| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Practical Number:** 2 | |
| **Aim of Practical:**  Study different types of layouts and Toast class.  A) Design a Registration form to show the working of different layouts.  B) Create an application to design a simple calculator to perform addition, subtraction, multiplication, and division. Show message for divide by zero error using Toast.  C) Create an application for Unit Conversion. | | | |
| **DOP:** 9.10.23 | | **DOS:** 28.10.23 | |
| **CO Mapped:** - | **PO Mapped:** - | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 2

**Aim:** Study different types of layouts and Toast class.

A) Design a Registration form to show the working of different layouts.

B) Create an application to design a simple calculator to perform addition, subtraction, multiplication, and division. Show message for divide by zero error using Toast.

C) Create an application for Unit Conversion.

**Theory:**

A layout defines the structure for a user interface in your app, such as in an [activity](https://developer.android.com/guide/components/activities). All elements in the layout are built using a hierarchy of [View](https://developer.android.com/reference/android/view/View) and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) objects. A View usually draws something the user can see and interact with. A ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects, as shown in figure 1.

**Figure 1.** Illustration of a view hierarchy, which defines a UI layout.

View objects are often called *widgets* and can be one of many subclasses, such as [Button](https://developer.android.com/reference/android/widget/Button) or [TextView](https://developer.android.com/reference/android/widget/TextView). The ViewGroup objects are usually called *layouts* and can be one of many types that provide a different layout structure, such as [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout) or [ConstraintLayout](https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout).

You can declare a layout in two ways:

* **Declare UI elements in XML.** Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts. You can also use Android Studio's [Layout Editor](https://developer.android.com/studio/write/layout-editor) to build your XML layout using a drag-and-drop interface.
* **Instantiate layout elements at runtime.** Your app can create View and ViewGroup objects and manipulate their properties programmatically.

Declaring your UI in XML lets you separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easier to provide different layouts for different screen sizes and orientations. This is discussed further in [Support different screen sizes](https://developer.android.com/training/multiscreen/screensizes).

The Android framework gives you the flexibility to use either or both of these methods to build your app's UI. For example, you can declare your app's default layouts in XML, and then modify the layout at runtime.

**Code:**

File: MainActivity.java

package vesit.ajayk57.mc.practical2;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.widget.Button;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button runABtn=findViewById(R.id.runABtn);

Button runBBtn=findViewById(R.id.runBBtn);

Button runCBtn=findViewById(R.id.runCBtn);

runABtn.setOnClickListener(view -> startActivity(new Intent(this,RegistrationActivity.class)));

runBBtn.setOnClickListener(view -> startActivity(new Intent(this,CalculatorActivity.class)));

runCBtn.setOnClickListener(view -> startActivity(new Intent(this, UnitConvertorActivity.class)));

}

}

File: activity\_main.java

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:id="@+id/titleTxt"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="@string/practical\_no\_2"

android:textAlignment="center"

android:textSize="24sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.5"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.0" />

<TextView

android:id="@+id/runTxt"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="@string/run"

android:textSize="20sp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.5"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/titleTxt"

app:layout\_constraintVertical\_bias="0.19" />

<Button

android:id="@+id/runABtn"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="@string/a\_reg"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.5"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/runTxt"

app:layout\_constraintVertical\_bias="0.0" />

<Button

android:id="@+id/runBBtn"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="@string/b\_cal"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.5"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/runABtn"

app:layout\_constraintVertical\_bias="0.0" />

<Button

android:id="@+id/runCBtn"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="@string/c\_unit\_conv"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.5"

