



BICOL UNIVERSITY

COLLEGE OF SCIENCE

Legaspi City



Middleware

LARAVEL ACTIVITY

RICHARD D. BILAN JR.

BSIT 3-C

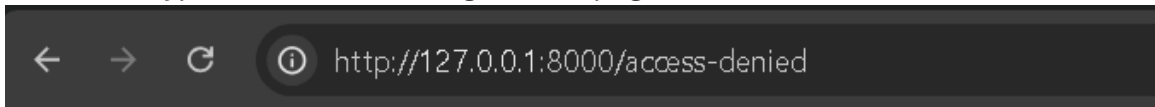
It Elec 1 - Web Development

LARAVEL

Firs is this is our Laravel works in this activity

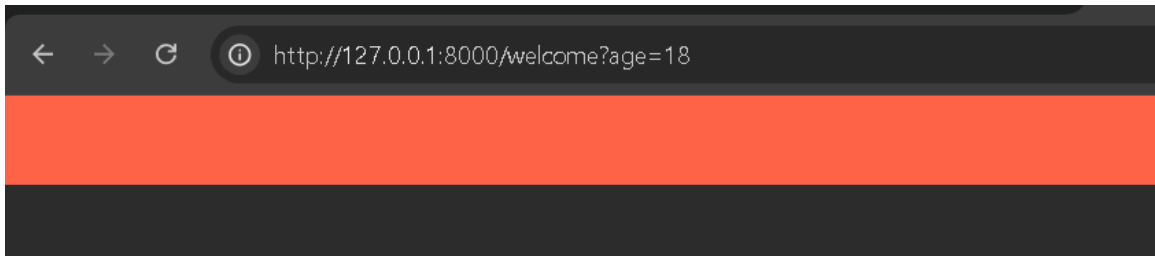


If the user Type less than 18 it will go to this page or the Access restriction

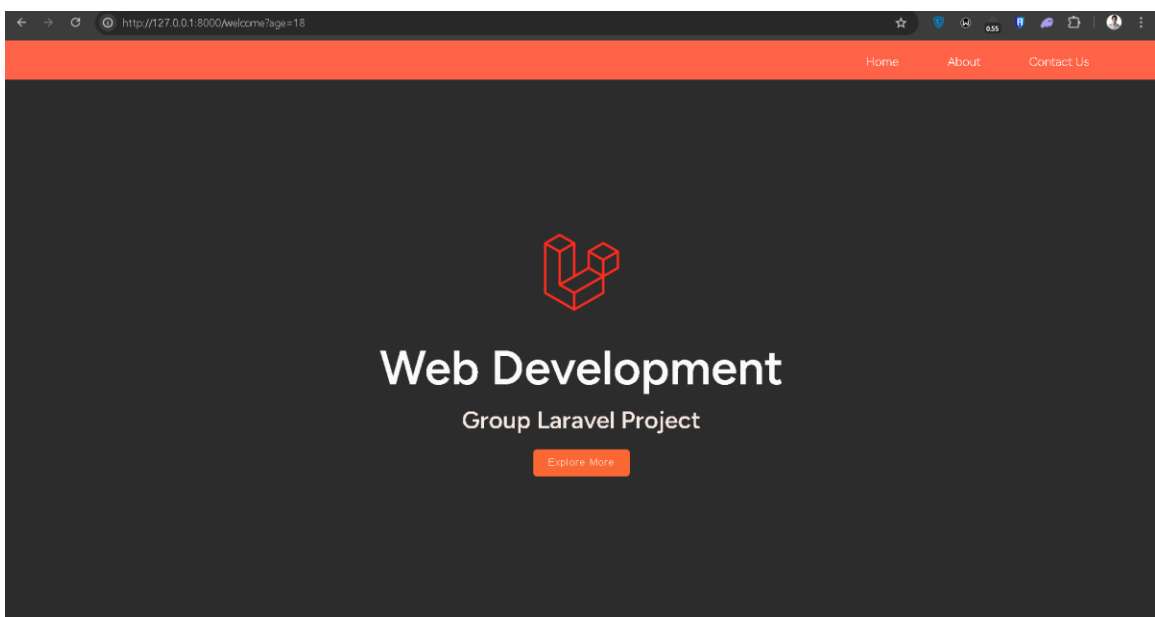


Access Denied PATAL MINOR KA!

