

## Experiment: 05

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

### Theory:

Flutter provides a complete system for navigating between screens and handling deep links. Small applications without complex deep linking can use Navigator, while apps with specific deep linking and navigation requirements should also use the Router to correctly handle deep links on Android and iOS, and to stay in sync with the address bar when the app is running on the web.

### Using the Navigator

The Navigator widget displays screens as a stack using the correct transition animations for the target platform. To navigate to a new screen, access the Navigator through the route's BuildContext and call imperative methods such as push() or pop():

```
onPressed: () {  
  Navigator.of(context).push(  
    MaterialPageRoute(  
      builder: (context) => const SongScreen(song: song),  
    ),  
  );  
},  
child: Text(song.name),
```

Because Navigator keeps a stack of Route objects (representing the history stack), The push() method also takes a Route object. The MaterialPageRoute object is a subclass of Route that specifies the transition animations for Material Design. For more examples of how to use the Navigator, follow the navigation recipes from the Flutter Cookbook or visit the Navigator API documentation.

### Using named routes

Applications with simple navigation and deep linking requirements can use the Navigator for navigation and the MaterialApp.routes parameter for deep links:

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    routes: {
      '/': (context) => HomeScreen(),
      '/details': (context) => DetailScreen(),
    },
  );
}
```

Routes specified here are called named routes. For a complete example, follow the [Navigate with named routes](#) recipe from the Flutter Cookbook.

### Limitations

Although named routes can handle deep links, the behavior is always the same and can't be customized. When a new deep link is received by the platform, Flutter pushes a new Route onto the Navigator regardless where the user currently is.

Flutter also doesn't support the browser forward button for applications using named routes. For these reasons, we don't recommend using named routes in most applications.

### Using the Router

Flutter applications with advanced navigation and routing requirements (such as a web app that uses direct links to each screen, or an app with multiple Navigator widgets) should use a routing package such as `go_router` that can parse the route path and configure the Navigator whenever the app receives a new deep link.

To use the Router, switch to the router constructor on `MaterialApp` or `CupertinoApp` and provide it with a Router configuration. Routing packages, such as `go_router`, typically provide a configuration for you. For example:

```
MaterialApp.router(
  routerConfig: GoRouter(
    // ...
  )
);
```

Code:

```
class MyAppBar extends StatelessWidget implements PreferredSizeWidget {
  @override
  Size get preferredSize => Size.fromHeight(kToolbarHeight);

  @override
  Widget build(BuildContext context) {
    return AppBar(
      title: Row(
        children: [
          Padding(
            padding: EdgeInsets.all(0.0),
            child: Image.asset(
              'assets/images/logo.png', // Replace with the actual path to
your image
              width: 50.0, // Adjust the width as needed
            ),
          ),
          Text(
            'AGRI',
            style: TextStyle(
              color: Colors.grey[900],
              fontSize: 20,
              fontWeight: FontWeight.w900,
            ),
          ),
          Text(
            'FARM',
            style: TextStyle(
              color: Color.fromARGB(255, 24, 156, 19),
              fontSize: 20,
              fontWeight: FontWeight.w900,
            ),
          ), // Replace with your app's logo icon
        ], // Replace with your app's name
      ),
    );
  }
}
```

```
        actions: [
          IconButton(
            icon: Icon(Icons.menu), // Replace with your toggle icon
            onPressed: () {
              // Open your toggle switch or drawer here
              _showNavigationMenu(context);
            },
          ),
        ],
      );
    }
  }

Widget _buildNavigationMenuItem(
  BuildContext context, String title, VoidCallback onPressed) {
  return ListTile(
    title: Text(title),
    onTap: onPressed,
  );
}

void _showNavigationMenu(BuildContext context) {
  final FirebaseAuth _auth = FirebaseAuth.instance;

