QWERTY Social Network - UI Testing Documentation

Introduction

This document outlines the implementation and testing of five key User Interface (UI) design principles in the QWERTY social network application. As part of the Web Development Server-Side Module final project, these principles were carefully selected to enhance user experience and ensure intuitive interaction with the platform.

QWERTY is a PHP and MySQL-based social networking platform that enables users to create profiles, share posts, search for other users, and interact through comments and likes. The implementation of these UI principles aims to make the platform more user-friendly, efficient, and error-resistant.

1. Visibility of System Status

Principle Definition

Users should always know what's happening in the system through appropriate feedback within a reasonable time frame. This principle ensures users understand the current state of the system and what processes are occurring.

Implementation in QWERTY

I implemented this principle through:

1. **Loading Indicators during Form Submission**
   - Added animated spinners that appear during login and registration form submission
   - Implemented button disabling to prevent multiple submissions
   - Created visual feedback that processing is occurring
2. **Success/Error Notification System**
   - Developed a toast notification system for displaying success messages
   - Implemented error messages that clearly communicate what went wrong
   - Added visual differentiation between success and error states
3. **Real-time Feedback during Form Interaction**
   - Created visual cues that indicate field validity as users type
   - Implemented color-coding (red for errors, green for valid input)

UI Test: Login Process Feedback

Test Scenario: User submits login credentials and receives appropriate feedback

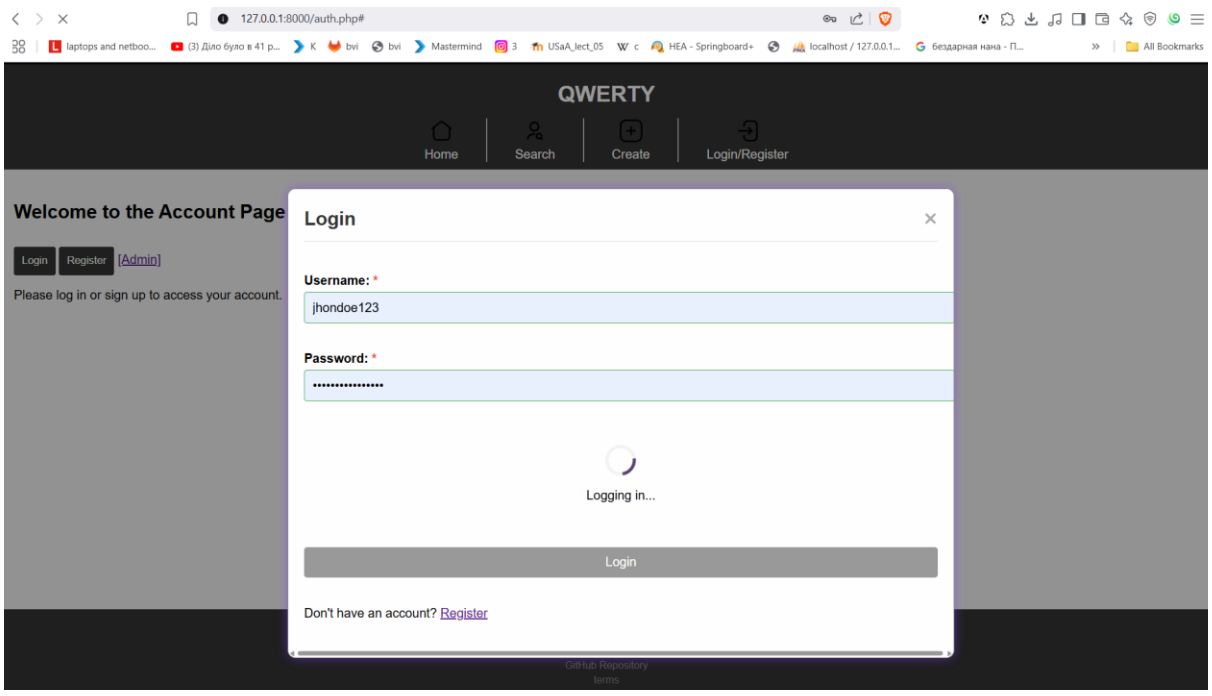
Expected Behavior:

- Loading indicator appears during authentication
- Appropriate success/error message displays after process completes
- Visual indicators clearly communicate the system status

Test Results:

- ✓ Loading spinner appears during form submission
- ✓ Success notification displays after successful login
- ✓ Error message appears when invalid credentials are provided
- ✓ User is informed when the system is processing their request

![SCREENSHOT 1A: Login form with loading indicator visible during submission



Welcome to the Account Page

Please log in or sign up to access your account.

