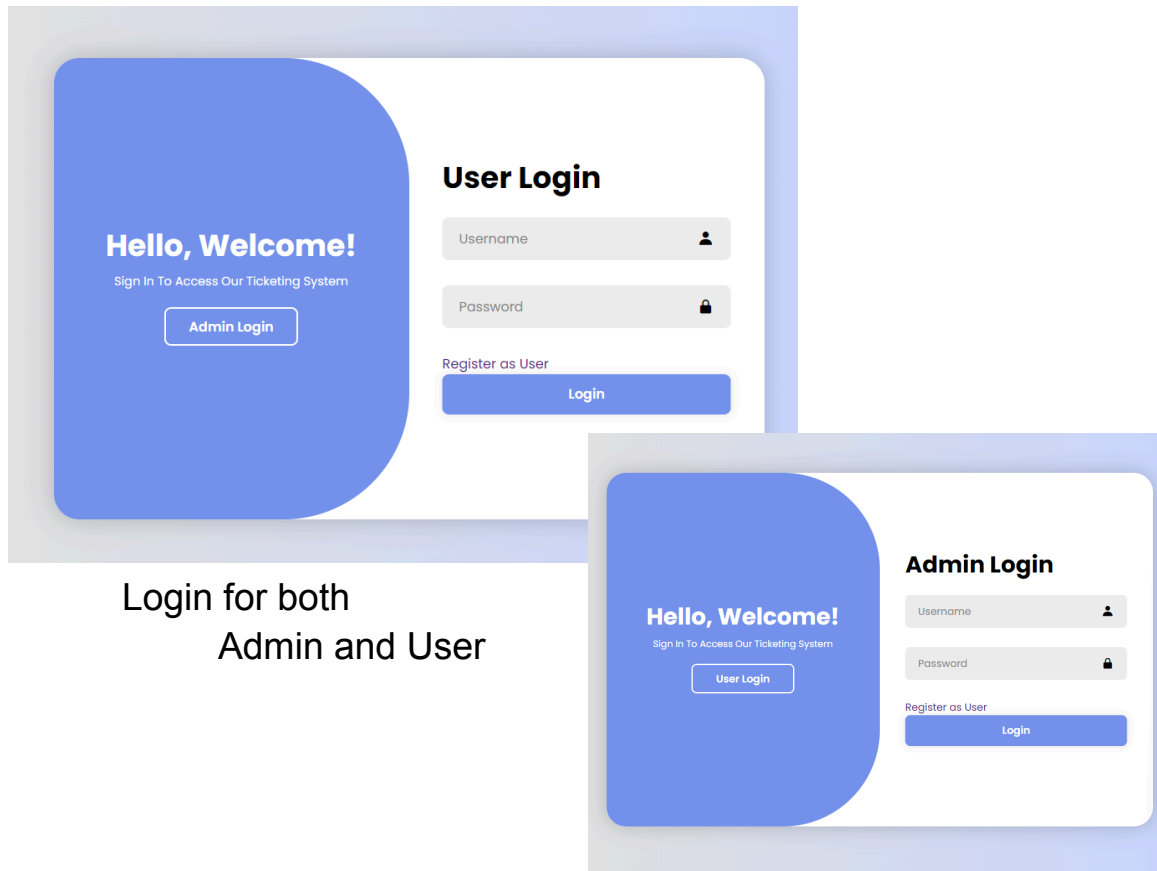
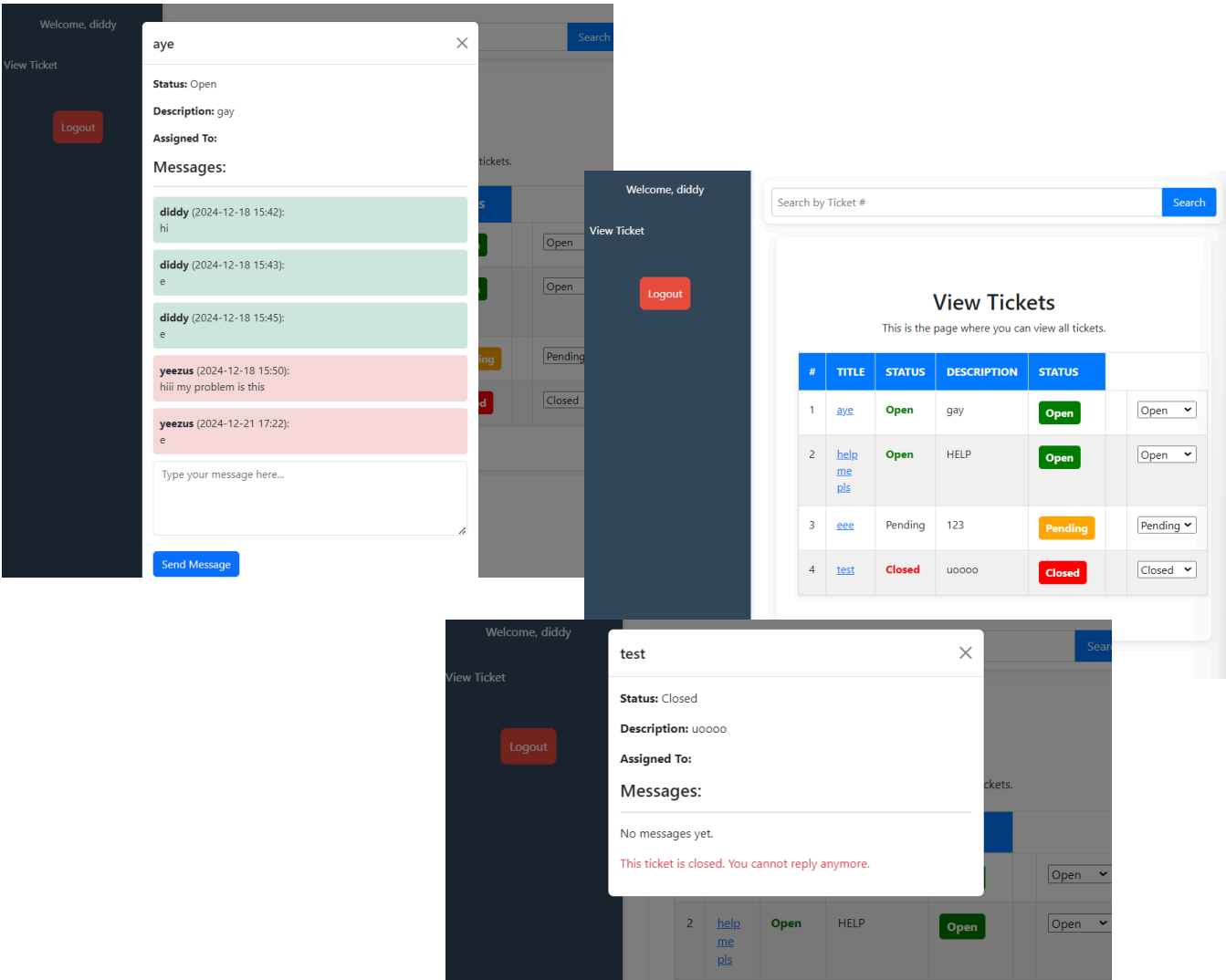


