



**BICOL UNIVERSITY COLLEGE OF SCIENCE**  
**LEGAZPI CITY, ALBAY**



**IT Elect 1 – WEB DEVELOPMENT**

# **DOCUMENTATION**

**(GROUP 5, LABORATORY 3)**

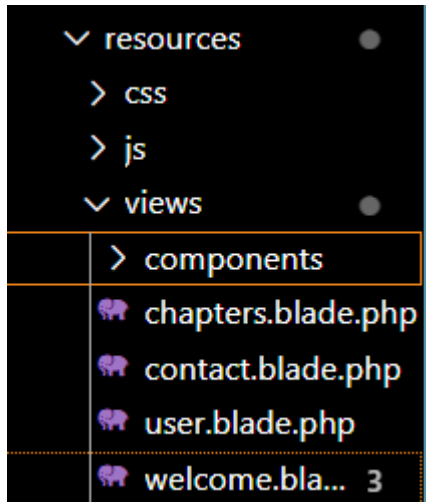
**SUBMITTED BY:**

**JERALD JAY G. BUBAN**

**BSIT – 3C**

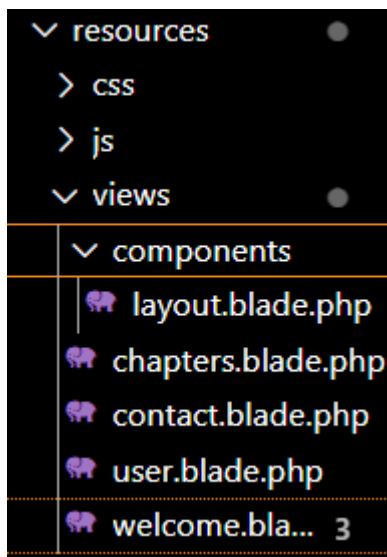
## Part 1: Creating a Layout File

In the resources/views directory, create a new folder named "Components".



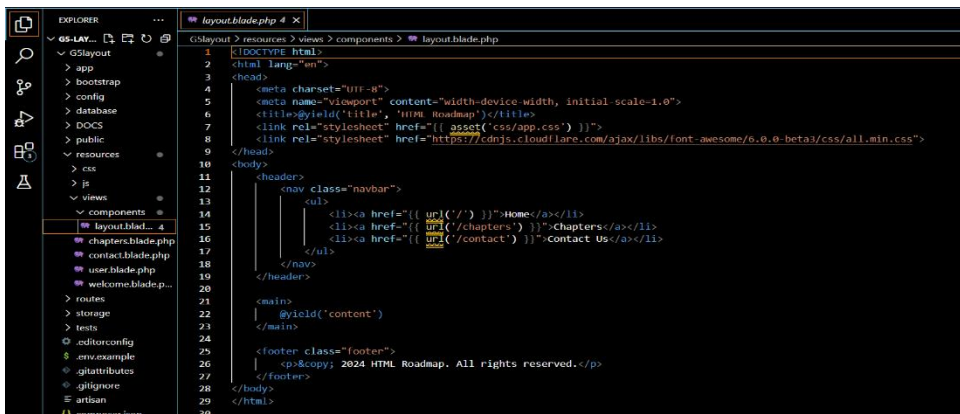
Creating a new folder named "Components" in the resources/views directory is important, because the purpose of this is that Laravel treats files in this folder as layout files (templates). This helps to keep things organized and makes it easier to reuse parts of the UI, like headers or footers. By separating these parts into their own files, the code stays clean and easier to manage.

Inside the "Components" folder, create a file named Layout.blade.php.



Creating a new folder named "Components" in the resources/views directory is important, because the purpose of this is that Laravel treats files in this folder as layout files (templates). This helps to keep things organized and makes it easier to reuse parts of the UI, like headers or footers. By separating these parts into their own files, the code stays clean and easier to manage.

