Prepared by: **Janine Ishe M. Matamorosa** **BSIT 3C**

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.

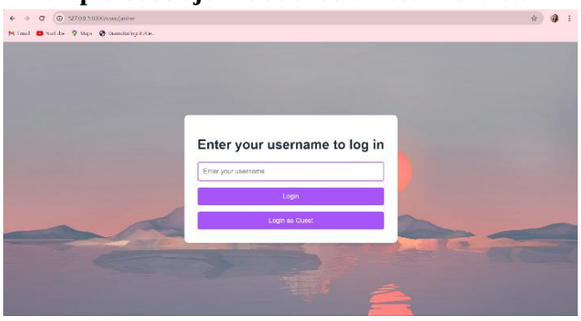
```php
// Root / home route
Route::get('/', function () {
    return view('home', ['message' => 'Welcome to the homepage!']);
})->name('home');

// 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');
```

Here are the four routes that we have defined for routing and these are are **home** which is the root, homepage, about, content, and contact. This **Route::get() method**, this specifies the contents of an HTTP GET request, this is used to specify each route. Optional username parameters are set for every route because if the username is not provided, the default username value will going to be Guest.

Part 2: Using Route Parameters

```php
// Route with required username parameter to display a welcome message
Route::get('/user/{username}', function ($username) {
    return view('home');
})->name('welcomeUser');      You, 11 seconds ago • Uncommitted changes

// Handle form submission and redirect to the homepage with username
```

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 with "Welcome, johndoe!".



Since all of the route have their own optional parameter, what this will do is, if the user type on the url (http://127.0.0.1:8000/user/janine) it will redirect them to the root url which where they are required to fill the forms with their username or just Log in as GUEST. (I did not update this on the github repository but it is included with different version.

```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');
```

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 (a-z, A-Z).

- Here are the different routes with an optional parameter if there is no username provided. But as you can see from above, We have two sign in options. Option (1) is to Log in with a provided username on the from. Option (2) is to log in as guest without providing a username on the forms. The route then returns a view called homepage with the username parameter. The *where* clause applies a regular expression constraint to the username parameter, allowing only alphabetic characters (a-z, A-Z).

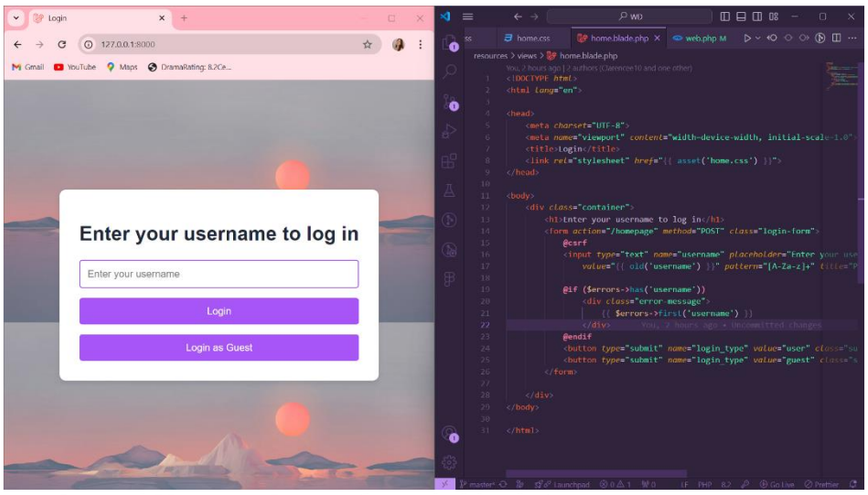
- The GET request for /contact-us and returns a view called contactForm where there is a form to fill up for the user to send email.

