**Lab 8**

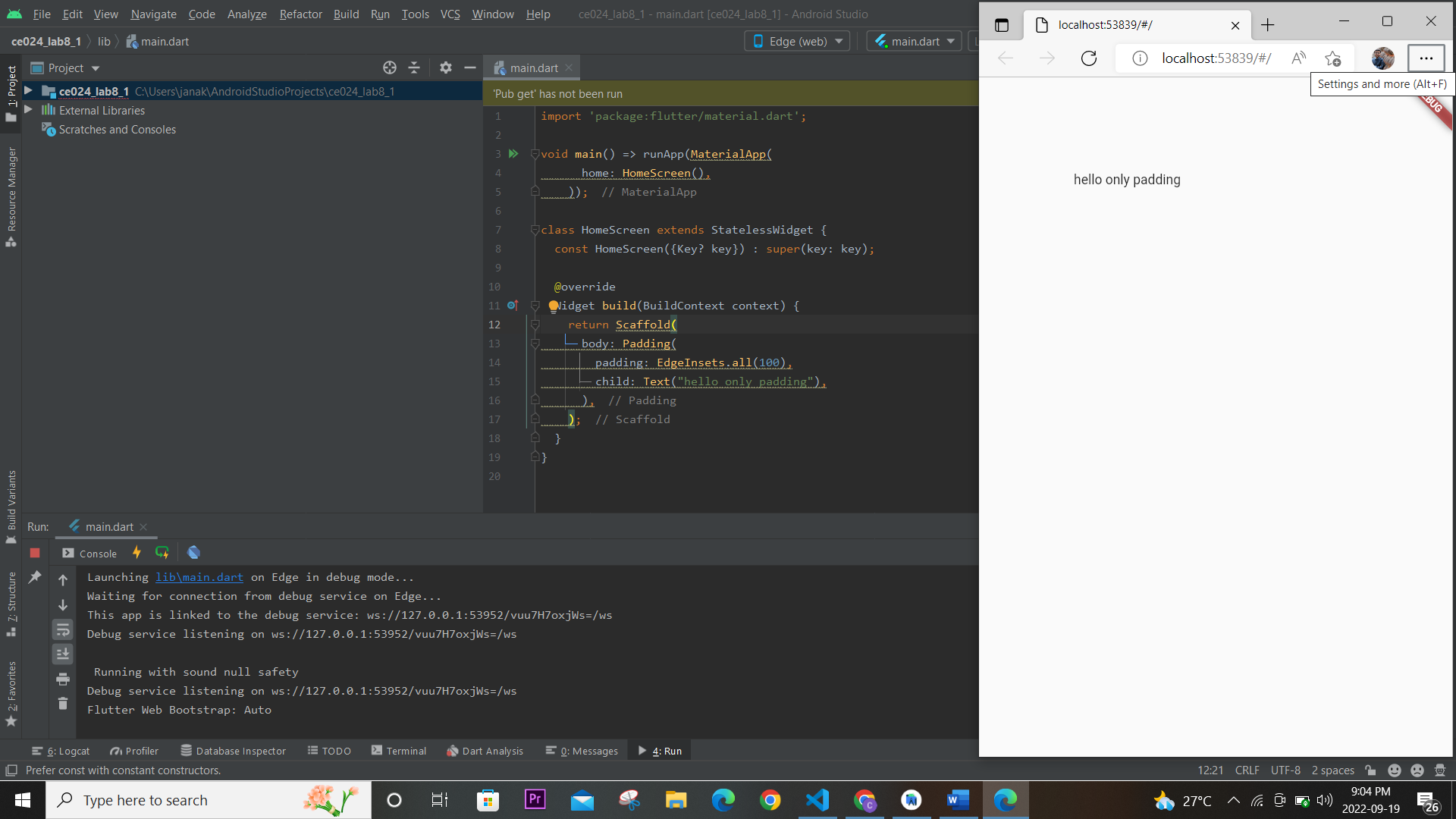
**Name: Chapla Krishna**

**Roll No: CE024**

**Id:21CEUOD006**

Tutorial 8.1

1)Code test: 1:Simple Padding



Padding : Padding widget in flutter does exactly what its name says, it adds padding or empty space around a widget or a bunch of widgets. We can apply padding around any widget by placing it as the child of the Padding widget. The size of the child widget inside padding is constrained by how much space is remaining after adding empty space around. The Padding widget adds empty space around any widget by using the abstract EdgeInsetsGeometry class.

