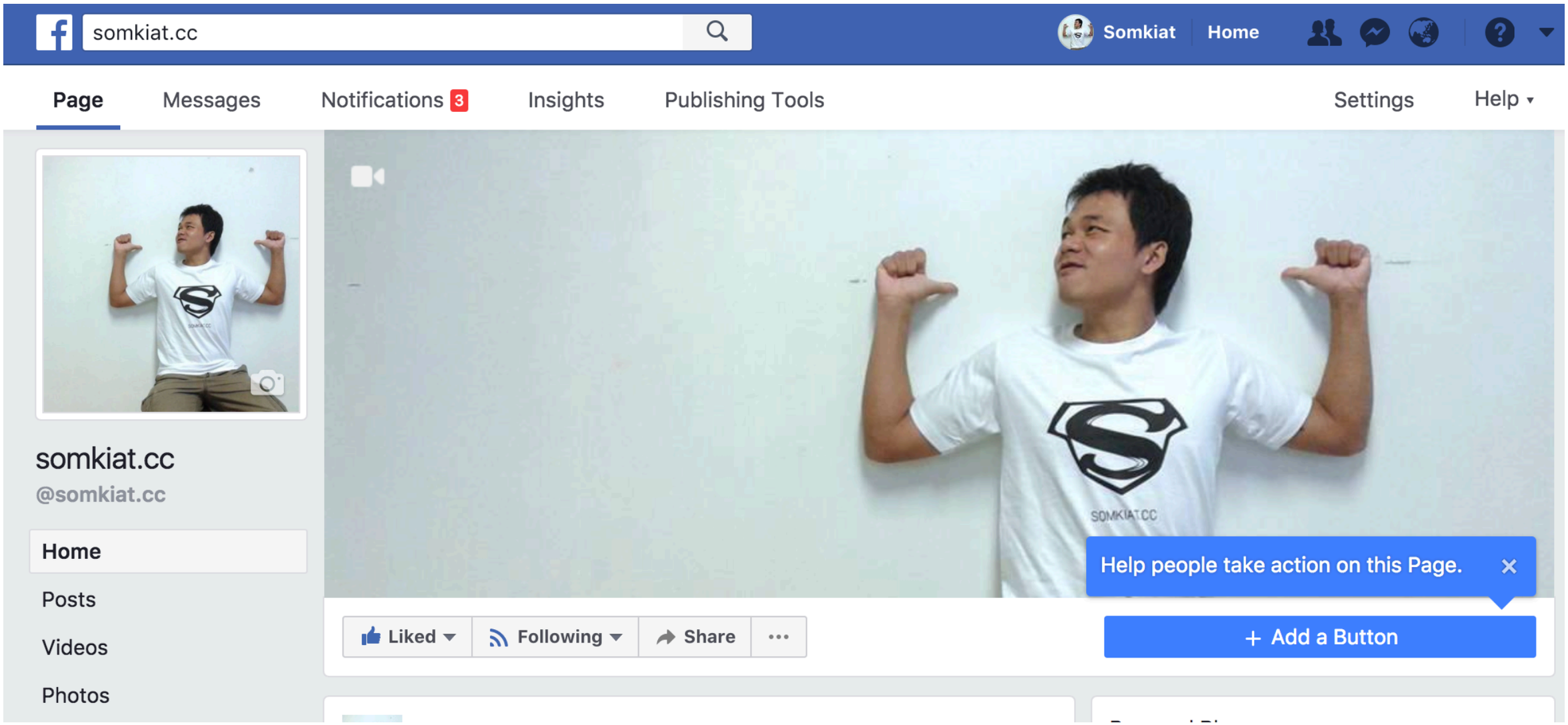
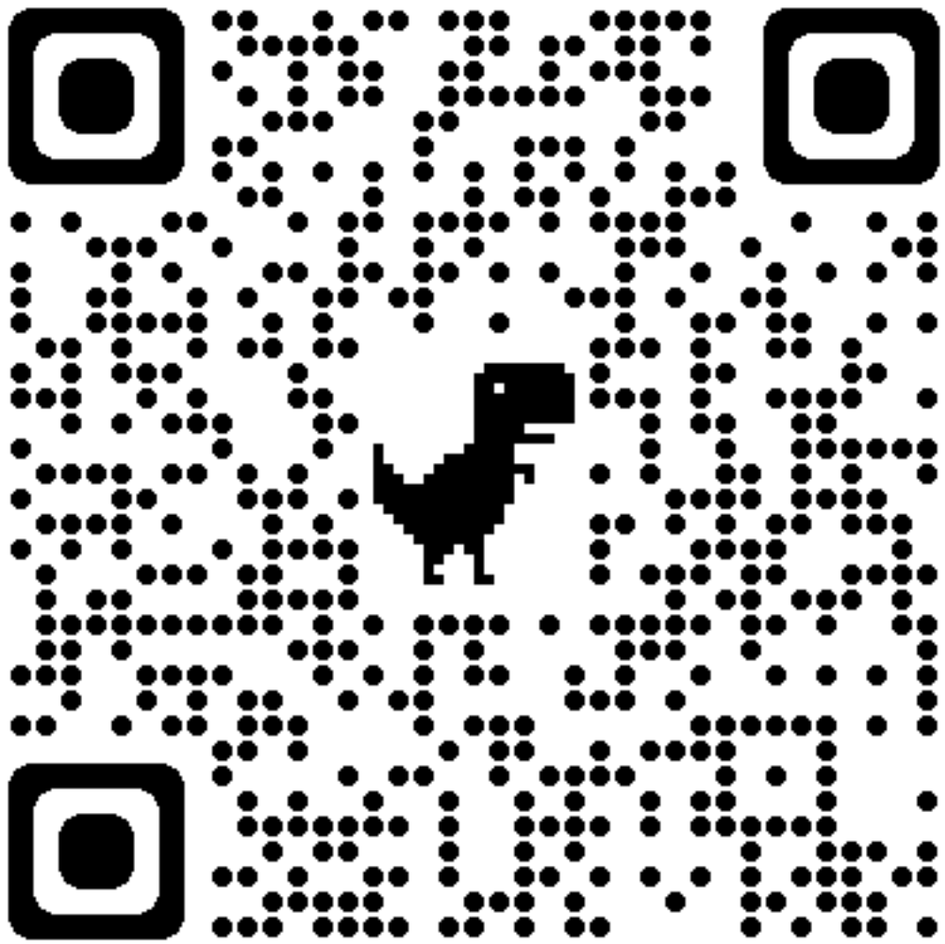
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Develop Mobile App with Flutter





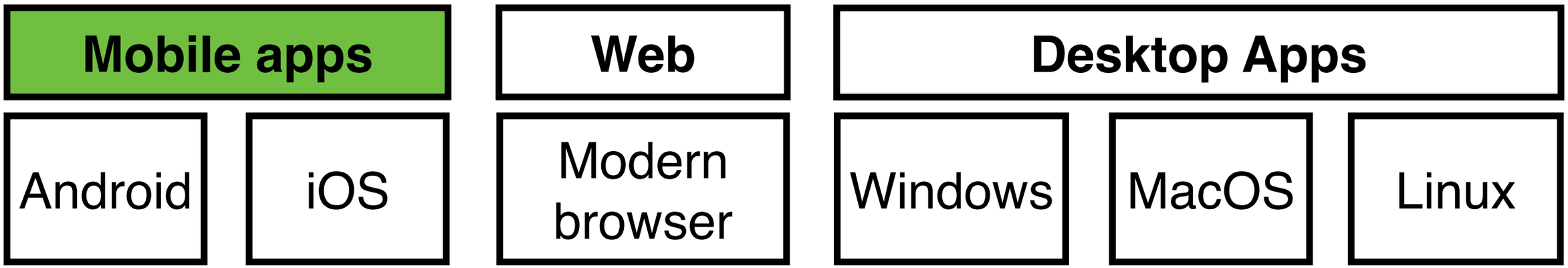
https://github.com/up1/course-flutter-101

Develop Mobile App

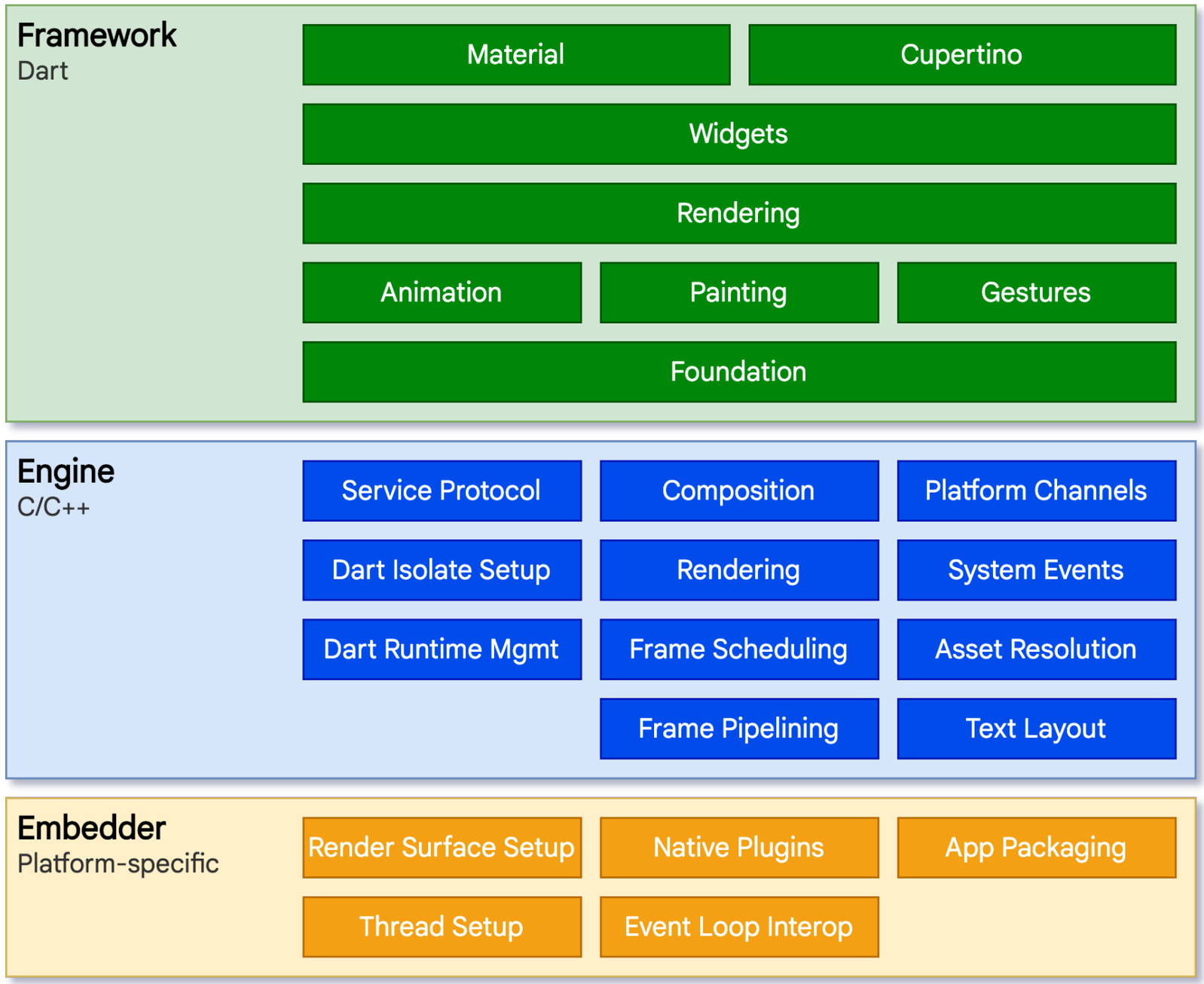
https://flutter.dev/

What is Flutter ?

UI framework developed with **Dart** programming
Write code for cross platform
One codebase, Multiple Apps

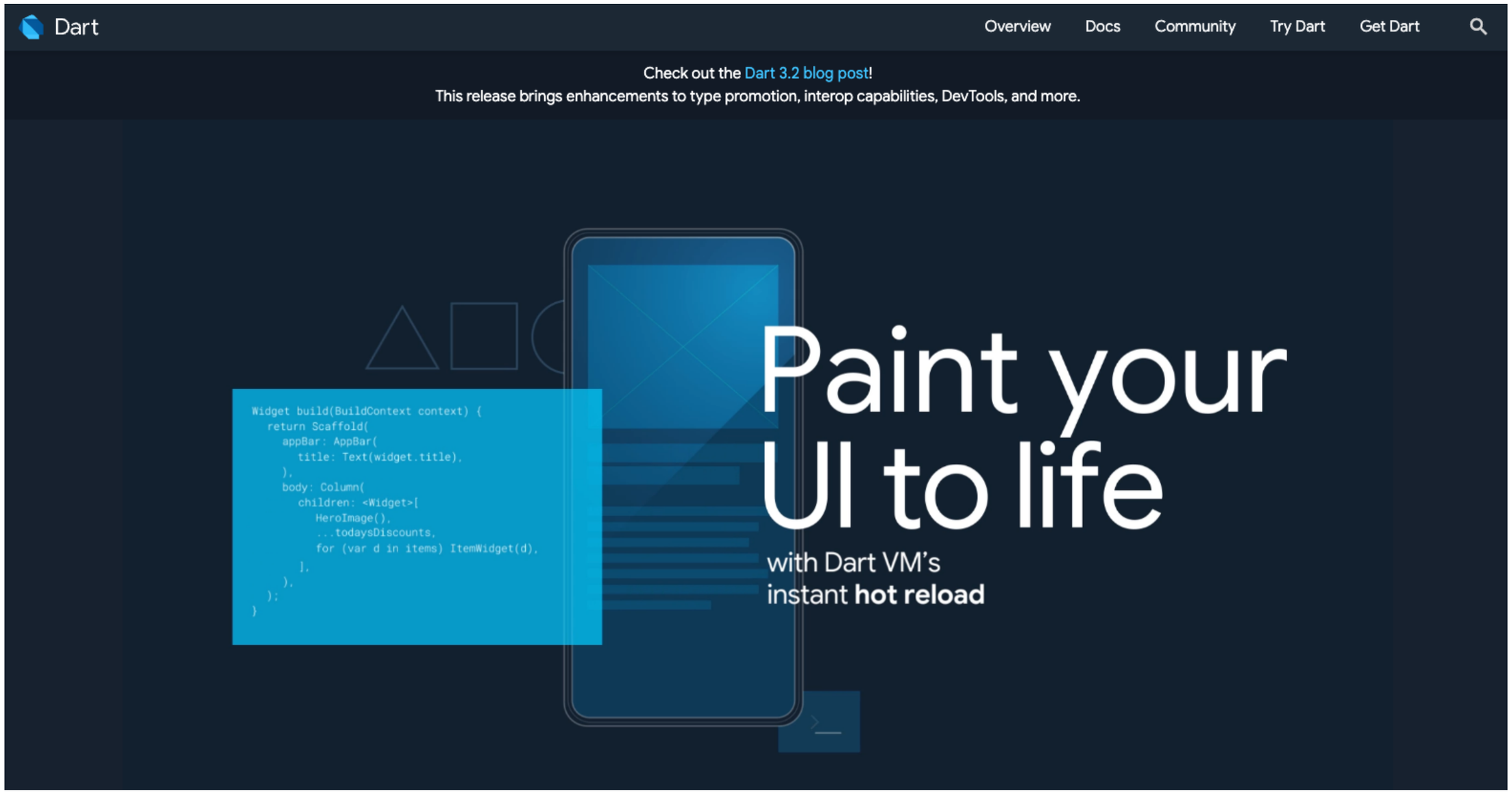


Architecture of Flutter



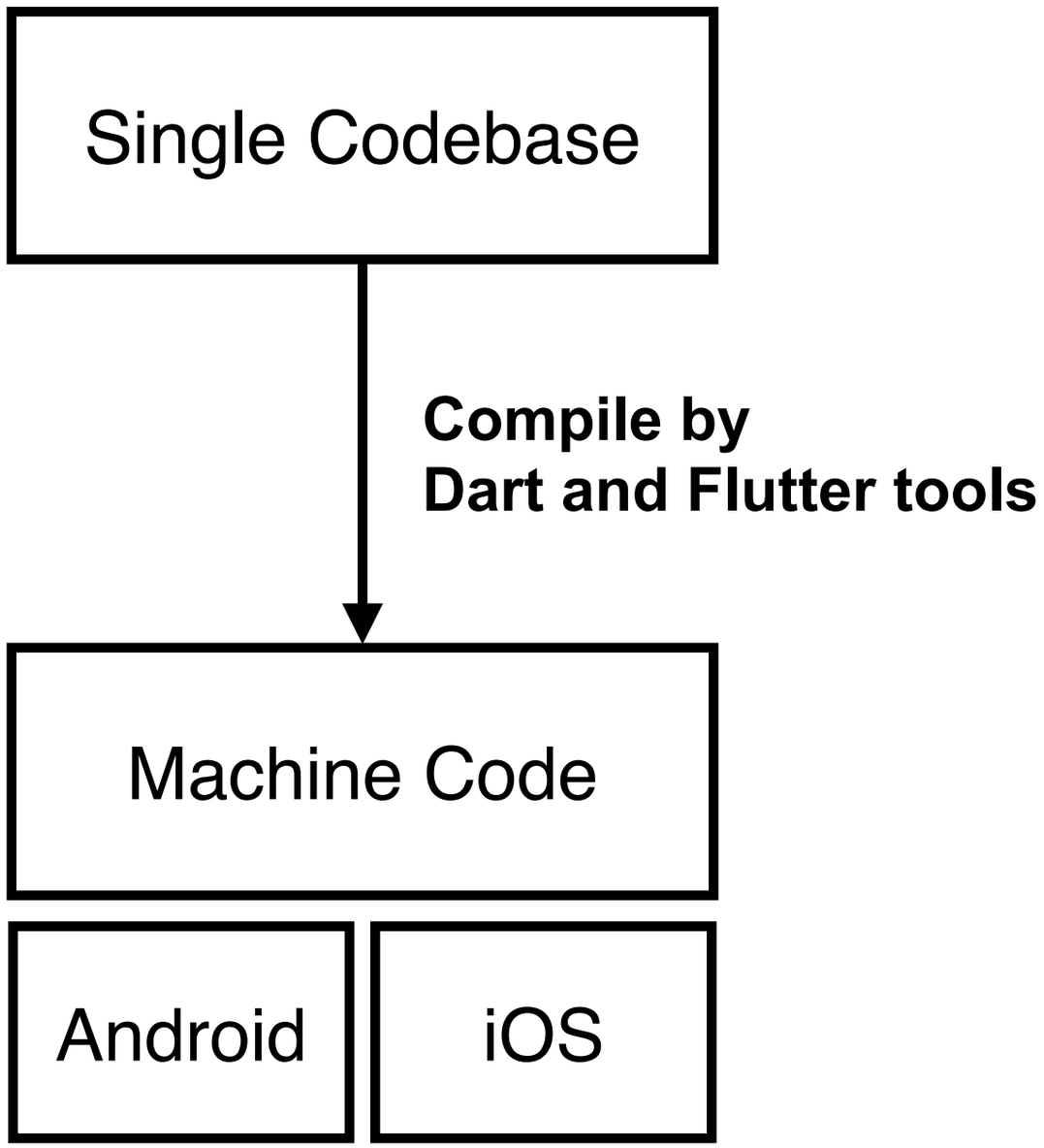
https://docs.flutter.dev/resources/architectural-overview

Dart



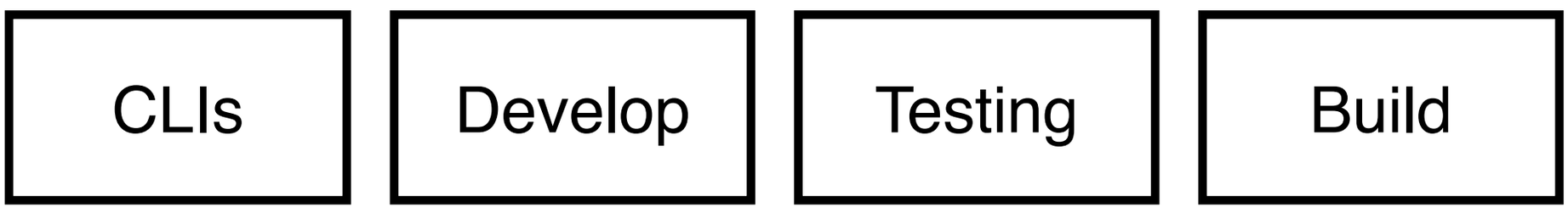
https://dart.dev/

Flutter to platform-specific machine code



What is Flutter ?

Provide collection of tools



Software Requirements

Android	**iOS**
JAVA	Xcode
Android SDK	CocoaPods

Flutter and Dart

Install Flutter and Dart

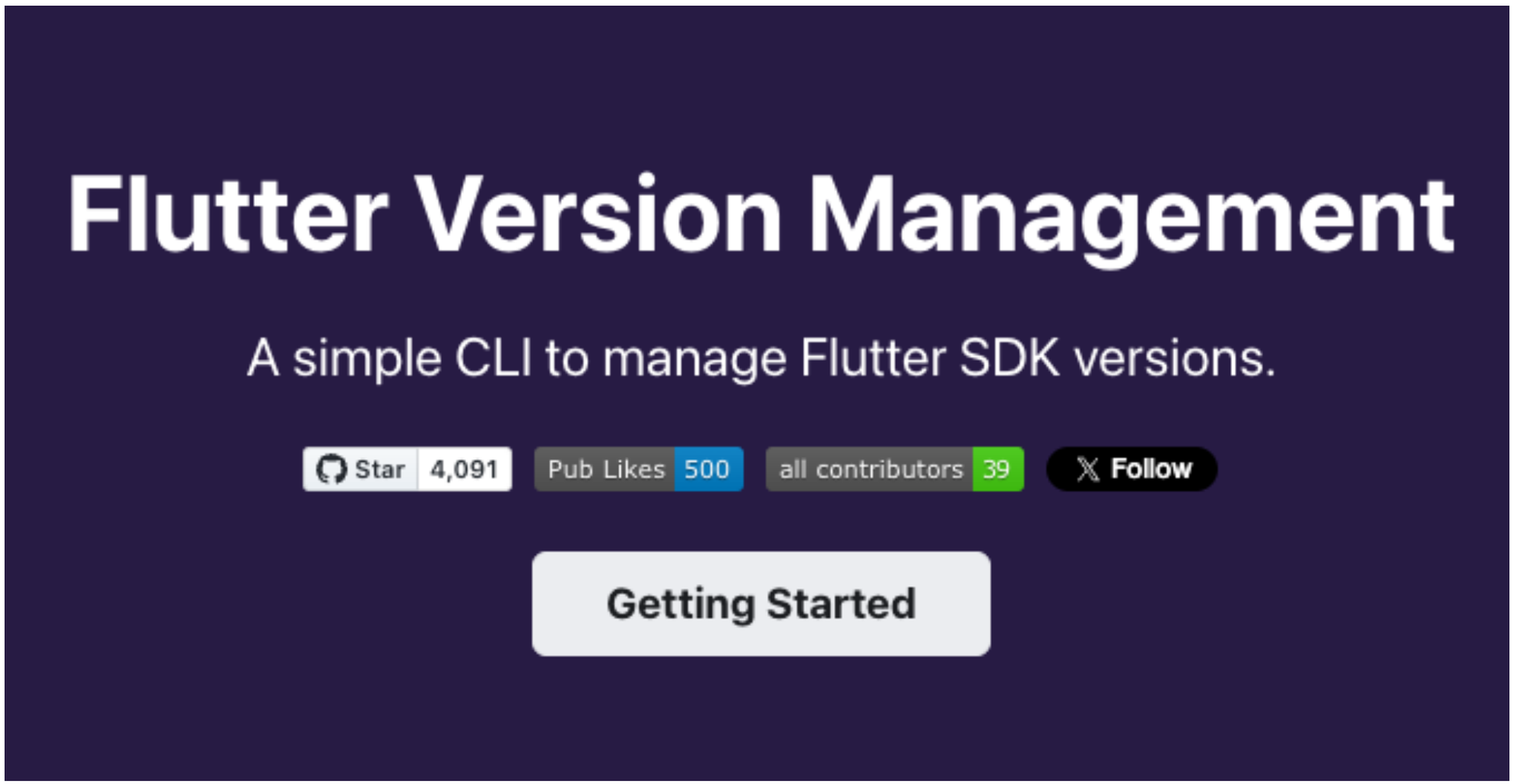
$flutter doctor

```
Doctor summary (to see all details, run flu
[✓] Flutter (Channel stable, 3.16.3, on mac
    en-TH)
[✓] Android toolchain - develop for Android
[✓] Xcode - develop for iOS and macOS (Xcod
[✓] Chrome - develop for the web
[✓] Android Studio (version 2022.3)
[✓] IntelliJ IDEA Community Edition (versio
[✓] VS Code (version 1.85.1)
[✓] Connected device (2 available)
[✓] Network resources
```

