Experiment No. 5

**Aim:-**

To apply navigation, routing and gestures in Flutter App

**Theory:-**

Navigation in Flutter allows users to move between different screens (or pages) in the app. Flutter uses the Navigator widget to handle navigation between routes (screens).

Types of Navigation

● Push Navigation (Forward Navigation) → Moves to a new screen.

● Pop Navigation (Backward Navigation) → Moves back to the previous screen.

● PushReplacement → Replaces the current screen with a new one.

● PushAndRemoveUntil → Moves to a new screen and removes previous screens from the stack.

Types of Routing

1.Direct Route Navigation (MaterialPageRoute)-Used for simple page-topage navigation. 2. Named Routes (Predefined Routes in main.dart)-Defined in the MaterialApp widget and used throughout the app.

Flutter uses the GestureDetector widget to detect user interactions like taps, swipes, pinches, and long presses. This is essential for making an app interactive.

Common Gestures & Their Uses:

● Tap → Detects simple taps on a widget.

● Double Tap → Recognizes double-clicking.

● Long Press → Triggers an action when the user presses and holds.

● Swipe (Drag) → Detects horizontal or vertical dragging.

● Pinch (Zoom In/Out) → Detects two-finger pinch for zooming.

**Code:-**

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:flutter\_svg/flutter\_svg.dart';

import 'package:foodie/constants/app\_constants.dart';

import 'package:foodie/router/scaffold\_with\_bottom\_nav\_controller.dart';

import 'package:go\_router/go\_router.dart';

import '../constants/string\_constants.dart';

import '../features/recipes/presentation/recipe\_controller.dart';

import '../providers/providers.dart';

import '../theme/theme.dart';

import 'app\_route.dart';

class ScaffoldWithBottomNavBar extends ConsumerStatefulWidget {

  const ScaffoldWithBottomNavBar({

    required this.child,

    super.key,

  });

  final Widget child;

  @override

  ConsumerState<ScaffoldWithBottomNavBar> createState() =>

      \_ScaffoldWithBottomNavBarState();

}

class \_ScaffoldWithBottomNavBarState

    extends ConsumerState<ScaffoldWithBottomNavBar> {

  void \_tap(int index) {

    ref.read(scaffoldBottomNavControllerProvider.notifier).setPosition(index);

    ref.invalidate(ingredientMultiplierProvider);

    ref.invalidate(instructionsCounterProvider);

    if (index == 0) {

      context.goNamed(AppRoute.home.name);

      ref.read(recipeControllerProvider.notifier).ref.invalidateSelf();

    } else if (index == 1) {

      context.goNamed(AppRoute.challenges.name);

    } else if (index == 2) {

      context.goNamed(AppRoute.saved.name);

    } else if (index == 3) {

      context.goNamed(AppRoute.friends.name);

    } else {

      context.goNamed(AppRoute.profile.name);

    }

  }

  @override

  Widget build(BuildContext context) {

    int position = ref.watch(scaffoldBottomNavControllerProvider);

    return Scaffold(

      body: widget.child,

      bottomNavigationBar: BottomNavigationBar(

        currentIndex: position,

        onTap: (index) => \_tap(index),

        items: [

          \_buildBottomNavBarItemWidget(Assets.icons.home, StringConstants.home),

          \_buildBottomNavBarItemWidget(

              Assets.icons.challenges, StringConstants.challenges),

          \_buildBottomNavBarItemWidget(

              Assets.icons.saved, StringConstants.saved),

          \_buildBottomNavBarItemWidget(

              Assets.icons.friends, StringConstants.friends),

          \_buildBottomNavBarItemWidget(

              Assets.icons.profile, StringConstants.profile),

        ],

      ),

    );

  }

  BottomNavigationBarItem \_buildBottomNavBarItemWidget(

      String iconName, String label) {

    return BottomNavigationBarItem(

        icon: SvgPicture.asset(

          iconName,

          height: AppConstants.bottomNavbarIconHeight,

        ),

        activeIcon: SvgPicture.asset(

          iconName,

          height: AppConstants.bottomNavbarIconHeight,

          theme: const SvgTheme(

            currentColor: ThemeColors.primary,

          ),

          colorFilter:

              const ColorFilter.mode(ThemeColors.primary, BlendMode.srcATop),

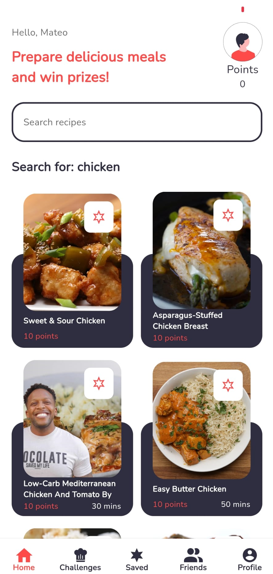
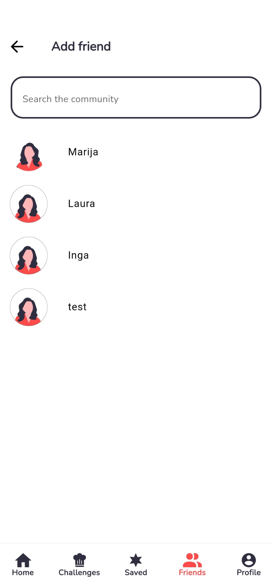
        ),

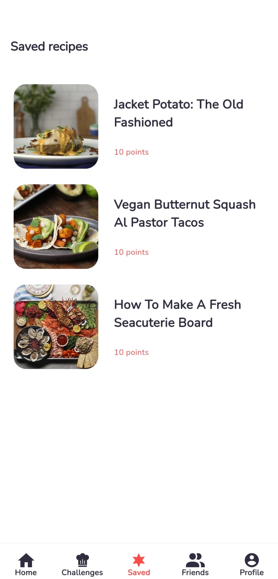
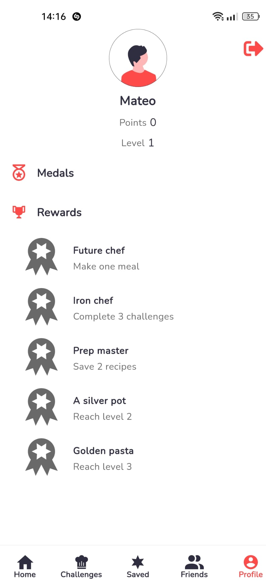
        label: label);

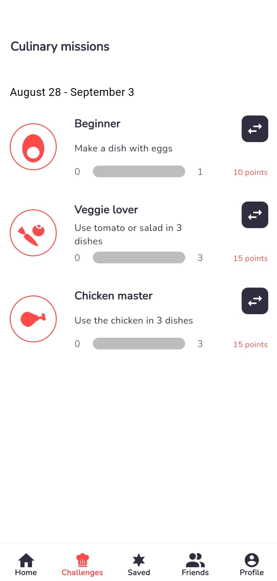
  }

}

**Output:-**

** **

** **

****