Bharati Vidyapeeth
(Deemed to be University)
College of Engineering, Pune
Department of Computer Engineering

Lab Manual

Mobile Architecture and Programming

B.Tech. Computer Sem VI

VISION OF THE DEPARTMENT

To pursue and excel in the endeavor for creating globally recognized computer engineers through quality education.

MISSION OF THE DEPARTMENT

- To impart engineering knowledge and skills conforming to dynamic curriculum.
- To develop professional, entrepreneurial and research competencies encompassing continuing intellectual growth.
- To produce qualified graduates exhibiting societal and ethical responsibilities in working environment.

Program Educational objectives

The B. Tech Computer Engineering Programme graduates upon completion of this Programme will able to:

1. Demonstrate technical and professional competencies by applying engineering fundamentals, computing principles and technologies.
2. Learn, Practice, and grow as skilled professionals adapting to the evolving computing landscape.
3. Demonstrate professional attitude, ethics, understanding of social context and interpersonal skills leading to a successful career.

Program Outcomes

a) To apply knowledge of computing and mathematics appropriate to the domain.
b) To logically define, analyse and solve real world problems
c) To apply design principles in developing hardware/software systems of varying complexity that meet the specified needs
d) To interpret and analyse data for providing solutions to complex engineering problems
e) To use and practice engineering and IT tools for professional growth
f) To understand and respond to legal and ethical issues involving the use of technology for societal benefits
g) To develop societal relevant projects using available resources
h) To exhibit professional and ethical responsibilities
i) To work effectively as an individual and a team member within the professional environment
j) To prepare and present technical documents using effective communication skills
k) To demonstrate effective leadership skills throughout the project management life cycle
l) To understand the significance of lifelong learning for professional development

Course Name: - Mobile Architecture and Programming

Hours Per Week: 2 Hrs.

Course Outcomes

1. Demonstrate foundational knowledge of mobile computing principles
2. Describe the fundamentals of mobile telecommunication system
3. Analyse the mechanism of mobile IP to support mobility
4. Compare TCP mechanisms in mobile environment
5. Develop Android mobile applications using industry standard frameworks
6. Design secure M- commerce applications

.

CO-PO mapping

PO →	1	2	3	4	5	6	7	8	9	10	11	12	PSO1	PSO2
CO1	3												3	
CO2	3					2							3	
CO3	3	2				2							3	
CO4	3	2				2							3	
CO5	3	3	3	3	3	3			3	3		3	3	2
CO6	3	3	2	2	3	3			3	3		3	3	2

Lab Requirements:

• Software: Android Studio, Java/Kotlin SDK, Emulator or physical Android device.

• Hardware: Computer with minimum 8GB RAM, Android-compatible smartphone (optional).

LIST OF PRACTICAL Assignments:

No	Detail of Assignment
1.	Create an application to print the multiplication table of any number entered by the user.
2.	Create a student registration form (Use 'hints' instead of 'Labels').
3.	Create an application that makes use of Shared Preferences to save and retrieve data in form of key-value pairs.
4.	Create an application that creates a notification after clicking a button.
5.	Create an application to take temperature as input from the user in degrees Celsius(°C) and display it in Fahrenheit(°F).
6.	Create an application using linear Layout(Horizontal/Vertical) enabling scrollable Images/Text.
7.	Create an application that will display toast (Message) on specific interval of Time.
8.	Create an application using list View to display a scrollable grocery shopping list.
9.	Create a Multi-screen application using intents.
10.	Creating Splash screen using handler in Android

Instructions to Install and configure java development kit (JDK), Android studio and android SDK.

Android is based on Linux with a set of native core C/C++ libraries. Android applications are written in Java. However, they run on Android's own Java Virtual Machine, called Dalvik Virtual Machine (DVM) (instead of JDK's JVM) which is optimized to operate on the small and mobile devices. SDK provides a selection of tools required to build Android apps or to ensure the process goes as smoothly as possible. Whether creating an app with Java, Kotlin or C#, SDK should run on an Android device and access unique features of the OS.

