EXPERIMENT NO: - 04

Name:- Swaraj Patil
Class:- D15A
Roll:No: - 39

AIM: - To create an interactive Form using form widget.

Theory: -

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

- While creating a form in Flutter, the **Form widget** is essential as it acts as a container for grouping multiple form fields and managing validation.

- A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.

- The **TextFormField widget** is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.

- To enhance the appearance and usability of input fields, **InputDecoration** is used, allowing customization of labels, icons, borders, and hint text.

- Validation plays a crucial role in forms, and the **validator property** within **TextFormField** ensures user input meets specific criteria before submission.

- Different types of input require appropriate **keyboard types**, such as TextInputType.number for numeric fields or TextInputType.emailAddress for email fields.

- Proper **state management** is needed to store and retrieve user input, ensuring the form data is processed correctly.

- A **submit button** is necessary to trigger form validation and submit the collected data for further processing.

Some Properties of Form Widget

- **key:** A GlobalKey that uniquely identifies the Form. You can use this

key to interact with the form, such as validating, resetting, or saving its state.

- **child:** The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- **autovalidateMode:** An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget

- **validate():** This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- **save():** This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- **reset():** Resets the form to its initial state, clearing any user-entered data.
- **currentState:** A getter that returns the current FormState associated with the Form.

Code: -

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


class OTPVerificationScreen extends
StatefulWidget {
  final String phoneNumber;


  const
OTPVerificationScreen({super.key,
required this.phoneNumber});


  @override
  _OTPVerificationScreenState
createState() =>
_OTPVerificationScreenState();
}


class _OTPVerificationScreenState
extends State<OTPVerificationScreen>
{
  final Color _primaryColor = const
Color(0xFF0088CC);
  final List<TextEditingController>
_otpControllers = List.generate(5,
(index) => TextEditingController());
  final List<FocusNode>
_otpFocusNodes = List.generate(5,
(index) => FocusNode());

  @override
  void dispose() {
    for (var controller in
_otpControllers) {
      controller.dispose();
    }
    for (var node in _otpFocusNodes)
{
      node.dispose();
    }
    super.dispose();
  }


