Name: Abhay Gupta

Div: D15 B

Roll No.: 18

MPL Practical 4

Aim:

To design and implement an interactive form using the Form widget in Flutter that validates user input for fields such as email and password.

Theory:

Forms are an essential part of user interaction in mobile applications, allowing users to input and submit data. In Flutter, the Form widget is used to group multiple form fields (TextFormField), enabling validation and submission handling.

- **Form Widget:**
  - A container for grouping multiple form fields.
  - Uses a GlobalKey<FormState> to validate and manage state.
- **TextFormField Widget:**
  - Accepts user input with built-in validation.
  - Provides error messages when validation fails.
- **Form Validation:**
  - Ensures input fields contain valid data before submission.
  - Uses the validator function to check conditions such as email format and password length.
- **Submission Handling:**
  - The FormState class provides methods like validate() and save() to handle form submission.
  - If validation succeeds, the form processes the data; otherwise, error messages are displayed.

Code:

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      debugShowCheckedModeBanner: false, // Removes the debug banner
      title: 'Technix Login',
      theme: ThemeData.dark().copyWith(
        primaryColor: Colors.red,
        scaffoldBackgroundColor: Colors.black,),
      home: const LoginPage(),
    );}}
class LoginPage extends StatefulWidget {
  const LoginPage({Key? key}) : super(key: key);
@override
  State<LoginPage> createState() => _LoginPageState();}
class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool _isPasswordVisible = false;
  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
  String? _validateEmail(String? value) {
    if (value == null || value.isEmpty) {
      return 'Email is required';
    }
    final emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}\$');
    if (!emailRegex.hasMatch(value)) {
      return 'Please enter a valid email';
    }    return null;
  }
```

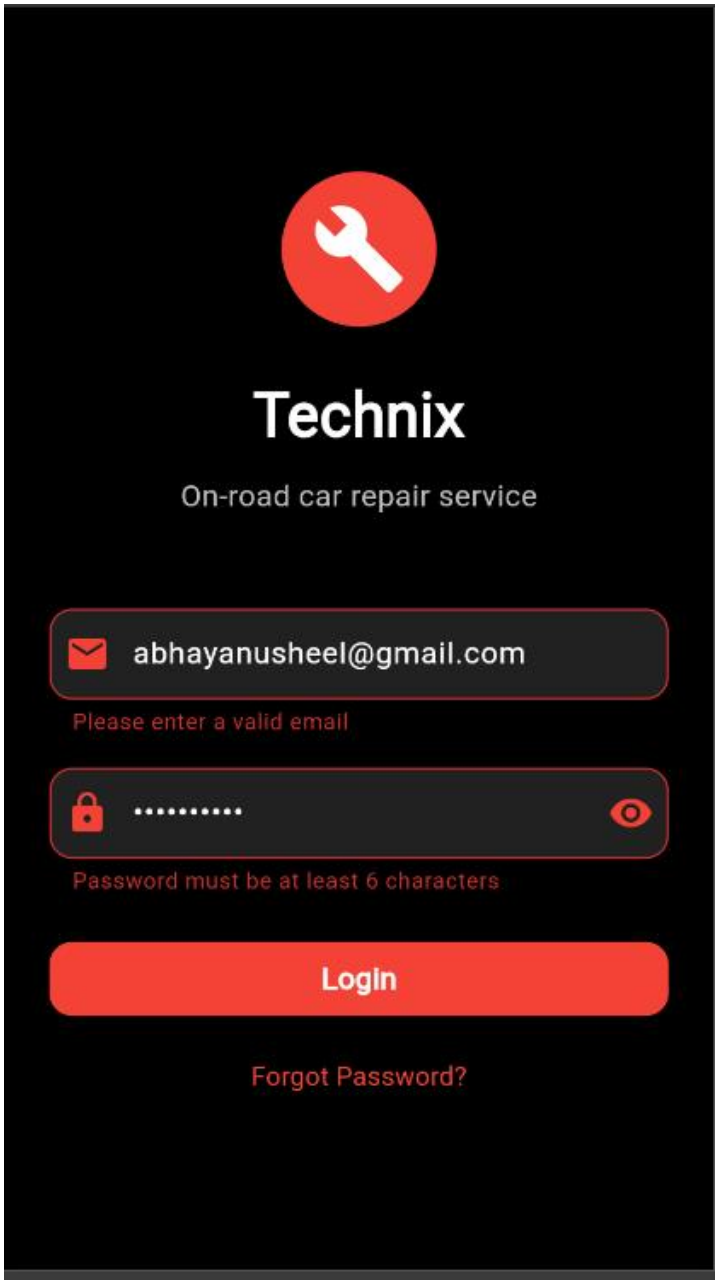
```
  String? _validatePassword(String? value) {
    if (value == null || value.isEmpty) {
      return 'Password is required';
    }
    if (value.length < 6) {
      return 'Password must be at least 6 characters';
    }
    return null;
  }

