Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

Experiment 04

Aim: To create an interactive Form using form widget

Theory:

Flutter forms use Form, TextFormField, and GlobalKey<FormState> for validation and user input handling. They are essential for collecting structured user data.

Reference

1. https://docs.flutter.dev/cookbook/forms/validation
2. https://www.javatpoint.com/flutter-forms
3. Example https://codelabs.developers.google.com/codelabs/first-flutter-app-pt2#6
4. https://flutterbyexample.com/lesson/stateful-widget-lifecycle

Steps

- Create a Form with a GlobalKey
- Add a TextFormField with validation logic
- Create a button to validate and submit the form
- I am using a Form widget with GlobalKey<FormState>.
- Validation, conditional fields (e.g., flat number for residents), and user input are handled properly.

Source code is as follows:

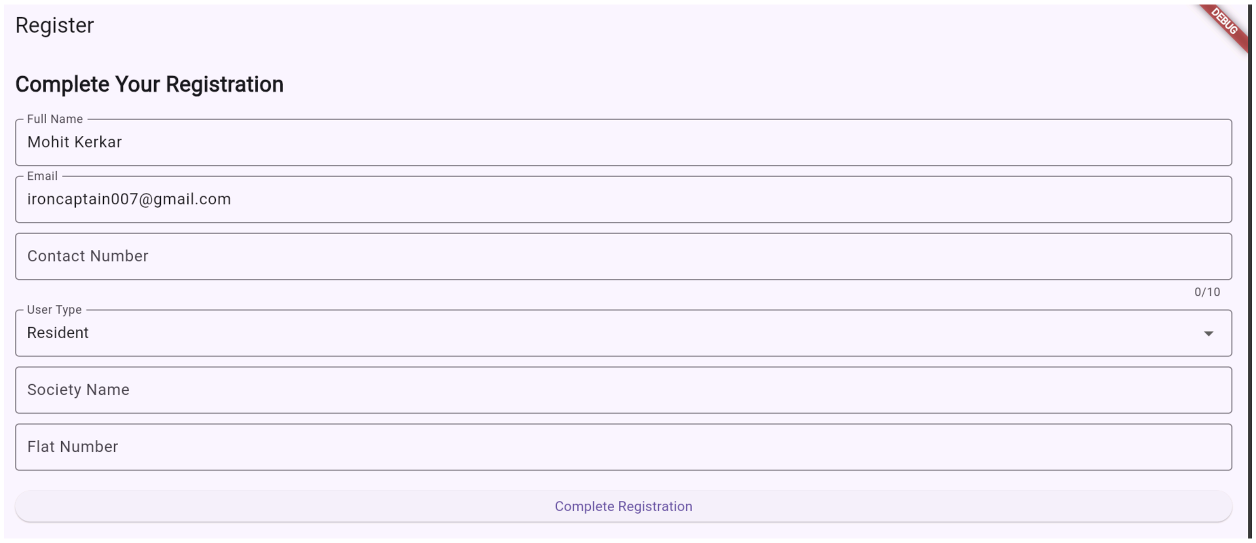
```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'society_list_screen.dart';

class RegisterScreen extends StatefulWidget {
  final User user;

  const RegisterScreen({Key? key, required this.user}) : super(key: key);

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

class _RegisterScreenState extends State<RegisterScreen> {
  final FirebaseFirestore _db = FirebaseFirestore.instance;
  final TextEditingController _contactController = TextEditingController();
  final TextEditingController _societyController = TextEditingController();
  final TextEditingController _flatNumberController = TextEditingController();
  final _formKey = GlobalKey<FormState>();

  String _userType = "Resident"; // Default user type
  bool _isResident = true;
  bool _isLoading = false;

  Future<void> _registerUser() async {
  if (!_formKey.currentState!.validate()) return;
  setState(() => _isLoading = true);
  try {
    String societyId = ""; // Initialize society ID

    // If user is a Manager, create a society
    if (_userType == "Manager") {
      DocumentReference societyRef = await _db.collection("societies").add({
        "name": _societyController.text.trim(),
        "location": "Not specified", // Default location, can be updated later
        "createdBy": widget.user.uid, // Store Manager's UID
      });
      societyId = societyRef.id; // Save society ID
    }

    // Store user details in Firestore
    await _db.collection("users").doc(widget.user.uid).set({
      "uid": widget.user.uid,
      "name": widget.user.displayName ?? "Unknown",
      "email": widget.user.email,
      "contactNumber": _contactController.text.trim(),
      "userType": _userType,
      "societyName": _societyController.text.trim(),
      "societyId": _userType == "Manager" ? societyId : "", // Assign society to Managers only
      "flatNumber": _isResident ? _flatNumberController.text.trim() : "N/A",
      "photoUrl": widget.user.photoURL ?? "",
      "createdAt": FieldValue.serverTimestamp(),
    });

    Navigator.pushReplacement(
  context,
  MaterialPageRoute(
    builder: (context) => SocietyListScreen(
      userId: widget.user.uid,
      userType: _userType, // Ensure correct value is passed
    ),
  ),
);


    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("✅ Registration Successful!")),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("❌ Error: ${e.toString()}")),
    );
  } finally {
    setState(() => _isLoading = false);
  }
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text("Register")),
    body: GestureDetector(
      onTap: () => FocusScope.of(context).unfocus(),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text(
                  "Complete Your Registration",
                  style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
                ),
                const SizedBox(height: 20),

                // Full Name
                TextFormField(
                  initialValue: widget.user.displayName ?? "Unknown",
                  readOnly: true,
                  decoration: const InputDecoration(
                    labelText: "Full Name",
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 10),

