1. Explain the structure and purpose of forms in Flutter.

Answer:

- **Structure of Forms in Flutter:**
  - **Flutter provides the Form widget, which acts as a container for form fields (like TextFormField).**
  - **A GlobalKey<FormState> is usually used to uniquely identify and manage the form.**
  - **Each input field can have validators to check correctness.**
- **Purpose of Forms in Flutter:**
  - **Forms are used to collect, validate, and submit user input such as login, registration, or feedback.**
  - **They help manage multiple input fields together instead of validating them individually.**

Example:

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


Form(
 key: _formKey,
 child: Column(
  children: [
   TextFormField(
    decoration: InputDecoration(labelText: "Email"),
    validator: (value) {
     if (value == null || value.isEmpty) return "Enter email";
     return null;
    },
   ),
   ElevatedButton(
    onPressed: () {
```

```
        if (_formKey.currentState!.validate()) {
          print("Form Submitted");
        }
      },
      child: Text("Submit"),
    ),
  ],
 ),
);
```

---

2. Describe how controllers and listeners are used to manage form input.

Answer:

- **Controllers:**
  - **TextEditingController is used to control and retrieve the text inside a TextField or TextFormField.**
  - **It allows reading the current value and modifying it programmatically.**
- **Listeners:**
  - **Controllers can have listeners attached to detect changes in input.**
  - **Useful for real-time validation, updating UI, or enabling/disabling buttons.**

Example:

```
final TextEditingController emailController = TextEditingController();


@override
void initState() {
 super.initState();
 emailController.addListener(() {
  print("Current email: ${emailController.text}");
 });
```

}

TextField(controller: emailController);

---

3. List some common form validation techniques and provide examples.

Answer:
Common validation techniques in Flutter:

1. **Required Field Validation**
    - **Ensures the user doesn’t leave a field empty.**
2. **validator: (value) =>**
3. **value == null || value.isEmpty ? "This field is required" : null;**
4. **Email Format Validation**
    - **Uses regex to check for proper email structure.**
5. **validator: (value) {**
6. **if (value == null || value.isEmpty) return "Enter email";**
7. **if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value))**
8. **return "Enter valid email";**
9. **return null;**
10. **}**
11. **Password Strength Validation**
    - **Checks for minimum length, uppercase letters, or special characters.**
12. **validator: (value) {**
13. **if (value == null || value.length < 6) return "Password too short";**
14. **return null;**
15. **}**
16. **Numeric / Phone Number Validation**
    - **Ensures only digits are entered.**
17. **validator: (value) {**

```
    if (value == null || !RegExp(r'^[0-9]+$').hasMatch(value))
      return "Enter valid number";
    return null;
}
```