Documentation

Prepared by: **Clarence John B. Maneja** **BSIT - 3C**

```php
<?php

use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;

// Root / home route
Route::get('/', function () {
    return view('home', ['message' => 'Welcome to the homepage!']);
})->name('home');

// Redirect from /home to /
Route::get('/home', function () {
    return redirect()->route('home');
});

// Homepage with an optional username parameter
Route::get('/homepage/{username?}', function ($username = 'Guest') {
    return view('homepage', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('homepage');

// About Us page with optional username parameter
Route::get('/about/{username?}', function ($username = 'Guest') {
    return view('about', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('about');

// Content page with optional username parameter
Route::get('/content/{username?}', function ($username = 'Guest') {
    return view('content', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('content');

// Contact Us page with optional username parameter
Route::get('/contact/{username?}', function ($username = 'Guest') {
    return view('contactPage', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('contactPage');

// Display a Contact Us form
Route::get('/contact-us', function () {
```

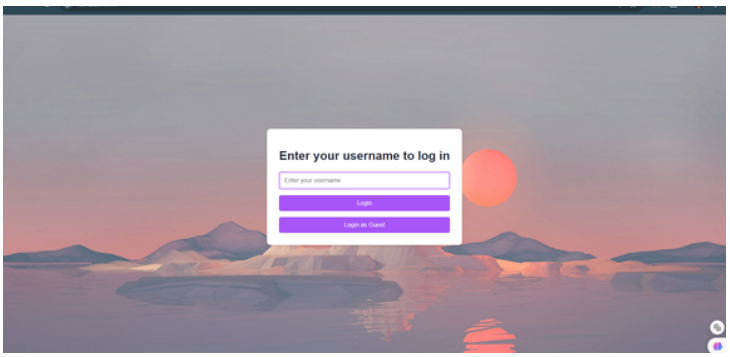
Part 1: Defining Basic Routes
Create a simple route that returns a view for the homepage.
The view should display a welcome message.
Create additional routes that:
Return a view for an "About Us" page.
Redirect from /home to / (the homepage).
Display a "Contact Us" form

We have defined four main routes: **home** (which is the root), homepage, about, content, and contact. Each route is specified using the **`Route::get()` method**, which handles HTTP GET requests. Optional username parameters are set for all routes, with the default username being 'Guest' if no username is provided.

Part 2: Using Route Parameters

Define a route with a required parameter: Create a route that accepts a username parameter and displays a welcome message that includes the username. Example: /user/johndoe should return a view

```php
// Route with required username parameter to display a welcome message
Route::get('/user/{username}', function ($username) {
    return view('welcomeUser', ['message' => "Welcome, $username!"]);
})->where('username', '[a-zA-Z]+')->name('welcomeUser');
```



When a user navigates to a URL such as `http://127.0.0.1:8000/user/clarence`, the system will redirect them to the root URL where they are prompted to either fill in their username or log in as a guest, displaying a message like **"Welcome, johndoe!"**. Each route has an optional username parameter, so if a username isn't provided, the default 'Guest' will be used.

```php
// Root / home route
Route::get('/', function () {
    return view('home', ['message' => 'Welcome to the homepage!']);
})->name('home');

// Redirect from /home to /
Route::get('/home', function () {
    return redirect()->route('home');
});

// Homepage with an optional username parameter
Route::get('/homepage/{username?}', function ($username = 'Guest') {
    return view('homepage', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('homepage');

// About Us page with optional username parameter
Route::get('/about/{username?}', function ($username = 'Guest') {
    return view('about', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('about');

// Content page with optional username parameter
Route::get('/content/{username?}', function ($username = 'Guest') {
    return view('content', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('content');

// Contact Us page with optional username parameter
Route::get('/contact/{username?}', function ($username = 'Guest') {
    return view('contactPage', ['username' => $username]);
})->where('username', '[a-zA-Z]+')->name('contactPage');

// Display a Contact Us form
Route::get('/contact-us', function () {
    return view('contactForm');
})->name('contactForm');

// Route with required username parameter to display a welcome message
```

