
Godavari College Of Engineering, Jalgaon.



**ACADEMIC YEAR
2020 - 2021**

VI SEMESTER

**LAB MANUAL
MOBILE APPLICATIONS DEVELOPMENT FOR
ANDROID.
[BTCOL607]**

FACULTY : PROF. GANESH CHAVAH

**DEPARTMENT OF
COMPUTER ENGINEERING**

Godavari Foundation's
Godavari College Of Engineering, Jalgaon.
(NAAC Accredited)
(An affiliated to Dr. Babasaheb Ambedkar Technological University)



CERTIFICATE

This is to certify that Miss. **Shweta Ravindra Patil** , Roll No: **34**, PRN No: **1951711245011** of **T.Y. COMPUTER** class has satisfactorily carried out the practical work in the Subject : **Mobile Application Development for Android [BTCOL607]** as per laid down in the syllabus, in this Laboratory and that this journal represent **her** bonafide work in the year **2020-2021**.

Date :

Signature
Prof. Ganesh Chavan
Faculty in Charge

Signature
Prof. Pramod B. Gosavi
H.O.D.

Dr. V. H. PATIL
PRINCIPAL
Godavari Foundation's
Godavari College of Engineering, Jalgaon

Index

Sr. No	Practical Name	Page No.	Dates
1	Install the Android SDK and developer tools and build a test project to confirm that those tools are properly installed and configured.	1 – 9	15 - 04 - 2021
2	Write a program using activity class to show different events.	10 – 12	22 - 04 - 2021
3	Write a program to send user from one application to another. (For example redirection to map).	13 – 20	29 - 04 - 2021
4	Write a program to play audio files.	21 – 25	06 - 05 - 2021
5	Write a program to play video files.	26 – 28	13 - 05 - 2021
6	Write a program to capture image using built in camera.	29 - 32	27 - 05 - 2021
7	Write a program to send SMS.	33 - 36	03 - 06 - 2021
8	Write a program to convert text to speech.	37 - 40	17 - 06 - 2021
9	Write a program to call a number.	41 - 45	19 - 06 - 2021

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class: T. Y. Btech. (Comp.)

Practical No. : 1

Date : 15 - 04 - 2021

Title : Install the Android SDK and developer tools and build a test project to confirm that those tools are properly installed and configured.

Aim : To Install the Android SDK and developer tools and build a test project to confirm that those tools are properly installed and configured.

Theory:

Android : Android is the best-selling **Operating System** among various mobile platforms across the globe. Hundreds of millions of mobile devices are powered by **Android** in more than 190 countries of the world. It conquered around **75%** of the global market share by the end of 2020, and this trend is growing bigger every other day.

Android Versions : Google launched the first version of the Android platform on Nov 5, 2007. Since then, Google released a lot of android versions such as Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Jellybeans, Kitkat, Lollipop, marshmallow, Nougat, Oreo, etc. with extra functionalities and new features.

Android Studio : Android Studio is the official **IDE (Integrated Development Environment)** for Android app development and it is based on **JetBrains' IntelliJ IDEA** software. Android Studio provides many excellent features that enhance productivity when building Android apps.

What is Android SDK ? : The Android SDK (Software Development Kit) is a set of development tools that are used to develop applications for the Android platform. This SDK provides a selection of tools that are required to build Android applications and ensures the process goes as smoothly as possible. SDK tools are generally platform independent and are required no matter which android platform you are working on.

- The Android SDK includes a complete set of development tools. It includes a debugger, libraries, a handset emulator.
- Software written in Java can be compiled to be executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use.

System Requirements :

- Microsoft Windows 7/8/10 (32-bit or 64-bit)

- 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Android Studio Installations & Android SDK set up Steps (on 64-bit Windows 10) :

Step 1) Download Android Studio from this link : <https://developer.android.com> Official Site



Android Studio provides the fastest tools for building apps on every type of Android device.

