MAD PWD LAB 5

AIM:TO APPLY NAVIGATION,ROUTING AND GESTURES IN FLUTTER APP.

Theory:

In Flutter, navigation and routing are essential for building dynamic and interactive user interfaces.
Navigation refers to the process of moving between different screens or "routes" within your application.
Routing, on the other hand, is the mechanism that manages the navigation flow, determining which screen to display based on user interactions or application logic.
Following are the methods to handle routing in flutter

1.Navigator class: Flutter provides the Navigator class, which manages a stack of Route objects. You can push new routes onto the stack to navigate forward, pop routes to navigate backward, or replace routes to update the current screen.

2.Named routes: Named routes allow you to define routes with unique names and associated widgets. This approach is particularly useful for managing navigation in larger applications. You can define named routes in your app's MaterialApp or CupertinoApp widget using the routes parameter.
3.Navigation methods: Flutter provides various methods for navigating between screens:

1.Navigator.push(): Pushes a new route onto the navigator's stack.
2.Navigator.pop(): Pops the current route off the navigator's stack.
3.Navigator.pushNamed(): Pushes a named route onto the navigator's stack.
4.Navigator.popAndPushNamed(): Pops the current route off the stack and pushes a named route.
5.Route generation: You can implement custom logic for generating routes dynamically based on certain conditions. This is often done using the onGenerateRoute callback in MaterialApp or CupertinoApp.

6.Passing data between screens: You can pass data between screens when navigating using constructor arguments or using the ModalRoute static methods to access the route's settings.

Code:

```
static Route<dynamic> onGenerateRoute(RouteSettings settings) {
 switch (settings.name) {
   case welcome:
     return MaterialPageRoute(
       builder: (context) => const WelcomePage(),
     );

   case login:
     return MaterialPageRoute(
       builder: (context) => const LoginPage(),
     );

   case verification:
     final Map args = settings.arguments as Map;
     return MaterialPageRoute(
       builder: (context) => VerificationPage(phoneNumber: '',
           //VerificationId: args['verification'],
           //PhoneNumber: args['phone Number'],
           ),
     );
```

OUTPUT:

SCREEN 1

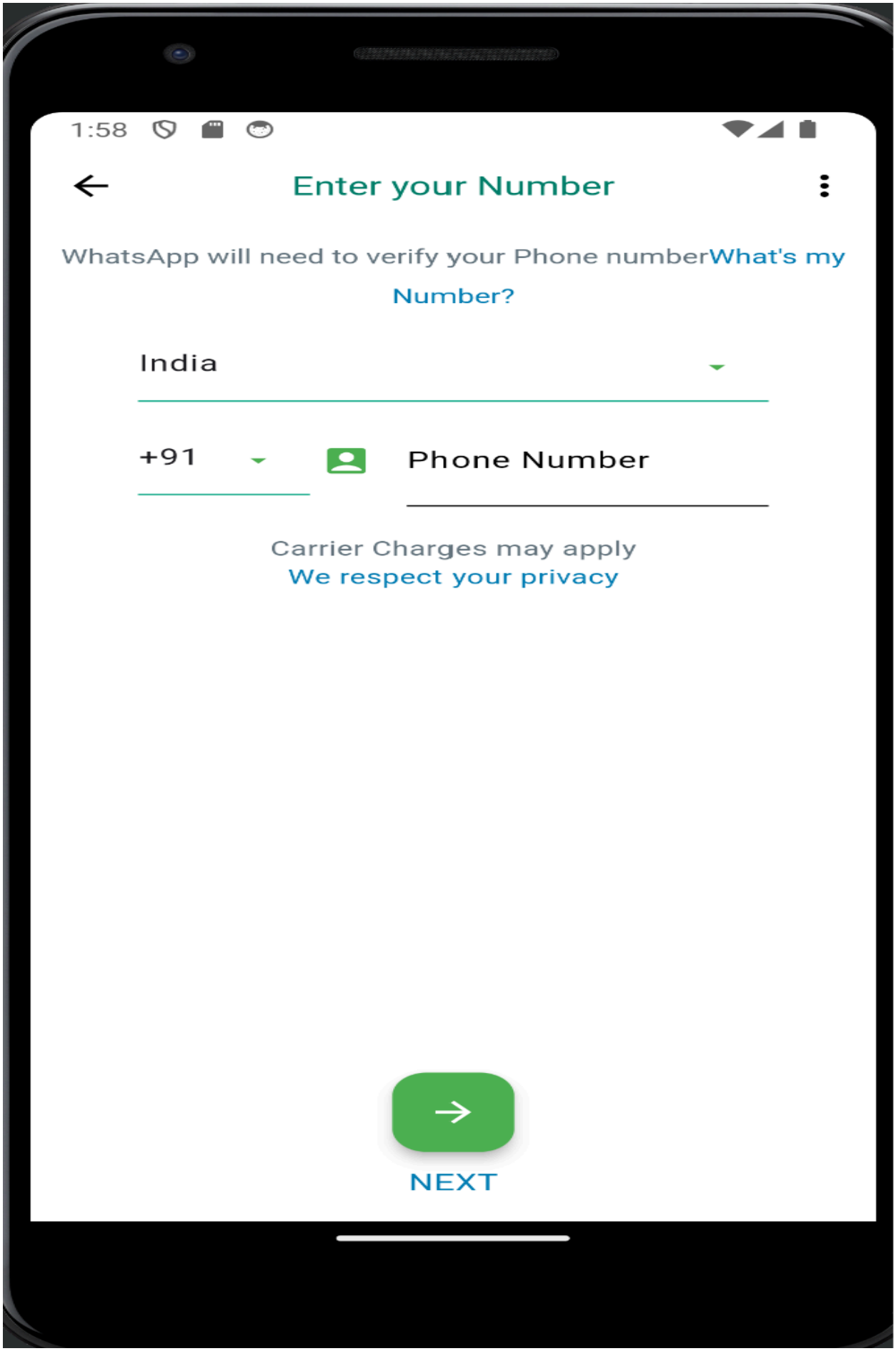
Welcome to WhatsApp

Read our Privacy Policy. Tap " Agree and continue" to accept our Terms of Services

AGREE AND CONTINUE

English

SCREEN 2


1:58
Enter your Number
WhatsApp will need to verify your Phone numberWhat's my Number?
India
+91
Phone Number
Carrier Charges may apply
We respect your privacy
NEXT

Conclusion:Successfully studied the concept of routing in flutter web app.