



ESTD. 2001

# **PRATHYUSHA ENGINEERING COLLEGE**

## **(An Autonomous Institution)**

Approved by AICTE & Affiliated to Anna University, NAAC "A" Grade Accredited  
Institution, National Board of Accrediting (NBA) accredited Programmes

## **INTERNSHIP REPORT**

*A Report submitted by in partial fulfillment of the requirement  
for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**In**

**INFORMATION TECHNOLOGY**

**By**

**BHARGAVI.K**  
**(111421205004)**

**PROJECT TITLE**

**ANDROID APPLICATION DEVELOPMENT**

**Under the Supervision of**

**Mr. A. ARUL PRABAHAR**  
**The National Small Scale Industries Corporation**

**Duration: 27<sup>th</sup> June 2024 to 10<sup>nd</sup> July 2024**



ESTD. 2001

# **PRATHYUSHA ENGINEERING COLLEGE**

## **(An Autonomous Institution)**

### **BONAFIDE CERTIFICATE**

Certified that this summer internship report is done by " **BHARGAVI.K**  
**(111421205004)**" who carried out the summer internship work during the year 2024-2025  
under my supervision.

**SIGNATURE**

**Ms. J. SAIKETHANA, M.E,**  
**ASSISTANT PROFESSOR,**  
**SUPERVISOR,**

Department of Information Technology,  
Prathyusha Engineering College,  
Tiruvallur – 602 025.

**SIGNATURE**

**Dr. R. THIAGARAJAN, Ph.D**  
**ASSOCIATE PROFESSOR,**  
**HEAD OF THE DEPARTMENT,**

Department of Information Technology,  
Prathyusha Engineering College,  
Tiruvallur – 602 025.

**PLACE: THIRVALLUR**

**DATE:**

Submitted for the Summer Internship held on..... at  
**PRATHYUSHA ENGINEERING COLLEGE, Tirvallur-602025**

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

No. : 44325



एन एस आई सी  
N S I C

राष्ट्रीय लघु उद्योग निगम लिमिटेड

THE NATIONAL SMALL INDUSTRIES CORPORATION LIMITED

(A Govt. of India Enterprise)

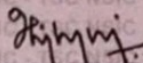
NSIC - TECHNICAL SERVICES CENTRE

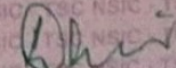
Sector B-24, Guindy Industrial Estate, Ekkaduthangal, Chennai - 600 032.

## CERTIFICATE

This is to certify that Ms. **BHARGAVI K.** D/o.  
Shri. **KUMAR**, student of final year B.Tech. (IT),  
Prathyusha Engineering College, has undergone  
**"Internship Training on ANDROID  
APPLICATION DEVELOPMENT"** conducted at  
our centre for a period of two weeks from  
27.06.2024 to 10.07.2024.



  
HEAD OF TRAINING

  
HEAD OF CENTRE

## ACKNOWLEDGEMENT

First I would like to thank Mr. A. Arul Prabahar, Deputy Manager of **The National Small Scale Industries Corporation**, for giving me the opportunity to do an internship within the organization.

I also would like all the people that worked along with me in **The National Small Scale Industries Corporation** with their patience and openness they created an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am highly indebted to Principal **Dr. R. S. KUMAR, M.TECH, Ph.D** for the facilities provided to accomplish this internship.

I would like to thank my Head of the Department **Dr. R. THIAGARAJAN, Ph.D** for his constructive criticism throughout my internship.

I would like to thank **Ms. J. SAIKETHANA, M.E** internship coordinator Department of **IT** for their support and advices to get and complete internship in above said organization.

I am extremely great full to my department staff members and friends who helped me in successful completion of this internship.

## **NATIONAL SMALL SCALE INDUSTRIES CORPORATION**

(A Government of India Enterprise)

### **ABOUT**

National Small Scale Industries Corporation (NSIC), is an ISO 9001:2015 certified Government of India Enterprise under Ministry of Micro, Small and Medium Enterprises (MSME). NSIC has been working to promote, aid and foster the growth of micro, small and medium enterprises in the country. NSIC operates through countrywide network of offices and Technical Centers in the Country. In addition, NSIC has set up Training cum Incubation Centre managed by professional manpower.



### **INTRODUCTION**

National Small Scale Industries Corporation Ltd. (NSIC), is an ISO 9001:2015 certified Government of India Enterprise under Ministry of Micro, Small and Medium Enterprises (MSME). NSIC has been working to fulfil its mission of promoting, aiding and fostering the growth of small industries and industry related micro, small and medium enterprises in the country. Over a period of five decades of transition, growth and development, NSIC has proved its strength within the country and abroad by promoting modernization, upgradation of technology, quality consciousness, strengthening linkages with large medium enterprises and enhancing exports - projects and products from small enterprises. NSIC operates through countrywide network of offices and Technical Centres in the Country. In addition, NSIC has set up Training cum incubation Centre & with a large professional manpower, NSIC provides a package of services as per the needs of MSME sector.

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	LIST OF FIGURES	8
	ABSTRACT	9
1	INTRODUCTION	10
2	MOBILE APPS PROGRAMS	
	HELLO WORLD	14
	TIMER	15
	LIST VIEW	18
	CHECK BOX	20
	IMAGE APP	22
	IMAGE SWITCHING	24
	CALCULATOR	26
	LOGIN PAGE	28
	WEB VIEW	30
	MUSIC PLAYER	32
	AUTO COMPLETE	34
	INCREMENT & DECREMENT	36
	SCROLL VIEW	38
	CALLER APP1	40

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>2.15 FRAGMENT</b>	<b>42</b>
	<b>2.16 GUESS GAME</b>	<b>44</b>
	<b>2.17 CALENDER</b>	<b>46</b>
	<b>2.18 MUSIC SWITCHING</b>	<b>47</b>
	<b>2.19 TIC TAC TOE</b>	<b>48</b>
	<b>2.20 VIDEO PLAYER</b>	<b>49</b>
	<b>2.21 QUIZ GAME</b>	<b>50</b>
	<b>2.22 PDF VIEWER</b>	<b>52</b>
<b>3</b>	<b>SUGRESSIONS AND FEEDBACKS</b>	<b>54</b>
<b>4</b>	<b>CONCLUSION</b>	<b>55</b>



## LIST OF FIGURES

FIGURENO	TITLE	PAGE NO
1.1	EVOLUTION OF THE ANDROID	11
1.2	ICON OF THE ANDROID STUDIO	12
2.1.1	OUTPUT OF HELLOWORLD	14
2.2.1	OUTPUT OF TIMER	17
2.3.1	OUTPUT OF LIST VIEW	18
2.4.1	OUTPUT OF CHECK BOX	20
2.5.1	OUTPUT OF IMAGE APP	22
2.6.1	OUTPUT OF IMAGE SWITCHING	24
2.7.1	OUTPUT OF CALCULATOR	26
2.8.1	OUTPUT OF LOGIN PAGE	28
2.9.1	OUTPUT OF WEB VIEW	31
2.10.1	OUTPUT OF MUSIC PLAYER	32
2.11.1	OUTPUT OF AUTO COMPLETE	34
2.12.1	OUTPUT OF INCREMENT &DECREMENT	36
2.13.1	OUTPUT OF SCROLL VIEW	38
2.14.1	OUTPUT OF CALLER APP	40
2.15.1	OUTPUT OF FRAGMENT	42
2.16.1	OUTPUT OF GUESS GAME	44
2.17.1	OUTPUT OF CALENDER	46
2.18.1	OUTPUT OF MUSIC SWITCHING	46
2.19.1	OUTPUT OF TIC TAC TOE	48
2.20.1	OUTPUT OF VIDEO PLAYER	49
2.21.1	OUTPUT OF QUIZ GAME	50
2.22.1	OUTPUT OF PDF VIEWER	52



## ABSTRACT

In the contemporary digital era, mobile applications have become integral to daily life, influencing sectors ranging from communication and entertainment to healthcare and finance. This paper explores mobile application development using Android Studio, focusing on its tools and features that streamline the creation of robust Android applications. Android Studio, the official IDE for Android development, leverages Java and Kotlin languages, providing a comprehensive suite for coding, debugging, and testing. Key components include an intuitive code editor, a powerful emulator, and integration with version control systems. The use of Gradle for automated builds and support for various device configurations enhances development efficiency. This study highlights best practices, common challenges, and solutions in Android app development, showcasing the versatility and capabilities of Android Studio in creating high-quality mobile applications.

## **CHAPTER 1**

### **INTRODUCTION**

Mobile application development in Android Studio is the process of creating software applications for Android devices using the official Integrated Development Environment (IDE) provided by Google. Android Studio, based on IntelliJ IDEA, offers a comprehensive suite of tools designed to streamline development. It supports programming languages such as Java and Kotlin, providing flexibility for developers. Key features include an advanced code editor, real-time code analysis, and intelligent code completion. The built-in emulator allows testing on various virtual devices, ensuring compatibility across different screen sizes and Android versions. Gradle, an integrated build automation tool, manages dependencies and automates the build process, enhancing efficiency. Android Studio also offers robust debugging tools, a visual layout editor, and seamless integration with version control systems like Git. These features collectively facilitate the development of high-quality, efficient, and scalable Android applications, making Android Studio a cornerstone of modern mobile app development.

### **HISTORY OF DEVELOPMENT**

Android app development began in 2003 with the founding of Android Inc., acquired by Google in 2005. The first Android device, HTC Dream, was released in 2008. Key early versions include Android 1.5 Cupcake (2009), introducing on-screen keyboards and third-party widgets, and Android 2.2 Froyo (2010), which added performance improvements and Wi-Fi hotspot functionality. Android 4.0 Ice Cream Sandwich (2011) unified the phone and tablet OS, introducing a refined UI and improved multitasking. Android 5.0 Lollipop (2014) brought Material Design, a major visual overhaul, while Android 10 (2019) introduced privacy enhancements and a system-wide dark mode. Android Studio, launched in 2013, replaced Eclipse as the primary IDE, providing a more integrated development environment. Significant updates like Android 12 (2021) with Material You and Android 13 (2022) continue to enhance customization and security. Today, Android dominates the global mobile OS market, fostering a robust app development ecosystem.