  @override
  Widget build(BuildContext context)
{
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        elevation: 0,
        backgroundColor:
Colors.white,
        leading: IconButton(
          icon: const
Icon(Icons.arrow_back, color:
Colors.black),
          onPressed: () =>
```

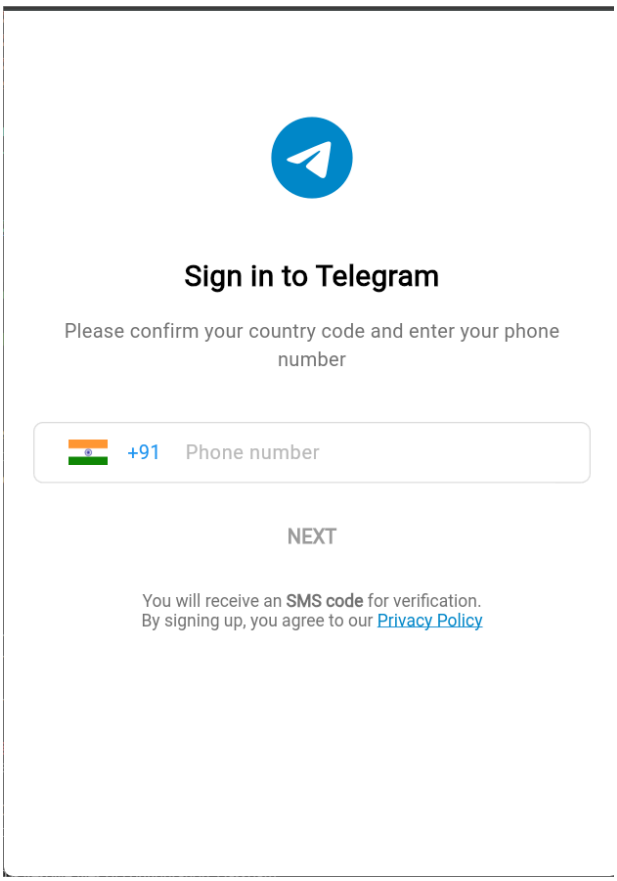
```
Navigator.pop(context),
        ),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal:
24),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              const SizedBox(height:
40),
              Text(
                'Verify your
number',
                style: TextStyle(
                  fontSize: 24,
                  fontWeight:
FontWeight.w600,
                  color:
Colors.grey[800],
                ),
              ),
              const SizedBox(height:
16),
              RichText(
                text: TextSpan(
                  style:
TextStyle(color: Colors.grey[600],
fontSize: 16),
                  children: [
                    const
TextSpan(text: 'We have sent a code
to '),
                    TextSpan(
                      text:
widget.phoneNumber,
                      style: const
TextStyle(fontWeight:
FontWeight.bold),
                    ),
                    const
TextSpan(text: '. Enter it below to
continue.'),
                  ],
                ),
              ),
              const SizedBox(height:
40),
              Row(
                mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                children:
List.generate(5, (index) {
                  return SizedBox(
```

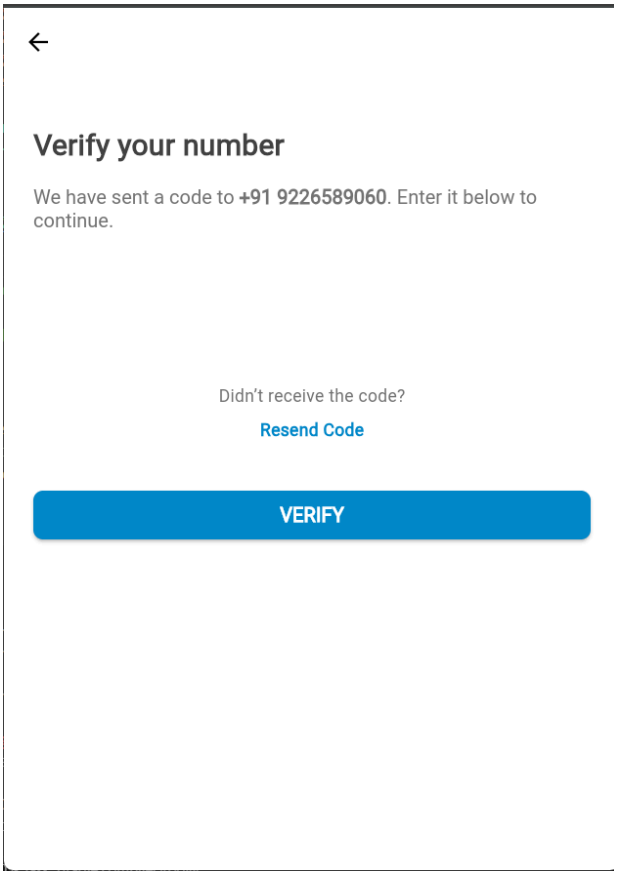
```
                    width: 50,
                    child:
TextField(
                      controller:
_otpControllers[index],
                      focusNode:
_otpFocusNodes[index],
                      textAlign:
TextAlign.center,
                      keyboardType:
TextInputType.number,
                      maxLength: 1,
                      style: const
TextStyle(fontSize: 24),
                      decoration:
InputDecoration(
                        counterText:
'',
                        border:
OutlineInputBorder(
borderRadius:
BorderRadius.circular(8),
borderSide: BorderSide(color:
Colors.grey[300]!),
                        ),
focusedBorder: OutlineInputBorder(
borderRadius:
BorderRadius.circular(8),
borderSide: BorderSide(color:
_primaryColor),
                        ),
                      ),
                      onChanged:
(value) {
                        if
(value.isNotEmpty && index < 4) {
_otpFocusNodes[index +
1].requestFocus();
                        } else if
(value.isEmpty && index > 0) {
_otpFocusNodes[index -
1].requestFocus();
                        }
                      },
                    ),
                  );
                }),
              ),
              const SizedBox(height:
24),
              Center(
                child: Text(
```

```
                'Didn't receive the code?',
                style: TextStyle(color: Colors.grey[600], fontSize: 14),
              ),
            ),
            const SizedBox(height: 8),
            Center(
              child: GestureDetector(
                onTap: () {
                  // Handle resend OTP logic
                },
                child: Text(
                  'Resend Code',
                  style: TextStyle(
                    color: _primaryColor,
                    fontSize: 14,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
            const SizedBox(height: 40),
            SizedBox(
              width: double.infinity,
              child: MaterialButton(
                height: 48,
                color: _primaryColor,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(8),
                ),
                onPressed: () {
                  // Handle OTP verification logic
                  String otp = _otpControllers.map((controller) => controller.text).join();
                  if (otp.length == 5) {
                    // Verify OTP
                  }
                },
                child: const Text(
```

```
                    'VERIFY',
                    style:
TextStyle(
                      color:
Colors.white,
                      fontSize: 16,
                      fontWeight:
FontWeight.bold,
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

OUTPUT:


Sign in to Telegram
Please confirm your country code and enter your phone number
+91
Phone number
NEXT
You will receive an SMS code for verification.
By signing up, you agree to our Privacy Policy


Verify your number
We have sent a code to +91 9226589060. Enter it below to continue.
Didn't receive the code?
Resend Code
VERIFY