Introduction to mobile development using Flutter

Mobile Hive - Palo IT (Thailand)

Krisada Vivek

Meet your instructor

Objective overview

In this course you will learn

- Understand how to use flutter framework
- Understand basic of Dart lang*
- Create your applications with multiple functionalities
- Know how to design mobile architecture
- Put your app into Production
- Know where to find changes of framework

Table of Contents

These are the topic that you are going to train throughout this course

Chapter 1: Installation and Setup for Flutter Development

Flutter SDK installation

Overview Process

- Go to Official Website -> https://flutter.dev
- Getting Start tab
- Proceed the instruction steps
  - Select your operating system (macOS)
  - Download .zip file
  - Extract the .zip
  - Place in desire location
  - Update system path ~> export PATH="$PATH:`pwd`/flutter/bin" to ~/.bash_profile or ~/.bash_rc
  - Restart your Terminal

Flutter SDK installation

Overview Process

- In you Terminal type "flutter" to identify is your setup success or not
  - If "Success" it's should prompt out lie this ->

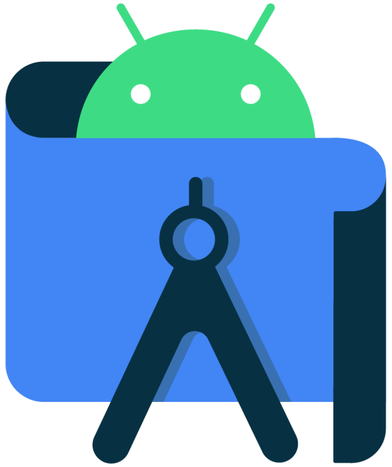
```
○ → flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.
```

  - If "No" please check your system variable path

macOS setup

Prerequisites

- For developing a mobile application you need 2 IDEs



macOS setup

Prerequisites

- For additional tools you will also need to install Homebrew => [The Missing Package Manager for macOS (or Linux) — Homebrew]

macOS setup

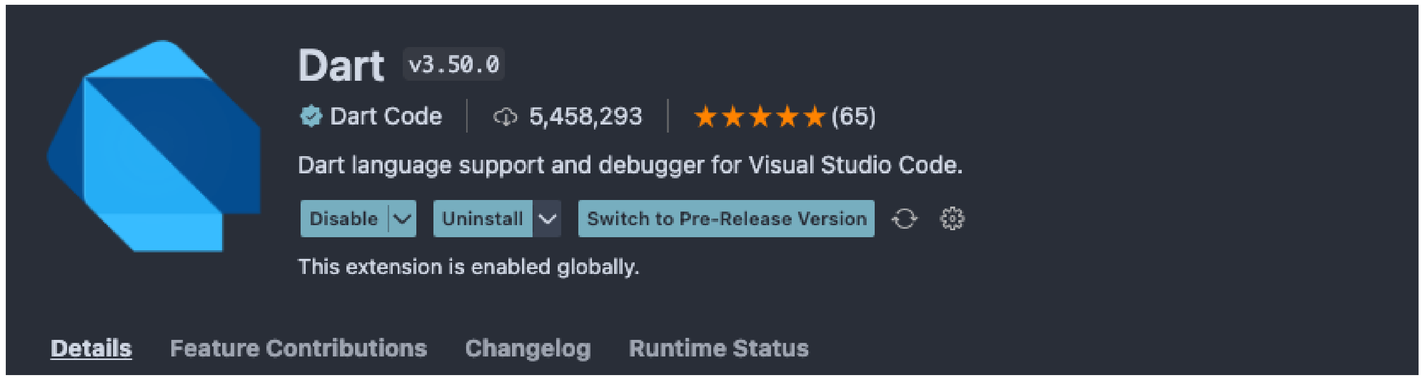
Select your IDEs

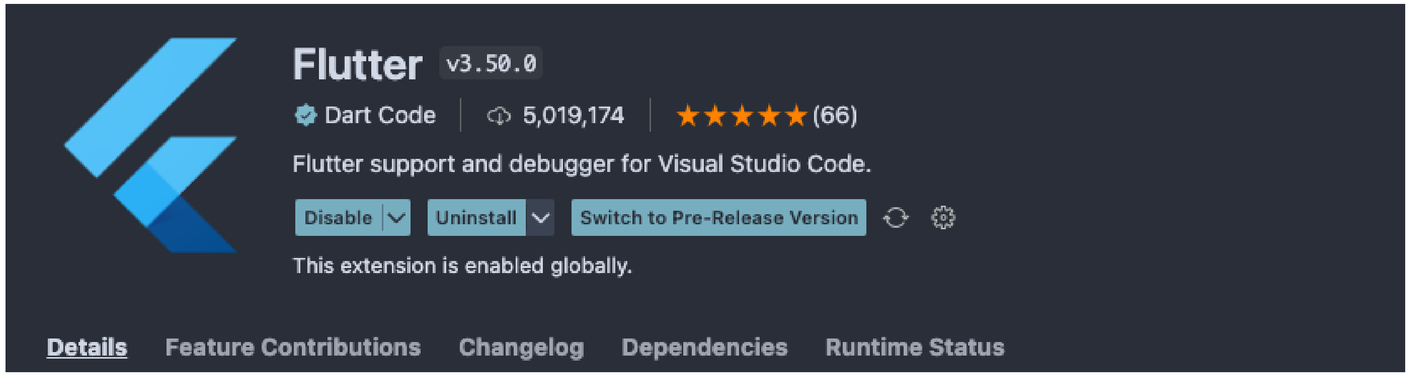
- VScode or Intelij

- But in this training we will use VScode as a main ide

macOS setup

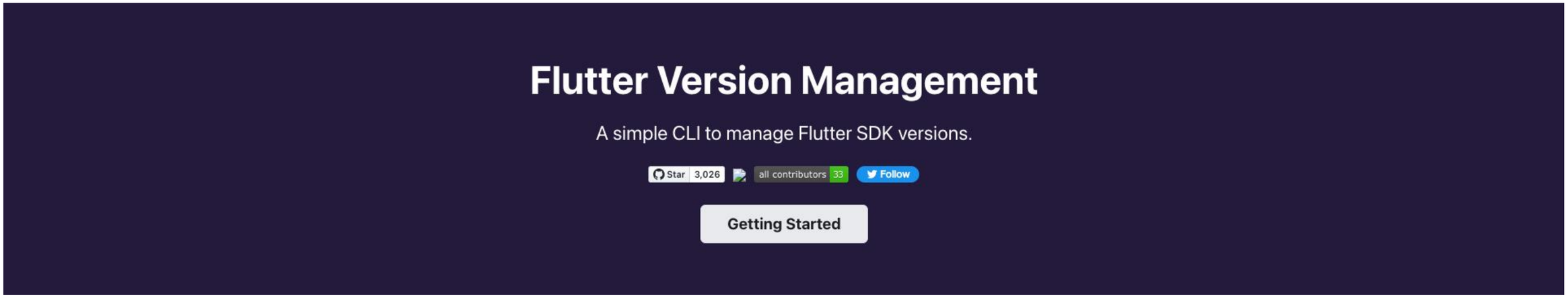
Setup an extensions





Can I install more than 1 flutter version?

