Experiment No. 5

AIM : To apply navigation, routing and gestures in Flutter App

Theory:

1. Navigation and Routing

In Flutter, navigation refers to moving from one screen (or "route") to another. There are several key concepts:

- **Routes**: These are the different screens or pages in your app. Every route is typically represented by a Widget in Flutter. The default route is usually the home screen of the app, but you can define multiple routes for different screens.

- **Navigator**: This is a widget that manages a stack of routes. You can "push" a new route onto the stack to navigate to another screen, or "pop" the top route off to go back.

- **Named Routes**: These are routes that are identified by a string. Instead of pushing or popping routes directly, you can refer to routes by their name (e.g., /home, /settings).

- **Custom Route Transitions**: Flutter allows you to define custom animations and transitions when navigating between routes. You can create smooth, custom page transitions using PageRouteBuilder.

- **Route Arguments**: You can pass data between routes using arguments. This is particularly useful when navigating to a screen that requires specific data (e.g., opening a product page with product details).

2. Gestures in Flutter

Gestures are interactions that a user performs with the screen, such as taps, swipes, or long presses. Flutter provides a flexible way to detect these gestures.

- **GestureDetector**: This is the most commonly used widget for detecting gestures. You can wrap it around any widget to detect gestures like tap, double tap, long press, swipe, and others.

- **Tap Gesture**: A simple touch interaction, typically detected using onTap or onLongPress callbacks.

- **Swipe Gestures**: Swiping is usually detected via onHorizontalDragUpdate, onVerticalDragUpdate, or onPanUpdate. These allow you to track the user's finger movement and respond accordingly.

- **Custom Gesture Detection**: Flutter also allows you to implement more complex gestures. For example, you can detect drag gestures to create features like a sliding menu or draggable elements.

- **Dismissible Widget**: This widget enables swipe-to-dismiss behavior, commonly used for items in a list that users can swipe left or right to remove.

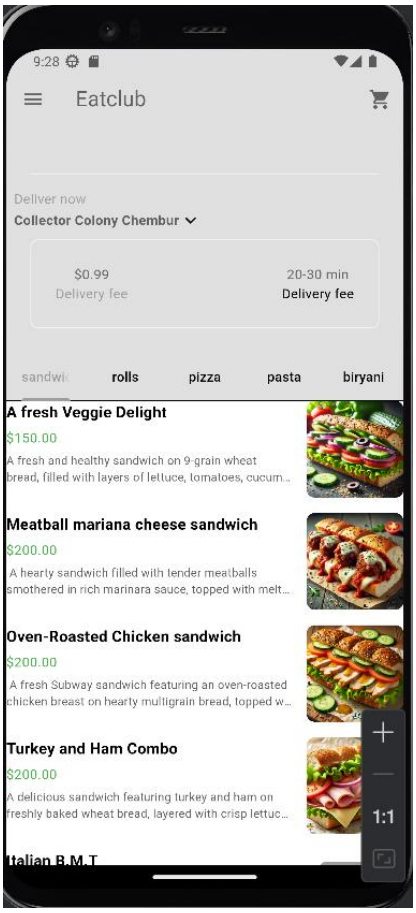
3. Managing Navigation and Gestures Together

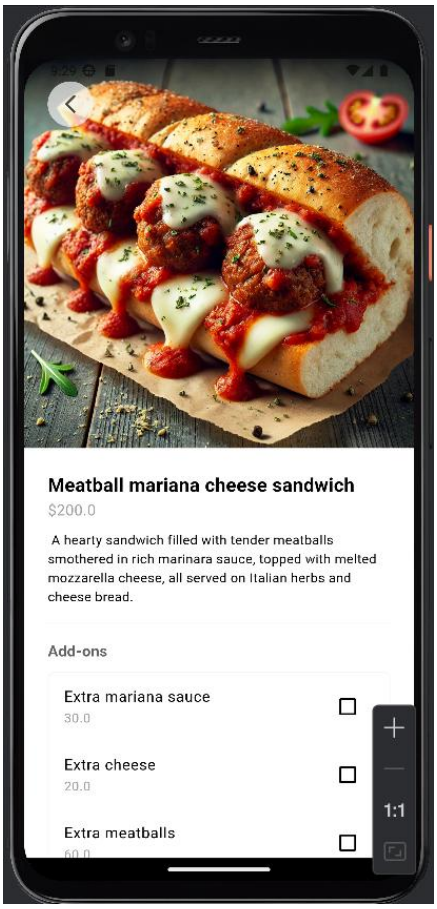
When you combine navigation with gestures, you can create more interactive and dynamic UIs. For instance, a user could swipe to navigate between screens, or tap a button that triggers navigation while performing a gesture on a different part of the screen.

