Building Forms in Flutter

Shuja Akbar

- Forms enable collecting user input data.
- Essential for applications needing user registration, feedback, data entry, etc.
- Purpose of Forms in Flutter Applications
  - Commonly used in login pages, surveys, payment, and registration.

Form Widget

- What is Form in Flutter?
  - A container for form elements such as text fields, buttons, checkboxes, etc.
  - Works in conjunction with FormState to validate and save data.
- Key Properties of Form Widget
  - key: Uniquely identifies the form.
  - autovalidateMode: Configures when to validate automatically.

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

Form(
  key: _formKey,
  child: Column(
    children: [
      TextFormField(),
      ElevatedButton(onPressed: () {
        if (_formKey.currentState.validate()) {
          // Process data
        }
      }),
    ],
  ),
);
```

FormState for Managing Form Data

- FormState Class
  - Tracks form state and manages validation.
  - Common methods:
    - .validate(): Validates all fields in the form.
    - .save(): Saves the form’s current state.
- Why Use FormState?
- Centralized form handling for efficient data validation and processing.

- TextFormField Widget
  - Most commonly used field for accepting user input.
  - Built-in validation, error handling, and decoration options.
- Properties of TextFormField
  - controller: Links the field to a TextEditingController.
  - keyboardType: Sets the type of input (e.g., email, phone).
  - obscureText: Secures text for passwords.

```
TextFormField(
  controller: _textController,
  decoration: InputDecoration(
    labelText: "Enter your name",
    hintText: "John Doe",
  ),
  validator: (value) {
    if (value.isEmpty) return "Please enter a name";
    return null;
  },
);
```

Checkboxes and Switches

- Checkbox Widget
  - Represents Boolean values.
  - Properties: value, onChanged, activeColor.
- Switch Widget
  - Alternative to Checkbox, primarily for toggling states.
  - Commonly used for settings and preferences.

```
Checkbox(
  value: isChecked,
  onChanged: (bool newValue) {
    setState(() {
      isChecked = newValue;
    });
  },
);

Switch(
  value: isSwitched,
  onChanged: (bool newValue) {
    setState(() {
      isSwitched = newValue;
    });
  },
);
```

Radio Buttons and Dropdown Menus

- Radio Widget
  - Used for selecting one option from a list.
  - Properties: value, groupValue, onChanged.
- DropdownButton Widget
  - Provides a drop-down list of items for user selection.
  - Properties: value, items, onChanged.

```
Radio(
  value: "option1",
  groupValue: selectedOption,
  onChanged: (value) {
    setState(() {
      selectedOption = value;
    });
  },
);

DropdownButton<String>(
  value: selectedValue,
  items: <String>['Option 1', 'Option 2', 'Option 3'].map((String
value) {
    return DropdownMenuItem<String>(
      value: value,
      child: Text(value),
    );
  }).toList(),
  onChanged: (newValue) {
    setState(() {
      selectedValue = newValue;
    });
  },
);
```

Form Validation Techniques

- Types of Validation
  - Client-side: Validating on the form itself.
  - Server-side: Validation upon submission.
- Flutter Form Validation Methods
  - .validate() on FormState for overall validation.
  - validator property on each TextFormField for field-level validation.

```
validator: (value) {
  if (value == null || value.isEmpty) {
    return 'Please enter some text';
  }
  return null;
}
```

TextEditingController

- Purpose of TextEditingController
  - Captures, modifies, and clears text field data.Also provides listening for text changes.
- Setting Up TextEditingController

```
final _controller = TextEditingController();

TextFormField(
  controller: _controller,
  decoration: InputDecoration(labelText: 'Username'),
);
```

```
class LoginForm extends StatefulWidget {
  @override
  _LoginFormState createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
  // Form key to uniquely identify the form
  final _formKey = GlobalKey<FormState>();

  // TextEditingControllers to manage input fields
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  // Function to handle form submission
  void _submitForm() {
    if (_formKey.currentState!.validate()) {
      // All validations have passed
      String email = _emailController.text;
      String password = _passwordController.text;

      // Displaying the entered values
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Email: $email, Password: $password')),
      );

      // TODO: Add further form submission logic here (e.g., sending data to
server)
    }
  }
```

```
 @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Email field
          TextFormField(
            controller: _emailController,
            decoration: InputDecoration(
              labelText: 'Email',
              border: OutlineInputBorder(),
            ),
            keyboardType: TextInputType.emailAddress,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter your email';
              } else if
(!RegExp(r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$")
                  .hasMatch(value)) {
                return 'Please enter a valid email';
              }
              return null;
            },
          ),
          SizedBox(height: 16.0),
          // Password field
          TextFormField(
            controller: _passwordController,
            decoration: InputDecoration(
              labelText: 'Password',
              border: OutlineInputBorder(),
            ),
            obscureText: true,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter your password';
              } else if (value.length < 6) {
                return 'Password must be at least 6 characters';
              }
              return null;
            },
          ),
          SizedBox(height: 24.0),

          // Submit button
          ElevatedButton(
            onPressed: _submitForm,
            child: Text('Submit'),
          ),
        ],
      ),
    );
  }

@override
  void dispose() {
    // Dispose the controllers when the widget is removed
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
}
```