE:

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

EXPLANATION:

1. Route Details:
   - This line of code creates a new route for the URL /adults.
   - When a user visits this URL, it will show the adults view.
2. Using Middleware:
   - The route uses the CheckAge middleware to check if the user is at least 21 years old.
   - The :21 part is the parameter passed to the middleware, specifying the minimum age required.
3. What Happens When Accessing the Route:
   - When a user tries to access /adults, the middleware will:
     - Check the user's age provided in the request.
     - If the age is 21 or older, the user will see the adults view.
     - If the age is below 21, the user will be redirected to the home page.
     - If the user is younger than the minimum age which is 18 below, it shows the denied page.
4. Summary
   - In summary, this route uses the CheckAge middleware to enforce an age restriction of 21 years for accessing the /adults page. It ensures that only users who meet this age requirement can view the content.