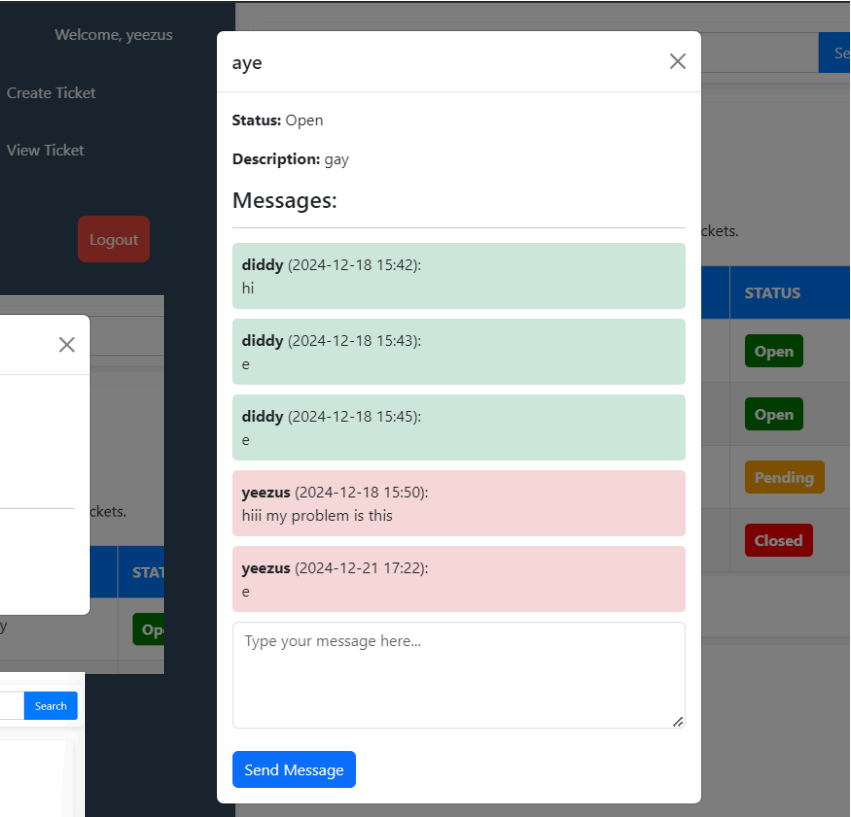
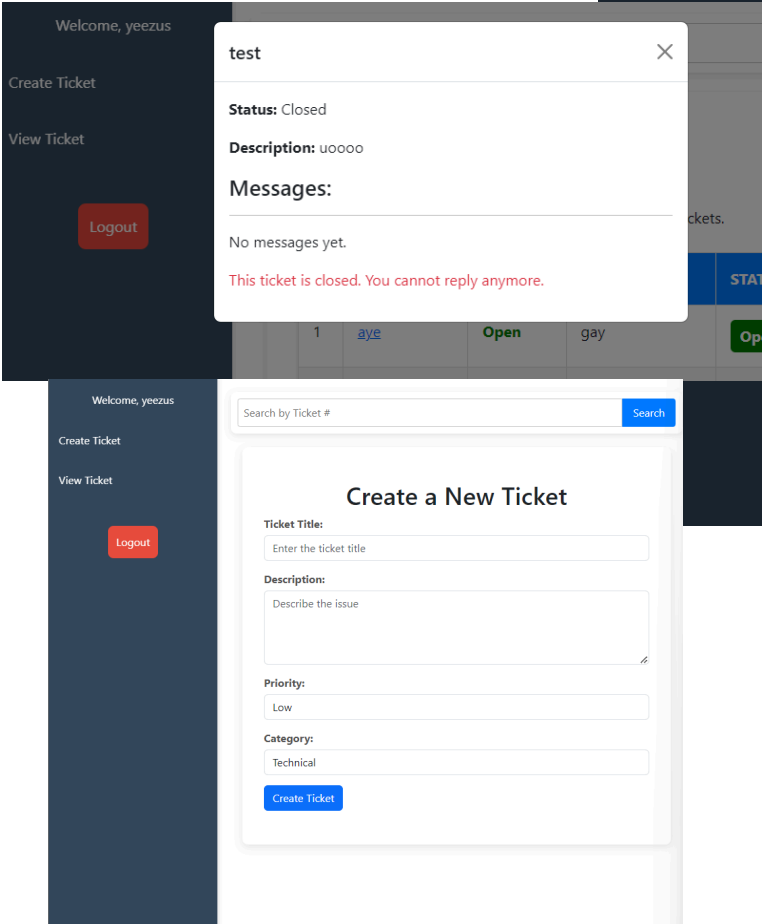
1. Front End



