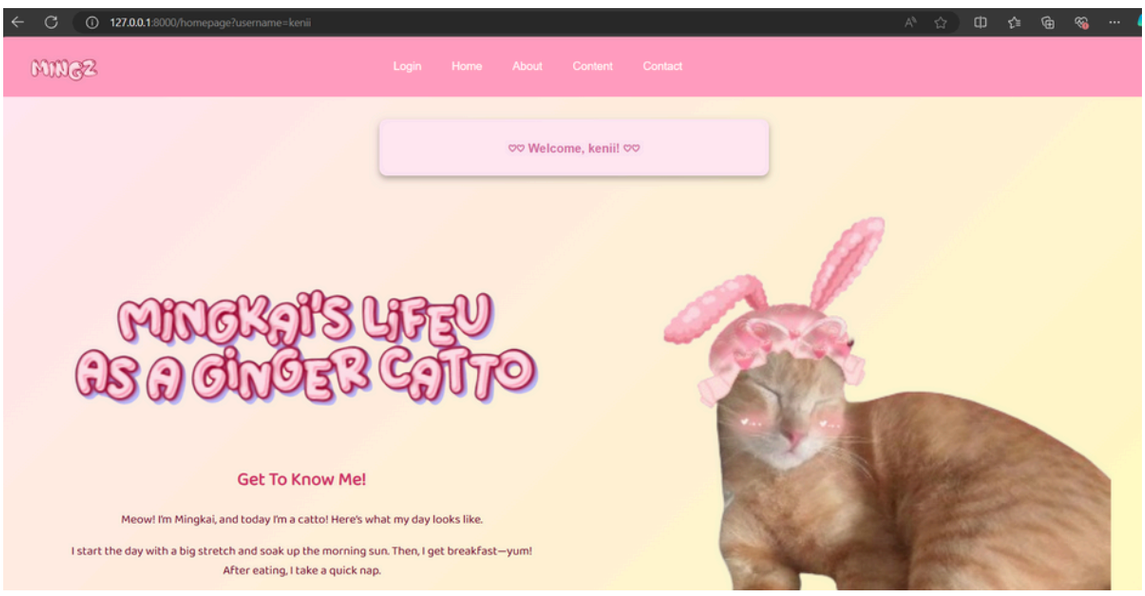
PART 1 : BASIC ROUTES

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

CODE (Route):

```
// Homepage route
Route::get('/homepage', [LoginController::class, 'homepage'])
    ->name('homepage');
```

OUTPUT:



EXPLANATION:

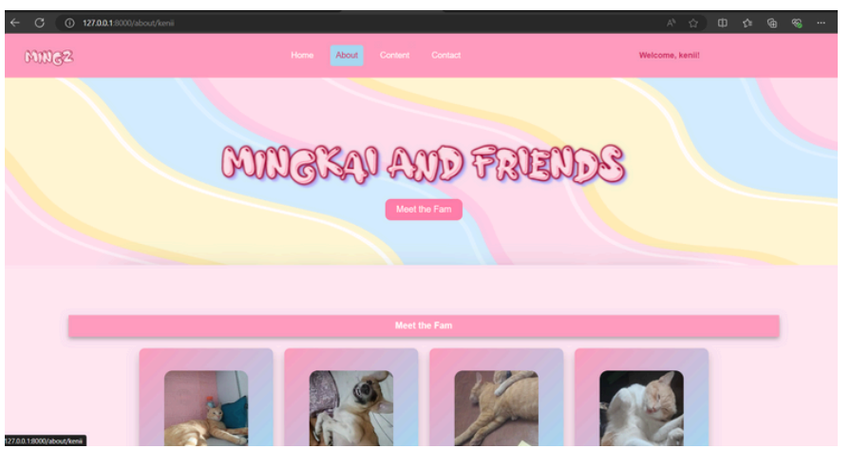
This route that returns a view for the homepage displaying a welcome message after you successfully access the web pages using the mingkai access.

- **Return a view for an "About Us" page.**

CODE (Route):

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

OUTPUT:



EXPLANATION:

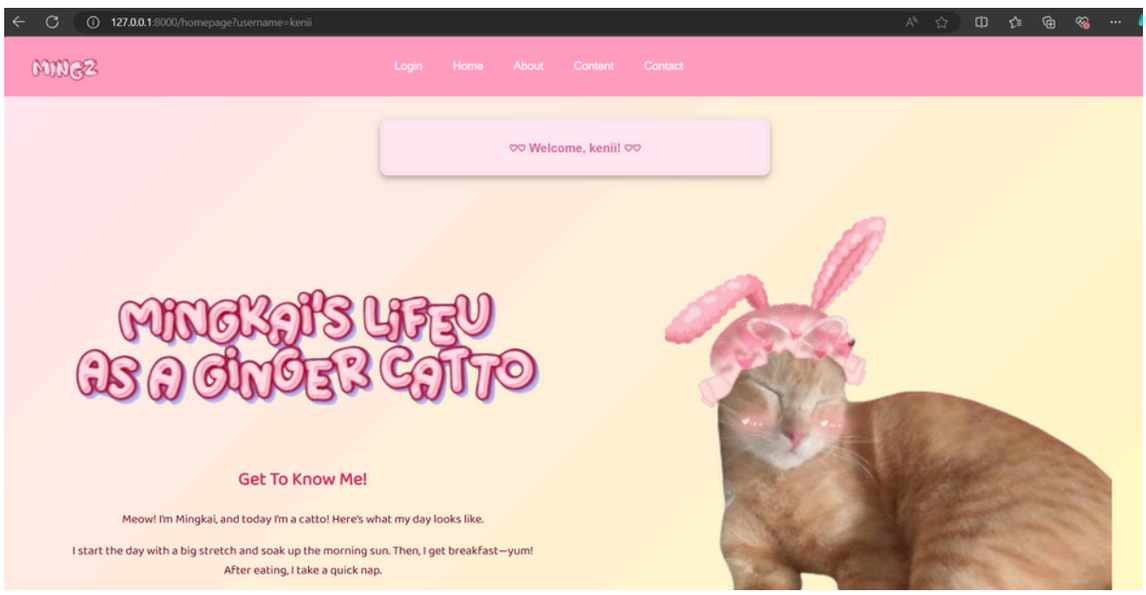
This route that returns a view for the about us page.

- **Redirect from /home to / (the homepage)**

CODE (Route):

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

OUTPUT:



EXPLANATION:

The `Route::redirect('/home', '/');` in Laravel automatically sends users from `/home` to the homepage (`/`). It's a simple way to ensure that anyone visiting `/home` is redirected to the root of our site.

- **Display a "Contact Us" form.**

CODE:

```blade
<!--SECTION OF FORM  -->

@if(session('flash'))
    <p style="color: #ff80aa">{{ session('flash')}}</p>
@endif
    <h1>CONTACT MINGKAI</h1>
    <p>If you want to pet me or have a little chat, just reach out to Mingkai!!!</p>
        <form class="contact" action="/contact" method="post">
            @csrf
            <div class="half left cf">
                <input type="text" id="input-name" placeholder="Name" name="name" required>
                <input type="email" id="input-email" placeholder="Email Address" name="email">
                <input type="text" id="input-subject" placeholder="Subject" name="subject" required>
                <textarea name="message" type="text" id="input-message" placeholder="Message" name="message" required></textarea>
                <input type="submit" class="submit-button" value="Submit" id="input-submit">
            </div>
            <div class="image-container right">
                <img src="/css/assets/images/contactmingkai.png" alt="Contact Mingkai">
            </div>
        </form>
<!--END OF SECTION OF FORM  -->
```

```blade
<body>
    <!--SECTION OF FORM  -->
        <p><b>Name: </b>{{ $dataReceived['name'] }}</p>
        <p><b>Email: </b>{{ $dataReceived['email'] }}</p>
        <p><b>Subject: </b>{{ $dataReceived['subject'] }}</p>
        <p><b>Message: </b>{{ $dataReceived['message'] }}</p>
    <!--END OF SECTION OF FORM  -->
</body>
```

web.php M | ContactMe.php

app > Mail > ContactMe.php > ContactMe > __construct

```php
<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class ContactMe extends Mailable
{
    use Queueable, SerializesModels;

    public $dataReceived;

    /**
     * Create a new message instance.
     *
     * @param array $data
     */
    public function __construct($data)
    {
        $this->dataReceived = $data;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->view('emailTemplate');
    }
}
```