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

Flutter Version Management

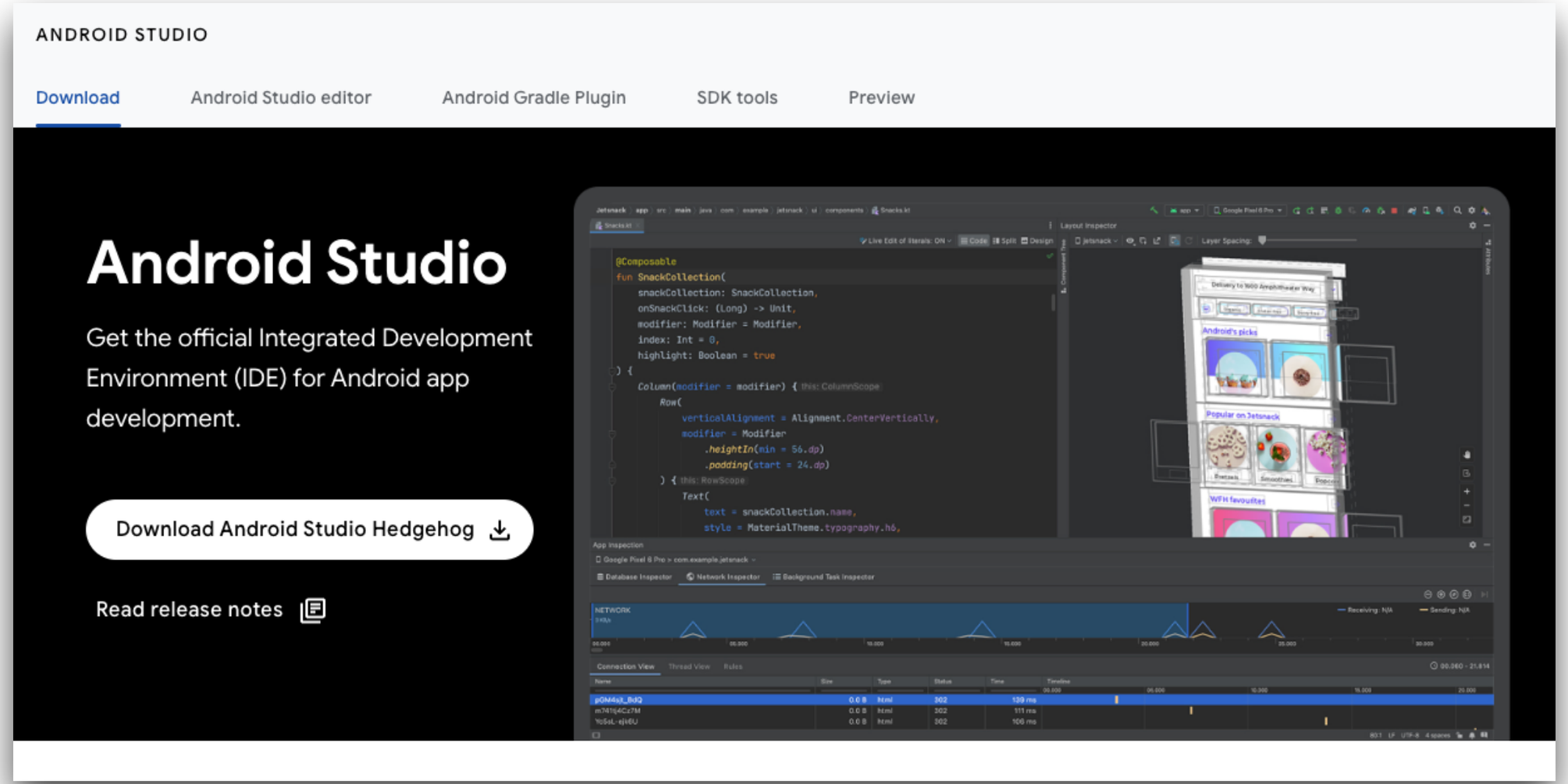
CLI to manage Flutter SDK versions



https://fvm.app/

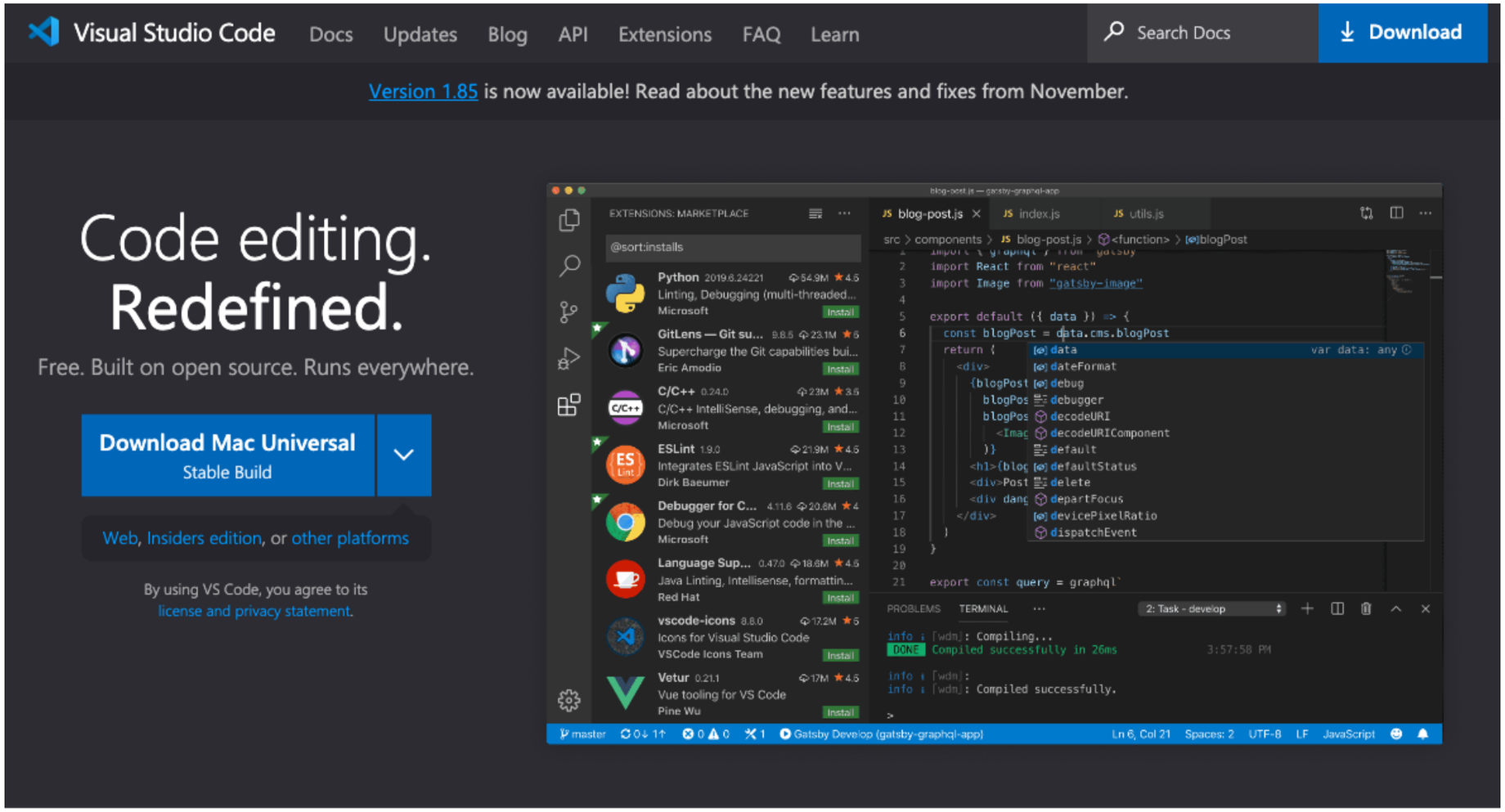
Android Studio

JAVA and Android SDK



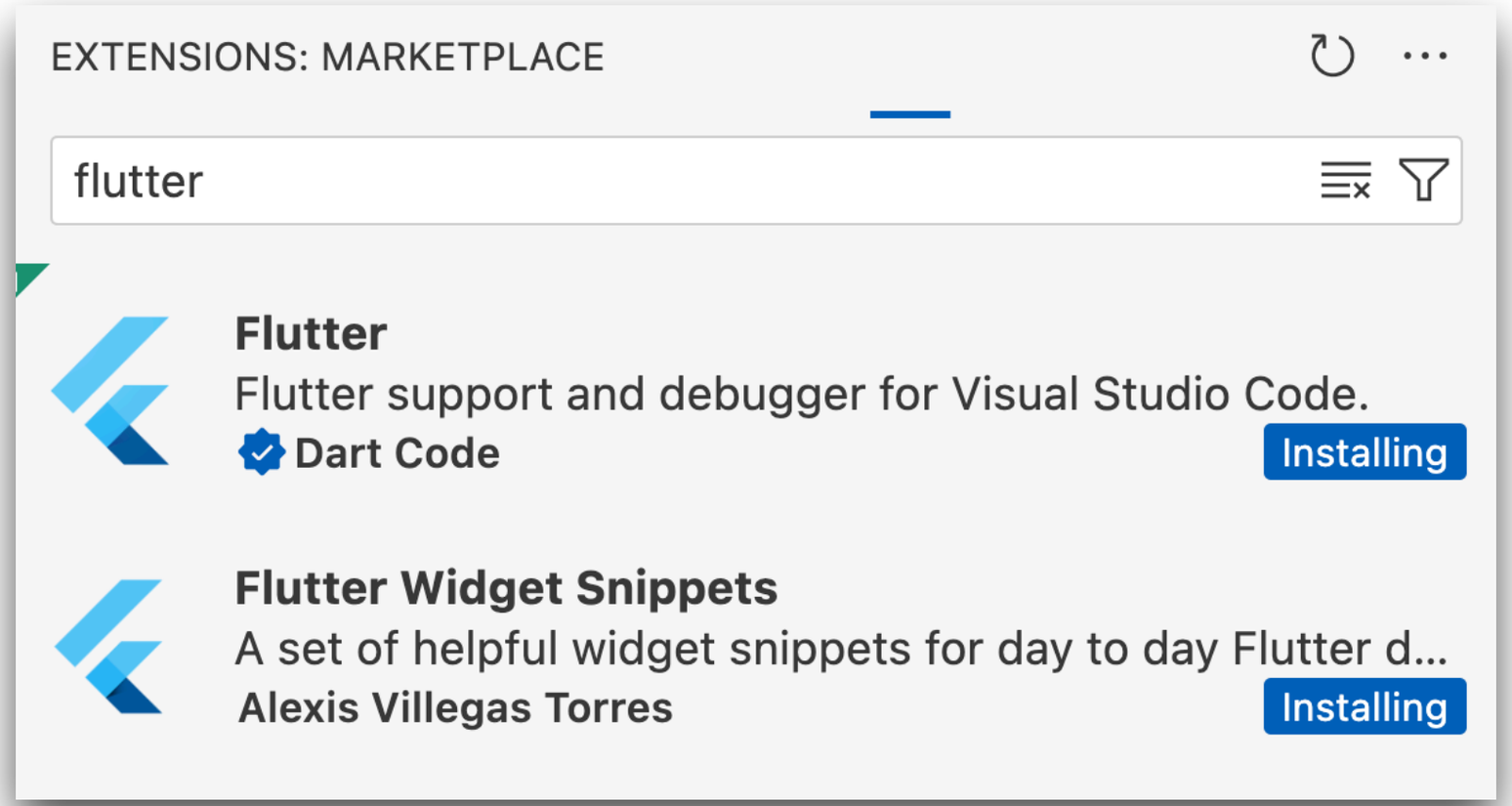
https://developer.android.com/studio

Code Editor with VSCode

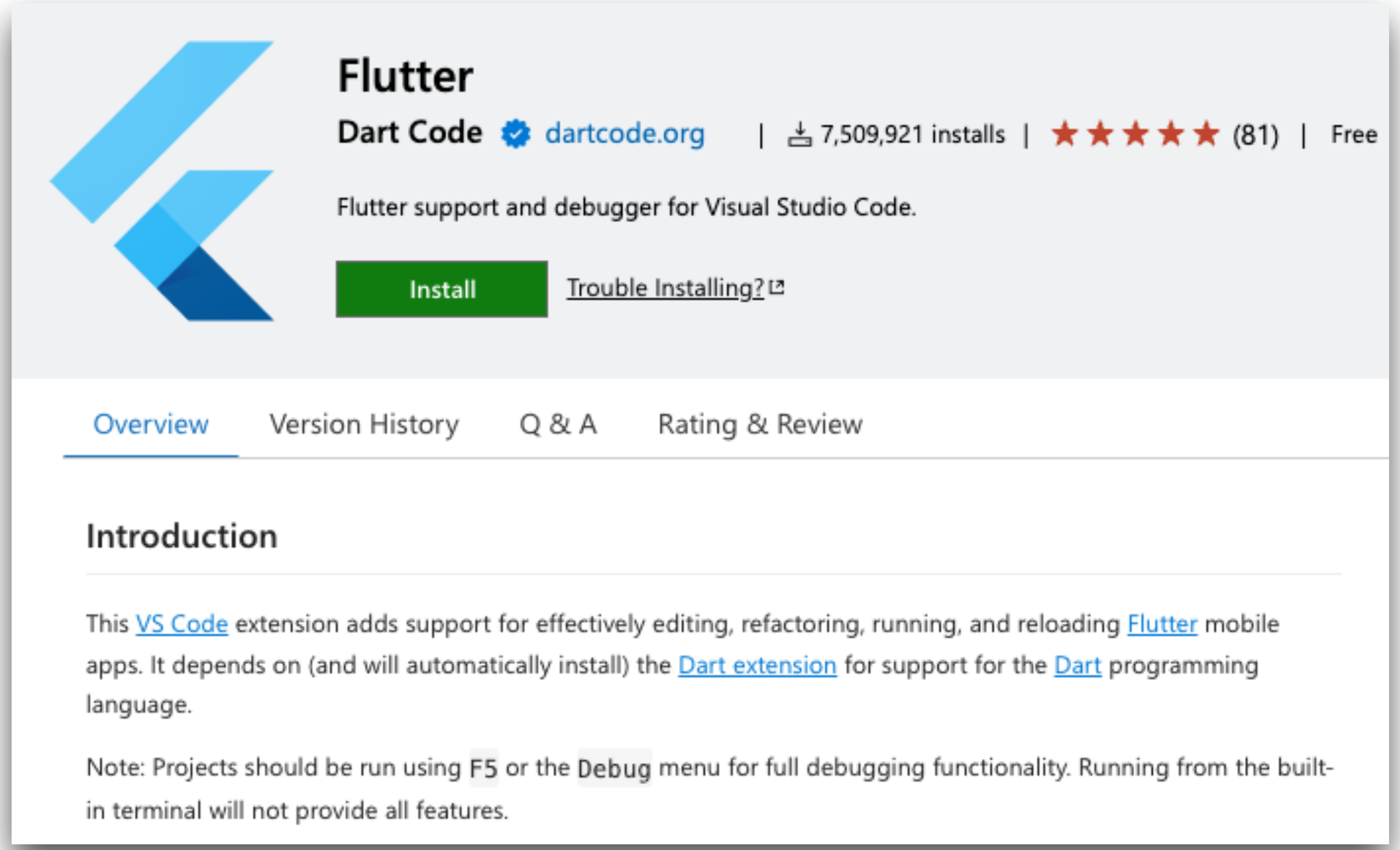


https://code.visualstudio.com/

VSCode Flutter Extension

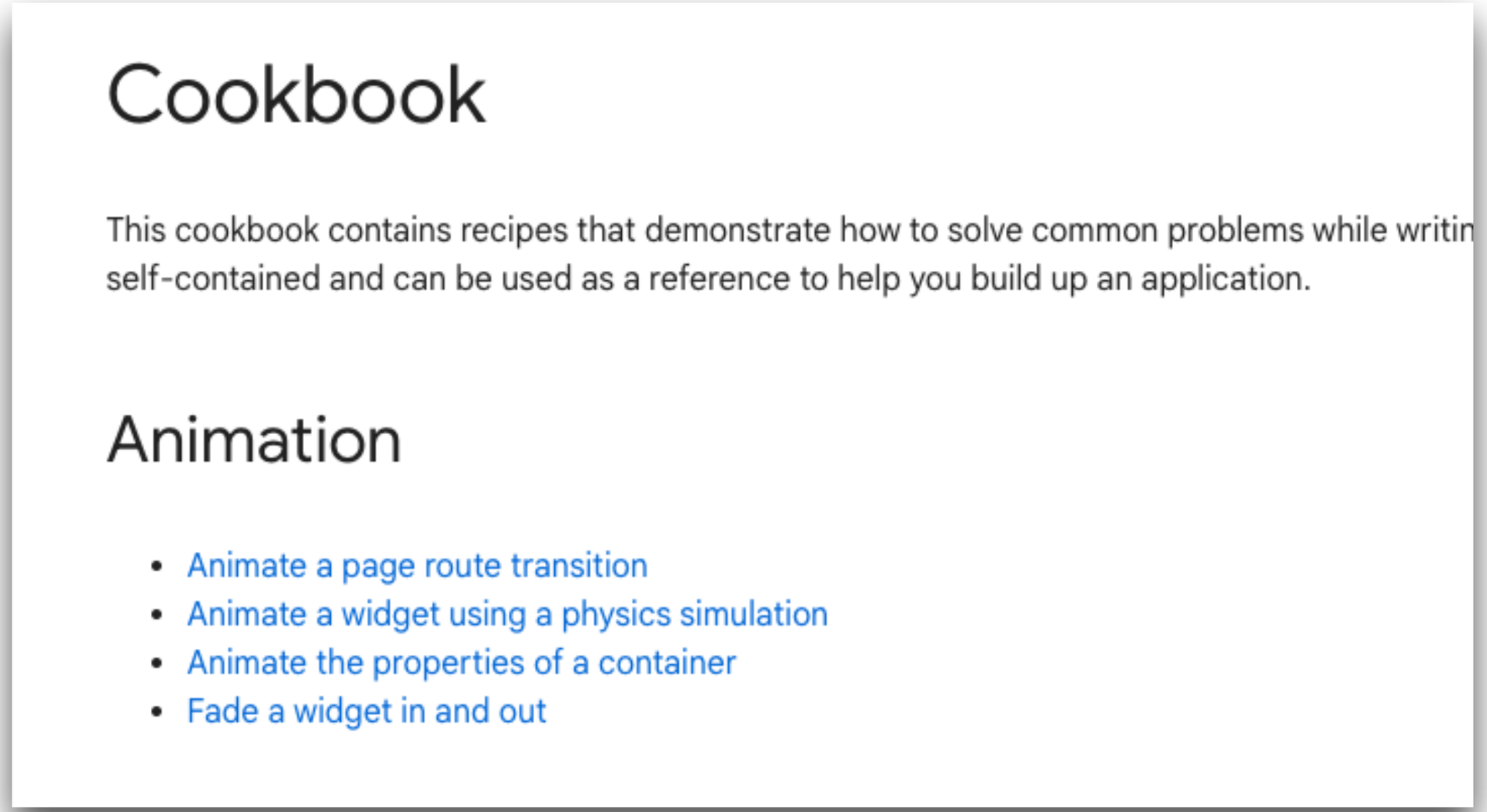


VSCode Flutter Extension



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

Flutter Cookbook

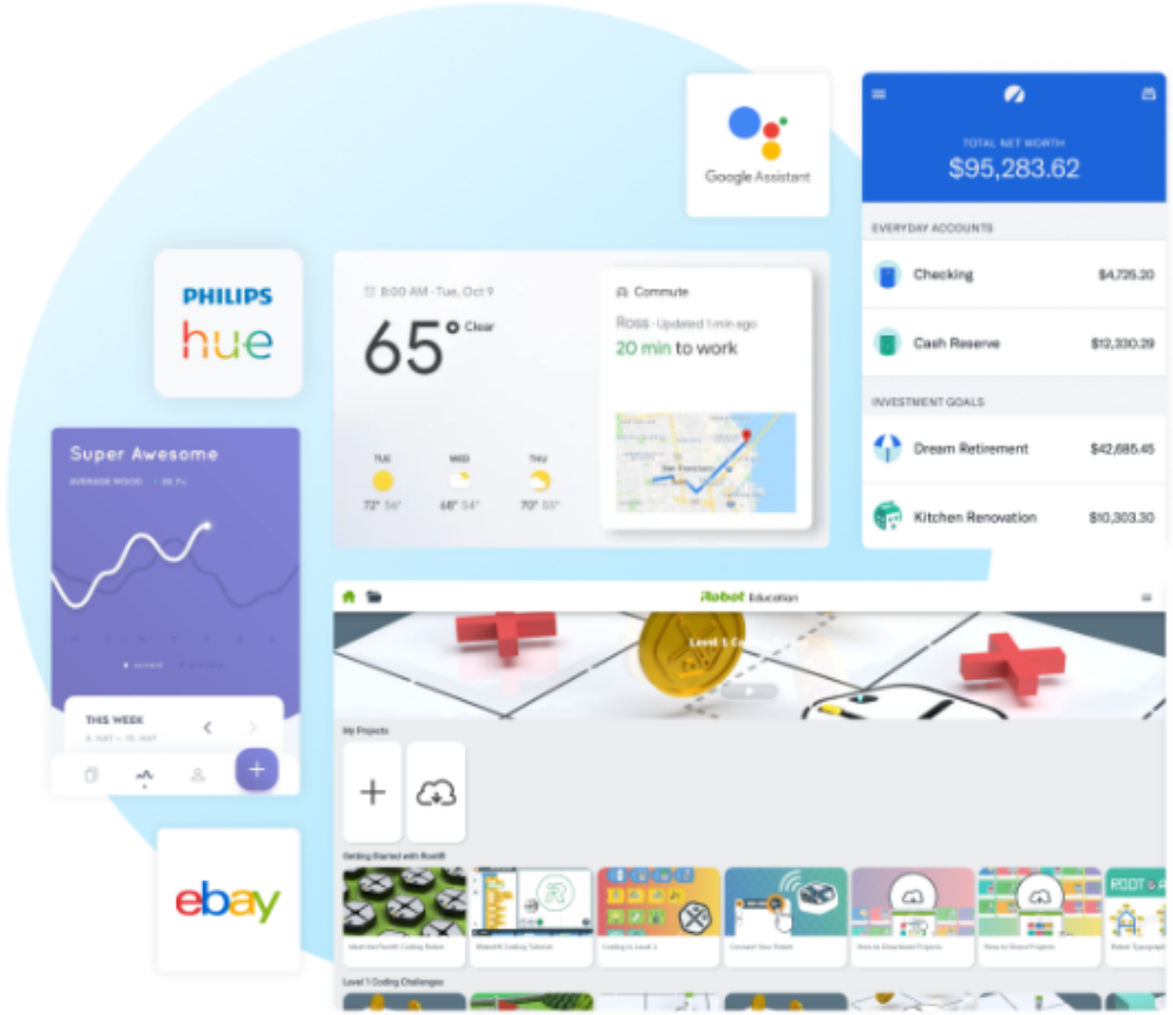


https://docs.flutter.dev/cookbook

Flutter Showcases

Flutter apps in production

Businesses of all sizes around the world are building with Flutter



https://flutter.dev/showcase

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 !!!

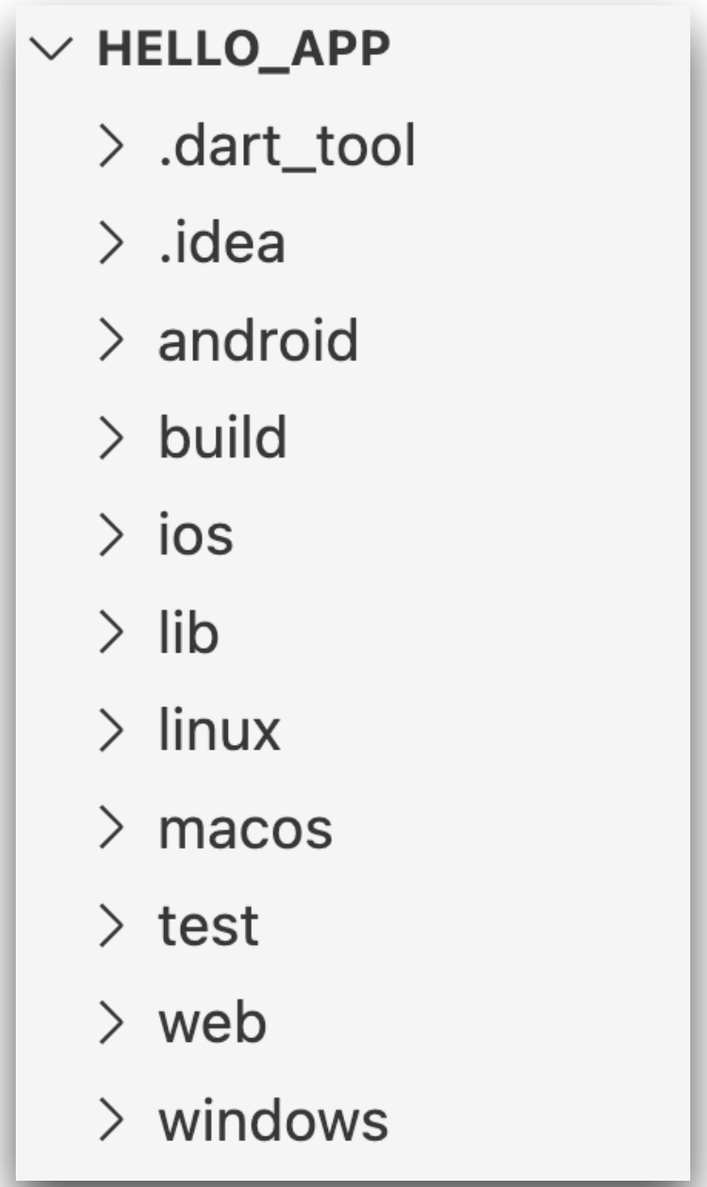
Android Devices

$adb devices

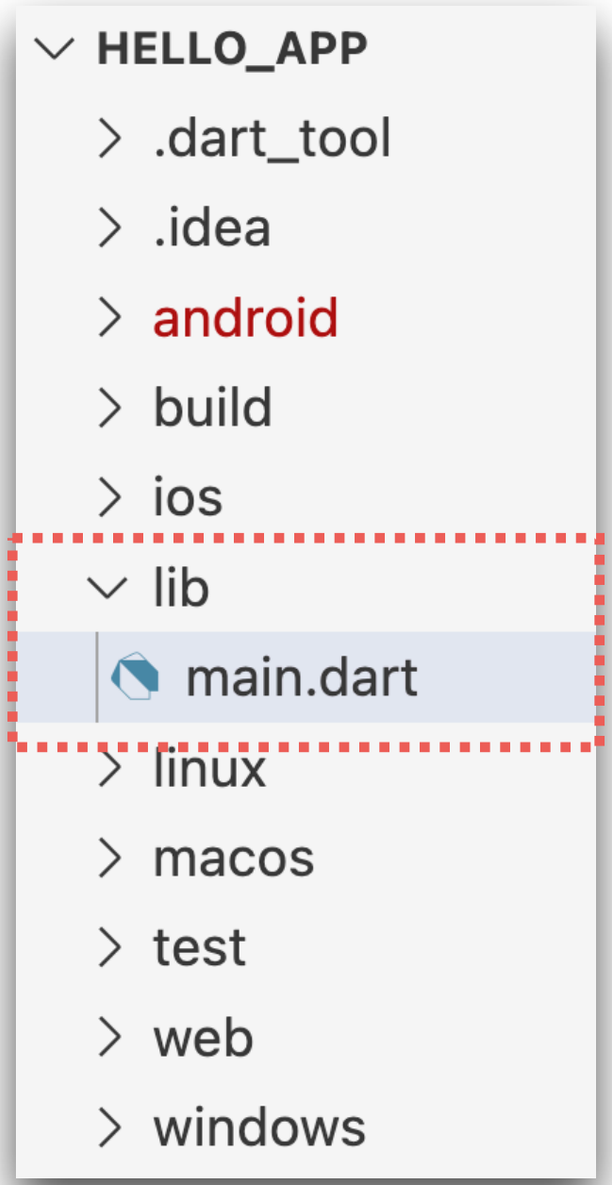
iOS Devices

$xcrun xctrace list devices

Open Project in VS Code



Let’ s start



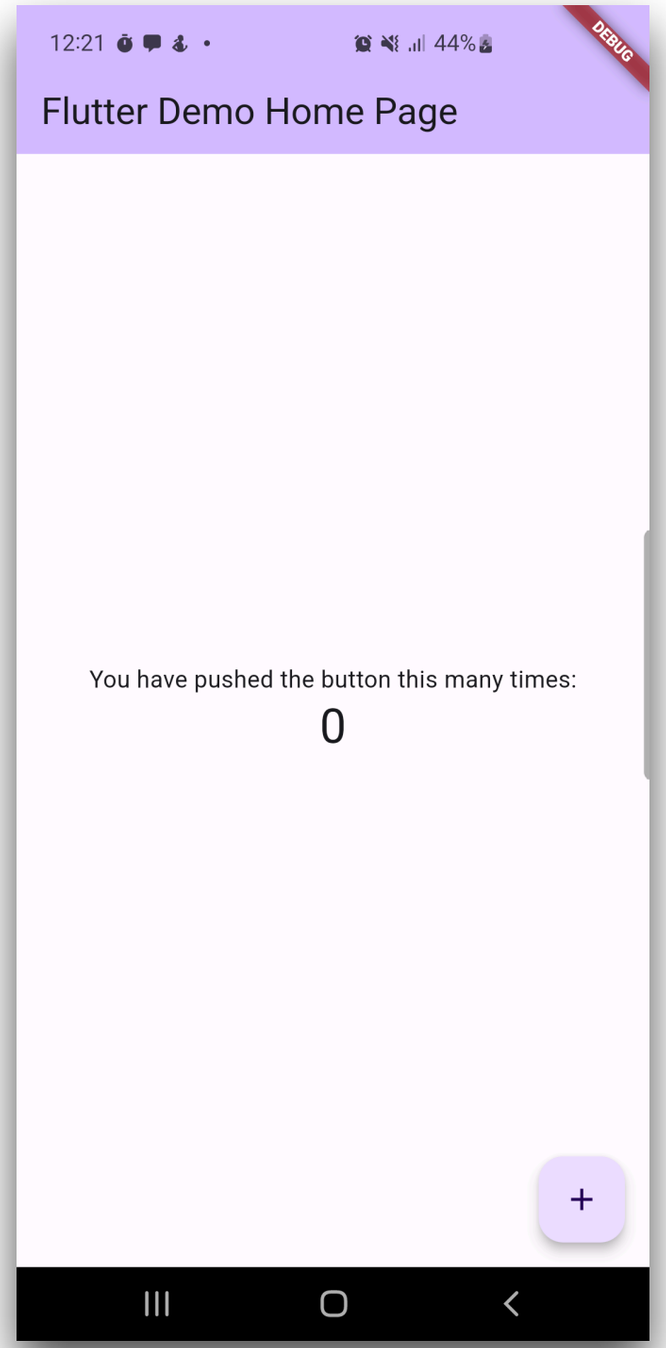
Running project

$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

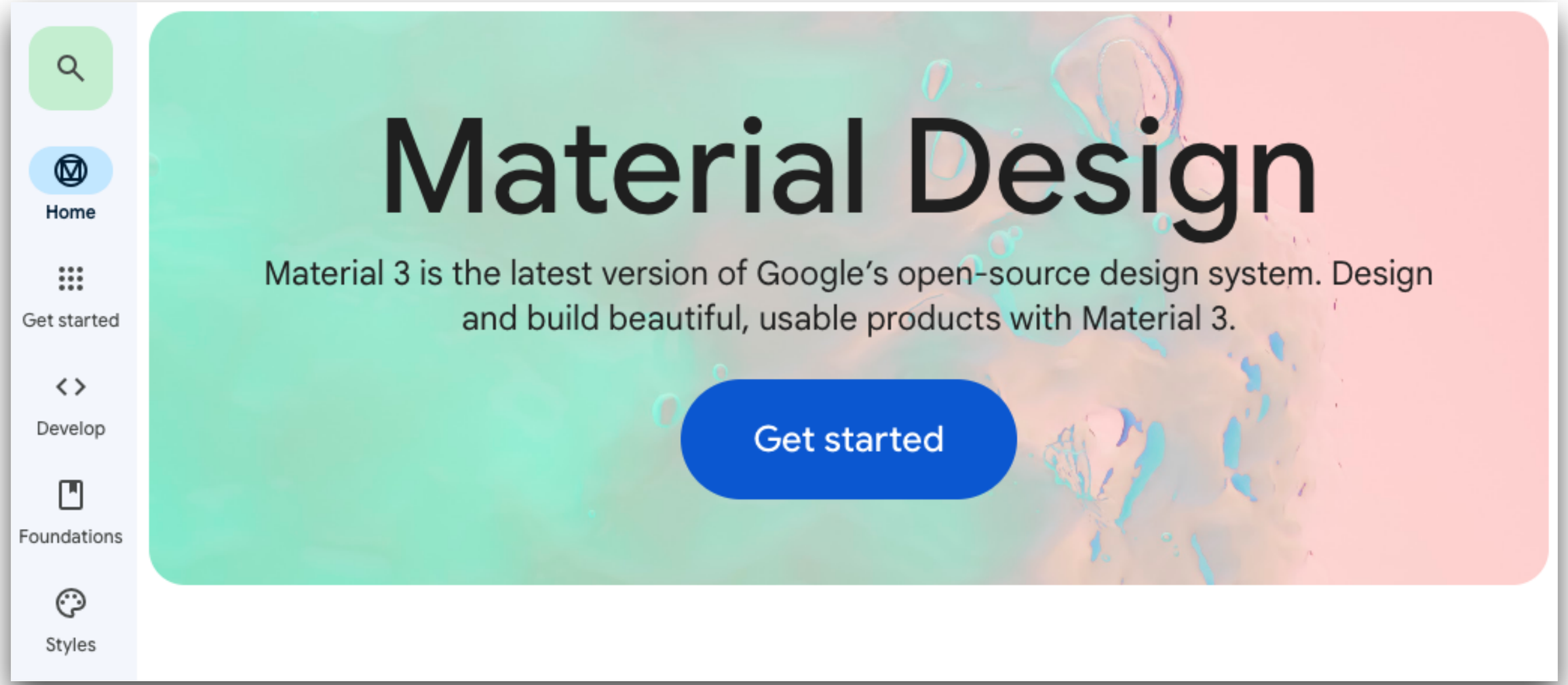
Utilities CLIs from Dart

$dart format

$dart analyze

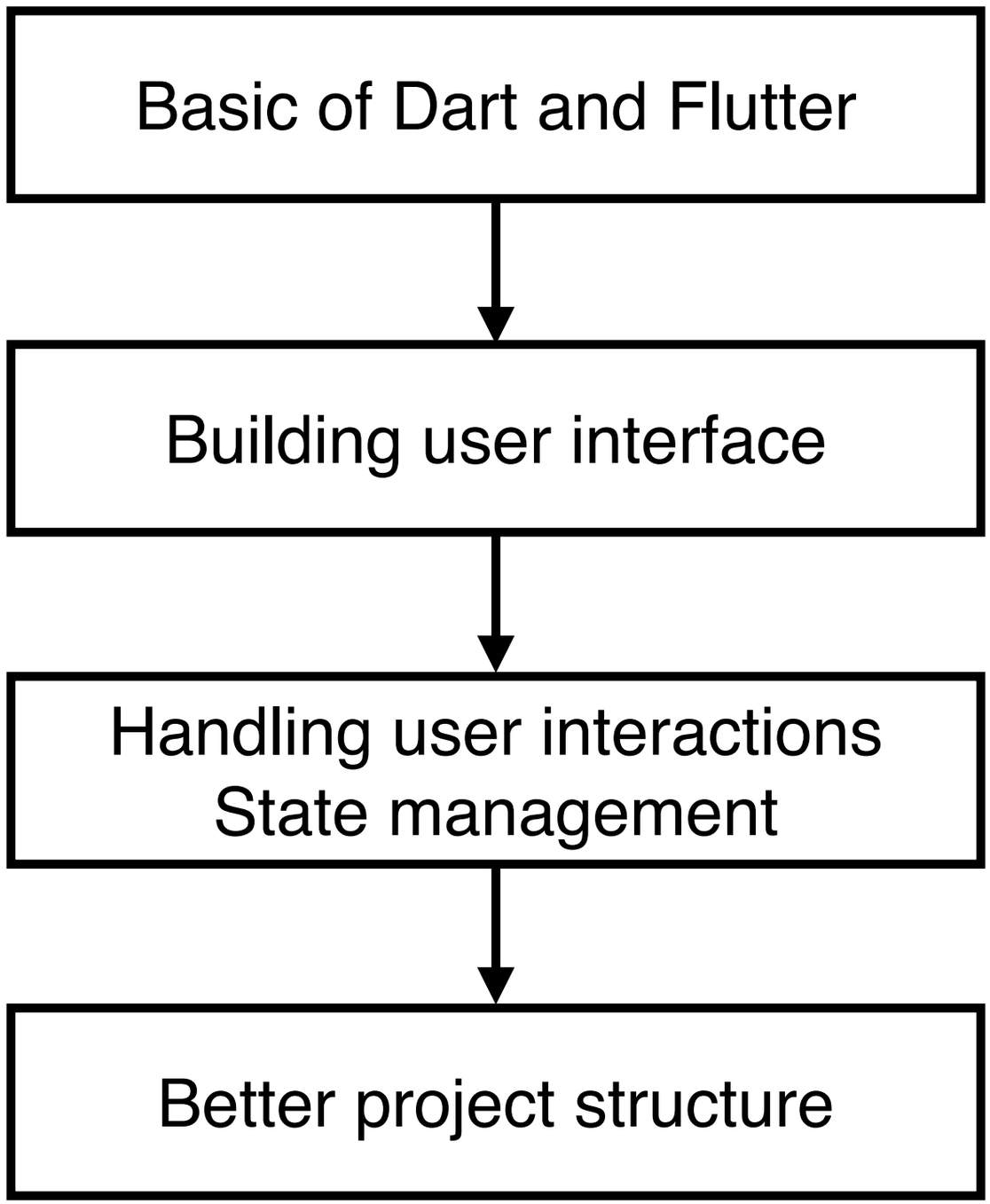
$dart fix

Material Design



https://m3.material.io/

Learning Path



Basic of Flutter and Dart

Basic of Flutter

Start from lib/main.dart

```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

```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(
```

Flutter UIs are build with Widget

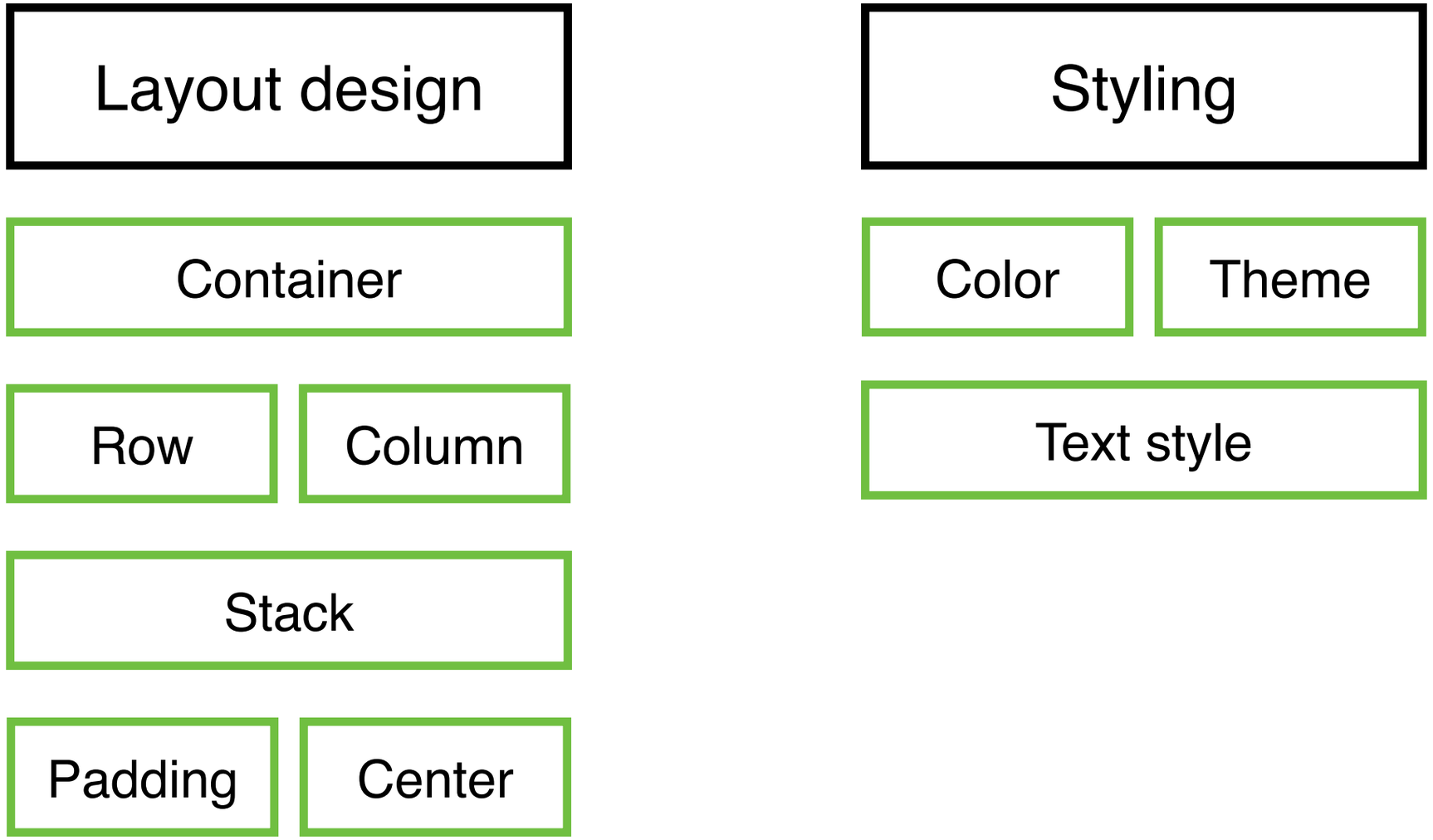
Build-in Widgets

Widgets

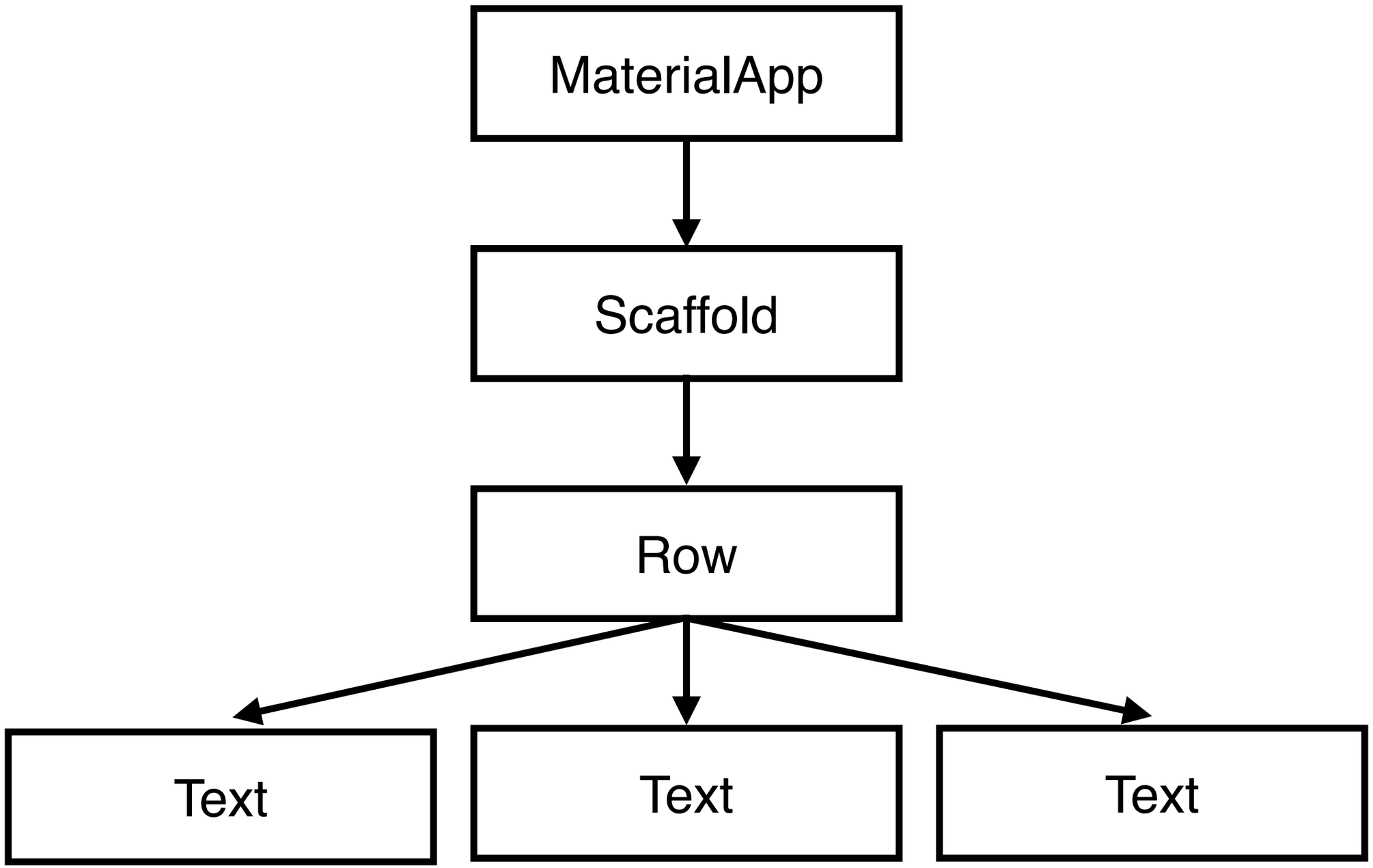
Building blocks that can be connected together

Reusable by nature

Essential of Widgets



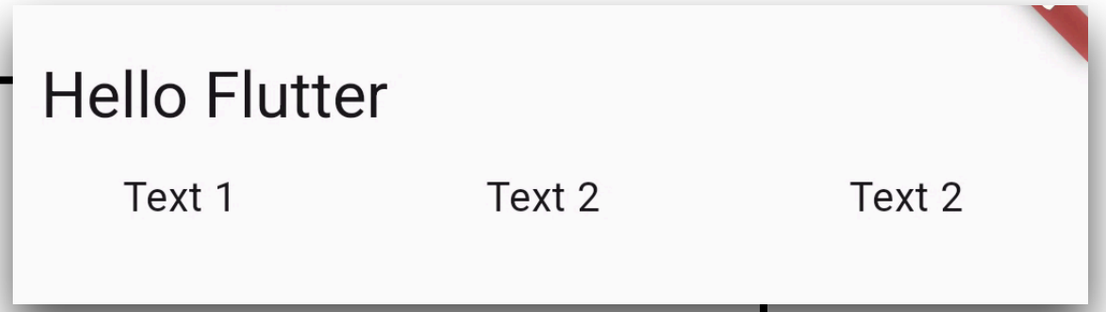
Widget tree



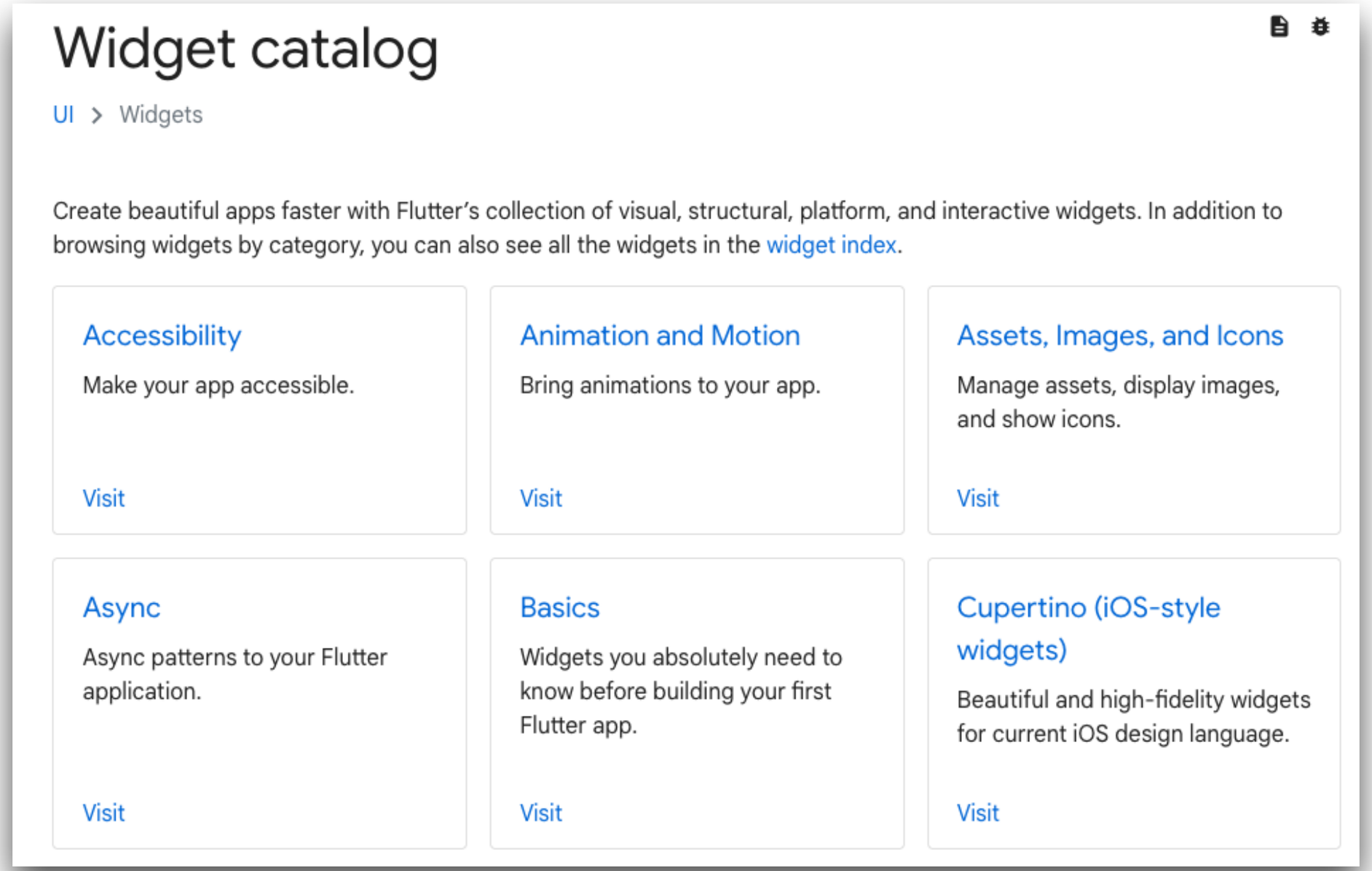
https://docs.flutter.dev/ui

Code Example

```dart
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),
            ),
          ],
        ),
      )));
}
```



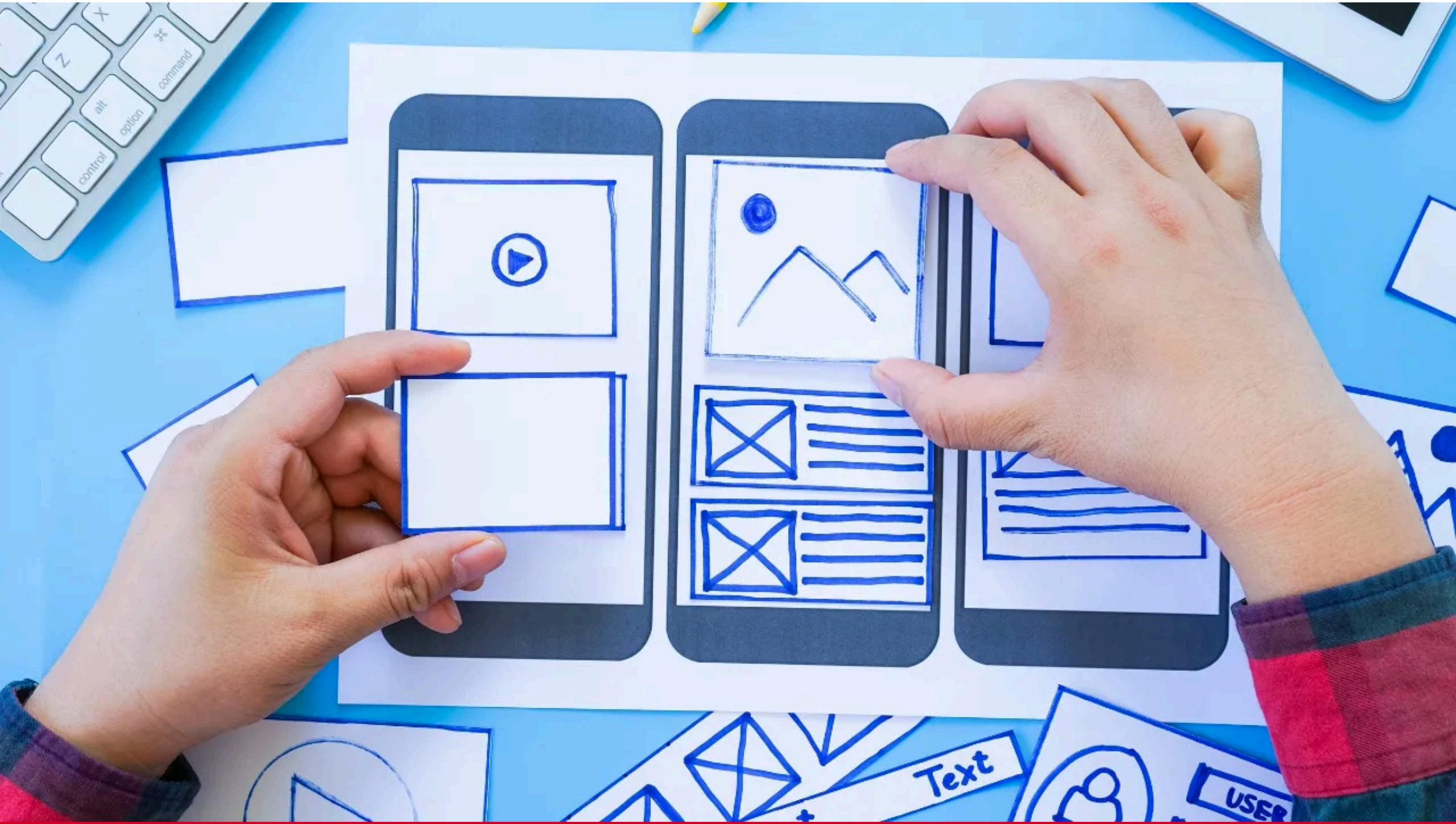
Widget Catalogs



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

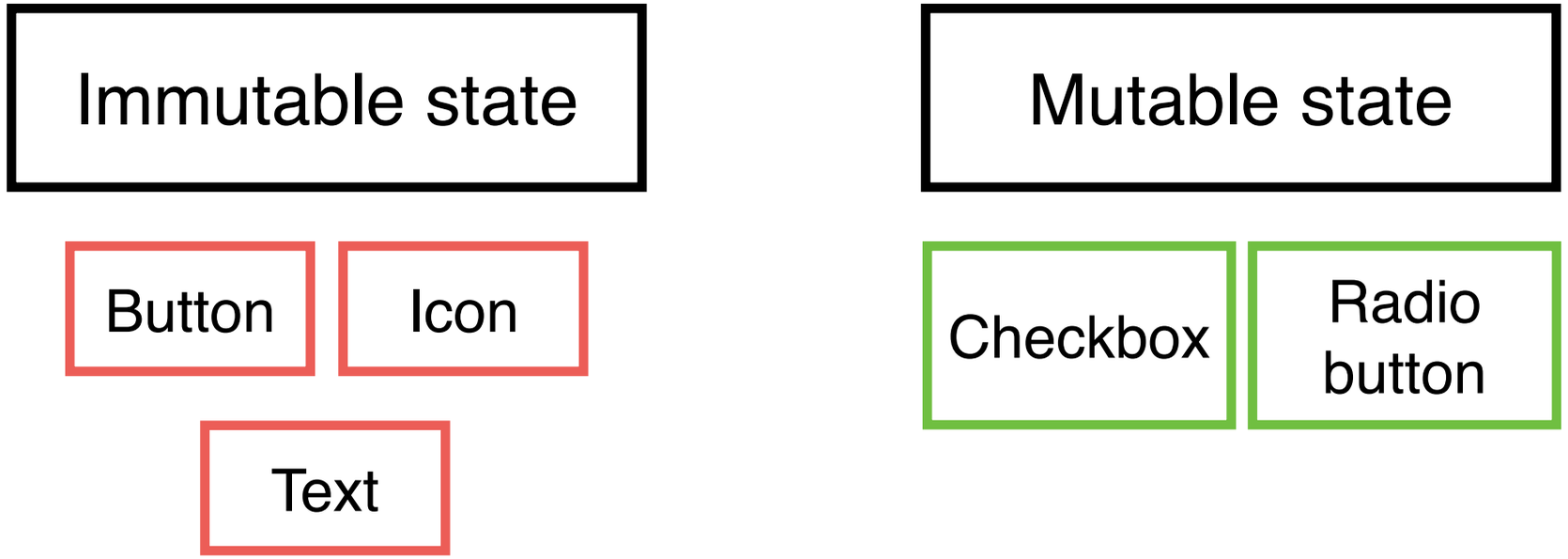
Types of Widgets

Types of Widgets



Types of Widgets

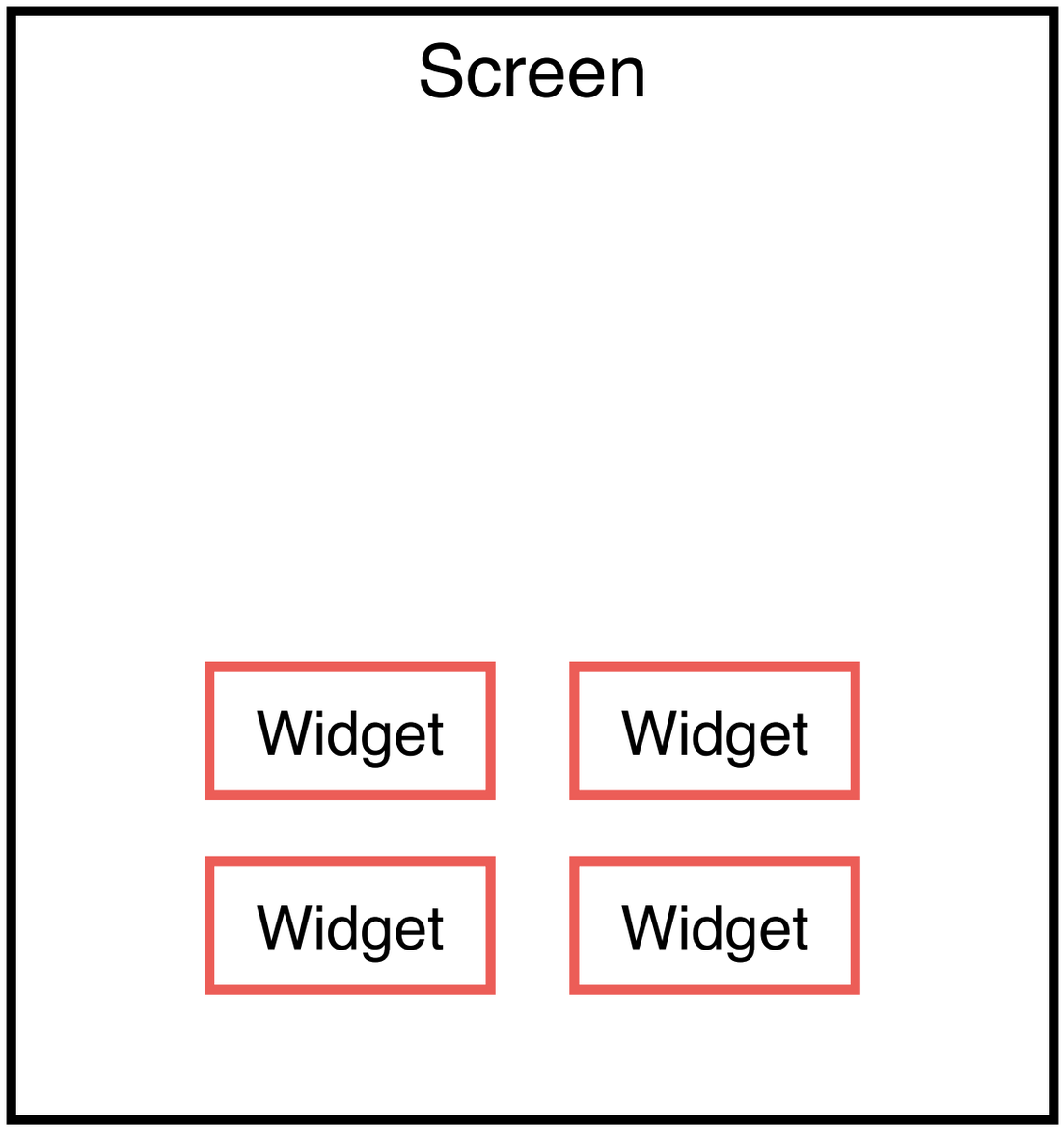
Stateless Widget
Stateful Widget



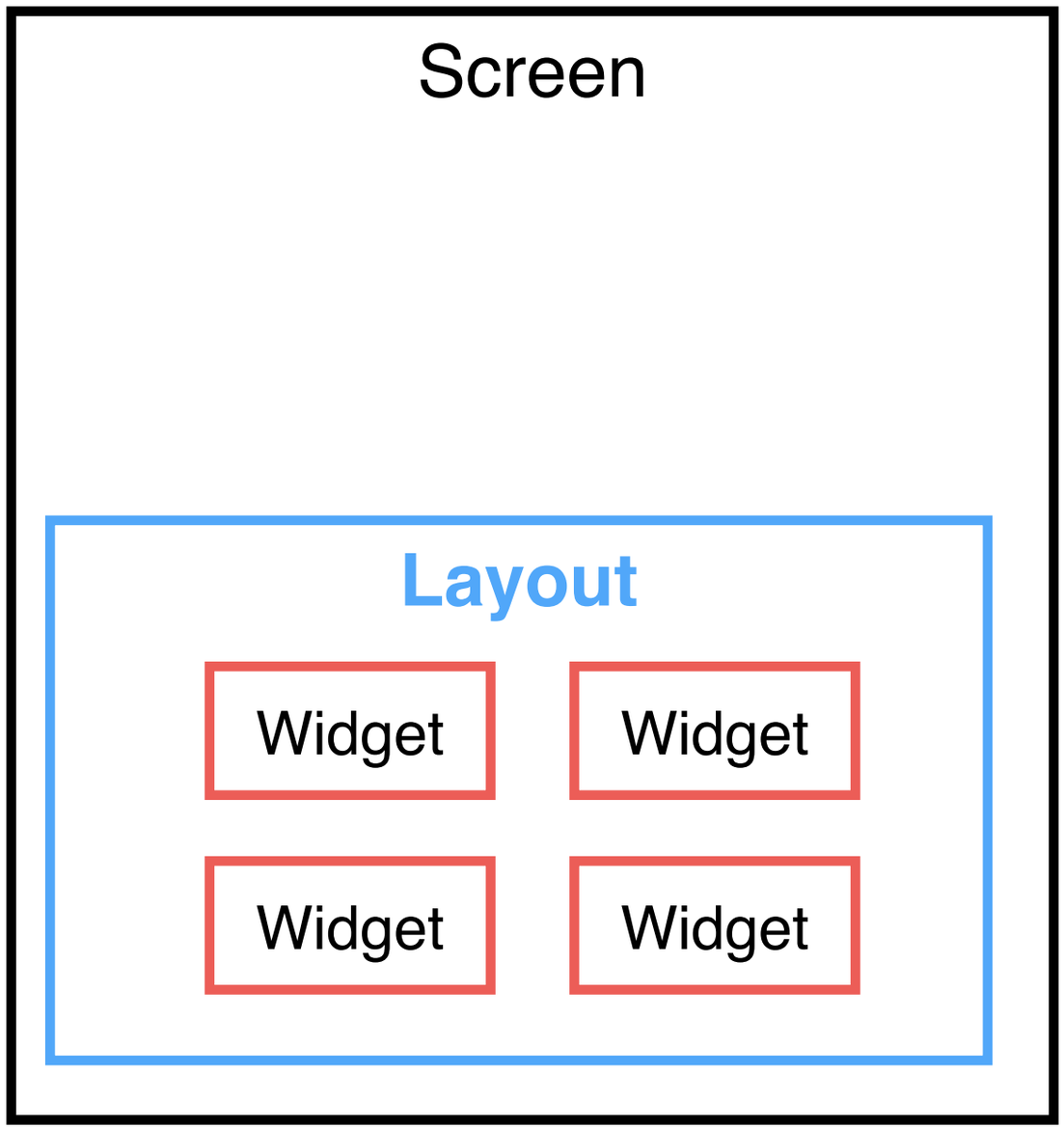
Stateless Widget

```dart
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

Layout in Flutter app

List and Grid

Working with long list

Scrolling

Adaptive and responsive design

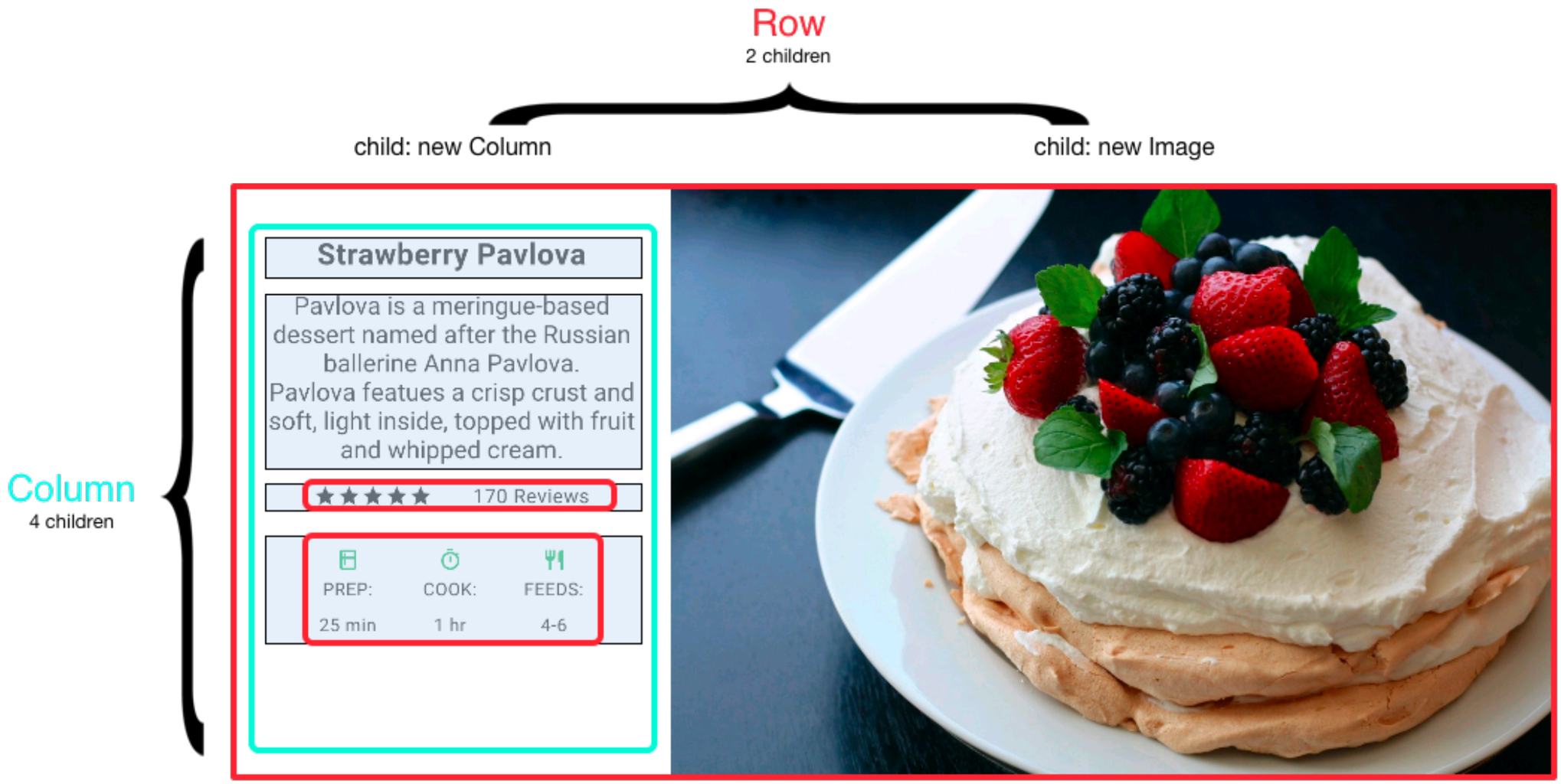
Container

Stack

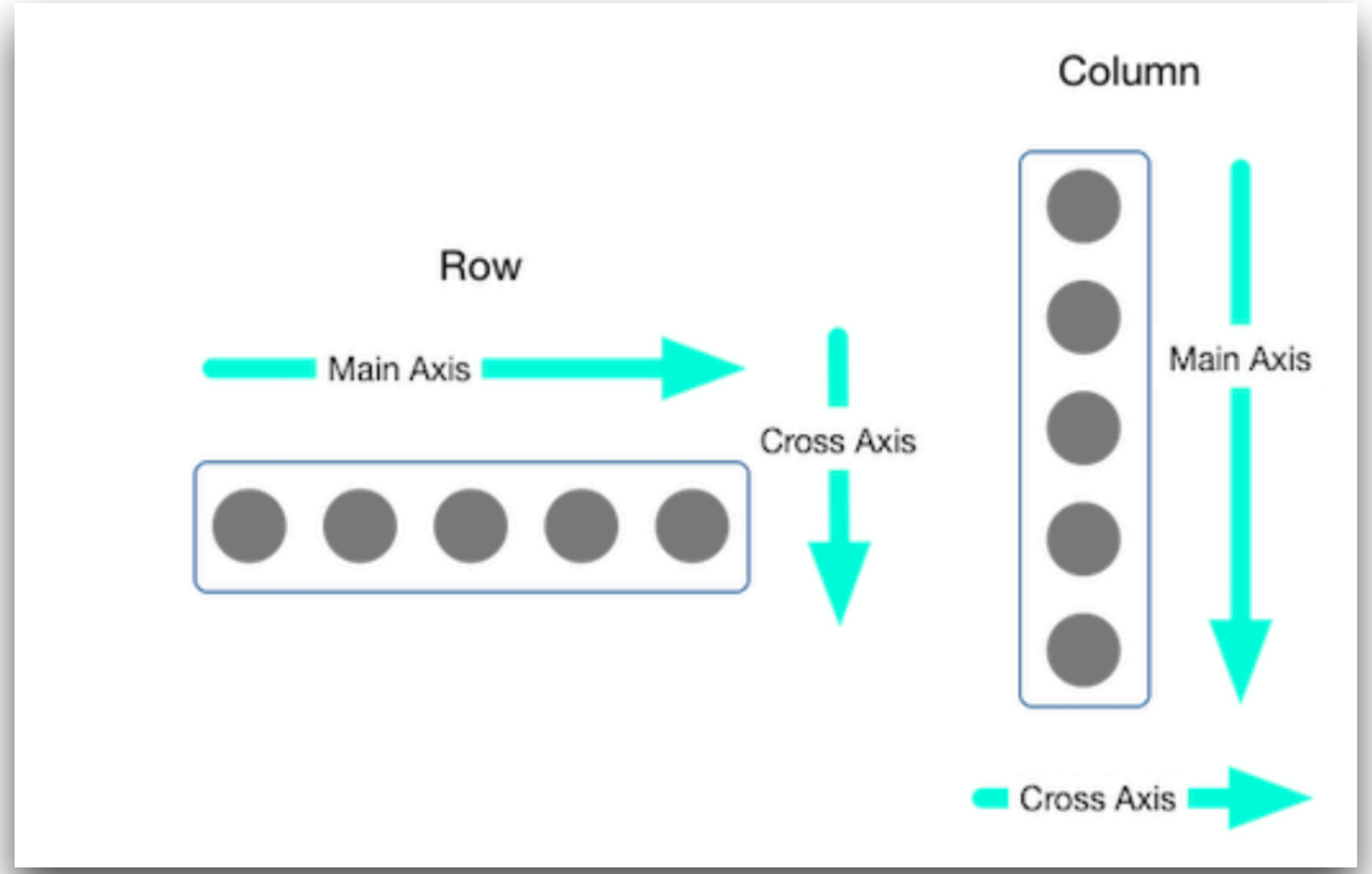
Card

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

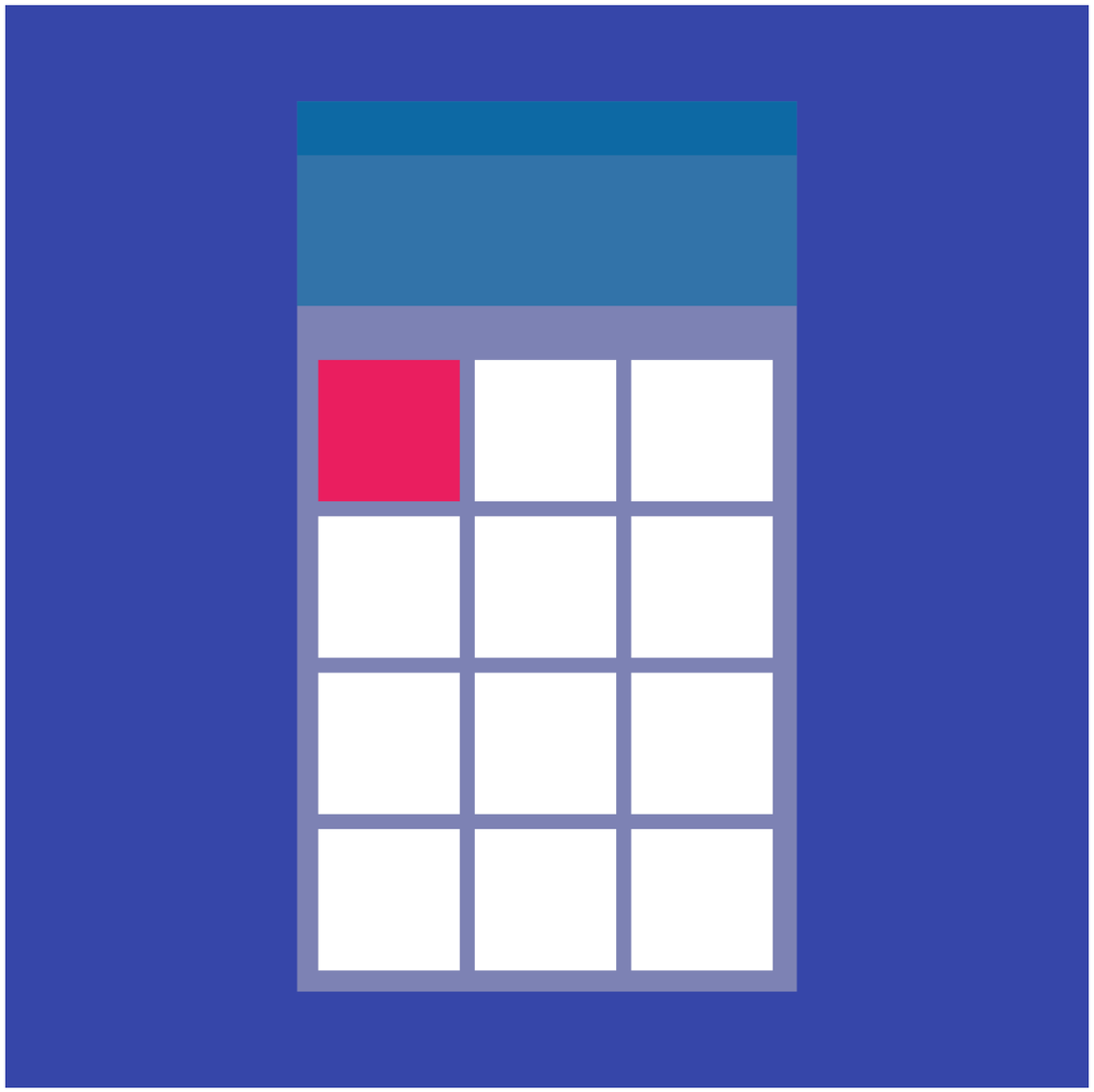
Row and Column



Align of widget

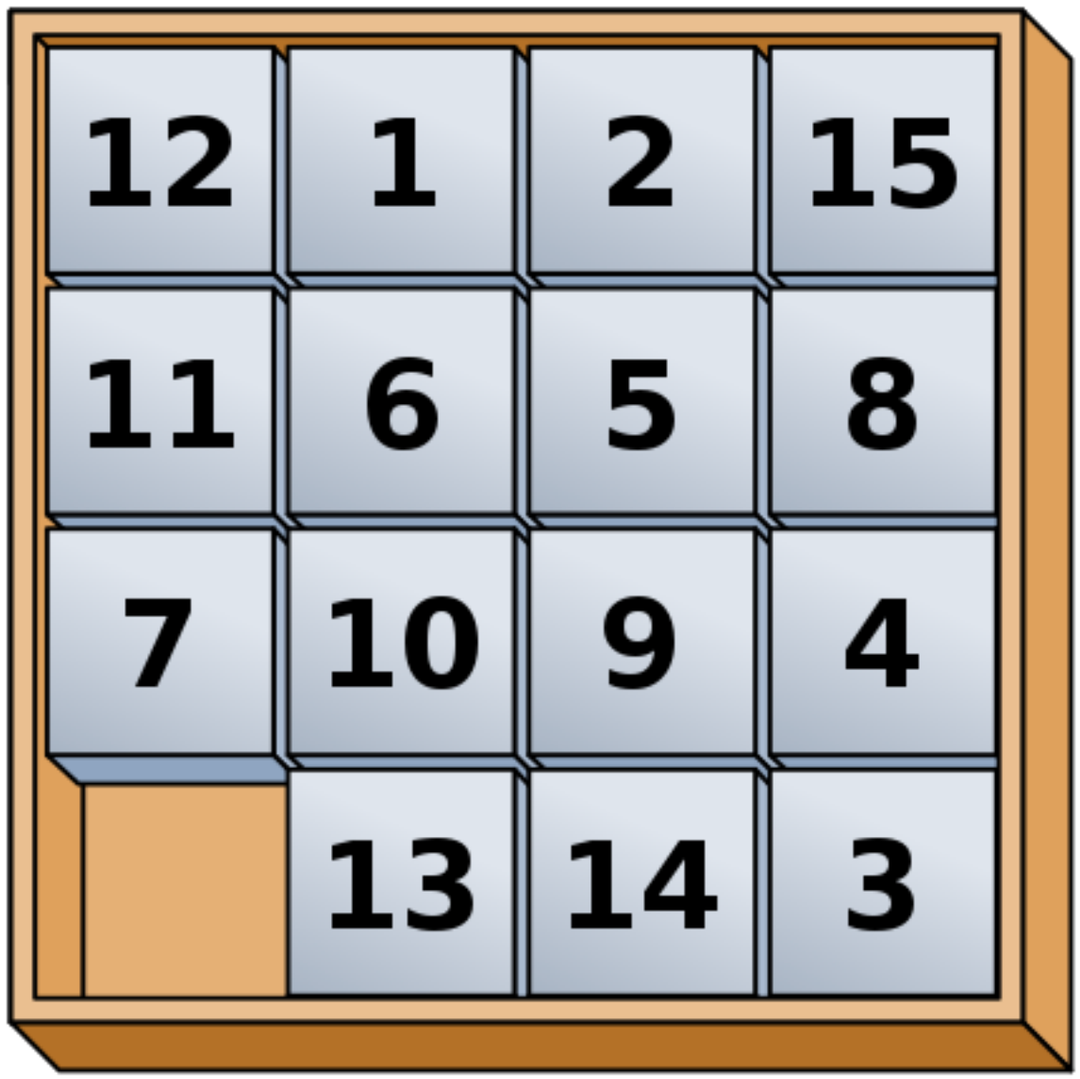


GridView



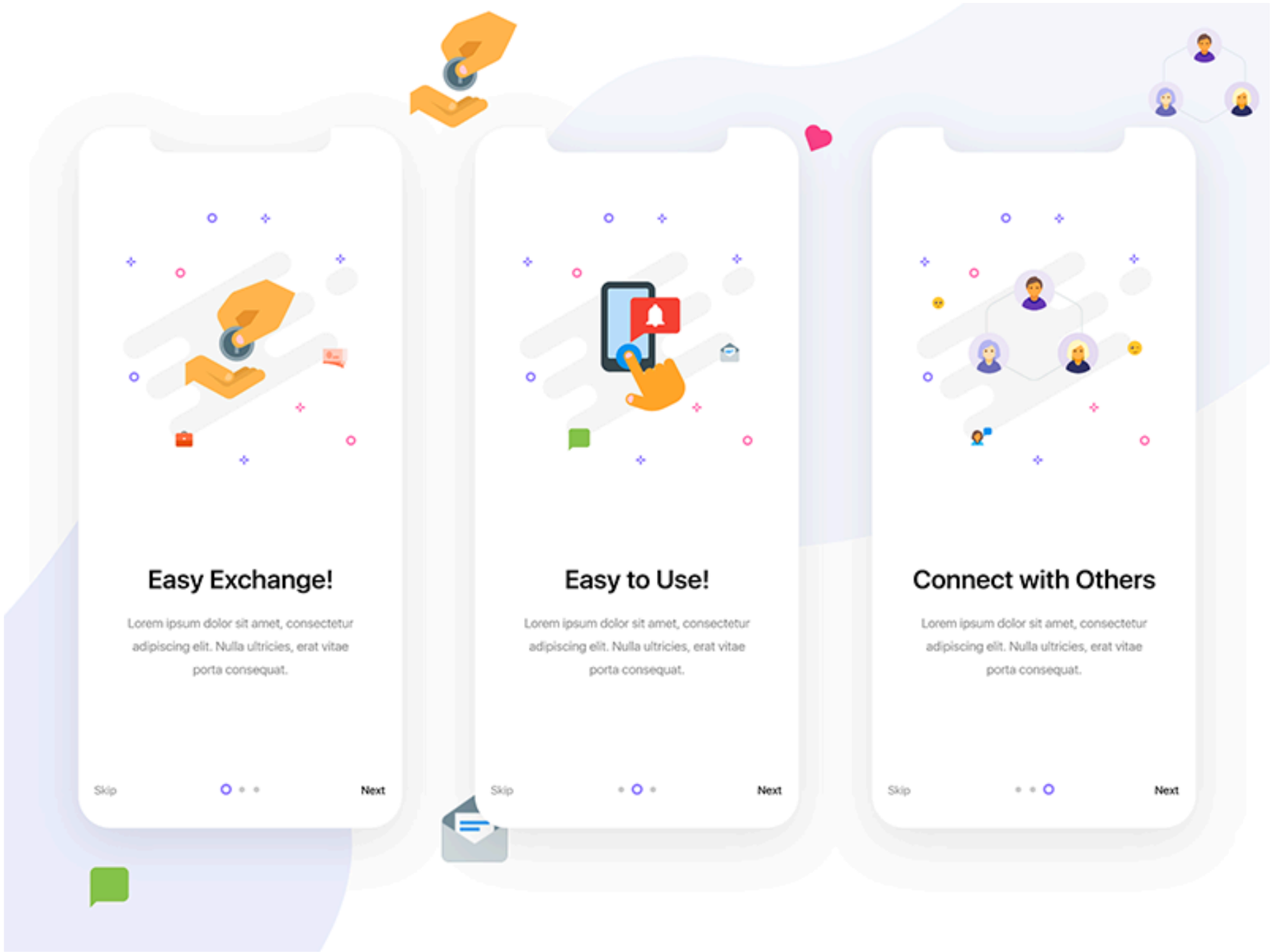
GridView

15 Puzzle

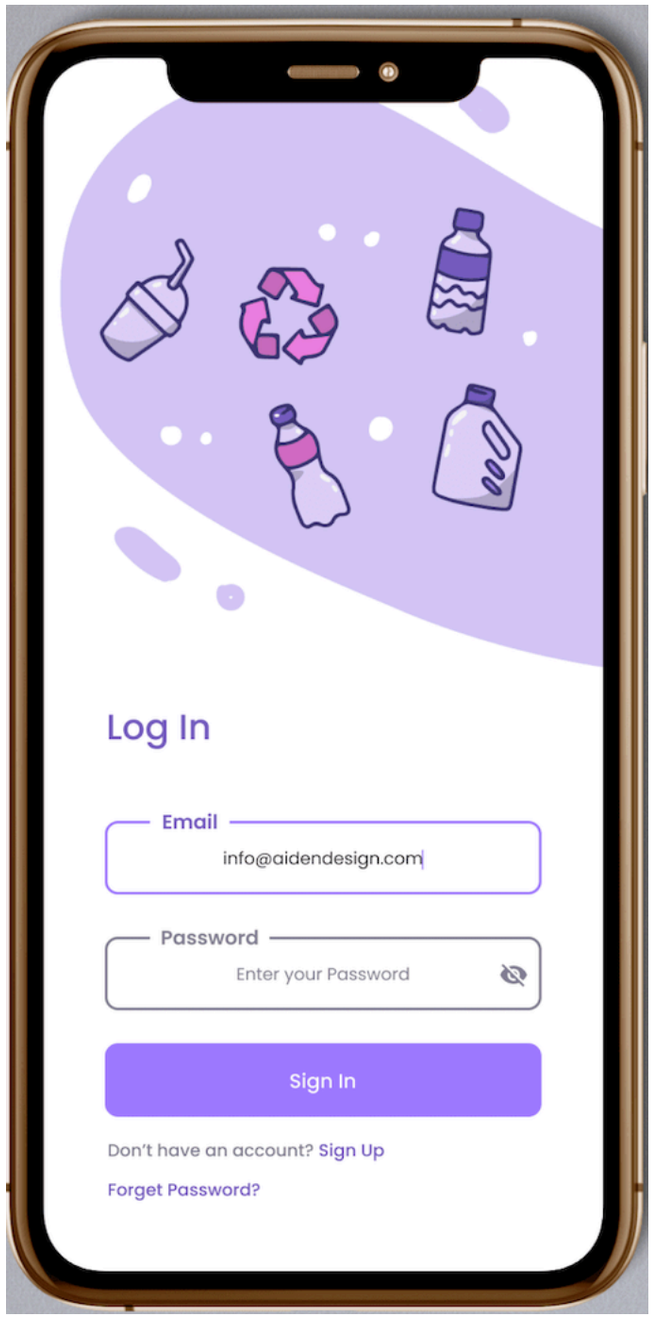


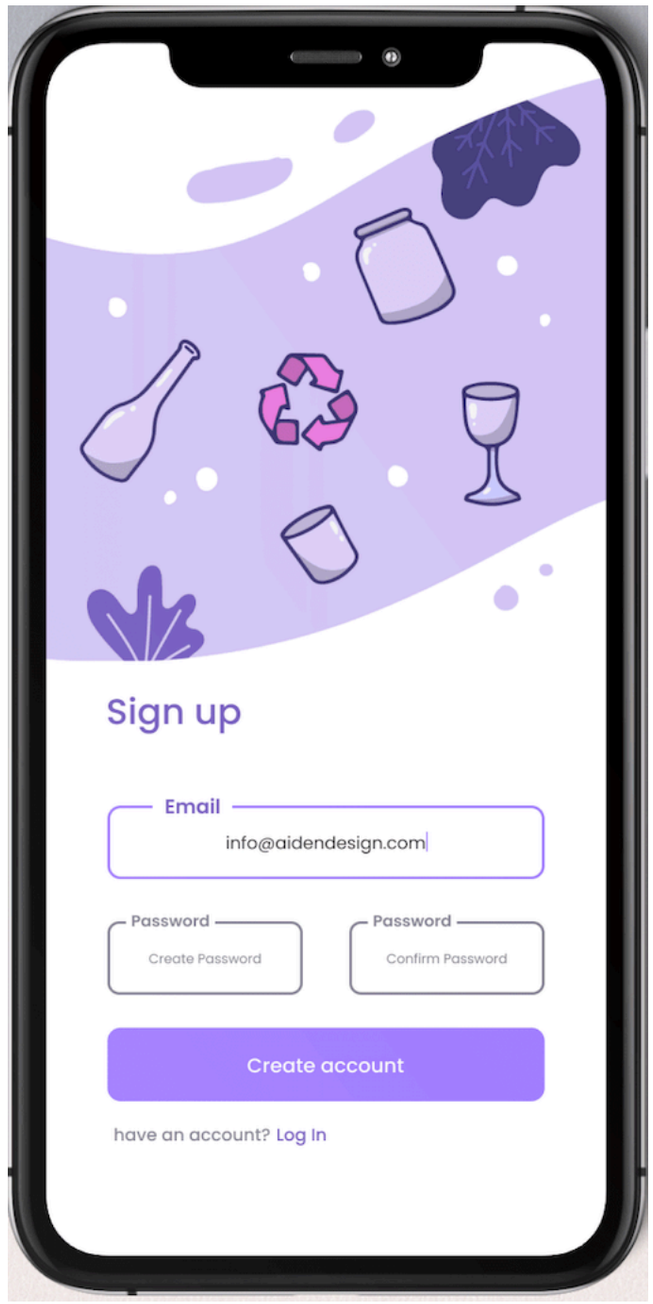
https://en.wikipedia.org/wiki/15_Puzzle

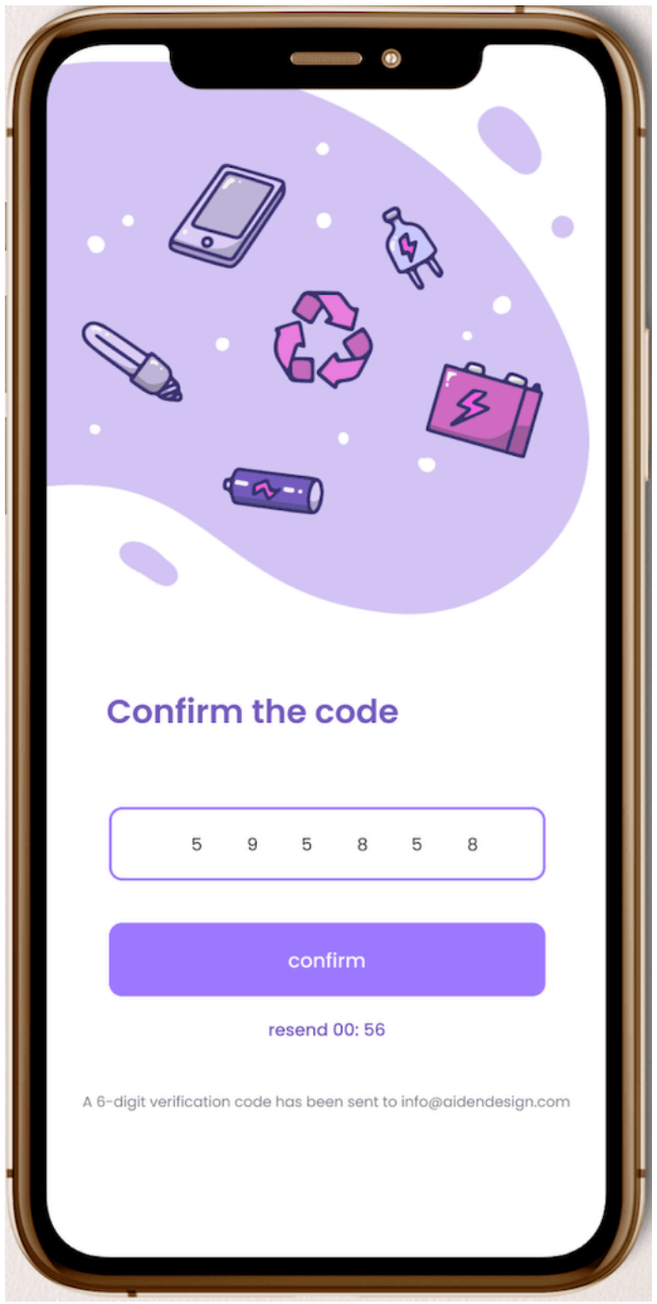
Onboarding screen



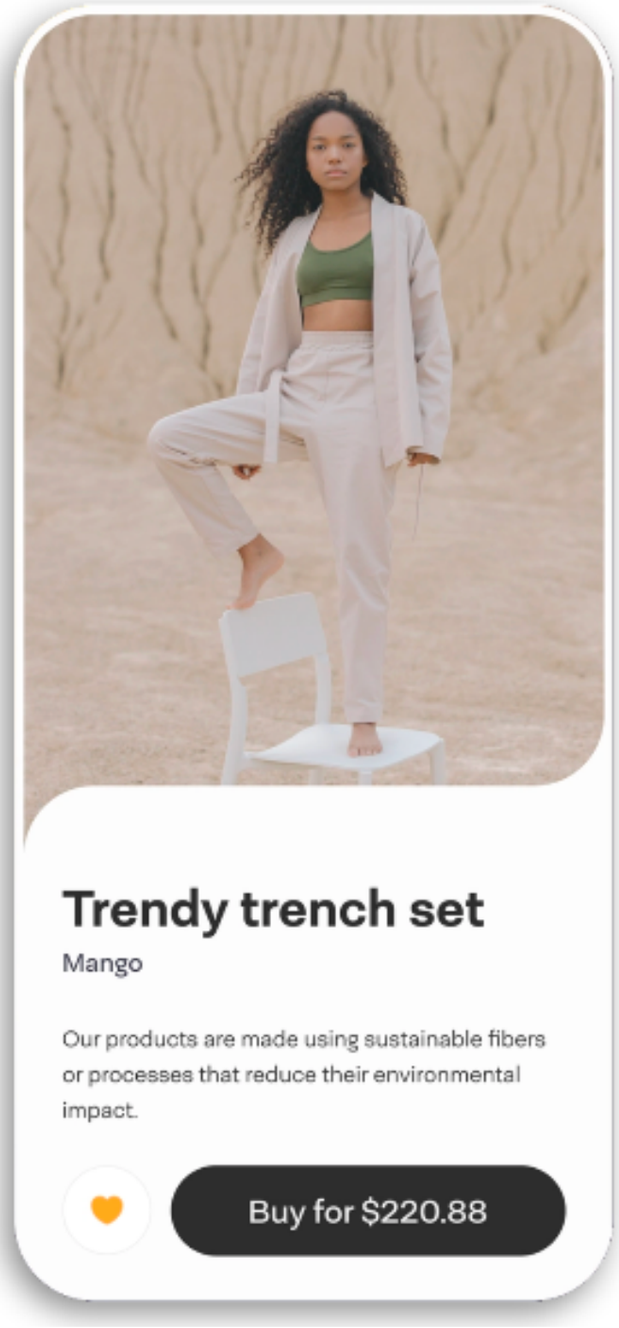
Login



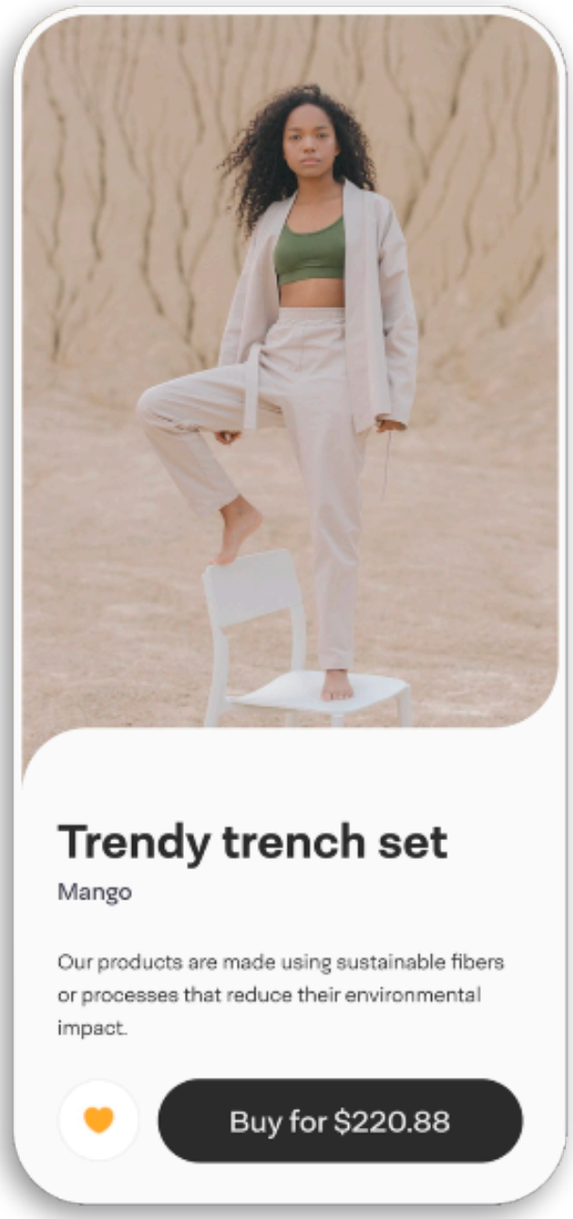


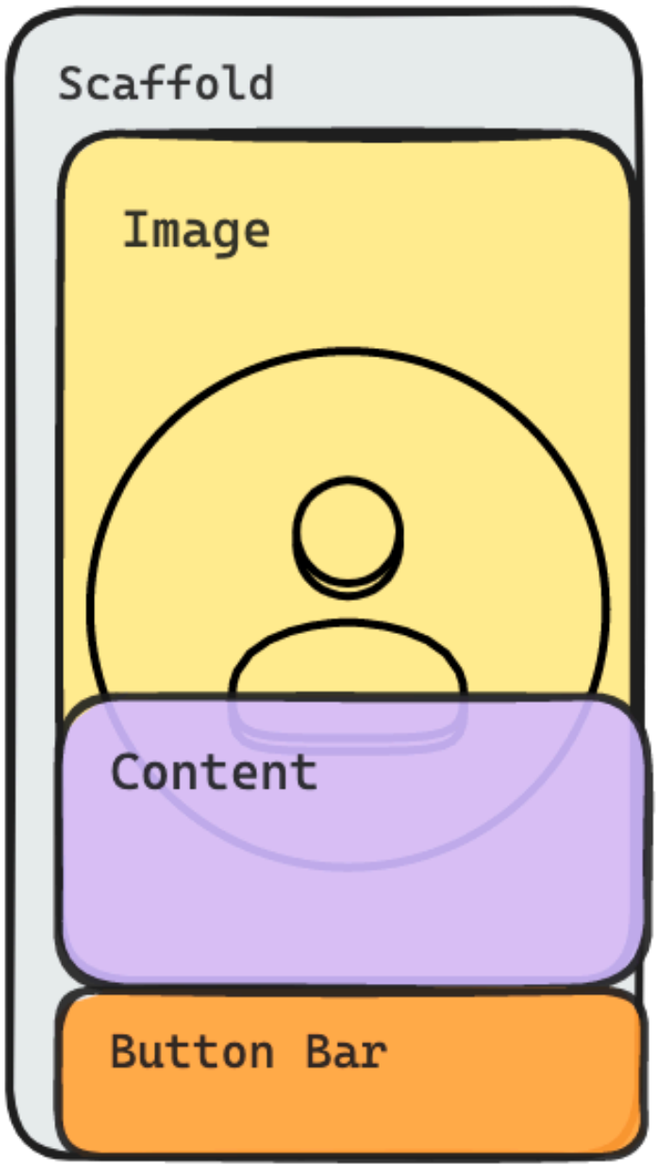


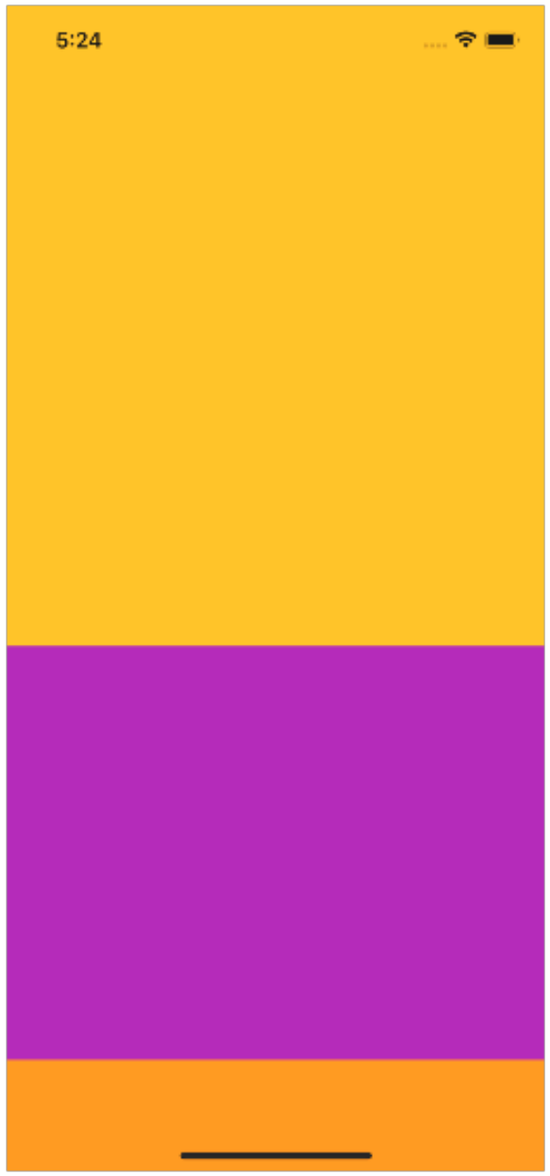
Product Page



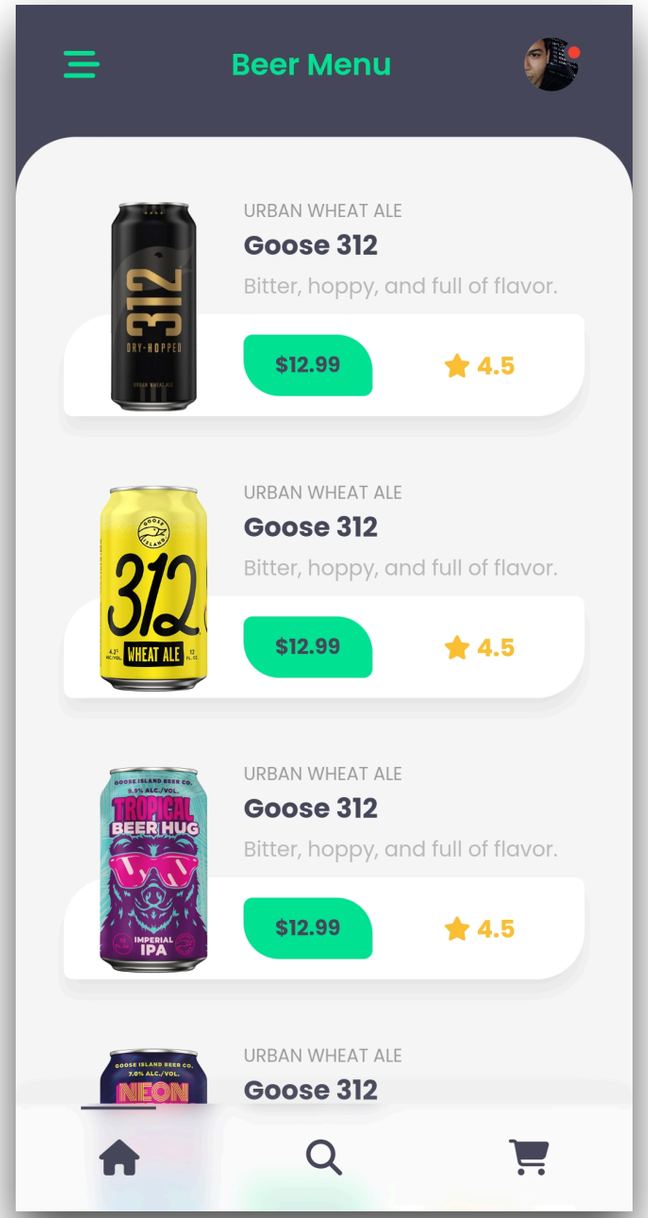
Product Page

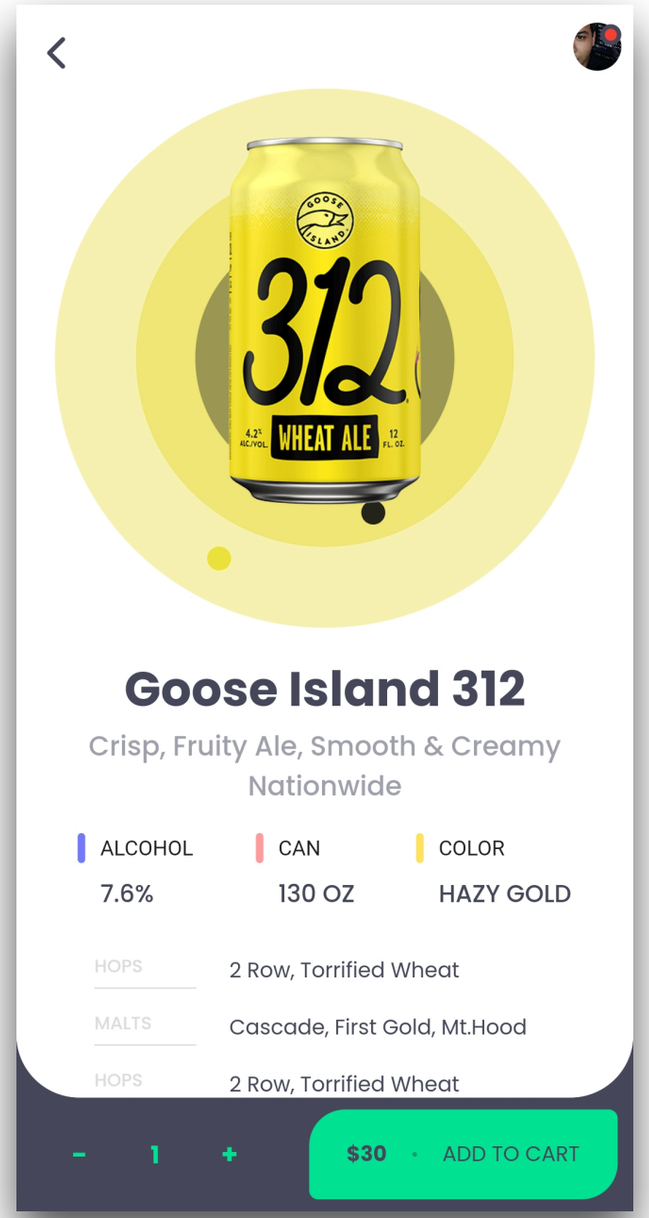






Listing and Detail

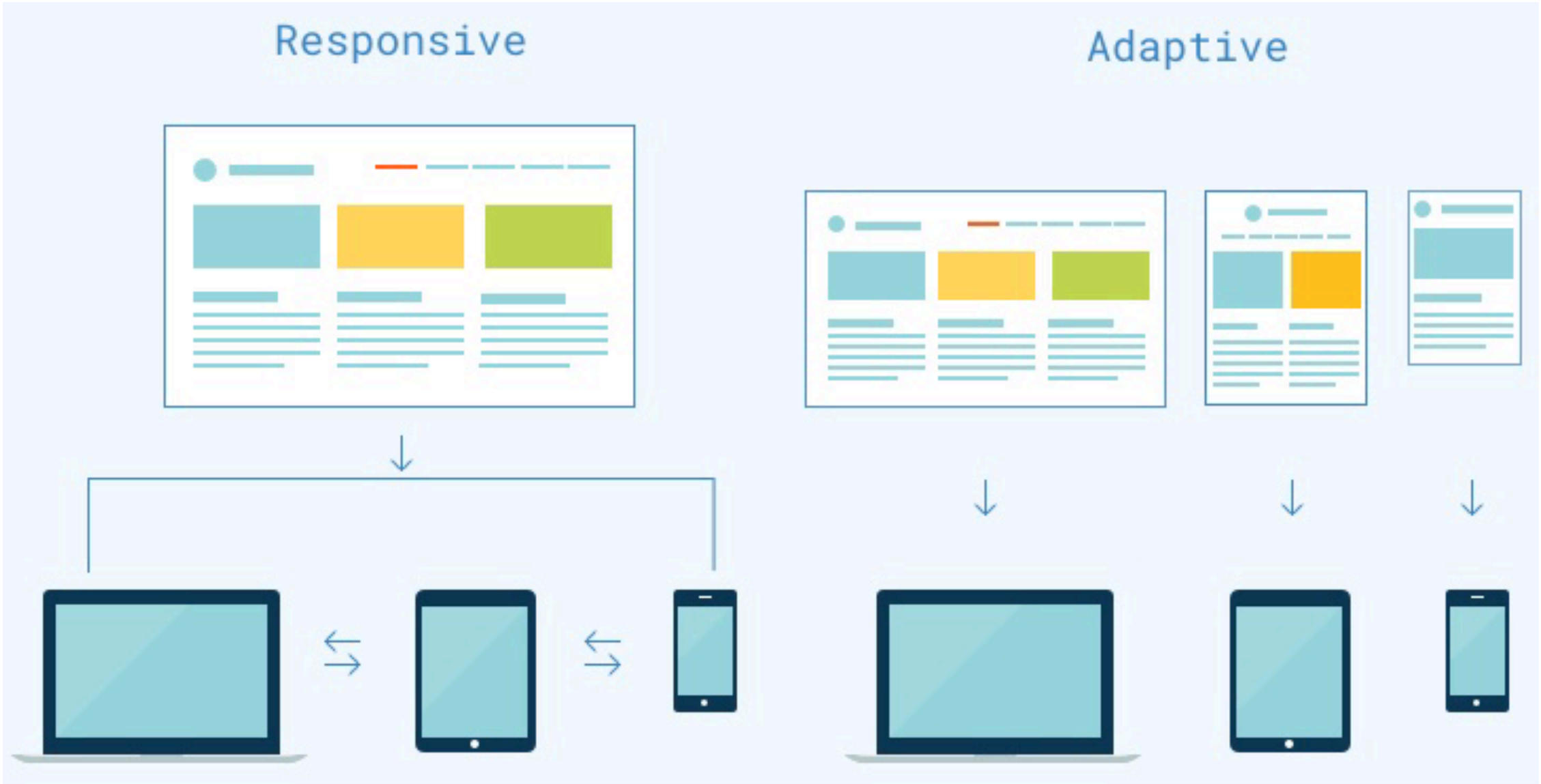




Adaptive vs Responsive App

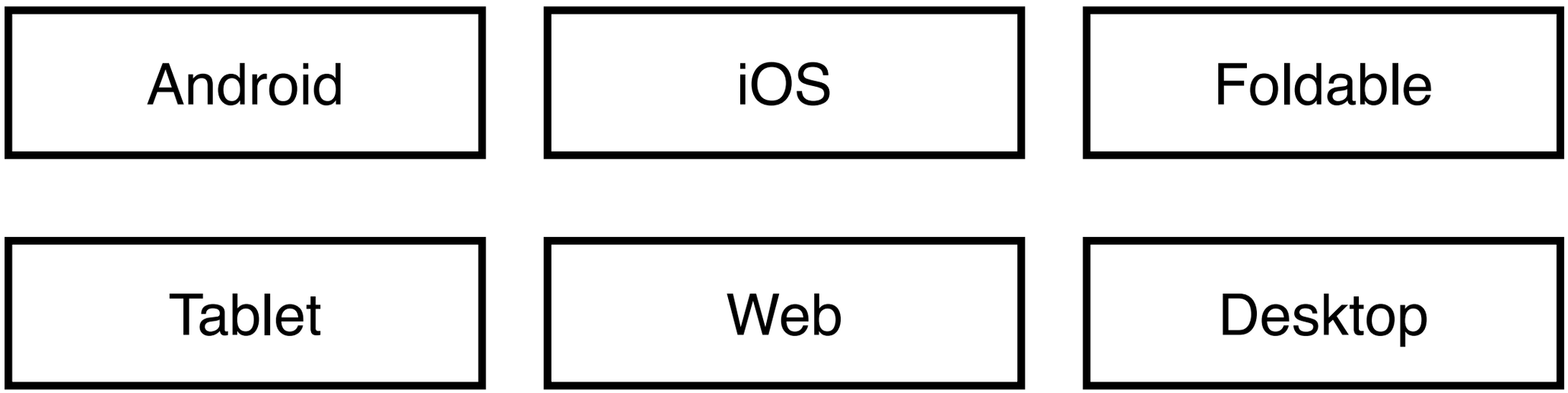
https://docs.flutter.dev/ui/layout/responsive/adaptive-responsive

Adaptive vs Responsive App



Adaptive

Change functionality based
on platform's capabilities



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**

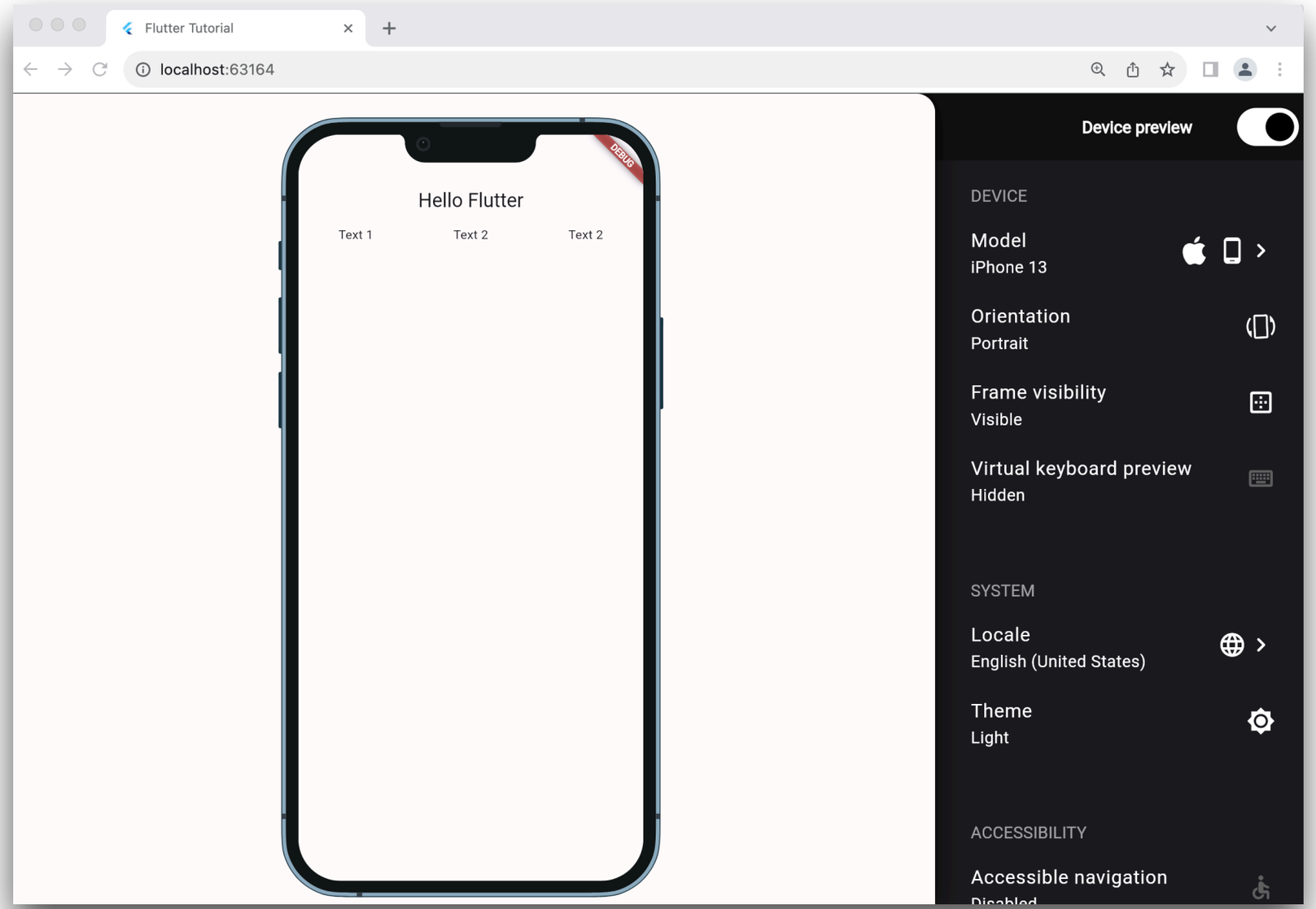
Size	Orientation	Device type

Responsive UI !!

Flutter Device Review

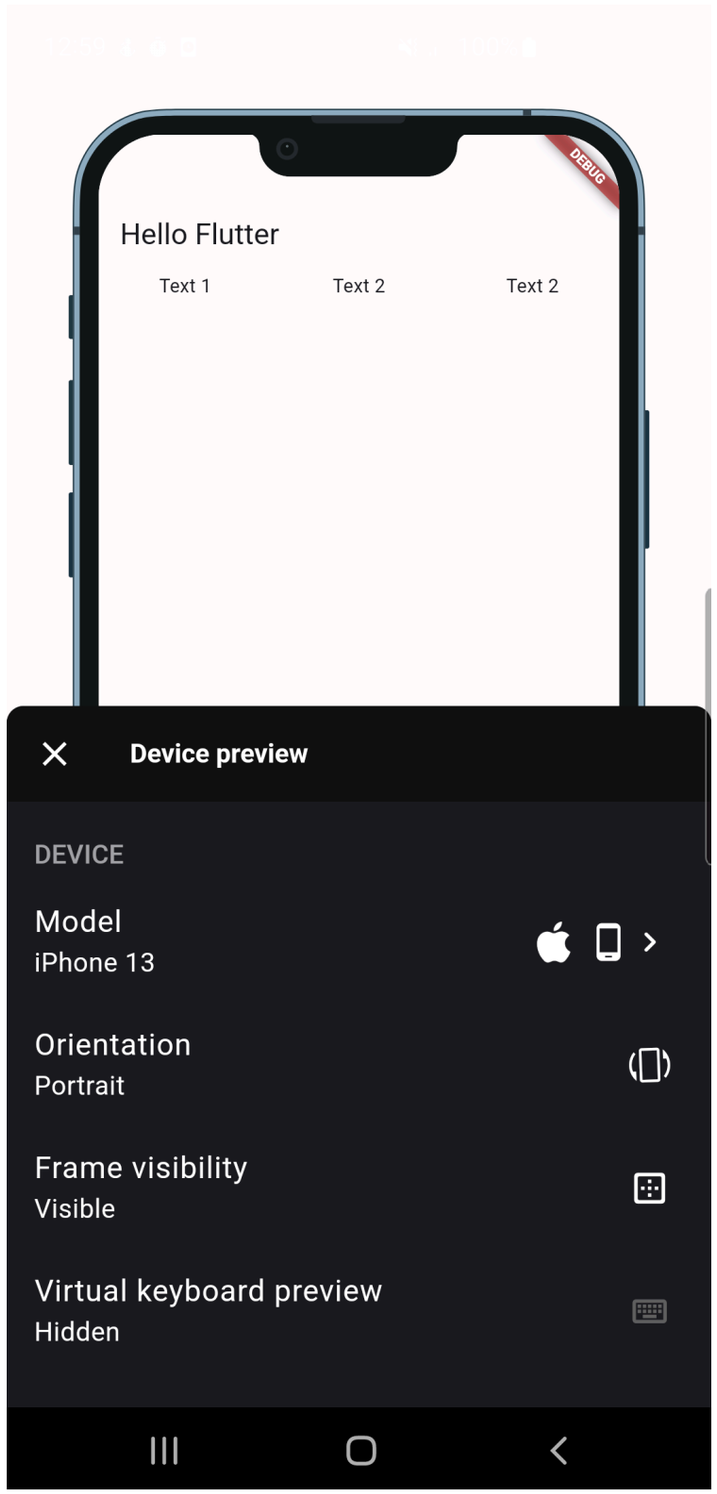
Flutter Device Review

Preview on any device and orientation



https://pub.dev/packages/device_preview

Flutter Device Review



Flutter Device Review

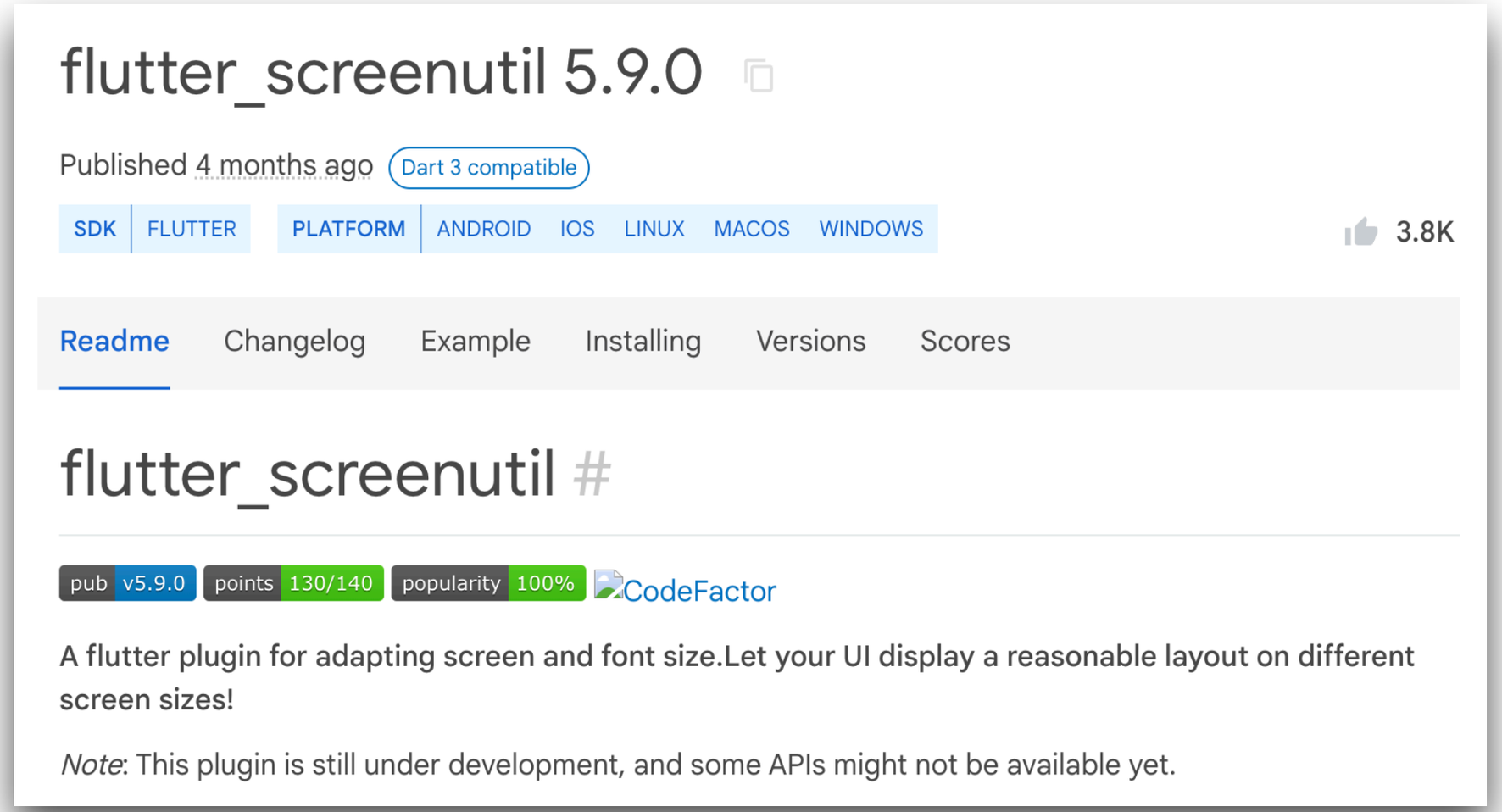
$flutter pub add device_preview

```dart
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

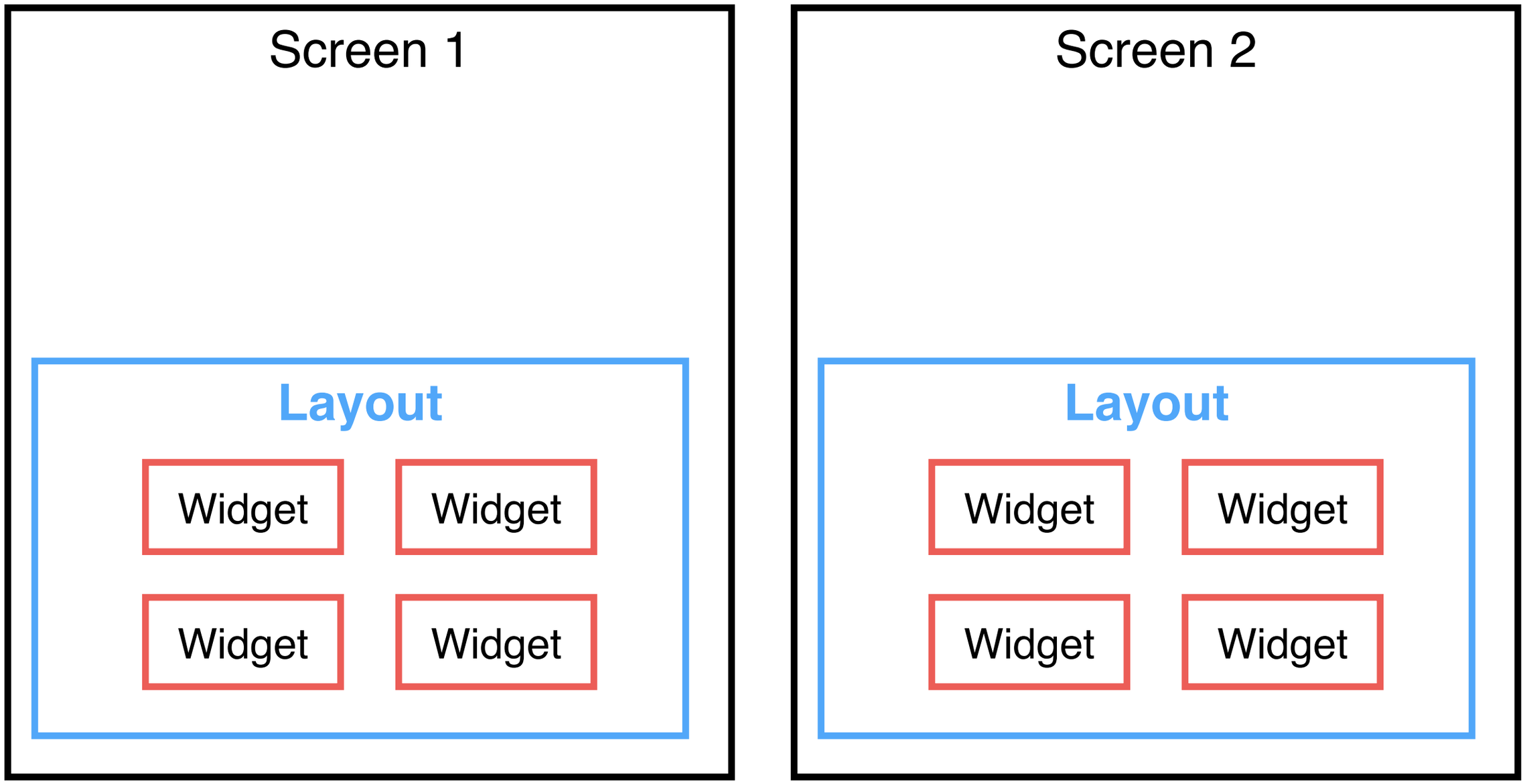
Using Flutter ScreenUtil

UI display a reasonable layout on difference screen

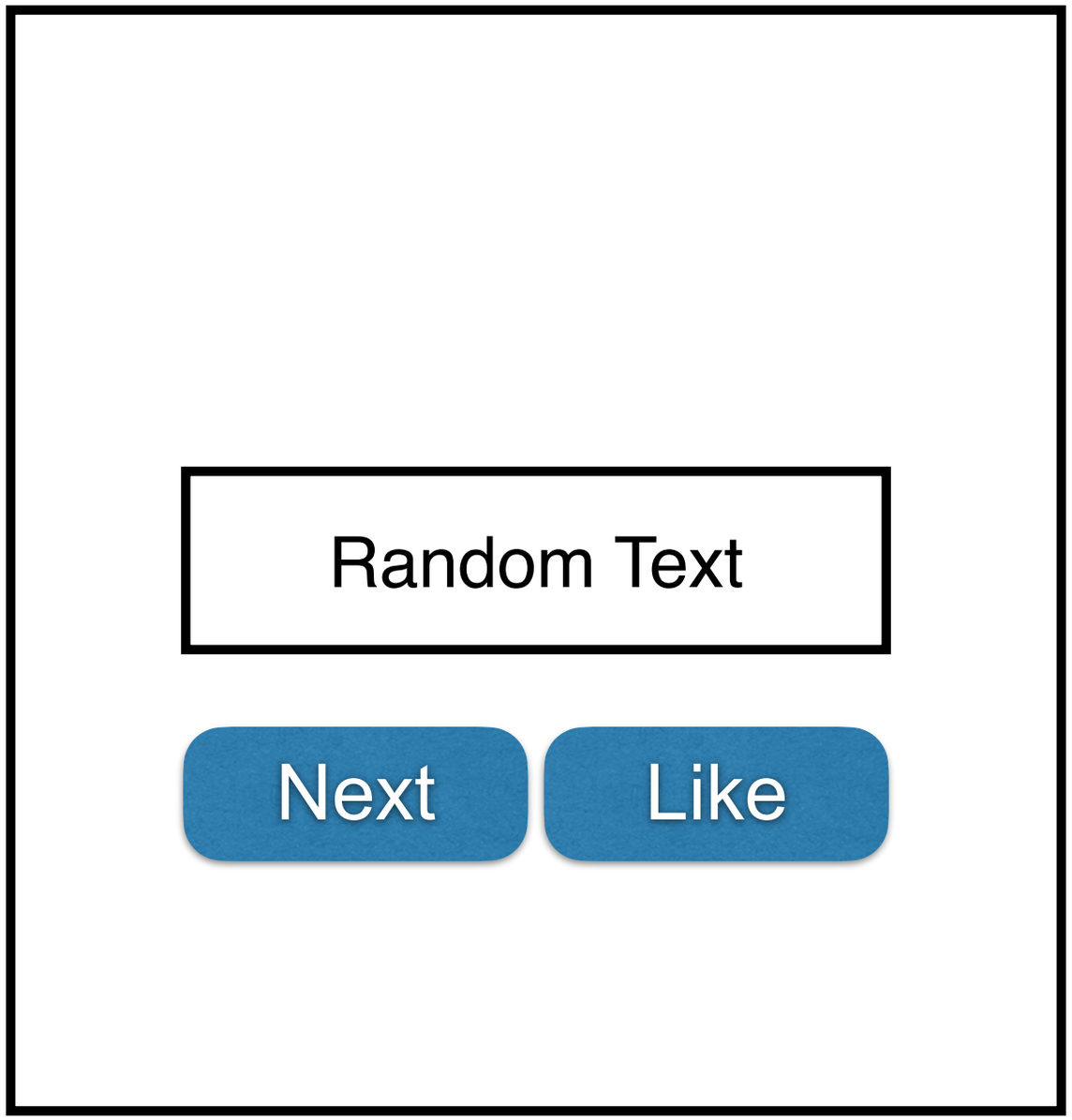


https://pub.dev/packages/flutter_screenutil

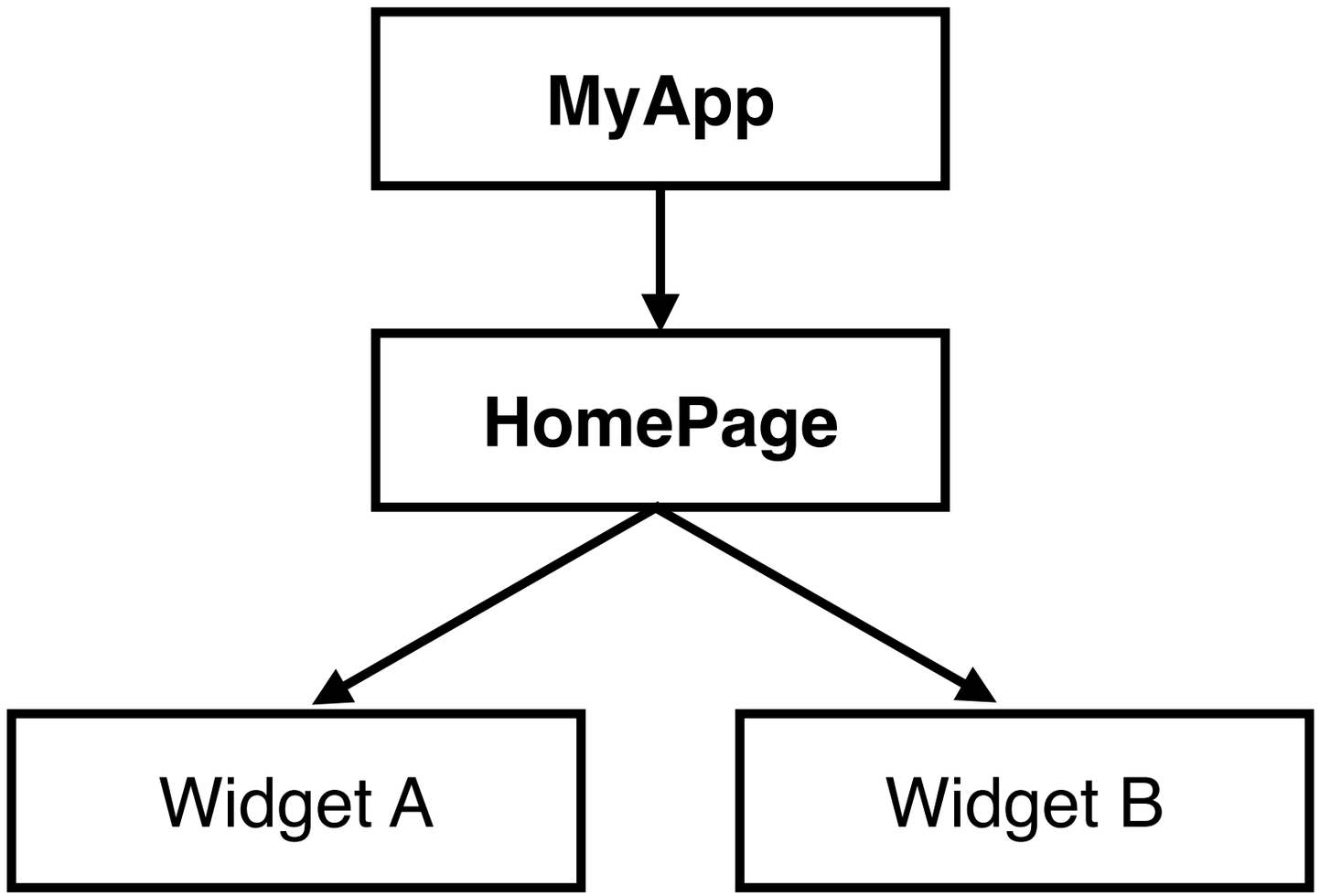
App Theme



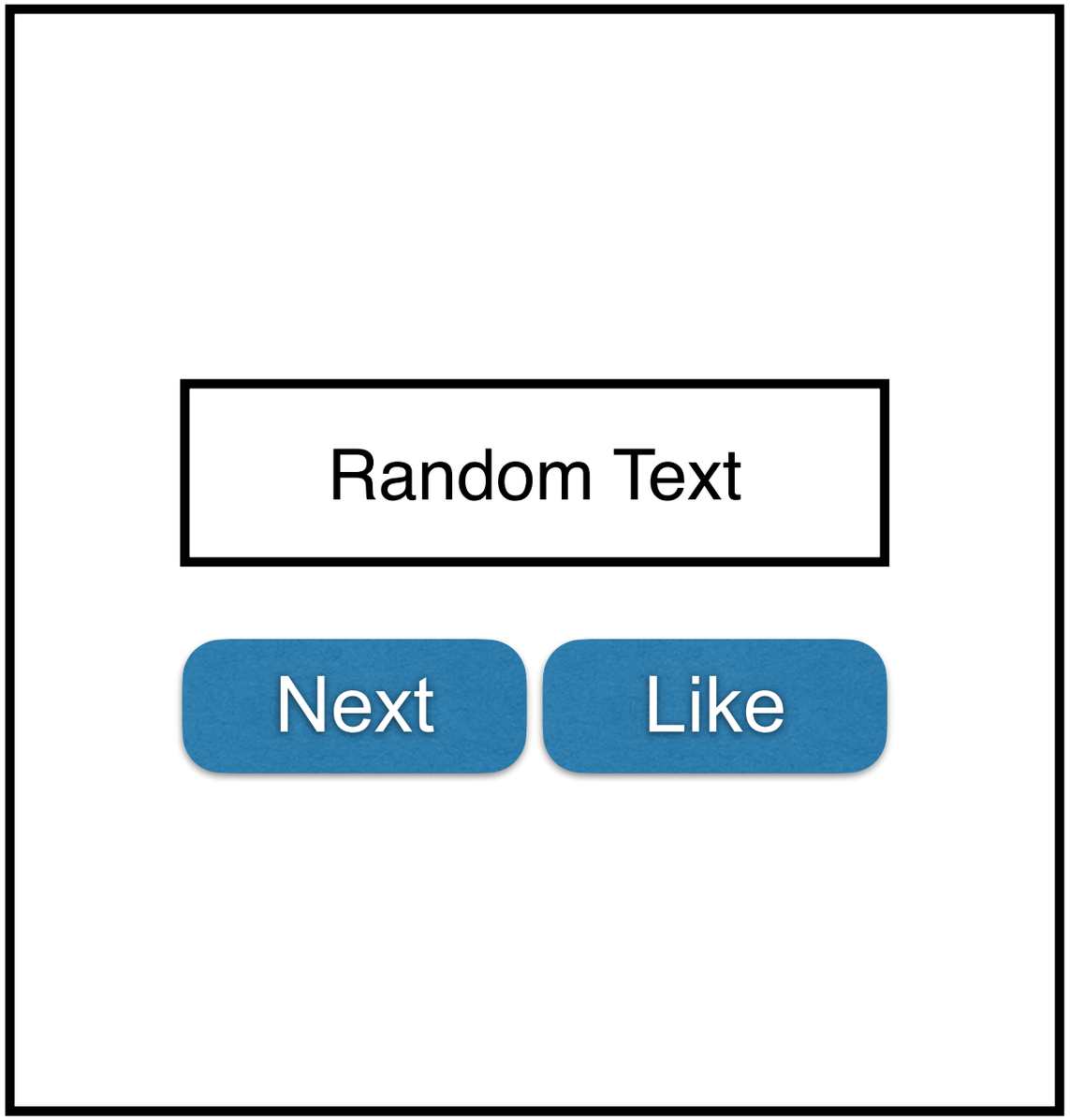
Workshop

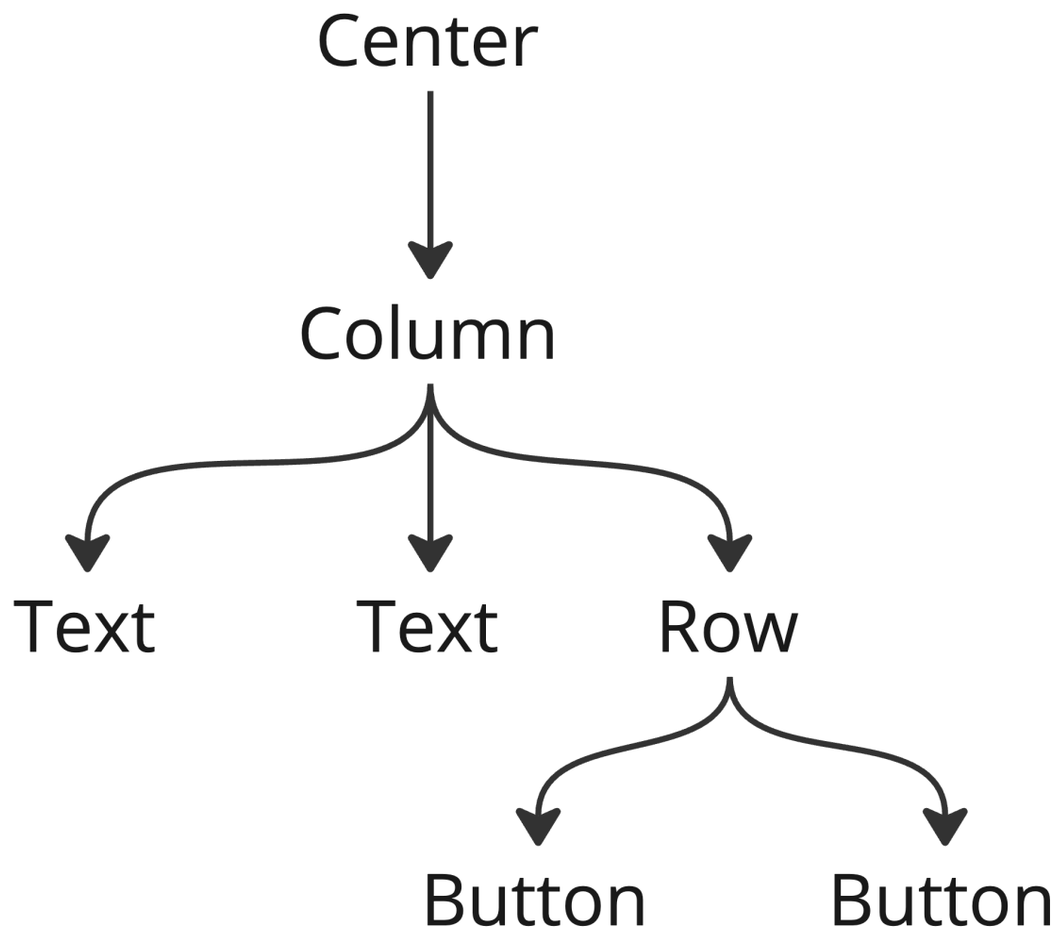


Design your app structure

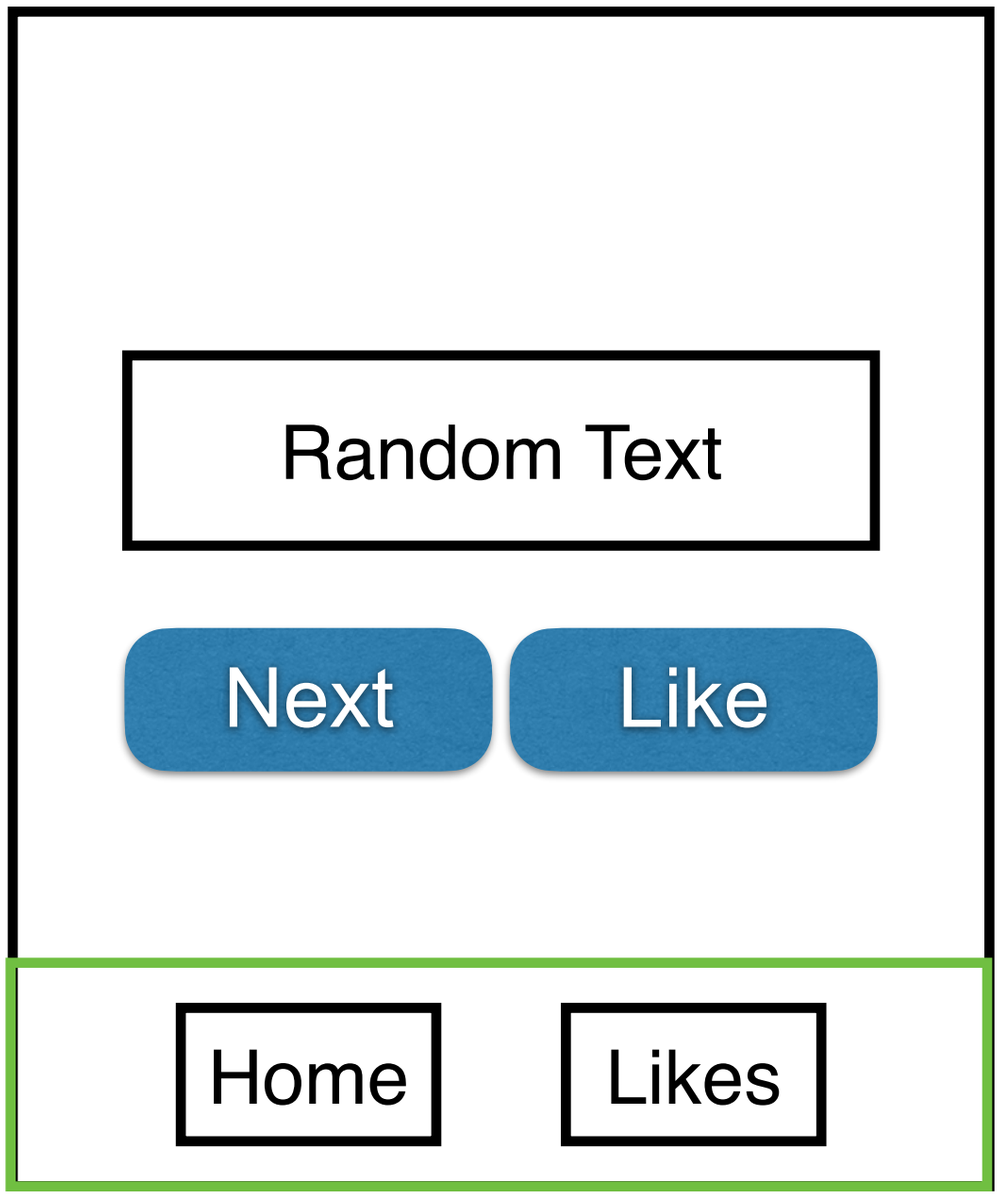


Widget tree



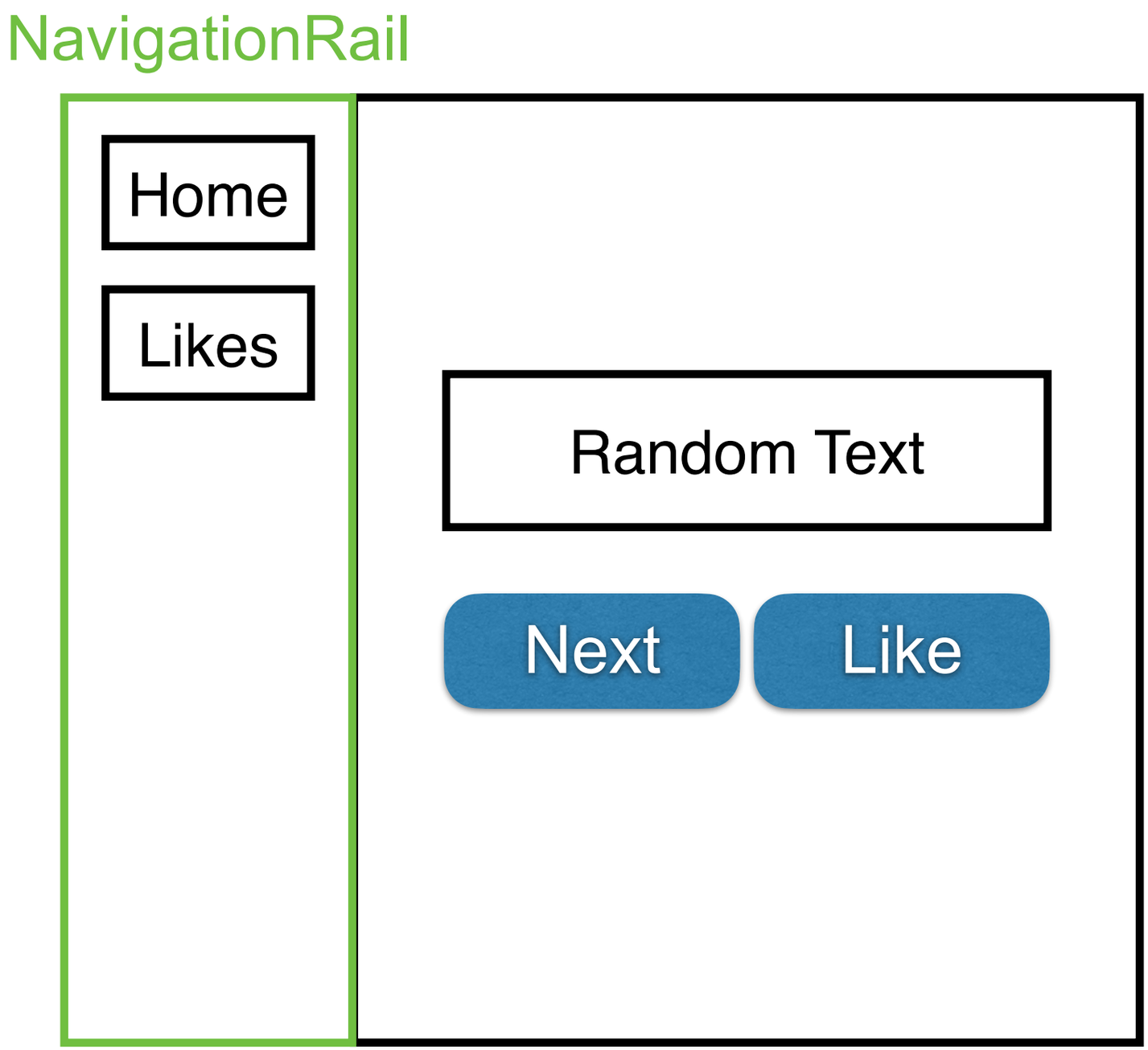


Add more pages

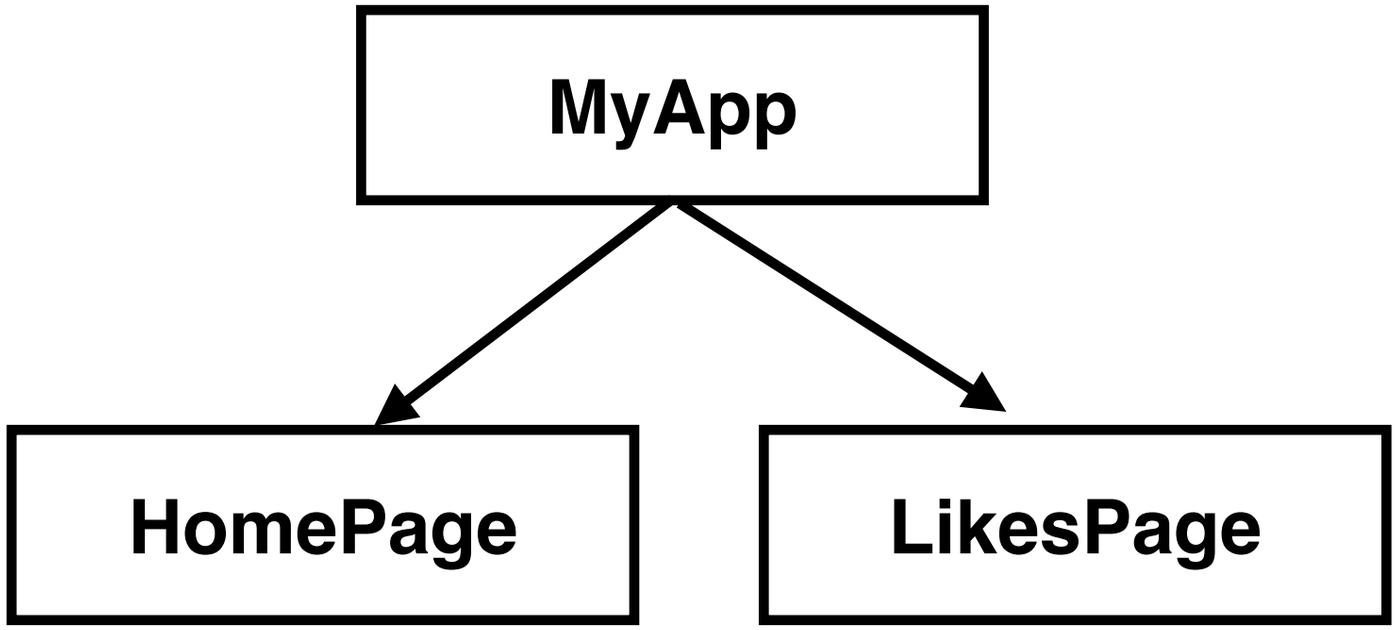


BottomNavigationBar

Responsive !!



Design your app structure



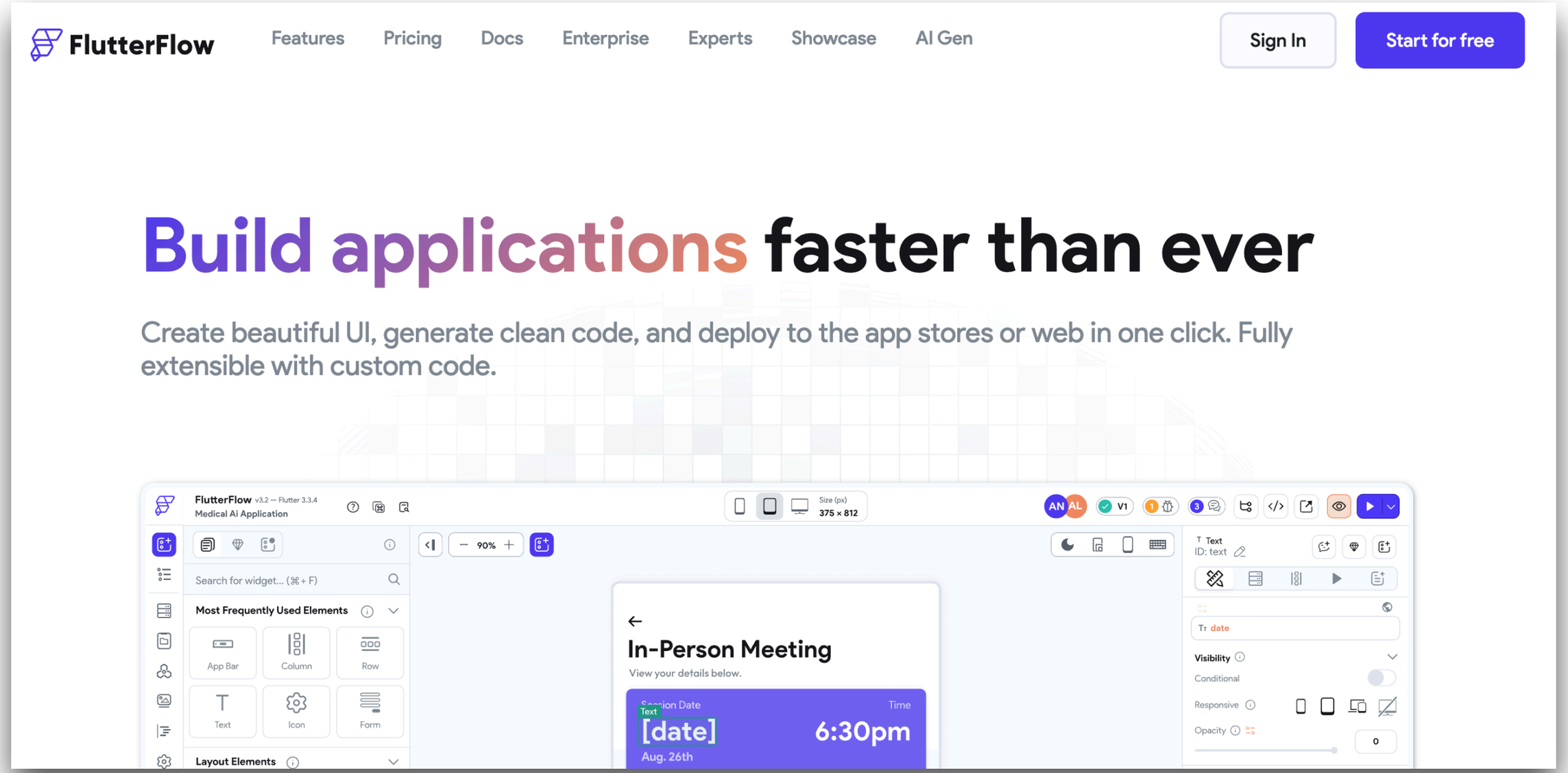
https://docs.flutter.dev/ui/navigation

Flutter Design UI Challenges

https://github.com/lohanidamodar/flutter_ui_challenges

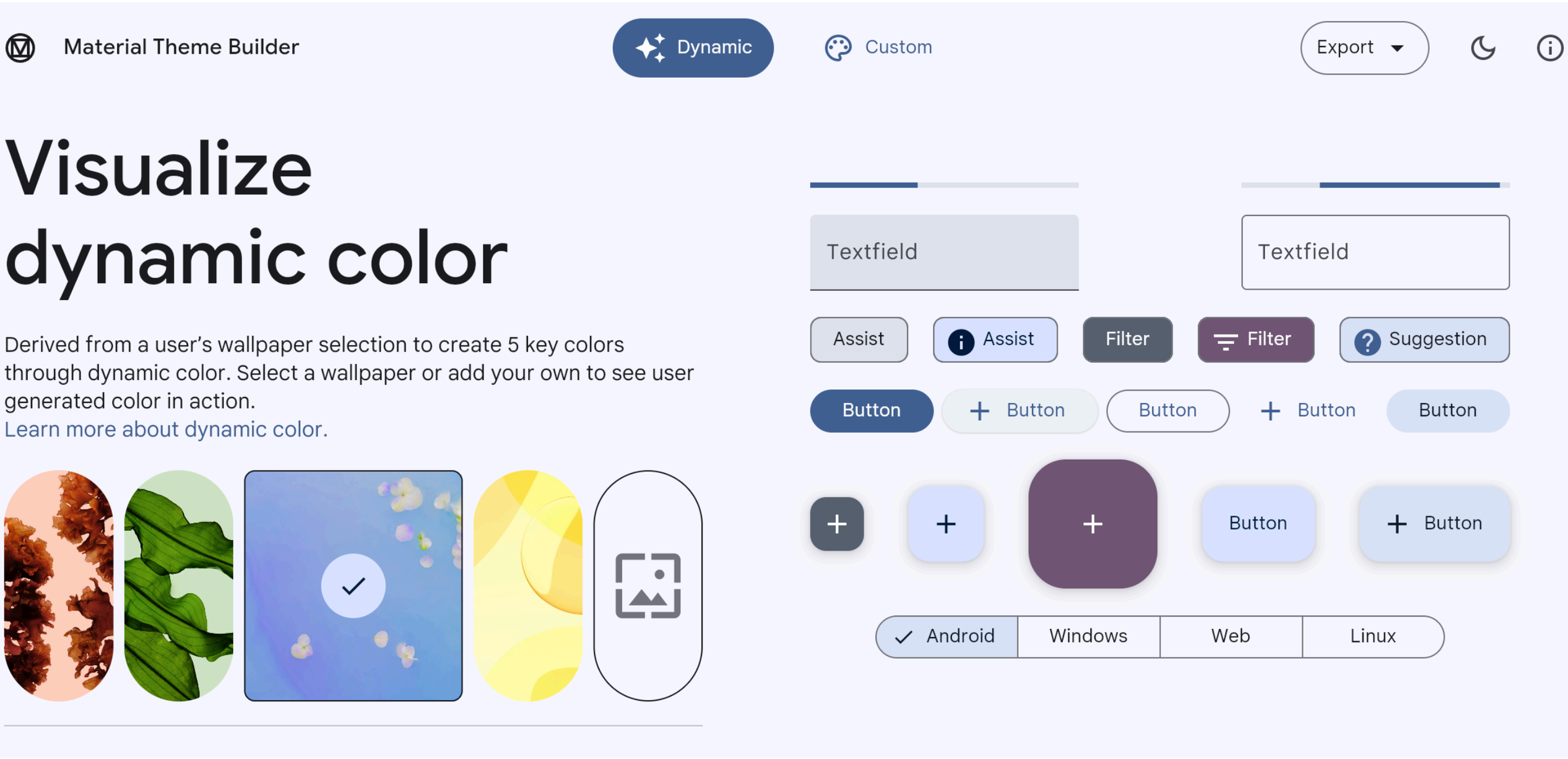
Design UIs Tools

FlutterFlow



https://flutterflow.io/

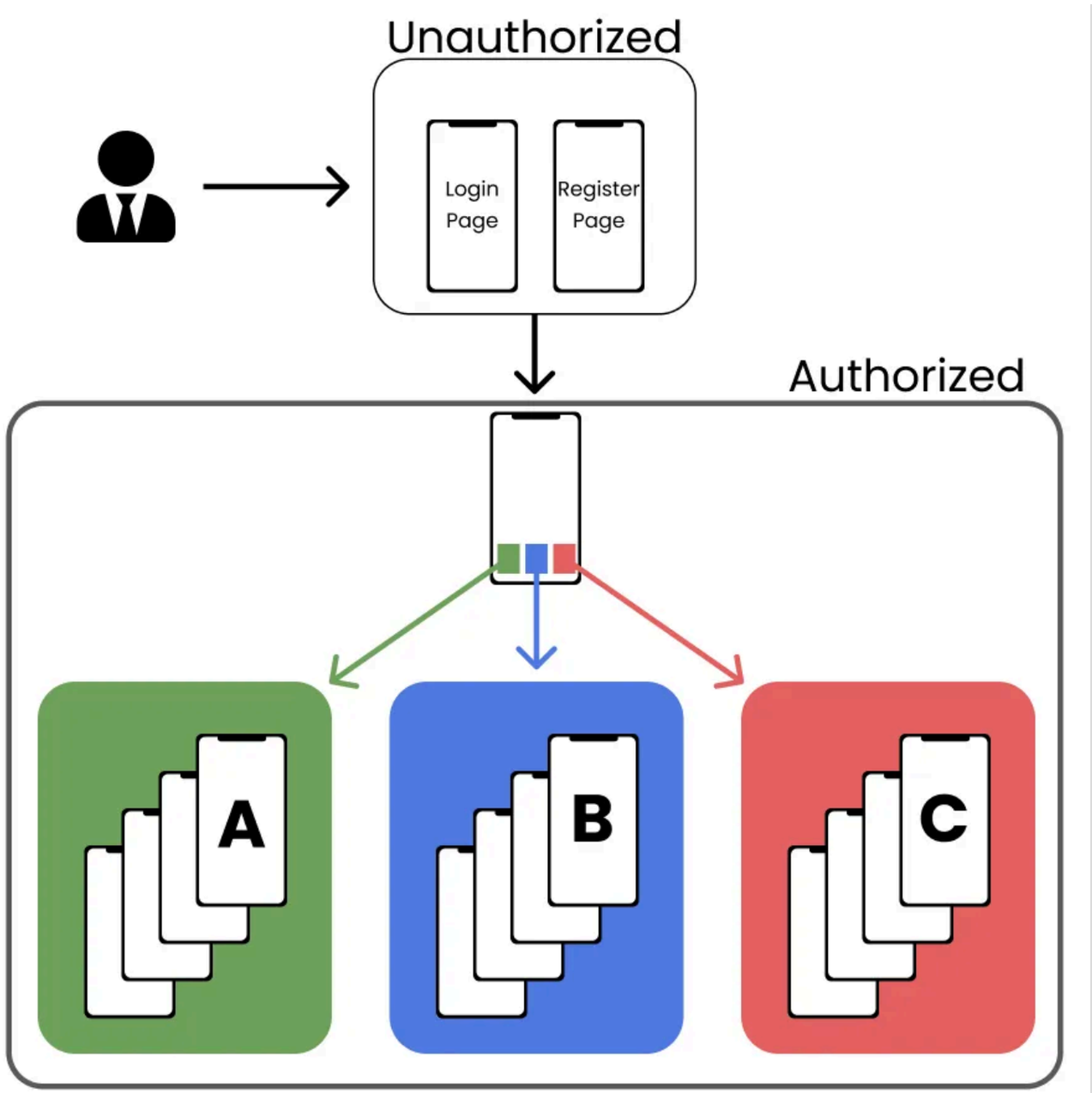
Material Theme Builder



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

Navigation/Router in app

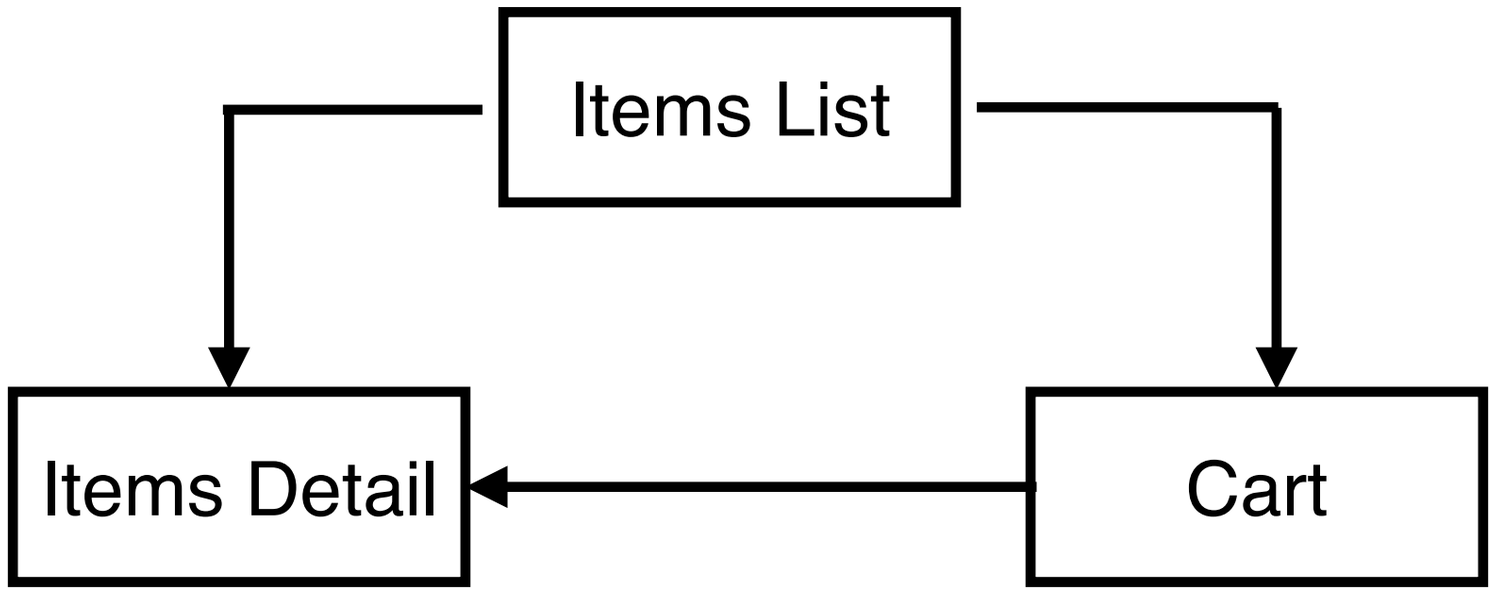
Why we need navigation ?



How to manage navigation ?

Navigation 2.0
Go Router
Fluro
Get

Workshop



Simple State management

State management

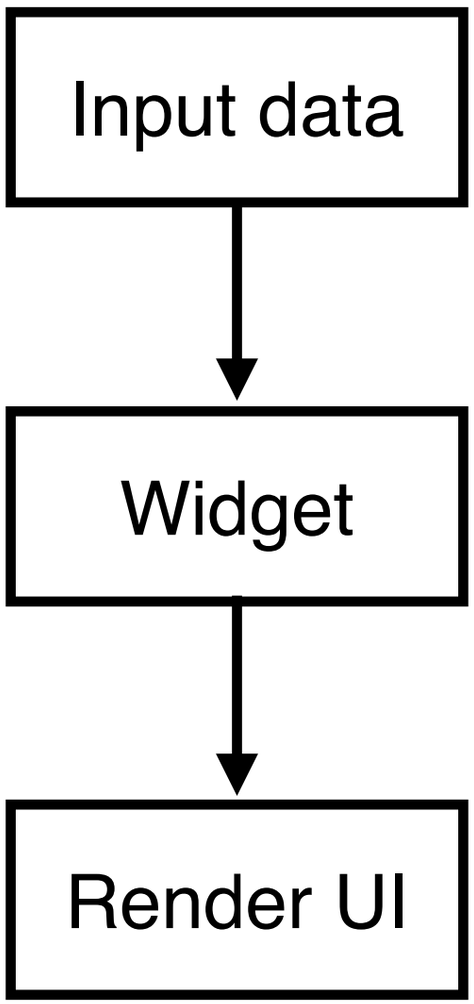
UI = f(state)

UI = layout of screen

State = application state

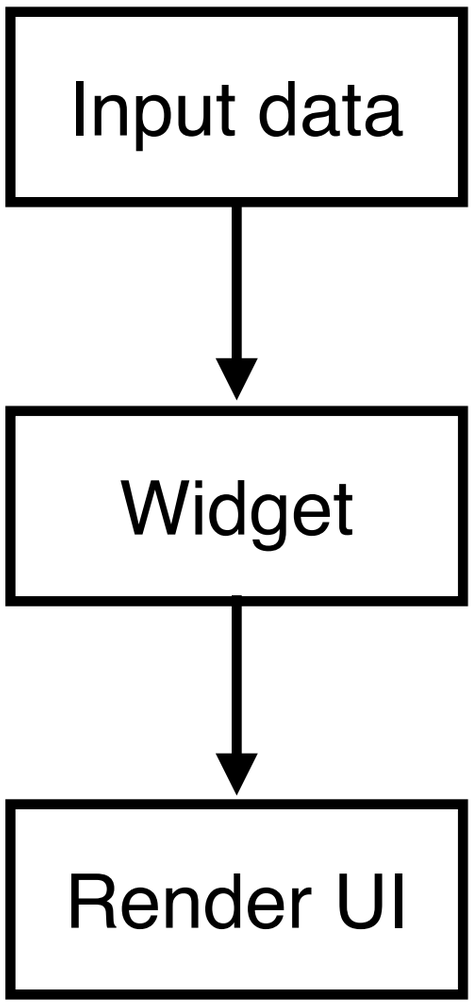
Any interactions that reflects UI changes

State management



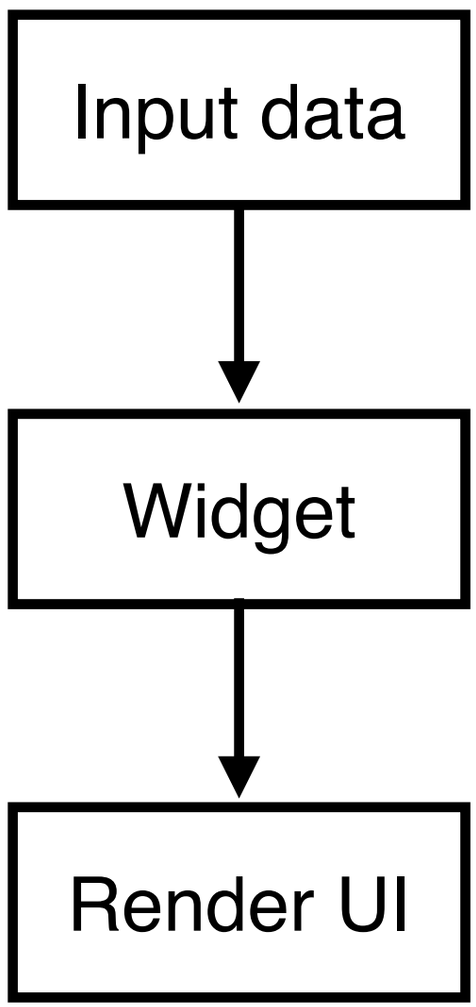
State management

Stateless widget



Input changes

Stateful widget



Input and local state changes

State ?

Form validation

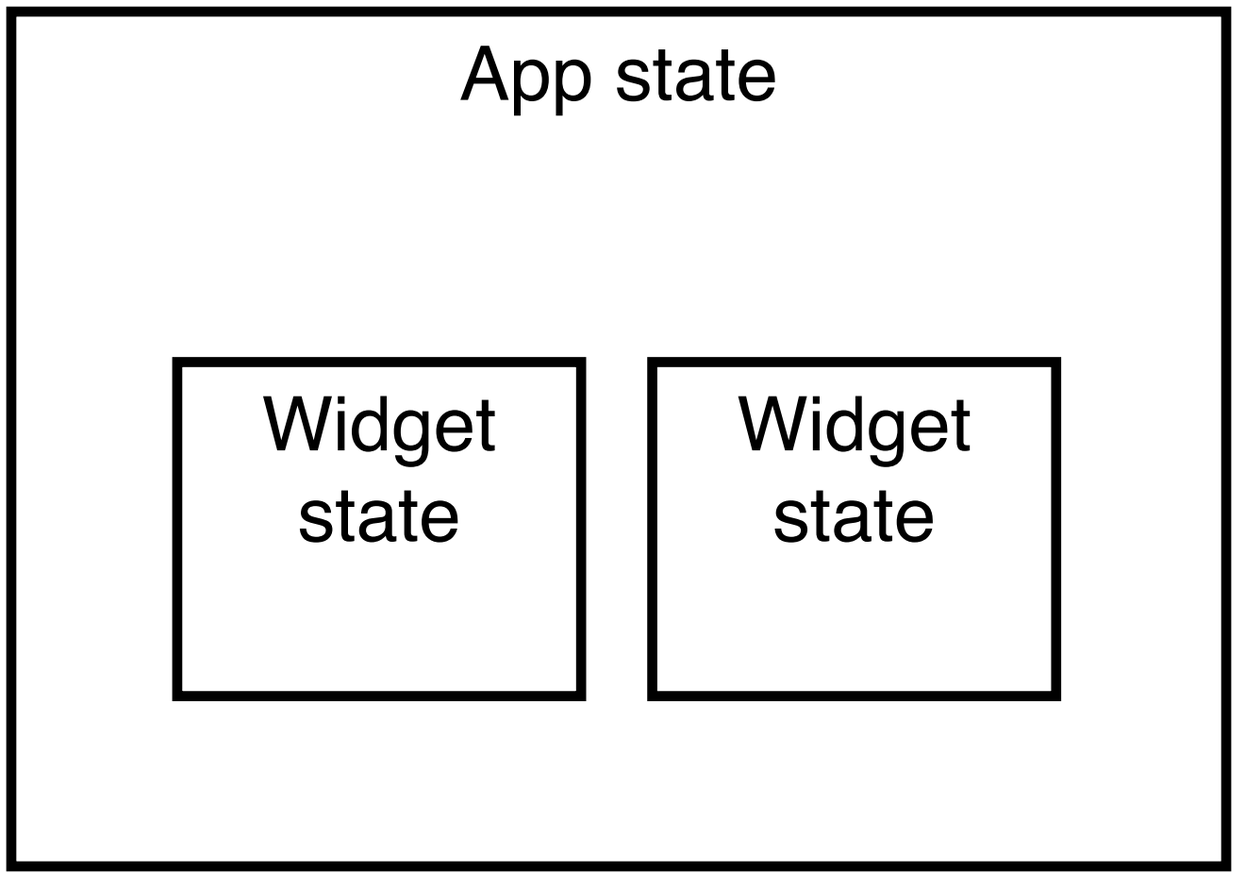
Network request

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

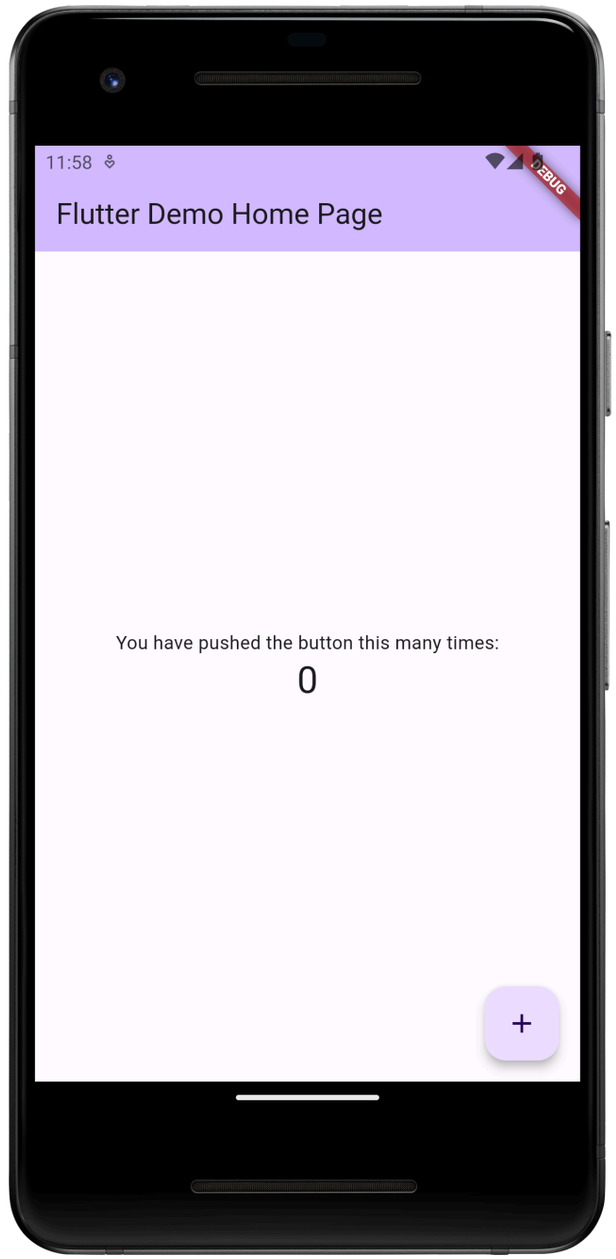
Current data

Current progress

Current tab

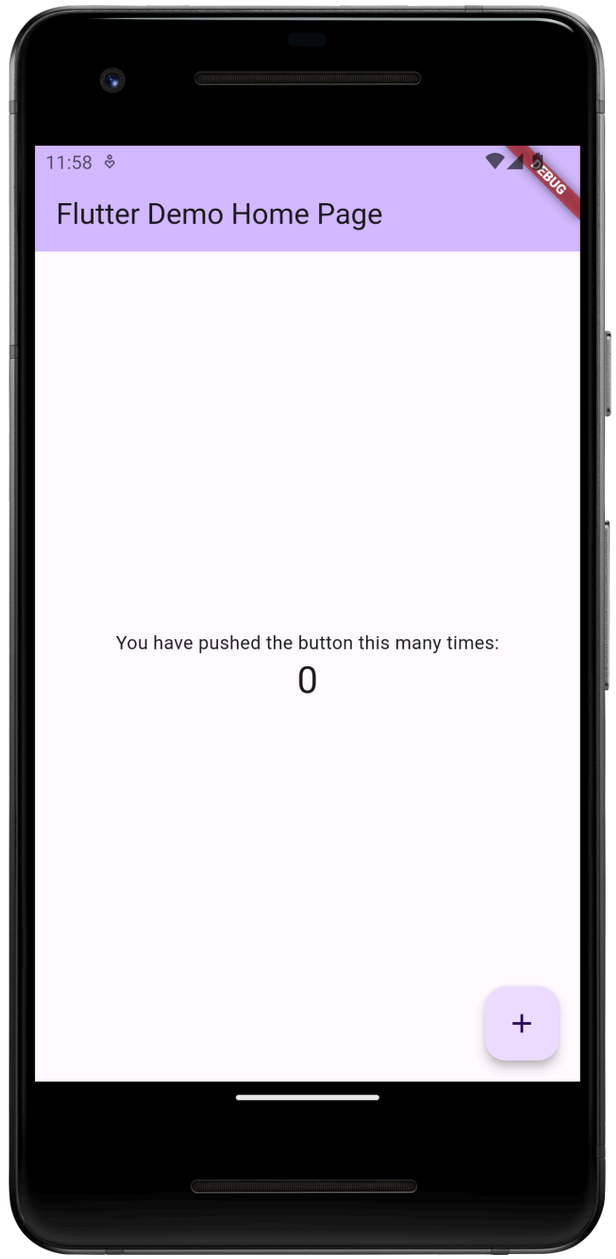
Stateful widget (1)

```dart
class MyHomePage extends StatefulWidget {

  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();

}
```

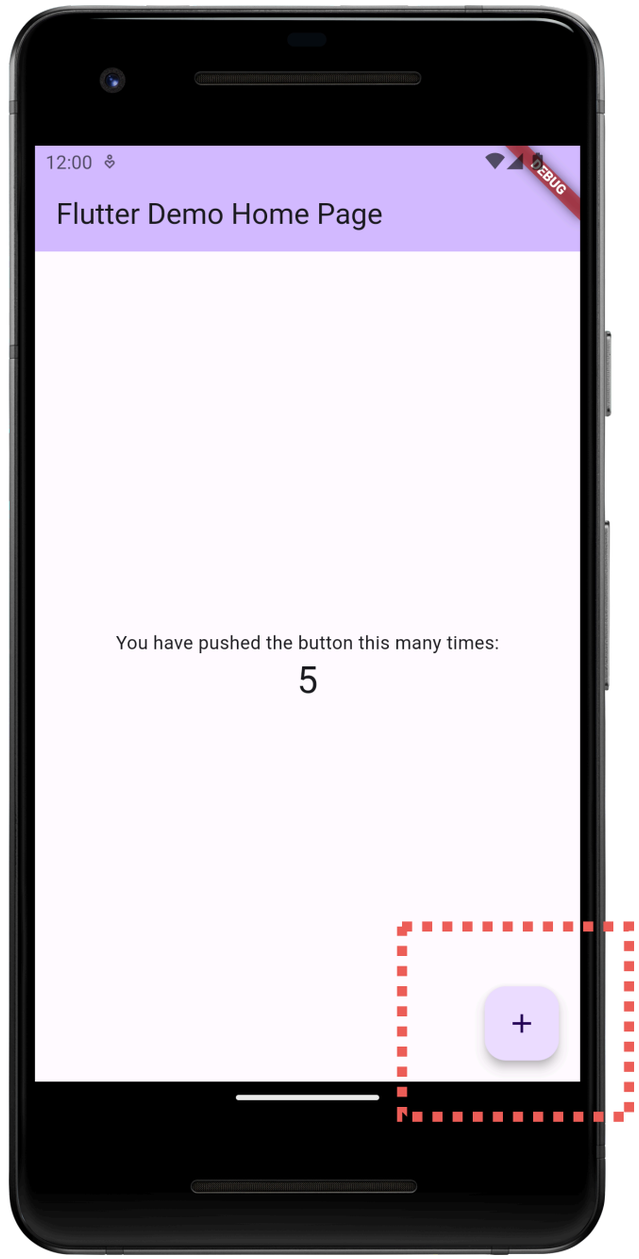


Stateful widget (2)

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

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

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



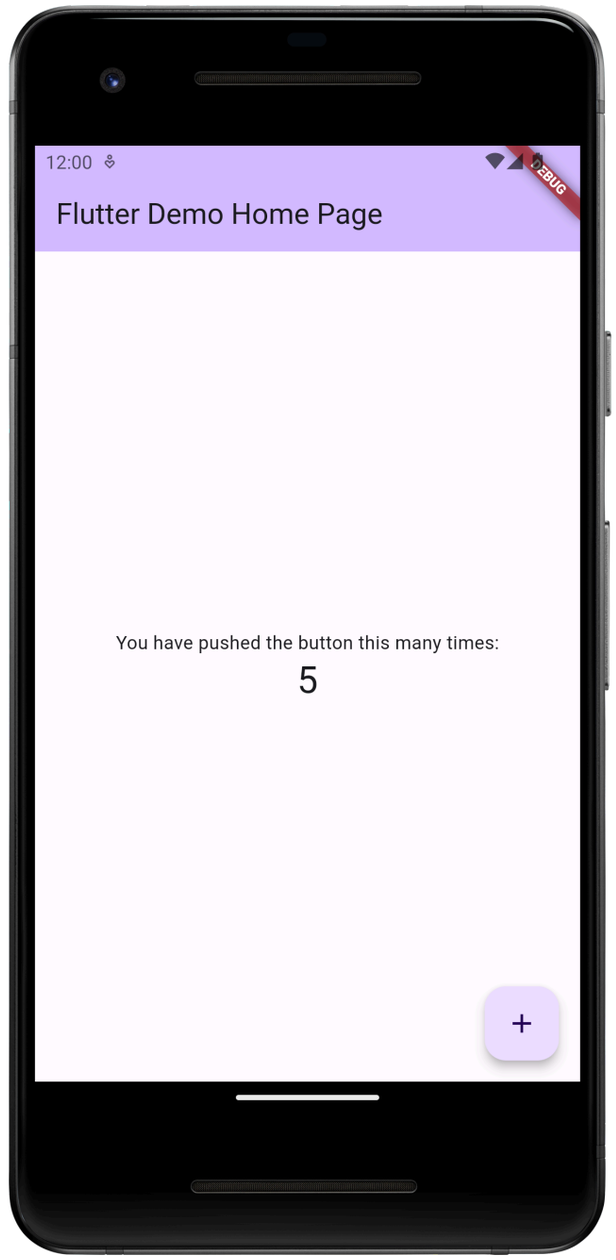
Stateful widget (3)




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

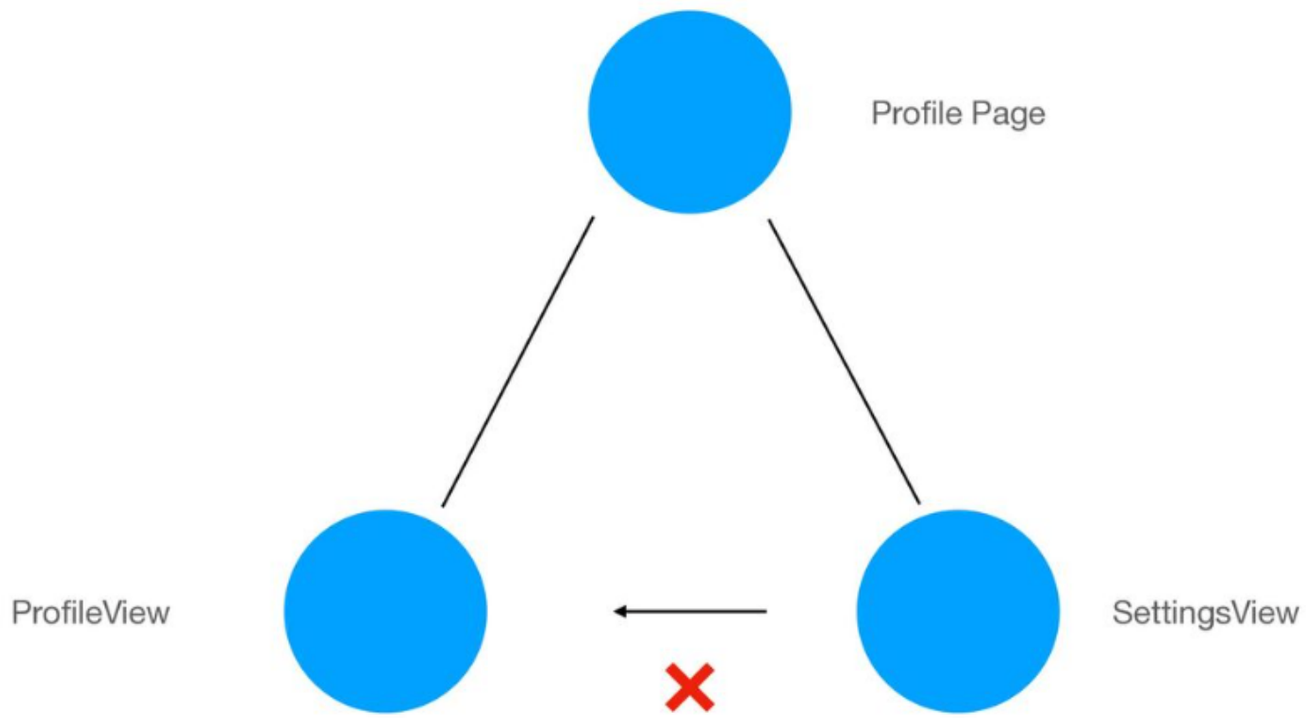
Stateful widget (4)

```dart
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

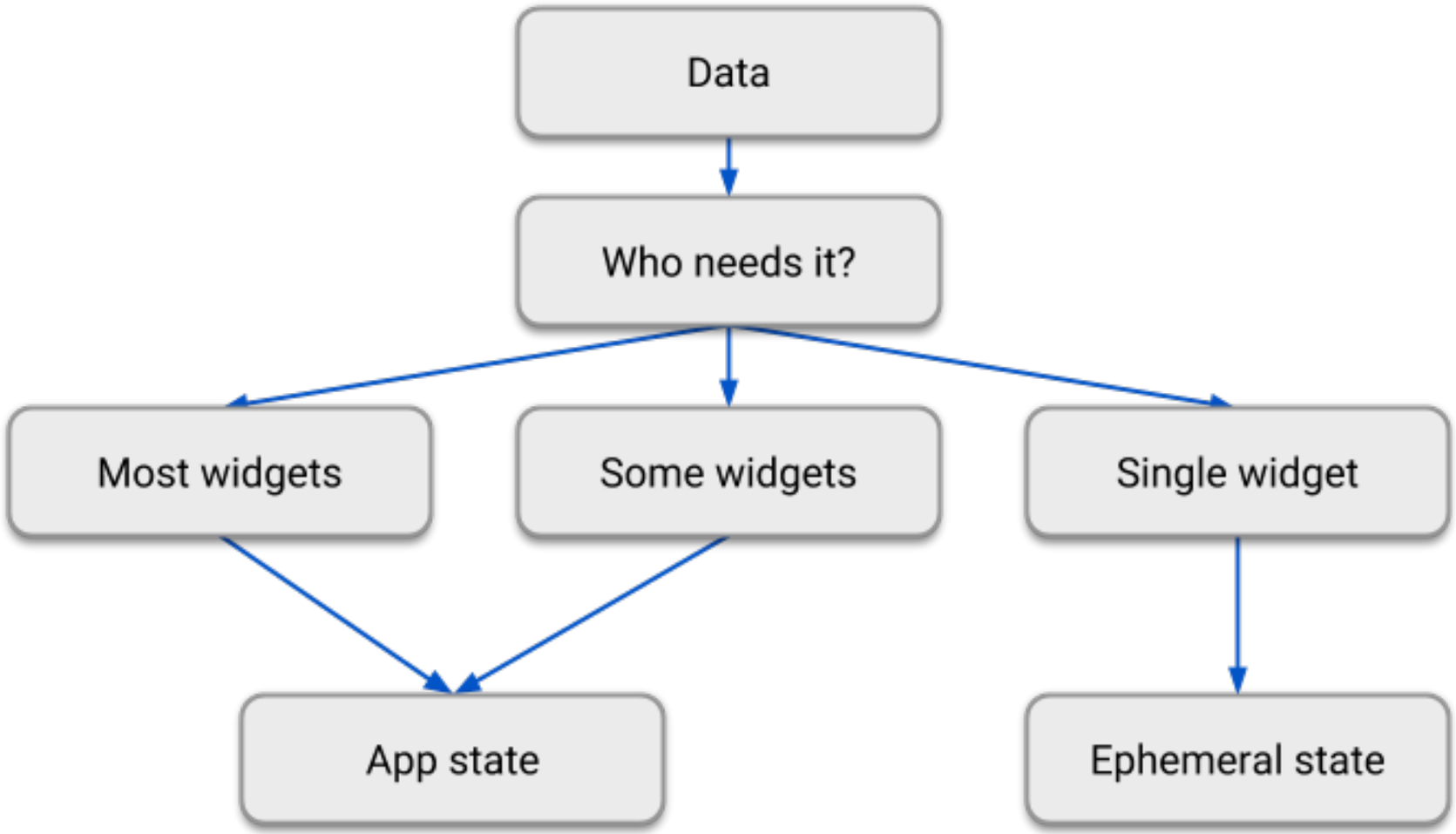
Shopping cart

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 ?

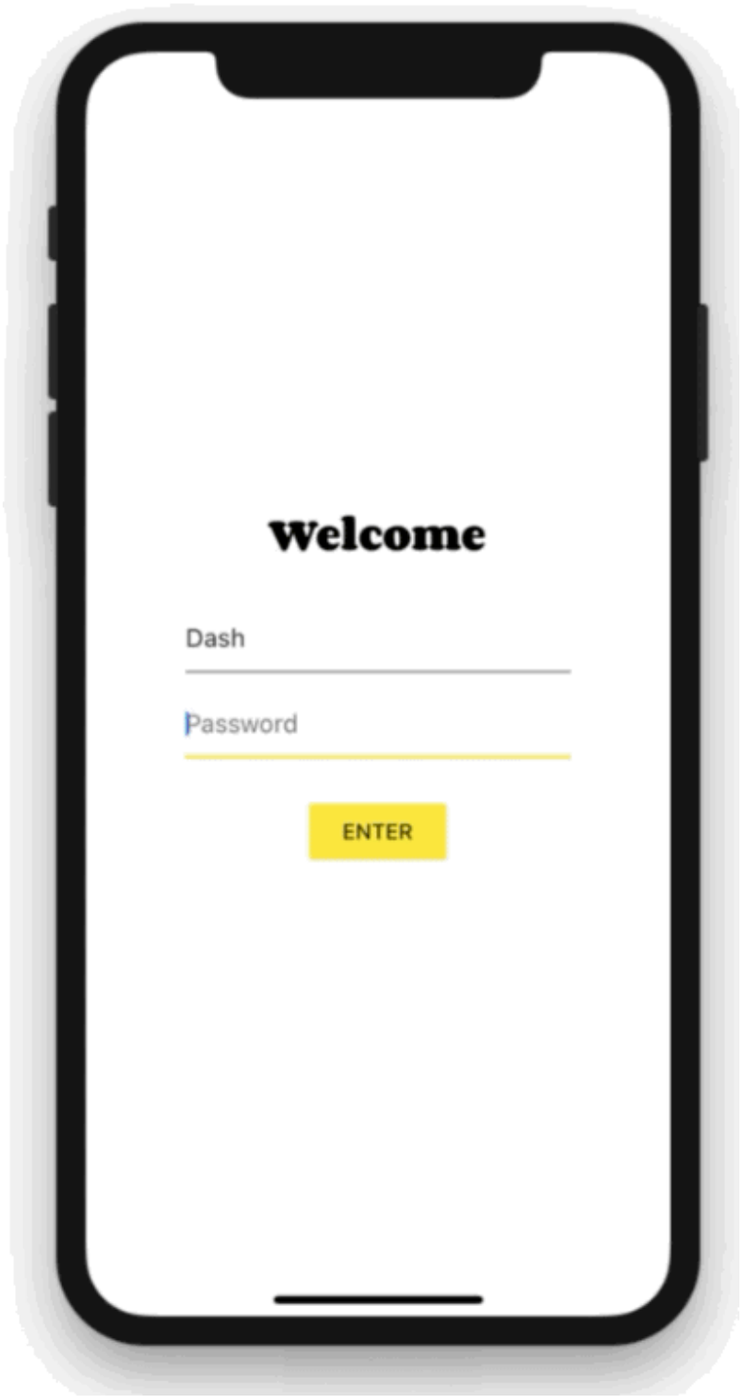


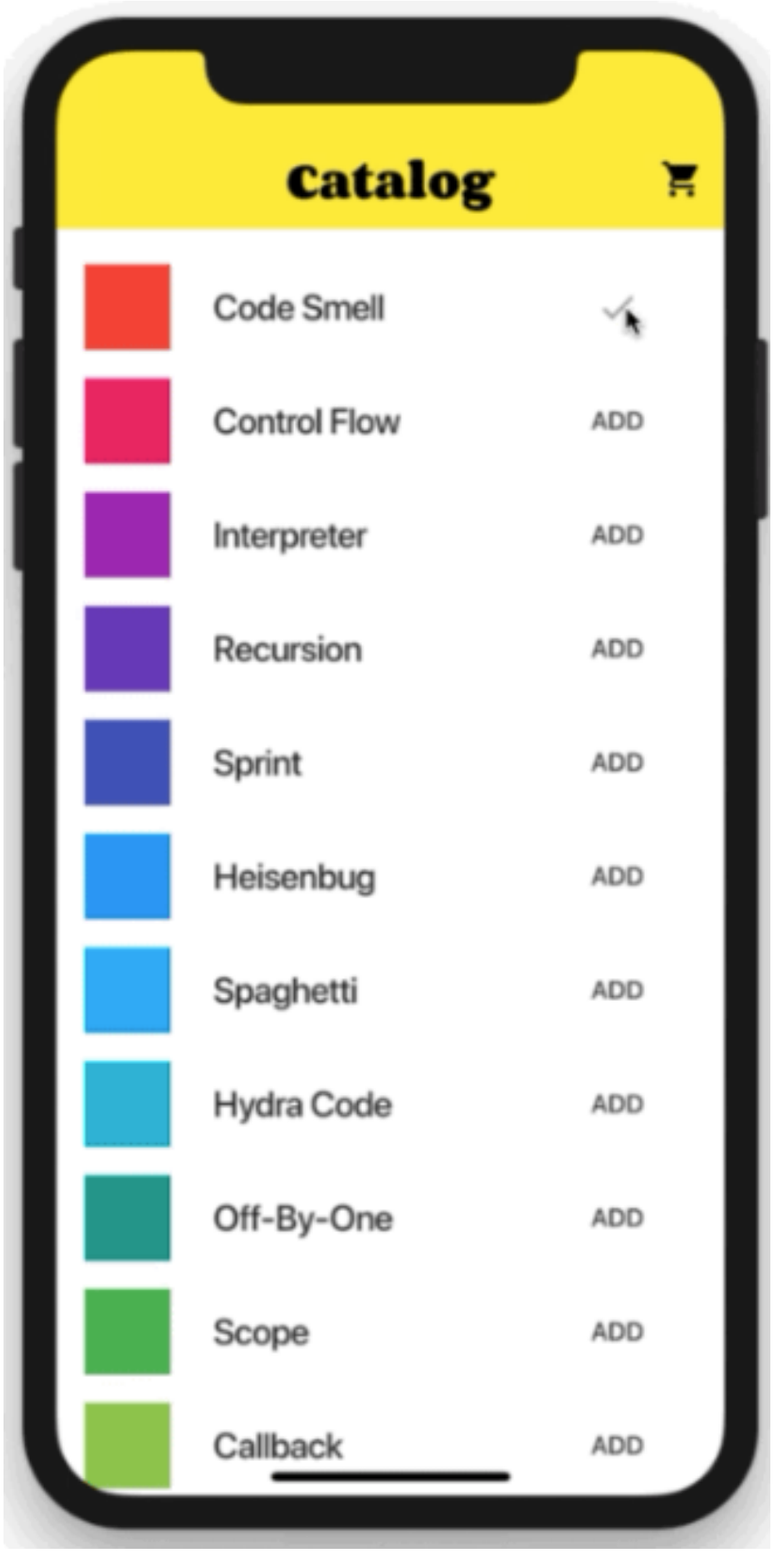
State management libraries !!

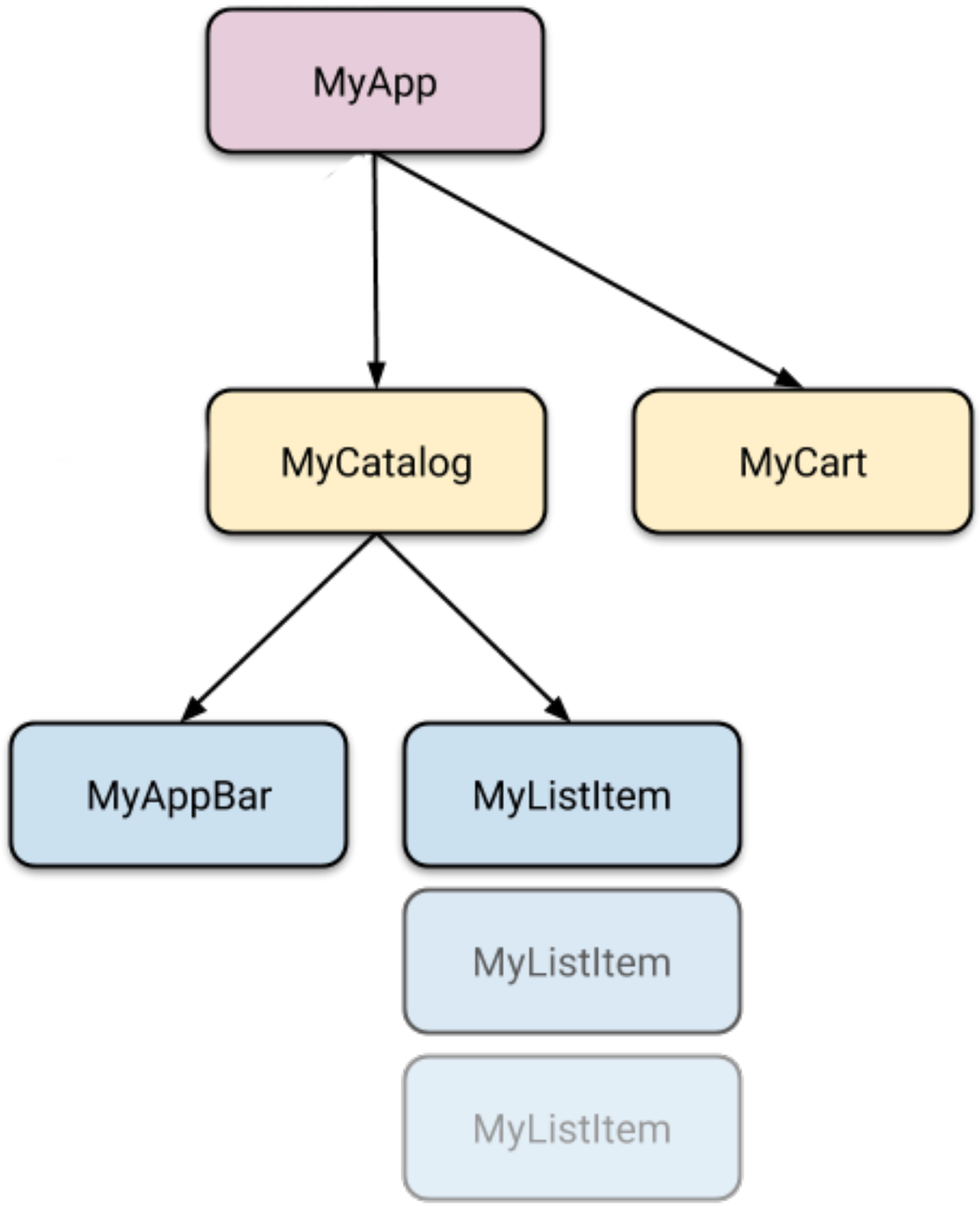
Provider	**setState**	Riverpod	MobX
BLoC	**GetX**	Redux	Reactive Value

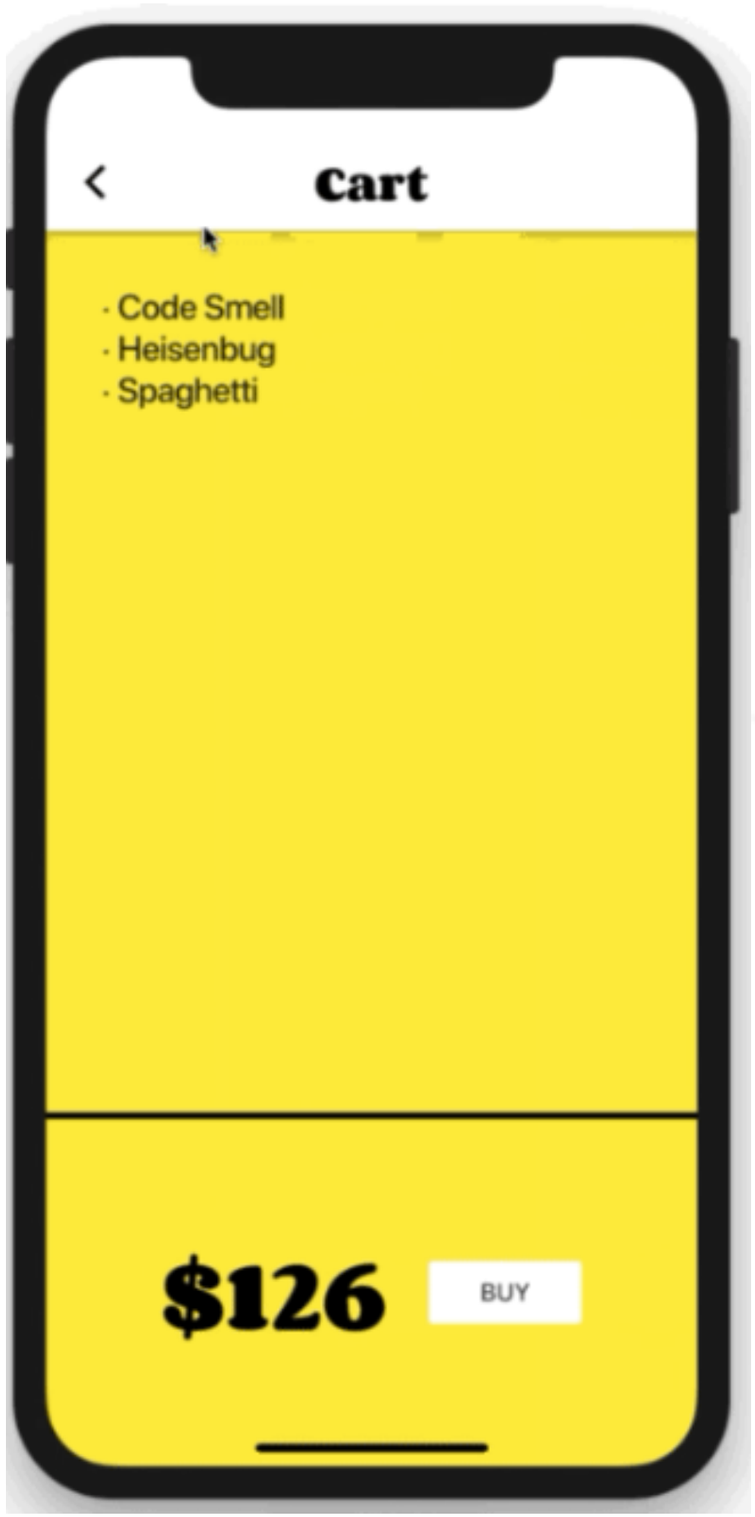
Workshop with State management

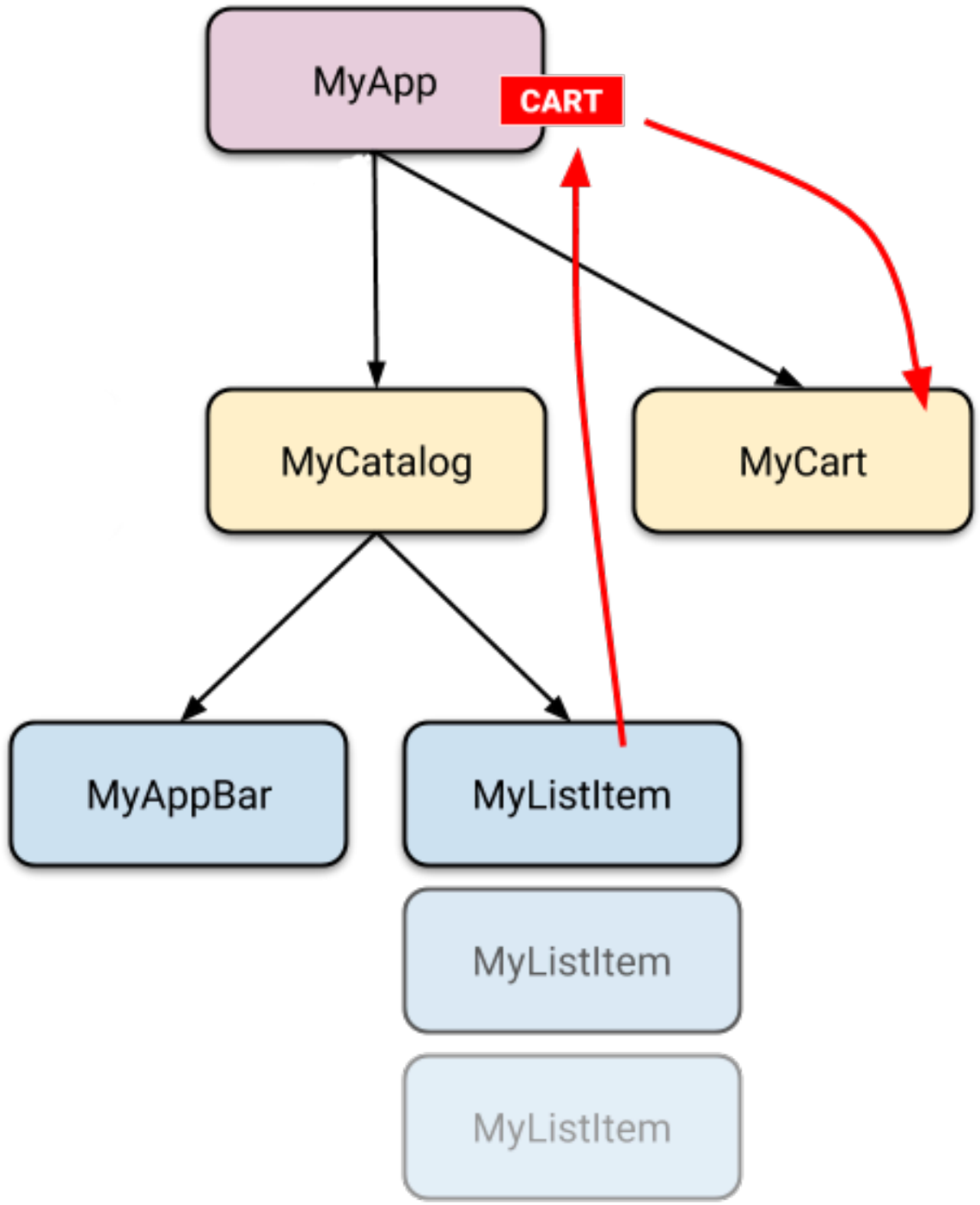
https://docs.flutter.dev/data-and-backend/state-mgmt/simple











Try to convert to GetX !!

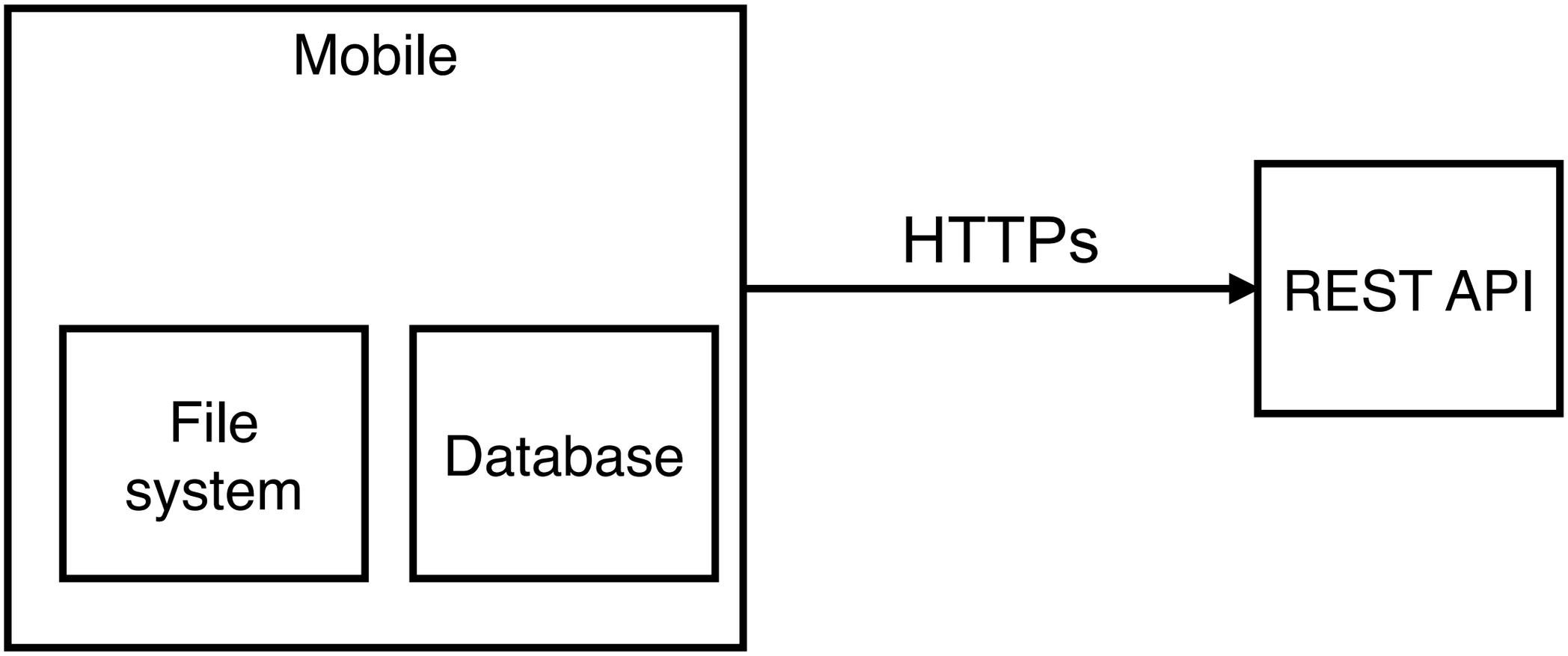
https://pub.dev/packages/get

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

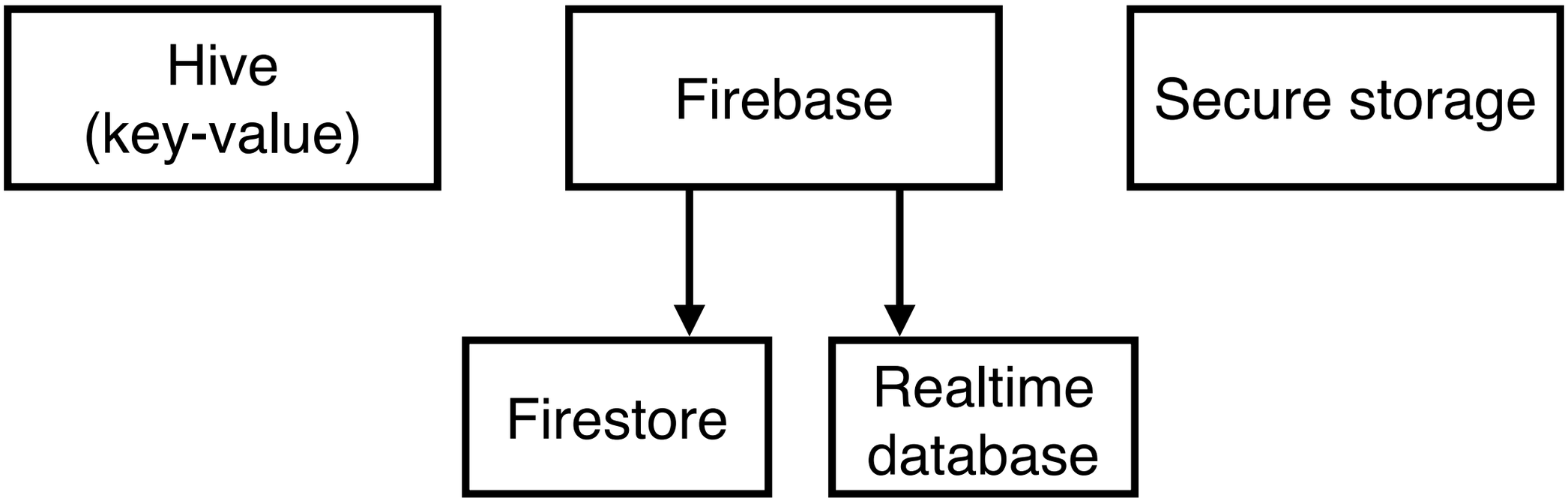
Working with HTTPs

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

```dart
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	Stream error
Single value	Multiple value

Fetch data from APIs

```dart
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
}
```

```dart
// 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

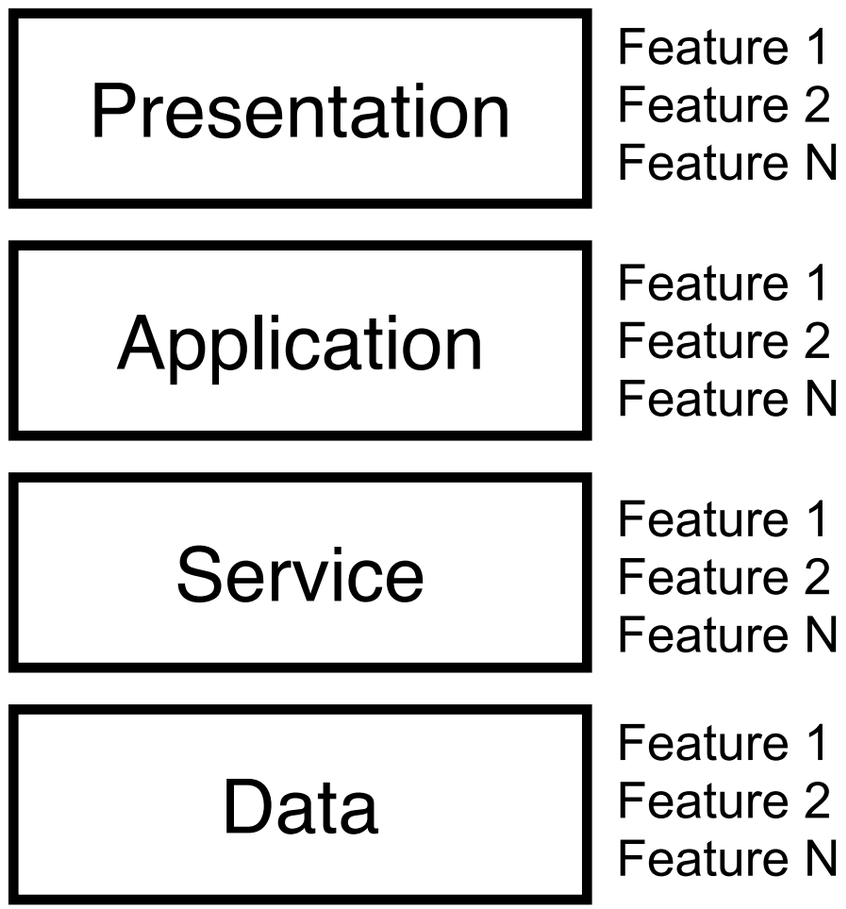
Ensure team can follow a clear convention
Add feature in a consistent way

Layer-first

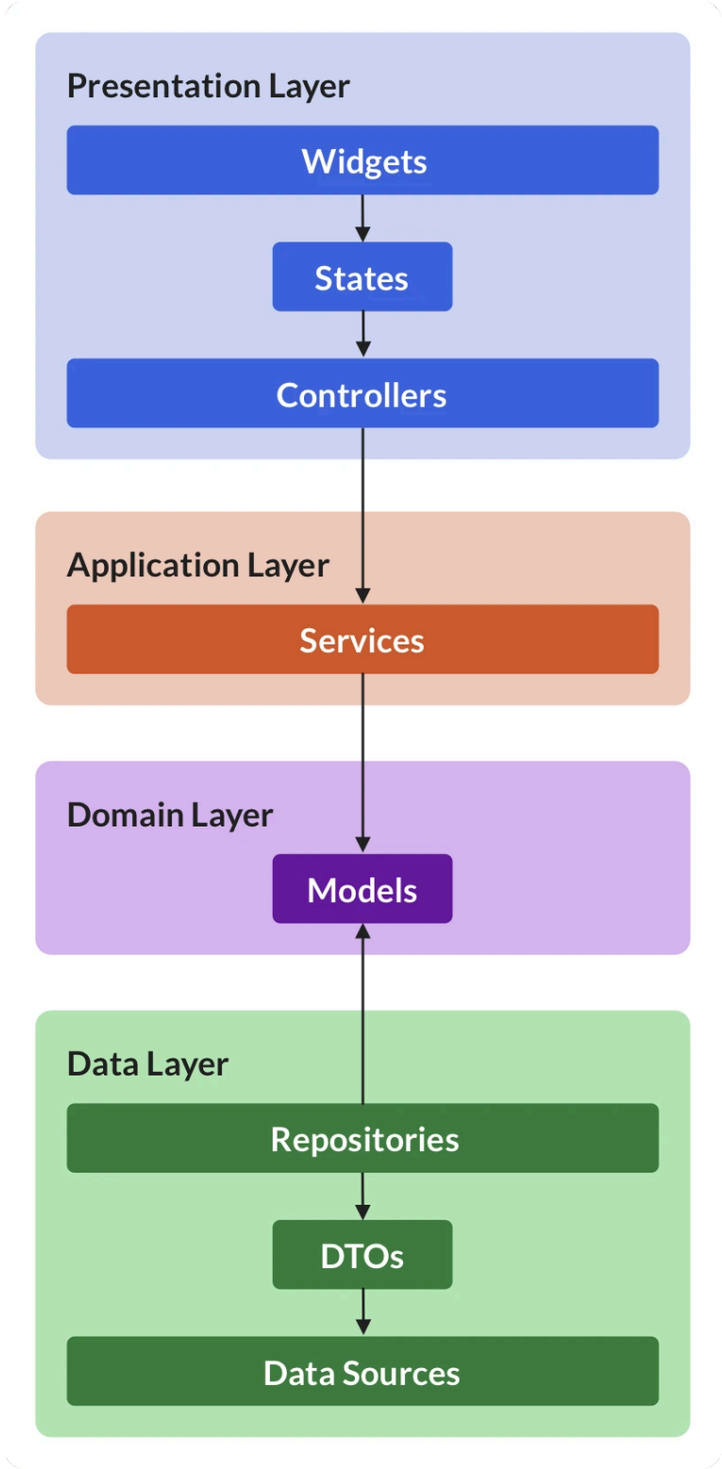
Feature-first

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)

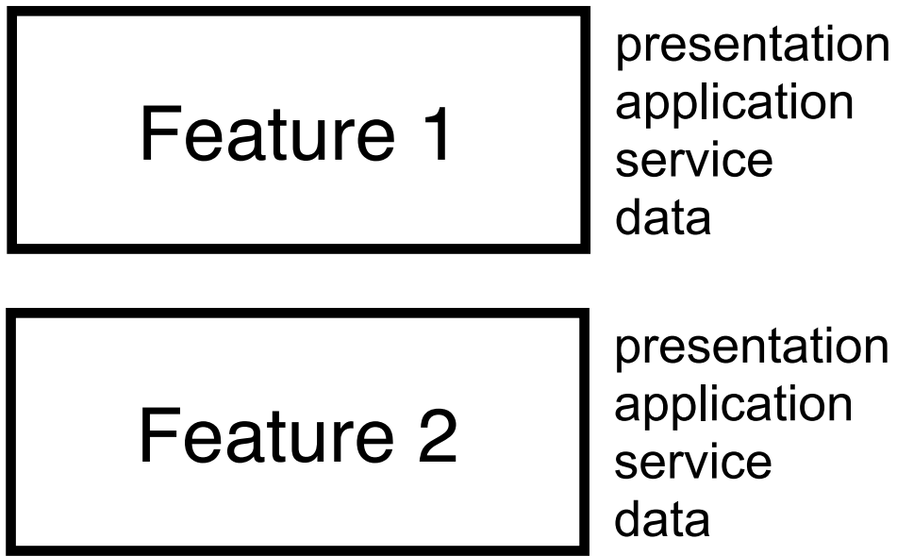
https://riverpod.dev/

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



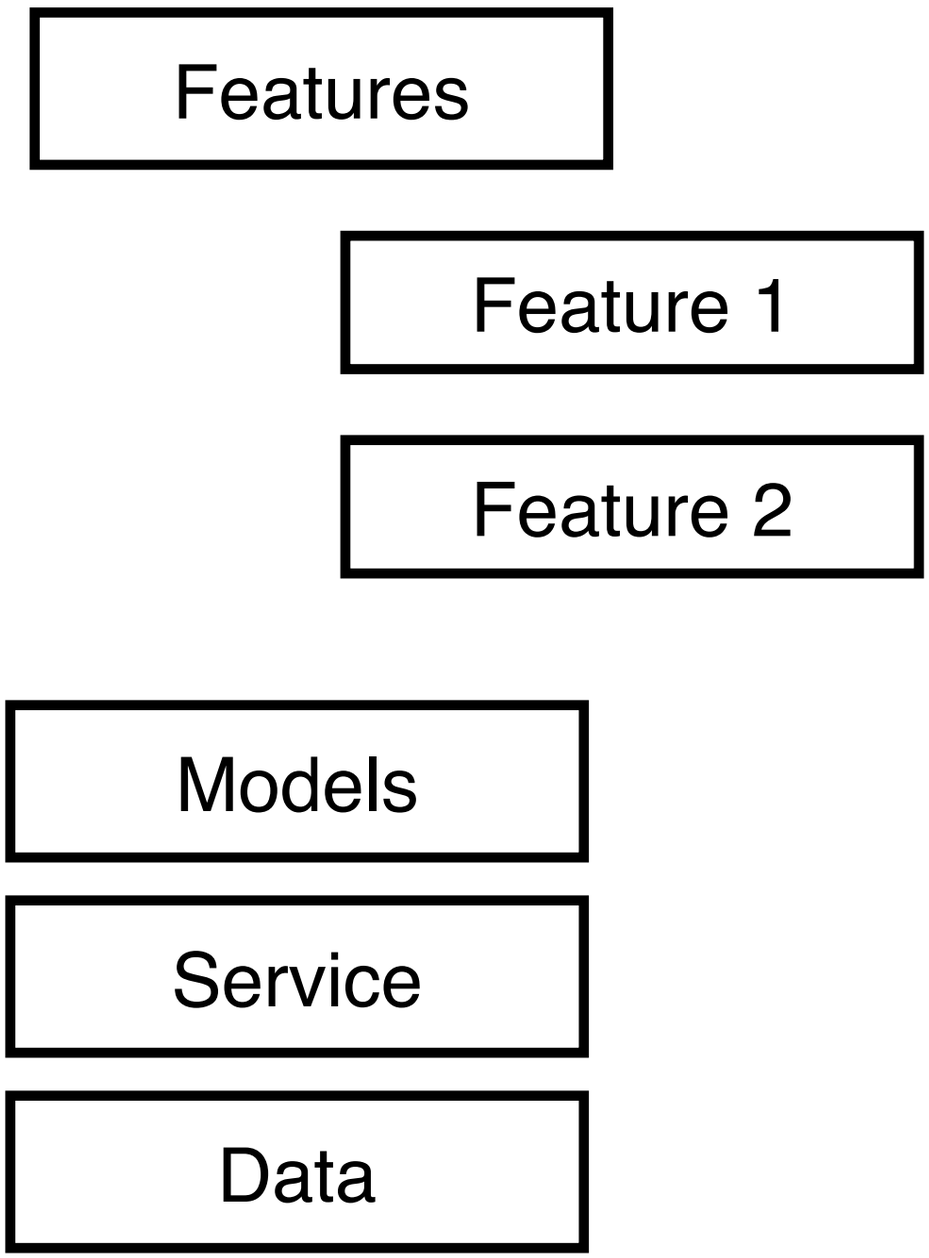
Feature-first

Not about the UI !!

What is a feature ?

Account	Product list	Product detail
Sign in	Order list	Checkout

Feature-first ?



What is a feature ?

Not use see, but what use does

Authenticate

Manage shopping cart

Checkout

Manage product

Manage order

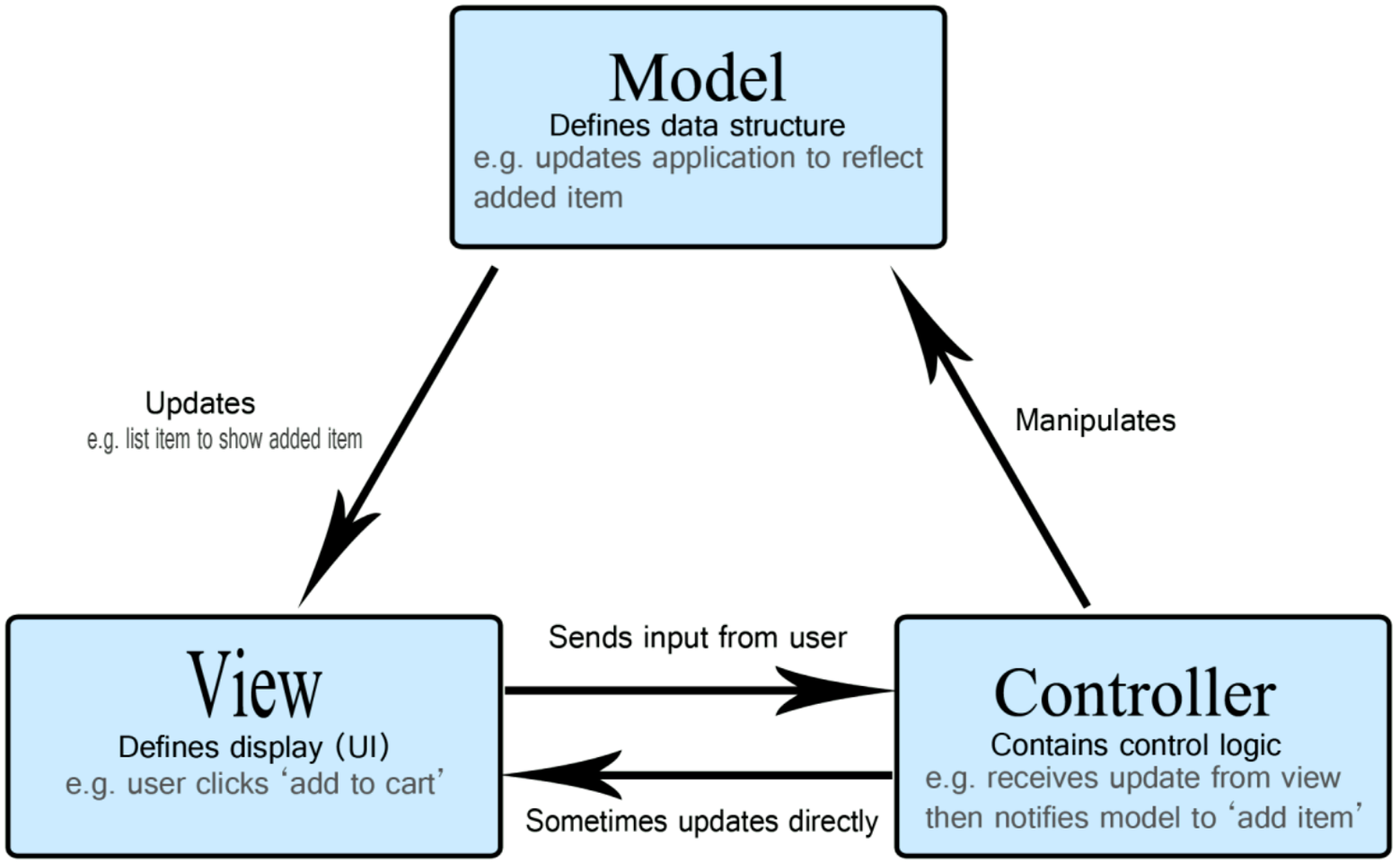
Manage user/ address

Review

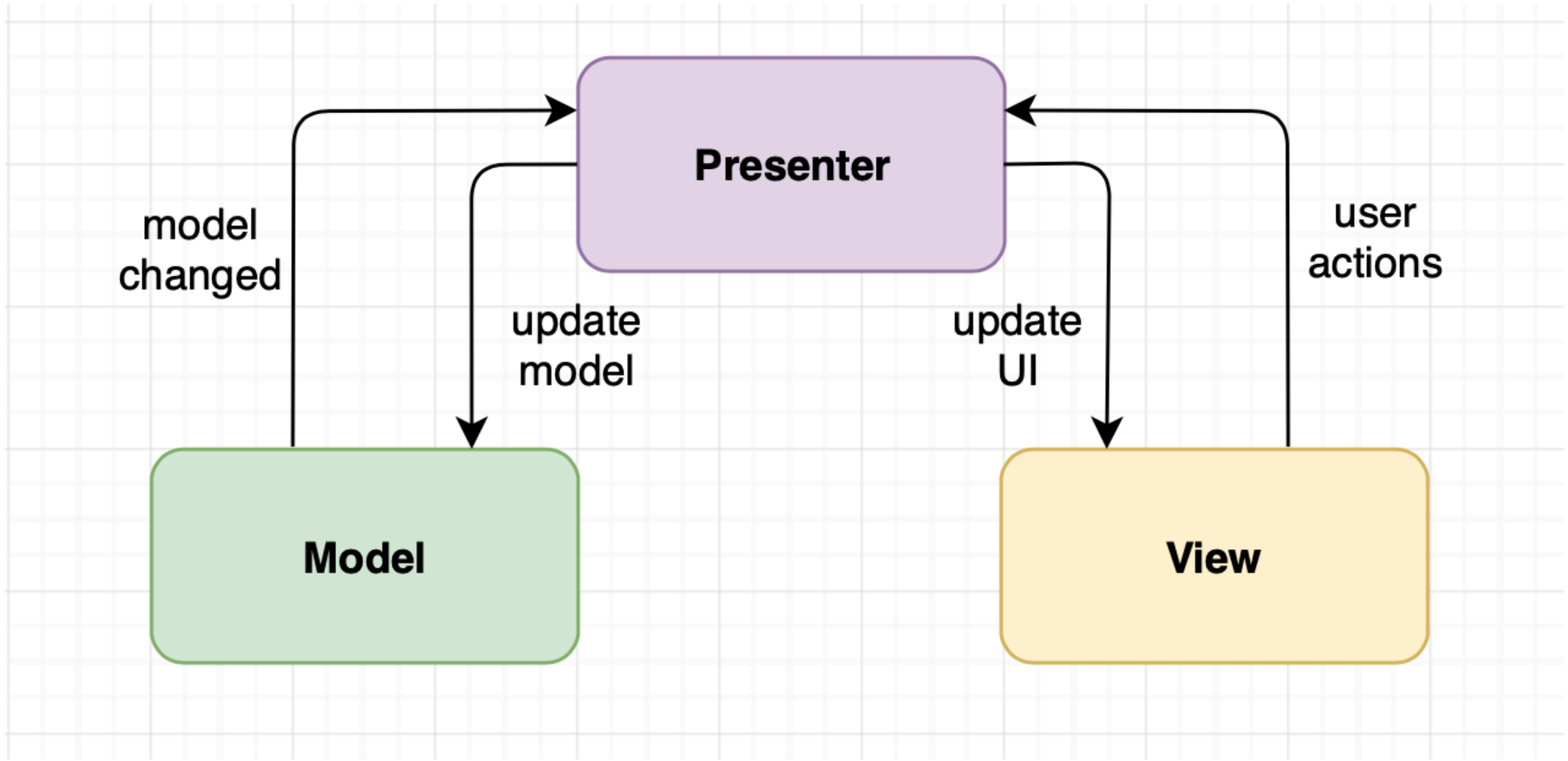
Popular project structures

MVC (Model View Controller)
MVP (Model View Presenter)
MVVM (Model View ViewModel)
Clean Architecture

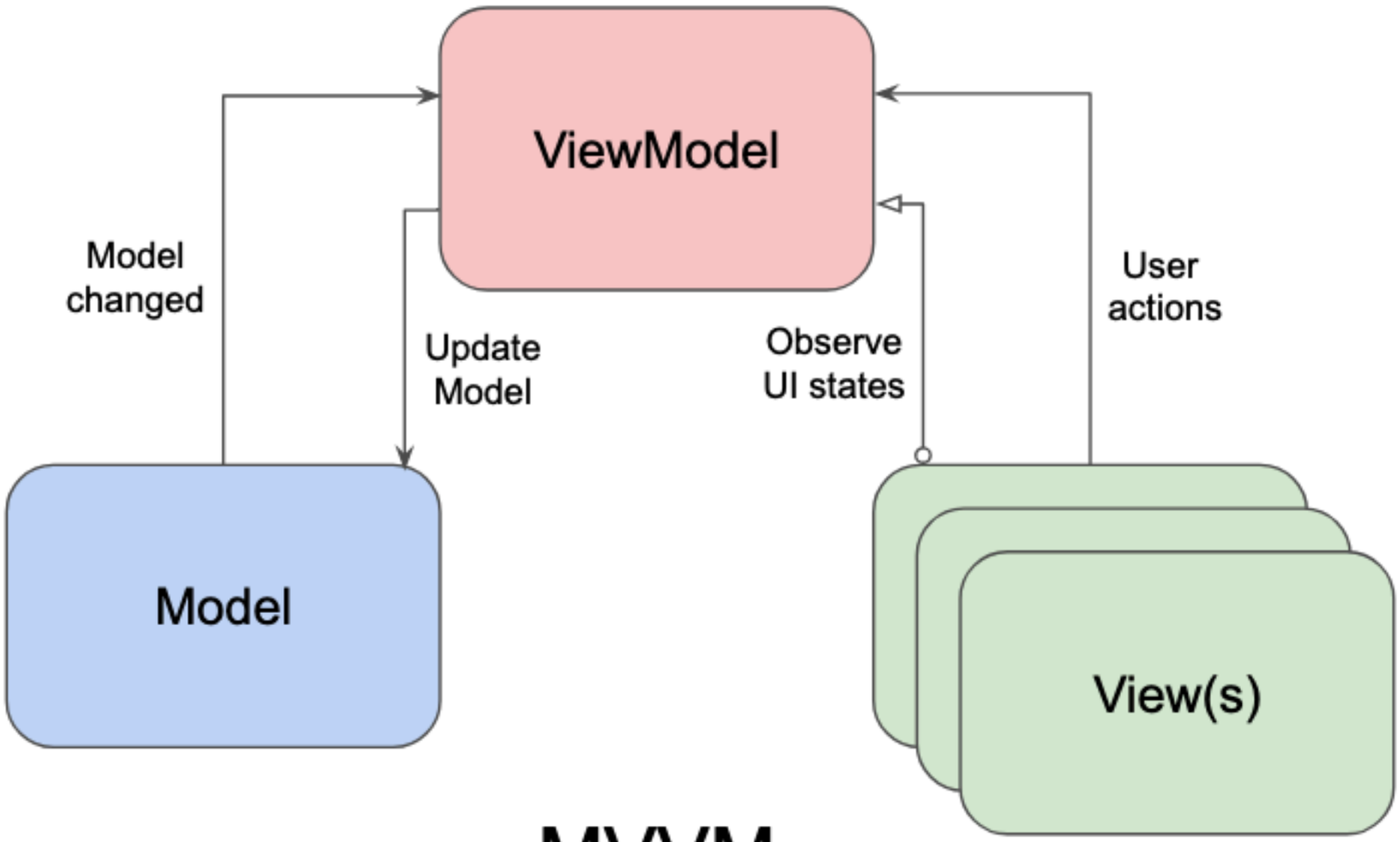
MVC



MVP

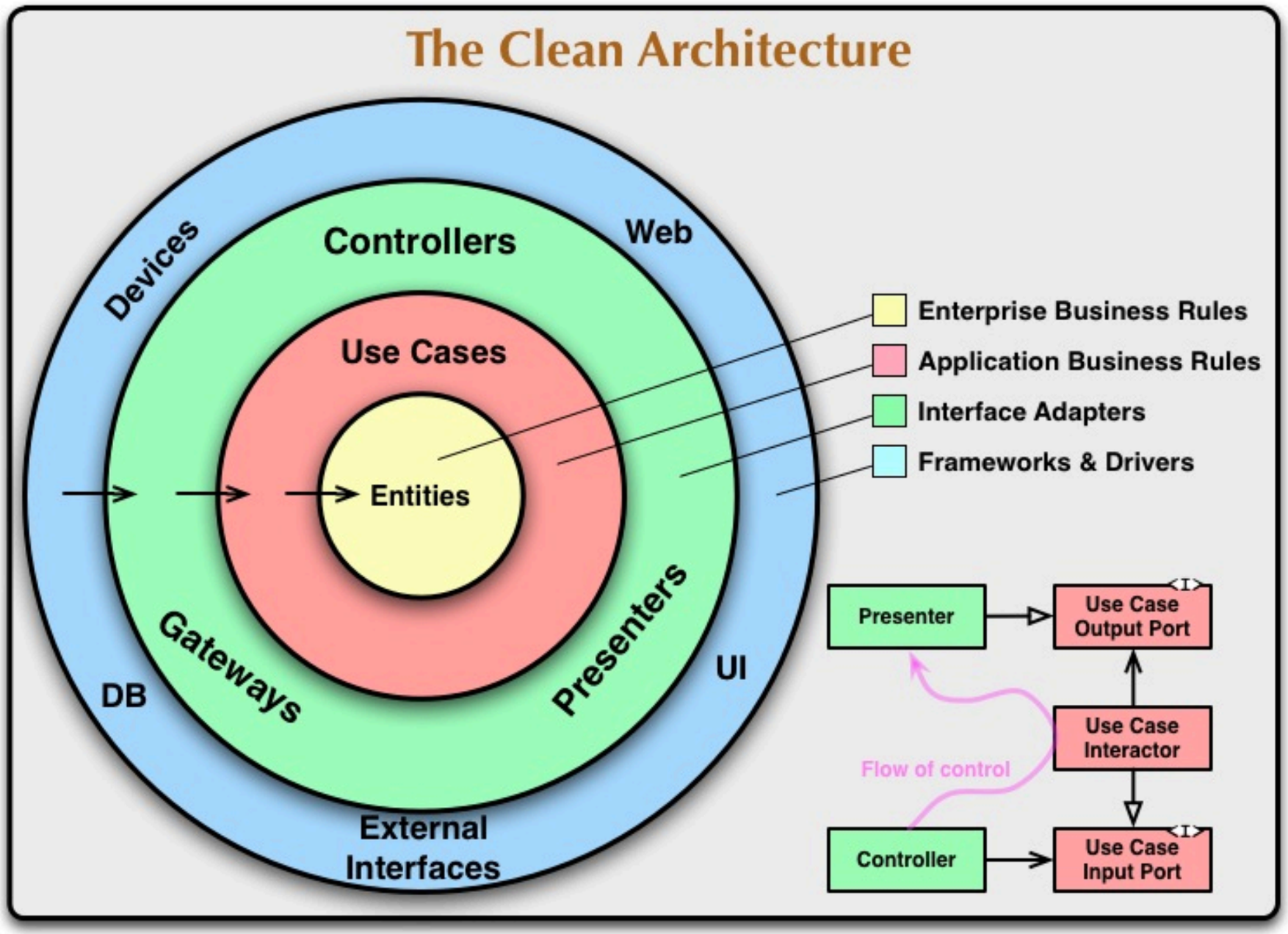


MVVM



MVVM

Clean Architecture



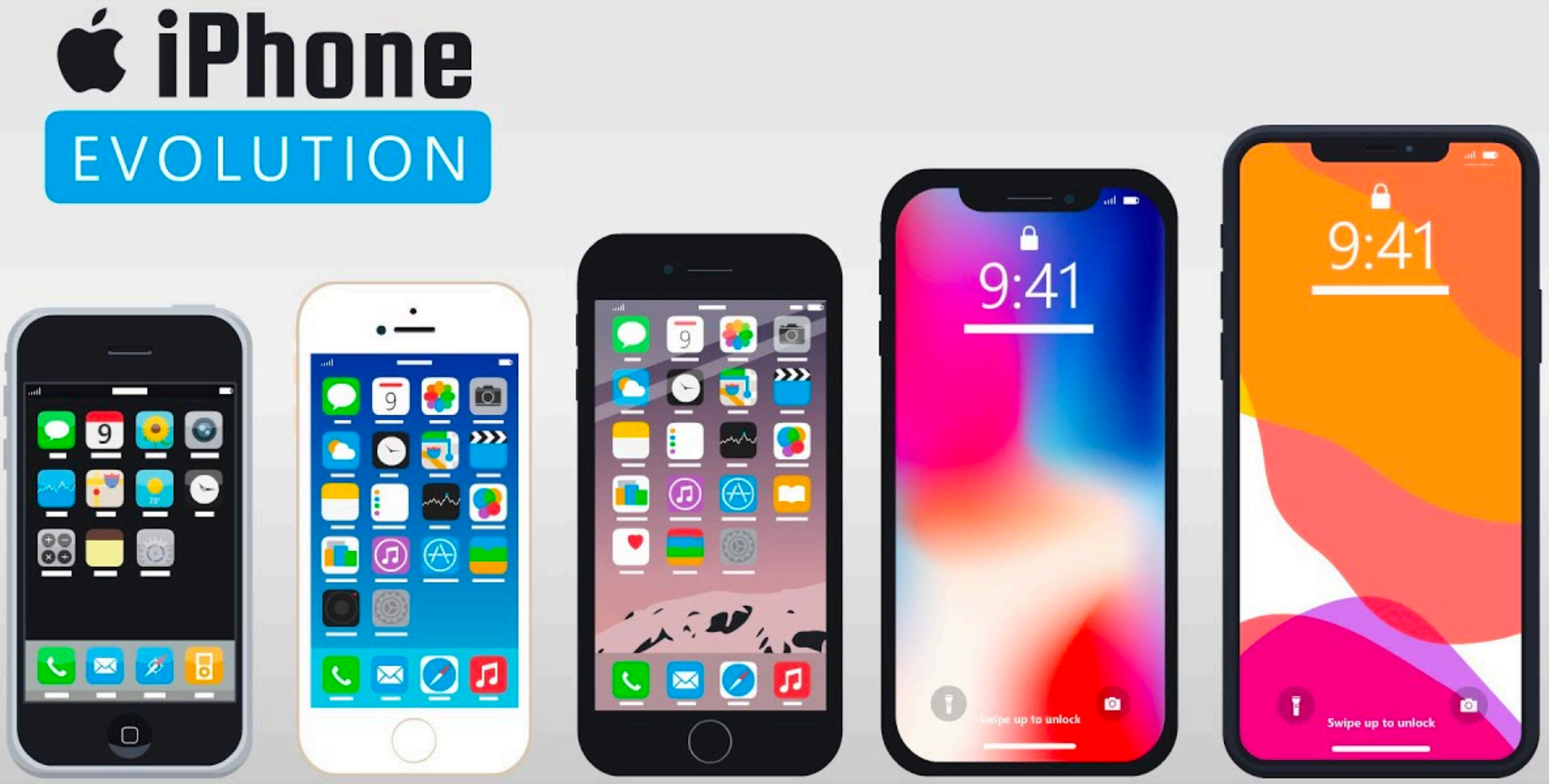
Workshop
with better project structure

Mobile App Testing

Important of Mobile Testing

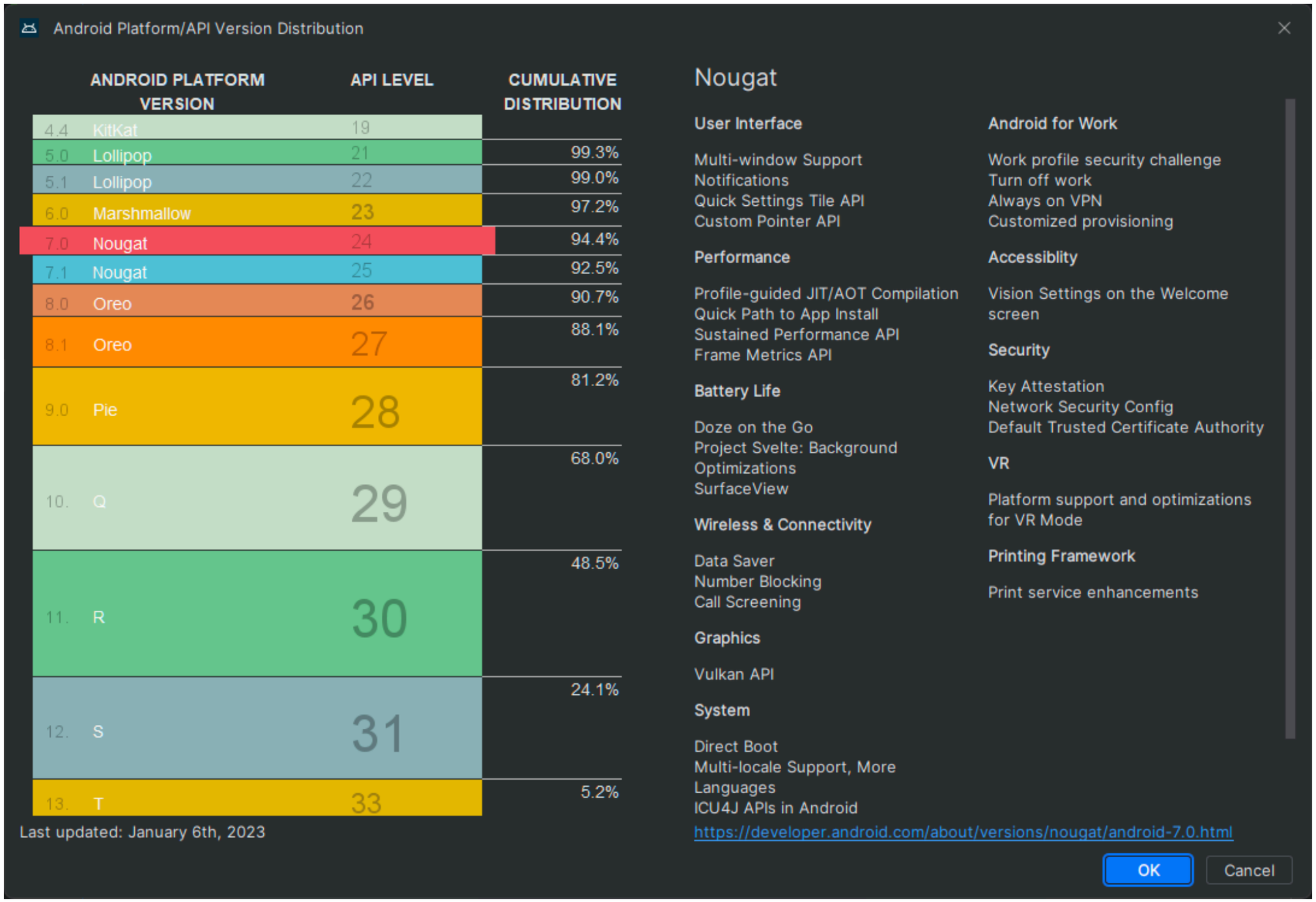
Variety of mobile devices
Different of operating system
Constant upgrades of operating system

iPhone ?



Android ?

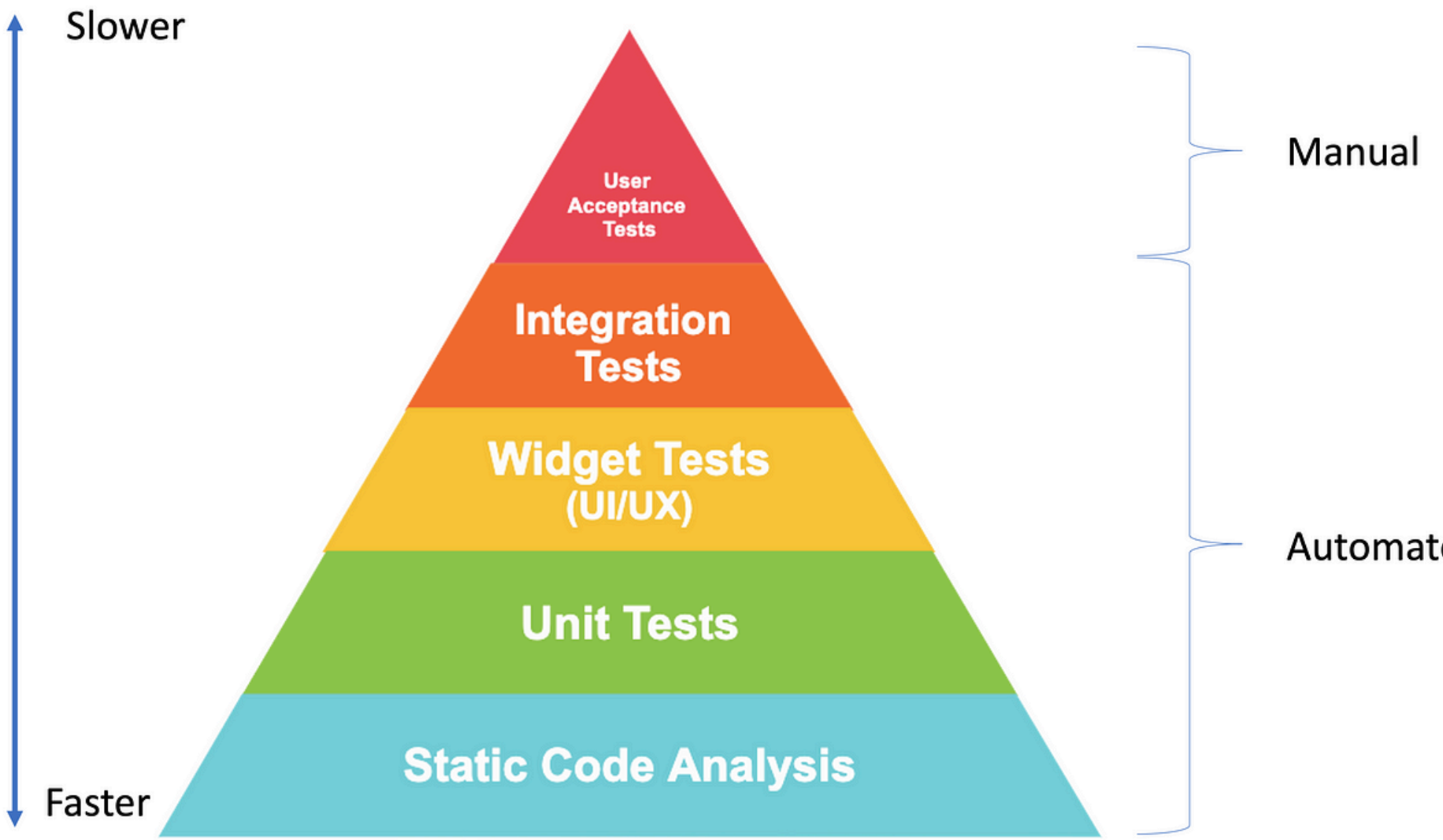
Android Version ?



Flutter testing

Unit test
Widget test
Integration test
External test with Appium and Flutter

Flutter testing



Static Code Analysis

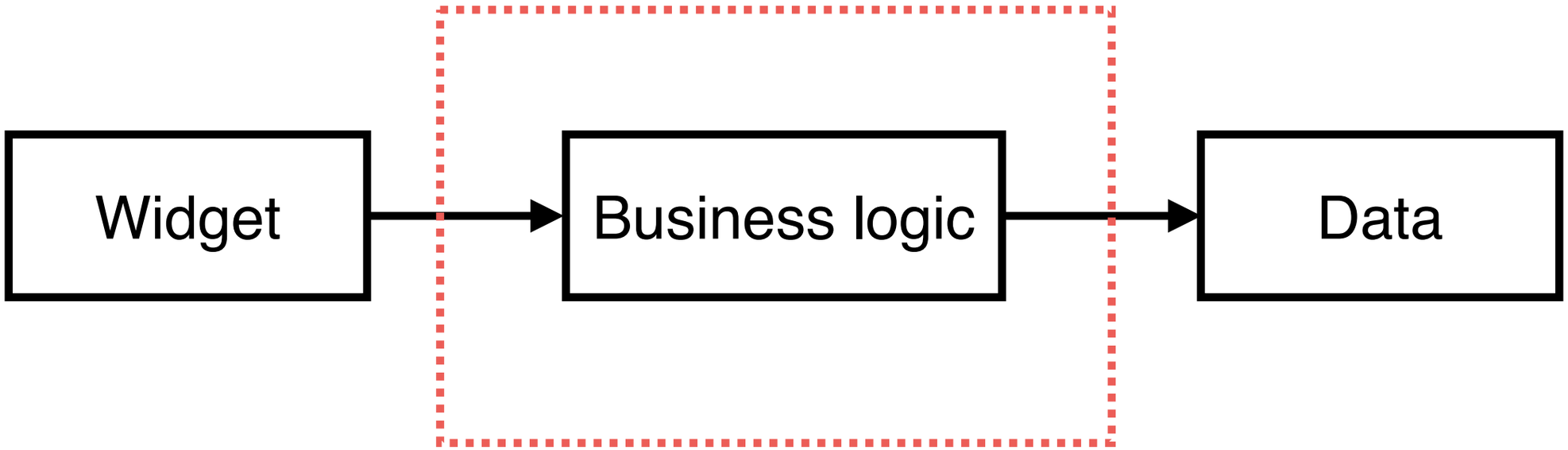
Working on compile-time before run unit test
Help to detect possible defect in early stage
Dart lint

$flutter analyze

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

Unit Tests

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

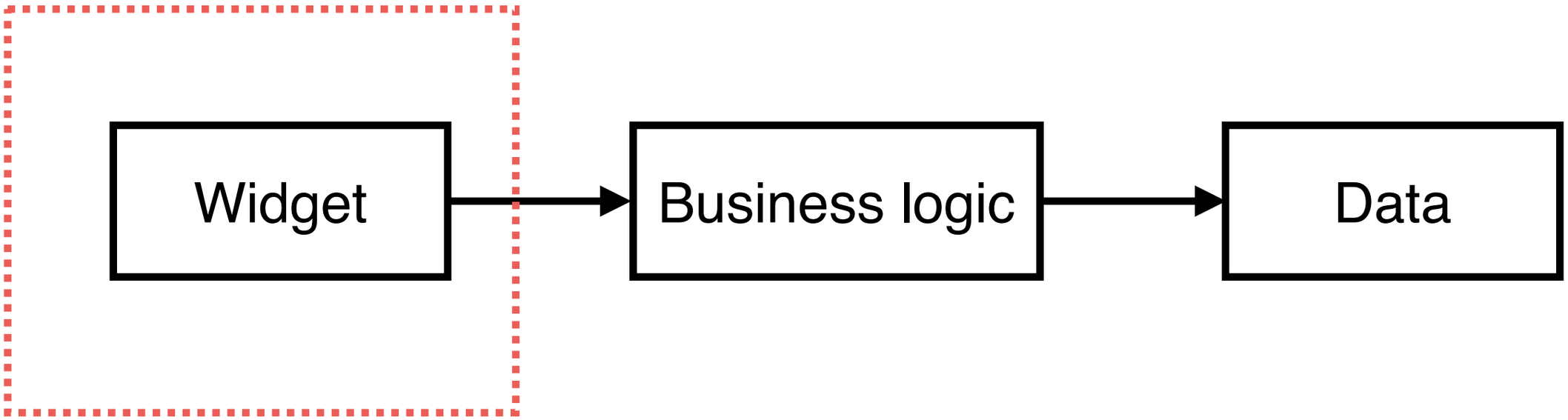


Widget Tests

Verify the widget's UI looks and interacts as expected

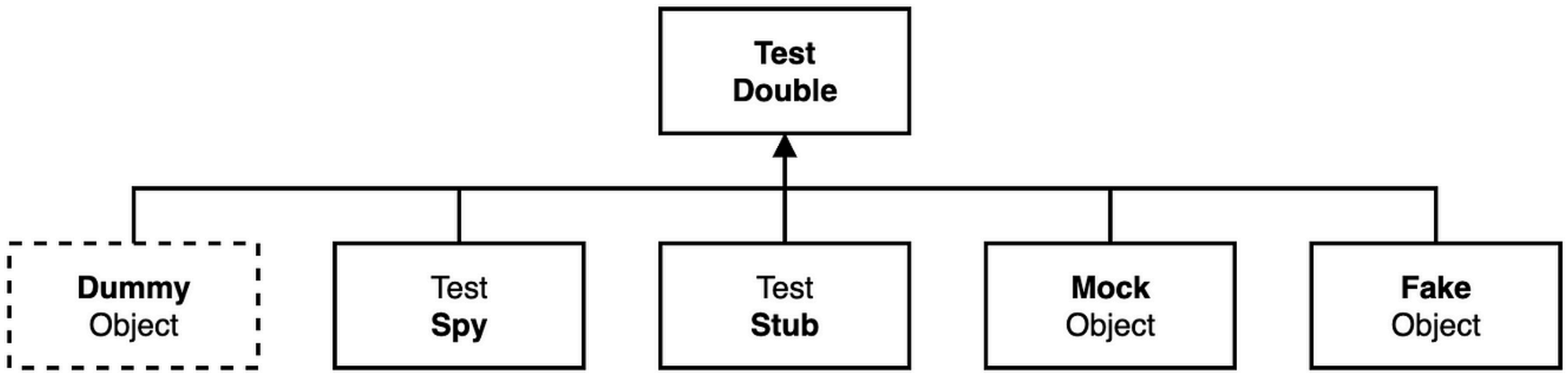
Not run on real device or emulator

Working with **Test double**



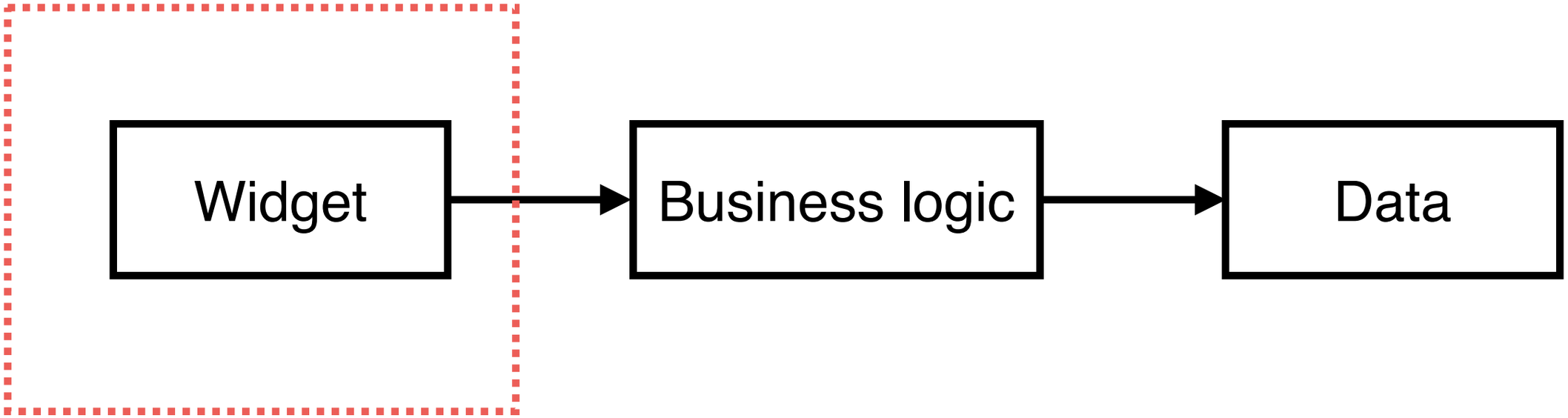
Test Double

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

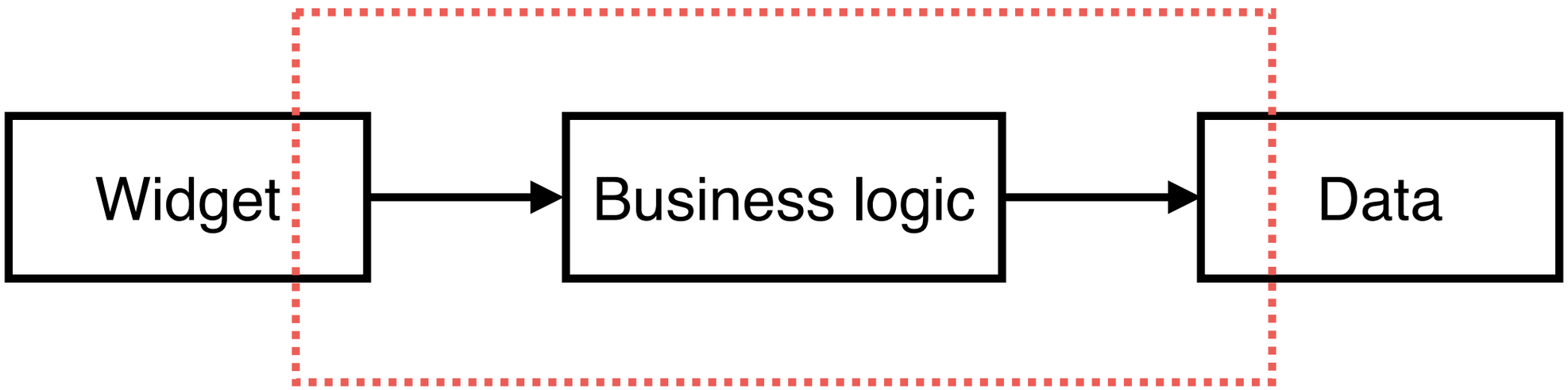
Ensure UI integrity when we are working in a large team

https://pub.dev/packages/golden_toolkit

Integration Tests

Individual modules are combined and tested as a group

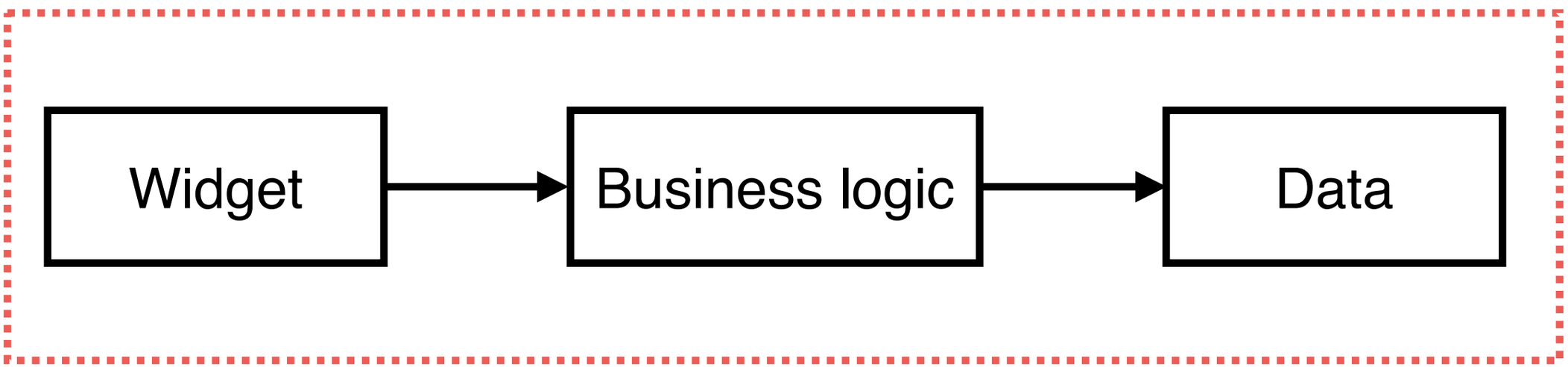
Working with **Test double**



Integration Tests

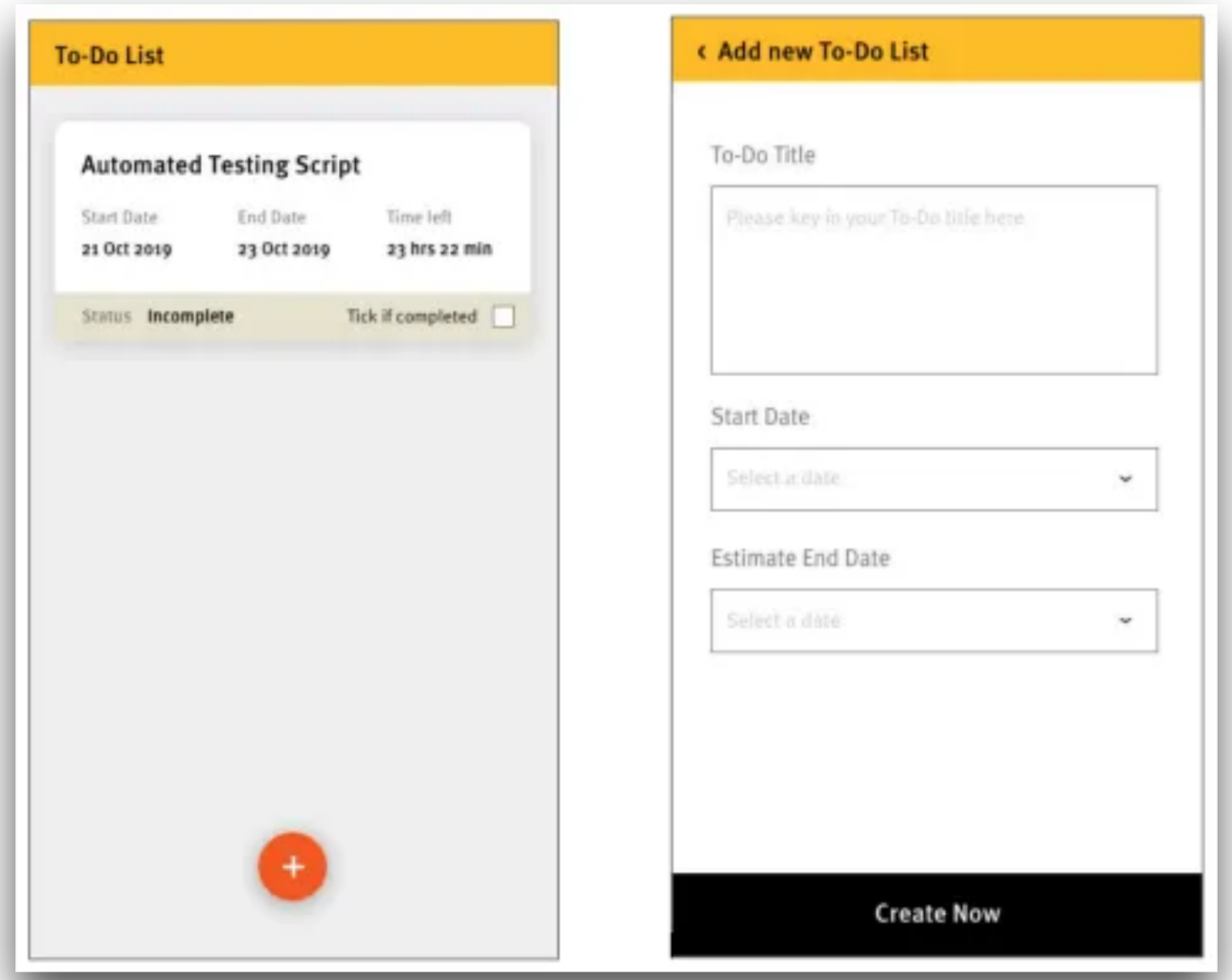
Individual modules are combined and tested as a group

Run on real device or emulator

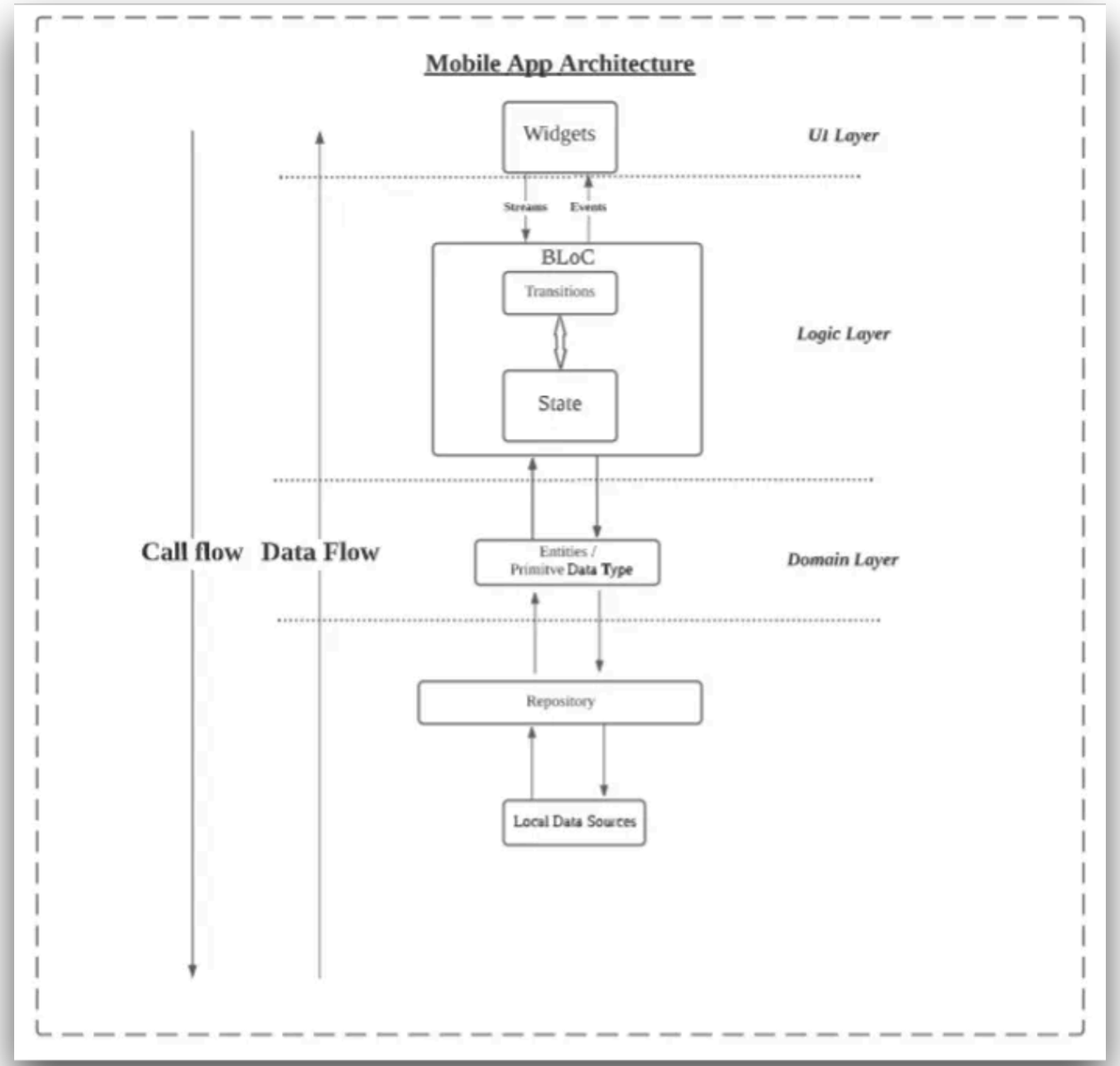


Workshop
with Flutter Testing

Sample Project



Project Structure



Q/A