Login Validation

This login authorization is performed in PHP, it is activated after the login credentials have been validated in javascript.

It utiulizes the password_verify() function to determine if the hashed password matches the submitted password for the submitted username.

- If SUCCESFUL: The user will be logged in to the site, and their credntials will be stored as $_SESSION variables.

- IF UNSUCCESFUL: The user will be redirected back to the login page, with a simple error message.

```php
//If the SQL returns a row, then the user exists.
if($result->num_rows > 0)
{
    $row = $result->fetch_assoc();

    // Compares submitted password against hashed password in database.
    // If correct, start a PHP session and set SESSION variables. (log the user in)
    if(password_verify($password, $row['password']))
    {
        //Destroys any (no longer required) stored sessions before startign a new one.
        session_destroy();
        session_start();
        $_SESSION['user_id'] = $row['userId'];
        $_SESSION['user_privilege'] = $row['privilege'];
        $_SESSION['user_name'] = $row['username'];
        $_SESSION['user_email'] = $row['email'];
        $_SESSION['user_fname'] = $row['firstName'];
        $_SESSION['user_lname'] = $row['lastName'];
        //Returns logged-in user to the index page.
        header("Location: ../pages/index.php");
    }
    else
    {
        // Return to login page with error message to be displayed in its dedicated error <div></div>.
        header("Location: ../pages/login.php?error=Your login details are incorrect. Please try again.");
    }

}
else
{
    // Return to login page with error message to be displayed in its dedicated error <div></div>.
    header("Location: ../pages/login.php?error=Your login details are incorrect. Please try again.");
}
```

Register Email Validation

This is a validation form that utilizes a REGEX pattern to check the validity of a submitted email address during registration.

It also verifies that the submitted email address is not null or empty.

The user is met with two visual prompts to help navigate any errors after submission:

1) The user is met with an error message above the field that is in question.

2) The field in question is highlighted red if invalid, or green if valid.

```
// Email validation
if(email.value == "" || email.value == null)
{
    // If the email is left empty, an error message will be displayed.
    emailErrorMsg.innerText = "Email cannot be left empty.";
    email.style.borderColor = "red";
}
else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email.value))
{
    // Uses a REGEX pattern to check for essential characters in an e-mail address.
    emailErrorMsg.innerText = "Please enter a valid email address.";
    email.style.borderColor = "red";
}
else
{
    // If the email is valid, the border color will turn green.
    email.style.borderColor = "green";
}
```

Register Password Validation

This script is fairly straightforward and utilizes the same structure as the previous.

It checks the submitted passwords for various conditions, namely the length requirements, and communicates any invalid fields.

```
// Password validation
if (password2.value !== password1.value)
{
    passwordErrorMsg.innerText = "Passwords do not match. Please try again.";
    password1.style.borderColor = "red";
    password2.style.borderColor = "red";
}
else if(password1.value == "" || password1.value == null || password2.value == "" || password2.value == null)
{
    passwordErrorMsg.innerText = "Password cannot be left empty.";
    password1.style.borderColor = "red";
    password2.style.borderColor = "red";
}
else if(password1.value.length >= 1 && password1.value.length <= 6)
{
    passwordErrorMsg.innerText = "Password must be longer than 6 characters.";
    password1.style.borderColor = "red";
    password2.style.borderColor = "red";
}
else if(password1.value.length >= 16)
{
    passwordErrorMsg.innerText = "Password must be shorter than 16 characters.";
    password1.style.borderColor = "red";
    password2.style.borderColor = "red";
}
else
{
    password1.style.borderColor = "green";
    password2.style.borderColor = "green";
}
```

Update Password Validation

This is the full document for updating a password within the database.

This screenshot showcases how we assign our valriables, as well as the event listener and prevention of form submission.

Users receive error messages in the same two visual ways as mentioned above.

The field border turns red, and the error message appears above it.

```
// Create all variables for field values.
const form = document.getElementById("change-password-form");
const old_password = document.getElementById("old_password");
const new_password1 = document.getElementById("new_password1");
const new_password2 = document.getElementById("new_password2");
const passError = document.getElementById("passError");

// Validate all field values and display error messages when necessary.
form.addEventListener("submit", (e) =>
{
    e.preventDefault(); // prevent form from submitting by default

    // Clear previous error messages
    passError.innerText = "";

    // Password validation
```

```
    if (new_password2.value !== new_password1.value)
    {
        passError.innerText = "New passwords do not match. Please try again.";
        new_password1.style.borderColor = "red";
        new_password2.style.borderColor = "red";
    }
    else if(old_password.value == "" || old_password.value == null
        || new_password1.value == "" || new_password1.value == null
        || new_password2.value == "" || new_password2.value == null)
    {
        passError.innerText = "Fields cannot be left empty.";
        old_password.style.borderColor = "yellow";
        new_password1.style.borderColor = "yellow";
        new_password2.style.borderColor = "yellow";
    }
    else if(new_password2.value.length >= 1 && new_password2.value.length <= 6)
    {
        passError.innerText = "New password must be longer than 6 characters.";
        new_password2.style.borderColor = "red";
        new_password2.style.borderColor = "red";
    }
    else if(new_password2.value.length >= 16)
    {
        passError.innerText = "New password must be shorter than 16 characters.";
        new_password2.style.borderColor = "red";
        new_password2.style.borderColor = "red";
    }
    else
    {
        old_password.style.borderColor = "green";
        new_password1.style.borderColor = "green";
        new_password2.style.borderColor = "green";
    }

    // Only submit the form when there are no validation errors
    if (!passError.innerText)
    {
        if(!confirm("Are you sure you want to change your password?"))
        {
            e.preventDefault();
        }
        form.submit();
    }
});
```

Update Profile Pic Validation

This is the PHP script that handles the upload of a new profile picture. It checks two conditions:

1) Checks the file size of the image, if it exceeds 125 kilobytes, it won't be accepted, and an error message will display above the submission field.

