EXPERIMENT NO: - 04

Name:- Himesh Pathai
Class:- D15A
Roll:No: - 35

AIM: - To create an interactive Form using form widget.

Theory: -

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

- While creating a form in Flutter, the **Form widget** is essential as it acts as a container for grouping multiple form fields and managing validation.
- A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.
- The **TextFormField widget** is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.
- To enhance the appearance and usability of input fields, **InputDecoration** is used, allowing customization of labels, icons, borders, and hint text.
- Validation plays a crucial role in forms, and the **validator property** within **TextFormField** ensures user input meets specific criteria before submission.
- Different types of input require appropriate **keyboard types**, such as TextInputType.number for numeric fields or TextInputType.emailAddress for email fields.
- Proper **state management** is needed to store and retrieve user input, ensuring the form data is processed correctly.
- A **submit button** is necessary to trigger form validation and submit the collected data for further processing.

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 :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:inventory/lumina/src/features/main_app/main_screen/main_screen.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'email_input_screen.dart';
import 'package:inventory/lumina/src/features/authentication/controllers/emailcontroller.dart';
import 'package:supabase_flutter/supabase_flutter.dart';

class LoginForm extends StatefulWidget {
  const LoginForm({
    super.key,
  });

  @override
  State<LoginForm> createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
  final supabase = Supabase.instance.client;

  final TextEditingController emailcontroller = TextEditingController();
  final TextEditingController passwordcontroller = TextEditingController();

  final Emailcontroller emailGet = Get.put(Emailcontroller());

  bool isTermsAccepted = false;

  Future<void> emailsignin() async {
    if (!isTermsAccepted) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content:
```

```
        Text("Please accept the terms and conditions to proceed.")),
  );
  return;
}

try {
  final response = await supabase.auth.signInWithPassword(
    email: emailcontroller.text,
    password: passwordcontroller.text,
  );

  if (response.user != null) {
    emailGet.emailget.value = emailcontroller.text;
    emailcontroller.clear();
    passwordcontroller.clear();
    Get.to(MainScreen(), transition: Transition.fade);
    final session = response.session;
    await supabase.auth.setSession(session as String);
    final prefs = await SharedPreferences.getInstance();
    prefs.setString('email', emailGet.emailget.value);
  }
} on AuthException catch (e) {
  // Handle Supabase AuthException explicitly
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text(e.message)),
  );
} on Exception catch (e) {
  // Handle other exceptions
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
        content: Text("Something went wrong! Please try again.")),
  );
}
```

```
}

@override
void initState() {
  super.initState();

  supabase.auth.refreshSession().then((session) {
    if (session != null) {
      emailGet.emailget.value = session.user!.email!;
      emailGet.mailchecker();
    }
  });
}

TextStyle termsTextStyle = TextStyle(
  fontSize: 16.0,
  color: Colors.black,
);

@override
Widget build(BuildContext context) {
  return Form(
    child: Container(
      padding: EdgeInsets.symmetric(vertical: 20),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          TextFormField(
            controller: emailcontroller,
            decoration: const InputDecoration(
                prefixIcon: Icon(Icons.person_outline_outlined),
                labelText: "Email",
                hintText: "Email",
```

```
    border: OutlineInputBorder()),
),
SizedBox(
 height: 10,
),
TextFormField(
 obscureText: true,
 controller: passwordcontroller,
 decoration: const InputDecoration(
    prefixIcon: Icon(Icons.fingerprint),
    labelText: "Password",
    hintText: "Password",
    border: OutlineInputBorder()),
),
const SizedBox(height: 20),
Row(
 children: [
  Checkbox(
    value: isTermsAccepted,
    onChanged: (bool? value) {
     setState(() {
      isTermsAccepted = value ?? false;
     });
    },
  ),
  const Text('I agree to the'),
  TextButton(
    onPressed: () {
     _showTermsAndConditions();
    },
    child: Text("Terms and Conditions"),
  ),
 ],
```

```
),
const SizedBox(height: 20),
Align(
 alignment: Alignment.centerRight,
 child: TextButton(
  onPressed: () {
   Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => const ForgotPasswordScreen()),
   );
  },
  child: const Text("Forgot Password"),
 ),
),
Padding(
 padding:
   const EdgeInsets.symmetric(vertical: 20.0, horizontal: 10.0),
 child: SizedBox(
  width: double.infinity,
  child: Container(
   decoration: BoxDecoration(
    gradient: const LinearGradient(
     colors: [Color(0xFF507DBC), Color(0xFF70A1D7)],
     begin: Alignment.topLeft,
     end: Alignment.bottomRight,
    ),
    borderRadius: BorderRadius.circular(
      16),
   ),
   child: OutlinedButton.icon(
    style: OutlinedButton.styleFrom(
     padding: const EdgeInsets.symmetric(
```

```
                vertical: 16.0,
                horizontal: 24.0),
            side: BorderSide(
                color: Colors
                    .transparent),
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(
                    16)),
          ),
          icon: Icon(Icons.email, color: Colors.white),
          onPressed: () {
            emailsignin();
          },
          label: const Text(
            "Log-In with Email",
            style: TextStyle(color: Colors.white, fontSize: 18.0),
          ),
        ),
      ),
    ),
  ],
 ),
),
);
}

void _showTermsAndConditions() {
 bool isDarkMode = Theme.of(context).brightness == Brightness.dark;
 showModalBottomSheet(
  context: context,
  builder: (context) => Container(
   padding: const EdgeInsets.all(20),
```

