**Experiment No:05**

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

**Theory:**

**Navigation and Routing in Flutter**

In Flutter, navigation and routing are managed by the Navigator widget. You can push a new route onto the navigator's stack to navigate to a new screen and pop a route from the stack to go back to the previous screen.

Navigation Routing:

1. Using Navigator Widget:

* Flutter's Navigator widget manages a stack of routes.
* To navigate to a new screen, you can push a new route onto the navigator's stack using Navigator.push()
* To go back to the previous screen, you can pop a route from the navigator's stack using Navigator.pop().

1. Named Routes:

* Named routes are identified by a unique string identifier.
* Define named routes in the app's main MaterialApp widget using the routes property.

1. Passing Data:

* You can pass data to a new screen when navigating by providing the data as arguments to Navigator.push().

1. Navigation using Tabs:

* DefaultTabController: The DefaultTabController widget manages the state of the TabBar and TabBarView, allowing you to switch between tabs and update the content accordingly.
* TabBar: The TabBar widget displays a horizontal row of tabs. Each tab represents a different screen or page.
* TabBarView: The TabBarView widget displays the content corresponding to the currently selected tab.

**Gesture Handling**

Gesture handling refers to the process of detecting and responding to user interactions, or gestures, on a touch-based interface, such as a smartphone or tablet. In the context of software development, especially in mobile app development, gesture handling involves recognizing various touch events, such as taps, swipes, pinches, and rotations, and translating them into meaningful actions within the application.Gesture handling involves detecting and responding to user interactions, such as taps, swipes, and pinches. In Flutter, you can use the GestureDetector widget to detect various gestures.

1. Gesture Detector:

* Wrap a widget with GestureDetector to detect gestures like tap, double tap, long press, etc.
* Specify the gesture callbacks, such as onTap, onDoubleTap, onLongPress, etc., to respond to the detected gestures.

1. Draggable and DragTarget:

* Use Draggable to make a widget draggable.
* Specify a child widget that can be dragged and provide a feedback widget that appears when the widget is being dragged.
* Use DragTarget to specify a target area for dropping a draggable widget.
* Implement the onWillAccept and onAccept callbacks to handle when a draggable widget is dragged over the DragTarget and dropped.

**Code:**

**main.dart:**

import 'package:flutter/material.dart';

import 'firstpage.dart';

import 'secondpage.dart';

import 'thirdpage.dart';

import 'loginpart.dart';

import 'ocr\_page.dart'; // Import the new OCR page file

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Home(),

theme: ThemeData(

visualDensity: VisualDensity.adaptivePlatformDensity,

primaryColor: Colors.indigo[900],

),

);

}

}

class Home extends StatelessWidget {

@override

Widget build(BuildContext context) {

return DefaultTabController(

length: 5, // Change the length to 5 for the new tabs

child: Scaffold(

appBar: AppBar(

backgroundColor: Colors.indigo[900],

bottom: TabBar(

tabs: [

Tab(

text: "Login", // Add a new tab for the login page

),

Tab(

text: "Original\nImage",

),

Tab(

text: "Scan\nImage",

),

Tab(

text: "Draw\nImage",

),

Tab(

text: "OCR", // Add a new tab for OCR

),

],

),

title: Text(

'Make PDF',

),

),

body: TabBarView(

children: [

LoginPage(), // Add the login page as a new tab

Firstpage(),

Secondpage(),

Thirdpage(),

OCRPage(), // Add the OCR page as a new tab

],

),

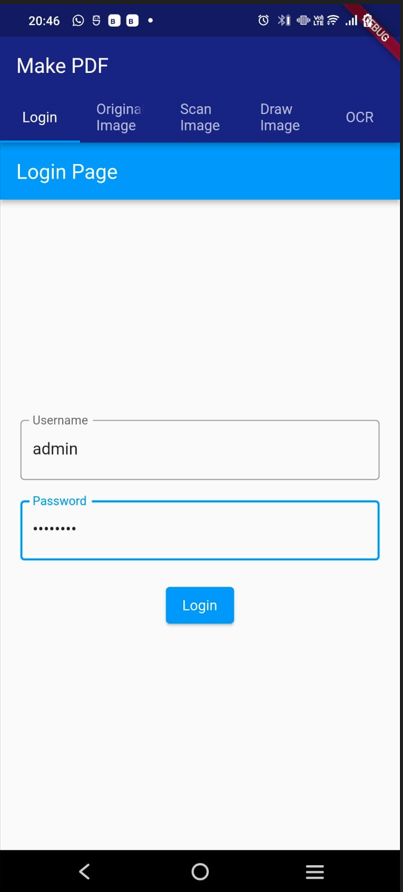
),

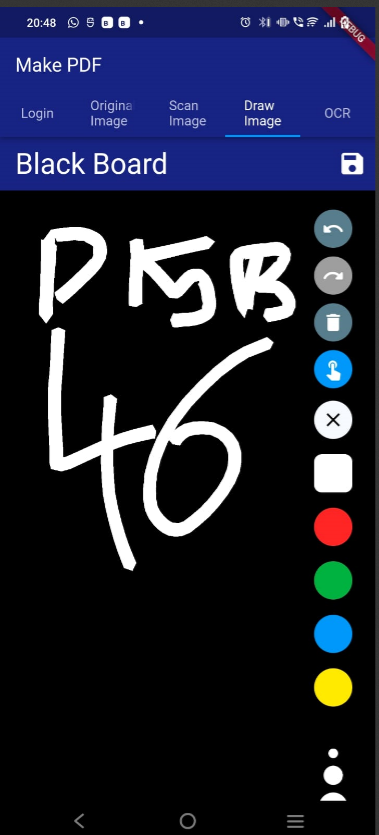
);

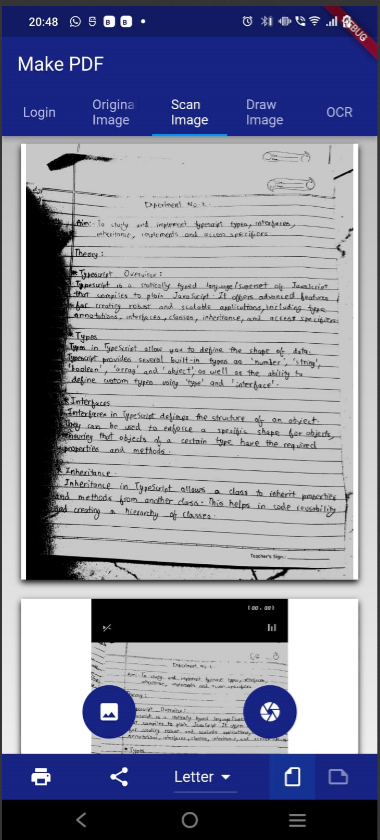
}

}

**Output:**







**Conclusion:**

We understood the concepts of navigation , routing and gestures in Flutter. We implemented navigation and routing for the above shown pages. We implemented gestures in a basic Flutter application.