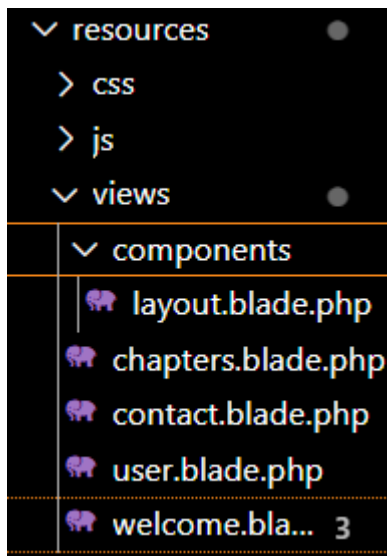
Define the basic HTML structure in Layout.blade.php.



- This is the structure of our Layout.blade.php. We put the navigation links in the header section, then a main section where each page's content will show using “@yield('content')”, and a footer. We also added placeholders for the page title and linked some CSS for styling, and used Laravel's “asset()” and “url()” functions to make sure everything is loaded properly.

## Part 2: Creating Views

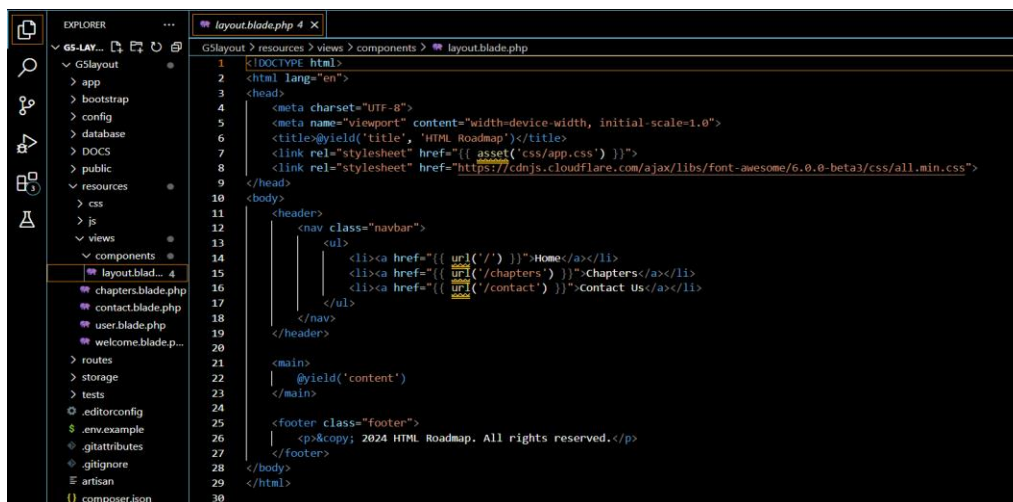
In the resources/views directory, create 3 new blade files.



In these blade files we put our codes for each page like the welcome page, chapters page, and contact page. Through this, we can put the information's we wanted and edit the content freely.

Each view should extend the layout file and include page-specific content.

The layout file:



- It defines a layout with a “header”, “main”, and “footer”, where “@yield('title')” and “@yield('content')” act as placeholders for content that will be provided by specific pages. The template also links to external CSS and Font Awesome for styling, and uses Laravel functions like “asset()” to load local assets and “url()” to generate URLs for navigation.

- The `<meta name="viewport" content="width=device-width, initial-scale=1.0">` ensures the website is responsive and adapts to different screen sizes, especially for mobile devices.
- The `@yield('title', 'HTML Roadmap')` directive lets each view define its own title. If no title is given, it automatically uses "HTML Roadmap" as the default title.
- The `<link rel="stylesheet" href="{{ asset('css/app.css') }}">` loads the local CSS file for styling using Laravel's `asset()` function to point to the correct path.
- The `<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-awesome.min.css">` loads the Font Awesome library for icons from a CDN.
- The `<nav class="navbar">` sets up the navigation bar using the `navbar` class for styling.
- The `<ul>` and `<li>` creates a list of navigation links.
- The `href="{{ url('/') }}"` generates the URL for the home page using Laravel's `url()` helper. It also generates the other links, like the "Chapters" and "Contact Us" pages.
- The `@yield('content')` acts as a placeholder for content that will be provided by the child views. Child views that extend this layout file will inject their specific content into this section.

