Name: Vidhi Gulhane
Roll No: 15
Class: D15B

Experiment No: 05

Aim: To Apply Navigation, Routing, and Gestures in a Flutter App

Objective:

The objective of this experiment is to understand and implement navigation between screens, routing using named and anonymous routes, and handling gesture detection in a Flutter application to create interactive and dynamic user interfaces.

Theory:

Flutter provides built-in support for **navigation**, **routing**, and **gesture handling**, which are essential for creating multi-screen, interactive applications.

1. Navigation in Flutter

Navigation allows users to move between different screens (also called routes or pages). Flutter uses a **stack-based navigation system** — meaning new screens are pushed onto the navigation stack, and going back pops them off.

- **Navigator Class:**
    - Used to manage routes.
    - Key methods:
        - Navigator.push(context, route) – navigates to a new screen.
        - Navigator.pop(context) – returns to the previous screen.

2. Routing in Flutter

Routing determines how users move between screens and how screens are identified.

- **Anonymous Routing:** Uses MaterialPageRoute directly.
- **Named Routing:** Defines route names in the MaterialApp and navigates using the name.

3. Gestures in Flutter

Gestures refer to user interactions like tapping, swiping, dragging, etc. Flutter provides gesture detection through the `GestureDetector` widget.

- Common gestures:
  - onTap
  - onDoubleTap
  - onLongPress
  - onPanUpdate (for drag)

Key Widgets and Classes:

Feature	**Widget/Class**	**Purpose**
Navigation	Navigator	Manages stack of routes
Routing	MaterialPageRoute, routes	Defines route transitions
Gestures	GestureDetector, InkWell	Detects and responds to gestures

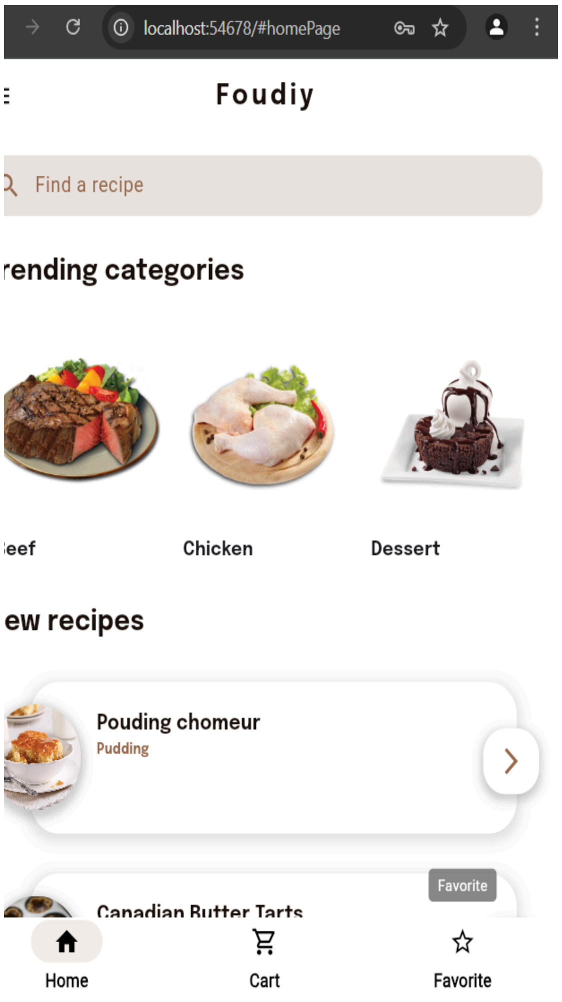
Benefits of Navigation & Gestures in Apps:

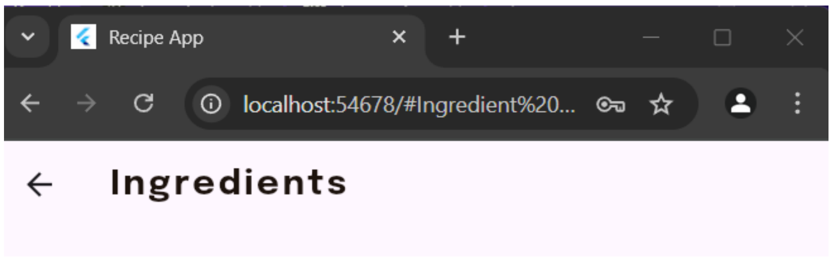
- Enables building multi-screen applications.
- Improves user experience with smooth transitions.
- Allows interactive UI elements using gestures.

Conclusion:

In this experiment, we successfully applied navigation, routing, and gesture detection in a Flutter app. We explored how to move between screens using `Navigator`, how to configure named and anonymous routes, and how to use `GestureDetector` for handling user interactions. Mastering these core features allows developers to build seamless, interactive, and scalable applications with a better user experience.

Output:


localhost:54678/#homePage
Foudiy
Find a recipe
rending categories
eef
Chicken
Dessert
ew recipes
Pouding chomeur
Pudding
Favorite
Canadian Butter Tarts
Home
Cart
Favorite


Recipe App
localhost:54678/#Ingredient%20...
Ingredients