



# Getting Started with Flutter

Jumail Bin Taliba  
School of Computing, UTM  
March 2020

# Outline

- Creating Flutter Projects
- Flutter Project Structure
- Building UI by example

# Several ways of creating Flutter projects

- From VS Code
- Using Flutter tool from Command Line (Git Bash, Cmd or Powershell)
- Clone from an existing project

# Creating Flutter Project from VS Code

1. In VS Code, open Command Palette, `Ctrl Shift P`
2. Choose from the menu, `Flutter New Project`

# Creating Flutter Project with CLI

1. Open Git Bash
2. Move to the directory where you save all your flutter projects.  
`cd d:/code/flutter`
3. Create a new directory in it for your new project  
`mkdir flutter_counter`
4. Move to the new project directory  
`cd flutter_counter`
5. Run flutter tool to start creating the project (*don't forget the dot*)  
`flutter create .`
6. Open the source code into VS Code (*don't forget the dot*)  
`code .`

# Creating Flutter Project by cloning

1. Copy the folder of any of your existing project.
  - You can clone from git file (created with git bundle), or from online repo (e.g. github)
  - You can copy from a zip file, or a folder

*In the following example, we are going to copy / clone from my project template on github.*

2. Clone my github repo (*command below should be in one-line*)

```
git clone https://github.com/jumail-  
utm/flutter_template.git my_project
```

*You can use any project name. Here, we use my\_project*

3. Move to the project directory

```
cd my_project
```

4. Run flutter tool to **properly** rename the project (*don't forget the dot*)

```
flutter create -project-name .
```

5. Open the source code into VS Code (*don't forget the dot*)

```
code .
```

# Flutter Project Structure

> .dart\_tool

> .idea

> android

> ios

✓ lib

📄 main.dart

*you will work on this file(s)*

> test

📄 .gitignore

≡ .metadata

≡ .packages

🔥 flutter\_counter.iml

≡ pubspec.lock

! pubspec.yaml

*to download dependencies*

📖 README.md

# Basic Flutter Program

```
lib > main.dart > ...
1  import 'package:flutter/material.dart';
2
3  void main() {
4      runApp(
5          MaterialApp(
6              title: 'Flutter Project',
7              home: Scaffold(
8                  appBar: AppBar(
9                      title: Text('Flutter'),
10                     ), // AppBar
11                 ), // Scaffold
12             ), // MaterialApp
13         ); // runApp
14     }
15
```

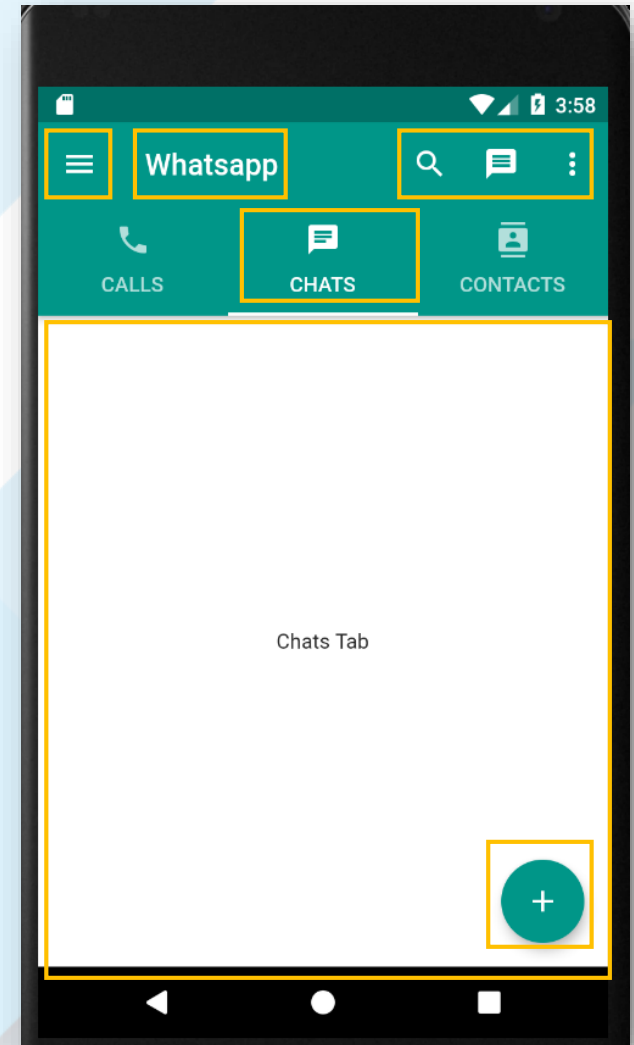


# For Convenience

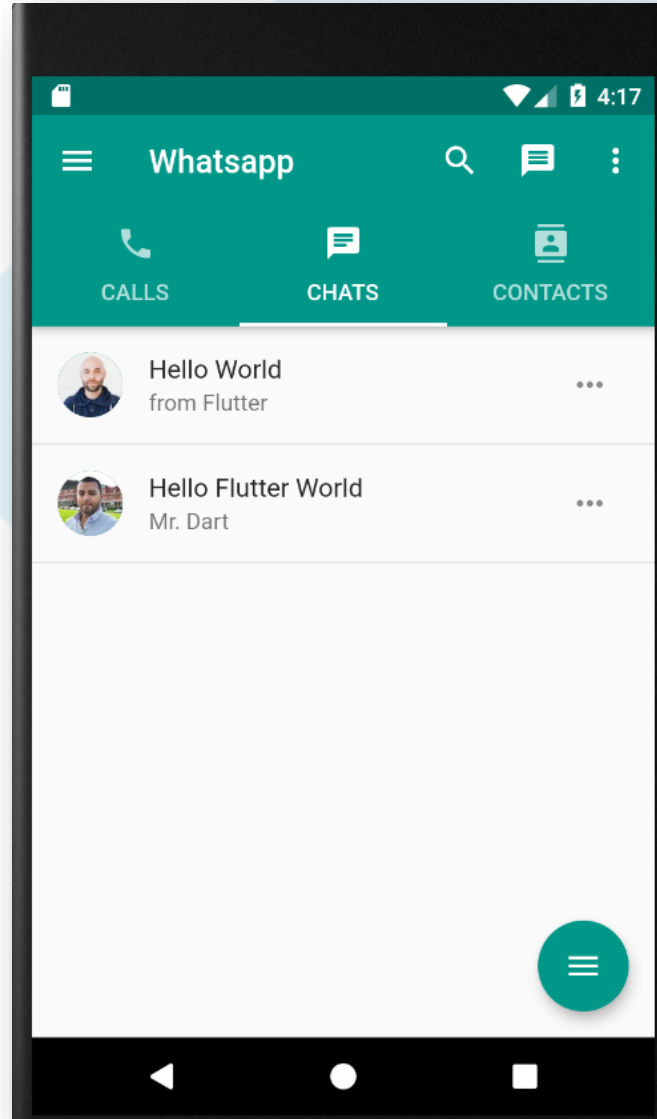
- Install VS Code extensions
  - Flutter
  - Enable: UI guides
  - Bracket Pair Colorizer
  - Pub spec Assist
- Use USB debugging (i.e. running directly on real phone)
  - Enable USB debugging and developer mode in your phone
- Use Hot Reload or Hot Restart when debugging your code
  - Hot Reload : `Ctrl F5`
  - Hot Restart : `Ctrl Shift F5`

# Introduction to Widgets

- What is a widget?
  - The building blocks that construct the application
- Widgets in Flutter are immutable
- Type of widgets:
  - Stateless – does not change at all
  - Stateful – can change to reflect the state
- Common widgets:
  - App widgets (e.g. MaterialApp)
  - Convenience widgets: Scaffold, NavigationBar, AppBar
  - Layout widgets: Column, Row, ListView
  - Text widgets: Text, RichText
  - Button widgets: IconButton, RaisedButton, FloatingActionButton
  - Graphics widgets: Image, Icon



# Building UI, example: Whatsapp Clone



# Preparing Starter Project (1)

1. Open Git Bash
2. Clone my github repo (*command below should be in one-line*)  

```
git clone https://github.com/jumail-utm/flutter_whatsapp_ui
```
3. Move to the project directory  

```
cd flutter_whatsapp_ui
```
4. Go to the starting point of the project and create a new branch to start editing  

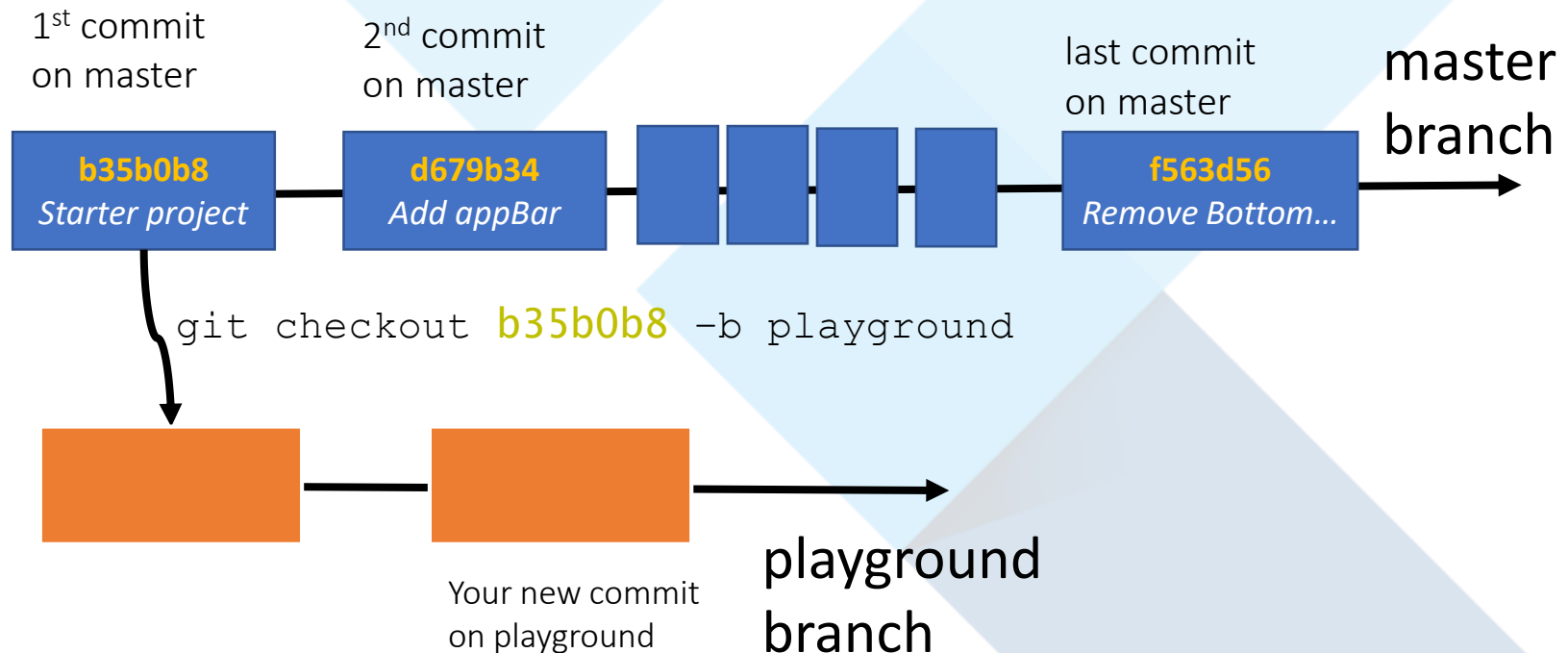
```
git checkout b35b0b8 -b playground
```
5. Open the source code into VS Code (*don't forget the dot*)  

```
code .
```
6. Try running the program by pressing F5 (*you want to choose an emulator first*)

# Git Basics

```
git log --oneline
```

```
f563d56 (HEAD -> master) remove BottomNavigationBar and do basic refactoring
37ac9ca add BottomNavigationBar
58ae5e6 add FloatingActionButton
362b386 add TabView
09bb5da add TabBar
d679b34 add appBar
b35b0b8 starter project
```