## Blade files code and page-specific contents:

### Welcome.blade.php code

```

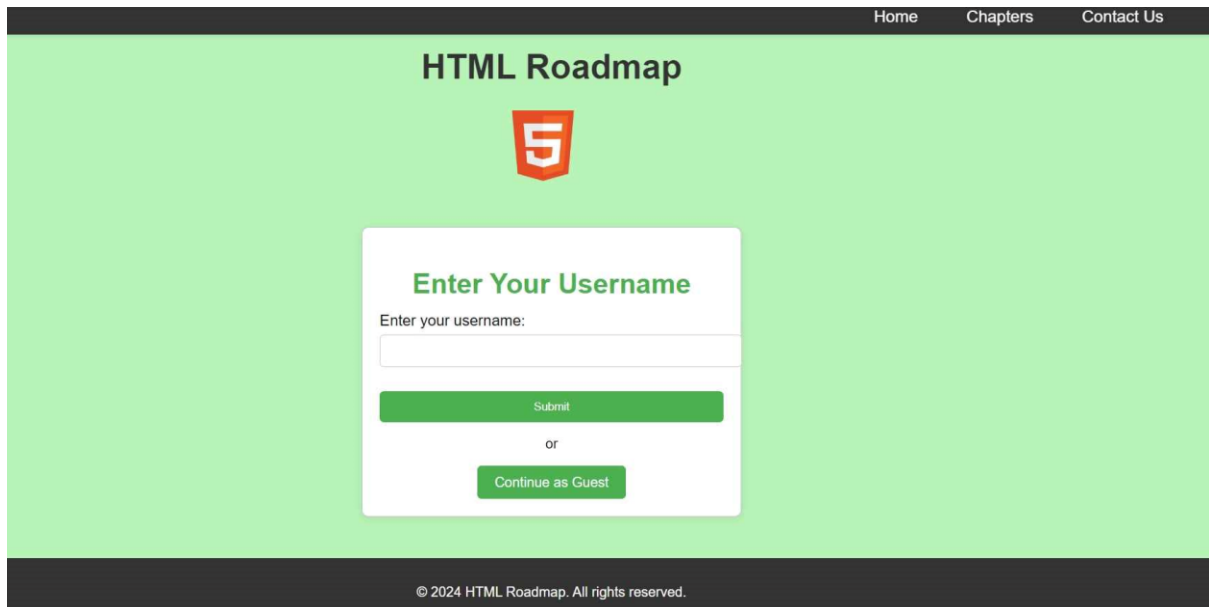
welcome.blade.php 3 x
GSLayout > resources > views > welcome.blade.php
1  @extends('components.Layout')
2
3  @section('title', 'HTML Roadmap - Welcome')
4
5  @section('content')
6  <header>
7      <h1>HTML Roadmap</h1>
8      
9  </header>
10
11 <div class="container">
12     <div class="form-box">
13         <h2>Enter Your Username</h2>
14         <form action="{{ url('/user') }}" method="GET">
15             <label for="username">Enter your username:</label>
16             <input type="text" name="username" id="username" pattern="[A-Za-z]+" title="Only alphabetic characters are allowed" required />
17             <br><br>
18             <button type="submit">Submit</button>
19         </form>
20
21
22         <div class="link-container">
23             <p>or</p>
24             <a href="{{ url('/user') }}" class="button-link">Continue as Guest</a>
25         </div>
26     </div>
27 </div>
28
29
30 @endsection
31

```

- The `@extends('components.Layout')` tells Laravel to extend the layout defined in `components.Layout`, which means that the content here will be placed into the layout's structure.
- The `@section('title', 'HTML Roadmap - Welcome')` sets the page title to "HTML Roadmap - Welcome", which will replace the placeholder defined in the layout file (from `@yield('title')`).
- The `@section('content')` starts the section for the main content of the page.

- The @endsection marks the end of the content section.

## Welcome page



- It prompts the user a welcome page, where they can enter their usernames to access the page or they can choose to continue as guest.

## Chapters.blade.php code

```

1  @extends('components.Layout')
2
3
4  @section('title', 'HTML Roadmap - Chapters')
5
6  @section('content')
7
8      <div class="card-container">
9          <div class="card">
10             <div class="card-header">
11                 <strong>Chapter 1. Introduction to HTML</strong>
12             </div>
13             <div class="card-body">
14                 <p>Learn about the basic concepts of HTML, its purpose, and how it integrates with web technologies.</p>
15                 <p>HTML is the backbone of web development. It allows you to create structured documents that browsers can render. This chapter cove
16             </div>
17         </div>
18
19         <div class="card">
20             <div class="card-header">
21                 <strong>Chapter 2. Basic HTML Structure</strong>
22             </div>
23             <div class="card-body">
24                 <p>Understand the structure of an HTML document, including the <code><html></code>, <code><head></code>, and <code><body></code> tags.
25                 <p>This chapter will guide you through the anatomy of an HTML page, emphasizing the importance of each section and how they interact
26             </div>
27         </div>
28
29         <div class="card">
30             <div class="card-header">
31                 <strong>Chapter 3. HTML Elements and Attributes</strong>
32             </div>
33             <div class="card-body">
34                 <p>Explore various HTML elements such as headings, paragraphs, links, images, and how to use attributes to enhance them.</p>
35                 <p>Learn about the most common elements and their attributes, including <code>href</code> for links, <code>src</code> for images, an
36             </div>
37         </div>
38     </div>
39 @endsection

```

```

chapters.blade.php 1,10
G5layout > resources > views > chapters.blade.php
8      <div class="card-container">
56         </div>
57         <div class="card">
58             <div class="card-header">
59                 <strong>Chapter 6. Semantic HTML</strong>
60             </div>
61             <div class="card-body">
62                 <p>Understand the importance of semantic elements for better accessibility and SEO, including <code><h1></code>, <code><h2></code>, <code><h3></code>, <code><h4></code>, <code><h5></code>, and <code><h6></code> tags.
63                 <p>Learn how semantic HTML enhances the meaning of your content, making it more accessible to screen readers and search engines. This chapter will cover the <code><h1></code>, <code><h2></code>, <code><h3></code>, <code><h4></code>, <code><h5></code>, and <code><h6></code> tags.
64             </div>
65         </div>
66         <div class="card">
67             <div class="card-header">
68                 <strong>Chapter 7. HTML Best Practices</strong>
69             </div>
70             <div class="card-body">
71                 <p>Learn best practices for writing clean and maintainable HTML code, including proper indentation, use of comments, and validation.
72                 <p>This chapter emphasizes the importance of writing readable code. You'll explore tools for validating HTML and ensuring your code follows best practices.
73             </div>
74         </div>
75         <div class="card">
76             <div class="card-header">
77                 <strong>Chapter 8. HTML5 New Features</strong>
78             </div>
79             <div class="card-body">
80                 <p>Explore new features introduced in HTML5, such as native video and audio support, the <code><canvas></code> element, and improved form controls.
81                 <p>This chapter introduces the latest HTML5 features that enhance user experience and interactivity. You'll learn how to implement these features in your web applications.
82             </div>
83         </div>
84     </div>
85
86     <div class="navigation">
87         <a href="{{ url('/contact') }}" class="next-navigation">Next</a>
88     </div>
89 @endsection
90

