Name:Yajat H Bhasin **Div:D15B** **Roll No:5**

Experiment 4

Aim:To create an interactive Form using form widget

Theory:

- Form Widget: The Form widget is the container for form fields. It manages the form state and provides methods to validate, save, reset, and submit the form.

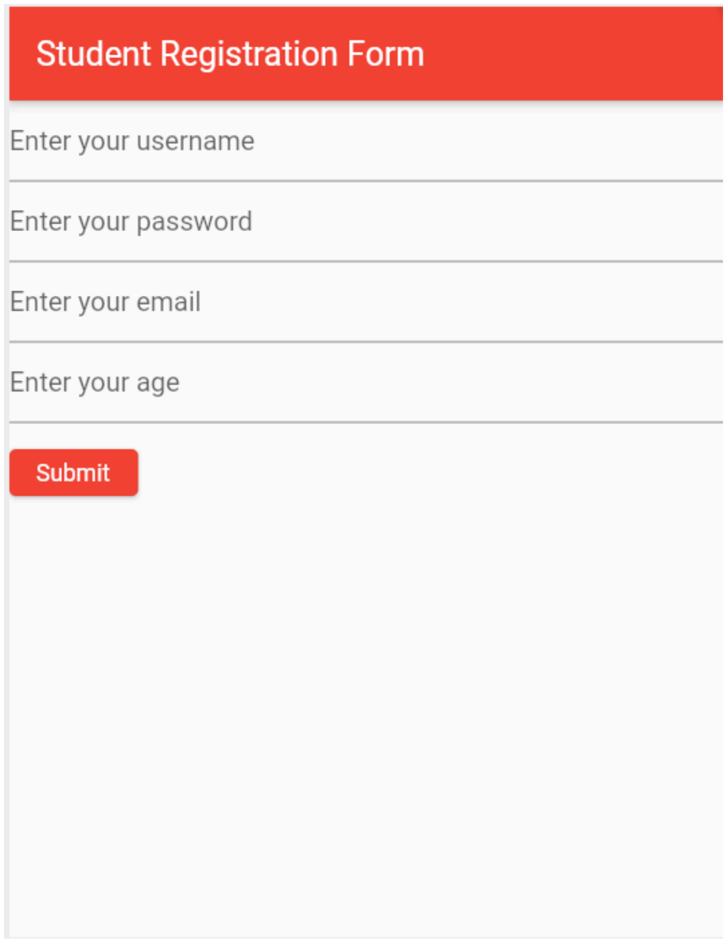
- Form Fields: Form fields are input fields where users can enter data. Flutter provides various types of form fields, such as TextFormField, DropdownButtonFormField, CheckboxFormField, RadioFormField, etc. Each form field has its own properties and behavior.

- Submission: Form submission involves taking the data entered by the user and processing it, such as sending it to a server, saving it locally, or performing other actions based on the application's logic. In Flutter, you can handle form submission by implementing a callback function that gets triggered when the form is submitted.

- Error Handling: Error handling in forms involves displaying error messages to the user when the data entered is invalid or when there are issues with form submission

Code:

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

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    const appTitle = 'Student Registration Form';

    return MaterialApp(
      title: appTitle,
      theme: ThemeData(primarySwatch: Colors.red),
      home: Scaffold(
        appBar: AppBar(title: const Text(appTitle)),
        body: MyCustomForm(),
      ),
    );
  }
}

class MyCustomForm extends StatefulWidget {
  const MyCustomForm({Key? key}) : super(key: key);

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

class MyCustomFormState extends State<MyCustomForm> {
  final _formKey = GlobalKey<FormState>();

  String _username = '';
  String _password = '';
  String _email = '';
  int _age = 0;

