Mobile Computing Lab

Subject Code: MCAL34

A Practical Journal Submitted in Fulfilment
of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2024-2025

By

Mr. Jadhav Harshal Sanjay
(Application Id- 53942)

Seat No.: 1030199

Semester-III

Under the Guidance of

Asst. Prof. Dnyaneshwar Deore

Centre for Distance and Online Education
Vidya Nagari, Kalina, Santacruz East – 400098.
University of Mumbai
PCP Center [Satish Pradhan Dnyanasadhana College, Thane]

Institute of Distance and Open Learning
Vidya Nagari, Kalina, Santacruz East – 400098.

CERTIFICATE

This to certify that, **"Jadhav Harshal Sanjay"** appearing **Master's in computer application (Semester III) Application Id: 53942** has satisfactorily completed the prescribed practical of **MCAL34 - Mobile Computing Lab** as laid down by the University of Mumbai for the academic year 2024-25.

Teacher In Charge	External Examiner	Coordinator – M.C.A

Date:

Place: -

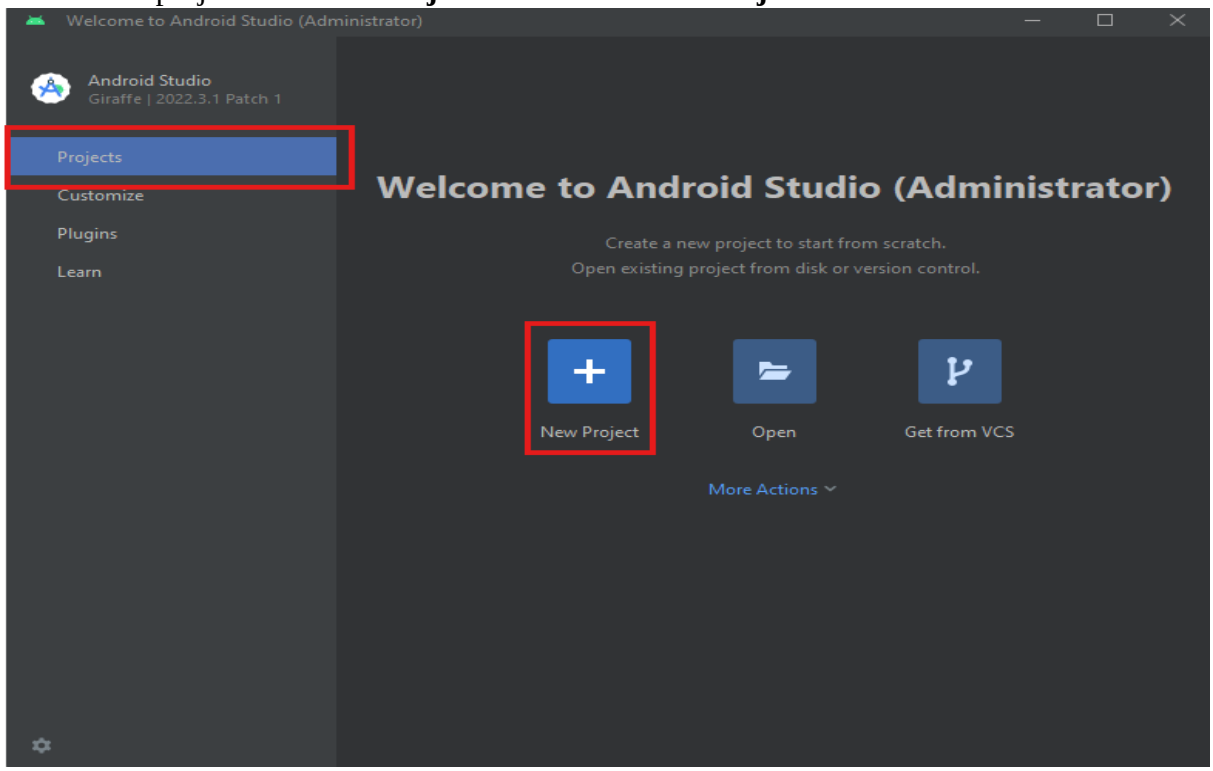
INDEX

Practical 1

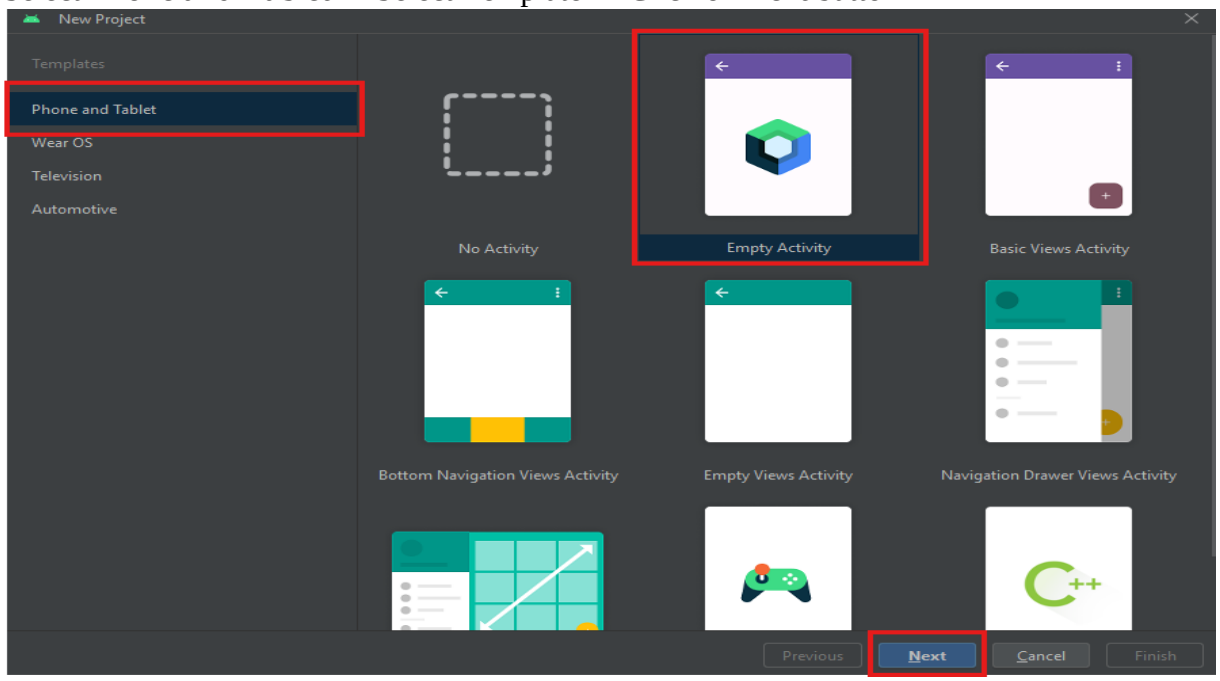
Aim: Steps to create a new android project and run the app.

