Experiment No:04

Aim: To create an interactive Form using form widget.

- Enhancing User Interaction with Forms in Flutter Applications

Forms play a crucial role in collecting user input and facilitating interactions within Flutter applications. Leveraging the Form widget effectively is essential for creating intuitive and functional user interfaces. Here's a revised overview with updated language and structure:

1. Understanding Form Widgets:

- Definition: The Form widget serves as a container for organizing and managing multiple input fields within a Flutter application.

- State Management: It holds the state of the form and provides methods for validation and submission, ensuring seamless interaction with users.

2. Constructing User-friendly Forms:

- Form Creation: Integrate form fields within a Form widget to establish a structured and interactive form layout.

- Global Key Integration: Employ the GlobalKey<FormState> to uniquely identify and access the form's state, enabling validation and submission functionality.

3. Utilizing Diverse Form Fields:

- Field Varieties: Employ various form fields like TextFormField, DropdownButtonFormField, etc., to capture diverse types of user input.

- Configuration Requirements: Associate each form field with a controller for controlled input management and a validator function to uphold data integrity.

4. Validating User Input:

- Purposeful Validation: Ensure user input meets predefined criteria before submission to maintain data accuracy and consistency.

- Implementation Approach: Define validation logic using the validator property of form fields. Validators return error messages if validation fails or null for valid input.

5. Efficient Form Submission Handling:

- Submission Triggering: Initiate form submission through user interactions, typically via a dedicated submit button or similar action.

- Submission Workflow: Validate the form using the validate method of the FormState within the submission handler. Proceed with submission logic only if the form passes validation checks.

6. Error Management Strategies:

- User Guidance: Facilitate user error correction by displaying informative error messages when form validation fails.

- Error Presentation:Choose between displaying errors below individual form fields or presenting a consolidated error message at the form's top for improved user experience.

7. Form Maintenance Practices:

- Resource Cleanup:Dispose of form controllers and associated resources within the dispose method of the State object to prevent memory leaks and optimize performance.

8.Exploring Additional Features:

- Advanced Functionality: Explore Flutter's rich ecosystem of widgets and utilities to enhance form interactions further. These include InputDecoration for customizing form field appearance, FocusNode for managing field focus, and SnackBar for providing user feedback.

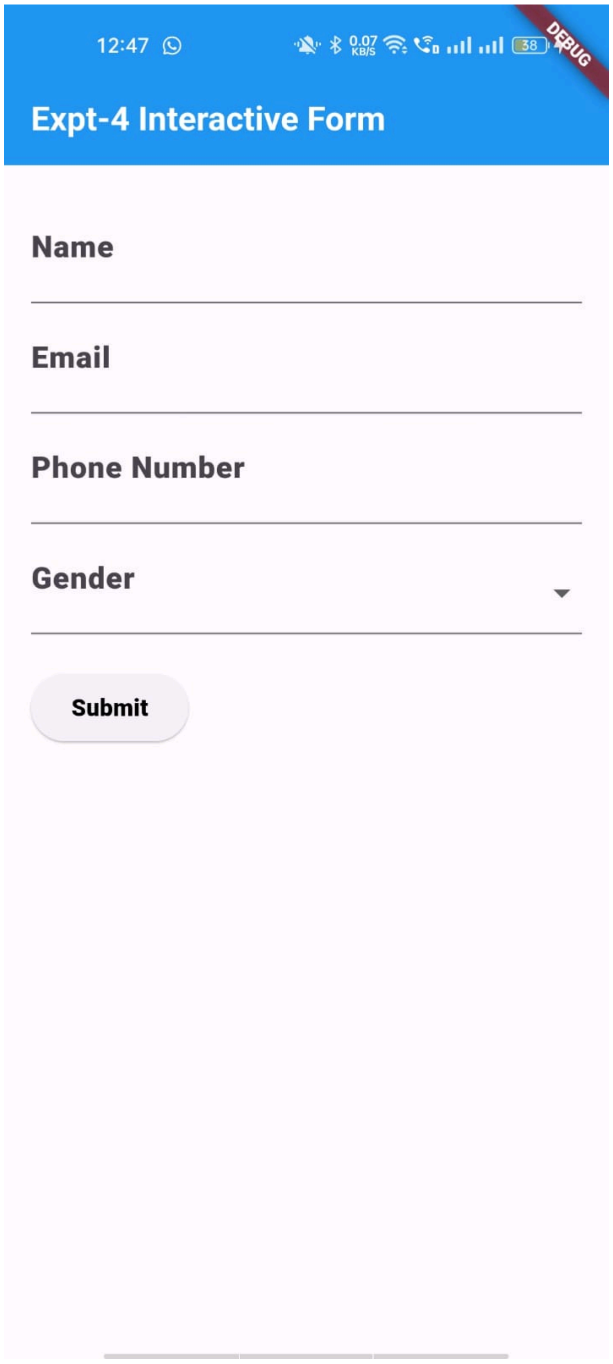
By implementing these principles and practices, Flutter developers can create forms that not only facilitate seamless user interactions but also enhance the overall user experience of their applications.