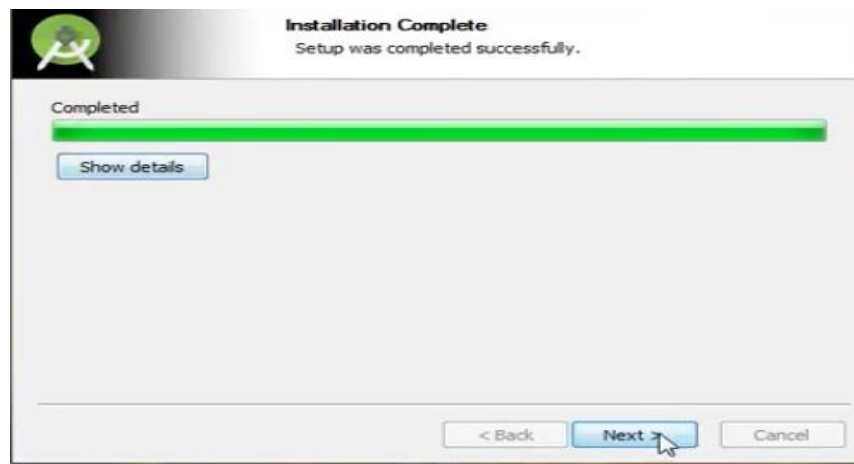
DOWNLOAD ANDROID STUDIO

4.1.3 for Windows 64-bit (896 MiB)

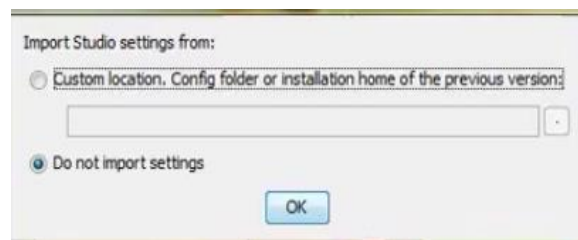
Step 2) After the downloading has finished, open the file from downloads and run it. It will prompt the following dialog box.



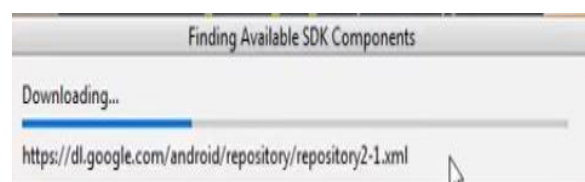
Step 3) It will start the installation, and once it is completed, it will be like the image shown below. Click on next.



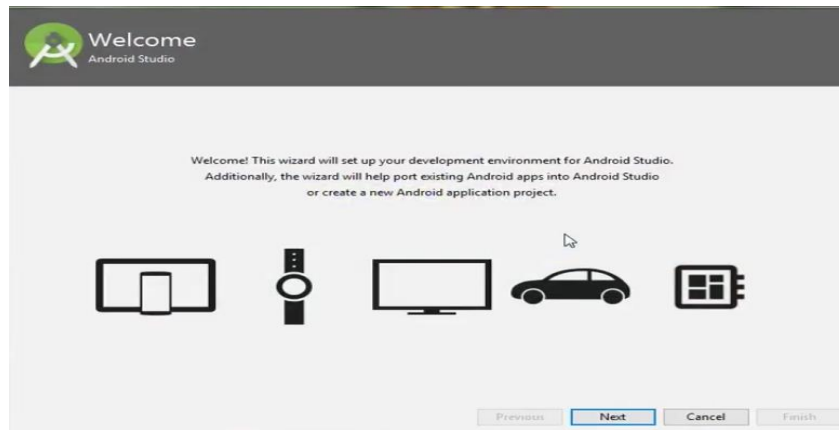
Step 4) Once “**Finish**” is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the ‘Don’t import Settings option’. Click the **OK** button.