Steps:

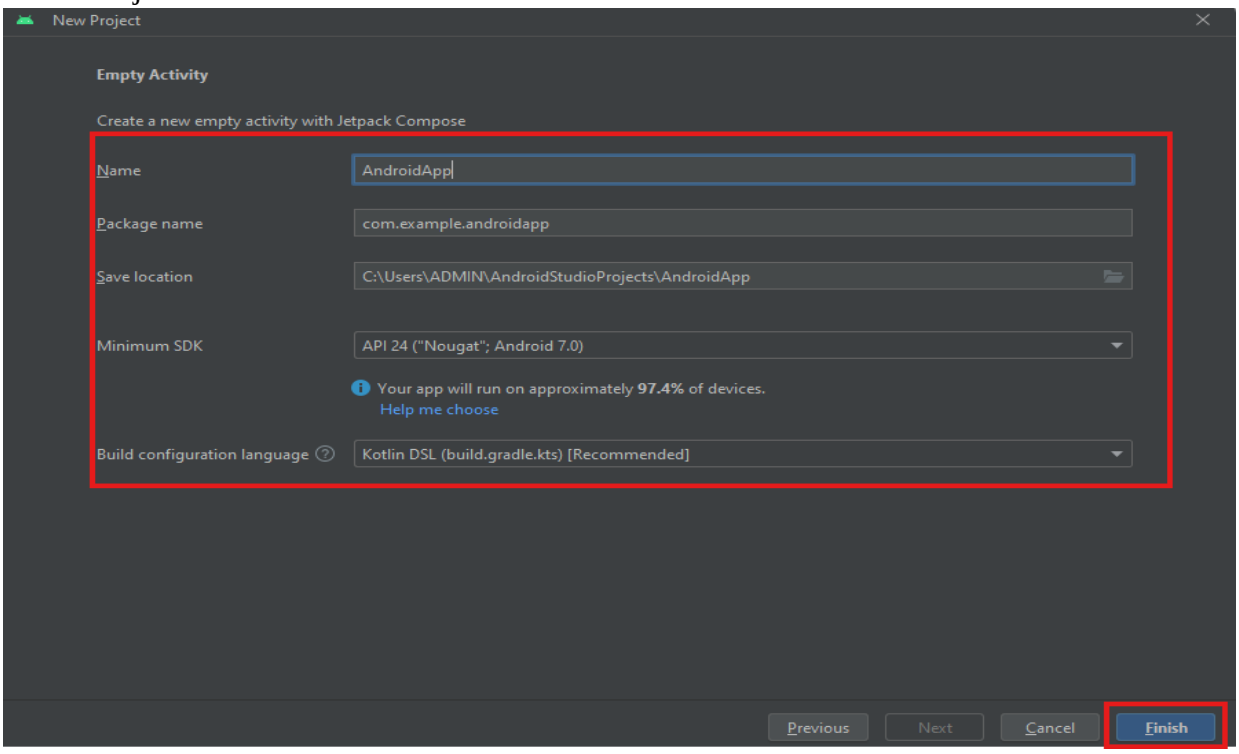
1. Create new project -> **Select Project** -> Click on **New Project**



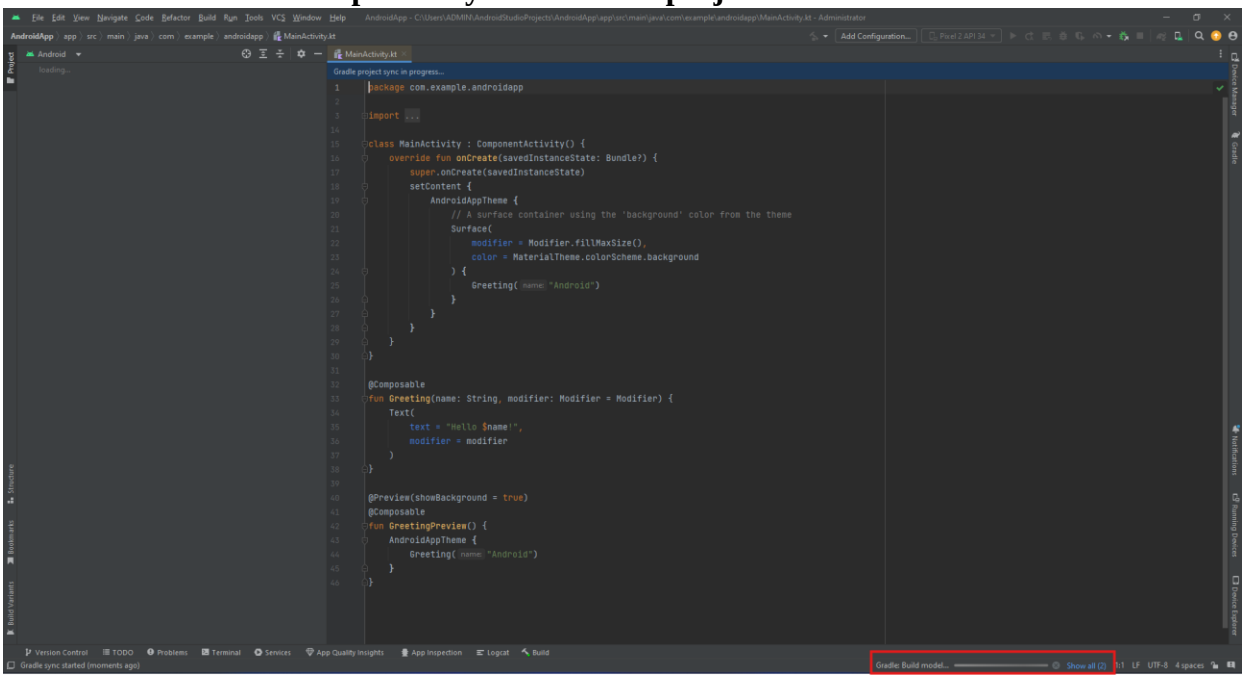
2. Select **Phone and Tablet** -> Select Template -> Click on **Next** button



