**Experiment 4**

**Aim:** To create an interactive Form using a 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.

**Github Link:** [**https://github.com/sadneya145/Activibe.git**](https://github.com/sadneya145/Activibe.git)

**Code in event.dart:**

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

class EventModel {

String id;

String title;

String description;

String date;

String time;

String qrCodeUrl;

bool isJoined;

EventModel({

required this.id,

required this.title,

required this.description,

required this.date,

required this.time,

required this.qrCodeUrl,

this.isJoined = false,

});

factory EventModel.fromJson(Map<String, dynamic> json) {

return EventModel(

id: json['\_id'] ?? '',

title: json['title'] ?? '',

description: json['description'] ?? '',

date: json['date'] ?? '',

time: json['time'] ?? '',

qrCodeUrl: json['qrCodeUrl'] ?? '',

);

}

}

class EventPage extends StatefulWidget {

@override

\_EventPageState createState() => \_EventPageState();

}

class \_EventPageState extends State<EventPage> {

List<EventModel> events = [];

final TextEditingController titleController = TextEditingController();

final TextEditingController descriptionController = TextEditingController();

final TextEditingController dateController = TextEditingController();

final TextEditingController timeController = TextEditingController();

@override

void initState() {

super.initState();

fetchEvents();

}

Future<void> fetchEvents() async {

final response = await http.get(Uri.parse("http://localhost:5000/events"));

if (response.statusCode == 200) {

final data = json.decode(response.body);

print("Fetched Events Data: $data"); // Debugging

setState(() {

events = (data['events'] as List)

.map((e) => EventModel.fromJson(e))

.toList();

});

} else {

print("Failed to load events. Status code: ${response.statusCode}");

}

}

Future<void> addEvent() async {

String title = titleController.text.trim();

String description = descriptionController.text.trim();

String date = dateController.text.trim();

String time = timeController.text.trim();

if (title.isNotEmpty &&

description.isNotEmpty &&

date.isNotEmpty &&

time.isNotEmpty) {

final response = await http.post(

Uri.parse("http://localhost:5000/createEvent"),

headers: {"Content-Type": "application/json"},

body: json.encode({

"title": title,

"description": description,

"date": date,

"time": time,

}),

);

if (response.statusCode == 200) {

fetchEvents();

Navigator.of(context).pop();

}

}

}

Future<void> joinEvent(String eventId, String userEmail) async {

final response = await http.post(

Uri.parse("http://localhost:5000/registerEvent/$eventId"),

headers: {"Content-Type": "application/json"},

body: json.encode({"email": userEmail}),

);

if (response.statusCode == 200) {

setState(() {

events.firstWhere((e) => e.id == eventId).isJoined = true;

});

}

}

void showAddEventDialog() {

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("Add New Event"),

content: Column(

mainAxisSize: MainAxisSize.min,

children: [

TextField(

controller: titleController,

decoration: InputDecoration(labelText: "Event Title")),

TextField(

controller: descriptionController,

decoration: InputDecoration(labelText: "Event Description")),

InkWell(

onTap: () async {

DateTime? pickedDate = await showDatePicker(

context: context,

initialDate: DateTime.now(),

firstDate: DateTime.now(),

lastDate: DateTime(2100),

);

if (pickedDate != null) {

String formattedDate =

DateFormat('yyyy-MM-dd').format(pickedDate);

dateController.text = formattedDate;

}

},

child: IgnorePointer(

child: TextField(

controller: dateController,

decoration: InputDecoration(

labelText: "Event Date", hintText: "Select Date"),

),

),

),

InkWell(

onTap: () async {

TimeOfDay? pickedTime = await showTimePicker(

context: context,

initialTime: TimeOfDay.now(),

);

if (pickedTime != null) {

String formattedTime = pickedTime.format(context);

timeController.text = formattedTime;

}

},

child: IgnorePointer(

child: TextField(

controller: timeController,

decoration: InputDecoration(

labelText: "Event Time", hintText: "Select Time"),

),

),

),

],

),

actions: [

TextButton(

onPressed: () => Navigator.of(context).pop(),

child: Text("Cancel")),

ElevatedButton(onPressed: addEvent, child: Text("Add Event")),

],

);

