AIM:To apply navigation, routing and gestures in Flutter App

In Flutter, you can implement navigation, routing, and gestures to create a smooth and intuitive user experience. Here's a step-by-step guide on how to achieve these functionalities in your Flutter app:

1) Navigating and Routing:

a. Define Screens:

Create separate Dart files for each screen in your app. For example, you might have HomePage.dart, DetailPage.dart, etc.

b. Create Routes:

In your main.dart file, define routes using the MaterialApp widget. This widget takes a routes parameter, where you can map route names to the corresponding screen widgets.

```
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => HomePage(),
        '/detail': (context) => DetailPage(),
      },
    );
  }
}
```

c. Navigate to a New Screen:

You can use the Navigator class to navigate between screens. For example, use Navigator.pushNamed to navigate to a new screen by its route name.
Navigator.pushNamed(context, '/detail');

d. Pass Data between Screens:

To pass data between screens, you can use the Navigator arguments.

Navigator.pushNamed(context, '/detail', arguments: {'data': 'Hello from HomePage!'});

2. Gestures:

a. GestureDetector:

Use the GestureDetector widget to detect various gestures like taps, swipes, etc.
dart

```
GestureDetector(
 onTap: () {
 // Handle tap gesture
 },
 onDoubleTap: () {
 // Handle double-tap gesture
 },
 onLongPress: () {
 // Handle long-press gesture
 },
 onHorizontalDragEnd: (DragEndDetails details) {
 // Handle horizontal swipe
 },
 child: YourWidget(),
)
```

b. Dismissible:

To implement dismissible gestures, you can use the Dismissible widget for swiping elements off the screen.
dart

```
Dismissible(
 key: Key("your_key"),
 onDismissed: (direction) {
 // Handle swipe direction (left or right)
 },
 background: Container(
 color: Colors.red,
 child: Icon(Icons.delete),
 ),
 child: YourListItem(),
)
```

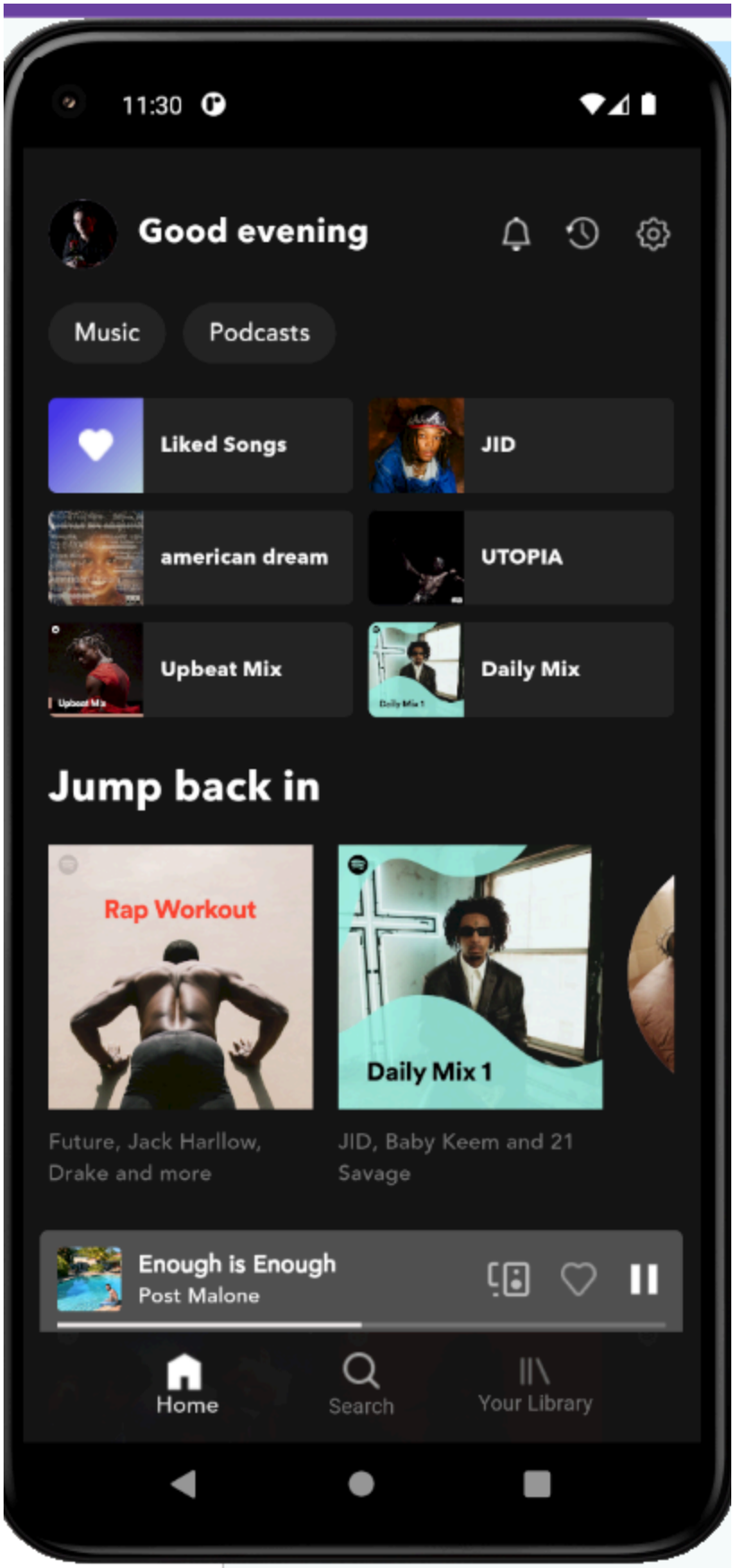
These steps should help you implement navigation, routing, and gestures in your Flutter app. Adjust them according to your specific requirements.

