Lab 5: Flutter Widget Practice Sheet

by: Mukhammadali Khayotov

September 22, 2025

1. The ‘Container‘ Widget

The **Container** is a fundamental building block used to apply styling like color, shape, or size constraints to its child widget.

Sample Usage

```
Container(
  padding: const EdgeInsets.all(16.0),
  color: Colors.blue,
  height: 100,
  width: 100,
  child: const Text('Hello!'),
)
```

Practice Tasks

1. **(Easy)** Change the ‘color‘ to ‘Colors.redAccent‘ and the ‘height‘ to ‘150‘.
2. **(Easy)** Replace the ‘Text‘ widget with an ‘Icon‘ widget, for example, ‘Icon(Icons.home)‘.
3. **(Medium)** Use the ‘decoration‘ property to give the ‘Container‘ rounded corners with a ‘BoxDecoration‘. *Hint: When using ‘decoration‘, move the ‘color‘ property inside the ‘BoxDecoration‘.*
4. **(Medium)** Add a ‘margin‘ around the ‘Container‘ using ‘margin: const EdgeInsets.all(20.0)‘. Observe how this differs from ‘padding‘.

2. The ‘Column‘ & ‘Row‘ Widgets

These are essential layout widgets. A **Column** arranges its children vertically, while a **Row** arranges them horizontally.

Sample Usage

```
Column(
  children: const <Widget>[
    Icon(Icons.star, size: 50),
    Icon(Icons.star, size: 50),
    Icon(Icons.star, size: 50),
  ],
)
```

Practice Tasks

1. **(Easy)** Add two more children to the ‘Column‘.
2. **(Easy)** Change the ‘Column‘ widget to a ‘Row‘ and observe the change.
3. **(Medium)** In your ‘Row‘, use ‘mainAxisAlignment: MainAxisAlignment.spaceEvenly‘ to distribute the children evenly.
4. **(Medium)** In your ‘Column‘, use ‘crossAxisAlignment: CrossAxisAlignment.start‘ to align all children to the left edge.

3. The ‘TextField‘ Widget

The **TextField** widget allows you to collect text input from the user via a hardware or on-screen keyboard.

Sample Usage

A basic ‘TextField‘ with placeholder text.

```
const TextField(
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    labelText: 'Enter your name',
  ),
)
```

Practice Tasks

1. **(Easy)** Change the ‘labelText‘ to "Enter your password".
2. **(Easy)** Add the property ‘obscureText: true‘ to the ‘TextField‘ to hide the input, which is common for password fields.
3. **(Medium)** Add an ‘Icon‘ to the ‘decoration‘. Use the ‘icon‘ property inside ‘InputDecoration‘ (e.g., ‘icon: Icon(Icons.person)‘).
4. **(Medium)** To read the value from a ‘TextField‘, you need a ‘TextEditingController‘. Create a controller and assign it to the ‘controller‘ property of the ‘TextField‘.

4. Advanced Layout (‘Expanded‘, ‘Flex‘, ‘Spacer‘)

While ‘Row‘ and ‘Column‘ are great for basic layouts, sometimes you need more control over how space is distributed. That’s where "flex" widgets come in.

The ‘Expanded‘ Widget

The **Expanded** widget forces its child to fill all available space along the main axis of a ‘Row‘ or ‘Column‘.

Sample Usage

In this ‘Row‘, the second ‘Container‘ is wrapped in ‘Expanded‘, so it takes up all the remaining horizontal space.

```
Row(
  children: <Widget>[
    Container(width: 100, height: 100, color: Colors.red),
    Expanded(
      child: Container(height: 100, color: Colors.green),
    ),
  ],
)
```

Practice Tasks

1. **(Easy)** In the sample, wrap the first (red) ‘Container‘ in an ‘Expanded‘ widget as well. What happens?
2. **(Medium)** Use the ‘flex‘ property. Give the red container ‘flex: 1‘ and the green container ‘flex: 2‘. The green container should now be twice as wide as the red one.
3. **(Medium)** Replace the ‘Row‘ with a ‘Column‘. How does the behavior of ‘Expanded‘ change?
4. **(Medium)** The ‘Spacer‘ widget is a simpler alternative to ‘Expanded‘. It creates an empty, flexible space. Create a ‘Row‘ with two ‘Container‘s and place a ‘Spacer()‘ between them to push them to opposite ends.

5. Widget Tree Challenges

Real UIs are made by nesting widgets inside other widgets. These challenges will test your ability to combine everything you've learned to build a structured "widget tree".

Challenge 1: Build a User Profile Card (Medium)

Create a card that displays user information. This is a very common UI pattern. Don't worry about perfect styling, just focus on the layout structure.

Your Goal: Create a widget that looks roughly like this:

```
+---------------------------------------------+
|  ---                                        |
| | A |  User Name                            |
|  ---   @username                            |
|                                             |
|  A short bio about the user goes here...    |
|                                             |
|  +----------+       +----------+            |
|  | Follow   |       | Message  |            |
|  +----------+       +----------+            |
+---------------------------------------------+
```

Required Widgets & Structure:

- A main ‘Container‘ with padding, rounded corners, and a light grey color.
- The child of the ‘Container‘ should be a ‘Column‘.
- The first item in the ‘Column‘ is a ‘Row‘.
- This ‘Row‘ should contain:
  - A ‘CircleAvatar‘ for the user's initial (’A’).
  - A ‘SizedBox‘ for spacing.
  - An ‘Expanded‘ widget containing another ‘Column‘ for the "User Name" and "@username" ‘Text‘ widgets. Use ‘CrossAxisAlignment.start‘ on this inner column.
- Add another ‘SizedBox‘ for vertical spacing.
- Add a ‘Text‘ widget for the bio.
- Add a final ‘Row‘ at the bottom with ‘MainAxisAlignment.spaceAround‘ containing two ‘ElevatedButton‘ widgets.

Challenge 2: Build a Product List Item (Medium)

Create a layout for a single product in a shopping list. This pattern combines a fixed-size image with flexible text content.

Your Goal: Create a widget that looks roughly like this:

```
+---------------------------------------------------------+
|  +-------+                                              |
|  |       |  Product Title                               |
|  | Image |                                              |
|  |       |  A brief description of the item...          |
|  +-------+                                              |
|          $99.99                                         |
+---------------------------------------------------------+
```

Required Widgets & Structure:

- The top-level widget should be a ‘Container‘ with some padding.

- Its child should be a ‘Row‘.
- The ‘Row‘ should contain:
  - An ‘Image.network()‘ widget (use a placeholder URL) inside a ‘SizedBox‘ to give it a fixed width and height (e.g., 100x100).
  - Another ‘SizedBox‘ for horizontal spacing.
  - An ‘Expanded‘ widget. Its child will be a ‘Column‘.
  - This inner ‘Column‘ should have ‘crossAxisAlignment: CrossAxisAlignment.start‘.
  - Inside this ‘Column‘, add three ‘Text‘ widgets for the title, description, and price. Style them with different font sizes and weights using ‘TextStyle‘.