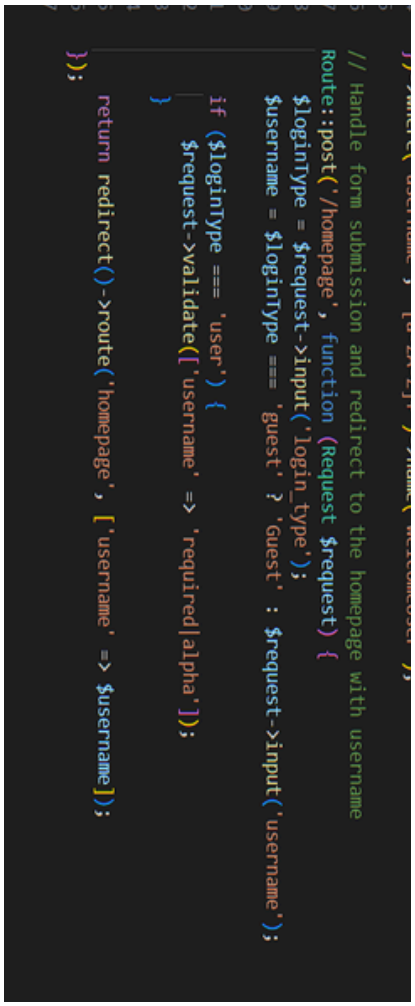
The GET request for `/contact-us` returns a view called `contactForm`, where the user can fill out a form to send an email.

This route handles POST requests to the `/homepage` URL, which is used for processing form submissions.

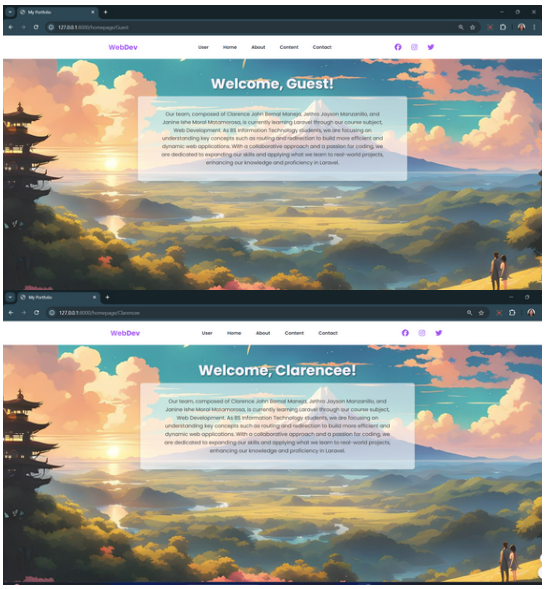
- Line 48: Retrieves the value from the input field of the submitted form data and stores it in the `$loginType` variable.
- Line 49: Checks whether the $loginType variable contains "guest" or "username." If it includes a username, the value of the username input field is retrieved and assigned to the $username variable. The user is then redirected to the homepage, where a "Welcome, User" message is displayed.

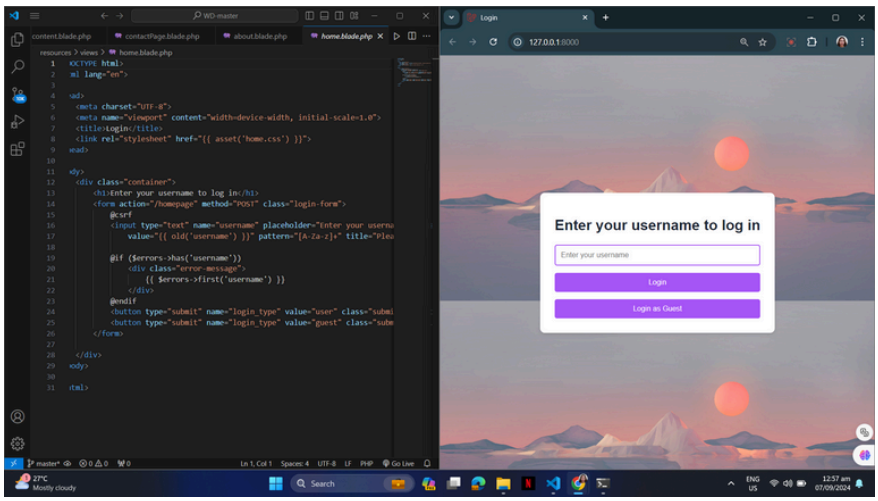
Define a route with an optional parameter: Modify the previous route to make the username optional. If no username is provided, display a generic welcome message. Example: /user should return a view with "Welcome, Guest!". Apply regular expression constraints to the route parameters: Ensure that the username only accepts alphabetic characters (az, A-Z).