2) code test: 2 Row widget : for multiple components ...

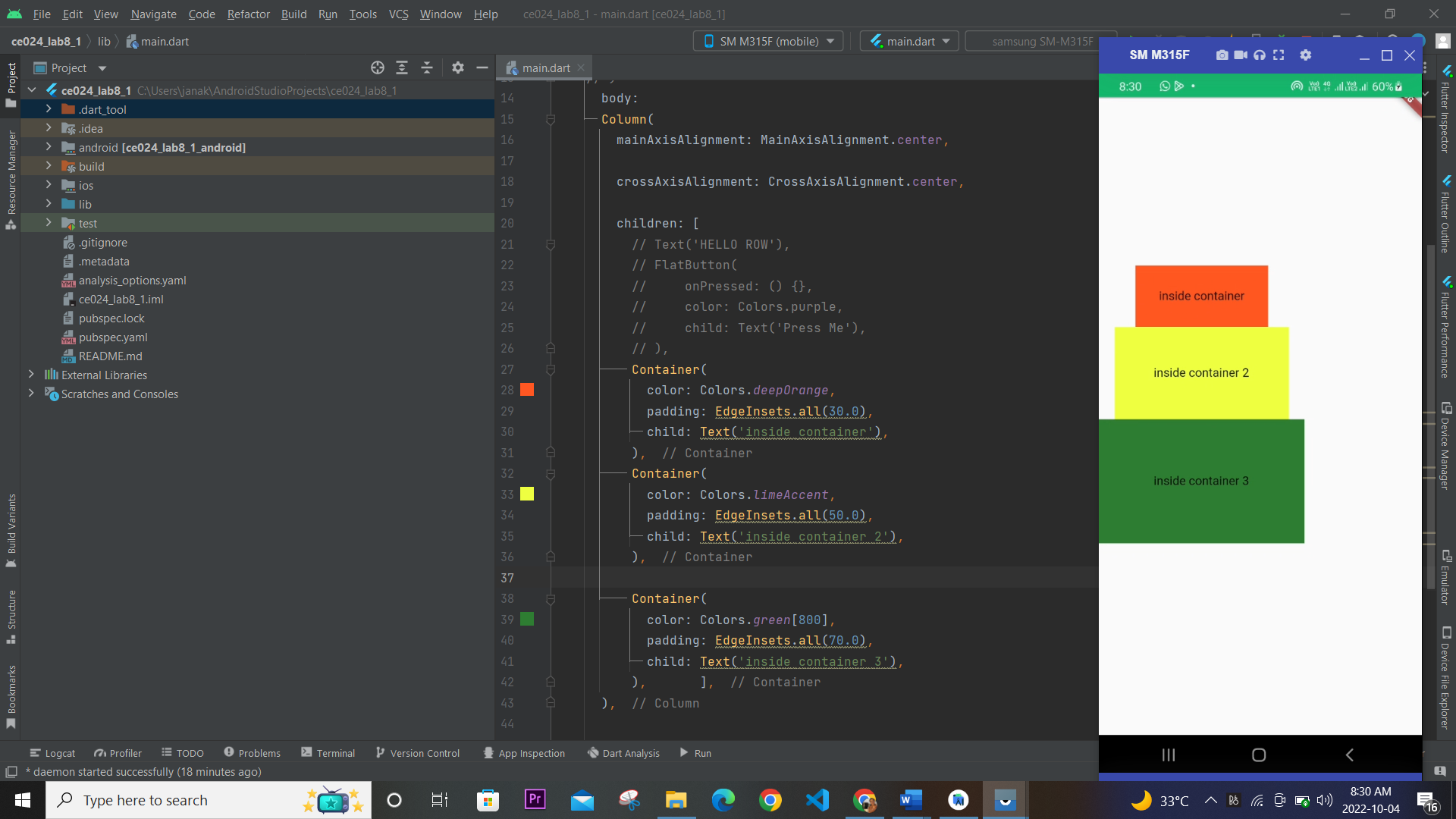
instead of ‘child’ – ‘children : []’ property is used

Row and column are the two essential widgets in [Flutter](https://www.javatpoint.com/flutter) that allows developers to **align children horizontally and vertically according to our needs**. These widgets are very necessary when we design the application user interface in Flutter.