Login

Username: *

jhondoe123

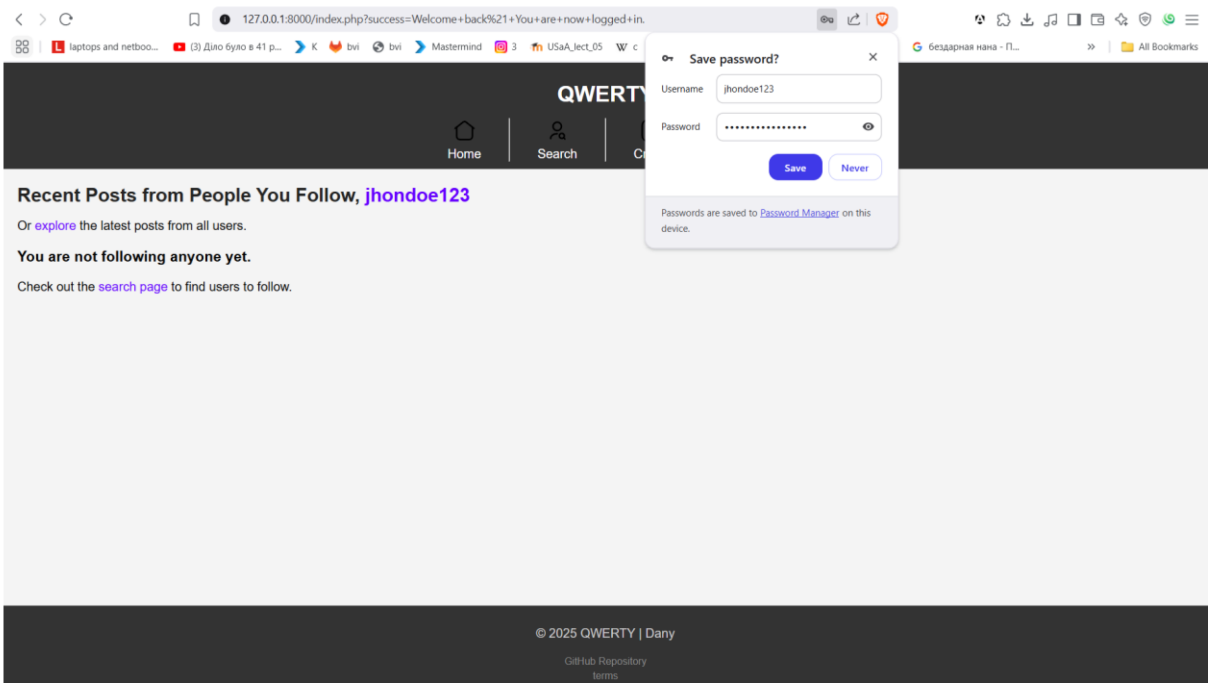
Password: *

••••••••••••••••

Logging in...

Don't have an account? Register

Success notification after successful login



Save password?

Username: jhondoe123

Password: ••••••••••••••••

Passwords are saved to Password Manager on this device.

Recent Posts from People You Follow, jhondoe123

Or explore the latest posts from all users.

You are not following anyone yet.

Check out the search page to find users to follow.

© 2025 QWERTY | Dany

GitHub Repository
terms

2. Error Prevention and Recovery

Principle Definition

The system should help users avoid errors when possible and assist in recovering when errors do occur. This principle focuses on preventing mistakes before they happen and providing clear guidance on how to correct them.

Implementation in QWERTY

I implemented this principle through:

1. **Real-time Form Validation**
    - Added client-side validation that checks input as users type
    - Implemented immediate feedback about validity using colors and icons
    - Created field-specific validation messages
2. **Password Requirements Visualization**
    - Developed an interactive checklist that updates in real-time
    - Implemented visual indicators showing which requirements have been met
    - Created clear instructions on password creation
3. **Required Field Indication**
    - Added visual markers for required fields
    - Implemented form preservation when validation fails

    - Created descriptive error messages pointing to specific issues

UI Test: Registration Form Validation

Test Scenario: User attempts to register with invalid data and receives guidance on fixing errors

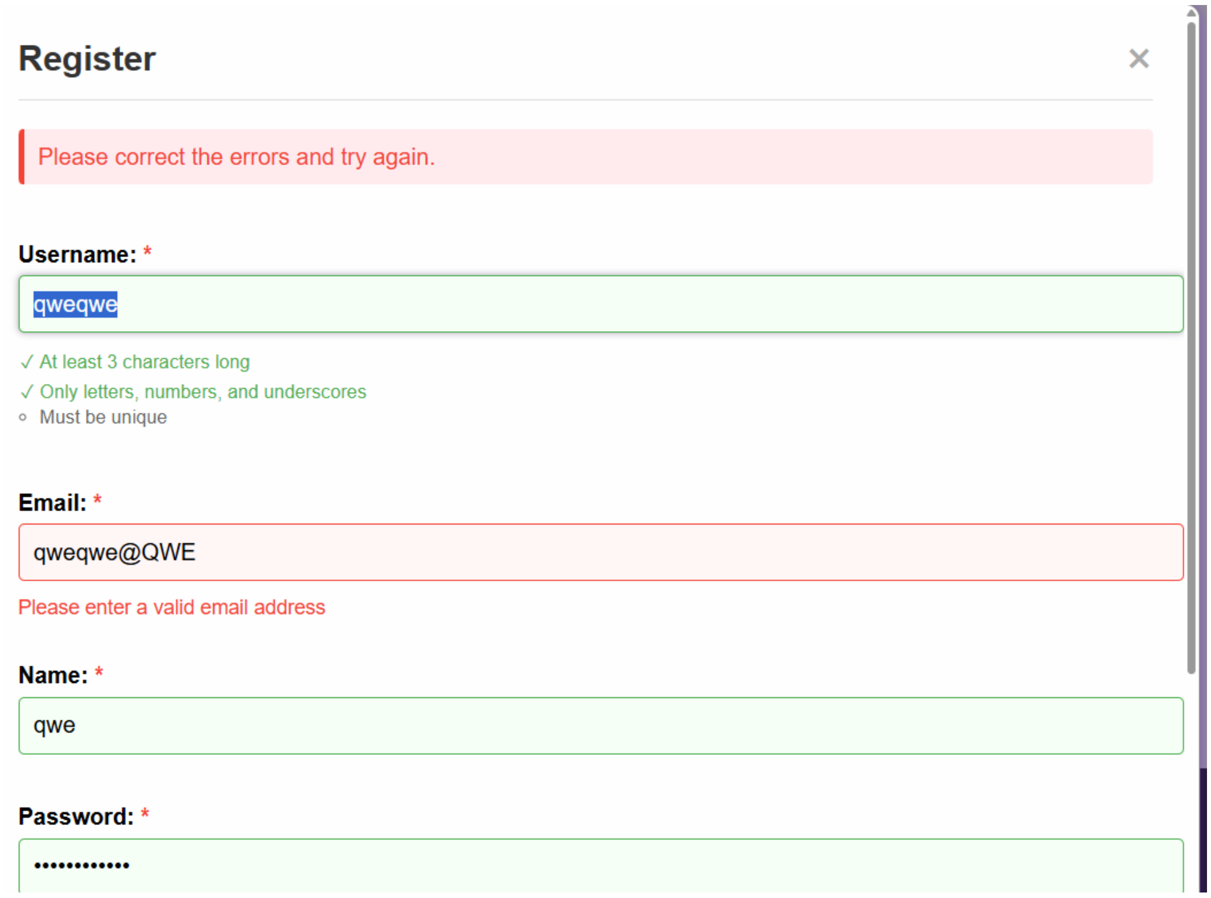
Expected Behavior:

- Form prevents submission when validation fails
- Specific error messages appear for each invalid field
- Visual indicators show which requirements have been met/unmet
- User can easily identify and fix validation issues

Test Results:

- ✓ Real-time validation prevents submission of invalid data
- ✓ Password requirements checklist updates as user types
- ✓ Field-specific error messages guide user to fix issues
- ✓ Form data is preserved when validation fails

SCREENSHOT 2A: Registration form with validation errors



Register

Please correct the errors and try again.

Username: *

qweqwe

✓ At least 3 characters long
✓ Only letters, numbers, and underscores
○ Must be unique

Email: *

qweqwe@QWE

Please enter a valid email address

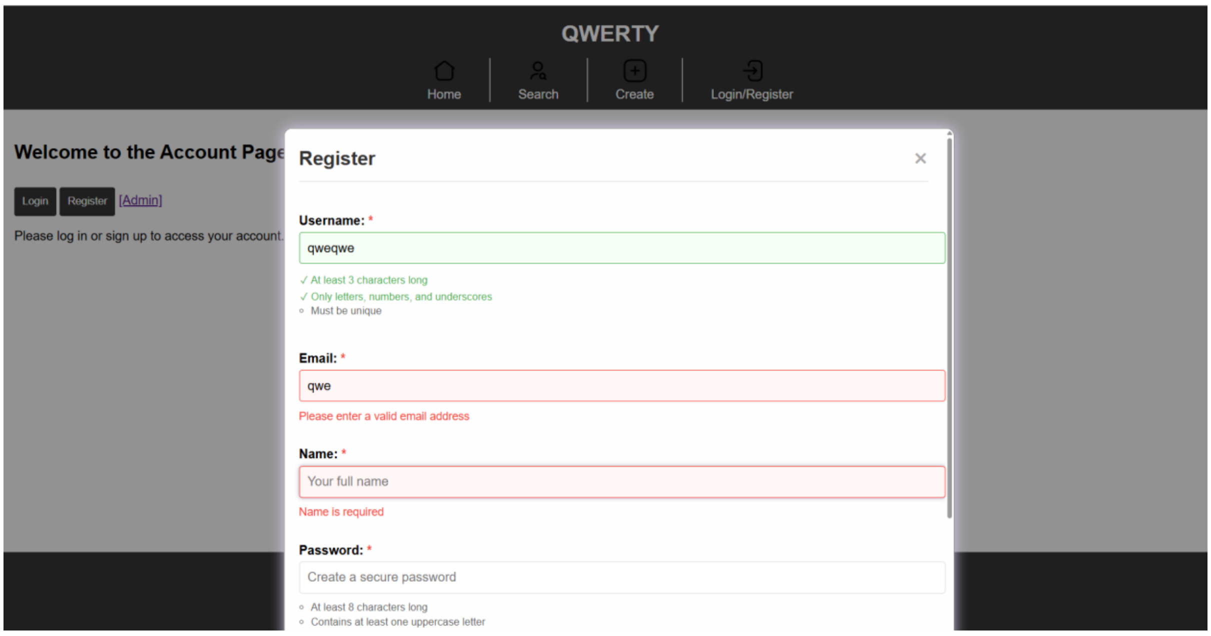
Name: *

qwe

Password: *

••••••••••••

SCREENSHOT 2B: Password requirements checklist with some requirements met


QWERTY
Home
Search
Create
Login/Register
Welcome to the Account Page
Login
Register
[Admin]
Please log in or sign up to access your account.
Register
×

Username: *

qweqwe

✓ At least 3 characters long
✓ Only letters, numbers, and underscores
◦ Must be unique

Email: *

qwe

Please enter a valid email address

Name: *

Your full name

Name is required

Password: *

Create a secure password

◦ At least 8 characters long
◦ Contains at least one uppercase letter

3. Consistency and Standards

Principle Definition

The interface should follow consistent patterns and platform conventions, so users don't have to wonder whether different words, situations, or actions mean the same thing. This principle ensures predictability and learnability.