Sorry po Sir pikon na po kase kami sa part nayan. But that's how it works when the user is minor



If the user is Legally in age there is no restriction the user will proceed to the Project



If the user is Over age or over 21 yrs old it will directly go to Access Denied



Access Restricted GURANG KANA! DAE KANA MAG KASTA!

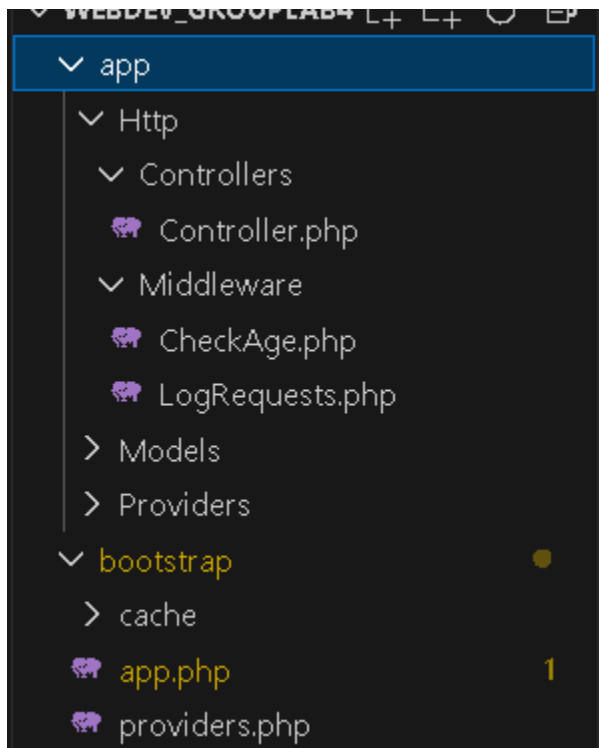
Sorry po ulit sir sa trippings HASAHS gutom na kami nyan. Anyways this is the page where the user is over age.

In Log request all the data or information that the user input will directly store ini this Log.txt