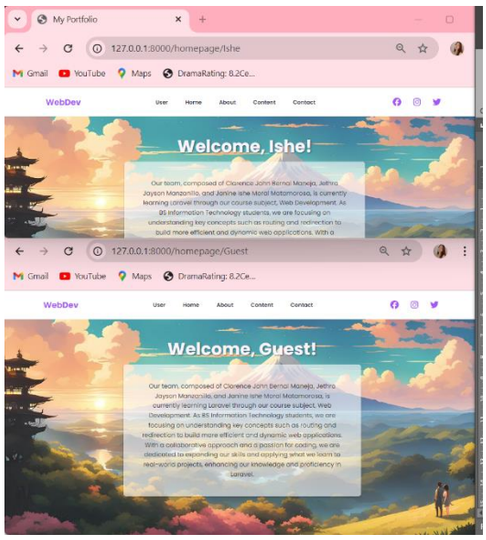
```php
// Handle form submission and redirect to the homepage with username
Route::post('/homepage', function (Request $request) {
    $loginType = $request->input('login_type');
    $username = $loginType === 'guest' ? 'Guest' : $request->input('username');

    if ($loginType === 'user') {
        $request->validate(['username' => 'required|alpha']);
    }

    return redirect()->route('homepage', ['username' => $username]);
});
```

Defines the route for handling POST requests to the /homepage URL. The route is used for processing the form submissions. The purpose of line 48 is to retrieve the value of the input field from the submitted form data and stores it in the $loginType variable. Line 49 is for checking if the value of the loginType contains “guest” or it has a “username” content. If it does have a username

included then the value of username input field is retrieved and assigned to the $username variable and will then be passed or redirected to the homepage and finally showing the Welcome, User message.

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.





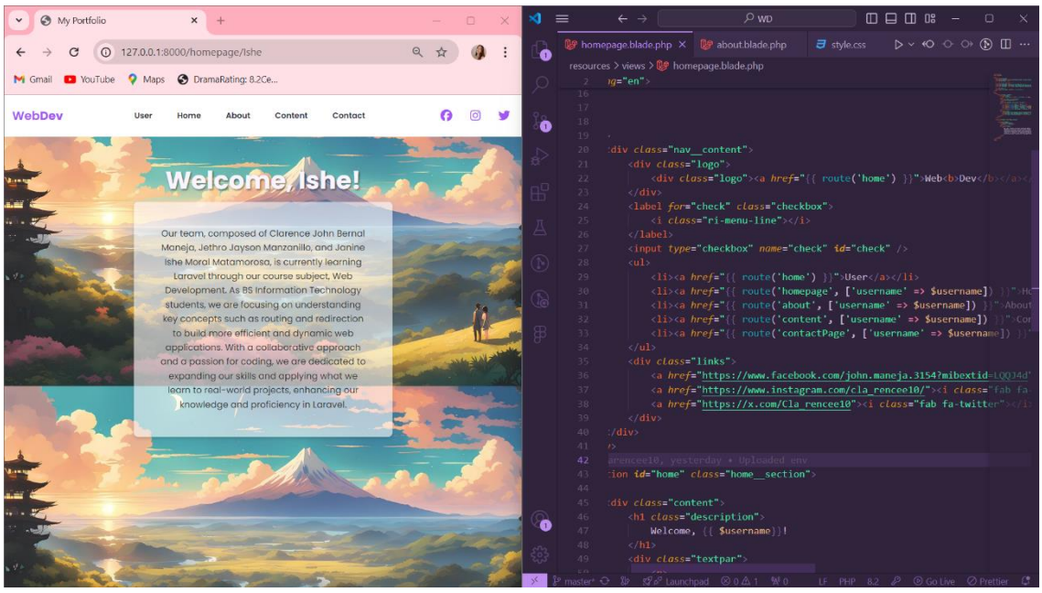
Optional Username Log in. **(home.balde.php)**

- User can Log in with a provided username
- User can log in as a guest.

How does the code work?

- The method that we use on submitting the from data is POST method.
- The @csrf (Cross-Site Request Forgery) a balde syntax used to protect against CSRF attacks by verifying that the request came from the same origin.
- The line 17 set the initial value fo the input field to the previously entered value (if any) using the old() helper function.
- Of course there will also be a regular expression pattern that the input value must match.
- Line 19 I also saw this on a video, this is a blade syntax that checks if there is a validation errors for the username, and the @endif it closes the @if statement.
- So, to conclude this form allows users to enter their username and choose to log in as a registered user or a guest. And the form data is sent to the /homepage URL using the POST method.