Introduction to FVMs/DVMs

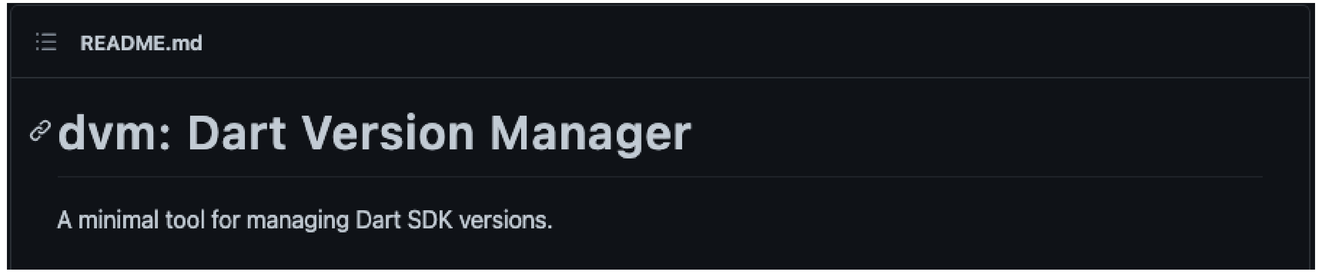


[Flutter Learning EP6: Managing Multiple version of Flutter SDK using FVM - YouTube]

Introduction to FVMs/DVMs

DVMs

- [cbracken/dvm: Dart Version Manager (github.com)]



Chapter 2: Creating 1st flutter app

Introductions

In this chapter we are going to

- Create 1st flutter application via CLI
- Know how to use the basic commands
- Fundamental Structure
- Run 1st Application on an Emulator/Simulator
- Learn other essential dev tools – DevTools/Debugger

Flutter Commands

These are basic command you are going to use in this course

- flutter create <application_name> => crate flutter project
- flutter doctor (-v) => command that will give you a prompt about version and additional dependencies that required to develop a flutter application
- flutter pub get => get and install dependencies in your project
- flutter run => run your 1st flutter app
- flutter upgrade

Create 1st flutter application using CLI

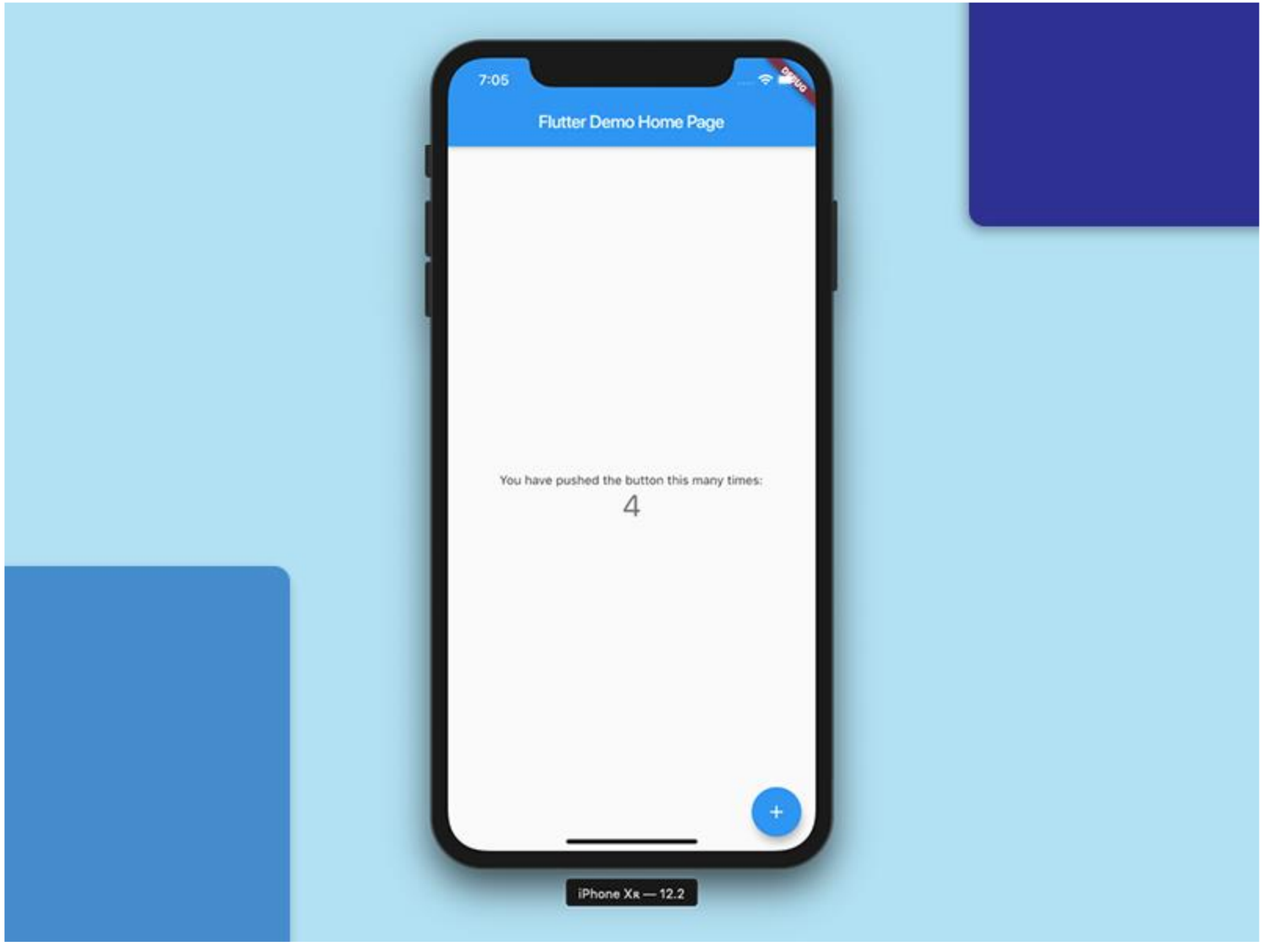
via command line

- Type "flutter create <application_name>" and lets magic happen
- Open IDE to see the code
- Take a quick look at the code

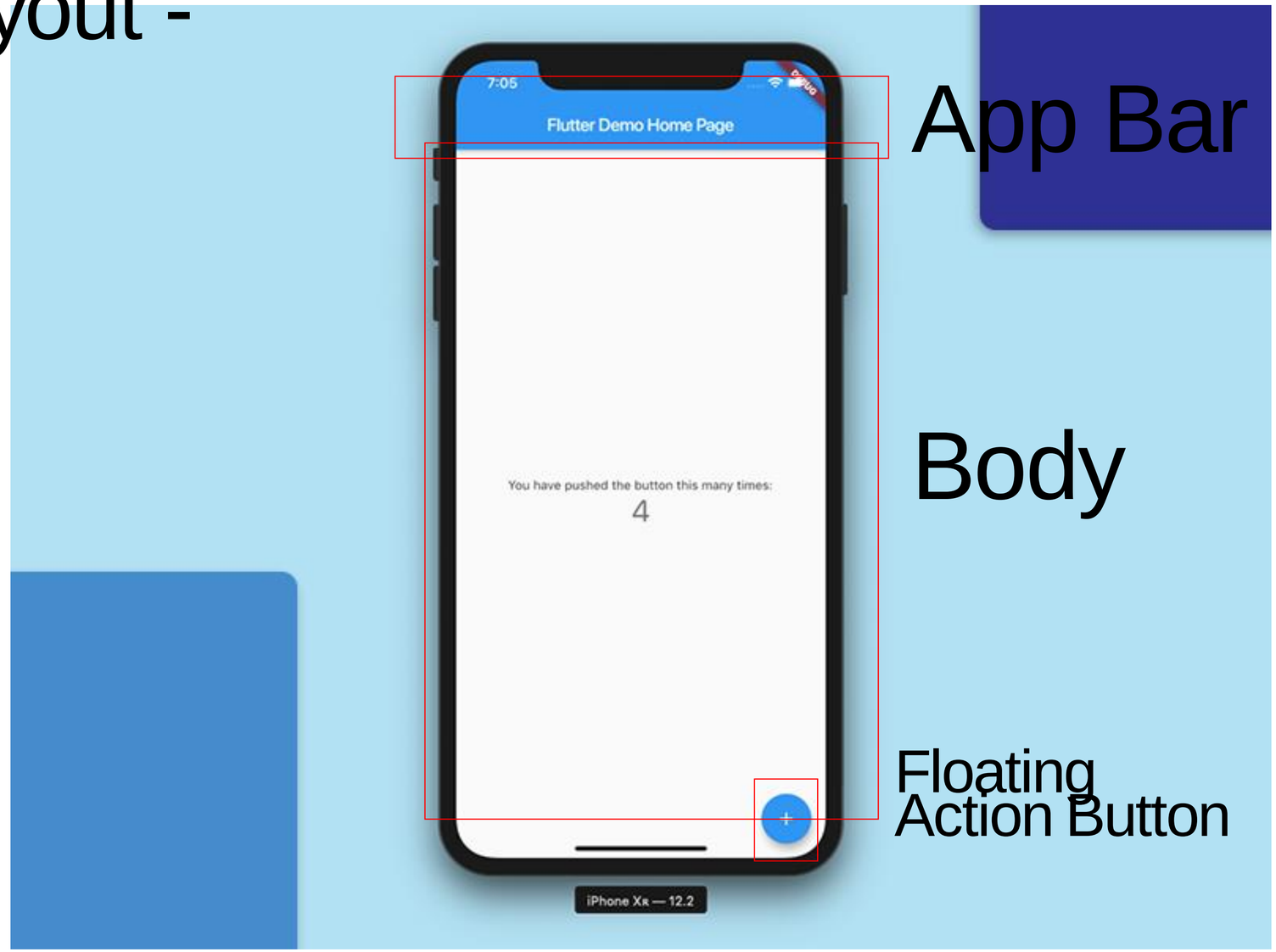
Run your 1st application

1st Checkpoint

- Start iOS Simulator ~> open –a Simulator
- Start Android Emulator ~> open Android Studio
- Type "flutter run" to start your application for the 1st time


7:05
DEBUG
Flutter Demo Home Page
You have pushed the button this many times:
4
+
iPhone XR — 12.2

Basic Layout - Scaffold



Run your 1st application

1st Checkpoint

- Look at your terminal, you will see additional information that essential to your development

```
○ → flutter run
Launching lib/main.dart on iPhone 14 Pro Max in debug
mode...
Running Xcode build...
 └Compiling, linking and signing...
Xcode build done.
20.4s
Syncing files to device iPhone 14 Pro Max...

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).

💪 Running with sound null safety 💪

An Observatory debugger and profiler on iPhone 14 Pro Max
```

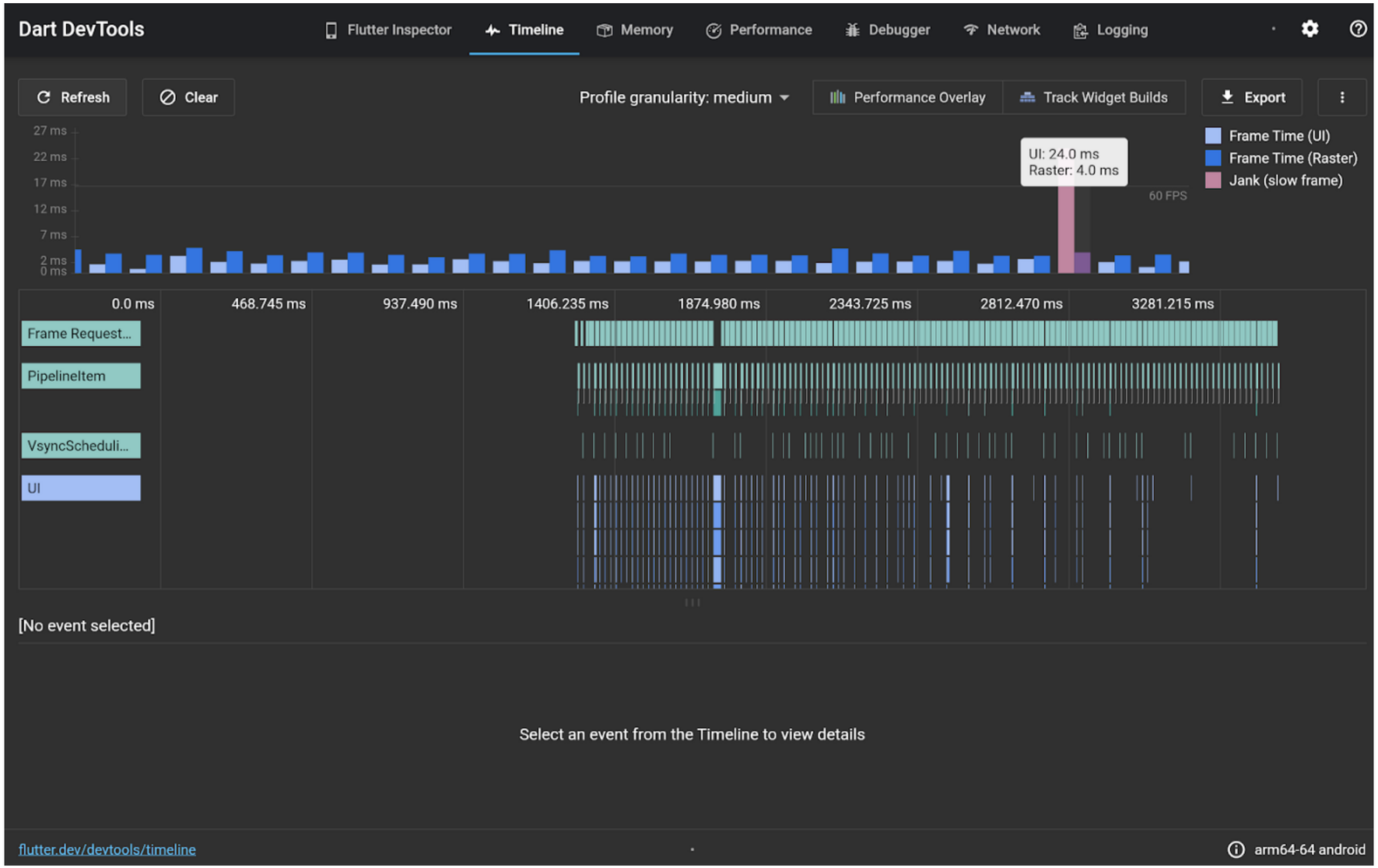
An essential Tools

DevTools

- Built-in tools that helps developer identify certain part of an application such as UIs, performance, memory consumptions

Debugger

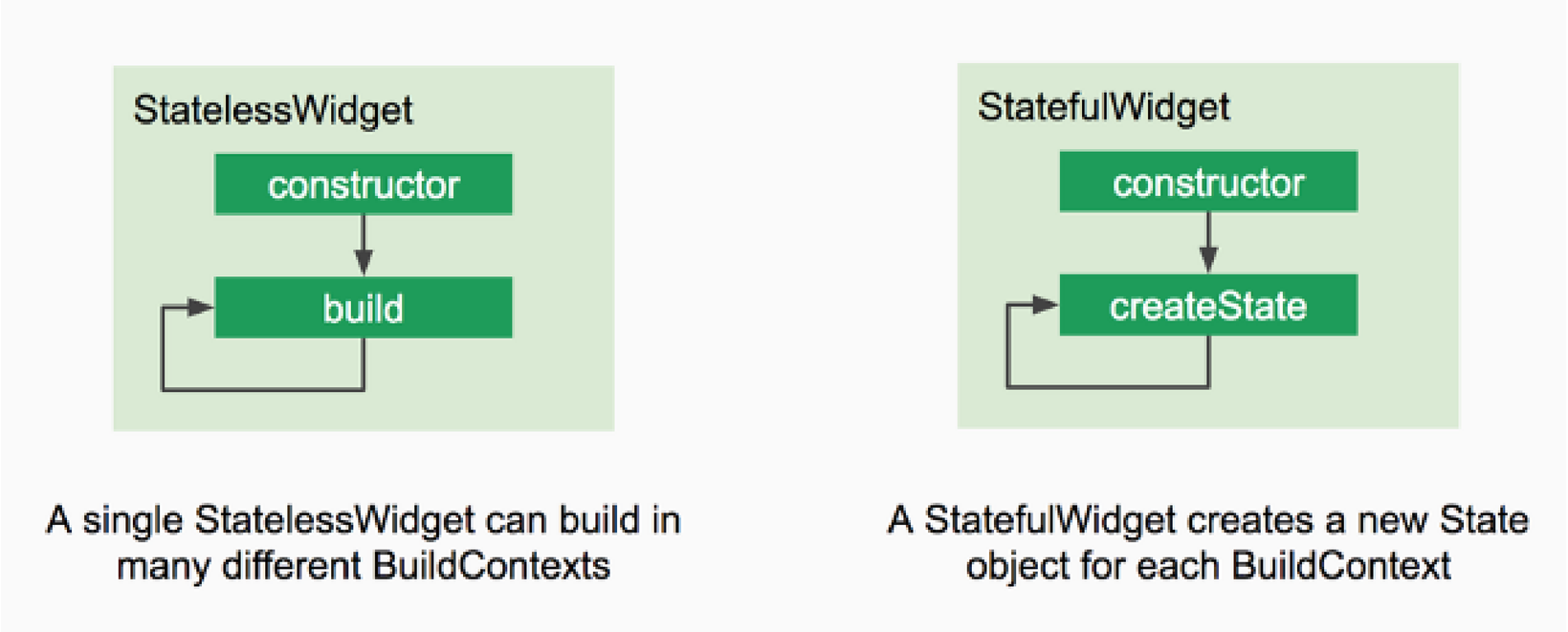
- Internal VScode debugger


Dart DevTools
Flutter Inspector
Timeline
Memory
Performance
Debugger
Network
Logging
Refresh
Clear
Profile granularity: medium
Performance Overlay
Track Widget Builds
Export
27 ms
22 ms
17 ms
12 ms
7 ms
2 ms
0 ms
UI: 24.0 ms
Raster: 4.0 ms
60 FPS
Frame Time (UI)
Frame Time (Raster)
Jank (slow frame)
0.0 ms
468.745 ms
937.490 ms
1406.235 ms
1874.980 ms
2343.725 ms
2812.470 ms
3281.215 ms
Frame Request...
PipelineItem
VsyncScheduli...
UI
[No event selected]
Select an event from the Timeline to view details
flutter.dev/devtools/timeline
arm64-64 android

Fundamental structure

Stateless vs Stateful

- In flutter we have 2 types of core widget
  - Stateless Widget
  - Stateful Widget



Stateful Widget

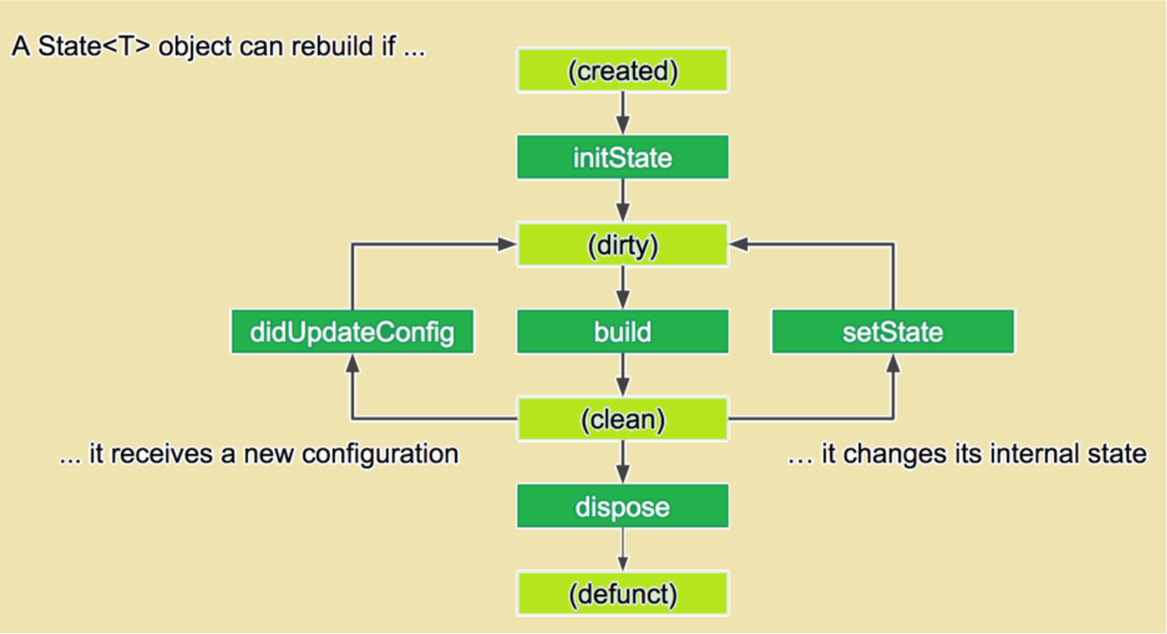
- Plenty of "state" -> mutable variable's value
- Change within its widget (local state)
- Life Cycle
- Can call setState(){})
- // add example of stateful state less (5 quick question)

```
class MyApp extends StatefulWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

Widget Lifecycle

Born – Live – Die – Repeat

- Terminologies in modern frontend framework,
- Manage things need to display – not display (UI element)
- Do some task before components appeared
- initState(), dispose(),build(), didChangeDependencies()



Mini Checkpoint 1

```
class MyHomePage extends <??> {
  const MyHomePage({super.key, required this.title});

  final String title;

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

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:
```

```
class MyApp extends <??> {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

```
class SamplePage extends <??> {
  const SamplePage({super.key});


  @override
  State<SamplePage> createState() => _SamplePageState();
}

class _SamplePageState extends State<SamplePage> {
  @override
  void initState() {
    super.initState();

    /// ...some code
  }

@override
  void dispose() {
    /// ...some code
    ///
    super.dispose();
  }
```

```
class MiniCheckpoint15 extends <??>{
  const MiniCheckpoint({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ScaffoldLayout(
      body: Center(
        child: Image.asset("path_to_image"),
      ),
    );
  }
}
```

Stateless Widget

- "not" retain any variable' value
- Give constructor, Then build
- State may control by it's parent

```
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(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

Recap

Stateless Stateful

- To identify which widget type are
  - Pattern
  - initState()
  - SetState

  - 5 mins break

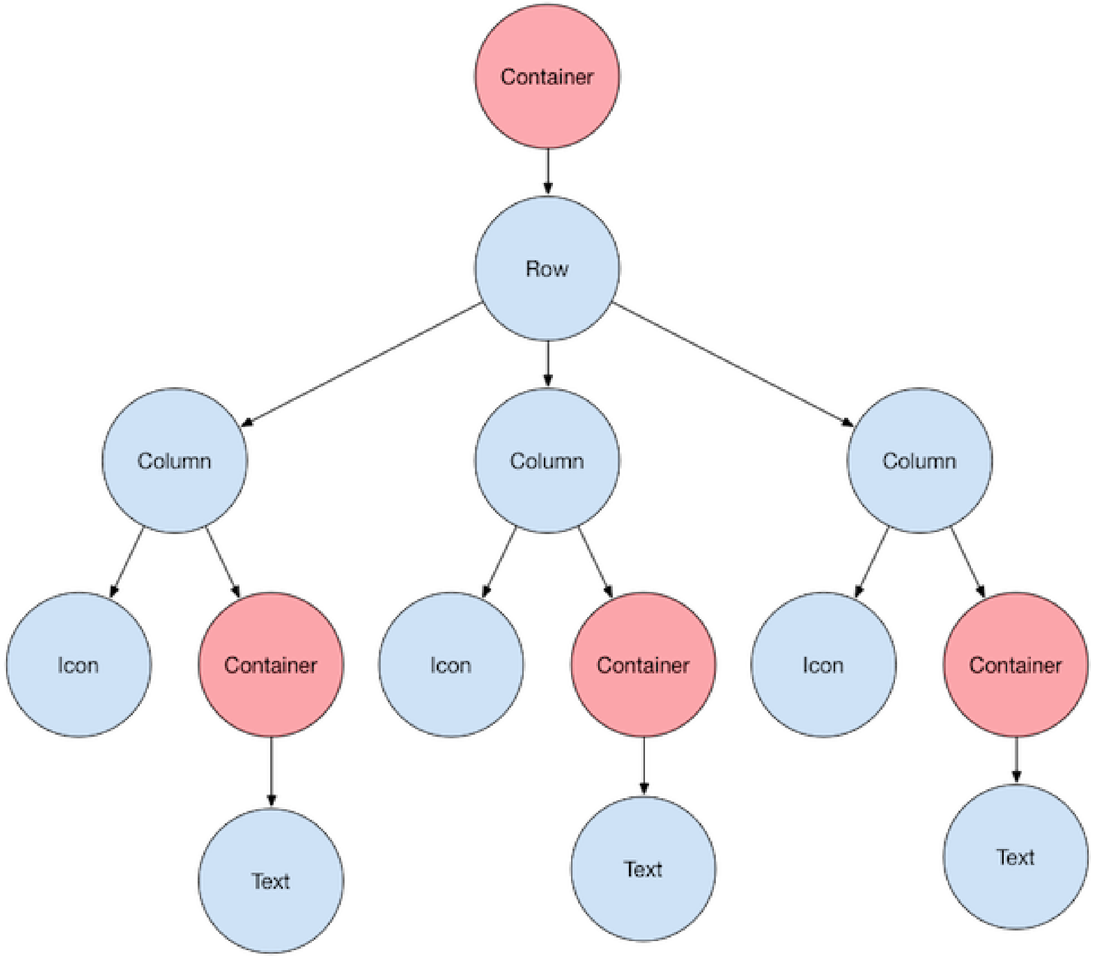
Widget Tree

How your code looks like

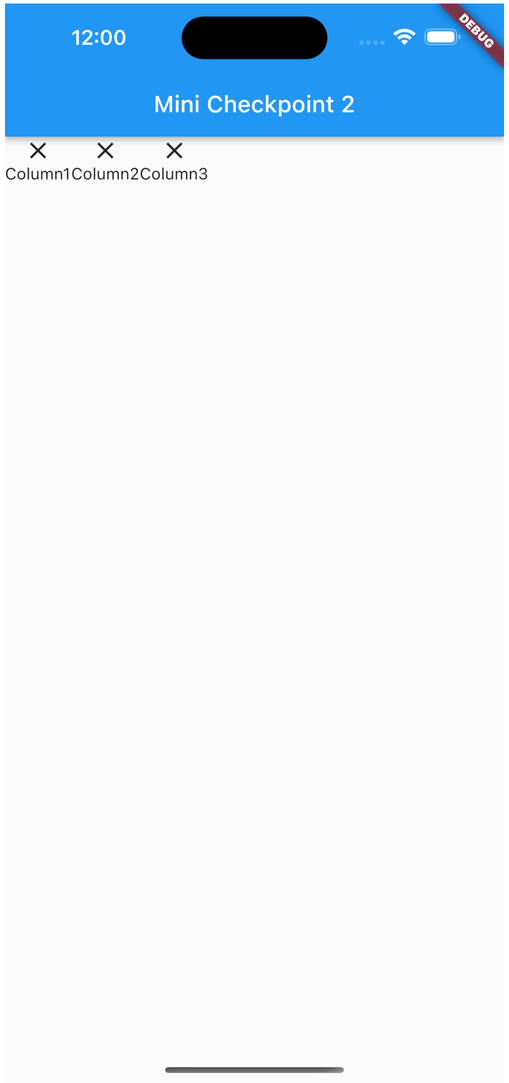
- Concept => Layers – Section - branches

- Terminologies
  - Parent
  - Child
  - Children
  - Nested Structure

- Performance – D.R.Y not W.E.T



Mini Checkpoint 2



Chapter 4: Why Flutter

TLDR;

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.

Advantages

- Native feeling
- Single code base for Multiple platform (Universal)
- Rich Animations – 60FPS smoothness
- Beautiful UIs – Material Design

Limitations

- Single Code base => if else conditions
- Multiple platform -> knowledge in multiple platform to implement additional native functionalities
- Rich Animations – 60FPS smoothness => performance / battery life/ memory consumption management
- iOS -> any new feature need to wait for 3rd party (open-source community) to develop and maintain
- iOS -> most of the patch focus on iOS improvement and optimization
- Flutter try to be jack of all trades

Common Pits Fall

- Using Flutter for only single platform
  - Alternatives – Jetpack Compose, SwiftUI
- Code once, you need to choose which platform that can goes together
- Replacing -> co-exists

Take A Break 15 mins

Chapter 5: Introduction to Widgets

Introduction to Widgets

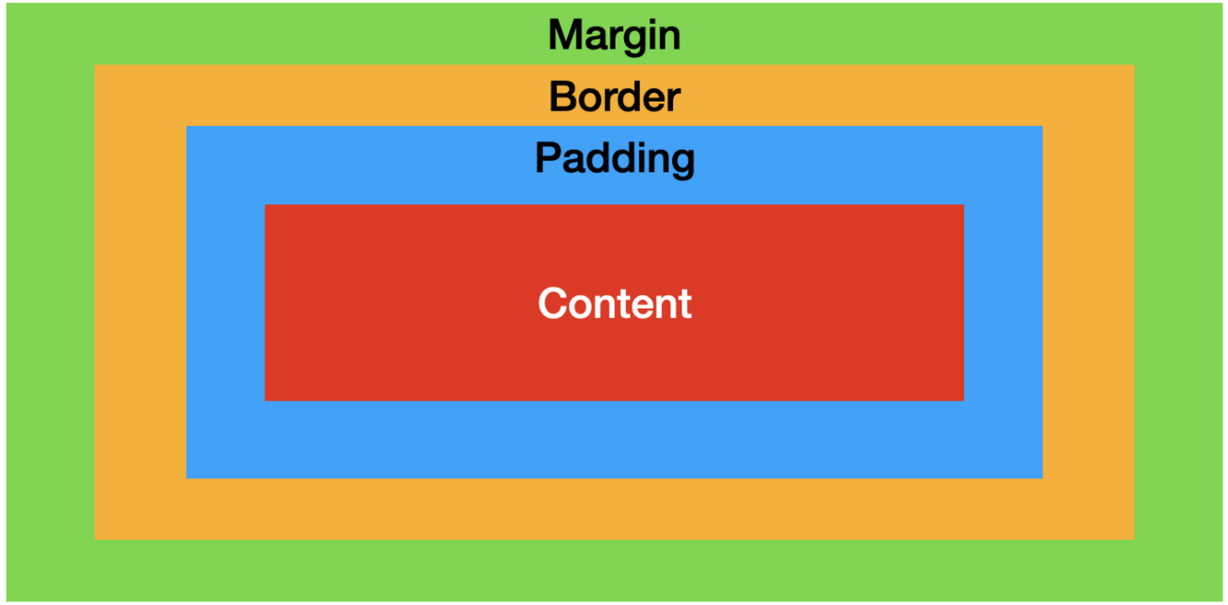
- Any UI components in flutter, either small or large are called "Widgets"

Basic widget - Container

- Multiple Properties
- BoxModel (CSS)
  - Padding
  - Margin
- Alternative
  - SizedBox()

```
Container(
  constraints: BoxConstraints.expand(
    height: Theme.of(context).textTheme.headline4!.fontSize! * 1.1 + 200.0,
  ),
  padding: const EdgeInsets.all(8.0),
  color: Colors.blue[600],
  alignment: Alignment.center,
  transform: Matrix4.rotationZ(0.1),
  child: Text('Hello World',
    style: Theme.of(context)
        .textTheme
        .headline4!
        .copyWith(color: Colors.white)),
)
```

Box Model



Mini Checkpoint 3

Fill colors and spaces to your container

Basic widget - Text

```
Text(
  'Hello, $_name! How are you?',
  textAlign: TextAlign.center,
  overflow: TextOverflow.ellipsis,
  style: const TextStyle(fontWeight: FontWeight.bold),
)
```

- we call this typography

Mini Checkpoint 4

Playing with Text

Basic widget - Image

To display image inside you application

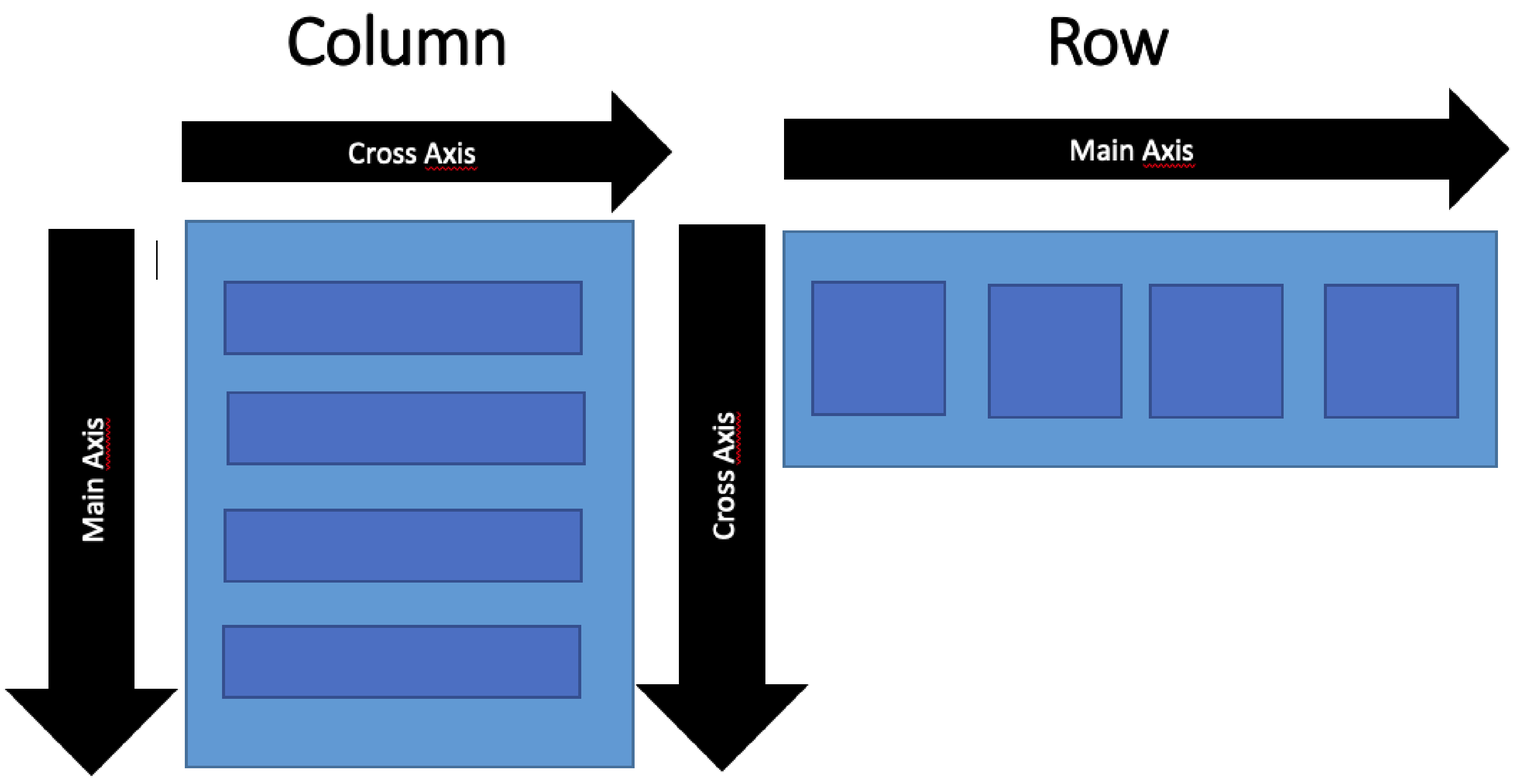
- Image.assets -> Load image from local
  - Create "assets" folder
  - Add path to pubspec.yaml
- Image.network -> Load image from network

```
const Image(
  image: NetworkImage('https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg'),
)
```

Mini Checkpoint 5

Playing with Image

Basic widget – Row and Column




Flutter Clutter
Implementing edge detection in Flutter
Google News
Amazing: Flutter ist the most famous mobile
OVERFLOWED BY 152 PIXELS
BOTTOM OVERFLOWED BY 32 PIXELS

Mini Checkpoint 6

Adding scroll functionality to your column/row

- You encounter an overflow
- Wrap SingleChildScrollView
- Tell the direction of scroll
  - Axis.horizontal
  - Axis.vertocal

Introduction to Alignments

- 2 main properties for both row and column
  - MainAxisAlignment
  - CrossAxisAlignment

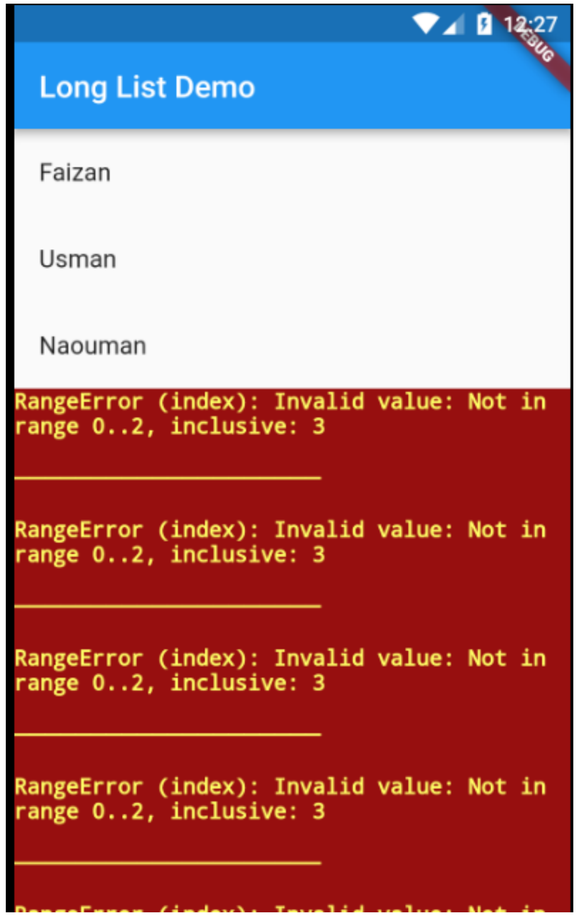
Mini Checkpoint 7

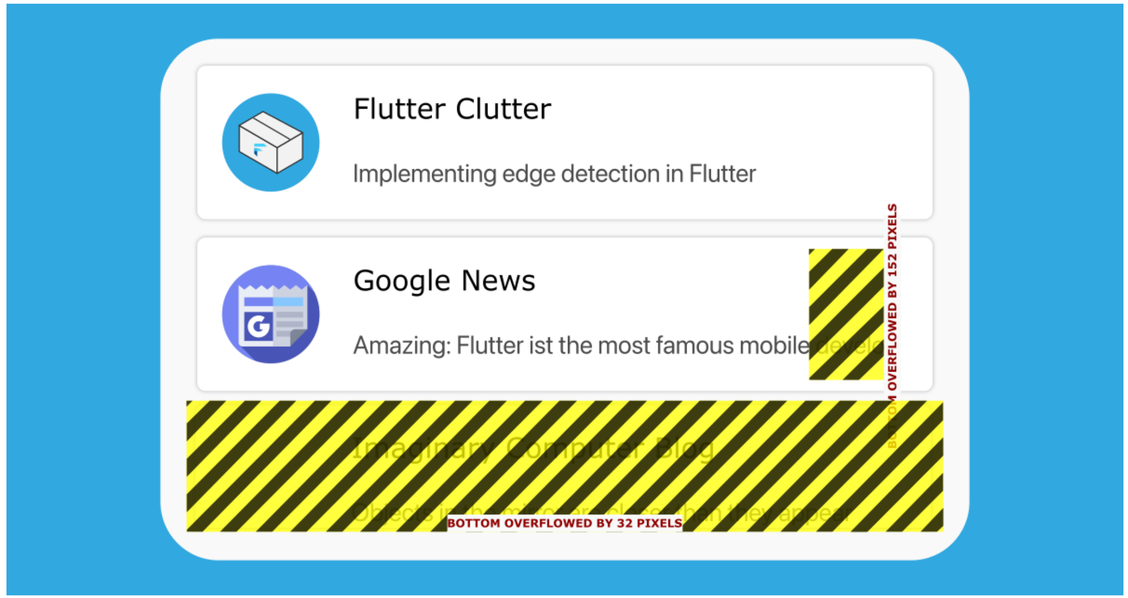
Align you widgets

Further Study

- Flutter Layout Cheat Sheet. Do you need simple layout samples for… | by Tomek Polański | Flutter Community

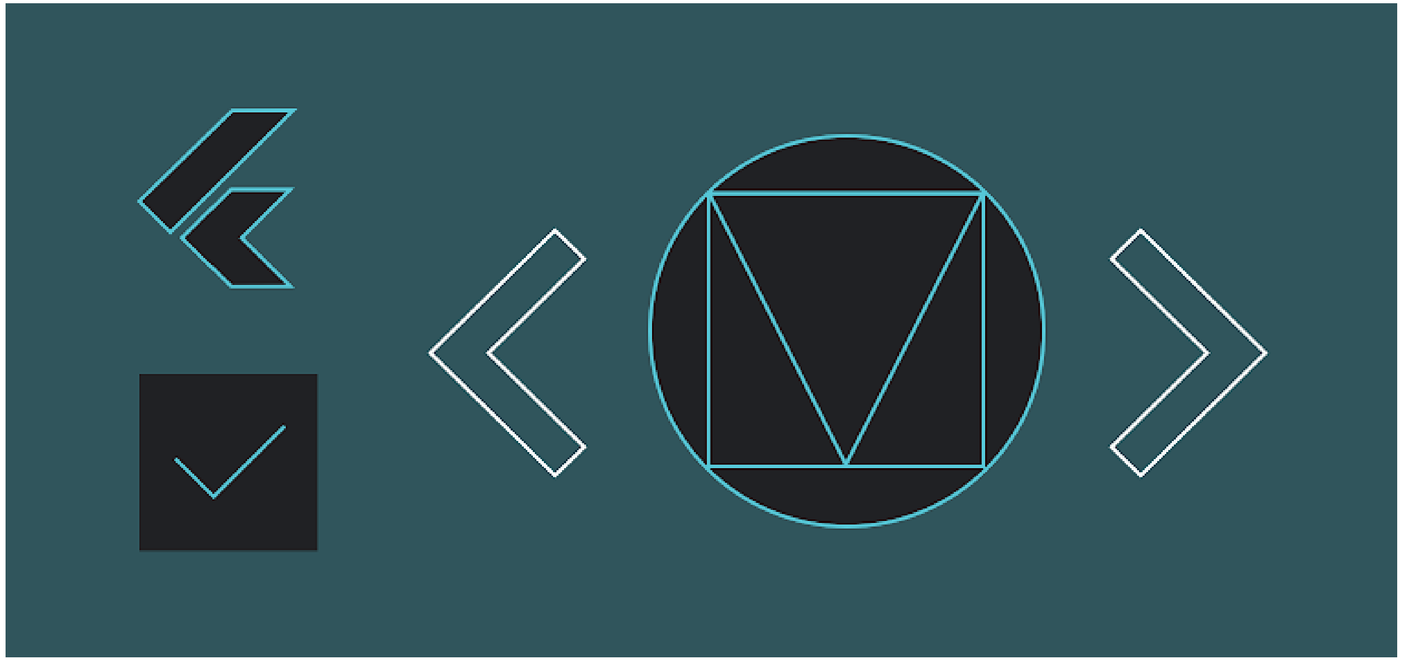
Widget Error Handling





Take A Break 10 mins

Chapter 7: Design System



Material Design

is an adaptable system of guidelines, components, and tools that support the best practices of user interface design

- Focus Platform
  - Mobile
    - Android (Native)
    - iOS (native) -> m3 not transition to iOS
    - Cross platform -> Flutter
  - Web
- Responsiveness
- Surface/ Shadows/ Reflection/ Ripple effect/ Transitions
- All the example in this section will refer to material design v2

Material Design

Typography

- Concept like CSS property
- Fonts
  - Size
  - Family
  - Weight – Bold, Regular,
  - Color
  - Etc.

Material Design

Iconography

- Refers to icons in material design
- Built-in to flutter framework
- Mostly 1:1 Ratio
- Multiple style – outlined, fill

Material Design

Components – buttons – show example

```
TextButton(
  style: ButtonStyle(
    foregroundColor: MaterialStateProperty.all<Color>
(Colors.blue),
    overlayColor: MaterialStateProperty.resolveWith<Color>(
      (Set<MaterialState> states) {
        if (states.contains(MaterialState.hovered))
          return Colors.blue.withOpacity(0.04);
        if (states.contains(MaterialState.focused) ||
            states.contains(MaterialState.pressed))
          return Colors.blue.withOpacity(0.12);
        return null; // Defer to the widget's default.
      },
    ),
  ),
  onPressed: () {},
  child: Text('TextButton')
)
```

```
ElevatedButton(
  style: ElevatedButton.styleFrom(
    onPrimary: Colors.black87,
    primary: Colors.grey[300],
    minimumSize: Size(88, 36),
    padding: EdgeInsets.symmetric(horizontal: 16),
    shape: const RoundedRectangleBorder(
      borderRadius:
BorderRadius.all(Radius.circular(2)),
    ),
  ),
  onPressed: () {},
  child: Text('Looks like a RaisedButton'),
)
```

- [Migrating to the New Material Buttons and their Themes - Google Docs]

- V1 - V2

Mini Checkpoint 8

- Add Buttons

Material Design

Theme

- Default theme for any modern platform – light / dark

```
MaterialApp(
   theme: ThemeData.light(), // or ThemeData.dark()
);
```

Material Design

Responsiveness

- Basic -> MediaQuery.of(context).size -> Apply example with button

```
MediaQuery.of(context).size.width
```

```
MediaQuery.of(context).size.height
```

Mini Checkpoint 9

Take A Break 10 mins

2nd Checkpoint – Creating UI from given image

aka Code in the dark

- Choose reference UIs – 5 min
- Think/discuss about how this will build using flutter (in terms of layout/ widgets) - 5 min
- Explain your plan - 5 min each
- 15 mins for coding
- Let discuss key take away - 5 mins each
- Q/A

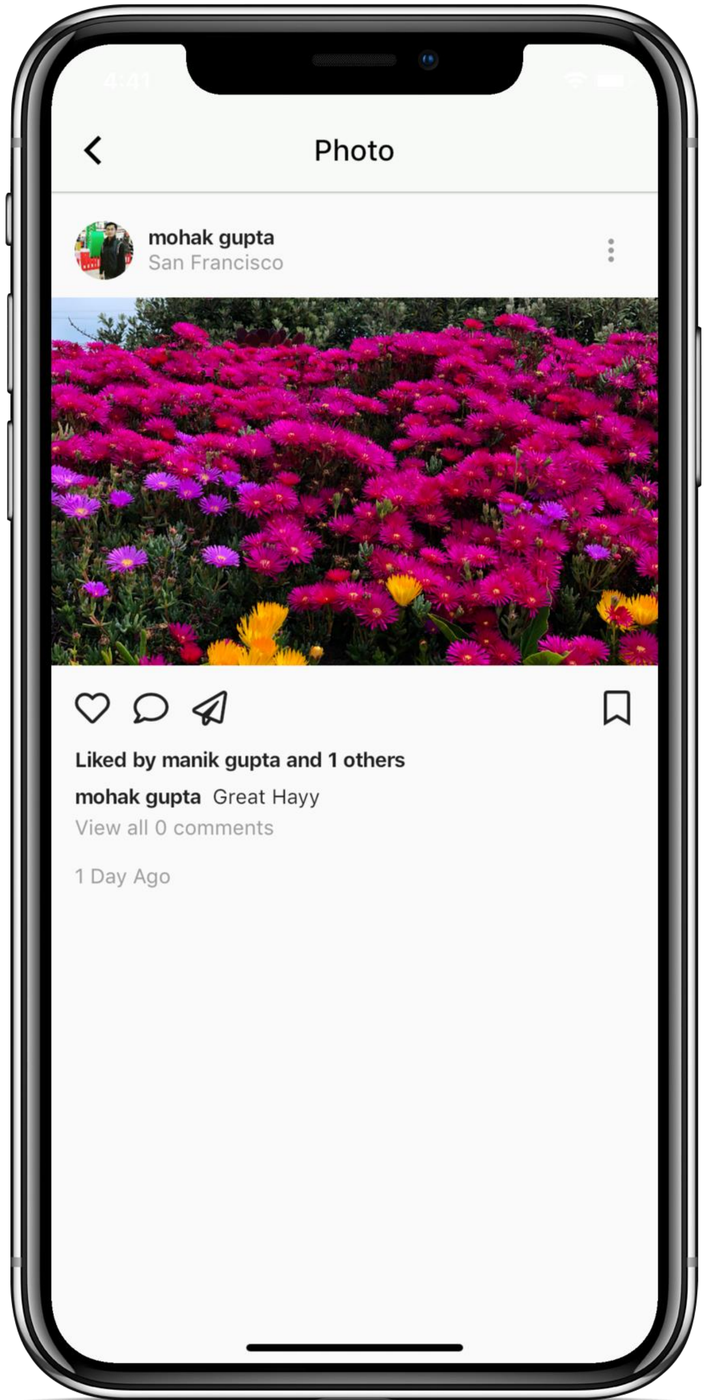
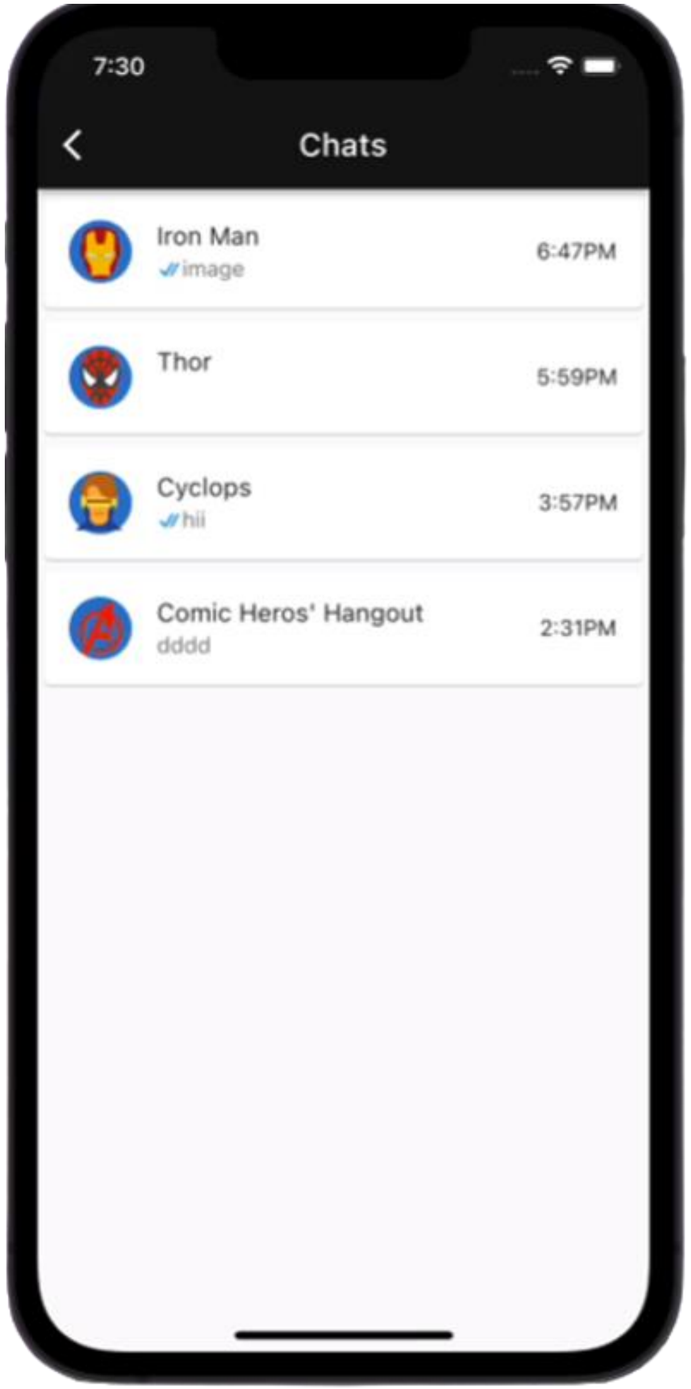


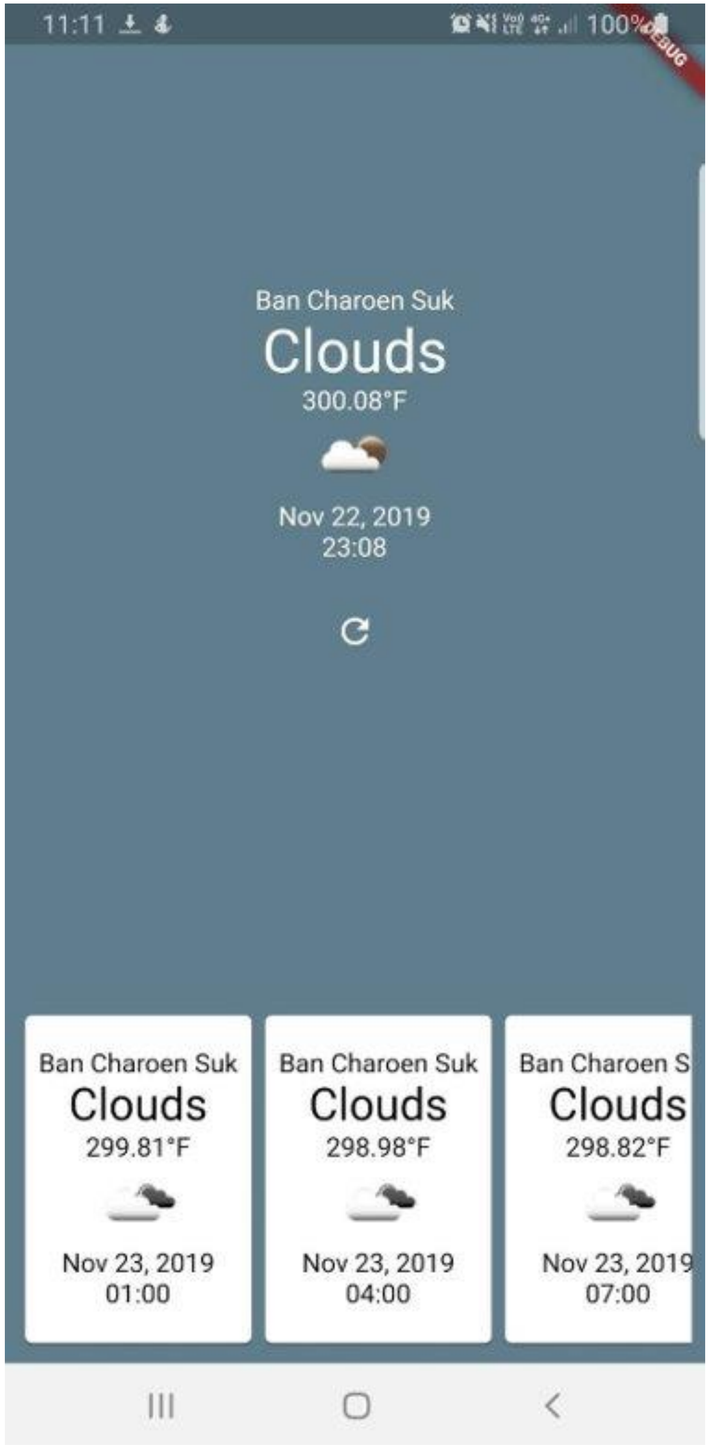
Photo
mohak gupta
San Francisco
Liked by manik gupta and 1 others
mohak gupta Great Hayy
View all 0 comments
1 Day Ago


7:30
Chats
Iron Man
image
6:47PM
Thor
5:59PM
Cyclops
hii
3:57PM
Comic Heros' Hangout
dddd
2:31PM

Hints - Helpers

Scope down

- Static
- Standard Fonts
- Image -> fill with any color
- Internet is Allow

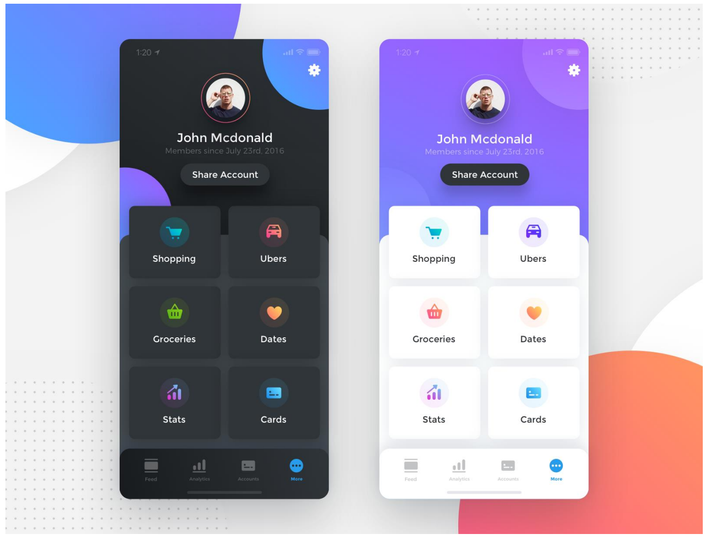

11:11
100%
DEBUG
Ban Charoen Suk
Clouds
300.08°F
Nov 22, 2019
23:08
Ban Charoen Suk
Clouds
299.81°F
Nov 23, 2019
01:00
Ban Charoen Suk
Clouds
298.98°F
Nov 23, 2019
04:00
Ban Charoen S
Clouds
298.82°F
Nov 23, 2019
07:00

Hints - Helpers

Scope down

- Static
- Standard Fonts
- Image -> fill with any color
- Internet is Allow

Key take away from this checkpoint



Further Study

- https://m3.material.io
- ~~Atomic Design | Brad Frost~~

Summaries

Now you know

- How to install/setup flutter SDK in your local machine
- Basic flutter command
- Fundamental of flutter components
- Dart Lang*
- Flutter Widgets
- Design system in general/ flutter~~/ atomic design~~

Homework

- Create GitHub account – individual -> studentID_groupName
- And upload all the checkpoint code to GitHub
  - mini_checkpoint_x
  - checkpoint_x
- And submit git username -> I'll post the form in Discord Channel
- I'll upload the code within this Friday -> so everyone can review
- I'll send the additional information -> in Discord Channel