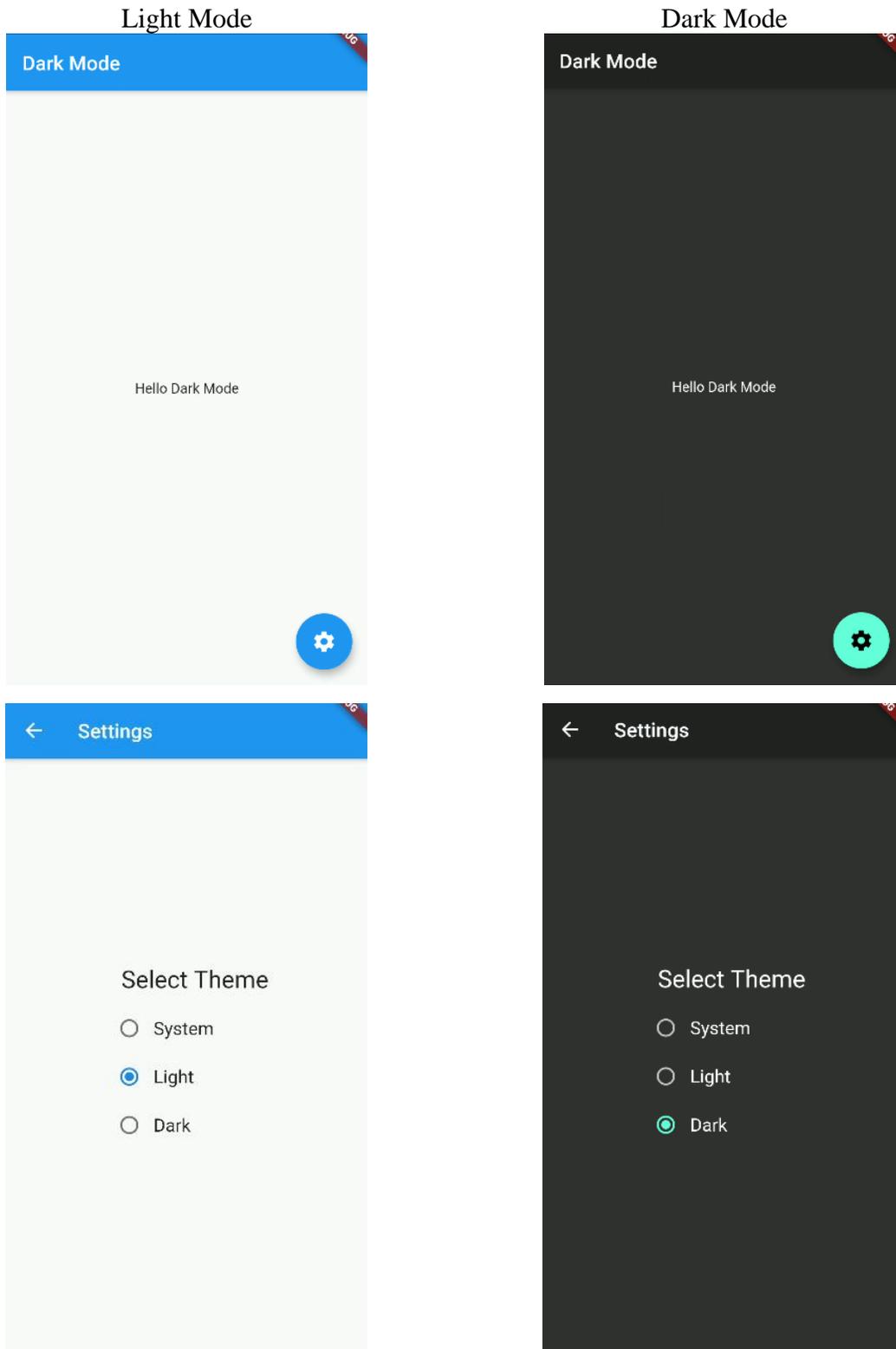


## State Management

**State Management** adalah sebuah desain dalam coding di mana kita dapat memisahkan antara logic dan view, Tujuannya adalah agar logic dapat kembali digunakan. Pada kotlin atau java biasanya kita mengenal istilah **MVVM** atau **MVP**, pada web **MVC**, maka di Flutter ada **BLoC**, **Provider**, **Redux**, atau **MobX**.