# Preparing Starter Project (2)



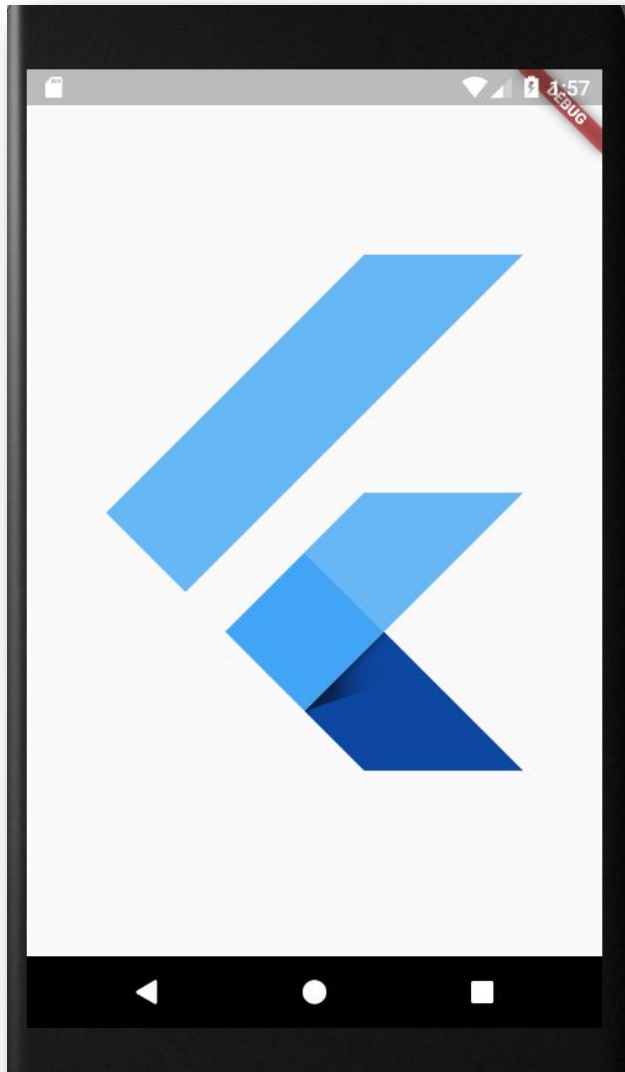
```
lib > main.dart > ...  
1  import 'package:flutter/material.dart';  
2  
3  void main() => runApp(  
4    |    | FlutterLogo(  
5    |    |   size: double.infinity,  
6    |    | ),  
7    |    | );
```

# Creating a Material App



```
lib > main.dart > ...
1   import 'package:flutter/material.dart';
2
3   void main() => runApp(
4     MaterialApp(
5       title: 'Whatsapp Clone',
6       home: FlutterLogo(),
7     ), // MaterialApp
8   );
9
```

# Creating a Home Screen (1)



```
lib > main.dart > ...
1  import 'package:flutter/material.dart';
2
3  void main() => runApp(
4    MaterialApp(
5      title: 'Whatsapp Clone',
6      home: Scaffold(
7        body: FlutterLogo(
8          size: double.infinity,
9        ), // FlutterLogo // Scaffold
10     ), // MaterialApp
11   );
```



# Creating a Home Screen (2)



```
2
3   void main() => runApp(
4     MaterialApp(
5       title: 'Whatsapp Clone',
6       home: Home(),
7     ), // MaterialApp
8   );
9
10  class Home extends StatelessWidget {
11    @override
12    Widget build(BuildContext context) {
13      return Scaffold(
14        appBar: AppBar(
15          title: Text('Whatsapp'),
16        ), // AppBar
17        body: FlutterLogo(
18          size: double.infinity,
19        ), // FlutterLogo
20      ); // Scaffold
21    }
22  }
23
```

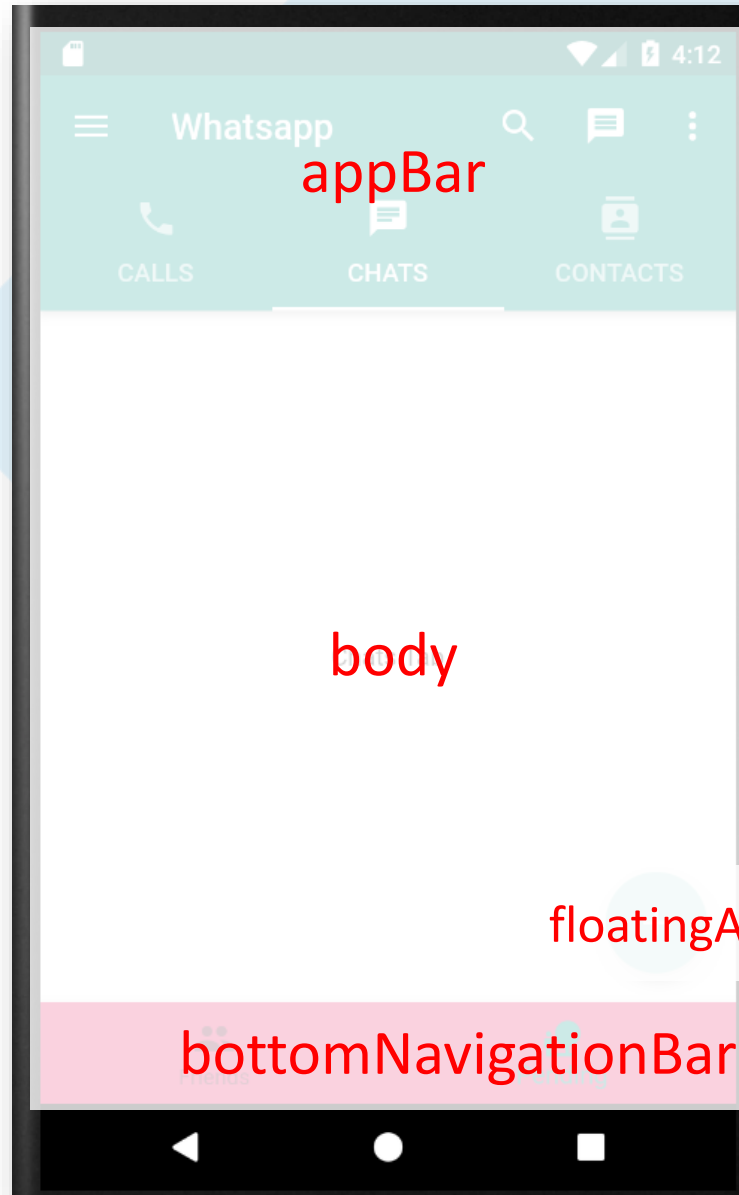
# What is MaterialApp?

- A convenience widget that wraps a number of widgets commonly required for applications that embraced material design.
- It provides a bunch of things including:
  - Theming
  - Routing
  - Main route (home)
  - Localization
- MaterialApp is used to organize multiple screens for the app

# Scaffold

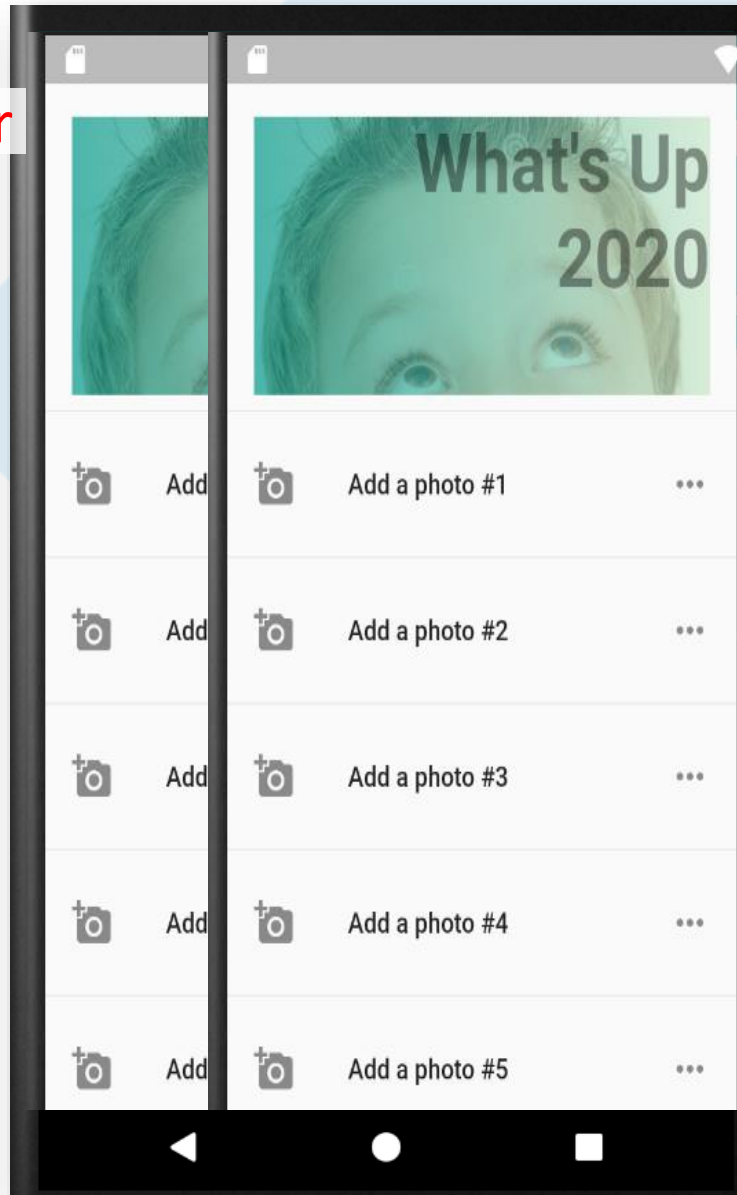
- Provides the basic visual layout structure based on material design.
- It provides a bunch of things including:
  - appBar
  - drawer and endDrawer
  - bottomNavigationBar
  - Body
  - floatingActionButton
  - SnackBar
- Scaffold is commonly used to build UI for a screen

# Scaffold Components (1)



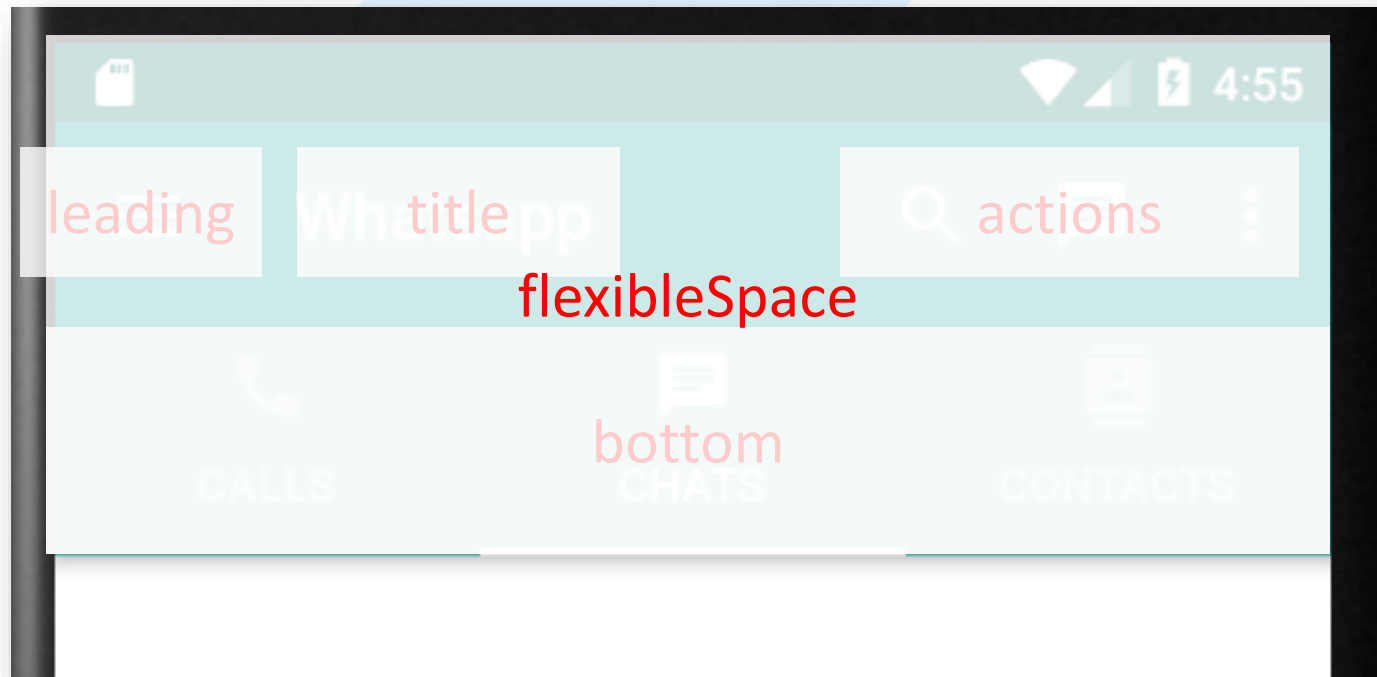
# Scaffold Components (1)

