Form validation

Workshop #3

Flutter Developer Bootcamp

# **Purpose**

This workshop demonstrates how to implement basic form validation for username and password fields in a Flutter application using text controllers, regular expressions, and user interface components.

# **Problem**

In the given workshop,there will be 2 text fields Username and password.(if the username is invalid it shows a error message in red color) Form validation ensures that the data entered by the user in form fields meets certain criteria or constraints before it's submitted.In this case, the validation checks if the username and password meet specific requirements:

1. Circle avatar
2. Username Validation:

* Must not be empty.
* Must contain 4 to 12 alphanumeric characters.

2 .Password Validation:

* Must not be empty.
* Must contain at least 8 characters.
* Must include at least one letter and one number.

3. If any error occurs, show a dialog box with the text message "Please fill in all fields correctly.

# **How to Solve**

1. Checkout the workshop from Git Repo:

git clone -b <user-branch> <repo-URL>

2. Open the root folder inside VS Code

3. To build the app click the run option in the main method{}

4. Type an invalid password in the field <textfield-name>

5. Click the Submit button, show a dialog box with the error text message "Please fill in all fields correctly.

6. Go To File: <specific-file-with-validation-method> à <method-name>, implement your validation logic. Make the validation work.

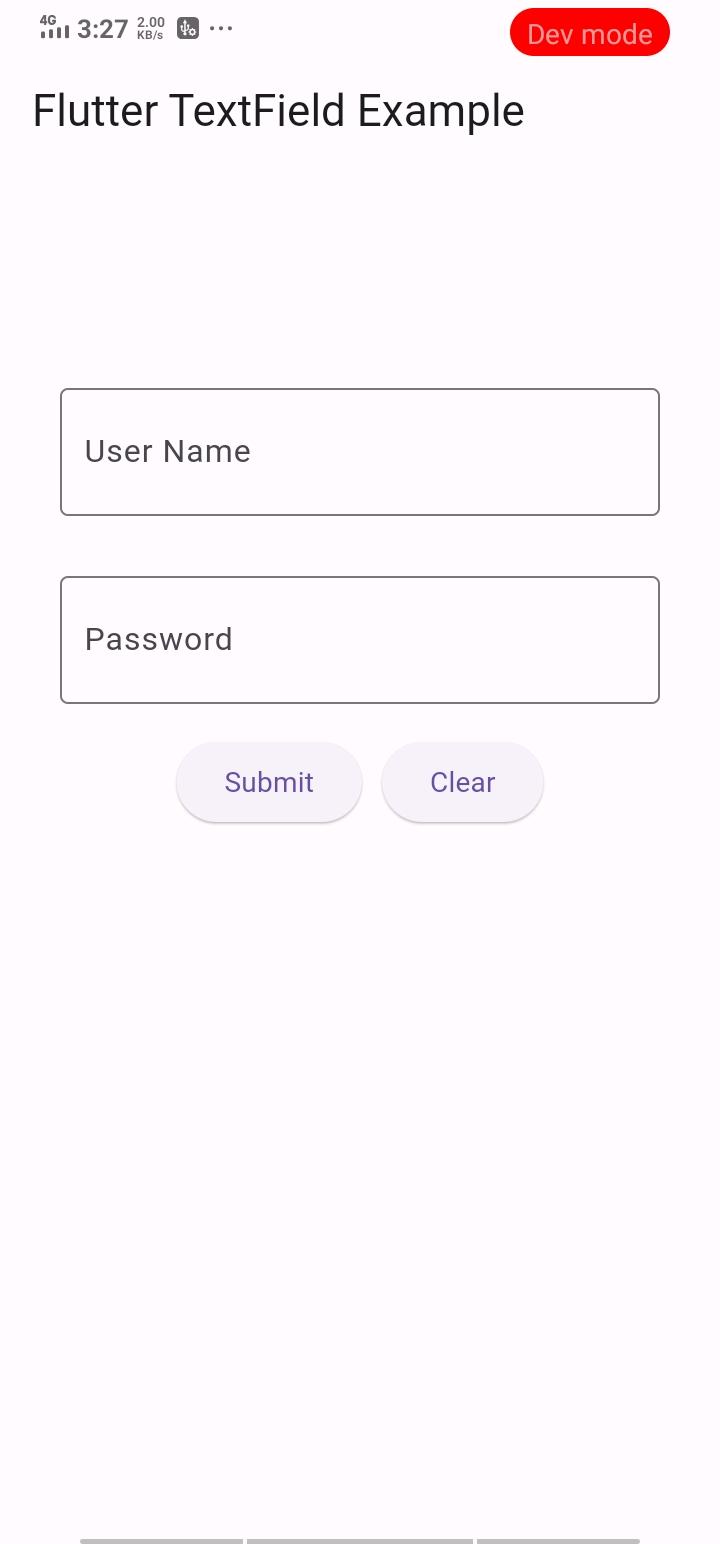
**You will Achieve**

When you complete this workshop you will learn the following:

