



Develop Mobile App with Flutter







Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

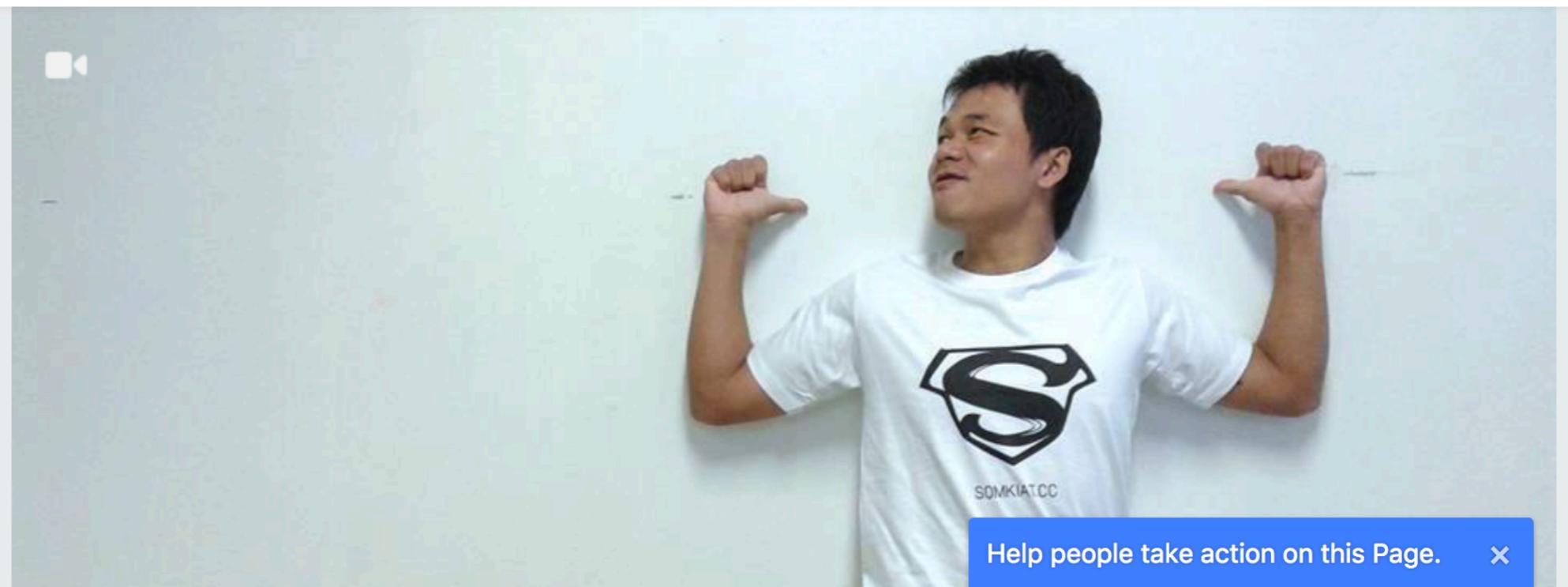
@somkiat.cc

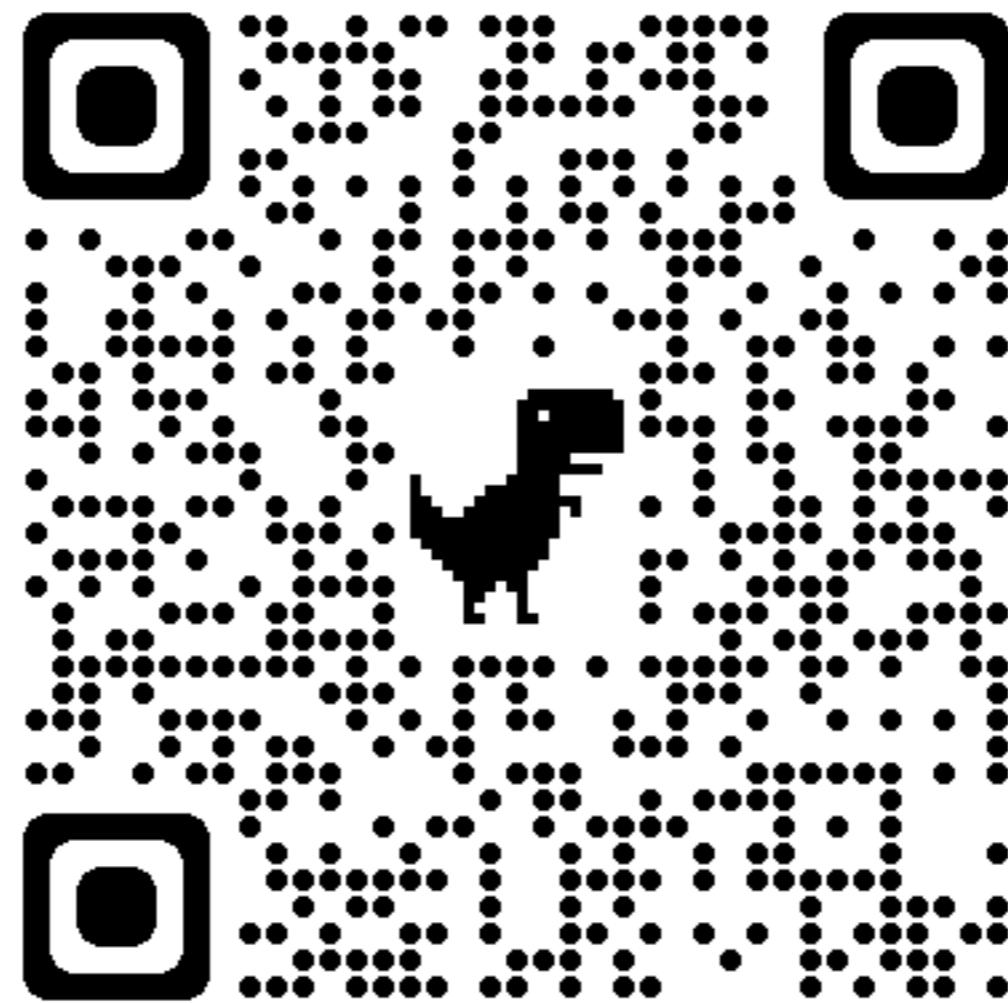
Home

Posts

Videos

Photos





<https://github.com/up1/course-flutter-101>



Develop Mobile App



<https://flutter.dev/>

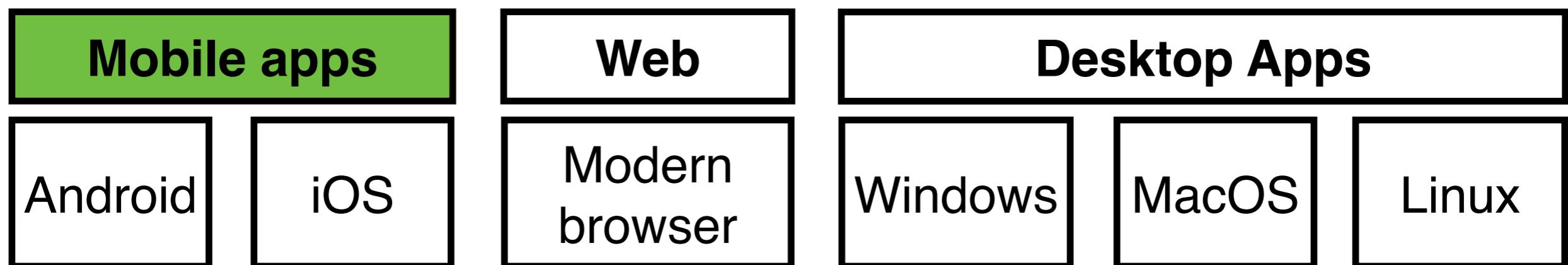


What is Flutter ?

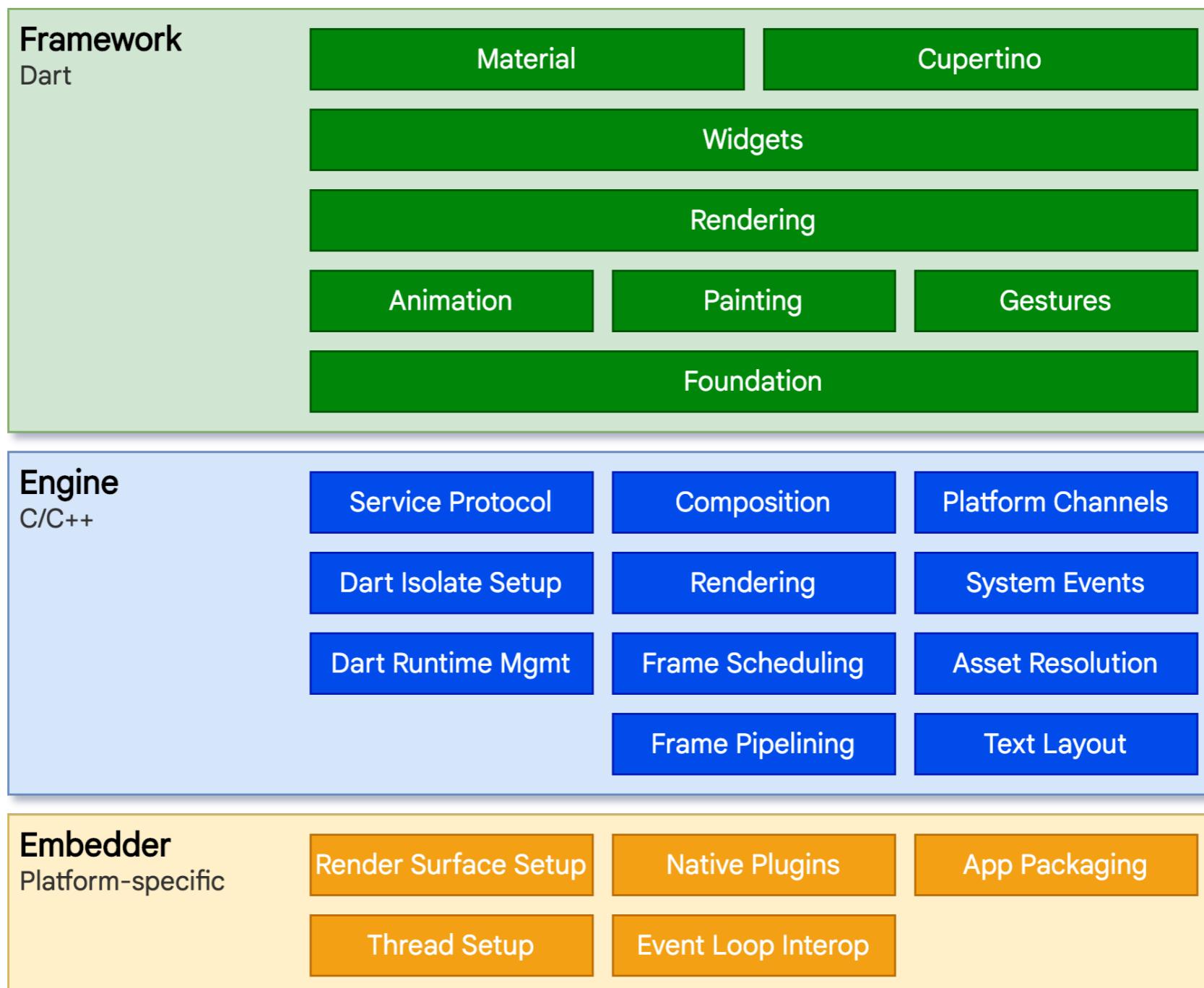
UI framework developed with **Dart** programming

Write code for cross platform

One codebase, Multiple Apps



Architecture of Flutter



<https://docs.flutter.dev/resources/architectural-overview>



Dart

The screenshot shows the official Dart website. At the top, there's a dark header with the Dart logo, navigation links for Overview, Docs, Community, Try Dart, Get Dart, and a search icon. A banner at the top of the main content area encourages users to check out the Dart 3.2 blog post, noting enhancements to type promotion, interop capabilities, DevTools, and more. The central visual features a smartphone displaying a blue UI interface with geometric shapes like triangles and squares. To the left of the phone is a code snippet in Dart:

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Column(
      children: <Widget>[
        HeroImage(),
        ...todaysDiscounts,
        for (var d in items) ItemWidget(d),
      ],
    );
}
```

To the right of the phone, large white text reads "Paint your UI to life" with the subtitle "with Dart VM's instant **hot reload**". The background of the main content area is a dark blue gradient.

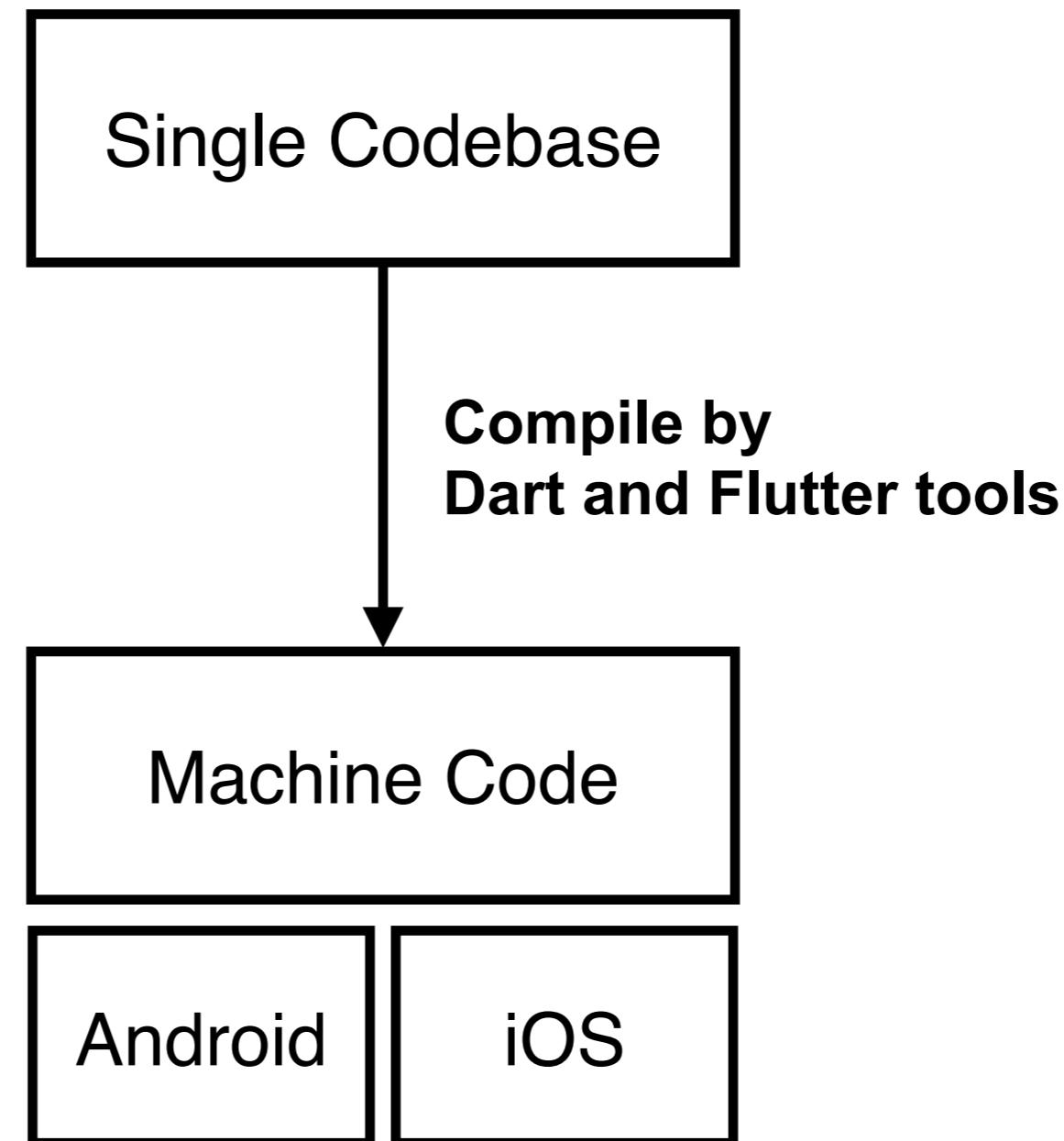
<https://dart.dev/>



Flutter 101

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Flutter to platform-specific machine code



What is Flutter ?

Provide collection of tools

CLIs

Develop

Testing

Build



Software Requirements

Android

iOS

JAVA

Xcode

Android SDK

CocoaPods

Flutter and Dart



Install Flutter and Dart

\$flutter doctor

```
Doctor summary (to see all details, run flutter doctor --verbose)
[✓] Flutter (Channel stable, 3.16.3, on macOS 12.1 21C52 darwin-arm)
    • Flutter version 3.16.3 at /Users/username/development/flutter
    • Framework revision 2a933edf0d (3 weeks ago) on 2023-07-31 18:32:52.807 +0200
    • Engine revision 6375f4d4d1
    • Tools: Dart 2.17.6, VS Code (version 1.85.1)

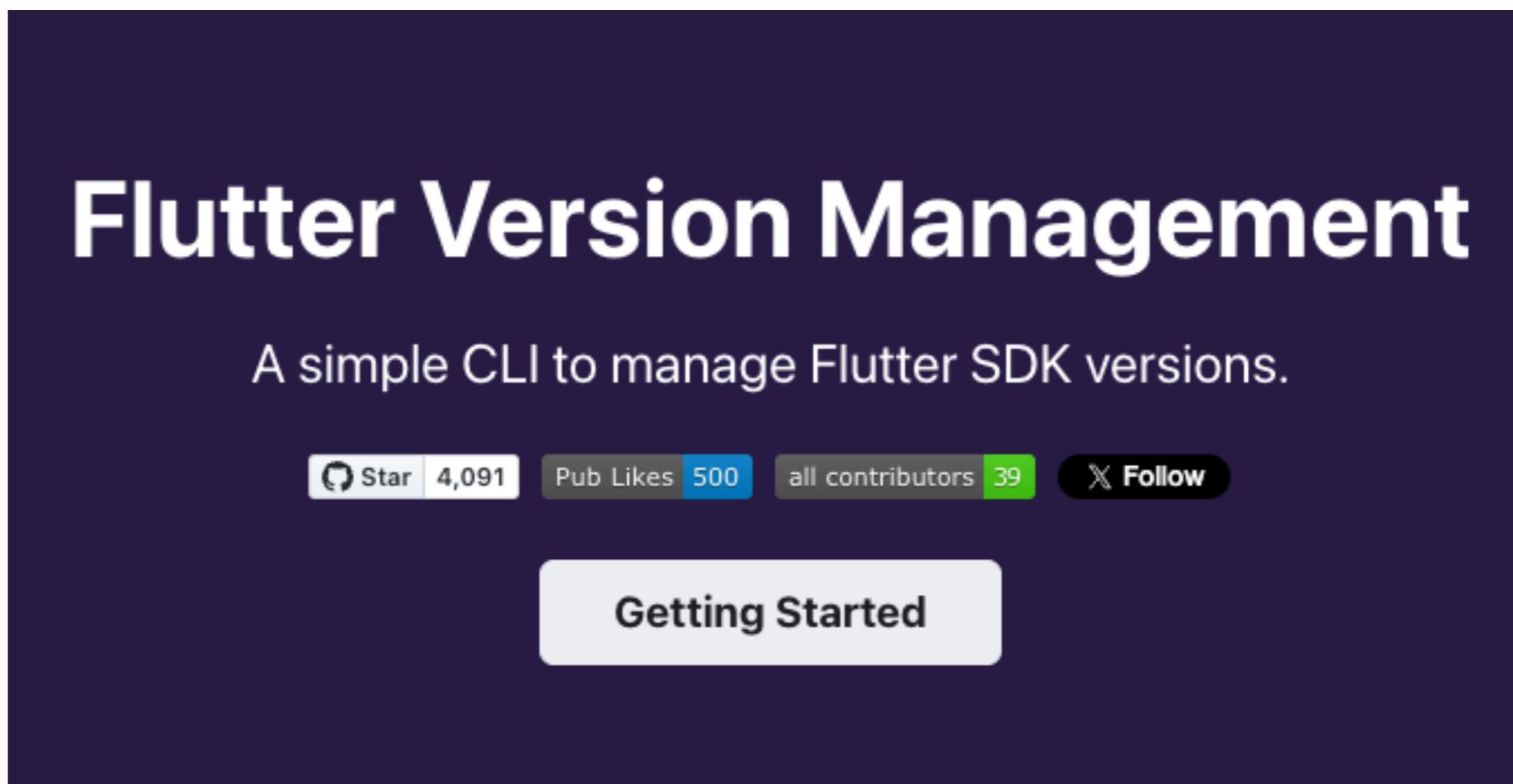
[✓] Android toolchain - develop for Android devices (Android SDK r33.0.1-7779505)
[✓] Xcode - develop for iOS and macOS (Xcode 14.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2022.3)
[✓] IntelliJ IDEA Community Edition (version 2022.3)
[✓] VS Code (version 1.85.1)
[✓] Connected device (2 available)
[✓] Network resources
```

<https://docs.flutter.dev/get-started/install>



Flutter Version Management

CLI to manage Flutter SDK versions



<https://fvm.app/>



Android Studio

JAVA and Android SDK

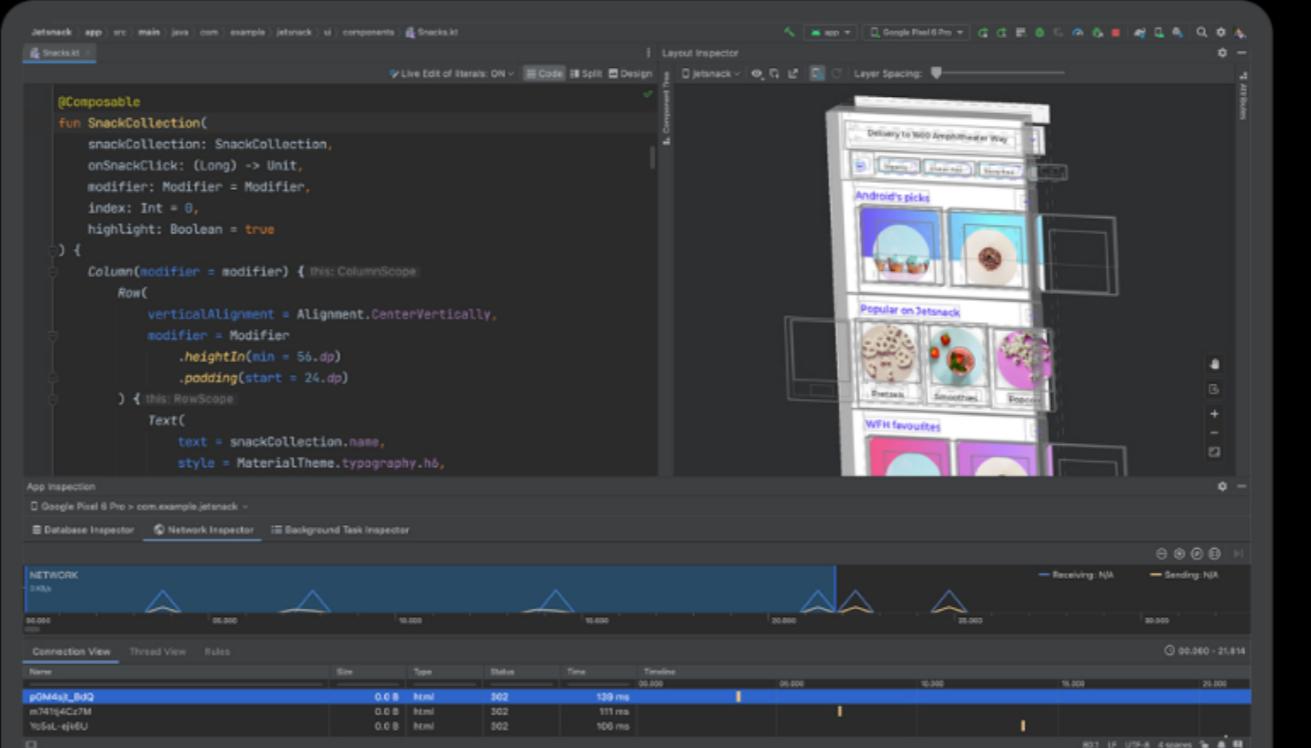
ANDROID STUDIO

Download Android Studio editor Android Gradle Plugin SDK tools Preview

Android Studio
Get the official Integrated Development Environment (IDE) for Android app development.

Download Android Studio Hedgehog ↴

Read release notes ↴



The screenshot shows the Android Studio interface. On the left, there's a large promotional image for Android Studio. On the right, the IDE window is open, displaying a Java file named SnackCollection.kt. The code defines a Composable function that creates a Column with a Row inside. The Row has vertical alignment set to center vertically, height in at least 56 dp, and padding start at 24 dp. It also contains a Text component with the name of the snack collection and its style. To the right of the code editor is the Layout Inspector, which shows a preview of an Android application screen for a Google Pixel 6 Pro. The screen displays a grid of snack items with labels like "Android's picks", "Popular on Jetpack", and "WFH favourites". Below the code editor is the Network Inspector, showing a timeline of network requests. One request from "IPDA492_R92" is highlighted, showing a size of 0.6 B, type of html, status of 302, and a duration of 139 ms.

<https://developer.android.com/studio>



Flutter 101

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Code Editor with VSCode

The screenshot shows the official Visual Studio Code website and a screenshot of the VS Code interface.

Visual Studio Code Website:

- Header: Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, Learn, Search Docs, Download.
- Middle: Version 1.85 is now available! Read about the new features and fixes from November.
- Section: Code editing. Redefined. Free. Built on open source. Runs everywhere.
- Buttons: Download Mac Universal (Stable Build), Web, Insiders edition, or other platforms.
- Text: By using VS Code, you agree to its license and privacy statement.

VS Code Interface Screenshot:

- Left sidebar: EXTENSIONS: MARKETPLACE showing extensions like Python, GitLens, C/C++, ESLint, Debugger for C#, Language Support, vscode-icons, and Vetur.
- Code editor: JS blog-post.js file open, showing code related to Gatsby and GraphQL.
- Bottom status bar: master, 0 0 0, Gatsby Develop (gatsby-graphql-app), info: [wdm]: Compiling..., DONE Compiled successfully in 26ms, 3:57:58 PM, Ln 6, Col 21, Spaces: 2, UTF-8, LF, JavaScript.



IntelliSense



Run and Debug



Built-in Git



Extensions

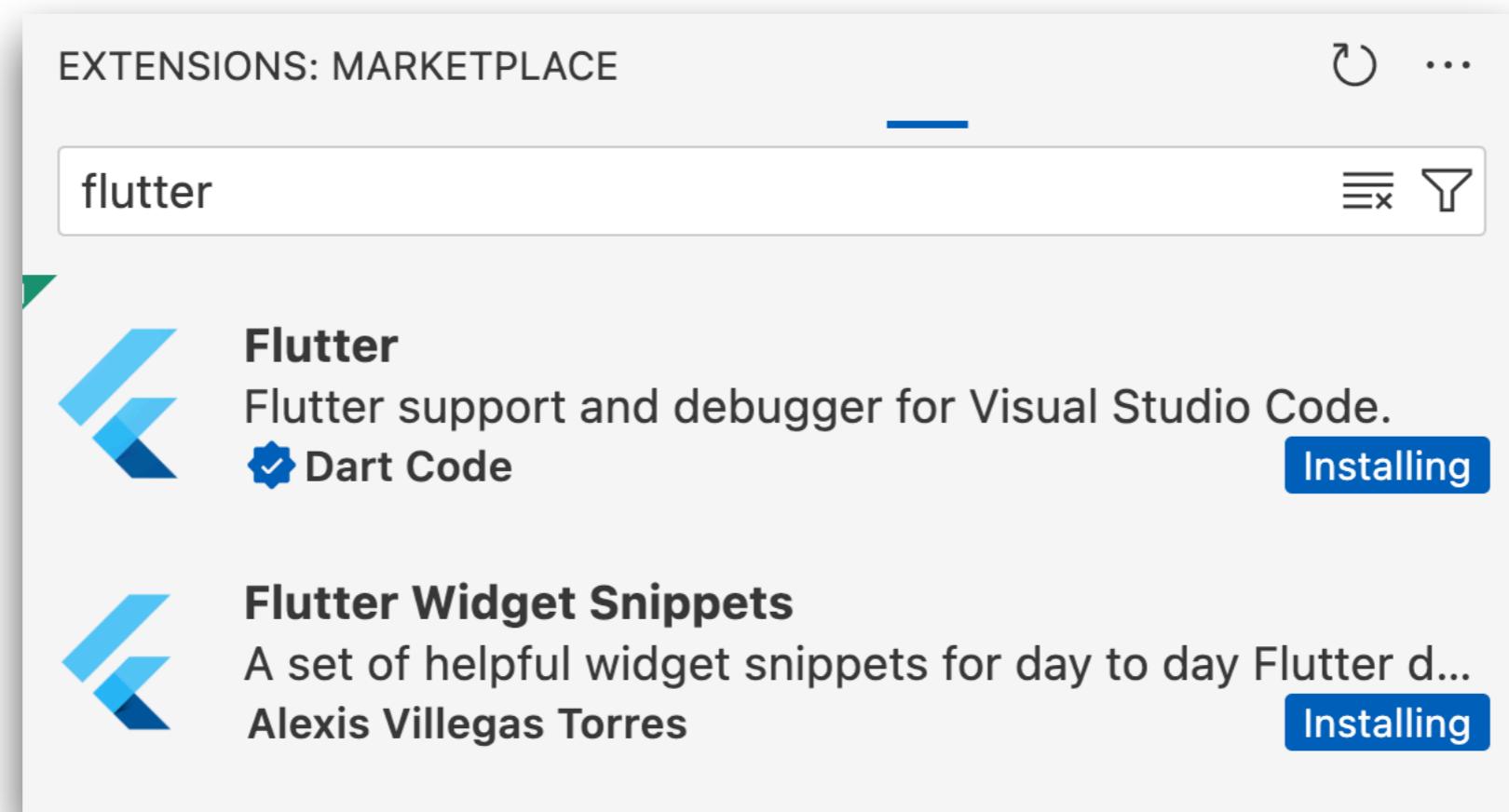
<https://code.visualstudio.com/>



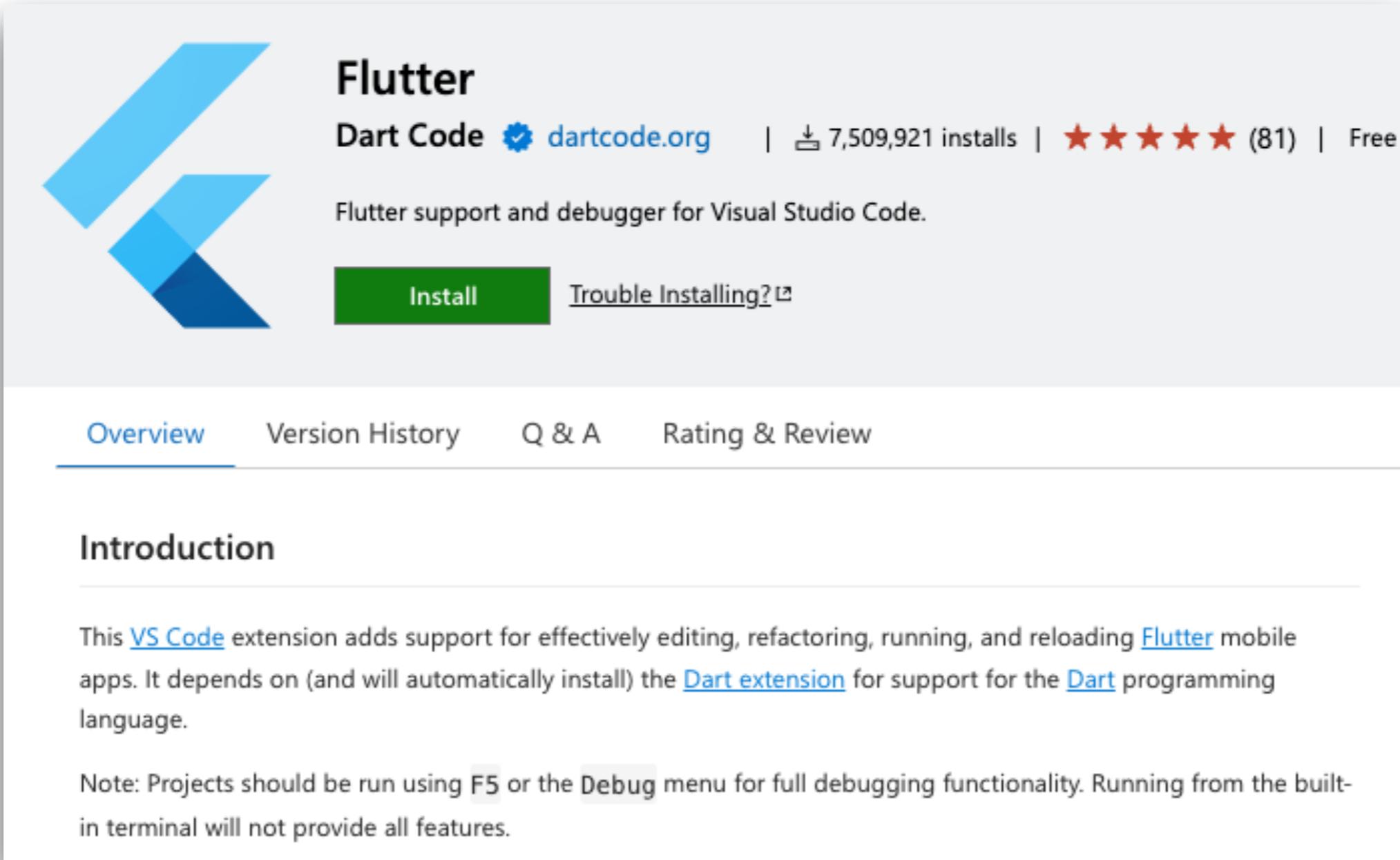
Flutter 101

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

VSCode Flutter Extension



VSCode Flutter Extension



The screenshot shows the 'Flutter' extension page on the Visual Studio Code Marketplace. The page features the Flutter logo (a blue 'F' icon) and the title 'Flutter'. Below the title, it says 'Dart Code' with a blue checkmark icon, a link to 'dartcode.org', and statistics showing '7,509,921 installs' and a rating of '★★★★★ (81) | Free'. A description below states 'Flutter support and debugger for Visual Studio Code.' There are two buttons: a green 'Install' button and a link 'Trouble Installing?'. Below this, there's a navigation bar with tabs: 'Overview' (which is active and underlined), 'Version History', 'Q & A', and 'Rating & Review'. The main content area is titled 'Introduction' and contains text about the extension's purpose and dependencies. It also includes a note about running projects.

This [VS Code](#) extension adds support for effectively editing, refactoring, running, and reloading [Flutter](#) mobile apps. It depends on (and will automatically install) the [Dart extension](#) for support for the [Dart](#) programming language.

Note: Projects should be run using F5 or the Debug menu for full debugging functionality. Running from the built-in terminal will not provide all features.

<https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter>



Flutter Cookbook

Cookbook

This cookbook contains recipes that demonstrate how to solve common problems while writing self-contained and can be used as a reference to help you build up an application.

Animation

- Animate a page route transition
- Animate a widget using a physics simulation
- Animate the properties of a container
- Fade a widget in and out

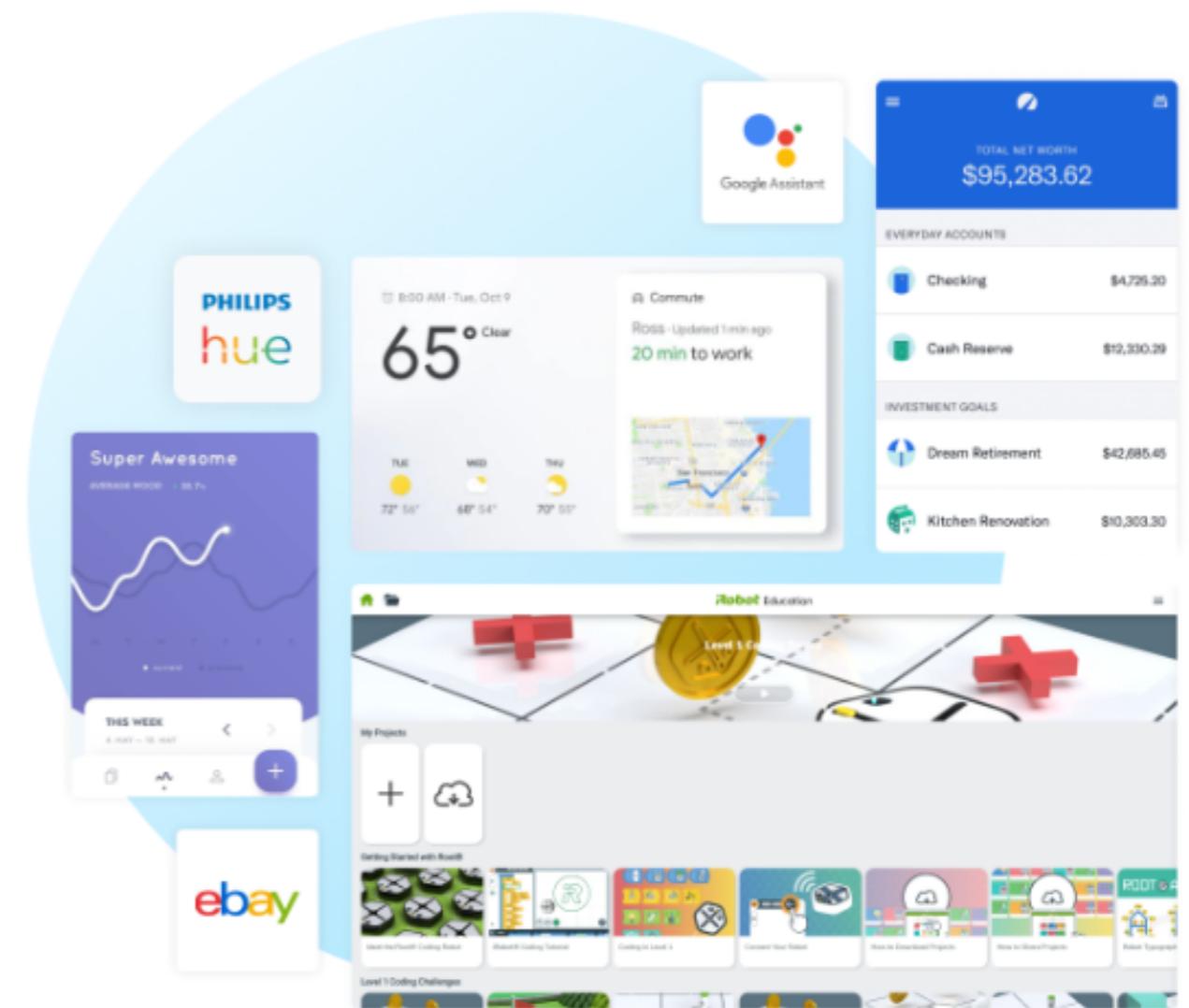
<https://docs.flutter.dev/cookbook>



Flutter Showcases

Flutter apps in production

Businesses of all sizes around the world are building with Flutter



<https://flutter.dev/showcase>



Flutter 101

© 2017 - 2018 Siam Chamnkit Company Limited. All rights reserved.

Awesome Flutter

Curate flutter libraries, tools , tutorials and articles



<https://github.com/Solido/awesome-flutter>



Develop and Running First App



Create a new project

```
$flutter create hello_app
```

Connect Mobile devices and Emulator !!!



Android Devices

\$adb devices

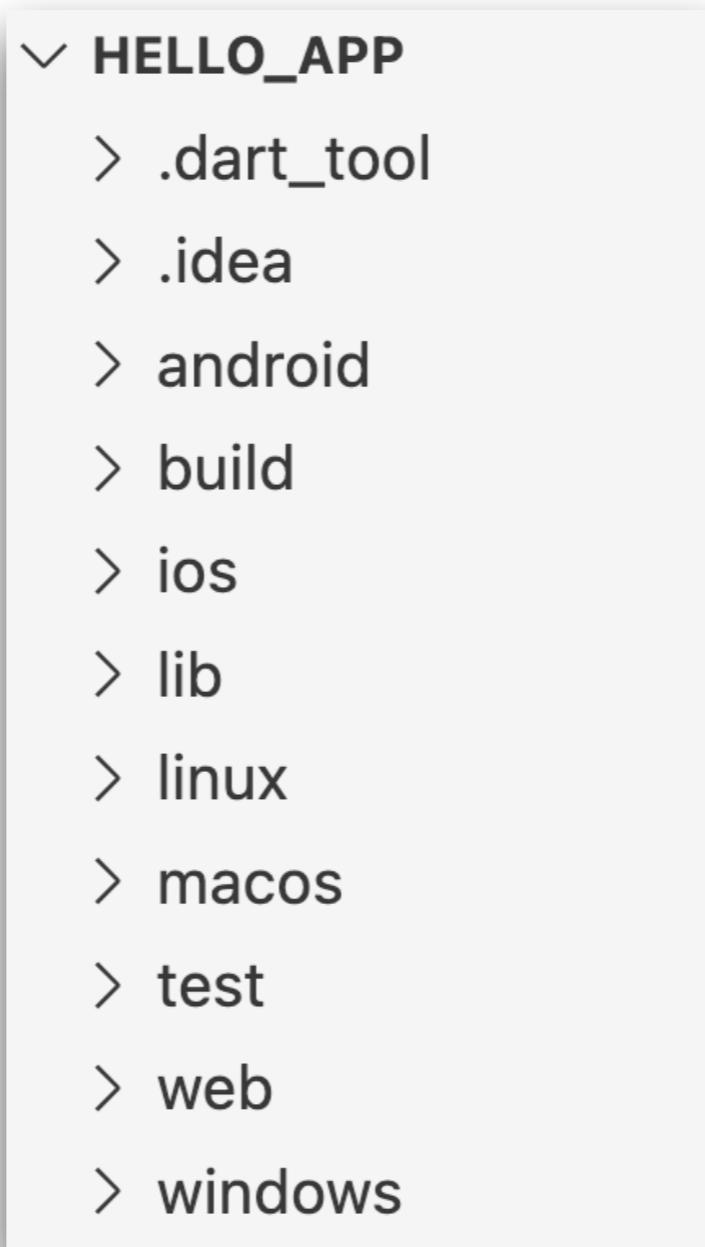


iOS Devices

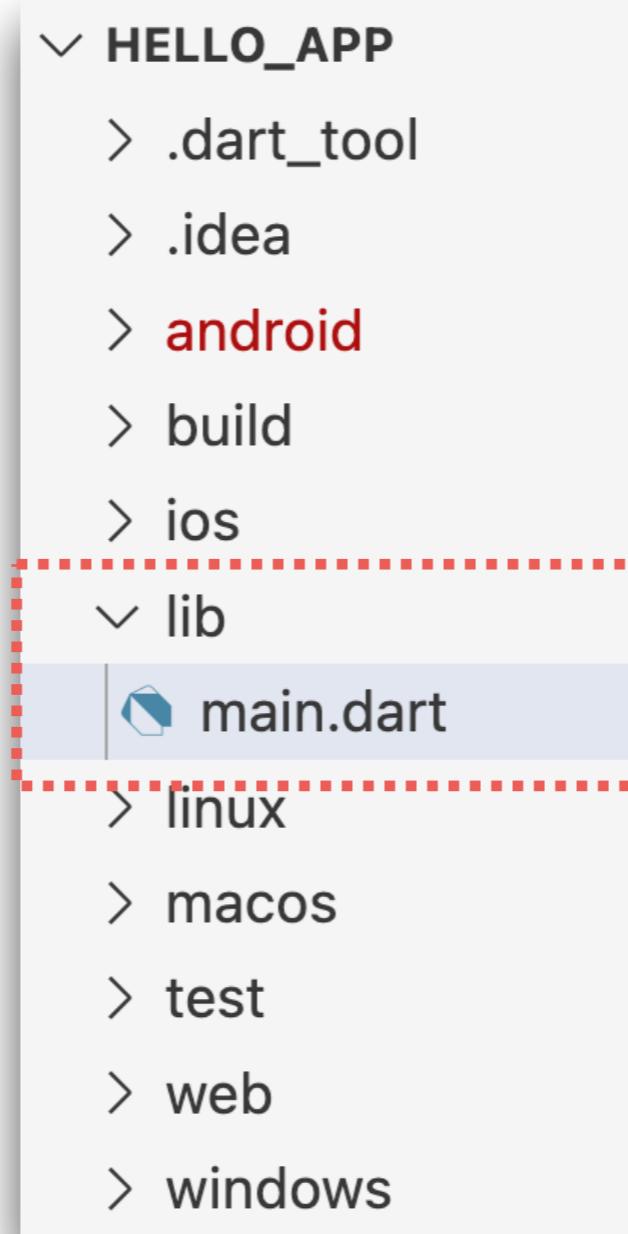
```
$xcrun xctrace list devices
```



Open Project in VS Code



Let's start



Running project

\$flutter run

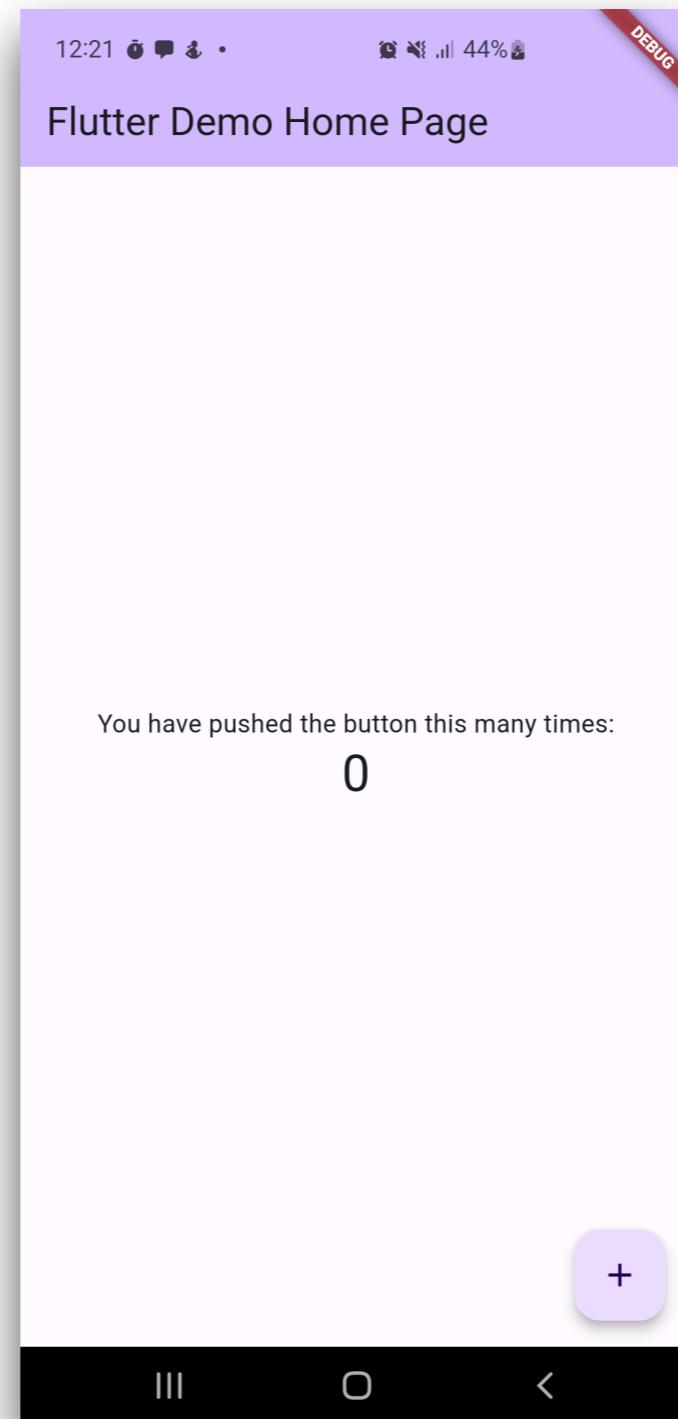
```
Launching lib/main.dart on SM G975F in debug mode...
Running Gradle task 'assembleDebug'...
✓ Built build/app/outputs/flutter-apk/app-debug.apk.
Installing build/app/outputs/flutter-apk/app-debug.apk...
Syncing files to device SM G975F...

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).
```

Run with Android app



Hello Flutter



Building APP

\$flutter build apk

\$flutter build ipa

Android app

iOS app



Utilities CLIs from Dart

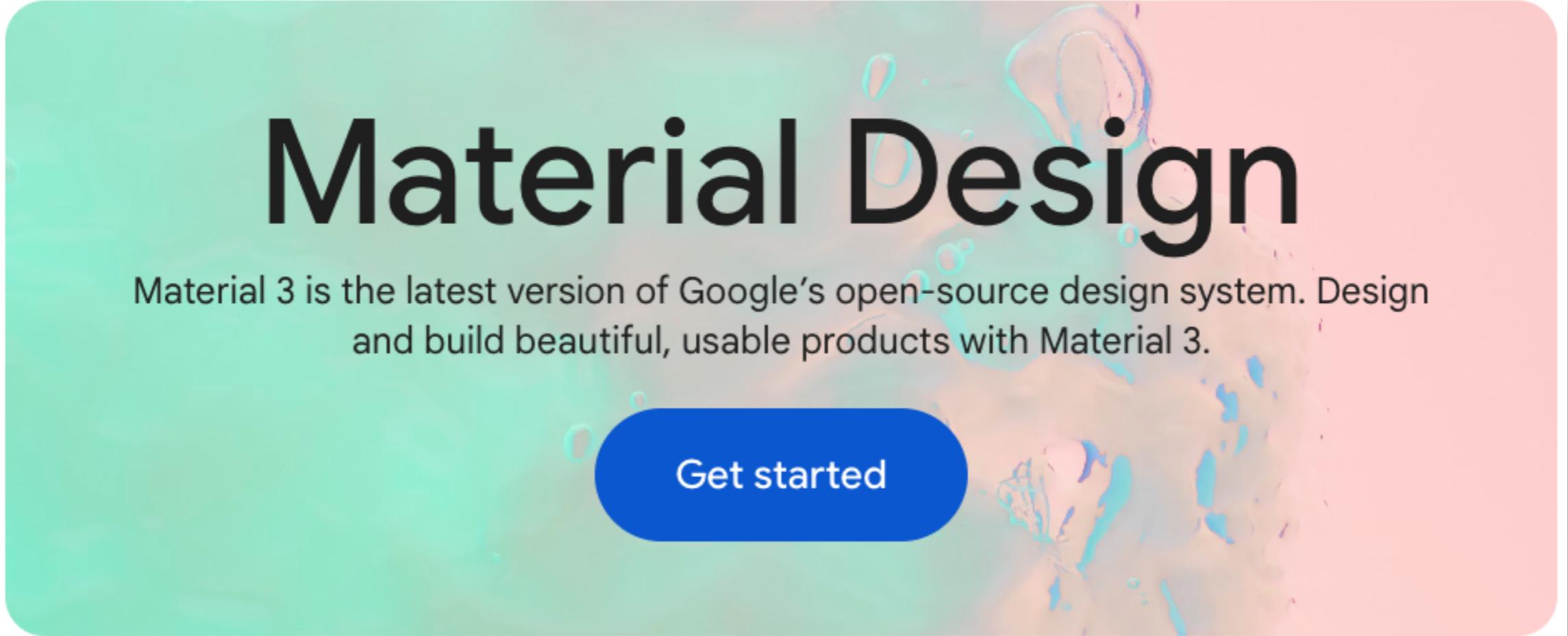
\$dart format

\$dart analyze

\$dart fix



Material Design

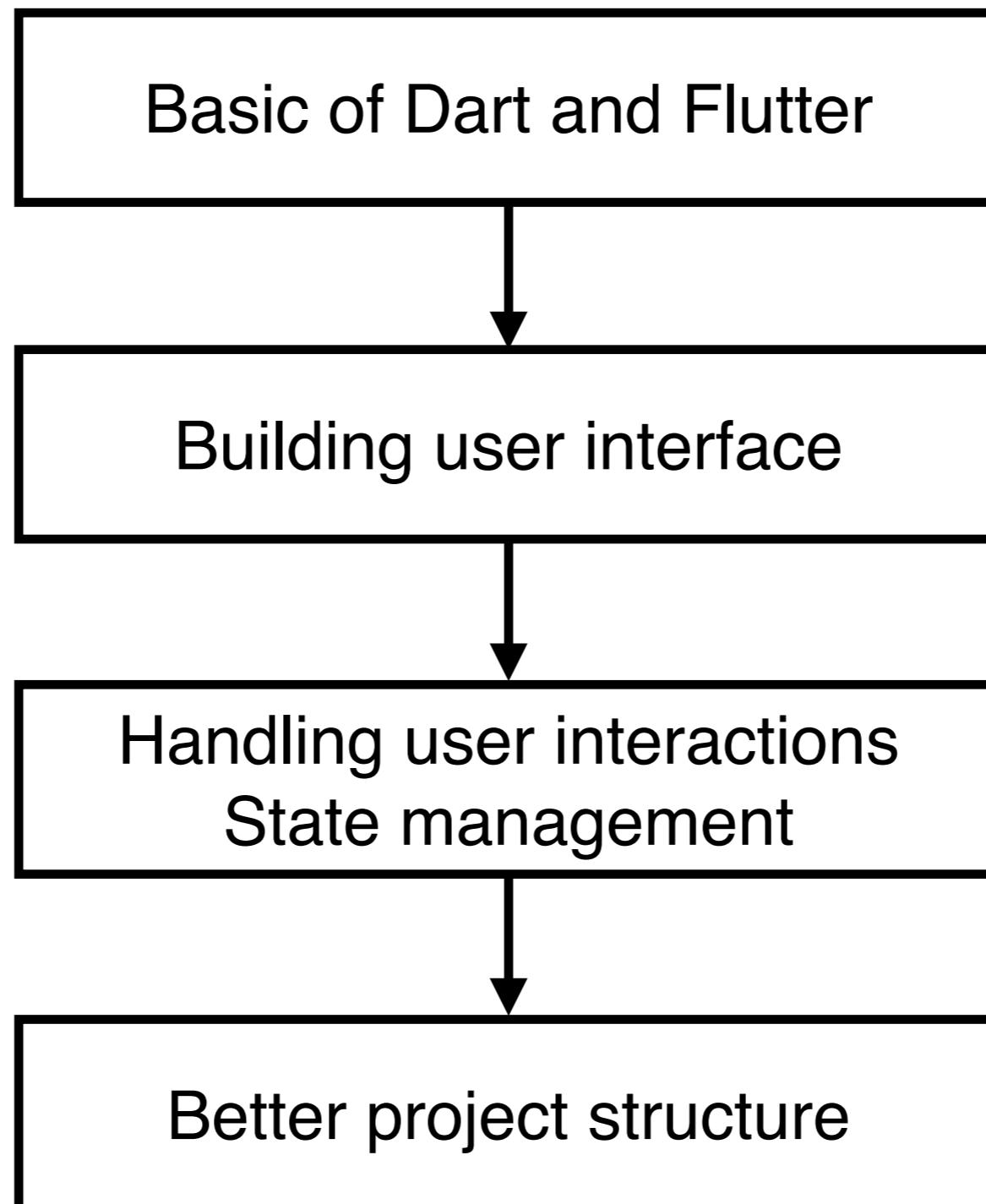


The screenshot shows the Material Design landing page. On the left, there's a vertical sidebar with icons and labels: a magnifying glass for search, a house for Home, three horizontal dots for Get started, a code editor for Develop, a book for Foundations, and a color palette for Styles. The main content area has a blurred background image of colorful, abstract shapes. At the top, the words "Material Design" are displayed in large, bold, black letters. Below them is a descriptive paragraph: "Material 3 is the latest version of Google's open-source design system. Design and build beautiful, usable products with Material 3." A prominent blue button with white text in the center says "Get started".

<https://m3.material.io/>



Learning Path



Basic of Flutter and Dart



Basic of Flutter

Start from lib/main.dart

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

void main() {
  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
```



Basic of Flutter

Widget tree of application

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

void main() {
  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
```



Flutter UIs are build with Widget



Build-in Widgets



Widgets

Building blocks that can be connected together
Reusable by nature



Essential of Widgets

Layout design

Styling

Container

Color

Theme

Row

Column

Text style

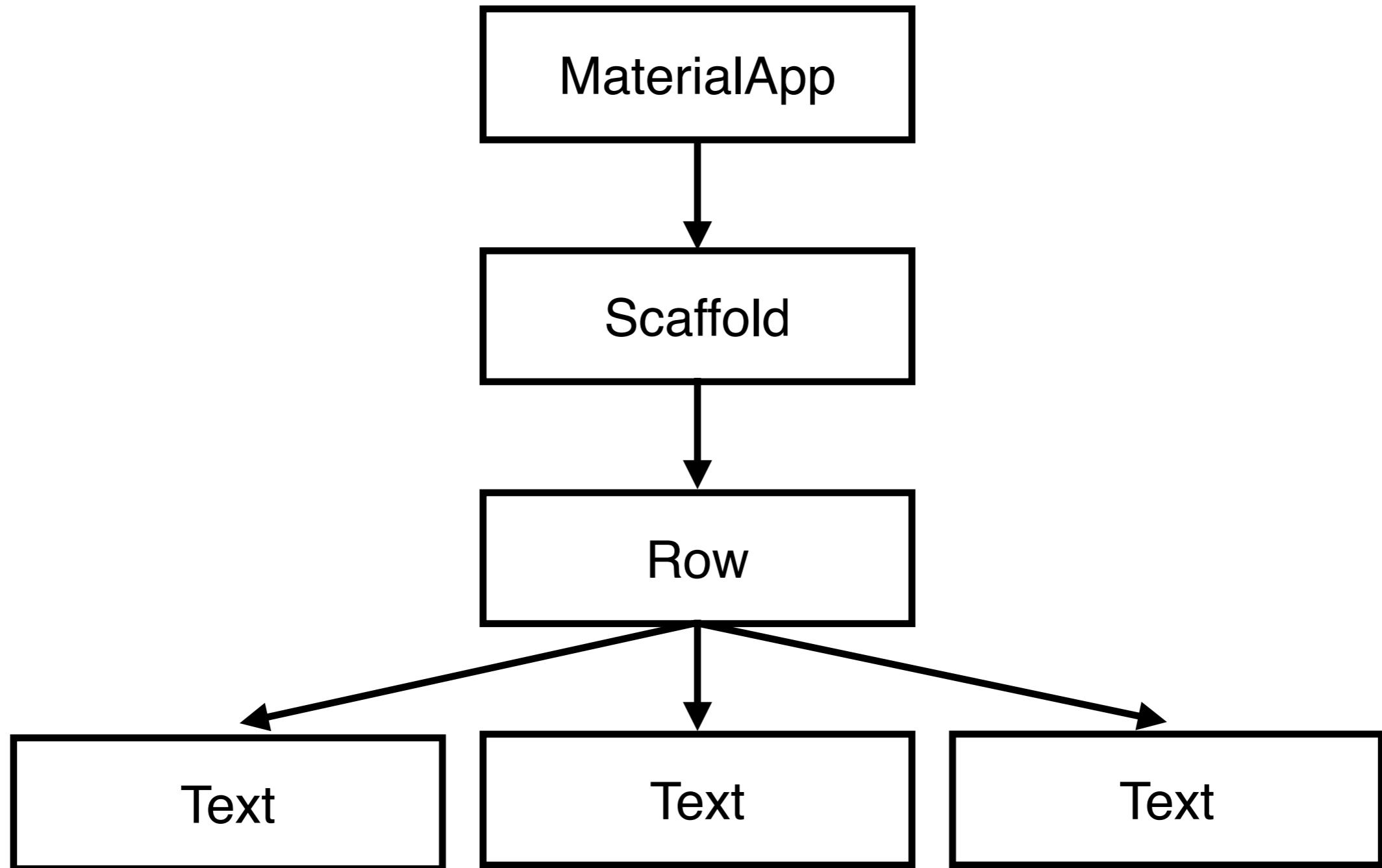
Stack

Padding

Center



Widget tree



<https://docs.flutter.dev/ui>



Code Example

```
void main() {  
  runApp(MaterialApp(  
    title: 'Flutter Tutorial',  
    home: Scaffold(  
      appBar: AppBar(  
        title: const Text('Hello Flutter'),  
      ),  
      body: const Row(  
        children: [  
          Expanded(  
            child: Text('Text 1', textAlign: TextAlign.center),  
          ),  
          Expanded(  
            child: Text('Text 2', textAlign: TextAlign.center),  
          ),  
          Expanded(  
            child: Text('Text 2', textAlign: TextAlign.center),  
          ),  
        ],  
      ),  
    )),  
};
```

Hello Flutter

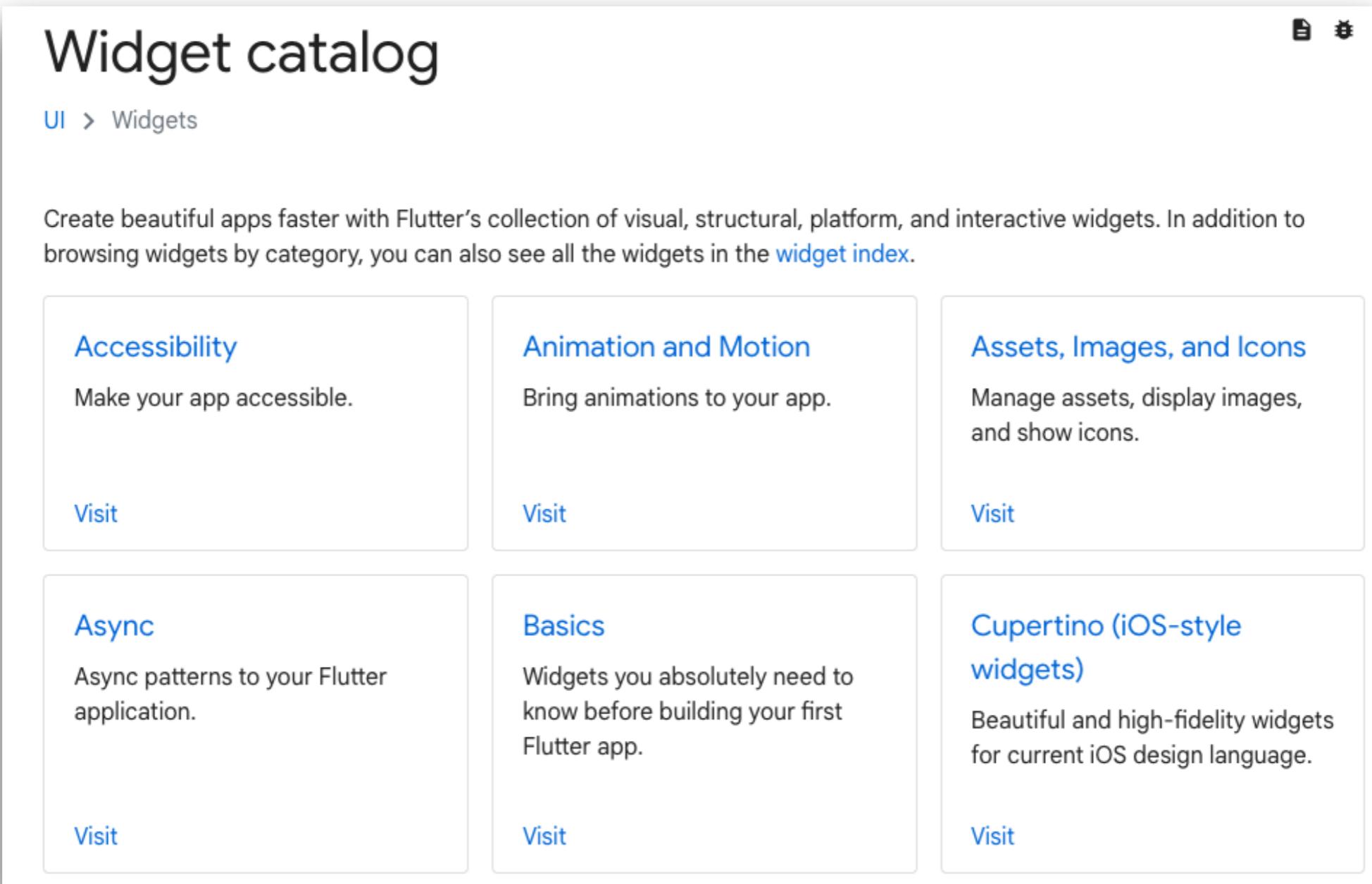
Text 1

Text 2

Text 2



Widget Catalogs



The screenshot shows the Flutter Widget Catalogs page. At the top, there's a navigation bar with a search icon, a gear icon, and a user profile icon. Below the navigation bar, the title "Widget catalog" is displayed, followed by a breadcrumb trail "UI > Widgets". A main text block encourages users to create beautiful apps faster with Flutter's collection of visual, structural, platform, and interactive widgets. It also mentions the "widget index". The page is organized into six cards arranged in a 2x3 grid:

- Accessibility**: Make your app accessible. [Visit](#)
- Animation and Motion**: Bring animations to your app. [Visit](#)
- Assets, Images, and Icons**: Manage assets, display images, and show icons. [Visit](#)
- Async**: Async patterns to your Flutter application. [Visit](#)
- Basics**: Widgets you absolutely need to know before building your first Flutter app. [Visit](#)
- Cupertino (iOS-style widgets)**: Beautiful and high-fidelity widgets for current iOS design language. [Visit](#)

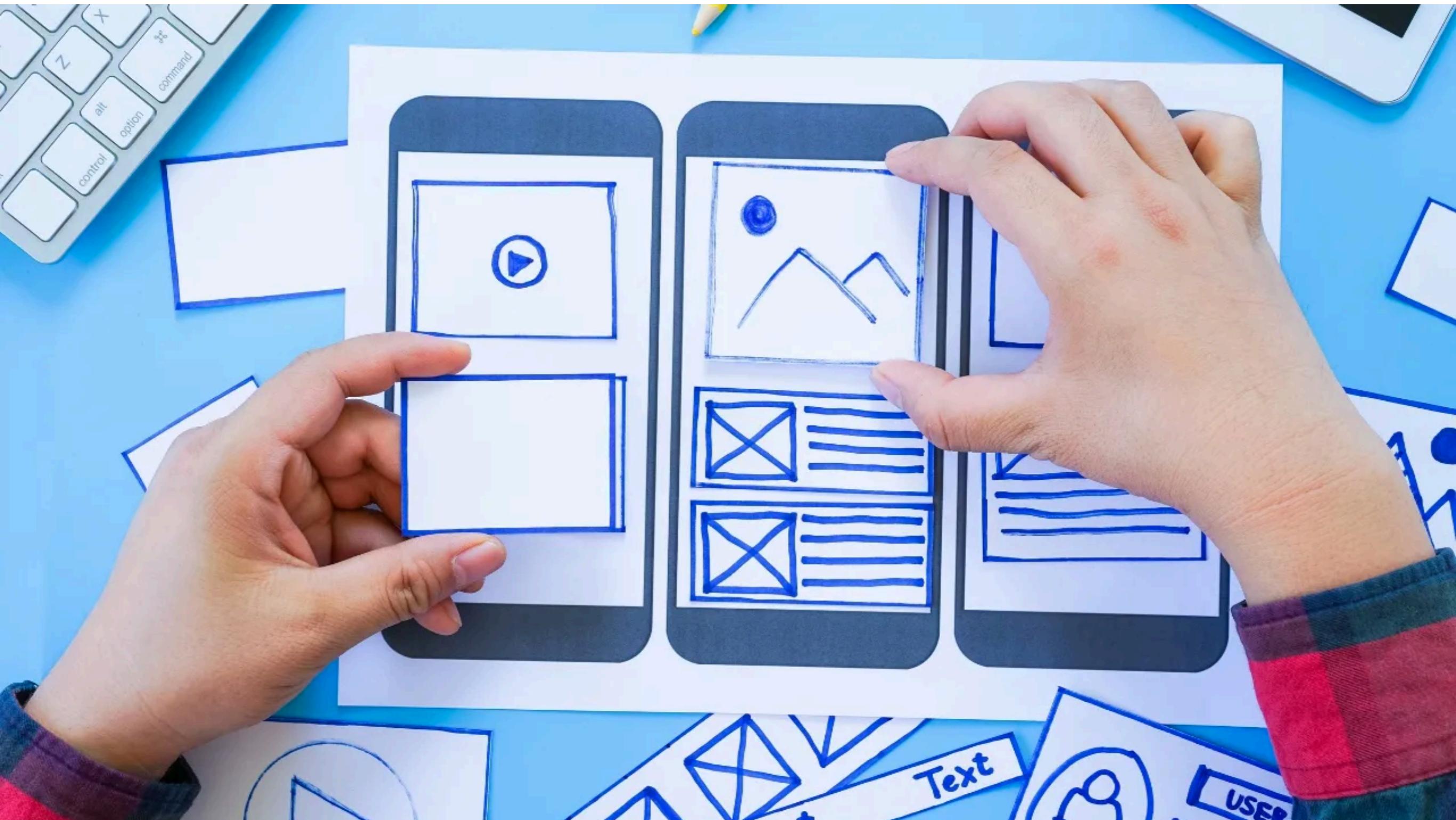
<https://docs.flutter.dev/ui/widgets>



Types of Widgets



Types of Widgets



Types of Widgets

Stateless Widget
Stateful Widget

Immutable state

Button

Icon

Text

Mutable state

Checkbox

Radio
button

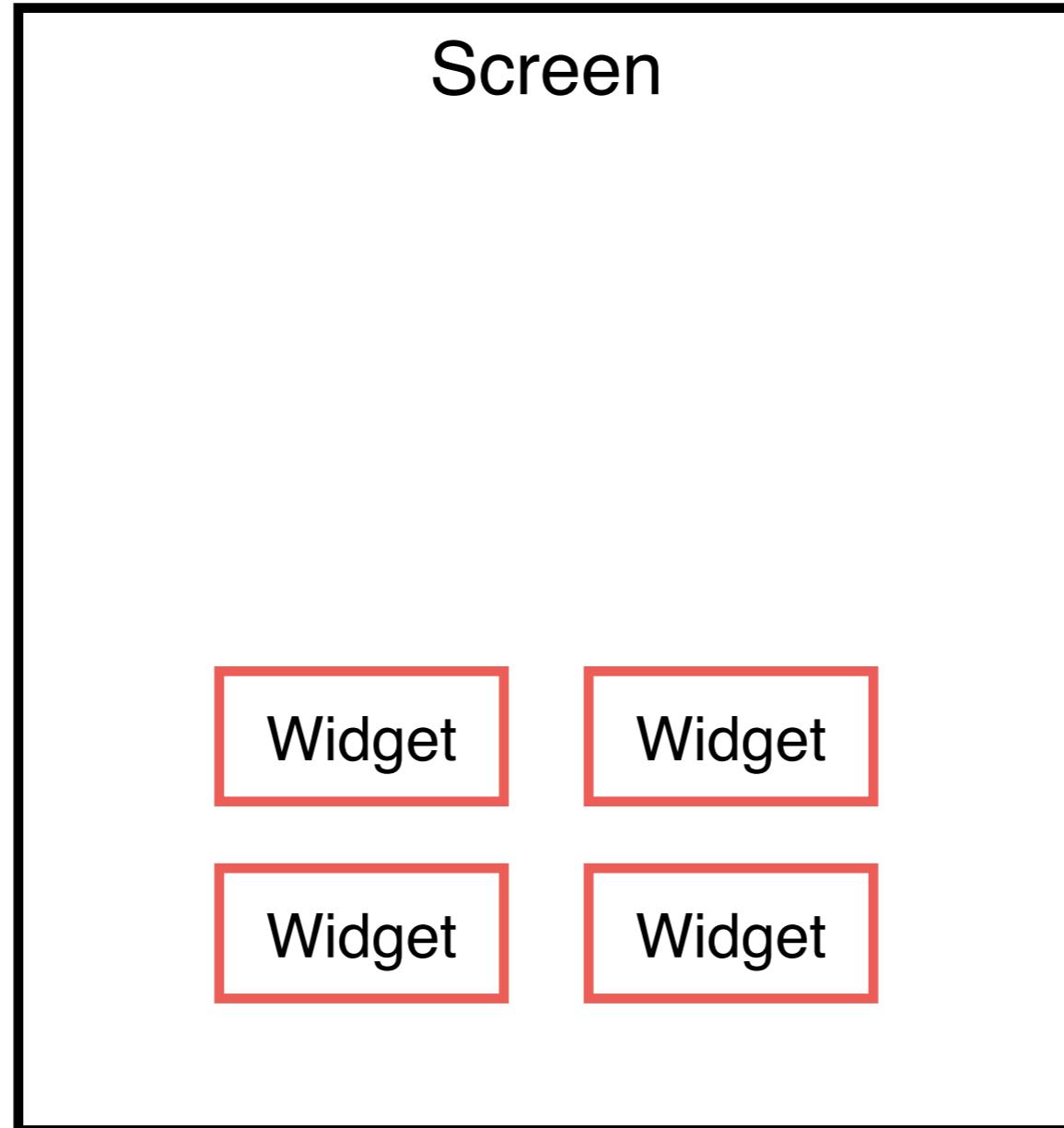


Stateless Widget

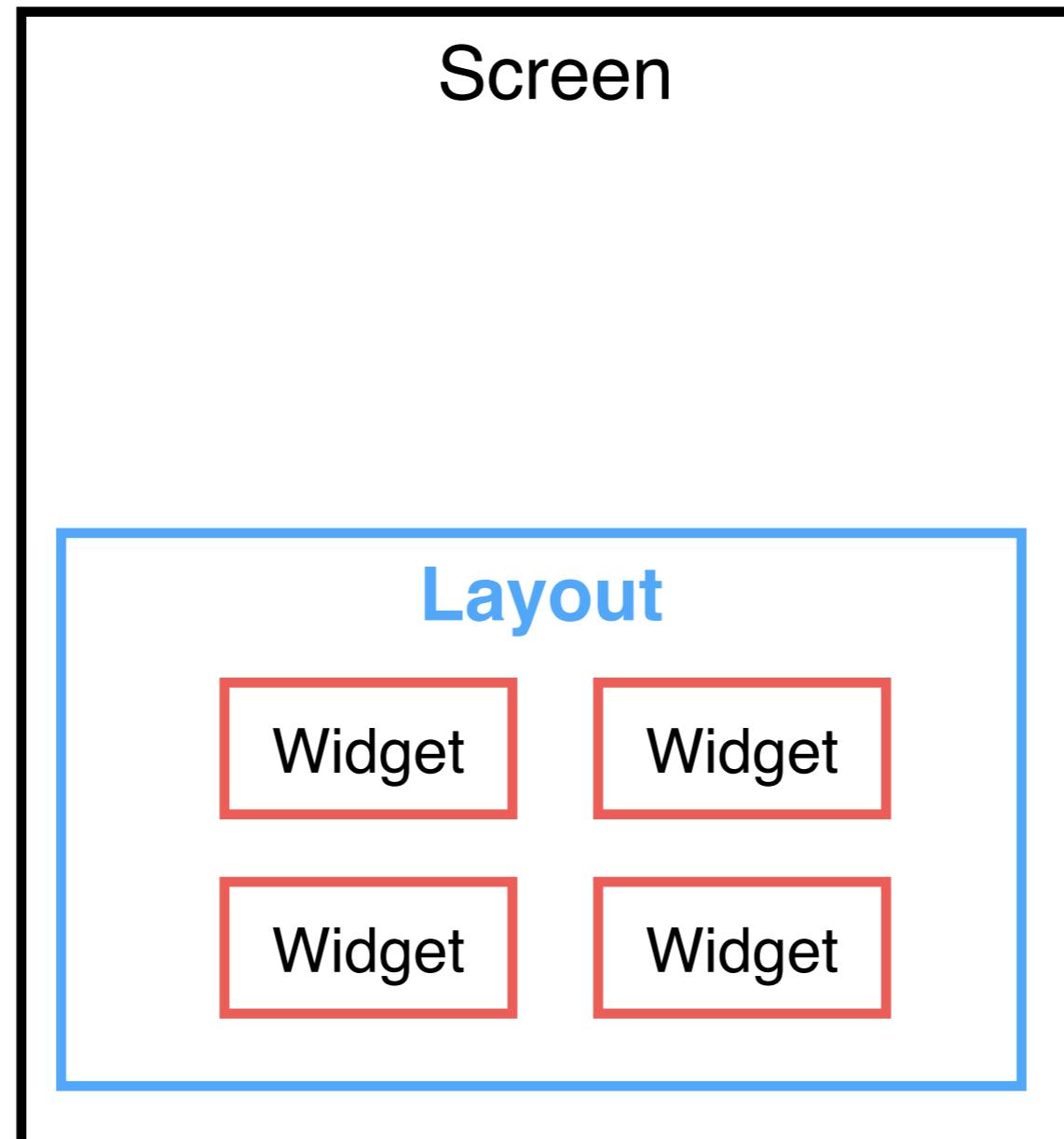
```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Flutter Tutorial',  
      home: Scaffold(  
        appBar: AppBar(  
          title: const Text('Hello Flutter'),  
        ),  
        body: const Row(  
          children: [  
            Expanded(  
              child: Text('Text 1', textAlign: TextAlign.center),  
            ),  
            Expanded(  
              child: Text('Text 2', textAlign: TextAlign.center),  
            ),  
            Expanded(  
              child: Text('Text 2', textAlign: TextAlign.center),  
            ),  
          ],  
        ),  
      );  
  }  
}
```



Widgets in screen



Grouping by Layout



Layout

<https://docs.flutter.dev/ui/widgets/layout>



Flutter 101

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Layout in Flutter app

List and Grid

Working with long list

Scrolling

Adaptive and responsive design

Container

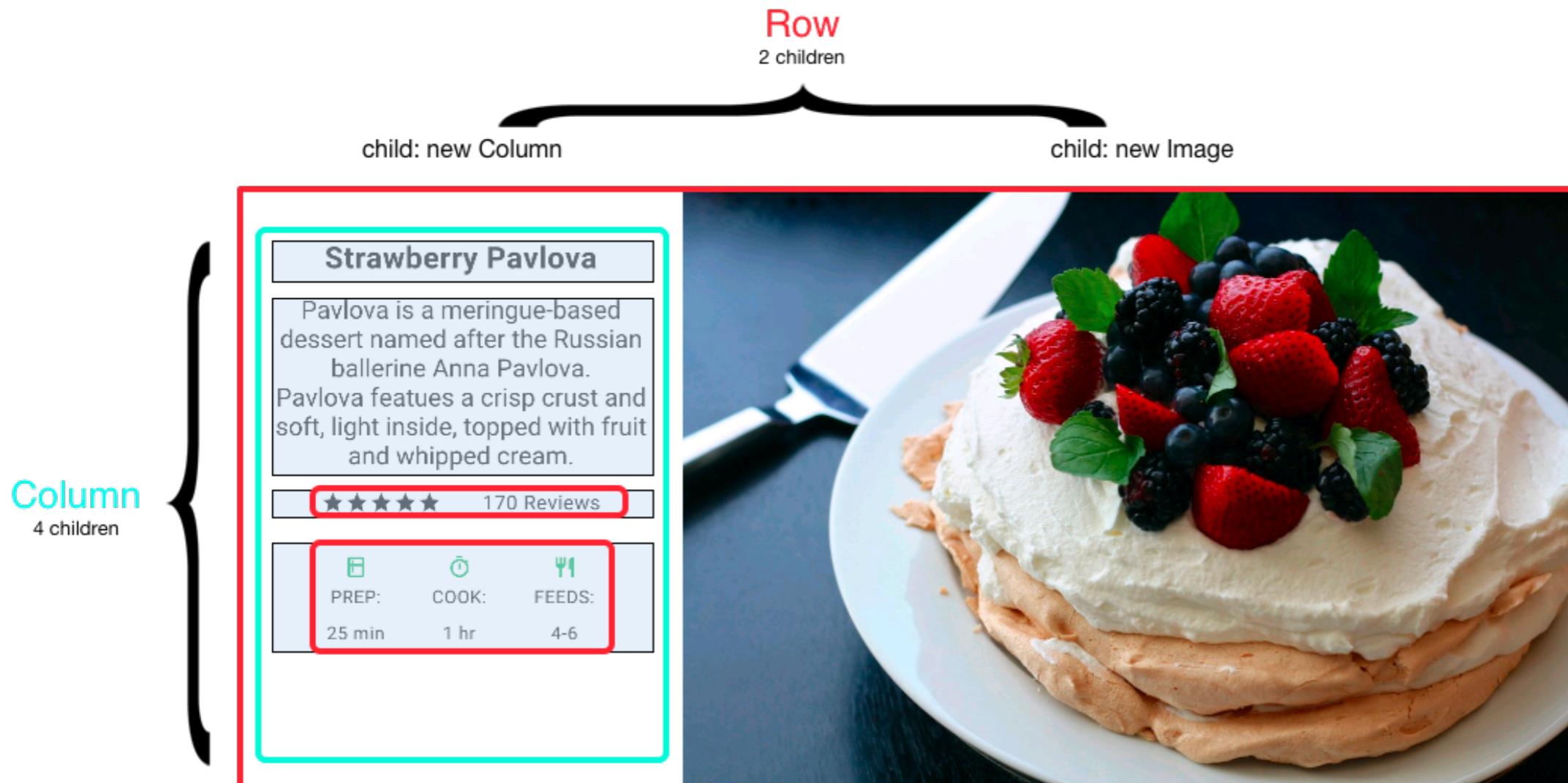
Stack

Card

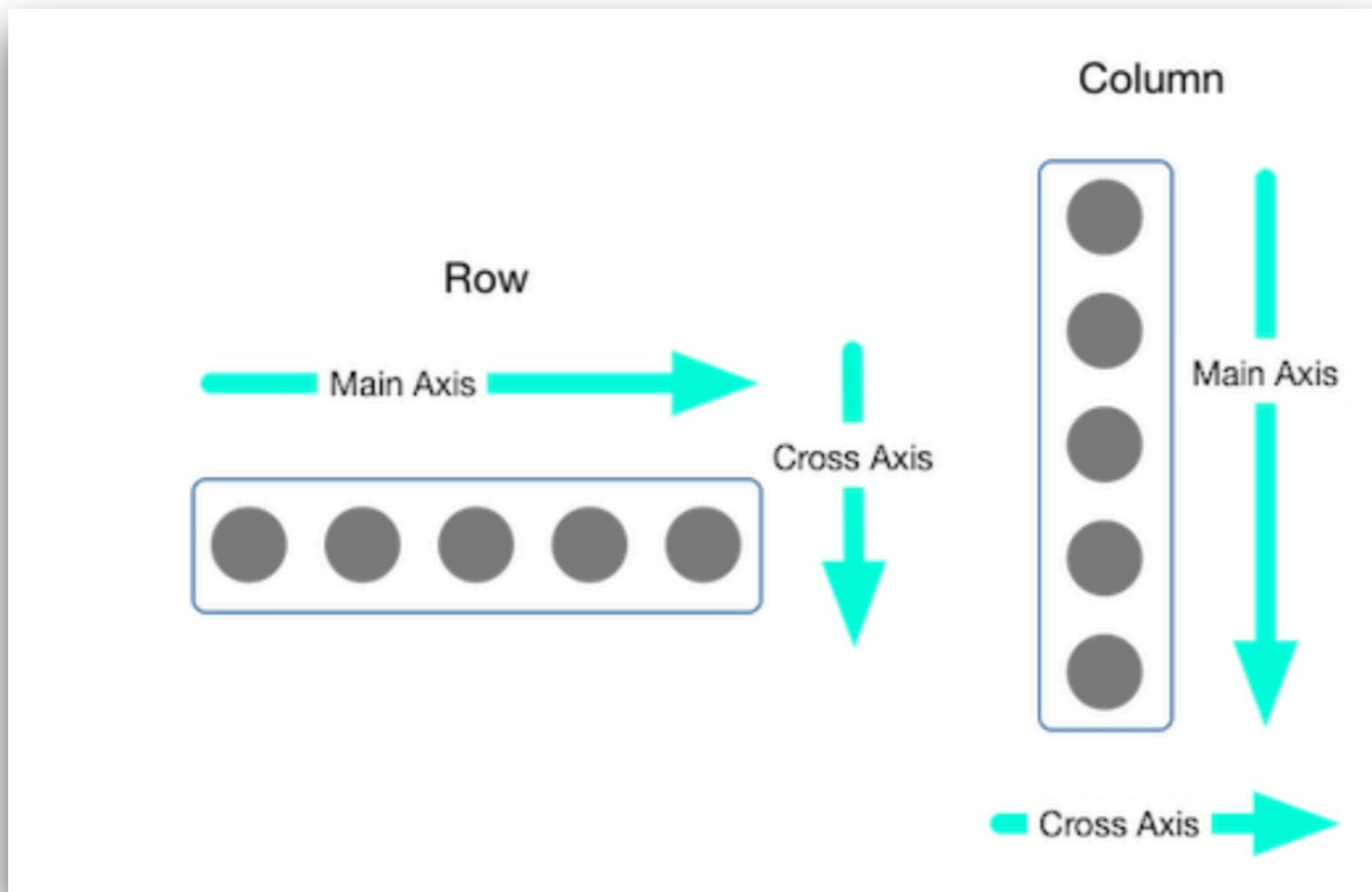
<https://docs.flutter.dev/ui/widgets/layout>



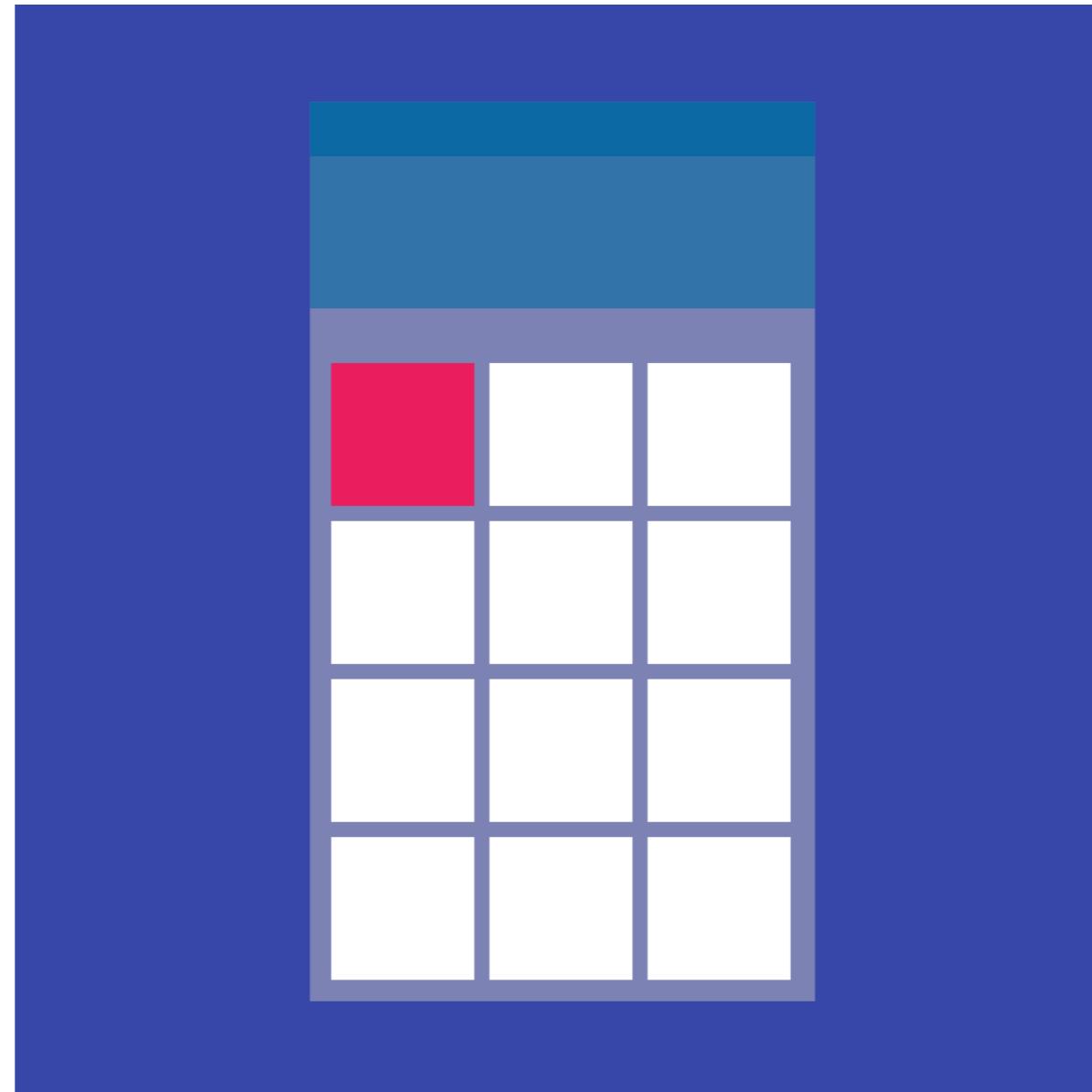
Row and Column



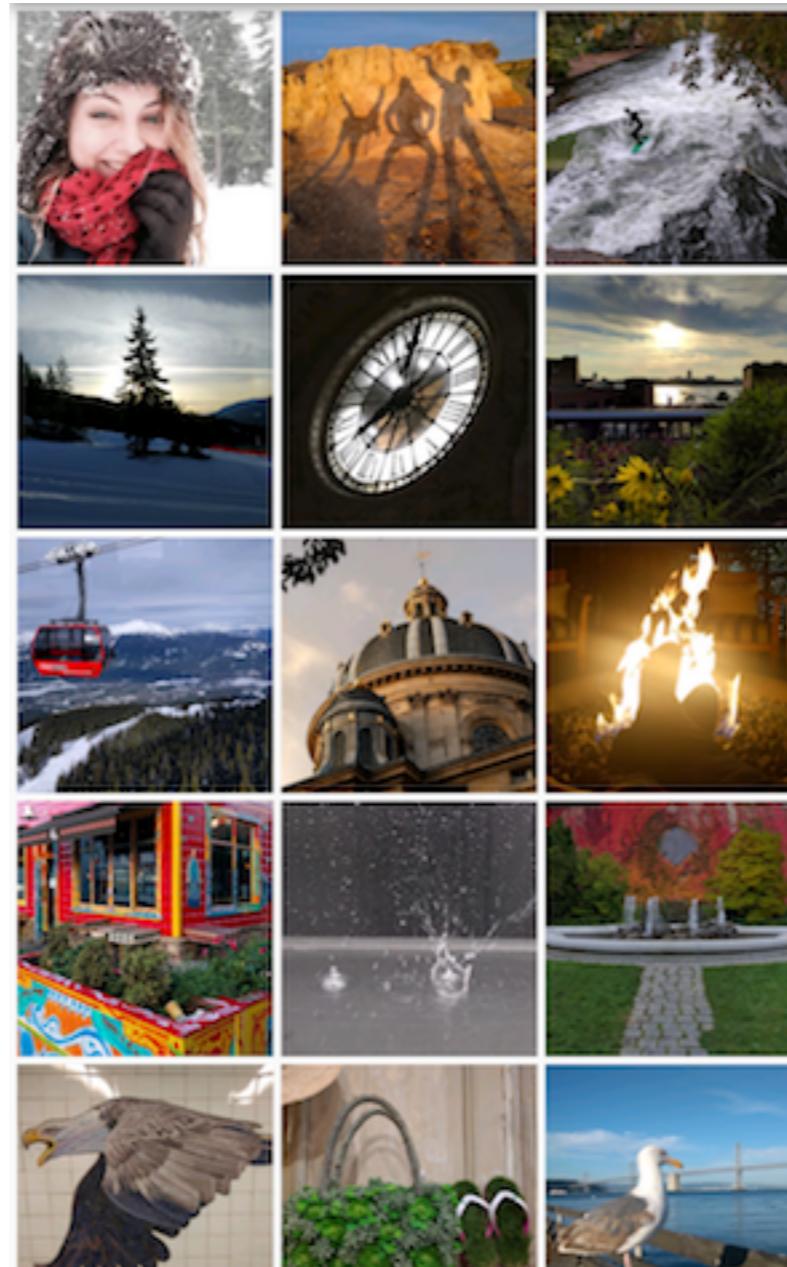
Align of widget



GridView



GridView



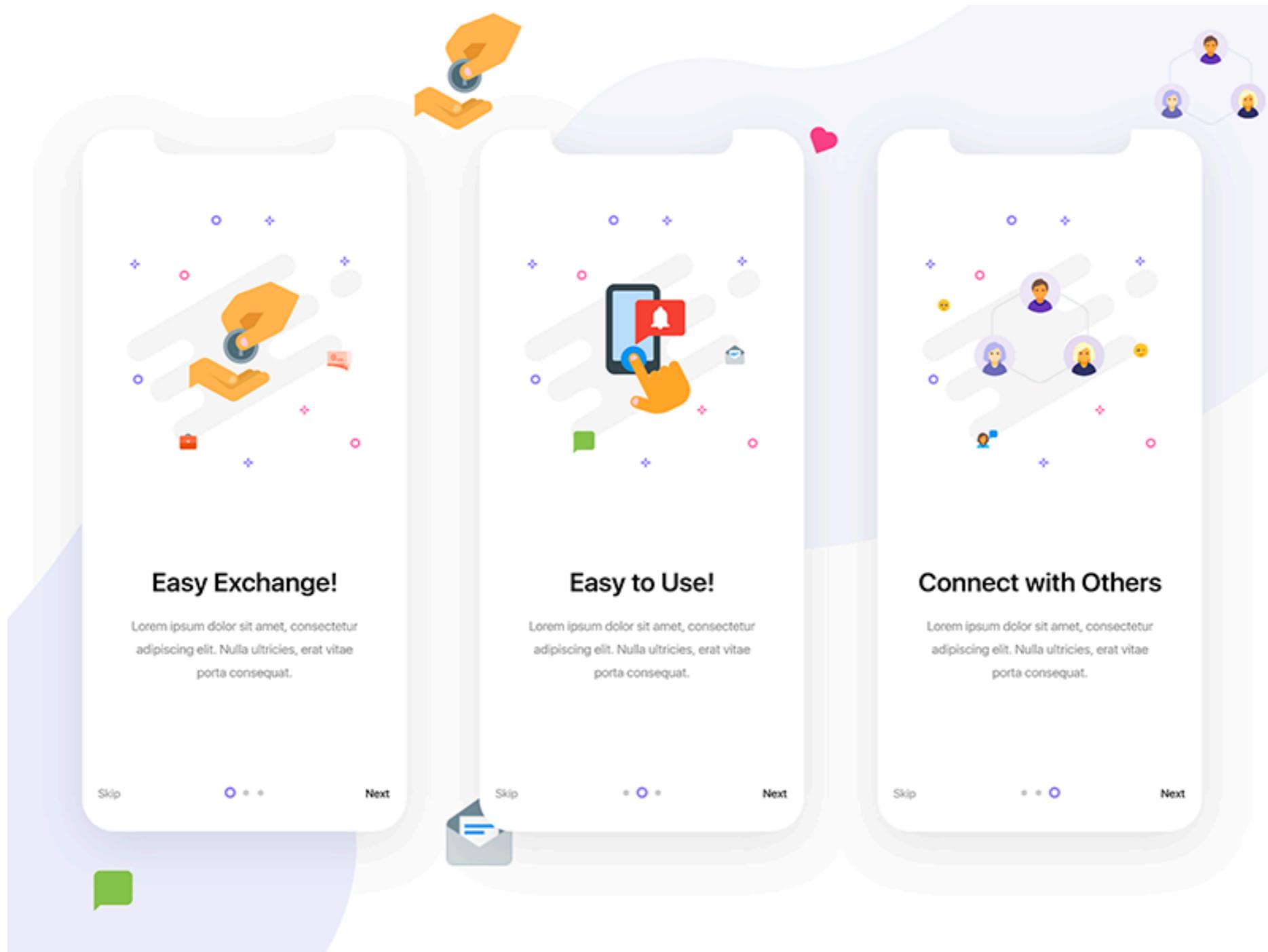
15 Puzzle



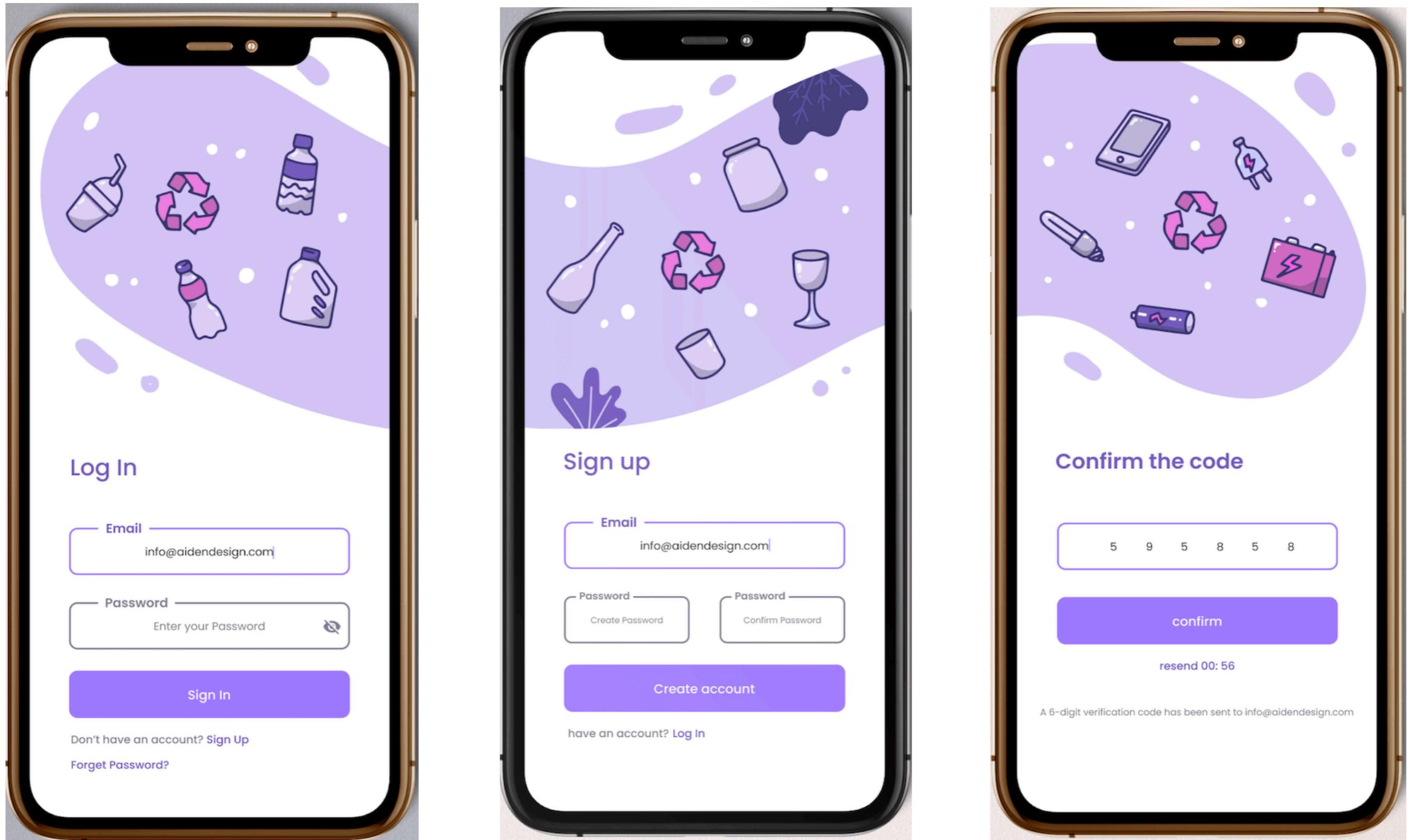
https://en.wikipedia.org/wiki/15_Puzzle



Onboarding screen



Login



Product Page



A woman with curly hair, wearing a light-colored trench coat over a green crop top and matching pants, stands on a white chair in a desert landscape with sand dunes in the background.

Trendy trench set

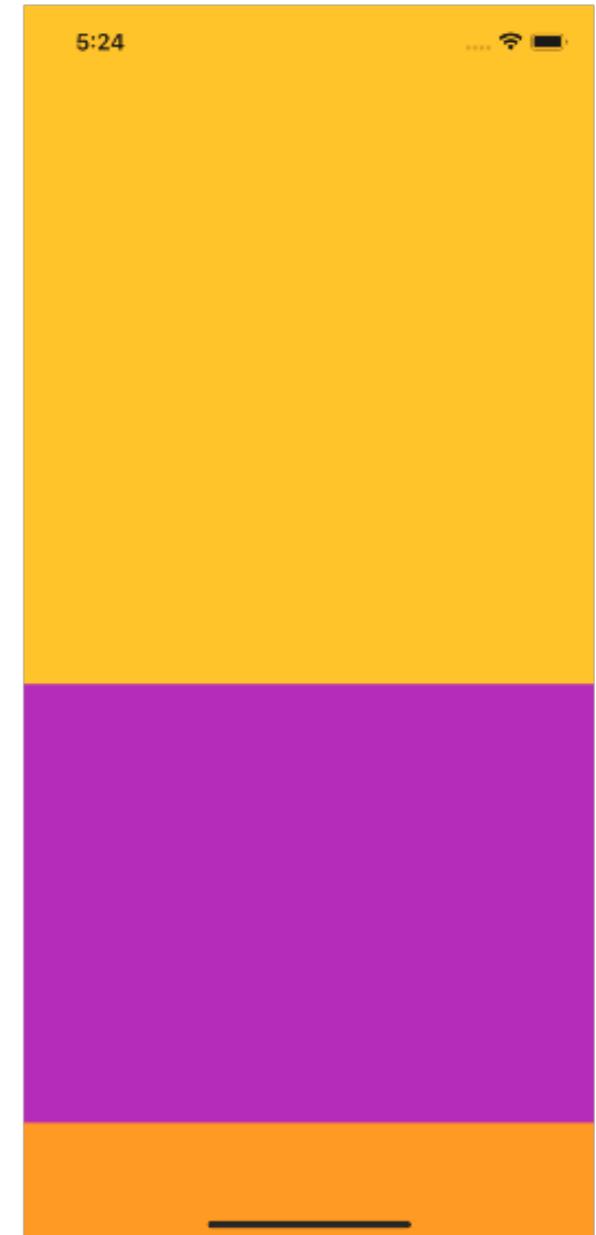
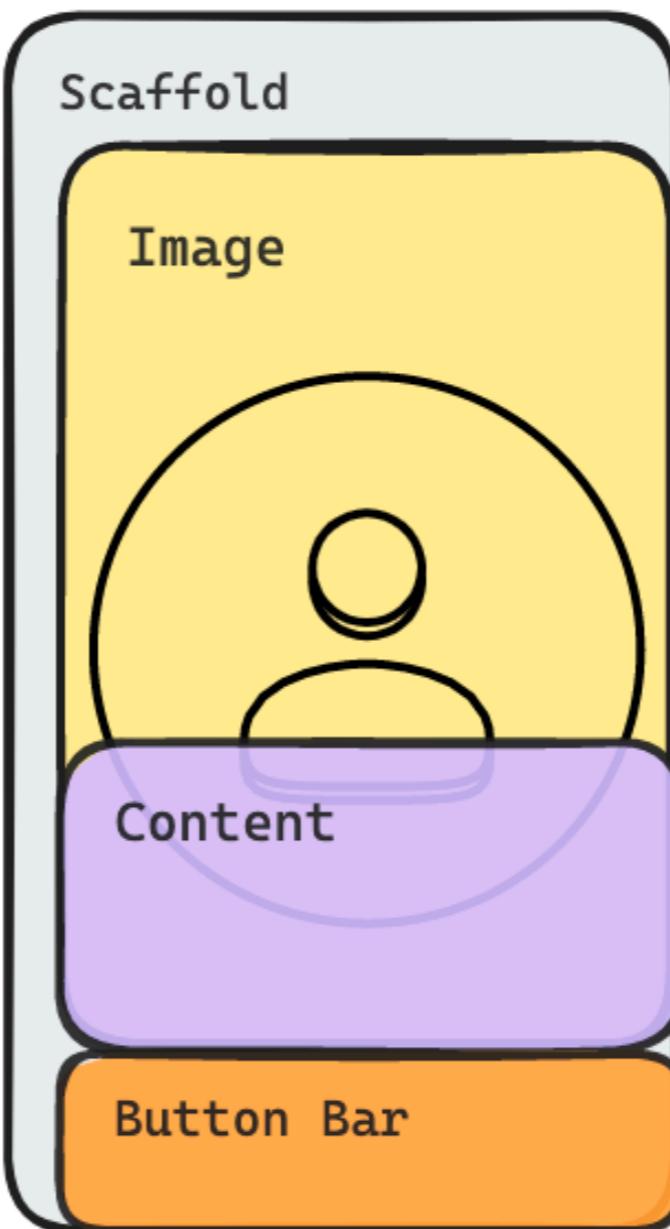
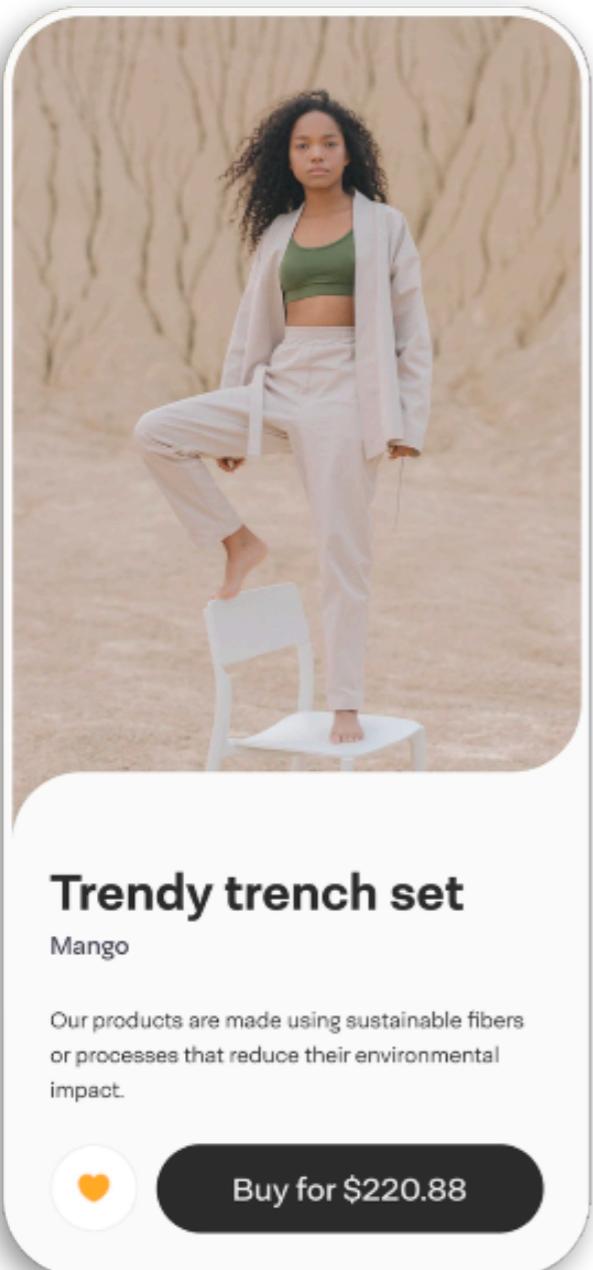
Mango

Our products are made using sustainable fibers or processes that reduce their environmental impact.

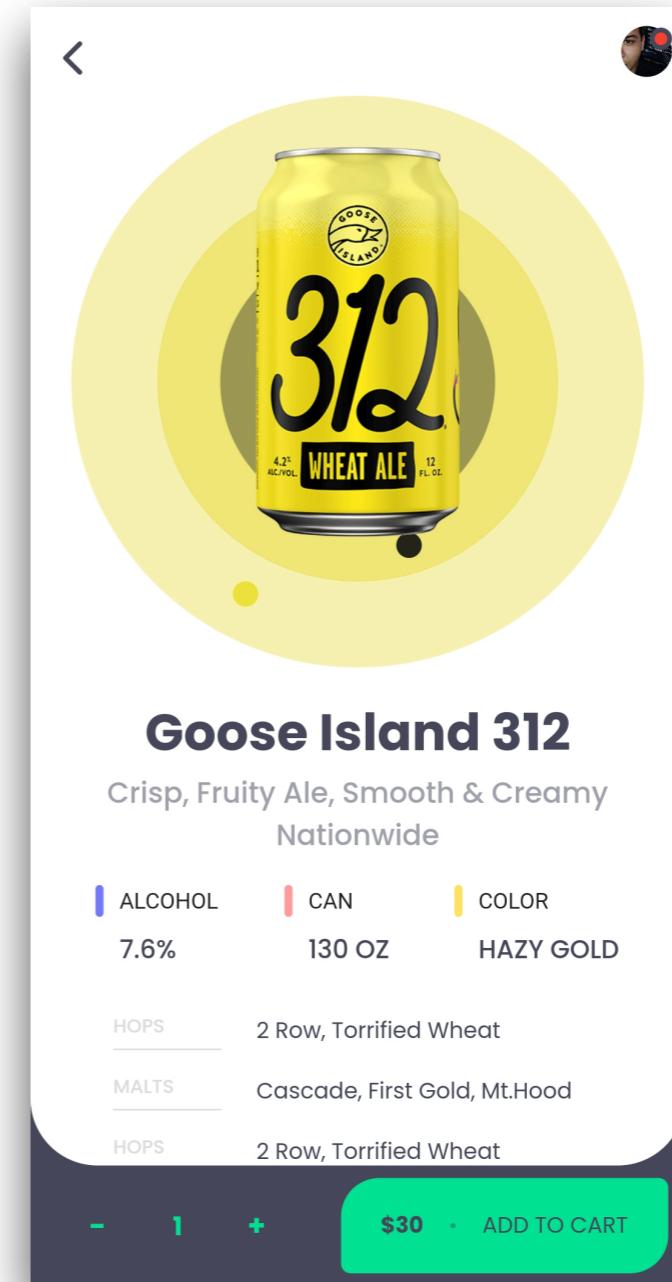
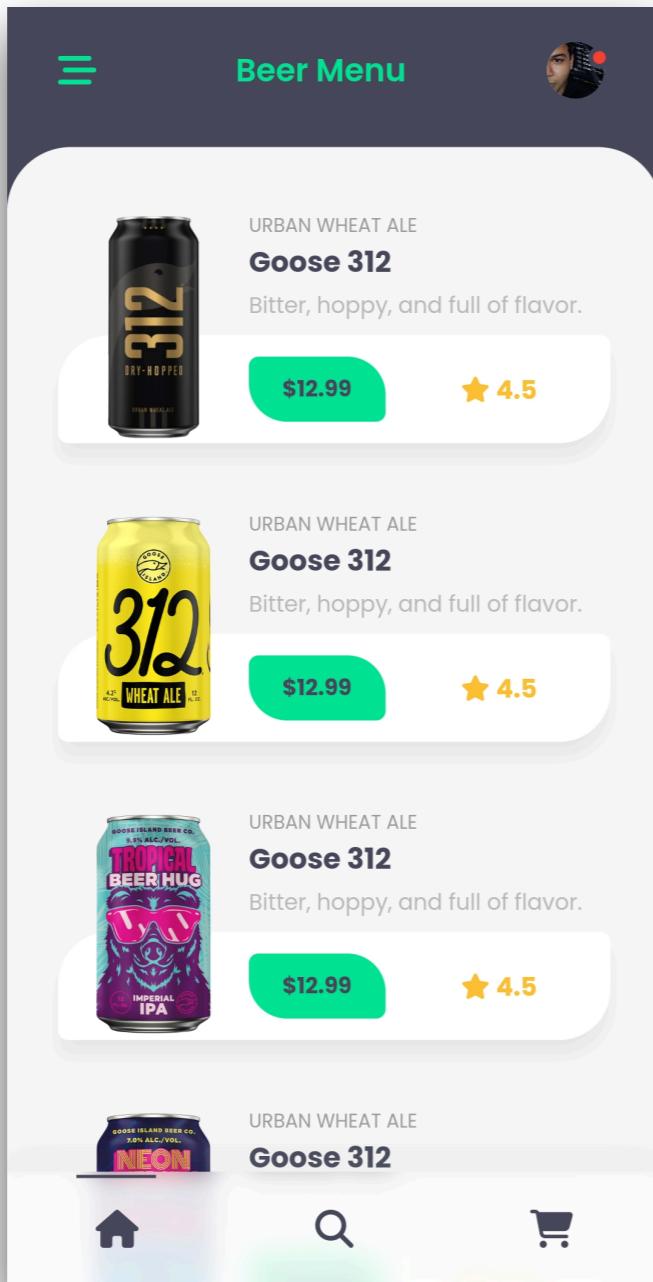
 Buy for \$220.88



Product Page



Listing and Detail

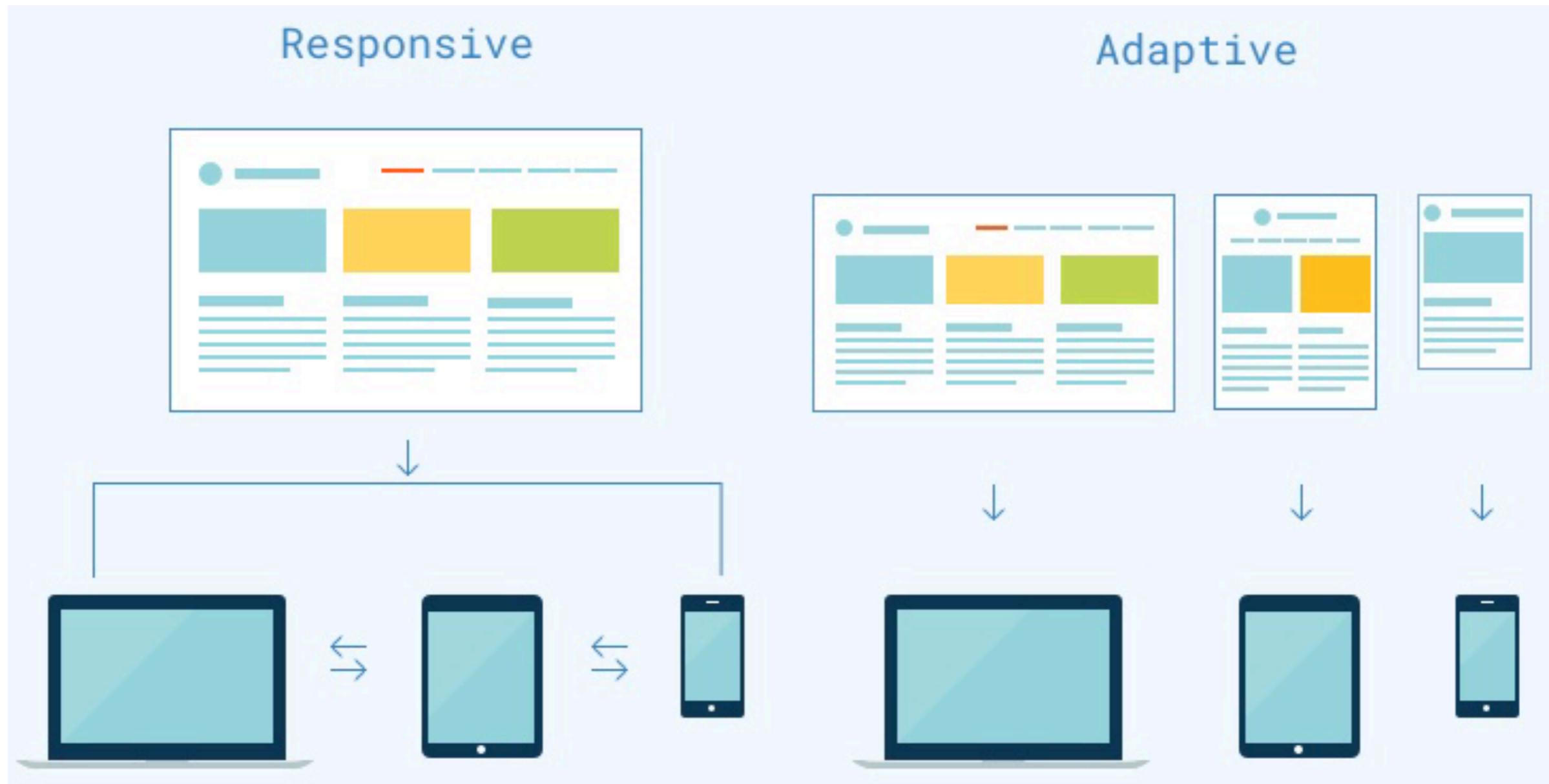


Adaptive vs Responsive App

<https://docs.flutter.dev/ui/layout/responsive/adaptive-responsive>



Adaptive vs Responsive App



Adaptive

Change functionality based
on platform's capabilities

Android

iOS

Foldable

Tablet

Web

Desktop



Responsive

Change how something looks based
on the available screens



Responsive UI

Don't hard code value (dimension, position)

Use **MediaQuery** to get current size of window

Use **Flexible** and **Expanded** widgets (percentage)

Use LayoutBuilder to get ConstraintBox from parent

Get Orientation of device with **MediaQuery**/
OrientationBuilder

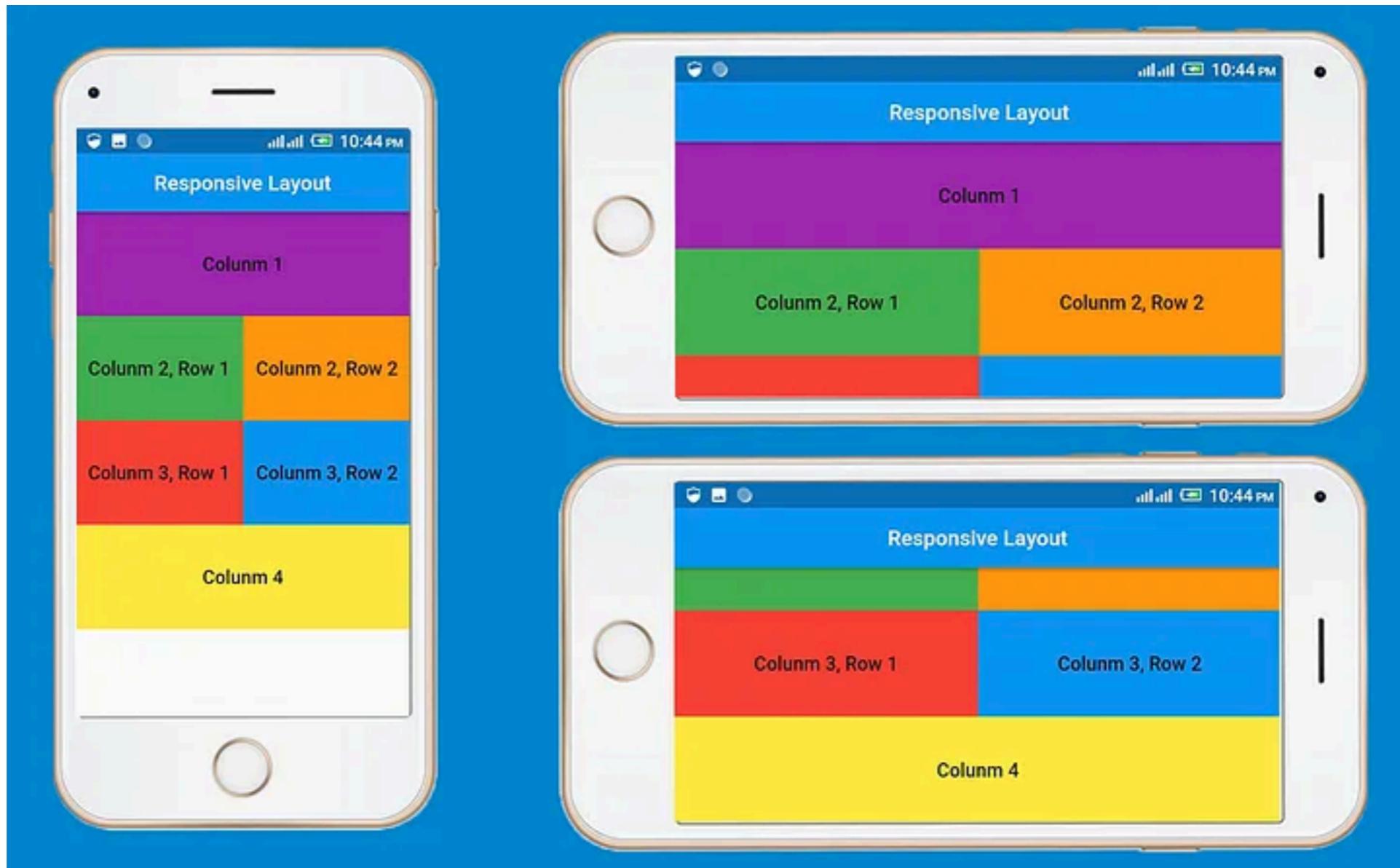
Size

Orientation

Device type



Responsive UI !!

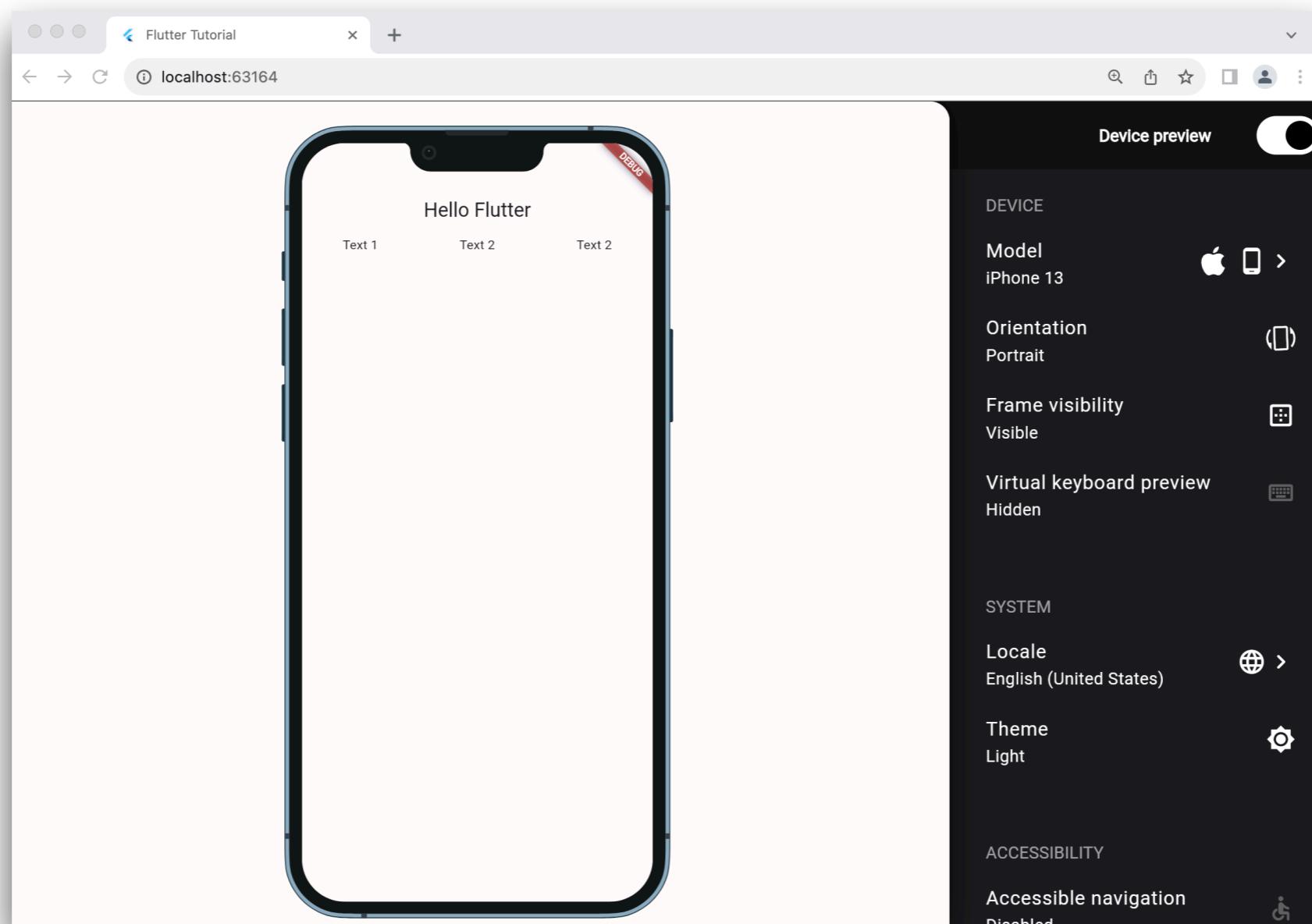


Flutter Device Review



Flutter Device Review

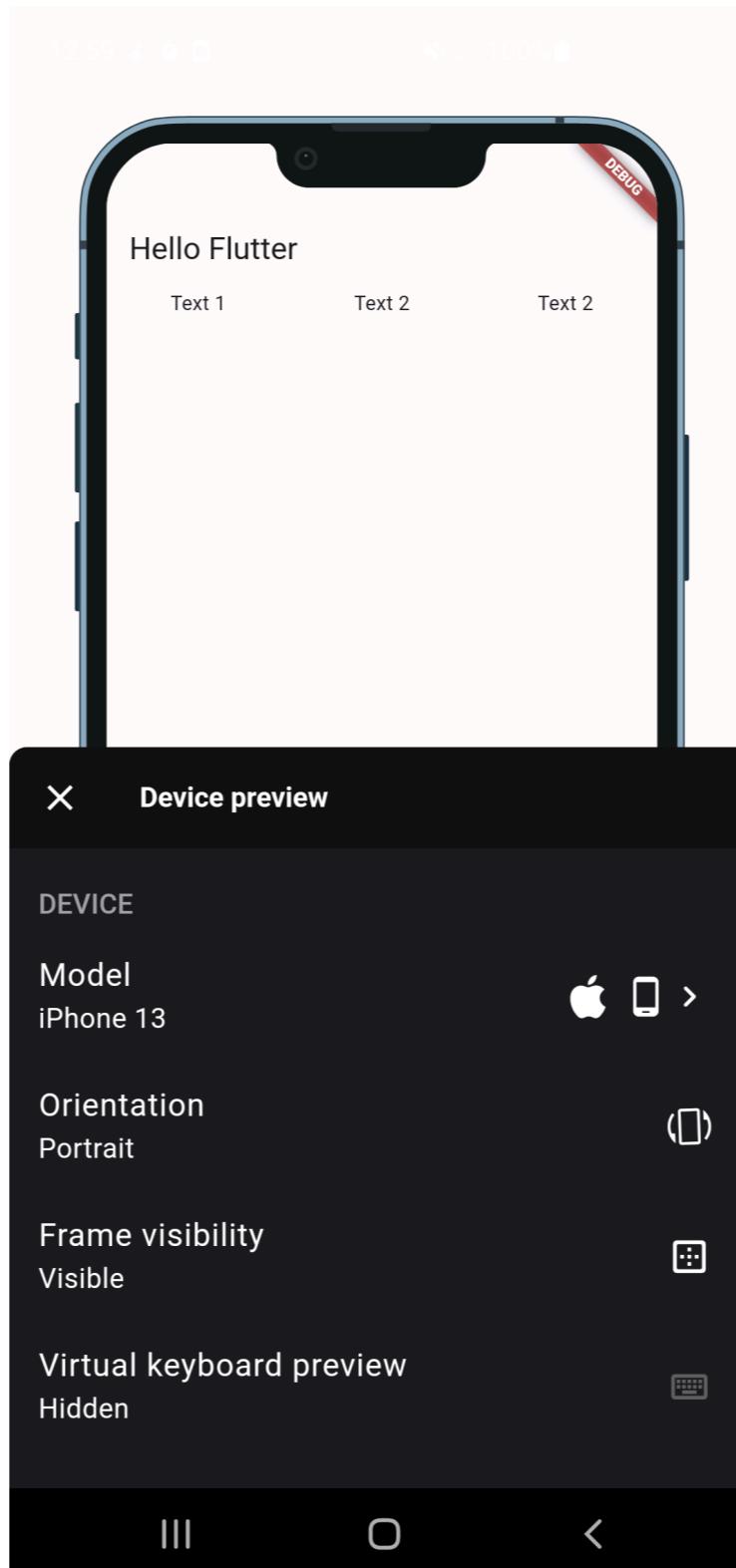
Preview on any device and orientation



https://pub.dev/packages/device_preview



Flutter Device Review



Flutter Device Review

\$flutter pub add device_preview

```
import 'package:device_preview/device_preview.dart';
import 'package:flutter/material.dart';

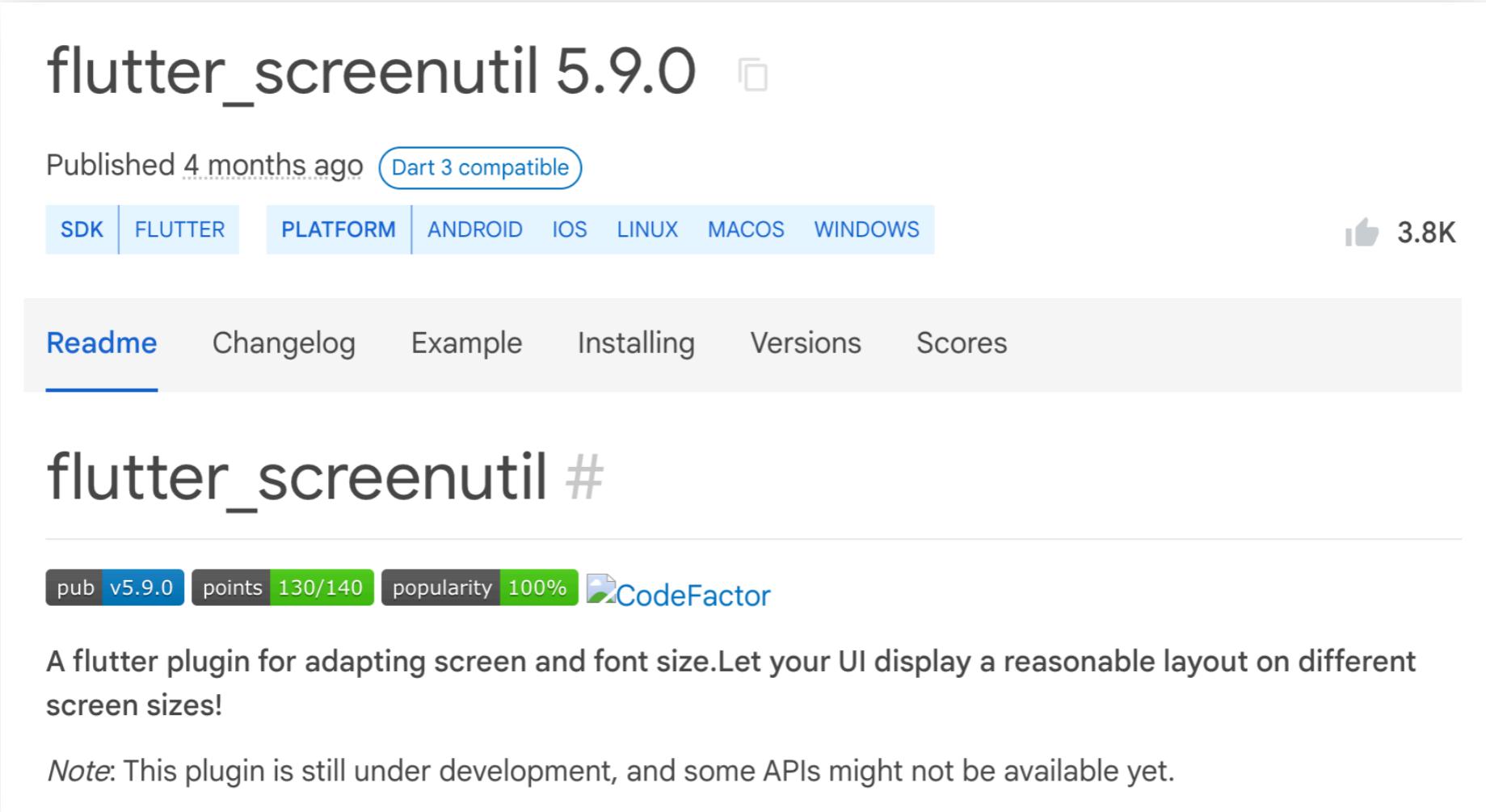
void main() {
  runApp(
    DevicePreview(
      builder: (context) => const MyApp()));
}
```

https://pub.dev/packages/device_preview



Using Flutter ScreenUtil

UI display a reasonable layout on difference screen



The screenshot shows the package page for `flutter_screenutil` version 5.9.0. The page includes the following details:

- Version:** 5.9.0
- Published:** 4 months ago
- Dart Compat:** Dart 3 compatible
- Platforms:** SDK, FLUTTER, PLATFORM, ANDROID, IOS, LINUX, MACOS, WINDOWS
- Rating:** 3.8K likes
- Links:** Readme, Changelog, Example, Installing, Versions, Scores
- Metrics:** pub v5.9.0, points 130/140, popularity 100%, CodeFactor
- Description:** A flutter plugin for adapting screen and font size. Let your UI display a reasonable layout on different screen sizes!
- Note:** This plugin is still under development, and some APIs might not be available yet.

https://pub.dev/packages/flutter_screenutil



App Theme

Screen 1

Screen 2

Layout

Widget

Widget

Widget

Widget

Layout

Widget

Widget

Widget

Widget



Workshop

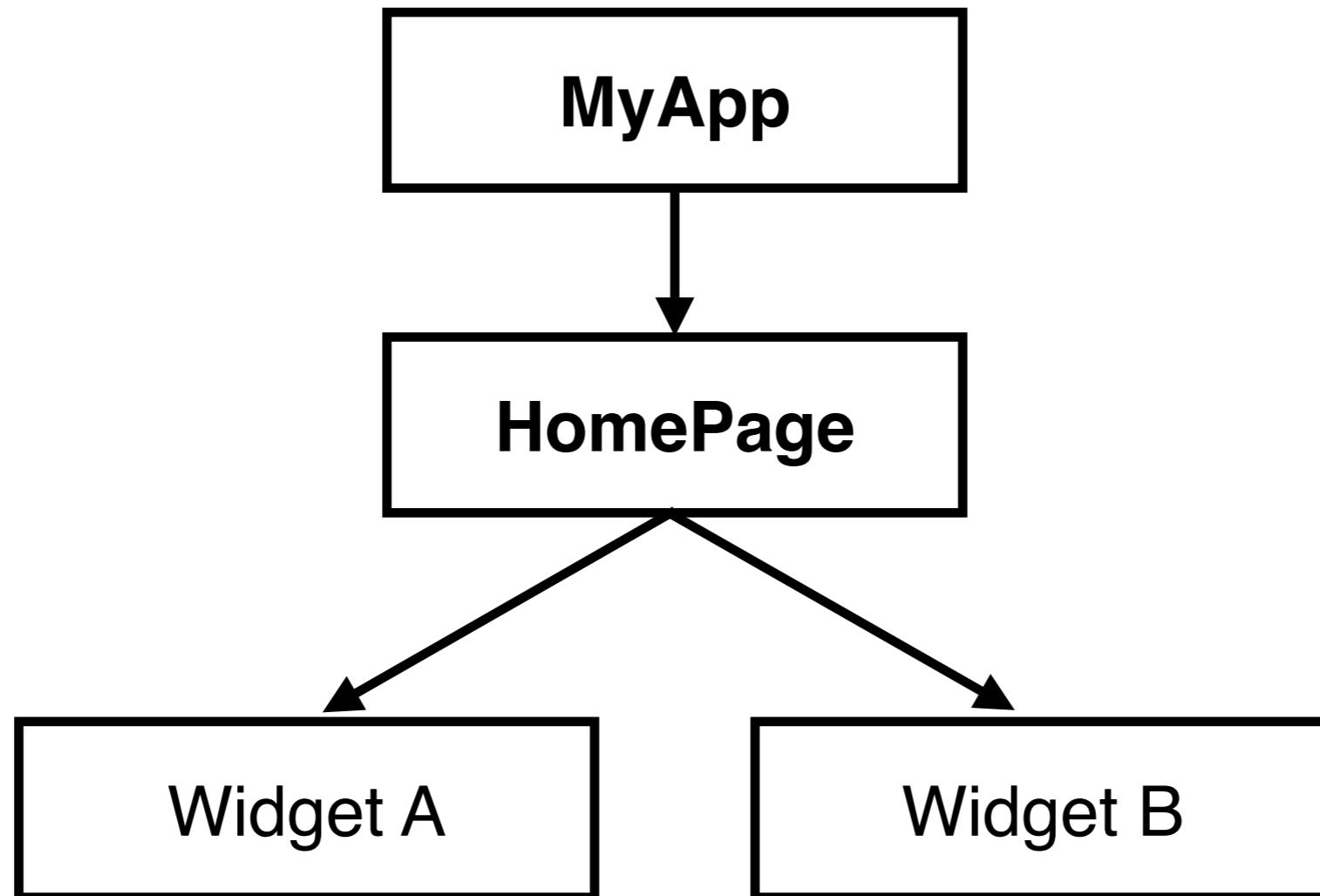
Random Text

Next

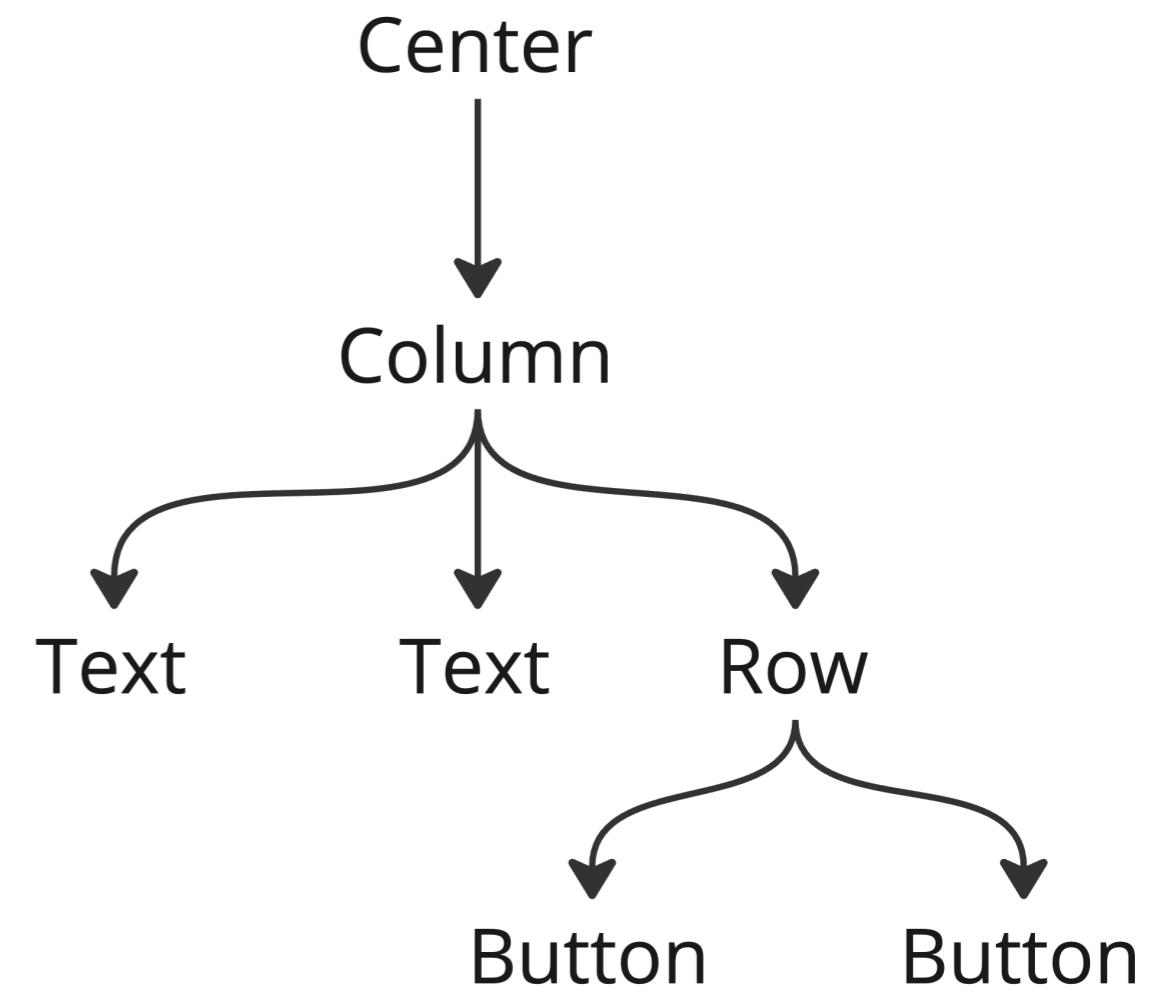
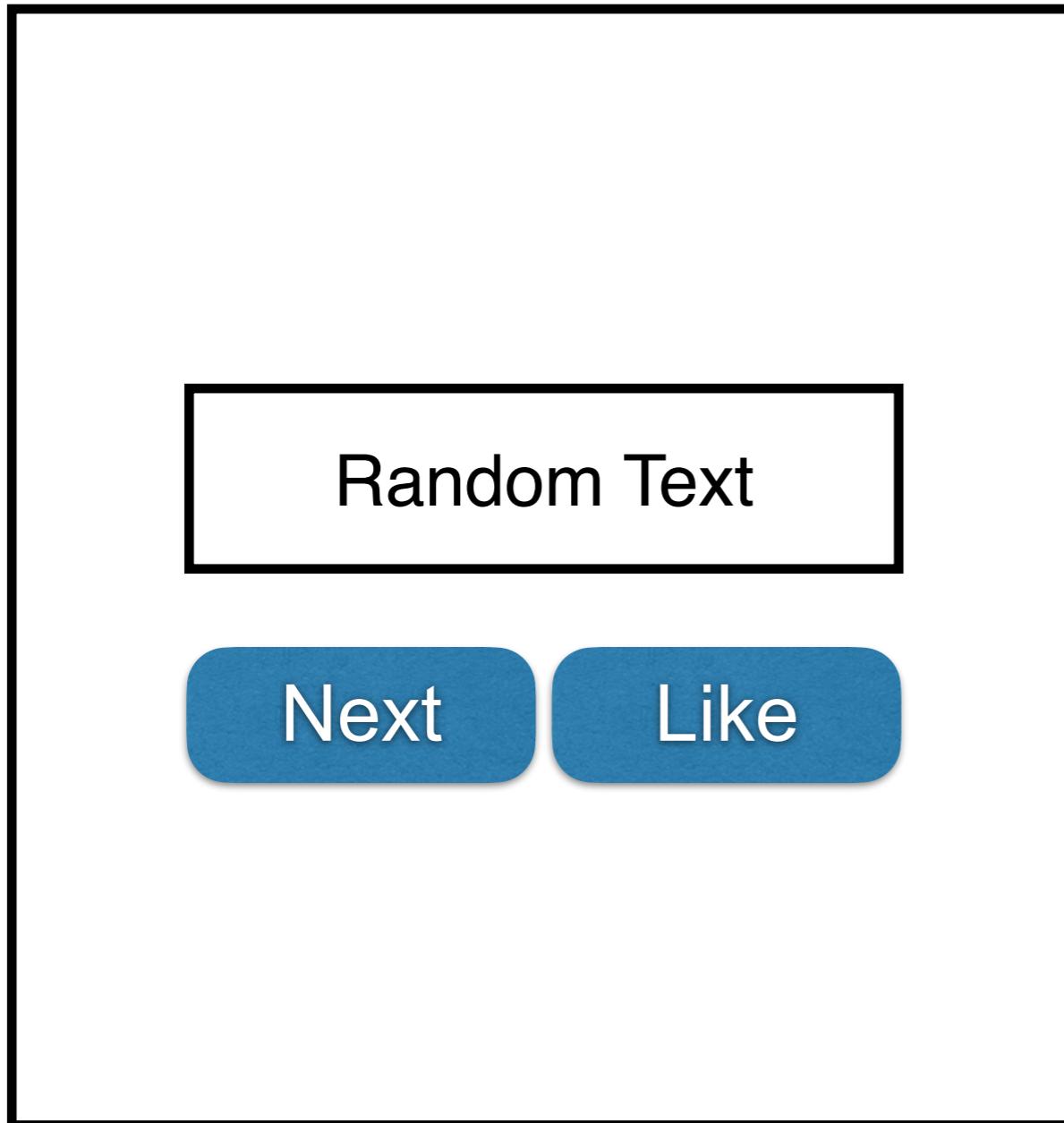
Like



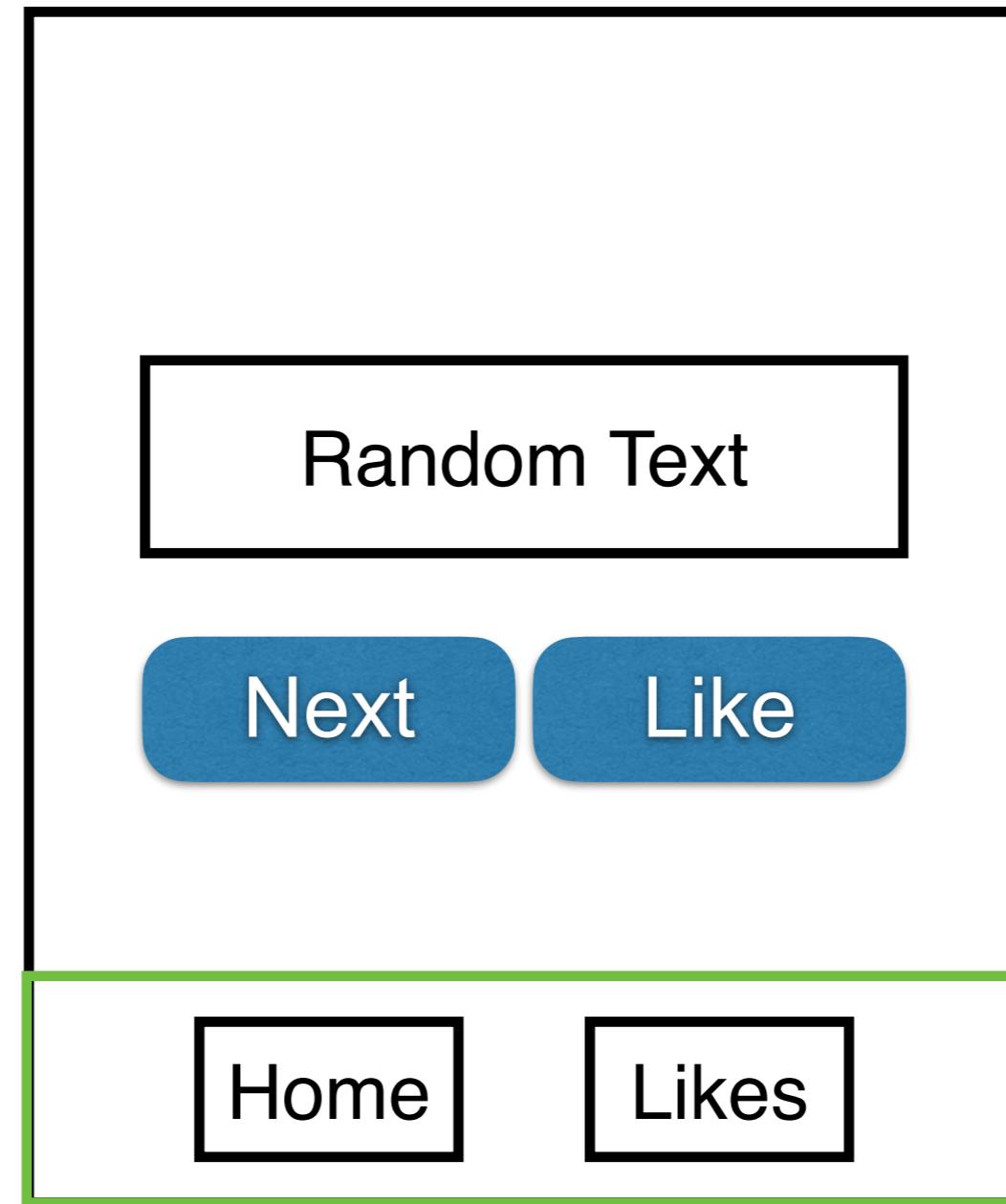
Design your app structure



Widget tree



Add more pages

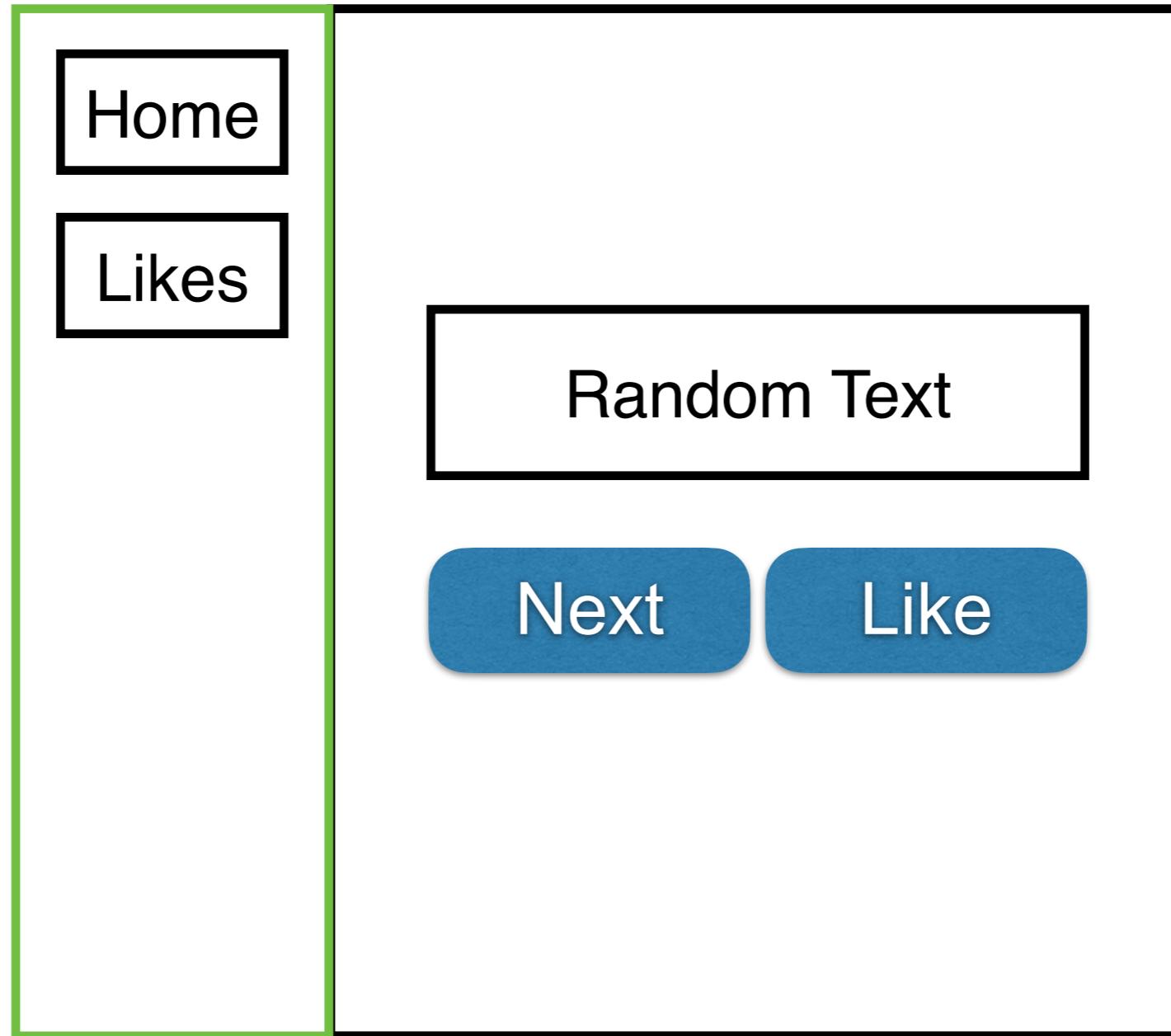


BottomNavigationBar

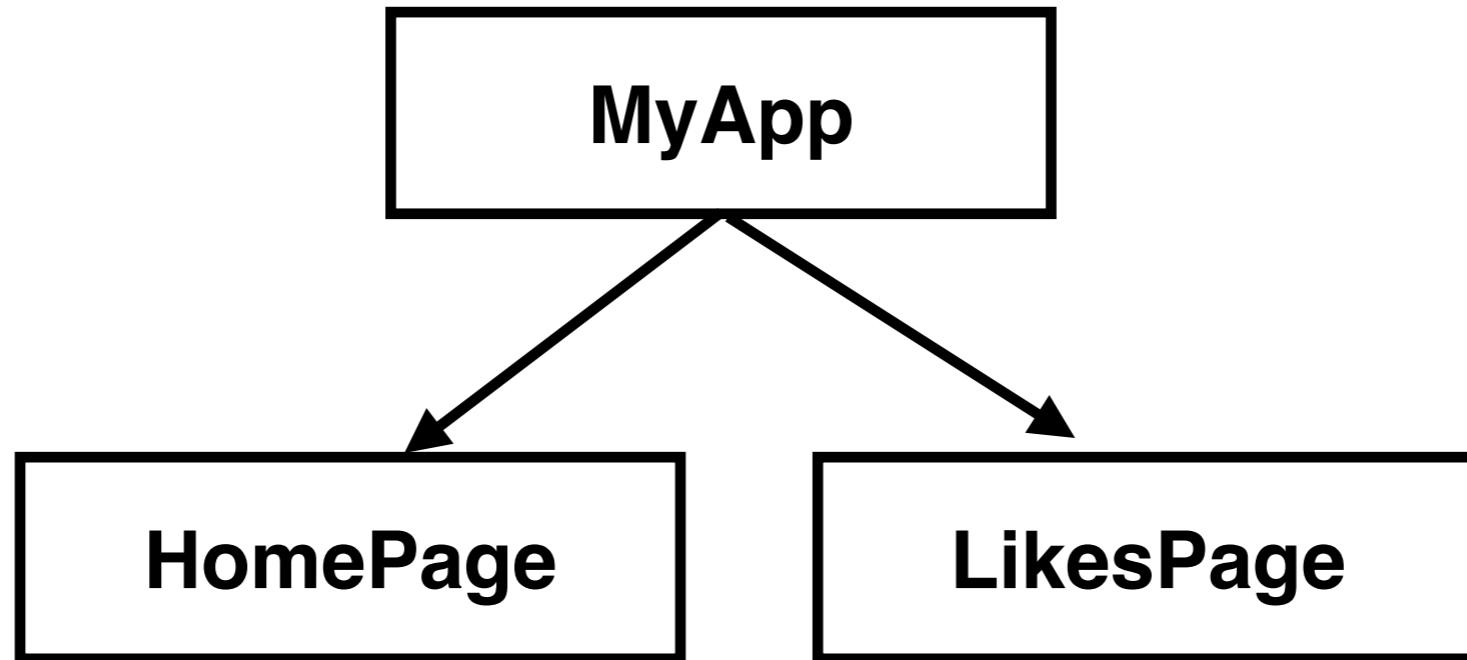


Responsive !!

NavigationRail



Design your app structure



<https://docs.flutter.dev/ui/navigation>



Flutter Design UI Challenges

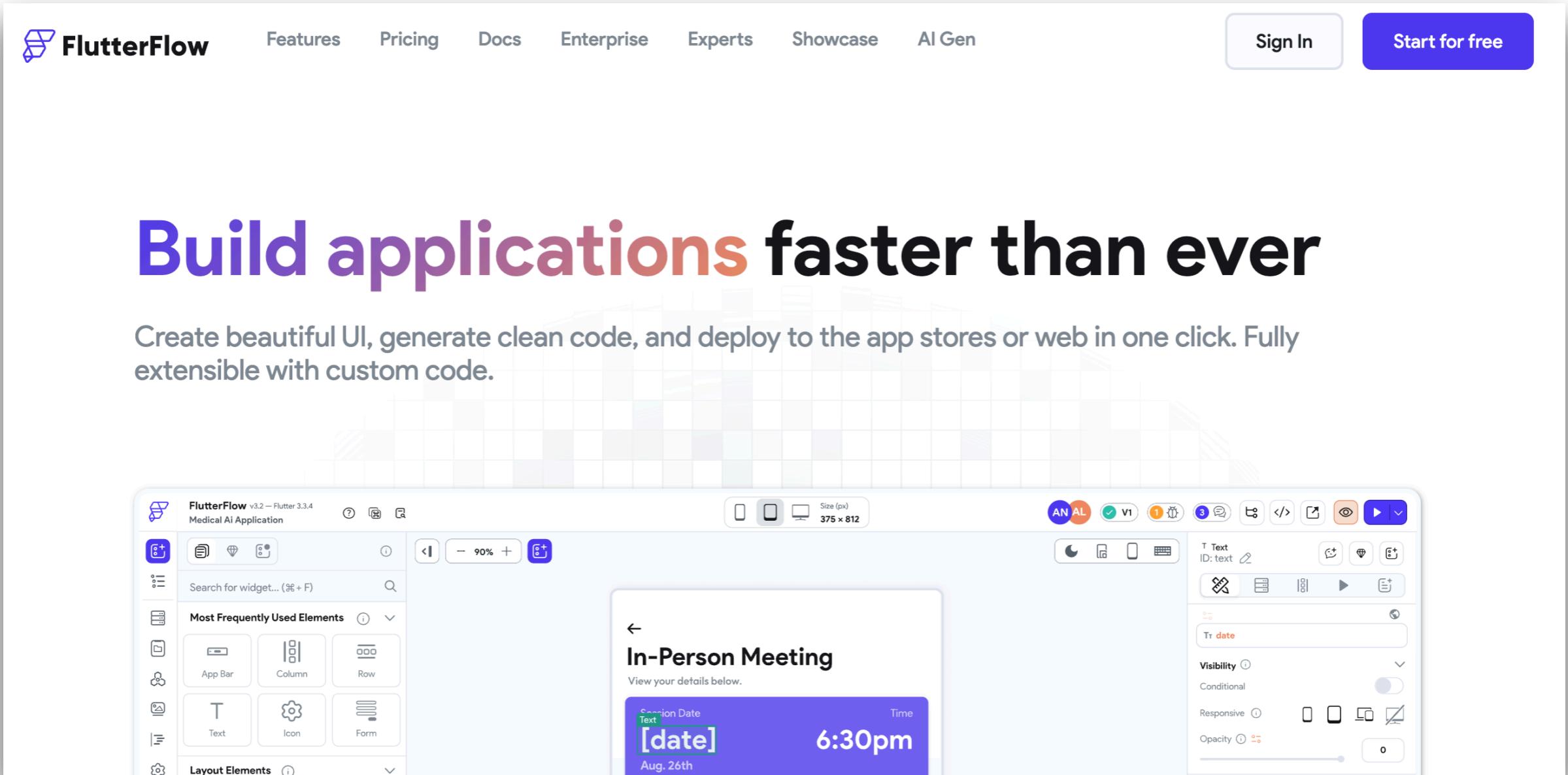
https://github.com/lohanidamodar/flutter_ui_challenges



Design UIs Tools



FlutterFlow



The screenshot shows the FlutterFlow web-based UI builder. At the top, there's a navigation bar with links for Features, Pricing, Docs, Enterprise, Experts, Showcase, AI Gen, Sign In, and Start for free. The main heading "Build applications faster than ever" is displayed in large, bold, purple and red text. Below it, a subtext reads: "Create beautiful UI, generate clean code, and deploy to the app stores or web in one click. Fully extensible with custom code." A large grid-based workspace shows a card component titled "In-Person Meeting". The card has a subtitle "View your details below.", a date section with "Selection Date [date] Aug. 26th", and a time section with "Time 6:30pm". The right side of the screen displays the FlutterFlow configuration sidebar, which includes sections for Text (ID: text), Tr date, Visibility (Conditional, Responsive), and Opacity. The sidebar also shows preview icons for mobile and desktop devices.

<https://flutterflow.io/>



Flutter 101

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Material Theme Builder

Material Theme Builder

Dynamic Custom Export

Visualize dynamic color

Derived from a user's wallpaper selection to create 5 key colors through dynamic color. Select a wallpaper or add your own to see user generated color in action.

Learn more about dynamic color.

The interface features a top navigation bar with tabs for 'Dynamic' (selected), 'Custom', 'Export', and icons for dark mode and help. Below is a large title 'Visualize dynamic color'. A text block explains how it generates colors from a wallpaper. A 'Learn more about dynamic color.' link is present. On the left, there are four circular thumbnails: two with foliage (brown and green), one with a blue gradient and flowers, and one with yellow and orange abstract shapes. To the right are several UI components: a horizontal slider, two 'Textfield' boxes, a row of buttons ('Assist', 'i Assist', 'Filter', 'Filter', '? Suggestion'), a row of plus-sign buttons ('Button', '+ Button', 'Button', '+ Button', 'Button'), and a row of colored buttons ('+'). At the bottom are platform-specific tabs: 'Android' (checked), 'Windows', 'Web', and 'Linux'.

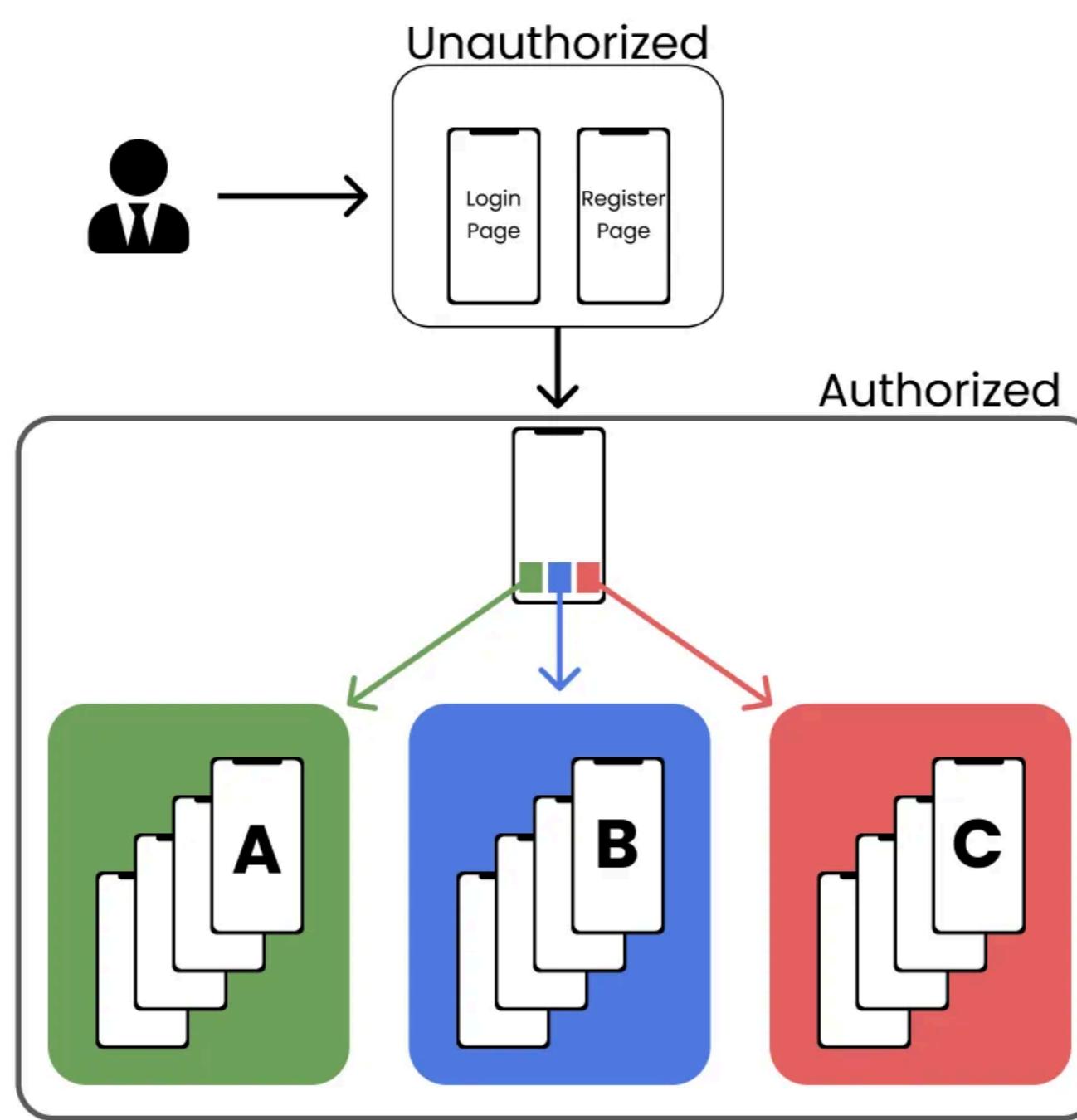
<https://material-foundation.github.io/material-theme-builder/>



Navigation/Router in app



Why we need navigation ?

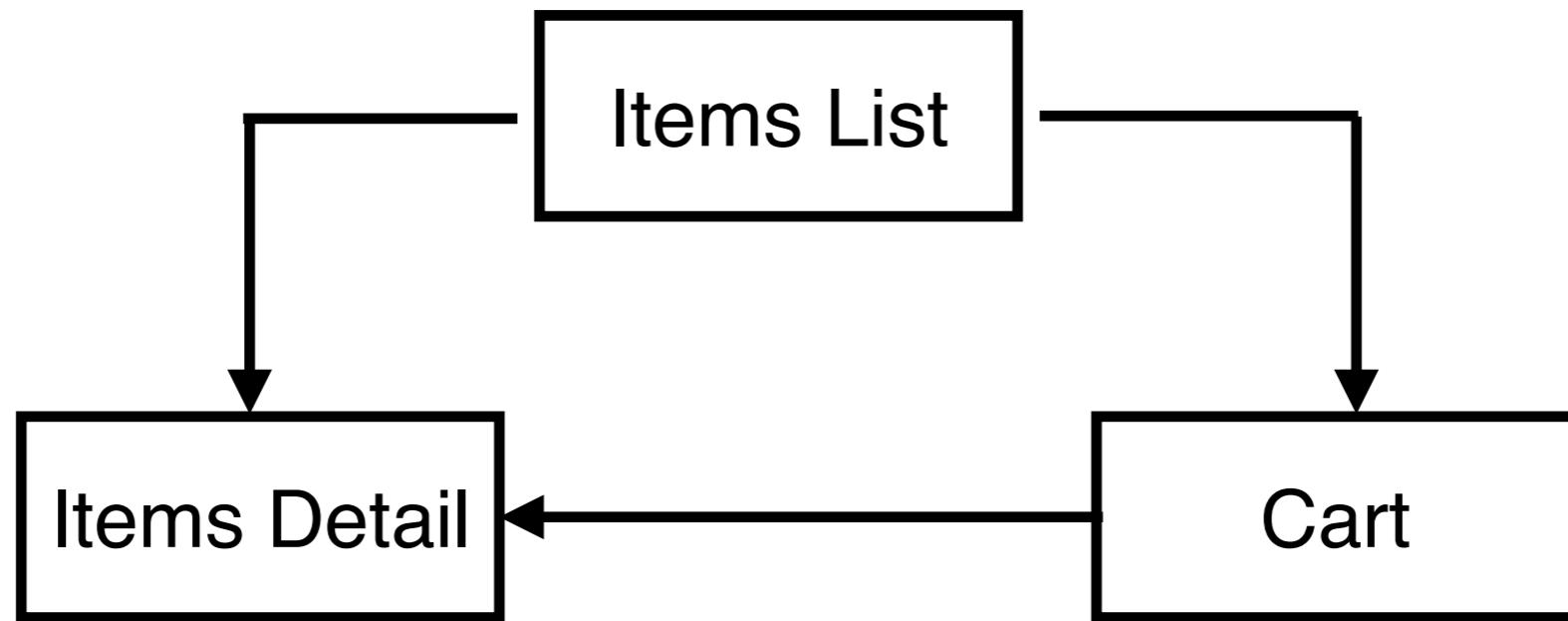


How to manage navigation ?

Navigation 2.0
Go Router
Fluro
Get



Workshop



Simple State management



State management

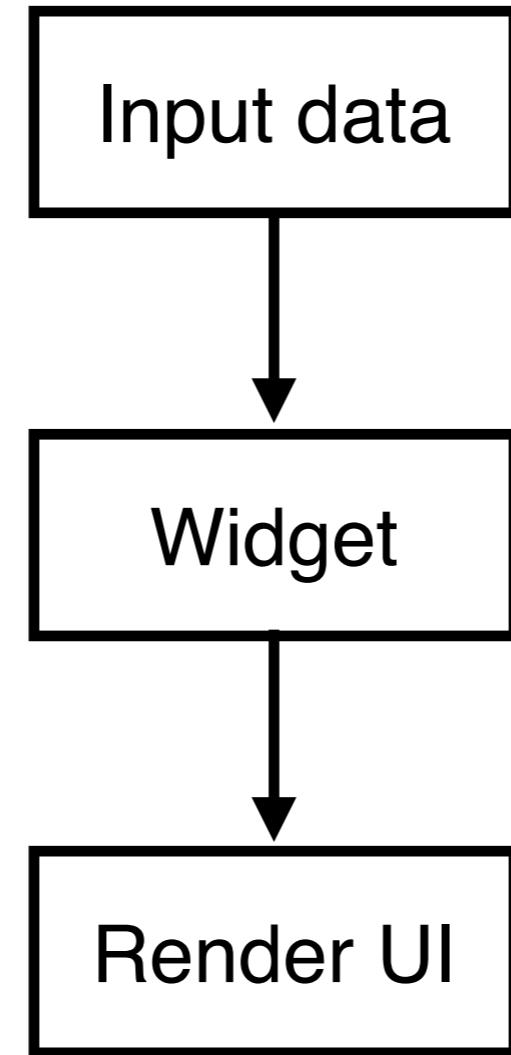
UI = f(state)

UI = layout of screen
State = application state

Any interactions that reflects UI changes

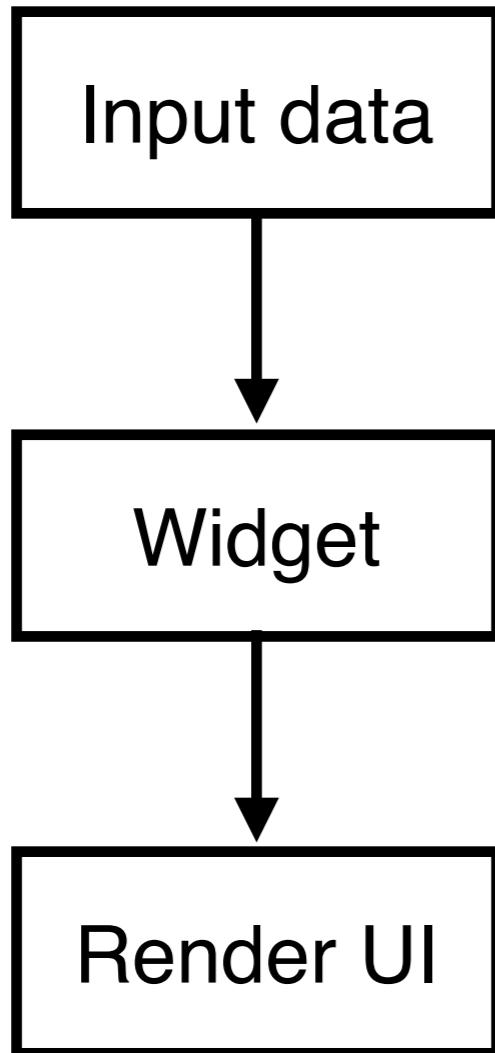


State management



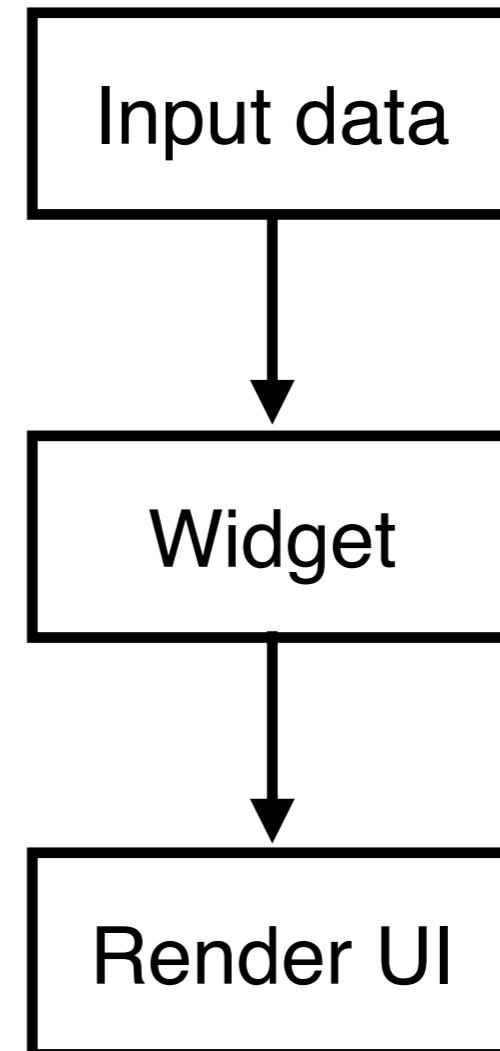
State management

Stateless widget



Input changes

Stateful widget



Input and local state changes



State ?

Form validation

Network request

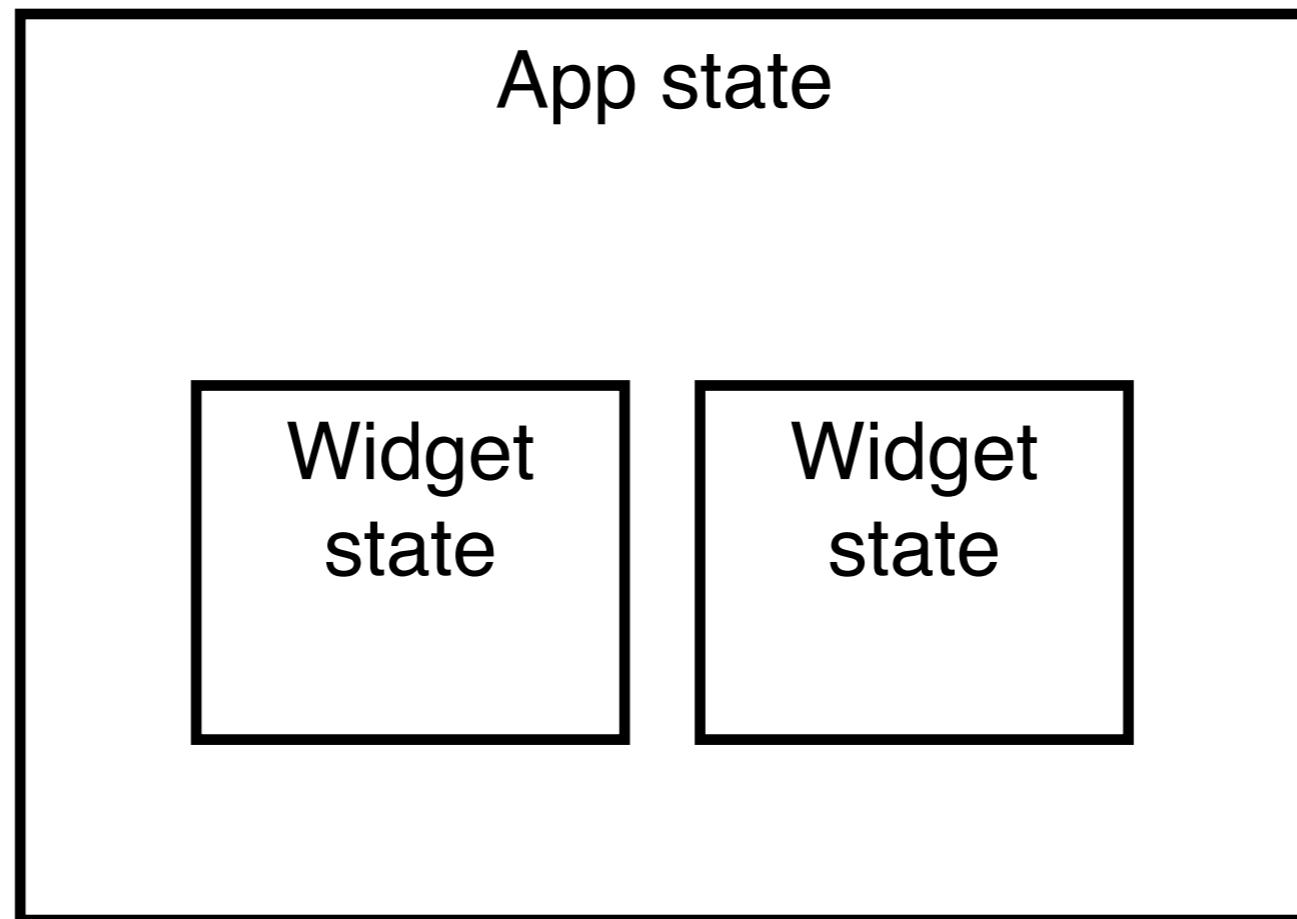
Database access

Data changes

Screen rotation



State in App ?



<https://docs.flutter.dev/data-and-backend/state-mgmt/ephemeral-vs-app>



Ephemeral state

Local state or UI state
Use Stateful Widget
Store in memory

Widget
state

Current
data

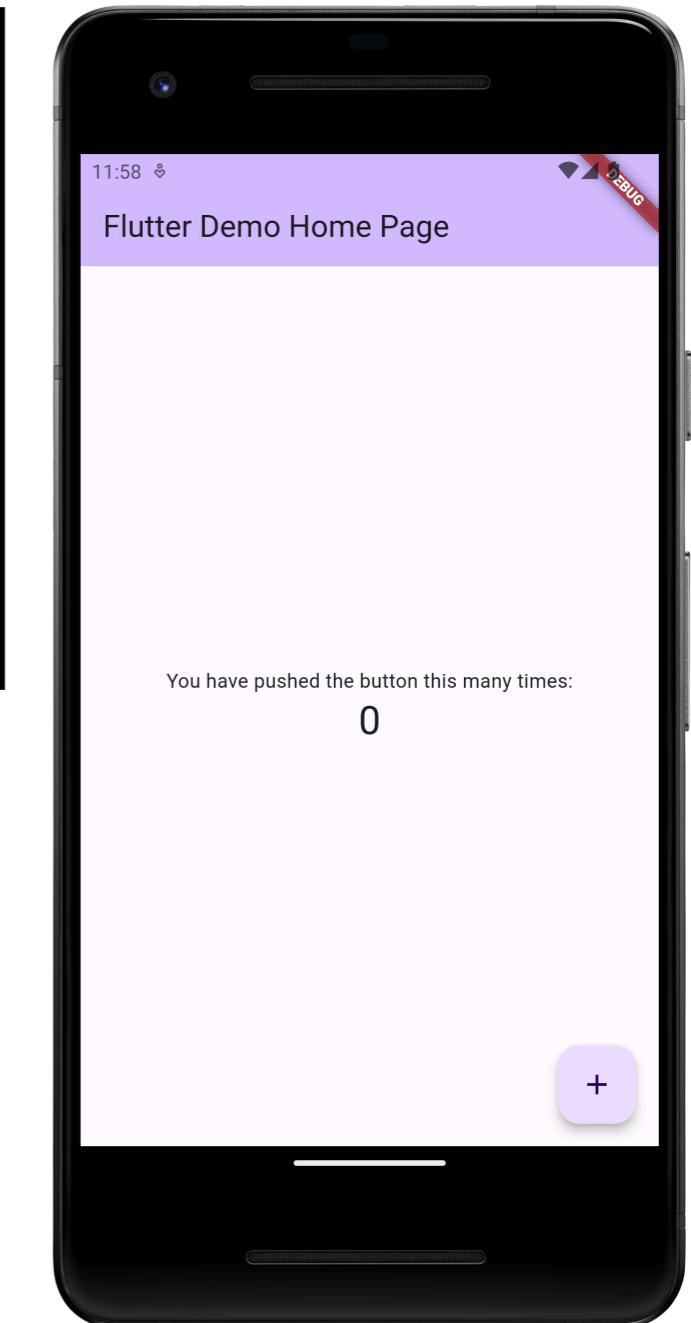
Current
progress

Current
tab



Stateful widget (1)

```
class MyHomePage extends StatefulWidget {  
  
  const MyHomePage({super.key, required this.title});  
  
  final String title;  
  
  @override  
  State<MyHomePage> createState() => _MyHomePageState();  
  
}
```

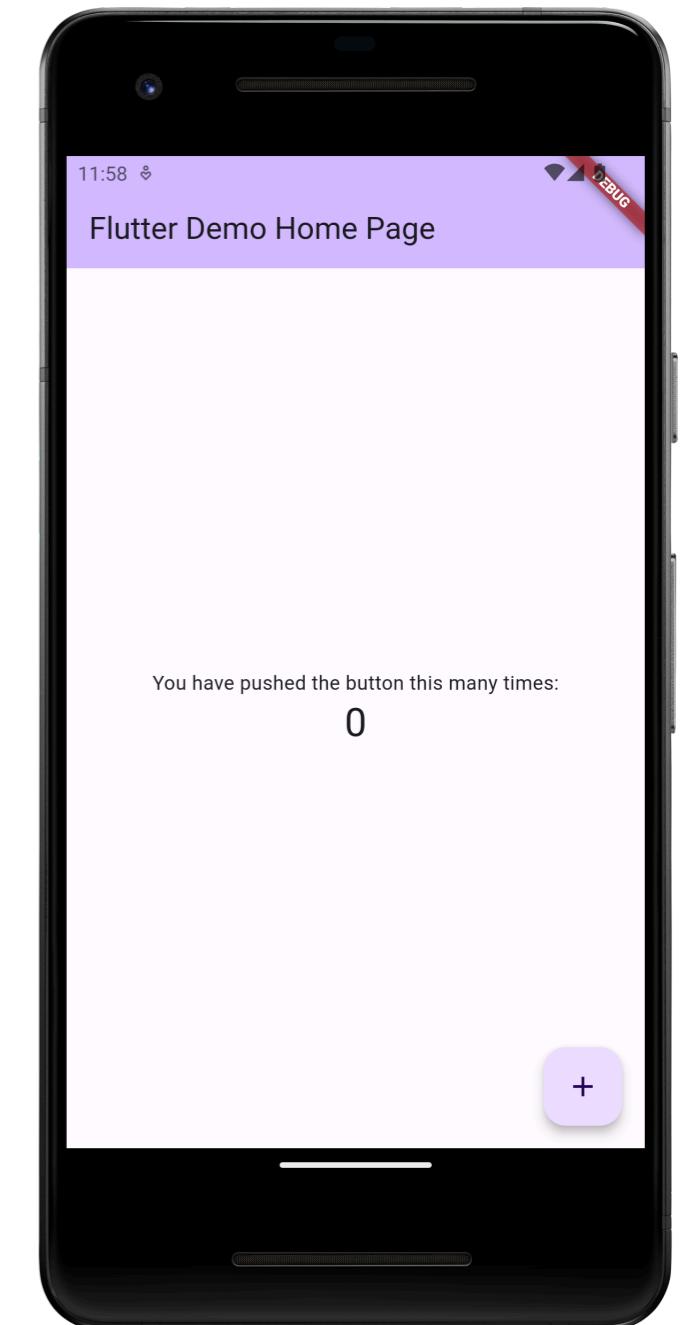


Stateful widget (2)

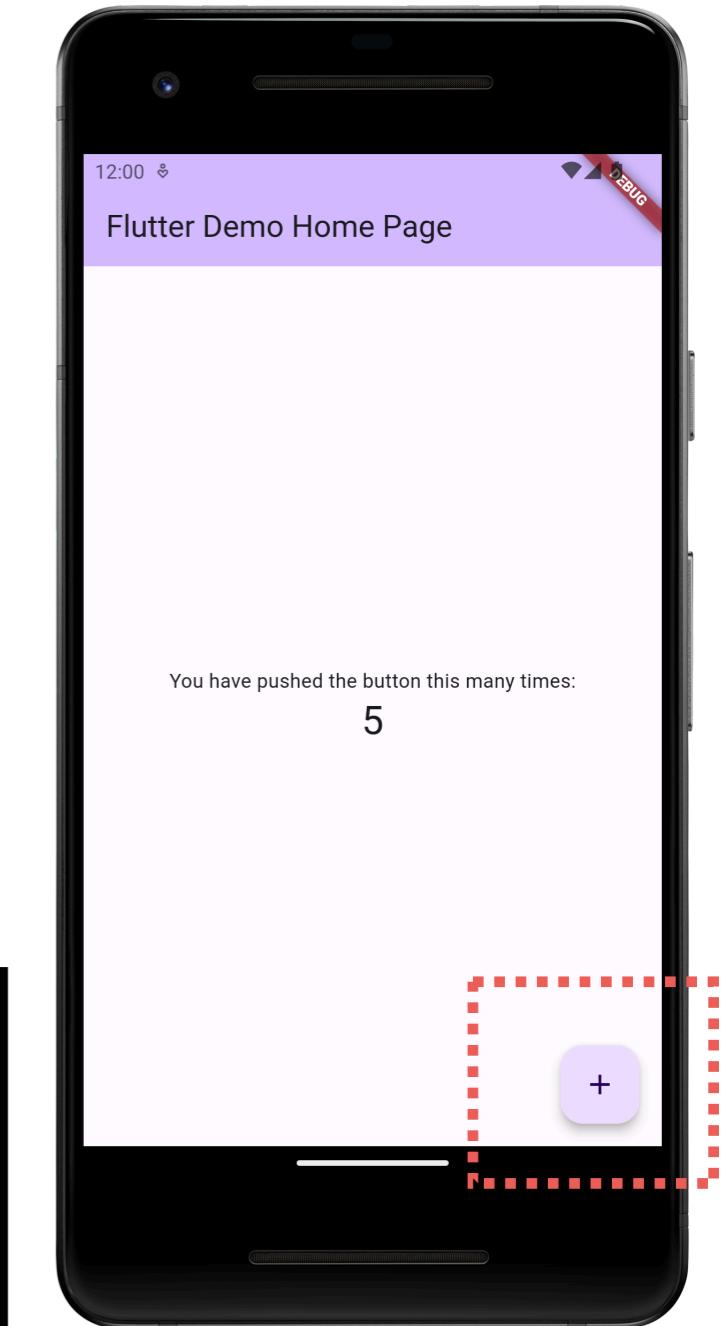
```
class _MyHomePageState extends State<MyHomePage> {
    int _counter = 0;

    void _incrementCounter() {
        setState(() {
            _counter++;
        });
    }

    @override
    Widget build(BuildContext context) {
        // Draw UI
    }
}
```



Stateful widget (3)



```
floatingActionButton: FloatingActionButton(  
    onPressed: _incrementCounter,  
    tooltip: 'Increment',  
    child: const Icon(Icons.add),  
) ,
```

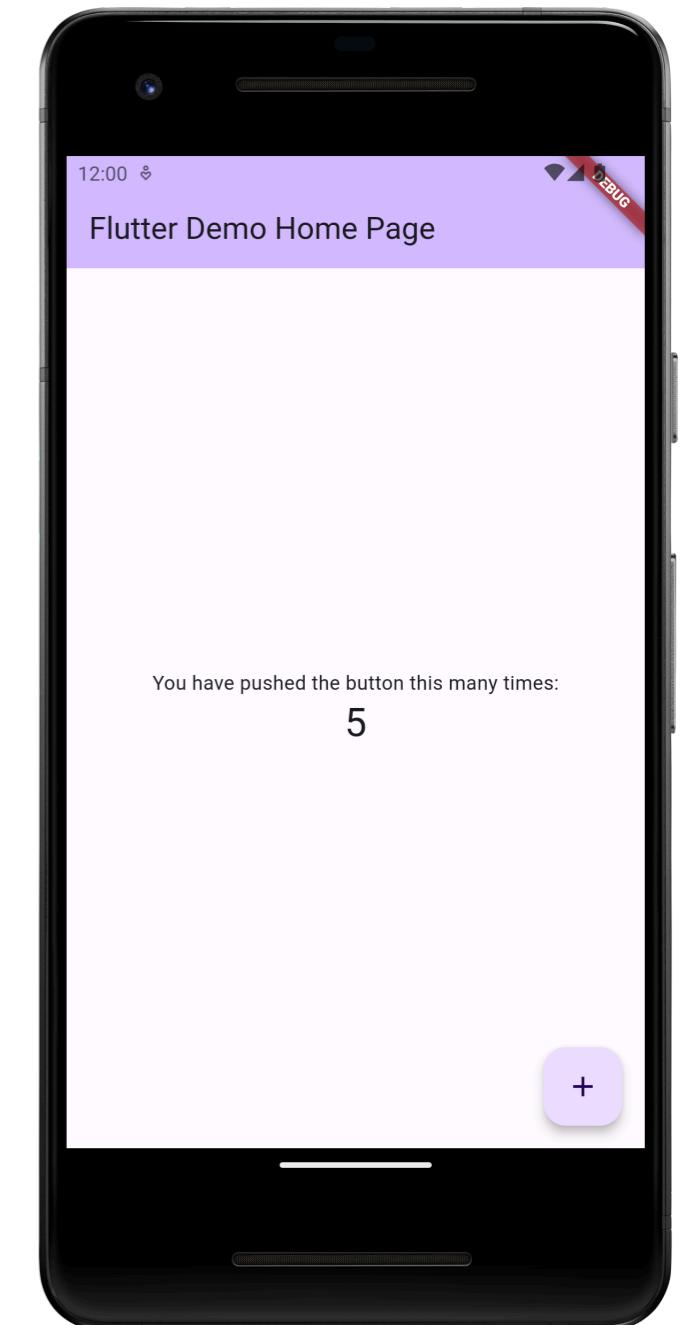


Stateful widget (4)

```
class _MyHomePageState extends State<MyHomePage> {
    int _counter = 0;

    void _incrementCounter() {
        setState(() {
            _counter++;
        });
    }

    @override
    Widget build(BuildContext context) {
        // Draw UI
    }
}
```

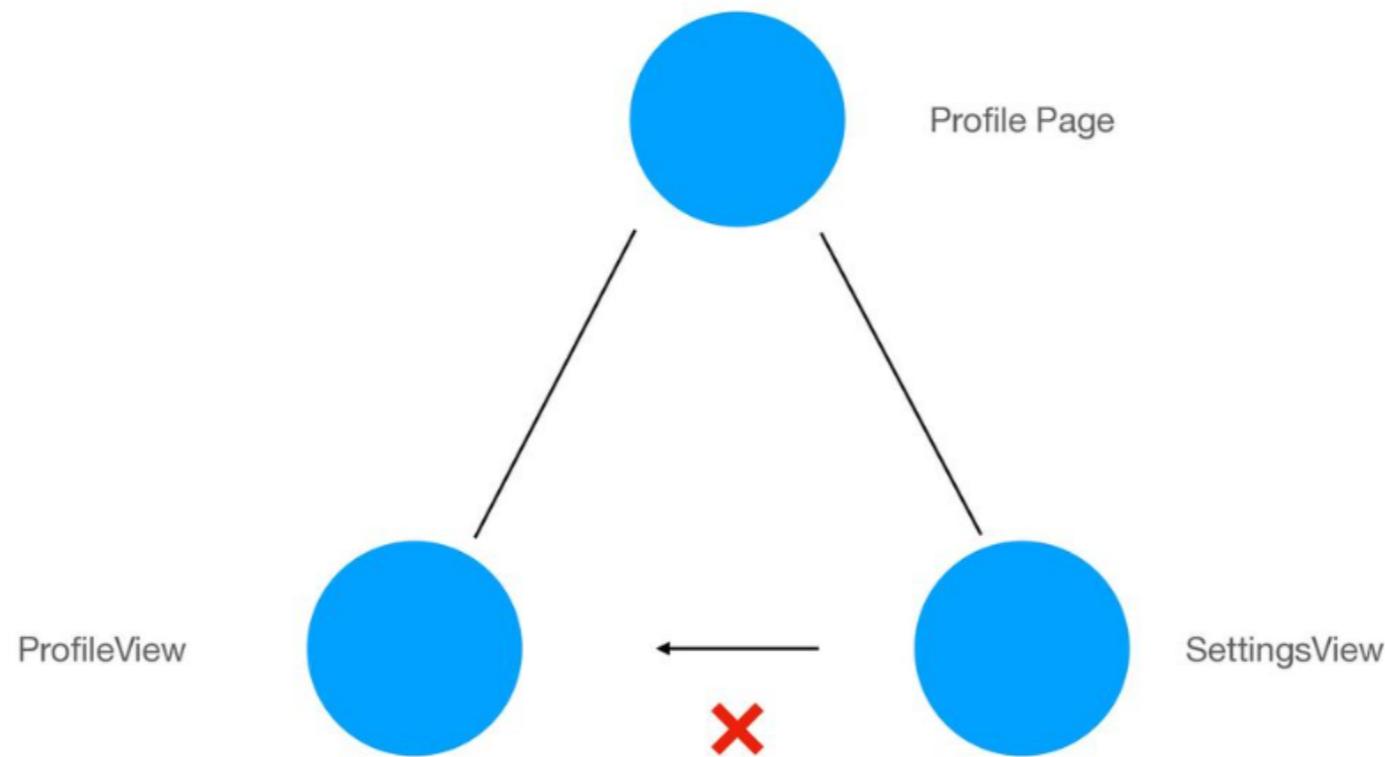


Cons of setState(){} ()

Logic inside presentation layer

Unnecessary widget rebuilds

Hard to maintain state across app level



App state

Shared state across many parts of your app

User
preference

Login
information

Notification

Shopping cart

Read/unread
of articles



How to keep App state ?

Shared preferences in Android app

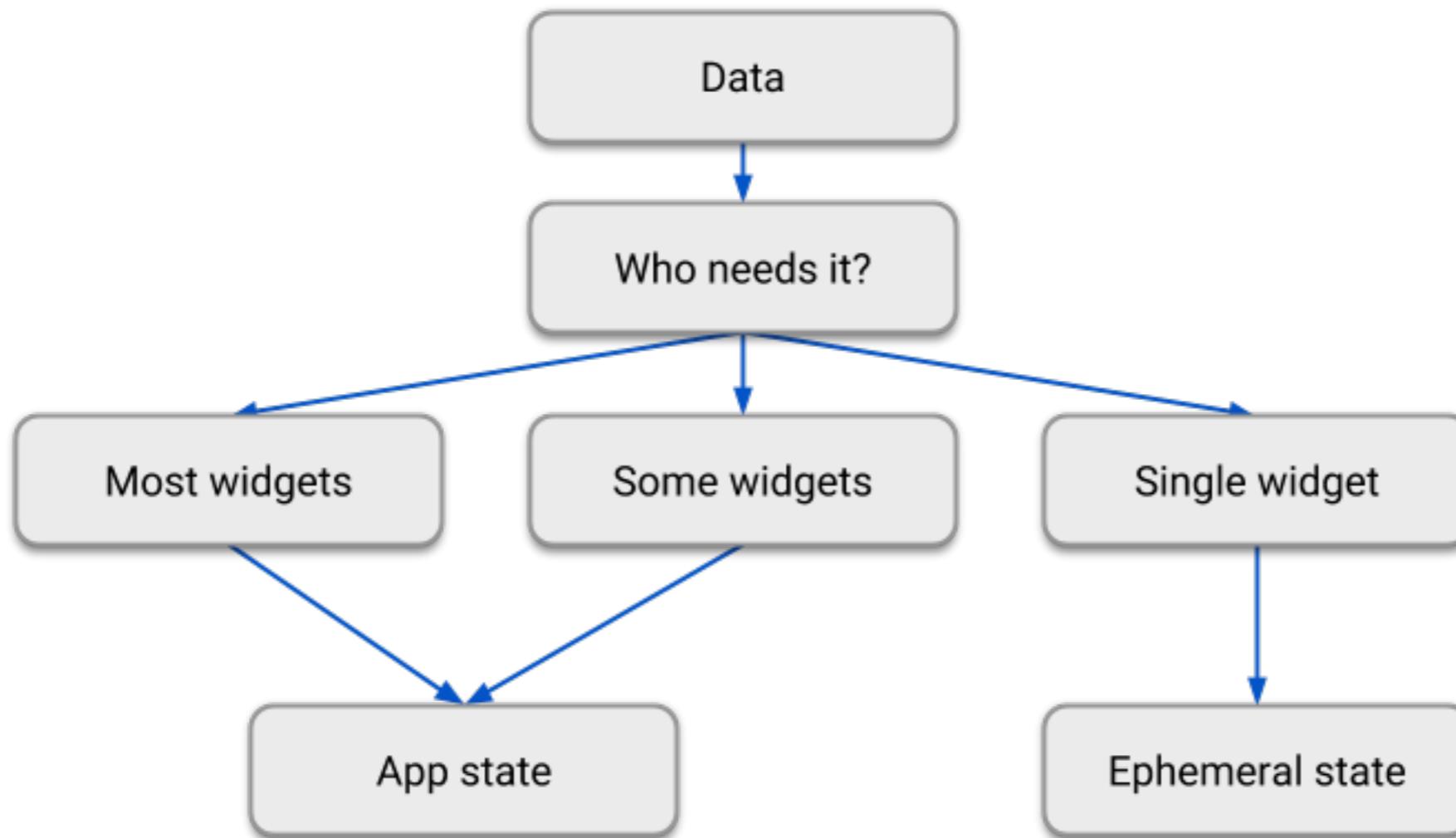
Sqlite is database in mobile

Local storage in web browser

NSUserDefaults in iOS app



State in App ?



State management libraries !!

Provider

setState

Riverpod

MobX

BLoC

GetX

Redux

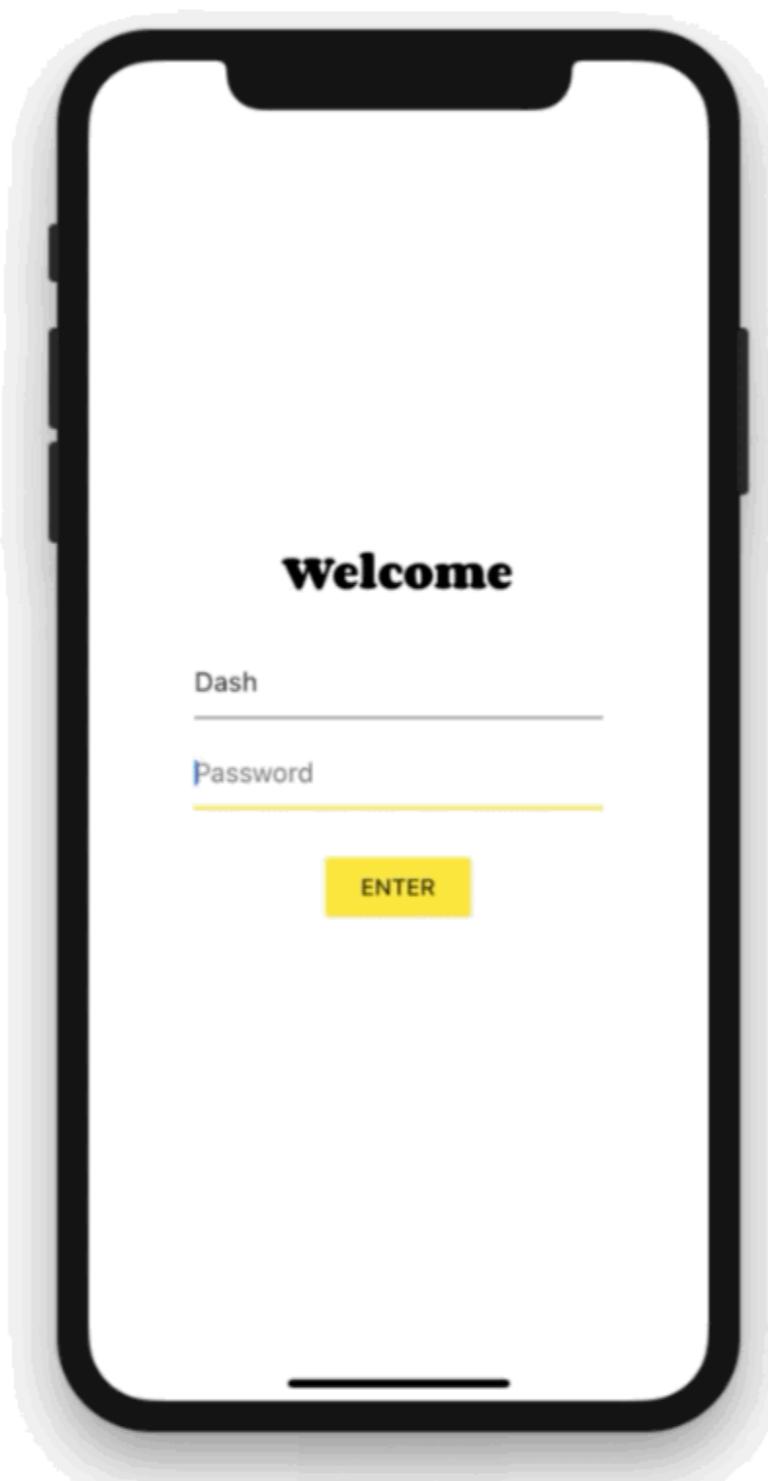
Reactive
Value

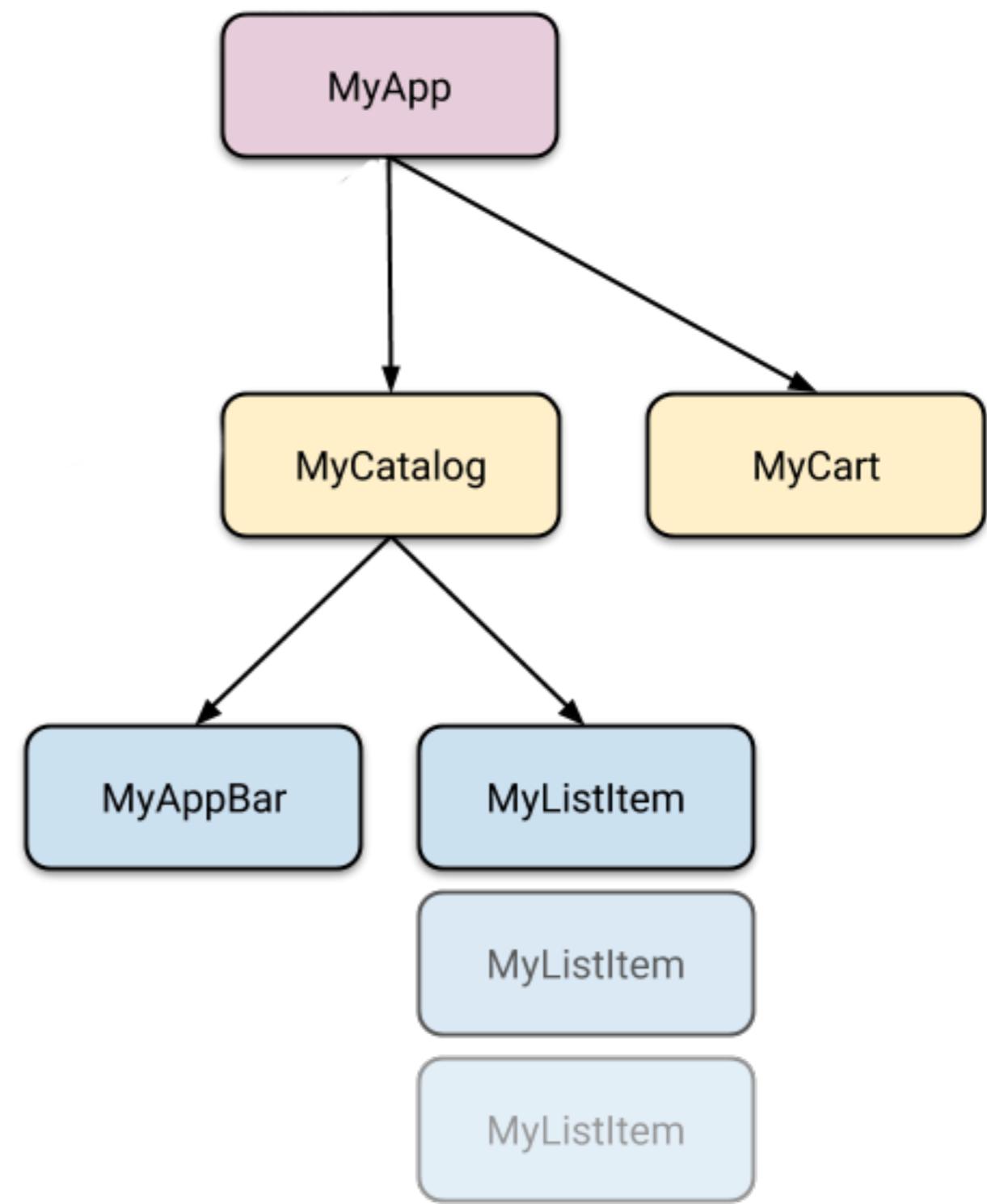
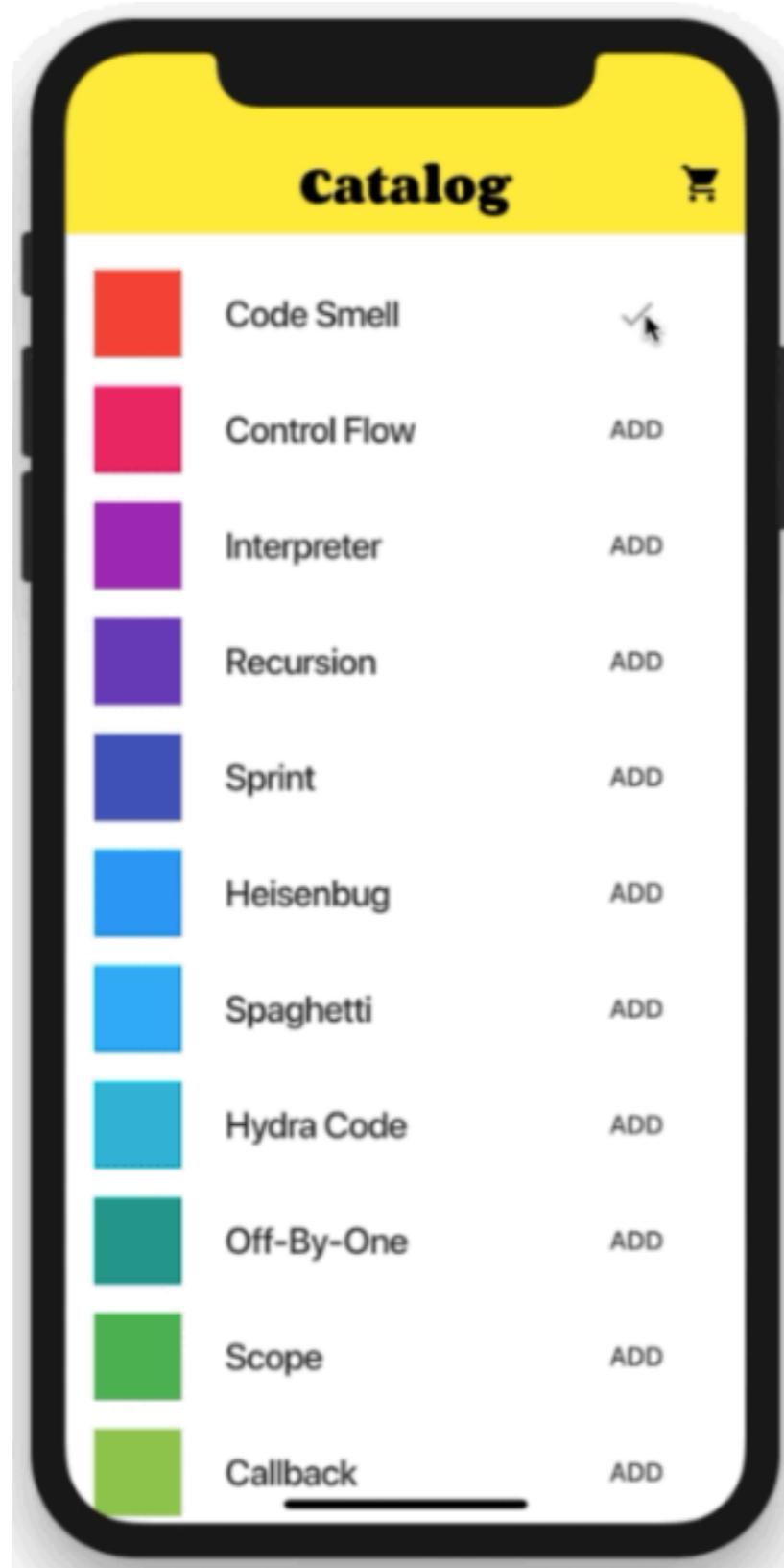


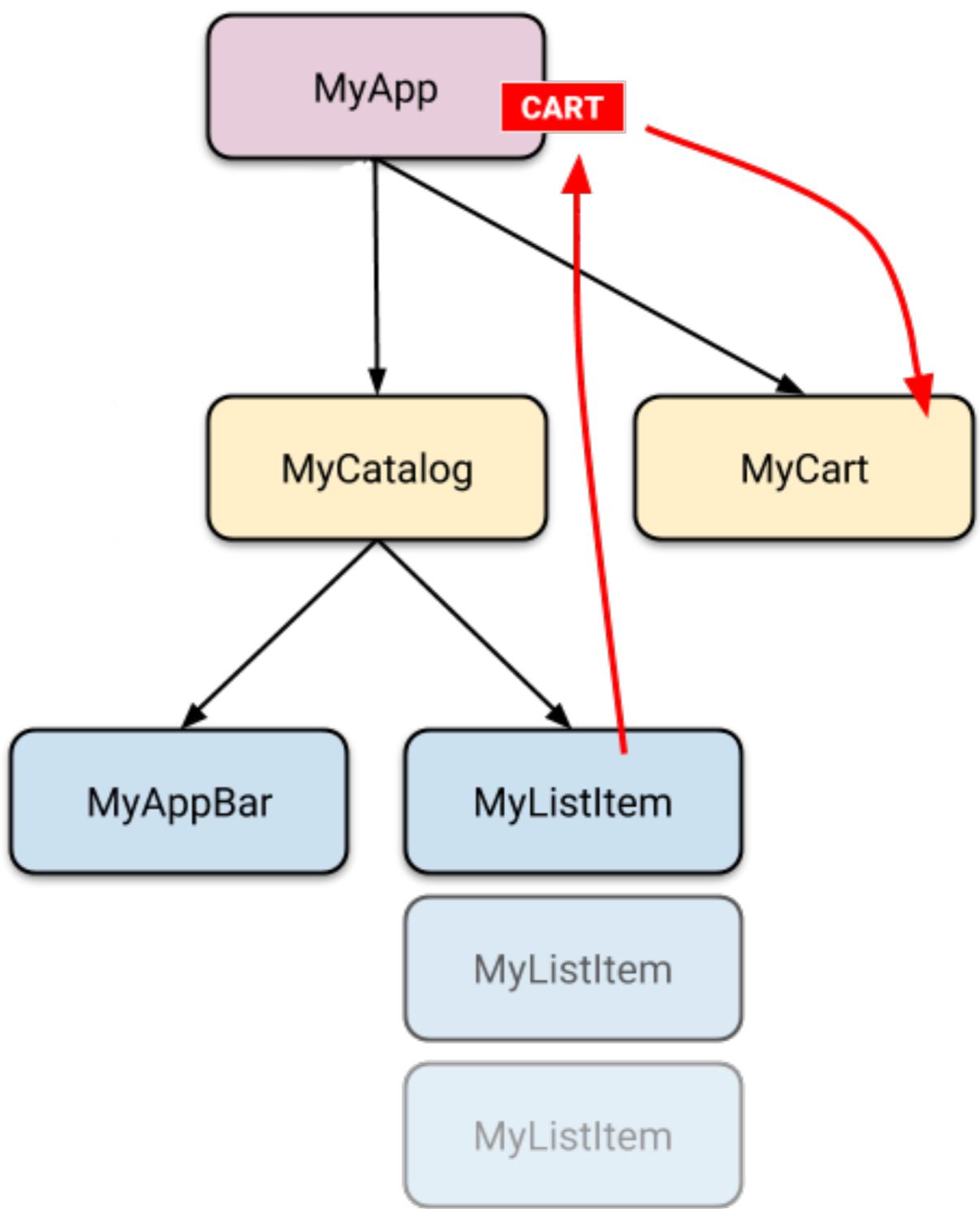
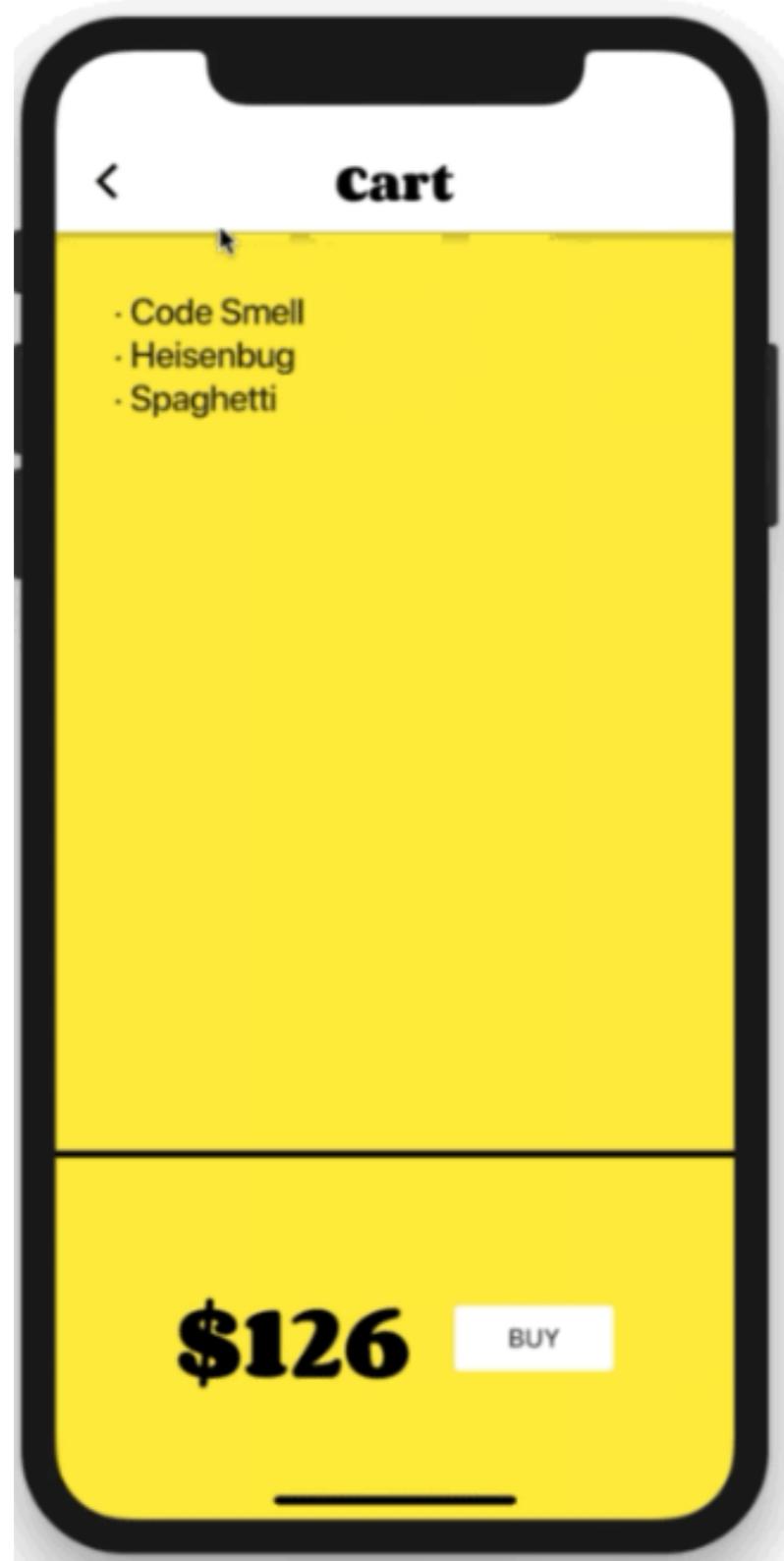
Workshop with State management

<https://docs.flutter.dev/data-and-backend/state-mgmt/simple>









Try to convert to GetX !!



<https://pub.dev/packages/get>



Working with External system



External system

File system on mobile (dart:io package)

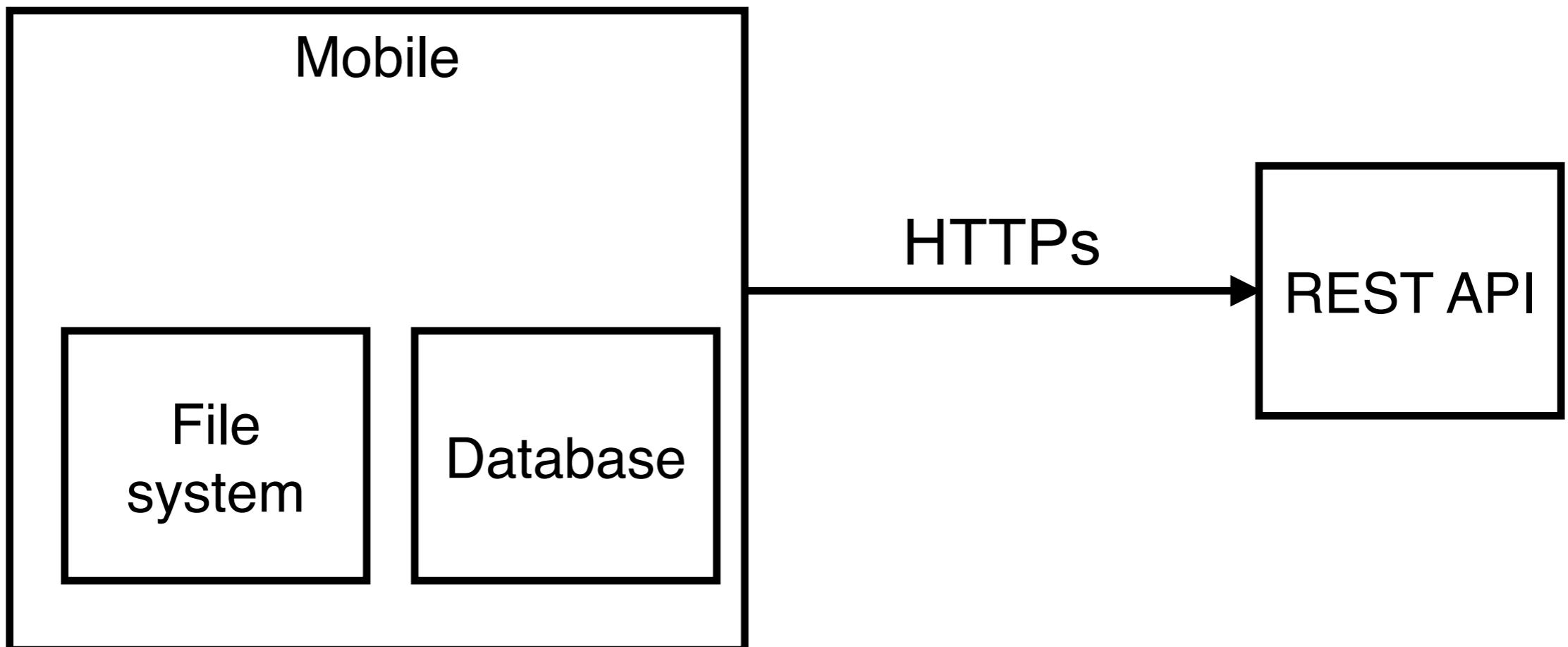
Key-value on mobile

Database on mobile (sqlite)

REST APIs via HTTPs



External system



Key-value data

Platform	Location
Android	SharedPreferences
iOS	NSUserDefaults
Linux	XDG_DATA_HOME directory
MacOS	NSUserDefaults
Web	LocalStorage
Windows	Roaming AppData directory

https://pub.dev/packages/shared_preferences



Database (sqlite)

Store app's data on mobile
CRUD (create, read, update, delete)

<https://docs.flutter.dev/cookbook/persistence/sqlite>
<https://pub.dev/packages/sqflite>



Working with HTTPS

Send and receive data from REST APIs

HTTP
package

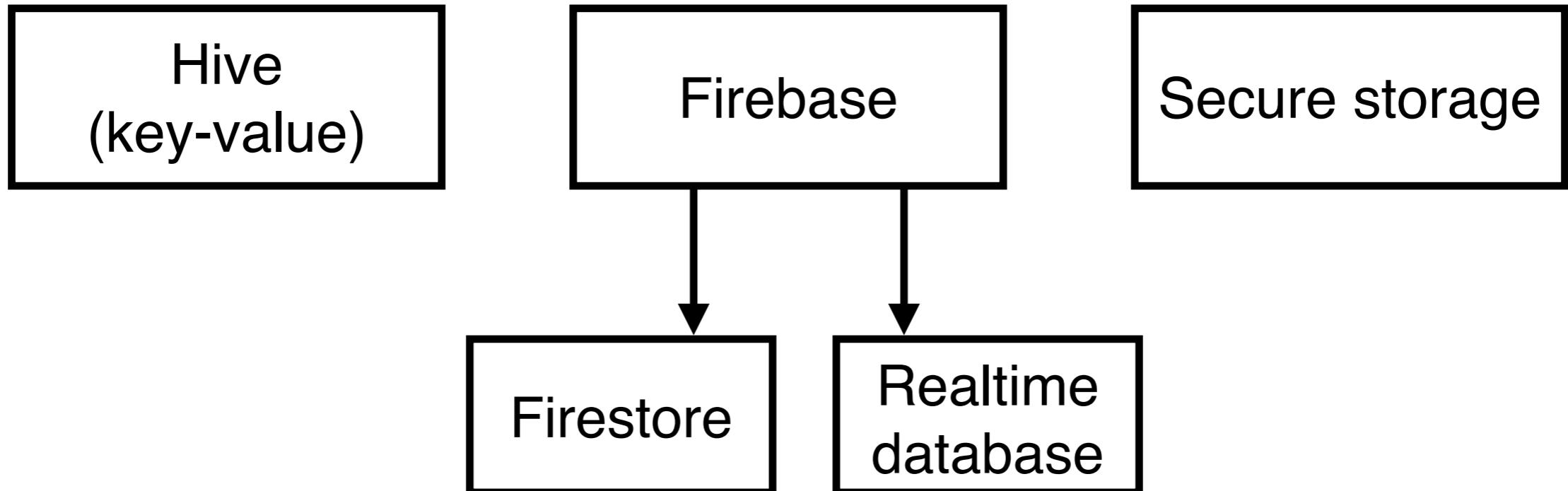
Dio
package

Retrofit
package

<https://docs.flutter.dev/cookbook/networking/send-data>



Other solutions



https://pub.dev/packages/flutter_secure_storage



Workshop with external system



Flutter Error Handling



Flutter Error Handling

Exceptions

Errors

Null reference errors

Asynchronous error



Basic of Error Handling

Try/catch

```
void divideNumbers(int numerator, int denominator) {  
    try {  
        double result = numerator / denominator;  
        print('Result of division: $result');  
    } catch (e) {  
        print('Error: $e');  
        print('Cannot perform the division.');  
    }  
}
```



Asynchronous Error !!

Working with database and external APIs

Future error

Single value

Stream error

Multiple value



Fetch data from APIs

```
Future<int> fetchUserData() async {
  await Future.delayed(Duration(seconds: 2)); // Simulating an async operation
  // Uncomment the line below to trigger an error
  // throw Exception('Error fetching user data');
  return 42; // Simulating a successful response
}
```

```
// Handling errors in a Future
fetchUserData()
  .then((value) => print('User data: $value'))
  .catchError((error) => print('Error fetching user data: $error'));
```



More Errors !!

Network error

Platform-specific

Global error with
Error widget

Report and logging

Firebase
crashlytics

<https://firebase.google.com/docs/crashlytics>



Best Practices

Use-friendly error message

Try/catch block use for exceptions

Future and stream use for async exceptions

Implement global error handling

Record error logging

Create custom error/exception

Test errors



Flutter Project Structure



Flutter Project Structure

Ensure team can follow a clear convention
Add feature in a consistent way

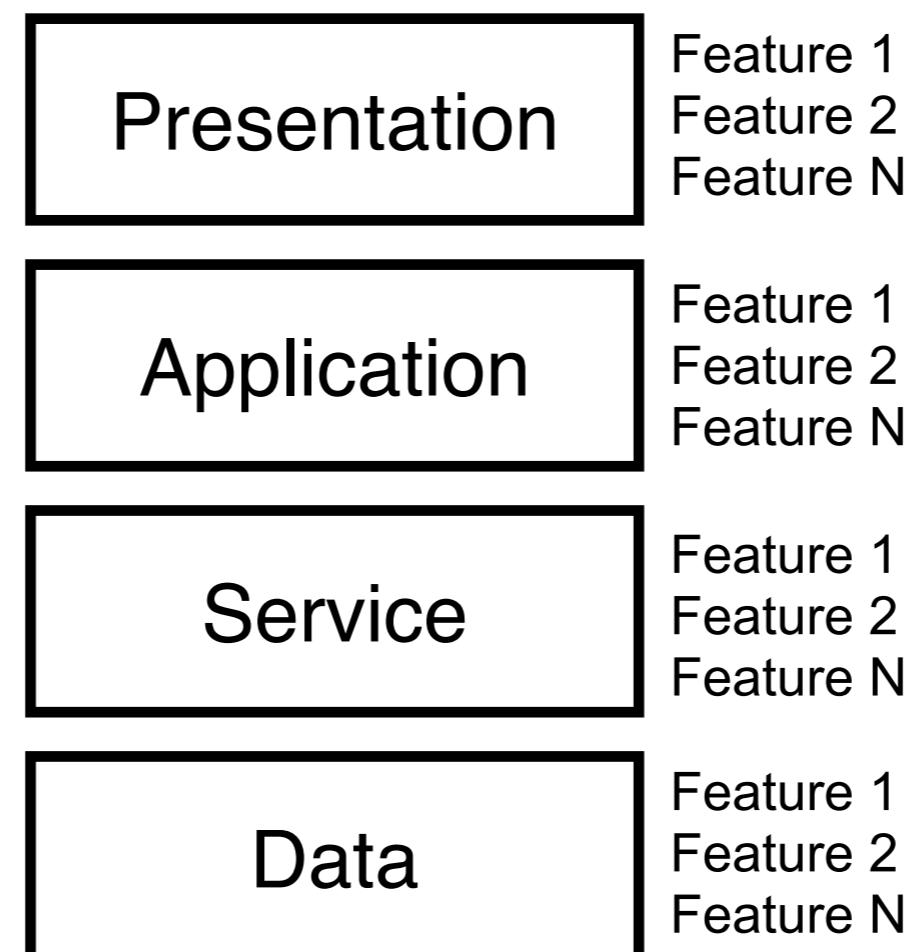
Layer-first

Feature-first

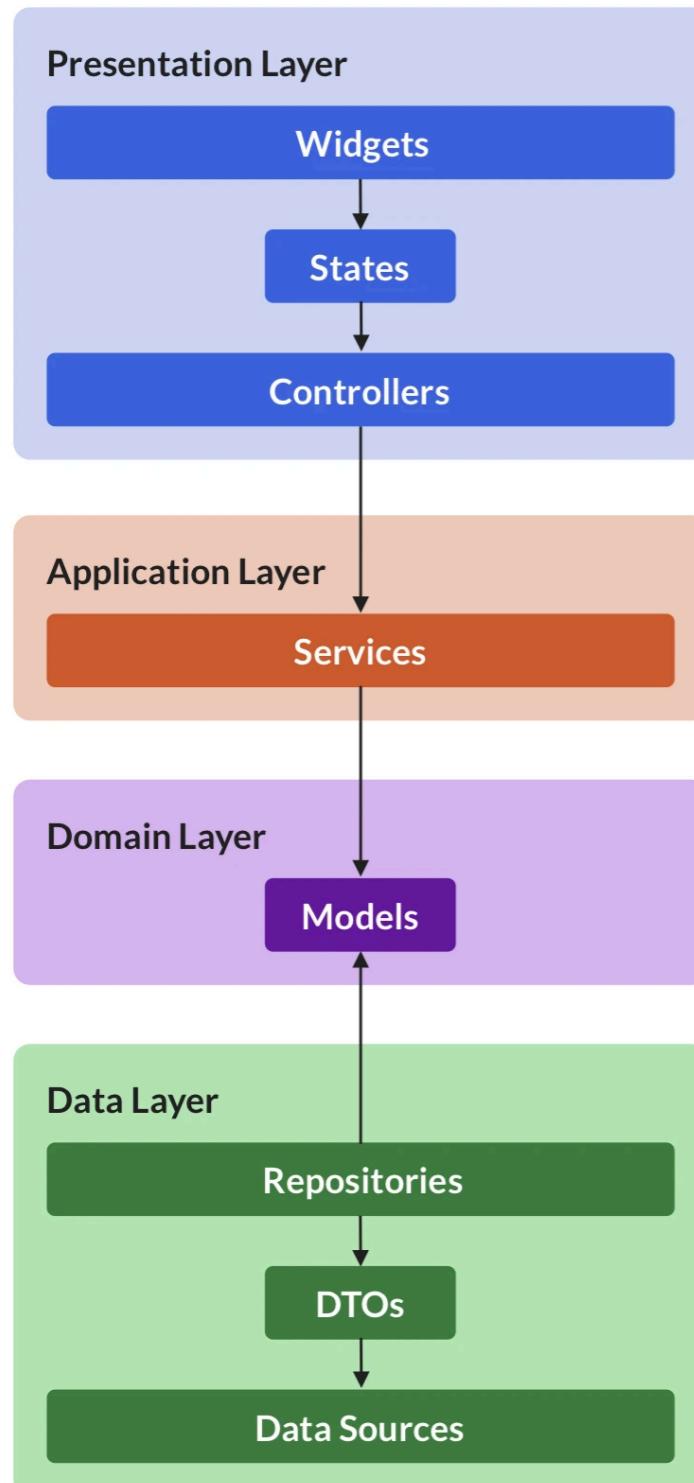


Layer-first

Separate by responsibility of work
Features inside layers
Suitable for small project



Riverpod App Architecture



Manage UI and state

Business logic and flow

Internal data model

Working with data (local and remote)

<https://riverpod.dev/>



Cons of Layer-first

Can't scale very well as the app grows
Keep jumping to different parts of project
Hard to add/remove features



Feature-first

Layers inside features

Separate by feature

Easy to see all files in each feature

Feature 1

presentation
application
service
data

Feature 2

presentation
application
service
data



Feature-first

Not about the UI !!

What is a feature ?

Account

Product list

Product detail

Sign in

Order list

Checkout



Feature-first ?

Features

Feature 1

Feature 2

Models

Service

Data



What is a feature ?

Not use see, but what use does

Authenticate

Manage
shopping cart

Checkout

Manage
product

Manage order

Manage user/
address

Review



Popular project structures

MVC (Model View Controller)

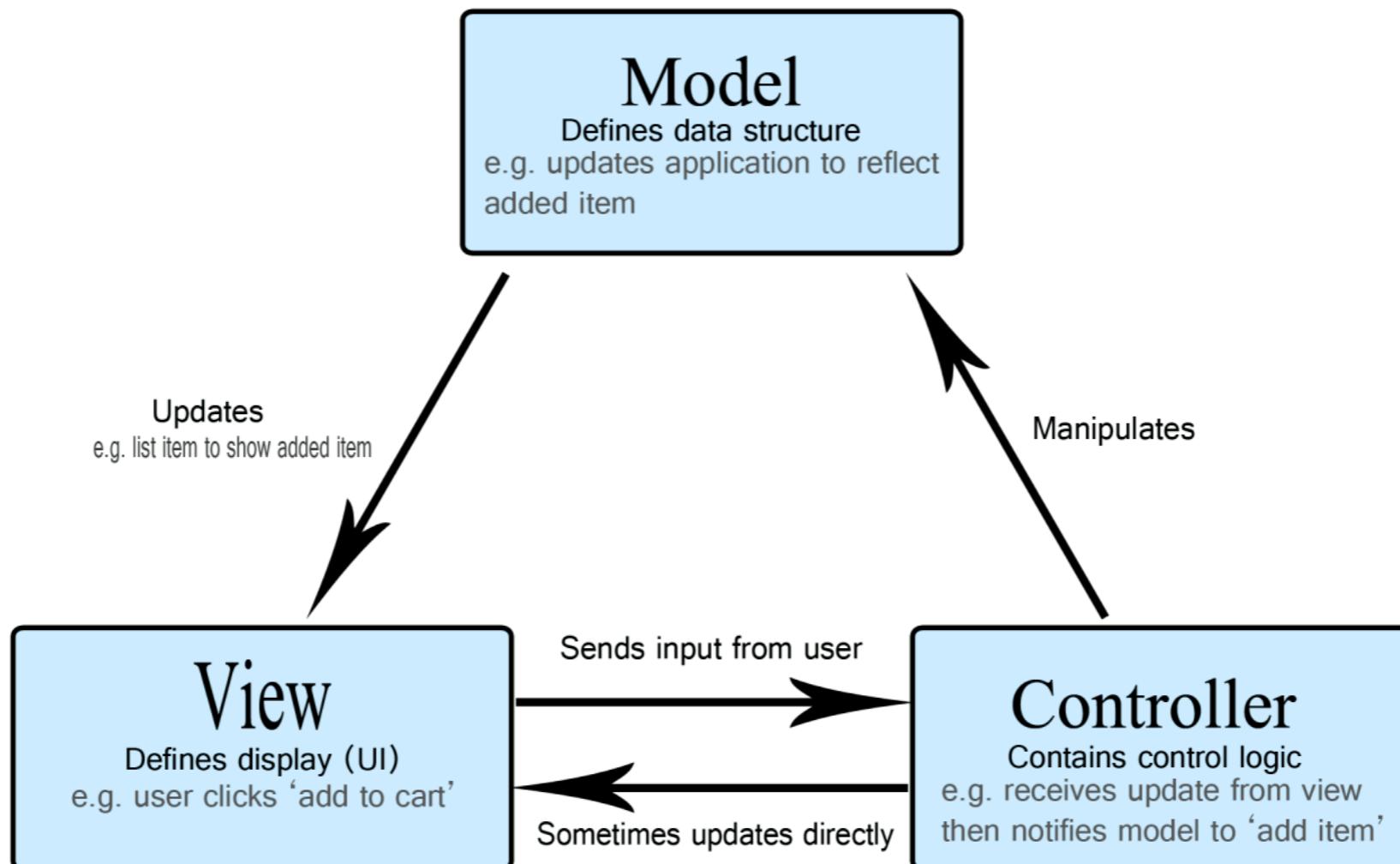
MVP (Model View Presenter)

MVVM (Model View ViewModel)

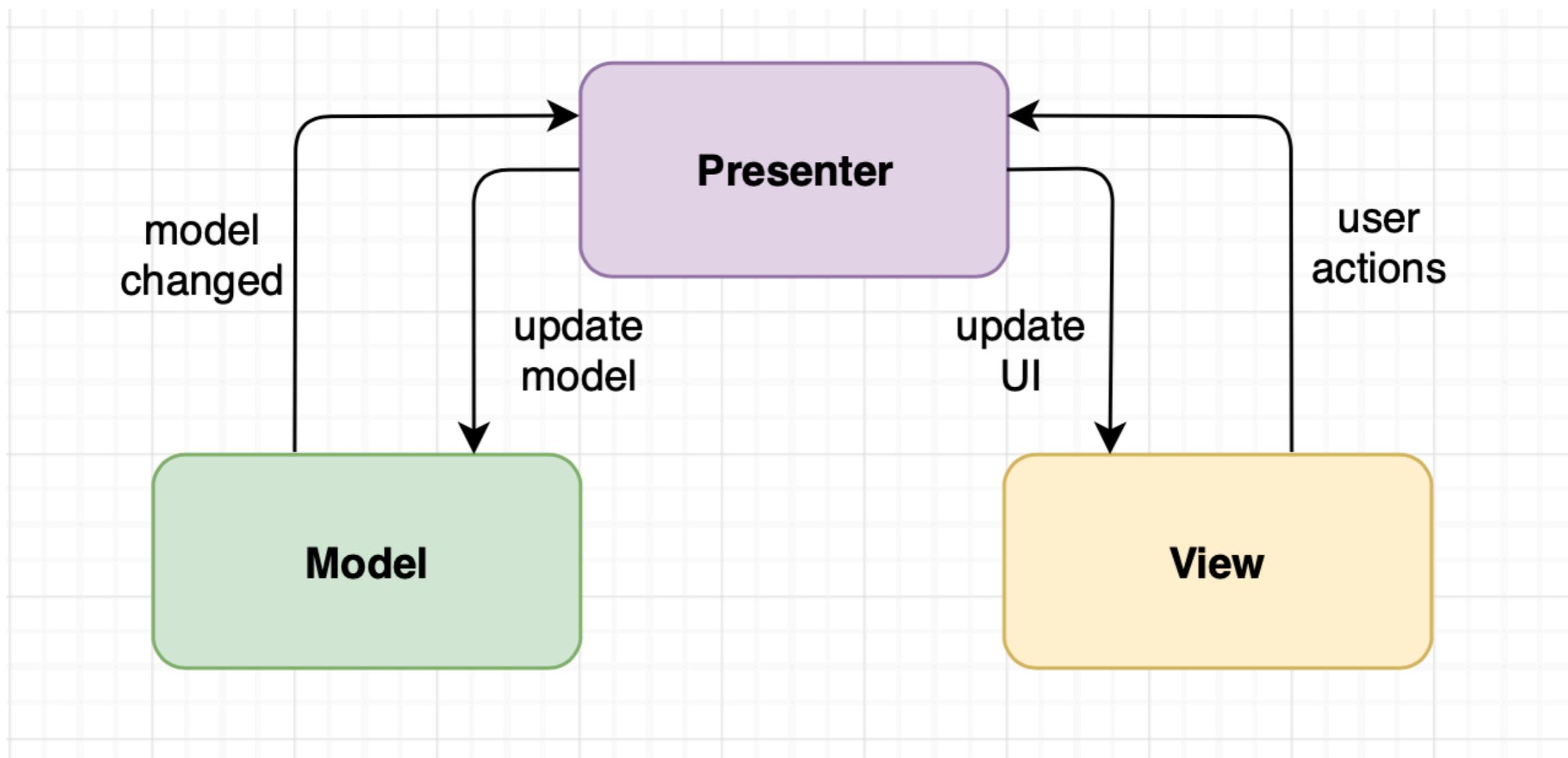
Clean Architecture



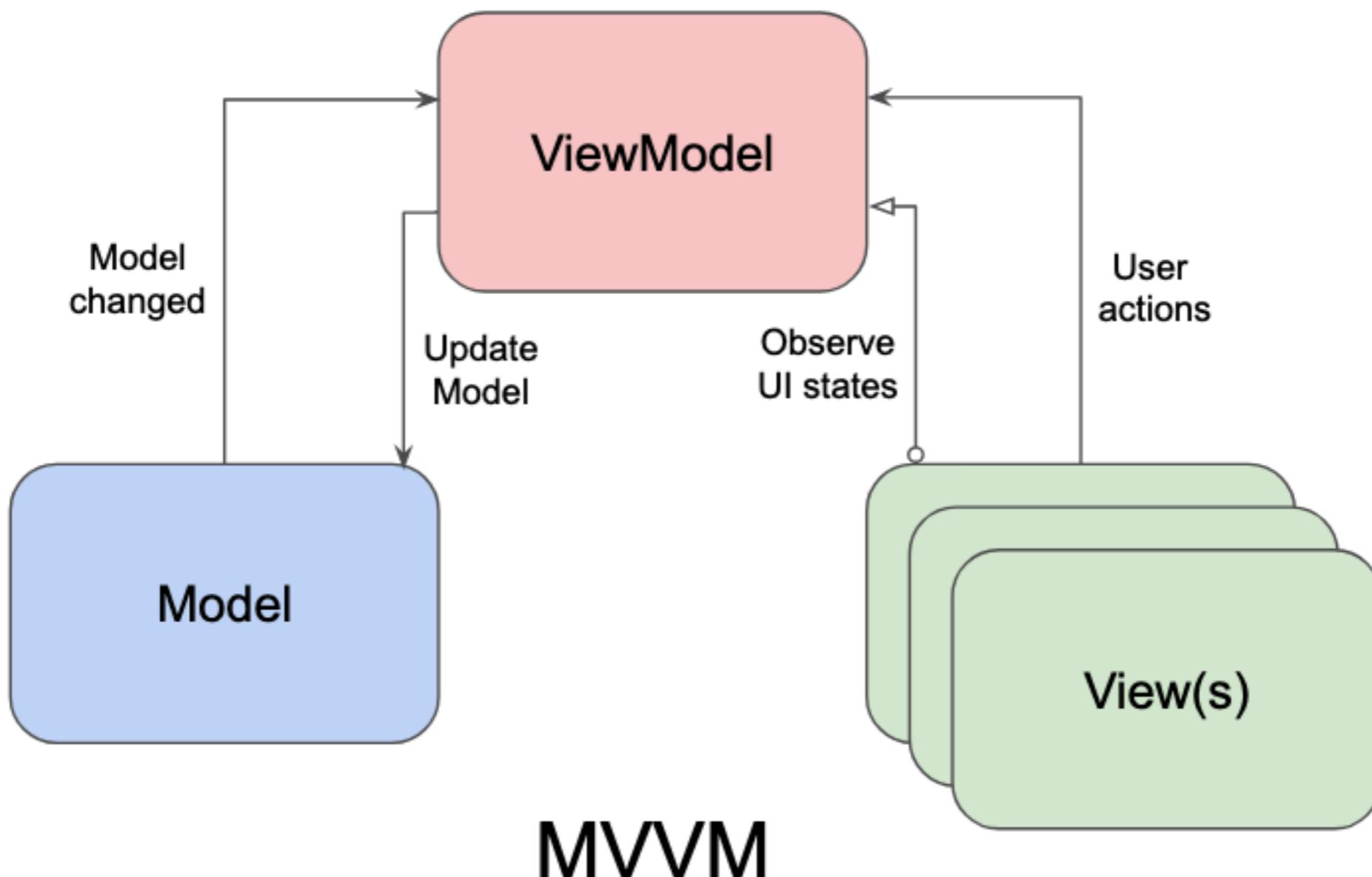
MVC



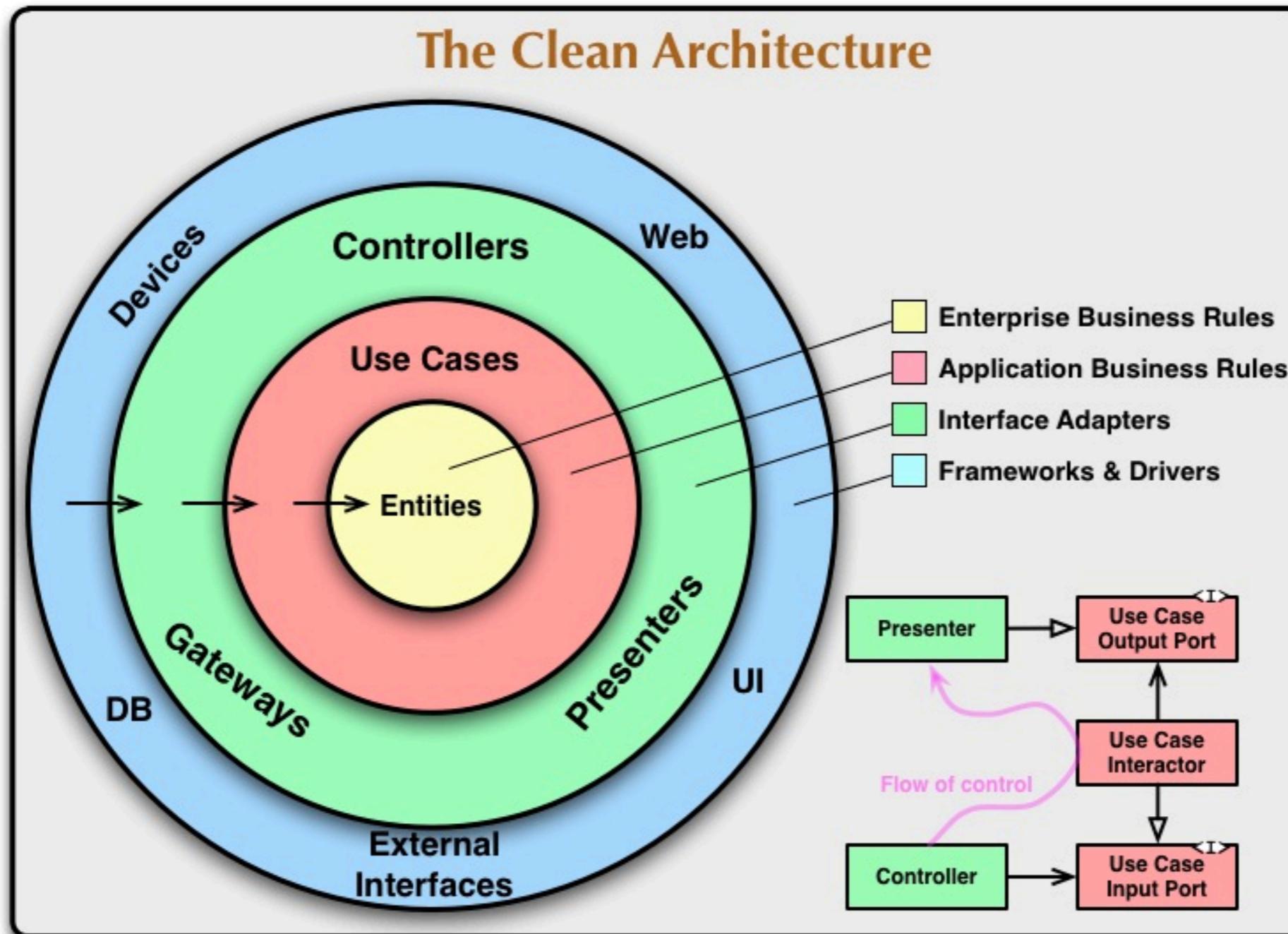
MVP



MVVM



Clean Architecture



Workshop with better project structure



Mobile App Testing



Important of Mobile Testing

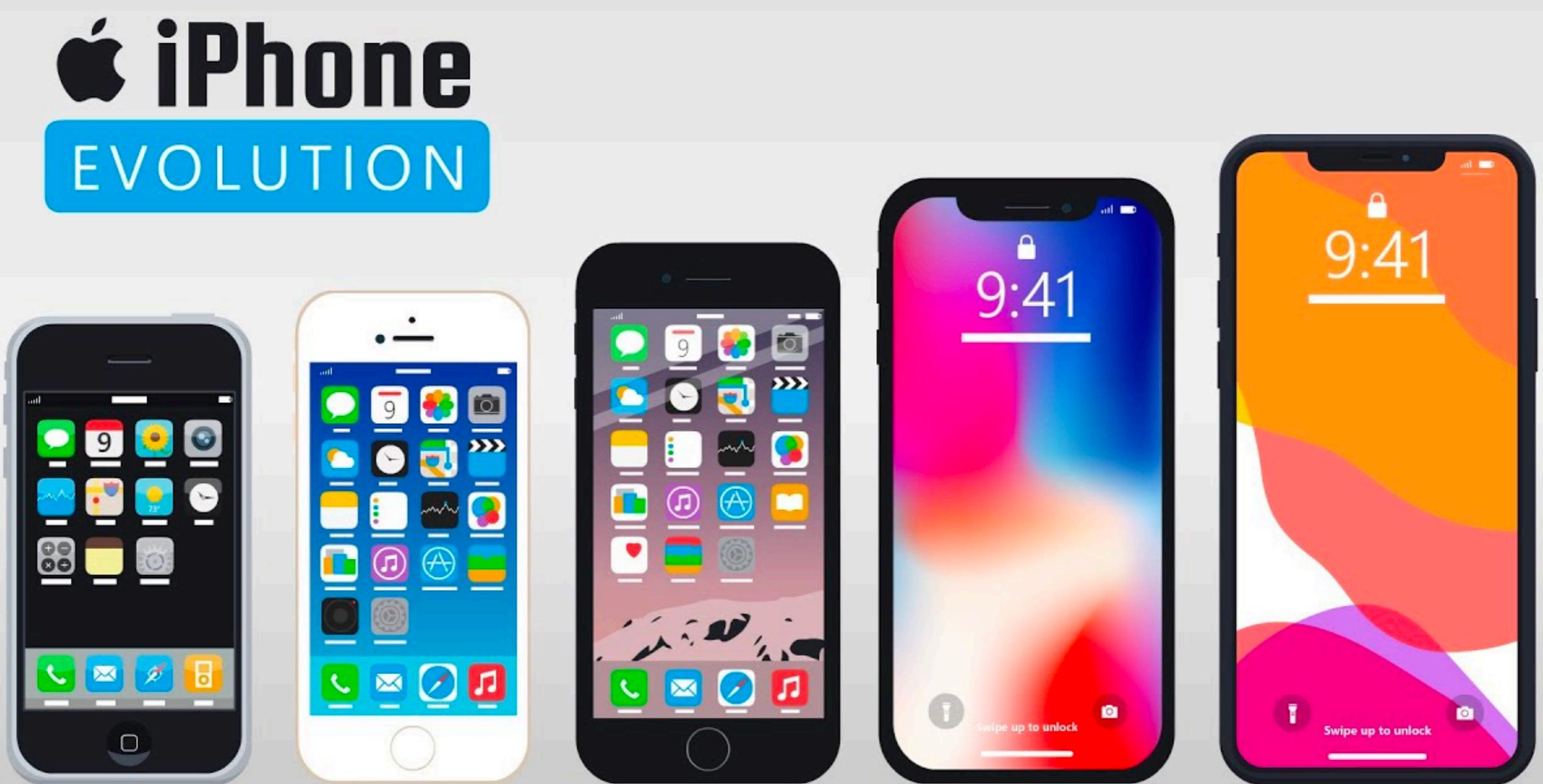
Variety of mobile devices

Different of operating system

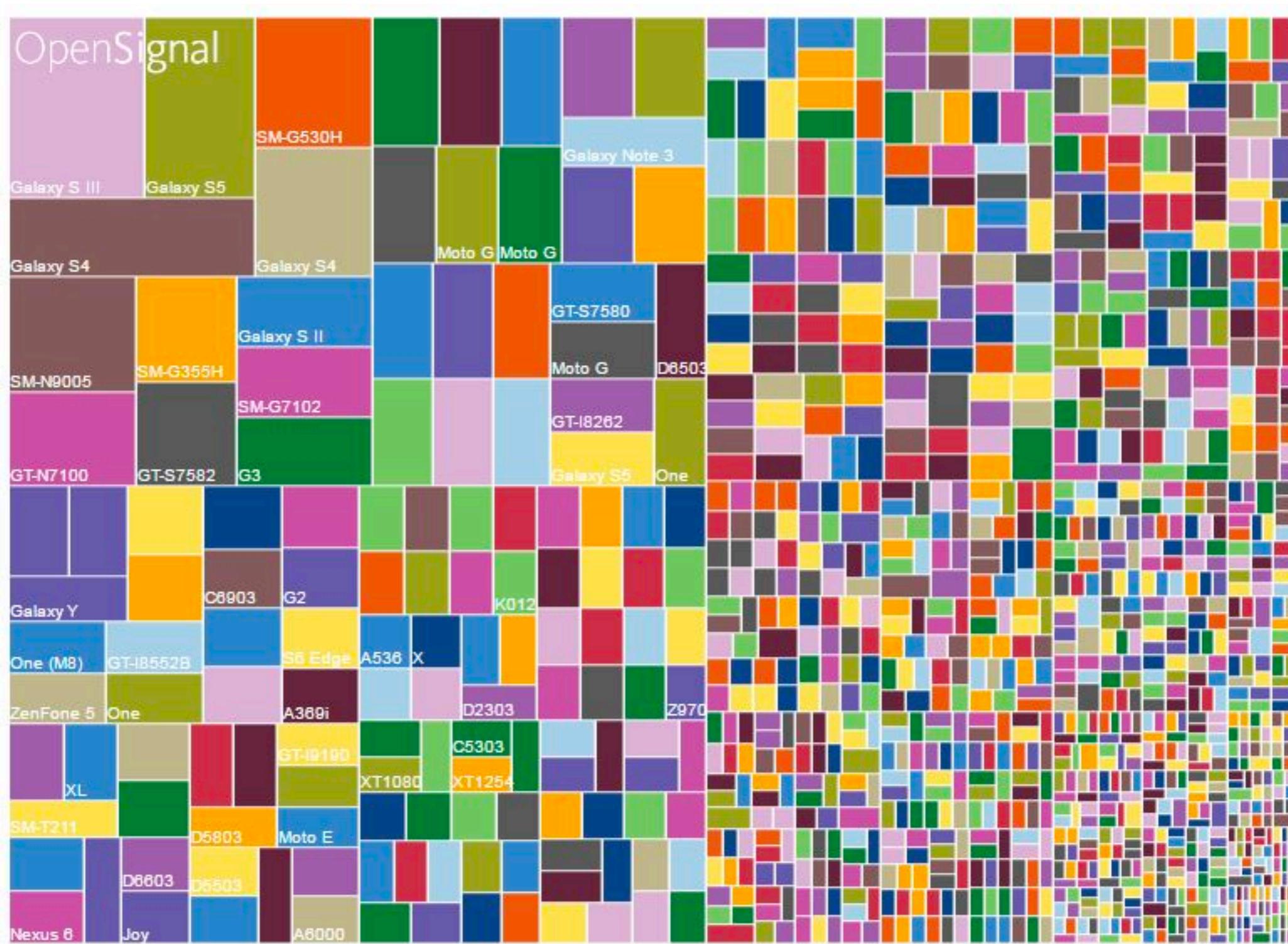
Constant upgrades of operating system



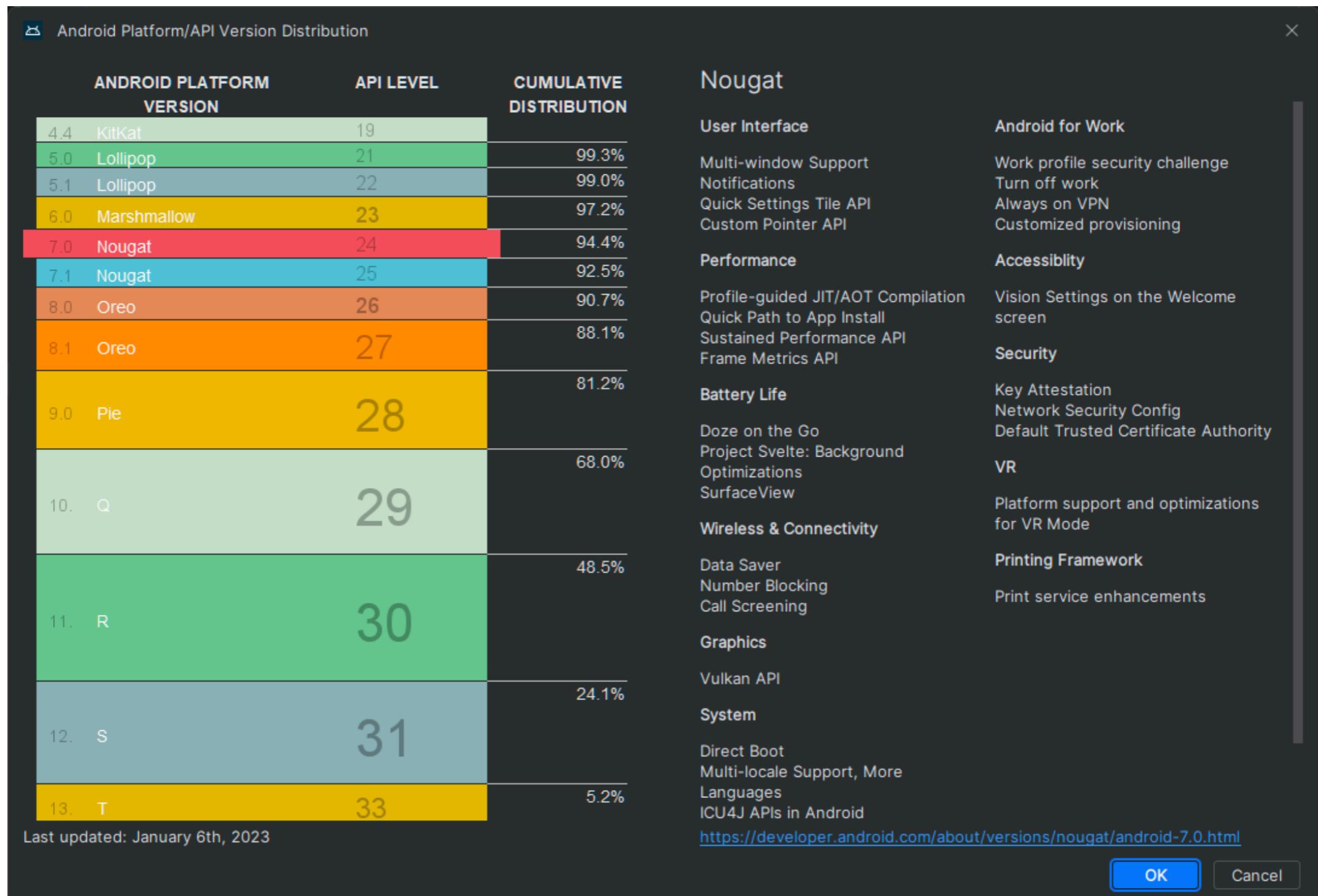
iPhone ?



Android ?



Android Version ?



Flutter testing

Unit test

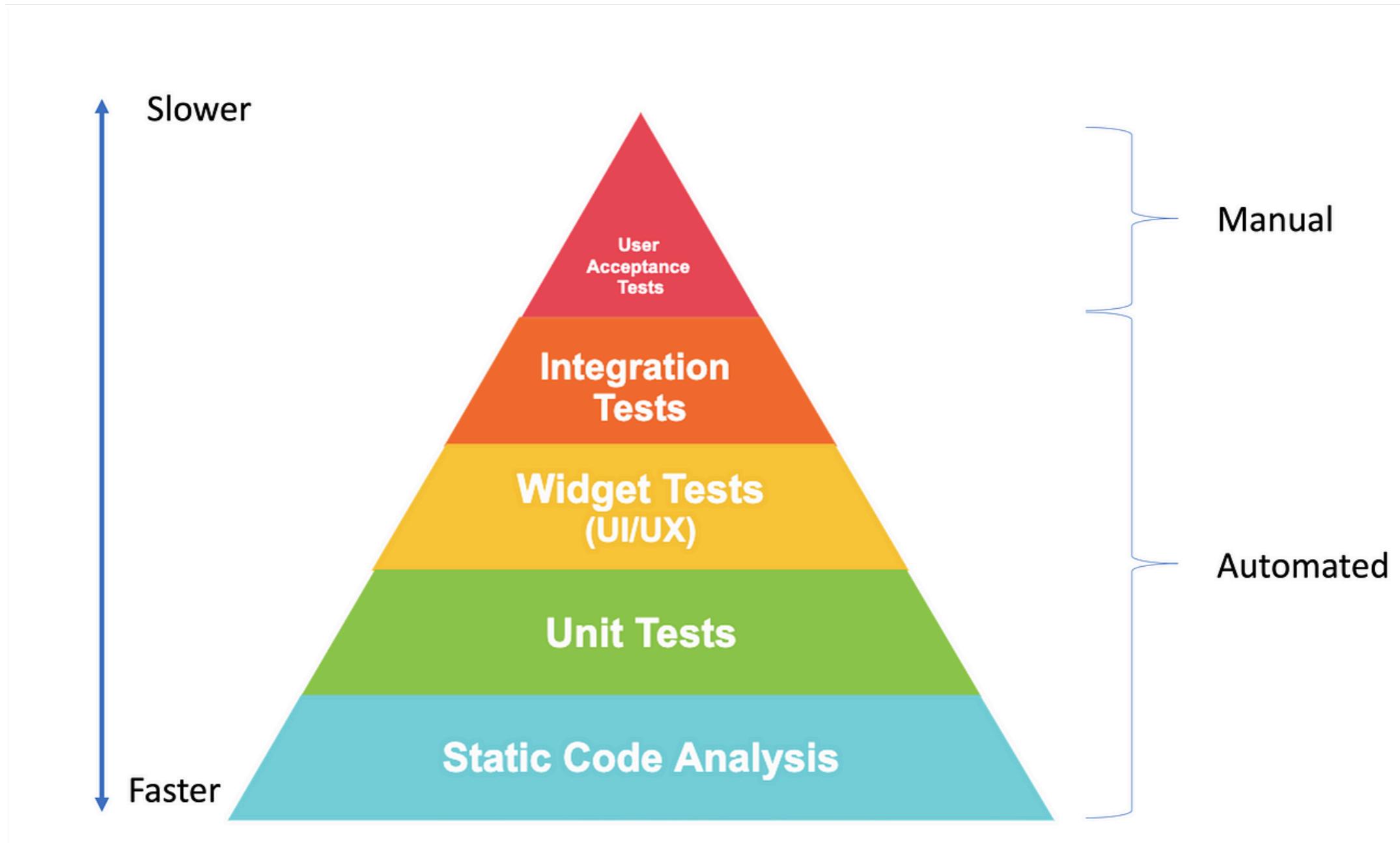
Widget test

Integration test

External test with Appium and Flutter



Flutter testing



Static Code Analysis

Working on compile-time before run unit test
Help to detect possible defect in early stage

Dart lint



\$flutter analyze

<https://dart.dev/effective-dart>
<https://pub.dev/packages/lint>

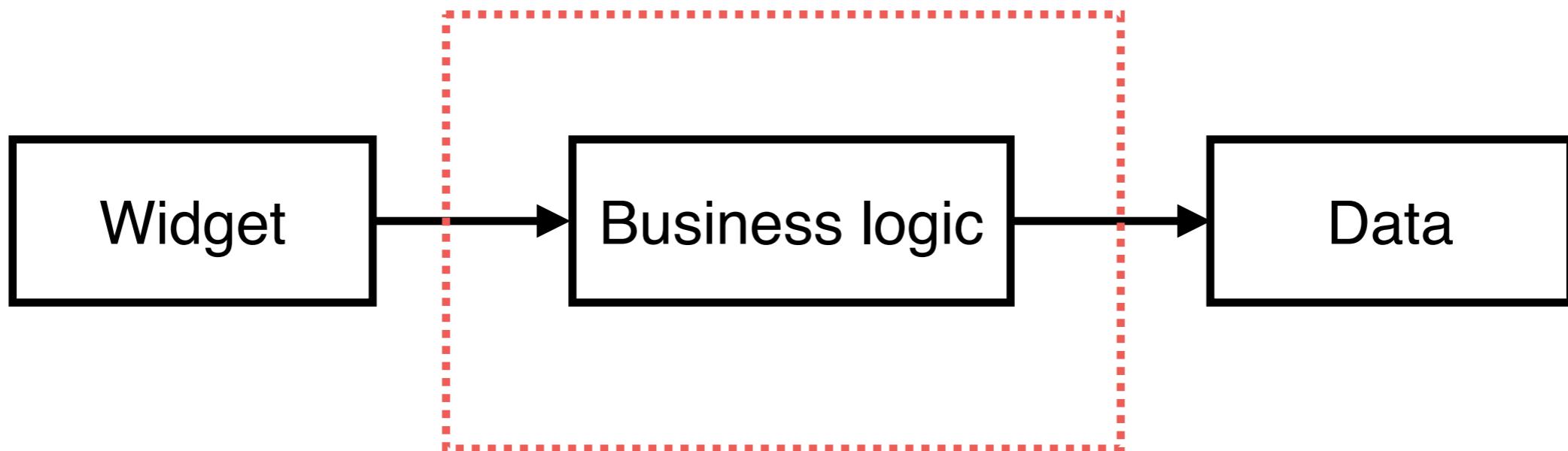


Flutter 101

© 2017 - 2018 Siam Chamnkit Company Limited. All rights reserved.

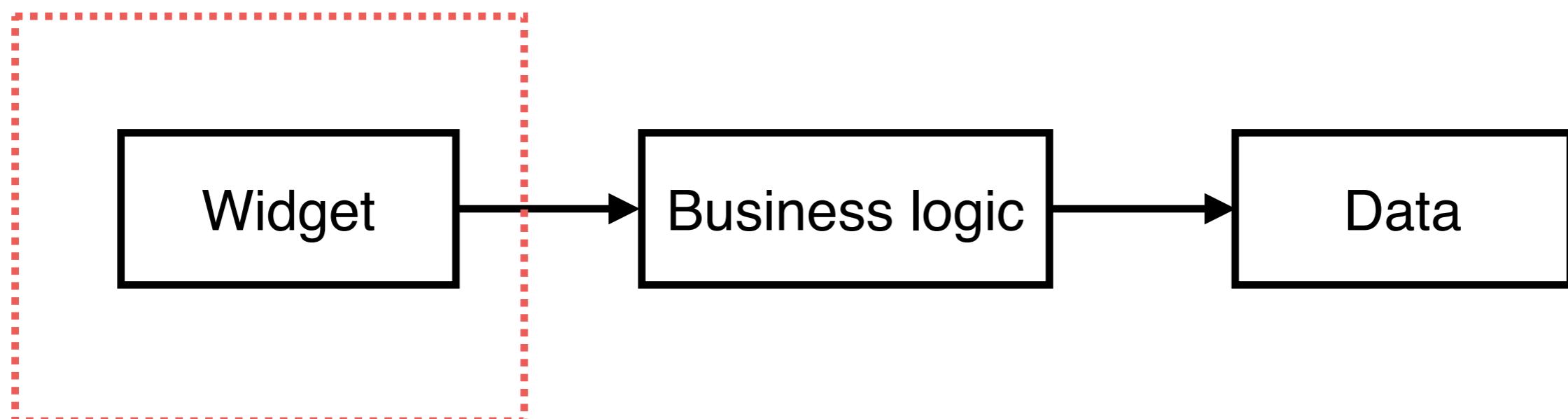
Unit Tests

Smallest piece of code
Code can be logically isolated in a system



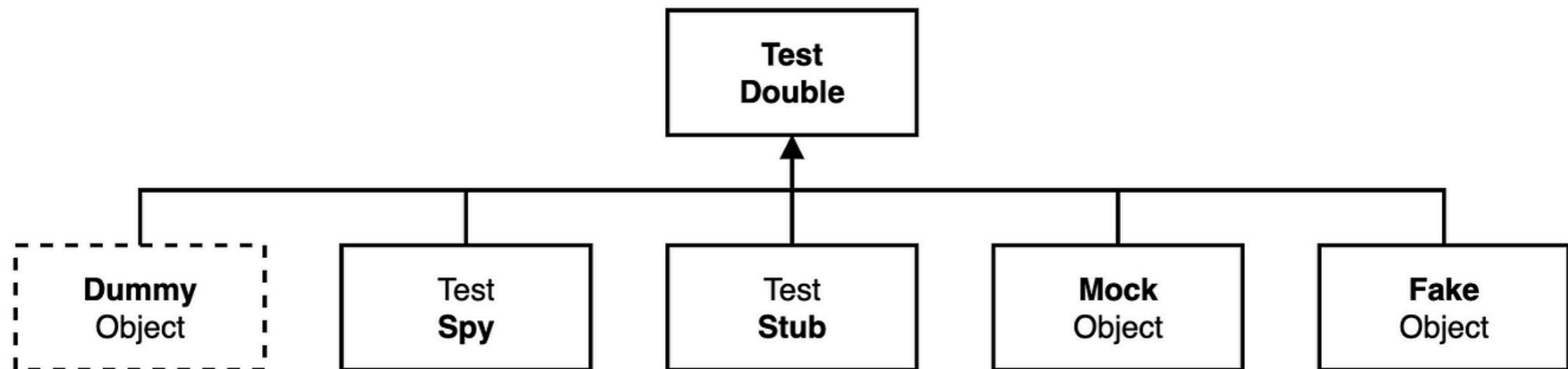
Widget Tests

Verify the widget's UI looks and interacts as expected
Not run on real device or emulator
Working with **Test double**



Test Double

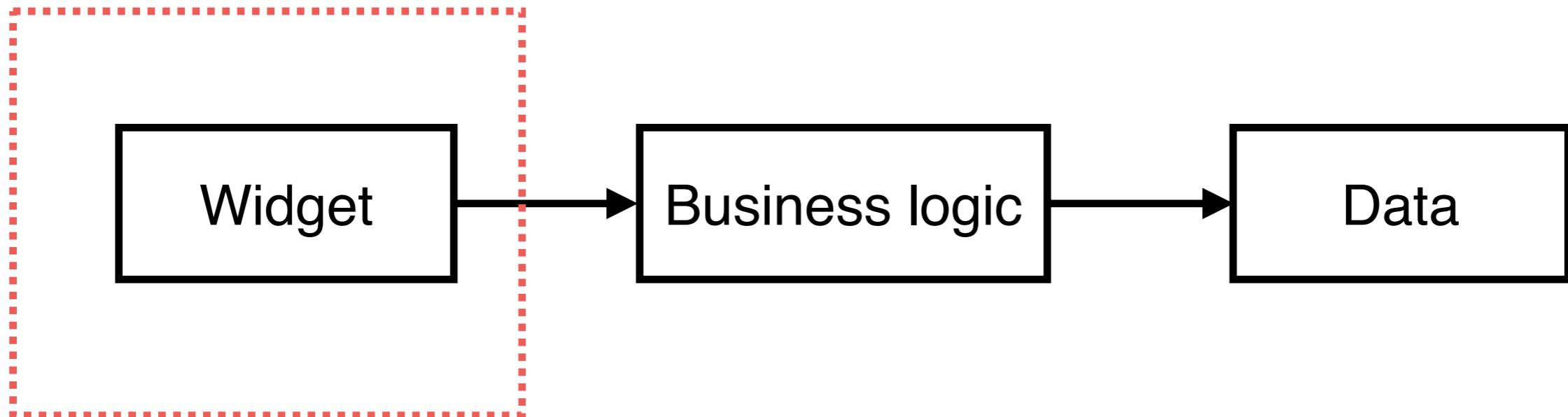
Generic term for any case where you replace a production code for testing purpose



<https://martinfowler.com/bliki/TestDouble.html>



More Widget Tests



Golden test or Screenshot test

Ensure UI integrity when we are working in a large team

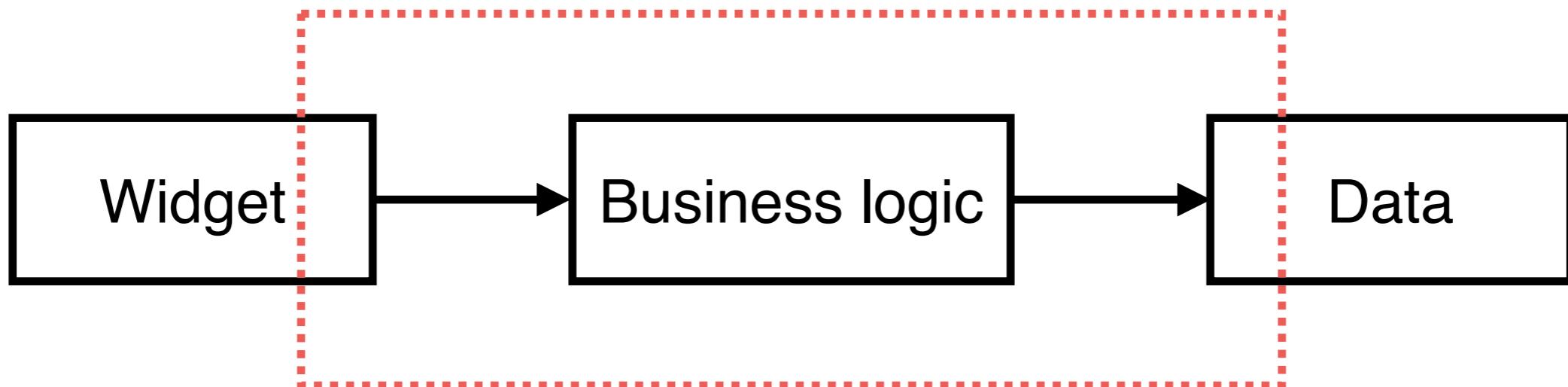
https://pub.dev/packages/golden_toolkit



Integration Tests

Individual modules are combined and tested as a group

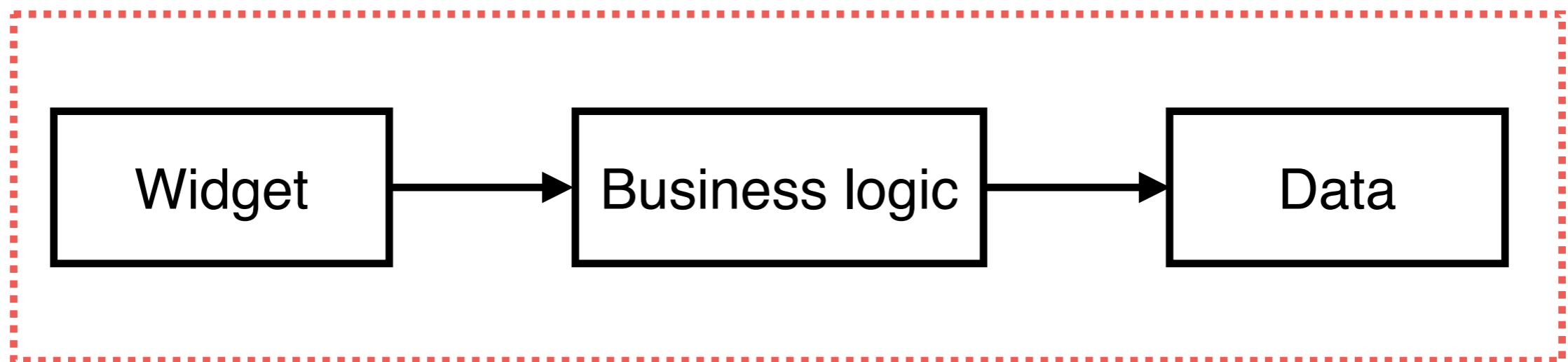
Working with **Test double**



Integration Tests

Individual modules are combined and tested as a group

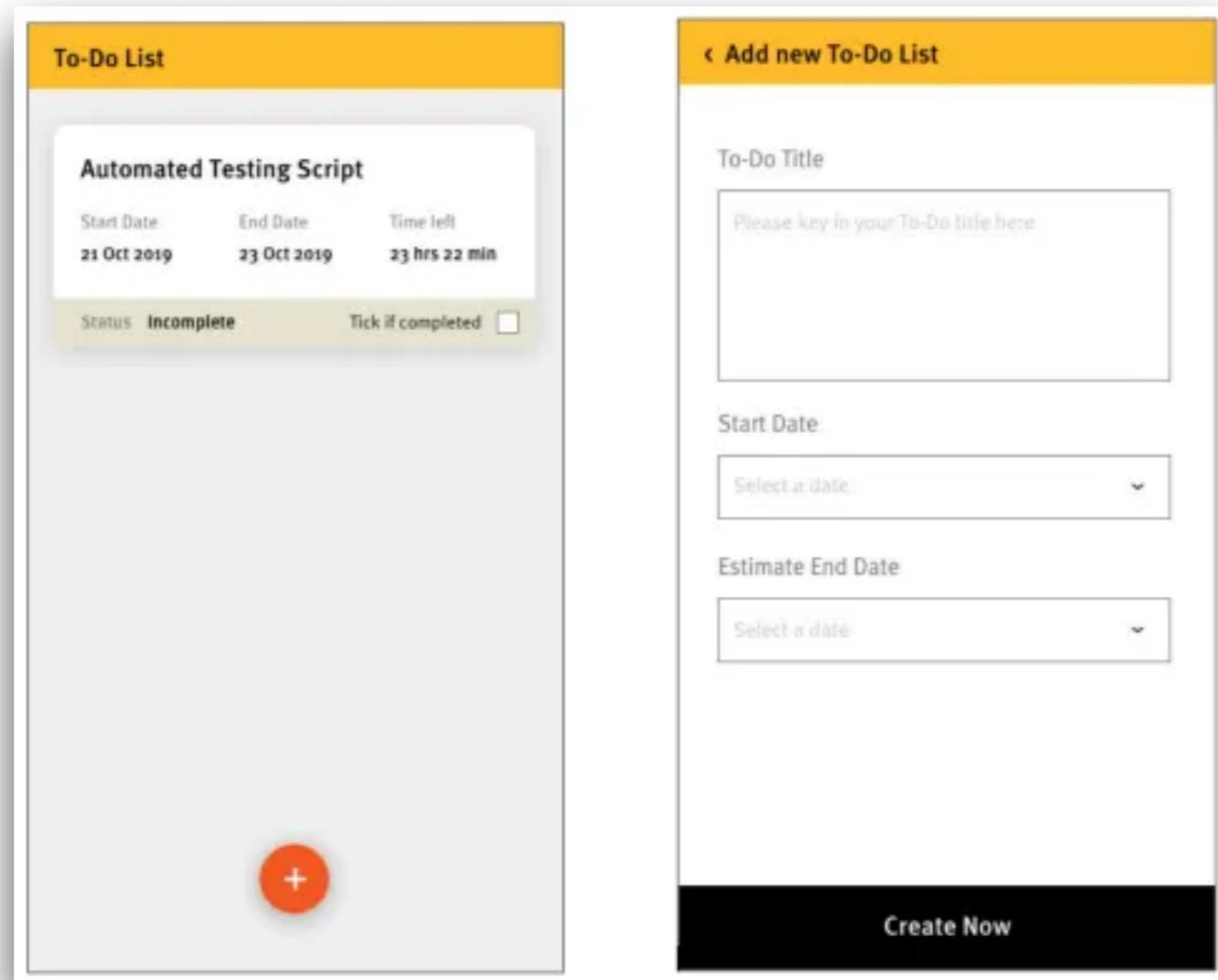
Run on real device or emulator



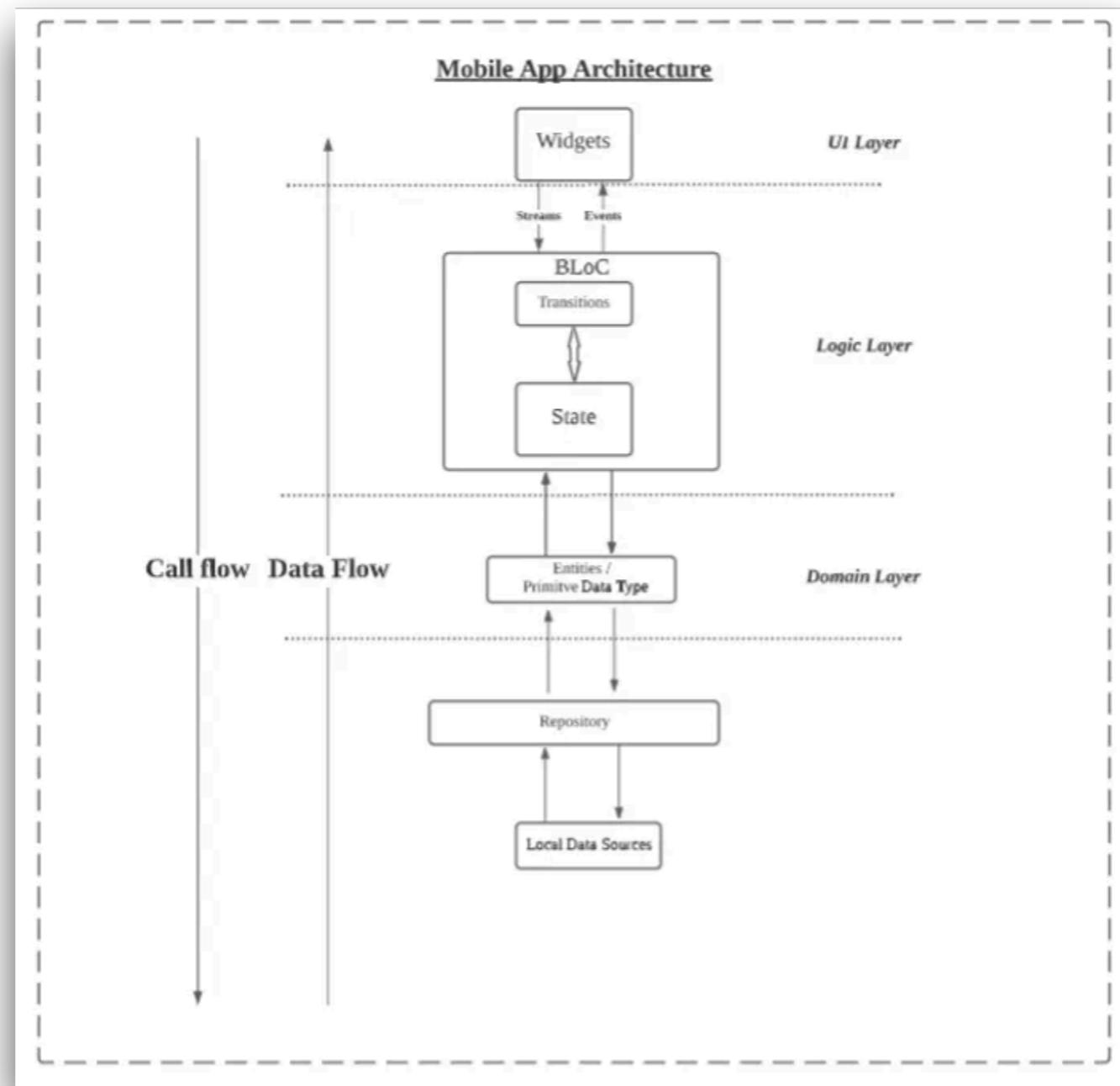
Workshop with Flutter Testing



Sample Project



Project Structure



Q/A