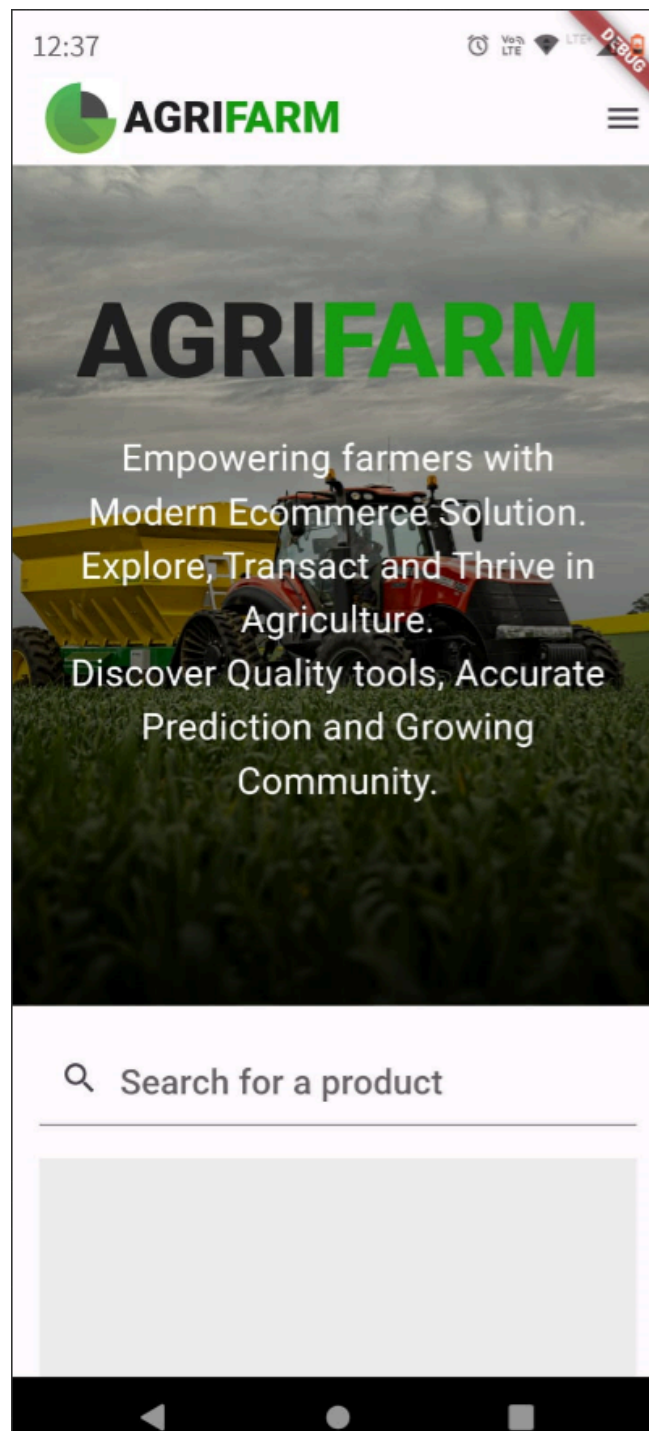
  User? _user;

