

# Documentation

## 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

```
routes > web.php
1 use Illuminate\Support\Facades\Route;
2 use Illuminate\Http\Request;
3
4 // Root / home route
5 Route::get('/', function () {
6     return view('home', ['message' => 'Welcome to the homepage!']);
7 })->name('home');
8
9 // Redirect from /home to /
10 Route::get('/home', function () {
11     return redirect()->route('home');
12 });
13
14 // Homepage with an optional username parameter
15 Route::get('/homepage/{username?}', function ($username = 'Guest') {
16     return view('homepage', ['username' => $username]);
17 })->where('username', '[a-zA-Z]*')->name('homepage');
18
19 // About Us page with optional username parameter
20 Route::get('/about/{username?}', function ($username = 'Guest') {
21     return view('about', ['username' => $username]);
22 })->where('username', '[a-zA-Z]*')->name('about');
23
24 // Content page with optional username parameter
25 Route::get('/content/{username?}', function ($username = 'Guest') {
26     return view('content', ['username' => $username]);
27 })->where('username', '[a-zA-Z]*')->name('content');
28
29 // Contact Us page with optional username parameter
30 Route::get('/contact/{username?}', function ($username = 'Guest') {
31     return view('contactPage', ['username' => $username]);
32 })->where('username', '[a-zA-Z]*')->name('contactPage');
33
34 // Display a Contact Us form
35 Route::get('/contact-us', function () {
```

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.

**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).**

## 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**

```
// 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');
```

This route handles GET requests to `/user/{username}`, capturing the `username` parameter to display a personalized welcome message in the `welcomeUser` view. The `username` is restricted to alphabetic characters, and the route is named `welcomeUser` for easy reference.

```
// 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
```

These routes define pages for a Laravel application with optional `username` parameters, defaulting to "Guest" if not provided.

The `homepage`, `about`, `content`, and `contactPage` views each display a page with the username, ensuring consistency and personalization, while the `contactForm` view presents a contact form without requiring a username. The `username` parameter is restricted to alphabetic characters, and each route is named for convenient referencing throughout the application.

```

)) >where( 'username', ['a-zA-Z'] ) >name( 'welcomeuser' );

// 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']);
    }

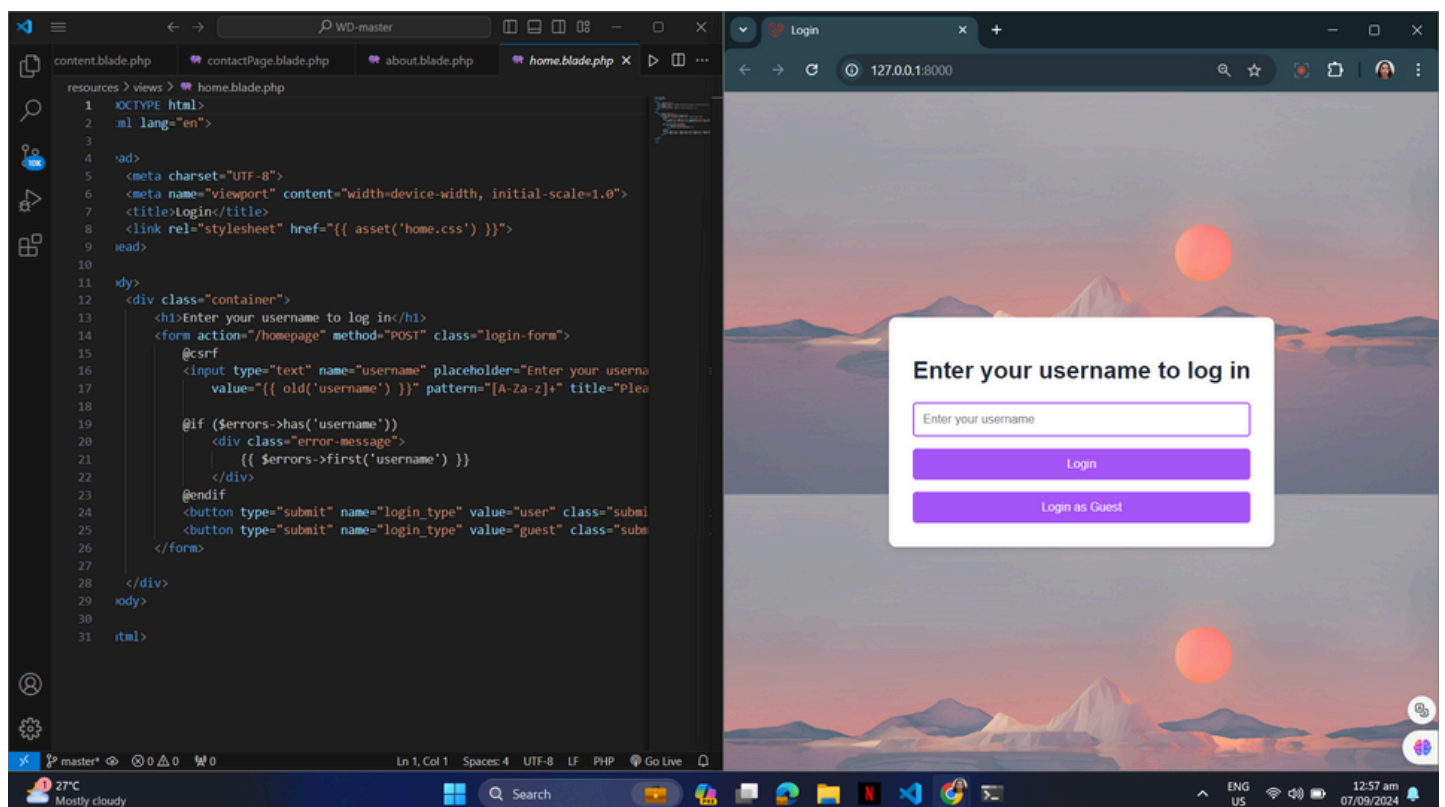
    return redirect()->route('homepage', ['username' => $username]);
});

