**Aim:** Adding user interactivity in the base components and customization of layout using functions provided by React Native and XAML.

**Description:**

This project focuses on enhancing user interaction and creating responsive layouts for mobile and cross-platform applications by leveraging the features of React Native and XAML. It combines modern tools and frameworks to build applications that are both user-friendly and visually appealing.

**Objective:**

* To implement interactivity in base components, such as buttons, text inputs, and dropdowns, using React Native and XAML.
* To design dynamic and responsive layouts tailored to user preferences using the layout capabilities of React Native and XAML.
* To utilize platform-specific functions for a seamless cross-platform experience.
* To build reusable and modular components for maintainable code architecture.

**Tools Required:**

* Node.js and React Native CLI for JavaScript and JSX (React Native development).
* Visual Studio 2022 with .NET MAUI for XAML and C# (cross-platform app development).
* Code editor: Visual Studio Code for JavaScript/JSX, Visual Studio 2022 for XAML and C#.
* Android Studio (for Android Emulator) or a physical device for testing mobile apps.

**Implementation:  
MainPage.xaml.cs**

using Microsoft.Maui.Controls;

using System;

using System.Threading.Tasks;

using Microsoft.Maui.Graphics;

namespace TemperatureConverterApp

{

    public partial class MainPage : ContentPage

    {

        public MainPage()

        {

            InitializeComponent();

        }

        private async Task ConvertButtonScaleAnimation()

        {

            // Scale up the button on click and revert it back

            await convertButton.ScaleTo(1.1, 100, Easing.CubicOut);

            await convertButton.ScaleTo(1.0, 100, Easing.CubicIn);

        }

        private async Task AnimateBackgroundColor(Color targetColor)

        {

            Color originalColor = this.BackgroundColor;

            uint duration = 500;

            // Define the color transition animation

            var animation = new Animation(v =>

            {

                // Interpolate each RGB component separately

                this.BackgroundColor = Color.FromRgba(

                    originalColor.Red + v \* (targetColor.Red - originalColor.Red),

                    originalColor.Green + v \* (targetColor.Green - originalColor.Green),

                    originalColor.Blue + v \* (targetColor.Blue - originalColor.Blue),

                    1 // Fully opaque

                );

            });

            // Commit the animation to apply the transition effect

            animation.Commit(this, "BackgroundColorAnimation", length: duration, easing: Easing.Linear);

            await Task.Delay((int)duration); // Ensures that the method waits for the animation to complete

        }

        private async void OnConvertClicked(object sender, EventArgs e)

        {

            // Animate the Convert button with a scale effect

            await ConvertButtonScaleAnimation();

            // Reset result label's opacity and background color before conversion

            resultLabel.Opacity = 0;

            resultLabel.TranslationX = -100; // Slide-in from the left

            if (double.TryParse(temperatureEntry.Text, out double temp))

            {

                string conversionType = conversionPicker.SelectedItem as string;

                double result;

                if (conversionType == "Celsius to Fahrenheit")

                {

                    result = (temp \* 9 / 5) + 32;

                    resultLabel.Text = $"{temp} °C = {result:F2} °F";

                }

                else if (conversionType == "Fahrenheit to Celsius")

                {

                    result = (temp - 32) \* 5 / 9;

                    resultLabel.Text = $"{temp} °F = {result:F2} °C";

                }

                else

                {

                    resultLabel.Text = "Please select a conversion type.";

                    return;

                }

                // Change color based on the converted temperature value

                if ((conversionType == "Celsius to Fahrenheit" && temp > 25) ||

                    (conversionType == "Fahrenheit to Celsius" && result > 25))

                {

                    resultLabel.TextColor = Colors.Red;

                    await AnimatePulseBackgroundColor(Color.FromRgb(255, 182, 193), Color.FromRgb(255, 100, 100)); // Pulse effect for high temperature

                }

                else

                {

                    resultLabel.TextColor = Colors.Green;

                    await AnimateBackgroundColor(Color.FromRgb(144, 238, 144)); // Smooth transition to light green

                }

                // Animate the label with slide-in, fade-in, and bounce effects

                await ShowResultLabelAnimation();

                await BounceResultLabelAnimation();