Code:

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


void main() {

  runApp(MyApp());

}


class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Interactive Form Example',

      theme: ThemeData(

        primaryColor: Colors.blue, // Change primary color of the app

        appBarTheme: AppBarTheme(

          titleTextStyle: TextStyle(

            fontSize: 20.0,

            fontWeight: FontWeight.bold,

            color: Colors.white, // Change text color of the AppBar title

          ),

        ),

      ),

      home: MyForm(),

    );

  }

}



class MyForm extends StatefulWidget {

  @override

  _MyFormState createState() => _MyFormState();

}
```

```
class _MyFormState extends State<MyForm> {

  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  TextEditingController _nameController = TextEditingController();

  TextEditingController _emailController = TextEditingController();

  TextEditingController _phoneNumberController = TextEditingController();
// Changed variable name

  String? _gender;



  @override

  void dispose() {

    _nameController.dispose();

    _emailController.dispose();

    _phoneNumberController.dispose(); // Updated variable name

    super.dispose();

  }



  void _submitForm() {

    if (_formKey.currentState != null &&
_formKey.currentState!.validate()) {

      // If the form is valid, perform actions like saving to a database
or API

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text('Form is validated and submitted!')),

      );
    }

  }



  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Expt-4 Interactive Form'),

        backgroundColor: Colors.blue, // Changed background color of the
AppBar

      ),

      body: Padding(

        padding: EdgeInsets.all(16.0),

        child: Form(

          key: _formKey,

          child: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            children: <Widget>[

              TextFormField(

                controller: _nameController,

                decoration: InputDecoration(

                  labelText: 'Name',

                  labelStyle: TextStyle(fontSize: 18), // Increased font
size of label
```

```
        ),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter your name';
          }
          return null;
        },
      ),
      TextFormField(
        controller: _emailController,
        decoration: InputDecoration(
          labelText: 'Email',
          labelStyle: TextStyle(fontSize: 18), // Increased font
size of label
        ),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter your email';
          }
          if (!value.contains('@')) {
            return 'Please enter a valid email';
          }
          return null;
        },
              ),
              TextFormField(
                controller: _phoneNumberController, // Updated variable
name
                decoration: InputDecoration(
                  labelText: 'Phone Number', // Changed field label
                  labelStyle: TextStyle(fontSize: 18), // Increased font
size of label
                ),
                keyboardType: TextInputType.phone,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your phone number'; // Updated
validation message
                  }
                  if (value.length != 10) {
                    return 'Please enter a valid 10-digit phone number';
// Updated validation message
                  }
                  return null;
                },
              ),
              DropdownButtonFormField<String>(
                decoration: InputDecoration(
                  labelText: 'Gender',
                  labelStyle: TextStyle(fontSize: 18), // Increased font size of label
                ),
                value: _gender,
                items: ['Male', 'Female', 'Other']
                    .map((gender) => DropdownMenuItem<String>(
                          value: gender,
                          child: Text(gender),
                        ))
                    .toList(),
                onChanged: (value) {
                  setState(() {
                    _gender = value;
                  });
                },
                validator: (value) {
                  if (value == null) {
                    return 'Please select your gender';
                  }
                  return null;
                },
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: _submitForm,

                child: Text(

                  'Submit',

                  style: TextStyle(color: Colors.black), // Changed text color of button

                ),

              ),

            ],

          ),

        ),

      ),

    );

