Experiment - 5

Name: Shivpratik Hande	Div-Roll no: **D15C-14**
DOP:	DOS:
Sign:	Grade:

AIM : To apply navigation, routing, and gestures in Flutter App

Theory :

Flutter is a powerful framework for building cross-platform mobile applications, and it provides efficient mechanisms for:

Navigation and Routing :

Navigation allows moving between different screens (also called routes or pages) in a Flutter app. Flutter provides multiple methods to implement navigation:

a) Basic Navigation (Navigator.push and Navigator.pop)

- Navigator.push(context, MaterialPageRoute(builder: (_) => SecondPage())); Pushes a new route onto the stack.
- Navigator.pop(context); Pops the top-most route from the stack and returns to the previous screen

b) Named Routing

- Define routes in MaterialApp's routes property:

c) Navigation Stack

Flutter uses a stack-based navigation model where each new screen is "pushed" onto a stack and can be "popped" to return to the previous screen.

2. Gestures in Flutter

Gestures are used to detect user interaction like taps, swipes, drags, etc.
Flutter uses the GestureDetector widget to handle gestures:

Gesture Type	Widget/Callback Used
Tap	onTap
Double Tap	onDoubleTap
Long Press	onLongPress
Vertical Drag on	VerticalDragUpdate
Horizontal Drag on	HorizontalDragUpdate

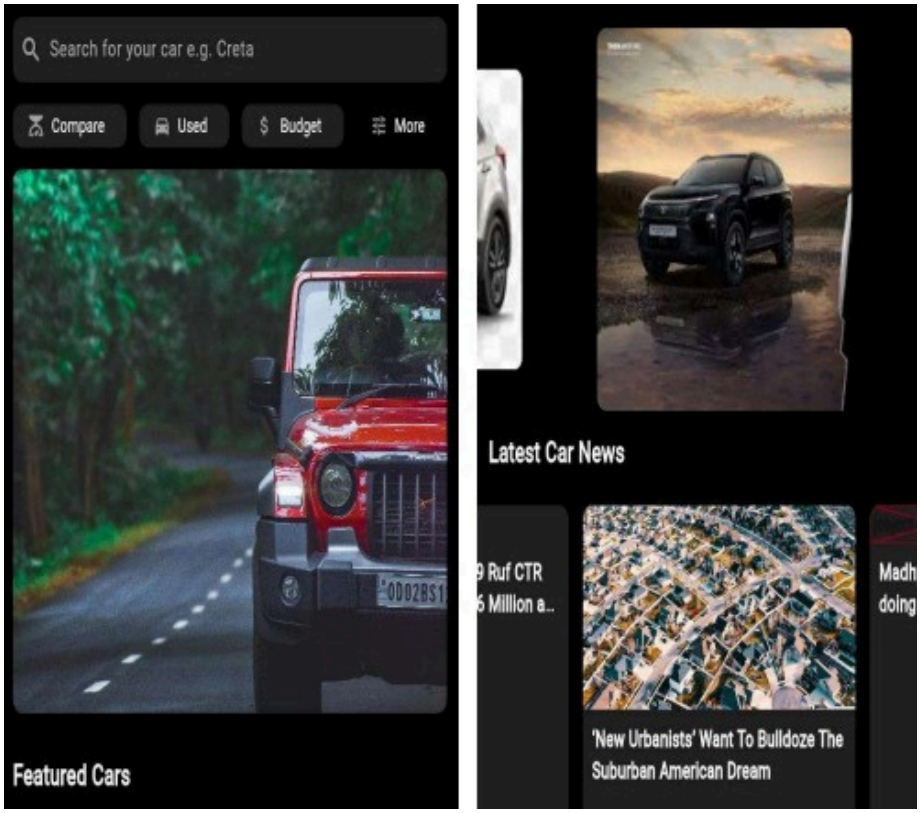
Experiment - 5

GestureDetector is a powerful tool for creating interactive UIs and responding to user inputs like swipe-to-dismiss, tap-to-select, or drag-to-move elements.

Example usage of navigation and routing in Flutter

```
MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => HomePage(),
    '/second': (context) => SecondPage(),
  },
);
// Navigate using named route
Navigator.pushNamed(context, '/second');
```

Pages and Navigation inside of my Disaster Preparedness Application



Conclusion : These are the various ways in which navigation, gestures and routing were implemented inside the application.