EXPERIMENT NO. 4
MAD and PWA Lab

Aim: To create an interactive Form using a form widget.

Theory:

Apps often require users to enter information into a **Text Field**. For example, you might require users to log in with an Email Address and Password combination.

To make apps secure and easy to use, check whether the information the user has provided is valid. If the user has correctly filled out the form, process the information. If the user submits incorrect information, display a friendly error message letting them know what went wrong.

First, create a Form. The **Form widget** acts as a container for grouping and validating multiple form fields. When creating the form, provide a **GlobalKey**. This uniquely identifies the Form, and allows validation of the form in a later step.

```dart
import 'package:flutter/material.dart';

class TextFieldInput extends StatelessWidget {
    const TextFieldInput({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final _formKey = GlobalKey<FormState>();

    return Form(
      key: _formKey,
      child: TextFormField(
      ),
    );
  }
}
```

Although the Form is in place, it doesn't have a way for users to enter text. That's the job of a TextFormField. The **TextFormField widget** renders a material design text field and can display validation errors when they occur.

Validate the input by providing a **validator()** function to the TextFormField. If the user's input isn't valid, the validator function returns a String containing an error message. If there are no errors, the validator must return null.

For this example, create a validator that ensures the TextFormField isn't empty. If it is empty, return a friendly error message.

```
import 'package:flutter/material.dart';

class TextFieldInput extends StatelessWidget {
    const TextFieldInput({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final _formKey = GlobalKey<FormState>();

    return Form(
      key: _formKey,
      child: TextFormField(
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter the respective details';
          }
          return null;
        },
      ),
    );
  }
}
```

Now that you have a form with a text field, provide a **Button** that the user can tap to submit the information.

When the user attempts to submit the form, check if the form is valid. If it is, display a success message. If it isn't (the text field has no content) display the error message.

First of all, I have created a separate class for **TextInputField** which can be reused whenever there is a form and the user has to enter the details. These are the properties I have used while creating the Input Field.

```dart
import 'package:flutter/material.dart';

class TextFieldInput extends StatelessWidget {
  final TextEditingController textEditingController;
  final bool isPass;
  final String hintText;
  final TextInputType textInputType;
  const TextFieldInput({
    Key? key,
    required this.textEditingController,
    this.isPass = false,
    required this.hintText,
    required this.textInputType,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final inputBorder = OutlineInputBorder(
      borderSide: Divider.createBorderSide(context),
    );

    final _formKey = GlobalKey<FormState>();

    return Form(
      key: _formKey,
      child: TextFormField(
        validator: (String? value) {
          if (value == null || value.isEmpty) {
            return 'Please enter the respective details';
          }
          return null;
        },
        cursorColor: Colors.white,
        cursorWidth: 1,
        controller: textEditingController,
        decoration: InputDecoration(
          hintText: hintText,
          border: inputBorder,
          focusedBorder: inputBorder,
          enabledBorder: inputBorder,
          filled: true,
          contentPadding: const EdgeInsets.all(
            17,
          ),
        ),
        keyboardType: textInputType,
        obscureText: isPass,
      ),
    );
  }
}
```

The essential properties I have used are :

Property	**Description**
cursorColor	It shows the color of the cursor. The cursor indicates the current location of the text insertion point in the field.
cursorWidth	It shows how thick the cursor will be. It defaults to 2.0 pixels.
controller	It controls the text being edited. If null, this widget will create its own TextEditingController.
decoration	It controls the BoxDecoration of the box behind the text input. It defaults to having a rounded rectangle grey border and can be null to have no box decoration.
hintText	It is displayed on top of the child of InputDecorator when the input isEmpty and either labelText is null or the input has the focus.
border	It is the shape of the border to draw around the decoration's container.
focusedBorder	It is the border to display when the InputDecorator has the focus and is not showing an error.
enabledBorder	It is the border to display when the InputDecorator is enabled and is not showing an error.
filled	If it is true, the decoration's container is filled with fillColor.
contentPadding	It is the padding for the input decoration's container. The decoration's container is the area which is filled if filled is true and bordered per the border.

keyboardType	It shows the type of keyboard to use for editing the text. It defaults to TextInputType.text if maxLines is one and TextInputType.multiline otherwise.
obscureText	It is used to hide the text being edited (e.g., for passwords). When this is set to true, all the characters in the text field are replaced by * .