drawer



endDrawer

# AppBar Components



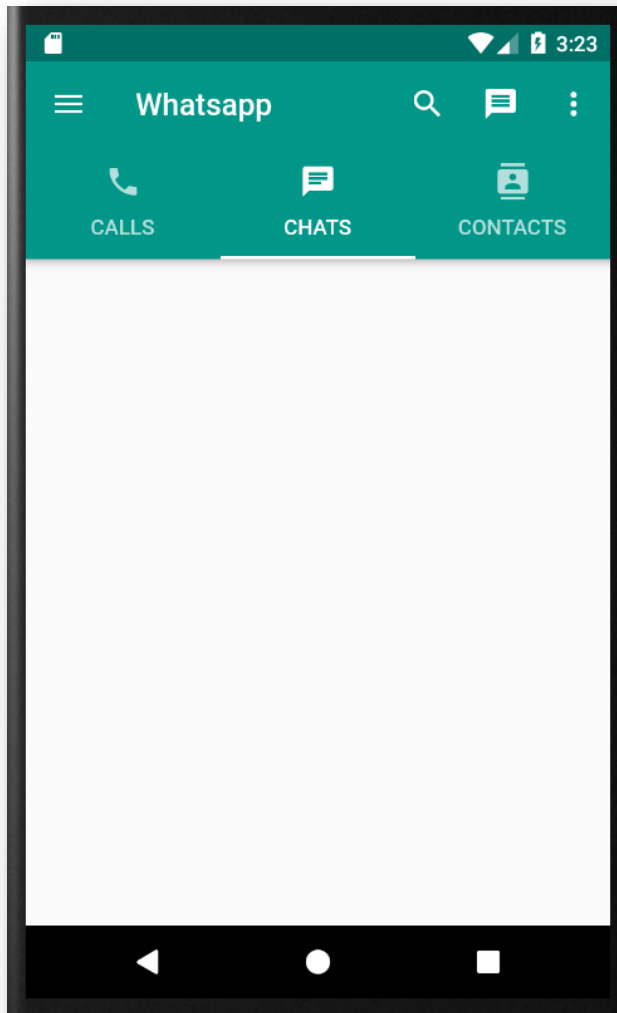
# Adding AppBar



```
3 void main() => runApp(  
4   MaterialApp(  
5     debugShowCheckedModeBanner: false,  
6     title: 'Whatsapp Clone',  
7     theme: ThemeData(primarySwatch: Colors.teal),  
8     home: Home(),  
9   ), // MaterialApp  
10 );
```

```
12 class Home extends StatelessWidget {  
13   void _doNothing() {}  
14  
15   @override  
16   Widget build(BuildContext context) {  
17     return Scaffold(  
18       appBar: AppBar(  
19         leading: IconButton(  
20           icon: Icon(Icons.menu),  
21           onPressed: _doNothing,  
22         ), // IconButton  
23         title: Text('Whatsapp'),  
24         actions: <Widget>[  
25           IconButton(  
26             icon: Icon(Icons.search),  
27             onPressed: _doNothing,  
28           ), // IconButton  
29           IconButton(  
30             icon: Icon(Icons.message),  
31             onPressed: _doNothing,  
32           ), // IconButton  
33           IconButton(  
34             icon: Icon(Icons.more_vert),  
35             onPressed: _doNothing,  
36           ), // IconButton  
37         ], // <Widget>[]  
38       ), // AppBar  
39     ); // Scaffold  
40   }  
41 }
```

# Adding TabBar

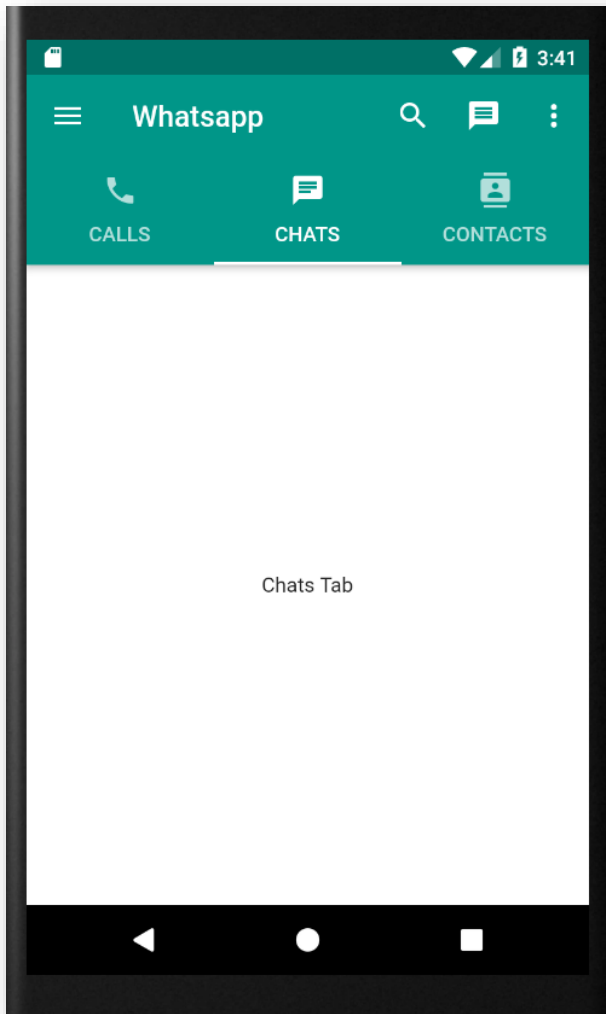


```
12 class Home extends StatelessWidget {
13   void _doNothing() {}
14
15   @override
16   Widget build(BuildContext context) {
17     return DefaultTabController(
18       length: 3,
19       initialIndex: 1,
20       child: Scaffold(
21         appBar: AppBar(
22           leading: IconButton(
23             icon: Icon(Icons.menu),
24             onPressed: _doNothing,
25           ), // IconButton
26           title: Text('Whatsapp'),
27           actions: <Widget>[
28             IconButton(icon: Icon(Icons.search), onPressed: _doNothing),
29             IconButton(icon: Icon(Icons.message), onPressed: _doNothing),
30             IconButton(icon: Icon(Icons.more_vert), onPressed: _doNothing),
31           ], // <Widget>[]
32         bottom: TabBar(tabs: [
33           Tab(text: 'CALLS', icon: Icon(Icons.call)),
34           Tab(text: 'CHATS', icon: Icon(Icons.chat)),
35           Tab(text: 'CONTACTS', icon: Icon(Icons.contacts)),
36         ]), // TabBar
37       ), // AppBar
38     ), // Scaffold
39   ); // DefaultTabController
40 }
41 }
```

*TabBar must have TabController*

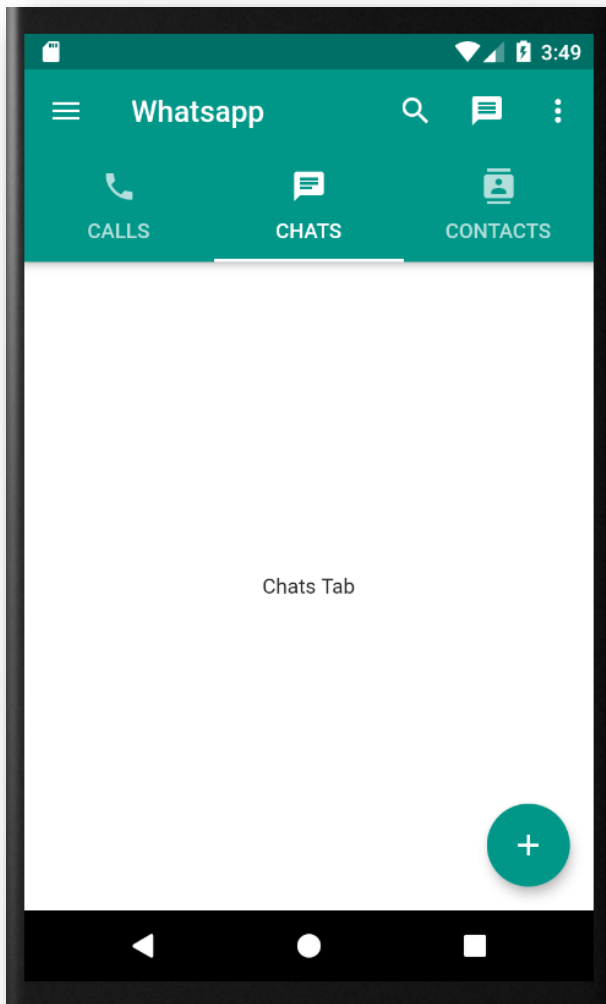


# Adding TabView



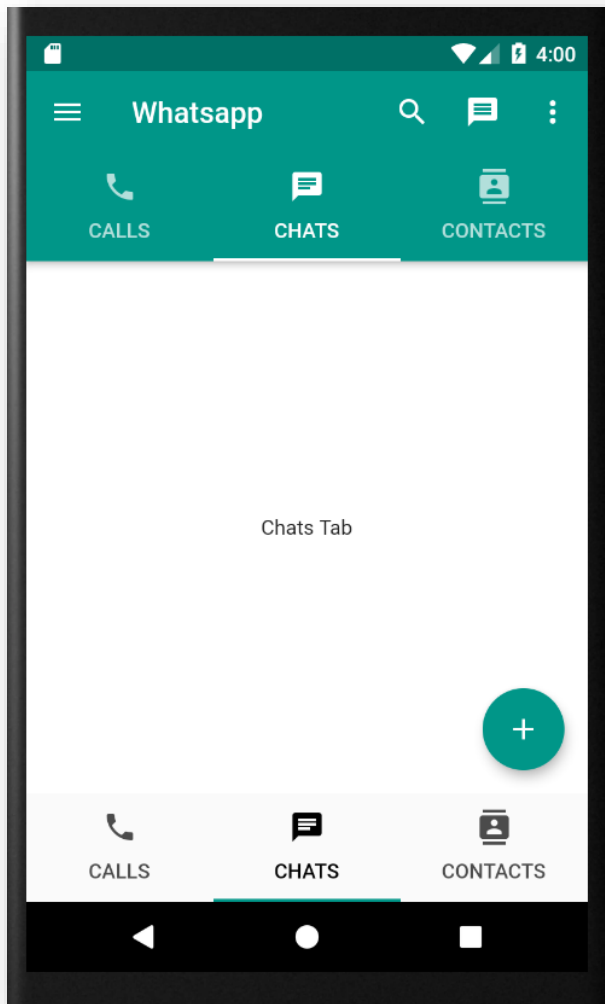
```
12 class Home extends StatelessWidget {  
13   void _doNothing() {}  
14  
15   @override  
16   Widget build(BuildContext context) {  
17     return DefaultTabController(  
18       length: 3,  
19       initialIndex: 1,  
20       child: Scaffold(  
21         appBar: AppBar( // AppBar  
38         body: TabBarView(children: [  
39           Container(color: Colors.pink, child: Center(child: Text('Calls Tab'))),  
40           Container(color: Colors.white, child: Center(child: Text('Chats Tab'))),  
41           Container(color: Colors.amber, child: Center(child: Text('Contact Tab'))),  
42         ]), // TabBarView  
43       ), // Scaffold  
44     ); // DefaultTabController  
45   }  
46 }
```

# Adding FloatingActionButton



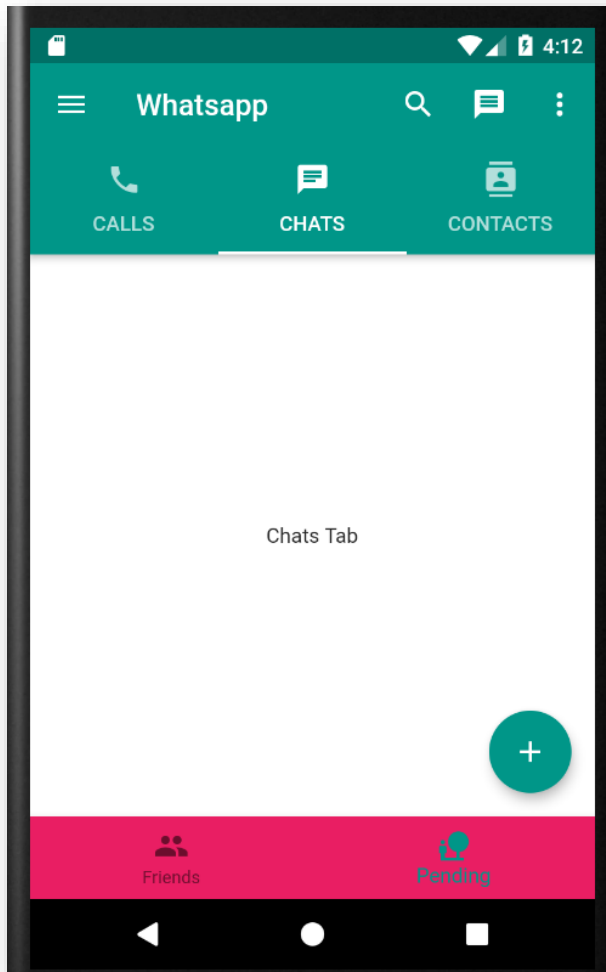
```
15  @override
16  Widget build(BuildContext context) {
17    return DefaultTabController(
18      length: 3,
19      initialIndex: 1,
20      child: Scaffold(
21        appBar: AppBar( // AppBar
22          body: TabBarView(children: [ // TabBarView
23            floatingActionButton: FloatingActionButton(
24              onPressed: _doNothing,
25              child: Icon(Icons.add),
26            ), // FloatingActionButton
27          ), // Scaffold
28        ); // DefaultTabController
29      }
30    }
31  }
```