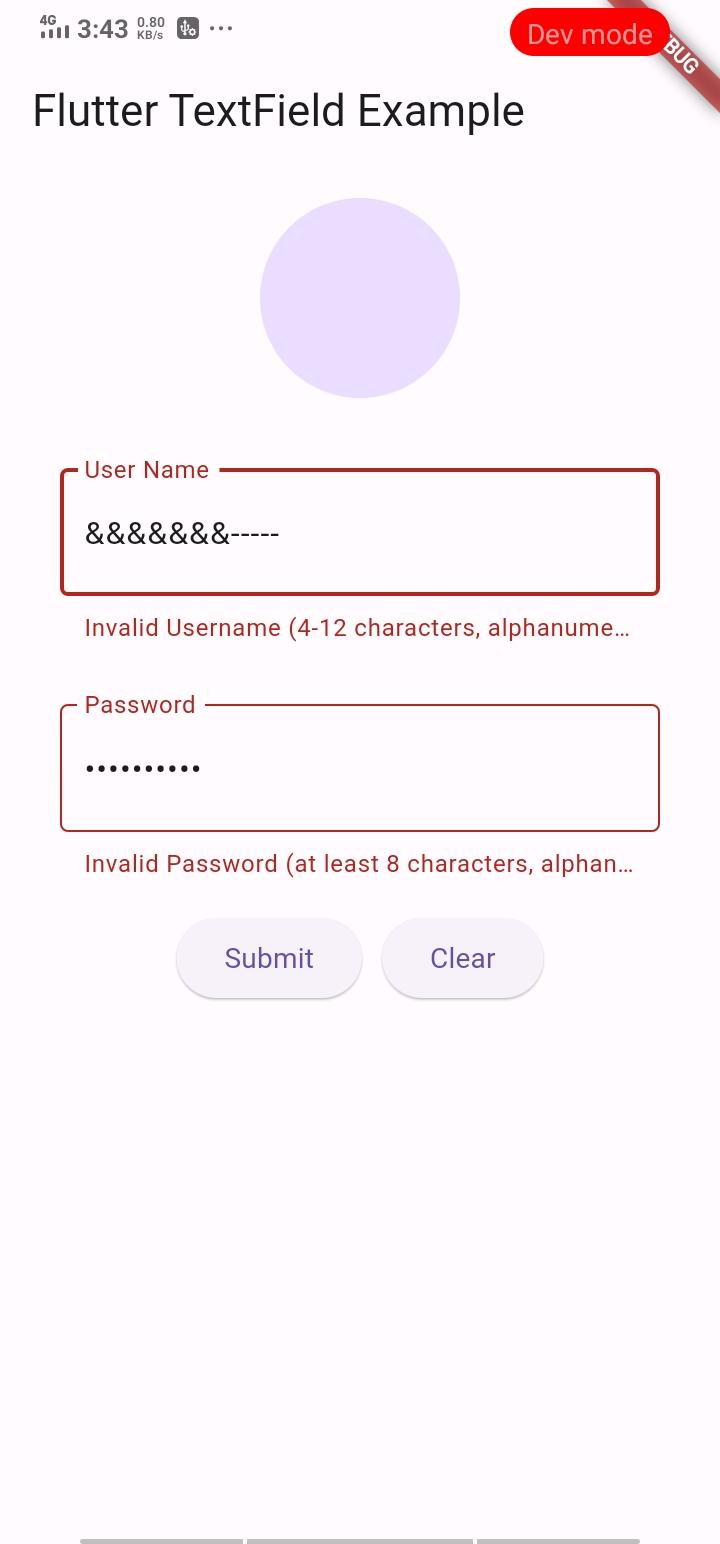
* **Circle Avatar**: Learn how to incorporate circular profile images or placeholders into your app's user interface, providing a visually appealing design.
* **Regex Validation**: Understand how to use regular expressions to validate user input in text fields, ensuring that the entered data meets specific criteria.
* **TextField**: Implement text fields for users to input text or data into your app, facilitating user interaction and data entry.
* **Decoration**: Explore techniques for applying decoration to widgets, such as text fields, buttons, or containers, to enhance their visual appearance and style.
* **Buttons**: Integrate buttons into your app to trigger actions or navigate to different screens, enhancing user interactivity and navigation.
* **Dialog Box**: Utilize dialog boxes to display important messages, alerts, or prompts to users, improving user communication and interaction.
* **Padding**: Adjust spacing around widgets using the Padding widget to improve the layout and aesthetics of your app's user interface.
* **SizedBox**: Manage widget sizes and spacing within your app's layout using the SizedBox widget, providing flexibility and control over the layout's structure.
* **Text**: Display text content within your app's interface, including headings, paragraphs, labels, or other textual information, ensuring effective communication with users.
* **Adjusting Text Alignment**: Control the alignment of text elements within your app's interface to improve readability and visual consistency, enhancing the overall user experience.

