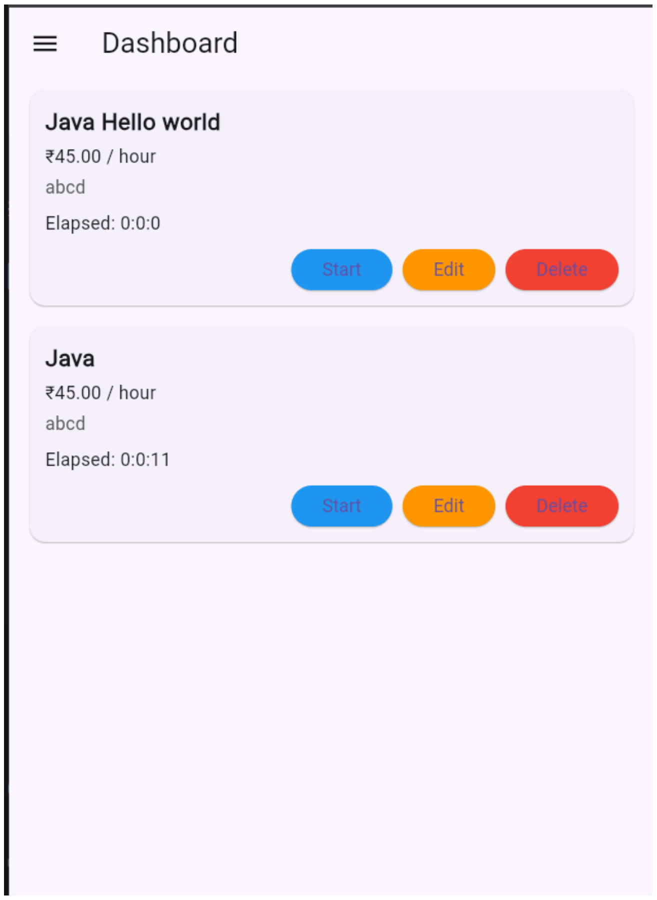
- **Aim: To apply navigation, routing and gestures in Flutter App Theory**

- **Theory**
    - Navigation in Flutter : Navigation refers to the mechanism that allows users to move between different screens or pages within a Flutter app. Flutter uses the Navigator widget to manage a stack of screens, providing a way to push and pop screens.
- **Key Concepts**
    - Navigator : The core widget for managing a stack of routes.
    - Routes: Represent different screens in an app.
    - Push and Pop Navigation:
      Push: Navigates to a new screen. Navigator.push(context, MaterialPageRoute(builder: (context) => SecondScreen()),);
      Pop: Returns to the previous screen. Navigator.pop(context);
    - Named Routes : Using named routes improves code readability: Navigator.pushNamed(context, '/second');
    - Routing in Flutter : Routing refers to how screens or pages are mapped and displayed in an app. In Flutter, routing is managed by the MaterialApp or CupertinoApp widget, which uses routes to determine which screen to show.
        - Types of Routes :
            - Static Routes: Defined at app launch. MaterialApp(routes: { '/': (context) => HomeScreen(), '/second': (context) => SecondScreen(), },);
            - Dynamic Routes: Allow passing parameters when navigating Navigator.push(context, MaterialPageRoute(builder: (context) => DetailScreen(itemId: 42)),);
    - Gestures in Flutter : Gestures are actions performed by the user on the screen, like tapping, swiping, or pinching. Flutter provides the GestureDetector widget to detect these actions.
        - Common Gestures :
            - Tap Gesture: Detects taps. GestureDetector(onTap: () => print("Tapped"), child: Container());
            - Long Press Gesture: Detects long presses. GestureDetector(onLongPress: () => print("Long Pressed"), child: Container());
            - Swipe Gesture: Detects swipes (horizontal or vertical). GestureDetector(onHorizontalDragEnd: (details) => print("Swiped"), child: Container(),);
            - Combining Navigation and Gestures: Gestures can also trigger navigation between screens. GestureDetector(onHorizontalDragEnd: (details) => Navigator.pushNamed(context, '/nextScreen'), child: Container(),);
- Code :

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:fluttertoast/fluttertoast.dart'; // Import FlutterToast
import 'login_screen.dart'; // Import Login screen
import '../app_colors.dart'; // Import AppColors
class VerifyEmailScreen extends StatelessWidget {
VerifyEmailScreen({Key? key}) : super(key: key);
Future<void> _verifyEmail(BuildContext context) async {
try {
User? user = FirebaseAuth.instance.currentUser;
if (user != null) {
// Check if the email is verified
await user.reload();
user = FirebaseAuth.instance.currentUser;
if (user?.emailVerified == true) {
// Show success toast message for email verification
Fluttertoast.showToast(
msg: "Email Verified Successfully!",
toastLength: Toast.LENGTH_SHORT,
gravity: ToastGravity.BOTTOM,
timeInSecForIosWeb: 1,
backgroundColor: Colors.green,
textColor: Colors.white,
fontSize: 16.0,
);
// Navigate to the login screen
Navigator.pushReplacement(
context,
MaterialPageRoute(
builder: (context) => const LoginScreen(),
),
);
} else {
// Show a toast if email is not verified
Fluttertoast.showToast(
msg: "Email is not verified. Please check your inbox.",
toastLength: Toast.LENGTH_SHORT,
gravity: ToastGravity.BOTTOM,
timeInSecForIosWeb: 1,
backgroundColor: Colors.orange,
textColor: Colors.white,
fontSize: 16.0,
}
);
} else {
```

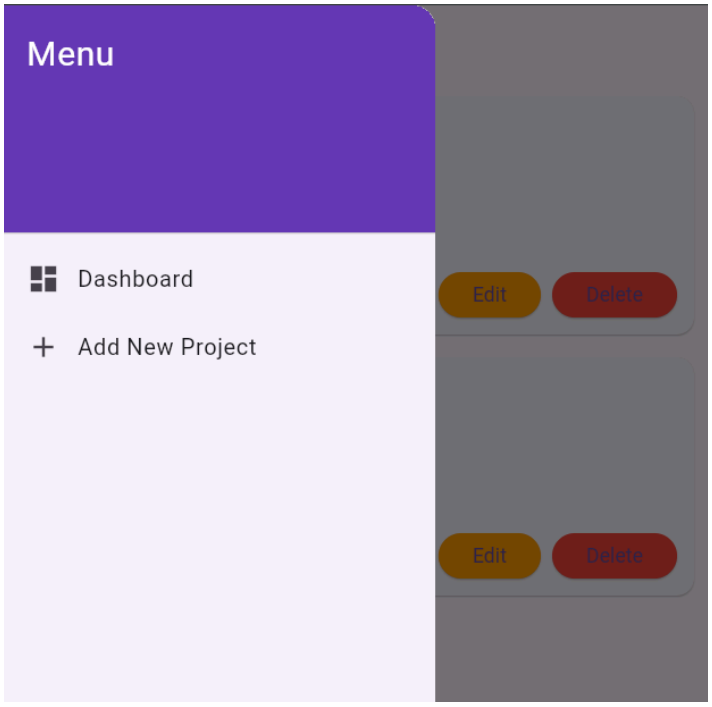
```
// Show toast if no user found
Fluttertoast.showToast(
msg: "No user found or email already verified!",
toastLength: Toast.LENGTH_SHORT,
gravity: ToastGravity.BOTTOM,
timeInSecForIosWeb: 1,
backgroundColor: Colors.red,
textColor: Colors.white,
fontSize: 16.0,
);
}
} catch (e) {
// Show error toast in case of exception
Fluttertoast.showToast(
msg: "Error: ${e.toString()}",
toastLength: Toast.LENGTH_SHORT,
gravity: ToastGravity.BOTTOM,
timeInSecForIosWeb: 1,
backgroundColor: Colors.red,
textColor: Colors.white,
fontSize: 16.0,
);
}
}
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
leading: IconButton(
icon: const Icon(Icons.arrow_back),
onPressed: () {
Navigator.pop(context);
},
color: AppColors.backgroundColor,
),
backgroundColor: AppColors.primaryColor,
elevation: 0,
),
body: Padding(
padding: const EdgeInsets.all(20.0),
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text(
'Verify Your Email',
style: TextStyle(
fontSize: 24,
fontWeight: FontWeight.bold,
color: AppColors.primaryColor,
),
),
const SizedBox(height: 20),
const Text(
'We have sent a verification email to your provided address. Please
check your inbox and verify your email.',
textAlign: TextAlign.center,
style: TextStyle(
fontSize: 16,
color: Colors.grey,
),
),
const SizedBox(height: 20),
GestureDetector(
onTap: () {
_verifyEmail(context);
},
              child: Container(
                height: 55,
                width: 300,
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(30),
                  gradient: const LinearGradient(
                    colors: [AppColors.primaryColor, AppColors.secondaryColor],
                  ),
                ),
                child: const Center(
                  child: Text(
                    'CHECK EMAIL VERIFICATION',
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 20,
                      color: AppColors.backgroundColor,
                    ),
                  ),
                ),
              ),
            ),
          ],
          ),
        ),
      );
    }
  }
```

- Output :


Dashboard

Java Hello world
₹45.00 / hour
abcd
Elapsed: 0:0:0

Java
₹45.00 / hour
abcd
Elapsed: 0:0:11



- Conclusion : In Flutter, navigation, routing, and gestures are integral for creating interactive and user-friendly applications.
  ● Navigation enables seamless transitions between screens, using methods like push and pop.
  ● Routing allows for both static and dynamic screen management, enhancing flexibility and modularity in app design.
  ● Gestures offer a way for users to interact with the app, making the experience more engaging. By mastering these concepts, developers can build well-structured, responsive, and intuitive applications in Flutter. These techniques are essential for handling user interactions, creating dynamic navigation flows, and optimizing the overall user experience in any mobile app.