

**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:

```
3  use App\Http\Middleware\LogRequests;
4  use Illuminate\Support\Facades\Route;
5  use Illuminate\Http\Request;
6
7  // Route for displaying the Age verification form
8  Route::get('/', function (): Factory|View {
9      return view('Age');
10 })->name('Age');
11
12 // Group routes logreq
13 Route::middleware([LogRequests::class])->group(function (): void {
14     Route::get('/home', function (): Factory|View {
15         return view('home');
16     })->name('home');
17
18     Route::get('/homepage/{username?}', function ($username = 'Guest'): Factory|View {
19         return view('homepage', data: ['username' => $username]);
20     })->where('username', '[a-zA-Z]+')->name('homepage');
21
22     Route::get('/about/{username?}', function ($username = 'Guest'): Factory|View {
23         return view('about', data: ['username' => $username]);
24     })->where('username', '[a-zA-Z]+')->name('about');
25
26     Route::get('/content/{username?}', function ($username = 'Guest'): Factory|View {
27         return view('content', data: ['username' => $username]);
28     })->where('username', '[a-zA-Z]+')->name('content');
29 });
30
31 // Form submission and redirect to the homepage with username
32 Route::post('/homepage', function (Request $request): mixed|RedirectResponse {
33     $loginType = $request->input('login_type');
34     $username = $loginType === 'guest' ? 'Guest' : $request->input('username');
35     if ($loginType === 'user') {
36         $request->validate(['username' => 'required|alpha']);
37     }
38     return redirect()->route('homepage', ['username' => $username]);
39 });
40 });
```