ROUTE:

```php
// Contact form route
Route::post('/contact', function () {
    $data = request()->all();
    Mail::to('mingkai103019@gmail.com')->send(new ContactMe($data));
    return redirect('/contact')->with('flash', 'Message Sent Successfully');
});
```

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

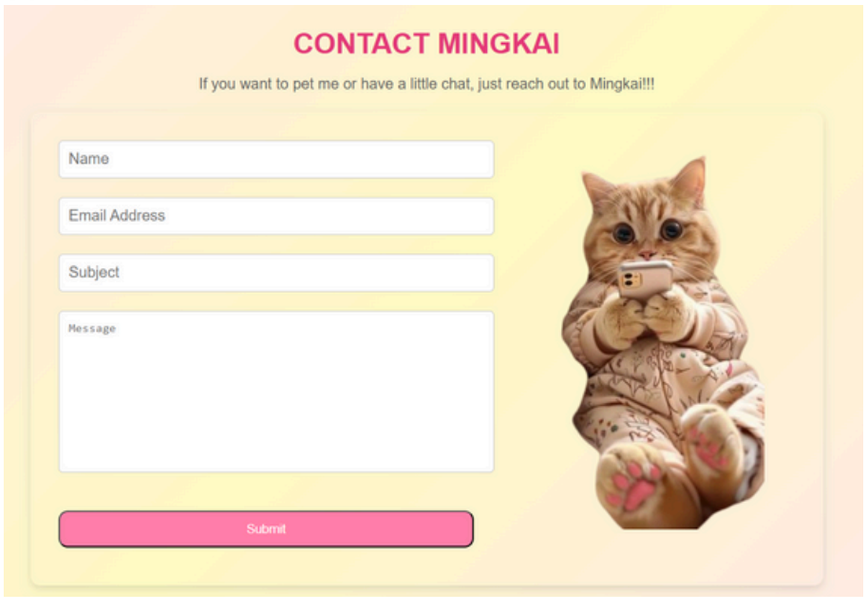
JAVASCRIPT:

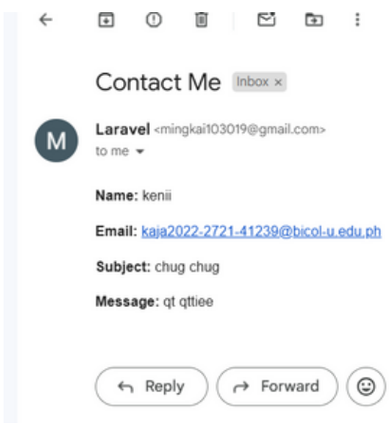
```javascript
<!-- JAVASCRIPT for (Contact form) -->
<script>

    document.getElementById('contact-form').addEventListener('submit', function(event) {
        var name = document.querySelector('input[name="name"]').value;
        var email = document.querySelector('input[name="email"]').value;
        var subject = document.querySelector('input[name="subject"]').value;
        var message = document.querySelector('input[name="message"]').value;

        if (!name || !email || !subject || !message) {
            alert('Please fill out all required fields.');
            event.preventDefault(); // Prevent form submission
        }
    });
</script>
```

.env:

