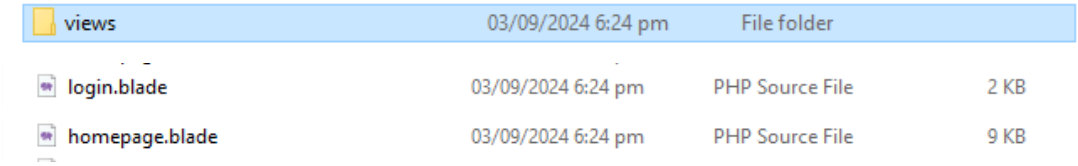
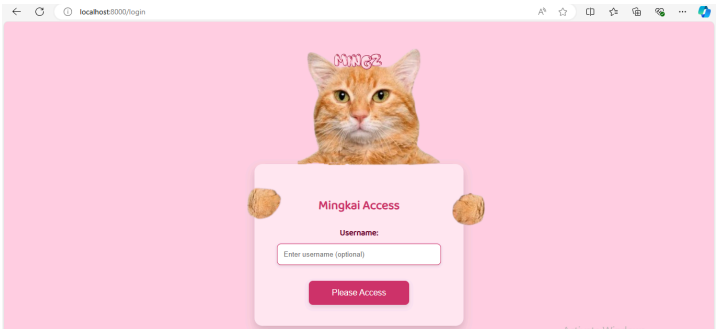
Sarah C. Abane BSIT 3C WebDev_Lab2

DOCUMENTATION

Part 1: Defining Basic Routes

1. Create a simple route that returns a view for the homepage. The view should display a welcome message.





Homepage Blade Template (homepage.blade.php):

- This is a simple Blade view where the welcome message is displayed. The message is passed to the view:

```
<div class="welcome-message">
<h4>♡♡ {{ $message }} ♡♡</h4>
</div>
```

Controller Logic for Displaying the Welcome Message

- In the **LoginController**, we have a method called homepage(), which handles the logic for retrieving the username from the request and passing it to the view
- This function checks if a username is present in the URL or request, defaulting to "Guest" if not provided. The username is then passed to the view as part of the $message variable

(logic/code for displaying the welcome message)

```
1 reference | 0 overrides
public function homepage(Request $request, $username = null)
{
    // Retrieve the username from the URL, or from the session if not provided
    $username = $username ?: $request->session()->get('username', 'guest');
    return view('homepage', ['message' => "Welcome, $username!"]); // Pass username to the homepage view
}
```

Route Logic for Returning the View:

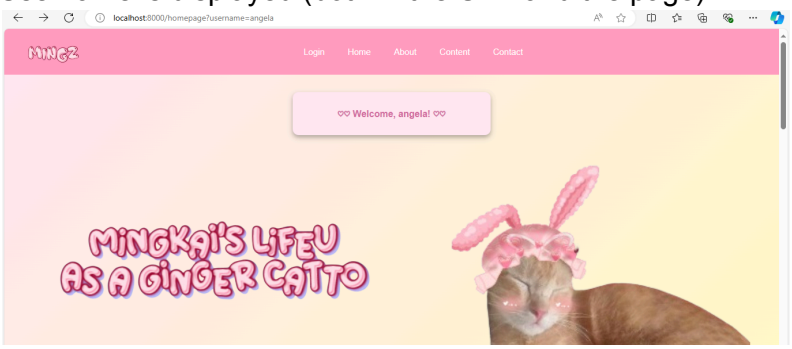
In the **web.php** file, the Route::get() functions are used to map HTTP GET requests to specific controller methods.

```
Route::get('/', [LoginController::class, 'homepage'])->name('homepage');
Route::get('/about/{username?}', [LoginController::class, 'about'])->name('about');
Route::get('/content/{username?}', [LoginController::class, 'content'])->name('content');
Route::get('/contact/{username?}', [LoginController::class, 'contact'])->name('contact');
Route::get('/user/{username?}', [LoginController::class, 'userProfile'])->name('user.profile');
```

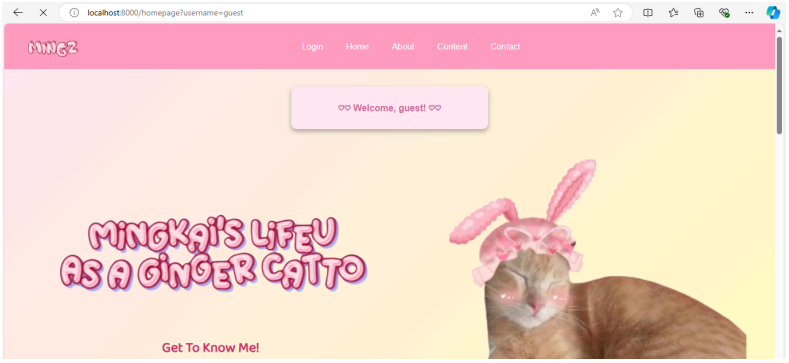
This code passes the username to the Blade view, ensuring that the personalized message ("Welcome, [username]") is displayed when a user visits the homepage.

```
return view('homepage', ['message' => "Welcome, $username!"]);
```

Username is displayed (both in the URL and the page)



If a username is not provided (guest)



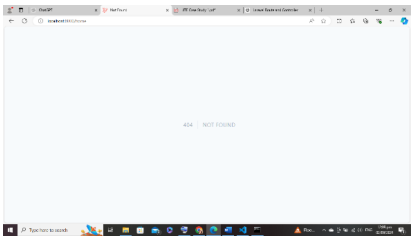
2. Create additional routes that:
   Return a view for an "About Us" page.
   - It links the /about URL to the about method in the LoginController and allows an optional username.

```
// About route with an optional parameter {username?}
Route::get('/about/{username?}', [LoginController::class, 'about'])
    ->name('about')
    ->where('username?', '[A-Za-z]'); // Constraint: only alphabetic characters
```