I have used the **TextFieldInput** class in the **Login Screen** and **Sign Up Screen**. The code in Login Screen is as follows :

```
// Text field input for email
Container(
      padding: const EdgeInsets.symmetric(horizontal: 32),
      child: TextFieldInput(
            hintText: 'Phone number, email or username',
            textInputType: TextInputType.emailAddress,
            textEditingController: _emailController,
      ),
),

const SizedBox(
      height: 19,
),

// Text field input for password
Container(
      padding: const EdgeInsets.symmetric(horizontal: 32),
      child: TextFieldInput(
            hintText: 'Password',
            textInputType: TextInputType.text,
            textEditingController: _passwordController,
            isPass: true,
      ),
),

const SizedBox(
      height: 19,
),
```

Now, adding the validation part to the form. To validate the form, use the **_formKey** created in Step 1. You can use the **_formKey.currentState()** method to access the **FormState**, which is automatically created by Flutter when building a Form.

The **FormState** class contains the **validate()** method. When the validate() method is called, it runs the **validator()** function for each text field in the form. If everything looks good, the validate() method returns true. If any text field contains errors, the validate() method rebuilds the form to display any error messages and returns false.

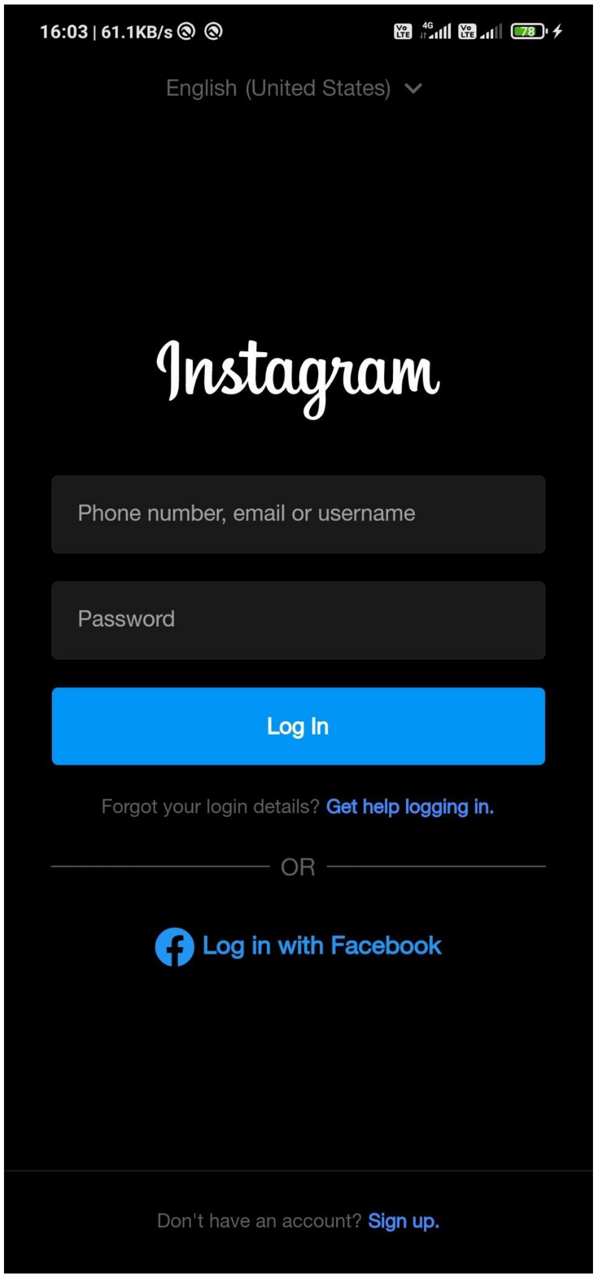
```
Form(
      key: _formKey,
      child: Column(
            children: [
                  // Text field input for email
                  Container(
                        padding: const EdgeInsets.symmetric(
                        horizontal: 32,
                  ),
                  child: TextFieldInput(
                        hintText: 'Phone number, email or username',
                        textInputType: TextInputType.emailAddress,
                        textEditingController: _emailController,
                  ),
            ),

            const SizedBox(
                  height: 19,
            ),

            // Text field input for password
            Container(
                  padding: const EdgeInsets.symmetric(
                        horizontal: 32,
                  ),
                  child: TextFieldInput(
                        hintText: 'Password',
                        textInputType: TextInputType.text,
                        textEditingController: _passwordController,
                        isPass: true,
                  ),
        ),

        const SizedBox(
            height: 19,
        ),

        // Button Login
        Container(
            padding: const EdgeInsets.symmetric(
                horizontal: 32,
            ),
            child: InkWell(
                onTap: () async {
                    if (_formKey.currentState!.validate()) {
                        loginUser;
                    }
                },
                child: Container(
                    child: _isLoading
                        ? const Center(
                            child: CircularProgressIndicator(
                                color: primaryColor,
                            ),
                        )
                        : const Text(
                            'Log In',
                            style: TextStyle(
                                fontSize: 16.0,
                                fontWeight: FontWeight.bold,
                            ),
                        ),
                    width: double.infinity,
                    alignment: Alignment.center,
                    padding: const EdgeInsets.symmetric(
                        vertical: 17,
                    ),
                    decoration: const ShapeDecoration(
                        shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.all(
                                Radius.circular(4),
                                        ),
                                    ),
                                    color: blueColor),
                                ),
                            ),
                        ),
                    ],
                ),
            ),
```