● **Key Features**

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

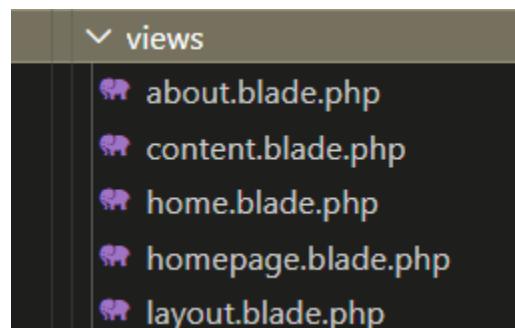
- effectively. This middleware is applied to a group of routes to maintain a clean and organized structure.
- **Parameterized Routes:** The homepage, about page, and content page include a parameterized route that accepts an optional `username`. This feature enhances the user experience by personalizing 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 }}!

```

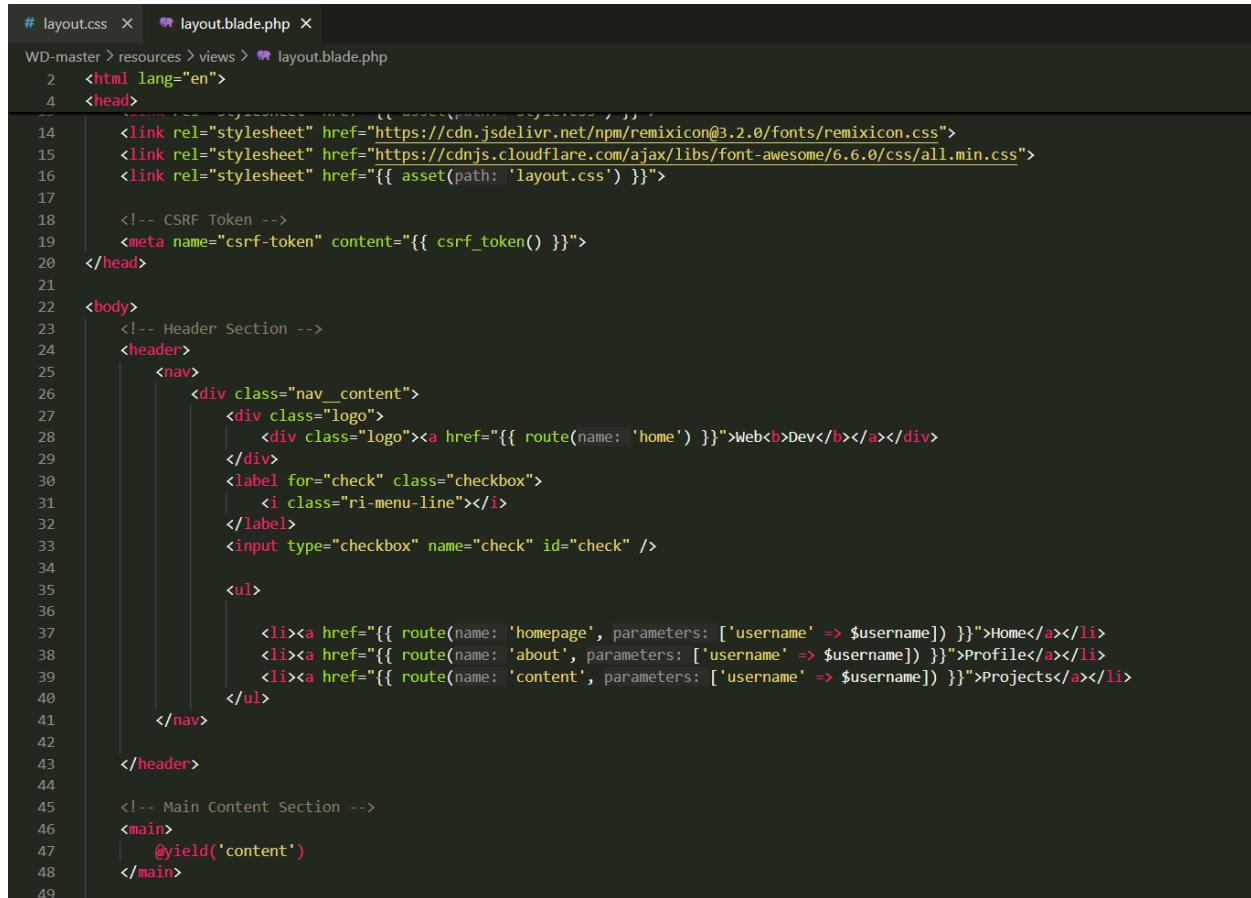
## Other Views

- **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.

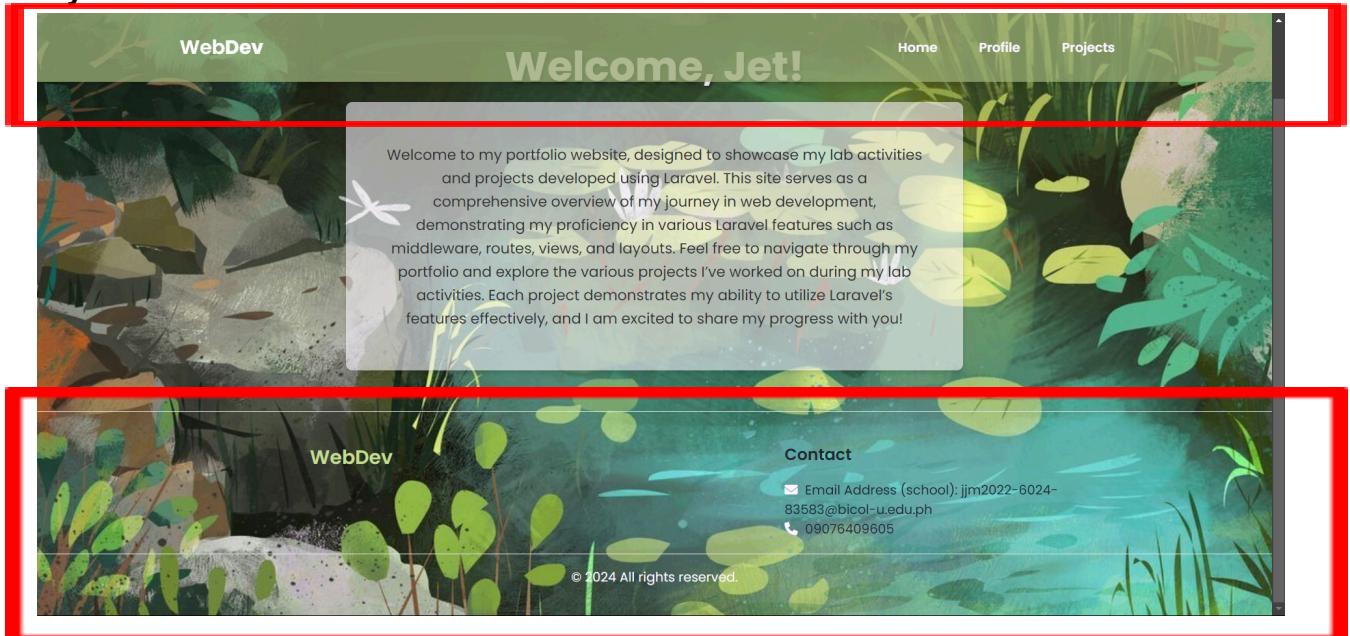
## Layouts:

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.

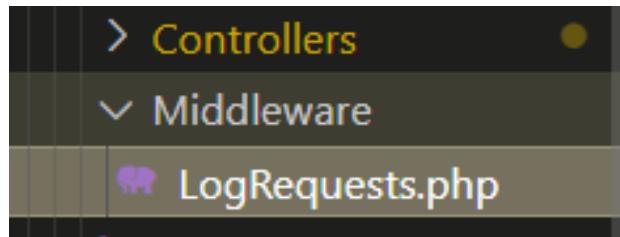