# bottomNavigationBar (1)



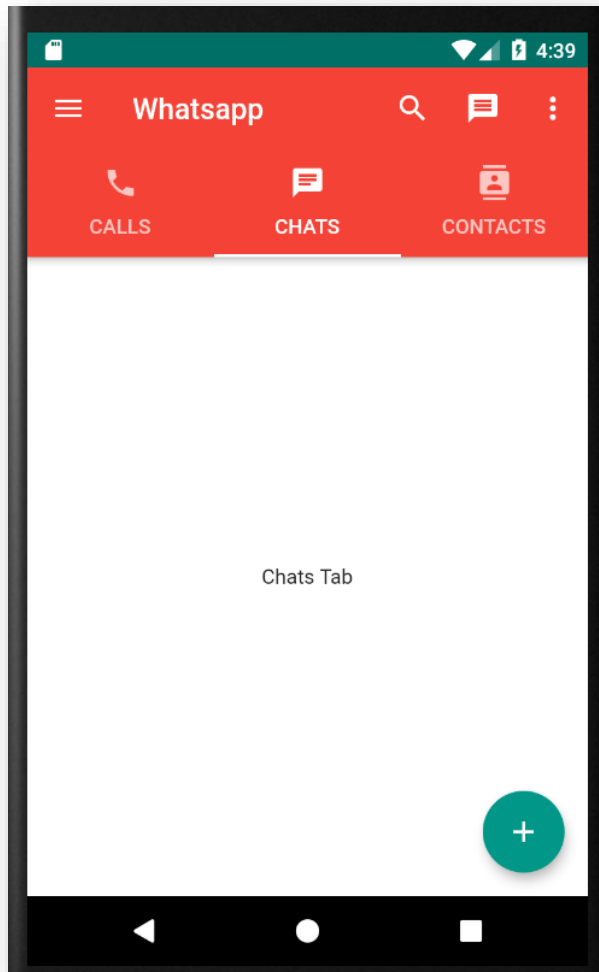
```
38 > |body: TabBarView(children: [ // TabBarView
46 |floatingActionButton: FloatingActionButton(
47 |  |onPressed: _doNothing,
+ 48 |  |child: Icon(Icons.add),
49 |  ), // FloatingActionButton
50 |bottomNavigationBar: TabBar(tabs: [
51 |  |Tab(text: 'CALLS', icon: Icon(Icons.call)),
52 |  |Tab(text: 'CHATS', icon: Icon(Icons.chat)),
53 |  |Tab(text: 'CONTACTS', icon: Icon(Icons.contacts)),
54 |  ], labelColor: Colors.black,)), // TabBar
55
```

# bottomNavigationBar (2)



```
38 > |body: TabBarView(children: [ // TabBarView
46 > |floatingActionButton: FloatingActionButton( // FloatingActionButton
50 |bottomNavigationBar: BottomNavigationBar(
51 |    currentIndex: 1,
52 |    backgroundColor: Colors.pink,
53 |    items: [
54 |        BottomNavigationBarItem(
55 |            icon: Icon(Icons.people),
56 |            title: Text('Friends'),
57 |        ), // BottomNavigationBarItem
58 |        BottomNavigationBarItem(
59 |            icon: Icon(Icons.nature_people),
60 |            title: Text('Pending'),
61 |        ), // BottomNavigationBarItem
62 |    ],
63 |), // BottomNavigationBar
64 |), // Scaffold
65 |); // DefaultTabController
66 |}
67 }
```

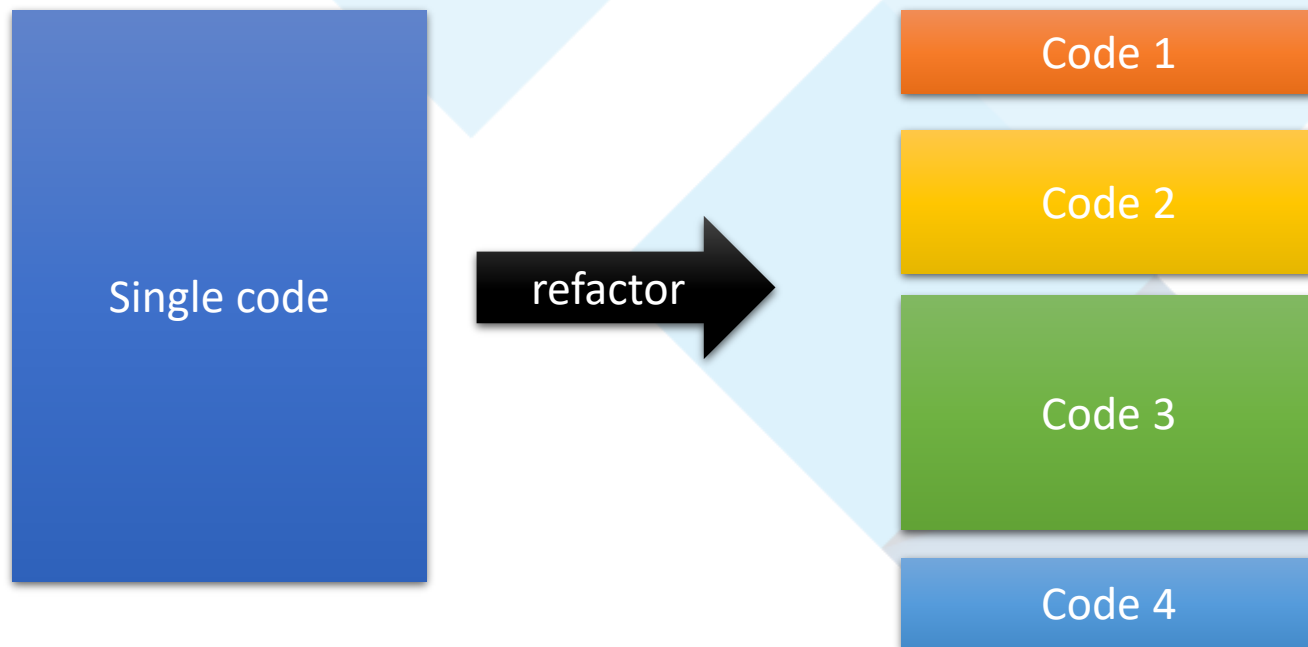
# AppBar's flexibleSpace and SafeArea



```
16   Widget build(BuildContext context) {
17     return DefaultTabController(
18       length: 3,
19       initialIndex: 1,
20       child: Scaffold(
21         appBar: AppBar(
22           leading: IconButton(
23             icon: Icon(Icons.menu),
24             onPressed: _doNothing,
25           ), // IconButton
26           title: Text('Whatsapp'),
27           actions: <Widget>[ // <Widget>[]
32         bottom: TabBar(tabs: [ // TabBar
37         flexibleSpace: SafeArea(
38           child: Container(color: Colors.red),
39         ), // SafeArea
40       ), // AppBar
41       body: TabBarView(children: [ // TabBarView
49       floatingActionButton: FloatingActionButton(
53     ), // Scaffold
54   ); // DefaultTabController
55 }
56 }
57
```

# Refactoring

- As you progress building the UI, the widget tree grows very quickly making the code **less readable**.
- Thus, it is important to **refactor** the code to make it more readable and much easier to work with.
- Basic strategy is by **splitting code** into several pieces of code rather than putting it in a single bulky one.



# How to Refactoring

- Refactor to constants
- Refactor to methods and functions
- Refactor to classes
- Further refactoring: files and directories

# How to Refactoring (1)

## Refactor to constant

## Delegate widget creation to variables

*In VS Code:*  
***Extract Local Variable***

```

20 |   child: Scaffold(
21 |     appBar: AppBar(
22 |       leading: IconButton(
23 |         icon: Icon(Icons.menu),
24 |         onPressed: _doNothing,
25 |       ), // IconButton
26 |       title: Text('Whatsapp'),
27 |       actions: <Widget>[
28 |         IconButton(icon: Icon(Icons.search), onPressed: _doNothing),
29 |         IconButton(icon: Icon(Icons.message), onPressed: _doNothing),
30 |         IconButton(icon: Icon(Icons.more_vert), onPressed: _doNothing),
31 |       ], // <Widget>[]
32 |       bottom: TabBar(tabs: [
33 |         Tab(text: 'CALLS', icon: Icon(Icons.call)),
34 |         Tab(text: 'CHATS', icon: Icon(Icons.chat)),
35 |         Tab(text: 'CONTACTS', icon: Icon(Icons.contacts)),
36 |       ]), // TabBar
37 |     ), // AppBar
38 |     body: TabBarView(children: [
39 |       Container(

```

```

16 Widget build(BuildContext context) {
17   final _appBar = AppBar(
18     | leading: IconButton( // IconButton
22     | title: Text('Whatsapp'),
23     | actions: <Widget>[ // <Widget>[]
28     | bottom: TabBar(tabs: [ // TabBar
33   ); // AppBar

34
35   return DefaultTabController(
36     | length: 3,
37     | initialIndex: 1,
38     | child: Scaffold(
39     | | appBar: _appBar,
40     | | body: TabBarView(children: [ // TabBarView
48     | | floatingActionButton: FloatingActionButton(

```



# How to Refactoring (2)

Refactor to method  
Split code within the class

*In VS Code:*  
***Extract Method***

After refactoring

```
15  @override
16  Widget build(BuildContext context) {
17    return DefaultTabController(
18      | length: 3,
19      | initialIndex: 1,
20      | child: Scaffold(
21        | appBar: _buildAppBar(),
22        | body: TabBarView(children: [ // TabBarView
30        | floatingActionButton: FloatingActionButton( // FloatingActionButton
34        | ), // Scaffold
35      ); // DefaultTabController
36    }
37
38  AppBar _buildAppBar() {
39    return AppBar(
40      | leading: IconButton( // IconButton
44      | title: Text('Whatsapp'),
45      | actions: <Widget>[ // <Widget>[]
50      | bottom: TabBar(tabs: [ // TabBar
55      ); // AppBar
56    }
57  }
```