- Step 1 - Setup Java Development Kit (JDK) You can download the latest version of Java JDK from Oracle's Java site: Java SE Downloads. You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally, set PATH and JAVA_HOME environment variables to refer to the directory that contains java and javac, typically java_install_dir/bin and java_install_dir respectively. If you are running Windows and have installed the JDK in C:\jdk1.6.0_15, you would have to put the following line in your C:\autoexec.batfile. set PATH=C:\jdk1.6.0_15\bin;%PATH% set JAVA_HOME=C:\jdk1.6.0_15

- Step 2 - Setup Android SDK You can download the latest version of Android SDK from Android's official website: http://developer.android.com/sdk/index.html. If you are installing SDK on Windows machine, then you will find ainstaller_rXXwindows.exe, so just download and run this exe which will launch Android SDK Tool Setup wizard to guide you throughout the installation, so just follow the instructions carefully. Finally, you will have Android SDK Tools installed on your machine. If you are installing SDK either on Mac OS or Linux, check the instructions provided along with the downloaded android-sdk_rXX-macosx.zip file for Mac OS and androidsdk_rXX-linux.tgz file for Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 7 operating system.

- Step 3 - Setup Android Development Tools (ADT) Plugin This step will help you in setting Android Development Tool plugin for Eclipse. Let's start with launching Eclipse and then, choose Help > Software Updates > Install New Software. This will display the following dialogue box.

- Step 4 - Create Android Virtual Device to test your Android applications you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager using Eclipse menu options Window > AVD Manager> which will launch Android AVD Manager. Use New button to create a new Android Virtual Device and enter the following information, before clicking Create AVD button.

Note:- **https://www.geeksforgeeks.org/how-to-clone-android-project-from-github-in-android-studio/**

EXPERIMENT NO: 1

Aim: - Create an application to print the multiplication table of any number entered by the user.

- Input field to enter a number.
- Button to generate the multiplication table.
- TextView to display the result.

UI Components: `EditText`, `Button`, `TextView` or `ListView`

Concepts Used: Input Handling, Loops, List Population

Step-by-step Android App (Java)

1. Create a new Android Project

Open Android Studio → New Project → Empty Activity

Project Name: MultiplicationTableApp

Main Activity: MainActivity

2. Design the Layout (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="24dp"
    android:gravity="center_horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/etNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter a number"
        android:inputType="number" />

    <Button
        android:id="@+id/btnGenerate"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Generate Table"
        android:layout_marginTop="16dp" />

    <ListView
        android:id="@+id/listViewTable"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp" />
</LinearLayout>
```

Step 3: Add Logic in Java File (MainActivity.java)

```
package com.example.multiplicationtable;

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 MainActivity extends AppCompatActivity {

    EditText inputNumber;
    Button generateButton;
    TextView tableOutput;

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

        inputNumber = findViewById(R.id.inputNumber);
        generateButton = findViewById(R.id.generateButton);
        tableOutput = findViewById(R.id.tableOutput);

        generateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String input = inputNumber.getText().toString().trim();

                if (!input.isEmpty()) {
                    int number = Integer.parseInt(input);
                    StringBuilder table = new StringBuilder();
```

```
                for (int i = 1; i <= 10; i++) {
                    table.append(number).append(" x ").append(i)
                            .append(" = ").append(number * i).append("\n");
                }
                tableOutput.setText(table.toString());
            } else {
                tableOutput.setText("Please enter a valid number.");
            }
        }
    });
  }
}
```

Output:

- User enters a number (e.g., 5)
- Presses **"Generate Table"**
- ListView displays:

```
5 x 1 = 5
5 x 2 = 10
...
5 x 10 = 50
```

Note:-

- The input from the user is validated.
- A for loop generates the multiplication table.
- The results are displayed in a ListView using an ArrayAdapter.

Conclusion:

Built an Android app that prints a **multiplication table** based on **user input**, showcasing input handling and dynamic list generation.

EXPERIMENT NO:2

Aim: - To design and implement a Student Registration Form in Android using EditText components with hint texts instead of TextView labels.

Procedure:

1. Create a New Project

- Open **Android Studio** → New Project → Empty Activity.
- Name it StudentRegistrationForm.

2. Design the UI (activity_main.xml)

Use EditText with the hint attribute. Use **LinearLayout**:

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

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <EditText
            android:id="@+id/etFullName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Full Name" />

        <EditText
            android:id="@+id/etEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email Address"
            android:inputType="textEmailAddress" />

        <EditText
            android:id="@+id/etPhone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Phone Number"
            android:inputType="phone" />
```

```
<EditText
   android:id="@+id/etAddress"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:hint="Home Address" />

<EditText
   android:id="@+id/etCourse"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:hint="Desired Course" />

<EditText
   android:id="@+id/etDob"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:hint="Date of Birth"
   android:inputType="date" />

<Spinner
   android:id="@+id/spinnerGender"
   android:layout_width="match_parent"
   android:layout_height="wrap_content" />

<EditText
   android:id="@+id/etNotes"
   android:layout_width="match_parent"
   android:layout_height="100dp"
   android:hint="Additional Notes"
   android:gravity="top"
   android:inputType="textMultiLine" />

 <Button
    android:id="@+id/btnSubmit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Register" />

  </LinearLayout>
</ScrollView>
```