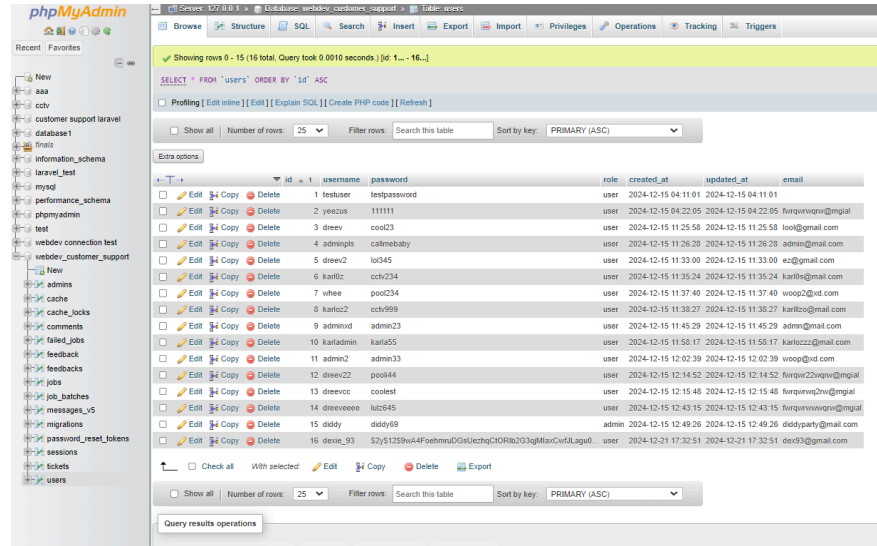
Admin Side



User Side



2. Back End



Showing rows 0 - 15 (16 total, Query took 0.0010 seconds) [id 1... - 16...]