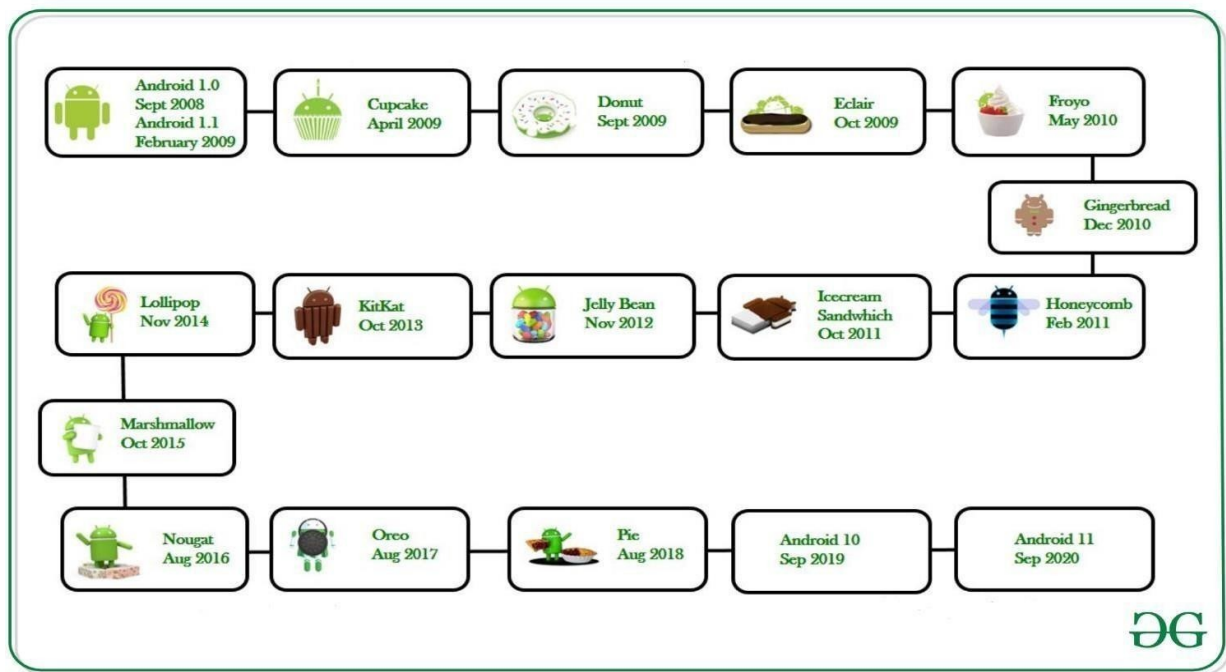


Fig 1.1 Evolution of the android

## DEVELOPMENT PLATFORMS AND TOOLS

Mobile application development hinges on robust platforms and tools that cater to the specific needs of different operating systems. The two most prominent platforms are Android and iOS, each with its unique development environment, tools, and programming languages.

### 1. Android Development

#### Development Environment:

- **Android Studio:** The official Integrated Development Environment (IDE) for Android app development. It is based on IntelliJ IDEA and offers a comprehensive suite of tools for coding, debugging, and testing.

#### Programming Language:

- **Java:** Traditionally the primary language for Android development, known for its reliability and widespread use.
- **Xml:** Is a type of markup language that establish a set of guidelines for encoding in a way that is both machine and human-readable .

## ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development, offering a powerful and user-friendly platform for developers. Built on IntelliJ IDEA, it provides a comprehensive suite of tools designed specifically for creating Android applications.

Key features of Android Studio include:

- **Advanced Code Editor:** Supports Java, Kotlin, and C++, offering syntax highlighting, code completion, and refactoring tools.
- **Visual Layout Editor:** Allows developers to design app interfaces visually, with drag-and-drop functionality for UI components.
- **Emulator:** A fast and feature-rich emulator to test apps on various Android devices and configurations.
- **Gradle Build System:** Automates the build process, manages dependencies, and supports custom build configurations.
- **Android Profiler:** Helps optimize app performance by monitoring CPU, memory, and network usage in real-time.
- **Version Control Integration:** Seamless integration with Git for version control, facilitating collaboration among team members.

Overall, Android Studio simplifies and enhances the development workflow, enabling developers to create high-quality Android apps efficiently.



Fig 1.2 Icon of the Android Studio

## **CHAPTER-2**

### **MOBILE APP PROGRAM**

#### **PROGRAM 2.1 - HELLO WORLD**

##### **SOURCE CODE:**

##### **XML CODE:**

- RelativeLayout: A layout that arranges child views relative to each other or to the parent.
- TextView: Displays text to the user and is a common widget for showing static text.  
`android:id="@+id/textView"`: Unique identifier for this view, which can be referenced in the Java (or Kotlin) code.
- `android:text="Hello World!"`: Text displayed by the TextView.
- `android:textSize="24sp"`: Text size in scaled pixels.
- `android:layout_centerInParent="true"`: Centers the TextView horizontally and vertically within its parent (RelativeLayout).

##### **JAVA CODE:**

- MainActivity: This is the main activity class that extends AppCompatActivity, which provides compatibility features for older versions of Android.
- `onCreate(Bundle savedInstanceState)`: This method is called when the activity is first created.
- `super.onCreate(savedInstanceState);`: Calls the superclass implementation of `onCreate()`.
- `setContentView(R.layout.activity_main);`: Sets the content view of the activity to the layout defined in `activity_main.xml`.
- `TextView textView = findViewById(R.id.textView);`: Finds the TextView with the id `textView` from the XML layout.
- `textView.setText("Hello World");`: Sets the text of the TextView to "Hello World".

## OUTPUT:



Fig 2.1.1 output of hello world

## PROGRAM 2.2 - TIMER

### SOURCE CODE:

#### XML CODE:

- RelativeLayout: Positions child views relative to each other or the parent.
- TextView (@+id/textViewClock): Displays the current time.
- android:textSize="48sp": Sets the text size to 48 scaled pixels.
- android:textColor="@android:color/black": Sets the text color to black.
- android:text="00:00:00": Initial text displayed (placeholder for clock display).
- android:layout\_centerInParent="true": Centers the TextView horizontally and vertically within its parent.

#### JAVA CODE:

- MainActivity: This is the main activity class that extends AppCompatActivity.

- `onCreate(Bundle savedInstanceState)`: Initializes the activity.
- `setContentView(R.layout.activity_main)`: Sets the content view to the layout defined in `activity_main.xml`.
- Initialization:
- `textViewClock`: Holds a reference to the `TextView` for displaying the clock.
- `handler`: Used to schedule and execute runnable to update the clock every second.
- `runnable`: A `Runnable` that updates the clock display (`textViewClock`) every second (1000 milliseconds).
- `updateTime()`: Method to format the current time (HH:mm:ss) and update `textViewClock`.
- `handler.post(runnable)`: Posts runnable for the first time to start updating the time.
- `updateTime()`:
- Creates a `SimpleDateFormat` instance ("HH:mm:ss") to format the current time in hours, minutes, and seconds.
- Uses `Locale.getDefault()` to get the device's default locale.
- Formats the current time (`Date()`) and sets it as the text for `textViewClock`.
- `onDestroy()`:
- Removes runnable callbacks from handler when the activity is destroyed to stop updating the time



**OUTPUT:**

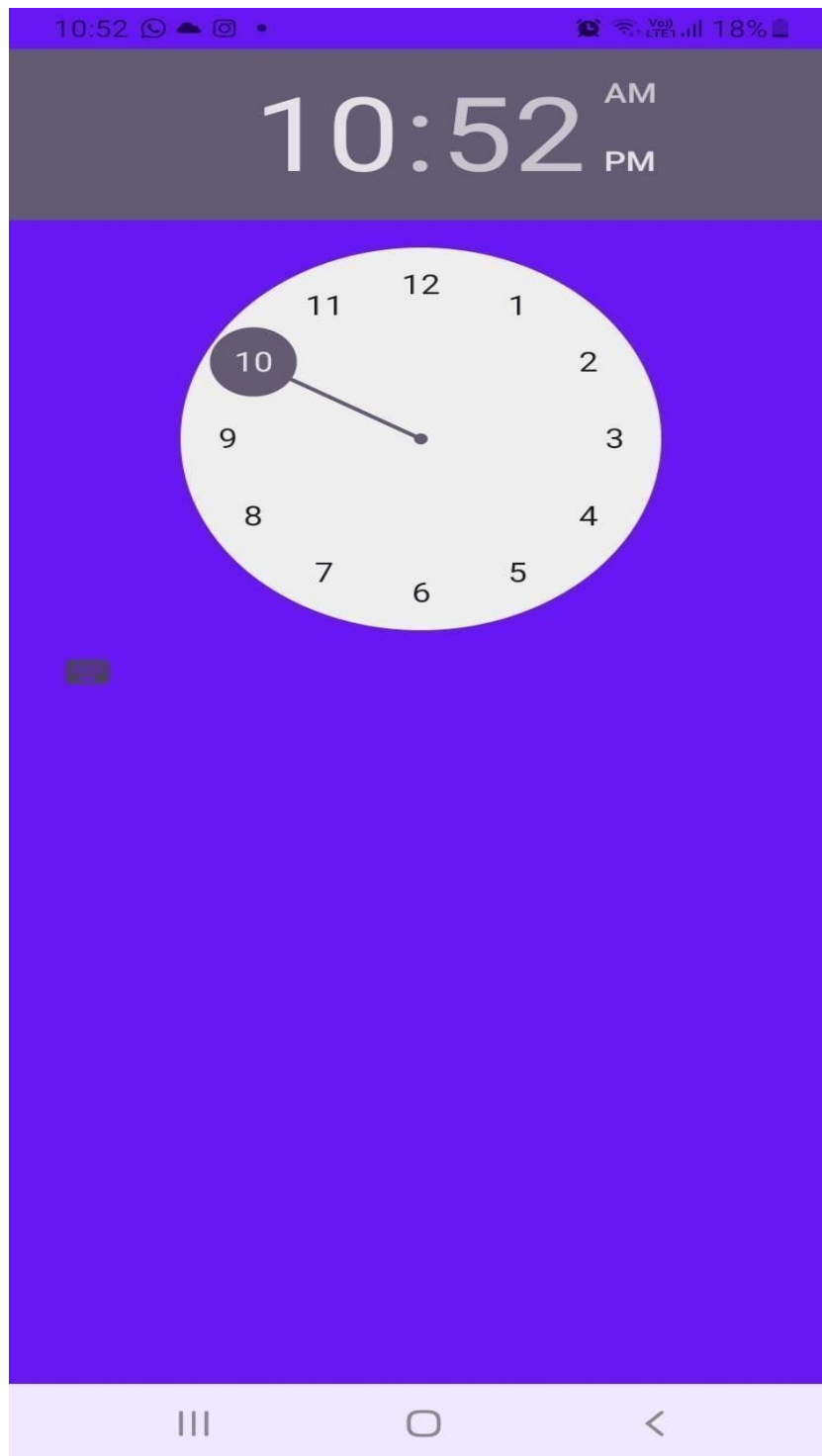


Fig 2.2.1 Output of Timer

## **PROGRAM 2.3 – LIST VIEW**

### **SOURCE CODE:**

#### **XML CODE:**

- `LinearLayout`: Vertical layout to stack views vertically.
- `ListView (@+id/listView)`: Displays a list of items.
- `android:divider="@android:color/darker_gray"`: Divider color between list items.
- `android:dividerHeight="1dp"`: Divider height

#### **JAVA CODE:**

- `MainActivity`: Main activity class that extends `AppCompatActivity`.
- `onCreate(Bundle savedInstanceState)`: Initializes the activity.
- `setContentView(R.layout.activity_main);`: Sets the content view to the layout defined in `activity_main.xml`.
- Initialization:
- `listView`: Reference to the `ListView` from XML layout.
- `dataList`: `ArrayList` to hold data for the list.
- Adding dummy data ("Item 1", "Item 2", etc.) to `dataList`.
- `ArrayAdapter<String> (adapter)`:
- Creates an `ArrayAdapter` to adapt `dataList` to the `ListView`.
- Uses `android.R.layout.simple_list_item_1` as the layout for each item in the list (default Android layout).
- Sets adapter to `listView` using `setAdapter(adapter)`

## OUTPUT:

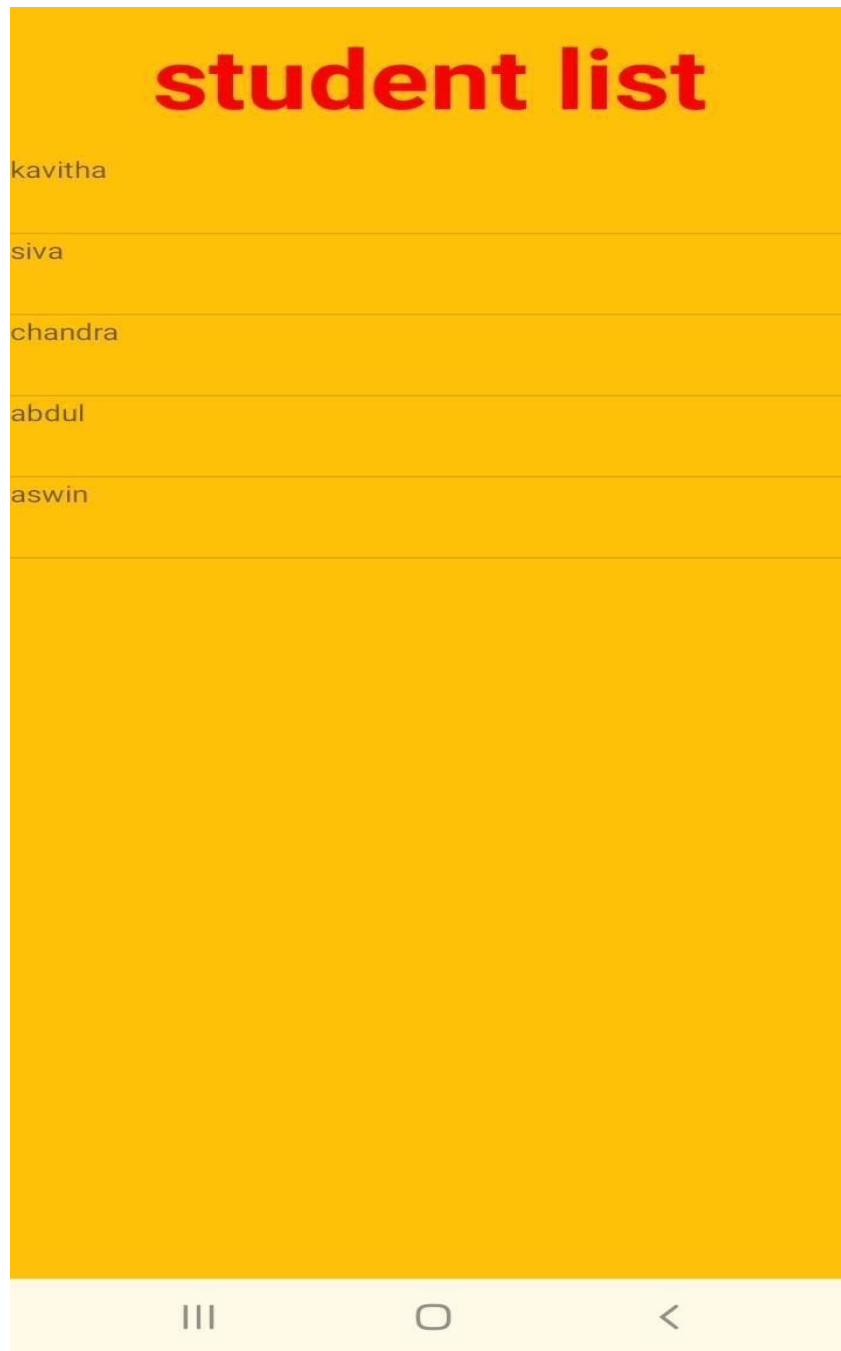


Fig 2.3.1 Output of listview

## **PROGRAM 2.4- CHECK BOX**

### **SOURCE CODE:**

#### **XML CODE:**

- `LinearLayout`: Vertical layout to stack views vertically.
- `CheckBox (@+id/checkbox)`:
- `android:text="Check me"`: Text displayed next to the checkbox.
- `android:checked="false"`: Initial checked state (unchecked).
- `android:textSize="18sp"`: Text size.
- `TextView (@+id/textView)`: Displays text indicating the state of the checkbox.

#### **JAVA CODE:**

- `MainActivity`: Main activity class that extends `AppCompatActivity`.
- `onCreate(Bundle savedInstanceState)`: Initializes the activity.
- `setContentView(R.layout.activity_main);`: Sets the content view to the layout defined in `activity_main.xml`.
- Initialization:
- `checkbox`: Reference to the `CheckBox` from XML layout.
- `textView`: Reference to the `TextView` that displays the checkbox state.
- `checkbox.setOnCheckedChangeListener()`: Sets a listener to monitor checkbox state changes.
- `onCheckedChanged(CompoundButton buttonView, boolean isChecked)`: Callback method invoked when the checkbox state changes.
- Updates `textView` based on whether the checkbox is checked (`isChecked`).

**OUTPUT:**

The image displays a mobile application interface on a light pink background. At the top, there are two radio button options: ☐ MALE and ☒ FEMALE. Below these options is a red button with the text "SHOW SELECTED". Further down, there is a dark grey button with the text "female". At the very bottom, there is a white navigation bar containing three icons: a hamburger menu icon (three horizontal lines), a home icon (a circle), and a back icon (a left-pointing chevron).

Fig 2.4.1 Output of check box

## **PROGRAM 2.5 – IMAGE APP**

### **SOURCE CODE:**

#### **XML CODE:**

- RelativeLayout: Positions child views relative to each other or the parent.
- ImageView (@+id/imageView):
- android:src="@drawable/ic\_launcher\_background": Initial image to display (replace with your own image).
- android:scaleType="centerCrop": Scales the image uniformly (maintaining aspect ratio) to fill the ImageView.
- Button (@+id/buttonChangeImage):
- android:text="Change Image": Text displayed on the button.
- android:layout\_alignParentBottom="true": Aligns the button to the bottom of the parent layout.
- android:layout\_centerHorizontal="true": Centers the button horizontally within the parent.
- android:layout\_marginBottom="24dp": Adds a margin of 24 density-independent pixels (dp) from the bottom.

#### **JAVA CODE:**

- MainActivity: Main activity class that extends AppCompatActivity.
- onCreate(Bundle savedInstanceState): Initializes the activity.
- setContentView(R.layout.activity\_main);: Sets the content view to the layout defined in activity\_main.xml.
- Initialization:
- imageView: Reference to the ImageView from XML layout.
- buttonChangeImage: Reference to the Button that changes the image.
- imageArray: Array holding drawable resource IDs of images (R.drawable.image1, R.drawable.image2, etc.).
- currentIndex: Tracks the index of the current image displayed.
- imageView.setImageResource(imageArray[currentIndex]): Sets the initial image in imageView using the first image in imageArray.
- buttonChangeImage.setOnClickListener(): Sets a click listener for buttonChangeImage to change the image.

- `onClick(View v)`: Callback method invoked when the button is clicked.
- Updates `currentImageIndex` to the next image in `imageArray`.
- Updates `imageView` to display the new image based on `currentImageIndex`.

**OUTPUT:**

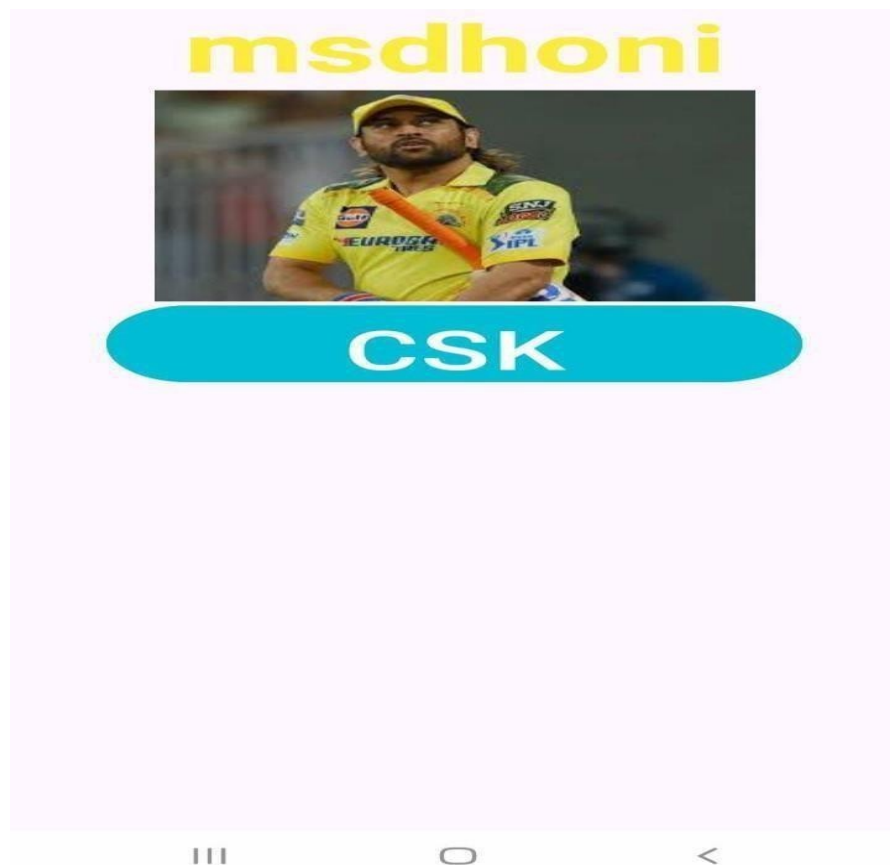


Fig 2.5.1 Output of image app



## **PROGRAM 2.6 – IMAGE SWITCHING**

### **SOURCE CODE:**

#### **XML CODE:**

- RelativeLayout: Positions child views relative to each other or the parent.
- ImageView (@+id/imageView):
- android:src="@drawable/image1": Initial image to display (replace with your own image).
- android:scaleType="fitCenter": Scales the image uniformly (maintaining aspect ratio) to fit within the ImageView's bounds.
- android:contentDescription="Image View": Accessibility description for the ImageView.
- android:clickable="true": Makes the ImageView clickable.
- android:onClick="switchImage": Specifies the method to invoke when the ImageView is clicked.
- TextView (@+id/textView):
- Positioned at the bottom center (android:layout\_centerHorizontal="true" and android:layout\_alignParentBottom="true").
- Provides instructions for the user to tap on the image.

#### **JAVA CODE:**

- MainActivity: Main activity class that extends AppCompatActivity.
- onCreate(Bundle savedInstanceState): Initializes the activity.
- setContentView(R.layout.activity\_main);: Sets the content view to the layout defined in activity\_main.xml.
- Initialization:
- imageView: Reference to the ImageView from XML layout.

- `imageArray`: Array holding drawable resource IDs of images (`R.drawable.image1`, `R.drawable.image2`, etc.).
- `currentImageIndex`: Tracks the index of the current image displayed.
- `imageView.setImageResource(imageArray[currentImageIndex])`: Sets the initial image in `imageView` using the first image in `imageArray`.
- `switchImage(View view)`: Method invoked when the `ImageView` is clicked (as specified by `android:onClick="switchImage"` in XML)
- Updates `currentImageIndex` to cycle through `imageArray`.
- Sets `imageView` to display the new image based on `currentImageIndex`.
- Shows a toast message indicating that the image has been switched.

