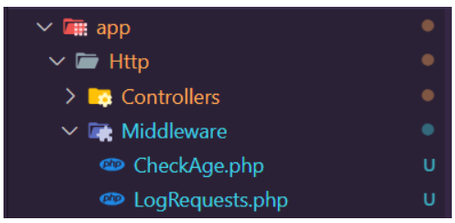
Janine Ishe M. Matamorosa **BSIT 3C**

Part 1: Create and Register New Middleware:

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



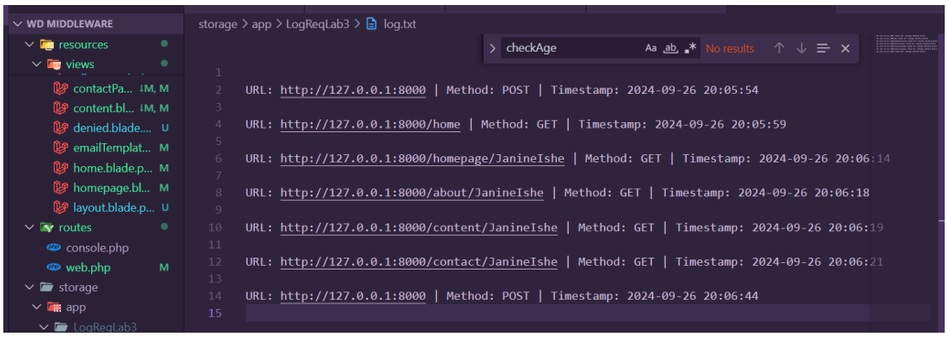
I have successfully created the two Middleware.

- 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 snippet for the middleware CheckAge.

```php
public function handle($request, Closure $next, $minAge = 18)
 {
     $age = $request->input('age', 0);
     if ($age >= $minAge) {
         return $next($request);
     }
     return response()->view('denied');
 }
```

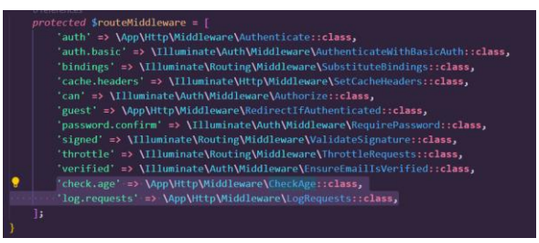
- LogRequests should log the details of all HTTP requests to a file called log.txt, include the URL method, and timestamp




```php
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);
}
```

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

```php
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,
    \App\Http\Middleware\CheckAge::class,
    \App\Http\Middleware\LogRequests::class,
];
```



```php
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,
    'check.age' => \App\Http\Middleware\CheckAge::class,
    'log.requests' => \App\Http\Middleware\LogRequests::class,
];
```

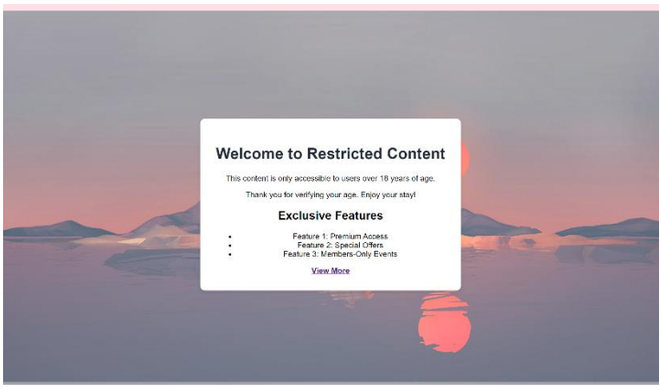
In this part I have already registered the middleware on global and routemiddleware. Below is the route-specific middleware.

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

Part 2: Assign Middleware to Routes:

- Create a route group that assigns the CheckAge middleware to a specific route.

