Aim: To create an interactive Form using a form widget.

Theory:

Forms are an essential part of mobile applications, allowing users to input and submit data. In Flutter, the Form widget is used to group multiple input fields together while managing user interactions and validation efficiently.

General Explanation

A form consists of input fields (TextField), validation logic, and submission actions. It helps in collecting structured data from users, ensuring that necessary fields are filled before processing. Forms also enhance user experience by providing feedback, such as error messages or success notifications.

Implementation in Our Code

- Used TextField widgets to collect item details (name and category).
- Implemented TextEditingController to manage input and clear fields after submission.
- Added validation checks to ensure required fields are not empty.
- Displayed feedback messages (SnackBar) to notify users of successful form submission or missing input.
- Used icons and UI styling to enhance form usability.

Code

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

void main() {
 runApp(LocalResourceApp());
}

class LocalResourceApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   title: 'Local Resource Sharing',
   theme: ThemeData(
    primaryColor: Colors.teal,
    scaffoldBackgroundColor: Colors.grey[200],
    textTheme: TextTheme(
     bodyLarge: TextStyle(fontFamily: 'Roboto', color: Colors.black),
     bodyMedium: TextStyle(fontFamily: 'Roboto', color: Colors.black87),
    ),
   ),
   home: AuthChecker(),
```

```
    );
  }
}

// Auth Checker to determine whether to show login or home screen
class AuthChecker extends StatefulWidget {
  @override
  _AuthCheckerState createState() => _AuthCheckerState();
}

class _AuthCheckerState extends State<AuthChecker> {
  bool isLoading = true;
  bool isLoggedIn = false;

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

  Future<void> checkLoginStatus() async {
    final prefs = await SharedPreferences.getInstance();
    final loggedIn = prefs.getBool('isLoggedIn') ?? false;

    setState(() {
      isLoggedIn = loggedIn;
      isLoading = false;
    });
  }

  @override
  Widget build(BuildContext context) {
    if (isLoading) {
      return Scaffold(
        body: Center(
          child: CircularProgressIndicator(
            color: Colors.teal,
          ),
        ),
      );
    }
```

```
    return isLoggedIn ? HomeScreen() : LoginScreen();
  }
}

// Login Screen
class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  bool _isPasswordVisible = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[200],
      body: SafeArea(
        child: Center(
          child: SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.all(24.0),
              child: Form(
                key: _formKey,
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    // Logo and App Name
                    Icon(
                      Icons.share_rounded,
                      size: 70,
                      color: Colors.teal,
                    ),
                    SizedBox(height: 15),
                    Text(
                      'Local Resource Sharing',
                      style: TextStyle(
```

```
    fontSize: 24,
    fontWeight: FontWeight.bold,
    color: Colors.teal,
   ),
  ),
  SizedBox(height: 40),

  // Email Field
  TextFormField(
   controller: _emailController,
   keyboardType: TextInputType.emailAddress,
   decoration: InputDecoration(
    labelText: 'Email',
    hintText: 'Enter your email',
    prefixIcon: Icon(Icons.email, color: Colors.teal),
    border: OutlineInputBorder(
     borderRadius: BorderRadius.circular(10),
    ),
    focusedBorder: OutlineInputBorder(
     borderRadius: BorderRadius.circular(10),
     borderSide: BorderSide(color: Colors.teal, width: 2),
    ),
   ),
   validator: (value) {
    if (value == null || value.isEmpty) {
     return 'Please enter your email';
    }
    // Email format validation using regex
    final emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');
    if (!emailRegex.hasMatch(value)) {
     return 'Please enter a valid email';
    }
    return null;
   },
  ),
  SizedBox(height: 20),

  // Password Field
  TextFormField(
   controller: _passwordController,
   obscureText: !_isPasswordVisible,
```