homepage.blade.php

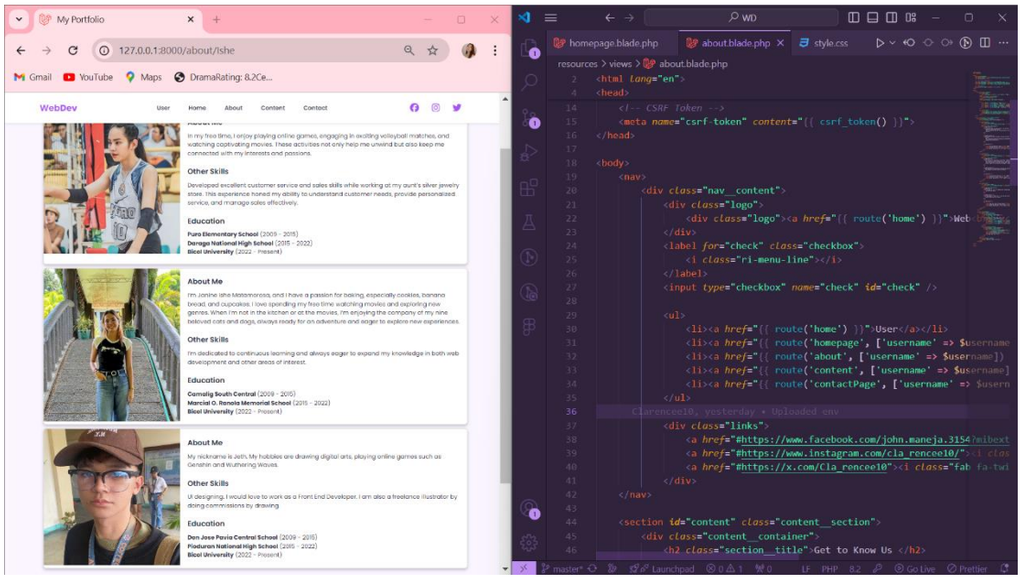


- As you can see from line 29 to line 33 these are the route helpers that generate URLs for the corresponding routes. It's a way to easily link to different pages of the website without hard-coding the exact URL.
- Line 47 displays the welcome message with the user's username.

About.blade.php

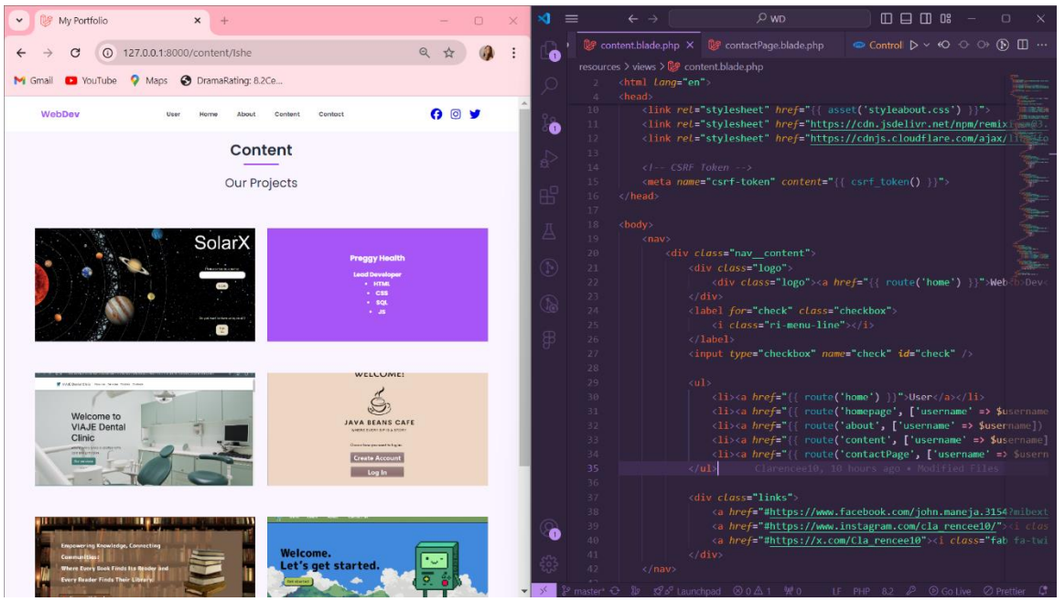
- The about page also uses route helpers that generate URLs for the corresponding routes.
- The about page just basically show information about our group.