16:03 | 61.1KB/s

English (United States)

Instagram

Phone number, email or username

Password

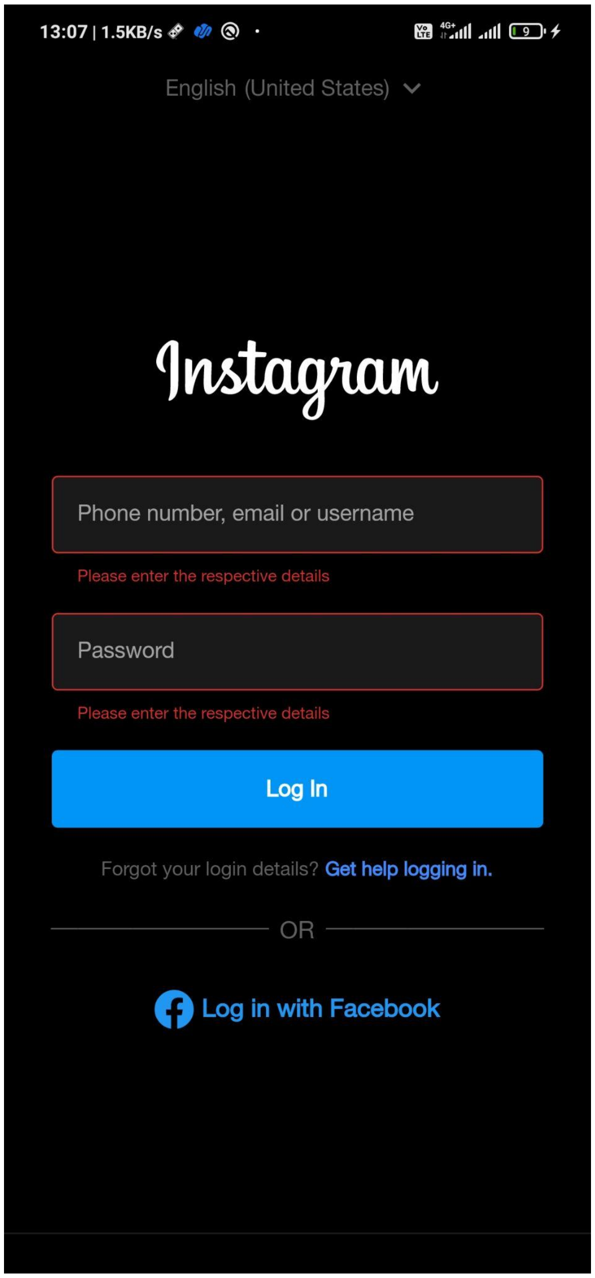
Log In

Forgot your login details? Get help logging in.