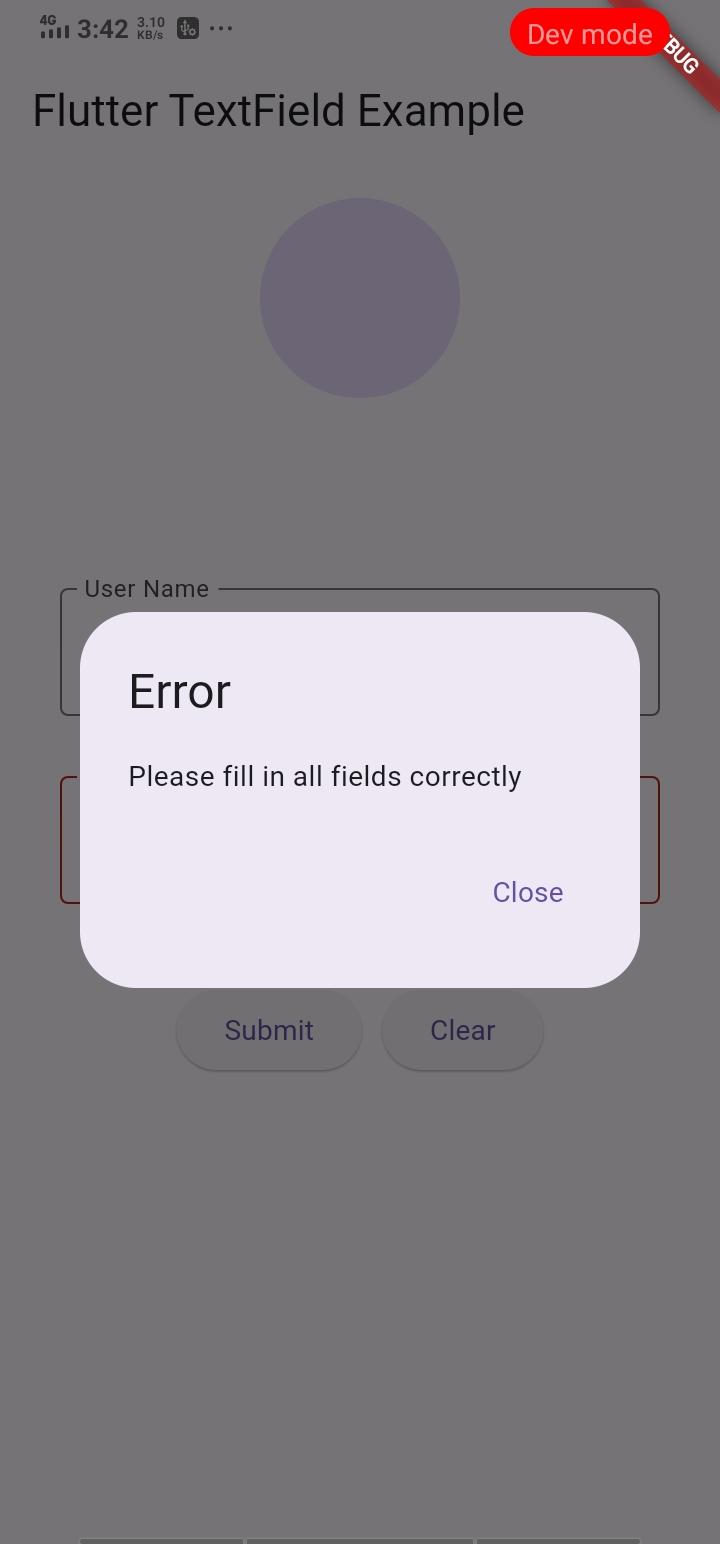
# **Screenshots**

## **Before implementation (without validation)**



## **After implementation (With validation)**





# **How to submit your workshop**

Push your project back to the same git branch using command:

<command name>

# **Happy Coding!**