## OUTPUT:



Fig 2.6.1 Output of image switcher

## PROGRAM 2.7 – CALCULATOR

### SOURCE CODE:

#### XML CODE:

- RelativeLayout: Positions child views relative to each other or the parent.
- TextView (@+id/textViewResult):
  - Displays the current calculation or result.
  - android:gravity="end": Aligns text to the end (right) of the TextView.
- GridLayout:
  - Organizes buttons in a grid with 4 columns and 5 rows.
- **Buttons:**
  - Digits (0 to 9) to input numbers.
  - Operations (+, -, \*, /) for calculations.
  - = to calculate the result.
  - C to clear/reset the calculation.

#### JAVA CODE:

- MainActivity: Main activity class that extends AppCompatActivity.
- onCreate(Bundle savedInstanceState): Initializes the activity.
- setContentView(R.layout.activity\_main);: Sets the content view to the layout defined in activity\_main.xml.
- Initialization:
  - textViewResult: Reference to the TextView for displaying the current calculation or result.
  - imageView: Placeholder for displaying an image result (to be implemented as needed).
  - currentNumber: StringBuilder to store the current input number.
  - operand1, operand2: Doubles to store operands for calculations.
  - operation: Char to store the operation (+, -, \*, /) to perform.
- Button Click Handlers (onClick methods in XML):

- `appendDigit(View view)`: Appends a digit to `currentNumber` and updates `textViewResult`.
- `performOperation(View view)`: Sets `operand1` and `operation` based on the clicked operation button.
- `calculateResult(View view)`: Calculates the result based on `operand1`, `operand2`, and `operation`.
- `clear(View view)`: Clears/reset the calculator state.
- `performOperation(double operand1, double operand2, char operation)`: Performs arithmetic operations (+, -, \*, /).

**OUTPUT:**

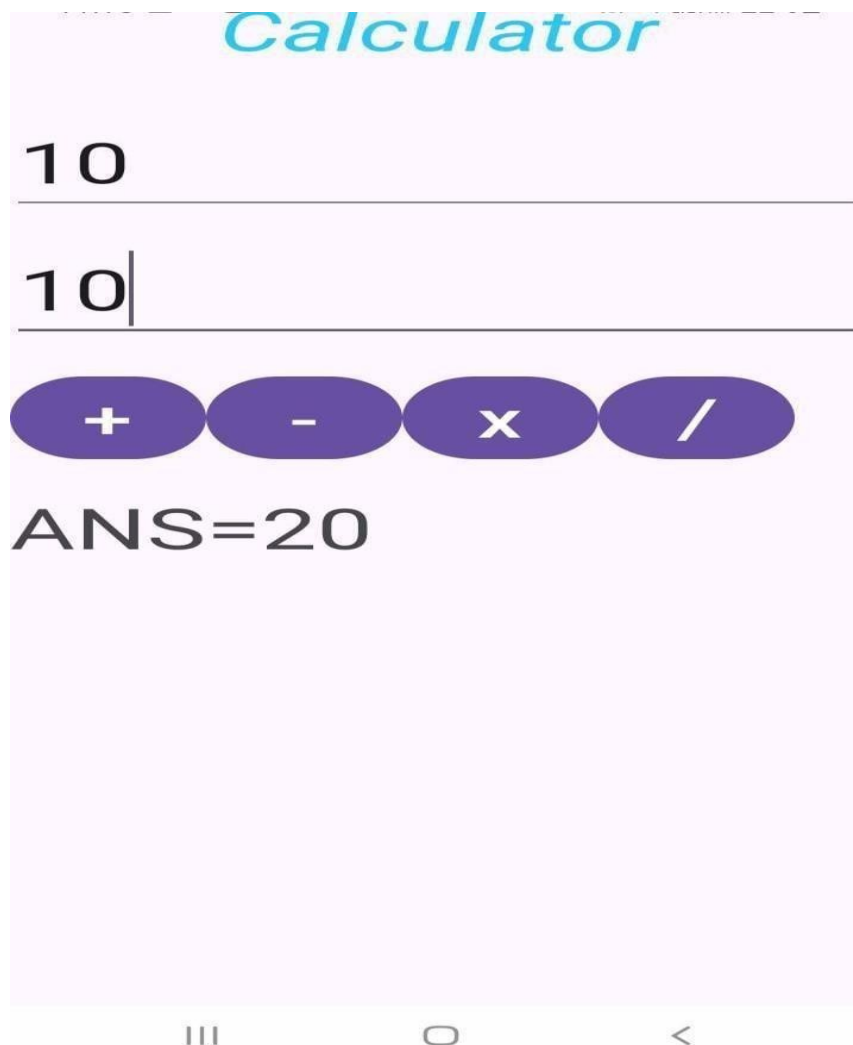


Fig 2.7.1 Output of calculator

## PROGRAM 2.8 – LOGIN PAGE

### SOURCE CODE:

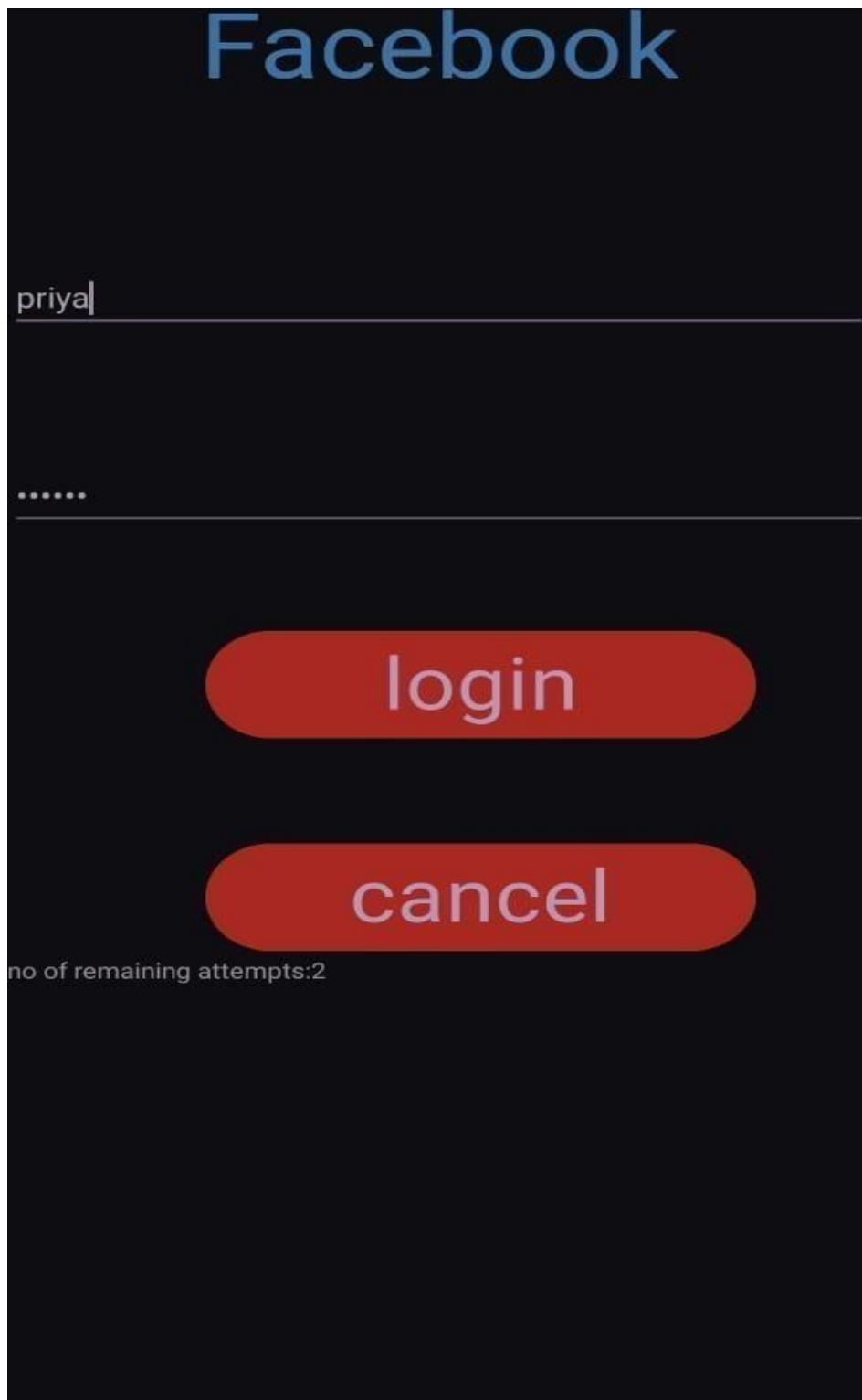
#### XML CODE:

- **RelativeLayout:** Used as the root layout here. It allows you to position child views relative to each other or to the parent view.
- **EditText:** Used for text input fields (`editTextUsername` and `editTextPassword`). `inputType="text"` specifies it's for general text input, while `inputType="textPassword"` hides the entered text.
- **Button:** Represents the login action (`buttonLogin`). It triggers the login functionality when clicked.

#### JAVA CODE:

- **extends AppCompatActivity:** `LoginActivity` inherits from `AppCompatActivity`, providing compatibility with older Android versions and support for modern features.
- **onCreate(Bundle savedInstanceState):** This method is called when the activity is created. Here, we set the content view to `activity_login.xml` and initialize UI components.
- **findViewById(R.id.\*):** Retrieves references to UI elements defined in XML layout by their IDs.
- **setOnClickListener:** Sets a click listener on the login button (`buttonLogin`), defining what happens when the button is clicked.
- **Toast:** Displays a short message to the user. It's used here to show whether the login was successful or not (`Toast.LENGTH_SHORT` specifies the duration of the toast message).

**OUTPUT:**



The image shows a dark-themed Facebook login interface. At the top, the word "Facebook" is displayed in its characteristic blue font. Below it, there are two input fields. The first field contains the text "priya" and has a cursor at the end. The second field contains six dots, indicating a password. Below the input fields are two large, rounded red buttons with white text: "login" and "cancel". At the bottom left, there is a small text label "no of remaining attempts:2".

Fig 2.8.1 Output of login page

## PROGRAM 2.9 – WEB VIEW

### SOURCE CODE:

#### XML CODE:

- **<RelativeLayout>**: Declares the RelativeLayout as the root element, allowing for relative positioning of child views.
- **xmlns:android="http://schemas.android.com/apk/res/android"**: Defines the XML namespace for Android attributes.
- **android:layout\_width="match\_parent"**: Sets the width of the RelativeLayout to match its parent's width.
- **android:layout\_height="match\_parent"**: Sets the height of the RelativeLayout to match its parent's height.
- **<WebView>**: Declares the WebView element within the RelativeLayout.
- **android:id="@+id/webview"**: Assigns a unique ID to the WebView element for reference in Java code.
- **android:layout\_width="match\_parent"**: Sets the width of the WebView to match its parent's width.
- **android:layout\_height="match\_parent"**: Sets the height of the WebView to match its parent's height.

