# Forms and Validation

*[Design System Image]*

Forms are an integral part of many applications, particularly during the user registration phase. One of the most important parts of working with forms is properly validating the information that users enter into them. In general, Brightlayer UI follows the form behaviors explained in detail by Material Design.

*[MaterialDesignDescription Component - Interactive React component]*

*[TOC Component - Interactive React component]*

---

# Anatomy

## Form Layout

Keep your forms short where possible, but make effective use of whitespace to avoid overwhelming your users. If your forms are larger than the height of your target device, consider breaking them into sections or multiple pages to improve the user experience.

*[Design System Image]*

*[DemoCard Component - Interactive React component]*

*[DemoCard Component - Interactive React component]*

*[DemoCard Component - Interactive React component]*

# Field Requirements

Required / optional fields should be identified according to the Material Design guidelines. If your form fields have complex requirements for valid values, you can list these criteria below the input. A common use case for this is when displaying password complexity requirements.

*[Design System Image]*

*[DemoCard Component - Interactive React component]*

# Error Messages

If there is an error with the data entered into a field, the field should indicate to the user that there is a problem. Label and helper text should be changed to a red color and the error message should be displayed in the helper text location below the field — any existing helper text should be replaced with the error message. You may optionally show an additional error icon in the text input.

Error messages should be succinct and should not provide information that could compromise security (e.g., use "Invalid Credentials" instead of "Incorrect Password").

## Success Screens

Once a user has reached the end of a form successfully, you should present them with a success screen. This screen should indicate to the user that their submission was accepted and offer information about the next steps (e.g., "Click the finish button below to log in."). Typically these success screens are stylized for your application and will include some bold graphics or illustrations.

## Mobile Keyboard

Mobile devices offer multiple different software keyboard variations for different types of inputs. Make sure that you are using the appropriate keyboard for your inputs (e.g., use the email keyboard for email inputs, phone number keyboard for a phone number field, etc.).

Typically, mobile devices also allow you to change the label on the return key of the software keyboard. This label should be appropriate to the context (see Submitting the Form below).

The focused input field should remain centered in the screen when the software keyboard is open — it should not be hidden behind the keyboard.

## Password Visibility

Depending on the security requirements of your application, it may be useful to add a password visibility toggle button to fields where users must enter a password (such as during login or changing a password). The visibility toggle switches the input type between hidden text and plain text so users can ensure that they typed their password correctly.

---

# Behaviors

## Verifying User Input

In general, there are three acceptable times to perform form field verification:

- on blur: the field is verified when the user clicks away from it
- on change: the field is dynamically verified on each keystroke or change in the value
- on action: fields are verified after performing a specific action, such as clicking a button

### Verifying on Blur

In order to avoid distracting or confusing users, most text fields should verify input when the field loses focus. Error messages should not appear until the user is finished entering data and the field is no longer in focus.

*[Design System Image]*

*[DemoCard Component - Interactive React component]*

### Verifying on Change

Some fields should verify the user input as it changes. The most common case for this type of verification is a new password field or a confirm password field. The verification feedback should be displayed / updated each time the user makes a change to the input field.

*[Design System Image]*

## Verifying on Action

In some specific situations, you may need to defer verification or perform it for all fields in a form at once. In this case, you need to link verification to a specific action, such as clicking a button.

This type of verification is common when the checks cannot be performed on the client, such as making an API call to verify a registration code, check user login credentials, or search for a device on the network. In most cases, you can do some preliminary checks using one of the two methods above (e.g., you can check if the email is a valid email before making the API call to verify the credentials).

If verification is being triggered by clicking on a button, you should replace the button text with a loading spinner while the verification is in progress. If the verification is successful, the user should be automatically taken to the next screen.

*[Design System Image]*

*[DemoCard Component - Interactive React component]*

# Submitting the Form

Forms should have a button to submit the data or continue to the next part of the form if there are multiple parts. Depending on what happens next, the button should be labeled _Next_, _Finish_, or similar. The button should be located at the bottom right corner of the form (bottom left for right-to-left languages).

*[Design System Image]*

On mobile, if there are multiple input fields on the screen, the keyboard action button should say "next", unless you are on the last item of the page, in which case the keyboard button should say "done" or "submit". This button can either submit the form or simply dismiss the keyboard (if you want users to have a chance to review their information first).

*[Design System Image]*

For screens where there is a single field with a known required length (e.g., a verification code), the submit button can be omitted and the form submitted as soon as the required length is reached.

*[Design System Image]*

*[DemoCard Component - Interactive React component]*

### Disabling the Submit Button

If there are errors indicated on any of the form fields, the submit button should be disabled. If you are performing verification when clicking the submit button (onAction), all the invalid fields should be indicated if they aren't already.

*[Design System Image]*

*[Design System Image]*

---

# Design Specifications

For additional instructions, refer to Material Design's specifications for "Text fields".

## Mobile

*[Design System Image]*

## Desktop and Tablet

If certain dimensions are not specified, refer to the dimensions suggested in the mobile section above.

*[Design System Image]*

---

# Developers

Use the following components to implement this pattern:

**Angular** — Angular Material

- Form Field
- Input

**React** — MUI

- Text Field

**React Native** — React-Native-Paper

- TextInput