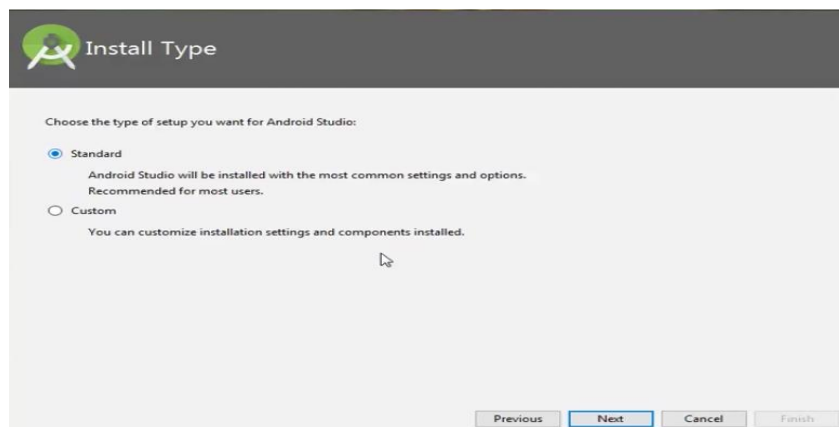
Step 5) This will start the Android Studio. Meanwhile, it will be finding the available SDK components.



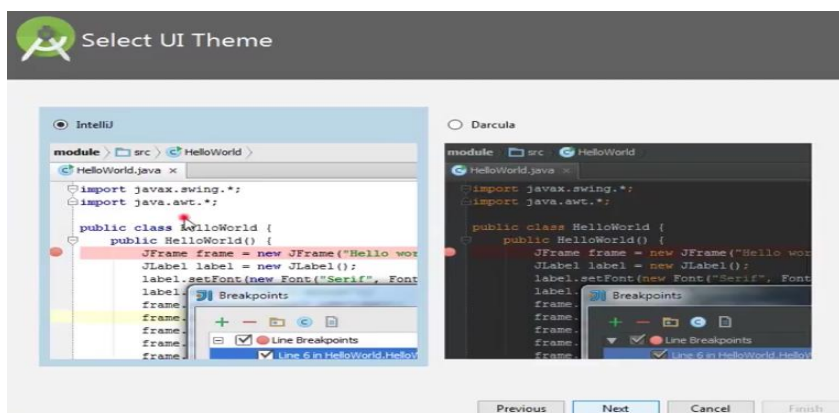
Step 6) After it has found the SDK components, it will redirect to the Welcome dialog box. Click on Next.



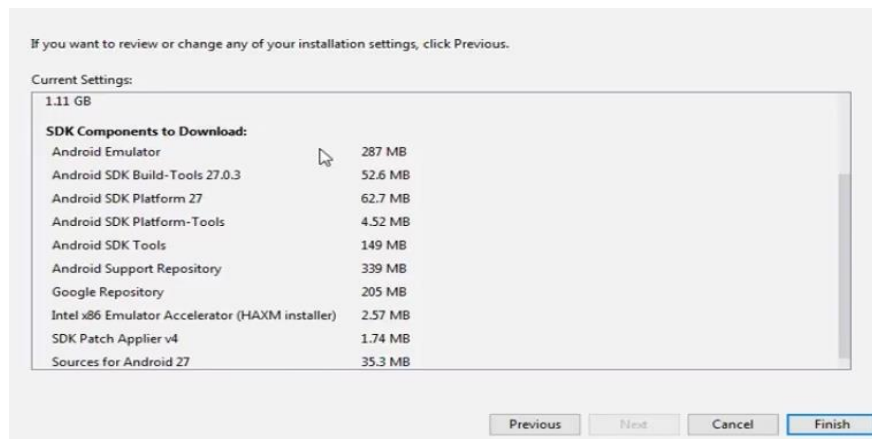
Step 7) Choose Standard and click on Next.



