NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

Experiment 05: To apply navigation, routing and gestures in Flutter App

Theory:

In Flutter, navigation, routing, and gestures are core elements that contribute to an app's interactivity and flow.
The provided HomePage code effectively demonstrates all three concepts:

1. What is Navigation in Flutter
   Navigation in Flutter allows switching between different screens (pages) within the app. It manages the user flow and screen hierarchy.

2. Routing Definition
   Routing refers to the process of defining and managing the paths (routes) that lead to different screens in the app. Each route corresponds to a screen or widget.

3. Navigator Class
   Flutter provides the Navigator class to manage a stack-based history of routes. It allows pushing new routes onto the stack and popping them when going back.

4. Named Routes vs Anonymous Routes

   - Anonymous Routes: Defined inline when pushing a new route.

   - Named Routes: Predefined in the app and accessed using a unique route name.

5. Route Settings
   RouteSettings is used to pass information to a new route such as name and arguments.

6. MaterialPageRoute
   A common way to define routes in Flutter, used for material design transitions between screens.

7. Defining Routes in main.dart
   Named routes are defined in the MaterialApp widget using the routes parameter, which maps route names to widget builders.

8. Initial Route
   The first screen to be displayed is set using the initialRoute property of MaterialApp.

9. Push and Pop Operations

- Navigator.push() adds a new route to the stack.
- Navigator.pop() removes the current route and returns to the previous one.

10. Passing Data Between Screens
    Data can be passed to another screen via constructors or route arguments and retrieved when the new screen loads.

What are Gestures
Gestures are user interactions like taps, swipes, long presses, and drags that can be detected and responded to in Flutter apps.

GestureDetector Widget
GestureDetector is a core widget in Flutter used to detect and respond to gestures. It wraps any widget and provides callback functions for various gesture events.

Common Gesture Events

- onTap: Detects tap on the widget.
- onDoubleTap: Detects double-tap.
- onLongPress: Detects long press.
- onPanStart, onPanUpdate, onPanEnd: Detect drag gestures.

InkWell and InkResponse Widgets
These widgets provide ripple effect and gesture detection in material design apps. They are often used with buttons and tappable cards.

Custom Gesture Handling
Flutter allows combining gesture callbacks with logic to create custom interactions such as dragging objects or resizing UI elements.

Gesture Arena in Flutter
Flutter uses a gesture recognition system called the gesture arena to resolve conflicts when multiple gestures compete for the same event.

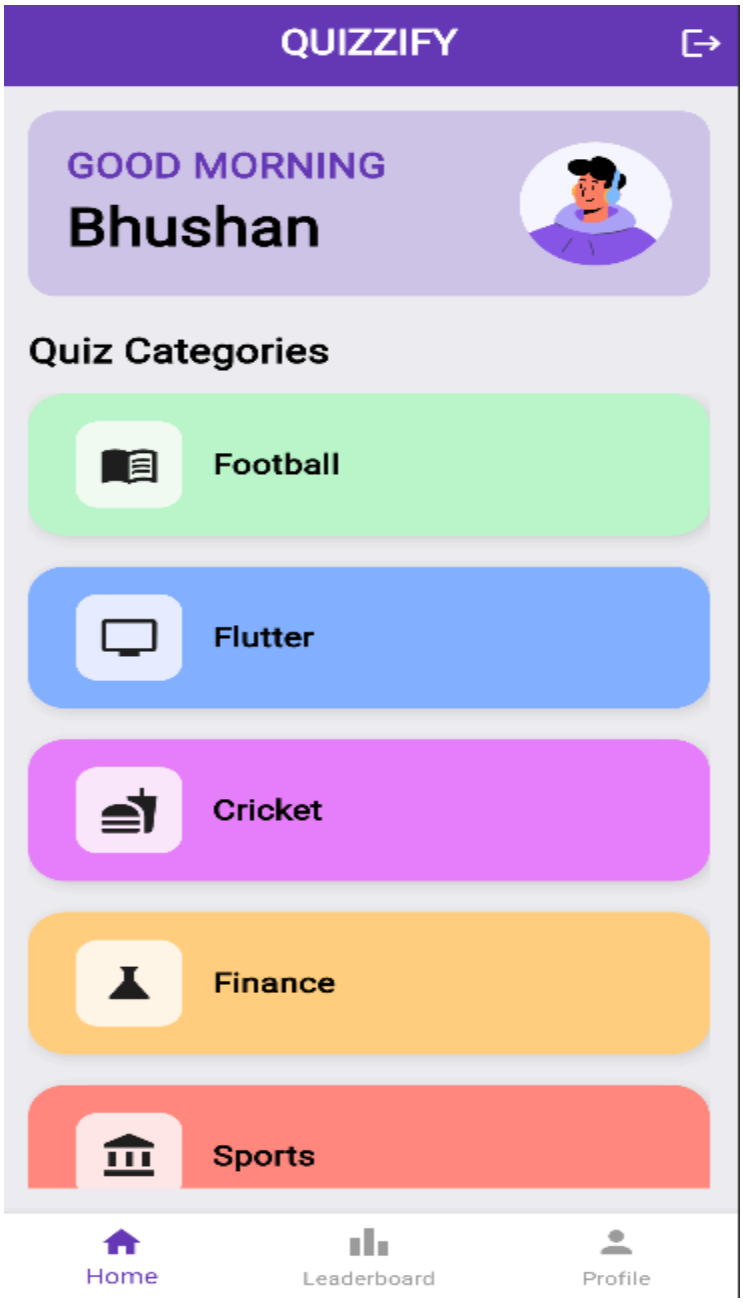
Pointer Events
Flutter also supports low-level gesture handling using pointer events like PointerDownEvent, PointerMoveEvent, and PointerUpEvent.

Interactive Widgets
Many Flutter widgets come with built-in gesture support, such as ElevatedButton, TextButton, ListTile, and Dismissible.

Scroll and Swipe Gestures
Flutter provides specialized widgets for handling scrolling (ListView, SingleChildScrollView) and swiping (Dismissible, PageView).

Importance of Gestures
Gestures improve user experience by enabling intuitive interaction patterns, supporting accessibility, and allowing natural app navigation.

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33



Conclusion :

This Flutter app successfully demonstrates routing and navigation using Navigator and MaterialPageRoute. Gesture detection is implemented using GestureDetector to enable navigation upon tapping quiz categories. The BottomNavigationBar enhances seamless switching between Home, Leaderboard, and Profile pages. User data and dynamic content from Firestore are integrated with stateful widgets and real-time updates. Overall, this experiment shows effective use of navigation, routing, and gesture handling for interactive app development.