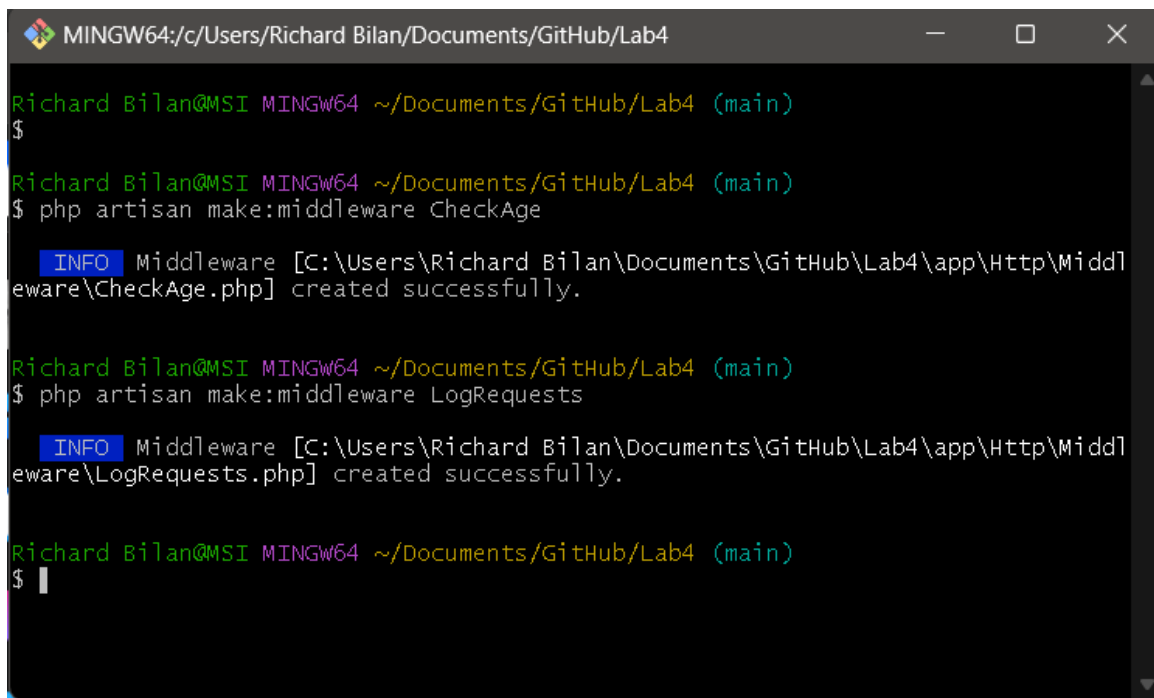
This includes the time, date, age, Type of Restrictions.

```
[2024-10-06 12:08:18] GET: http://127.0.0.1:8000
[2024-10-06 12:08:37] GET: http://127.0.0.1:8000/restricted-dashboard
[2024-10-06 13:26:33] GET: http://127.0.0.1:8000/access-denied
[2024-10-06 21:45:00] GET: http://127.0.0.1:8000
[2024-10-06 21:45:05] GET: http://127.0.0.1:8000/restricted-dashboard
[2024-10-06 21:45:14] GET: http://127.0.0.1:8000/?age=18
[2024-10-06 21:45:18] GET: http://127.0.0.1:8000/?age=17
[2024-10-06 21:45:26] GET: http://127.0.0.1:8000/access-denied
[2024-10-06 21:58:22] GET: http://127.0.0.1:8000/restricted-dashboard, Age: N/A
[2024-10-06 21:58:46] GET: http://127.0.0.1:8000/restricted-dashboard, Age: N/A
[2024-10-06 21:59:17] GET: http://127.0.0.1:8000/access-denied, Age: N/A
[2024-10-06 22:22:38] GET: http://127.0.0.1:8000/welcome?age=19, Age: 19
[2024-10-06 22:22:41] GET: http://127.0.0.1:8000/welcome?age=19, Age: 19
[2024-10-06 22:22:45] GET: http://127.0.0.1:8000/welcome?age=21, Age: 21
[2024-10-06 22:22:45] GET: http://127.0.0.1:8000/restricted-dashboard
```

This is the Middleware we Created.



```
php artisan make:middleware CheckAge
php artisan make:middleware LogRequests
```



```
MINGW64:/c/Users/Richard Bilan/Documents/GitHub/Lab4
Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$
Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php artisan make:middleware CheckAge

[INFO] Middleware [C:\Users\Richard Bilan\Documents\GitHub\Lab4\app\Http\Middleware\CheckAge.php] created successfully.

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php artisan make:middleware LogRequests

[INFO] Middleware [C:\Users\Richard Bilan\Documents\GitHub\Lab4\app\Http\Middleware\LogRequests.php] created successfully.

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$
```

After downloading the Middleware
we Installed Composer

```
MINGW64:/c/Users/Richard Bilan/Documents/GitHub/Lab4

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php artisan make:middleware CheckAge

  INFO  Middleware [C:\Users\Richard Bilan\Documents\GitHub\Lab4\app\Http\Middleware\CheckAge.php] created successfully.

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php artisan make:middleware LogRequests

  INFO  Middleware [C:\Users\Richard Bilan\Documents\GitHub\Lab4\app\Http\Middleware\LogRequests.php] created successfully.

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php artisan --version
Laravel Framework 11.21.0

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ composer create-project --prefer-dist laravel/laravel new_project
bash: composer: command not found

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ AC

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ AC

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 2.7.9) successfully installed to: C:\Users\Richard Bilan\Documents\GitHub\Lab4\composer.phar
Use it: php composer.phar

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ php -r "unlink('composer-setup.php');"

Richard Bilan@MSI MINGW64 ~/Documents/GitHub/Lab4 (main)
$ █
```

