

MAD PWA Lab Exp 4

Aim: To create an interactive Form using form widget

Theory:

Forms play a vital role in collecting and processing user input in mobile applications. Flutter provides a powerful Form widget that simplifies the process of creating interactive and responsive forms. This theory covers the fundamental concepts and practices involved in creating an interactive form using the Form widget.

Key Concepts:

Form Widget:

The Form widget serves as a container for form elements, helping to organize and manage user input.

It enables the grouping of various form fields and facilitates form-level operations like validation and submission.

TextFormFields:

Text input is a common element in forms, and Flutter provides the TextFormField widget for this purpose.

TextFormField allows developers to collect textual input from users and supports customization through attributes like decoration and validator.

Form Validation:

The Form widget includes a powerful validation mechanism using the validator property in form fields.

Developers can define custom validation functions to ensure that user input meets specific criteria before submission.

Form Submission:

Forms typically include a mechanism for submitting user input. Flutter provides the onSaved property to handle the form data once it passes validation.

Developers can implement custom logic to process and store the validated form data.

Form Key:

Managing the form's state is crucial for interactivity. Flutter uses a GlobalKey<FormState> to interact with the form's state.

The key enables actions such as resetting the form, validating all form fields, and accessing the form's current state.

Best Practices:**Organizing Form Elements:**

Structure the form using a Column or ListView for better organization and readability. Group related form fields together to enhance user experience.

Error Handling:

Leverage the validator property to provide meaningful error messages to users when form input is invalid.

Display errors in a user-friendly manner, enhancing the overall user experience.

Responsive Design:

Ensure that the form is responsive to different screen sizes and orientations.

Use Flutter's layout widgets to create a consistent and visually appealing form across devices.

Accessibility:

Implement accessibility features to ensure users with disabilities can interact with and understand the form.

Use proper labels, hints, and error messages to enhance accessibility.

Code & Implementation:

```
//mobilenumber.dart
```

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

```
import 'package:jio_saavn_app/screens/home.dart';
```

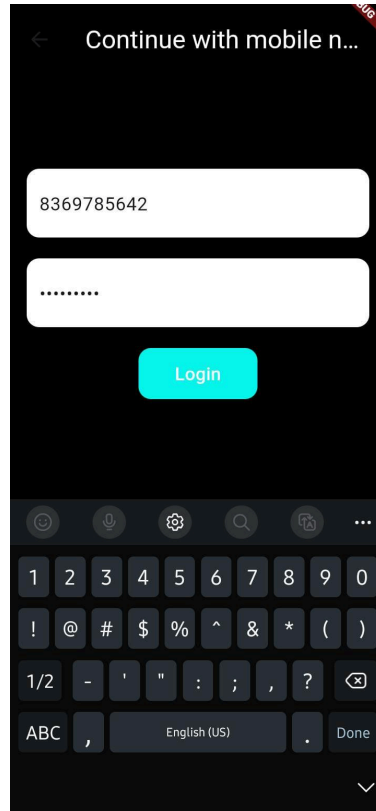
```
class MobileNumberPage extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(  
          'Continue with mobile number',  
          style: TextStyle(color: Colors.white),  
        ),  
      ),  
    );  
  }  
}
```

```
        backgroundColor: Colors.black,
      ),
      backgroundColor: Colors.black,
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              keyboardType: TextInputType.phone,
              decoration: InputDecoration(
                hintText: 'Enter mobile number',
                fillColor: Colors.white,
                filled: true,
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(12.0),
                  borderSide: BorderSide.none,
                ),
              ),
            ),
            SizedBox(height: 20),

            // Password TextField
            TextField(
              obscureText: true, // Hide the entered text for passwords
              decoration: InputDecoration(
                hintText: 'Enter password',
                fillColor: Colors.white,
                filled: true,
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(12.0),
                  borderSide: BorderSide.none,
                ),
              ),
            ),
            SizedBox(height: 20),

            ElevatedButton(
              onPressed: () {
                Navigator.push(
```

```
        context,
        MaterialPageRoute(builder: (context) => HomeScreen()),
    );
    // Handle mobile number login
    // You can implement your logic to verify the mobile
    number and navigate to the next screen
    },
    style: ElevatedButton.styleFrom(
        primary: Color.fromRGBO(0, 255, 240, 1), // Aqua color
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12.0),
        ),
    ),
    child: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Text(
            'Login',
            style: TextStyle(fontSize: 16, color: Colors.white),
        ),
    ),
),
],
),
),
);
}
```



Conclusion: Creating interactive forms in Flutter involves leveraging the Form widget, managing form state, and incorporating best practices for a seamless user experience. By following these principles, developers can design robust and user-friendly forms in their Flutter applications.