Tutorial 4.1
Stateful Widgets

Overview

- This is an ungraded activtiy.
- The goal of this acitivty is to apply what you have learned from the active learning activity.
- You will be doing online collaborative coding with your Home Group to implement about the topic “Stateful widgets”.

Preparation for the Activity

- Download the zip file and extract it to a folder.
- You should get the following materials:
    a. A codebase Flutter project
    b. A video about the expected result (expected_result.mp4)
    c. The question script (question.pdf)
- Open the the tutorial folder into VS Code
- Inside VS Code, open an integrated git Bash terminal. You can also open a git bash terminal externally.

Git Commands

The followings are some git commands you may need to use in this exercise.

Initializing Git

- Go to the exercise folder (using Git Bash)
- Type the following command to initialize the folder with git

```
git init
```

Committing Changes

- Type the following command to commit (or to record) any changes you have made. This will add new commit to the git repo.

```
git add .
git commit -a -m "a good commit message"
```

- If you need to update the recent commit, type

```
git add .
git commit --amend
```

- Create your first commit containing your room number and the room members. Invoke the following command on git bash.

```
git commit --verbose
```

It will open a new file COMMIT_EDITMSG. Enter the followings into the file:

```
Tutorial Group Information, Room <Number>

Member 1
Member 2
Member 3
Member 4
```

The Problem

In this tutorial you will be implementing a state management technique using Stateful Widgets. Develop a mobile application using Flutter Framework that shows an icon the screen. The user is then allowed to change the **color**, **size** and **roundness** of the icon using some button and slider widgets. Figure 1 illustrates some details about the User Interface of the application.

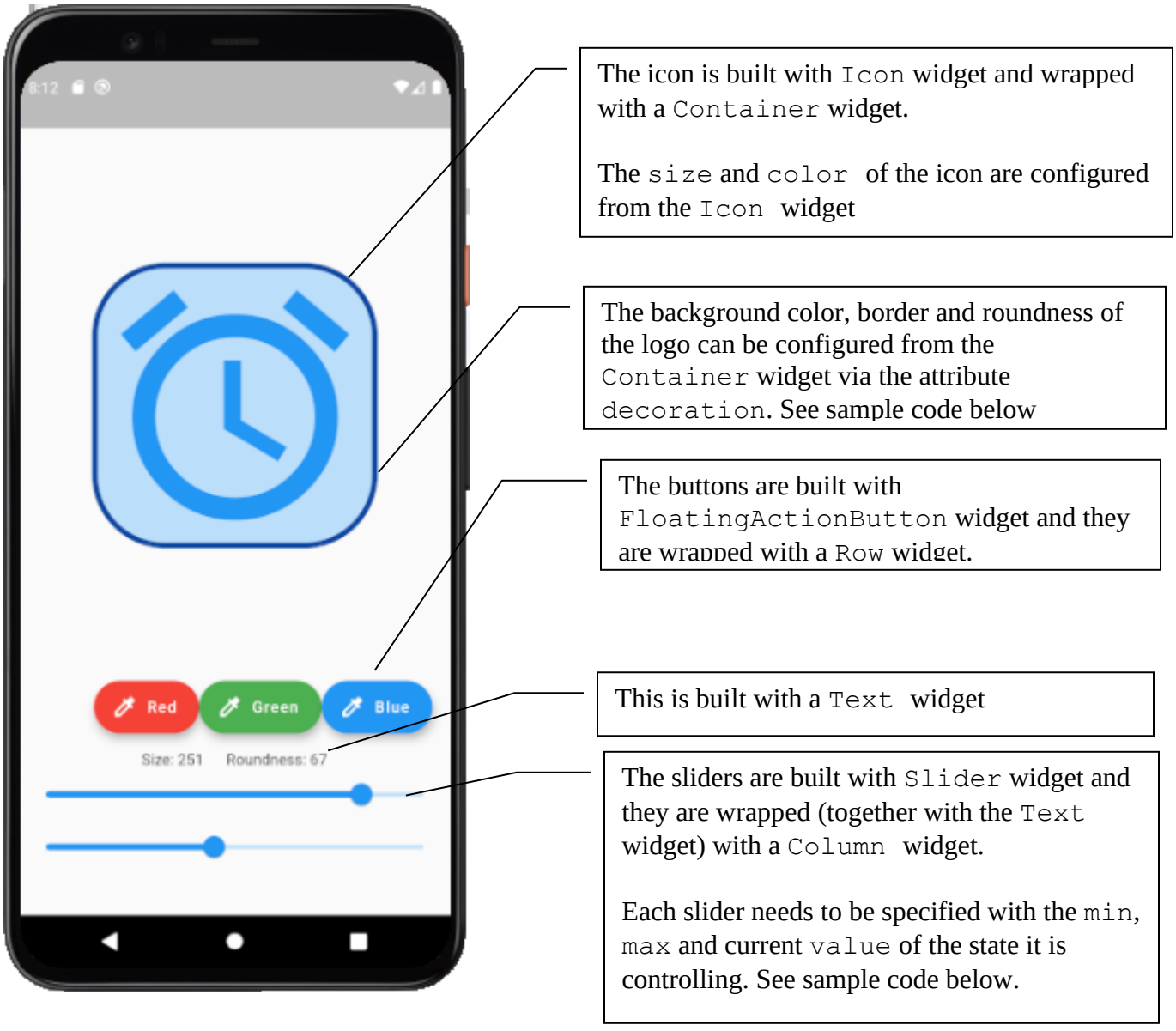

Figure 1: Details of the UI built

Sample Code

1. Configuring the background color, border and roundness of the container.

```
Container(
  decoration: BoxDecoration(
    color: Colors.blue[100],
    border: Border.all(width: 5.0, color: Colors.blue[900]),
    borderRadius: BorderRadius.all(
      Radius.circular(50.0),
    ),
  ), //BoxDecoration
  child: ….
  ),
), //Container
```

In the above sample code, the colors for the background and border are fixed to blue. However, in your code you need make it dynamic. Note also that using array indices to the colors means that we want to choose certain shades for the particular color. `Colors.blue [100]` is a lighter blue and `Colors.blue [900]` is a darker blue

2. Creating a slider

```
// Slider for controlling the size of the logo
Slider(
  value: 200.0,
  min: 0.0,
  max: 300.0,
  onChanged: (newValue){….},
) //Slider
```

In the above sample code, the current value of the logo size is fixed to 200. However, in your code you need make it dynamic that reflects the changes made by user interactions .

Refactoring

Once you have done with the basic requirements above, you will need to do refactoring on your code. You will refactor your code within the same source code fille. No need to use separate files.

1. Refactor the logo to a method or function.
2. You may notice that the three buttons share some commonility. Thus, it is a good idea to do refactoring on them. Do that by creating a new widget class.
3. You may also notice that the two sliders share some commonility. Thus, do refactoring using a new widget class.

File Structure

You can further refactor the code into separate files. Below is an example file structure that you may use for this project.