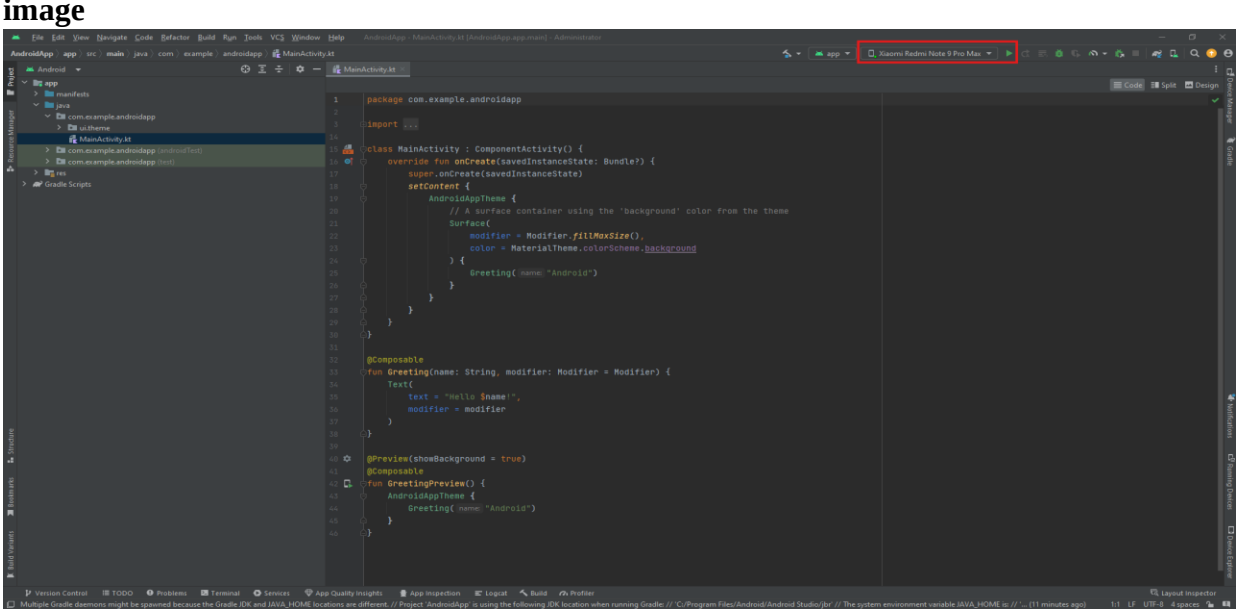
3. Select Project name and other details and click on **Finish**



4. **Wait for download the dependency and load the project**



5. After loading project **select your device and click on Run button as shown in below image**



Output: App will run on selected device

Practical 2

Aim: Creating an android application for basic calculation by using all basic UI controls.

Steps

1. Create a new project with blank activity.
2. Now add the following code in activity_main.xml file

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- First Input -->
    <EditText
        android:id="@+id/input1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter first number"
        android:inputType="numberDecimal" />

    <!-- Second Input -->
    <EditText
        android:id="@+id/input2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter second number"
        android:inputType="numberDecimal" />
```

```
<!-- Result -->
<EditText
    android:id="@+id/result"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Result"
    android:inputType="none"
    android:focusable="false" />

<!-- Add Button -->
<Button
    android:id="@+id/buttonAdd"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Add" />

<!-- Subtract Button -->
<Button
    android:id="@+id/buttonSubtract"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Subtract" />

<!-- Multiply Button -->
<Button
    android:id="@+id/buttonMultiply"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Multiply" />

    <!-- Divide Button -->
    <Button
        android:id="@+id/buttonDivide"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Divide" />
</LinearLayout>
```

3. Add the following code in MainActivity.java file

Code:

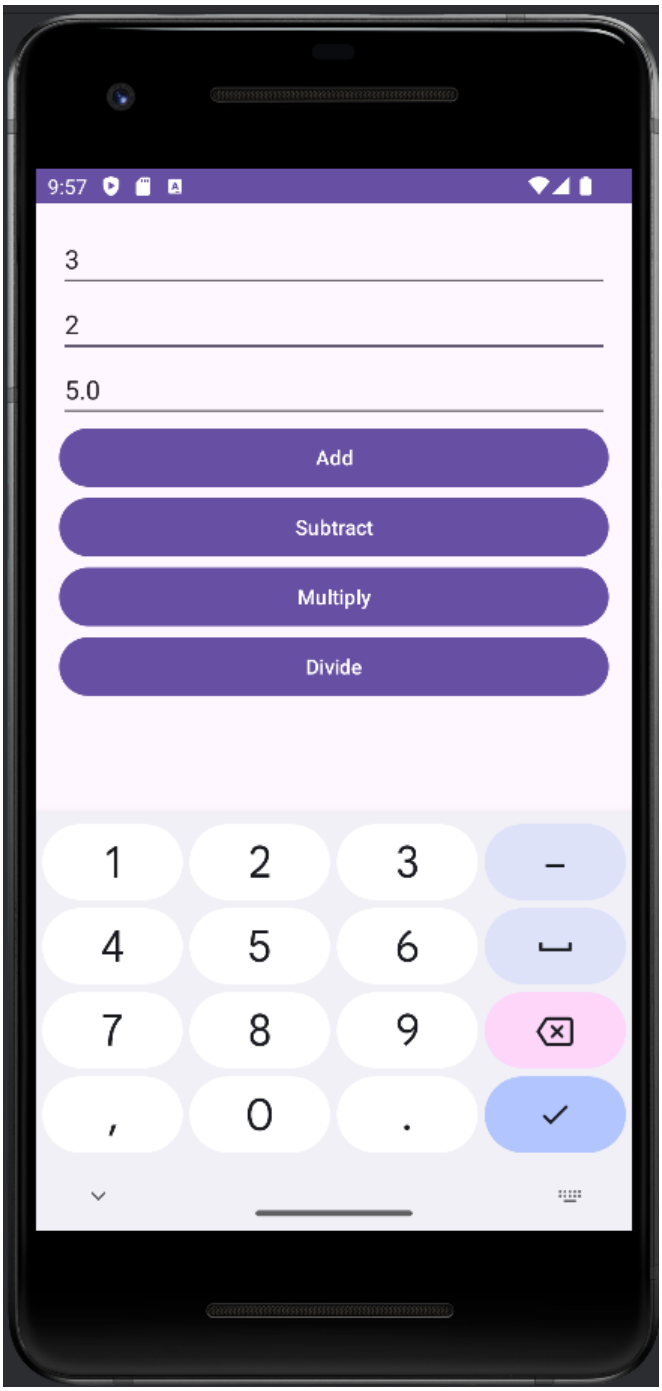
```
package com.example.calculator;

import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private EditText input1, input2, result;
    private Button buttonAdd, buttonSubtract, buttonMultiply, buttonDivide;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Initialize views
    input1 = findViewById(R.id.input1);
    input2 = findViewById(R.id.input2);
    result = findViewById(R.id.result);

    buttonAdd = findViewById(R.id.buttonAdd);
    buttonSubtract = findViewById(R.id.buttonSubtract);
    buttonMultiply = findViewById(R.id.buttonMultiply);
    buttonDivide = findViewById(R.id.buttonDivide);

    // Set click listeners for each button
    buttonAdd.setOnClickListener(v -> calculate("+"));
    buttonSubtract.setOnClickListener(v -> calculate("-"));
    buttonMultiply.setOnClickListener(v -> calculate("*"));
    buttonDivide.setOnClickListener(v -> calculate("/"));
}

private void calculate(String operator) {
    // Validate inputs
    if (TextUtils.isEmpty(input1.getText()) || TextUtils.isEmpty(input2.getText())) {
        Toast.makeText(this, "Please enter both numbers", Toast.LENGTH_SHORT).show();
        return;
    }
```

```
        // Parse inputs
        double num1 = Double.parseDouble(input1.getText().toString());
        double num2 = Double.parseDouble(input2.getText().toString());
        double resultValue = 0;

        // Perform calculation
        switch (operator) {
            case "+":
                resultValue = num1 + num2;
                break;
            case "-":
                resultValue = num1 - num2;
                break;
            case "*":
                resultValue = num1 * num2;
                break;
            case "/":
                if (num2 == 0) {
                    Toast.makeText(this, "Cannot divide by zero",
Toast.LENGTH_SHORT).show();
                    return;
                }
                resultValue = num1 / num2;
                break;
        }

        // Display result
        result.setText(String.valueOf(resultValue));
```

```
    }
}
```

Output:



Practical 3

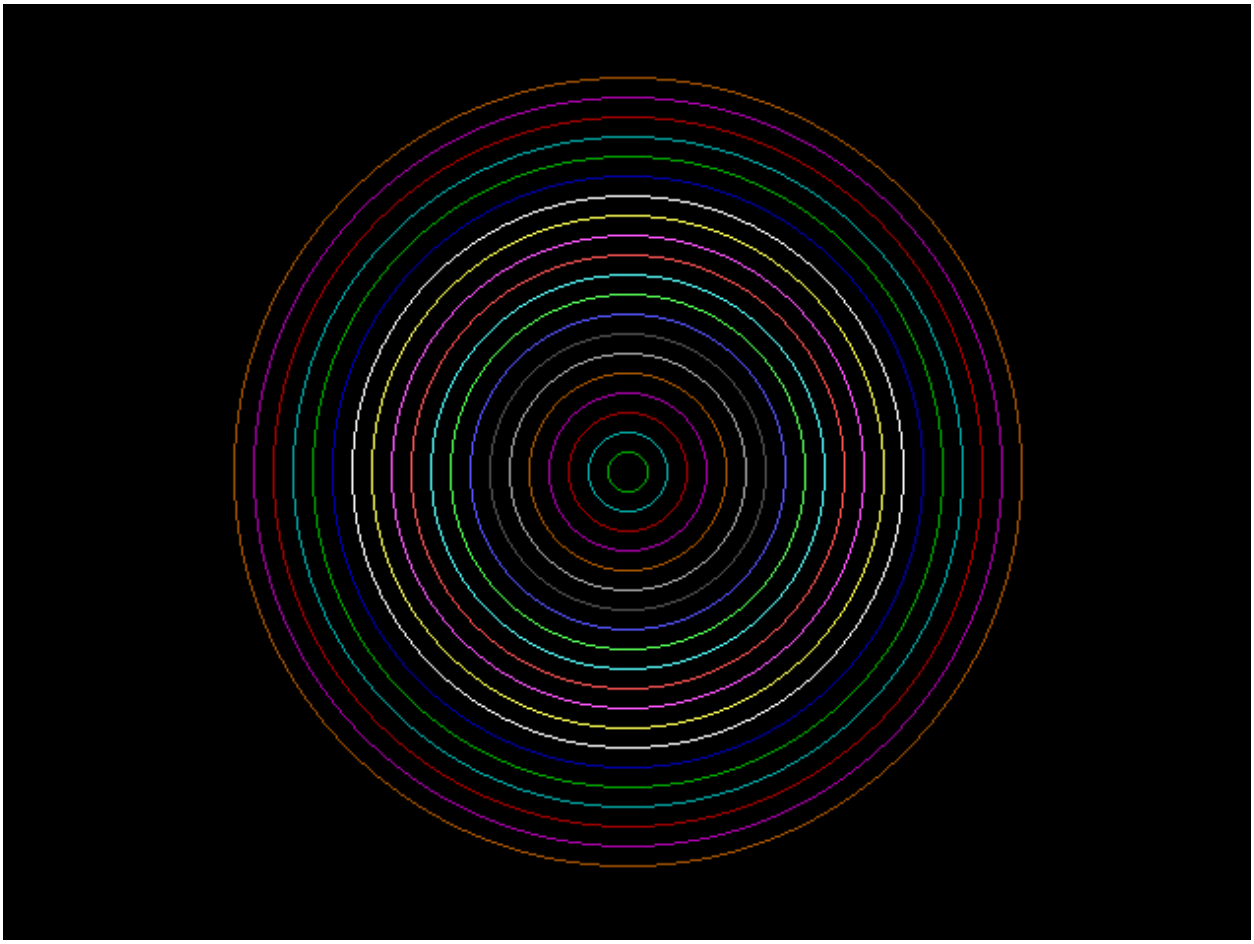
Aim: Write a Program to draw animation using increasing circles filled with different colors and patterns.

Code:

```
#include <graphics.h>

#include <conio.h>

#include <dos.h>


void drawDynamicCircle() {

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

    int x = getmaxx() / 2;

    int y = getmaxy() / 2;

    int radius = 10;

    int maxRadius = 200;

    while (radius <= maxRadius) {

        int color = (radius / 10) % 15 + 1;

        setcolor(color);


        // Draw the circle

        circle(x, y, radius);


        // Increase the radius

        radius += 10;


        // Add delay for better visualization

        delay(200);

    }
```

```
    getch();  // Wait for a key press
    closegraph(); // Close the graphics mode
}

int main() {
    drawDynamicCircle();
    return 0;
}
```

Output:



Practical 4

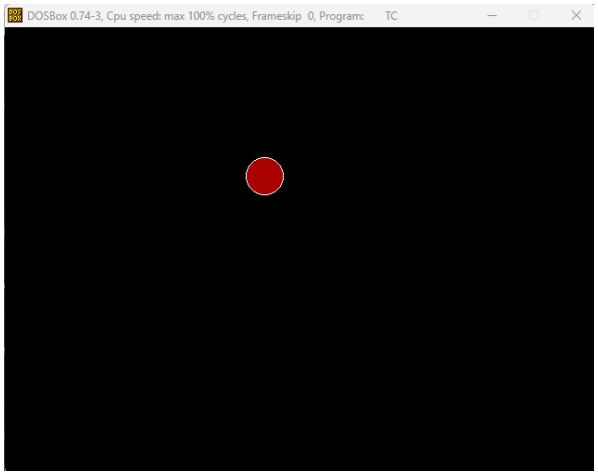
Aim: Write CPP program for bouncing ball graphics animation

Code:

```
#include <graphics.h>
#include <conio.h> // For getch()
#include <dos.h>
void bouncingBall() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI"); // Adjust the path to your setup

    int x = 50, y = 50;          // Initial position of the ball
    int radius = 20;             // Radius of the ball
    int x_speed = 5, y_speed = 5; // Speed of the ball
    int screenWidth = getmaxx();  // Width of the screen
    int screenHeight = getmaxy(); // Height of the screen

    while (!kbhit()) { // Continue until a key is pressed
        cleardevice(); // Clear the screen

        // Draw the ball
        setcolor(WHITE);
        setfillstyle(SOLID_FILL, RED); // Fill color
        fillellipse(x, y, radius, radius);

        // Update the ball's position
        x += x_speed;
        y += y_speed;
```

```
        // Check for collisions with the walls and reverse direction
        if (x - radius <= 0 || x + radius >= screenWidth)
            x_speed = -x_speed;

        if (y - radius <= 0 || y + radius >= screenHeight)
            y_speed = -y_speed;

        delay(30); // Small delay for smooth animation
    }

    closegraph(); // Close the graphics mode
}

int main() {
    bouncingBall();
    return 0;
}
```

Output:

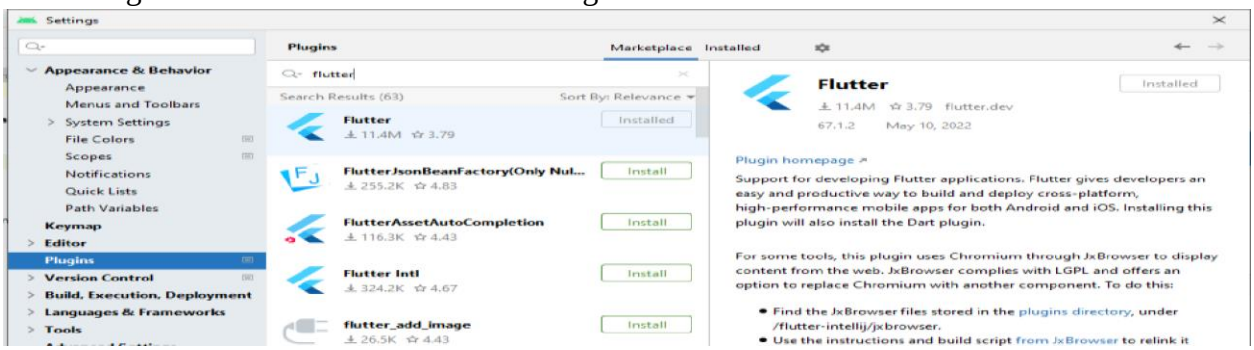


Practical 5

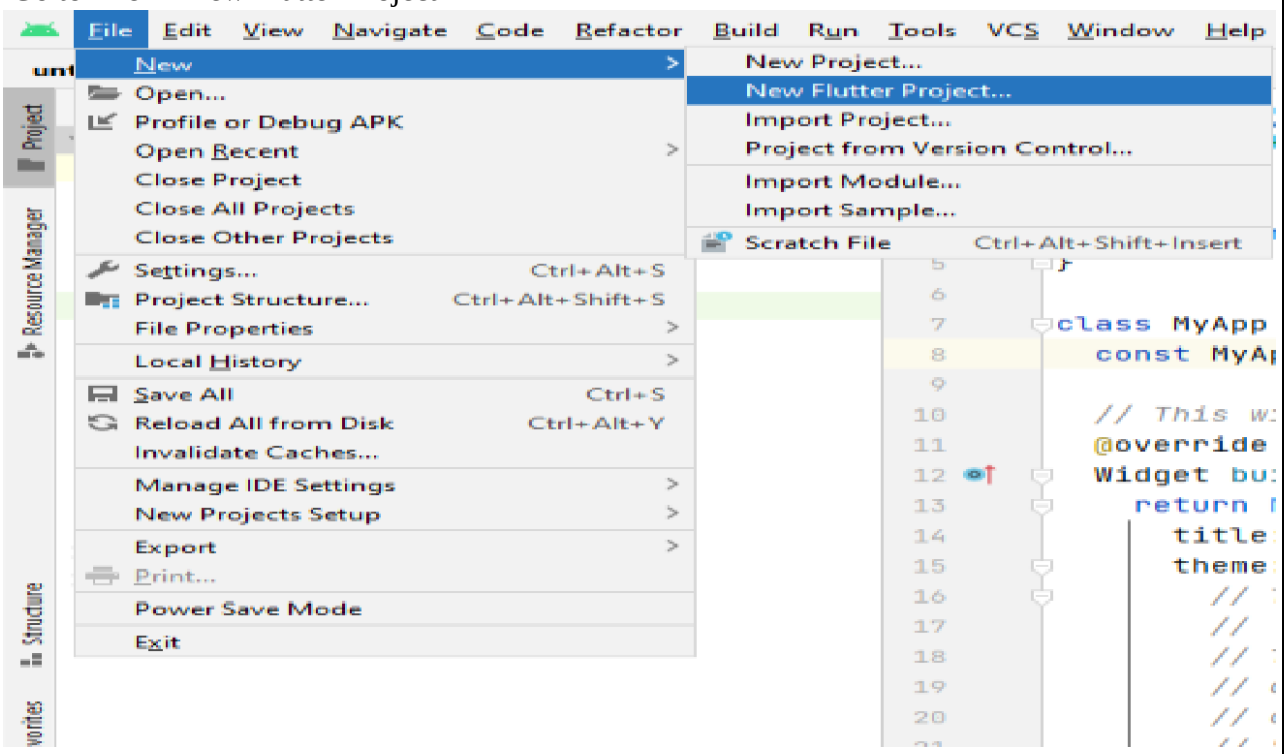
Aim: Demonstrate Step by Step installation and Creation of new flutter app in Android Studio

Steps:

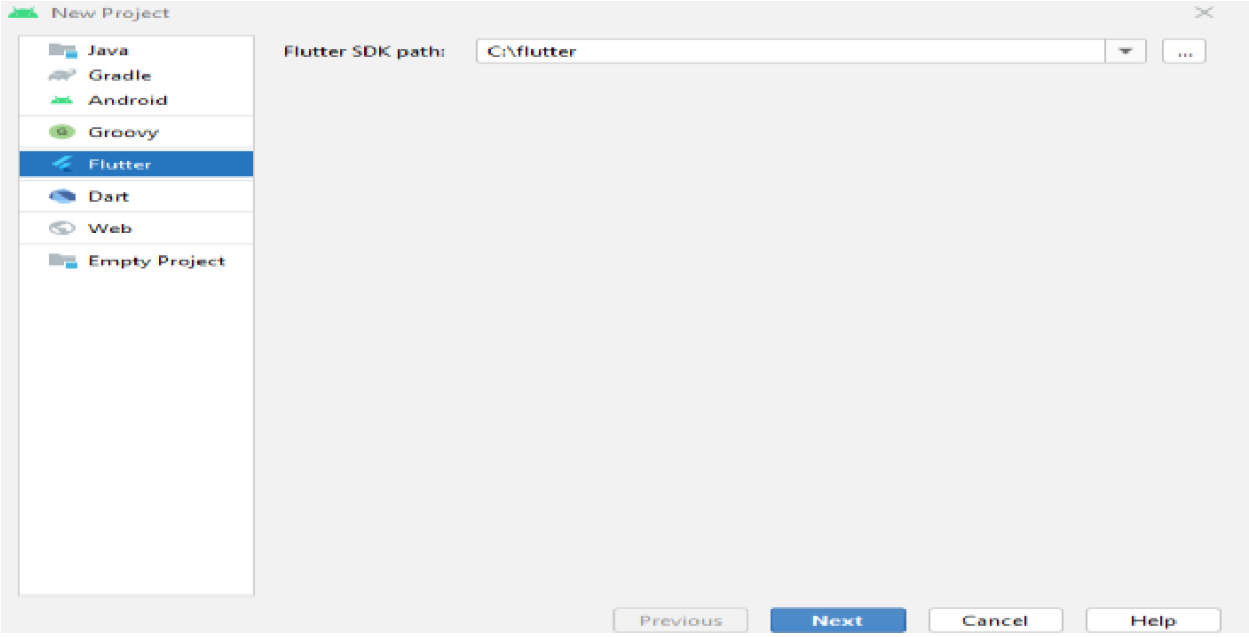
1. Open or create any new project
2. Go to Files->Settings
3. Go to Plugins and install Flutter and Dart Plugin and Restart the IDE



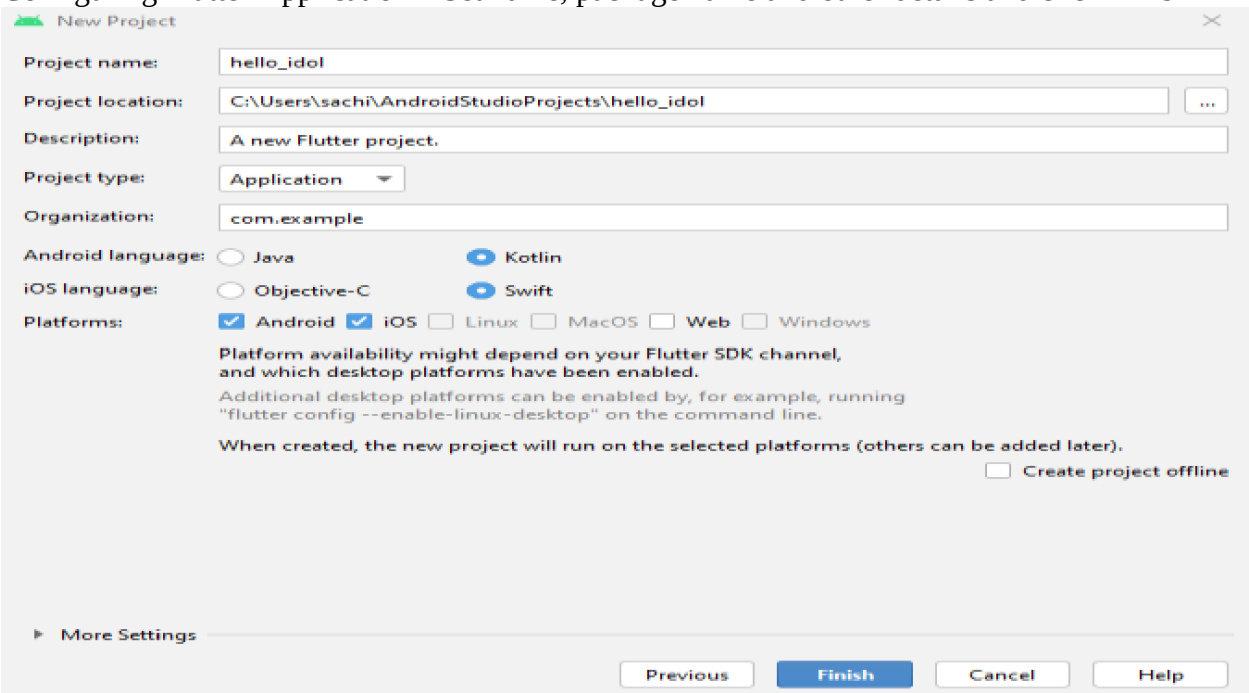
4. Go to File -> New Flutter Project



5. Add path to flutter SDK and click next

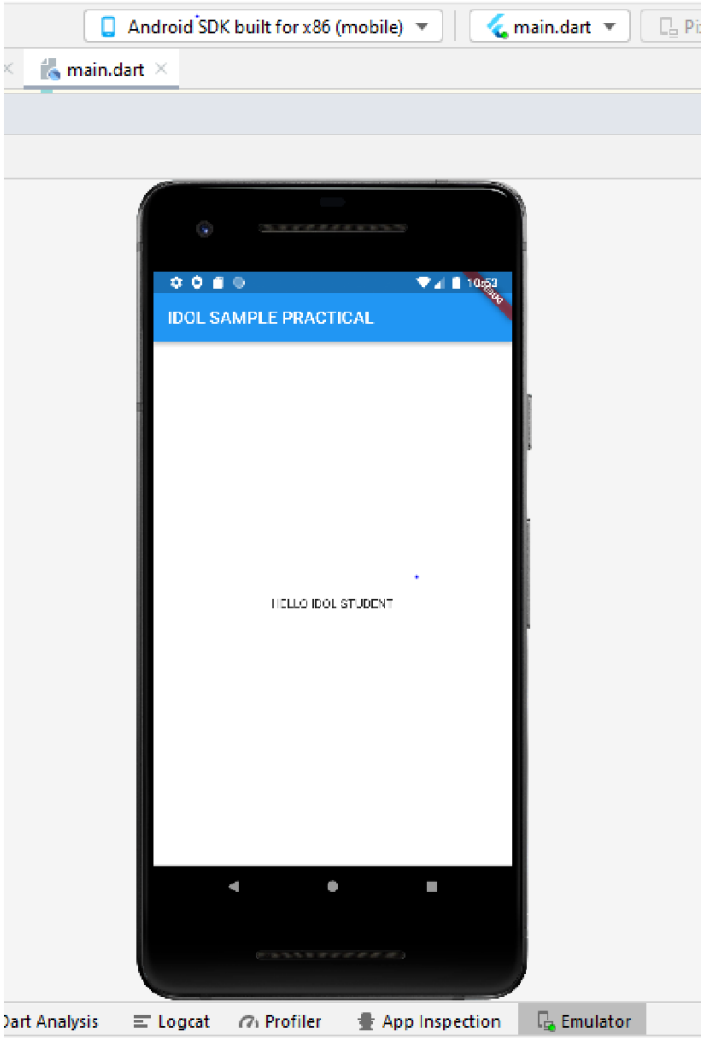


6. Configuring Flutter Application – Set name, package name and other details and click Finish



7. Now the project will load on the IDE, now we can run the project on device or emulator.

Output:


Android SDK built for x86 (mobile)
main.dart
main.dart
IDOL SAMPLE PRACTICAL
HELLO IDOL STUDENT
Logcat
Profiler
App Inspection
Emulator

Practical 6

Aim: To demonstrate the flutter UI widgets like Stateless Widgets, Events etc.

App Bar
Code:

```
import 'package: flutter/material.dart';

void main() => runApp (const IDOLMCA());
class IDOLMCA extends StatelessWidget {
const IDOLMCA({Key? key): super(key: key);
@override Widget build(BuildContext context)
{return MaterialApp(
home: Scaffold(
backgroundColor: Colors.grey,-appBar: AppBar(backgroundColor: Colors.green, title: const
Text ("IDOL MCA"),
), // AppBar
body: const Center (child: Text("Stateless Widget"),
 ), // Center
), // Scaffold
); // MaterialApp
}
}
```

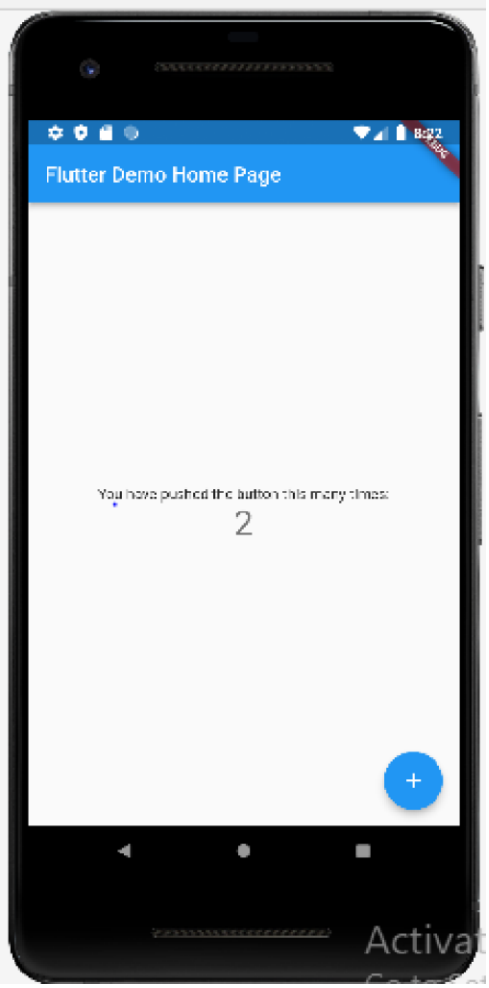
Output:

Counter App

Code:

```
import 'package: flutter/material.dart';

void main() => runApp (MyApp());
class MyApp extends StatefulWidget {
}
@override
_MyAppState createState() => _MyAppState();
class MyAppState extends State<MyApp> { @override
Widget build (BuildContext context) { return Container();
}
}
```

Output:


Flutter Demo Home Page
You have pushed the button this many times:
2

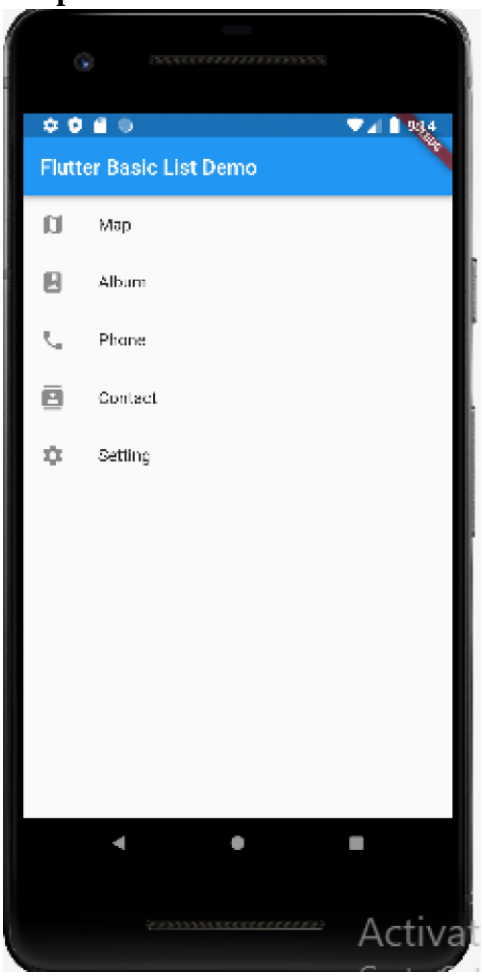
Practical 7

Aim: demonstrate Flutter different types of List views

Basic List
Code:

```
import 'package: flutter/material.dart';
void main() => runApp (MyApp());
class MyApp extends StatelessWidget { @override
Widget build(BuildContext context) {
final appTitle = 'Flutter Basic List Demo';
return Material App( title: appTitle,
home: Scaffold appBar: AppBar (
title: Text(appTitle), ), // AppBar
body: ListView(
children: <Widget>[
ListTile(
leading: Icon(Icons.map),
title: Text('Map'),
), // ListTile
ListTile(
leading: Icon(Icons.photo_album),
title: Text('Album'),
// ListTile
ListTile(
leading: Icon(Icons.phone),
title: Text('Phone'),
// ListTile
ListTile(
leading: Icon(Icons.contacts),
title: Text('Contact'),2. // ListTile
ListTile(
leading: Icon(Icons.settings),
title: Text('Setting'),
// ListTile
], //<Widget>[] ), // ListView ), // Scaffold
): // Material App
```

Output:



Grid View List

Code:

```
import 'package: flutter/material.dart';
void main() {
}
runApp(const MyApp());
class MyApp extends StatelessWidget { const MyApp({super.key});
}
@override
```

```
Widget build(BuildContext context) {
}
const title = 'Grid List';
return MaterialApp(
title: title,
home: Scaffold(
appBar: AppBar (
title: const Text(title),
), // AppBar
body: GridView.count(
// Create a grid with 2 columns. If you change the scrollDirection to // horizontal, this produces 2 rows.
crossAxisCount: 2,
// Generate 100 widgets that display their index in the List. children: List.generate (100, (index) {
return Center (
child: Text(
'Item $index',
style: Theme.of(context).textTheme.headline5,
), // Text
); // Center
}), // List.generate ), // GridView.count
), // Scaffold
); // Material App
```

Output:

Horizontal List
Code:

```
import 'package:flutter/material.dart';

void main() => runApp (const MyApp();

class MyApp extends Stateless widget { Const MyApp({super.key});

@override
```

```
widget build(BuildContext context) {
}
const title = 'Horizontal List'; return MaterialApp(
title: title.
-home: Scaffold
appBar: AppBar(
title: const Text (title).
). // AppBar
-body: Container(
margin: const Edge Insets.symmetric (vertical: 20.0).
height: 200.0.
child: ListViewC
// This next line does the trick.
scrollDirection: Axis.horizontal, children: <widget>[
Container(
width: 160.0.
color: Colors.red.
). // Container
Container(
width: 160.0.
color: Colors.blue. ). // Container Container(
width: 160.0.
color: colors.green. ). // Container Container(
width: 160.0.
color: Colors.yellow.
). // Container Container(
width: 160.0.
color: Colors.orange. ). // Container
```

1. // <widget>[]). // ListView). // Container

). // Scaffold); // MaterialApp

Output:

Practical 8

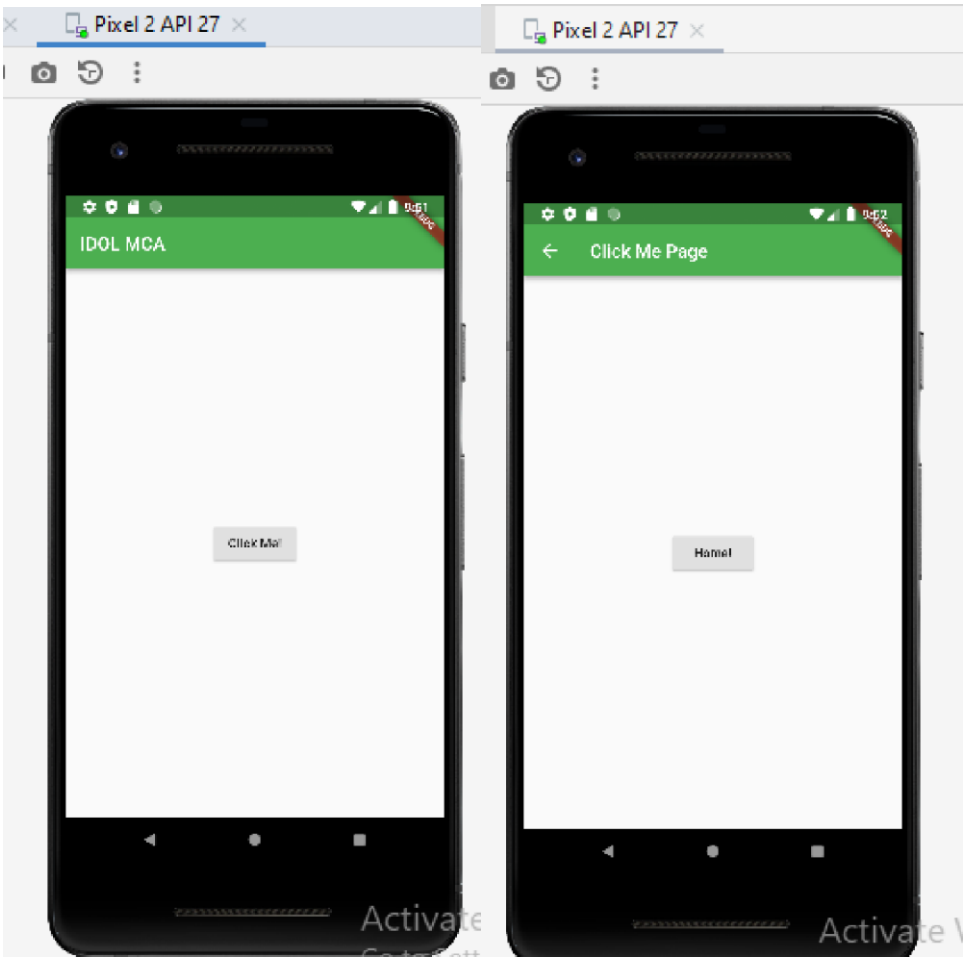
Aim: Demonstrate page navigation in flutter to show multiple views in one app

Code:

```
import 'package:flutter/material.dart';
void main()
runApp MaterialApp(home: HomeRoute().)); // MaterialApp
class HomeRoute extends Stateless widget {
@override
Widget build(BuildContext context) {
return Scaffold(appBar: AppBar(
title: Text("IDOL MCA").
backgroundColor: Colors.green,
). // AppBar
body: Center(child: RaisedButton(
child: Text('click Me!').
onPressed: () {Navigator.push( context.);
}
MaterialPageRoute(builder:
(context) => SecondRoute(). // MaterialPageRoute
). // RaisedButton
). // Center
}
);
// Scaffold
class SecondRoute extends Stateless widget {
@override
widget build(BuildContext context) {
return Scaffold(appBar: AppBar(
```

```
title: Text("click Me Page").
backgroundColor: Colors.green.
). // AppBar
body: Center(child: RaisedButton(
onPressed: {
}.
Navigator.pop(context);
child: Text C'Home!").
). // RaisedButton
). // Center
); // Scaffold
```

Output:



Practical 9

Aim: Demonstrate dart basic syntax print, String Interpolation, Arithmetic Operations in dart programming.

Print

Code: print.dart file

```
void main(){
   print("Welcome to Dart World");
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Welcome to Dart World

C:\Users\ADMIN\Desktop>
```

String Interpolation

Code: stringInt.dart file

```
void main(){
   var name="Student";
   var rollNo = 21;
print("My name is ${name} and roll number is ${rollNo}");
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
My name is Student and roll number is 21

C:\Users\ADMIN\Desktop>
```

Arithmetic Operation

Code print.dart file

```
import 'dart.io';
```

```
void main() {
  // Prompt the user for the first number
  stdout.write("Enter the first number: ");
  double? num1 = double.tryParse(stdin.readLineSync()!);

  // Prompt the user for the second number
  stdout.write("Enter the second number: ");
  double? num2 = double.tryParse(stdin.readLineSync()!);

  if (num1 == null || num2 == null) {
    print("Invalid input. Please enter valid numbers.");
    return;
  }

  // Perform arithmetic operations
  double addition = num1 + num2;
  double subtraction = num1 - num2;
  double multiplication = num1 * num2;
  double division = num2 != 0 ? num1 / num2 : double.nan; // Avoid division by zero

  // Print the results
  print("\nResults:");
  print("Addition: $addition");
  print("Subtraction: $subtraction");
  print("Multiplication: $multiplication");
  print("Division: ${division.isNaN ? "Cannot divide by zero" : division}");
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Enter the first number: 10
Enter the second number: 2

Results:
Addition: 12.0
Subtraction: 8.0
Multiplication: 20.0
Division: 5.0

C:\Users\ADMIN\Desktop>
```

Practical 10

Aim: Demonstrate if else, switch case and loop operations in dart

If..Else

Code:

```
void main() {
  int number = 10;


  if (number > 0) {
    print("The number is positive");
  } else if (number < 0) {
    print("The number is negative");
  } else {
    print("The number is zero");
  }
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
The number is positive

C:\Users\ADMIN\Desktop>
```

Switch Case

Code:

```
void main() {
  String day = "Monday";
  switch (day) {
    case "Monday":
      print("Start of the week!");
      break;
```

```
    case "Friday":
      print("Weekend is near!");
      break;
    case "Sunday":
      print("It's a holiday!");
      break;
    default:
      print("Just another day!");
  }
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Start of the week!

C:\Users\ADMIN\Desktop>
```

For Loop

Code:

```
void main() {
  for (int i = 1; i <= 5; i++) {
    print("Number: $i");
  }
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5

C:\Users\ADMIN\Desktop>
```

While Loop

Code:

```
void main() {
  int counter = 1;
  while (counter <= 5) {
    print("Counter: $counter");
    counter++;
  }
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5

C:\Users\ADMIN\Desktop>
```

Do-While Loop

Code:

```
void main() {
  int counter = 1;
  do {
    print("Counter: $counter");
    counter++;
  } while (counter <= 5);
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5

C:\Users\ADMIN\Desktop>
```

For-Each Loop

Code:

```
void main() {
  List<String> fruits = ["Apple", "Banana", "Cherry"];


  for (String fruit in fruits) {
    print("Fruit: $fruit");
  }
}
```

Output:

```
C:\Users\ADMIN\Desktop>dart print.dart
Fruit: Apple
Fruit: Banana
Fruit: Cherry

C:\Users\ADMIN\Desktop>
```