

## CROSS-PLATFORM MOBILE APP DEVELOPMENT

(503107)

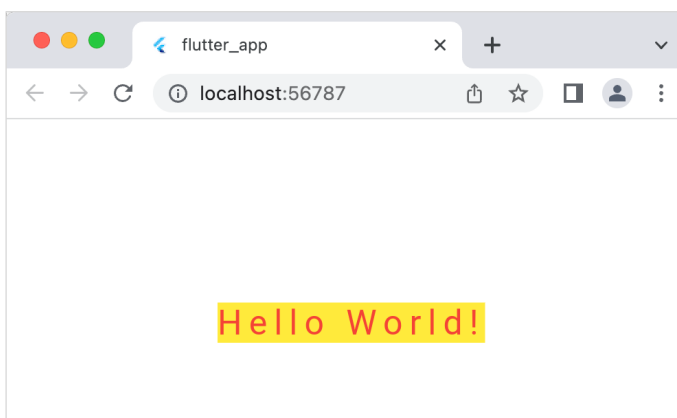
### LAB 1

## EXERCISE 1

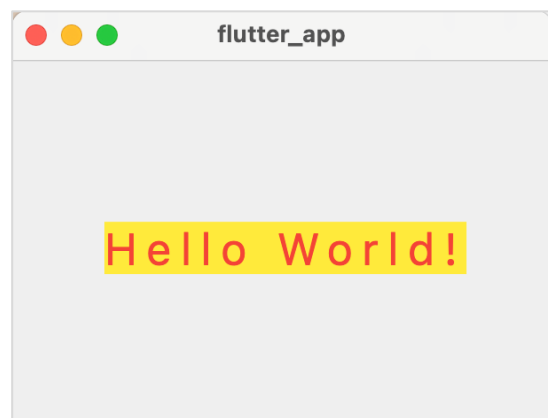
Create a simple Flutter application that displays a **Text** widget in the center of the screen (in combination with the **Center** widget) with the following settings:

- Text direction: left to right
- Text color: red
- Background color: yellow
- Size: 25
- Spacing between letters: 5.0

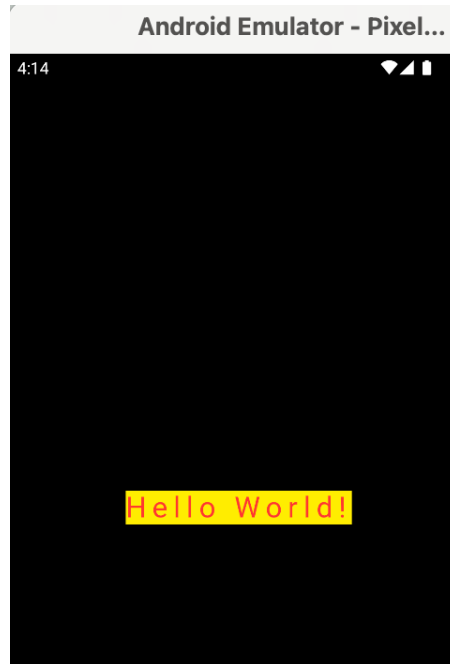
Since the app only uses the Text widget, when running on a mobile device, the app will display a black background. Some sample images of the application are given bellow:



*Example on Chrome*



*Example on macOS*



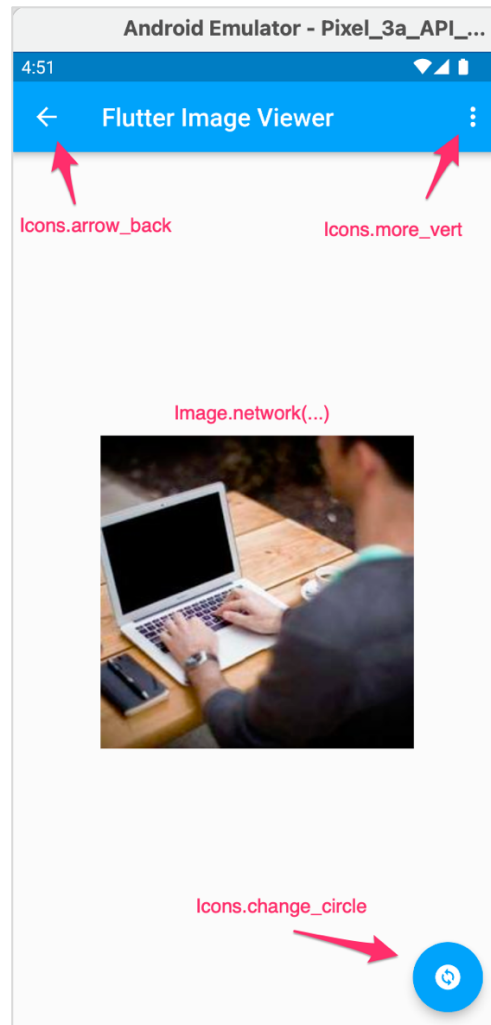
*Example on Android device*

Run your app on at least **3 different devices** (e.g. Windows, Chrome, Android) and **take screenshots**. Try **adjusting** the 'Hello World' value to the new value and save the source code to test the **Hot Reloading** feature. For Hot Reloading to work, you need to place UI components inside a custom **Stateless widget class**. Do not put Text widget directly in runApp() function.

Flutter's hot reload feature helps you quickly and easily experiment, build UIs, add features, and fix bugs. Hot reload works by injecting updated source code files into the running Dart Virtual Machine (VM). After the VM updates classes with the new versions of fields and functions, the Flutter framework automatically rebuilds the widget tree, allowing you to quickly view the effects of your changes.

## EXERCISE 2

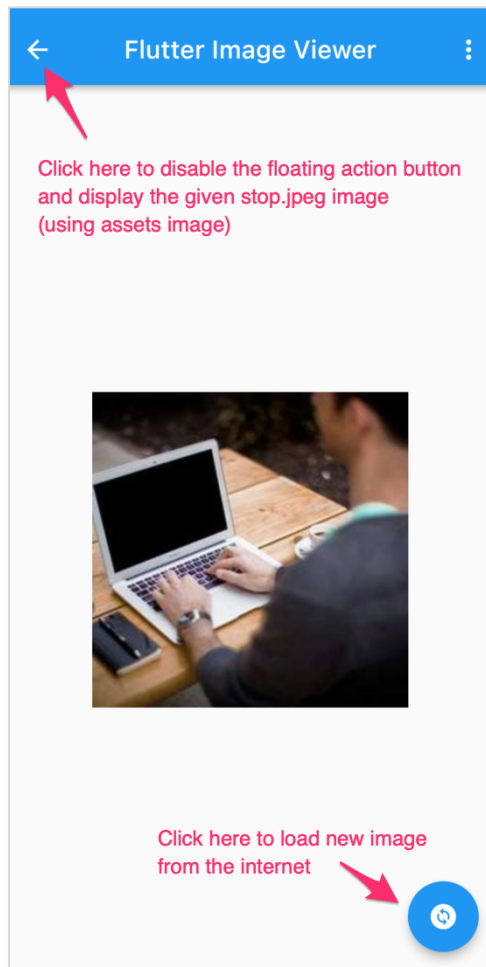
Create a Flutter application with an interface similar to the image below using **Stateless Widget**. The app uses **Material Design** with an **AppBar** and a **FloatingActionButton** and an **Image** widget centered on the screen. Images are downloaded from the internet from any url, for example you could use the following url (replace **1** with another value to change the image if desired)



<https://picsum.photos/250?image=1>

## EXERCISE 3

Continuing to work on the previous exercise, convert the application from Stateless Widget to **Stateful Widget** and write the code so that every time the floating action button is clicked, a new image will be displayed. When the back button is clicked, disable floating action button (not clickable anymore).



*Before clicking the back button*

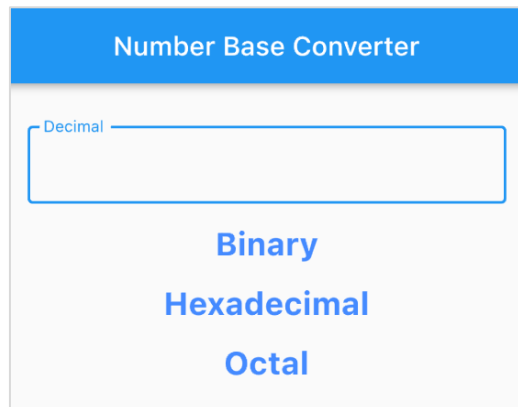


*After clicking the back button*

After clicking the back button, the floating action button cannot be clicked anymore. The application will then display the image given stop.jpeg as local assets.

## EXERCISE 4

Create an application that allows to convert base 10 to binary, octal and hexadecimal base systems. Every time you enter any number into the TextField widget, the corresponding values will be updated in the Text widgets below. Hava a look at the given animated gif illustration to better understand the requirements of the exercise.



Number Base Converter

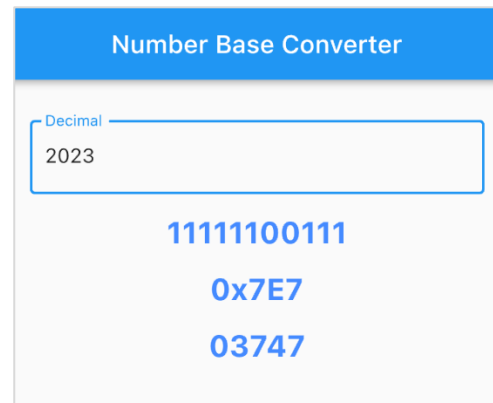
Decimal

Binary

Hexadecimal

Octal

*Before typing in the TextField*



Number Base Converter

Decimal  
2023

11111100111

0x7E7

03747

*After typing in the TextField*

On mobile devices, make sure that only numeric values are allowed in the textfield. The software keyboard then only displays the numeric keys, not the full keyboard with the alphabet.

#### Hint:

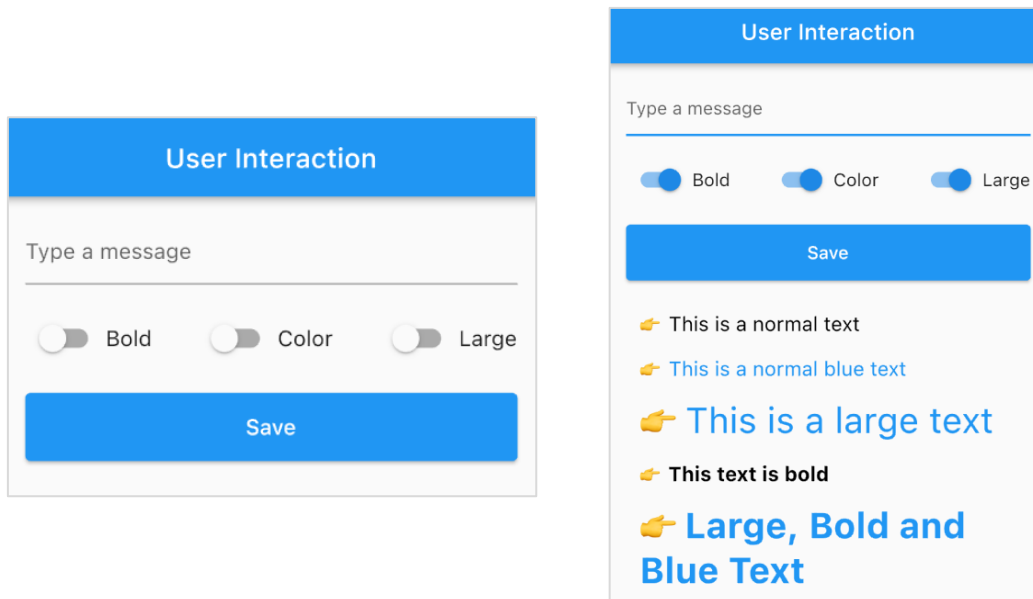
- Use the **units\_converter** package to perform conversion functions between radix systems. See more at: [https://pub.dev/packages/units\\_converter](https://pub.dev/packages/units_converter)
- Use `InputDecoration()` with `border = OutlineInputBorder()` property for the TextField's **decoration** property to add a border around it.
- Set up the TextField's **onChange()** event to update the numeric values every time the user adds or removes a character in the textfield.
- Use the **keyboardType** property to set the display mode of the soft keyboard.

## EXERCISE 5

Use widgets like **TextField**, **Switch**, **ElevatedButton**, **Text** and layout widgets to design UI as shown below. When the save button is clicked, the following requirements need to be fulfilled:

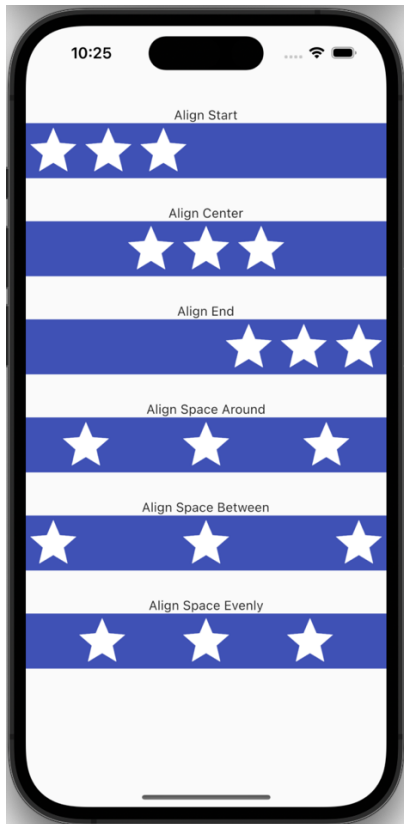
- The text in the TextField will be added to the listview below the button. TextField validation is required before adding to the list, displaying a snackbar message if the textfield is blank.

- If the corresponding switches are enabled, the text will be styled accordingly.
- Once done, the textfield should be automatically focused again.
- Set the onSubmitted property for the textfield so that when the user hits the enter (return) key, it will have the same effect as clicking the button.

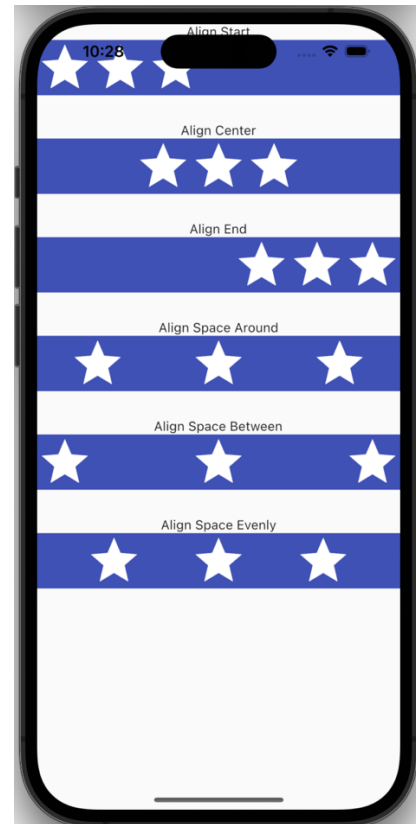


## EXERCISE 6

Use the **Row** and **Column** widget to design an application with an interface similar to the one below. To set the background color for the Row and Column widget, we can enclose it with a **Container** widget and then set the color property for that container. Use **SafeArea** widgets to ensure widgets are in the visible area of the device, avoiding the case where widgets are partially or completely obscured by elements like the iPhone's notch.



*Meet the requirement*



*Does not meet the requirement*