2) Determines the file extension of the photo against a pre-constructed array of accepted file extensions (.png, .jpg, .gif). If it is not an acceptable file type, an error message will be displayed above the submission field.

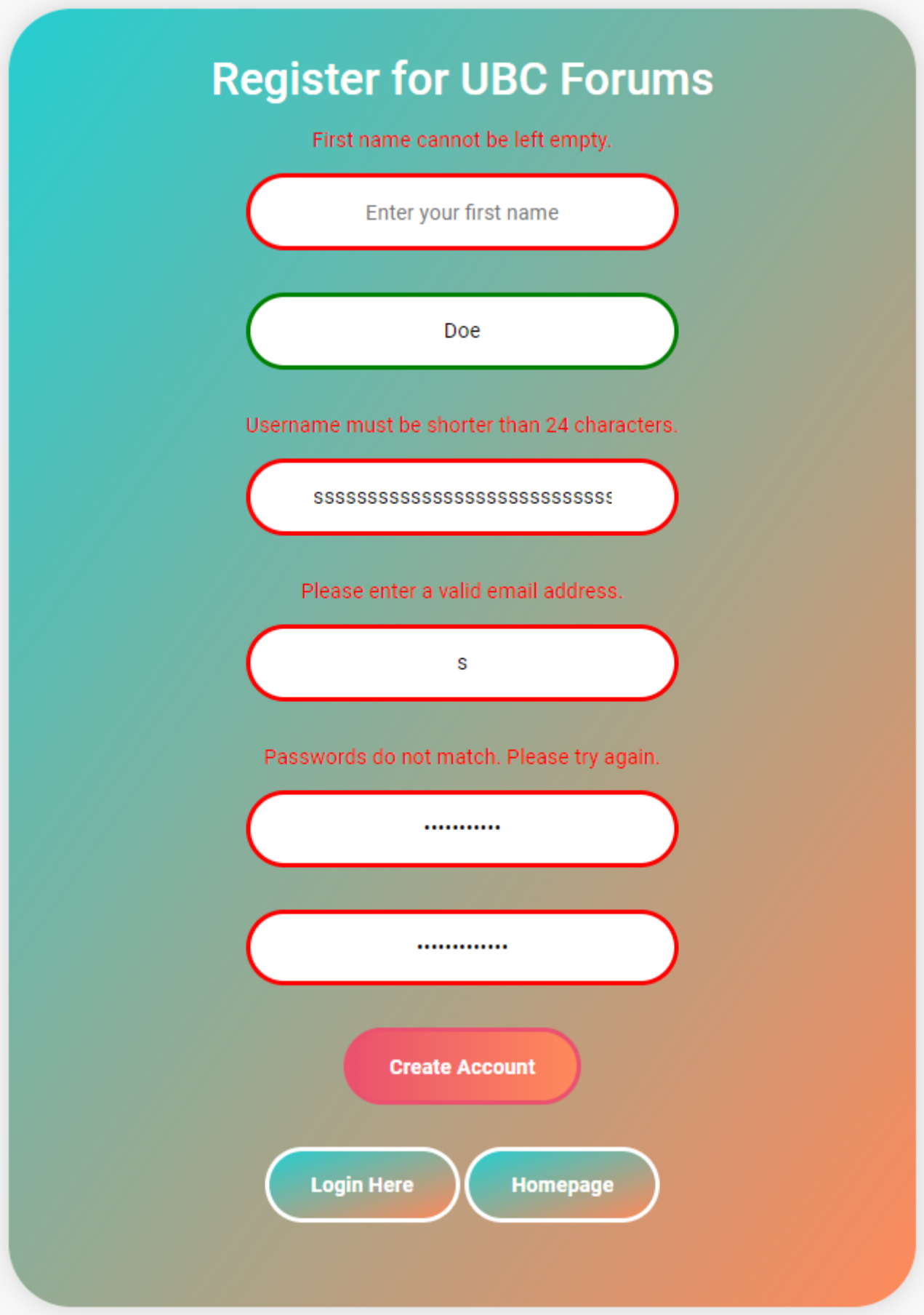
```php
// Checks if the file is too large to be uploaded.
if($pic_size > 125000)
{
    header("Location: ../pages/account.php?picError=Your file is too large. Please try again.");
    $conn->close();
}
// File is an appropriate size
else
{
    // Checks if the file is a valid image type by extracting its file extension.
    $pic_ext = pathinfo($pic_name, PATHINFO_EXTENSION);
    $pic_ext_lc = strtolower($pic_ext);
    $valid_exts = array("jpg", "jpeg", "png");

    if(in_array($pic_ext_lc, $valid_exts))
    {
        // Generates a unique name for the new profile picture and stores it on the server.
        $new_pic_name = uniqid("img-", true) . "." . $pic_ext_lc;
        $pic_path = "../images/" . $new_pic_name;
        move_uploaded_file($tmp_name, $pic_path);
```