After downloading the composer we begin to code the Given Activity First is we Fix the checkage for its Condition

```
env M CheckAge.php X Release Notes: 1.94.0
app > Http > Middleware > CheckAge.php > ...
1 <?php
2
3 // app/Http/Middleware/CheckAge.php
4
5 namespace App\Http\Middleware;
6
7 use Closure;
8 use Illuminate\Http\Request;
9 use Carbon\Carbon;
10
11 3 references | 0 implementations
12 class CheckAge
13 {
14     0 references | 0 overrides
15     public function handle(Request $request, Closure $next, $minAge): mixed
16     {
17         // Log the request before redirecting
18         $logData = sprintf(
19             format: "[%s] %s: %s, Age: %s\n",
20             values: Carbon::now()->format(format: 'Y-m-d H:i:s'),
21             $request->method(),
22             $request->fullurl(),
23             $request->query(key: 'age', default: 'N/A')
24         );
25
26         file_put_contents(filename: storage_path(path: 'logs/log.txt'), data: $logData, flags: FILE_APPEND);
27
28         if ($request->age < $minAge) {
29             return redirect(to: 'access-denied');
30         } elseif ($request->age == 21) {
31             return redirect(to: 'restricted-dashboard');
32         }
33
34         return $next($request);
35     }
36 }
```

The CheckAge middleware is designed to intercept incoming requests and determine whether a user meets a specific age requirement before allowing access to certain routes.

Also,, It achieves this by first logging each request's details, including the timestamp, request method (e.g., GET, POST), the full URL, and the user's provided age. The middleware writes this information into a log file for future reference.

```
.env M app.php X Release Notes: 1.94.0
bootstrap > app.php
1
2 <?php
3
4 use Illuminate\Foundation\Application;
5 use Illuminate\Foundation\Configuration\Exceptions;
6 use Illuminate\Foundation\Configuration\Middleware;
7
8 return Application::configure(basePath: dirname(path: __DIR__))
9     ->withRouting(
10         web: __DIR__.'/../routes/web.php',
11         commands: __DIR__.'/../routes/console.php',
12         health: '/up',
13     )
14     ->withMiddleware(callback: function (Middleware $middleware):
15         //
16     )
17     ->withExceptions(using: function (Exceptions $exceptions): vo
18         //
19     )->create();
20
21 $app->middleware([
22     App\Http\Middleware\CheckAge::class,
23     App\Http\Middleware\LogRequests::class,
24 ]);
25
26
```

Instead of using the Kernel.php for middleware management, this is what we use bcs in our case the Kernel is not reading any kinds of middleware. these part of middleware are likely responsible for handling user age validation and logging HTTP requests, respectively, as part of the app's workflow.

Global Middleware Group (LogRequests)

```
.env M web.php X Release Notes: 1.94.0
routes > web.php
3 use Illuminate\Support\Facades\Route;
4
5 use App\Http\Middleware\CheckAge;
6 use App\Http\Middleware\LogRequests;
7
8 Route::middleware([LogRequests::class])>group(callback: function (): void {
9     Route::get(uri: '/', action: function (): Factory|View {
10         return view(view: 'welcome');
11     }->name(name: 'home'));
12
13     Route::get(uri: '/contactus', action: function (): Factory|View {
14         return view(view: 'contactus');
15     }->name(name: 'contact'));
16
17     Route::get(uri: '/about', action: function (): Factory|View {
18         return view(view: 'about');
19     }->name(name: 'about'));
20
21     Route::get(uri: '/restricted-dashboard', action: function (): string {
22         return "Access Restricted GURANG KANA! DAE KANA MAG KASTA!";
23     });
24
25     Route::get(uri: '/access-denied', action: function (): string {
26         return "Access Denied PATAL MINOR KA!";
27     });
28 });
29
30 // Apply CheckAge middleware with a minimum age parameter
31 Route::middleware([CheckAge::class.':18'])>group(callback: function (): void {
32     Route::get(uri: '/welcome', action: function (): Factory|View {
33         return view(view: 'welcome');
34     }->name(name: 'welcome'));
35
36     Route::get(uri: '/dashboard', action: function (): Factory|View {
37         return view(view: 'dashboard');
38     }->name(name: 'dashboard'));
```

The first group of routes uses the LogRequests middleware. This middleware will log details of HTTP requests that access these routes, including the URL and timestamp.

The second group of routes uses the CheckAge middleware with a minimum age parameter of 18. This means that the user's age will be checked, and if it is below 18, they will be redirected (likely to /access-denied or another specified page).

LogRequests middleware is applied to the first group of routes to track activity.