  void _handleLogin() {
    if (_formKey.currentState!.validate()) {
      print('Email: ${_emailController.text}');
      print('Password: ${_passwordController.text}');
    }
  }
@override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: SafeArea(
        child: Center(
          child: SingleChildScrollView(
            padding: const EdgeInsets.all(24.0),
            child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Container(
                    padding: const EdgeInsets.all(16.0),
                    decoration: BoxDecoration(
```

```
    color: Colors.red,
    shape: BoxShape.circle,
  ),
  child: Icon(
    Icons.build,
    size: 50,
    color: Colors.white,
  ),
),
const SizedBox(height: 24),
Text(
  'Technix',
  style: TextStyle(
    fontSize: 32,
    fontWeight: FontWeight.bold,
    color: Colors.white,
  ),
),
const SizedBox(height: 8),
Text(
  'On-road car repair service',
  style: TextStyle(
    fontSize: 16,
    color: Colors.grey[400],
  ),
),
const SizedBox(height: 48),
TextFormField(
  controller: _emailController,
  style: TextStyle(color: Colors.white),
  validator: _validateEmail,
  keyboardType: TextInputType.emailAddress,
```

```
  decoration: _inputDecoration('Email', Icons.email),
),
const SizedBox(height: 16),
TextFormField(
  controller: _passwordController,
  obscureText: !_isPasswordVisible,
  style: TextStyle(color: Colors.white),
  validator: _validatePassword,
  decoration: _inputDecoration('Password', Icons.lock).copyWith(
    suffixIcon: IconButton(
      icon: Icon(
        _isPasswordVisible ? Icons.visibility_off : Icons.visibility,
        color: Colors.red,
      ),
      onPressed: () {
        setState(() {
          _isPasswordVisible = !_isPasswordVisible;
        });
      },
    ),
  ),
),
const SizedBox(height: 24),
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: _handleLogin,
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.red,
      padding: const EdgeInsets.symmetric(vertical: 16),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(12),
```

```
                ),
              ),
              child: Text(
                'Login',
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                ),
              ),
            ),
          ),
          const SizedBox(height: 16),
          TextButton(
            onPressed: () {},
            child: Text(
              'Forgot Password?',
              style: TextStyle(
                color: Colors.red,
                fontSize: 14,
              ),),),],),),),),), );  }
InputDecoration _inputDecoration(String hintText, IconData icon) {
  return InputDecoration(
    hintText: hintText,
    hintStyle: TextStyle(color: Colors.grey[400]),
    prefixIcon: Icon(icon, color: Colors.red),
    filled: true,
    fillColor: Colors.grey[900],
    enabledBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.transparent),      ),
    focusedBorder: OutlineInputBorder(
```

);}}



Conclusion:

By using the Form widget in Flutter, we can create interactive and user-friendly forms with validation. This approach enhances user experience by preventing incorrect data entry and ensuring only valid input is processed. The use of GlobalKey<FormState> enables efficient form validation and submission, making it an essential component in Flutter applications requiring user input.