

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

**Theory:** In Flutter, forms are created using the Form widget, which serves as a container for input fields (like TextFormField) and allows you to handle the validation and submission of data.

**Key Components:**

**1. Form Widget:**

- The Form widget is used to group multiple input fields. It provides the ability to manage and validate the form data.
- A GlobalKey<FormState> is required to track the form's state for validation and saving the data.

**2. TextFormField:**

- TextFormField is the primary widget for collecting text input in Flutter forms.
- It comes with built-in support for validation and user input handling.

**3. Validation:**

- The validator property of the TextFormField allows you to define rules that ensure user inputs are valid (e.g., email format, required fields).
- Flutter will automatically show error messages when the validation fails.

**4. FormState:**

- To manage the state of the form, you use FormState which allows you to validate, save, and reset the form.
- You can trigger validation by calling formKey.currentState?.validate().

**5. Saving Data:**

- Once the form is valid, you can save the data using the onSave property of the TextFormField.

## **Steps to Create an Interactive Form:**

### **1. Create a Form Widget:**

- You use the Form widget to wrap the form fields and manage validation.
- A `GlobalKey<FormState>` is used to access the form's state.

### **2. Add Form Fields (TextFormField):**

- For each input field, you use the TextFormField widget.
- Each TextFormField can have a validator to ensure the input is valid (e.g., ensuring that the user provides a valid email, password, etc.).
- Use `onSaved` to store the user input when the form is submitted.

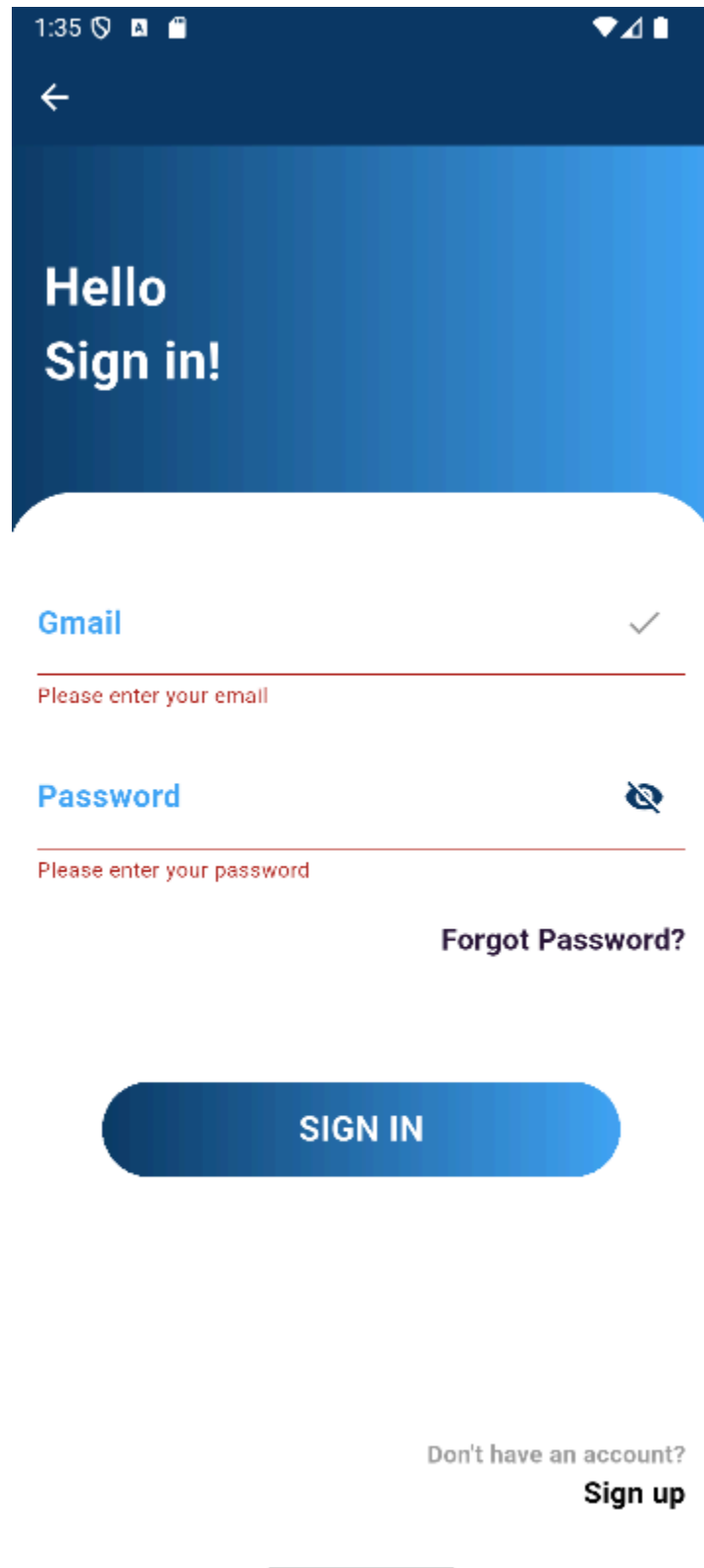
### **3. Validate the Form:**

- You can validate the form using the `formKey.currentState?.validate()` method, which triggers the validator for each field. If any field is invalid, it prevents form submission.

### **4. Handle Form Submission:**

- Once the form is validated, the form data is saved by calling `formKey.currentState?.save()`.
- The form can then be processed (e.g., sending data to a server, storing locally).

Output:



A mobile application login screen with a dark blue header and a light blue gradient background. The header contains a back arrow, the time 1:35, and status icons. The main content area has a white rounded rectangle containing two input fields: 'Email' with a checkmark icon and 'Password' with an eye icon. Below the inputs are links for 'Forgot Password?' and 'Sign up', and a 'SIGN IN' button. A horizontal line is at the bottom.

1:35

←

# Hello Sign in!

Email ✓

Please enter your email

Password 👁

Please enter your password

[Forgot Password?](#)

[Sign up](#)

**SIGN IN**

Don't have an account?

—

1:36



# Create Your Account

Full Name

Please enter Full Name

Email

Please enter Email

Password



Please enter Password

Confirm Password



Please enter Confirm Password

SIGN UP



## **Conclusion**

**In this experiment, you learned how to create an interactive form in Flutter using the Form widget. The Form widget allows for easy management of form fields, validation, and submission, while TextFormField provides the necessary functionality for input fields. By using validation and form state management, you can ensure that data entered by the user is valid before proceeding with any further operations, such as sending the data to a backend or storing it locally. This is a fundamental concept in Flutter for collecting and processing user input effectively.**