AIM: - To create an interactive Form using the **Form widget** in Flutter.

Theory:

In Flutter, a **Form** is a container for grouping and validating multiple form fields. It provides an easy way to manage user input, validate it, and handle form submission.

Flutter offers two key widgets for creating forms:

1. **Form Widget** – a container that holds multiple form fields and tracks their state.
2. **TextFormField Widget** – a specialized input field that integrates with the Form widget for validation and state tracking.

To use a form:

- We first create a GlobalKey<FormState> to uniquely identify the Form and allow validation.
- Inside the Form widget, we include one or more TextFormField widgets.
- Each TextFormField can have built-in validation using the validator parameter.
- We use a RaisedButton or ElevatedButton to trigger form submission and validation.

Important Components:

- GlobalKey<FormState>: A key to access the form state.
- Form: Widget to group multiple form fields.
- TextFormField: Input field with built-in validation support.
- validator: A function that returns an error string if validation fails or null if valid.
- formKey.currentState!.validate(): Triggers validation for all fields.

Form Validation Process:

1. User fills in data.
2. On submit, formKey.currentState!.validate() is called.
3. Each TextFormField runs its validator.
4. If all return null, form is valid.
5. Data can then be saved or processed.

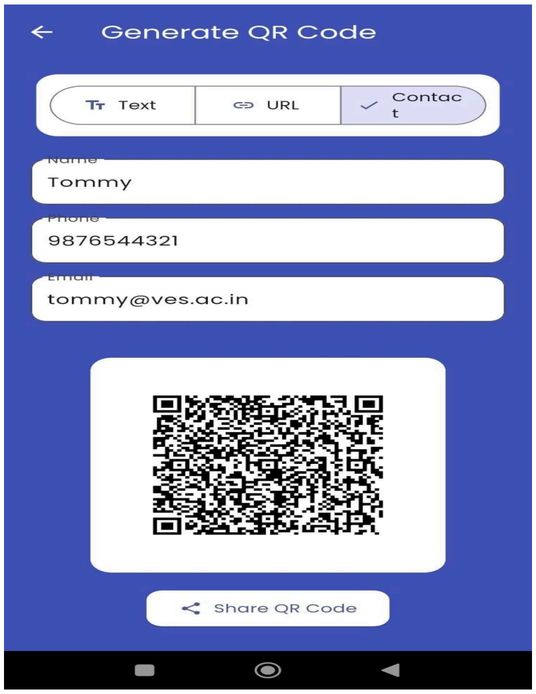
Use Cases:

- Login or signup forms
- Profile update pages
- Feedback or contact forms
- Surveys and data collection apps
- In our case to encode name ,contact and email in qr code.

Code:

```
final _formKey = GlobalKey<FormState>();
Form(
  key: _formKey,
  child: Column(
    children: [
      TextFormField(
        controller: _controllers['name'],
        decoration: InputDecoration(
          labelText: "Name",
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          filled: true,
          fillColor: Colors.white,
        ),
        validator: (value) {
          if (value == null || value.trim().isEmpty) {
            return 'Please enter a name';
          }
          return null;
        },
        onChanged: (_) {
          if (_formKey.currentState!.validate()) {
            setState(() {
              qrData = _generateQRData();
            });
          }
        },
      ),
      const SizedBox(height: 16),
      TextFormField(
        controller: _controllers['phone'],
        decoration: InputDecoration(
    labelText: "Phone",
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    filled: true,
    fillColor: Colors.white,
  ),
  keyboardType: TextInputType.phone,
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return 'Please enter a phone number';
    }
    if (!RegExp(r'^\d{10}$').hasMatch(value.trim())) {
      return 'Enter a valid 10-digit phone number';
    }
    return null;
  },
  onChanged: (_) {
    if (_formKey.currentState!.validate()) {
      setState(() {
        qrData = _generateQRData();
      });
    }
  },
),
const SizedBox(height: 16),
TextFormField(
  controller: _controllers['email'],
  decoration: InputDecoration(
    labelText: "Email",
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    filled: true,
    fillColor: Colors.white,
  ),
  keyboardType: TextInputType.emailAddress,
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return 'Please enter an email';
    }
    if (!RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w]{2,4}$')
          .hasMatch(value.trim())) {
        return 'Enter a valid email';
      }
      return null;
    },
    onChanged: (_) {
      if (_formKey.currentState!.validate()) {
        setState(() {
          qrData = _generateQRData();
        });
      }
    },
  ),
],
```

Output:



Name
Tommy
Phone
9876544321
Email
tommy@ves.ac.in

Conclusion:

In this experiment, we successfully created an interactive form using Flutter's Form and TextFormField widgets. We learned how to validate user input using validators and how to manage form state with a GlobalKey. This allows us to build structured, responsive, and user-friendly form-based interfaces in mobile applications.