CODE:
(homescreen)

lib > main.dart > MyApp > build

```dart
import 'package:flutter/material.dart';
import 'package:spotify_clone/DI/service_locator.dart';
import 'package:spotify_clone/ui/splash_screen.dart';

Run | Debug | Profile
void main() {
  initServiceLocator();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        splashColor: Colors.transparent,
      ), // ThemeData
      debugShowCheckedModeBanner: false,
      home: const SplashScreen(),
    ); // MaterialApp
  }
}
```

OUTPUT:

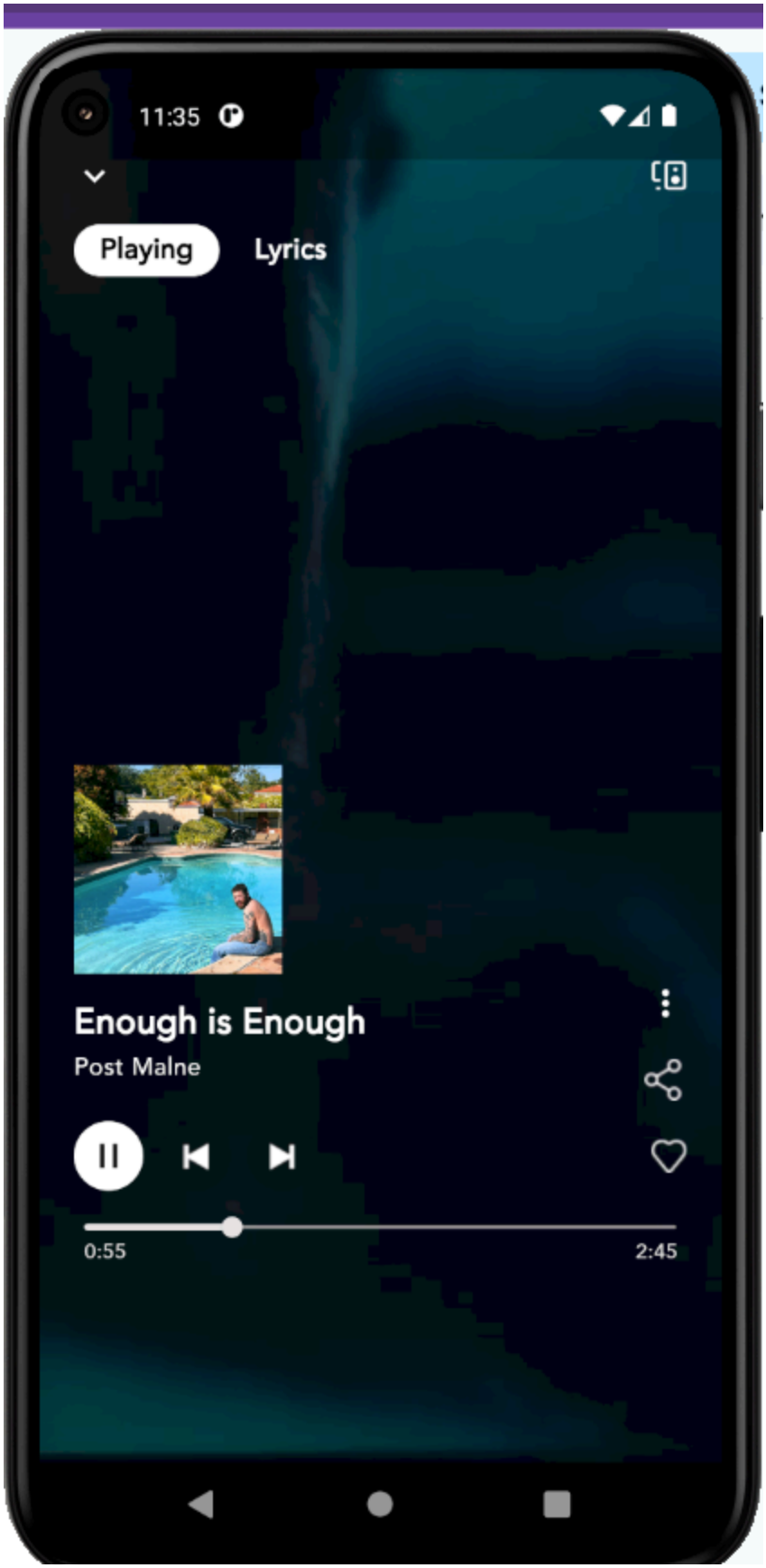

11:30
Good evening
Music
Podcasts
Liked Songs
JID
american dream
UTOPIA
Upbeat Mix
Daily Mix
Jump back in
Rap Workout
Daily Mix 1
Future, Jack Harllow, Drake and more
JID, Baby Keem and 21 Savage
Enough is Enough
Post Malone
Home
Search
Your Library

Now navigating to video player:
CODE:

lib > widgets > video_player.dart > ...

```dart
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';
import 'package:video_player/video_player.dart';

class BackVideoPlayer extends StatefulWidget {
  const BackVideoPlayer({super.key});

  @override
  State<BackVideoPlayer> createState() => _BackVideoPlayerState();
}

class _BackVideoPlayerState extends State<BackVideoPlayer> {
  late final VideoPlayerController controller;

  @override
  void initState() {
    super.initState();
    controller = VideoPlayerController.asset('video/music-video.mp4')
      ..initialize().then((value) {
        controller.play();
        controller.setLooping(true);
        setState(() {});
      });
  }

  @override
  void dispose() {
    controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: [
        FittedBox(
```

OUTPUT:


11:35
Playing
Lyrics
Enough is Enough
Post Malne
0:55
2:45

conclusion:

In conclusion, implementing navigation, routing, and gestures in a Flutter app can greatly enhance the user experience and make the app more intuitive and engaging. By following the steps outlined above, you can structure your app with separate screens, define routes for navigation, and incorporate gestures for interactive elements. Flutter provides powerful widgets like Navigator for navigation, GestureDetector for handling gestures, and Dismissible for swipe actions. By combining these tools, you can create a seamless and user-friendly Flutter application. Remember to customize the implementation based on your specific app requirements and design preferences.