Visual Representations of Error Handling

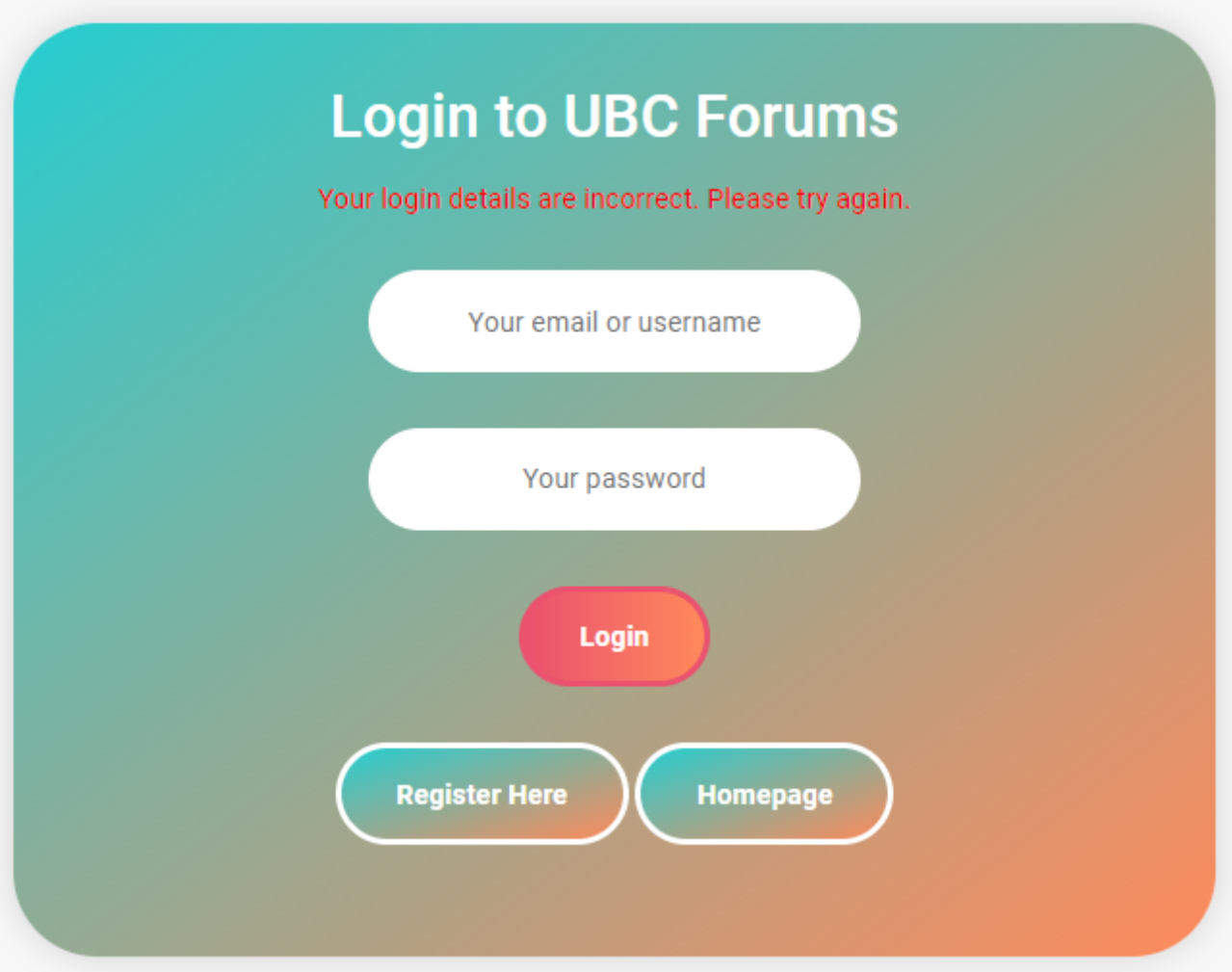
Below we have attached a collection of screenshots showing off some of the common errors that users may run into as they learn the flow of the website.

We have also tried to showcase how we have handled other requests: such as having sql queries return empty-handed, and critical interactions requiring browser confirmations.