Prepared by: **Janine Ishe M. Matamorosa** **BSIT 3C**

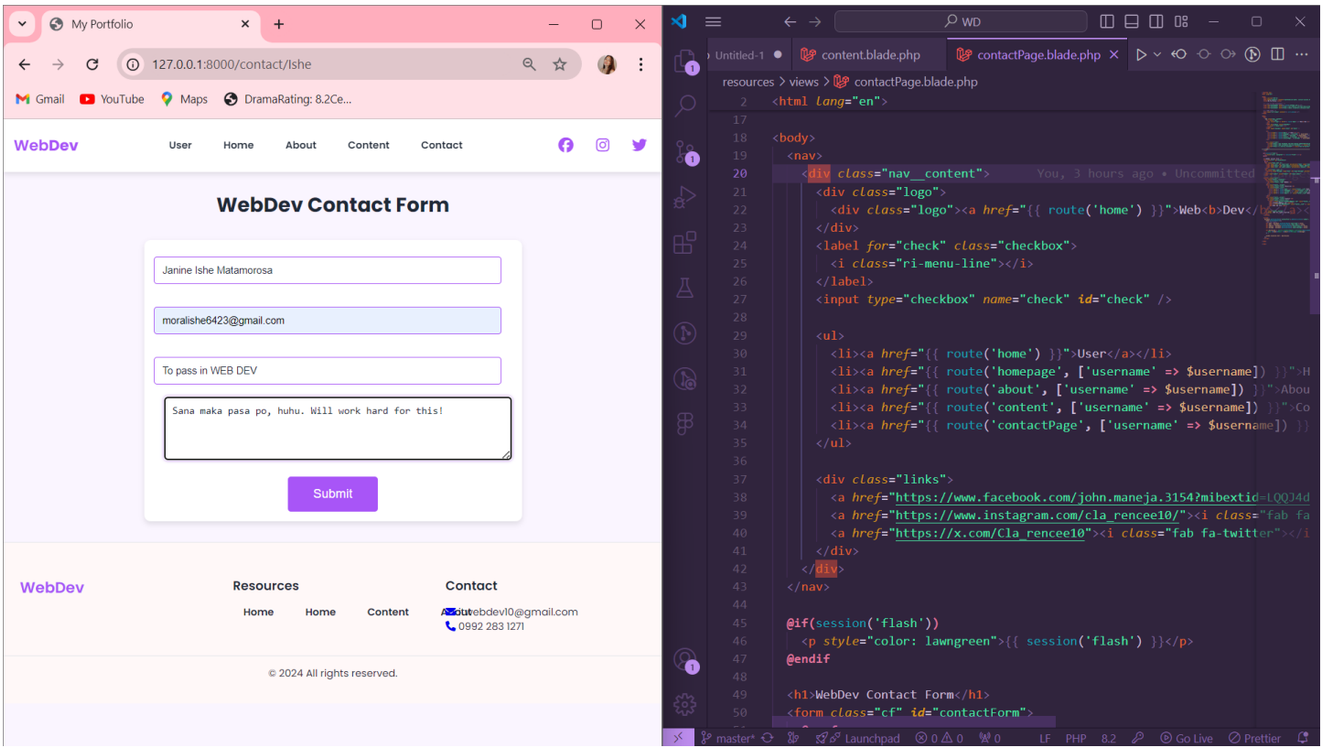


Content.balde.php

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



Contact.blade.php



Prepared by: **Janine Ishe M. Matamorosa** **BSIT 3C**

- I used the same navigation links from *homepage, about, contents, and contact* because, each link's URL is created dynamically based on the routes that have been set and any parameters which have been provided making it simple to navigate between the various parts of our website.
- In this script, we capture the form submit event, stop the default behavior, and retrieve the values we entered in the name, email, subject, and message fields. Then, we create a mailto: link with the recipient's email address, itwebdev10@gmail.com, automatically filling in the subject and body of the email with our details. Finally, the script opens our default email client with the pre-filled information and redirects us to this mailto: link, allowing us to send the message directly.