Name: Niyati V. Gaonkar
Class: D15B Roll no.: 17

MAD-PWD Experiment - 4

AIM: To create interactive forms using flutter Widgets.

THEORY:

Flutter Forms

Forms are an integral part of all modern mobile and web applications. It is mainly used to interact with the app as well as gather information from the users. They can perform many tasks, which depend on the nature of your business requirements and logic, such as authentication of the user, adding user, searching, filtering, ordering, booking, etc. A form can contain text fields, buttons, checkboxes, radio buttons, etc.

Creating Form

Flutter provides a Form widget to create a form. The form widget acts as a container, which allows us to group and validate the multiple form fields. When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields.

The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Let us create a form. First, create a Flutter project and replace the following code in the main.dart file. In this code snippet, we have created a custom class named MyCustomForm. Inside this class, we define a global key as _formKey. This key holds a FormState and can use to retrieve the form widget. Inside the build method of this class, we have added some custom style and use the TextFormField widget to provide the form fields such as name, phone number, date of birth, or just a normal field. Inside the TextFormField, we have used InputDecoration that provides the look and feel of your form properties such as borders, labels, icons, hint, styles, etc. Finally, we have added a button to submit the form.

Build a form with validation

Apps often require users to enter information into a text field. For example, you might require users to log in with an email address and password combination.

- Create a Form with a GlobalKey.
- Add a TextFormField with validation logic.
- Create a button to validate and submit the form.

1)Create a Form with a GlobalKey.

If you made this a StatelessWidget, you'd need to store this key somewhere. As it is resource expensive, you wouldn't want to generate a new GlobalKey each time you run the build method.

```
import 'package:flutter/material.dart';
// Define a custom Form widget.
class MyCustomForm extends StatefulWidget {
  const MyCustomForm({super.key});
  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}
// Define a corresponding State class.
// This class holds data related to the form.
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
  //
  // Note: This is a `GlobalKey<FormState>`,
  // not a GlobalKey<MyCustomFormState>.
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    // Build a Form widget using the _formKey created above.
    return Form(
      key: _formKey,
```

```
      child: const Column(
        children: <Widget>[
          // Add TextFormFields and ElevatedButton here.
        ],
      ),
    );
  }
}
```

2). Add a TextFormField with validation logic

Although the Form is in place, it doesn't have a way for users to enter text. That's the job of a TextFormField. The TextFormField widget renders a material design text field and can display validation errors when they occur.

content_copy

```
TextFormField(
  // The validator receives the text that the user has entered.
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter some text';
    }
    return null;
  },
),
```

3). Create a button to validate and submit the form

Now that you have a form with a text field, provide a button that the user can tap to submit the information.

content_copy

```
ElevatedButton(
  onPressed: () {
    // Validate returns true if the form is valid, or false otherwise.
    if (_formKey.currentState!.validate()) {
      // If the form is valid, display a snackbar. In the real world,
      // you'd often call a server or save the information in a database.
```

```
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Processing Data')),
      );
    }
  },
  child: const Text('Submit'),
),
```

How does this work?
To validate the form, use the _formKey created in step 1. You can use the _formKey.currentState() method to access the FormState, which is automatically created by Flutter when building a Form.

The FormState class contains the validate() method. When the validate() method is called, it runs the validator() function for each text field in the form. If everything looks good, the validate() method returns true. If any text field contains errors, the validate() method rebuilds the form to display any error messages and returns false.

CODE:

main.dart :