#### JAVA CODE:

- **package com.example.webviewexample;** Declares the package name where this class is located.
- **import android.os.Bundle;** Imports the Bundle class used for passing data to the onCreate method.
- **public class MainActivity extends AppCompatActivity {}**: Defines the MainActivity class, inheriting from AppCompatActivity.
- **@Override**: Annotation indicating that the onCreate method overrides a method in the superclass.
- **protected void onCreate(Bundle savedInstanceState) {}**: Defines the onCreate method, called when the activity is first created.
- **super.onCreate(savedInstanceState);**: Calls the superclass's onCreate method to perform initializations.



- **setContentView(R.layout.activity\_main);** Sets the activity's content to the layout defined in activity\_main.xml.
- **WebView myWebView = findViewById(R.id.webview);** Finds the WebView element in the layout by its ID.
- **myWebView.loadUrl("https://www.example.com");** Loads the specified URL into the WebView.

## OUTPUT:

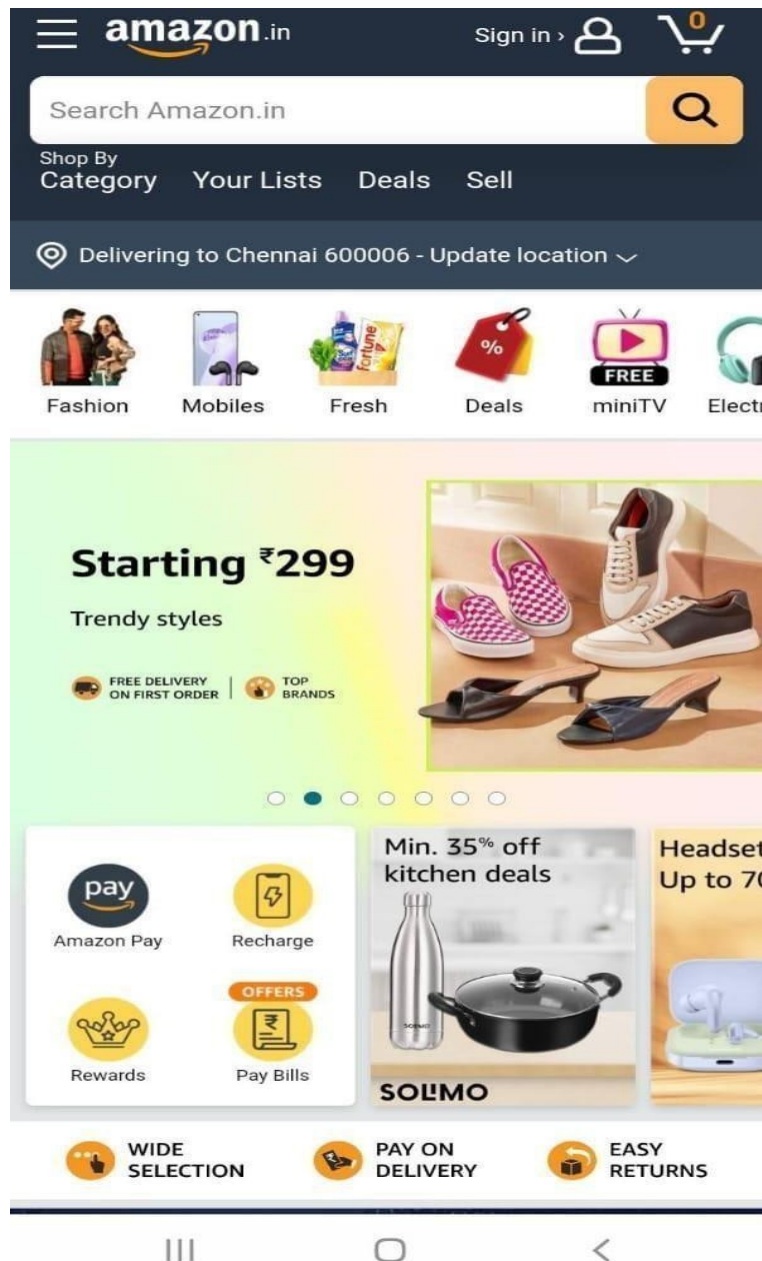


Fig 2.9.1 Output of web view

## PROGRAM 2.10 – MUSIC PLAYER

### SOURCE CODE:

#### XML CODE:

- **<RelativeLayout>**: Declares a RelativeLayout as the root element, allowing for relative positioning of child views.
- **xmlns:android="http://schemas.android.com/apk/res/android"**: Defines the XML namespace for Android attributes.
- **<ImageView>**: Declares an ImageView for displaying album art.
- **android:id="@+id/album\_art"**: Assigns a unique ID to the ImageView for reference in Java code.
- **android:src="@drawable/album\_art\_placeholder"**: Sets the source image for the ImageView.
- **<TextView>**: Declares a TextView for displaying the song title.
- **android:id="@+id/song\_title"**: Assigns a unique ID to the TextView for reference in Java code.
- **<TextView>**: Declares another TextView for displaying the artist name.
- **android:id="@+id/artist\_name"**: Assigns a unique ID to the TextView for reference in Java code.
- **</androidx.constraintlayout.widget.ConstraintLayout>**

#### JAVA code:

- **package com.example.musicplayer;**: Declares the package name where this class is located.
- **public class MainActivity extends AppCompatActivity {}**: Defines the MainActivity class, inheriting from AppCompatActivity.
- **private MediaPlayer mediaPlayer;**: Declares a MediaPlayer object to play audio..
- **protected void onCreate(Bundle savedInstanceState) {}**: Defines the onCreate method, called when the activity is first created.
- **super.onCreate(savedInstanceState);**: Calls the superclass's onCreate method to perform initializations.
- **setContentView(R.layout.activity\_main);**: Sets the activity's content to the layout defined in activity\_main.xml.

- **mediaPlayer = MediaPlayer.create(this, R.raw.sample);**: Initializes the MediaPlayer object with an audio file (sample.mp3) located in the res/raw directory.
- **public void playMusic(View view) {**: Defines the playMusic method, called when the play button is clicked.
- **if (mediaPlayer != null) {**: Checks if the MediaPlayer is not null.
- **mediaPlayer.release();**: Releases the resources used by the MediaPlayer.
- **mediaPlayer = null;**: Sets the MediaPlayer to null.
- **super.onDestroy();**: Calls the superclass's onDestroy method to perform additional clean

## OUTPUT:

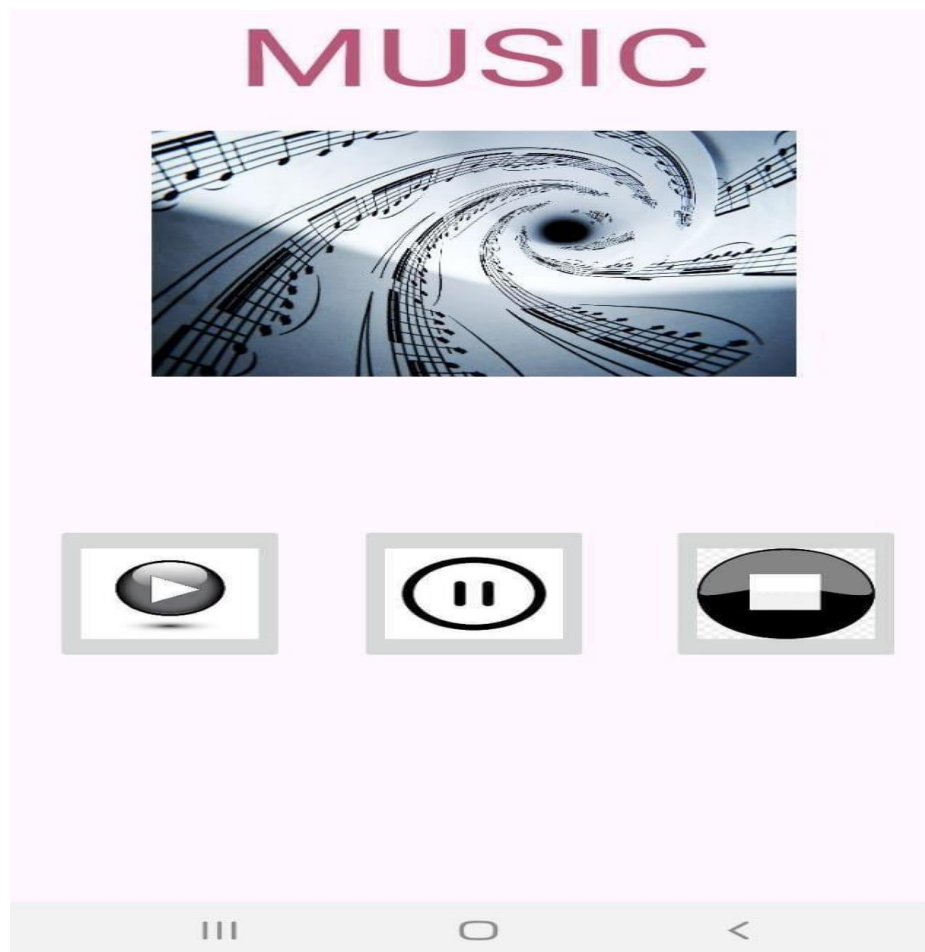


Fig 2.10.1 Output of music player

## PROGRAM 2.11 – AUTO COMPLETE

### SOURCE CODE:

#### XML CODE

- **<RelativeLayout**  
**xmlns:android="http://schemas.android.com/apk/res/android"**: Declares the RelativeLayout as the root element.
- **<AutoCompleteTextView**: Defines the AutoCompleteTextView element.
- **android:id="@+id/autoCompleteTextView"**: Assigns a unique ID to the AutoCompleteTextView for reference in Java code.
- **android:completionThreshold="1"**: Specifies that autocomplete suggestions should appear after typing at least one character.

#### JAVA CODE:

- **public class MainActivity extends AppCompatActivity {}**: Defines the MainActivity class.
- **super.onCreate(savedInstanceState);**: Calls the superclass's onCreate method for initializations.
- **setContentView(R.layout.activity\_main);**: Sets the activity's layout from activity\_main.xml.
- **AutoCompleteTextView autoCompleteTextView = findViewById(R.id.autoCompleteTextView);**: Finds the AutoCompleteTextView in the layout by its ID.
- **String[] countries = {"Australia", "Brazil", "Canada", "Denmark", "Egypt"};**: Defines an array of string suggestions.
- **ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_dropdown\_item\_1line, countries);**: Creates an ArrayAdapter with context, layout, and data array.
- **autoCompleteTextView.setAdapter(adapter);**: Sets the adapter to the AutoCompleteTextView to provide suggestions.