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=mingkai103019@gmail.com
MAIL_PASSWORD=iyluffhncjvcomzx
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=mingkai103019@gmail.com
MAIL_FROM_NAME="${APP_NAME}"
```

OUTPUT:





EXPLANATION:

- *Form UI: A form for users to enter their name, email, subject, and message, with a cute theme featuring Mingkai. After submitting the form, users are redirected back to the contact page with a flash message displayed if the submission is successful.*
- *Flash Message: If a message is successfully sent, a flash message ("Message Sent Successfully") appears in pink.*
- *Form Validation: JavaScript is used to ensure that all required fields are filled out before the form is submitted. If any required field is missing, an alert will be shown.*
- *Mailable Class: The ContactMe.php file defines a ContactMe Mailable class, which prepares the email to be sent, passing along the user's data.*
- *Email Sending: When the form is submitted, Laravel processes the data and sends an email to mingkai103019@gmail.com with the form details using the Mail facade.*
- *Email Template: The form fields (name, email, subject, message) are displayed in the email using an email template called emailTemplate.*

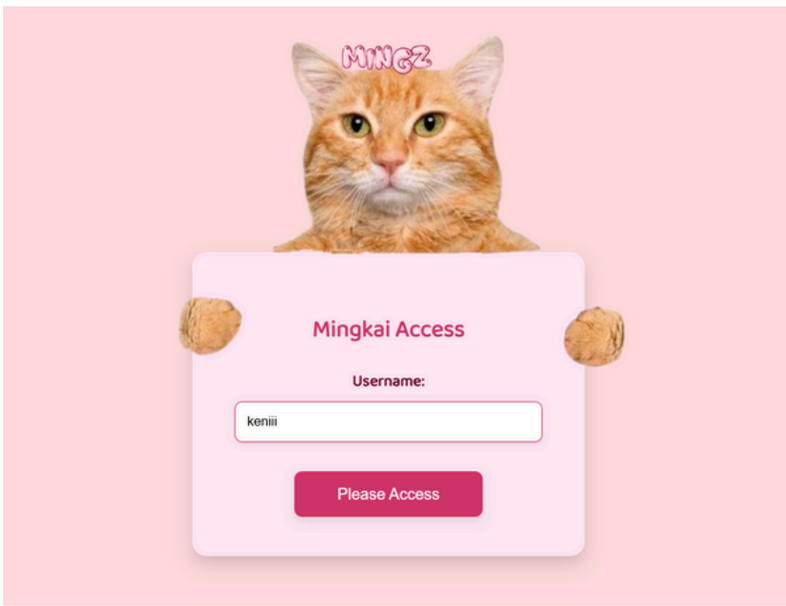
PART 2 : USING ROUTE PARAMETERS

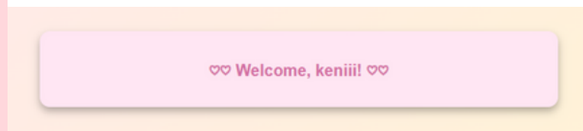
1. **DEFINE A ROUTE WITH A REQUIRED PARAMETER:**
   - **Create a route that accepts a username parameter and displays a welcome message that includes the username.**

CODE (Route):

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

OUTPUT:





EXPLANATION:

- *Route::get('/user/{username?}'):*

-This is a GET route, meaning it responds to GET requests (like when a user visits a page in their browser).

-The {username?} part is a dynamic parameter, where username can be passed in the URL. The ? indicates that this parameter is optional, so /user and /user/{username} both work.

- *[LoginController::class, 'userProfile']:*

- This specifies that when someone visits this route, the userProfile method inside the LoginController will be executed to handle the request.

- *->name('user.profile'):*

-This names the route as user.profile, making it easier to reference elsewhere in your code (like in redirects or generating URLs).

2. 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.**

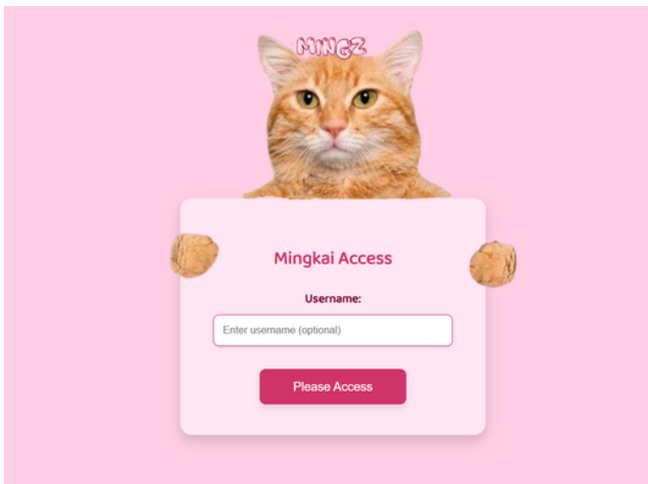
CODE (Route):

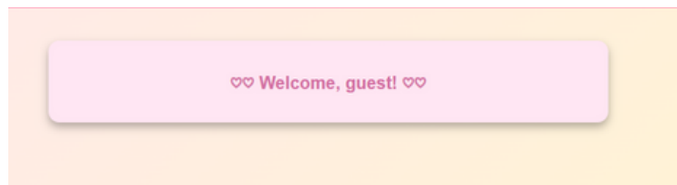
```php
Route::get('/user/{username?}',
    [LoginController::class, 'userProfile'])
    ->name('user.profile')
    ->where('username?', '[A-Za-z]'); // Constraint: only alphabetic characters
```

CODE (Controller):