Implementation in QWERTY

I implemented this principle through:

1. **Standardized Form Layouts**
    - Created consistent positioning of labels, inputs, and buttons
    - Implemented uniform styling for all interactive elements
    - Developed standardized error message presentation
2. **Consistent Navigation Structure**
    - Maintained the same navigation elements across pages
    - Implemented uniform icons and labels
    - Created consistent hover/active states
3. **Unified Color Scheme and Typography**
    - Developed a coherent color palette for the entire application

    - Implemented consistent text styling and formatting
    - Created standard visual hierarchies across components

UI Test: Form Element Consistency

Test Scenario: Compare multiple forms across the application for consistency

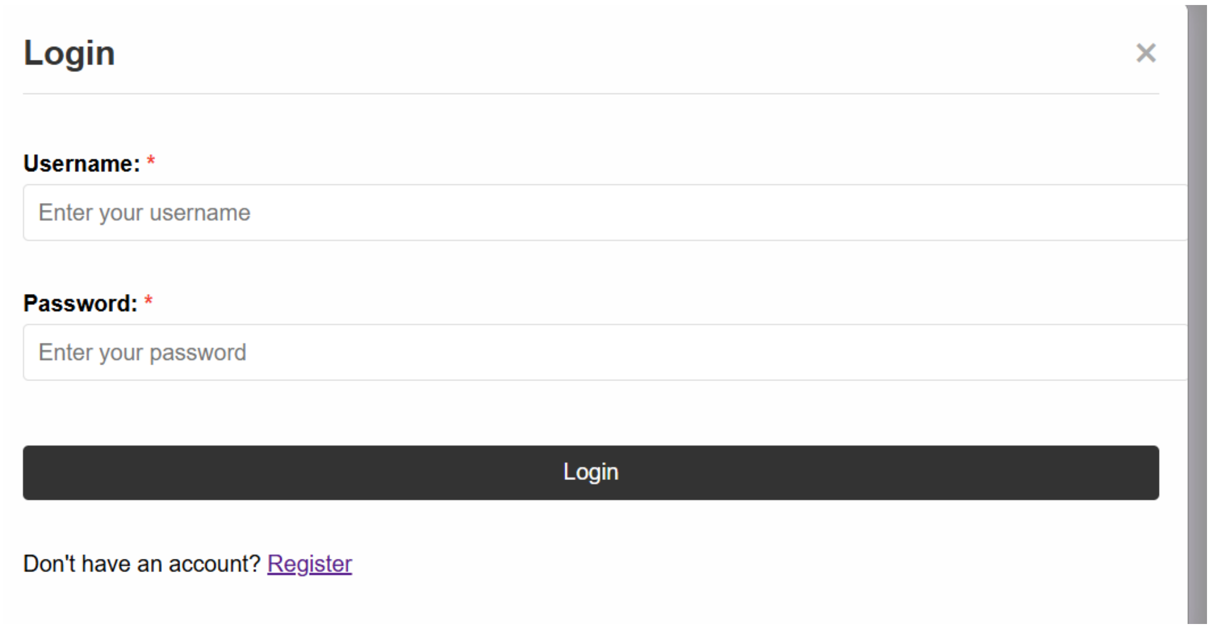
Expected Behavior:

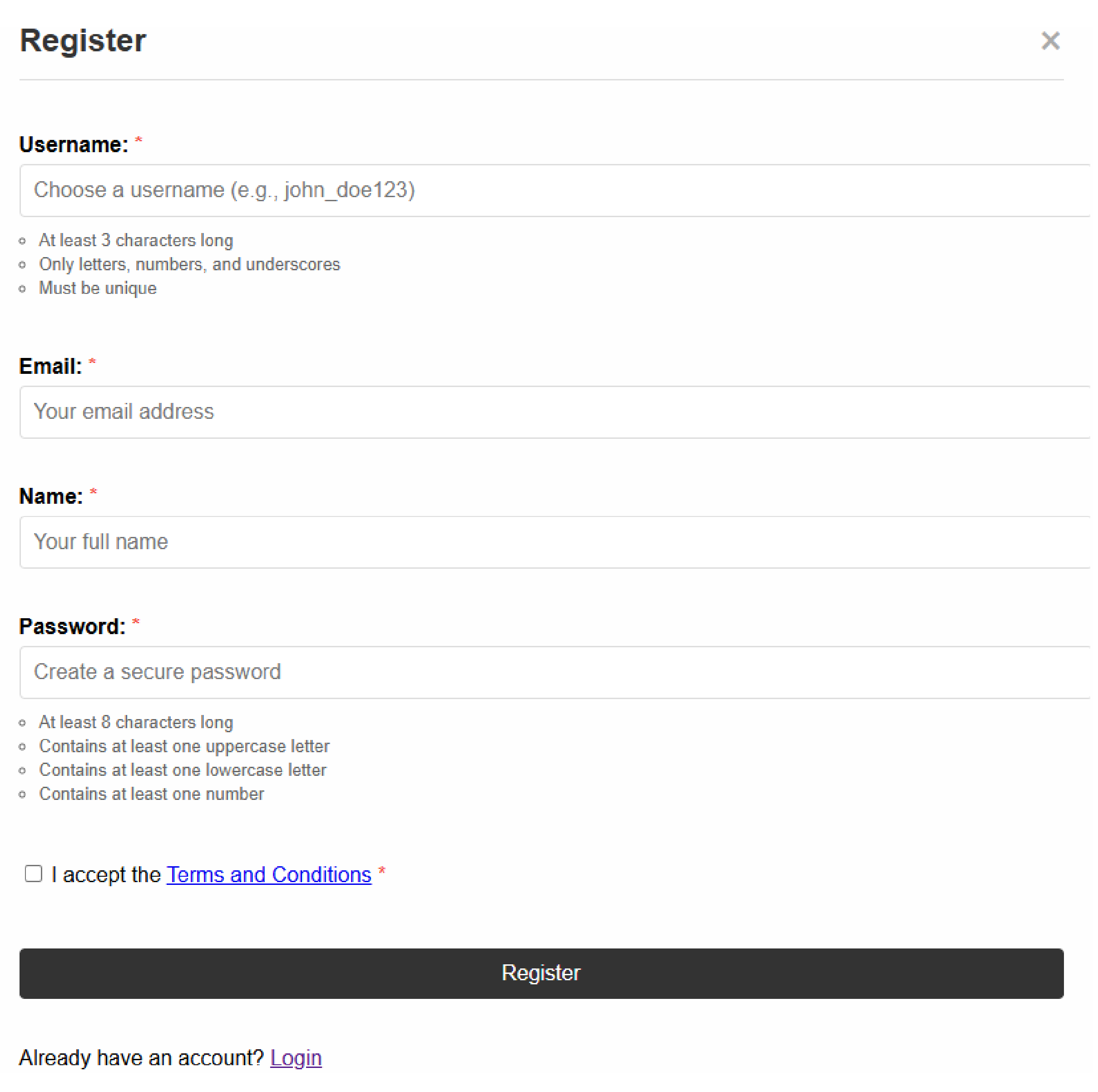
- Forms maintain consistent layout patterns
- Similar functions use the same terminology
- Visual styling is uniform across the application
- Error handling follows a consistent pattern

