EXPERIMENT 4

Aim : To create an interactive Form using form widget

Theory:

- Flutter is Google's UI toolkit for crafting beautiful, natively compiled iOS and Android apps from a single code base. To build any application we start with widgets – The building block of flutter applications.
- Widgets describe what their view should look like given their current configuration and state. It includes a text widget, row widget, column widget, container widget, and many more.
- Widgets: Each element on a screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of an app is a tree of widgets.

- Creating a login page with validation in Flutter involves several key steps, from designing the user interface to implementing validation logic. Below is a theoretical outline of how you can approach this task:

1. Designing the User Interface (UI):

Decide on the layout of the login page, including the placement of username and password fields, and the login button. Consider using Flutter's Material design widgets for a consistent look and feel.
Use TextFormField widgets for input fields to allow users to enter their username and password securely.
Add any additional UI elements such as error messages for validation feedback.

2. State Management:

Decide on the state management approach. For a simple login page, you can use the StatefulWidget to manage the state of the page.
Create state variables to hold the values of the username and password

fields, as well as variables to track the validation status of these fields.

3. Validation Logic:
Implement validation logic for the username and password fields.
Use Flutter's form validation capabilities, such as the validator parameter in TextFormField, to validate user input.
Provide feedback to the user if the input is invalid, such as displaying error messages below the input fields.

4. Handling User Input:
Implement logic to handle user input, such as updating the state variables when the user types in the input fields.

5. Handling Form Submission:
Implement logic to handle form submission when the user taps the login button.
Validate the entire form to ensure all input fields are valid before proceeding with login.
If the form is valid, perform any necessary authentication logic, such as verifying the username and password against a database or an API.

In Flutter, a GlobalKey is a unique identifier for widgets. It's used to maintain state or reference specific widgets across the widget tree. GlobalKey provides a way to access and interact with widgets that are not directly in the widget hierarchy where they are defined.
Here are some key points about GlobalKey:

1. Unique Identifier: GlobalKey is used to uniquely identify a widget throughout the application.
2. Accessing Widget State: It allows you to access the state of stateful widgets, such as retrieving the state of a StatefulWidget or triggering methods within

Code:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:demo_alumnet/services/auth/auth_service.dart';
import 'package:demo_alumnet/widgets/helper_functions.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:demo_alumnet/components/widgets.dart';
import 'package:provider/provider.dart';
```

```
class RegisterPage extends StatefulWidget {
  final void Function()? onTap;

  const RegisterPage({Key? key, required this.onTap});

  @override
  State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  // Form key for validation
  final _formKey = GlobalKey<FormState>();

  // Text controllers
  final nameController = TextEditingController();
  final emailController = TextEditingController();
  final passwordController = TextEditingController();
  final confirmPasswordController = TextEditingController();
  final bioController = TextEditingController();

  // Sign up user
  Future<void> signUp() async {
    // Validate form
    if (!_formKey.currentState!.validate()) {
      return;
    }

    // Get the auth service
    final authService = Provider.of<AuthService>(context, listen: false);

    try {
      // Sign up with email and password
      UserCredential userCredential = await authService.signUpWithEmailAndPassword(
        emailController.text,
        passwordController.text,
        nameController.text,
        bioController.text,
      );

      // Create user document in Cloud Firestore
      await createUserDocument(userCredential);

      // Show success message in SnackBar
```

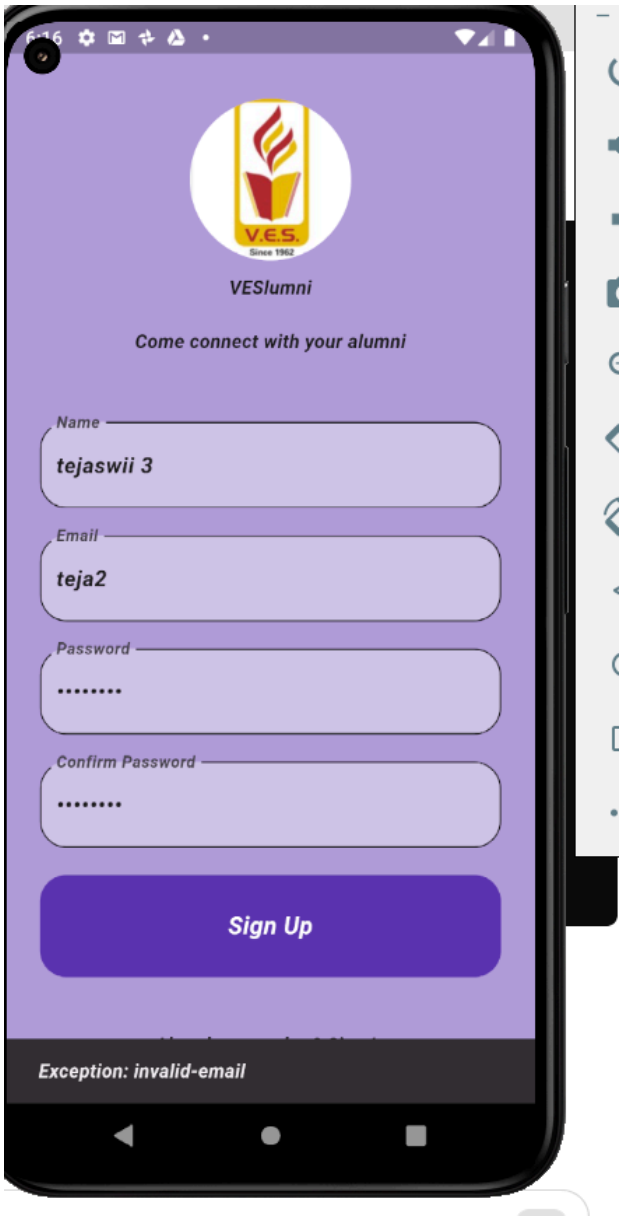
```
    showSnackBar("Registration successful");
  } catch (e) {
    // Show error message in SnackBar
    showSnackBar(e.toString());
  }
}