Kali ini, kita akan menggunakan salah satunya yakni **Provider**. Contohnya adalah untuk mengubah tema warna tampilan aplikasi seperti contoh di bawah:



Untuk menggunakan package **Provider**, pastikan Anda memasukkannya dalam **pubspec.yaml**

## 1. Main

```
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2 import 'package:provider/provider.dart';
3 import 'settings_screen.dart';
4 import 'theme_manager.dart';
5
6 void main() => runApp(MyApp());
7
8 class MyApp extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return ChangeNotifierProvider<ThemeManager>([
12      create: (context) => ThemeManager(),
13      child: MaterialAppWithTheme(),
14    ]); // ChangeNotifierProvider
15  }
16 }
17
18 class MaterialAppWithTheme extends StatelessWidget {
19   @override
20  Widget build(BuildContext context) {
21    final themeManager = Provider.of<ThemeManager>(context);
22    return MaterialApp(
23      home: StartScreen(),
24      theme: ThemeData.light(),
25      darkTheme: ThemeData.dark(),
26      themeMode: themeManager.themeMode,
27    ); // MaterialApp
28  }
29 }
30
31 class StartScreen extends StatelessWidget {
32   StartScreen({Key key}) : super(key: key);
33
34   @override
35   Widget build(BuildContext context) {
36     return Scaffold(
37       appBar: AppBar(
38         title: Text("Dark Mode"),
39       ), // AppBar
40       body: Center(
41         child: Column(
42           mainAxisAlignment: MainAxisAlignment.center,
43           children: <Widget>[
44             Text("Hello Dark Mode"),
45           ], // <Widget>[]
46         ), // Column
47       ), // Center
48       floatingActionButton: FloatingActionButton(
49         onPressed: () {
50           Navigator.push(
51             context,
52             MaterialPageRoute(
53               builder: (context) => SettingsScreen(),
54             ), // MaterialPageRoute
55           );
56         },
57         tooltip: 'Settings',
58         child: Icon(Icons.settings),
59       ), // FloatingActionButton
60     ); // Scaffold
61   }
62 }
```

```
lib > main.dart > MyApp > build
25   darkTheme: ThemeData.dark(),
26   themeMode: themeManager.themeMode,
27 ); // MaterialApp
28 }
29 }
30
31 class StartScreen extends StatelessWidget {
32   StartScreen({Key key}) : super(key: key);
33
34   @override
35   Widget build(BuildContext context) {
36     return Scaffold(
37       appBar: AppBar(
38         title: Text("Dark Mode"),
39       ), // AppBar
40       body: Center(
41         child: Column(
42           mainAxisAlignment: MainAxisAlignment.center,
43           children: <Widget>[
44             Text("Hello Dark Mode"),
45           ], // <Widget>[]
46         ), // Column
47       ), // Center
48       floatingActionButton: FloatingActionButton(
49         onPressed: () {
50           Navigator.push(
51             context,
52             MaterialPageRoute(
53               builder: (context) => SettingsScreen(),
54             ), // MaterialPageRoute
55           );
56         },
57         tooltip: 'Settings',
58         child: Icon(Icons.settings),
59       ), // FloatingActionButton
60     ); // Scaffold
61   }
62 }
```

## 2. Settings\_screen

```
1 import 'package:flutter/material.dart';
2 import 'package:provider/provider.dart';
3 import 'theme_manager.dart';
4
5 class _SettingsScreenState extends State<SettingsScreen> {
6   ThemeMode _groupValue;
7   ThemeManager _themeManager;
8
9   @override
10  void initState() {
11    super.initState();
12    _themeManager = Provider.of<ThemeManager>(context, listen: false);
13    _groupValue = _themeManager.themeMode;
14  }
15
16  void _updateTheme(ThemeMode themeMode) {
17    _themeManager.themeMode = themeMode;
18  }
19
20  @override
21  Widget build(BuildContext context) {
22    return Scaffold(
23      appBar: AppBar(
24        title: Text('Settings'),
25      ), // AppBar
26      body: Center(
27        child: Column(
28          mainAxisAlignment: MainAxisAlignment.max,
29          mainAxisAlignment: MainAxisAlignment.center,
30          crossAxisAlignment: CrossAxisAlignment.start,
31          children: <Widget>[
32            Padding(
33              padding: EdgeInsets.only(left: 16.0, bottom: 10.0)
```

```
32            Padding(
33              padding: EdgeInsets.only(left: 16.0, bottom: 10.0),
34              child: Text("Select Theme", style: TextStyle(fontSize: 24.0)),
35            ), // Padding
36          Row(
37            mainAxisAlignment: MainAxisAlignment.start,
38            mainAxisAlignment: MainAxisAlignment.min,
39            children: <Widget>[
40              Radio(
41                onChanged: (val) => setState() {
42                  _groupValue = val;
43                  _updateTheme(val);
44                },
45                value: ThemeMode.system,
46                groupValue: _groupValue,
47              ), // Radio
48              GestureDetector(
49                onTap: () => setState() {
50                  var val = ThemeMode.system;
51                  _groupValue = val;
52                  _updateTheme(val);
53                },
54                child: Text(
55                  "System",
56                  style: TextStyle(fontSize: 18.0),
57                ), // Text
58              ), // GestureDetector
59            ], // <Widget>[]
60          ), // Row
61          Row(
62            mainAxisAlignment: MainAxisAlignment.start,
63            mainAxisAlignment: MainAxisAlignment.min,
```

```
63            mainAxisAlignment: MainAxisAlignment.min,
64            children: <Widget>[
65              Radio(
66                onChanged: (val) => setState() {
67                  _groupValue = val;
68                  _updateTheme(val);
69                },
70                value: ThemeMode.light,
71                groupValue: _groupValue,
72              ), // Radio
73              GestureDetector(
74                onTap: () => setState() {
75                  var val = ThemeMode.light;
76                  _groupValue = val;
77                  _updateTheme(val);
78                },
79                child: Text(
80                  "Light",
81                  style: TextStyle(fontSize: 18.0),
82                ), // Text
83              ), // GestureDetector
84            ], // <Widget>[]
85          ), // Row
86          Row(
87            mainAxisAlignment: MainAxisAlignment.start,
88            mainAxisAlignment: MainAxisAlignment.min,
89            children: <Widget>[
90              Radio(
91                onChanged: (val) => setState() {
92                  _groupValue = val;
93                  _updateTheme(val);
94                },
95                value: ThemeMode.dark,
```

```
92                _groupValue = val;
93                _updateTheme(val);
94              ),
95              value: ThemeMode.dark,
96              groupValue: _groupValue,
97            ), // Radio
98            GestureDetector(
99              onTap: () => setState() {
100                var val = ThemeMode.dark;
101                _groupValue = val;
102                _updateTheme(val);
103              },
104              child: Text(
105                "Dark",
106                style: TextStyle(fontSize: 18.0),
107              ), // Text
108            ), // GestureDetector
109          ], // <Widget>[]
110        ), // Row
111      ], // <Widget>[]
112    ), // Column
113  ), // Center
114 ); // Scaffold
115 }
116 }
117
118 class SettingsScreen extends StatefulWidget {
119   @override
120   _SettingsScreenState createState() => _SettingsScreenState();
121 }
122
```

### 3. Theme\_manager

```
1 import 'package:flutter/material.dart';
2 import 'package:shared_preferences/shared_preferences.dart';
3
4 class ThemeManager with ChangeNotifier {
5   ThemeMode _themeMode;
6   ThemeMode defaultThemeMode = ThemeMode.system;
7
8   ThemeManager() {
9     _getThemeModeFromSharedPrefs();
10  }
11
12  get themeMode {
13    return _themeMode;
14  }
15
16  set themeMode(ThemeMode themeMode) {
17    _themeMode = themeMode;
18    _saveThemeModeInSharedPrefs(themeMode);
19    notifyListeners();
20  }
21
22  void _getThemeModeFromSharedPrefs() async {
23    SharedPreferences prefs = await SharedPreferences.getInstance();
24    String themeModeFromPrefs = prefs.getString('themeMode');
25
26    _themeMode = ThemeMode.values.firstWhere(
27      (element) => themeModeFromPrefs == element.toString(),
28      orElse: () => defaultThemeMode
29    );
30    notifyListeners();
31  }
32
33  _saveThemeModeInSharedPrefs(ThemeMode themeMode) async {
34    SharedPreferences prefs = await SharedPreferences.getInstance();
35    prefs.setString('themeMode', themeMode.toString());
36  }
37 }
38
```