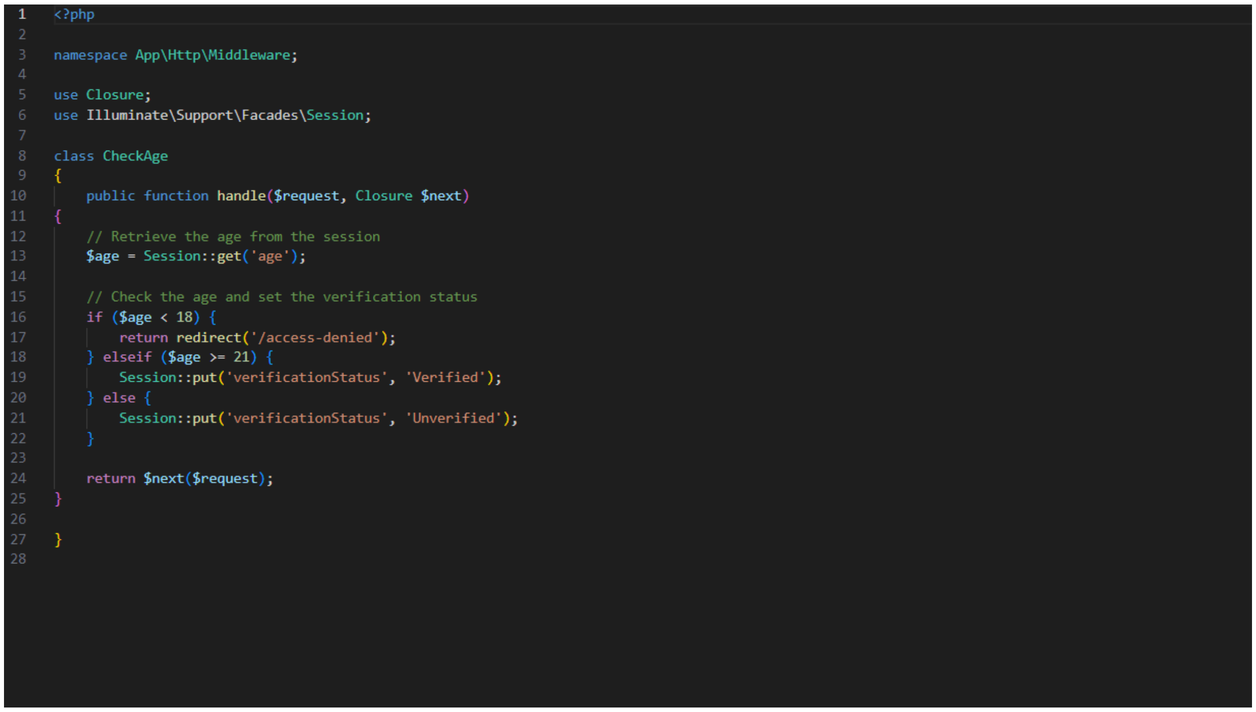
Andreev Albert D. Ataiza Website Development BSIT-3C