SELECT * FROM `users` ORDER BY `id` ASC

Number of rows: 25 Filter rows: Search this table Sort by key: PRIMARY (ASC)

		id	username	password	role	created_at	updated_at	email
<input type="checkbox"/>	Edit	1	testuser	testpassword	user	2024-12-15 04:11:01	2024-12-15 04:11:01	
<input type="checkbox"/>	Edit	2	yeecus	111111	user	2024-12-15 04:22:05	2024-12-15 04:22:05	fonquvovr@gmail
<input type="checkbox"/>	Edit	3	dreev	cosi23	user	2024-12-15 11:25:58	2024-12-15 11:25:58	look@gmail.com
<input type="checkbox"/>	Edit	4	admipls	calmebaby	user	2024-12-15 11:26:28	2024-12-15 11:26:28	adren@gmail.com
<input type="checkbox"/>	Edit	5	dreev2	lol345	user	2024-12-15 11:33:00	2024-12-15 11:33:00	ez@gmail.com
<input type="checkbox"/>	Edit	6	karficz	ctch234	user	2024-12-15 11:35:24	2024-12-15 11:35:24	karficz@gmail.com
<input type="checkbox"/>	Edit	7	whee	pool234	user	2024-12-15 11:37:40	2024-12-15 11:37:40	wuop@gmail.com
<input type="checkbox"/>	Edit	8	karficz2	ctch999	user	2024-12-15 11:38:27	2024-12-15 11:38:27	karficz@gmail.com
<input type="checkbox"/>	Edit	9	admi2d	admi23	user	2024-12-15 11:45:29	2024-12-15 11:45:29	admi@gmail.com
<input type="checkbox"/>	Edit	10	karfadmin	karf555	user	2024-12-15 11:58:17	2024-12-15 11:58:17	karficz@gmail.com
<input type="checkbox"/>	Edit	11	admi2n	admi33	user	2024-12-15 12:02:39	2024-12-15 12:02:39	wuop@gmail.com
<input type="checkbox"/>	Edit	12	dreev22	pool844	user	2024-12-15 12:14:52	2024-12-15 12:14:52	fonquvovr@gmail
<input type="checkbox"/>	Edit	13	dreevcs	coolst	user	2024-12-15 12:15:48	2024-12-15 12:15:48	fonquvovr@gmail
<input type="checkbox"/>	Edit	14	dreevcees	ku2645	user	2024-12-15 12:43:15	2024-12-15 12:43:15	fonquvovr@gmail
<input type="checkbox"/>	Edit	15	diddy	diddy69	admin	2024-12-15 12:49:26	2024-12-15 12:49:26	diddy@gmail.com
<input type="checkbox"/>	Edit	16	dreevcs_93	52y\$1239uA4foetmuD9aUczhqCfORib203qMftrCwKtLagu8	user	2024-12-21 17:32:51	2024-12-21 17:32:51	diddy@gmail.com

Register function (authcontroller.php) for both admin and users

```
4 }
5
6 // Handle User registration
7 public function register(Request $request)
8 {
9     // Validate input
10     $validated = $request->validate([
11         'username' => 'required|string|unique:users,username',
12         'email' => 'required|email|unique:users,email',
13         'password' => 'required|confirmed|min:6',
14         'role' => 'required|string',
15     ]);
16
17     // Hash password
18     $validated['password'] = ($validated['password']);
19
20     // Save user
21     $user = User::create($validated);
22
23     // Check if the user was successfully created
24     if ($user->wasRecentlyCreated) {
25         return redirect()->route('login')->with('success', 'User registered successfully!');
26     }
27
28     return back()->withErrors(['registration' => 'Failed to register the user.']);
29 }
30 }
```