CheckAge middleware ensures that only users meeting the age requirement can access the welcome and dashboard routes.


```

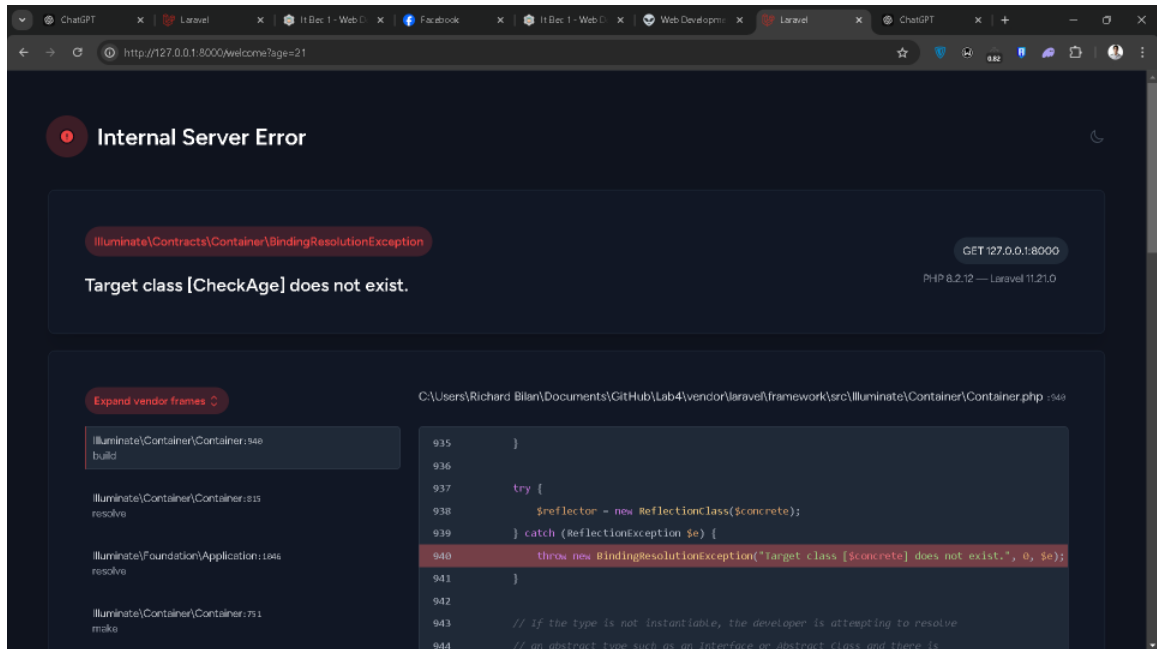
app > Http > Middleware > LogRequests.php > LogRequests > handle()
1  <?php
2
3  // app/Http/Middleware/LogRequests.php
4
5  namespace App\Http\Middleware;
6
7  use Closure;
8  use Illuminate\Http\Request;
9  use Carbon\Carbon;
10
11  3 references | 0 implementations
12  class LogRequests
13  {
14      0 references | 0 overrides
15      public function handle(Request $request, Closure $next): mixed
16      {
17          return $next($request);
18      }
19
20      0 references | 0 overrides
21      public function terminate(Request $request, $response): void
22      {
23          $logData = sprintf(
24              format: "[%s] %s: %s\n",
25              values: Carbon::now()->format(format: 'Y-m-d H:i:s'),
26              $request->method(),
27              $request->fullUrl()
28          );
29
30          file_put_contents(filename: storage_path(path: 'logs/log.txt'), data: $logData, flags: FILE_APPEND);
31      }
32  }

```

This middleware tracks every request made to the routes it is applied to, logging the time, method, and URL. This is useful for monitoring user activity, debugging, or auditing purposes.

The `handle` method is the standard part of middleware that processes the request before it reaches the route or controller. In this case, the `handle` method simply calls `$next($request)` to allow the request to continue without any modification. It doesn't log anything at this stage.

The **terminate** method is where the logging happens. The `terminate` method is executed after the response is sent to the browser. It logs details of the request, such as the request method (e.g., GET, POST) and the full URL that was accessed.



This is the most Difficult part and very time consuming we ever encounter in this entire activity but one of our member managed to Figure it out after that the Project goes smoothly.