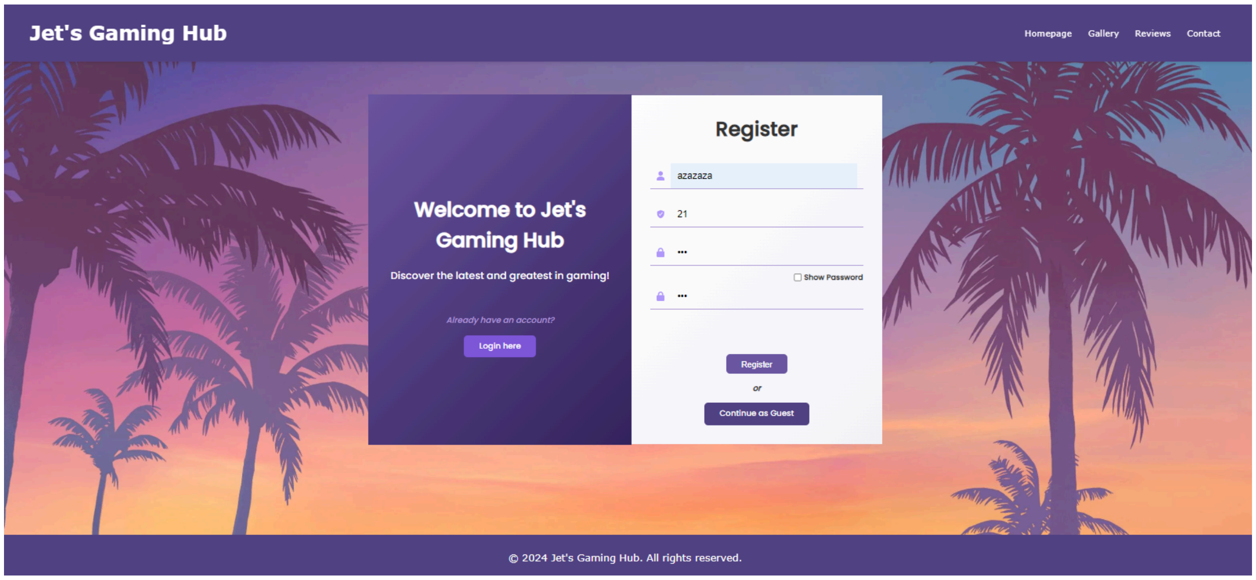
1. CheckAge.php - Middleware

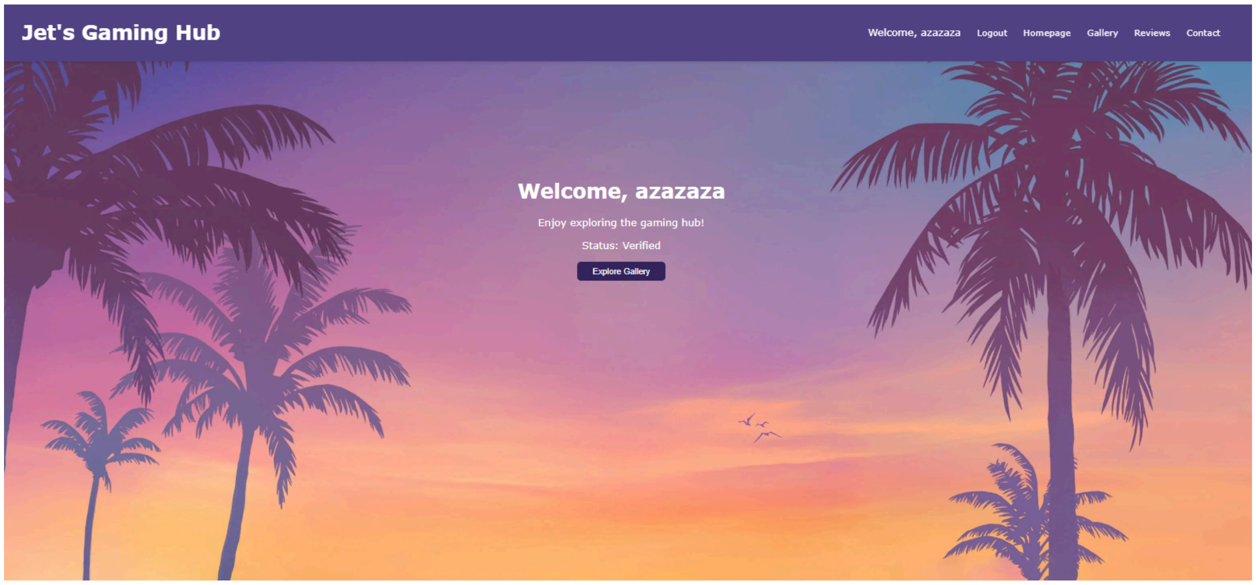


```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Session;

class CheckAge
{
    public function handle($request, Closure $next)
{
    // Retrieve the age from the session
    $age = Session::get('age');

    // Check the age and set the verification status
    if ($age < 18) {
        return redirect('/access-denied');
    } elseif ($age >= 21) {
        Session::put('verificationStatus', 'Verified');
    } else {
        Session::put('verificationStatus', 'Unverified');
    }

    return $next($request);
}

}
```

This is the checkage middleware, here you can see that the user must be 18+ to register in the website, but if they are less than 21 of age but greater than 18 they will become an unverified user, and verified user if they meet the age requirements of being 21.