Row Widget: **This widget arranges its children in a horizontal direction on the screen**. In other words, it will expect child widgets in a horizontal array. If the child widgets need to fill the available horizontal space, we must wrap the children widgets in an Expanded widget.

A row widget does not appear scrollable because it displays the widgets within the visible view. So it is considered wrong if we have more children in a row which will not fit in the available space. If we want to make a scrollable list of row widgets, we need to use the ListView widget.

3)Column Widget:

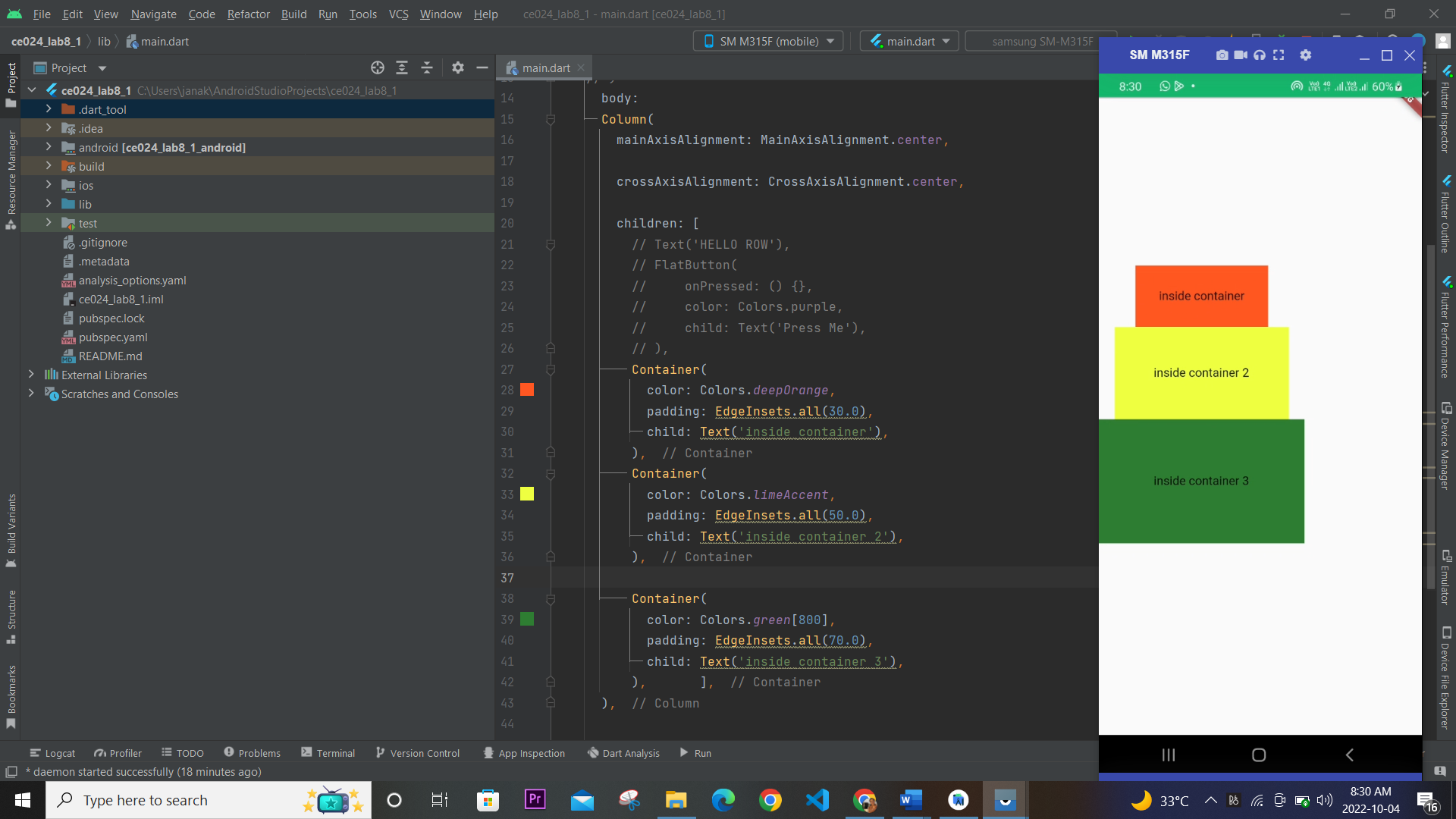