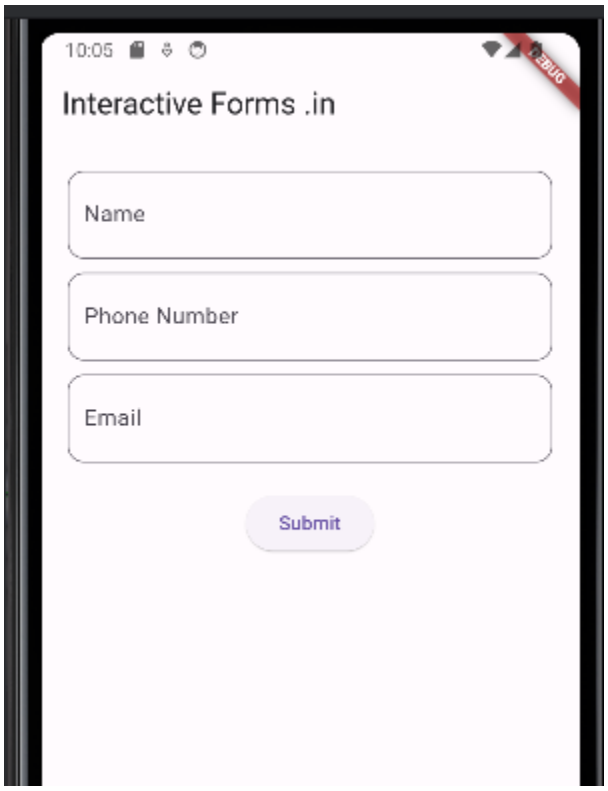
```dart
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Form Validation',
     theme: ThemeData(
       primarySwatch: Colors.blue,
     ),
     home: MyForm(),
```

```
    );
  }
}

class MyForm extends StatefulWidget {
  @override
  MyFormState createState() => MyFormState();
}

class MyFormState extends State<MyForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _phoneController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  String? _nameErrorText;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Form Validation'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              TextFormField(
                controller: _nameController,
                decoration: InputDecoration(
                  labelText: 'Name',
                  errorText: _nameErrorText,
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                ),
                validator: (value) {
                  if (value!.isEmpty) {
                    return 'Please enter your name';
                  }
                  if (value.contains(RegExp(r'[0-9]'))) {
                    return 'Invalid username format';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 10,),
              TextFormField(
```

```
            controller: _phoneController,
            decoration: InputDecoration(
              labelText: 'Phone Number',
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(12),
              ),
            ),
            keyboardType: TextInputType.phone,
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter your phone number';
              }
              if (!value.contains(RegExp(r'^[0-9]+$'))) {
                return 'Please enter valid phone number';
              }
              return null;
            },
          ),
          const SizedBox(height: 10,),
          TextFormField(
            controller: _emailController,
            decoration: InputDecoration(
              labelText: 'Email',
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(12),
              ),
            ),
            keyboardType: TextInputType.emailAddress,
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter your email';
              }
              if (!value.contains('@')) {
                return 'Please enter valid email';
              }
              return null;
            },
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              if (_formKey.currentState!.validate()) {
                // Form is valid, submit data
                // For now, let's print the form data
                print('Name: ${_nameController.text}');
                print('Phone Number: ${_phoneController.text}');
                print('Email: ${_emailController.text}');
              }
            },
```

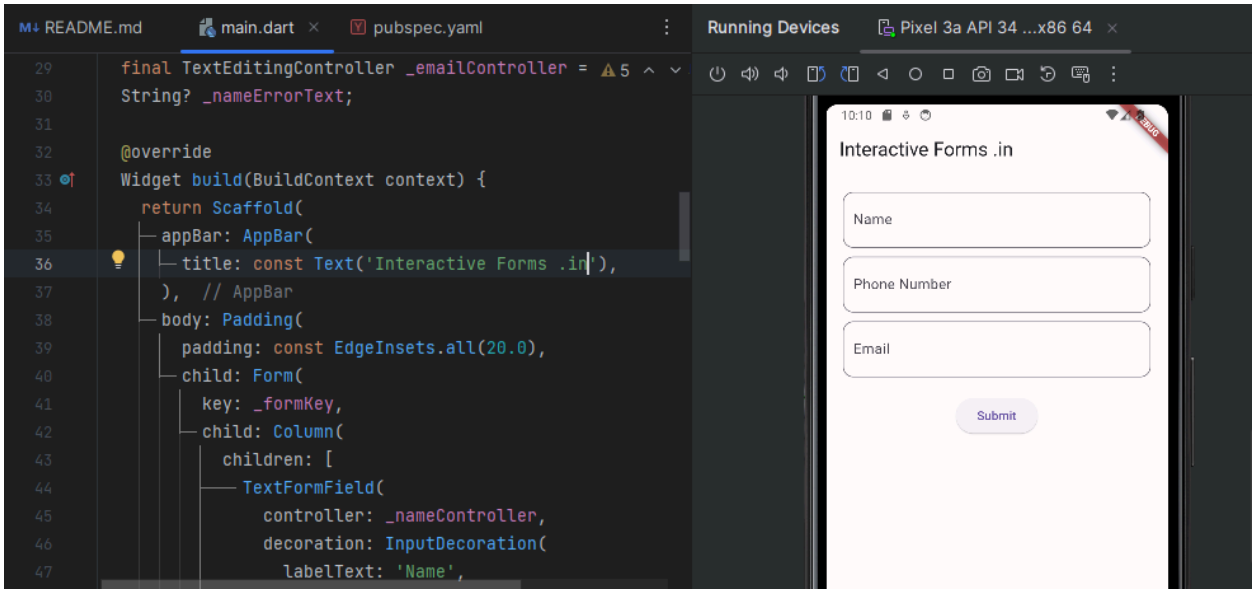
```
              child: const Text('Submit'),
            ),
          ],
        ),
      ),
    ),
  );
 }

 @override
 void dispose() {
   _nameController.dispose();
   _phoneController.dispose();
   _emailController.dispose();
   super.dispose();
 }
}
```