Example:

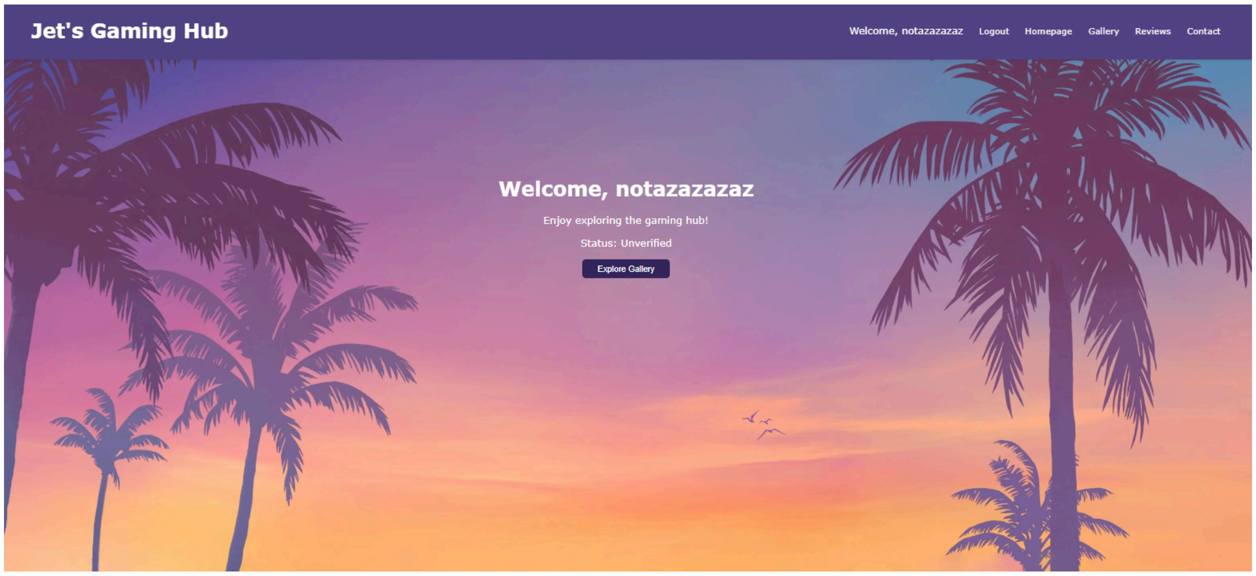




Welcome, azazaza

Enjoy exploring the gaming hub!

Status: Verified

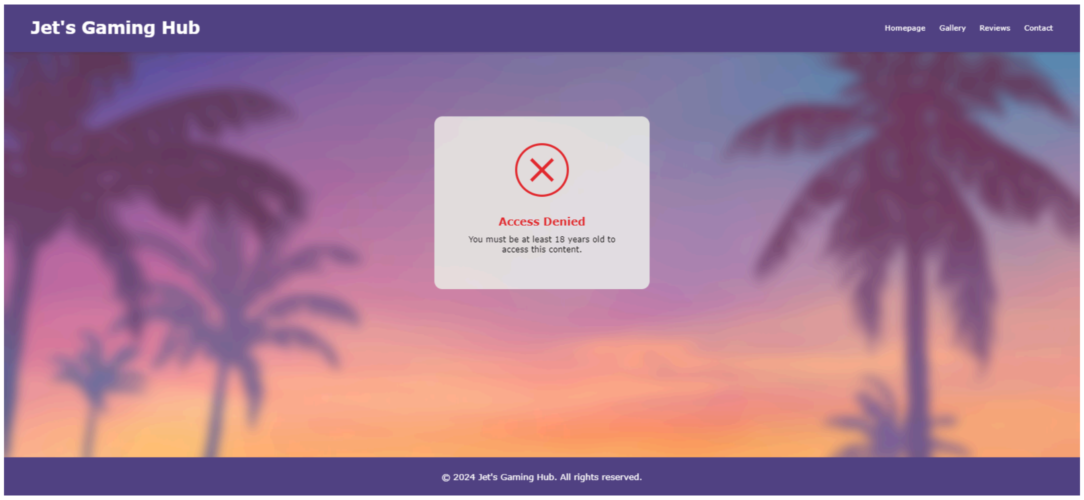


Welcome, notazazazaz

Enjoy exploring the gaming hub!

Status: Unverified

Or if they are under 18

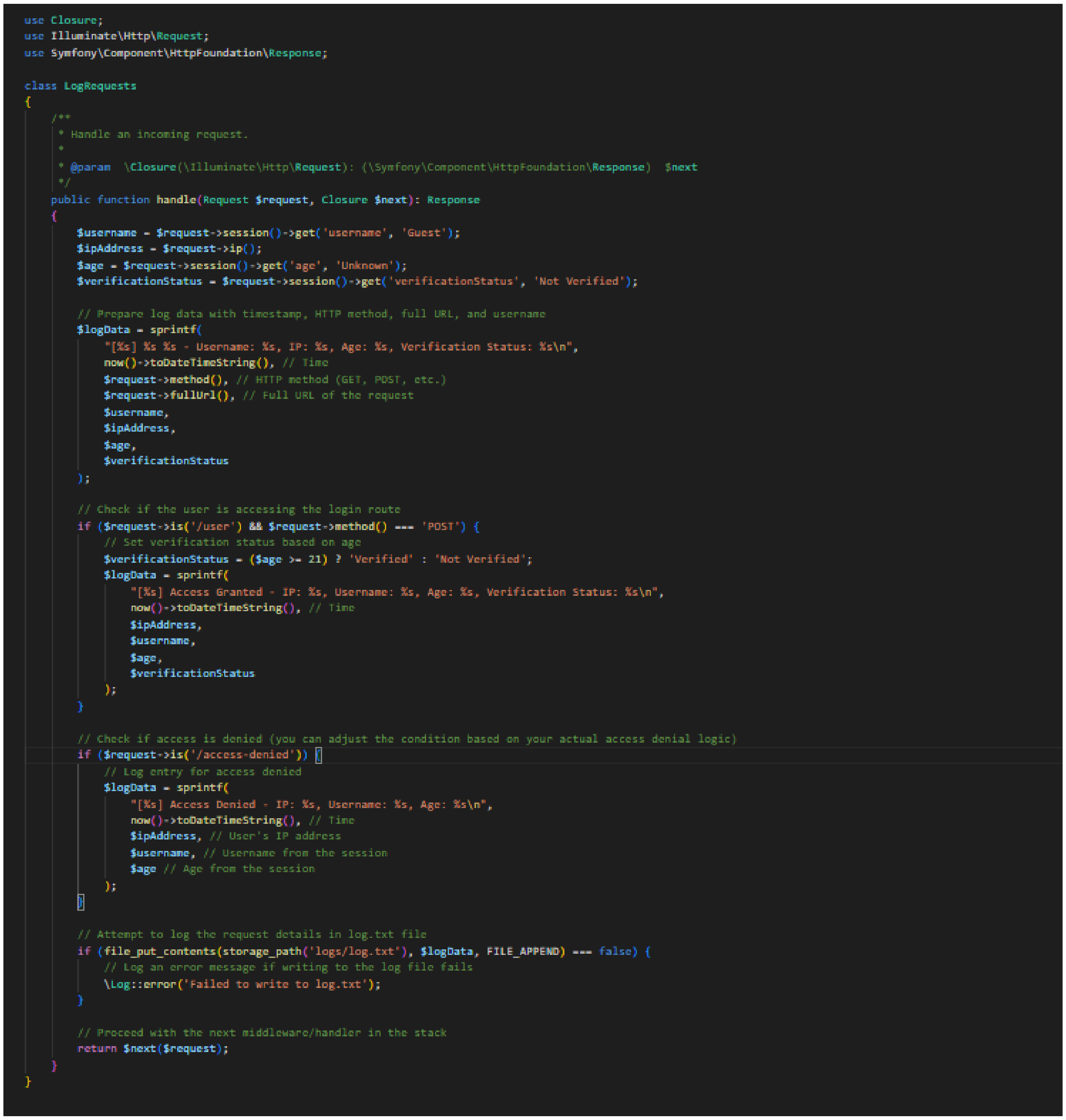


Access Denied

You must be at least 18 years old to access this content.

© 2024 Jet's Gaming Hub. All rights reserved.