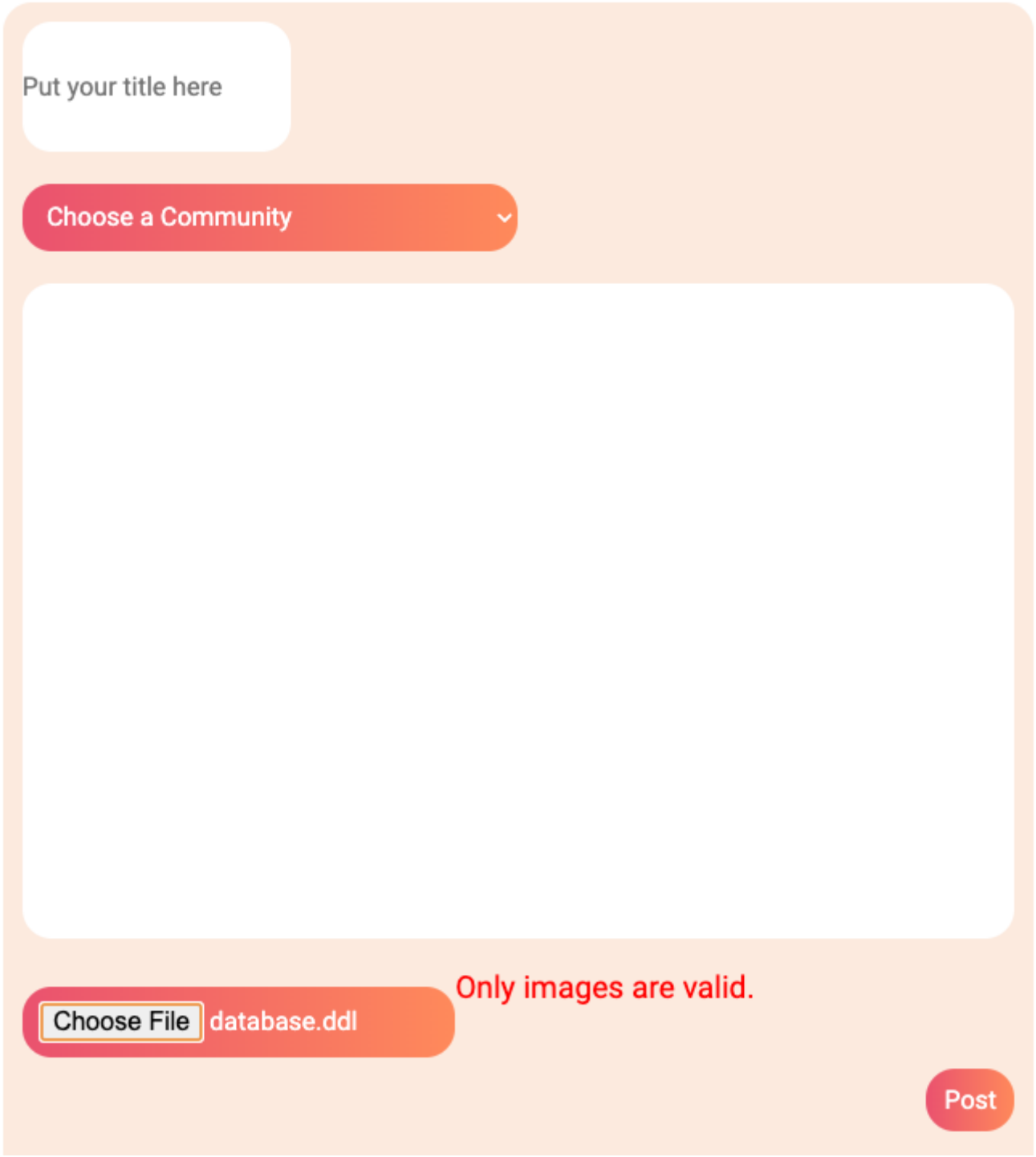
Registration Error Handling



