# EXP 4

**Aim:** To create an interactive Form using form widget

**Theory:**

A Form Widget is a user interface element used to collect input from users. Forms are essential in web and application development, as they allow users to enter data such as login credentials, personal details, or feedback. Interactive forms enhance user experience by providing validation, responsiveness, and dynamic interactions.

Key Concepts in Interactive Forms

1. Form Widgets – These are components like text fields, checkboxes, radio buttons, dropdowns, and buttons that allow users to input data.
2. Validation – Ensuring that user input meets specific criteria (e.g., email format, required fields).
3. State Management – Forms can store data dynamically and update based on user interactions.
4. Event Handling – Capturing user actions like clicks, typing, or selections to trigger specific responses.
5. UI/UX Considerations – Forms should be visually appealing, easy to navigate, and provide clear feedback.

Implementation Approach

- Create a form using a programming language or framework (HTML, React, Flutter, etc.).
- Integrate form widgets such as input fields, dropdowns, and buttons.
- Implement validation to ensure correct user input.
- Provide real-time feedback through messages or UI changes.
- Ensure responsiveness across devices for a better user experience.

**Code for New Journal Entry page:**

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

class NewEntryScreen extends StatefulWidget {
  @override
  _NewEntryScreenState createState() => _NewEntryScreenState();
}

class _NewEntryScreenState extends State<NewEntryScreen> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController _titleController = TextEditingController();
  final TextEditingController _contentController = TextEditingController();

  static const Color pastelPurple = Color(0xFFDAAAFF);
  static const Color pastelPink = Color(0xFFF35E74);
  static const Color textDark = Color(0xFF4A4A4A);

  Future<void> saveNewEntry() async {
    if (_formKey.currentState!.validate()) {
      try {
        FirebaseFirestore firestore = FirebaseFirestore.instance;
```

```
  await firestore.collection('journalEntries').add({
    'title': _titleController.text.trim(),
    'content': _contentController.text.trim(),
    'timestamp': FieldValue.serverTimestamp(),
  });

  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text('New Entry Saved!'),
      backgroundColor: Colors.pink.shade100,
    ),
  );

  _titleController.clear();
  _contentController.clear();
  Navigator.pop(context);
} catch (e) {
  print("Error saving new entry: $e");
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text('Failed to save the entry. Please try again.'),
```

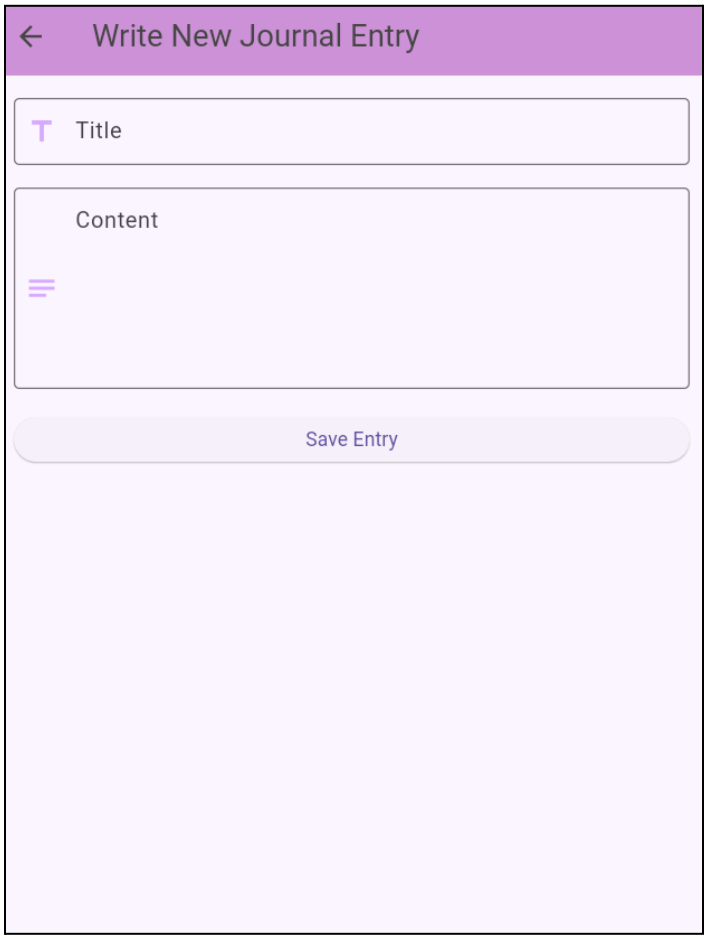
```
        backgroundColor: pastelPink,
      ),
    );
  }
 }
}

@override
Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
   title: Text("Write New Journal Entry", style: TextStyle(color: textDark)),
   backgroundColor: Colors.purple.shade200,
   iconTheme: IconThemeData(color: textDark),
  ),
  body: SingleChildScrollView(
   child: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Form(
     key: _formKey, // <--- New
     child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
```

```
children: [
  TextFormField(
    controller: _titleController,
    decoration: InputDecoration(
      labelText: 'Title',
      border: OutlineInputBorder(),
      prefixIcon: Icon(Icons.title, color: pastelPurple),
    ),
    validator: (value) {
      if (value == null || value.trim().isEmpty) {
        return 'Title cannot be empty';
      }
      return null;
    },
  ),
  SizedBox(height: 16),
  TextFormField(
    controller: _contentController,
    decoration: InputDecoration(
      labelText: 'Content',
      border: OutlineInputBorder(),
      alignLabelWithHint: true,
```

```
                prefixIcon: Icon(Icons.notes, color: pastelPurple),
              ),
              maxLines: 5,
              validator: (value) {
                if (value == null || value.trim().isEmpty) {
                  return 'Content cannot be empty';
                }
                return null;
              },
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: saveNewEntry,
              child: Text('Save Entry'),
            ),
          ],
        ),
      ),
    ),
  );
}}
```

**Output:**




**Conclusion:**

In this experiment, a form was created using the Form widget to allow users to add new journal entries by entering a title and content. The use of TextFormField with validation made the form interactive and user-friendly. This helped ensure proper input before saving the data to Firestore, improving both functionality and user experience.