```

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).

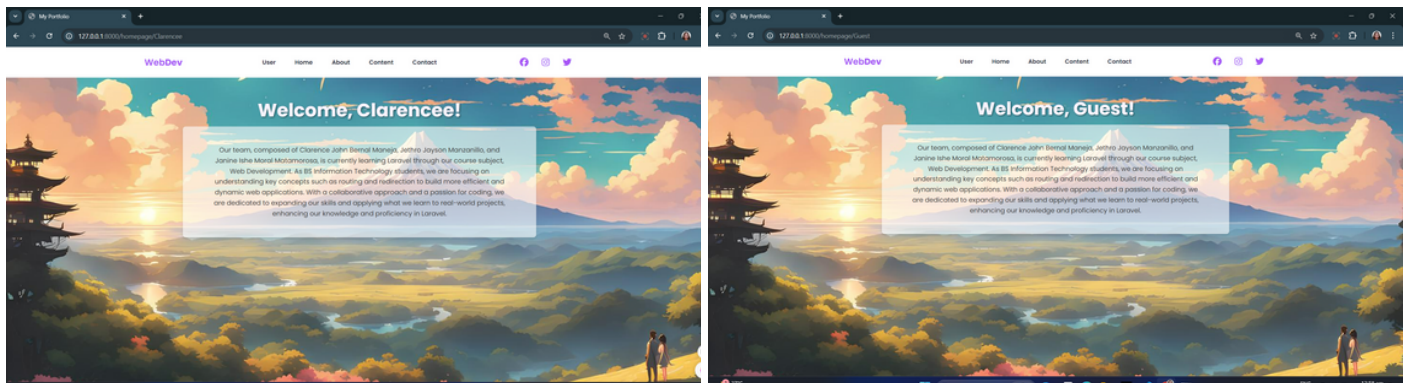
The GET request for `/contact-us` returns a view called `contactForm`, where the user can fill out a form to send an email.