```

- The `@extends('components.Layout')` tells Laravel to extend the layout defined in `components.Layout`, which means that the content here will be placed into the layout's structure.
- The `@section('title', 'HTML Roadmap - Chapters')` sets the title for this page to "HTML Roadmap - Chapters", which will appear in the browser tab or page title area, replacing the layout's title placeholder.
- The `@section('content')` marks the start of the content section where the actual content of the page (chapters) will be inserted into the layout's `@yield('content')`.
- The `<div class="navigation">` provides navigation to move to the next page. In this case, a "Next" button directs the user to the contact page.
- The `<a href="{{ url('/contact') }}" class="next-navigation">Next</a>` is a link to the `/contact` route, styled as a "Next" button to guide users to the next part of the website.
- The `@endsection` marks the end of the content section.

## Chapters page

[Home](#)
[Chapters](#)
[Contact Us](#)

### Chapter 1. Introduction to HTML

Learn about the basic concepts of HTML, its purpose, and how it integrates with web technologies.

HTML is the backbone of web development. It allows you to create structured documents that browsers can render. This chapter covers the history of HTML and its evolution into HTML5.

### Chapter 2. Basic HTML Structure

Understand the structure of an HTML document, including the `<html>`, `<head>`, and `<body>` tags.

This chapter will guide you through the anatomy of an HTML page, emphasizing the importance of each section and how they interact with CSS and JavaScript.

### Chapter 3. HTML Elements and Attributes

Explore various HTML elements such as headings, paragraphs, links, images, and how to use attributes to enhance them.

Learn about the most common elements and their attributes, including `href` for links, `src` for images, and how attributes affect the rendering and functionality of your elements.

### Chapter 4. Forms and Input

Learn how to create forms for user input, including text fields, radio buttons, checkboxes, and submit buttons.

This chapter focuses on form creation and handling user input. You'll learn about the `<form>` element and various input types, as well as form validation and accessibility considerations.

### Chapter 5. Multimedia in HTML

Discover how to embed images, audio, and video into your web pages using the appropriate HTML elements.

This chapter will cover the `<img>`, `<audio>`, and `<video>` elements, showing you how to use them effectively in your web applications.

### Chapter 6. Semantic HTML

Understand the importance of semantic elements for better accessibility and SEO, including `<header>`, `<footer>`, `<article>`, and `<section>`.

Learn how semantic HTML enhances the meaning of your content, making it more accessible to screen readers and search engines.

### Chapter 7. HTML Best Practices

Learn best practices for writing clean and maintainable HTML code, including proper indentation, use of comments, and validation.

This chapter emphasizes the importance of writing readable code and following best practices to ensure your HTML is valid and easy to maintain.

### Chapter 8. HTML5 New Features

Explore new features introduced in HTML5, such as native video and audio support, the `<canvas>` element, and improved form controls.

This chapter introduces the latest HTML5 features that enhance user experience and interactivity. You'll learn how to implement these features in your web applications.