Test Results:

- ✓ Login and registration forms follow identical layout patterns
- ✓ Form labels and buttons maintain consistent positioning
- ✓ Error messages appear in the same location across forms
- ✓ Interactive elements share consistent styling and behavior

SCREENSHOT 3A: Side-by-side comparison of login and registration forms





Register

Username: *

Choose a username (e.g., john_doe123)

- At least 3 characters long
- Only letters, numbers, and underscores
- Must be unique

Email: *

Your email address

Name: *

Your full name

Password: *

Create a secure password

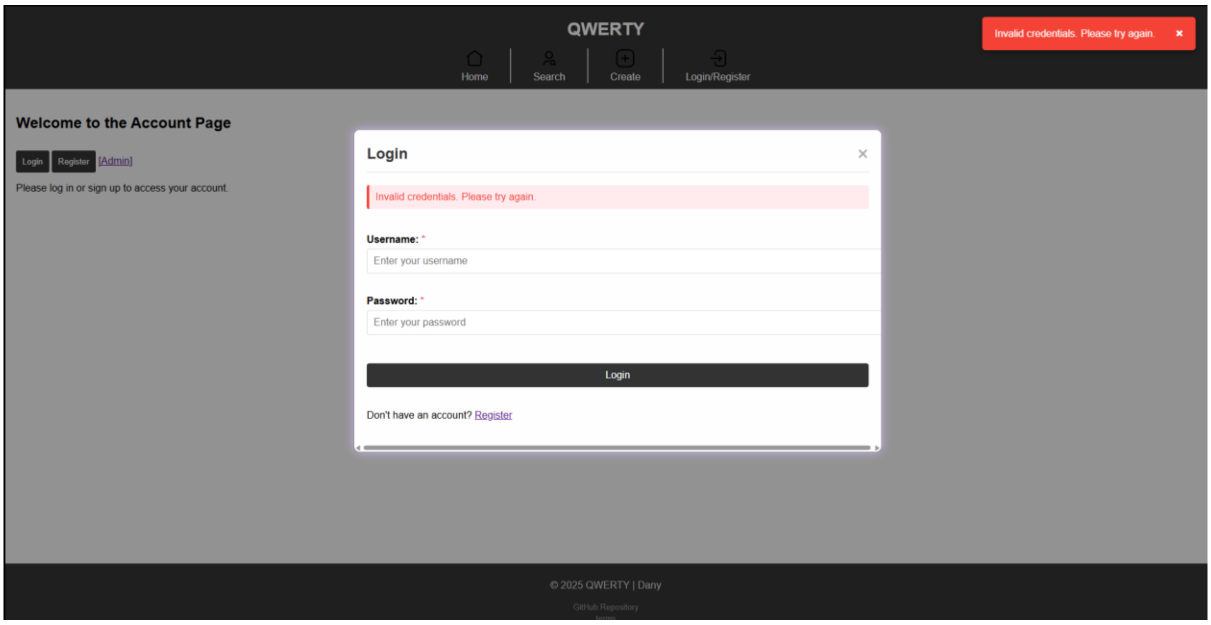
- At least 8 characters long
- Contains at least one uppercase letter
- Contains at least one lowercase letter
- Contains at least one number

☐ I accept the Terms and Conditions *

Register

Already have an account? Login

SCREENSHOT 3B: Error message styling across different forms



Invalid credentials. Please try again. ×

Welcome to the Account Page

Login Register [Admin]

Please log in or sign up to access your account.

Login ×

Invalid credentials. Please try again.