Here are the various routes that include an optional username parameter. If no username is provided, the user has two sign-in options: (1) logging in with a provided username through the form, or (2) logging in as a guest without providing a username. The route then returns the `homepage` view with the username parameter. Additionally, the `where` clause applies a regular expression constraint to the username, allowing only alphabetic characters (a-z, A-Z).



Part 3: Documentation (Individual) Take screenshots of your application runtime along with its code. Provide a detailed explanation of how your code works, highlighting the purpose of each key section.



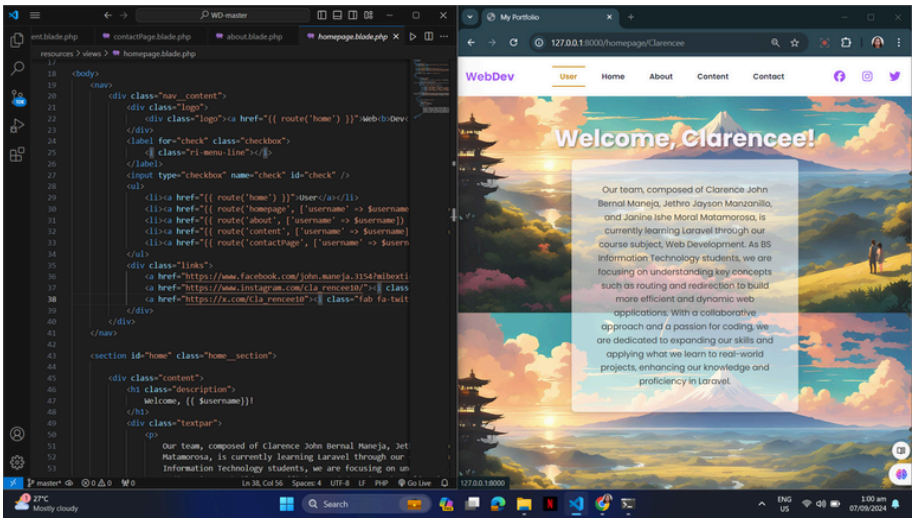


Optional Username Log In (home.blade.php)
User Options:
- Users can log in with a provided username.
- Users can log in as a guest.

How Does the Code Work?
- Form Submission: The form data is submitted using the POST method.
- CSRF Protection: The `@csrf` directive, a Blade syntax, is used to protect against Cross-Site Request Forgery (CSRF) attacks by ensuring the request originates from the same origin.
- Initial Input Value: On line 17, the initial value of the input field is set to the previously entered value (if any) using the `old()` helper function.
- Regular Expression Pattern: A regular expression pattern is applied to ensure that the input value meets certain criteria.
- Validation Check: On line 19, the Blade syntax checks for validation errors related to the username. The `@endif` directive is used to close the `@if` statement.

