Module - 6

1. Structure and Purpose of Forms in Flutter

In Flutter, a **form** is a container for grouping and validating multiple input fields. It is built using the Form widget, which works with FormField widgets such as TextFormField.

Structure:

- The Form widget is used to wrap multiple input fields.
- Each input field (like TextFormField) is a form field that can be validated.
- A GlobalKey<FormState> is typically used to manage the form's state for validation and saving.

Purpose:

- To collect and validate user input.
- To manage multiple input fields in a structured and consistent way.
- To allow validation and saving logic to be centralized and reusable.

---

2. Controllers and Listeners in Form Input Management

Controllers (e.g., TextEditingController) are used to programmatically access and modify the value of input fields.

- Each TextFormField or TextField can be linked to a controller.
- The controller allows reading the input value or setting it dynamically.

Listeners can be added to controllers to track changes in real-time.

- A listener function can be attached to a controller to execute code when the user types or changes input.
- Useful for live validation, UI updates, or dynamic behavior based on user input.

---

3. Common Form Validation Techniques and Examples

Form validation ensures that user input meets certain criteria before being processed. In Flutter, validation is often handled using the validator property of TextFormField.

Techniques:

- **Required field check**
  Ensures the field is not left empty.
  Example:
- validator: (value) => value == null || value.isEmpty ? 'This field is required' : null;
- **Email format validation**
  Checks if input matches a valid email pattern.
  Example:
- validator: (value) => !value.contains('@') ? 'Enter a valid email' : null;
- **Password strength check**
  Validates minimum length, special characters, etc.
  Example:
- validator: (value) => value.length < 8 ? 'Password too short' : null;
- **Custom logic validation**
  Allows complex conditions specific to the application.

Validation is typically triggered using the FormState.validate() method.