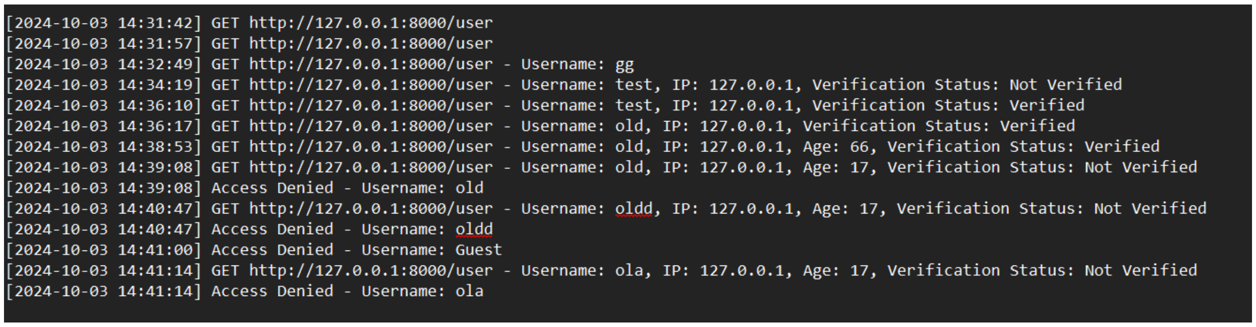
2. LogRequests.php



```php
use Closure;
use Illuminate\Http\Request;
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
    {
        $username = $request->session()->get('username', 'Guest');
        $ipAddress = $request->ip();
        $age = $request->session()->get('age', 'Unknown');
        $verificationStatus = $request->session()->get('verificationStatus', 'Not Verified');

        // Prepare log data with timestamp, HTTP method, full URL, and username
        $logData = sprintf(
            "[%s] %s %s - Username: %s, IP: %s, Age: %s, Verification Status: %s\n",
            now()->toDateTimeString(), // Time
            $request->method(), // HTTP method (GET, POST, etc.)
            $request->fullUrl(), // Full URL of the request
            $username,
            $ipAddress,
            $age,
            $verificationStatus
        );

        // Check if the user is accessing the login route
        if ($request->is('/user') && $request->method() === 'POST') {
            // Set verification status based on age
            $verificationStatus = ($age >= 21) ? 'Verified' : 'Not Verified';
            $logData = sprintf(
                "[%s] Access Granted - IP: %s, Username: %s, Age: %s, Verification Status: %s\n",
                now()->toDateTimeString(), // Time
                $ipAddress,
                $username,
                $age,
                $verificationStatus
            );
        }

        // Check if access is denied (you can adjust the condition based on your actual access denial logic)
        if ($request->is('/access-denied')) {
            // Log entry for access denied
            $logData = sprintf(
                "[%s] Access Denied - IP: %s, Username: %s, Age: %s\n",
                now()->toDateTimeString(), // Time
                $ipAddress, // User's IP address
                $username, // Username from the session
                $age // Age from the session
            );
        }

        // Attempt to log the request details in log.txt file
        if (file_put_contents(storage_path('logs/log.txt'), $logData, FILE_APPEND) === false) {
            // Log an error message if writing to the log file fails
            \Log::error('Failed to write to log.txt');
        }

        // Proceed with the next middleware/handler in the stack
        return $next($request);
    }
}
```

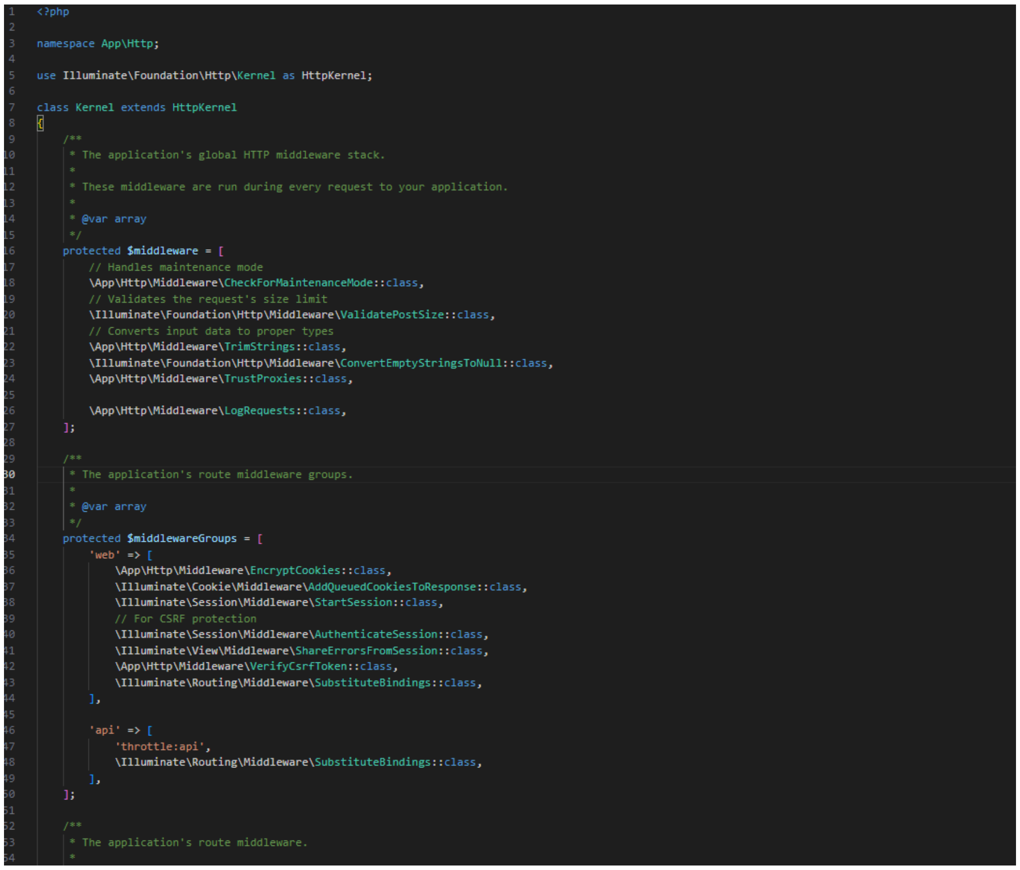
Here we can see that this middleware handles the appends or logs for the login requests, whether the user has been denied or granted access. This also logs the users with their usernames, age, and IP Addresses in the logs.txt file