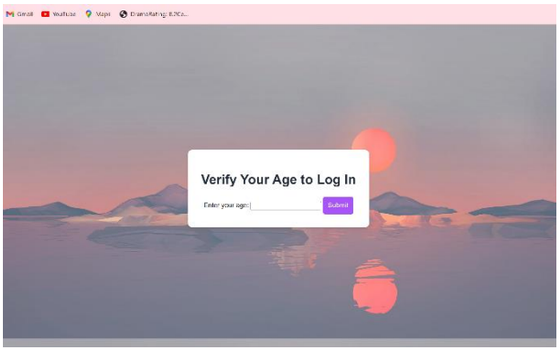
```php
// CheckAge middleware to restricted contents
Route::get(uri: '/adults', action: function (): Factory|View {
    return view(view: 'adults');
})->name(name: 'adults')->middleware(middleware: CheckAge::class.':21');
```



This is the root and it will first ask for the age of the user, If the user is <18 then it will show access denied since the website is created for ages 18 and up.

This snippet below links to the middleware so that it can filter the age.

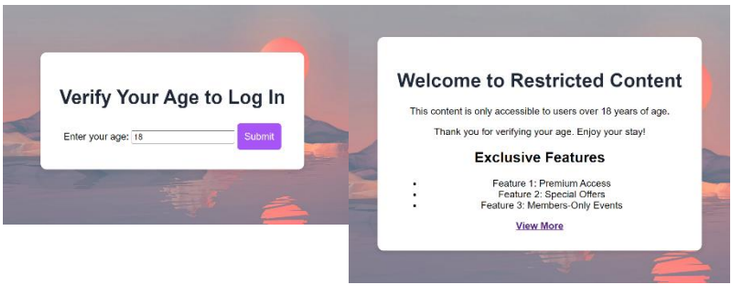
```php
Route::post(uri: '/', action: function (Request $request): Factory|View {
    return view(view: '/adults');
})->name(name: 'age.verify')->middleware(middleware: CheckAge::class);//route specific
```



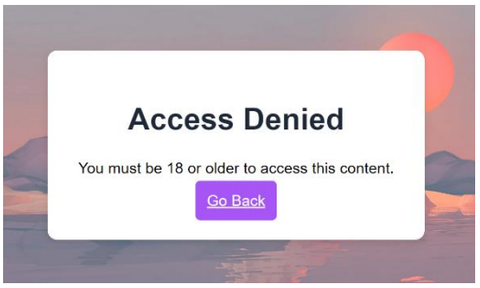
- Ensure the middleware applies to the following routes:
    - A route that loads a welcome page.
    - A route that loads a dashboard page.

In our case our middleware applies to the root which is the **Age.blade.php** which verifies or filters the age and the **Adult.blade.php.**

- Test the middleware by simulating different age values in the request (Test various scenarios where the middleware passes or fails the request).
- **User is above 18**



User is below 18



Part 3: Create Middleware with Parameters:

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

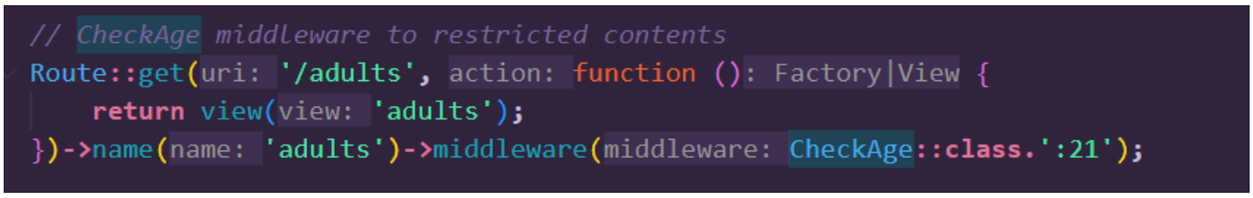
```php
9 references | 0 implementations
class CheckAge
{
    0 references | 0 overrides
    public function handle($request, Closure $next, $minAge = 18): mixed|Response
    {
        $age = $request->input('age', 0);
        if ($age >= 18 && $age <= 20) {
            return response()->view(view: 'home');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

In this middleware, I implemented age verification to control access to certain routes based on the user's age input. The logic is designed to direct users based on their age, specifically focusing on those who are 18 to 20 years old and those who are 21 and older.

If the user is aged 18, 19, or 20, I redirect them to the home view. This means that they can access the home page directly.

If the user is 21 or older, they are directed to the adults view. The adult view shows one view that can only be seen by ages 21 and above.

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



I created a route that uses the CheckAge middleware to enforce age restrictions for accessing certain content. Specifically, I defined a route for the /adults path, which returns a view named adults. This route is named adults for easy reference throughout the application.