4. Back Button Handling

On Android devices, there is a system-wide back button that users can press to navigate backward. Flutter provides a way to intercept and customize this behavior using WillPopScope, allowing you to decide what happens when the user tries to go back (e.g., prevent the user from leaving the current screen, show a confirmation dialog, or allow normal back navigation).

Screenshots:





Code Snippets:

Food Page:

```
import 'package:flutter/material.dart';
import 'package:flutter_eats/components/my_button.dart';
import 'package:flutter_eats/models/food.dart';
import 'package:flutter_eats/models/restaurant.dart';
import 'package:provider/provider.dart';

class FoodPage extends StatefulWidget {
  final Food food;
  final Map<Addon,bool> selectedAddons ={};

  FoodPage({
    super.key,
    required this.food,
  }) {
    // initialize selected addons to be false
    for (Addon addon in food.availableAddons) {
      selectedAddons[addon] = false;
    }
  }

  @override
  State<FoodPage> createState() => _FoodPageState();
}

class _FoodPageState extends State<FoodPage> {

  //method to add to cart
  void addTocart(Food food, Map<Addon, bool>selectedAddons) {
    // close the current food page to go back to menu
    Navigator.pop(context);

    //format the selected addon
    List<Addon> currentlySelectedAddons = [];
    for (Addon addon in widget.food.availableAddons) {
      if (widget.selectedAddons[addon] == true) {
        currentlySelectedAddons.add(addon);
      }
    }

    // add to cart
    context.read<Restaurant>().addTocart(food, currentlySelectedAddons);
  }
```

```
@override
Widget build(BuildContext context) {
  return Stack(children: [
    //scaffold UI
  Scaffold(
  body: SingleChildScrollView(
  child: Column(
  children: [
  //food image
  Image.asset(widget.food.imagePath),

  Padding(
  padding: const EdgeInsets.all(25.0),
  child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
  // food name
  Text(
  widget.food.name,
  style: const TextStyle(
  fontWeight: FontWeight.bold,
  fontSize: 20,
  ),
  ),

  //food price
  Text(
  '\$'+ widget.food.price.toString(),
  style: TextStyle(
  fontSize: 16,
  color: Theme.of(context).colorScheme.primary,
  ),
  ),

  const SizedBox(height: 10),

  //food description
  Text(
  widget.food.description,
  ),
  const SizedBox(height: 10),

  Divider(color: Theme.of(context).colorScheme.secondary),
```

```
const SizedBox(height: 10),

//addons
Text(
"Add-ons",
style: TextStyle(
color: Theme.of(context).colorScheme.inversePrimary,
fontSize: 16,
fontWeight: FontWeight.bold,
),
),
const SizedBox(height: 10),

Container(
decoration: BoxDecoration(
border: Border.all(
color: Theme.of(context).colorScheme.secondary),
borderRadius: BorderRadius.circular(8),
),
child: ListView.builder(
shrinkWrap: true,
physics: const NeverScrollableScrollPhysics(),
padding: EdgeInsets.zero,
itemCount: widget.food.availableAddons.length,
itemBuilder: (context, index) {
//get individual addons
Addon addon = widget.food.availableAddons[index]; // Fixed syntax

//return check box UI
return CheckboxListTile(
title: Text(addon.name),
subtitle: Text(
addon.price.toString(),
style: TextStyle(
color: Theme.of(context).colorScheme.primary,
),

),
value: widget.selectedAddons[addon],
onChanged: (bool? value) {
setState(() {
widget.selectedAddons[addon] = value!;
```

```
});
},
);
},
),
)
],
),
),

// button add to cart
MyButton(
onTap: () => addTocart(widget.food, widget.selectedAddons),
text: "Add to cart"),

const SizedBox(height: 25),
],
),)
),



  //back button
  SafeArea(
     child: Opacity(
       opacity: 0.7,
       child: Container(
         margin: const EdgeInsets.only(left:25),
         decoration: BoxDecoration(
           color: Theme.of(context).colorScheme.secondary,
           shape: BoxShape.circle,
         ),
         child: IconButton(
            onPressed: ()=> Navigator.pop(context),
            icon: const Icon(Icons.arrow_back_ios_new_rounded)),
       ),
     ))
  ],);
 }
}
```

Food.dart file

```
class Food{
  final String name;
```

```
  final String description;
  final String imagePath;
  final double price;
  final FoodCategory category;
  List<Addon> availableAddons;

  Food( {
    required this.name,
    required this.description,
    required this.imagePath,
    required this.price,
    required this.category,
    required this.availableAddons,
  });
}

// food categories
enum FoodCategory{
  sandwiches,
  rolls,
  pizza,
  pasta,
  biryani,
}

// food addons
class Addon {
  String name;
  double price;

  Addon({
    required this.name,
    required this.price,
});
}
```