                // Email
                TextFormField(
                  initialValue: widget.user.email,
                  readOnly: true,
                  decoration: const InputDecoration(
                    labelText: "Email",
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 10),

                // Contact Number
                TextFormField(
                  controller: _contactController,
                  keyboardType: TextInputType.phone,
                  decoration: const InputDecoration(
                    labelText: "Contact Number",
                    border: OutlineInputBorder(),
                  ),
                  maxLength: 10,
                  validator: (value) {
                    if (value!.isEmpty) return "Enter Contact Number";
                    // if (!RegExp(r'^[0-9]{10}\$').hasMatch(value)) return "Enter a valid 10-digit
        number";
                    return null;
                  },
                ),
                const SizedBox(height: 10),

                // User Type Dropdown
                DropdownButtonFormField<String>(
                  value: _userType,
                  items: ["Resident", "Manager"].map((type) {
                    return DropdownMenuItem(value: type, child: Text(type));
                  }).toList(),
                  onChanged: (value) {
                    setState(() {
                      _userType = value!;
                      _isResident = _userType == "Resident";
                    });
                  },
                  decoration: const InputDecoration(
                    labelText: "User Type",
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 10),

                // Society Name
                TextFormField(
                  controller: _societyController,
                  decoration: const InputDecoration(
                    labelText: "Society Name",
                    border: OutlineInputBorder(),
                  ),
                  validator: (value) => value!.isEmpty ? "Enter Society Name" : null,
                ),
                const SizedBox(height: 10),

                // Flat Number (Only for Residents)
                if (_isResident)
                  TextFormField(
                    controller: _flatNumberController,
                    decoration: const InputDecoration(
                      labelText: "Flat Number",
                      border: OutlineInputBorder(),
                    ),
                    validator: (value) => value!.isEmpty ? "Enter Flat Number" : null,
                  ),
                const SizedBox(height: 20),
                // Register Button
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton(
                    onPressed: _isLoading ? null : _registerUser,
                    child: _isLoading
                        ? const CircularProgressIndicator(color: Colors.white)
                        : const Text("Complete Registration"),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    ),
  );
 }
}
```

In registration form:

- A Form widget is wrapped around the entire input structure to maintain a unified validation context.
- A GlobalKey<FormState> is used to validate and manage form state globally when the form is submitted.
- Various TextFormField widgets are used to collect user inputs like name, email, society, etc.
- Conditional rendering is implemented: for instance, the Flat Number field appears only if the selected user type is 'Resident'. This is achieved using Flutter’s stateful widget logic and conditional checks.
- Validation logic is applied to fields to ensure correctness of data (e.g., checking if email is valid, or mandatory fields are not left blank).

Name: Mohit Kerkar **Div: D15C** **Roll No: 23**


Register

Complete Your Registration

Full Name: Mohit Kerkar

Email: ironcaptain007@gmail.com

Contact Number

0/10

User Type: Resident

Society Name

Flat Number

Conclusion:
We built a dynamic, validated registration form that efficiently captures resident and manager details in the Society app.