Module - 4

1. How the Navigator Widget Works in Flutter

The **Navigator** widget in Flutter is used to manage a stack of route objects. It enables navigation between different screens (or "routes") by **pushing** new routes onto the stack or **popping** them off the stack. The top-most route in the stack is the currently visible screen.

- **Push**: Adds a new route to the stack and displays it.
- **Pop**: Removes the current route from the stack and returns to the previous one.

This stack-based approach allows Flutter to keep track of the user's navigation history and manage transitions smoothly.

---

2. Concept of Named Routes and Their Advantages Over Direct Route Navigation

Named routes are a way to navigate between screens using a string identifier instead of directly instantiating a widget.

- Defined in the MaterialApp or CupertinoApp widget using the routes property.
- Navigation is done using Navigator.pushNamed(context, '/routeName').

Advantages:

- **Centralized route management**: All routes are defined in one place.
- **Code readability**: Easier to understand and maintain large apps.
- **Parameter passing**: Cleaner way to pass arguments between screens.
- **Scalability**: Better suited for apps with many screens or complex navigation.

---

3. How Data Can Be Passed Between Screens Using Route Arguments

In Flutter, data can be passed between screens using the **arguments** parameter in named routes.

- When navigating, use Navigator.pushNamed(context, '/screen', arguments: data).
- In the receiving screen, access the data using ModalRoute.of(context)?.settings.arguments.

This allows passing complex objects or values from one screen to another in a structured way, improving modularity and reusability of screens.