**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.**

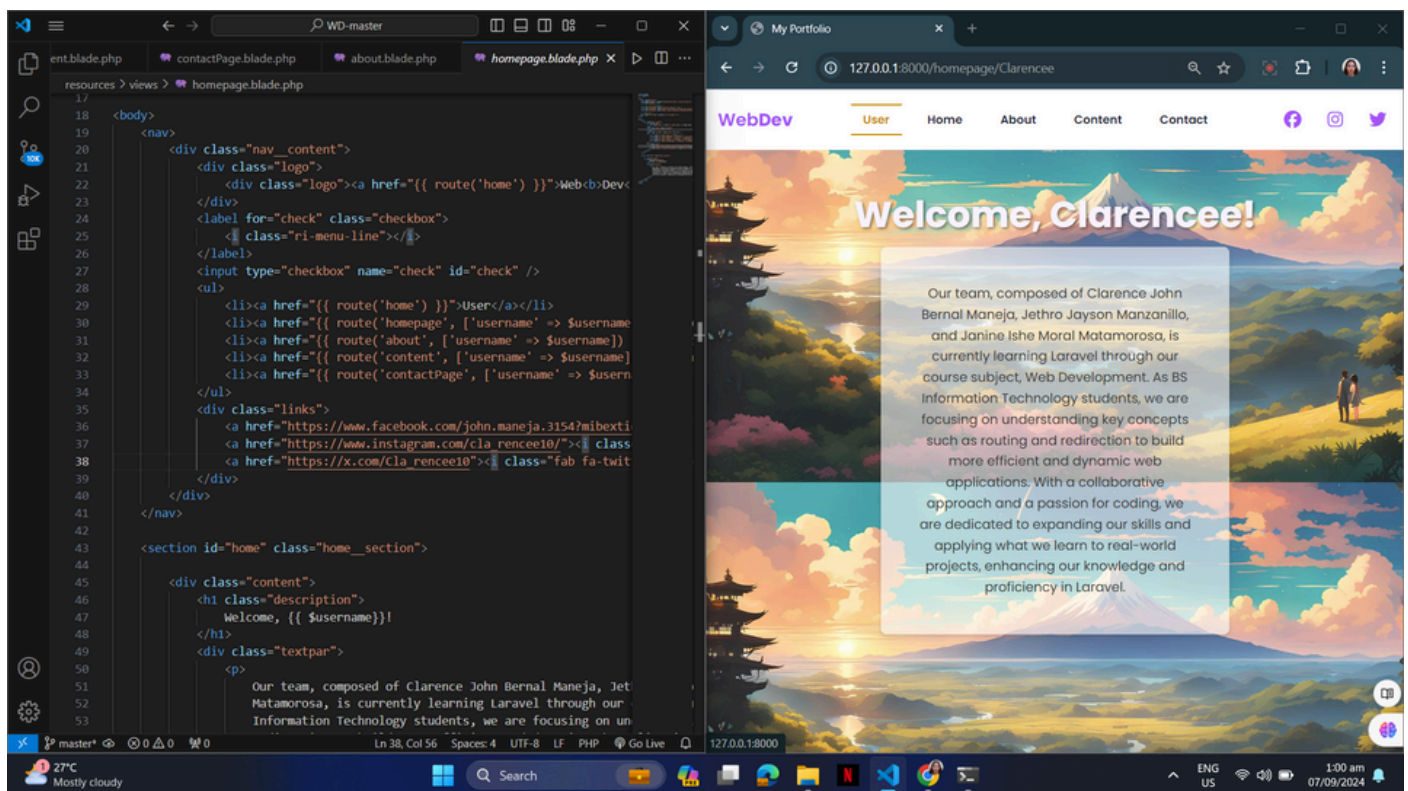


The image shows a side-by-side comparison of code and its runtime output. On the left, a code editor displays the Blade template for the login form in `home.blade.php`. The code includes a form with a text input for the username, a CSRF token, and two submit buttons for 'user' and 'guest' login types. It also includes error handling for the username field. On the right, a web browser at the address `127.0.0.1:8000` shows the rendered login form. The form has a title 'Enter your username to log in', a text input field with the placeholder 'Enter your username', and two buttons: 'Login' and 'Login as Guest'. The background of the browser window shows a sunset over mountains.