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          TextFormField(
            decoration: const InputDecoration(hintText: 'Enter your username'),
            validator: (value) => value!.isEmpty ? 'Please enter a username' : null,
            onSaved: (value) => _username = value!,
          ),
          TextFormField(
            decoration: const InputDecoration(hintText: 'Enter your password'),
            validator: (value) => value!.isEmpty ? 'Please enter a password' : null,
            obscureText: true,
            onSaved: (value) => _password = value!,
          ),
          TextFormField(
            decoration: const InputDecoration(hintText: 'Enter your email'),
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter an email';
              } else if (!value.contains('@')) {
                return 'Please enter a valid email';
              }
              return null;
            },
            keyboardType: TextInputType.emailAddress,
            onSaved: (value) => _email = value!,
          ),
          TextFormField(
            decoration: const InputDecoration(hintText: 'Enter your age'),
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter an age';
              } else if (int.tryParse(value) == null || int.parse(value) < 1) {
                return 'Please enter a valid age';
              }
              return null;
            },
            keyboardType: TextInputType.number,
            onSaved: (value) => _age = int.parse(value!),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 16),
                    child: ElevatedButton(
                      onPressed: () {
                        if (_formKey.currentState!.validate()) {
                          _formKey.currentState!.save();
                          showDialog(
                            context: context,
                            builder: (BuildContext context) => AlertDialog(
                              title: const Text('Form Data'),
                              content: SingleChildScrollView(
                                child: ListBody(
                                  children: <Widget>[
                                    Text('Username: $_username'),
                                    Text('Password: $_password'),
                                    Text('Email: $_email'),
                                    Text('Age: $_age'),
                                  ],
                                ),
                              ),
                              actions: <Widget>[
                                TextButton(
                                  onPressed: () => Navigator.of(context).pop(),
                                  child: const Text('Close'),
                                ),
                              ],
                            ),
                          );
                        }
                      },
                      child: const Text('Submit'),
                    ),
                  ),
                ],
              ),
            );
           }
          }
```



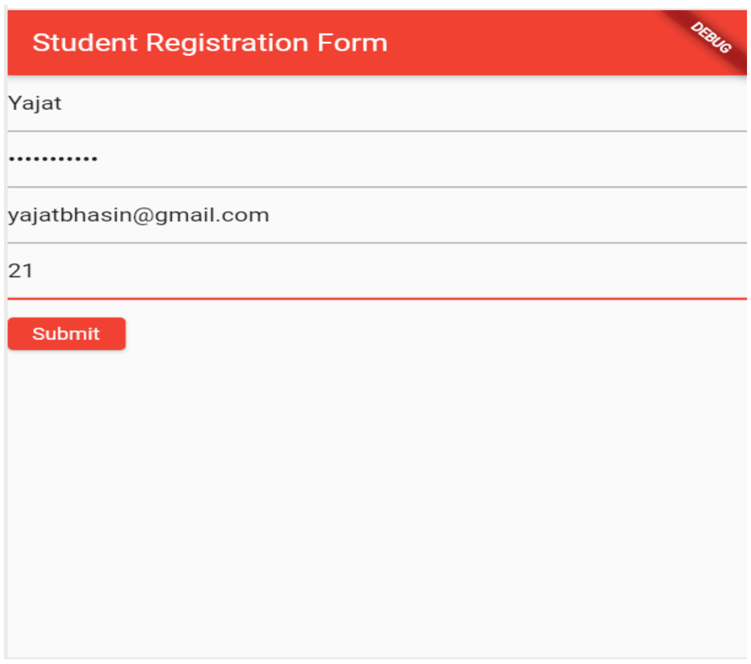
Student Registration Form

Enter your username

Enter your password

Enter your email

Enter your age

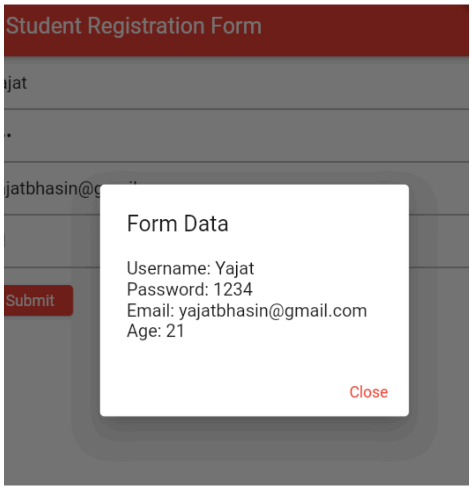


Yajat

•••••••••••

yajatbhasin@gmail.com

21



Form Data

Username: Yajat
Password: 1234
Email: yajatbhasin@gmail.com
Age: 21

Conclusion: In summary, using a form widget to create interactive forms enhances user engagement and data collection. It streamlines information input, offers customization options, and improves overall user experience. Interactive forms are versatile tools for surveys, registrations, and feedback mechanisms, facilitating efficient data capture and processing in digital environments.