},

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Activibe")),

body: ListView.builder(

itemCount: events.length,

itemBuilder: (context, index) {

return Card(

margin: EdgeInsets.symmetric(horizontal: 16, vertical: 8),

elevation: 4,

shape:

RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),

child: Padding(

padding: EdgeInsets.all(12),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

events[index].title,

style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),

),

SizedBox(height: 6),

Text(

events[index].description,

style: TextStyle(fontSize: 14, color: Colors.grey[700]),

),

SizedBox(height: 10),

Row(

children: [

Icon(Icons.calendar\_today,

size: 16, color: Colors.blueGrey),

SizedBox(width: 4),

Text(

"Date: ${events[index].date}",

style: TextStyle(

fontSize: 14, fontWeight: FontWeight.w500),

),

],

),

SizedBox(height: 4),

Row(

children: [

Icon(Icons.access\_time, size: 16, color: Colors.blueGrey),

SizedBox(width: 4),

Text(

"Time: ${events[index].time}",

style: TextStyle(

fontSize: 14, fontWeight: FontWeight.w500),

),

],

),

SizedBox(height: 10),

Center(

child: ClipRRect(

borderRadius: BorderRadius.circular(8),

child: Image.network(

events[index].qrCodeUrl,

width: 120, // Adjusted width

height: 120, // Adjusted height

fit: BoxFit.cover,

),

),

),

SizedBox(height: 10),

Align(

alignment: Alignment.centerRight,

child: ElevatedButton(

onPressed: () =>

joinEvent(events[index].id, "test@example.com"),

child: Text(events[index].isJoined ? "Joined" : "Join"),

style: ElevatedButton.styleFrom(

padding:

EdgeInsets.symmetric(horizontal: 24, vertical: 10),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(8)),

backgroundColor:

events[index].isJoined ? Colors.green : Colors.blue,

),

),

),

],

),

),

);

},

),

floatingActionButton: FloatingActionButton(

onPressed: showAddEventDialog,

child: Icon(Icons.add),

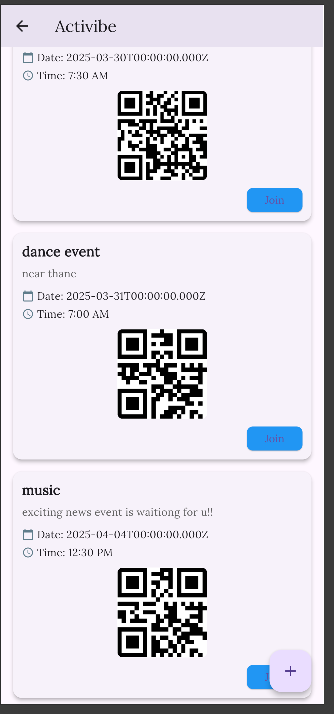
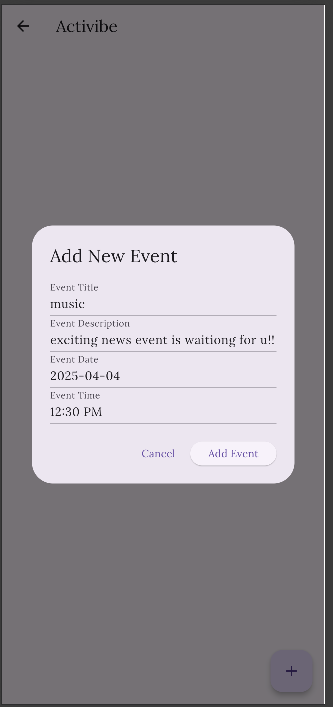
),

);

}

}

**Output:**



## **Conclusion**

Interactive forms play a crucial role in modern applications by enhancing usability and data collection. Using form widgets with validation and event handling ensures a smooth user experience. Proper design and implementation of interactive forms lead to better engagement, accurate data submission, and improved overall functionality of an application.