  }

}
```



12:47

DEBUG

Expt-4 Interactive Form

Name

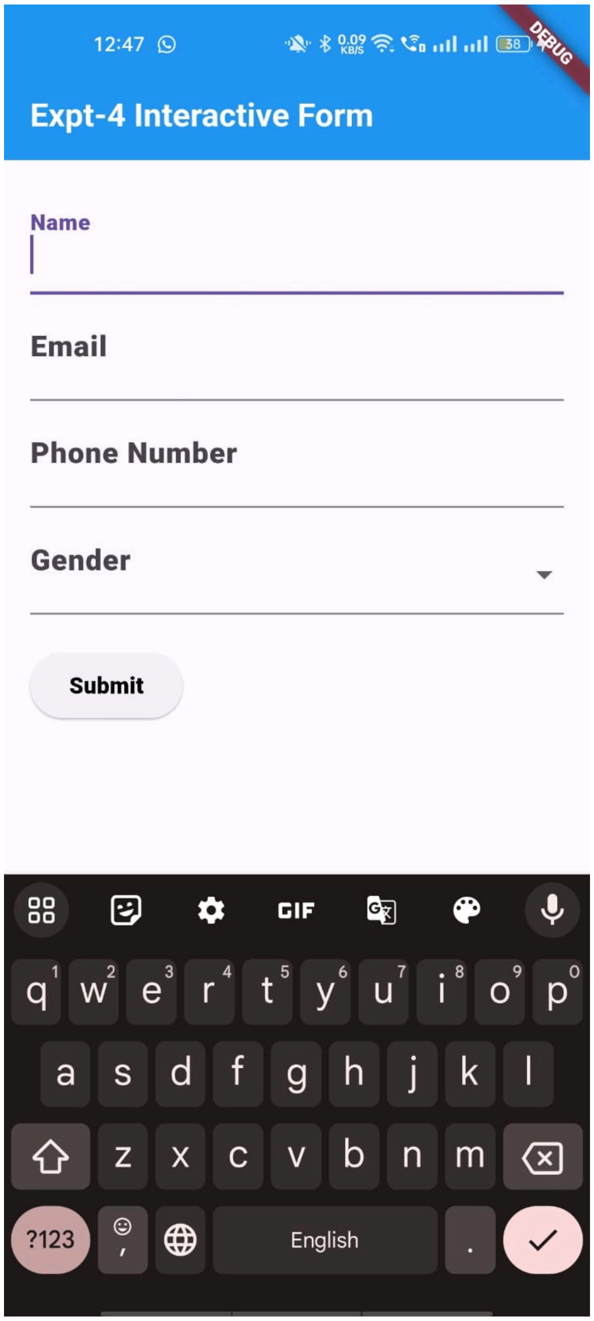
Email

Phone Number

Gender

Submit

Output:



Expt-4 Interactive Form

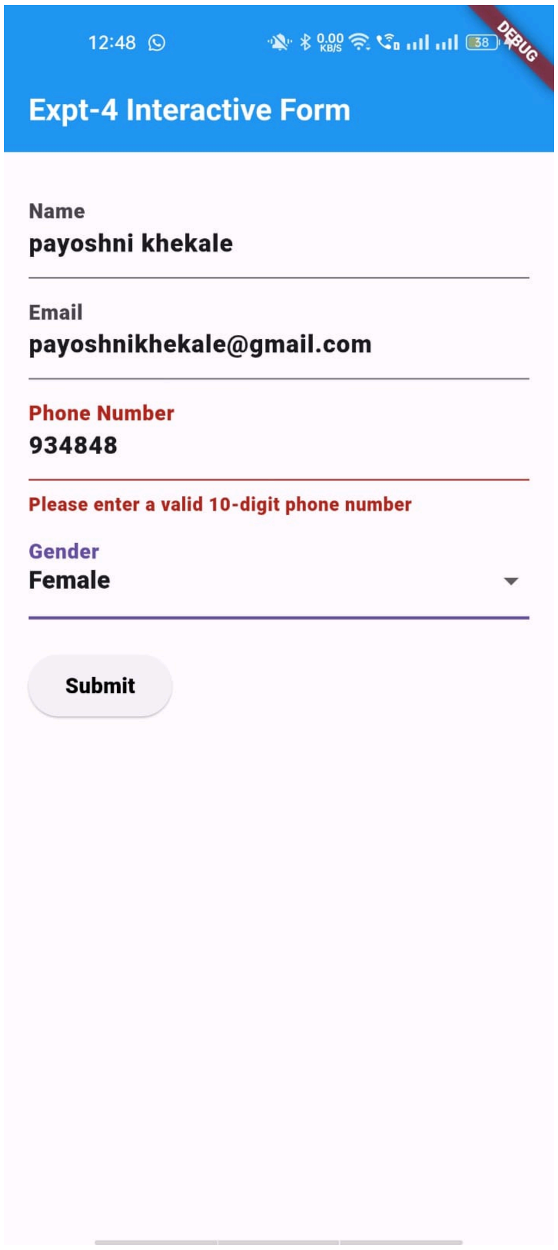
Name

Email

Phone Number

Gender

Submit


12:48

DEBUG

Expt-4 Interactive Form

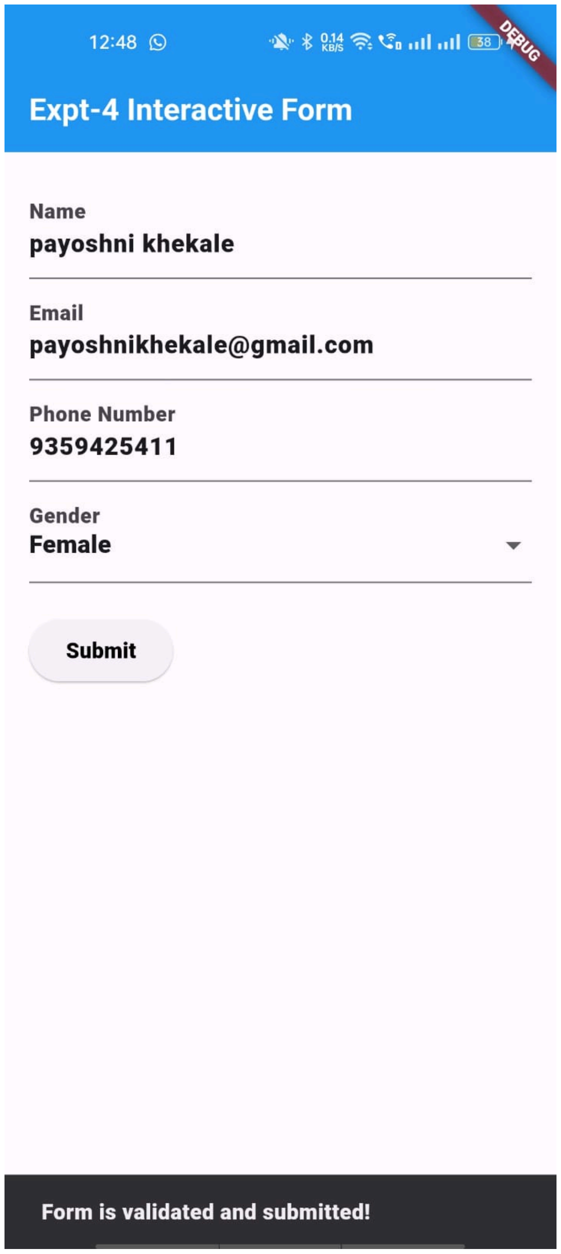
Name
payoshni khekale

Email
payoshnikhekale@gmail.com

Phone Number
934848

Please enter a valid 10-digit phone number

Gender
Female


12:48

DEBUG

Expt-4 Interactive Form

Name
payoshni khekale

Email
payoshnikhekale@gmail.com

Phone Number
9359425411

Gender
Female

Form is validated and submitted!

Conclusion: I have successfully created an interactive Form using form widget in Flutter.