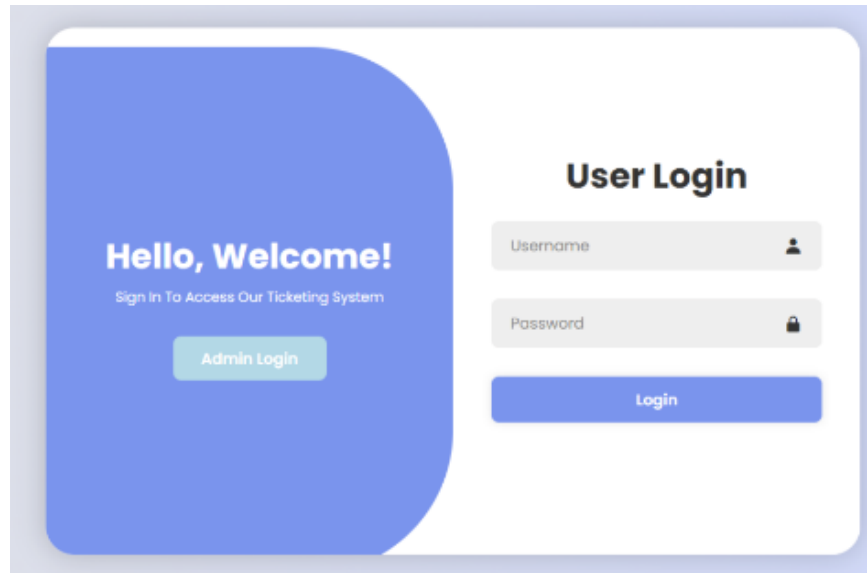1. Login View
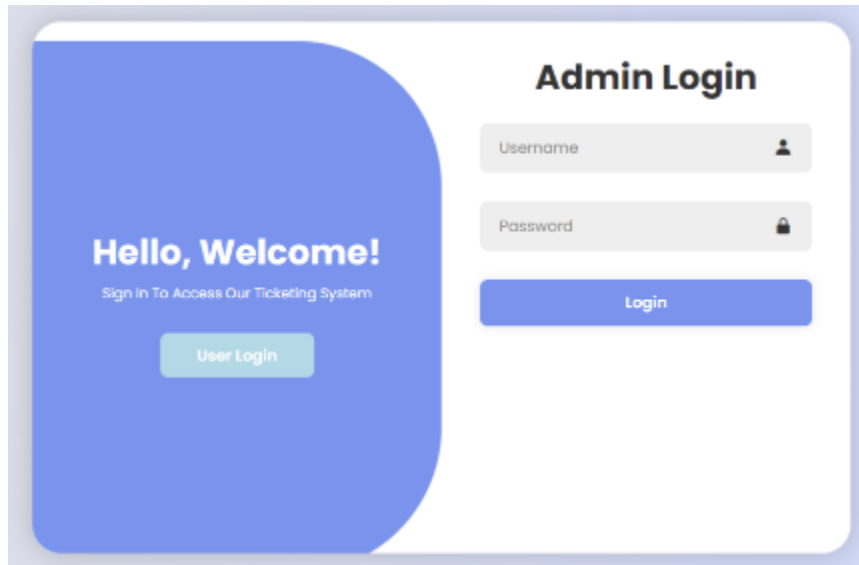   - A user interface with simple design that lets the user input their user and pass so they can access the website.

2. Admin View
   - The same as the login view but it's for the admin of the website to access and manage the requests made by the user/s.

3. AuthController
   - The code here serves as a login function so that it will only accept the given parameters that we inputted for the login of both user and the admin.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AuthController extends Controller
{
    // Show the login form
    public function showLoginForm()
    {
        return view('login');
    }

    // Handle User login
    public function loginUser(Request $request)
    {
        $credentials = $request->only('username', 'password');

        if ($credentials['username'] === 'user' && $credentials['password'] === 'userpassword') {
            session(['user_type' => 'user']);
            return redirect()->route('create-ticket');
        }

        return redirect()->back()->with('error', 'Invalid username or password');
    }

    // Handle Admin login
    public function loginAdmin(Request $request)
    {
        $credentials = $request->only('username', 'password');

        if ($credentials['username'] === 'admin' && $credentials['password'] === 'adminpassword') {
            session(['user_type' => 'admin']);
            return redirect()->route('view-ticket');
        }
```

4. TicketController
   - This code handles the tickets that are submitted by the users.

AuthController.php   TicketController.php ×

C: > Users > User > Desktop > webdev proj > Controller > app > Http > Controllers > TicketController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TicketController extends Controller
{
    public function create()
    {
        return view('create-ticket');
    }

    public function store(Request $request)
    {
        // Logic to store the ticket data
        // For example: Ticket::create($request->all());
        return redirect()->route('create-ticket')->with('success', 'Ticket created successfully.');
    }
}
```

5. CheckUserType
   - This middleware checks whether the user is authenticated before it allows it to proceed to the desired destination.

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class CheckUserType
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @param  string  $type
     * @return mixed
     */
    public function handle(Request $request, Closure $next, $type)
    {
        if (session('user_type') !== $type) {
            return redirect()->route('login');
        }
        return $next($request);
    }
}
```

6. CheckUserRole
   - This checks if the user has a role assigned to it and redirects it to the views that only their role is allowed to access.

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class CheckUserRole
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @param  string  $role
     * @return mixed
     */
    public function handle(Request $request, Closure $next, $role)
    {
        // Check if the session has a user type and if it matches the required role
        if ($request->session()->get('user_type') !== $role) {
            return redirect()->route('login')->withErrors(['Unauthorized access']);
        }

        return $next($request);
    }
}
```

7. App styles
   - This gives the home view where the ticket can be created and viewed styles and designs.

8. Login styles
   - This gives the Login view the design and style so that it is easier to interact with and pleasing to the eye.