```
    decoration: InputDecoration(
     labelText: 'Password',
     hintText: 'Enter your password',
     prefixIcon: Icon(Icons.lock, color: Colors.teal),
     suffixIcon: IconButton(
      icon: Icon(
       _isPasswordVisible ? Icons.visibility : Icons.visibility_off,
       color: Colors.grey,
      ),
      onPressed: () {
       setState(() {
        _isPasswordVisible = !_isPasswordVisible;
       });
      },
     ),
     border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
     ),
     focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
      borderSide: BorderSide(color: Colors.teal, width: 2),
     ),
    ),
    validator: (value) {
     if (value == null || value.isEmpty) {
      return 'Please enter your password';
     }
     if (value.length < 6) {
      return 'Password must be at least 6 characters';
     }
     return null;
    },
   ),
   SizedBox(height: 20),

   // Login Button
   SizedBox(
    width: double.infinity,
    height: 50,
    child: ElevatedButton(
     onPressed: () async {
```

```
          if (_formKey.currentState!.validate()) {
            // For demo purpose, just save login state
            final prefs = await SharedPreferences.getInstance();
            await prefs.setBool('isLoggedIn', true);
            await prefs.setString('userEmail', _emailController.text);

            Navigator.pushReplacement(
              context,
              MaterialPageRoute(builder: (context) => HomeScreen()),
            );
          }
        },
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.teal,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
        ),
        child: Text(
          'Login',
          style: TextStyle(
            color: Colors.white,
            fontSize: 16,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
    ),
    SizedBox(height: 20),

    // Sign Up Link
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text("Don't have an account? "),
        TextButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SignupScreen()),
            );
```

```
                    },
                    child: Text(
                     'Sign Up',
                     style: TextStyle(
                      color: Colors.teal,
                      fontWeight: FontWeight.bold,
                     ),
                    ),
                   ),
                  ],
                 ),
                ],
               ),
              ),
             ),
            ),
           ),
          ),
         ),
        );
       }
}

// Sign Up Screen
class SignupScreen extends StatefulWidget {
 @override
 _SignupScreenState createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
 final _formKey = GlobalKey<FormState>();
 final TextEditingController _nameController = TextEditingController();
 final TextEditingController _emailController = TextEditingController();
 final TextEditingController _passwordController = TextEditingController();
 final TextEditingController _confirmPasswordController = TextEditingController();
 bool _isPasswordVisible = false;
 bool _isConfirmPasswordVisible = false;

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.grey[200],
```

```
  appBar: AppBar(
   backgroundColor: Colors.teal,
   title: Text('Create Account'),
   foregroundColor: Colors.white,
  ),
  body: SafeArea(
   child: Center(
    child: SingleChildScrollView(
     child: Padding(
      padding: const EdgeInsets.all(24.0),
      child: Form(
       key: _formKey,
       child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
         // Name Field
         TextFormField(
          controller: _nameController,
          decoration: InputDecoration(
           labelText: 'Full Name',
           hintText: 'Enter your full name',
           prefixIcon: Icon(Icons.person, color: Colors.teal),
           border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
           ),
           focusedBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
            borderSide: BorderSide(color: Colors.teal, width: 2),
           ),
          ),
          validator: (value) {
           if (value == null || value.isEmpty) {
            return 'Please enter your name';
           }
           return null;
          },
         ),
         SizedBox(height: 20),

         // Email Field
         TextFormField(
```

```
    controller: _emailController,
    keyboardType: TextInputType.emailAddress,
    decoration: InputDecoration(
     labelText: 'Email',
     hintText: 'Enter your email',
     prefixIcon: Icon(Icons.email, color: Colors.teal),
     border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
     ),
     focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
      borderSide: BorderSide(color: Colors.teal, width: 2),
     ),
    ),
    validator: (value) {
     if (value == null || value.isEmpty) {
      return 'Please enter your email';
     }
     // Email format validation using regex
     final emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');
     if (!emailRegex.hasMatch(value)) {
      return 'Please enter a valid email';
     }
     return null;
    },
   ),
   SizedBox(height: 20),