```
child: SingleChildScrollView(
 child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
   Text(
    "Terms and Conditions",
    style: Theme.of(context).textTheme.headlineMedium,
   ),
   SizedBox(height: 10),
   Text(
    '1. Objective\n'
    'The purpose of these guidelines is to provide a clear understanding of the rules and procedures for the ISA-VESIT inventory system.\n\n'
    '2. Issuance of Components\n'
    '● Eligibility: Only students who are currently enrolled for ISA Memberships are eligible to borrow components.\n'
    '● Issuance Procedure:\n'
    '  - Components will be issued based on availability and necessity.\n'
    '  - A record of issued components will be maintained by the ISA Council.\n\n'
    '3. Student Responsibilities\n'
    '● Care: Students are responsible for the proper care and handling of the components.\n'
    '● Usage: Components must be used only for their intended educational or project purposes.\n'
    '● Return: Components must be returned by the due date specified at the time of issuance.\n'
    '● Modification: Modification of the components is not allowed.\n'
    '● Damage: No damage is allowed. Students will have to pay the entire amount if the component is damaged.\n'
    '● Loss: If a component is lost, the student is responsible for it and has to pay the entire amount of the component as listed below.\n'
    '● Issuance/Reissuance: Components must be issued or reissued in the presence of and with the approval of a council member only.\n'
    '● Reissue: The component should be reissued within 1 month of time after issuing the component.\n'
    '● Timing: For issuance/reissuance of the component the timings are 1) 1:00 pm - 1:30 pm 2) 3:30 pm - 4:00 pm\n\n'
    '4. Return Policy\n'
```

'● Due Date: Components must be returned by the due date specified during issuance.\n'

'● Condition: Components must be returned in the same condition as they were issued.\n'

'● All the components should be returned to the council before the end semester exam.\n'

'● Refer Fine Structure for more terms and conditions regarding the fine payment.\n\n'

'5. Fine Structure and Payment\n'

'● Late Returns:\n'

' - A fine will be imposed for each day the component is returned late.\n'

' - The maximum late fine will not exceed 2500 Rs.\n'

'● Component Damage:\n'

' - No damage to the components would be accepted.\n'

' - In case of damage: Replacement Cost of the new component.\n'

'● Loss of Component:\n'

' - Full replacement cost of the component will be charged.\n'

'● Payment:\n'

' - Fines must be paid within 5 days of notification.\n'

' - Payment should be made online.\n\n'

'6. Consequences of Non-Payment\n'

'● Failure to pay fines may result in:\n'

' - Suspension of borrowing privileges.\n'

' - Membership Suspension.\n'

' - Clearance for collecting Leaving Certificate would not be provided by the Central Library of College.\n\n'

'7. Dispute Resolution\n'

'● Students who wish to dispute a fine may do so by submitting a written appeal to the ISA committee within 3 days of fine notification.\n'

'● The decision of the ISA committee will be final.\n\n'

'8. Policy Review\n'

'● This policy will be reviewed annually and is subject to change. Updates will be communicated to all students via email and WhatsApp.\n\n'

'9. Component Price List:\n'

'https://bit.ly/ComponentPrice\n'

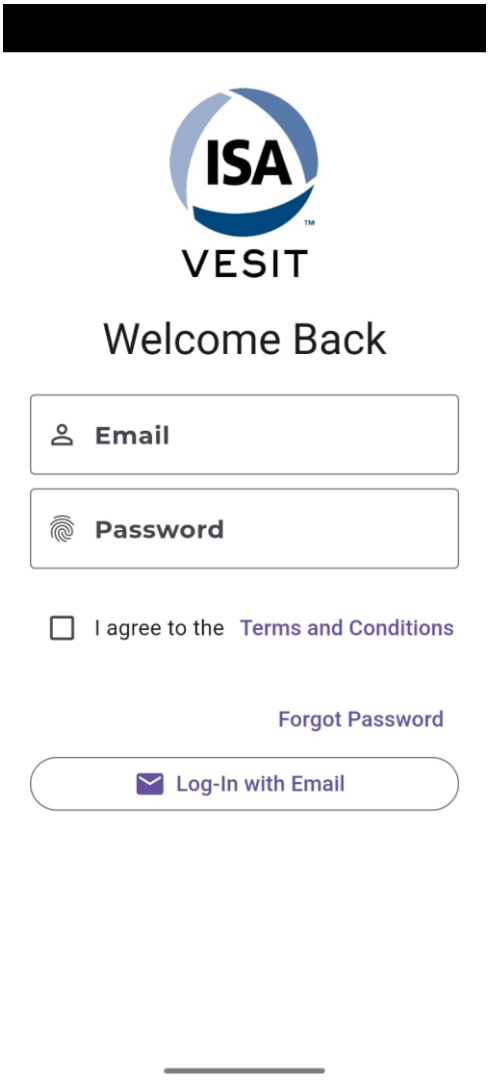
'Note: Subject to Change of Price\n\n'

'10. Contact Information\n'

'For any questions or concerns regarding this policy, please contact:\n'

```
            'Sr. Treasurer: Atishkar Singh\n'
            'Phone No: 9049120954',
            style: termsTextStyle.copyWith(
              color: isDarkMode
                  ? Colors.white
                  : Colors.black), // Set color to white
          ),
        ],
      ),
    ),
  ),
 );
}
```

Output :


ISA
VESIT
Welcome Back
Email
Password
I agree to the Terms and Conditions
Forgot Password
Log-In with Email