Jethro J. Manzanillo
BSIT - 3C
Web Development 2

Routes:

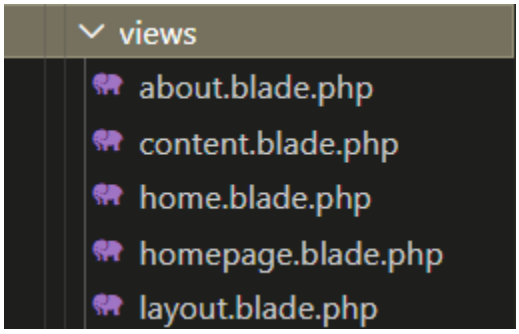
I defined routes for each lab activity in my Laravel application, incorporating a parameterized route for user interaction. Below is an overview of how I implemented the required features:

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

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

// Group routes logreq
Route::middleware([LogRequests::class])->group(function (): void {
    Route::get('/home', function (): Factory|View {
        return view(view: 'home');
    })->name('home');

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

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

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

});

// Form submission and redirect to the homepage with username
Route::post('/homepage', function (Request $request): mixed|RedirectResponse {
    $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]);
});
```

- **Key Features**
  - **Middleware:** I implemented a custom middleware called `LogRequests` that logs incoming requests, ensuring that I can monitor user interactions with my site

effectively. This middleware is applied to a group of routes to maintain a clean and organized structure.
    - **Parameterized Routes:** The homepage, about page, and content page include a parameterized route that accepts an optional `username`. This feature enhances the user experience by personalizing greetings based on the user's input or defaulting to "Guest."
    - **Form Submission:** I set up a form on the homepage to allow users to specify their login type (guest or user). Upon submission, the form processes the input, validates the username if provided, and redirects to the personalized homepage.
- **View Files**
  Each route corresponds to a specific view file within the `resources/views` directory. I created the following views:
    - `home.blade.php`: The home page layout.
    - `homepage.blade.php`: Displays personalized greetings.
    - `about.blade.php`: Contains information about the portfolio.
    - `content.blade.php`: Displays various lab activities.

Views:



Master Layout (`layout.blade.php`)

I used the master layout file to maintain consistency across all views. This layout includes the basic structure such as the header, footer, and shared styles or scripts, allowing the views to focus on their specific content.

In each view file, I extended the master layout using:

```
1   @extends('layout')
2
3   @section('content')
4
5   <section id="home" class="image_section">
6
7       <div class="content">
8           <h1 class="description">
9               Welcome, {{ $username }}!
```

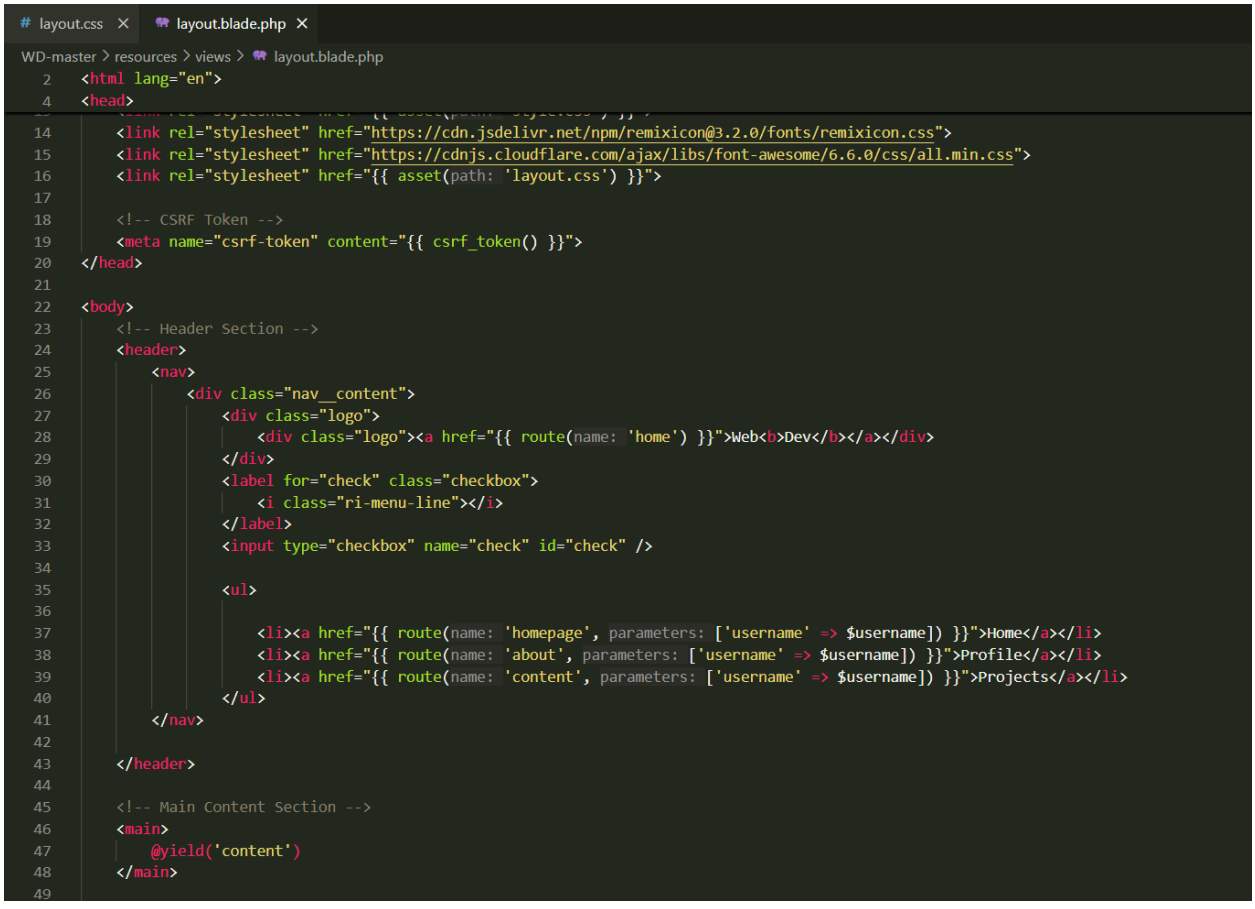
Other Views

- **`about.blade.php`**: This view provides details about the lab or project, explaining my approach and key information.
- **`content.blade.php`**: This view showcases specific content related to a lab or project, with a summary of what I achieved.

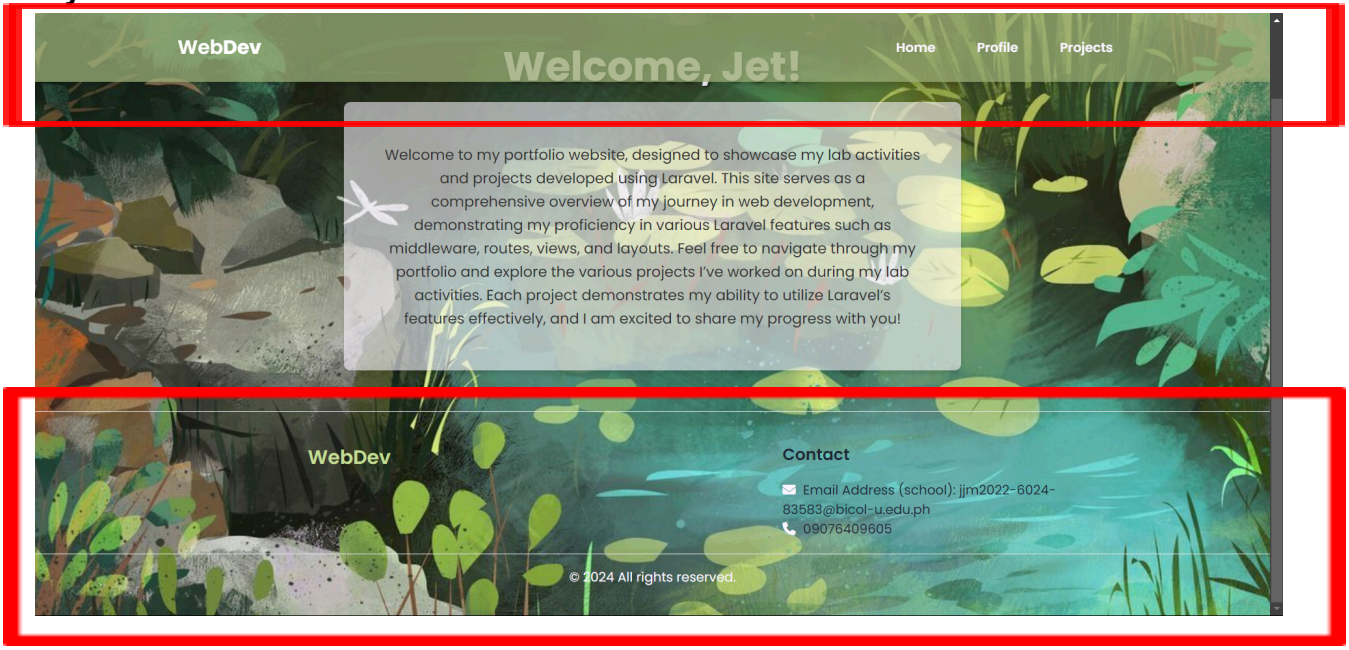
By using the `@extends('layout')` method, I ensure that all my views maintain a consistent style, focusing on the content and lab summaries.