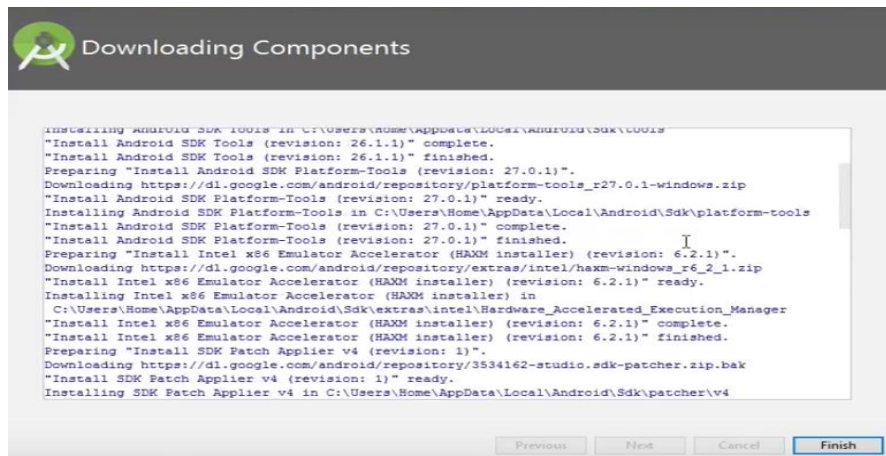
Step 8) Now choose the theme, whether the Light theme or the Dark one. The light one is called the IntelliJ theme whereas the dark theme is called Darcula. Choose as required. Click on the Next button.



Step 9) Now it is time to download the SDK components. Click on Finish. Components begin to download let it complete.



Step 10) The Android Studio has been successfully configured. Now it's time to launch and build apps. Click on the Finish button to launch it.

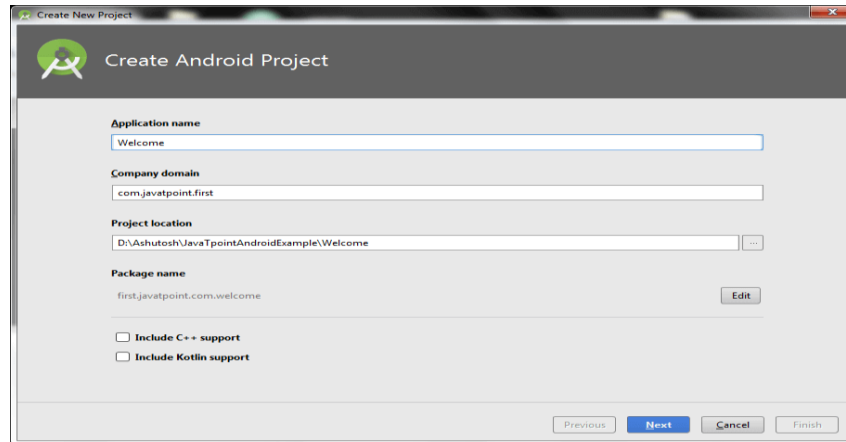


Steps For Building New Test Android Project :

Step 1) Open **Android Studio** & Click on **Start a new Android Studio project** to build a new app.

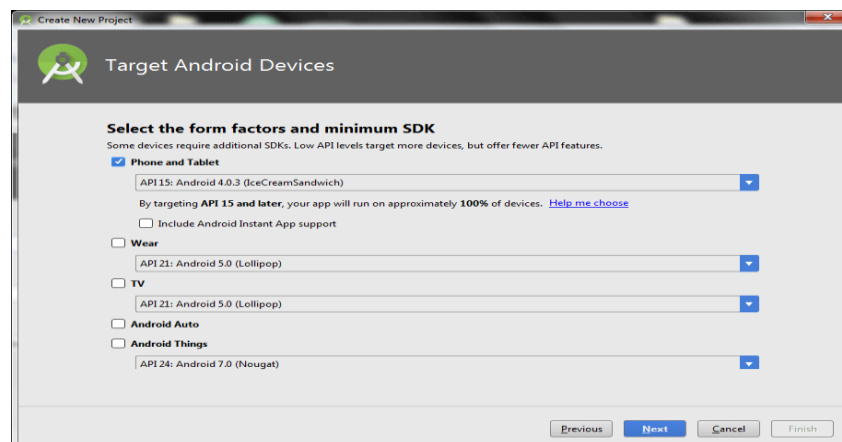


Step 2) Provide the following information: Application name, Company domain, Project location and Package name of application and click next.



