

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

### Theory:

Navigation, routing, and gestures are fundamental aspects of building interactive mobile applications. They define how users move between different screens (navigation), how specific URLs or paths map to particular screens or functionalities (routing), and how users interact with on-screen elements (gestures).

- **Navigation:** This involves moving between different screens or parts of an application. It typically involves using widgets like `Navigator` to push and pop routes (screens) onto a stack.
- **Routing:** This is the process of mapping a specific URL or path to a particular screen or functionality within an application. Named routes provide a way to navigate to different parts of the app using a string identifier.
- **Gestures:** These are user interactions like tapping, swiping, dragging, and pinching that can trigger specific actions in an application. Flutter provides `GestureDetector` and other widgets to recognize and respond to these gestures.

### Implemented in our Code

In our Local Resource Sharing Flutter app, we used navigation, routing and gestures in following ways.

- **Navigation:** We implemented navigation to allow users to move between the login screen, sign-up screen, home screen, item detail screen, and add item screen using the `Navigator` widget. When a user successfully logs in or signs up, they are navigated to the home screen.
- **Routing:** We defined named routes in the `MaterialApp` widget to map route names to specific screens. This makes it easy to navigate to a specific screen by its name. The routes allowed us to move between Login, signup and the home screen.
- **Gestures:** We used the `GestureDetector` widget to make the resource cards on the home screen tappable. When a user taps on a resource card, they are navigated to the item detail screen, where they can see more information about the resource.

### Code

```
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

```
void main() {
  runApp(LocalResourceApp());
}
```

```
class LocalResourceApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Local Resource Sharing',
      theme: ThemeData(
        primaryColor: Colors.teal,
        scaffoldBackgroundColor: Colors.grey[200],
        textTheme: TextTheme(
```

```
bodyLarge: TextStyle(fontFamily: 'Roboto', color: Colors.black),
bodyMedium: TextStyle(fontFamily: 'Roboto', color: Colors.black87),
),

// New Color Scheme based on the theme
colorScheme: ColorScheme.fromSwatch().copyWith(
  primary: Colors.teal, // Primary color (teal)
  secondary: Colors.amber, // Secondary color (amber)
  background: Colors.grey[200], // Light gray background
  surface: Colors.white, // White cards
  onBackground: Colors.black87, // Dark gray text
  onSurface: Colors.black87, // Dark gray text
  onPrimary: Colors.white,
  onSecondary: Colors.white,
),
appBarTheme: AppBarTheme(
  backgroundColor: Colors.teal,
  foregroundColor: Colors.white,
),
elevatedButtonTheme: ElevatedButtonThemeData(
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.teal,
    foregroundColor: Colors.white,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(10),
    ),
  ),
),
inputDecorationTheme: InputDecorationTheme(
  border: OutlineInputBorder(
    borderRadius: BorderRadius.circular(10),
  ),
  focusedBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Colors.teal, width: 2),
    borderRadius: BorderRadius.circular(10),
  ),
  labelStyle: TextStyle(color: Colors.black87),
),
home: AuthChecker(),
routes: {
  '/home': (context) => HomeScreen(),
  '/login': (context) => LoginScreen(),
  '/signup': (context) => SignupScreen(),
  // '/item_detail': (context) => ItemDetailScreen(), // Route for Item Detail Screen
  '/add_item': (context) => AddItemScreen(), // Route for Add Item Screen
}
```

```
    },  
  );  
}  
}  
  
// Home Screen  
class HomeScreen extends StatefulWidget {  
  @override  
  _HomeScreenState createState() => _HomeScreenState();  
}  
  
class _HomeScreenState extends State<HomeScreen> {  
  String userName = "";  
  String userEmail = "";  
  List<ResourceItem> items = [ // Changed to List<ResourceItem>  
    ResourceItem(  
      name: 'Drill Machine',  
      category: 'Home Appliances',  
      image: 'assets/images/drill_machine.png',  
      icon: Icons.construction,  
      owner: 'Rahul Sharma',  
      description: 'Powerful drill for various tasks',  
    ),  
    ResourceItem(  
      name: 'Lawn Mower',  
      category: 'Gardening',  
      image: 'assets/images/lawn_mower.png',  
      icon: Icons.grass,  
      owner: 'Priya Patel',  
      description: 'Efficient lawn mower for a perfect lawn',  
    ),  
    ResourceItem(  
      name: 'Projector',  
      category: 'Electronics',  
      image: 'assets/images/projector.png',  
      icon: Icons.tv,  
      owner: 'Amit Verma',  
      description: 'High-resolution projector for home theater',  
    ),  
    ResourceItem(  
      name: 'Electric Kettle',  
      category: 'Kitchen Appliances',  
      image: 'assets/images/default.png',  
      icon: null,  
      owner: 'Sneha Reddy',  
      description: 'Fast boiling electric kettle for daily use',  
    ),  
  ],  
}
```

```
),  
];
```

```
@override  
void initState() {  
  super.initState();  
  loadUserData();  
}
```

```
Future<void> loadUserData() async {  
  final prefs = await SharedPreferences.getInstance();  
  setState(() {  
    userName = prefs.getString('userName') ?? 'User';  
    userEmail = prefs.getString('userEmail') ?? 'user@example.com';  
  });  
}
```

```
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(  
      title: Text('Local Resource Sharing'),  
      actions: [  
        IconButton(  
          icon: Icon(Icons.add),  
          onPressed: () {  
            Navigator.pushNamed(context, '/add_item'); // Navigate to Add Item Screen  
          },  
        ),  
        IconButton(  
          icon: Icon(Icons.logout),  
          onPressed: () async {  
            final prefs = await SharedPreferences.getInstance();  
            await prefs.setBool('isLoggedIn', false);  
            Navigator.pushReplacementNamed(context, '/login');  
          },  
        ),  
      ],  
    ),  
    body: Column(  
      crossAxisAlignment: CrossAxisAlignment.start,  
      children: [  
        Padding(  
          padding: const EdgeInsets.all(16.0),  
          child: Column(  

```

```
crossAxisAlignment: CrossAxisAlignment.start,
children: [
  Text('Welcome, $userName!',
    style:
      TextStyle(fontSize: 24, fontWeight: FontWeight.bold)),
  SizedBox(height: 8),
  Text('Email: $userEmail',
    style: TextStyle(fontSize: 16, color: Colors.grey[600])),
],
),
),
Expanded(
  child: ListView.builder(
    itemCount: items.length,
    itemBuilder: (context, index) {
      return ResourceCard(item: items[index]);
    },
  ),
),
],
),
);
}
}

// Add Item Screen
class AddItemScreen extends StatefulWidget {
  @override
  _AddItemScreenState createState() => _AddItemScreenState();
}

class _AddItemScreenState extends State<AddItemScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _descriptionController = TextEditingController();
  final TextEditingController _categoryController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('List an Item'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Form(
```

```
key: _formKey,  
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.start,  
  children: [  
    TextFormField(  
      controller: _nameController,  
      decoration: InputDecoration(  
        labelText: 'Item Name',  
        border: OutlineInputBorder(),  
      ),  
      validator: (value) {  
        if (value == null || value.isEmpty) {  
          return 'Please enter item name';  
        }  
        return null;  
      },  
    ),  
    SizedBox(height: 20),  
    TextFormField(  
      controller: _descriptionController,  
      decoration: InputDecoration(  
        labelText: 'Description',  
        border: OutlineInputBorder(),  
      ),  
      validator: (value) {  
        if (value == null || value.isEmpty) {  
          return 'Please enter item description';  
        }  
        return null;  
      },  
    ),  
    SizedBox(height: 20),  
    TextFormField(  
      controller: _categoryController,  
      decoration: InputDecoration(  
        labelText: 'Category',  
        border: OutlineInputBorder(),  
      ),  
      validator: (value) {  
        if (value == null || value.isEmpty) {  
          return 'Please enter item category';  
        }  
        return null;  
      },  
    ),  
    SizedBox(height: 20),
```

```
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      // Implement adding item logic here
      String itemName = _nameController.text;
      String itemDescription = _descriptionController.text;
      String itemCategory = _categoryController.text;


      // For now, just show a snackbar
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(
            'Adding $itemName to category $itemCategory',
          ),
        ),
      );

      // You can also navigate back to the HomeScreen or update the item list
      Navigator.pop(context);
    }
  },
  child: Text('Add Item'),
),
],
),
),
),
);
}
}

class ResourceItem {
  final String name;
  final String category;
  final String image;
  final String owner; // Added owner
  final IconData? icon;
  final String description;

  ResourceItem({
    required this.name,
    required this.category,
    required this.image,
    required this.icon,
    required this.owner,
    required this.description,
  });
}
```

localhost:51352/#/login



## Local Resource Sharing


Login

Don't have an account? [Sign Up](#)


localhost:51352

Local Resource Sharing


Welcome, Ganesh!  
Email: ganesh@gmail.com



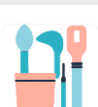
**Drill Machine**  
Powerful drill for various tasks  
Owner: Rahul Sharma      Category: Home Appliances



**Lawn Mower**  
Efficient lawn mower for a perfect lawn  
Owner: Priya Patel      Category: Gardening



**Projector**  
High-resolution projector for home theater  
Owner: Amit Verma      Category: Electronics



**Electric Kettle**  
Fast boiling electric kettle for daily use  
Owner: Sneha Reddy      Category: Kitchen Appliances


localhost:51352/#/add\_item

List an Item

Add Item

localhost:51352/#/add\_item

Drill Machine



**Drill Machine**  
**Description:**  
Powerful drill for various tasks  
**Owner:** Rahul Sharma  
**Category:** Home Appliances  

Borrow



**Conclusion**

In this experiment, we successfully implemented an interactive form using TextField, validation checks, and SnackBar feedback for user inputs. Initially, we faced issues with clearing input fields and displaying validation messages, which we resolved by using TextEditingController and condition checks before submission.