**Name:Bhushan Mukund Kor** **Academic Year:2024-2025**

**Division: D15C** **Roll No: 28**

# Experiment 4

**Aim:** To create an interactive Form using form widget

**Theory:**
Forms are a vital component of mobile applications, used to gather user inputs through elements like text fields, checkboxes, radio buttons, and dropdowns. Flutter simplifies form creation by offering the Form widget, which works alongside input widgets such as TextFormField, DropdownButtonFormField, and CheckboxListTile to build dynamic and validated user forms.

**Key Components:**
- Form Widget: Serves as a container that organizes and manages the state of multiple form fields.
- FormField Widgets: Input elements like TextFormField for user text input, DropdownButtonFormField for dropdown selections, and more.
- GlobalKey: A special key that helps in accessing the form's current state to perform validation and save operations.
- Validation: Ensures the data entered by the user is correct by applying conditions such as required fields and proper formats like email.

**Form Validation:**
- validator Property: Each field can include a validator function that returns an error message if the input doesn't meet specific criteria.
- FormState.validate(): Invokes all validators within the form and verifies the input across fields.
- FormState.save(): Saves the values of all form fields once validation is successful.

**Form Submission :**
A button, such as ElevatedButton, typically handles form submission.
When clicked, formKey.currentState!.validate() is used to verify all inputs.
If all entries are valid, formKey.currentState!.save() is executed to store or process the form data.

**Interactive Features:**
- Dynamic Fields: Allows adding or removing fields in response to user interactions or conditions.
- Real-time Validation: Provides immediate input feedback while the user is typing or selecting values.
- Custom Input Controls: Enables integration of advanced input types like sliders, switches, and date pickers within forms.

**Name:Bhushan Mukund Kor** **Academic Year:2024-2025**

**Division: D15C** **Roll No: 28**

**State Management in Forms:**
For straightforward forms, local state using setState() is sufficient.
However, in more advanced use cases, especially when dealing with multiple fields or app-wide state, it's advisable to use state management solutions such as Provider or Bloc for better control and maintainability.

**Code:**
**Form creation with global key and form field validation**

```
final _formKey = GlobalKey<FormState>();

Form(
          key: _formKey,
          child: Column(
            children: [
              _buildInput("Username", _usernameController),
              _buildInput("Email", _emailController, isEmail: true),
              _buildPasswordInput("Password", _passwordController, true),
              _buildPasswordInput("Confirm Password", _confirmController, false),
            ],
          ),
        ),
```

**Form submission**

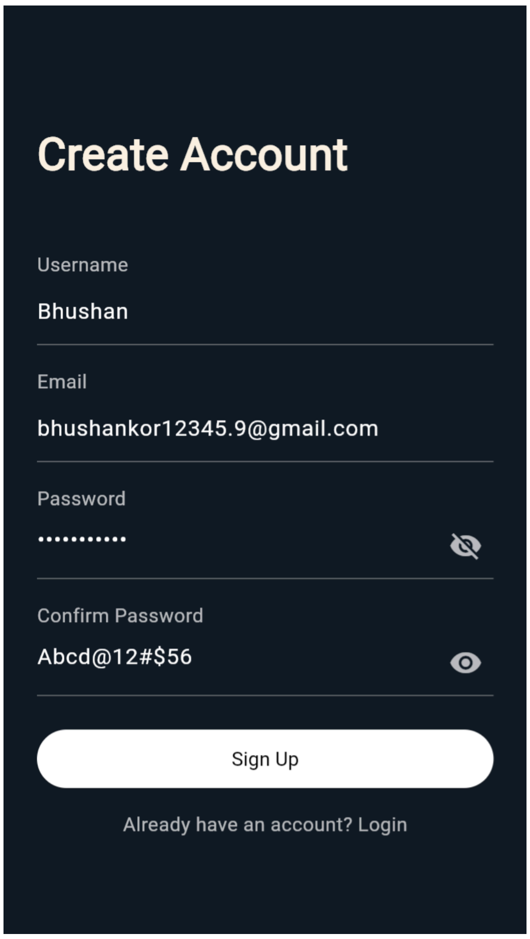
```
const SizedBox(height: 24),
        ElevatedButton(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.white,
            foregroundColor: Colors.black,
            minimumSize: const Size.fromHeight(50),
          ),
          onPressed: _handleSignup,
          child: const Text("Sign Up"),
        ),
 Future<void> _handleSignup() async {
   if (_formKey.currentState!.validate()) {
     if (_passwordController.text != _confirmController.text) {
       Fluttertoast.showToast(msg: "Passwords do not match");
       return;
     }
```

```
  try {
   UserCredential userCredential = await FirebaseAuth.instance
       .createUserWithEmailAndPassword(
         email: _emailController.text.trim(),
         password: _passwordController.text.trim(),
       );

   // Save user data to Firestore
   await FirebaseFirestore.instance
       .collection('users')
       .doc(userCredential.user!.uid)
       .set({
     'username': _usernameController.text.trim(),
     'email': _emailController.text.trim(),
     'uid': userCredential.user!.uid,
     'createdAt': Timestamp.now(),
   });

   Fluttertoast.showToast(msg: "Signup successful!");
   Navigator.pushReplacement(
       context, MaterialPageRoute(builder: (_) => LoginPage(onLoginSuccess: () {})));
  } catch (e) {
   if (e is FirebaseAuthException) {
     String errorMessage = '';
     switch (e.code) {
       case 'email-already-in-use':
         errorMessage = 'The email address is already in use by another account.';
         break;
       case 'invalid-email':
         errorMessage = 'The email address is not valid.';
         break;
       case 'weak-password':
         errorMessage = 'The password is too weak. It should be at least 6 characters.';
         break;
       default:
         errorMessage = 'An unknown error occurred: ${e.message}';
     }
     Fluttertoast.showToast(msg: errorMessage);
   } else {
     Fluttertoast.showToast(msg: "Error: ${e.toString()}");
   }
  }
 }
```

```
  }
```