```
# layout.css X  layout.blade.php X
WD-master > resources > views > layout.blade.php
  2   <html lang="en">
  4     <head>
  5       <!-- CSS References --> {{ asset('css/app.css') }}</head>
  6
  7       <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/remixicon@3.2.0/fonts/remixicon.css">
  8       <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css">
  9       <link rel="stylesheet" href="{{ asset(path: 'layout.css') }}>
  10
  11     <!-- CSRF Token -->
  12     <meta name="csrf-token" content="{{ csrf_token() }}>
  13   </head>
  14
  15   <body>
  16     <!-- Header Section -->
  17     <header>
  18       <nav>
  19         <div class="nav__content">
  20           <div class="logo">
  21             <div class="logo"><a href="{{ route(name: 'home') }}>WebDev</a></div>
  22           </div>
  23           <label for="check" class="checkbox">
  24             <i class="ri-menu-line"></i>
  25           </label>
  26           <input type="checkbox" name="check" id="check" />
  27
  28         <ul>
  29           <li><a href="{{ route(name: 'homepage', parameters: ['username' => $username]) }}>Home</a></li>
  30           <li><a href="{{ route(name: 'about', parameters: ['username' => $username]) }}>Profile</a></li>
  31           <li><a href="{{ route(name: 'content', parameters: ['username' => $username]) }}>Projects</a></li>
  32         </ul>
  33       </div>
  34     </nav>
  35
  36   </header>
  37
  38   <!-- Main Content Section -->
  39   <main>
  40     @yield('content')
  41   </main>
  42
  43
  44
  45
  46
  47
  48
  49
```

## 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.

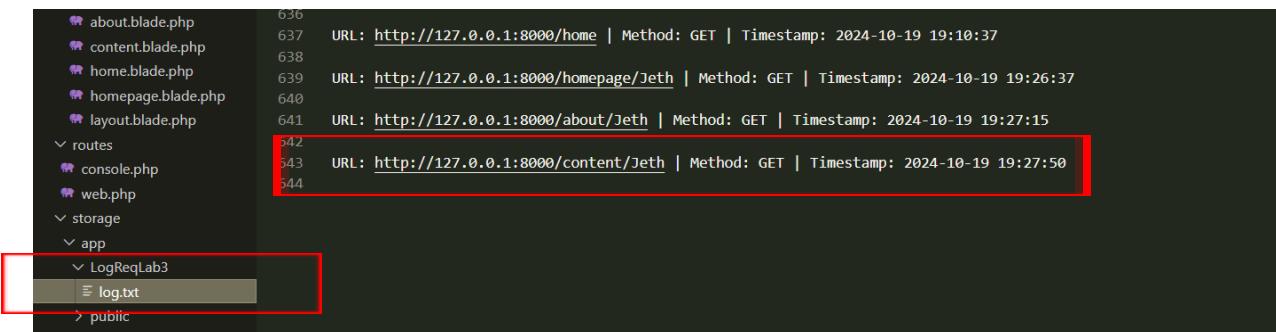
```
<?php