   // Password Field
   TextFormField(
    controller: _passwordController,
    obscureText: !_isPasswordVisible,
    decoration: InputDecoration(
     labelText: 'Password',
     hintText: 'Create a password',
     prefixIcon: Icon(Icons.lock, color: Colors.teal),
     suffixIcon: IconButton(
      icon: Icon(
       _isPasswordVisible ? Icons.visibility : Icons.visibility_off,
       color: Colors.grey,
      ),
```

```
          onPressed: () {
            setState(() {
              _isPasswordVisible = !_isPasswordVisible;
            });
          },
        ),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(10),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(10),
          borderSide: BorderSide(color: Colors.teal, width: 2),
        ),
      ),
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Please enter a password';
        }
        if (value.length < 6) {
          return 'Password must be at least 6 characters';
        }
        // Password strength validation
        bool hasUppercase = value.contains(RegExp(r'[A-Z]'));
        bool hasDigits = value.contains(RegExp(r'[0-9]'));
        bool hasSpecialCharacters = value.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'));

        if (!(hasUppercase && hasDigits && hasSpecialCharacters)) {
          return 'Password must contain uppercase, number, and special character';
        }
        return null;
      },
    ),
    SizedBox(height: 20),

    // Confirm Password Field
    TextFormField(
      controller: _confirmPasswordController,
      obscureText: !_isConfirmPasswordVisible,
      decoration: InputDecoration(
        labelText: 'Confirm Password',
        hintText: 'Confirm your password',
```

```
      prefixIcon: Icon(Icons.lock_outline, color: Colors.teal),
      suffixIcon: IconButton(
       icon: Icon(
        _isConfirmPasswordVisible ? Icons.visibility : Icons.visibility_off,
        color: Colors.grey,
       ),
       onPressed: () {
        setState(() {
         _isConfirmPasswordVisible = !_isConfirmPasswordVisible;
        });
       },
      ),
      border: OutlineInputBorder(
       borderRadius: BorderRadius.circular(10),
      ),
      focusedBorder: OutlineInputBorder(
       borderRadius: BorderRadius.circular(10),
       borderSide: BorderSide(color: Colors.teal, width: 2),
      ),
     ),
     validator: (value) {
      if (value == null || value.isEmpty) {
       return 'Please confirm your password';
      }
      if (value != _passwordController.text) {
       return 'Passwords do not match';
      }
      return null;
     },
    ),
    SizedBox(height: 30),