  showModalBottomSheet(
    context: context,
    builder: (BuildContext context) {
      return Column(
        mainAxisAlignment: MainAxisAlignment.min,
        children: [
          _buildNavigationMenuItem(context, 'Profile', () {
            Navigator.pop(context);
            // Navigate to the Profile page
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => ProfilePage()),
            );
          }),
        ],
      );
    },
  );
}
```

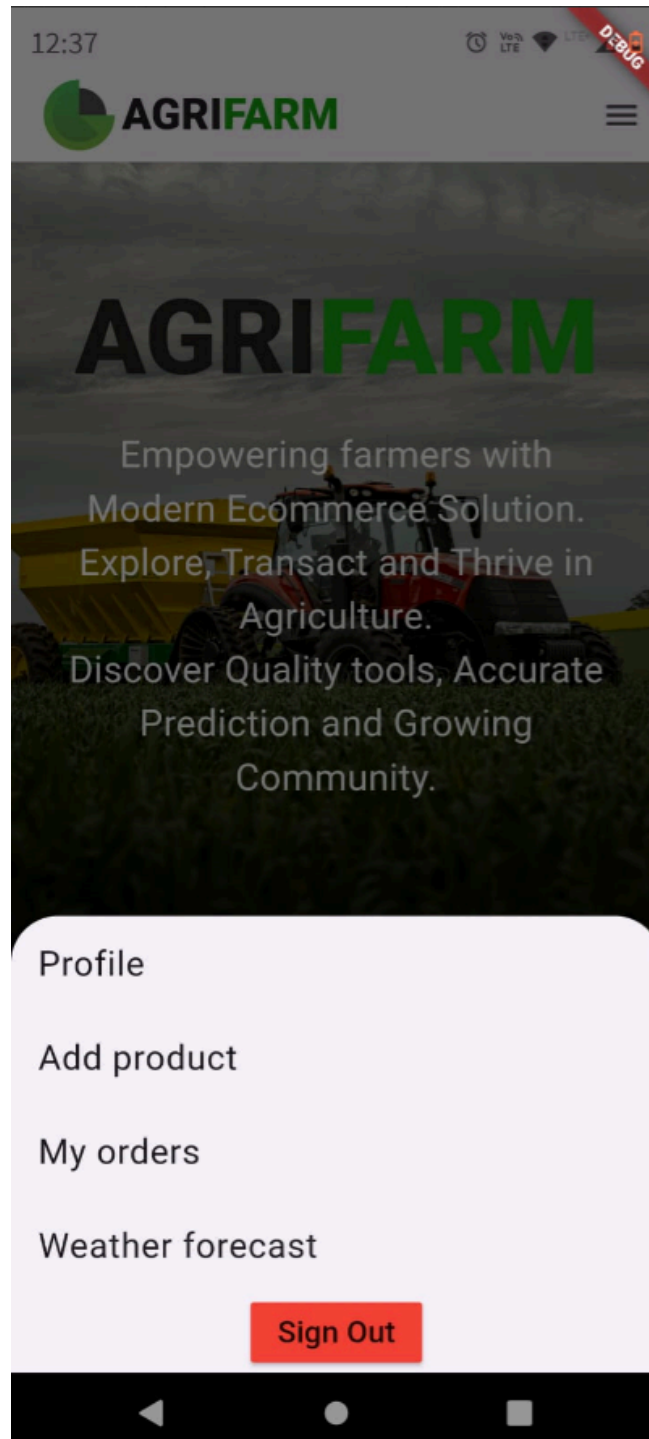
```
    _buildNavigationMenuItem(context, 'Add product', () {
      Navigator.pop(context);
      // Navigate to the Add product page
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => AddProductPage()),
      );
    }),
    _buildNavigationMenuItem(context, 'My orders', () {
      Navigator.pop(context);
      // Navigate to the My orders page
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => MyOrdersPage()),