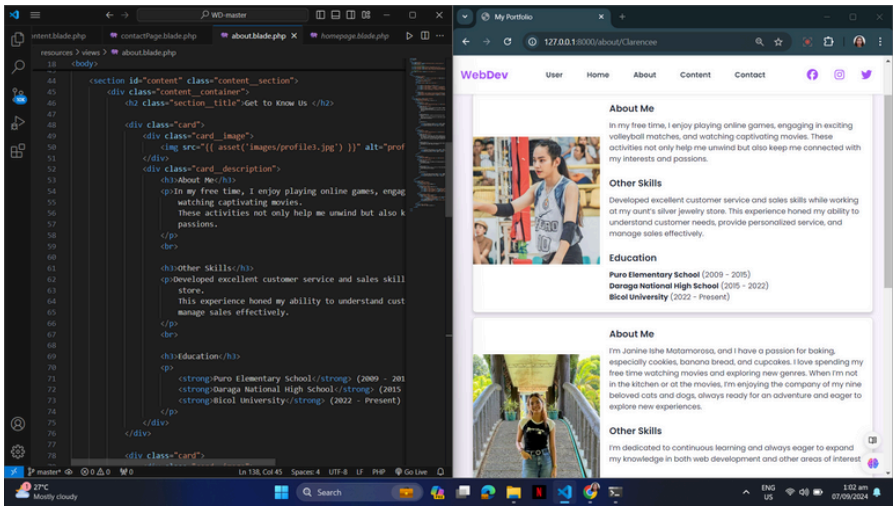
- Conclusion: This form allows users to either enter their username and log in as a registered user or log in as a guest. The form data is then sent to the `/homepage` URL using the POST method.

homepage.blade.php



- Route Helpers (Lines 29 to 33): These lines use route helpers to generate URLs for the corresponding routes. This approach allows for easy linking to different pages of the website without hardcoding the exact URLs.

- Welcome Message (Line 47): This line displays a welcome message that includes the user's username.



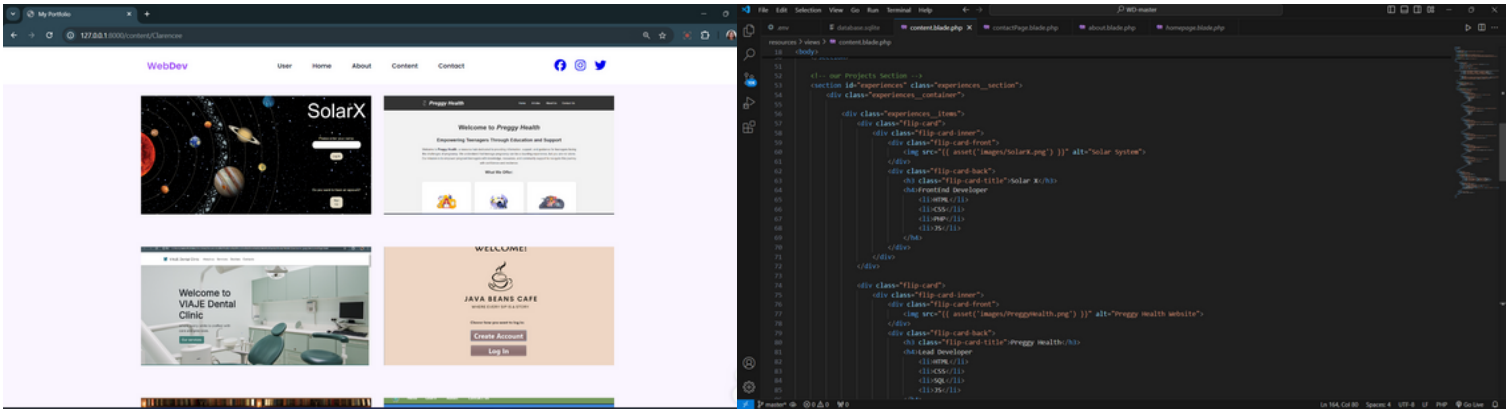
about.blade.php

- Route Helpers: The about page utilizes route helpers to generate URLs for the corresponding routes, making navigation consistent and avoiding hardcoding URLs.

- Page Content: The about page primarily provides information about our group.

content.balde.php

- Basically, this uses the same navigation as the homepage and the about page



contact.balde.php

- Navigation Links: I used the same navigation links for the homepage, about, contents, and contact pages. Each link's URL is dynamically generated based on the defined routes and any provided parameters, making it easy to navigate between different sections of our website.

- Form Submission Script: This script captures the form submit event, prevents the default behavior, and retrieves the values entered in the name, email, subject, and message fields. It then creates a mailto: link with the recipient's email address (itwebdev10@gmail.com), automatically filling in the subject and body of the email with the provided details. Finally, the script opens the default email client with the pre-filled information, allowing us to send the message directly by redirecting to this mailto: link.