            }

            else

            {

                resultLabel.Text = "Invalid temperature input.";

                resultLabel.TextColor = Colors.Gray;

                await AnimateBackgroundColor(Color.FromRgb(211, 211, 211)); // LightGray equivalent for invalid input

                await ShakeEntryAnimation(temperatureEntry); // Shake animation for invalid input

                await ShowResultLabelAnimation();

            }

        }

        private void OnPickerSelectedIndexChanged(object sender, EventArgs e)

        {

            // Get the selected item from the picker

            var selectedConversion = conversionPicker.SelectedItem as string;

            // Optional: Display the selected item in the result label as feedback

            if (!string.IsNullOrEmpty(selectedConversion))

            {

                resultLabel.Text = $"Selected: {selectedConversion}";

            }

        }

        private async Task ShowResultLabelAnimation()

        {

            // Set initial properties

            resultLabel.Opacity = 0;

            resultLabel.TranslationX = -100;

            // Animate opacity and translation

            await Task.WhenAll(

                resultLabel.FadeTo(1, 750, Easing.CubicIn),

                resultLabel.TranslateTo(0, 0, 750, Easing.CubicOut)

            );

        }

        private async Task BounceResultLabelAnimation()

        {

            // Bounce effect for the result label

            await resultLabel.ScaleTo(1.1, 100, Easing.SpringIn);

            await resultLabel.ScaleTo(1.0, 100, Easing.SpringOut);

        }

        private async Task ShakeEntryAnimation(Entry entry)

        {

            // Shake effect for invalid entry input

            const int shakeDistance = 10;

            for (int i = 0; i < 3; i++)

            {

                await entry.TranslateTo(-shakeDistance, 0, 50);

                await entry.TranslateTo(shakeDistance, 0, 50);

            }

            await entry.TranslateTo(0, 0, 50);

        }

        private async Task AnimatePulseBackgroundColor(Color startColor, Color endColor)

        {

            uint duration = 500;

            var animation = new Animation(v =>

            {

                this.BackgroundColor = Color.FromRgba(

                    startColor.Red + v \* (endColor.Red - startColor.Red),

                    startColor.Green + v \* (endColor.Green - startColor.Green),

                    startColor.Blue + v \* (endColor.Blue - startColor.Blue),

                    1

                );

            });

            animation.Commit(this, "PulseAnimation", length: duration, easing: Easing.CubicInOut, repeat: () => true);

            await Task.Delay((int)(duration \* 2)); // Pulses twice

        }

    }

}

**MainPage.xaml**

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

             x:Class="TemperatureConverterApp.MainPage"

        BackgroundColor="#f5f5f5">

<VerticalStackLayout Padding="20" Spacing="25" VerticalOptions="Center">

        <!-- Title Label -->

        <Frame HasShadow="True" CornerRadius="10" Padding="15" BackgroundColor="#2196F3">

            <Label Text="Temperature Converter"

                   FontSize="26"

                   HorizontalOptions="Center"

                   TextColor="White"

                   FontAttributes="Bold" />

        </Frame>

        <!-- Temperature Input -->

        <Frame HasShadow="True" CornerRadius="10" Padding="10" BackgroundColor="White">

            <Entry x:Name="temperatureEntry"

                   Placeholder="Enter temperature"

                   Keyboard="Numeric"

                   FontSize="18"

                   TextColor="#333333" />

        </Frame>

        <!-- Conversion Type Picker -->

        <Frame HasShadow="True" CornerRadius="10" Padding="10" BackgroundColor="White">

            <Picker x:Name="conversionPicker"

                    Title="Select Conversion Type"

                    FontSize="18"

                    HorizontalOptions="FillAndExpand"

SelectedIndexChanged="OnPickerSelectedIndexChanged">

                <Picker.ItemsSource>

                    <x:Array Type="{x:Type x:String}">

                        <x:String>Celsius to Fahrenheit</x:String>

                        <x:String>Fahrenheit to Celsius</x:String>

                    </x:Array>

                </Picker.ItemsSource>

            </Picker>

        </Frame>

        <!-- Convert Button -->

       <Button x:Name="convertButton"

        Text="Convert"

        FontSize="18"

        BackgroundColor="#4CAF50"

        TextColor="White"

        CornerRadius="10"

        HeightRequest="50"