    // Sign Up Button
    SizedBox(
     width: double.infinity,
     height: 50,
     child: ElevatedButton(
      onPressed: () async {
       if (_formKey.currentState!.validate()) {
        // For demo purpose, save user info and login
        final prefs = await SharedPreferences.getInstance();
```

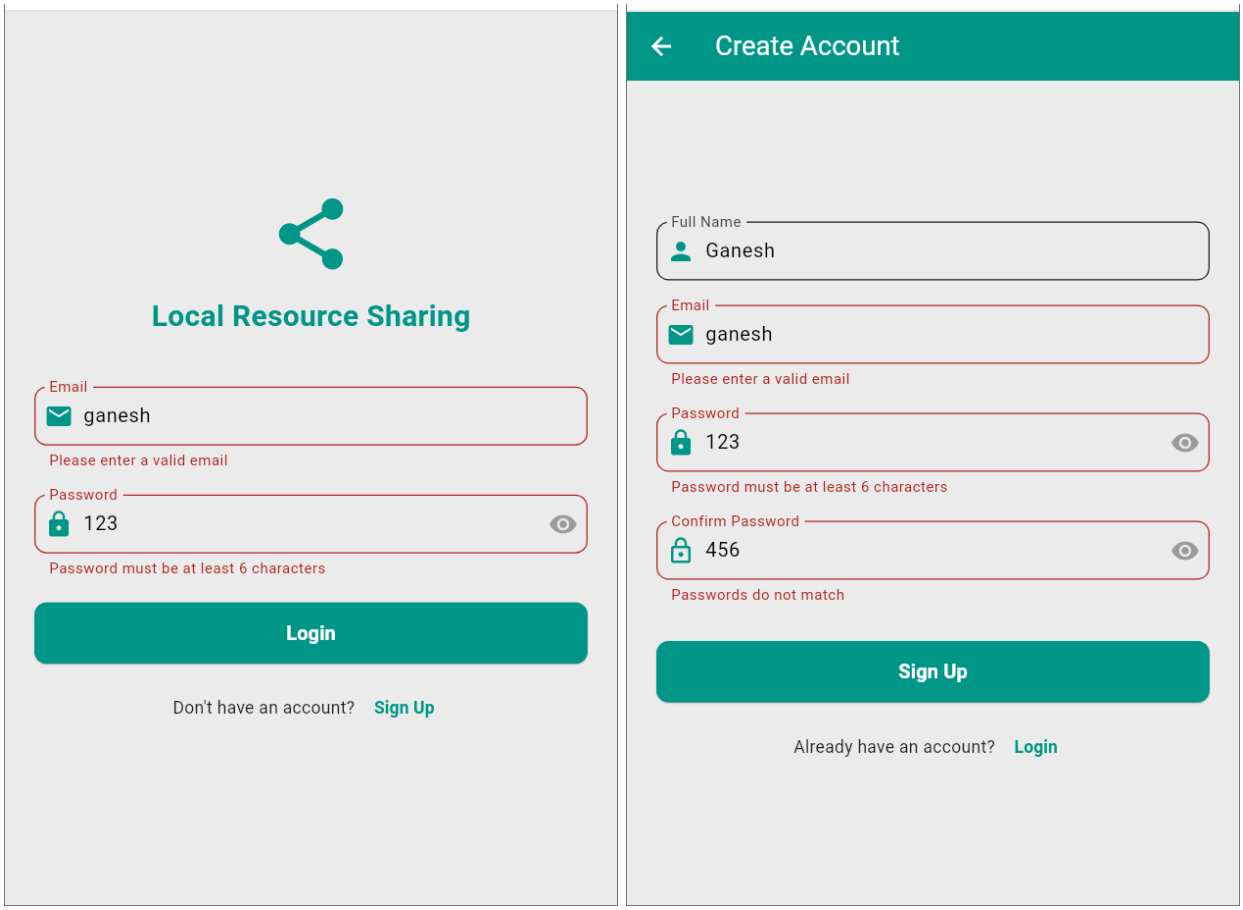
```
            await prefs.setBool('isLoggedIn', true);
            await prefs.setString('userName', _nameController.text);
            await prefs.setString('userEmail', _emailController.text);

            // Show success message
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(
                content: Text('Account created successfully!'),
                backgroundColor: Colors.green,
              ),
            );

            // Navigate to Home Screen
            Navigator.pushAndRemoveUntil(
              context,
              MaterialPageRoute(builder: (context) => HomeScreen()),
              (route) => false,
            );
          }
        },
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.teal,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
        ),
        child: Text(
          'Sign Up',
          style: TextStyle(
            color: Colors.white,
            fontSize: 16,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
    ),
  ),
  SizedBox(height: 20),

  // Login Link
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
```

```
                children: [
                  Text("Already have an account? "),
                  TextButton(
                    onPressed: () {
                      Navigator.pop(context);
                    },
                    child: Text(
                      'Login',
                      style: TextStyle(
                        color: Colors.teal,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```



Conclusion

In this experiment, we successfully implemented an interactive form using TextField, validation checks, and SnackBar feedback for user inputs. Initially, we faced issues with clearing input fields and displaying validation messages, which we resolved by using TextEditingController and condition checks before submission.