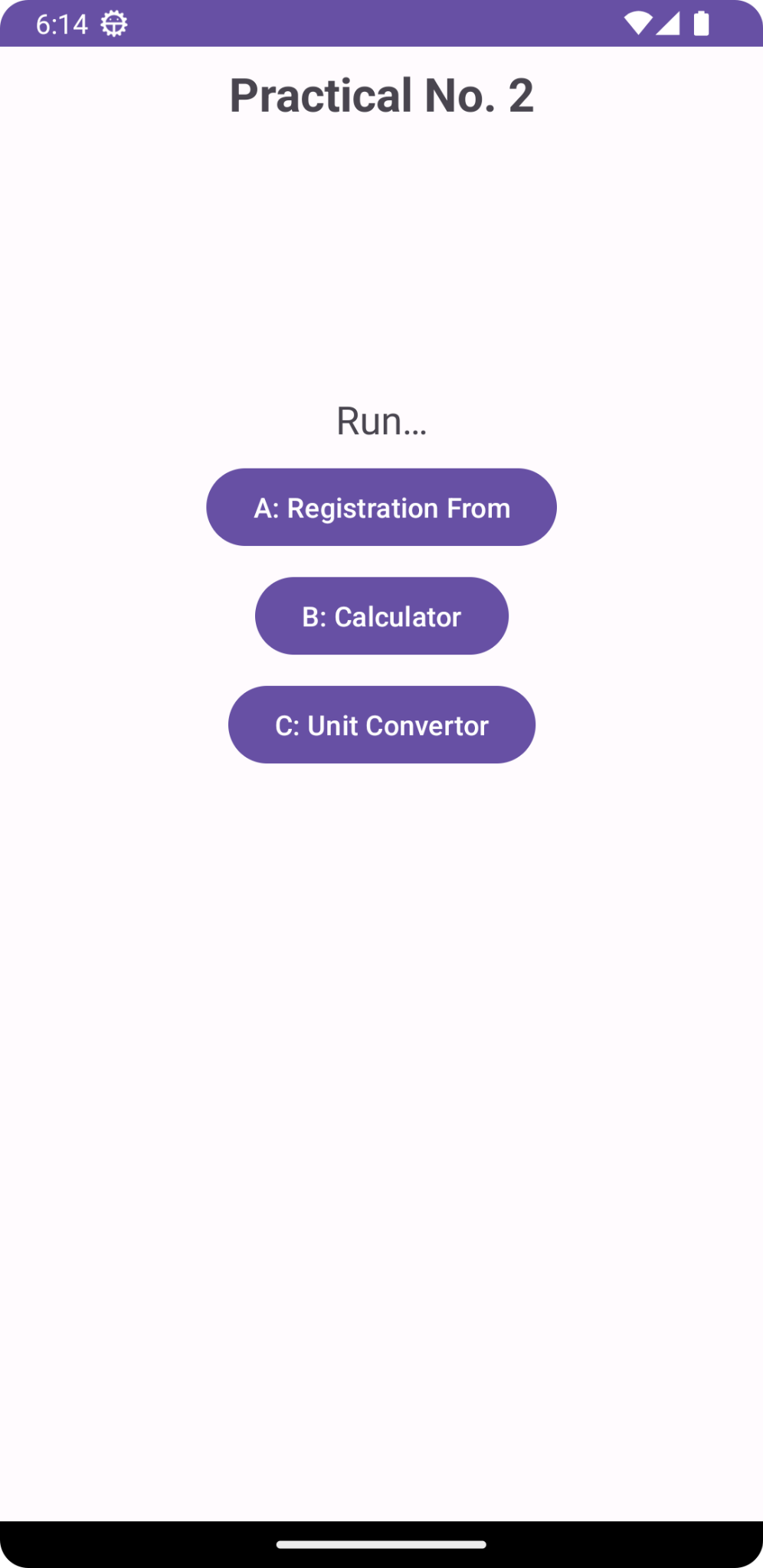
app:layout\_constraintStart\_toStartOf="parent"