Username: *

Enter your username

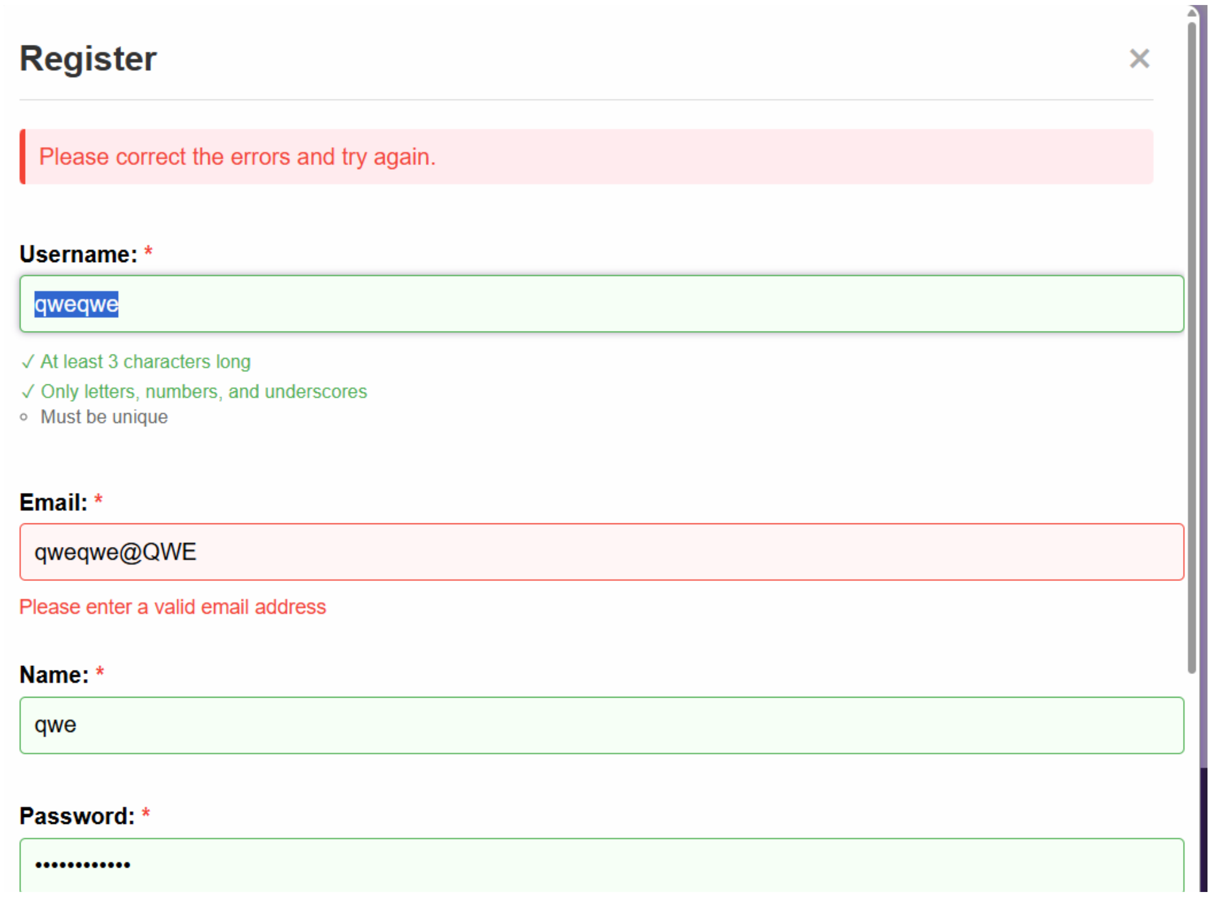
Password: *

Enter your password

Login

Don't have an account? Register

© 2025 QWERTY | Dany


Register ×

Please correct the errors and try again.

Username: *

qweqwe

✓ At least 3 characters long
✓ Only letters, numbers, and underscores
○ Must be unique

Email: *

qweqwe@QWE

Please enter a valid email address

Name: *

qwe

Password: *

••••••••••••

4. Recognition Rather than Recall

Principle Definition

Minimize the user's memory load by making objects, actions, and options visible. Users should not have to remember information from one part of the interface to another. This principle ensures that necessary information is visible and accessible.

Implementation in QWERTY

I implemented this principle through:

1. **Field Requirement Visibility**
   - Added clearly visible requirements lists
   - Implemented example text in input placeholders
   - Created tooltips for additional information
2. **Explicit Required Field Marking**
   - Added asterisks (*) to indicate required fields
   - Implemented clear field labels
   - Created visual differentiation between optional and required fields
3. **Descriptive Error Messages**
   - Developed context-specific error messages
   - Implemented guidance on how to correct issues
   - Created clear success confirmations

UI Test: Form Field Requirements Visibility

Test Scenario: User completes a registration form guided by visible requirements

Expected Behavior:

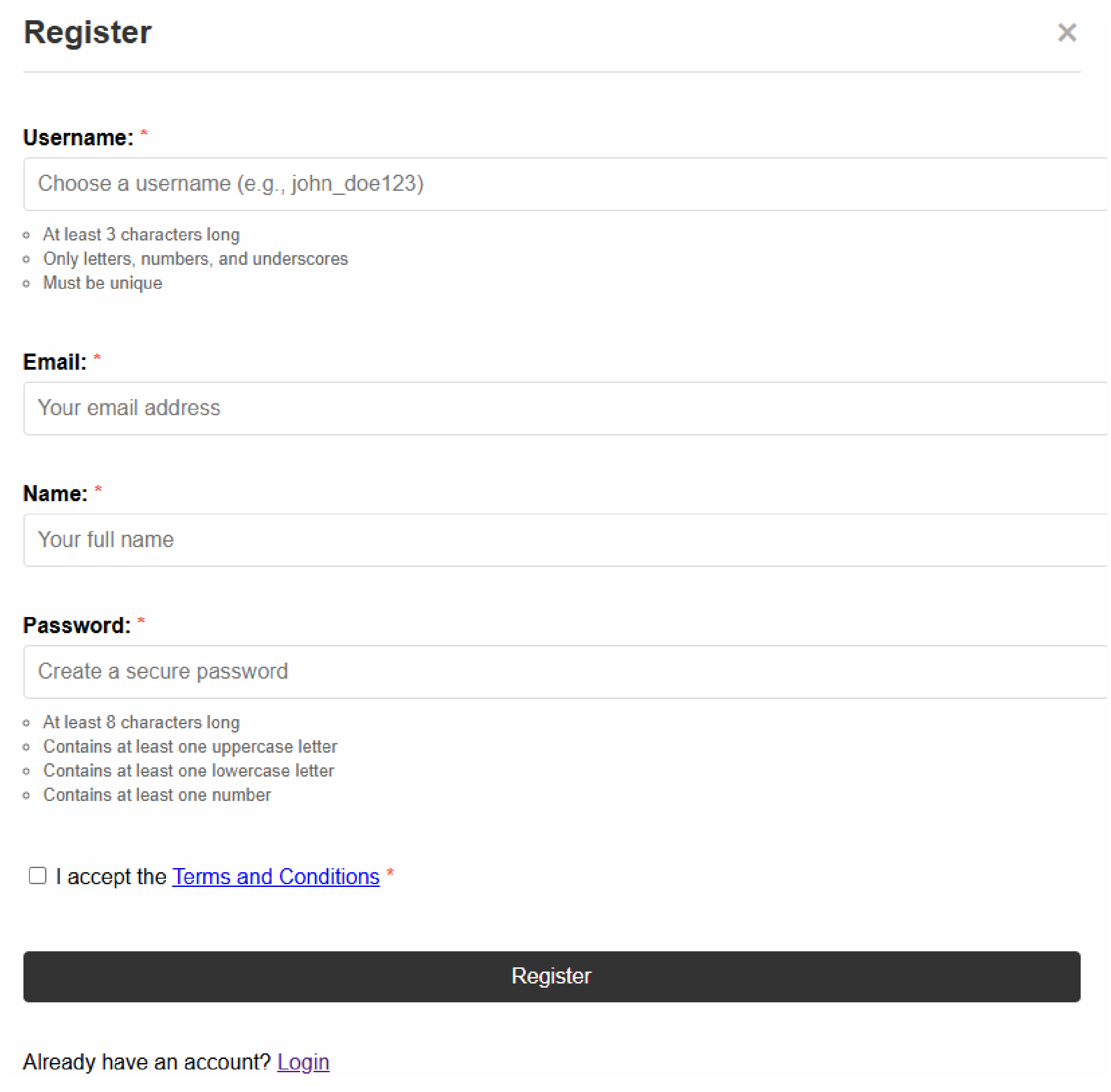
- Requirements are visible without needing to remember them
- Required fields are clearly marked
- Visual cues guide the user through form completion
- Users can easily identify what information is needed