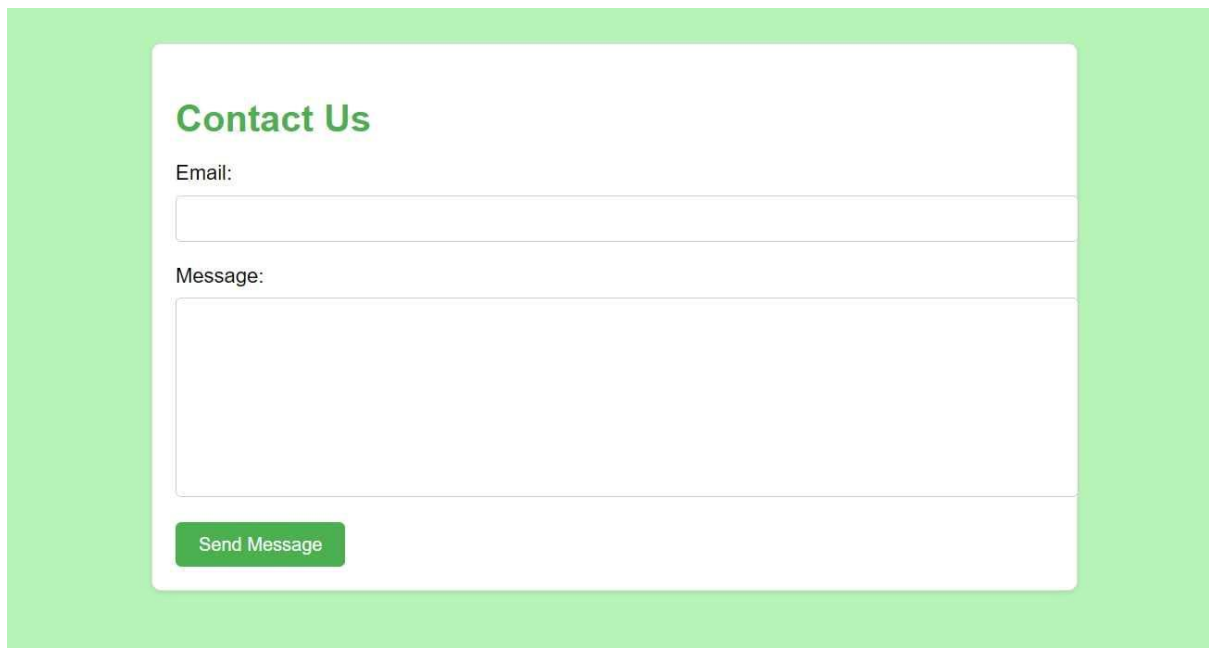
- In this page, it shows the chapters or the content of the webpage.

## Contact.blade.php code

```
contact.blade.php 1 X
G5layout > resources > views > contact.blade.php
1 @extends('components.Layout')
2
3 @section('title', 'HTML Roadmap - Contact Us')
4
5 @section('content')
6 <div class="container contact-container">
7   <div class="card">
8     <div class="card-header">
9       <h2>Contact Us</h2>
10    </div>
11    <div class="card-body">
12      <form action="{{ url('/submit-contact') }}" method="POST">
13        @csrf
14        <div class="form-group">
15          <label for="email">Email:</label>
16          <input type="email" class="input-box" id="email" name="email" required>
17        </div>
18        <div class="form-group">
19          <label for="message">Message:</label>
20          <textarea class="input-box" id="message" name="message" required></textarea>
21        </div>
22        <button type="submit" class="submit-btn">Send Message</button>
23      </form>
24    </div>
25  </div>
26 </div>
27 @endsection
```

- The `@extends('components.Layout')` tells Laravel to extend the layout defined in `components.Layout`, which means that the content here will be placed into the layout's structure.
- The `@section('title', 'HTML Roadmap - Contact Us')` sets the title for this page to "HTML Roadmap – Contact Us", which will appear in the browser tab or page title area, replacing the layout's title placeholder.
- The `@section('content')` marks the start of the content section where the actual content of the page (chapters) will be inserted into the layout's `@yield('content')`.
- The `@csrf` directive is added to the form to have a protection against crosssite request forgery.
- The `@endfor` <fThe `@endsection` marks the end of the content section.

## Contact Us page

A screenshot of a web form titled "Contact Us" in green text. The form is set against a light green background. It contains two input fields: "Email:" with a single-line text box, and "Message:" with a multi-line text area. Below these fields is a green button labeled "Send Message".

**Contact Us**

Email:

Message:

Send Message

- Prompts the user a contact us form, where you can input your email and the message you want to say.