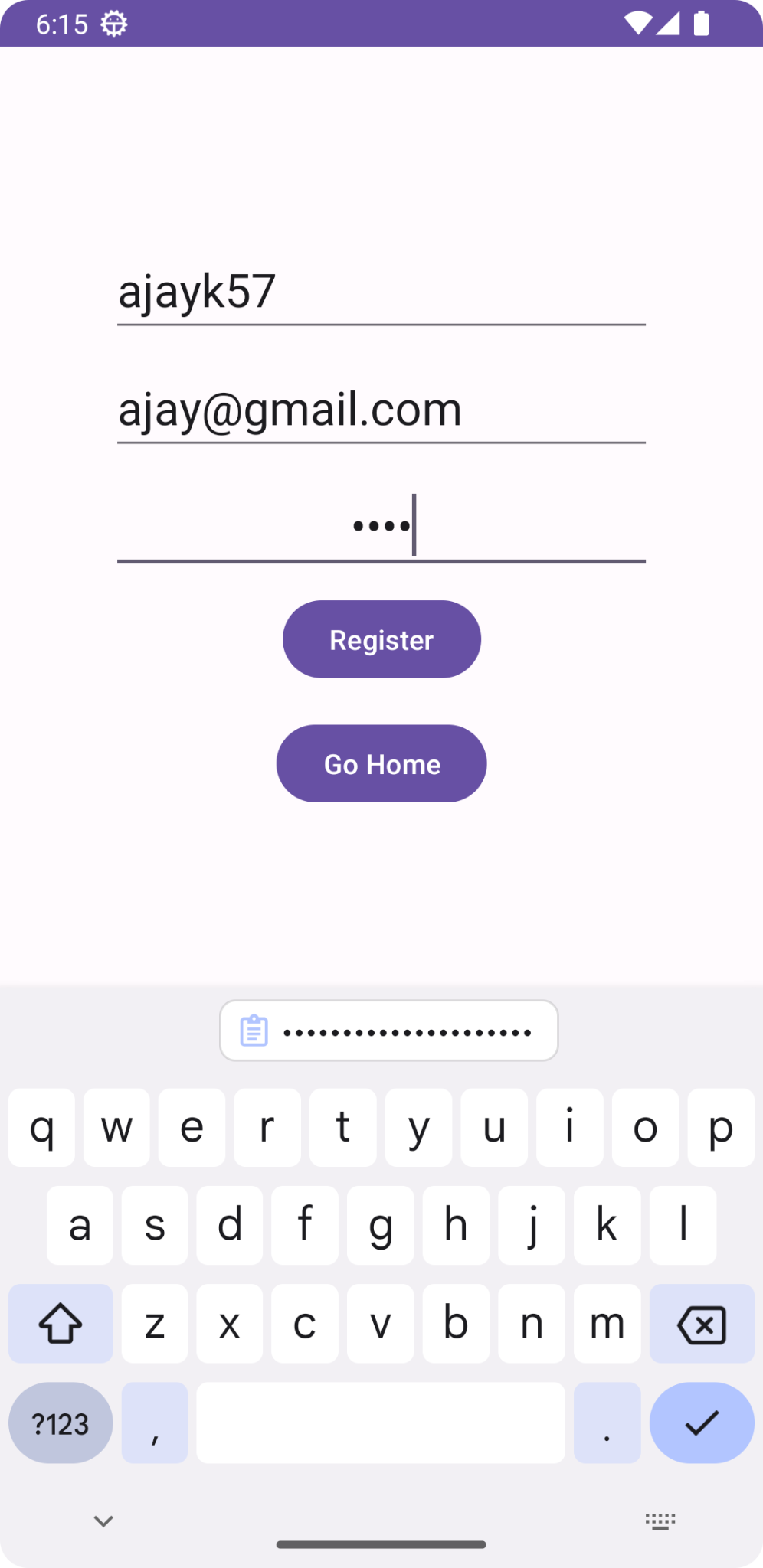
app:layout\_constraintTop\_toBottomOf="@+id/runBBtn"