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.



```

about.blade.php      636
content.blade.php   637  URL: http://127.0.0.1:8000/home | Method: GET | Timestamp: 2024-10-19 19:10:37
home.blade.php       638
homepage.blade.php  639  URL: http://127.0.0.1:8000/homepage/Jeth | Method: GET | Timestamp: 2024-10-19 19:26:37
layout.blade.php     640
routes              641  URL: http://127.0.0.1:8000/about/Jeth | Method: GET | Timestamp: 2024-10-19 19:27:15
routes              642
console.php          643  URL: http://127.0.0.1:8000/content/Jeth | Method: GET | Timestamp: 2024-10-19 19:27:50
web.php              644

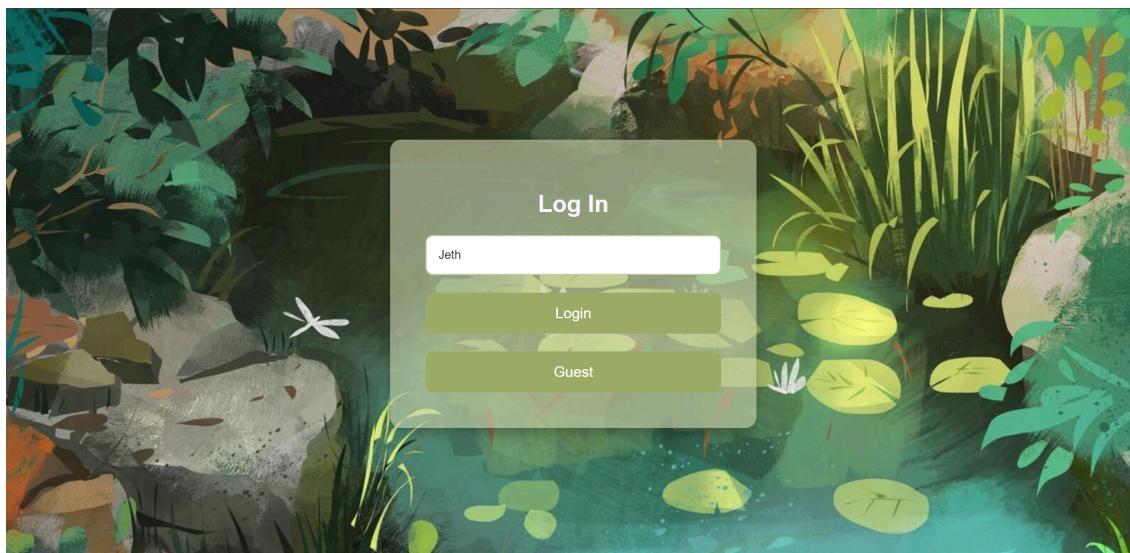
log.txt
public