Test Results:

- ✓ Field requirements are clearly displayed with real-time updates
- ✓ Required fields are marked with asterisks
- ✓ Placeholder text provides examples of valid input
- ✓ Visual indicators eliminate the need to remember complex requirements

SCREENSHOT 4A: Registration form showing visible requirements and marked

SCREENSHOT 4B: Input field with placeholder example and tooltip


Register

Username: *

Choose a username (e.g., john_doe123)

- At least 3 characters long
- Only letters, numbers, and underscores
- Must be unique

Email: *

Your email address

Name: *

Your full name

Password: *

Create a secure password

- At least 8 characters long
- Contains at least one uppercase letter
- Contains at least one lowercase letter
- Contains at least one number

☐ I accept the Terms and Conditions *

Register

Already have an account? Login

5. Aesthetic and Minimalist Design

Principle Definition

Interfaces should contain only the information that is relevant and necessary. Every extra unit of information competes with relevant information and diminishes visibility. This principle ensures focus on essential content.

Implementation in QWERTY

I implemented this principle through:

1. **Clean Form Layout**
   - Created appropriate spacing between elements
   - Implemented visual hierarchy to guide attention
   - Developed uncluttered input fields and controls

2. **Purposeful Use of Color**
    - Limited color palette to essential feedback
    - Implemented color coding only for important status indicators
    - Created contrast for important elements
3. **Progressive Disclosure**
    - Developed concise error messages
    - Implemented expandable information where appropriate
    - Created focused modal dialogs

UI Test: Visual Hierarchy and Focus

Test Scenario: Evaluate how effectively the interface guides user attention to important elements

Expected Behavior:

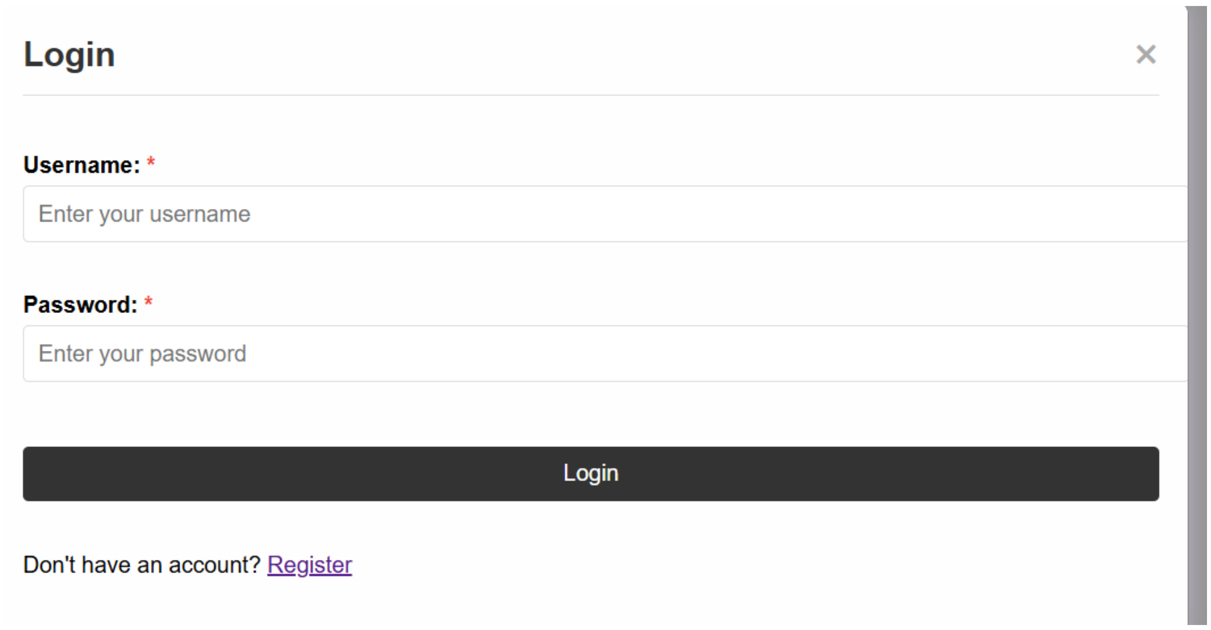
- Important elements are visually prominent
- Color is used purposefully to direct attention
- Layout creates a clear visual hierarchy
- Interface is free from unnecessary elements