Layouts:

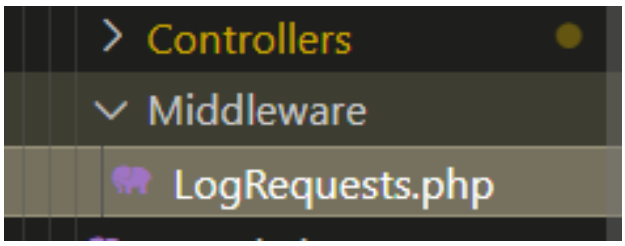
I implemented a master layout to maintain a uniform design across all the pages of my portfolio website. This layout includes shared components like the header, footer, and navigation, ensuring consistency throughout the site**.**



Layout Interface:



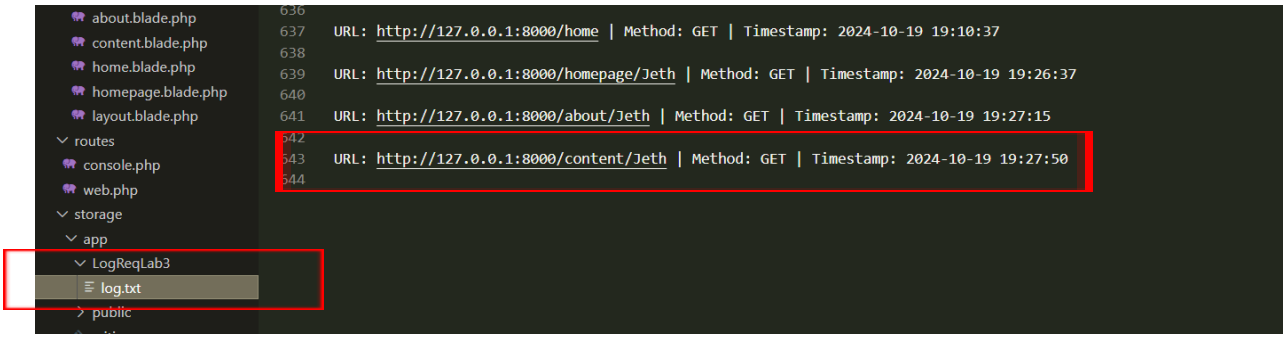
Middleware:



I reused this middleware from a previous project to handle request logging. It logs each incoming request, recording details such as the URL, HTTP method, and a timestamp.

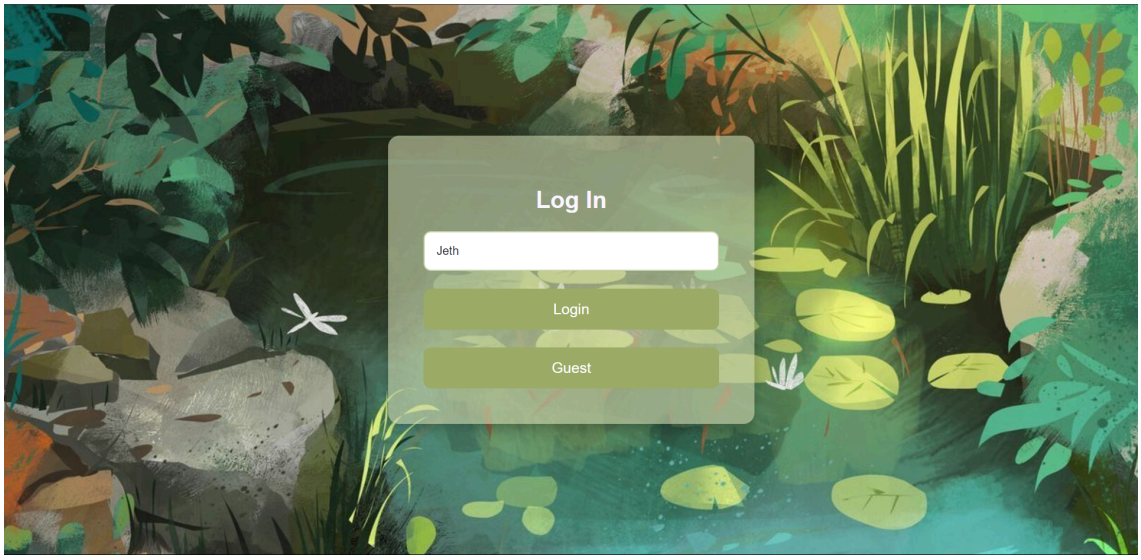
```php
<?php