Output:

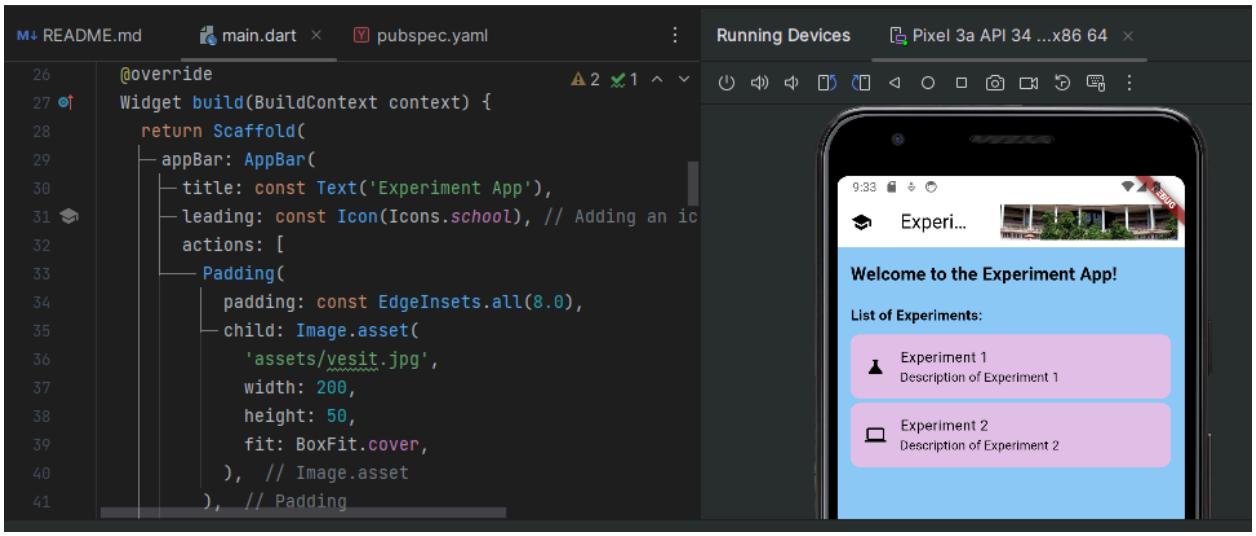


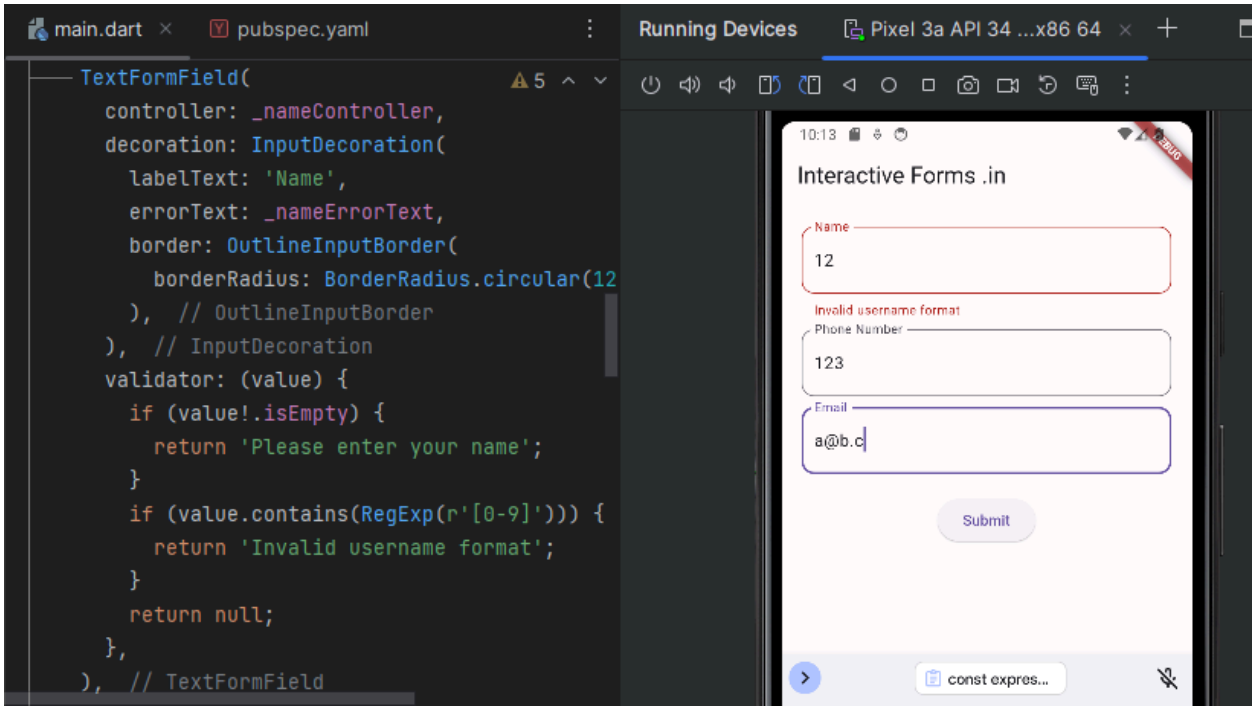
EXPLANATION :

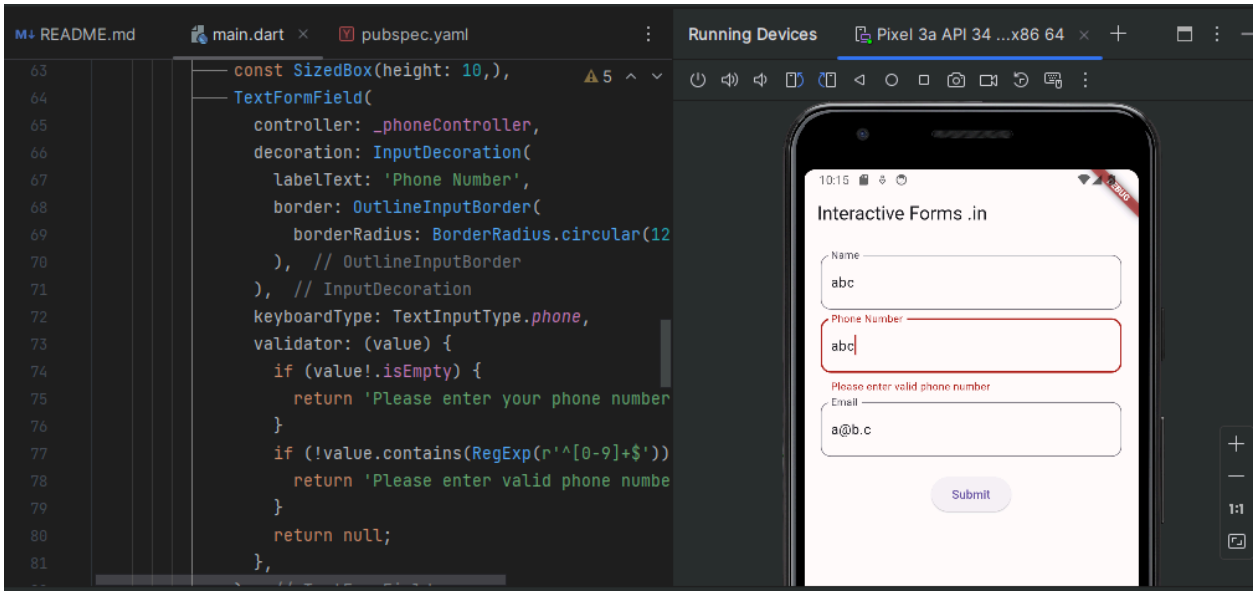
Step 1: Create a Form with a GlobalKey.



Step 2: Add validations to it.







Step 3: Form is ready!

CONCLUSION: Thus we have created an interactive form in Flutter using forms in flutter and it's different operations and functions. We have also added validations of name and number that shows error in case of invalid format upon submission.