Column Widget: **This widget arranges its children in a vertical direction on the screen**. In other words, it will expect a vertical array of children widgets. If the child widgets need to fill the available vertical space, we must wrap the children widgets in an Expanded widget.

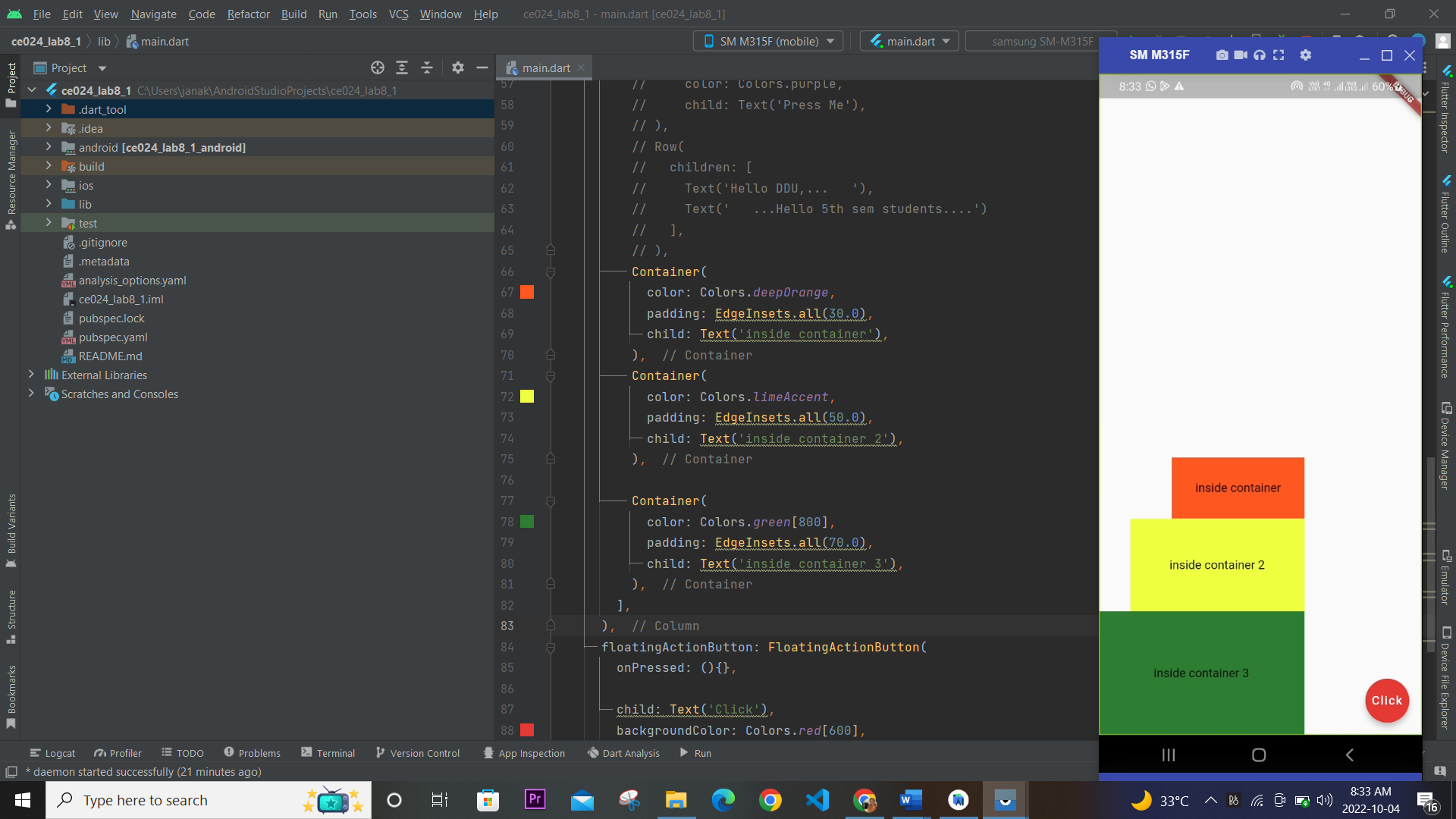
A column widget does **not appear scrollable** because it displays the widgets within the visible view. So it is considered wrong if we have more children in a column which will not fit in the available space. If we want to make a scrollable list of column widgets, we need to use the ListView Widget.

