Experiment No.5

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

- **Theory:**

Navigation in Flutter:

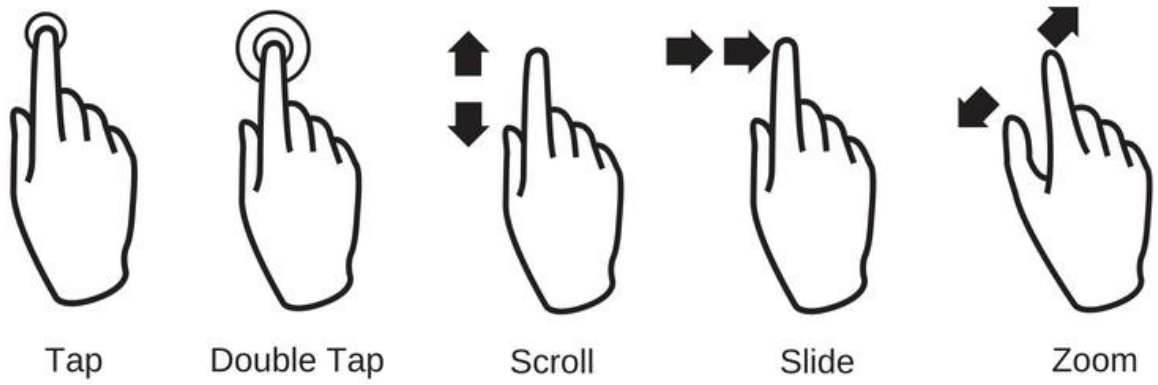
Navigation in Flutter refers to the process of moving between different screens or pages within a mobile application. It involves transitioning from one view to another seamlessly. In Flutter, the Navigator class facilitates navigation by managing a stack of routes. Developers use methods like Navigator.push() to navigate forward, adding a new screen to the stack, and Navigator.pop() to move backward, removing the current screen. Additionally, named routes provide a declarative way to define and access specific screens. Overall, navigation is crucial for creating a structured and user-friendly app experience, allowing users to explore different sections of the application effortlessly.

Routing in Flutter:

Routing in Flutter involves defining the structure of an application, specifying how different screens or views are organized and accessed. The MaterialApp widget serves as the root of the app, providing a scaffold for navigation and routing. Developers can define routes using named routes or the default '/' route. The MaterialPageRoute class is used to specify the transition and content for each route. Effective routing enhances code organization and maintenance, contributing to a more modular and scalable Flutter application.

Gestures in Flutter:

Gestures in Flutter enable user interactions through touch-based actions, enhancing the interactivity of an application. The GestureDetector widget is fundamental for handling gestures like taps, long presses, and drags. By wrapping widgets with GestureDetector or using specialized widgets like InkWell, developers can respond to various touch events, providing a more intuitive user experience. Gestures play a vital role in mobile app development, allowing users to navigate, select, and interact with content seamlessly, contributing to a fluid and engaging UI. Understanding and implementing gestures effectively is crucial for creating responsive and user-friendly Flutter applications.



- **Code:-**

```
GestureDetector(
  onTap: () => Navigator.pushReplacement(
    context,
    MaterialPageRoute(
      builder: (context) {
        return HomePage();
      },
    ), // MaterialPageRoute
  ),
  child: Container(
    padding: const EdgeInsets.all(24),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(16),
      color: Colors.black,
    ), // BoxDecoration
    child: const Text(
      "Get Started",
      style: TextStyle(
        color: Colors.white,
        // fontWeight: FontWeight.bold,
        fontSize: 16,
      ), // TextStyle
```

```
return GestureDetector(
  onTap: () {
    print(categoriesName[index]);
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) =>
                BookList(name: categoriesName[index]))); // MaterialPageRoute
  },
  child: Stack(
    alignment: Alignment.bottomLeft,
    children: [
      Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(12),
          image: DecorationImage(
              image: NetworkImage(categoriesImage[index]),
              fit: BoxFit.cover), // DecorationImage
        ), // BoxDecoration
```