        Clicked="OnConvertClicked" />

        <!-- Result Label -->

        <Frame HasShadow="True" CornerRadius="10" Padding="15" BackgroundColor="White">

            <Label x:Name="resultLabel"

                   FontSize="20"

                   HorizontalOptions="Center"

                   TextColor="#333333"

                   FontAttributes="Bold"

                   HorizontalTextAlignment="Center"

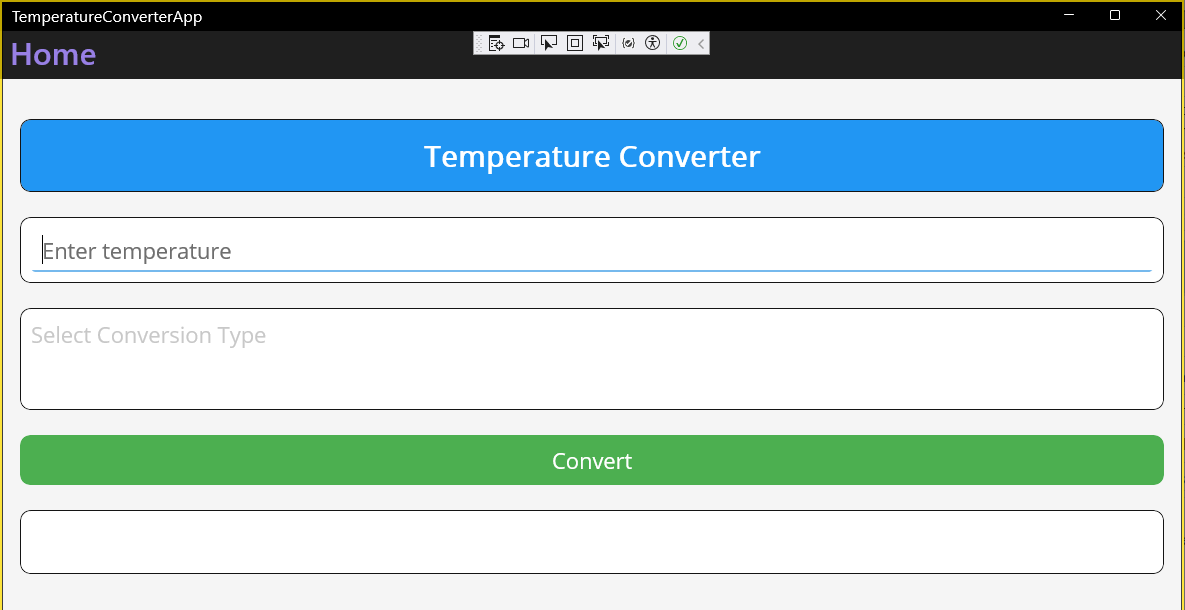
                   Opacity="0" /> <!-- Initial opacity set to 0 for fade-in animation -->