We can also control how a column widget aligns its children using the property mainAxisAlignment and crossAxisAlignment. The column's **cross-axis**will run **horizontally**, and the **main axis** will run **vertically**. The below visual representation explains it more clearly.

4)Row and column test 1:



5)Row and column test 2:



CrossAxisAlignment: CrossAxisAlignment is a property of the Column widget. It is used to arrange children’s widgets in Horizontally format according to the given axis.

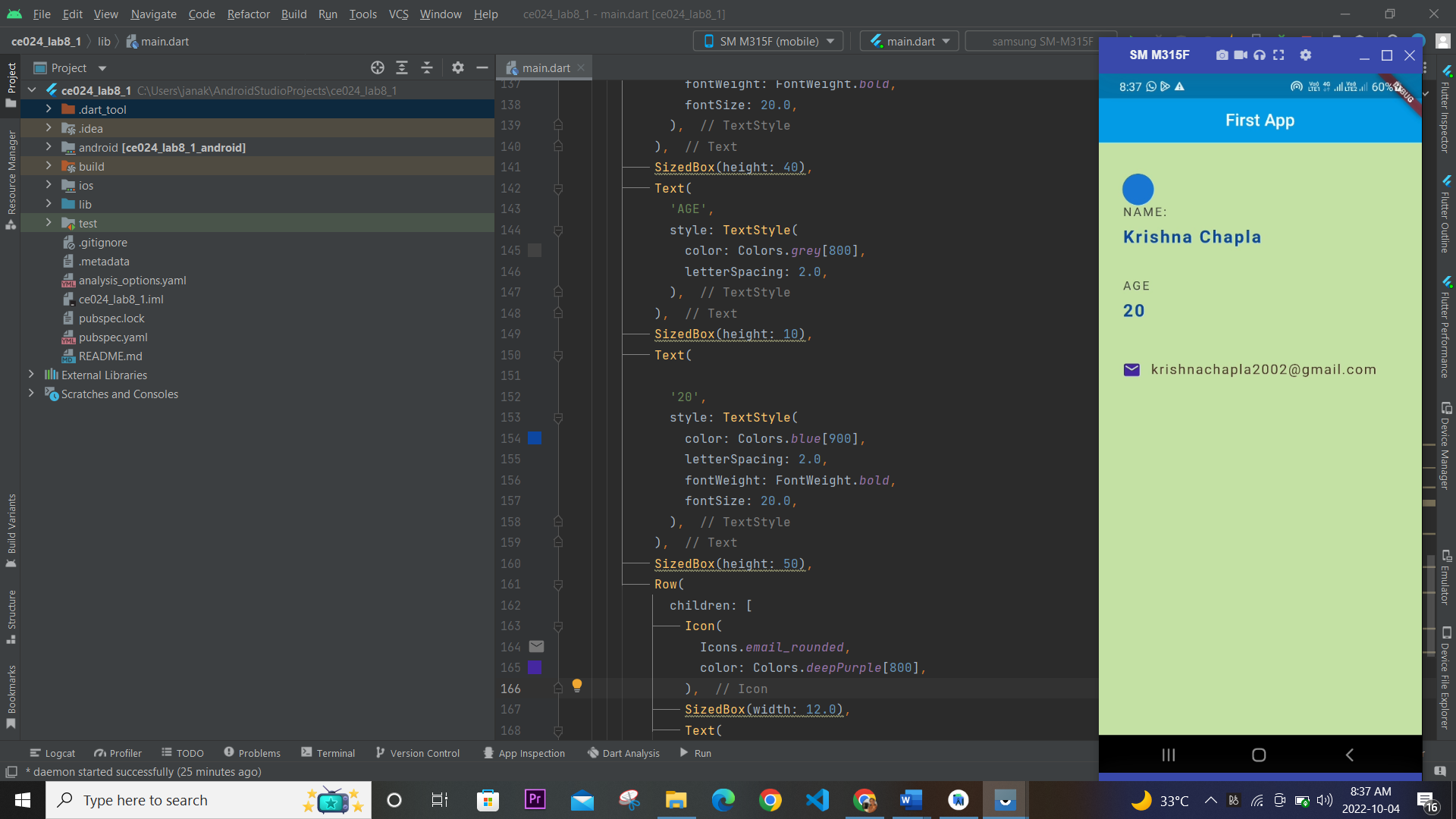
mainAxisAlignment: MainAxisAlignment is a property of the Column widget. It is used to arrange children’s widgets in vertical format according to the given axis.

