Param Keswani
D15A 27
Batch B
Experiment no 4
Aim - To create an interactive Form using form widget
Theory -
Interactive Form Creation in Flutter

Flutter offers an array of powerful widgets and techniques to build adaptable and user-friendly forms that seamlessly integrate with your app's design and functionality. This guide delves into the key concepts and strategies involved in crafting interactive forms.

Fundamental Widgets:

● Form: The overarching container that encompasses form fields and manages validation. Create a unique GlobalKey<FormState> for global access and validation.

● FormField: A base class for input widgets, providing styling, validation, and error handling. Choose the appropriate concrete widget based on the input type (e.g., TextField, Checkbox, DropdownButton).

● TextInputAction: Set the textInputAction property on TextField to control the on-screen keyboard's behavior (e.g., TextInputAction.next, TextInputAction.done).

● FocusNode: Control keyboard focus navigation between form fields or widgets within a field (e.g., for multi-line text editing). Use FocusNode objects with FocusManager for management.

Dynamic Form Building:

● ListViews: Dynamically render form fields by creating a ListView.builder that builds FormField instances based on data (e.g., from a list or JSON). Customize field types and validation based on data properties.

● State Management: Employ state management solutions like Bloc, Provider, or raw setState to handle form state effectively. Store form data, validation errors, and other state information dynamically.

User Interaction and Validation:

● Input Validation: Implement validation rules within FormField or its descendant widgets using a validator callback. Provide clear error messages for invalid input to guide users.

● Focus Management: Use FocusNode to control keyboard focus flow, enabling a natural form-filling experience. Consider using packages like auto_animated for field auto-focusing when entering the screen.

● Interactive UI: Incorporate widgets like Checkbox, DropdownButton, Radio, and custom interactive elements to offer a variety of input options and enhance user engagement.

Additional Considerations:

● Performance: For large forms, consider techniques like lazy loading or pagination to manage memory and rendering overhead.

- Accessibility: Ensure your forms are accessible to users with disabilities by following WCAG guidelines and using appropriate semantic elements.
- Styling: Customize form aesthetics using Flutter's rich customization options to match your app's design and provide visual feedback (e.g., underline active fields, highlight errors)

Code :

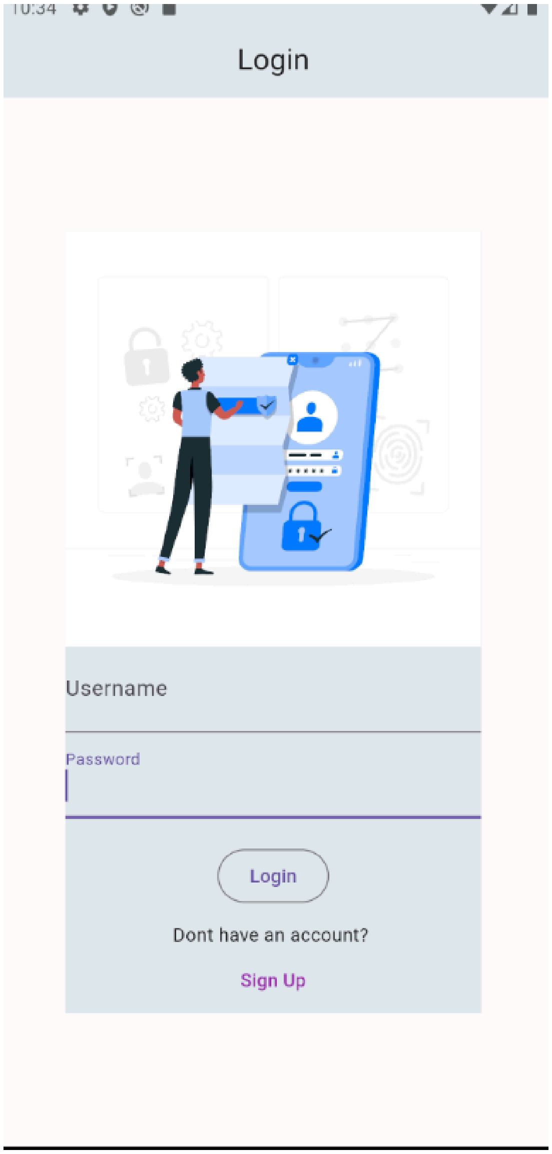
```
import 'package:flutter/material.dart';
import 'signup.dart';
import 'home.dart';

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

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Center(child: Text("Login")),
        backgroundColor: Color.fromARGB(255, 223, 230, 236),
      ),
      body: Container(
        width: double.infinity,
        height: 700,
        color: const Color.fromARGB(255, 223, 230, 236),
        margin: EdgeInsets.fromLTRB(50, 100, 50, 100),
        child: SingleChildScrollView(
          child: Column(
            children: [
              Image.asset("images/login.jpg"),
              TextFormField(
                decoration: const InputDecoration(
                  labelText: 'Username',
                ),
              ),
              TextFormField(
                obscureText: true,
                decoration: const InputDecoration(
                  labelText: 'Password',
                ),
              ),
              SizedBox(
                height: 20,
                width: double.infinity,
              ),
              OutlinedButton(
                  onPressed: () {
                    Navigator.push(context,
                        MaterialPageRoute(builder: (context) => Home()));
                  },
                  child: Text("Login")),
              SizedBox(
                height: 10,
              ),
              Text("Dont have an account? "),
              TextButton(
                  onPressed: () {
                    Navigator.push(context,
                        MaterialPageRoute(builder: (context) => Signup()));
                  },
                  child: const Text(
                    "Sign Up",
                    style: TextStyle(color: Colors.purple),
                  ))
            ],
          ),
        ),
      ),
    );
  }
}
```

Screenshot:



Dont have an account?