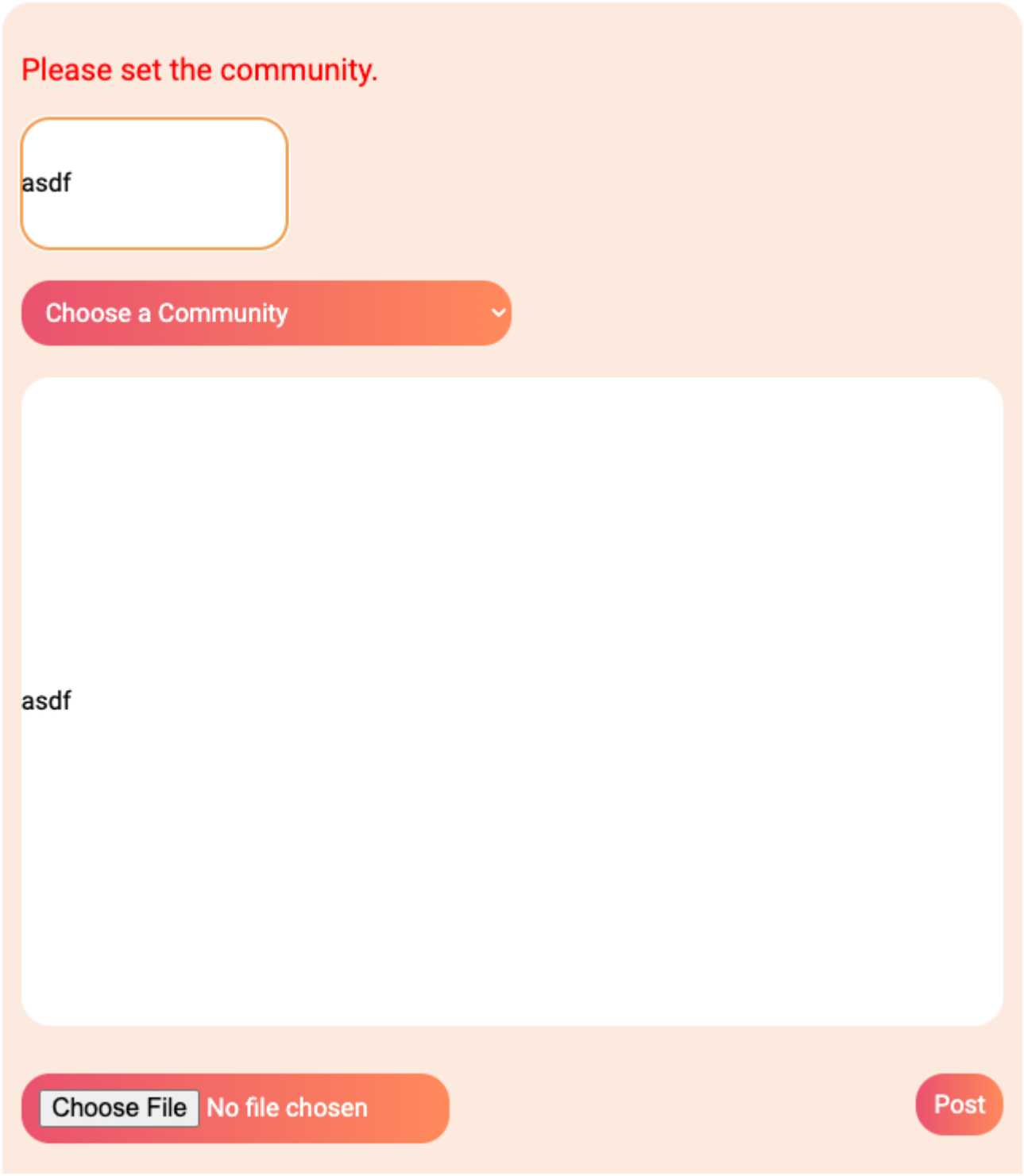
Login Error Handling



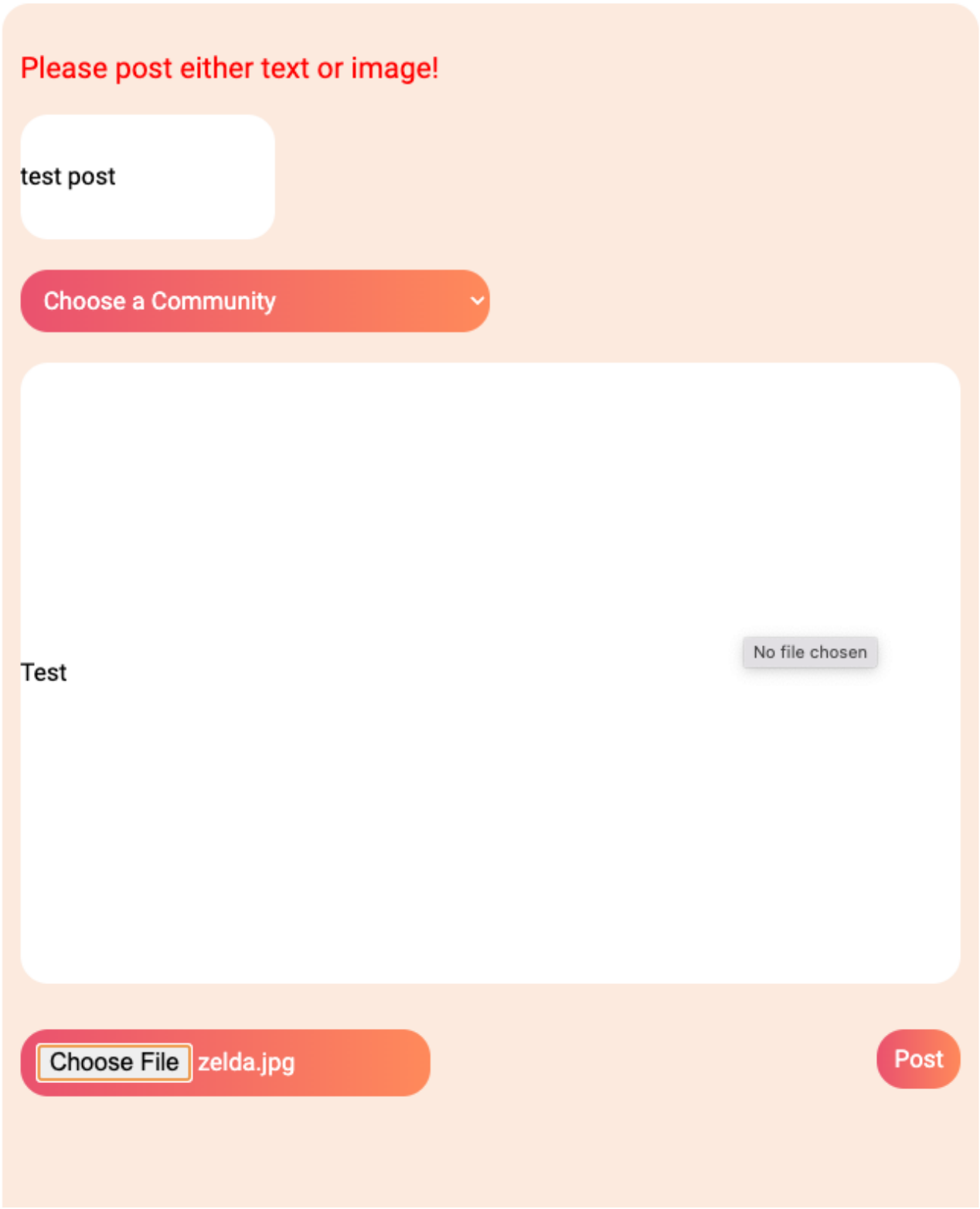