OR

Log in with Facebook

Don't have an account? Sign up.


13:07 | 1.5KB/s

English (United States)

Instagram

Phone number, email or username

Please enter the respective details

Password

Please enter the respective details

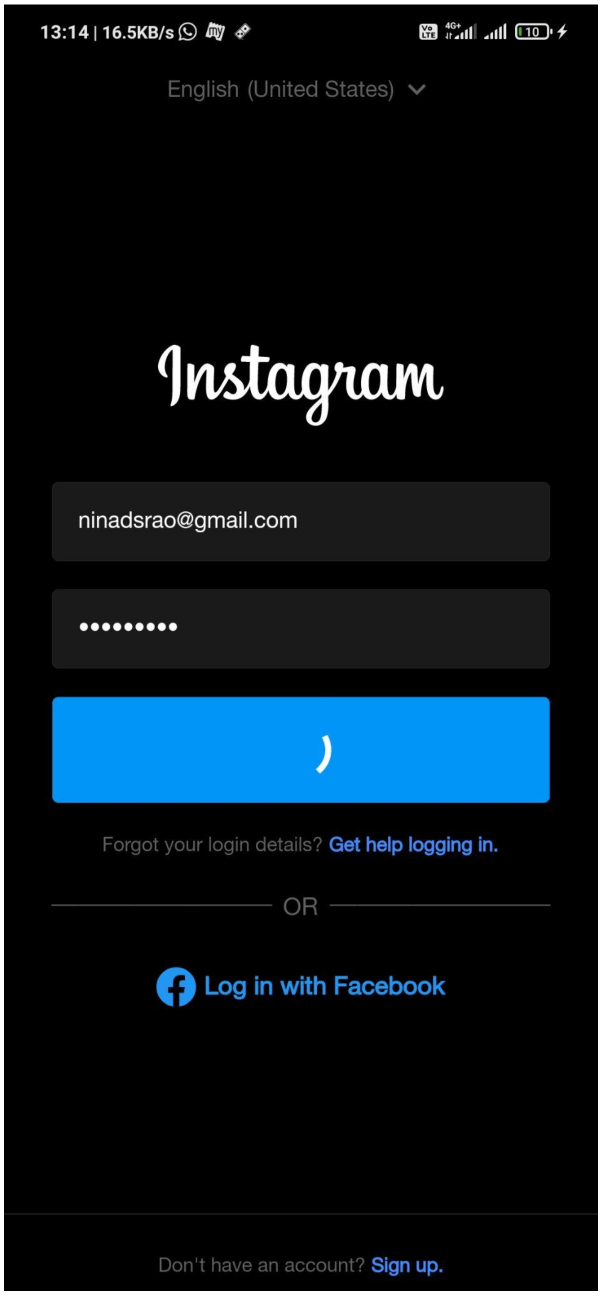
Log In

Forgot your login details? Get help logging in.

OR

Log in with Facebook

The above screens are the **Login Screen** of my application. The first screen is the normal Login Screen where it has two input fields for Email Address and Password. The second screen appears like this when the user doesn't enter the details or inputs in the **TextField** of Email Address or Password. This is done by validating the fields using the global form key.



Once the user enters the correct details, the Log In button changes to the **CircularProgressIndicator** which is used as a widget over here. After that, the user is logged in.

Conclusion: Hence, we understood how to create an interactive Form using a form widget that has been used in making the Login Screen of our application (Instagram).