      );
    }),
    _buildNavigationMenuItem(context, 'Weather forecast', () {
      Navigator.pop(context);
      // Navigate to the Weather forecast page
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) =>
WeatherForecastPage()),
      );
    }),
    MaterialButton(
      color: Colors.red,
      child: const Text("Sign Out"),
      onPressed: _auth.signOut,
    ),
  ],
);
},
);
}
```

**Output:**

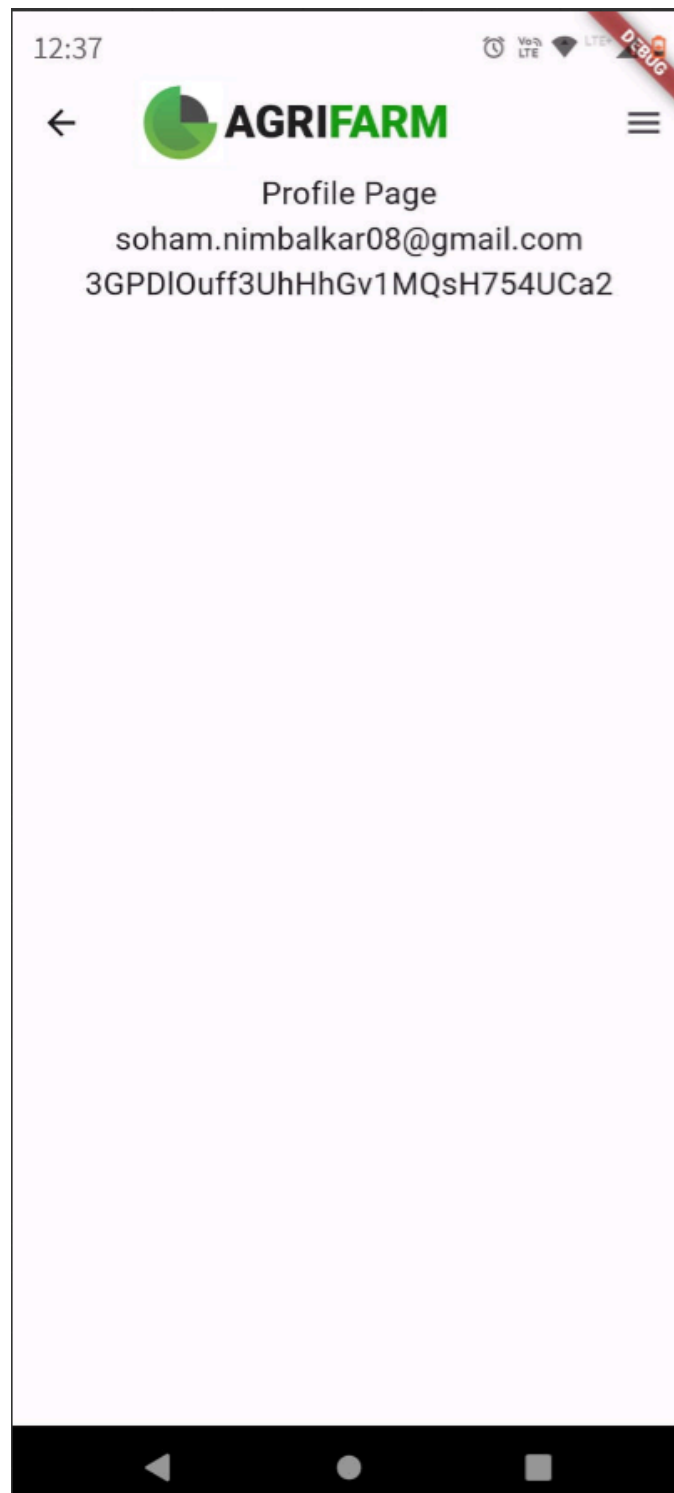
Landing page:



Navigation menu:

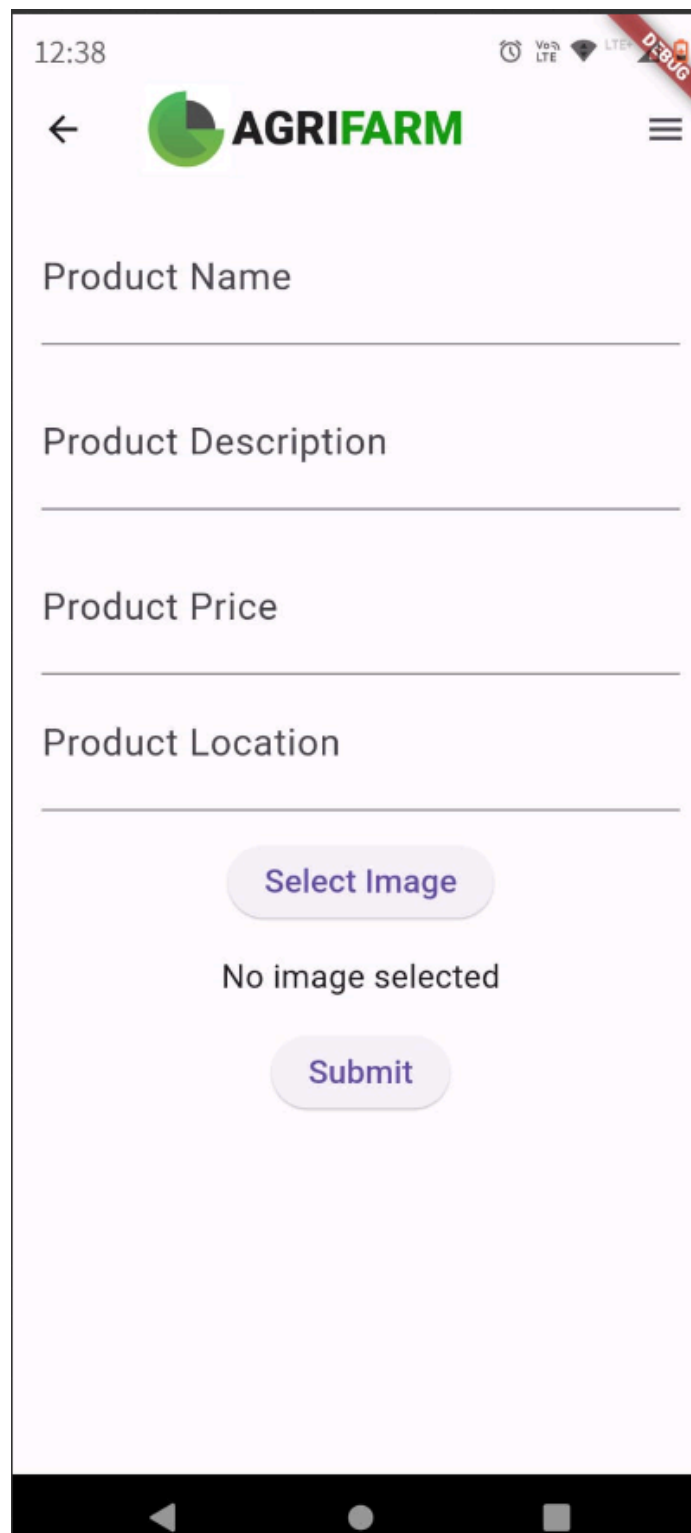


Profile page:





Add product:



The screenshot shows the 'Add product' screen of the AGRIFARM app. At the top, the status bar displays the time 12:38, VoLTE, LTE, and a battery icon. The app's header includes a back arrow, the AGRIFARM logo, and a menu icon. The form contains four text input fields: 'Product Name', 'Product Description', 