# Assignment

## MODULE: 4

## Navigation And Routing

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

- **Ans.**

The `Navigator` widget in Flutter manages a **stack of routes (pages)**, allowing navigation between screens using `push()` to go to a new page and `pop()` to return. It's like a page stack where the top page is visible. Flutter has:

**Navigator 1.0 (Imperative)**: Uses `push`/`pop` methods.

**Navigator 2.0 (Declarative)**: Uses a `pages` list and is better for deep linking and state-based navigation.

Use it to switch screens, show dialog, or handle complex navigation flows.

**Q.2_Describe the concept of named routes and their advantages over direct route Navigation.**

- **Ans.**

**Named routes** in Flutter use string identifiers to navigate between screens instead of directly creating route objects.

Example:

```
Navigator.pushNamed(context, '/details');
```

Advantages over direct navigation:

- **Centralized route management** in `MaterialApp`'s `routes` map.
- **Cleaner code**: Avoids repeating route creation logic.
- **Easier navigation** with dynamic routing and deep links.
- **Better maintainability** in large apps.

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

- **Ans.**

In Flutter, data can be passed between screens using **route arguments** during navigation.

### Passing Data (with Named Routes)

**1. Pass arguments when navigating:**

```
Navigator.pushNamed(
  context,
  '/details',
  arguments: 'Hello from Home!',
);
```

**2. Receive arguments in the target screen:**

```
class DetailsPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final String message = ModalRoute.of(context)!.settings.arguments
as String;

    return Scaffold(
      body: Center(child: Text(message)),
    );
  }
}
```

Use the `arguments` parameter in `pushNamed()`.

Access it via `ModalRoute.of(context)!.settings.arguments` in the destination screen..