Sample Log.txt file output

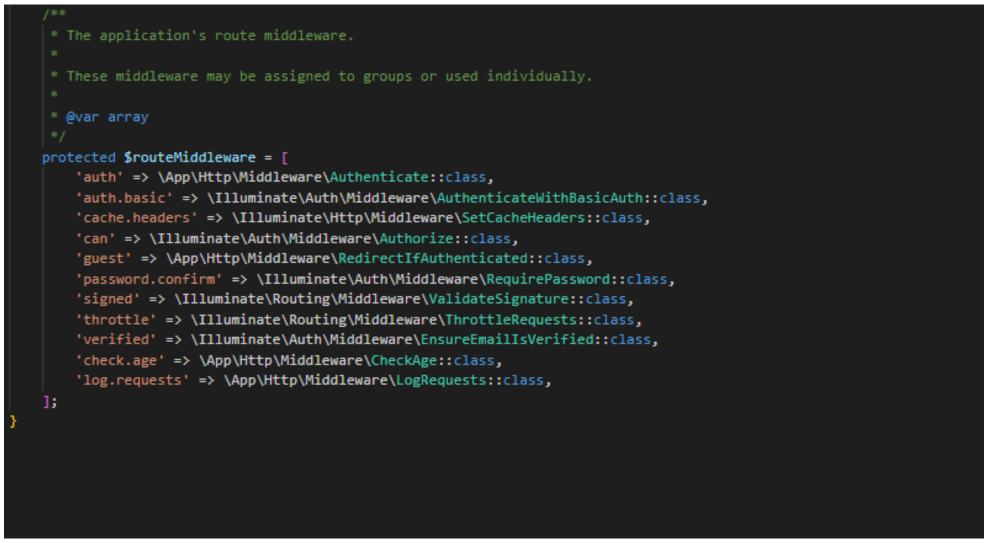


```
[2024-10-03 14:31:42] GET http://127.0.0.1:8000/user
[2024-10-03 14:31:57] GET http://127.0.0.1:8000/user
[2024-10-03 14:32:49] GET http://127.0.0.1:8000/user - Username: gg
[2024-10-03 14:34:19] GET http://127.0.0.1:8000/user - Username: test, IP: 127.0.0.1, Verification Status: Not Verified
[2024-10-03 14:36:10] GET http://127.0.0.1:8000/user - Username: test, IP: 127.0.0.1, Verification Status: Verified
[2024-10-03 14:36:17] GET http://127.0.0.1:8000/user - Username: old, IP: 127.0.0.1, Verification Status: Verified
[2024-10-03 14:38:53] GET http://127.0.0.1:8000/user - Username: old, IP: 127.0.0.1, Age: 66, Verification Status: Verified
[2024-10-03 14:39:08] GET http://127.0.0.1:8000/user - Username: old, IP: 127.0.0.1, Age: 17, Verification Status: Not Verified
[2024-10-03 14:39:08] Access Denied - Username: old
[2024-10-03 14:40:47] GET http://127.0.0.1:8000/user - Username: oldd, IP: 127.0.0.1, Age: 17, Verification Status: Not Verified
[2024-10-03 14:40:47] Access Denied - Username: oldd
[2024-10-03 14:41:00] Access Denied - Username: Guest
[2024-10-03 14:41:14] GET http://127.0.0.1:8000/user - Username: ola, IP: 127.0.0.1, Age: 17, Verification Status: Not Verified
[2024-10-03 14:41:14] Access Denied - Username: ola
```