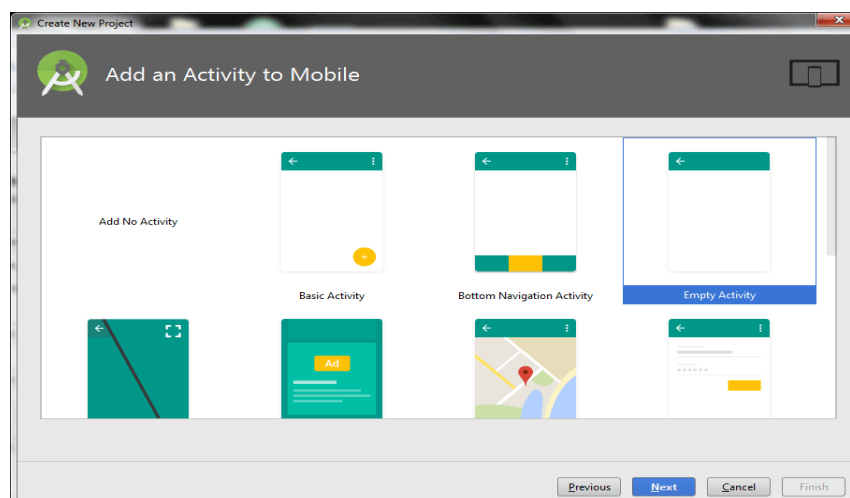
The screenshot shows the 'Create Android Project' dialog box. It has a title bar with the Android Studio logo and the text 'Create Android Project'. The main area contains several input fields: 'Application name' with the text 'Welcome', 'Company domain' with 'com.javatpoint.first', 'Project location' with 'D:\Ashutosh\JavaTpointAndroidExample\Welcome', and 'Package name' with 'first.javatpoint.com.welcome'. There are checkboxes for 'Include C++ support' and 'Include Kotlin support', both of which are unchecked. At the bottom right, there is an 'Edit' button next to the package name field. At the very bottom, there are four buttons: 'Previous', 'Next' (highlighted in blue), 'Cancel', and 'Finish'.

Step 3) Select the API level of application and click next.



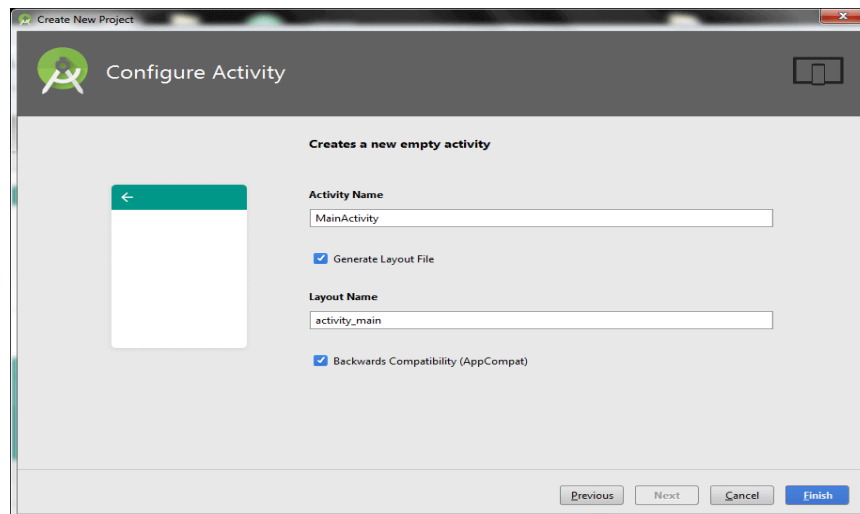
The screenshot shows the 'Target Android Devices' dialog box. It has a title bar with the Android Studio logo and the text 'Target Android Devices'. The main area is titled 'Select the form factors and minimum SDK' and includes a note: 'Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.' There are four main sections: 'Phone and Tablet' (checked), 'Wear', 'TV', and 'Android Auto'. Each section has a dropdown menu for the API level. The 'Phone and Tablet' section is expanded, showing 'API 15: Android 4.0.3 (IceCreamSandwich)' selected. Below this, there is a note: 'By targeting API 15 and later, your app will run on approximately 100% of devices. [Help me choose](#)'. There is also an unchecked checkbox for 'Include Android Instant App support'. At the bottom right, there are four buttons: 'Previous', 'Next' (highlighted in blue), 'Cancel', and 'Finish'.