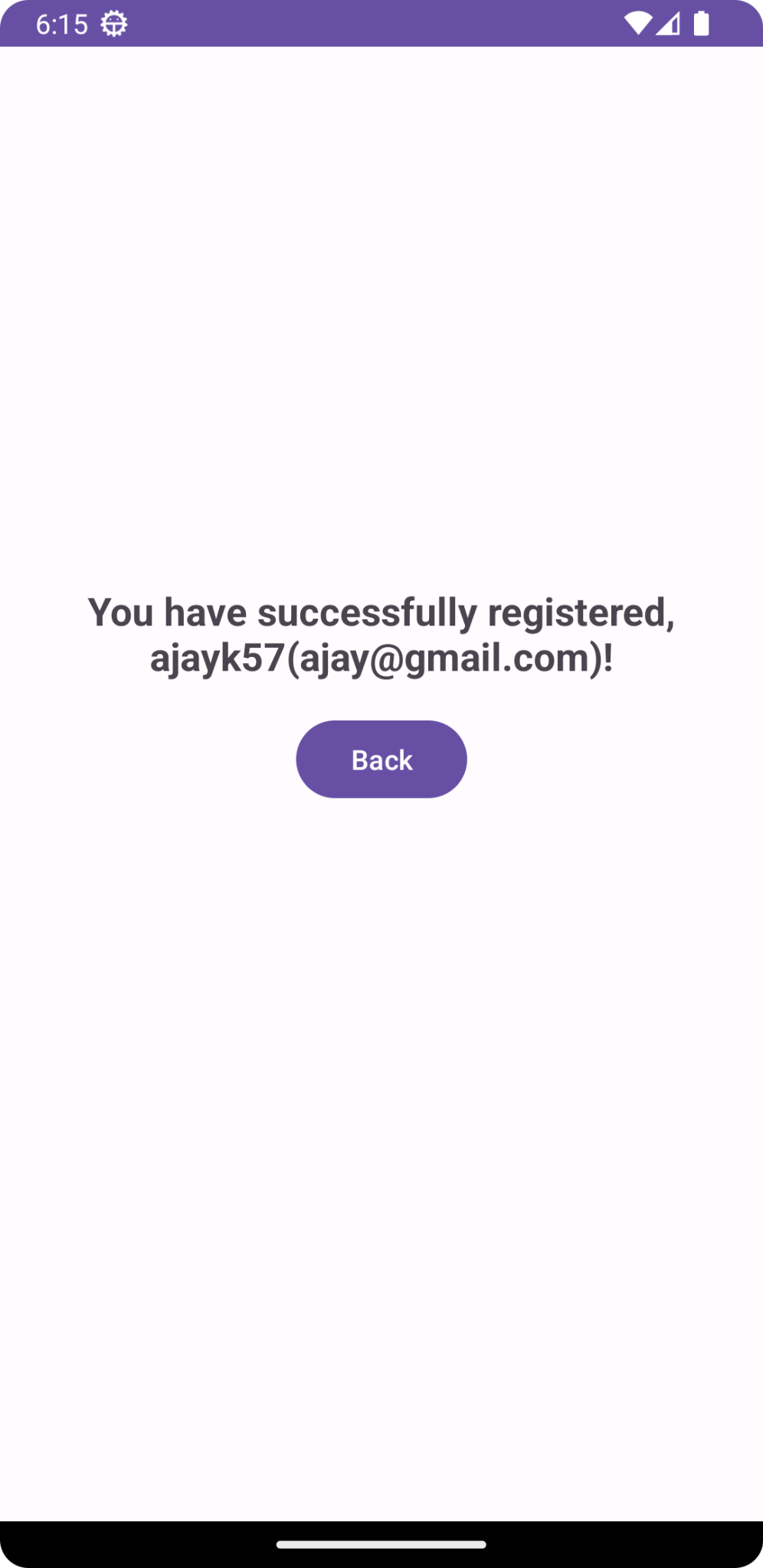
app:layout\_constraintVertical\_bias="0.0" />

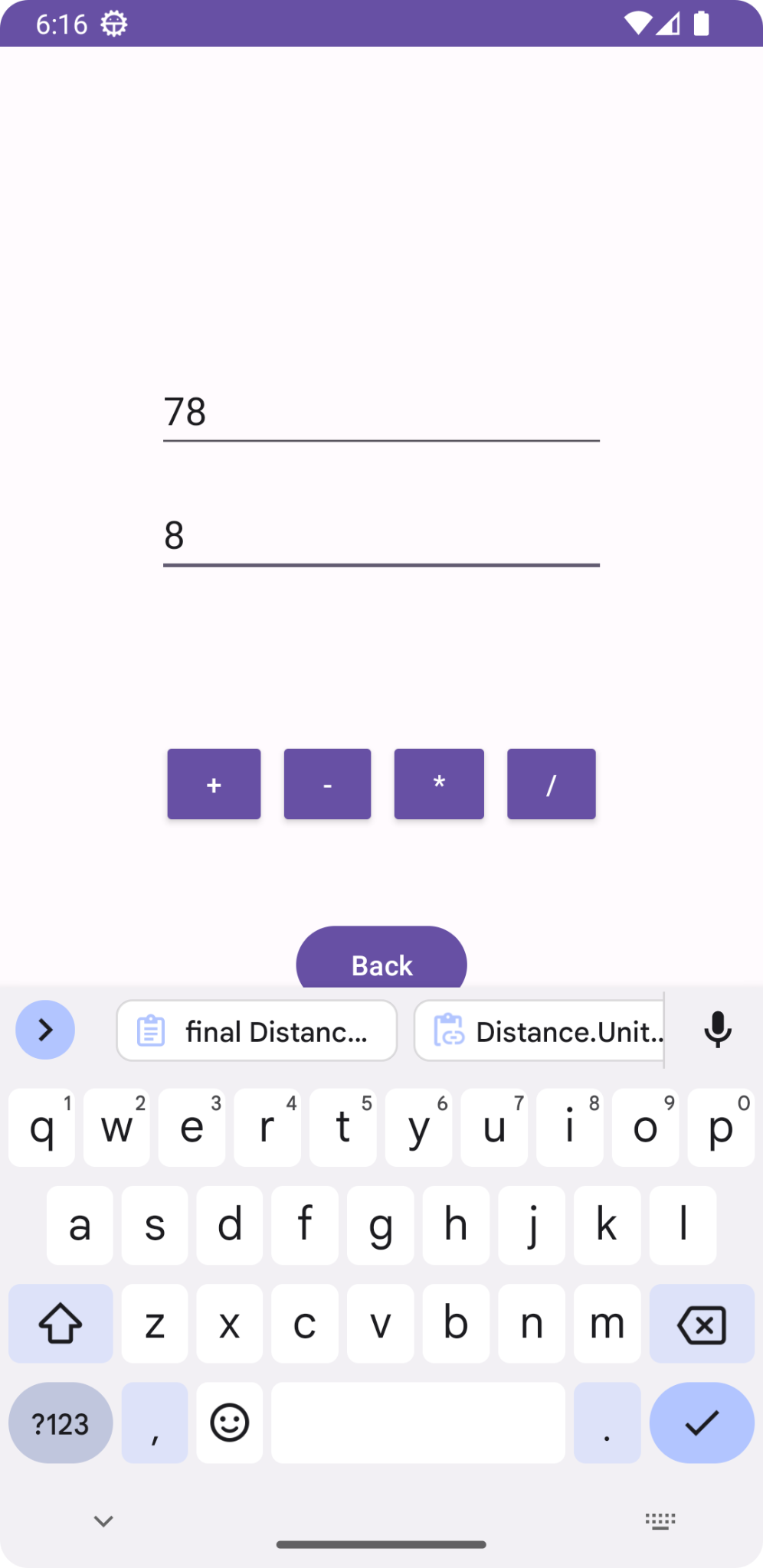
</androidx.constraintlayout.widget.ConstraintLayout>