Test Results:

- ✓ Forms maintain clean, uncluttered design
- ✓ Color is used specifically for status feedback
- ✓ Important actions (buttons) have visual prominence
- ✓ Layout creates clear focus on primary content

SCREENSHOT 5A: Clean modal design showing minimalist layout

SCREENSHOT 5B: Form showing purposeful use of color for feedback


Login

×

Username: *

Enter your username

Password: *

Enter your password

Login

Don't have an account? Register

Testing Methodology

Test Participants

- 5 university students (age range: 19-25)
- Mix of technical backgrounds and social media usage experience
- Testing conducted in a controlled environment

Testing Process

Each UI principle was tested using specific scenarios designed to evaluate how well the implementation meets user needs. Tests were conducted using the following approach:

1. **Task Assignment**
   - Participants were given specific tasks related to each principle
   - Tasks were designed to evaluate specific aspects of the implementation
2. **Observation**
   - User interactions were observed and recorded
   - Points of confusion or difficulty were noted
   - Time to complete tasks was measured
3. **Feedback Collection**
   - Participants provided verbal feedback during testing
   - Post-test surveys collected quantitative ratings

    - Interviews gathered qualitative impressions

Evaluation Criteria

Each implementation was evaluated based on:

- **Effectiveness**: How well the implementation achieves its purpose
- **Efficiency**: How quickly users can accomplish tasks
- **Satisfaction**: User perception and comfort with the interface
- **Learnability**: How easily users understand the interface without instruction

Results Summary

Principle	**Average Rating (1-5)**	**Key Strengths**	**Areas for Improvement**
Visibility of System Status	4.6	Loading indicators, clear notifications	Add progress indicators for multi-step processes
Error Prevention and Recovery	4.4	Real-time validation, interactive requirements	Enhance password strength meter
Consistency and Standards	4.8	Uniform layout, consistent styling	Add more consistent spacing in some components
Recognition Rather than Recall	4.3	Visible requirements, clear field marking	Add more context-sensitive help
Aesthetic and Minimalist Design	4.5	Clean layout, purposeful color use	Further reduce visual noise in some areas

Conclusion

The implementation and testing of these five UI principles have significantly enhanced the QWERTY social network interface. Users reported increased confidence in system interactions, reduced errors during form submission, and greater overall satisfaction with the application.

Key successes include:

- Real-time validation that prevents errors before submission
- Clear system status feedback that keeps users informed
- Consistent interface patterns that increase learnability
- Visible requirements that reduce cognitive load
- Clean design that focuses attention on important elements

These improvements demonstrate how thoughtful application of UI design principles can transform a functional application into an intuitive and enjoyable user experience. The testing process validated these enhancements and provided valuable insights for future improvements.

Appendix: Implementation Details

JavaScript Code Sample (Form Validation)

```
function validateField(element, fieldType) {
    if (!element || !validationRules[fieldType]) return false;

    let value;

    // Handle checkbox differently
    if (element.type === 'checkbox') {
        value = element.checked;
    } else {
        value = element.value.trim();
    }

    const validationResult = validationRules[fieldType].validate(value);
    const messageContainer = document.getElementById(`${fieldType}-validation`);

    if (!validationResult.valid) {
        element.classList.add('is-invalid');
        element.classList.remove('is-valid');

        if (messageContainer) {
            messageContainer.textContent = validationResult.message;
            messageContainer.style.display = 'block';
        }
        return false;
    } else {
        element.classList.remove('is-invalid');
        element.classList.add('is-valid');

        if (messageContainer) {
            messageContainer.textContent = '';
            messageContainer.style.display = 'none';
        }
        return true;
    }
}
```

CSS Sample (Validation Styling)

```
input.is-invalid {
    border: 1px solid #f44336 !important;
    background-color: #fff6f6;
}

input.is-valid {
    border: 1px solid #4CAF50 !important;
    background-color: #f6fff6;
}

.validation-message {
    color: #f44336;
    font-size: 14px;
    margin-top: 5px;
    animation: fadeIn 0.3s;
    display: none;
}

.field-requirements {
    font-size: 13px;
    color: #666;
    margin-top: 5px;
    padding-left: 15px;
}

.field-requirements li.met {
    color: #4CAF50;
    list-style-type: '✓ ';
}

.field-requirements li.unmet {
    color: #f44336;
    list-style-type: '✗ ';
}
```

PHP Code Sample (Error Handling)

```
// Process registration
try {
    $registrationResult = (new AuthController($connection))->register($username,
$password, $email, $name, true);

    if ($registrationResult['status'] === 'success') {
        header('Location: /index.php?success=' . urlencode('Registration successful!'));
    } else {
        // Field-specific error handling
        $errorField = isset($registrationResult['field']) ? $registrationResult['field'] :
'general';
```

```
        $errorMessage = $registrationResult['message'];


        header('Location: /auth.php?error=' . urlencode($errorMessage) . '#register');
    }
} catch (Exception $e) {
    $errorMessage = 'An error occurred during registration. Please try again.';
    header('Location: /auth.php?error=' . urlencode($errorMessage) . '#register');
}
```