Invalid Image Validation



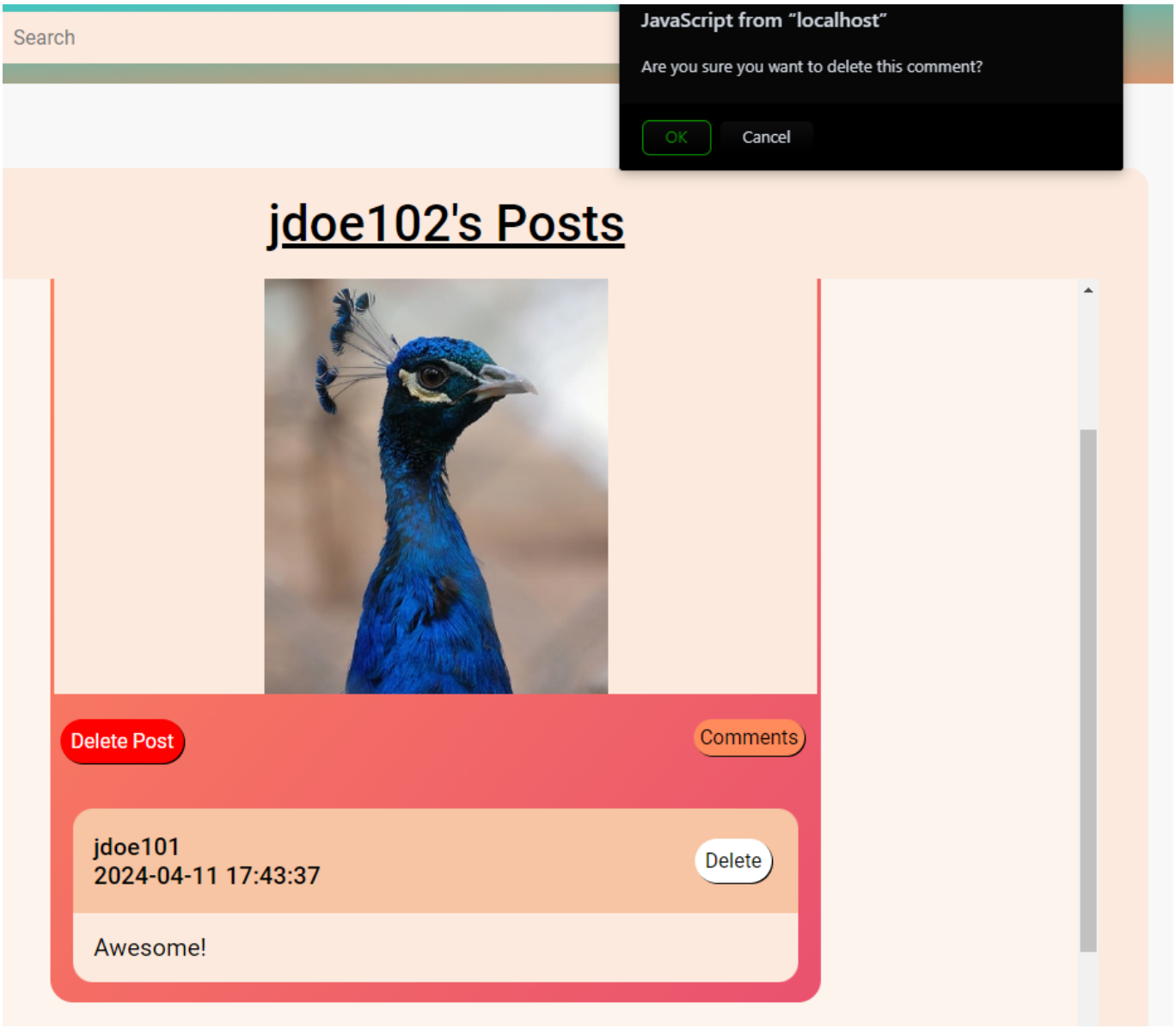
Proper Post form validation



