# Lab 25

Instructor: Sidra Khatoon

E-mail: skhatoon[@uit.edu](mailto:ad@uit.edu)

# Objective

# The objective of lab is to explore animation in flutter

**Student Information**

|  |  |
| --- | --- |
| **Student Name** |  |
| **Student ID** |  |
| **Date** |  |

**Assessment**

|  |  |
| --- | --- |
| **Marks Obtained** |  |
| **Remarks** |  |
| **Signature** |  |

# Objective

# The objective of lab is to explore animation in flutter

# Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

# How to Submit?

Submit lab work using Teams.

**Introduction**

# At the most basic level, animations in a Flutter app can be seen as one of two types: drawing-based, and code-based animations. Drawing-based animations are animated graphics, vectors, characters, or anything that is “drawn” then animated. On the other hand, code-based animations are focused on widget layouts and styles (lists, colors, text, ..etc). Code-based animations in Flutter have 2 types: Implicit animations & Explicit animations. In each of these types, you can either use ready-to-use widgets, or create your own widgets. Let’s dive deeper into each type with some examples. In this lab we will explore implicit animation with ready to used widget. They are called AnimatedFoo widgets.

# AnimatedFoo Widgets

In this Widget Foo is animated property. Most of them are animated versions of the widgets you already know and use, like AnimatedContainer, AnimatedPadding, AnimatedPositioned, …etc.

# AnimatedContainer Widget:

In Flutter a container is a simple widget with well-defined properties like height, width, and color, etc. The AnimatedContainer widget is a simple container widget with animations. These types of widgets can be animated by altering the values of their properties which are the same as the Container widget. These types of animation in Flutter is known as ‘Implicit Animation. We will discuss then in detail in this article by building a simple app with AnimatedContainer widget.

# Constructor of AnimatedContainer class:

AnimatedContainer(

{Key key,

AlignmentGeometry alignment,

EdgeInsetsGeometry padding,

Color color,

Decoration decoration,

Decoration foregroundDecoration,

double width,

double height,

BoxConstraints constraints,

EdgeInsetsGeometry margin,

Matrix4 transform,

Widget child,

Curve curve: Curves.linear,

@required Duration duration,

VoidCallback onEnd}

)

**Properties of AnimatedContainer Widget:**

**alignment:** This property takes AlignmentGeometry class as the object. It controls the alignment of the child widget with the container.

child: This property holds a widget as the object to show inside the AnimatedContainer.

**constraints**: BoxConstraints class is the object to this property. It applies some extra constraints to the child widget in the AnimatedContainer.

decoration: This property takes in Decoration class as the object to apply color behind the child widget.

**foregroundDecoration:** This property controls the default color of the text inside the AnimatedContainer.

**margin:** The margin property holds EdgeInsetsGeometry class as the object. It adds empty space around the widget.

**padding:** This property also takes EdgeInsetsGeometry class as the object to add empty space inside the AnimatedContainer and the child widget.

**transform:** This property takes Matrix4 as the object to apply matrix transformation before painting the AnimatedContainer.

Follow the below steps to build an application with AnimatedContainer widget:

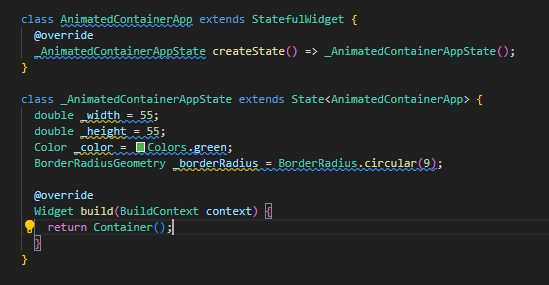
Create a StatefulWidget and define its properties.

Add an AnimatedContainer widget and define its properties.

Create animation by altering those properties.

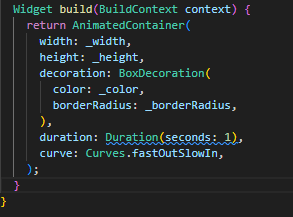
**Creating a StatefulWidget:**

Use the custom State class to create a StatefulWidget and define its properties as shown below:



**Adding AnimatedContainer widget:**

Add an AnimatedContainer widget with its duration property defined that determines how long the container is going to animate as shown below:



**Altering the properties:**

Rebuilding and changing the properties after the end of duration specified property is done as shown below:



**Assessment:**

Create a Flutter app that uses the AnimatedContainer widget to simulate a "shape-shifting" animation.

Requirements:

1. Display a container at the center of the screen.
2. Add a toggle button below the container to switch between two states:
   * State 1: The container is a small square (100x100 pixels) with a red color.
   * State 2: The container is a large circle (200x200 pixels) with a blue color.
3. Use the AnimatedContainer widget to smoothly transition between the two states over 800 milliseconds.
4. Ensure the animation includes changes to:

* Height and width.
* Border radius (to achieve the circle).
* Background color.