```php
class LoginController extends Controller
{
    public function login(Request $request)
    {
        // Validate the request (allowing optional username)
        $request->validate([
            'username' => 'nullable|alpha', // Ensure username contains only alphabetic characters
        ]);

        // Get the username from the request, default to 'guest' if not provided
        $username = $request->input('username');
        if (empty($username)) {
            $username = 'guest'; // Default to 'guest' if username is empty
        }

        // 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]);
    }

    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');

        // Pass the username to the homepage view
        return view('homepage', ['message' => "Welcome, $username!"]);
    }
```

OUTPUT:





EXPLANATION:

- *User Accesses /user or /user/{username}:*
  - The user can visit /user or /user/{username}, which triggers the userProfile method in LoginController.
- *Login Method:*
  - If the user submits a form (handled by the login() method), it checks if a username is provided. If not, it defaults to 'guest'. The username is then stored in the session for future use.
- *Redirect to Homepage:*
  - After logging in or submitting the username, the user is redirected to the homepage route with the username parameter.
- *Homepage Method:*
  - The homepage() method checks if a username was passed through the URL or retrieves it from the session. It then displays a personalized message on the homepage, either welcoming the specific user or greeting them as "guest."

3. APPLY REGULAR EXPRESSION CONSTRAINTS TO THE ROUTE PARAMETERS:

- **Ensure that the username only accepts alphabetic characters (a-z, A-Z)**

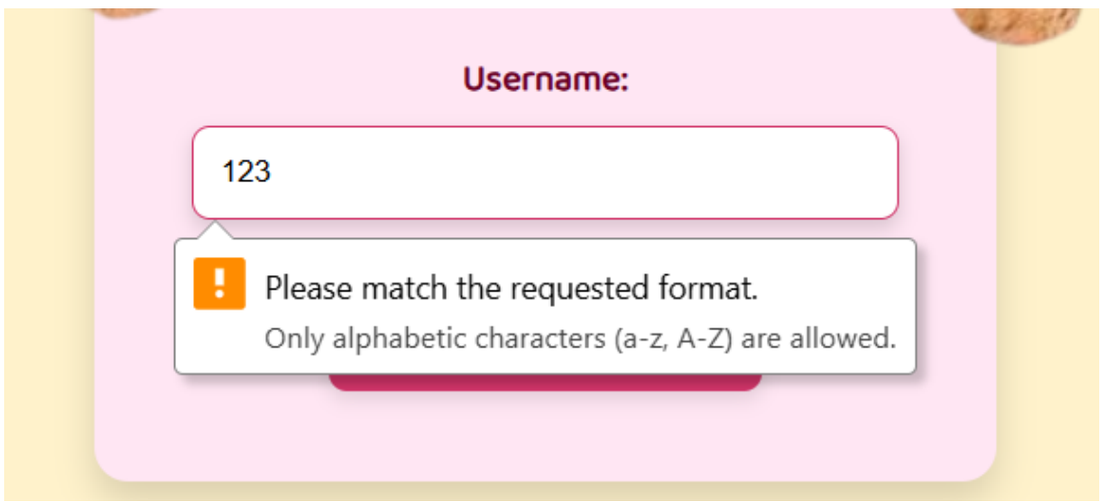
CODE (Route):

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

CODE (Controller):

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

OUTPUT:



EXPLANATION:

- *->where('username', '[A-Za-z]+'):*
  - This is a route constraint that uses a regular expression to define what values are allowed for the username parameter.
- *[A-Za-z]+:*
  - [A-Za-z] means the route will only accept alphabetic characters (both uppercase and lowercase).
  - + ensures that at least one character is required if the username is provided.
- *As a result, the route will only match URLs where the username consists of alphabetic characters, such as /user/keniii or /user/Asarah. It will not match /user/123 or /user/ken_anne.*
- *'nullable|alpha'*
  - The method starts by validating the incoming request to ensure that the username, if provided, only contains alphabetic characters. The 'nullable|alpha' rule means that the username field can be either empty or alphabetic.