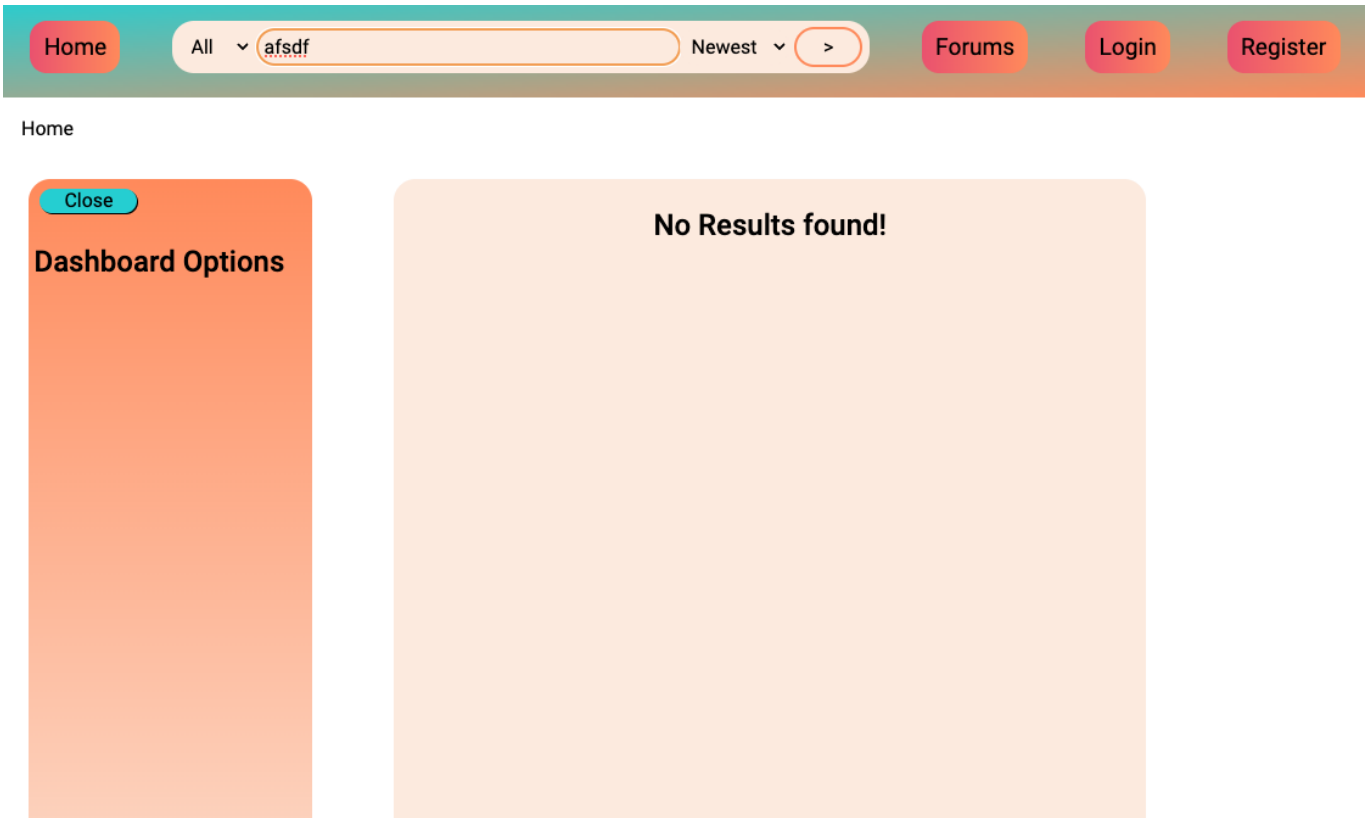
Cant post both text and image Validation