// Create user document and store it in Cloud Firestore
Future<void> createUserDocument(UserCredential? userCredential) async {
  if (userCredential != null && userCredential.user != null) {
    await FirebaseFirestore.instance
        .collection("Users")
        .doc(userCredential.user!.email)
        .set({
      'email': userCredential.user!.email,
      'username': nameController.text,
      'bio': "Enter your Bio",
    });
  }
}

// Show SnackBar with a message
void showSnackBar(String message) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.deepPurple[200],
    body: SafeArea(
      child: Center(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 25.0),
            child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
```

```
                children: [
                  // Circular logo image
                  CircleAvatar(
                    radius: 60,
                    backgroundImage: AssetImage('assets/Vesitlogo.png'),
                  ),
                  const SizedBox(
                    height: 10,
                  ),
                  const Text(
                    'VESlumni',
                    style: TextStyle(fontWeight: FontWeight.bold),
                  ),
                  const SizedBox(
                    height: 20,
                  ),
                  const Text(
                    'Come connect with your alumni',
                    style: TextStyle(fontWeight: FontWeight.w500),
                  ),
                  const SizedBox(
                    height: 50,
                  ),
                  // Name
                  MyTextField(
                    controller: nameController,
                    hintText: 'Enter your name',
                    labelText: 'Name',
                    obscureText: false,
                    validator: (value) {
                      if (value == null || value.isEmpty) {
                        return 'Name is required';
                      }
                      return null;
                    },
                  ),
                  const SizedBox(
                    height: 20,
                  ),
                  // Email
                  MyTextField(
                    controller: emailController,
                    hintText: 'Enter your email',
```

```
                    labelText: 'Email',
                    obscureText: false,
                    validator: (value) {
                      if (value == null || value.isEmpty) {
                        return 'Email is required';
                      }
                      if
(!RegExp(r'^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$').hasMatch(value)) {
                        return 'Enter a valid email address';
                      }
                      return null;
                    },
                  ),
                  const SizedBox(
                    height: 20,
                  ),
                  // Password
                  MyTextField(
                    controller: passwordController,
                    hintText: 'Password',
                    labelText: 'Password',
                    obscureText: true,
                    validator: (value) {
                      if (value == null || value.isEmpty) {
                        return 'Password is required';
                      }
                      return null;
                    },
                  ),
                  const SizedBox(
                    height: 20,
                  ),
                  // Confirm Password
                  MyTextField(
                    controller: confirmPasswordController,
                    hintText: 'Confirm Password',
                    labelText: 'Confirm Password',
                    obscureText: true,
                    validator: (value) {
                      if (value == null || value.isEmpty) {
                        return 'Confirm Password is required';
                      }
                      if (value != passwordController.text) {
```

```
                    return 'Passwords do not match';
                  }
                  return null;
                },
              ),
              const SizedBox(
                height: 20,
              ),
              // Sign up button
              MyCustomBtn(onTap: signUp, text: "Sign Up"),
              const SizedBox(
                height: 40,
              ),
              // Already a member? Sign In
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const Text('Already a member?'),
                  const SizedBox(
                    width: 4,
                  ),
                  GestureDetector(
                    onTap: widget.onTap,
                    child: const Text(
                      'Sign In',
                      style: TextStyle(
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    ),
  ),
  ),
  );
  }
}
```

Output:



Conclusion : Hence we have understood and studied about the form validation in flutter and implemented it in our application.