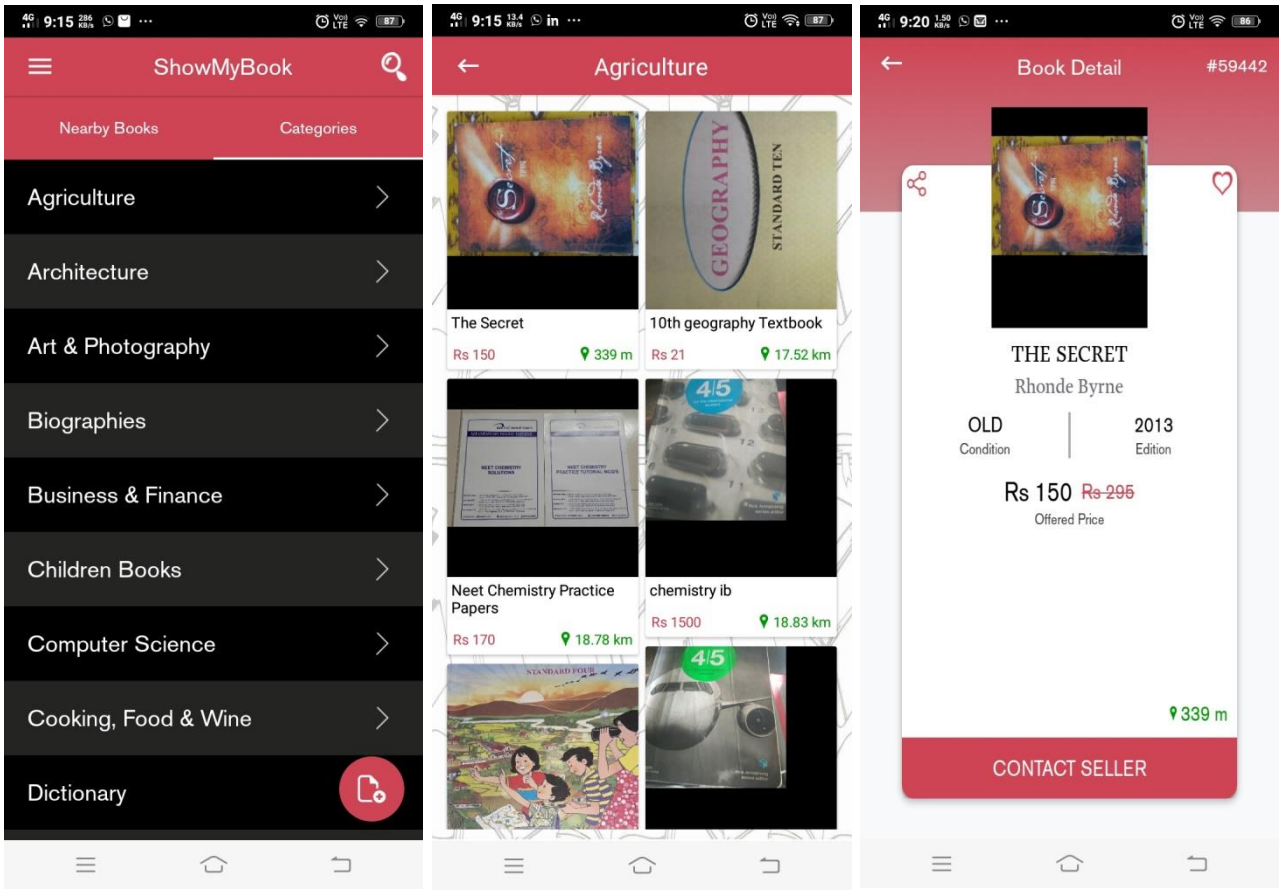
```
class Categories extends StatelessWidget {
  Widget build(BuildContext context) {
                        begin: Alignment.topCenter,
                        end: Alignment.bottomCenter,
                      )), // LinearGradient // BoxDecoration
                ), // Container
                Padding(
                  padding: const EdgeInsets.all(10.0),
                  child: Text(
                    categoriesName[index],
                    style: Theme.of(context)
                        .textTheme
                        .headline4
                        ?.copyWith(color: Colors.white),
                  ), // Text
                ), // Padding
              ],
            ), // Stack
          ); // GestureDetector
```

GestureDetector: The GestureDetector widget is used to detect user interactions, specifically the onTap event.

Navigator.push: The Navigator.push method is employed to navigate to a new screen when a category and book is tapped. It takes a MaterialPageRoute as an argument, and the builder function creates the destination screen with the corresponding category name ansd book name.

BookList Screen: The BookList screen is instantiated with the selected category's name as a parameter

- **Output:-**



- **Conclusion:**

Adding features like moving between different screens, setting up clear paths, and using taps and gestures in my book-selling app made it more user-friendly. This made it easy for users to explore and interact with the app. Working on these aspects in Flutter improved my understanding of creating a smooth and enjoyable experience for users as they navigate through different sections of the app.