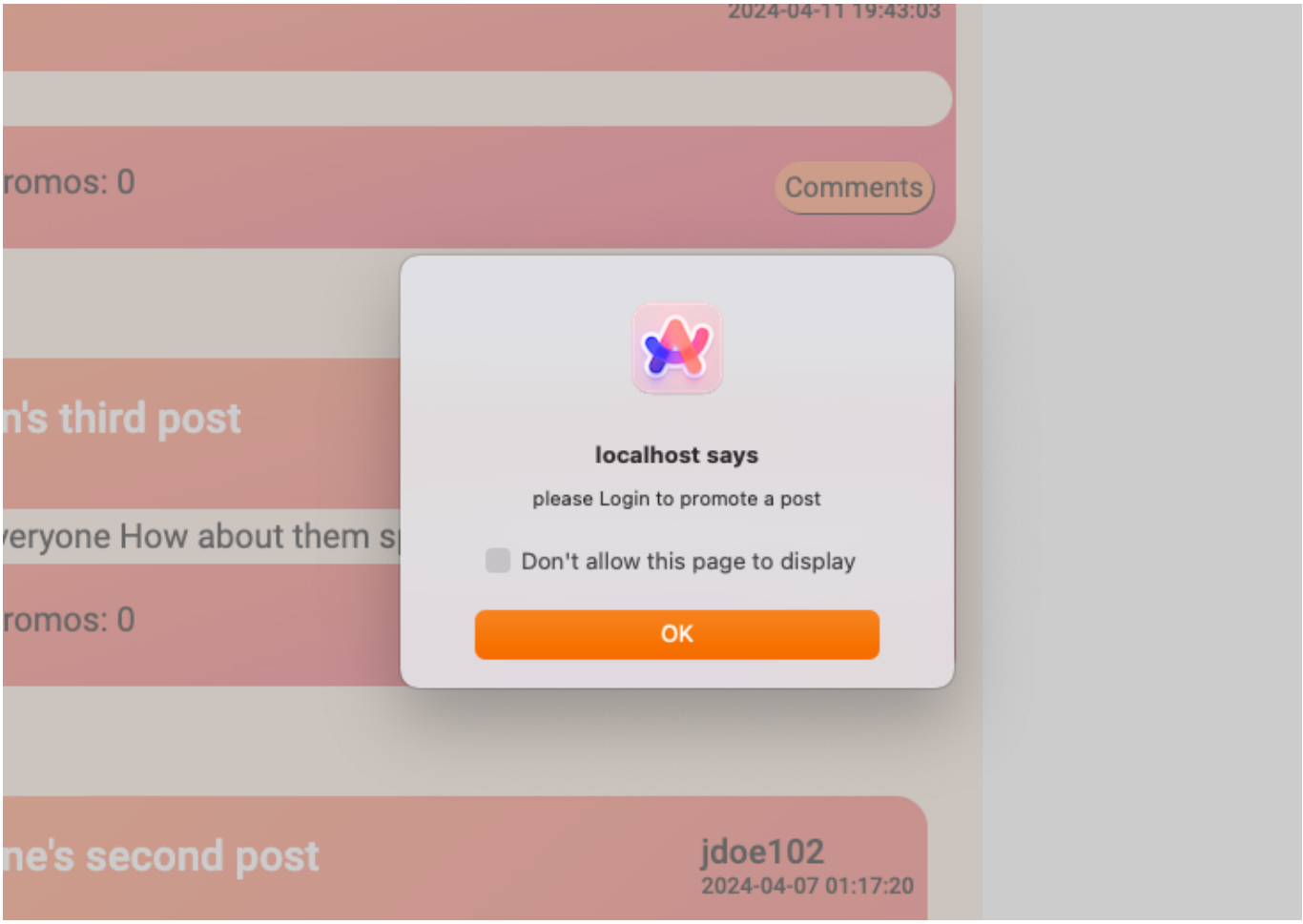
Deletion confirmation



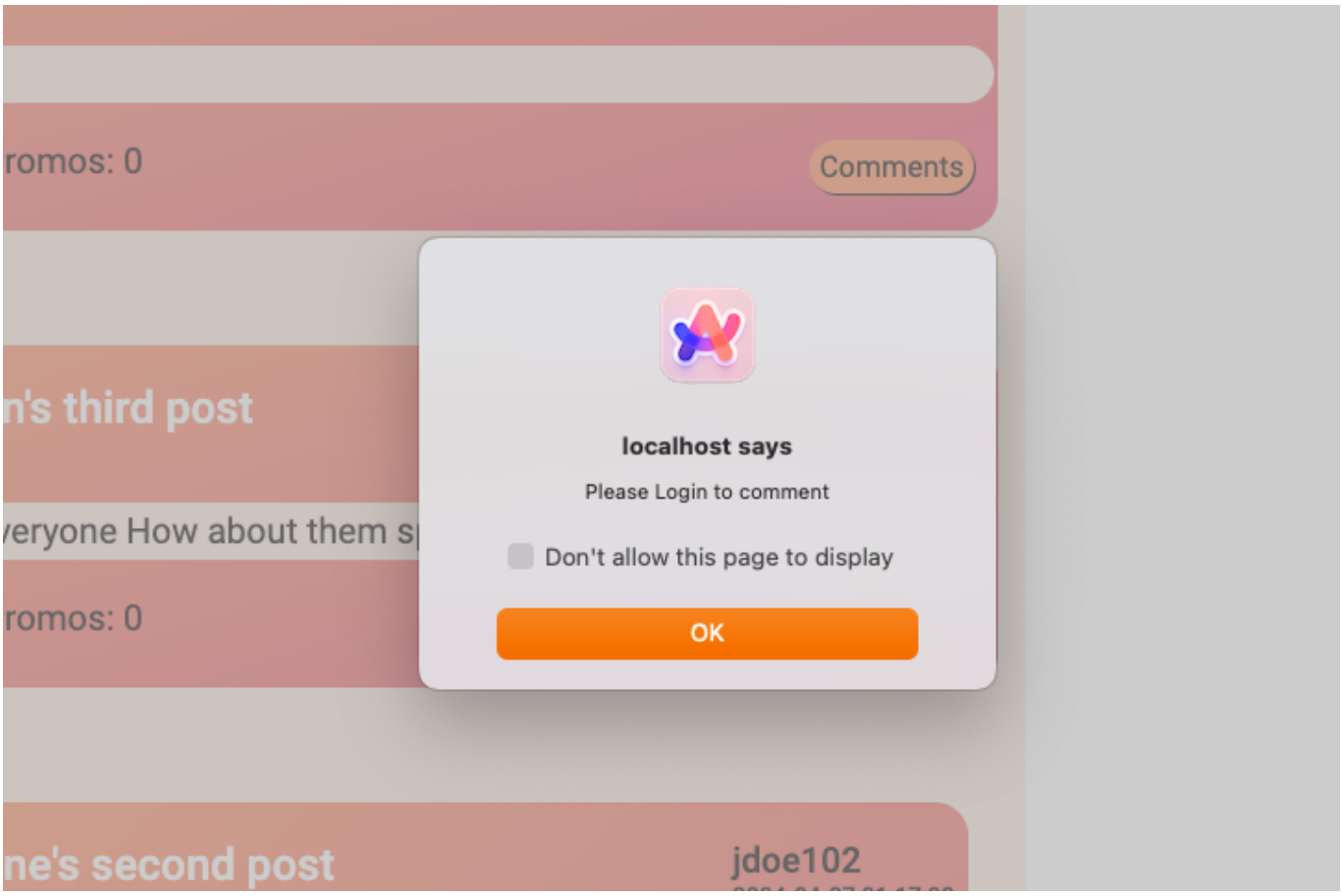
Search form No results handling



Trying to promote a post while not logged in



Trying to comment while not logged in



Other Errors

Other Errors are hard to show as they are often redirects, such as not being logged in but accessing a restricted site, but we welcome any bug/penetration testing!