3. Kernel.php



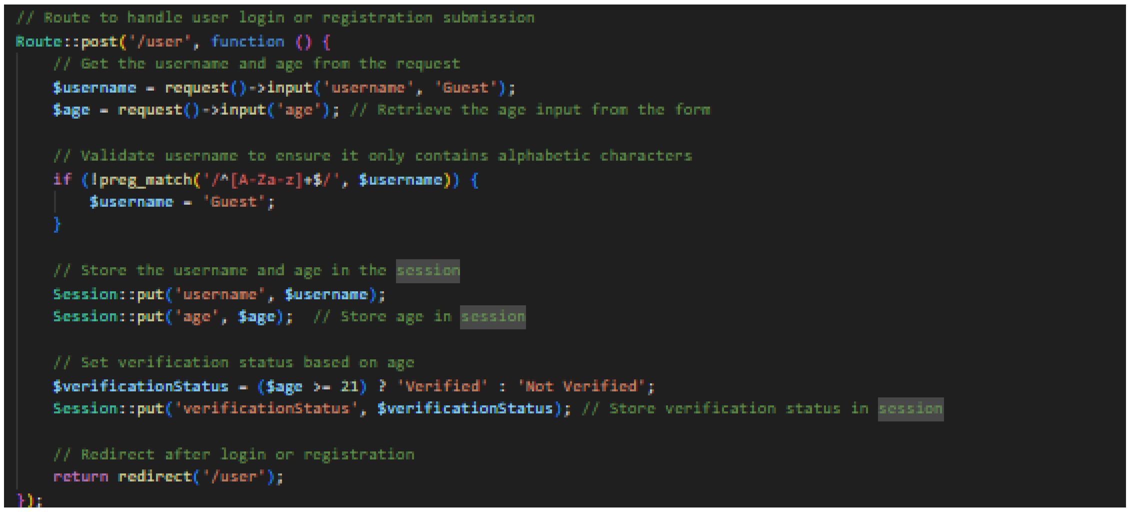
```php
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array
     */
    protected $middleware = [
        // Handles maintenance mode
        \App\Http\Middleware\CheckForMaintenanceMode::class,
        // Validates the request's size limit
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        // Converts input data to proper types
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
        \App\Http\Middleware\TrustProxies::class,

        \App\Http\Middleware\LogRequests::class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            // For CSRF protection
            \Illuminate\Session\Middleware\AuthenticateSession::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.
     *
```