- This login form allows users to either enter their username to log in as a registered user or log in as a guest, with the form data submitted via the POST method to the `/homepage` URL. The form is protected against CSRF attacks using the `@csrf` directive, and the `old()` helper function sets the initial input value to any previously entered data. A regular expression pattern is used to validate the input, and any validation errors related to the username are handled within the form using Blade syntax.

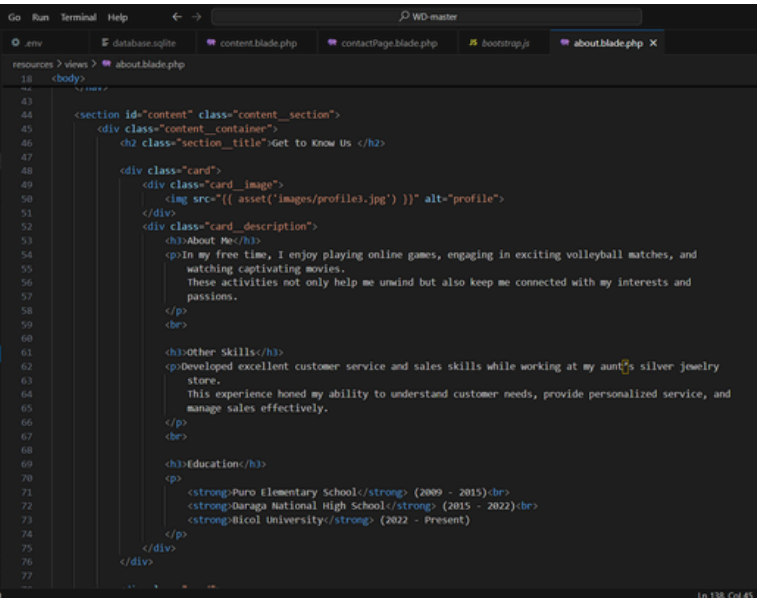


## homepage.blade.php



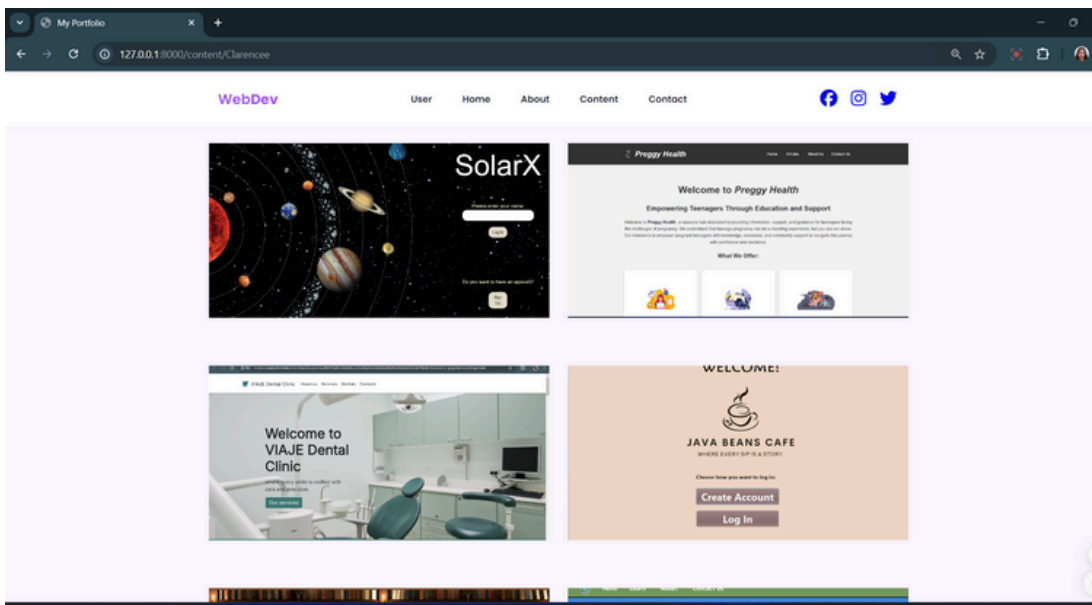
- The code uses route helpers (lines 29 to 33) to generate URLs dynamically, ensuring easy navigation between different pages without the need to hardcode URLs. This method enhances flexibility and maintainability as the routes can be updated without modifying the entire code. Additionally, the welcome message on line 47 incorporates the user's username, personalizing the experience for each user who logs in.