```

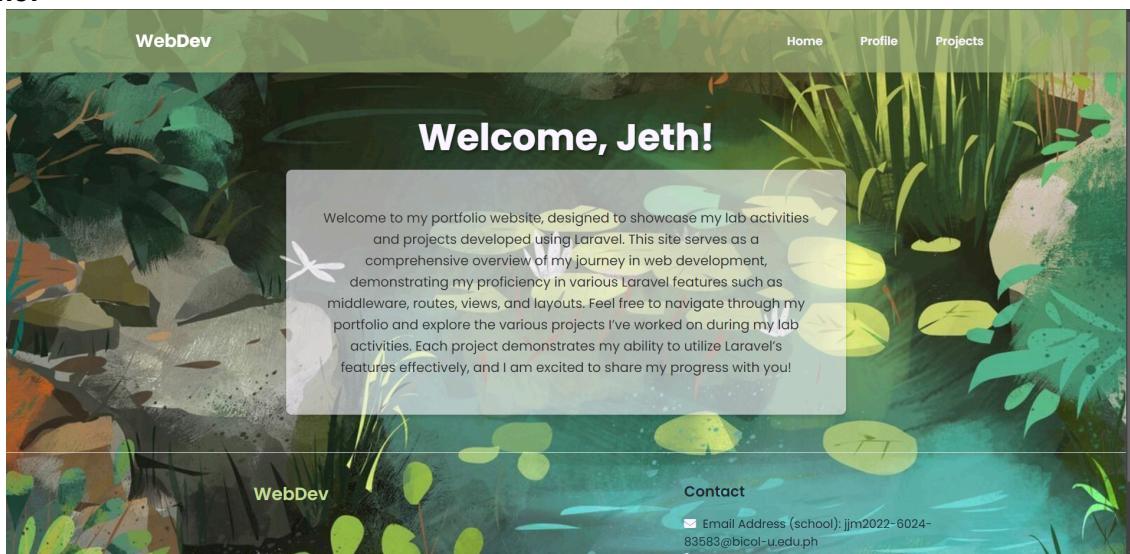
I reused this middleware from a previous project where I had implemented similar request logging functionality. It was easy to adapt it to fit this project's needs, and it's proven to be efficient for tracking request activity.

## Interfaces:

### Log In:



### Home:



## Profile:

**Jethro Jayson Manzanillo**

**About Me**

- Email Address (school): jjm2022-6024-83583@bicol-u.edu.ph
- Email Address (personal): manzanillojethro@gmail.com
- Contact Number: 09076409605
- Date of Birth: December 16, 2003
- Address: Purok 3 Orog Street, Brgy. 2 Piordan, Albay

**Other Skills**

- UI designing.
- Drawing (Digital Art)

**Education**

- Don Jose Pavia Central School (2009 – 2018)
- Piordan National High School (2015 – 2022)
- Bicol University (2022 – Present)

## Projects:

**Setting Up Laravel**

For this project, my goal was to dive into Laravel and get hands-on experience with its setup and core features. First, I installed Laravel and all necessary dependencies, then I created a new project and set up the .env file to connect the database. After that, I built three basic views: the homepage, about, and content pages, and linked them using routes. While doing this, I explored and documented the purpose of Laravel's key directories and files. To wrap it up, I took screenshots of each step and compiled them into a PDF for easy reference. This project helped me better understand how Laravel works and how to navigate its structure.

**Routes**

In this part of the project, we focused on defining basic routes in Laravel and exploring route parameters. Initially, we created a simple

**Routes**

In this part of the project, we focused on defining basic routes in Laravel and exploring route parameters. Initially, we created a simple route that returns a view for the homepage, displaying a welcoming message to users. Following that, we expanded our routes to include an 'About Us' page, a redirection from '/home' to the homepage, and a 'Contact Us' form. Next, we delved into route parameters by defining a route that accepts a required username parameter. This allowed us to create dynamic greeting messages, such as 'Welcome, johndoe!' based on the URL. We also modified this route to make the username optional, displaying a default message like 'Welcome, Guest!' when no username is provided. To ensure data validation, we implemented regular expression constraints to allow only alphabetic characters in the username. This exercise not only reinforced our understanding of routing in Laravel but also enhanced our skills in creating dynamic web applications.

**Layouts**

In this section of our project, we focused on creating a layout file and corresponding views in Laravel. We organized our project by creating