3. Add Spinner Data (MainActivity.java or MainActivity.kt)

4. Handle Button Click (Optional Submission Logic)

```
Button submit = findViewById(R.id.btnSubmit);
submit.setOnClickListener(v -> {
   String name = ((EditText)findViewById(R.id.etFullName)).getText().toString();
   // Retrieve other fields similarly
   Toast.makeText(this, "Registration Successful!", Toast.LENGTH_SHORT).show();
});
```

Output:

A clean, scrollable Android form where users fill in:

- Full Name
- Email
- Phone Number
- Address
- Desired Course
- Date of Birth
- Gender (Spinner)
- Additional Notes

Each field has a **hint** for clarity instead of a label.

Conclusion:

Created a **student registration form** using Android's EditText with **hints**, achieving a clean and modern UI without traditional labels.

EXPERIMENT NO:3

Aim: - Create an application that makes use of Shared Preferences to save and retrieve data in form of key-value pairs.

Procedure:

1. Create a New Android Project

- Open Android Studio → New Project → Empty Activity.
- Name it SharedPrefsApp.

2. Design the Layout (activity_main.xml)

```
<?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="24dp">
    <EditText
        android:id="@+id/etName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name" />
    <Button
        android:id="@+id/btnSave"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Save Name" />
    <Button
        android:id="@+id/btnRetrieve"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Retrieve Name" />
    <TextView
        android:id="@+id/tvResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Saved Name Will Appear Here"
        android:textSize="18sp"
        android:paddingTop="16dp" />
</LinearLayout>
```

3. Write Java Code (MainActivity.java)

```
package com.example.sharedprefsapp;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    EditText etName;
    Button btnSave, btnRetrieve;
    TextView tvResult;
    public static final String PREFS_NAME = "MyPrefsFile";
    public static final String KEY_NAME = "username";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etName = findViewById(R.id.etName);
        btnSave = findViewById(R.id.btnSave);
```

```
        btnRetrieve = findViewById(R.id.btnRetrieve);
        tvResult = findViewById(R.id.tvResult);
        SharedPreferences sharedPreferences = getSharedPreferences(PREFS_NAME,
MODE_PRIVATE);
        btnSave.setOnClickListener(v -> {
            String name = etName.getText().toString();
            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.putString(KEY_NAME, name);
            editor.apply();
            Toast.makeText(this, "Name saved!", Toast.LENGTH_SHORT).show();
        });
        btnRetrieve.setOnClickListener(v -> {
            String name = sharedPreferences.getString(KEY_NAME, "No name found");
            tvResult.setText("Retrieved: " + name);
        });
    }
}
```

Output:

- When the user enters a name and taps **Save**, the app stores it in **SharedPreferences**.
- On tapping **Retrieve**, the app fetches and displays the saved name using the key "username".

Conclusion:

Created an Android application using SharedPreferences to store and retrieve data in a key-value pair format.

EXPERIMENT NO: 4

Aim: - Create an application that creates a notification after clicking a button.

Procedure:

1. Create a New Android Project

- Open Android Studio → New Project → Empty Activity.
- Name it NotificationApp.

2. Design the Layout (activity_main.xml)

```
<?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:padding="24dp"
    android:gravity="center"
    android:orientation="vertical">
    <Button
        android:id="@+id/btnNotify"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Notification" />
</LinearLayout>
```

3. Add Notification Code (MainActivity.java)

```
package com.example.notificationapp;
import android.app.*;
import android.content.Context;
import android.os.Bundle;
```

```
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;

public class MainActivity extends AppCompatActivity {
    private static final String CHANNEL_ID = "default_channel";
    private static final int NOTIFICATION_ID = 1;

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

        Button btnNotify = findViewById(R.id.btnNotify);
        btnNotify.setOnClickListener(v -> showNotification());
    }

    private void createNotificationChannel() {
        if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
            CharSequence name = "Default Channel";
            String description = "Used for general notifications";
            int importance = NotificationManager.IMPORTANCE_DEFAULT;
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
importance);
            channel.setDescription(description);
            NotificationManager notificationManager =
getSystemService(NotificationManager.class);
```

```
            notificationManager.createNotificationChannel(channel);
        }
    }

    private void showNotification() {
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
                .setSmallIcon(android.R.drawable.ic_dialog_info)
                .setContentTitle("Button Clicked!")
                .setContentText("You just triggered a notification.")
                .setPriority(NotificationCompat.PRIORITY_DEFAULT);
        NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);
        notificationManager.notify(NOTIFICATION_ID, builder.build());
    }
}
```

Output:

When the user clicks the **"Show Notification"** button, a notification with a title and message appears in the system tray.

Conclusion:

Created an Android app that triggers a **notification on button click**, utilizing **NotificationManager** and **NotificationCompat**.

EXPERIMENT NO: 5

Aim: - Create an application to take temperature as input from the user in degrees Celsius(°C) and display it in Fahrenheit(°F).

The formula used for conversion is:
Fahrenheit = (Celsius × 9/5) + 32
This application responds to user input and performs a live unit conversion.

Procedure:

1. Create a New Android Project

- Open **Android Studio** → New Project → Empty Activity.
- Name it TemperatureConverter.

2. Design the Layout (activity_main.xml)

```
<?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:padding="24dp"

    android:orientation="vertical"

    android:gravity="center">

    <EditText

        android:id="@+id/etCelsius"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Enter temperature in °C"

        android:inputType="numberDecimal" />

    <Button
```

```
        android:id="@+id/btnConvert"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Convert to °F"
        android:layout_marginTop="16dp" />
    <TextView
        android:id="@+id/tvResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Temperature in °F will appear here"
        android:textSize="18sp"
        android:layout_marginTop="16dp" />
</LinearLayout>
```

3.Write Java Code (MainActivity.java)

```
package com.example.temperatureconverter;
import android.os.Bundle;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    EditText etCelsius;
    Button btnConvert;
    TextView tvResult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        etCelsius = findViewById(R.id.etCelsius);
        btnConvert = findViewById(R.id.btnConvert);
        tvResult = findViewById(R.id.tvResult);
        btnConvert.setOnClickListener(v -> {
            String celsiusStr = etCelsius.getText().toString();
            if (!celsiusStr.isEmpty()) {
                double celsius = Double.parseDouble(celsiusStr);
                double fahrenheit = (celsius * 9 / 5) + 32;
                tvResult.setText("Temperature in °F: " + fahrenheit);
            } else {
                Toast.makeText(this, "Please enter a value",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Output:

When the use enters a temperature in **Celsius** and taps the **Convert** button:
The corresponding **Fahrenheit** value is calculated and displayed.

Conclusion:

Created an Android application that accepts input in Celsius and converts it to Fahrenheit using basic UI elements and arithmetic logic.

EXPERIMENT NO: 6

Aim: - Create an application using linear Layout(Horizontal/Vertical) enabling scrollable Images/Text.

Procedure:

Step 1: Create a New Android Project

- Open Android Studio → New Project → Empty Activity
- Name the app: ScrollableContentApp

Step 2: Design Layout with Scrollable Images and Text

Example 1: Vertical Scroll (Text and Images)

File: res/layout/activity_main.xml

Example 2: Horizontal ScrollView (Image Gallery)

Step 3: Add Sample Images

- Place sample1.png, sample2.png, sample3.png into the res/drawable folder.
- Use right-click on drawable → **New > Image Asset** or paste images directly.

Step 4: No Java/Kotlin Code Required for Basic Scrolling

This UI is fully XML-driven unless you want to load text or images dynamically.

Output:

- A vertically scrollable layout containing **text and multiple images**.
- A **horizontal image gallery** that scrolls left to right.

Conclusion:

Created an Android layout using **LinearLayout** with vertical and horizontal scrolling for text and images using ScrollView and HorizontalScrollView.

EXPERIMENT NO: 7

Aim: - Create an application that will display toast (Message) on specific interval of Time.

Note:-

- Key Classes: Handler, Runnable, or TimerTask
- The use of Handler and Runnable enables timed repetition.
- Proper cleanup with removeCallbacks() avoids memory leaks.

Procedure:

Step 1: Create a New Android Project

- Open **Android Studio** → New Project → Empty Activity
- Name it: TimedToastApp

Step 2: Design Basic Layout (activity_main.xml)

Step 3: Add Toast Logic in Java (MainActivity.java) app doesn't require interactive UI, but you can include a simple message

---

Output:

- Every 5 seconds, a **Toast** message saying "This is a timed toast!" will appear on the screen.
- The cycle continues until the app is closed.

Conclusion:

You successfully built an Android application that shows a Toast message at regular intervals, demonstrating use of Handler, Runnable, and Toast.

EXPERIMENT NO: 8

Aim: - To develop an Android application using **ListView** to display a **scrollable grocery shopping list**.

- UI Component: ListView, ArrayAdapter

Procedure:

Step 1: Create a New Android Project

- Open **Android Studio** → New Project → Empty Activity
- Name the project: GroceryShoppingListApp

Step 2: Define the Layout (activity_main.xml)

In this step, define the layout with a ListView to display the shopping list.

Step 3: Create the Java Code (MainActivity.java)

Step 4: Run the Application

Once you've added the necessary Java code, click **Run** in Android Studio to start the application.

Output:

The app will display a scrollable list of items (grocery shopping list) using ListView. The list is scrollable, and the items are populated from the ArrayList in the MainActivity.java file.

Notes :

- ArrayAdapter helps in linking data (the grocery list) to the ListView.
- The list is scrollable, and each grocery item appears as a separate row.
- The default simple_list_item_1 layout is used to display each item in the list.

Conclusion:

You have successfully created an Android application that displays a **grocery shopping list** using **ListView**. This application demonstrates the use of **ArrayList**, **ArrayAdapter**, and **ListView**.

EXPERIMENT NO: 9

Aim: - To develop an Android application with **multiple screens (activities)** and navigate between them using **Intents**..

- UI Components: Button, TextView, EditText
- Key Concept: **Explicit Intents**
- Intent is used to switch between screens.
- putExtra() and getStringExtra() allow data transfer between activities.

Procedure:

Step 1: Create a New Android Project

- Open Android Studio → **New Project** → Empty Activity
- Project Name: MultiScreenApp
- Main Activity: MainActivity

Step 2: Create Layout for MainActivity (activity_main.xml)

Step 3: Create Second Activity

- Right-click java > your package name → New → **Activity > Empty Activity**
- Name: SecondActivity

Step 4: Layout for SecondActivity (activity_second.xml)

Step 5: Code for MainActivity.java

Step 6: Code for SecondActivity.java

Step 7: Update AndroidManifest.xml

Ensure SecondActivity is declared:

Output:

- **Screen 1:** User enters a message and taps "Send".
- **Screen 2:** Displays the message passed from the first screen.

Conclusion:

You successfully created a multi-screen Android application using explicit intents to pass data and navigate between screens.

EXPERIMENT NO: 10

Aim: - To develop an Android application that displays a **Splash Screen** for a few seconds using a Handler, and then navigates to the main screen.

Key Components: Handler, Intent, Activity

Note:

- Handler().postDelayed() introduces a delay before launching the next activity.
- finish() ensures the Splash screen doesn't remain in the back stack.

Step 1: Create a New Android Project

- Open Android Studio → **New Project** → Empty Activity
- Project Name: SplashScreenApp
- Activity Name: SplashActivity

Step 2: Create a Splash Screen Layout (activity_splash.xml)

Step 3: Create MainActivity Layout (activity_main.xml)

Step 4: SplashActivity.java

Step 5: MainActivity.java

Step 6: Update AndroidManifest.xml

Output:

- App launches with a **Splash screen** showing the app logo and name.
- After 3 seconds, it transitions to the **MainActivity**.

Conclusion:

You successfully created a **Splash Screen using a Handler** in Android and learned how to use delayed navigation with Intent.