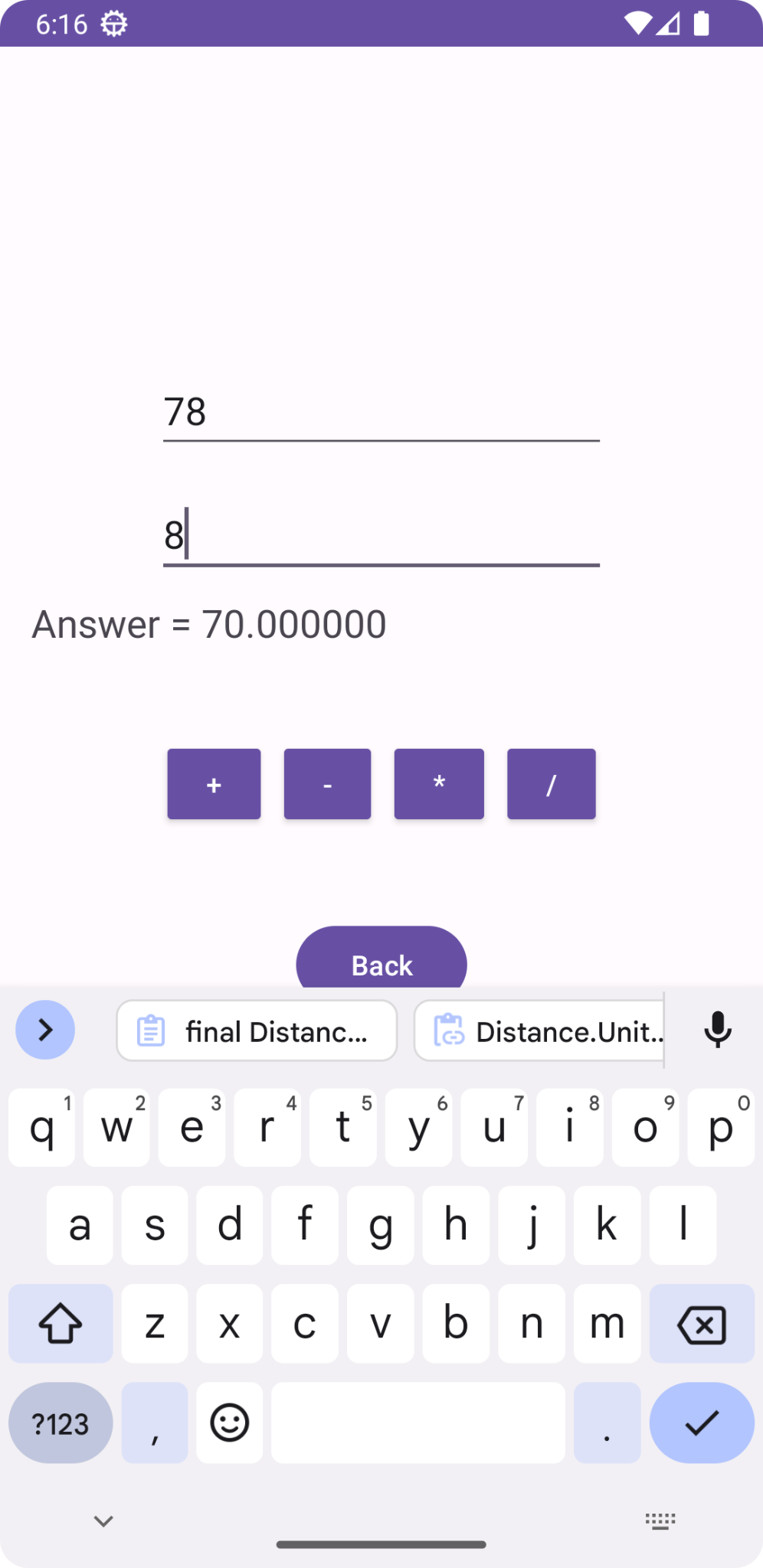
**Output:**

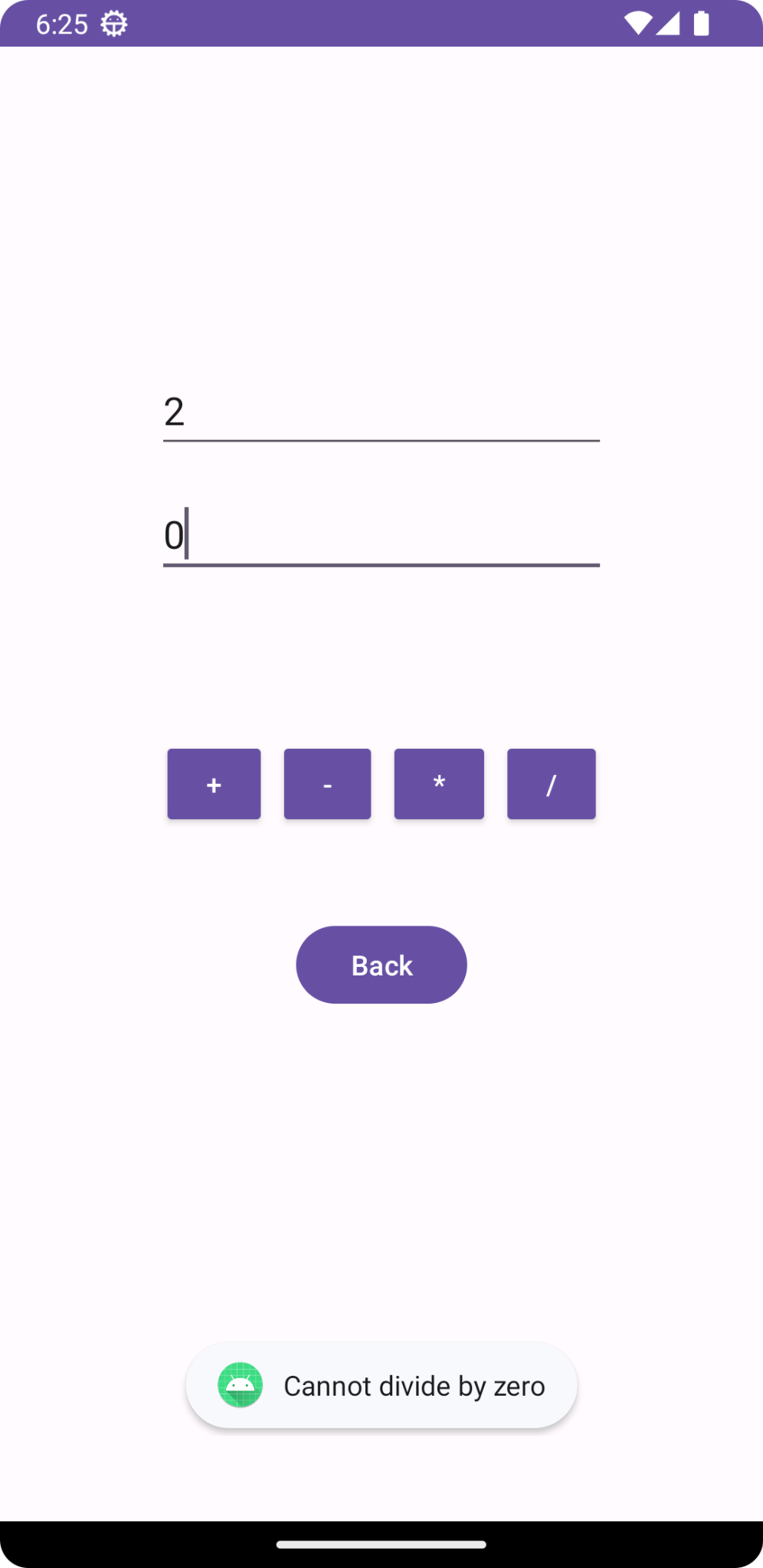