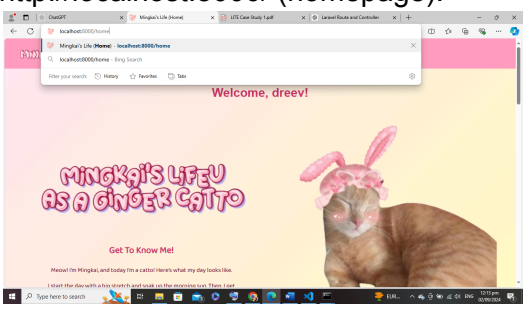
   Redirect from /home to / (the homepage).
   This route redirects requests from /home to the root URL /, effectively redirecting users to the homepage.

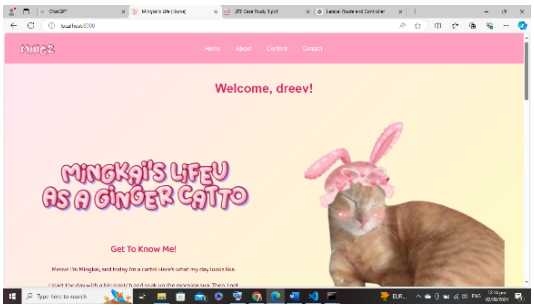
```php
// Redirect /home to /
Route::redirect('/home', '/');
```

- Without redirection: Returns a 404 page because /home is not defined.

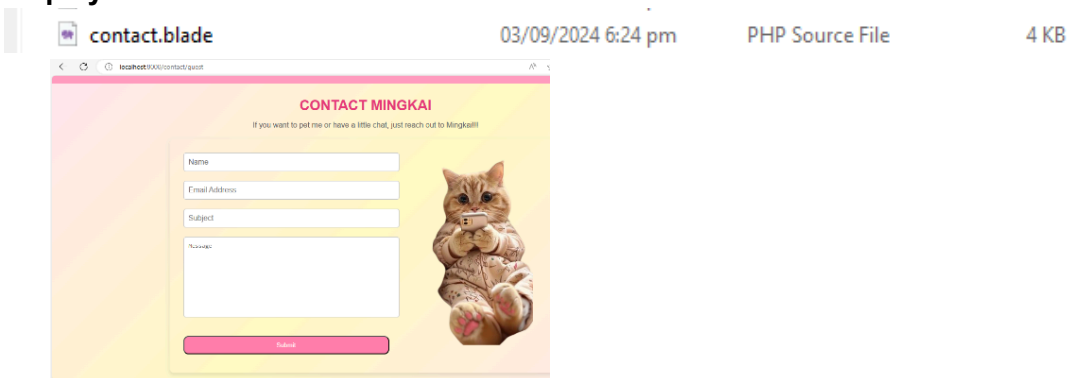


- With redirection: The route configuration automatically redirects the user to http://localhost:8000/ (homepage).





Display a "Contact Us" form.



Part 2: Using Route Parameters

1. Define a route with a **required** parameter:
    - The **{username}** parameter is required, meaning the route /user/johndoe (Ex: the entered username is johndoe) will pass "johndoe" to the **userProfile method** in the LoginController.

```php
Route::get('/user/{username?}', [LoginController::class, 'userProfile'])
    ->name('user.profile')
    ->where('username?', '[A-Za-z]'); // Constraint: only alphabetic characters
```

2. Define a route with an **optional** parameter:

- The **{username?}** makes the username optional. If no username is provided, the **userProfile method** defaults to using 'guest' as the username.
  If no username is provided, it will display "Welcome, Guest!".

- In the **LoginController,** we have a method like this: The method sets a default username if not provided, stores it in the session, and redirects the user to the homepage with the username in the URL.

```
    1 reference | 0 overrides
    public function userProfile(Request $request, $username = null)
    {
        // Default to 'guest' if no username is provided
        $username = $username ?: 'guest';

        // Store the username in the session
        $request->session()->put('username', $username);

        // Redirect to the homepage with username in the URL
        return redirect()->route('homepage', ['username' => $username]);
    }
}
```

3. Apply **regular expression constraints** to the route parameters:
   - The `where()` method applies a regular expression constraint on the parameter: **`[A-Za-z]+`**: Ensures that only alphabetic characters are allowed, both uppercase (`A-Z`) and lowercase (`a-z`). The + means one or more characters are required if the username is provided.

```
->where('username?', '[A-Za-z]'); // Constraint: only alphabetic characters
```

   - We also provide logic in the login form/**login.blade.php**: The * allows for an empty string, meaning the username is optional. There's also an error message in case the user inputs invalid characters.

```
<form action="{{ route('login') }}" method="POST">
    @csrf
    <label for="username">Username:</label>
    <input type="text" id="username" name="username"
        pattern="[A-Za-z]*"
        title="Only alphabetic characters (a-z, A-Z) are allowed."
        placeholder="Enter username (optional)">
        <button type="submit">Please Access</button>
</form>
```

- Validation in the Controller
  - **`'nullable|alpha'`**: This server-side validation rule ensures that the username can either be `null` (optional) or, if provided, it must only consist of alphabetic characters.

```
$request->validate([  // Validate the request (allowing optional username)
    'username' => 'nullable|alpha', // Ensure username contains only alphabetic characters
]);
```