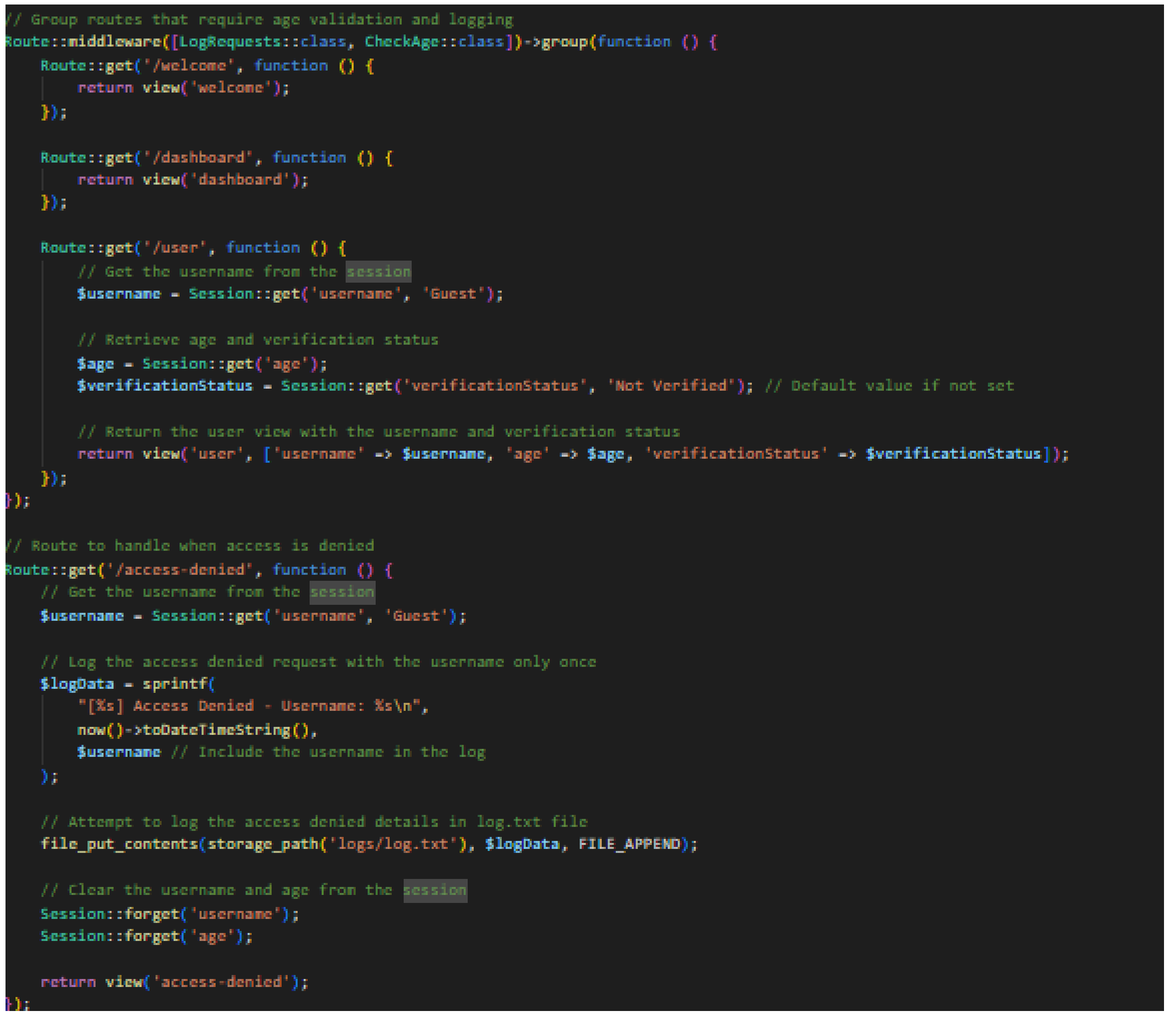
```
    /**
     * The application's route middleware.
     *
     * These middleware may be assigned to groups or used individually.
     *
     * @var array
     */
    protected $routeMiddleware = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        '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 the kernel file, here we can see it controls and coordinates the middleware to make it act as a filter in our website, we can also see the custom middleware here like CheckAge and LogRequests.

3. Routes:



```
// Route to handle user login or registration submission
Route::post('/user', function () {
    // Get the username and age from the request
    $username = request()->input('username', 'Guest');
    $age = request()->input('age'); // Retrieve the age input from the form

    // Validate username to ensure it only contains alphabetic characters
    if (!preg_match('/^[A-Za-z]+$/', $username)) {
        $username = 'Guest';
    }

    // Store the username and age in the session
    Session::put('username', $username);
    Session::put('age', $age);  // Store age in session

    // Set verification status based on age
    $verificationStatus = ($age >= 21) ? 'Verified' : 'Not Verified';
    Session::put('verificationStatus', $verificationStatus); // Store verification status in session

    // Redirect after login or registration
    return redirect('/user');
});
```



```php
// Group routes that require age validation and logging
Route::middleware([LogRequests::class, CheckAge::class])->group(function () {
    Route::get('/welcome', function () {
        return view('welcome');
    });

    Route::get('/dashboard', function () {
        return view('dashboard');
    });

    Route::get('/user', function () {
        // Get the username from the session
        $username = Session::get('username', 'Guest');

        // Retrieve age and verification status
        $age = Session::get('age');
        $verificationStatus = Session::get('verificationStatus', 'Not Verified'); // Default value if not set

        // Return the user view with the username and verification status
        return view('user', ['username' => $username, 'age' => $age, 'verificationStatus' => $verificationStatus]);
    });
});

// Route to handle when access is denied
Route::get('/access-denied', function () {
    // Get the username from the session
    $username = Session::get('username', 'Guest');

    // Log the access denied request with the username only once
    $logData = sprintf(
        "[%s] Access Denied - Username: %s\n",
        now()->toDateTimeString(),
        $username // Include the username in the log
    );

    // Attempt to log the access denied details in log.txt file
    file_put_contents(storage_path('logs/log.txt'), $logData, FILE_APPEND);

    // Clear the username and age from the session
    Session::forget('username');
    Session::forget('age');

    return view('access-denied');
});
```

In the routes, we will save the age from the form and then add a logic of a verification status. Where they must be 21 to display a “verified” status, else “unverified” status once the homepage loads in. (Example in page 2)