6) Code test: Expanded and Flex widget

Expanded Widget: **Expanded**widget in flutter comes in handy when we want a child widget or children widgets to take up all the available space along the main axis (for Row the main axis is horizontal & vertical for the Column).  The expanded widget can be taken as the child of Row, Column, and Flex. And in case we don’t want to give equal spaces to our children widgets we can distribute the available space as our will using the flex factor. **The expanded** widget is similar to the Flexible widget in a flutter, with its fit property set to FlexFit.tight as default. **An expanded** widget is basically a shorthand of a Flexible widget.

Properties :child,flex,fit,debugTypicalAncestorWidgetClass

7)Final Custom Design:



10)Final Code for tutorial 8:

// import 'package:flutter/material.dart';  
// class HomeScreen extends StatelessWidget {  
// @override  
// Widget build(BuildContext context) {  
// return Scaffold(  
//  
// // body: Column(  
// // mainAxisAlignment: MainAxisAlignment.spaceAround,  
// // crossAxisAlignment: CrossAxisAlignment.stretch,  
// // children: [  
// // Text('hello ddu'),  
// // ],  
// // )  
// // body:  
// // Column(  
// // mainAxisAlignment: MainAxisAlignment.center,  
// //  
// // crossAxisAlignment: CrossAxisAlignment.center,  
// //  
// // children: [  
// // // Text('HELLO ROW'),  
// // // FlatButton(  
// // // onPressed: () {},  
// // // color: Colors.purple,  
// // // child: Text('Press Me'),  
// // // ),  
// // Container(  
// // color: Colors.deepOrange,  
// // padding: EdgeInsets.all(30.0),  
// // child: Text('inside container'),  
// // ),  
// // Container(  
// // color: Colors.limeAccent,  
// // padding: EdgeInsets.all(50.0),  
// // child: Text('inside container 2'),  
// // ),  
// //  
// // Container(  
// // color: Colors.green[800],  
// // padding: EdgeInsets.all(70.0),  
// // child: Text('inside container 3'),  
// // ), ],  
// // ),  
//  
// body:  
// Column(  
// // mainAxisAlignment: MainAxisAlignment.center,  
// mainAxisAlignment: MainAxisAlignment.end,  
//  
// // crossAxisAlignment: CrossAxisAlignment.center,  
// crossAxisAlignment: CrossAxisAlignment.end,  
//  
// children: [  
// // Text('HELLO ROW'),  
// // FlatButton(  
// // onPressed: () {},  
// // color: Colors.purple,  
// // child: Text('Press Me'),  
// // ),  
// // Row(  
// // children: [  
// // Text('Hello DDU,... '),  
// // Text(' ...Hello 5th sem students....')  
// // ],  
// // ),  
// Container(  
// color: Colors.deepOrange,  
// padding: EdgeInsets.all(30.0),  
// child: Text('inside container'),  
// ),  
// Container(  
// color: Colors.limeAccent,  
// padding: EdgeInsets.all(50.0),  
// child: Text('inside container 2'),  
// ),  
//  
// Container(  
// color: Colors.green[800],  
// padding: EdgeInsets.all(70.0),  
// child: Text('inside container 3'),  
// ),  
// ],  
// ),  
// floatingActionButton: FloatingActionButton(  
// onPressed: (){},  
//  
// child: Text('Click'),  
// backgroundColor: Colors.red[600],  
// ),  
// );  
//  
// }  
// }  
//  
// void main() => runApp(  
// MaterialApp(  
// home: HomeScreen()));  
  
