Module-4

Q.1 Explain how the Navigator widget works in Flutter.

- Ans=>**Navigator = stack of screens (routes).**
- push → add new screen on top.
- pop → remove current screen, go back.
- pushReplacement → replace current screen.
- pushAndRemoveUntil → clear old screens, open new one.
- **Named routes** → use route names instead of widgets for navigation.

Simply: **Navigator works like a stack → last screen opened is the first one closed.**

Q.2 Describe the concept of named routes and their advantages over direct route navigation.

Ans=>**Named Routes**

- Instead of writing MaterialPageRoute everywhere, you **assign a name (string)** to each screen.
- Define all routes in MaterialApp → routes.

- Navigate using Navigator.pushNamed(context, '/routeName').

Advantages:

- Cleaner & shorter code (no need to repeat MaterialPageRoute).
- Easier to manage in large apps (all routes in one place).
- Supports **initialRoute** & deep linking.
- Improves readability & maintainability.

Q.3 Explain how data can be passed between screens using route arguments.

Ans=>1.Direct route (constructor)

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => SecondScreen(data: "Hello")),
);
```

2. Named routes (arguments):

```
Navigator.pushNamed(
  context,
```

```
  '/second',
  arguments: "Hello",
);
```

Module-6

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

Ans=>**Structure:**

- **Form widget** → container for form fields.
- **GlobalKey<FormState>** → tracks form state (validate, save, reset).
- **TextFormField / other fields** → input widgets inside the form.
- **validator** → function to check input validity.
- **Buttons** → to submit or reset the form.

Purpose:

- Collect multiple inputs (like login, signup, feedback).
- Validate user input (e.g., email format, required fields).
- Manage form state (save, reset, validate).

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

Ans=>**1. Controllers (TextEditingController)**

- Used to **control and access input field values**.

2. Listeners

- Detect changes in input field.

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

Ans=>**Common Form Validation Techniques in Flutter**

1. **Required field check** → Input must not be empty.
2. **Format check** → Correct pattern (email, phone, etc.).
3. **Length check** → Minimum/maximum characters (like password).
4. **Range check** → Numbers within a valid range (like age).
5. **Comparison check** → Fields must match (like confirm password).
6. **Custom validation** → App-specific rules (like unique username).