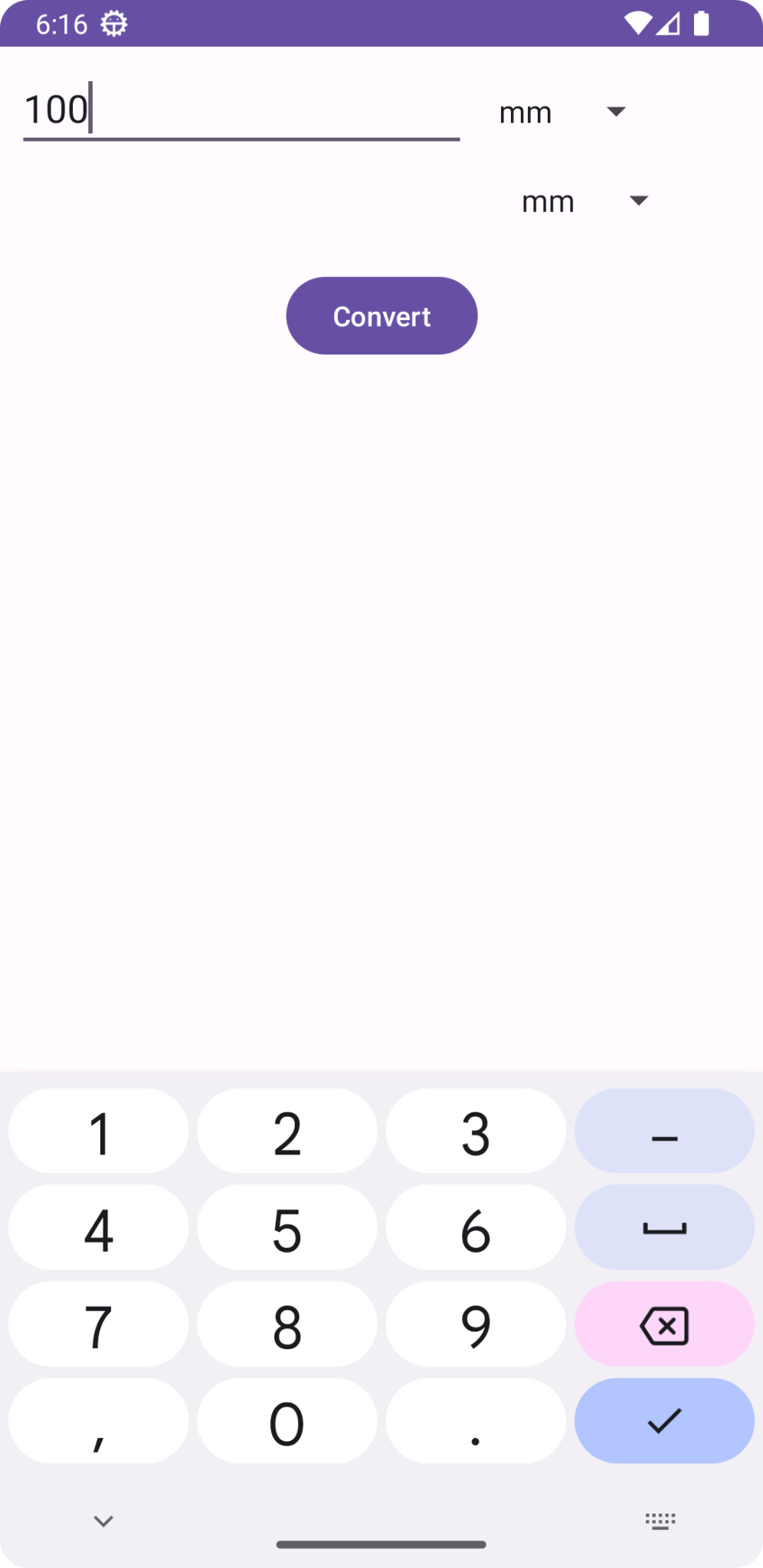


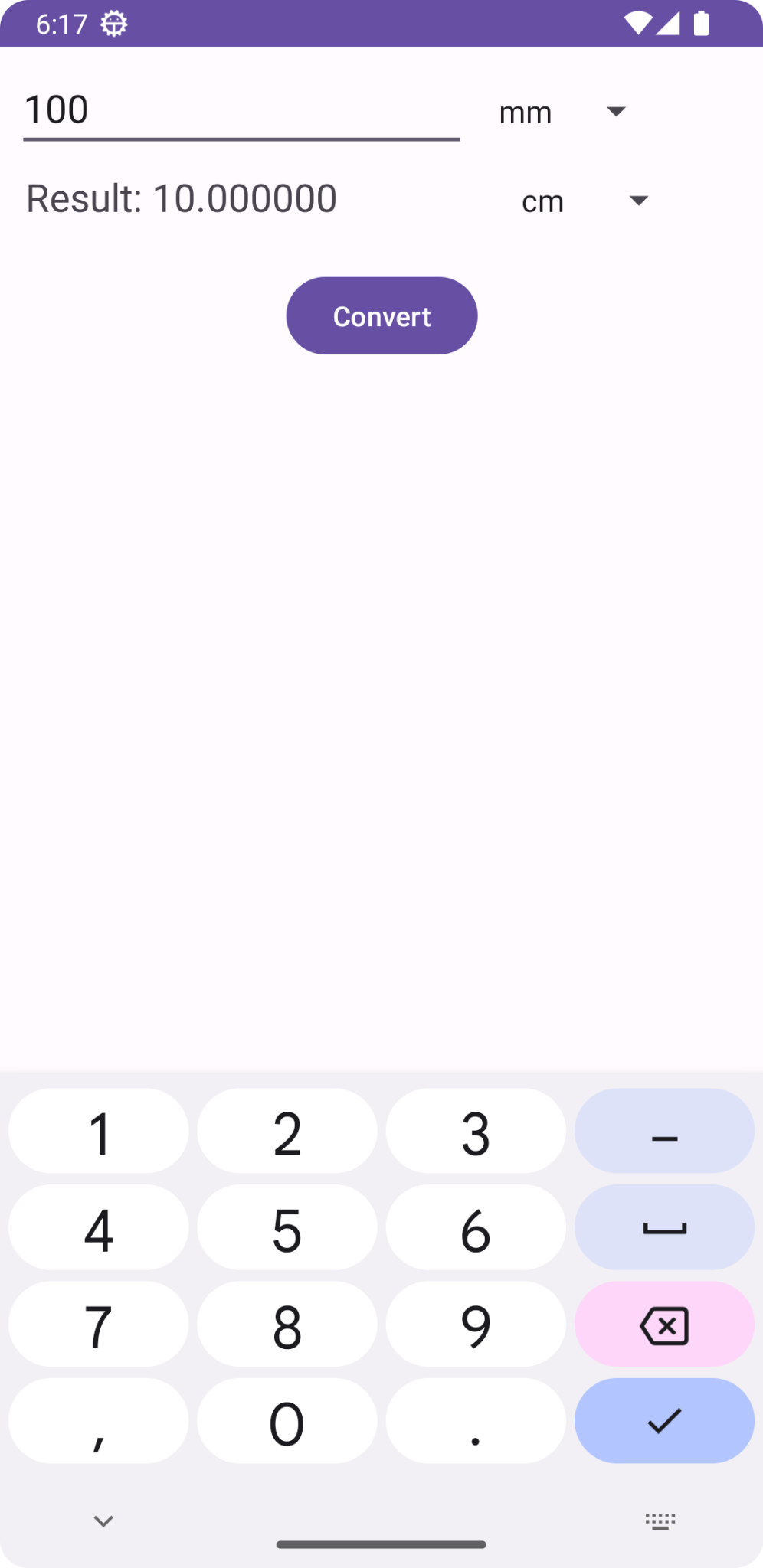












**Conclusion:**

I learnt about layouts in android.