Name: Abujaid Ansari **Batch-** A

Class: D15B **Roll No.** 02

Experiment No. 4

Aim: To create an interactive form using form widget

Theory:

The Form widget in Flutter is a fundamental widget for building forms. It provides a way to group multiple form fields together, perform validation on those fields, and manage their state.

Some Properties of Form Widget:

- key: A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- child: The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- autovalidateMode: An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget:

- validate(): This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- save(): This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- reset(): Resets the form to its initial state, clearing any user-entered data.
- currentState: A getter that returns the current FormState associated with the Form.

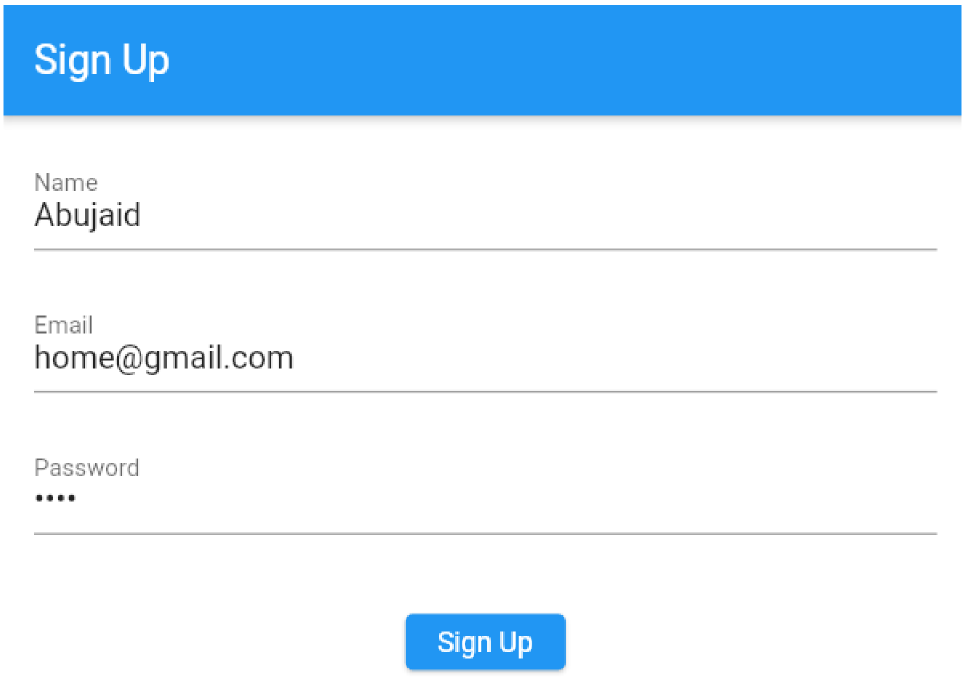
Code:

Main.dart

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

void main() {
  runApp(SignUpApp());
}

class SignUpApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Sign Up',
      theme: ThemeData(
        primaryColor: Colors.blue,
        scaffoldBackgroundColor: Colors.white,
        textTheme: TextTheme(
          bodyText2: TextStyle(color: Colors.grey[800]),
        ),
      ),
      home: SignUpScreen(),
    );
  }
}

class SignUpScreen extends StatefulWidget {
  @override
  _SignUpScreenState createState() => _SignUpScreenState();
}
```

```
class _SignUpScreenState extends State<SignUpScreen> {
  final _formKey = GlobalKey<FormState>(); // GlobalKey for the Form widget
  TextEditingController _nameController = TextEditingController();
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Sign Up'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Form(
          key: _formKey, // Assigning the GlobalKey to the Form widget
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              TextFormField(
                controller: _nameController,
                decoration: InputDecoration(labelText: 'Name'),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your name';
                  }
                  return null;
                },
              ),
              SizedBox(height: 20.0),
              TextFormField(
                controller: _emailController,
                decoration: InputDecoration(labelText: 'Email'),
                keyboardType: TextInputType.emailAddress,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your email';
                  }
                  // You can add more complex email validation logic here
                  return null;
                },
              ),
              SizedBox(height: 20.0),
              TextFormField(
                controller: _passwordController,
                decoration: InputDecoration(labelText: 'Password'),
                obscureText: true,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter a password';
                  }
                  // You can add more complex password validation logic here
                  return null;
                },
              ),
              SizedBox(height: 40.0),
              ElevatedButton(
                              onPressed: () {
                                if (_formKey.currentState!.validate()) {
                                  // Form is valid, perform signup logic here
                                  String name = _nameController.text;
                                  String email = _emailController.text;
                                  String password = _passwordController.text;

