# TIERS LIMITED SUMMER INTERNSHIP 2024

# MOBILE APP DEVELOPMENT

## Navigation and Routing

CLASS # 6

## Overview

In this class, we will explore navigation and routing in Flutter, which are essential for building multi-screen applications. We will cover the basics of navigation, different navigation methods, and how to implement routing in a Flutter app.

## Objectives

- Understand the concept of navigation and routing in Flutter.
- Learn how to use Navigator for basic navigation.
- Explore different methods of navigation including named routes.
- Implement navigation using the Navigator widget.
- Learn to pass data between screens.

## Navigation and Routing in Flutter

**1. Introduction to Navigation**

- **Navigation:** The process of moving from one screen to another in an application.
- **Routing:** The mechanism that determines which screen to display when a particular route is requested.

**2. Basic Navigation**

- **Navigator Class:** A widget that manages a stack of route objects.
- **push Method:** Adds a route to the stack of routes managed by the Navigator.
- **pop Method:** Removes the current route from the stack of routes managed by the Navigator.

```
class FirstScreen extends StatelessWidget {
  const FirstScreen({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Screen'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const SecondScreen()),
            );
          },
        ),
      ),
    );
  }
}
```

```
class SecondScreen extends StatelessWidget {
  const SecondScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go Back'),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
  }
}
```

### 3. Named Routes

- **Named Routes:** A way to manage routes by assigning them names.
- **Route Settings:** Can be configured in the MaterialApp widget.

**Example: Named Routes**

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => const FirstScreen(),
        '/second': (context) => const SecondScreen(),
      },
    );
  }
}
```

### 4. Passing Data Between Screens

- **Arguments:** Data can be passed between screens using the constructor of the destination screen.

**Example: Passing Data**

```
class FirstScreen extends StatelessWidget {
  const FirstScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Screen'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const SecondScreen(
                  message: 'Hello from First Screen!',
                ),
              ),
            );
          },
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  final String message;

  const SecondScreen({Key? key, required this.message}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: Text(message),
      ),
    );
  }
}
```

## 5. Returning Data from a Screen

- **Navigator.pop:** Can return data to the previous screen.

**Example: Returning Data**

```
class FirstScreen extends StatelessWidget {
  const FirstScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Screen'),
          onPressed: () async {
            final result = await Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const SecondScreen()),
            );

            ScaffoldMessenger.of(context)
                .showSnackBar(SnackBar(content: Text('Result: $result')));
          },
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  const SecondScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Return Data'),
          onPressed: () {
            Navigator.pop(context, 'Hello from Second Screen!');
          },
        ),
      ),
    );
  }
}
```

## Exercises:

1. **Exercise 1:** Insert Navigation in your mini project.
2. **Exercise 1:** Pass Data in your screens.