- The ``about.blade.php`` page leverages route helpers to dynamically generate URLs for navigation, ensuring consistency across the site and eliminating the need for hardcoded URLs. The main content of the page focuses on providing detailed information about the group.

## content.balde.php



```

18 <body>
19
20 <!-- our Projects Section -->
21 <section id="experiences" class="experiences_section">
22   <div class="experiences_container">
23     <div class="experiences_items">
24       <div class="flip-card">
25         <div class="flip-card-inner">
26           <div class="flip-card-front">
27             
28           </div>
29           <div class="flip-card-back">
30             <h3 class="flip-card-title">Solar X</h3>
31             <h4>FrontEnd Developer</h4>
32             <ul>
33               <li><HTML</li>
34               <li><CSS</li>
35               <li><PHP</li>
36               <li><JS</li>
37             </ul>
38           </div>
39         </div>
40       </div>
41       <div class="flip-card">
42         <div class="flip-card-inner">
43           <div class="flip-card-front">
44             
45           </div>
46           <div class="flip-card-back">
47             <h3 class="flip-card-title">Preggy Health</h3>
48             <h4>Lead Developer</h4>
49             <ul>
50               <li><HTML</li>
51               <li><CSS</li>
52               <li><SQL</li>
53               <li><JS</li>
54             </ul>
55           </div>
56         </div>
57       </div>
58     </div>
59   </div>
60 </section>
61 </body>

```

- Basically, this uses the same navigation as theThe navigation structure for this page is essentially the same as that used on both the homepage and the about page. Each navigation link is dynamically generated, ensuring consistency and ease of navigation across the entire website. homepage and the about page

## contact.blade.php

```

18 <body>
19
20 <!-- our Projects Section -->
21 <section id="experiences" class="experiences_section">
22   <div class="experiences_container">
23     <div class="experiences_items">
24       <div class="flip-card">
25         <div class="flip-card-inner">
26           <div class="flip-card-front">
27             
28           </div>
29           <div class="flip-card-back">
30             <h3 class="flip-card-title">Solar X</h3>
31             <h4>FrontEnd Developer</h4>
32             <ul>
33               <li><HTML</li>
34               <li><CSS</li>
35               <li><PHP</li>
36               <li><JS</li>
37             </ul>
38           </div>
39         </div>
40       </div>
41       <div class="flip-card">
42         <div class="flip-card-inner">
43           <div class="flip-card-front">
44             
45           </div>
46           <div class="flip-card-back">
47             <h3 class="flip-card-title">Preggy Health</h3>
48             <h4>Lead Developer</h4>
49             <ul>
50               <li><HTML</li>
51               <li><CSS</li>
52               <li><SQL</li>
53               <li><JS</li>
54             </ul>
55           </div>
56         </div>
57       </div>
58     </div>
59   </div>
60 </section>
61 </body>

```

The navigation links for the homepage, about, contents, and contact pages are consistently implemented across the site, with each URL dynamically generated based on the defined routes and any provided parameters, facilitating seamless navigation.

The form submission script enhances user interaction by capturing the form's submit event, preventing the default behavior, and retrieving input values such as name, email, subject, and message. It constructs a `mailto:` link directed to `itwebdev10@gmail.com`, automatically filling in the email's subject and body with the user's details. This action then triggers the default email client, pre-populated with the user's information, ready to send the message directly through the `mailto:` link.

```

<script>
document.getElementById('contactForm').addEventListener('submit', function (event) {
  event.preventDefault();
  //get the inputs
  var name = document.getElementById('input-name').value;
  var email = document.getElementById('input-email').value;
  var subject = document.getElementById('input-subject').value;
  var message = document.getElementById('input-message').value;

  var mailtoLink = `mailto:itwebdev10@gmail.com?subject=${encodeURIComponent(subject)}&body=${encodeURIComponent(
    `Name: ${name}\nEmail: ${email}\n\nMessage:\n${message}`
  )}`;

  window.location.href = mailtoLink;
});
</script>

```