                                  // You can navigate to another screen or perform signup action here
                                  // For now, just print the signup details
                                  print('Name: $name');
                                  print('Email: $email');
                                  print('Password: $password');
                                }
                              },
                              child: Text('Sign Up'),
                            ),
                          ],
                        ),
                      ),
                    ),
                  );
                }
              }
```

Output:



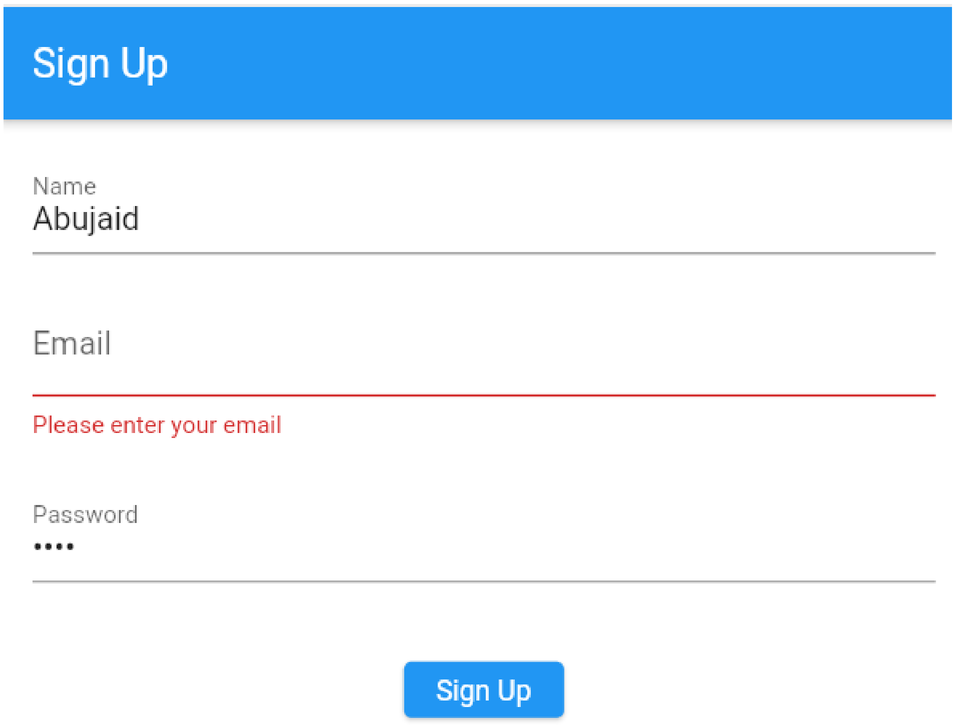
Sign Up

Name
Abujaid

Email
home@gmail.com

Password
••••

Sign Up



Sign Up

Name
Abujaid

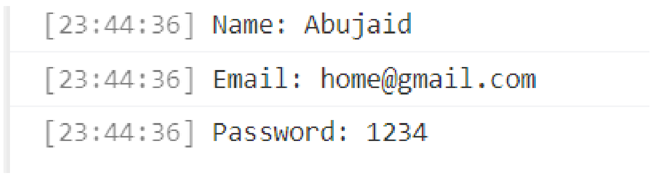
Email

Please enter your email

Password
••••

Sign Up

In console:



```
[23:44:36] Name: Abujaid
[23:44:36] Email: home@gmail.com
[23:44:36] Password: 1234
```

Explanation:

1. We created a new Flutter app called `SignUpApp` and defined its theme and home screen as `SignUpScreen`.

2. Inside `SignUpScreen`, I defined a `Form` widget to encapsulate the signup form. The `Form` widget requires a `GlobalKey<FormState>` to uniquely identify the form.

3. We used `TextFormField` widgets for collecting the user's name, email, and password. Each `TextFormField` has a `controller` property that controls the text input and a `validator` function to validate the input.

4. In the form submission button (`ElevatedButton`), I used the `onPressed` callback to validate the form using `_formKey.currentState!.validate()`. If the form is valid, I extracted the signup details from the text controllers and performed a signup action. For now, I just print the signup details to the console.

Conclusion:

Hence, in this experiment, we have learnt how to use form widget in our Flutter app. We have successfully created a form which takes input as email, name and password for new user sign up in our BMI calculator. The form validation ensured that users entered valid information before submitting. The form submission process was implemented to print the collected data to the console.