Name:Mahek Taneja
Roll no:58

Exp 5 To apply navigation, routing and gestures in Flutter App

Aim:
To apply navigation, routing, and gesture detection in a Flutter application to enhance user interaction and app flow.

Theory:
Flutter provides powerful tools for creating seamless and interactive user interfaces. Navigation, routing, and gestures are essential for managing screen transitions and handling user interactions in a structured and responsive way.
1. Navigation and Routing:
Navigation allows users to move between different screens (widgets) in the app. Flutter offers two primary ways to implement navigation:

- Named Routes: Defined in the MaterialApp's routes table for organized screen management.

dart
CopyEdit

```dart
Navigator.pushNamed(context, '/second');
```

- Anonymous Routes (Direct Navigation): Used for one-time navigation without predefined names.

dart
CopyEdit

```dart
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => SecondScreen()),
);
```

Routing helps manage complex navigation flows in apps with multiple screens like an E-commerce or Social Media app.
2. Gesture Detection:
Flutter provides the GestureDetector widget, which listens for user gestures such as taps, swipes, long presses, and drags.
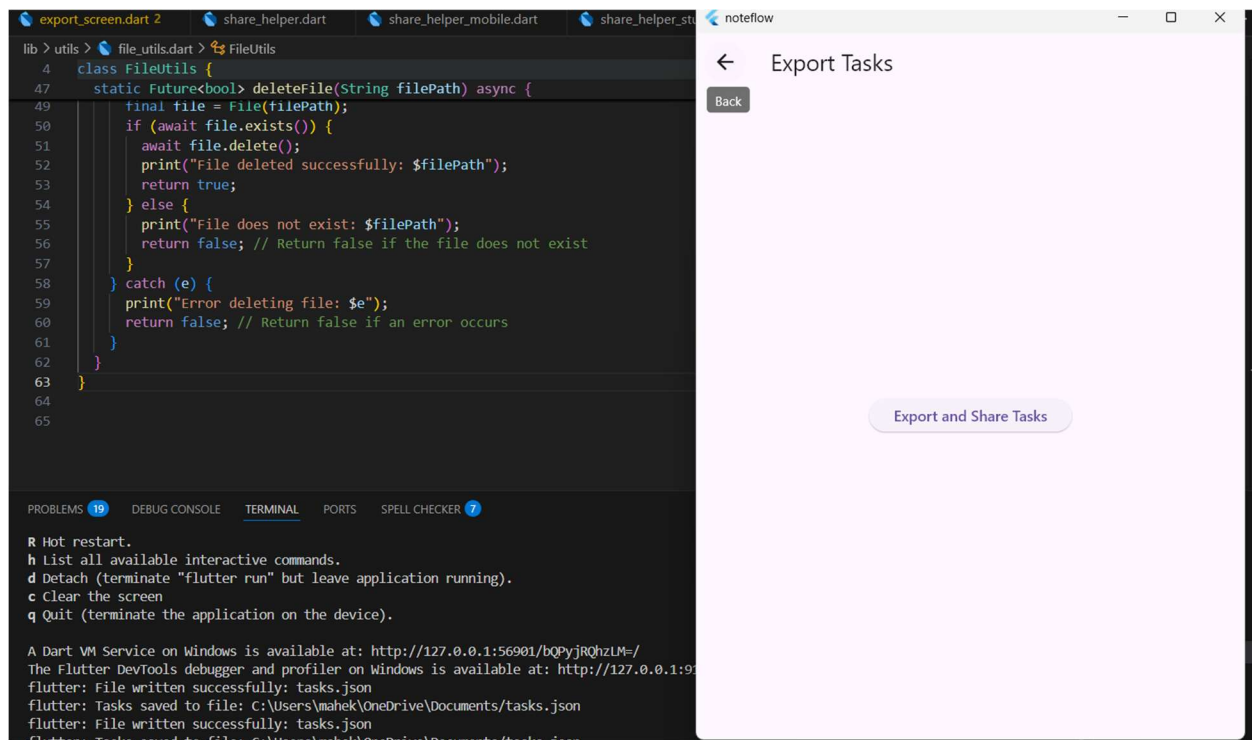Example:
dart
CopyEdit

```dart
GestureDetector(
  onTap: () {
    print("Tapped!");
  },
  child: Container(
    color: Colors.blue,
```
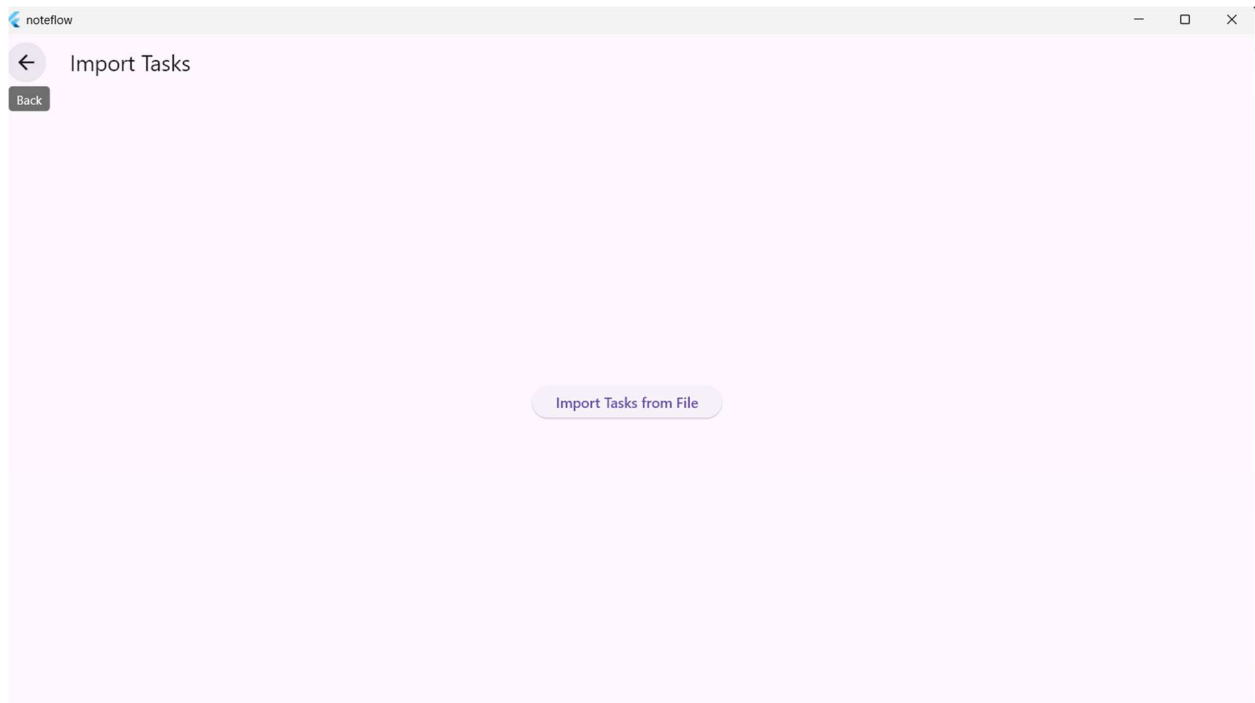
```
    width: 100,
    height: 100,
  ),
);
```
Gestures enhance interactivity, allowing users to perform actions by touching the screen in specific ways—like swiping to delete or tapping to navigate.

Conclusion:

By applying navigation, routing, and gestures in a Flutter app, we can create a more interactive, intuitive, and user-friendly experience. Navigation enables smooth screen transitions, routing structures the app's flow, and gesture detection makes the interface responsive to touch-based user actions. These concepts are fundamental to building modern mobile apps that are both functional and engaging.