namespace App\Http\Middleware;

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

4 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('Y-m-d H:i:s') . PHP_EOL;
        Storage::append('LogReqLab3/log.txt', $logData);
        return $next($request);
    }
}
```

- **Logging Information**: The middleware logs the request URL, HTTP method, and the current timestamp using Laravel's `now()` helper function. This data is then appended to a log file stored in the `LogReqLab3` directory.

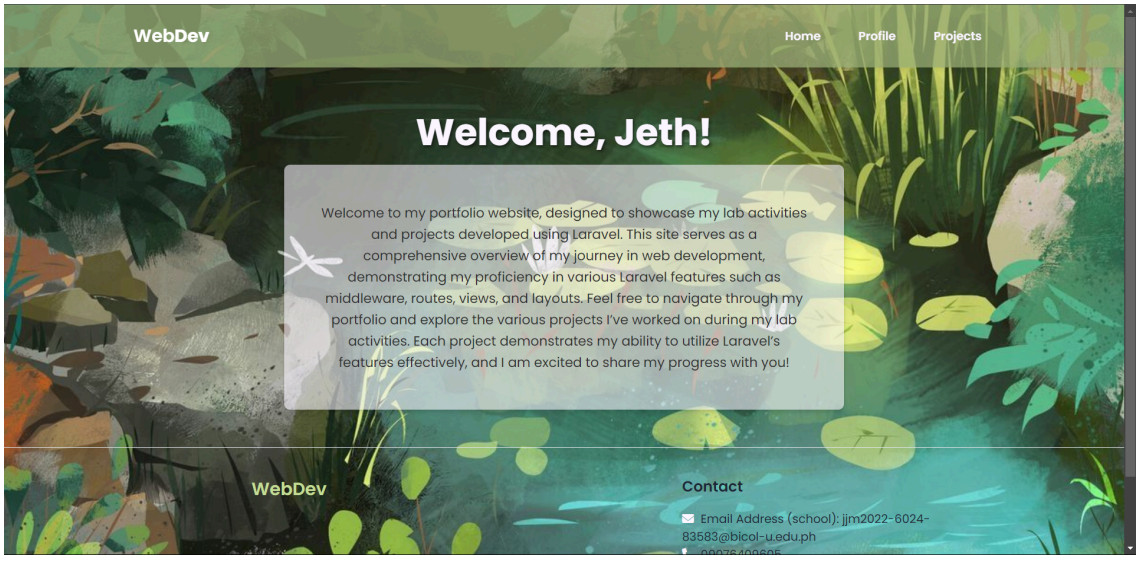


I reused this middleware from a previous project where I had implemented similar request logging functionality. It was easy to adapt it to fit this project's needs, and it's proven to be efficient for tracking request activity.

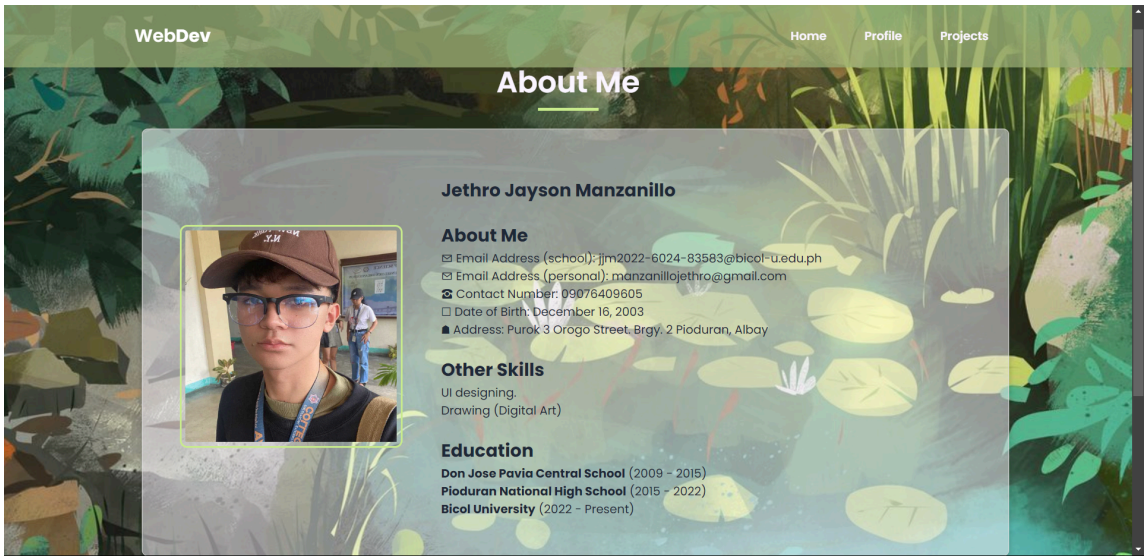
Interfaces:
Log In:



Home:



Profile:



Projects: