# *W5 PRACTICE*

## *STATELESS WIDGETS*

### *Learning objectives*

- ✓ Manipulate **Column** layout with **stretch** alignment
- ✓ Use **Icons**, **Image** and **Card** widgets
- ✓ Use **Enums** with attributes to specify a data model
- ✓ Create re-usable **StatelessWidget**
- ✓ Manage **required** and **optional** widget properties

### *No AI tools allowed to solve this practice*

### *How to submit?*

- ✓ **Push** your final code on **your GitHub repository**
- ✓ Then **attach the GitHub path** to the MS Team assignment and **turn it in**

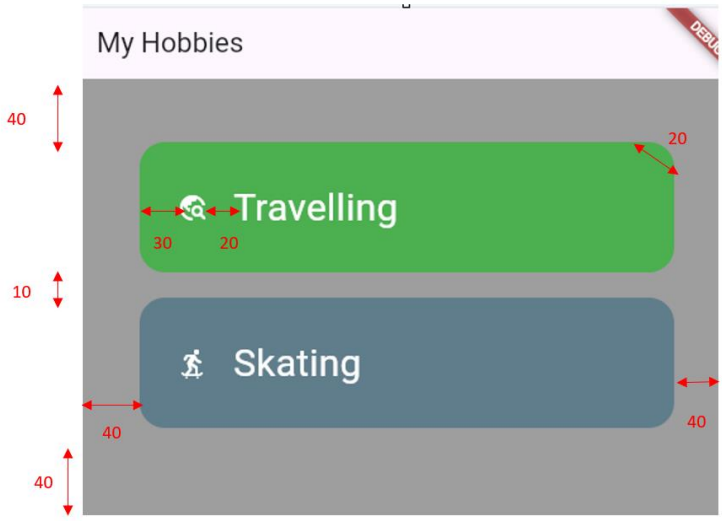
### *Before practice, to be prepared!*

*Read the following documentation to be ready for this practice:*

https://www.classcentral.com/classroom/youtube-flutter-tutorial-for-beginners-45851/60c82bddaba3e
https://www.classcentral.com/classroom/youtube-flutter-tutorial-for-beginners-45851/60c82bddaba15
https://www.youtube.com/watch?v=GPoRjSjd1cI
https://api.flutter.dev/flutter/material/Card-class.html
https://www.classcentral.com/classroom/youtube-flutter-tutorial-for-beginners-45851/60c82bddaba0a
https://api.flutter.dev/flutter/widgets/Image-class.html

# *EX 1 – The hobbies*

In this exercise, you need to arrange **hobbies cards** vertically, each containing a hobby with an icon and text.

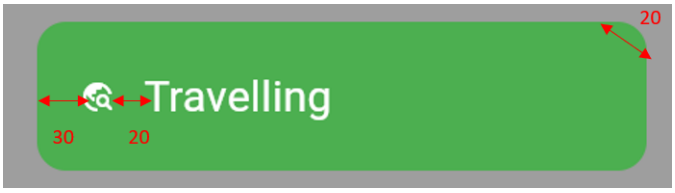


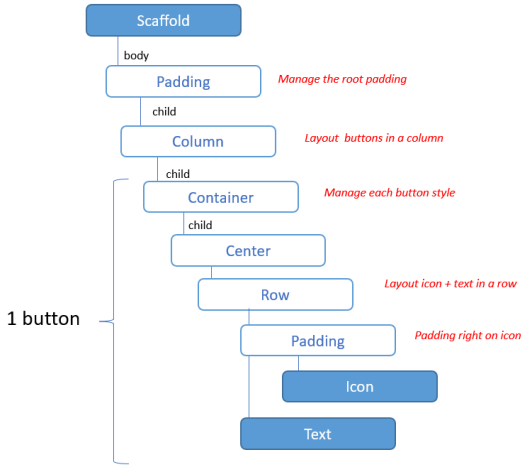

This exercise has 2 parts:

- **Part 1:** create a single hobby card
- **Part 2:** extract the card into a stateless widget

## PART 1 – Build the UI

Start by create a single hobby card






- We use `CrossAxisAlignment.stretch` on Column, so that the children to take up the entire width of the parent. *More about Colum here*

What is the difference between `crossAxisAlignment` and `mainAxisAlignment` in Column widget? Try it out to understand.

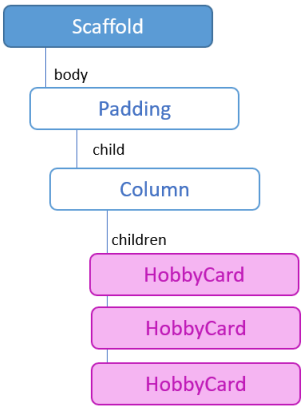
- We use predefined icons from the **Icons class**. *More about icons here*

- The button radius is performed using a **BoxDecoration** on the container. More about box decoration here

## PART 2 – Create a HobbyCard widget

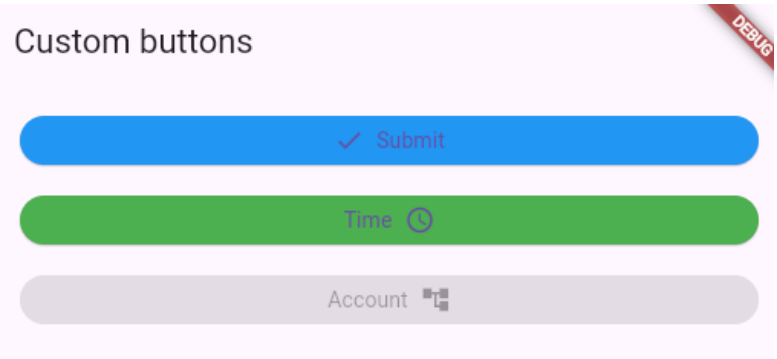
Extract the hobby card into a stateless widget called **HobbyCard** that takes the following parameters:

- The hobby title (String, *required*)
- The hobby icon (IconData, *required*)
- *The card background color (Color, optional, default value = BLUE)*



# *EX 2– The buttons*

In this exercise, you need to build a Custom Button

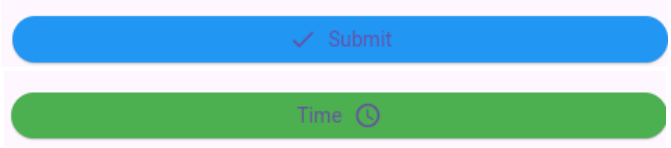



The button can have 3 **types**. Each type has a **specific color**:

| Button type | Color |
|---|---|
| Primary | Blue |
| Secondary | green |
| Disabled | grey |

*Note: we recommend managing this requirement with a dedicated enum!*

The button **icon** can be **either** before or after the button **label**:



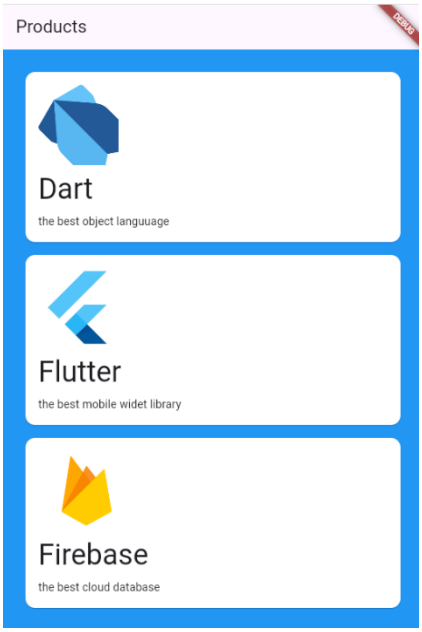

*Note: we recommend managing this requirement with a dedicated enum!*

The custom button must be a stateless widget called **CustomButton** that takes the following parameters:

- The button **label** (String, *required*)
- The button **icon** (IconData, *required)*
- The icon position (*left or right*, *optional, by default left*)
- The button type (*primary, secondary, disabled*, *optional, by primary)*

# *EX 3 – The products*

In this exercise, you need to arrange **product cards** vertically, each containing a product image, title and description.

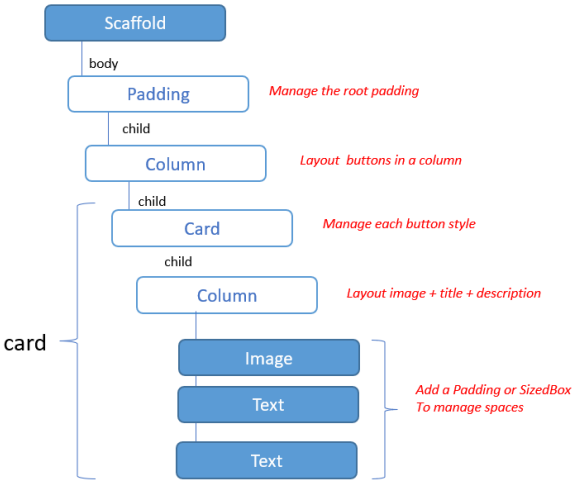



This exercise has 3 parts:

- **Part 1:** create a single hobby card
- **Part 2:** create a **Enum** gathering each product information
- **Part 2:** extract the card into a stateless widget

## PART 1 – Build the UI

Start by create a single card



- We use Card widget, to style the card. *More information about card here*

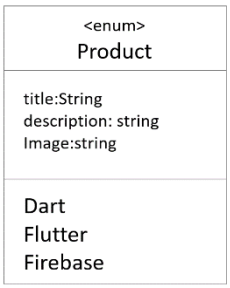
- We use Image widget, to display images. More information here and here.
    - *We will store images in asset folder*
    - *The 3 images are provided in the ZIP folder*

## PART 2 – Create a Enum product

To store our product, we are using enum.TO learn more about enum: here

Enums are a **special kind of class** used to represent **fixed number of constant values**.
*Why enum? Because our products are always the same and will not change over time*

Enum can also have attributes and constructors, and this is what we are doing right now.

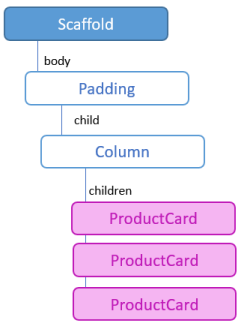



- Start to create a **Product enum**, following the above UML.
- The enum has 3 values: dart, flutter, firebase.

## PART 3 – Create a Product widget

Extract the card into a stateless widget called **ProductCard** that takes the following parameters (named):

- The product (type: Product, required)



# *EX 4 – The weather Forecast*

Recreate the **given weather app** mockup or create your **own weather app**!


### ✅ Mandatory parts

- ✓ Each card represents a city weather
- ✓ Cards must be rounded with **gradients** and **shadow** (you will need the `PhysicalModel` widget)
- ✓ Card must contain the min, max, current temperature of the city and a `CircleAvatar` with the weather type
- ✓ Ensure good **layout alignment** and spacing using `Rows/Columns`.

### 🌟 Optional parts (to explore)

- ✓ **Decorative oval** on the right side (you will need the `Stack` widget)
- ✓ Display multiple cards in a **scrollable view** (you will need the `ListView` widget)

### 💭 Hint topics

PhysicalModel
https://api.flutter.dev/flutter/widgets/PhysicalModel-class.html

Stack
https://api.flutter.dev/flutter/widgets/Stack-class.html

CircleAvatar
https://api.flutter.dev/flutter/material/CircleAvatar-class.html

Positioned
https://api.flutter.dev/flutter/widgets/Positioned-class.html

💭 **Do you need weather images?**

You can download the image ZIP file here.

You will then need to put image on /assets folder and update the pub spec file.
More information here and here.