Name: Vedang Khandagale
Division: D15A
Roll No: 29
Batch: B

Experiment No 4

Aim: To create interactive form using form widget

Theory: In Flutter, a "Form" is a widget that represents a container for a collection of form fields. It helps manage the state of the form and facilitates the validation and submission of user input. Here are some key concepts and theories about forms in Flutter:

1. Widget Hierarchy:
   Forms in Flutter are composed of the `Form` widget, which contains a list of `FormField` widgets. Each `FormField` represents an individual input field like text fields, checkboxes, or dropdowns.

2. Form State:
   The `Form` widget maintains the state of the form, including the current values of the form fields and their validation statuses. The form state is automatically managed by Flutter.

3. Validation:
   Forms provide built-in validation through the `validator` property of each `FormField`. Validators are functions that determine whether the input is valid. The form's overall validity is determined by the validity of all its fields.

4. Form Submission:
   Form submission is typically triggered by a button press. The `onPressed` callback of the button can call the `FormState.save()` method, which invokes the `onSaved` callback for each form field and then calls the `onFormSaved` callback.

5. GlobalKey<FormState>:
   To interact with the form state, a `GlobalKey<FormState>` is commonly used. This key allows access to the form state and is used to validate and save the form.

6. Auto-validation:
   Flutter provides automatic validation by calling the `validator` function whenever the user input changes. This allows for real-time feedback to the user about the validity of their input.

7. Form Submission Lifecycle:
   The form submission process involves validation, saving, and then handling the saved data. Developers can customize this process by providing their own logic within the `onSaved` and `onFormSaved` callbacks.

8. Focus Management:
Forms handle the focus of input fields, making it easy to navigate through the form using keyboard input or programmatically setting focus on specific fields.

9. FormKey and GlobalKey:
Using a `GlobalKey<FormState>` allows for more control over the form, such as triggering form validation or resetting the form. It is usually defined as a global key in the widget tree.

10. Form Persistence:
Form data can be persisted across different screens or app sessions by passing the data down the widget tree or using state management solutions like Provider or Riverpod.

Forms play a crucial role in user interaction, data collection, and validation in Flutter applications, providing a structured and efficient way to handle user input.

Code:

```
import 'dart:io';
import 'package:facebook_clone/core/constants/app_colors.dart';
import 'package:facebook_clone/core/constants/constants.dart';
import 'package:facebook_clone/core/utils/utils.dart';
import 'package:facebook_clone/core/widgets/pick_image_widget.dart';
import 'package:facebook_clone/core/widgets/round_button.dart';
import 'package:facebook_clone/core/widgets/round_text_field.dart';
importpackage:facebook_clone/features/auth/presentation/widgets/birthday_p
icker.dart';
import'package:facebook_clone/features/auth/presentation/widgets/gender_pi
cker.dart';
import
'package:facebook_clone/features/auth/providers/auth_provider.dart';
import 'package:facebook_clone/features/auth/utils/utils.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
final _formKey = GlobalKey<FormState>();
class CreateAccountScreen extends ConsumerStatefulWidget {
  const CreateAccountScreen({super.key});
  static const routeName = '/create-account';
  @override
  ConsumerState<CreateAccountScreen> createState() =>
      _CreateAccountScreenState();
}
```

```dart
class _CreateAccountScreenState extends ConsumerState<CreateAccountScreen>
{
  File? image;
  DateTime? birthday;
  String gender = 'male';
  bool isLoading = false;
  late final TextEditingController _fNameController;
  late final TextEditingController _lNameController;
  late final TextEditingController _emailController;
  late final TextEditingController _passwordController;
  @override
  void initState() {
    _fNameController = TextEditingController();
    _lNameController = TextEditingController();
    _emailController = TextEditingController();
    _passwordController = TextEditingController();
    super.initState();
  }
  @override
  void dispose() {
    _fNameController.dispose();
    _lNameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
  Future<void> createAccount() async {
    if (_formKey.currentState!.validate()) {
      _formKey.currentState!.save();
      setState(() => isLoading = true);
      await ref
          .read(authProvider)
          .createAccount(
            fullName: '${_fNameController.text} ${_lNameController.text}',
            birthday: birthday ?? DateTime.now(),
            gender: gender,
            email: _emailController.text,
            password: _passwordController.text,
            image: image,
          )
```

```
          .then((credential) {
        if (!credential!.user!.emailVerified) {
          Navigator.pop(context);
        }
      }).catchError((_) {
        setState(() => isLoading = false);
      });
      setState(() => isLoading = false);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: AppColors.realWhiteColor,
      appBar: AppBar(),
      body: SingleChildScrollView(
        child: Padding(
          padding: Constants.defaultPadding,
          child: Form(
            key: _formKey,
            child: Column(
              children: [
                GestureDetector(
                  onTap: () async {
                    image = await pickImage();
                    setState(() {});
                  },
                  child: PickImageWidget(image: image),
                ),
                const SizedBox(height: 20),
                Row(
                  children: [
                    // First Name Text Field
                    Expanded(
                      child: RoundTextField(
                        controller: _fNameController,
                        hintText: 'First name',
                        textInputAction: TextInputAction.next,
                        validator: validateName,
```

```
          ),
        ),
        const SizedBox(width: 10),
        // Last Name Text Field
        Expanded(
          child: RoundTextField(
            controller: _lNameController,
            hintText: 'Last name',
            textInputAction: TextInputAction.next,
            validator: validateName,
          ),
        ),
      ],
    ),
    const SizedBox(height: 20),
    BirthdayPicker(
      dateTime: birthday ?? DateTime.now(),
      onPressed: () async {
        birthday = await pickSimpleDate(
          context: context,
          date: birthday,
        );
        setState(() {});
      },
    ),
    const SizedBox(height: 20),
    GenderPicker(
      gender: gender,
      onChanged: (value) {
        gender = value ?? 'male';
        setState(() {});
      },
    ),
    const SizedBox(height: 20),
    // Phone number / email text field
    RoundTextField(
      controller: _emailController,
      hintText: 'Email',
      textInputAction: TextInputAction.next,
      keyboardType: TextInputType.emailAddress,
```

```
                validator: validateEmail,
              ),
              const SizedBox(height: 20),
              // Password Text Field
              RoundTextField(
                controller: _passwordController,
                hintText: 'Password',
                textInputAction: TextInputAction.done,
                keyboardType: TextInputType.visiblePassword,
                validator: validatePassword,
                isPassword: true,
              ),
              const SizedBox(height: 20),
              isLoading
                  ? const Center(child: CircularProgressIndicator())
                  : RoundButton(
                      onPressed: createAccount,
                      label: 'Create Account',
                    ),
            ],
          ),
        ),
      ),
    ),
  );
}
```

Output: