Experiment 1

Title: Install Android Studio with Latest Configuration in your System.

Aim: Installation Steps

1. Install Android Studio on Windows

a) Open the folder where you downloaded and saved the Android Studio installation file.

b) Double-click the downloaded file.

c) If you see a User Account Control dialog about allowing the installation to make changes to your computer, click Yes to confirm the installation. (Figure 1)

d) Click Next to start the installation. (Figure 2)

e) Accept the default installation settings for all steps.

f) Click Finish when the installation is done to launch Android Studio. (Figure 3)

g) Choose your preference of light or dark theme when Android Studio first launches. Screenshots in this course use the light theme, but choose whichever one you prefer. (Figure 4)

h) During the installation, the setup wizard downloads and installs additional components and tools needed for Android app development. This may take some time depending on your internet speed. During this time, you may see a User Account Control dialog for Windows

i) You may also receive a Windows Security Alert about adb.exe. Click Allow Access,if needed, to continue the installation. (Figure 6)

j) When the download and installation completes, click Finish. (Figure 7)

Screenshots:

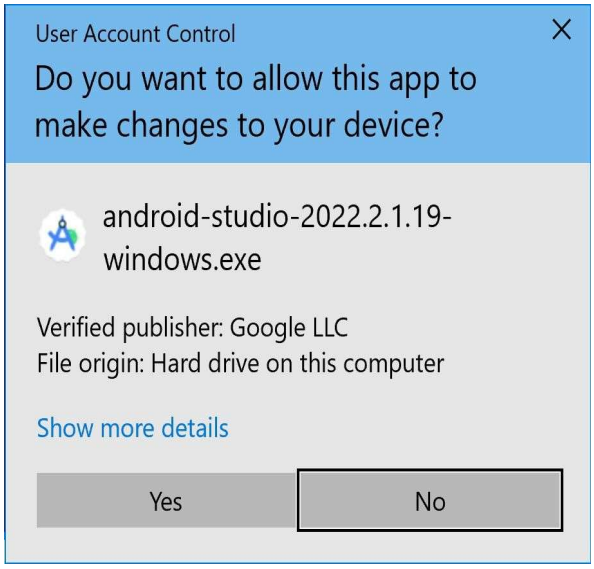


Figure 1: security pop up box

Figure 2: installation steps

Figure 3: completion of installation

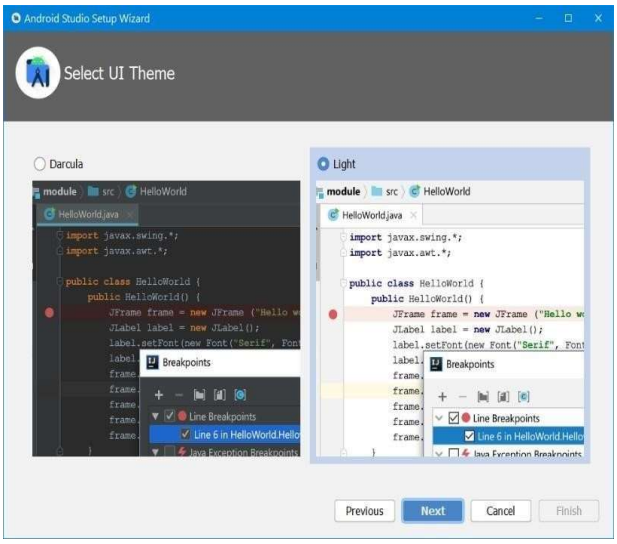


Figure 4: select ui of android studio

Figure 5: give the access

Figure 6: allow access

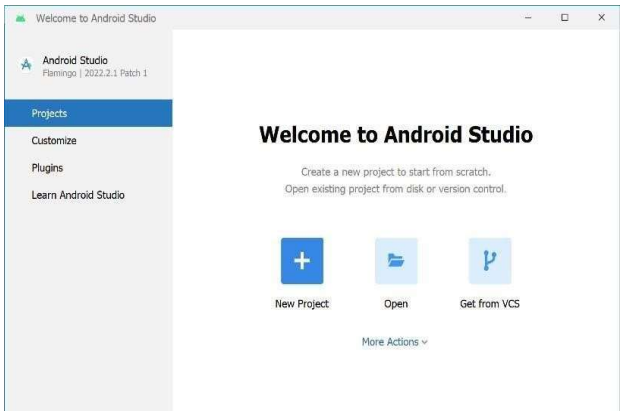


Figure 7: starting android studio

2.Introduction to Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android App development. It is a powerful tool that allows developers to build high-quality applications for the Android platform. It has complete tools for the process of Android App development. From writing code to testing and deployment, Android studio has all the functionalities for developers to develop an Android App.

3.Manifests File:

AndroidManifest.xml

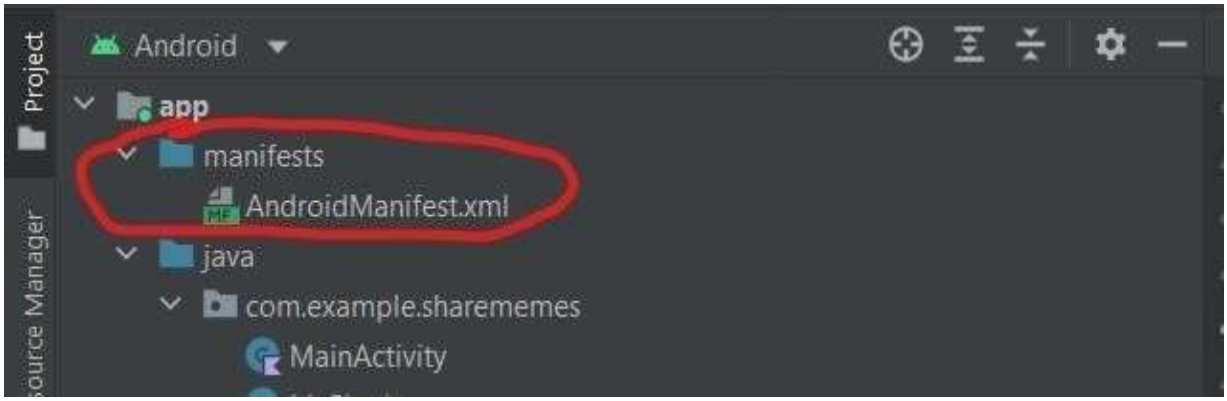


Figure 8: Manifests file structure

b) Java Folder:

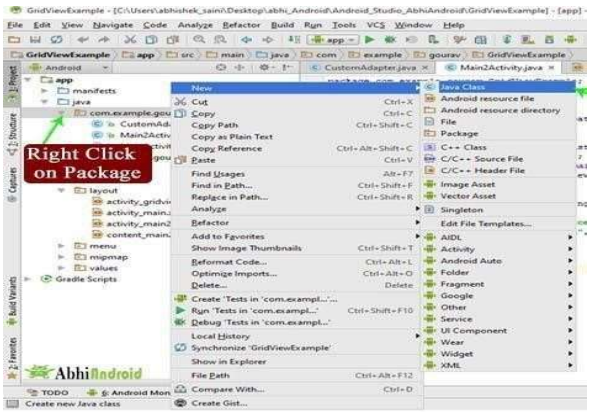


Figure 9:create steps of java file

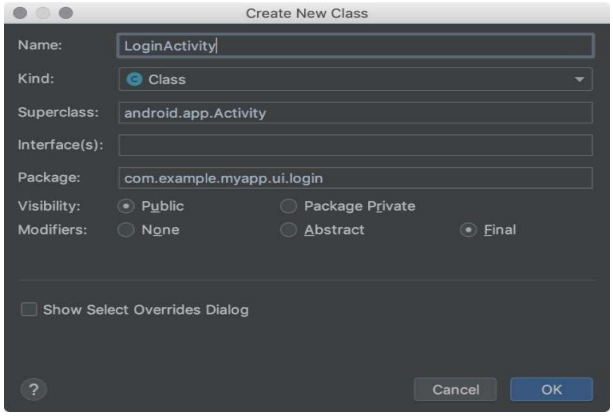


Figure 10: file name

c) Gradel File:

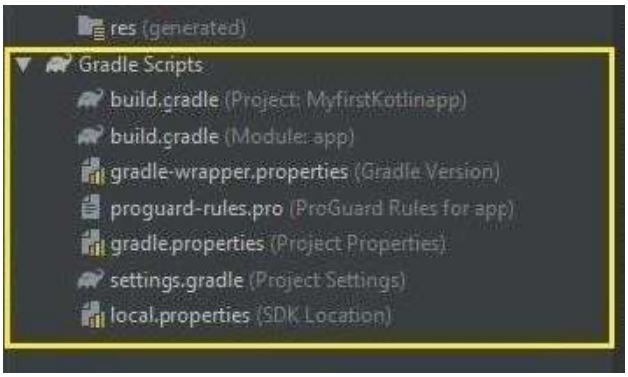


Figure 11: gradle file structure

The Gradle file is a configuration script used in Android projects to manage dependencies, build settings, and tasks. It defines project structure, dependencies, and plugins, allowing automated build processes and version control. The build.gradle file is essential for compiling the project, packaging the APK, and integrating third-party libraries or tools.

d) Res Folder:

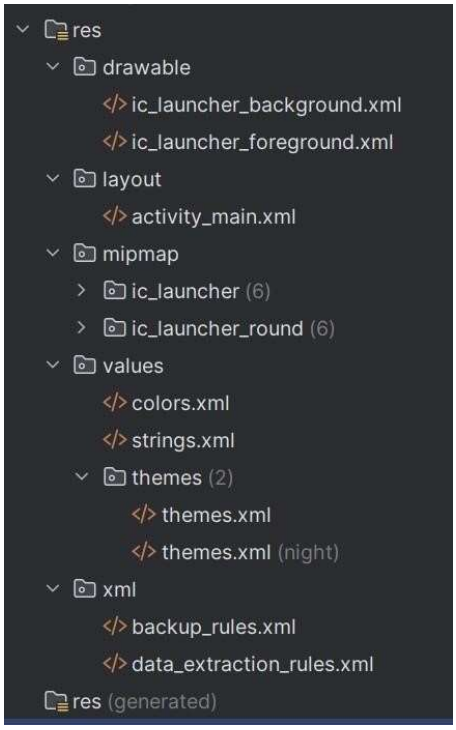


Figure 12: resource file structure

2. Steps to enable developer mode

a) Open Settings on your Android device and go to the About phone section. (Figure 13)

b) Locate the phone's build number and tap on it multiple times until you see a message regarding developer options and how many times you have to tap to enable it. On Samsung devices, you need to tap on Software information. (Figure 14, 15, 16)

c) After tapping the required amount of times, you will see a message saying that the developer options are on. You may also have to enter your phone's security code.

d) Now go back to your phone Settings and go to System. Scroll down to the bottom, and you will find the Developer options. (Figure 17)

e) Turn on USB debugging and wireless debugging, make sure your laptop and mobile with same network connection.

Screenshots:

Figure 13

Figure 14

Figure

Figure 16

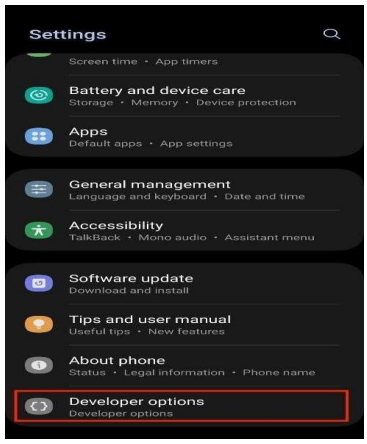


Figure 17

Figure 18

15

4.Application Installation Methods

There are 3 methods given below :

a) APK Installation

The APK (Android Package Kit) installation method involves downloading the app's APK file directly onto the user's device and manually installing it. This method is useful when the app is not available on official app stores like Google Play Store.

b) Wired Installation

Wired installation involves connecting the user's Android device to a computer via a USB cable and transferring the app directly from the computer to the device. This method is commonly used by developers or for sideloading apps.

c) Wireless Installation

Wireless installation allows users to install the Internship EviApp without needing a physical connection to a computer. This can be done through app distribution platforms, cloud services, or direct download links.

Steps for installing :

APK Installation steps

1. **Build the APK:**
   - In Android Studio, go to Build > Build Bundle(s)/APK(s) > Build APK(s).
   - Once the build is complete, you'll find the APK file in the app/build/outputs/apk/ directory.
2. **Transfer the APK to your device:**
   - Connect your Android device to your computer via USB or use a cloud service/email totransfer the APK.
3. **Install the APK:**
   - On your Android device, navigate to the location where you stored the APK file.
   - Tap on the APK file to begin the installation process.
   - You may need to enable "Unknown Sources" or "Install unknown apps" you're your device settings to allow installation.
4. **Wired Installation steps**

i). Enable Developer Options and USB Debugging:

On your Android device, go to Settings > About phone and tap Build number seven times toenable Developer Options.

Go to Settings > Developer options and enable USB debugging.

ii). Connect the device:

Connect your Android device to your computer using a USB cable. o Ensure your device is recognized by Android Studio.

iii). Run the app:

In Android Studio, click the Run button or go to Run > Run 'app'.

Select your connected device from the list and click OK. o Android Studio will build the app and install it on your device.

5. Wireless Installation steps

i). Enable Developer Options and Wireless Debugging:

On your Android device, go to Settings > About phone and tap Build number seven timesto enable Developer Options. o Go to Settings > Developer options and enable Wireless debugging (available on Android11 and above).

ii). Connect to the same network:

Ensure your Android device and your computer are connected to the same Wi-Fi network.

iii). Pair the device:

In Android Studio, go to View > Tool Windows > Device Manager.

iv). Steps to enable developer mode

a) Open Settings on your Android device and go to the About phone section. (Figure 13)

b) Locate the phone's build number and tap on it multiple times until you see a message regarding developer options and how many times you have to tap to enable it. On Samsung devices, you need to tap on Software information. (Figure 14, 15, 16)

c) After tapping the required amount of times, you will see a message saying that the developer options are on. You may also have to enter your phone's security code.

d) Now go back to your phone Settings and go to System. Scroll down to the bottom, and you will find the Developer options. (Figure 17)

e) Turn on USB debugging and wireless debugging, make sure your laptop and mobile with same network connection.

Experiment 2

Title: Study of layouts and UI components used for UI Design.

Step1: List Out UI components used for designing Layout.

1. **TextView:** Displays text to the user. It can be customized with different fonts, sizes, colors, and text styles.
2. **Button:** A clickable element used to perform an action when tapped. It can display text, icons, or both.
3. **EditText:** Allows users to input and edit text. It supports various input types like passwords, numbers, and email.
4. **ImageView:** Displays images and can be used to load images from resources, assets, or URLs with different scaling options.
5. **RecyclerView:** A flexible view for displaying large data sets in a scrollable list with customizable layouts and animations.
6. **CheckBox:** A two-state button that can be either checked or unchecked, often used in forms and selection lists.
7. **RadioButton:** Allows users to select one option from a set of mutually exclusive choices, typically used within a RadioGroup.
8. **Switch:** A two-state toggle switch that allows users to turn on/off settings or functionality with a simple gesture.
9. **ProgressBar:** Shows the progress of a task, either determinate with a specific value or indeterminate with a looping animation.
10. **Spinner:** A drop-down menu that allows users to select a single item from a list of options, similar to a combo box.
11. **SeekBar**: A slider that allows users to choose a value from a range by moving the thumb along the bar.
12. **CardView**: A container that provides a flexible, rounded-corner layout with shadow effects, often used to group content with a clean design.
13. **Toolbar**: A customizable action bar at the top of the screen that provides navigation, branding, and actions like search or settings.

Step 2 : List out all various Layouts use for designing Android Application.

1. LinearLayout

Description: Aligns its child views in a single direction, either vertically or horizontally.

Details:

- You can use android:orientation to specify the direction (vertical or horizontal).
- Views are stacked one after another in the specified orientation.
- Supports weight distribution through the android:layout_weight attribute, allowing views to take up space proportionally.

2. RelativeLayout

Description: Positions its child views relative to each other or the parent layout.

Details:

- Child views can be aligned to the left, right, top, or bottom of other views.
- You can use attributes like android:layout_alignParentTop, android:layout_toRightOf, etc., for precise positioning.
- Provides flexible layout designs, especially for complex UI where views need to overlap or position relative to one another.

3. ConstraintLayout

Description: A powerful layout that allows complex layouts with a flat hierarchy.

Details:

- Views are positioned relative to each other using constraints.
- Offers more flexibility and efficiency than RelativeLayout, with features like bias, chains, and guidelines.
- Helps reduce nested view hierarchies, leading to better performance.

4. FrameLayout

Description: A simple layout that allows child views to be stacked on top of each other.

Details:

- Typically used for displaying a single view or overlapping views.
- Child views are positioned at the top left corner by default, but can be positioned using android:layout_gravity.
- Commonly used for displaying fragments, custom drawing, or loading indicators.

5. TableLayout

Description: Organizes child views into rows and columns, similar to an HTML table.

Details:

- Each row is defined using a TableRow element.

- Columns are created automatically based on the number of views in each row.
- You can control the span of views across multiple columns using android:layout_span.

6. GridLayout

Description: Organizes child views into a grid with rows and columns.

Details:

- Offers a more flexible grid system compared to TableLayout.
- Allows precise control over row and column spans.
- Supports dynamic layouts, with views that can be positioned based on their index in the grid.

7. AbsoluteLayout *(Deprecated)*

Description: Positions child views at absolute coordinates (x, y).

Details:

- Requires setting android:layout_x and android:layout_y for each view.
- Not recommended due to difficulties in handling different screen sizes and resolutions.

8. CoordinatorLayout

Description: A super-powered FrameLayout that provides advanced control for animations and interactions between child views.

Details:

- Commonly used as the root layout for implementing Material Design components like the collapsing toolbar.
- Supports behaviors that allow child views to react to scroll events, touch events, and window insets.

9. ScrollView

Description: A layout that allows vertical scrolling of its child views.

Details:

- Can contain only one direct child view, but that child can be a layout containing multiple views.
- Often used to handle content that is too large to fit on the screen.
- There's also a HorizontalScrollView for horizontal scrolling.

10. DrawerLayout

Description: A layout used to implement a navigation drawer, a panel that slides in from the edge of the screen.

Details:

- Typically used as the root layout for activities that include a navigation drawer.
- Contains two children: the main content view and the drawer view.
- Allows users to navigate between different sections of the app.

11. RelativeLayout

Description: Allows child views to be positioned relative to each other or the parent layout.

Details:

- Enables complex layouts without nesting, leading to better performance.
- Child views can be positioned using attributes like layout_alignParentTop, layout_centerHorizontal, etc.

12. ViewGroup

Description: A special view that can contain other views, serving as a base class for layouts.

Details:

- It's an invisible container that helps structure the UI by grouping views together.
- Handles layout measurement, drawing, and interaction with its child views.
- Custom layouts are usually built by extending ViewGroup.

13. FlexboxLayout

Description: A layout that aligns its child views similarly to the CSS Flexbox model.

Details:

- Supports flexible sizing and aligning of child views in both horizontal and vertical directions.
- Provides control over wrapping, spacing, and alignment, making it suitable for responsive designs.

Step3: Demonstrate the different Ui components.

Layout/Screen XML code:

1. **activity_main.xml**

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity2">

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Label"
        android:fontFamily="sans-serif"
        android:textStyle="bold"
        android:textColor="@color/red"
        android:textAllCaps="true"
        tools:layout_editor_absoluteX="120dp"
        tools:layout_editor_absoluteY="20dp"
        tools:ignore="MissingConstraints" />
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingRight="60dp"
```

```
        android:paddingLeft="60dp"
        tools:layout_editor_absoluteX="100dp"
        tools:layout_editor_absoluteY="80dp"
        android:text="@string/B"
        android:drawableTop="@mipmap/ic_launcher_round"
        tools:ignore="MissingConstraints" />
    <ImageView
        android:id="@+id/imgV1"
        android:contentDescription="@string/todo"
        android:src="@mipmap/ic_launcher_round"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:layout_editor_absoluteX="160dp"
        tools:layout_editor_absoluteY="230dp"
        tools:ignore="MissingConstraints" />

    <EditText
        android:id="@+id/edt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:autofillHints="Enter your name"
        android:inputType="text"
        android:minHeight="48dp"
        android:paddingHorizontal="30dp"
        android:text="@string/enterName"
        tools:ignore="LabelFor,MissingConstraints"

        tools:layout_editor_absoluteX="100dp"
        tools:layout_editor_absoluteY="320dp" />
    <CheckBox
```

```
        android:id="@+id/ckBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="100dp"
        tools:layout_editor_absoluteY="400dp" />
    <RadioButton
        android:id="@+id/rdBtn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/yes"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="222dp"
        tools:layout_editor_absoluteY="400dp" />
    <RadioGroup
        android:id="@+id/rg1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="100dp"
        tools:layout_editor_absoluteY="486dp">

        <RadioButton
            android:id="@+id/male"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/male" />
        <RadioButton
            android:id="@+id/female"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/female" />
    </RadioGroup>
    <ImageButton
        android:id="@+id/imgBtn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/Label"
        android:minWidth="50dp"
        android:minHeight="50dp"
        tools:ignore="MissingConstraints,TouchTargetSizeCheck"
        tools:layout_editor_absoluteX="150dp"
        tools:layout_editor_absoluteY="627dp"
        android:src="@mipmap/ic_launcher_round"
        />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="Intent"
        android:text="Second"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="272dp"
        tools:layout_editor_absoluteY="627dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

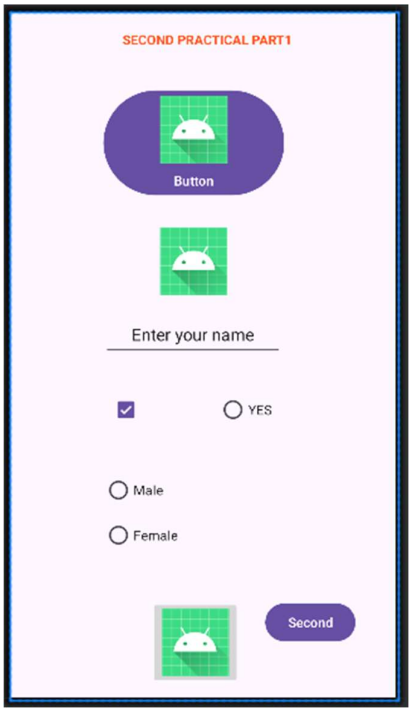
Class File/Java code:

MainActivity.java

```
package com.example.myapplication;
```

```
import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener
       (findViewById(R.id.main), (v, insets) -> {
                   Insets systemBars =
       insets.getInsets(WindowInsetsCompat.Type.systemBars());
                   v.setPadding(systemBars.left, systemBars.top,
       systemBars.right, systemBars.bottom);
                   return insets;
            });
        }

    }
```

Screenshots:



First Screen

Step3: Demonstrate the different Layout.

Layout/Screen XML code:

Second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity">
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginTop="100dp"
        android:layout_marginStart="40dp"
        android:padding="30dp"
        android:layout_alignParentTop="true"
        >
        <EditText
            android:id="@+id/edt1"
            android:layout_width="283dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="@string/first"
            android:inputType="number"
            android:minHeight="48dp"
            android:padding="10dp" />
        <EditText
            android:id="@+id/etd2"
            android:layout_width="283dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="@string/second"
            android:inputType="number"
            android:minHeight="48dp"
            android:padding="10dp" />

        <TextView
```

```
            android:id="@+id/tv1"
            android:layout_width="280dp"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:text="@string/res"
            android:textSize="25sp"
            android:textStyle="bold"
            />
    </LinearLayout>
    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/constraintLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/linearLayout"
        android:layout_marginTop="100dp">
        <Button
            android:id="@+id/btn1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:text="@string/add"
            app:layout_constraintEnd_toStartOf="@+id/btn2"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintWidth_percent="0.4"/>
        <Button
            android:id="@+id/btn2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:text="@string/sub"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/btn1"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintWidth_percent="0.4"/>
        <Button
            android:id="@+id/btn3"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:text="@string/mul"
            app:layout_constraintEnd_toStartOf="@+id/btn4"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/btn1"
```

```
app:layout_constraintWidth_percent="0.4"/>
        <Button
            android:id="@+id/btn4"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:text="@string/div"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/btn3"
            app:layout_constraintTop_toBottomOf="@+id/btn2"
            app:layout_constraintWidth_percent="0.4"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
</RelativeLayout>
```

Class File/Java code:

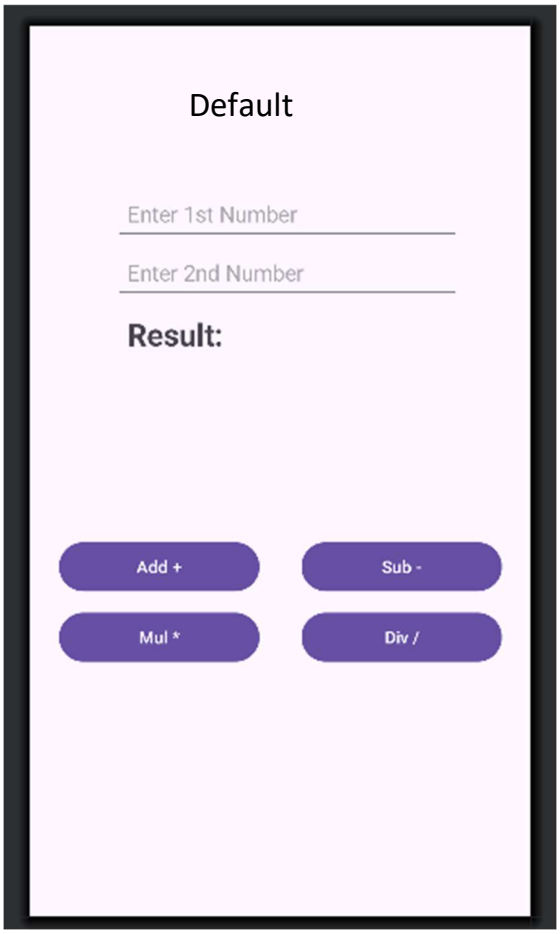
2. SecondActivity.java

```
package com.example.myapplication;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

    private EditText editText1, editText2;
    private TextView textViewResult;
    private Button buttonAdd, buttonSub, buttonMul, buttonDiv;

    @Override


protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.calculator);

        // Initialize UI elements
        editText1 = findViewById(R.id.editText1);
        editText2 = findViewById(R.id.editText2);
        textViewResult = findViewById(R.id.textView3);
        buttonAdd = findViewById(R.id.button1);
        buttonSub = findViewById(R.id.button2);
        buttonMul = findViewById(R.id.button3);
        buttonDiv = findViewById(R.id.button4);

        // Add button click listeners
        buttonAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                calculate('+');
            }
        });

        buttonSub.setOnClickListener(new View.OnClickListener() {
            @Override
```

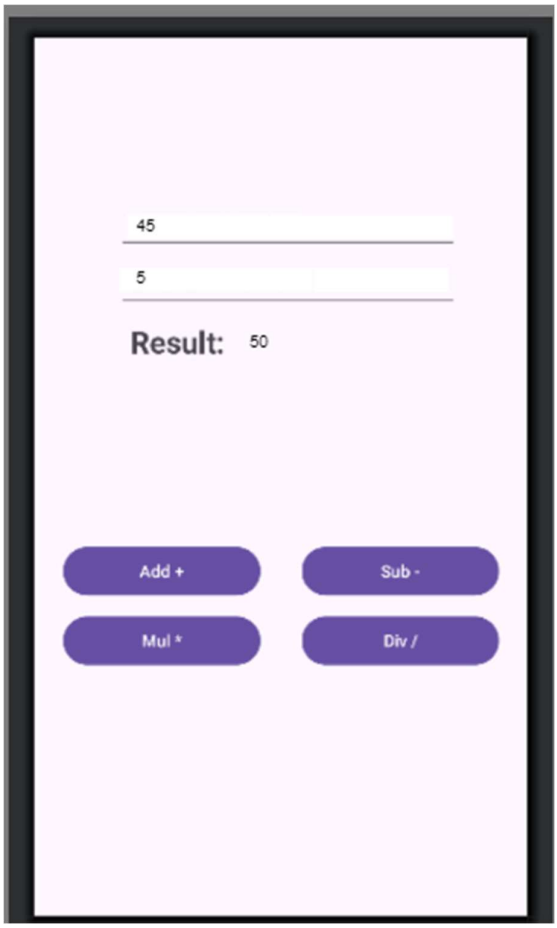
```
            public void onClick(View v) {
                calculate('-');
            }
        });

        buttonMul.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                calculate('*');
            }
        });

        buttonDiv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                calculate('/');
            }
        });
    }

    private void calculate(char operator) {String value1 =
editText1.getText().toString();
        String value2 = editText2.getText().toString();

        if (value1.isEmpty() || value2.isEmpty()) {
            textViewResult.setText("Please enter numbers");
            return;
        }

        double num1 = Double.parseDouble(value1);
        double num2 = Double.parseDouble(value2);
        double result = 0;

        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
```

```
if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    textViewResult.setText("Cannot divide by
zero");
                    return;
                }
                break;
        }

        textViewResult.setText(String.valueOf(result));
    }
}
```

Screenshots:



Default



Add