  
import 'package:flutter/material.dart';  
void main() => runApp(MaterialApp(  
 home: HomeScreen(),  
));  
class HomeScreen extends StatelessWidget {  
// const FinalTest1({Key? key}) : super(key: key);  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*lightGreen*[200],  
 appBar: AppBar(  
 title: Text('First App'),  
 centerTitle: true,  
 backgroundColor: Colors.*lightBlue*[600],  
 elevation: 0.0,  
 ),  
 body: Padding(  
 padding: EdgeInsets.fromLTRB(30, 40, 30, 0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 CircleAvatar(  
 backgroundImage: AssetImage('assets/dog3.jpg'),  
 ),  
 Text(  
 'NAME: ',  
 style: TextStyle(  
 color: Colors.*grey*[800],  
 letterSpacing: 2.0,  
 ),  
 ),  
 SizedBox(height: 10),  
 Text(  
 'Krishna Chapla',  
 style: TextStyle(  
 color: Colors.*blue*[900],  
 letterSpacing: 2.0,  
 fontWeight: FontWeight.*bold*,  
 fontSize: 20.0,  
 ),  
 ),  
 SizedBox(height: 40),  
 Text(  
 'AGE',  
 style: TextStyle(  
 color: Colors.*grey*[800],  
 letterSpacing: 2.0,  
 ),  
 ),  
 SizedBox(height: 10),  
 Text(  
  
 '20',  
 style: TextStyle(  
 color: Colors.*blue*[900],  
 letterSpacing: 2.0,  
 fontWeight: FontWeight.*bold*,  
 fontSize: 20.0,  
 ),  
 ),  
 SizedBox(height: 50),  
 Row(  
 children: [  
 Icon(  
 Icons.*email\_rounded*,  
 color: Colors.*deepPurple*[800],  
 ),  
 SizedBox(width: 12.0),  
 Text(  
 'krishnachapla2002@gmail.com',  
  
 style: TextStyle(  
  
 color: Colors.*brown*[800],  
 fontSize: 16.0,  
 letterSpacing: 1.5,  
  
 ),  
 )  
 ],  
 )  
 ],  
 ),  
 ),  
 );  
 }  
}

Tutorial 8.2 :

1)Stateful Widget: The widgets whose state can not be altered once they are built are called stateless widgets. These widgets are immutable once they are built i.e any amount of change in the variables, icons, buttons, or retrieving data can not change the state of the app. Below is the basic structure of a stateless widget. A stateless widget overrides the build() method and returns a widget. For example, we use Text or the Icon in our flutter application where the state of the widget does not change in the runtime. It is used when the UI depends on the information within the object itself. Other examples can be Text, RaisedButton, and IconButtons.

Card Widget:The card is a built-in widget in flutter which derives its design from Google’s Material Design Library. The functionality of this widget on screen is, that it is a bland space or panel with round corners and a slight elevation on the lower side. It comes with many properties like color, shape, shadow color, etc which lets developers customize it the way they like.

Properties :child,borderOnForeground,elevation,margin,etc.