Name - Gaurang A Raorane Div - D15A
Roll no - 49 Batch - C

Experiment - 5

Aim:- To apply navigation, routing and gestures in Flutter App

Theory:-

Navigation, routing, and gestures are essential aspects of mobile app development that enable users to navigate between screens, interact with app elements, and perform various actions. In Flutter, these features are implemented using widgets and event handlers to create intuitive and engaging user experiences. Let's delve into the theory behind navigation, routing, and gestures in a Flutter app:

Navigation and Routing:

- Navigation refers to the process of moving between different screens or pages within an app.
- Routing involves defining the routes or paths that users can take to navigate to specific screens.
- Flutter provides a built-in routing mechanism using the Navigator widget and MaterialApp widget's routes property to manage app navigation.
- Routes are mapped to unique route names or paths, making it easy to navigate to a specific screen using its route name.

Navigator Widget:

- The Navigator widget manages a stack of Route objects representing the app's navigation history.
- It allows pushing new routes onto the navigation stack, popping routes off the stack, and replacing routes with new ones.
- Routes can be pushed onto the stack using Navigator.push() and popped using Navigator.pop() methods.

Routes:

- A route represents a screen or page in the app.
- Each route is associated with a unique route name and a builder function that constructs the UI for the corresponding screen.
- Routes can be declared statically using the routes property of MaterialApp or dynamically using the Navigator widget.

Gesture Detection:

- Gestures enable users to interact with app elements through touch-based actions like tapping, swiping, dragging, pinching, etc.
- Flutter provides gesture detection widgets such as GestureDetector, InkWell, InkResponse, etc., to handle user input gestures.
- These widgets detect user gestures and trigger corresponding event handlers to perform specific actions.

Gesture Recognition:

- Gesture recognition involves identifying and interpreting user gestures to determine the intended action.
- Flutter's gesture detection widgets come with built-in gesture recognizers that recognize common gestures like taps, drags, long presses, etc.
- Developers can customize gesture recognition behavior and implement complex gestures using gesture recognizer callbacks.

BottomNavBar.dart

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:unicons/unicons.dart';
import 'package:flutter/services.dart';
import 'package:osiris/Services/consts.dart';

class BottomNavBar extends StatefulWidget {
  BottomNavBar({Key? key, required this.currentIndex}) : super(key: key);
  int currentIndex = 0;

  @override
  _BottomNavBarState createState() => _BottomNavBarState();
}

class _BottomNavBarState extends State<BottomNavBar> {
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 50,
      margin: const EdgeInsets.fromLTRB(8, 8, 8, 16),
      decoration: BoxDecoration(
          color: accent_t.withOpacity(0.95),
          borderRadius: BorderRadius.circular(12)),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          IconButton(
            icon: Icon(
              UniconsLine.home_alt,
              color: widget.currentIndex == 0 ? Colors.white : inactive_accent,
            ),
            onPressed: () {
              HapticFeedback.mediumImpact();
              setState(() {
```

```
            widget.currentIndex = 0;
          });
          GoRouter.of(context).go('/main');
        },
      ),
      IconButton(
        icon: Icon(
          UniconsLine.search,
          color: widget.currentIndex == 1 ? Colors.white : inactive_accent,
        ),
        onPressed: () {
          HapticFeedback.mediumImpact();
          setState(() {
            widget.currentIndex = 1;
          });
          GoRouter.of(context).go('/search');
        },
      ),
      IconButton(
        icon: Icon(
          UniconsLine.heart,
          color: widget.currentIndex == 2 ? Colors.white : inactive_accent,
        ),
        onPressed: () {
          HapticFeedback.mediumImpact();
          setState(() {
            widget.currentIndex = 2;
          });
          GoRouter.of(context).go('/profile');
        },
      ),
    ],
  ),
 );
 }
}
```

Routes.dart

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:osiris/Screens/LoginScreen.dart';
import 'package:osiris/Screens/MainScreen.dart';
import 'package:osiris/Screens/MovieScreen.dart';
import 'package:osiris/Screens/NavScreen.dart';
```

```
import 'package:osiris/Screens/ProfileScreen.dart';
import 'package:osiris/Screens/SearchScreen.dart';
import 'package:osiris/Screens/TvShowScreen.dart';
import 'package:osiris/Screens/SignUpScreen.dart';



GoRouter router = GoRouter(initialLocation: '/', routes: [
  GoRoute(
    path: '/',
    builder: (context, state) => const NavScreen(),
  ),

  GoRoute(
    path: '/',
    builder: (context, state) => const SignUpScreen(),
    pageBuilder: defaultPageBuilder<SignUpScreen>(const SignUpScreen()),
  ),
  GoRoute(
    path: '/login',
    builder: (context, state) => const LoginScreen(),
    pageBuilder: defaultPageBuilder<LoginScreen>(const LoginScreen()),
  ),
  GoRoute(
    path: '/main',
    builder: (context, state) => const MainScreen(),
    pageBuilder: defaultPageBuilder<MainScreen>(const MainScreen()),
  ),
  GoRoute(
    path: '/search',
    builder: (context, state) => const SearchScreen(),
    pageBuilder: defaultPageBuilder<SearchScreen>(const SearchScreen()),
  ),
  GoRoute(
    path: '/profile',
    builder: (context, state) => const ProfileScreen(),
    pageBuilder: defaultPageBuilder<ProfileScreen>(const ProfileScreen()),
  ),
  GoRoute(
    path: '/movie/:id',
    builder: (context, state) => MovieScreen(state.params['id']!),
  ),
  GoRoute(
    path: '/tv/:id',
```

```
    builder: (context, state) => TVShowScreen(state.params['id']!),
  )
]);

CustomTransitionPage buildPageWithDefaultTransition<T>({
  required BuildContext context,
  required GoRouterState state,
  required Widget child,
}) {
  return CustomTransitionPage<T>(
    key: state.pageKey,
    child: child,
    transitionsBuilder: (context, animation, secondaryAnimation, child) =>
        FadeTransition(opacity: animation, child: child),
  );
}

Page<dynamic> Function(BuildContext, GoRouterState) defaultPageBuilder<T>(
        Widget child) =>
    (BuildContext context, GoRouterState state) {
      return buildPageWithDefaultTransition<T>(
        context: context,
        state: state,
        child: child,
      );
    };
```

Output:-