**Form toggle**

```
Center(
              child: GestureDetector(
                onTap: () {
                  Navigator.pushReplacement(
                        context, MaterialPageRoute(builder: (_) => LoginPage(onLoginSuccess: ()
{})));
                },
                      child: const Text("Already have an account? Login", style: TextStyle(color:
Colors.white70)),
              ),
            )
```



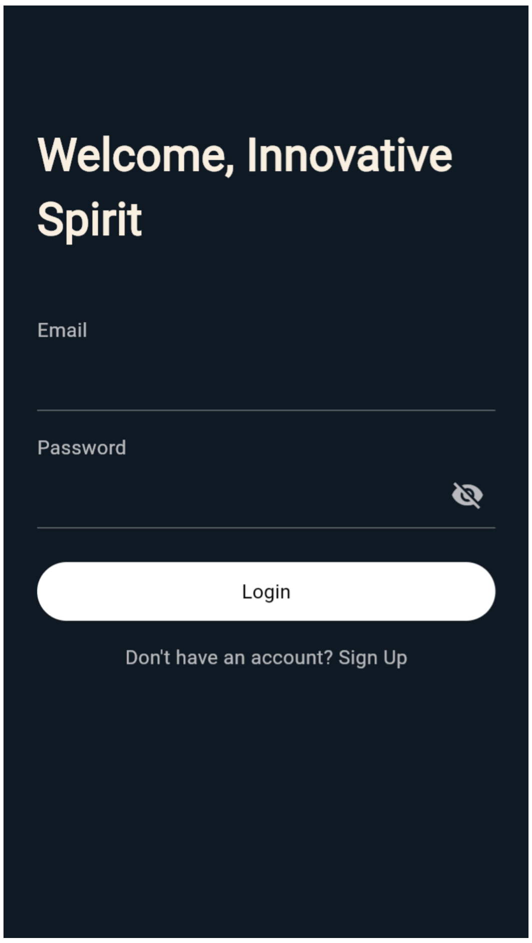
Create Account

Username: Bhushan

Email: bhushankor12345.9@gmail.com

Password: •••••••••••

Confirm Password: Abcd@12#$56



Welcome, Innovative Spirit

Email:

Password:

**Name:Bhushan Mukund Kor** **Academic Year:2024-2025**

**Division: D15C** **Roll No: 28**

**GitHub Link: https://github.com/BKCODE2003/ToDo**

**Conclusion:**
This experiment successfully demonstrated the creation of an interactive form in Flutter using the Form widget, form validation, and submission handling. By implementing TextFormField with validators, a toggle between login and sign-up modes, and Firebase authentication integration, we built a functional and user-friendly form that ensures data integrity and provides real-time feedback. This approach can be extended to more complex forms with additional input fields and advanced state management techniques.