**OUTPUT:**

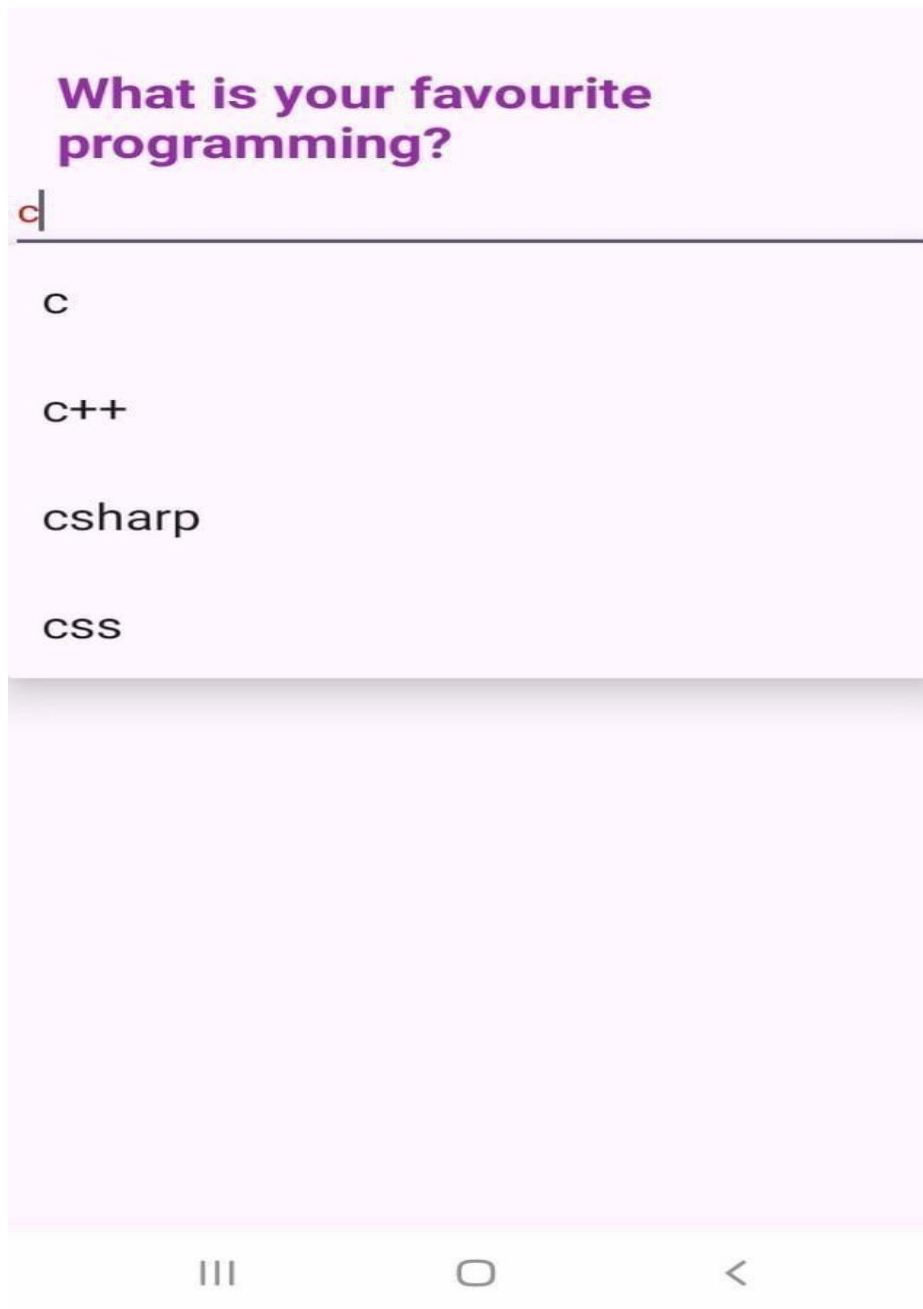


Fig 2.11.1 Output of Auto Complete

## PROGRAM 2.12 – INCREMENT & DECREMENT

### SOURCE CODE:

#### XML CODE

- **<RelativeLayout**  
**xmlns:android="http://schemas.android.com/apk/res/android"**: Declares the RelativeLayout as the root element.
- **<Button**: Defines the Button element for incrementing.
- **android:id="@+id/btnIncrement"**: Assigns a unique ID to the button.
- **android:onClick="incrementCounter"**: Specifies the method to call when the button is clicked.
- **<TextView**: Defines the TextView element for displaying the counter value.
- **android:id="@+id/tvCounter"**: Assigns a unique ID to the TextView.
- **<Button**: Defines the Button element for decrementing.
- **android:id="@+id/btnDecrement"**: Assigns a unique ID to the button.

#### JAVA CODE:

- **private int counter = 0;** Initializes a counter variable.
- **private TextView tvCounter;** Declares a TextView variable for displaying the counter.
- **@Override protected void onCreate(Bundle savedInstanceState) {}**: Overrides onCreate method.
- **super.onCreate(savedInstanceState);** Calls superclass onCreate method.
- **setContentView(R.layout.activity\_main);** Sets activity layout.
- **tvCounter = findViewById(R.id.tvCounter);** Finds TextView by ID.
- **updateCounterText();** Updates TextView with initial counter value.
- **public void incrementCounter(View view) {}** Handles increment button click.
- **counter++;** Increments the counter.
- **updateCounterText();** Updates TextView with new counter value.
- **public void decrementCounter(View view) {}** Handles decrement button click.
- **if (counter > 0) {}** Checks if counter is greater than zero.

- **counter--;** Decrements the counter.
- **updateCounterText();** Updates TextView with new counter value.

**OUTPUT:**

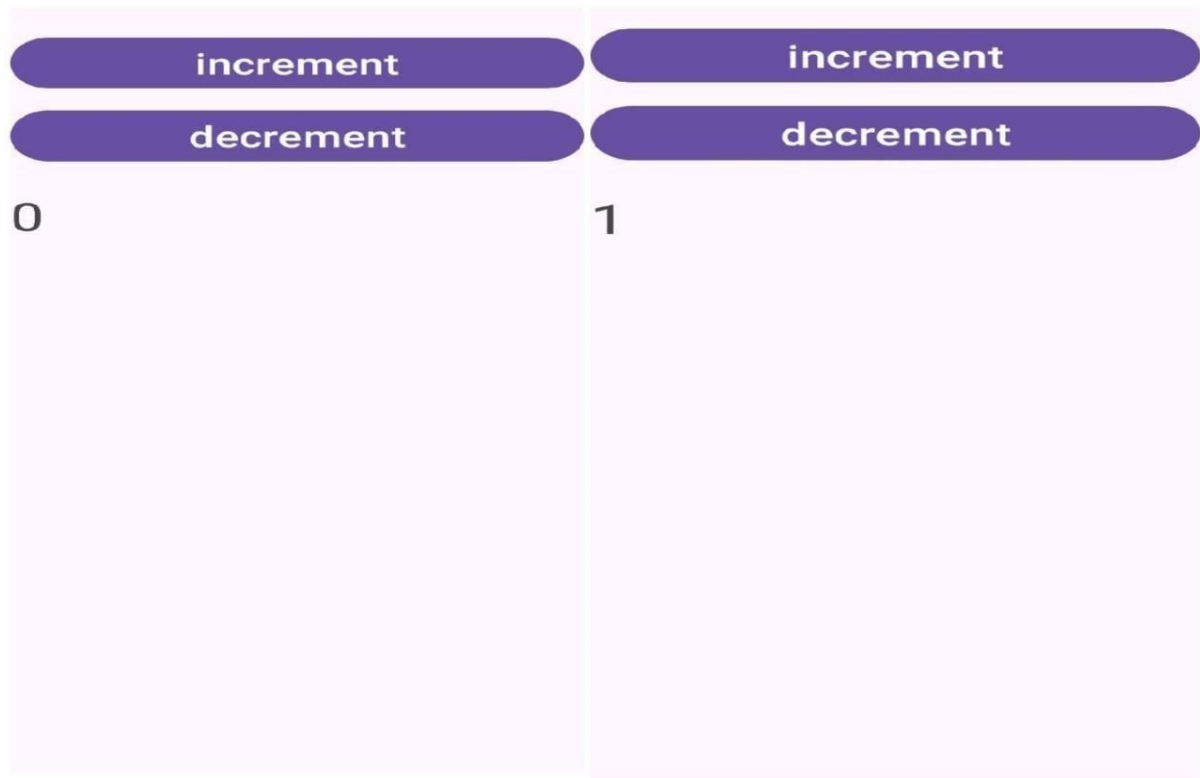


Fig 2.11.1 Output of increment & decrement



## PROGRAM 2.13- SCROLL VIEW

### SOURCE CODE:

#### XML CODE:

- **<ScrollView>**: Defines the ScrollView container with a namespace declaration and sets its width and height to match the parent.
- **<LinearLayout>**: Contains child views arranged vertically, with width matching the parent and height wrapping content.
- **<TextView>**: Represents a simple text element within the LinearLayout.

#### JAVA CODE:

- **package com.example.scrollviewexample;** Declares the package name where this class is located.
- **import android.os.Bundle;** Imports the Bundle class used for passing data to the onCreate method.
- **import androidx.appcompat.app.AppCompatActivity;** Imports AppCompatActivity for backward compatibility and modern features.
- **public class MainActivity extends AppCompatActivity {** Defines the MainActivity class, inheriting from AppCompatActivity.
- **@Override**: Annotation indicating that the onCreate method overrides a method in the superclass.
- **protected void onCreate(Bundle savedInstanceState) {** Defines the onCreate method, called when the activity is first created.
- **super.onCreate(savedInstanceState);** Calls the superclass's onCreate method to perform initializations.
- **setContentView(R.layout.activity\_main);** Sets the activity's content to the layout defined in activity\_main.xml.

## OUTPUT:

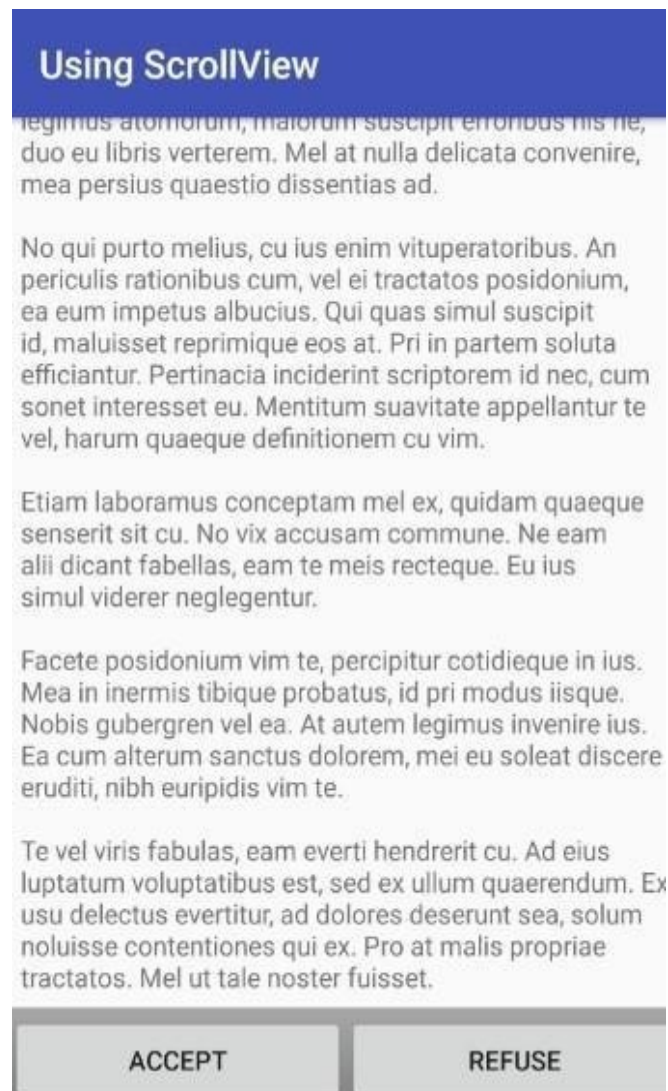


Fig 2.13.1 output of scroll view

## **PROGRAM 2.14-CALLER APP**

### **SOURCE CODE:**

#### **XML CODE:**

- Create a New Project in Android Studio
- To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. The code for that has been given in both Java and Kotlin Programming Language for Android.
- Add Permission to AndroidManifest.xml File
- You need to take permission from the user for a phone call and for that CALL\_PHONE permission is added to the manifest file. Here is the code of the manifest file:
  - <!-- Permission for phone call -->
  - <uses-permission android:name="android.permission.CALL\_PHONE" />
- Step 3: Working with the XML Files
- Next, go to the activity\_main.xml file, which represents the UI of the project. Below is the code for the activity\_main.xml file. Comments are added inside the code to understand the code in more detail. This file contains a Relative Layout which contains EditText to write the phone number on which you want to make a phone call and a button for starting intent or making calls:

#### **JAVA CODE:**

- Working with the MainActivity File
- Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File. Comments are added inside the code to understand the code in more detail.
- In the MainActivity Intent, the object is created to redirect activity to the call manager, and the action attribute of intent is set as ACTION\_CALL.
- Phone number input by the user is parsed through Uri and that is passed as data in the Intent object which is then used to call that phone number .
- setOnClickListener is attached to the button with the intent object in it to make intent with action as ACTION\_CALL to make a phone call.

## OUTPUT:

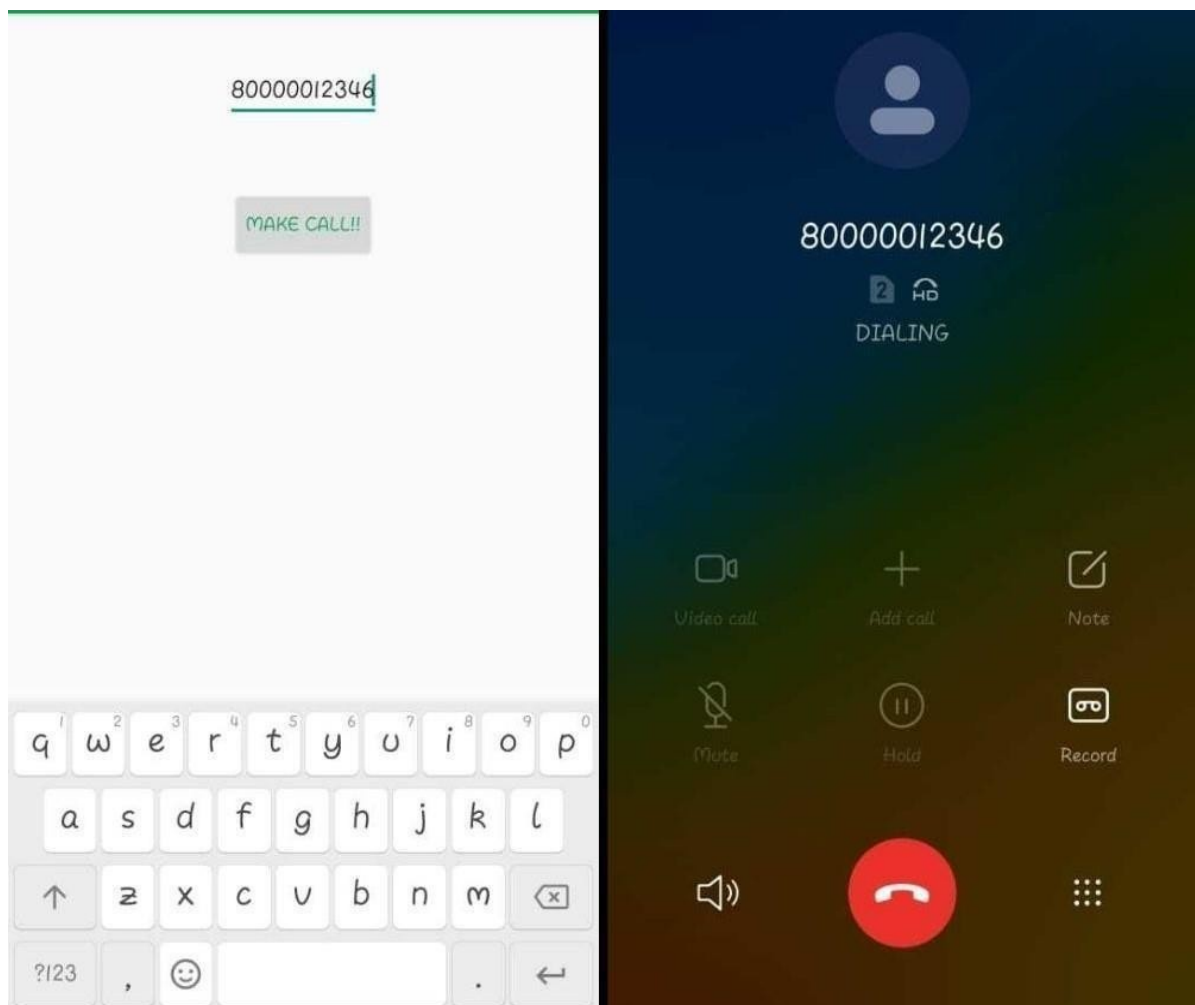


Fig 2.14.1 output of caller app

## PROGRAM 2.15-FRAGMENT

### SOURCE CODE:

#### XML CODE(1):

- **<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android":**  
Declares the LinearLayout as the root element.
- **<TextView:** Defines the TextView element.
- **android:id="@+id/tvFragment":** Assigns a unique ID to the TextView for reference in Java code.
- **<Button:** Defines the Button element.
- **</LinearLayout>:** Closes the LinearLayout element.

#### JAVA CODE:

- **public class ExampleFragment extends Fragment {:** Defines ExampleFragment as a subclass of Fragment.
- **@Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {:** Overrides onCreateView to create the fragment's view.
- **View view = inflater.inflate(R.layout.fragment\_example, container, false);:**  
Inflates the fragment layout defined in fragment\_example.xml.
- **return view;:** Returns the inflated view to display the fragment's UI.
- This Java code defines a basic fragment that inflates a layout file (fragment\_example.xml) to display its UI components within an activity.

**OUTPUT:**



Fig 2.14.1 output of fragment

## **PROGRAM 2.16-GUESS GAME:**

### **SOURCE CODE:**

#### **XML CODE:**

- Create a new Android Studio Project:
- Select "Empty Activity" template.
- :Design the Layout (activity\_main.xml)
- Define the UI components of our Guessing Game app.
- Java Code (MainActivity.java)
- Implementing the logic for the Guessing Game.
- Layout (activity\_main.xml):
- Uses a LinearLayout to arrange UI components vertically.
- TextView (textViewHint) displays instructions.
- EditText (editTextGuess) allows user input for their guess.
- Button (buttonGuess) triggers the guess submission.
- TextView (textViewResult) displays feedback based on the guess.

#### **JAVA CODE:**

- Initializes UI components (editTextGuess and textViewResult).
- Generates a random number (randomNumber) between 1 and 100 using `java.util.Random`.
- `onGuessButtonClick(View view)` method is invoked when the "Guess" button is clicked.
- Parses user input (guessString), checks if the guess is within the valid range (1-100), and compares it with randomNumber.
- Provides feedback (`textViewResult.setText()`) whether the guess is higher, lower, or correct.
- Resets randomNumber for a new game when the correct guess is made.

## OUTPUT:

45

Submit

Try a lower number.

1 2 3 -

4 5 6 =

7 8 9 ↵

, 0 . ✓

Fig 2.16.1 output of guess game



## PROGRAM 2.17-CALENDER:

### SOURCE CODE:

### XML CODE:

- Create a new project and you will have a layout XML file and java file. Your screen will look like the image below.
- Open your xml file and add CalendarView and TextView. And assign id to TextView and CalendarView. After completing this process, the xml File looks in screen.

### JAVA CODE:

- Now, open up the activity java file and define the CalendarView and TextView type variable, and also use findViewById() to get the Calendarview and textview.
- Now, add setOnDateChangeListener interface in object of CalendarView which provides setOnDateChangeListener method. In this method, we get the Dates(days, months, years) and set the dates in TextView for Display
- Now run the app and set the current date which will be shown on the top of the screen.

### OUTPUT:



	June 2016						
	M	T	W	T	F	S	S
22	30	31	1	2	3	4	5
23	6	7	8	9	10	11	12
24	13	14	15	16	17	18	19
25	20	21	22	23	24	25	26
26	27	28	29	30	1	2	3
27	4	5	6	7	8	9	10

Fig 2.17.1 output of calender

## PROGRAM 2.18-MUSIC SWITCHING

### SOURCE CODE:

### XML CODE:

- In a music switching app, create a MediaPlayer for audio playback.
- Design the UI with buttons for play, pause, next, and previous.
- Implement OnClickListener for button controls.
- Use a list or array to manage multiple tracks, updating the MediaPlayer source on track switch, ensuring lifecycle management for seamless playback.

### JAVA PROGRAM:

- **Declare Media Player:** Initialize two instances of MediaPlayer for playing two different tracks.
- **Load and Prepare:** Use MediaPlayer.create() to load audio files into each instance and call prepare() on both to ensure they are ready for playback.
- **Switch Button:** Implement a button with an OnClickListener that stops the currently playing MediaPlayer and starts the other.
- **Release Resources:** Properly release resources using release() method when the app is stopped or finished to avoid memory leaks.
- **Error Handling:** Implement error handling using try-catch blocks for IOException and other potential exceptions to ensure robustness.

### OUTPUT:



Fig2.18.1 output of music switching

## PROGRAM 2.19-TIC TAC TOE

### SOURCE CODE:

### XML CODE:

- Create a new Android Studio Project:
- Select "Empty Activity" template.
- Design the Layout (activity\_main.xml)
- This will define the UI components of our Tic Tac Toe game.

### JAVA CODE:

- Initializes the 2D array buttons to hold references to all buttons on the game board.
- `onButtonClick(View view)` is called when any button is clicked, handling player turns, updating button text (X or O), and checking for a win/draw condition.
- `checkForWin()` method checks all possible win conditions (rows, columns, diagonals).
- `player1Wins()`, `player2Wins()`, and `draw()` methods display Toast messages for game outcomes and reset the board with `resetBoard()`.
- Notes:
- This example doesn't include advanced features like AI for single-player mode or networked multiplayer.
- For a more polished game, consider adding animations, sound effects, a scoreboard, and improving the user interface.

### OUTPUT:

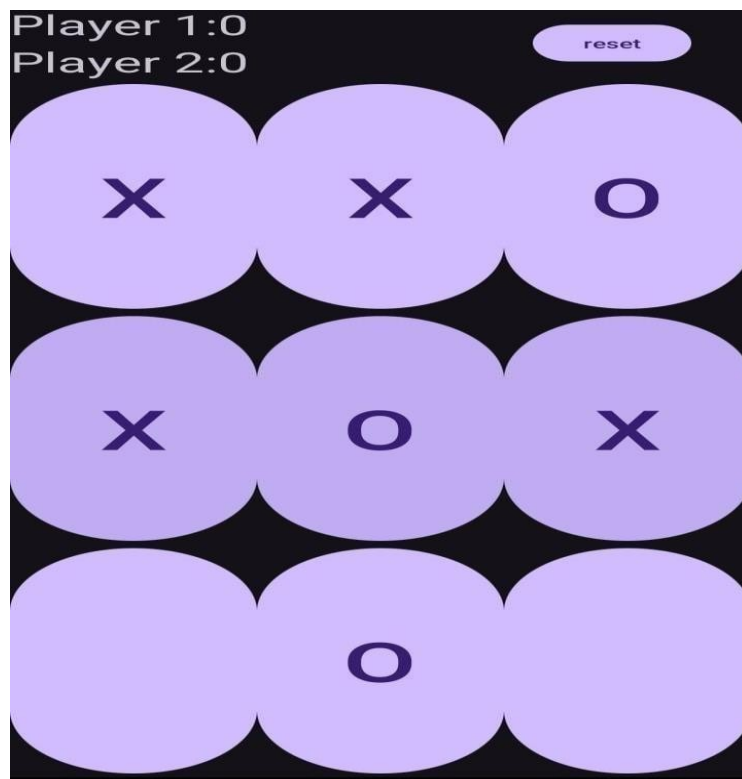


Fig 2.19.1 output of tic tac toe

## **PROGRAM 2.20- VIDEO PLAYER: SOURCE CODE:**

### **XML CODE:**

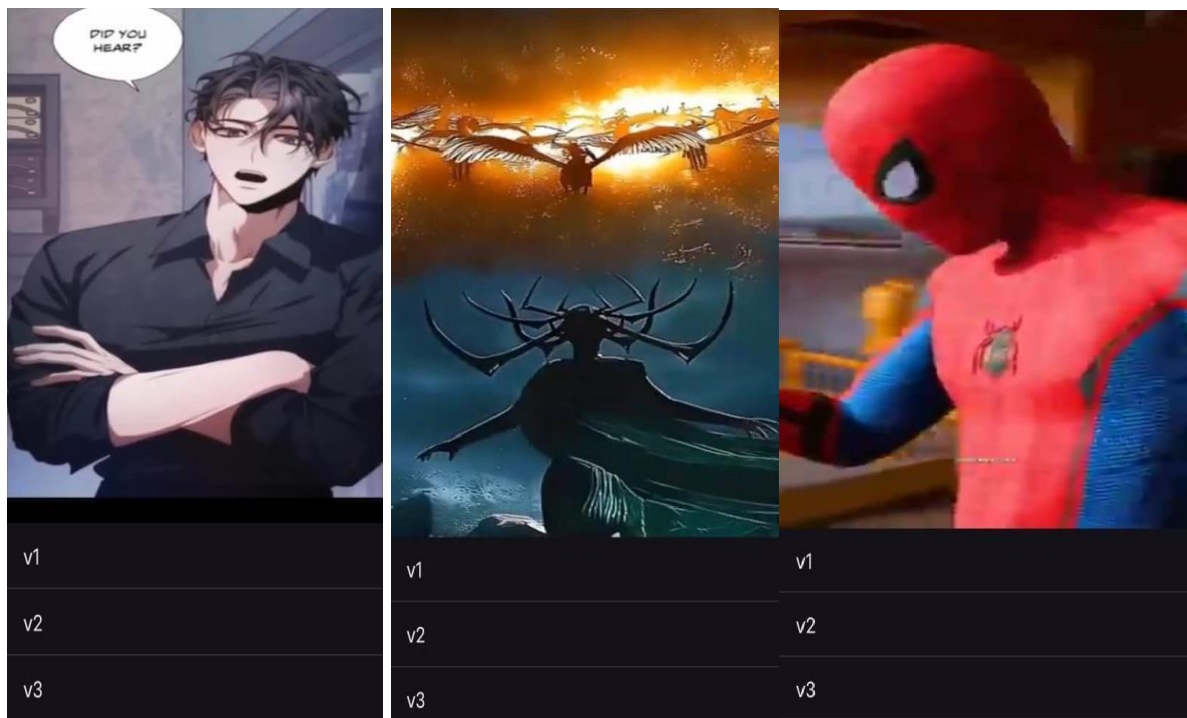
- Setting up the Project
- Create a new Android Studio Project:
- Select "Empty Activity" template.
- Add Permissions (if needed):
- If targeting Android 6.0 (API level 23) or higher, add permissions in AndroidManifest.xml for accessing storage.
- Prepare a video file:
- Place a video file (e.g., MP4) in the res/raw directory of your project. If you don't have the raw directory, create it under res.
- Layout XML (activity\_main.xml)
- Define the UI components of our video player.

### **JAVA CODE:**

- Initializes VideoView and MediaController.
- Sets the video file path (videoPath) using Uri.
- Starts video playback with videoView.start().
- Pauses video playback when the activity is paused (onPause()).
- Stops video playback and releases resources when the activity is destroyed (onDestroy()).
- Notes:
- Replace R.raw.your\_video\_file with the appropriate resource identifier for your video file in the raw directory.
- This example doesn't cover all edge cases (like handling errors, managing multiple videos, etc.). For a production-quality app, you'd need additional features and error handling.
- Ensure proper resource management (like stopping playback and releasing resources) to avoid memory leaks.

## OUTPUT:

Fig 2.20.1 output of Video player



## PROGRAM 2.21-QUIZ GAME:

### SOURCE CODE:

### XML CODE:

- Here the parent layout is a `LinearLayout` whose orientation is set to vertical. Inside it, there is one `ImageView`, one `TextView`, two `Buttons`, and two `ImageButton`.
- The `Button` and `ImageButton` are inside a child `LinearLayout` for horizontal orientation
- . `ImageView` is used for displaying image and `TextView` is used to display the question and `Button` is used to indicate true/false and `ImageButton` for navigating to next/previous question

### JAVA CODE:

- `onCreate()` method is invoked first when the app is launched. `Question[]` array is instantiated with question Id and right answer to the question.
- `setOnClickListener()` method is invoked whenever `Button/ImageButton` is clicked, so when the user clicks a button it checks for its Id by `getId()` method and performs

actions as per our logic.

- () updates question by setText() method of TextView and changes images by keeping track of question number. checkAnswer() method checks the original answer with the button clicked and uses Toast to display text accordingly.

## OUTPUT:

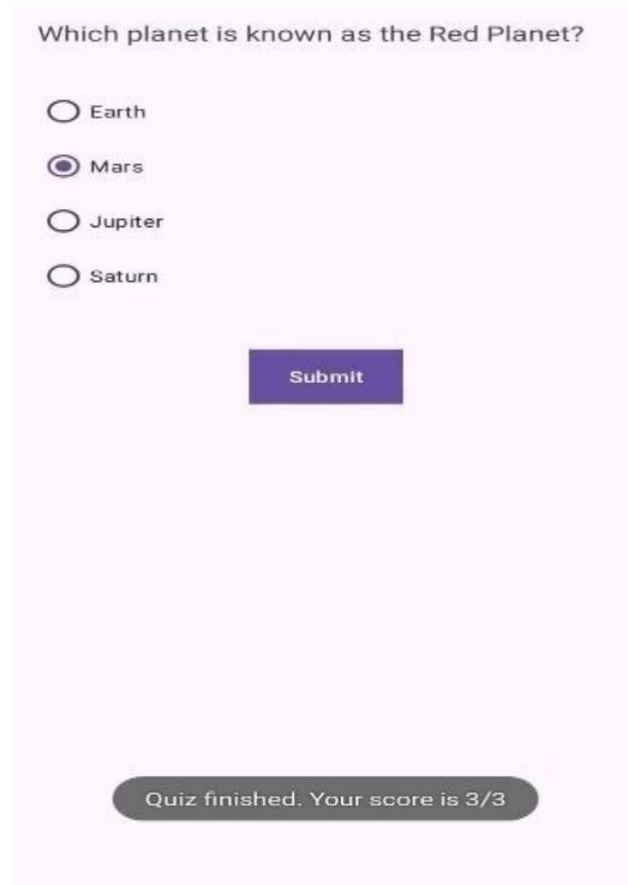


Fig 2.21.1 output quiz game

## **PROGRAM 2.22-PDF VIEWER:**

### **SOURCE CODE:**

#### **XML CODE:**

- Defines a RelativeLayout and specifies the XML namespace for Android attributes.
- Sets the width of the RelativeLayout to match the width of its parent.
- Sets the height of the RelativeLayout to match the height of its parent.
- Begins a TextView element to display the title of the PDF.
- Assigns a unique ID to the TextView for the PDF title.
- Sets the width of the TextView to wrap around its content.
- Sets the height of the TextView to wrap around its content.
- Centers the TextView horizontally within the parent RelativeLayout.
- Adds a top margin of 20dp to the TextView.
- Sets the default text to display in the TextView.
- Sets the text size of the TextView to 24sp.
- Begins a PDFView element to display the PDF content.
- Assigns a unique ID to the PDFView element.
- Sets the width of the PDFView to match the width of its parent.
- Sets the height of the PDFView to match the height of its parent.
- Positions the PDFView below the TextView.
- Adds a top margin of 10dp to the PDFView.
- Ends the RelativeLayout element.

#### **JAVA CODE:**

- Defines the package for the application.
- Imports the Bundle class for activity state storage.
- Imports the AppCompatActivity class for compatibility support.
- Imports the PDFView class to display PDFs.
- Defines the MainActivity class, inheriting from AppCompatActivity.
- Overrides a method from the superclass.
- Defines the onCreate method, called when the activity is created.
- Calls the superclass's onCreate method.

- Sets the activity's layout from XML.
- Finds the PDFView element by its ID.
- Loads a PDF from the assets folder and displays it.
- Ends the onCreate method.
- Ends the MainActivity class.

## OUTPUT:



Fig 2.22.1 output of pdf viewer



## **SUGGESTION AND FEED BACK**

- The overall experience was wholesome and we came to know about the mobile app development.
- Practical experiences which we had during these 4 weeks made us learn more.
- We have learnt about the mobile app applications in real life mode which will be helpful to us we are going to implement.
- We also learned some basic and advanced programs and helped us gaining practical knowledge and training.
- By focusing on these areas, you can create a high-quality, user-friendly, and secure Android app that meets user expectations and performs well in the market.

## **CONCLUSION**

Mobile app development is a dynamic and crucial aspect of today's digital landscape. This report has highlighted key phases and considerations in the app development lifecycle, emphasizing the importance of thorough planning, robust design, efficient development, rigorous testing, and effective deployment strategies. The iterative nature of development, supported by agile methodologies, allows for flexibility and responsiveness to changing requirements and user feedback. User experience (UX) and user interface (UI) design play pivotal roles in ensuring app usability and engagement, contributing significantly to its success. Furthermore, the integration of backend services, security measures, and performance optimization are critical to delivering a reliable and scalable app. Looking forward, the mobile app development industry continues to evolve with advancements in technologies such as AI, AR/VR, and IoT, offering new avenues for innovation. Success in app development hinges on collaboration across multidisciplinary teams, adherence to industry best practices, and a keen understanding of user expectations and market trends. By aligning these elements, developers can create impactful, user-centric mobile applications that meet both business objectives and user needs effectively.