## Part 3: Updating Routes

In routes /web.php, define routes to return the views:

A screenshot of a code editor (VS Code) showing the file explorer on the left and the code editor on the right. The file explorer shows a project structure with folders like G5layout, app, bootstrap, config, database, DOCS, public, resources, css, js, views, components, and routes. The routes folder is expanded, showing files like console.php, web.php, and storage. The code editor shows the content of web.php, which defines several routes using the Laravel routing system. The routes are: a root route returning 'welcome', a route for '/chapters' returning 'chapters', a route for '/contact' returning 'contact', and a route for '/user' that checks for a valid username and returns 'user' if valid, otherwise 'Guest'.

```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4
5 Route::get('/', function (): mixed {
6     return view('welcome');
7 });
8
9 Route::get('/chapters', function (): mixed {
10    return view('chapters');
11 });
12
13 Route::get('/contact', function (): mixed {
14    return view('contact');
15 });
16
17 Route::get('/user', function (): mixed {
18     $username = request()->input('username', 'Guest');
19     if (!preg_match(pattern: '/^[A-Za-z]+$/', subject: $username)) {
20         $username = 'Guest';
21     }
22     return view('user', ['username' => $username]);
23 });
```

- The “use Illuminate\Support\Facades\Route;” allows us to use Laravel’s routing system to set up page links.



- The “Route::get('/', function () { return view('welcome'); });” sets up a route for the home page (/). When someone goes to the home page, it shows the welcome page.
- The “Route::get('/chapters', function () { return view('chapters'); });” sets up a route for /chapters. When someone visits /chapters, the chapters page is shown.
- The “Route::get('/contact', function () { return view('contact'); });” sets up a route for /contact. When someone goes to /contact, the contact page is shown.
- The “Route::get('/user', function () { ... });” sets up a route for /user, which handles usernames from the form.
- The “\$username = request()->input('username', 'Guest');” gets the username from the URL. If no username is provided, it uses 'Guest' instead.
- The “if (!preg\_match('/^[A-Za-z]+\$/', \$username)) { \$username = 'Guest'; }” checks if the username has only letters (A-Z, a-z). If it contains anything else, it changes the name to "Guest".
- The “return view('user', ['username' => \$username]);” shows the user page and passes the username to the view, which can display it.

## Explain any challenges you faced and how you resolved them.

In doing this layout laboratory activity, we experienced some challenges like:

- **Maintaining a Consistent Layout Across Pages**

Issue: Keeping all the pages to have a uniform structure (with header, navigation, and footer) without repeating code.

Solution: A main layout file (Layout.blade.php) was created and Blade's `@extends` and `@yield` directives were used to apply this layout to individual views, allowing each page to follow the same design while inserting custom content.

- **Positioning of Layout Elements**

Issue: Keeping the navbar and footer fixed in place while allowing the content of each page to change.

Solution: Flexbox was implemented in the layout's CSS (`display: flex`, `flex-direction: column`) to ensure the header and footer stay in their positions, while the content section expands and adjusts to fill the remaining space.

- **Validating User Input**

Issue: Ensuring that the username input followed specific rules, only allowing alphabetic characters.

Solution: A regular expression (`preg_match('/^[A-Za-z]+$/', $username)`) was applied in the user route to check the username. If the input didn't match, the system defaulted to "Guest" as the username.

## Explore the difference between `{{slot}}` and `@yield`.

### `{{slot}}`

Use case: Used inside Blade components (not layouts) to define a placeholder for content that the component's user can pass.

How it works: It allows passing content directly into a reusable component when the component is created.

### `@yield`

Use case: Primarily used for defining sections in a layout file (like `Layout.blade.php`) where different views can inject their content.

How it works: It defines a placeholder for content that will be filled by the child views using `@section`.

## Comparison of `{{slot}}` and `@yield`.

- `{{slot}}` is used for passing content into Blade components, while `@yield` is used for defining sections in layout files for specific pages.
- `{{slot}}` gets its content when the component is created, whereas `@yield` requires `@section` to fill the placeholder.
- `{{slot}}` is for component-based development (smaller, reusable parts of a page), while `@yield` is for layout inheritance (typically whole-page structures).