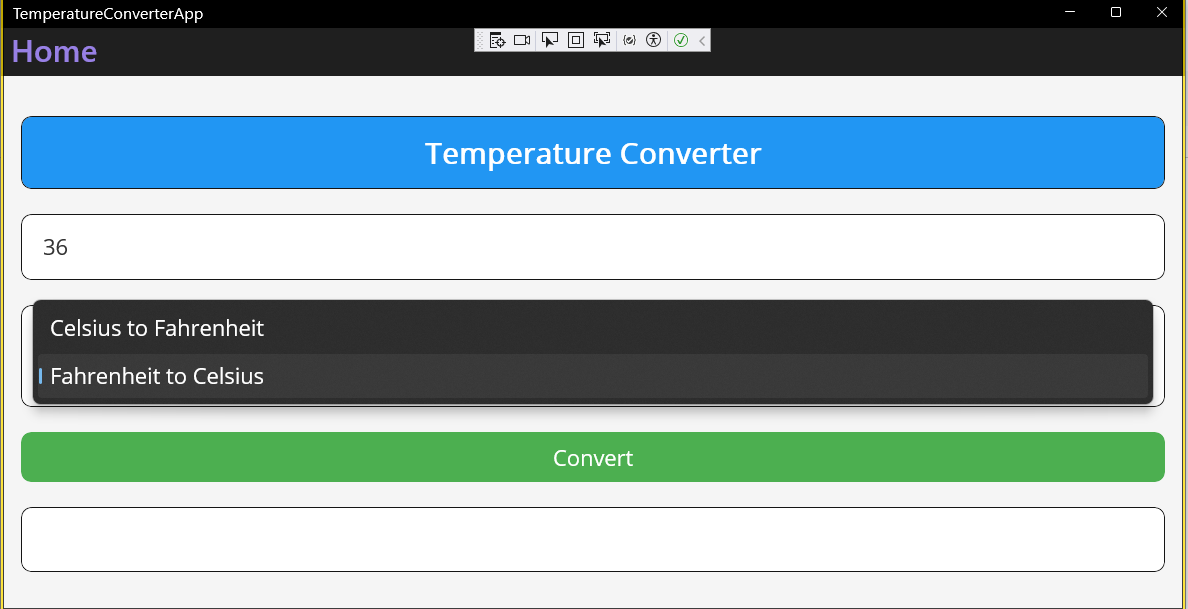
        </Frame>

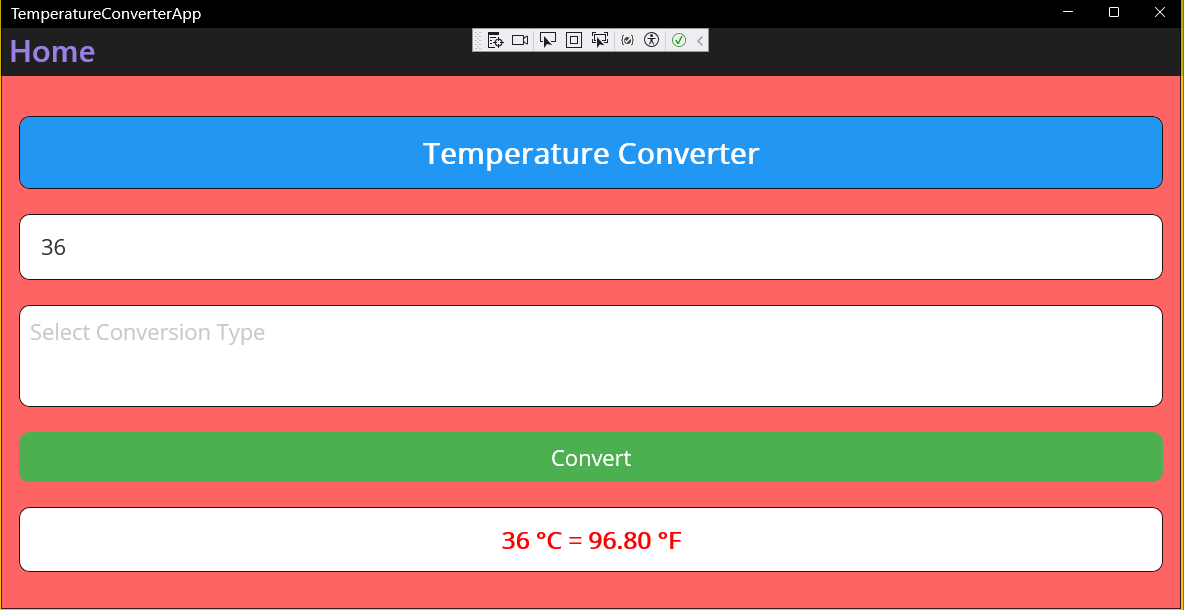
    </VerticalStackLayout>

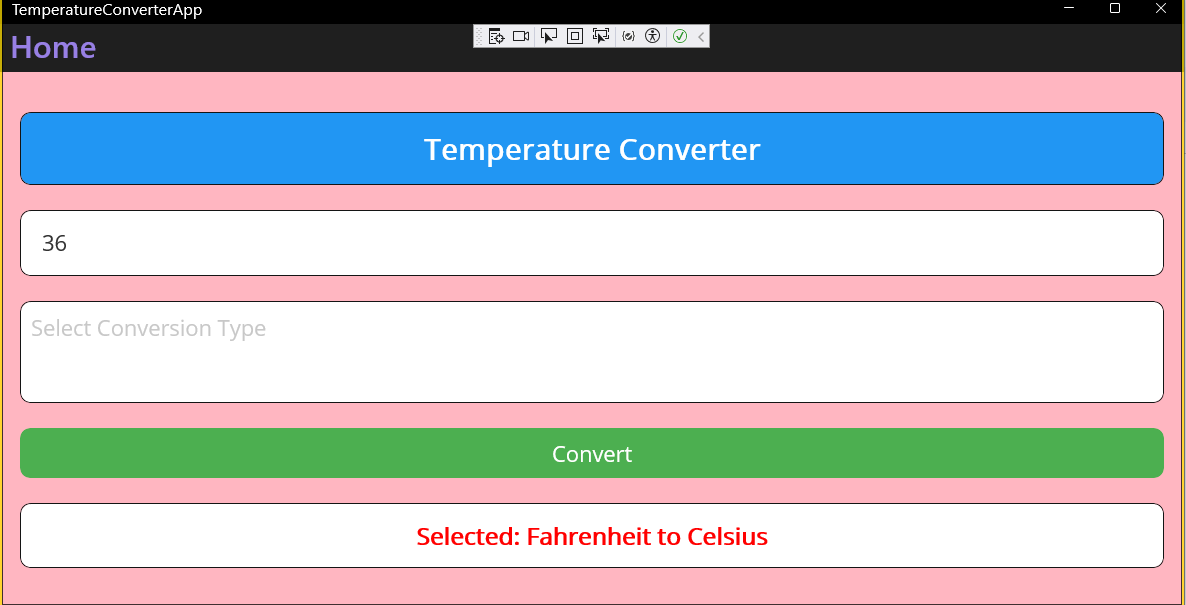
</ContentPage>

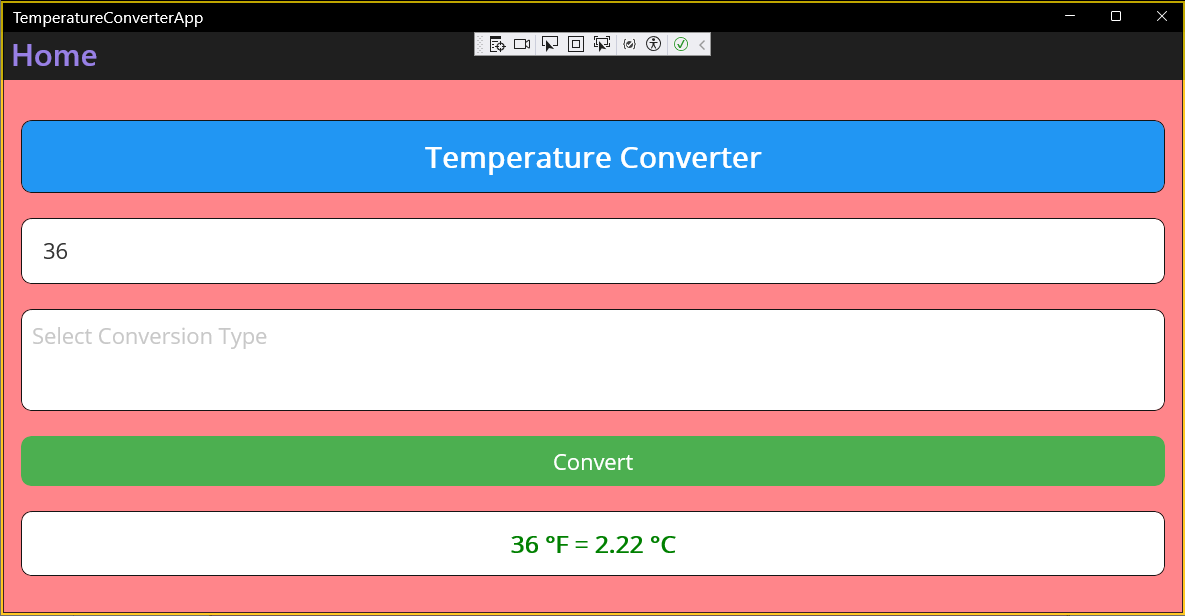
**Output Screens:**











**Conclusion:**

This project demonstrates how to create highly interactive and customizable user interfaces for cross-platform applications. By combining these frameworks, developers can build modern, responsive, and user-friendly applications suitable for a wide range of platforms and devices.