## How to Refactoring (3)

## Refactor to function

## Split code to outside of class

## *In VS Code:*

### ***Extract Method***

After  
refactoring

```

12 class Home extends StatelessWidget {
13     void _doNothing() {}
14
15     @override
16     Widget build(BuildContext context) {
17         return DefaultTabController(
18             | length: 3,
19             | initialIndex: 1,
20             | child: Scaffold(
21                 | appBar: _buildAppBar(),
22                 | body: TabBarView(children: [ // TabBarView
23                 | floatingActionButton: FloatingActionButton(
24                 | ), // Scaffold
25             ); // DefaultTabController
26         }
27     }
28
29     void _doNothing() {}
30
31     AppBar _buildAppBar() {
32         return AppBar(
33             | leading: IconButton( // IconButton
34             | title: Text('Whatsapp'),
35             | actions: <Widget>[ // <Widget>[]
36             | bottom: TabBar(tabs: [ // TabBar
37             ); // AppBar
38         }
39     }
40
41     void _doNothing() {}
42
43     AppBar _buildAppBar() {
44         return AppBar(
45             | leading: IconButton( // IconButton
46             | title: Text('Whatsapp'),
47             | actions: <Widget>[ // <Widget>[]
48             | bottom: TabBar(tabs: [ // TabBar
49             ); // AppBar
50         }
51     }
52
53     void _doNothing() {}
54
55     AppBar _buildAppBar() {
56         return AppBar(
57             | leading: IconButton( // IconButton
58             | title: Text('Whatsapp'),
59             | actions: <Widget>[ // <Widget>[]
60             | bottom: TabBar(tabs: [ // TabBar
61             ); // AppBar
62         }
63     }
64
65     void _doNothing() {}
66
67     AppBar _buildAppBar() {
68         return AppBar(
69             | leading: IconButton( // IconButton
70             | title: Text('Whatsapp'),
71             | actions: <Widget>[ // <Widget>[]
72             | bottom: TabBar(tabs: [ // TabBar
73             ); // AppBar
74         }
75     }
76
77     void _doNothing() {}
78
79     AppBar _buildAppBar() {
80         return AppBar(
81             | leading: IconButton( // IconButton
82             | title: Text('Whatsapp'),
83             | actions: <Widget>[ // <Widget>[]
84             | bottom: TabBar(tabs: [ // TabBar
85             ); // AppBar
86         }
87     }
88
89     void _doNothing() {}
90
91     AppBar _buildAppBar() {
92         return AppBar(
93             | leading: IconButton( // IconButton
94             | title: Text('Whatsapp'),
95             | actions: <Widget>[ // <Widget>[]
96             | bottom: TabBar(tabs: [ // TabBar
97             ); // AppBar
98         }
99     }
100
101     void _doNothing() {}
102
103     AppBar _buildAppBar() {
104         return AppBar(
105             | leading: IconButton( // IconButton
106             | title: Text('Whatsapp'),
107             | actions: <Widget>[ // <Widget>[]
108             | bottom: TabBar(tabs: [ // TabBar
109             ); // AppBar
110         }
111     }
112
113     void _doNothing() {}
114
115     AppBar _buildAppBar() {
116         return AppBar(
117             | leading: IconButton( // IconButton
118             | title: Text('Whatsapp'),
119             | actions: <Widget>[ // <Widget>[]
120             | bottom: TabBar(tabs: [ // TabBar
121             ); // AppBar
122         }
123     }
124
125     void _doNothing() {}
126
127     AppBar _buildAppBar() {
128         return AppBar(
129             | leading: IconButton( // IconButton
130             | title: Text('Whatsapp'),
131             | actions: <Widget>[ // <Widget>[]
132             | bottom: TabBar(tabs: [ // TabBar
133             ); // AppBar
134         }
135     }
136
137     void _doNothing() {}
138
139     AppBar _buildAppBar() {
140         return AppBar(
141             | leading: IconButton( // IconButton
142             | title: Text('Whatsapp'),
143             | actions: <Widget>[ // <Widget>[]
144             | bottom: TabBar(tabs: [ // TabBar
145             ); // AppBar
146         }
147     }
148
149     void _doNothing() {}
150
151     AppBar _buildAppBar() {
152         return AppBar(
153             | leading: IconButton( // IconButton
154             | title: Text('Whatsapp'),
155             | actions: <Widget>[ // <Widget>[]
156             | bottom: TabBar(tabs: [ // TabBar
157             ); // AppBar
158         }
159     }
160
161     void _doNothing() {}
162
163     AppBar _buildAppBar() {
164         return AppBar(
165             | leading: IconButton( // IconButton
166             | title: Text('Whatsapp'),
167             | actions: <Widget>[ // <Widget>[]
168             | bottom: TabBar(tabs: [ // TabBar
169             ); // AppBar
170         }
171     }
172
173     void _doNothing() {}
174
175     AppBar _buildAppBar() {
176         return AppBar(
177             | leading: IconButton( // IconButton
178             | title: Text('Whatsapp'),
179             | actions: <Widget>[ // <Widget>[]
180             | bottom: TabBar(tabs: [ // TabBar
181             ); // AppBar
182         }
183     }
184
185     void _doNothing() {}
186
187     AppBar _buildAppBar() {
188         return AppBar(
189             | leading: IconButton( // IconButton
190             | title: Text('Whatsapp'),
191             | actions: <Widget>[ // <Widget>[]
192             | bottom: TabBar(tabs: [ // TabBar
193             ); // AppBar
194         }
195     }
196
197     void _doNothing() {}
198
199     AppBar _buildAppBar() {
200         return AppBar(
201             | leading: IconButton( // IconButton
202             | title: Text('Whatsapp'),
203             | actions: <Widget>[ // <Widget>[]
204             | bottom: TabBar(tabs: [ // TabBar
205             ); // AppBar
206         }
207     }
208
209     void _doNothing() {}
210
211     AppBar _buildAppBar() {
212         return AppBar(
213             | leading: IconButton( // IconButton
214             | title: Text('Whatsapp'),
215             | actions: <Widget>[ // <Widget>[]
216             | bottom: TabBar(tabs: [ // TabBar
217             ); // AppBar
218         }
219     }
220
221     void _doNothing() {}
222
223     AppBar _buildAppBar() {
224         return AppBar(
225             | leading: IconButton( // IconButton
226             | title: Text('Whatsapp'),
227             | actions: <Widget>[ // <Widget>[]
228             | bottom: TabBar(tabs: [ // TabBar
229             ); // AppBar
230         }
231     }
232
233     void _doNothing() {}
234
235     AppBar _buildAppBar() {
236         return AppBar(
237             | leading: IconButton( // IconButton
238             | title: Text('Whatsapp'),
239             | actions: <Widget>[ // <Widget>[]
240             | bottom: TabBar(tabs: [ // TabBar
241             ); // AppBar
242         }
243     }
244
245     void _doNothing() {}
246
247     AppBar _buildAppBar() {
248         return AppBar(
249             | leading: IconButton( // IconButton
250             | title: Text('Whatsapp'),
251             | actions: <Widget>[ // <Widget>[]
252             | bottom: TabBar(tabs: [ // TabBar
253             ); // AppBar
254         }
255     }
256
257     void _doNothing() {}
258
259     AppBar _buildAppBar() {
260         return AppBar(
261             | leading: IconButton( // IconButton
262             | title: Text('Whatsapp'),
263             | actions: <Widget>[ // <Widget>[]
264             | bottom: TabBar(tabs: [ // TabBar
265             ); // AppBar
266         }
267     }
268
269     void _doNothing() {}
270
271     AppBar _buildAppBar() {
272         return AppBar(
273             | leading: IconButton( // IconButton
274             | title: Text('Whatsapp'),
275             | actions: <Widget>[ // <Widget>[]
276             | bottom: TabBar(tabs: [ // TabBar
277             ); // AppBar
278         }
279     }
280
281     void _doNothing() {}
282
283     AppBar _buildAppBar() {
284         return AppBar(
285             | leading: IconButton( // IconButton
286             | title: Text('Whatsapp'),
287             | actions: <Widget>[ // <Widget>[]
288             | bottom: TabBar(tabs: [ // TabBar
289             ); // AppBar
290         }
291     }
292
293     void _doNothing() {}
294
295     AppBar _buildAppBar() {
296         return AppBar(
297             | leading: IconButton( // IconButton
298             | title: Text('Whatsapp'),
299             | actions: <Widget>[ // <Widget>[]
300             | bottom: TabBar(tabs: [ // TabBar
301             ); // AppBar
302         }
303     }
304
305     void _doNothing() {}
306
307     AppBar _buildAppBar() {
308         return AppBar(
309             | leading: IconButton( // IconButton
310             | title: Text('Whatsapp'),
311             | actions: <Widget>[ // <Widget>[]
312             | bottom: TabBar(tabs: [ // TabBar
313             ); // AppBar
314         }
315     }
316
317     void _doNothing() {}
318
319     AppBar _buildAppBar() {
320         return AppBar(
321             | leading: IconButton( // IconButton
322             | title: Text('Whatsapp'),
323             | actions: <Widget>[ // <Widget>[]
324             | bottom: TabBar(tabs: [ // TabBar
325             ); // AppBar
326         }
327     }
328
329     void _doNothing() {}
330
331     AppBar _buildAppBar() {
332         return AppBar(
333             | leading: IconButton( // IconButton
334             | title: Text('Whatsapp'),
335             | actions: <Widget>[ // <Widget>[]
336             | bottom: TabBar(tabs: [ // TabBar
337             ); // AppBar
338         }
339     }
340
341     void _doNothing() {}
342
343     AppBar _buildAppBar() {
344         return AppBar(
345             | leading: IconButton( // IconButton
346             | title: Text('Whatsapp'),
347             | actions: <Widget>[ // <Widget>[]
348             | bottom: TabBar(tabs: [ // TabBar
349             ); // AppBar
350         }
351     }
352
353     void _doNothing() {}
354
355     AppBar _buildAppBar() {
356         return AppBar(
357             | leading: IconButton( // IconButton
358             | title: Text('Whatsapp'),
359             | actions: <Widget>[ // <Widget>[]
360             | bottom: TabBar(tabs: [ // TabBar
361             ); // AppBar
362         }
363     }
364
365     void _doNothing() {}
366
367     AppBar _buildAppBar() {
368         return AppBar(
369             | leading: IconButton( // IconButton
370             | title: Text('Whatsapp'),
371             | actions: <Widget>[ // <Widget>[]
372             | bottom: TabBar(tabs: [ // TabBar
373             ); // AppBar
374         }
375     }
376
377     void _doNothing() {}
378
379     AppBar _buildAppBar() {
380         return AppBar(
381             | leading: IconButton( // IconButton
382             | title: Text('Whatsapp'),
383             | actions: <Widget>[ // <Widget>[]
384             | bottom: TabBar(tabs: [ // TabBar
385             ); // AppBar
386         }
387     }
388
389     void _doNothing() {}
390
391     AppBar _buildAppBar() {
392         return AppBar(
393             | leading: IconButton( // IconButton
394             | title: Text('Whatsapp'),
395             | actions: <Widget>[ // <Widget>[]
396             | bottom: TabBar(tabs: [ // TabBar
397             ); // AppBar
398         }
399     }
400
401     void _doNothing() {}
402
403     AppBar _buildAppBar() {
404         return AppBar(
405             | leading: IconButton( // IconButton
406             | title: Text('Whatsapp'),
407             | actions: <Widget>[ // <Widget>[]
408             | bottom: TabBar(tabs: [ // TabBar
409             ); // AppBar
410         }
411     }
412
413     void _doNothing() {}
414
415     AppBar _buildAppBar() {
416         return AppBar(
417             | leading: IconButton( // IconButton
418             | title: Text('Whatsapp'),
419             | actions: <Widget>[ // <Widget>[]
42
```

# How to Refactoring (4)

## Refactor to class

Split code to different classes.  
It is recommended approach

*In VS Code:*

***Extract Widget***

After refactoring

```
12 class Home extends StatelessWidget {
13   void _doNothing() {}
14   @override
15   Widget build(BuildContext context) {
16     return DefaultTabController(
17       length: 3,
18       initialIndex: 1,
19       child: Scaffold(
20         appBar: _AppBar(),
21         body: TabBarView(children: [ // TabBarView
22           floatingActionButton: FloatingActionButton(
23             ), // Scaffold
24         ); // DefaultTabController
25   }
26 }
27
28 class _AppBar extends StatelessWidget
29   implements PreferredSizeWidget {
30   void _doNothing() {}
31   const _AppBar({ Key key, }) : super(key: key);
32   @override
33   Size get preferredSize => Size.fromHeight(100);
34
35   @override
36   Widget build(BuildContext context) {
37     return AppBar(
38       leading: IconButton( // IconButton
39         title: Text('Whatsapp*'),
40         actions: <Widget>[ // <Widget>[]
41         bottom: TabBar(tabs: [ // TabBar
42         ); // AppBar
43   }
44 }
```

# How to Refactoring (5)

- Further refactoring to separate files
- It is a good practice to use dedicated files and directory structure
  - Easy to maintain
  - Easy to scale
  - Better for teamwork
- Example project directory structure:

```
[project_root_dir]
|
+---[assets]
|   + ---[images]
|       |           + ---photo.jpg
|       |           + ---background.png
|       |
|       + ---[fonts]
|
+---[backends]
|   + ---[firebase_functions]
|   + ---[nodejs_code]
|   + ---[php_code]
|
+---[lib]
|   + ---[models]
|       |           + ---user.dart
|       |           + ---chat.dart
|       |           + ---call.dart
|   + ---[services]
|   + ---[widgets]
|   + ---[screens]
|       |           + ---[home]
|       |               |           + ---index.dart
|       |               |           + ---app_bar.dart
|       |               |           + ---[tabs]
|       |               |               |           + ---chat_tab_view.dart
|       |               |               |           + ---call_tab_view.dart
|       |               |               |           + ---contact_tab_view.dart
|       |               |
|       |               + ---[login]
|       |               + ---[splash]
```

# Refactoring Our Code (1)

Here, we are using  
Refactoring to functions

After  
refactoring

```
12 class Home extends StatelessWidget {  
13   @override  
14   Widget build(BuildContext context) {  
15     return DefaultTabController(  
16       length: 3,  
17       initialIndex: 1,  
18       child: Scaffold(  
19         appBar: _buildAppBar(),  
20         body: _buildTabBarView(),  
21         floatingActionButton: _buildFloatingActionButton(),  
22       ), // Scaffold  
23     ); // DefaultTabController  
24   }  
25 } // class Home  
26
```

# Refactoring Our Code (2)

## Refactoring AppBar

```
26 |  
27 | void _doNothing() {}  
28 |  
29 | AppBar _buildAppBar() {  
30 |   return AppBar(  
31 |     leading: IconButton(  
32 |       icon: Icon(Icons.menu),  
33 |       onPressed: _doNothing,  
34 |     ), // IconButton  
35 |     title: Text('Whatsapp'),  
36 |     actions: <Widget>[  
37 |       IconButton(icon: Icon(Icons.search), onPressed: _doNothing),  
38 |       IconButton(icon: Icon(Icons.message), onPressed: _doNothing),  
39 |       IconButton(icon: Icon(Icons.more_vert), onPressed: _doNothing),  
40 |     ], // <Widget>[]  
41 |     bottom: TabBar(tabs: [  
42 |       Tab(text: 'CALLS', icon: Icon(Icons.call)),  
43 |       Tab(text: 'CHATS', icon: Icon(Icons.chat)),  
44 |       Tab(text: 'CONTACTS', icon: Icon(Icons.contacts)),  
45 |     ]), // TabBar  
46 |   ); // AppBar  
47 | }  
48 |
```

# Refactoring Our Code (3)

## Refactoring TabBarView and FloatingActionButton

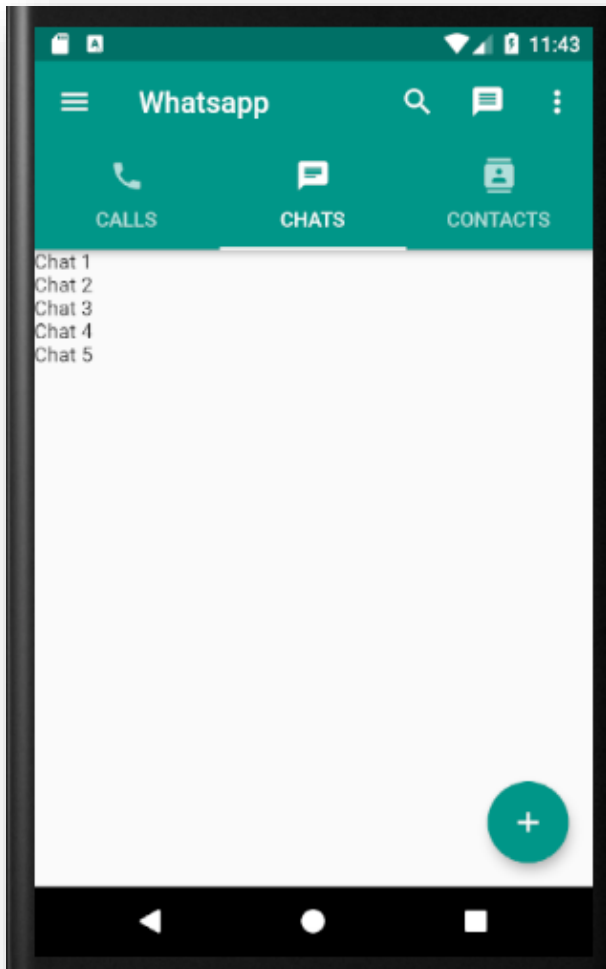
```
49 TabBarView _buildTabBarView() {  
50   return TabBarView(children: [  
51     Container(color: Colors.pink, child: Center(child: Text('Calls Tab'))),  
52     Container(color: Colors.white, child: Center(child: Text('Chats Tab'))),  
53     Container(color: Colors.amber, child: Center(child: Text('Contact Tab'))),  
54   ]); // TabBarView  
55 }  
56  
57 FloatingActionButton _buildFloatingActionButton() {  
58   return FloatingActionButton(  
59     onPressed: _doNothing,  
60     child: Icon(Icons.add),  
61   ); // FloatingActionButton  
62 }
```

# Several ways to add ListView

- ListView (....)
- ListView with collection-for
- ListView.builder (....)



# Adding ListView – Approach 1, ListView()

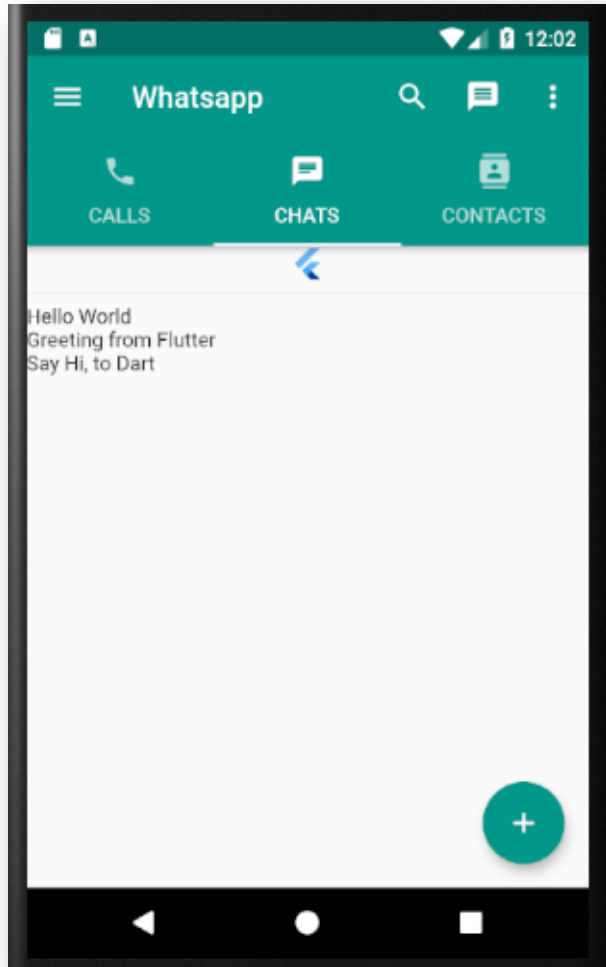


Add to CHATS Tab, in function  
`_buildTabBarView()`

```
49 TabBarView _buildTabBarView() {  
50   return TabBarView(children: [  
51     Container(color: Colors.pink, child: Center(child: Text('Calls Tab'))),  
52     _ChatsTabView(),  
53     Container(color: Colors.amber, child: Center(child: Text('Contact Tab'))),  
54   ]);  
55 }  
56  
57 class _ChatsTabView extends StatelessWidget {  
58   @override  
59   Widget build(BuildContext context) {  
60     return ListView(  
61       children: <Widget>[  
62         Text('Chat 1'),  
63         Text('Chat 2'),  
64         Text('Chat 3'),  
65         Text('Chat 4'),  
66         Text('Chat 5'),  
67       ],  
68     );  
69   }  
70 }  
71
```

This approach is suitable for list with known size

# Adding ListView – Approach 2, with collection-for

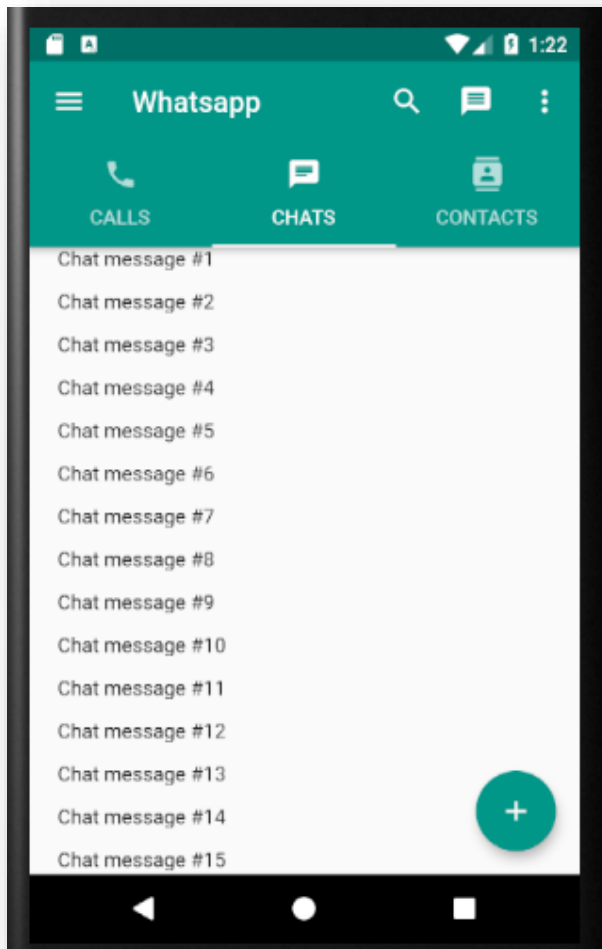


This approach can be used for dynamic list

```
49 TabBarView _buildTabBarView() {
50   return TabBarView(children: [
51     Container(color: Colors.pink, child: Center(child: Text('Calls Tab'))),
52     _ChatsTabView(
53       ['Hello World', 'Greeting from Flutter', 'Say Hi, to Dart'],
54     ),
55     Container(color: Colors.amber, child: Center(child: Text('Contact Tab'))),
56   ]);
57 }
58
59 class _ChatsTabView extends StatelessWidget {
60   final List items;
61
62   _ChatsTabView(this.items);
63
64   @override
65   Widget build(BuildContext context) {
66     return ListView(
67       children: <Widget>[
68         FlutterLogo(),
69         Divider(),
70         for (var item in items) Text(item),    The for, if, and spread elements v
71       ],
72     );
73   }
74 }
75
```

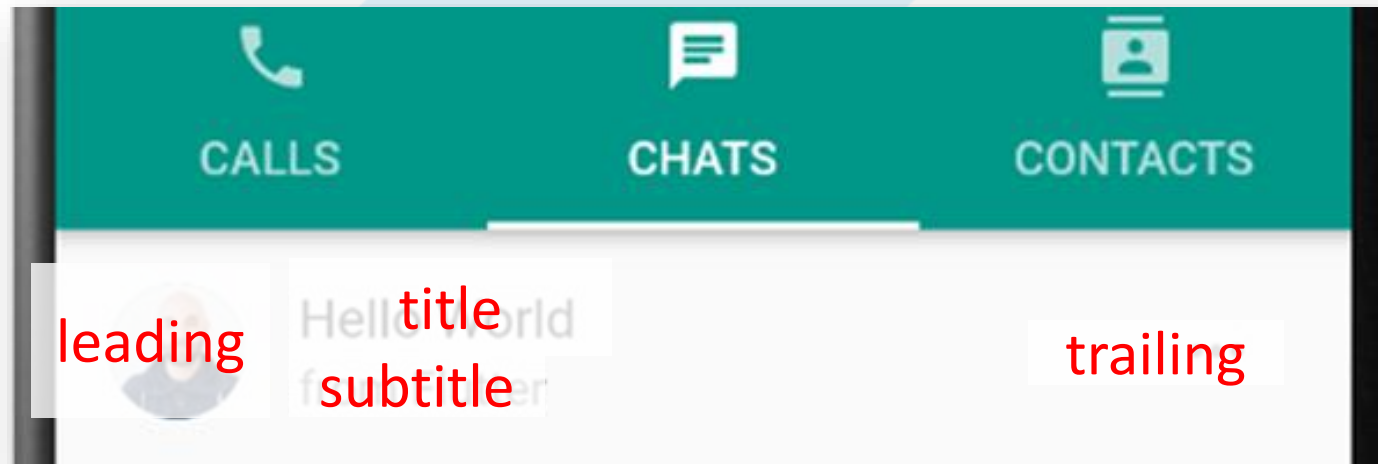
# Adding ListView – Approach 3, ListView.builder()

- Another approach for dynamic list.
- It is recommended approach.
- Suitable for large list
- Removing `itemCount` will create an infinite list

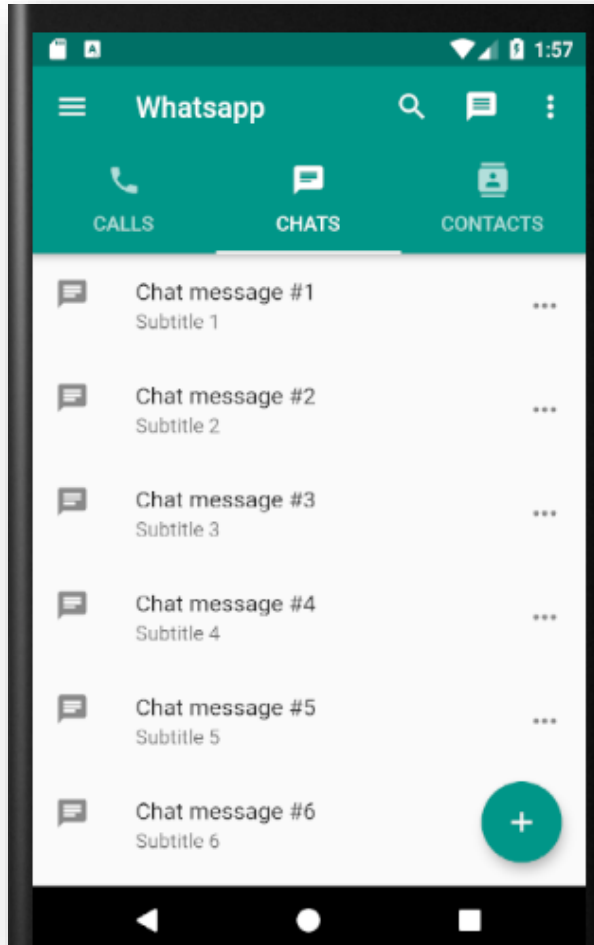


```
49  TabBarView _buildTabBarView() {
50      final chats = List<String>.generate(100, (i) => 'Chat message #${i + 1}');
51
52      return TabBarView(children: [
53          Container(color: Colors.pink, child: Center(child: Text('Calls Tab'))),
54          _ChatsTabView(chats),
55          Container(color: Colors.amber, child: Center(child: Text('Contact Tab'))),
56      ]);
57  }
58
59  class _ChatsTabView extends StatelessWidget {
60      final List items;
61
62      _ChatsTabView(this.items);
63
64      @override
65      Widget build(BuildContext context) {
66          return ListView.builder(
67              itemExtent: 30,
68              padding: EdgeInsets.only(left: 20.0),
69              itemCount: items.length,
70              itemBuilder: (context, index) => Text(items[index]),
71          );
72      }
73  }
```

# Adding ListTile, the components



# Adding ListTile



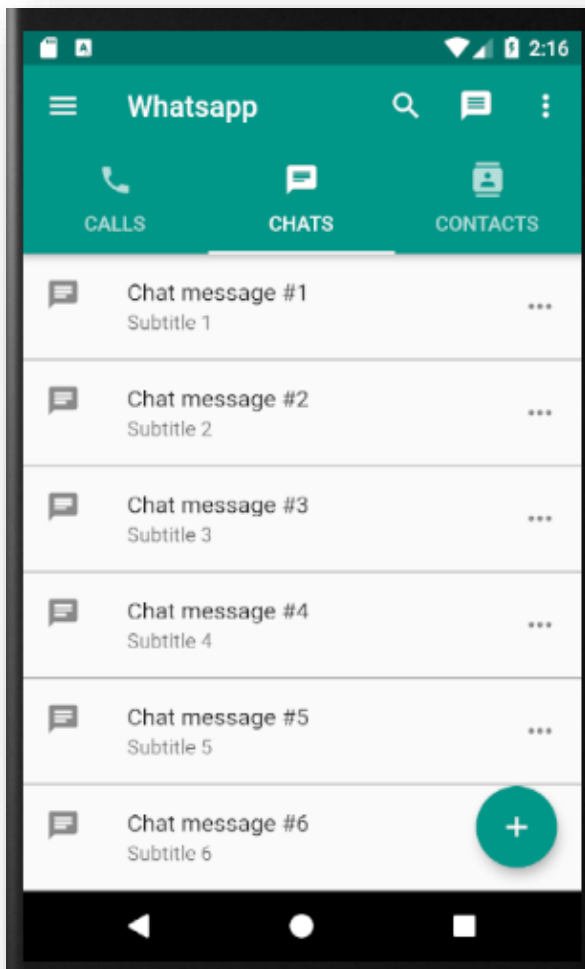
```
49 TabBarView _buildTabBarView() {
50   final chats = List<String>.generate(10, (i) => 'Chat message #${i + 1}');
51
52   return TabBarView(children: [
53     Container(color: Colors.pink, child: Center(child: Text('Calls Tab'))),
54     _ChatsTabView(chats),
55     Container(color: Colors.amber, child: Center(child: Text('Contact Tab'))),
56   ]);
57 }
58
59 class _ChatsTabView extends StatelessWidget {
60   final List items;
61
62   _ChatsTabView(this.items);
63
64   @override
65   Widget build(BuildContext context) {
66     return ListView.builder(
67       itemCount: items.length,
68       itemBuilder: (context, index) => ListTile(
69         leading: Icon(Icons.chat),
70         title: Text(items[index]),
71         subtitle: Text('Subtitle ${index + 1}'),
72         trailing: Icon(Icons.more_horiz),
73       ),
74     );
75   }
76 }
```

# Several ways to add separators

- Wrapped with Container widget and specify decoration
- Wrapped with Card widgets
- ListTile.divideTiles()
- ListView.separated (....)

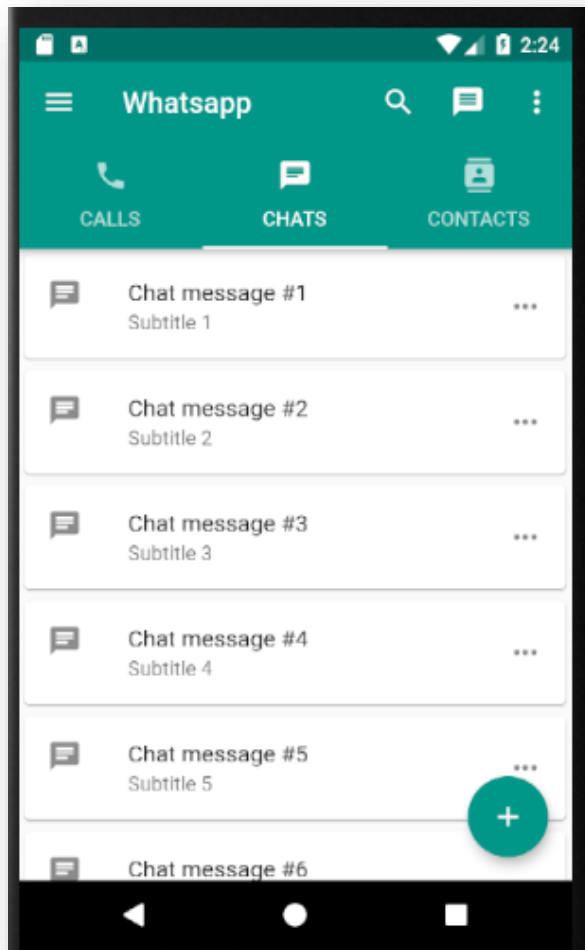
# Adding Separator – Approach 1, wrapped with Container

Decorate the bottom border of the container from the attribute decoration



```
59 class _ChatsTabView extends StatelessWidget {
60   final List items;
61
62   _ChatsTabView(this.items);
63
64   @override
65   Widget build(BuildContext context) {
66     return ListView.builder(
67       itemCount: items.length,
68       itemBuilder: (context, index) => Container(
69         decoration: BoxDecoration(
70           border: Border(
71             bottom: BorderSide(color: Colors.grey, style: BorderStyle.solid),
72           ),
73       ),
74       child: ListTile(
75         leading: Icon(Icons.chat),
76         title: Text(items[index]),
77         subtitle: Text('Subtitle ${index + 1}'),
78         trailing: Icon(Icons.more_horiz),
79       ),
80     ),
81   );
82 }
83 }
```

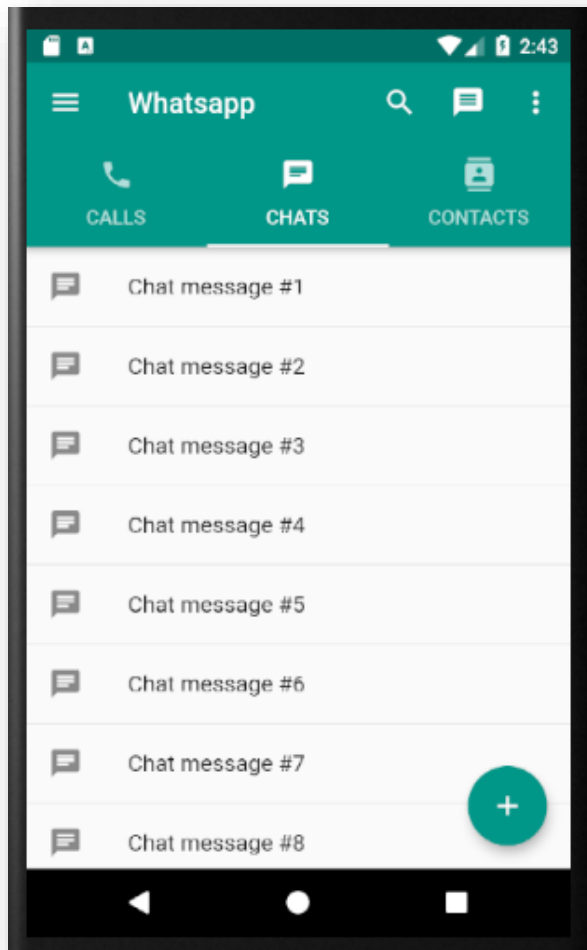
# Adding Separator – Approach 2, wrapped with Card



```
59 class _ChatsTabView extends StatelessWidget {  
60   final List items;  
61  
62   _ChatsTabView(this.items);  
63  
64   @override  
65   Widget build(BuildContext context) {  
66     return ListView.builder(  
67       itemCount: items.length,  
68       itemBuilder: (context, index) => Card(  
69         child: ListTile(  
70           leading: Icon(Icons.chat),  
71           title: Text(items[index]),  
72           subtitle: Text('Subtitle ${index + 1}'),  
73           trailing: Icon(Icons.more_horiz),  
74         ),  
75       ),  
76     );  
77   }  
78 }
```



# Adding Separator – Approach 3, ListTile.divideTiles()

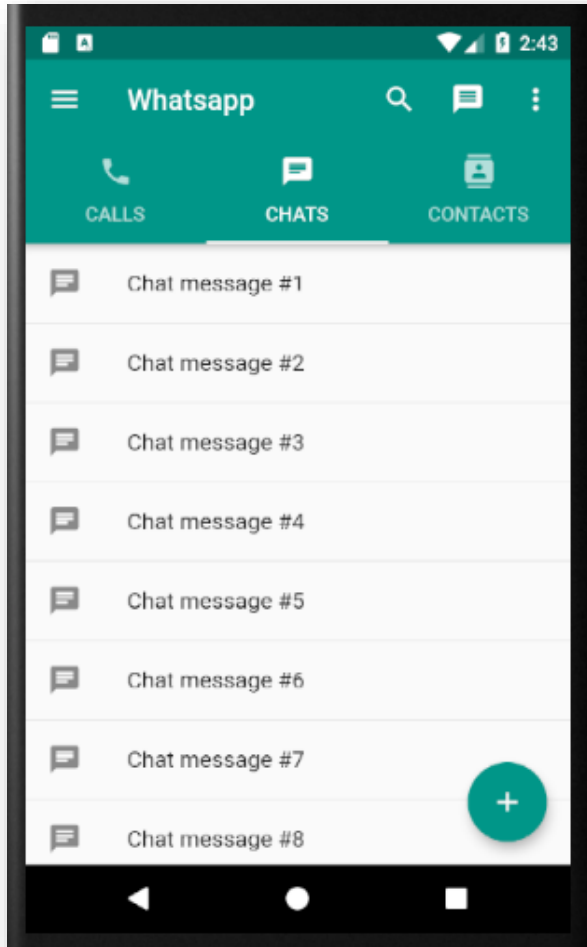


For static list, used with `ListView()` constructor

```
59 class _ChatsTabView extends StatelessWidget {  
60   final List items;  
61  
62   _ChatsTabView(this.items);  
63  
64   @override  
65   Widget build(BuildContext context) {  
66     return ListView(  
67       children: ListTile.divideTiles(  
68         context: context,  
69         tiles: [  
70           for (var item in items) The for, if, and spread elements wer  
71             ListTile(  
72               leading: Icon(Icons.chat),  
73               title: Text(item),  
74             ),  
75           ],  
76         ).toList(),  
77     );  
78   }  
79 }
```

# Adding Separator – Approach 3, ListTile.divideTiles()

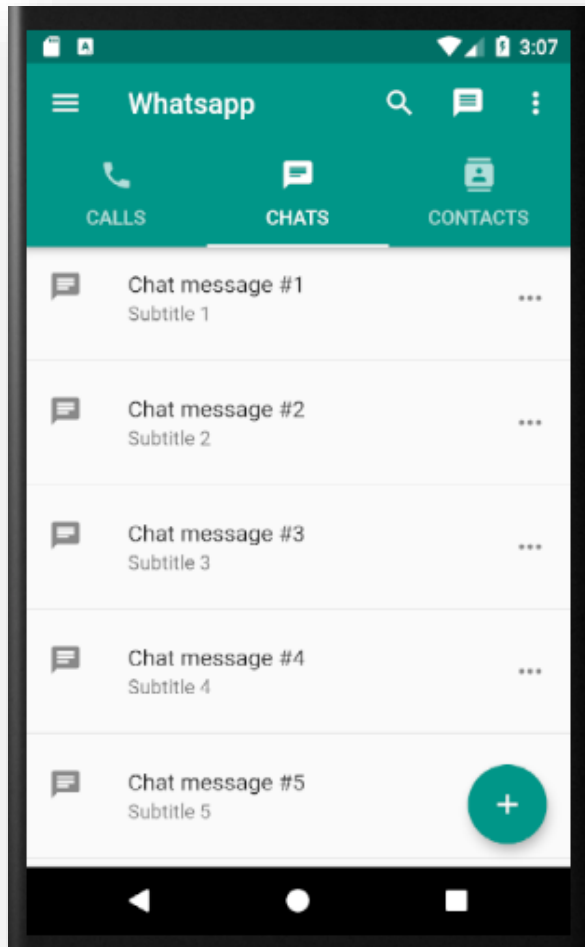
This example is the same as in the previous slide, except this time high order method **map** is used instead of collection for



```
59 class _ChatsTabView extends StatelessWidget {  
60   final List items;  
61  
62   _ChatsTabView(this.items);  
63  
64   @override  
65   Widget build(BuildContext context) {  
66     return ListView(  
67       children: ListTile.divideTiles(  
68         context: context,  
69         tiles: items.map<Widget>(  
70           (item) => ListTile(  
71             leading: Icon(Icons.chat),  
72             title: Text(item),  
73           ),  
74         ),  
75       ).toList(),  
76     );  
77   }  
78 }
```

# Adding Separator – Approach 4, ListView.separated()

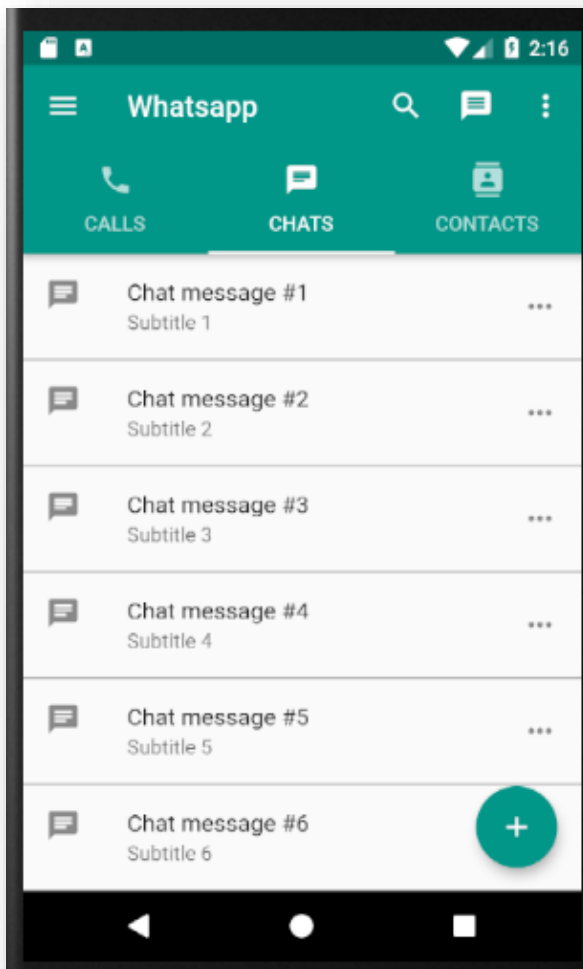
Has dedicated builder (callback) to build separator



```
59 class _ChatsTabView extends StatelessWidget {  
60   final List items;  
61  
62   _ChatsTabView(this.items);  
63  
64   @override  
65   Widget build(BuildContext context) {  
66     return ListView.separated(  
67       itemCount: items.length,  
68       separatorBuilder: (context, index) => Divider(),  
69       itemBuilder: (context, index) => ListTile(  
70         leading: Icon(Icons.chat),  
71         title: Text(items[index]),  
72         subtitle: Text('Subtitle ${index + 1}'),  
73         trailing: Icon(Icons.more_horiz),  
74       ),  
75     );  
76   }  
77 }
```

# Adding SnackBar

- In this example, a snakebar will pop up when a chat message is long pressed
- Snackbar is called from a Scaffold



```
59 class _ChatsTabView extends StatelessWidget {  
60   final List items;  
61  
62   _ChatsTabView(this.items);  
63  
64   @override  
65   Widget build(BuildContext context) {  
66     return ListView.separated(  
67       itemCount: items.length,  
68       separatorBuilder: (context, index) => Divider(),  
69       itemBuilder: (context, index) => ListTile(  
70         leading: Icon(Icons.chat),  
71         title: Text(items[index]),  
72         subtitle: Text('Subtitle ${index + 1}'),  
73         trailing: Icon(Icons.more_horiz),  
74         onLongPress: () => Scaffold.of(context).showSnackBar(SnackBar(  
75           duration: Duration(seconds: 5),  
76           backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
77           content: Text('Item ${items[index]} has been long pressed'),  
78         )),  
79       ),  
80     );  
81   }  
82 }
```

# Drawer

- Adding drawers: drawer and endDrawer
- Open drawer programmatically

# Adding Drawer (1)

Add to Scaffold, in class Home

```
12 class Home extends StatelessWidget {  
13   @override  
14   Widget build(BuildContext context) {  
15     return DefaultTabController(  
16       length: 3,  
17       initialIndex: 1,  
18       child: Scaffold(  
19         appBar: _buildAppBar(),  
20         body: _buildTabBarView(),  
21         floatingActionButton: _buildFloatingActionButton(),  
22         drawer: _buildDrawer(),  
23         endDrawer: _buildEndDrawer(),  
24       ),  
25     );  
26   }  
27 } // class Home
```

- Drawer is part of Scaffold.
- Drawer is specified with the attributes **drawer** and **endDrawer**.
- To add the drawer menu on AppBar automatically, **do not specify** the attribute leading and actions.
- If leading and actions are specified manually, the drawers will need to be opened programmatically.
- In the next example we are going to add a drawer that will open without needing explicit code.

## Adding Drawer (2)

By not specifying the `leading` attribute, it means

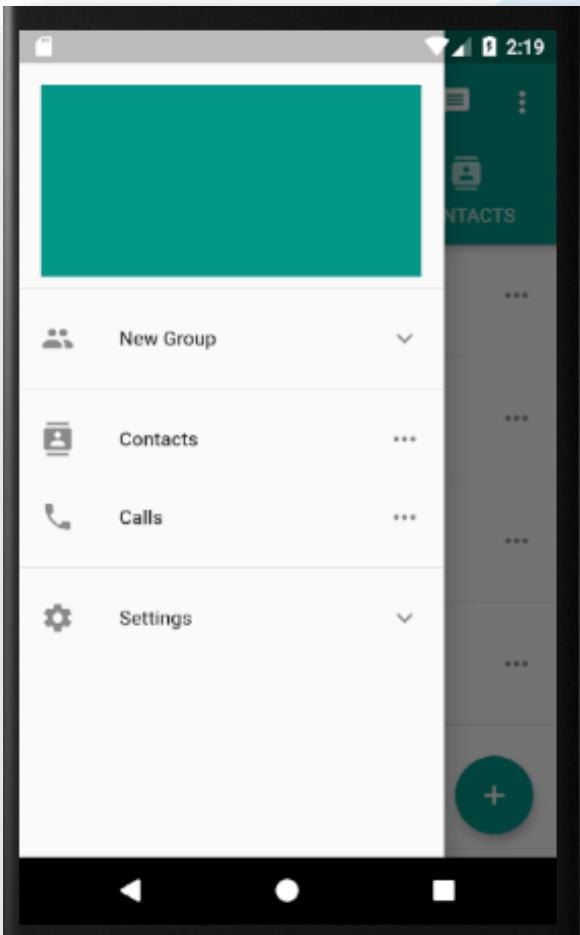
- Icon button for the drawer on the AppBar will be added automatically
- No explicit code for opening the drawer. Default code will be used

Unset the `leading` attribute of AppBar

```
35  AppBar _buildAppBar() {  
36    return AppBar(  
37      // leading: IconButton(  
38        //   icon: Icon(Icons.menu),  
39        //   onPressed: _doNothing,  
40        // ),  
41      title: Text('Whatsapp'),  
42      > actions: <Widget>[ ...  
47      > bottom: TabBar(tabs: [ ...  
52    );  
53  }
```

# Adding Drawer (3)

- Drawer can be built with any widget.
- Common use case, using ListView widget with DrawerHeader on top followed by ListTile



## Build the drawer with Drawer widget

```
29 Drawer _buildDrawer() {
30   return Drawer(
31     child: ListView(
32       children: <Widget>[
33         DrawerHeader(
34           child: Container(color: Colors.teal),
35         ),
36         ListTile(
37           leading: Icon(Icons.group),
38           title: Text('New Group'),
39           trailing: Icon(Icons.expand_more),
40         ),
41         Divider(),
42         ListTile(
43           leading: Icon(Icons.contacts),
44           title: Text('Contacts'),
45           trailing: Icon(Icons.more_horiz),
46         ),
47         ListTile(
48           leading: Icon(Icons.call),
49           title: Text('Calls'),
50           trailing: Icon(Icons.more_horiz),
51         ),
52         Divider(),
53         ListTile(
54           leading: Icon(Icons.settings),
55           title: Text('Settings'),
56           trailing: Icon(Icons.expand_more),
57         ),
58       ],
59     ),
60   );
61 }
```



# Open Drawer Programmatically (1)

By specifying the `leading` or `actions` attribute of `AppBar`, it means

- Custom Icon button will be used for the drawer.
- The drawer will need to be open programmatically.

In the next example, we are going to open the `endDrawer` programmatically

Note that drawers are under Scaffold, thus we need to open the drawer from Scaffold

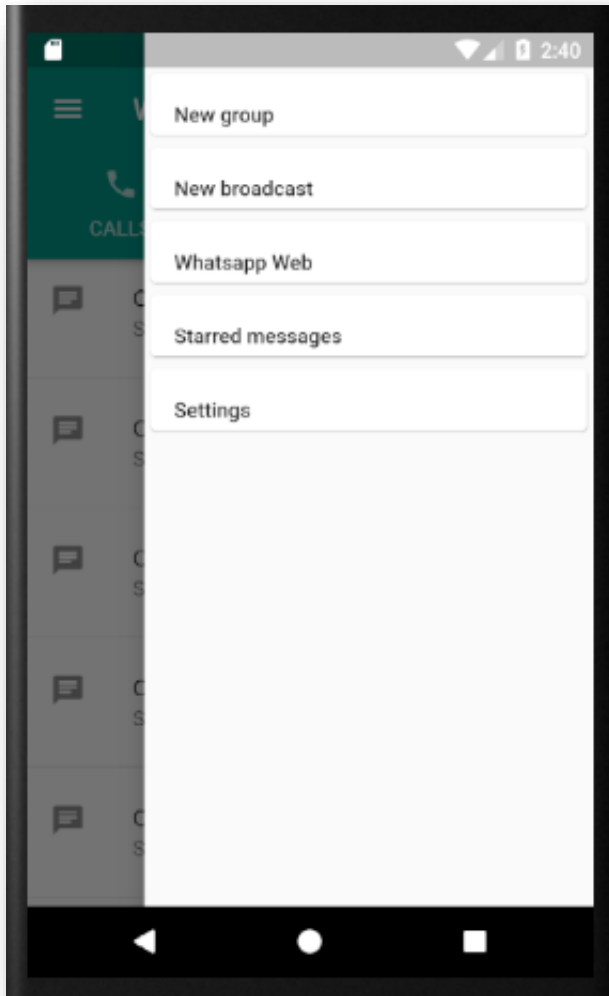
Specify a callback for `onTap` of the last action button of `AppBar`

```
84  AppBar _buildAppBar() {
85    return AppBar(
86      // leading: IconButton(
87      //   icon: Icon(Icons.menu),
88      //   onPressed: _doNothing,
89      // ),
90      title: Text('Whatsapp'),
91      actions: <Widget>[
92        IconButton(icon: Icon(Icons.search), onPressed: _doNothing),
93        IconButton(icon: Icon(Icons.message), onPressed: _doNothing),
94        _AppBarActionMoreButton(),
95      ],
96      bottom: TabBar(tabs: [
97        Tab(text: 'CALLS', icon: Icon(Icons.call)),
98        Tab(text: 'CHATS', icon: Icon(Icons.chat)),
99        Tab(text: 'CONTACTS', icon: Icon(Icons.contacts)),
100      ]),
101    );
102  }
```

```
104  class _AppBarActionMoreButton extends StatelessWidget {
105    @override
106    Widget build(BuildContext context) {
107      return IconButton(
108        icon: Icon(Icons.more_vert),
109        onPressed: () => Scaffold.of(context).openEndDrawer();
110      );
111    }
112  }
```

# Open Drawer Programmatically (2)

Build the drawer with Drawer widget



```
63  Drawer _buildEndDrawer() {
64    const menu = const <String>[
65      'New group',
66      'New broadcast',
67      'Whatsapp Web',
68      'Starred messages',
69      'Settings'
70    ];
71
72    return Drawer(
73      child: ListView.builder(
74        itemExtent: 50,
75        itemCount: menu.length,
76        itemBuilder: (context, index) =>
77          Card(child: ListTile(title: Text(menu[index]))),
78      ),
79    );
80  }
```

```
12  class Home extends StatelessWidget {
13    @override
14    Widget build(BuildContext context) {
15      return DefaultTabController(
16        length: 3,
17        initialIndex: 1,
18        child: Scaffold(
19          appBar: _buildAppBar(),
20          body: _buildTabBarView(),
21          floatingActionButton: _buildFloatingActionButton(),
22          drawer: _buildDrawer(),
23          endDrawer: _buildEndDrawer(),
24        ),
25      );
26    }
27  } // class Home
```

# Summary

- Different ways to create flutter projects
- Introduction to Widgets
- Common widgets
- Refactoring