Admin login and logout (login is based on roles on the usertype which is saved in the db)

```
// Handle Admin login
public function loginAdmin(Request $request)
{
    // Validate the incoming request data
    $validated = $request->validate([
        'username' => 'required',
        'password' => 'required',
    ]);

    // Fetch the user from the database
    $user = User::where('username', $validated['username'])->first();

    // Check if the user exists and the role is 'admin'
    if (!$user || $user->role !== 'admin') {
        return redirect()->back()->with('error', 'Unauthorized access. Admins only.');
```

```
    }

    // Check if the password matches
    if ($user->password !== $validated['password']) {
        return redirect()->back()->with('error', 'Invalid credentials.');
```

```
    }

    // Store user information in the session
    session(['user_id' => $user->id, 'user_type' => $user->role, 'username' => $user->username]);

    // Redirect the admin to the ticketing system
    return redirect()->route('view-ticket')->with('success', 'Welcome, Admin!');
```

```
}

// Handle logout
public function logout()
{
    Log::info('User logging out, clearing session data');
    session()->flush(); // Clear session data

    // Log session data after logout
    Log::info('Session data after logout: ', session()->all());

    return redirect()->route('login'); // Redirect to login page
}
```

```
// Handle User login
public function loginUser(Request $request)
{
    $validated = $request->validate([
        'username' => 'required',
        'password' => 'required',
    ]);

    // Fetch user from the database
    $user = User::where('username', $validated['username'])->first();

    // Check if the user exists and the password matches
    if (!$user || $user->password !== $validated['password']) {
        return back()->with('error', 'Invalid credentials.');
```

```
    }

    // Store user information in the session
    session(['user_id' => $user->id, 'user_type' => $user->role, 'username' => $user->username]);

    // Log session data after login
    Log::info('User logged in, session data: ', session()->all());

    // Redirect user based on their role (user/admin)
    return redirect()->route('view-ticket');
```

Login for user

Addmessage function (ticketcontroller.php) and show ticket function (ticketcontroller.php)

- Had a hard time implementing this, but it indicates and it's just a functionality of messages between user and admin

```
}  
public function addMessage(Request $request, $ticketId)  
{  
    $request->validate([  
        'message' => 'required|string|max:1000',  
    ]);  
  
    // Store the new message  
    $message = new Message();  
    $message->ticket_id = $ticketId;  
    $message->user_id = session('user_id'); // assuming the user ID is in the session  
    $message->message = $request->message;  
    $message->save();  
  
    return back()->with('success', 'Message sent successfully!');  
}  
  
// Method to show ticket details with related messages  
public function showTicket($ticketId)  
{  
    // Eager load messages with the associated user  
    $ticket = Ticket::with('messages.user')->find($ticketId);  
  
    if (!$ticket) {  
        return redirect()->route('tickets.index')->with('error', 'Ticket not found.');    }  
  
    return view('ticket.show', compact('ticket'));  
}  
  
// Method to show a ticket (admin or user) without comments  
public function show($ticketId)  
{  
    // Find the ticket by ID  
    $ticket = Ticket::findOrFail($ticketId);  
  
    // Optionally, load feedbacks or related information  
    $feedbacks = $ticket->feedbacks;  
  
    return view('ticket.details', compact('ticket', 'feedbacks'));  
}
```

Search and store ticket (ticketcontroller.php)

- It stores ticket and searches by keywords, number, description

```
// Store a new ticket (for users)
public function store(Request $request)
{
    // Validation (optional, but recommended)
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:1000',
    ]);

    // Create and store the new ticket
    $ticket = new Ticket();
    $ticket->title = $request->title;
    $ticket->status = 'open'; // Default status
    $ticket->description = $request->description;
    $ticket->user_id = session('user_id'); // Associate with logged-in user
    $ticket->save();

    return redirect()->route('view-ticket');
}

// Method to search for tickets based on the title or description
public function search(Request $request)
{
    $searchTerm = $request->input('search');

    // Search for tickets by title, description, or other attributes, eager
    $tickets = Ticket::with('user') // Eager load user relationship
        ->where('title', 'like', "%$searchTerm%")
        ->orWhere('description', 'like', "%$searchTerm%")
        ->get();

    return view('view-ticket', ['tickets' => $tickets]);
}
```

Update status (ticketcontroller.php)

```
// Method to update the status of a ticket (admin only)
public function updateStatus(Request $request, $id)
{
    // Validate the incoming request
    $request->validate([
        'status' => 'required|in:open,pending,closed',
    ]);

    // Find the ticket by ID
    $ticket = Ticket::findOrFail($id);

    // Update the status
    $ticket->status = $request->input('status');
    $ticket->save();
    Log::info('Test log entry');

    // Redirect back with a success message
    return back()->with('success', 'Ticket status updated successfully!');
}
```

In admin side they can change the status of ticket, if ticket is closed no one can reply anymore