'Product Price', and 'Product Location'. Below these fields is a 'Select Image' button, followed by the text 'No image selected', and a 'Submit' button. A red 'Debug' banner is visible in the top right corner of the app interface.

12:38

VoLTE LTE

Debug

← AGRIFARM ≡

Product Name

Product Description

Product Price

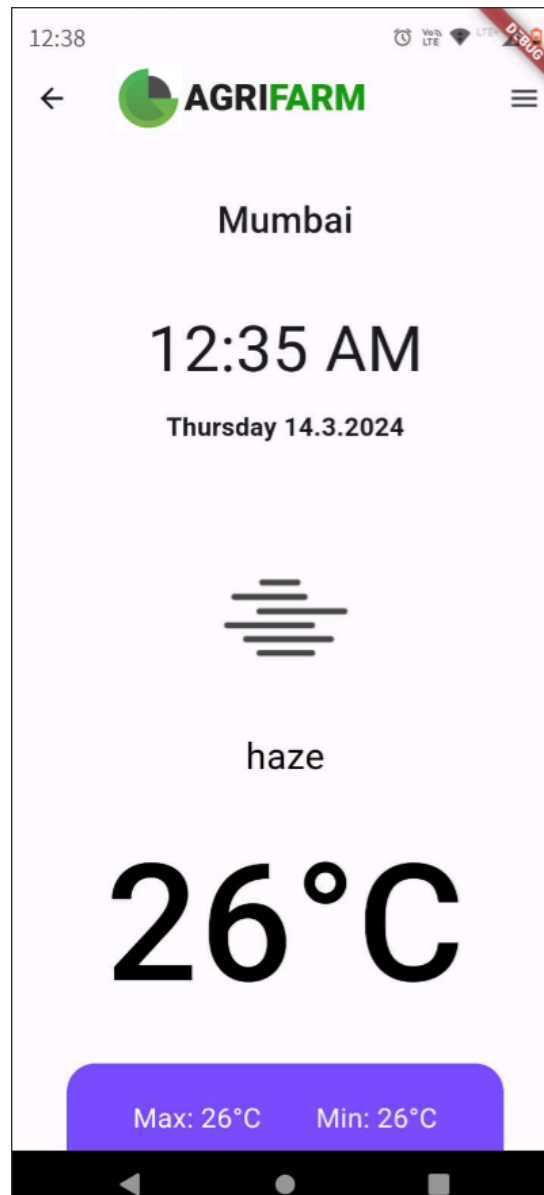
Product Location

Select Image

No image selected

Submit

Weather page:



**Conclusion:** We have learned how to navigate to a new screen by creating a new route and manage it by using the Navigator. The Navigator maintains the stack-based history of routes. If there is a need to navigate to the same screen in many parts of the app, this approach is not beneficial because it results in code duplication. The solution to this problem can be removed by defining the named routes and can use the named routes for navigation.