Step 4) Select the Activity type (Empty Activity).



The screenshot shows the 'Add an Activity to Mobile' dialog box. It has a title bar with the Android Studio logo and the text 'Add an Activity to Mobile'. The main area displays a grid of activity templates. The first template is 'Add No Activity'. The second is 'Basic Activity'. The third is 'Bottom Navigation Activity'. The fourth is 'Empty Activity', which is highlighted with a blue border. At the bottom right, there are four buttons: 'Previous', 'Next' (highlighted in blue), 'Cancel', and 'Finish'.

Step 5) Provide the Activity Name and click finish.



Step 6) Write the Program Code for a) activity_main.xml & 2) MainActivity.java

a) activity_main.xml Code :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="first.javatpoint.com.welcome.MainActivity">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Android!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
```

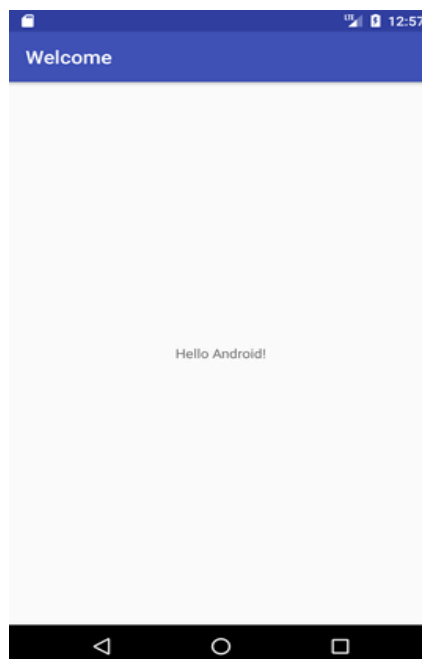
```
app:layout_constraintRight_toRightOf="parent"  
app:layout_constraintTop_toTopOf="parent" />
```

```
</android.support.constraint.ConstraintLayout>
```

b) MainActivity.java Code :

```
package first.javatpoint.com.welcome;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
  
    @Override    protected void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.activity_main);    } } 
```

Step 7) Run the android application : To run the android application, click the run icon on the toolbar or simply press Shift + F10. The android emulator might take 2 or 3 minutes to boot. So please have patience. After booting the emulator, the android studio installs the application and launches the activity. You will see something like this:



Conclusion:- In this Practical i learned to install the Android SDK and developer tools and build a test project to confirm that those tools are properly installed and configured.

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 2

Date : 22 - 04 - 2021

Title : Write a program using activity class to show different events.

Aim : To Implement a program using activity class to show different events.

Theory:

Events : Events are a useful way to collect data about a user's interaction with interactive components of Applications. Like button presses or screen touch etc. The Android framework maintains an event queue as first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

There are following three concepts related to Android Event Management –

1. **Event Listeners** – An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
2. **Event Listeners Registration** – Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
3. **Event Handlers** – When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Program Steps :

Step 1: Create a New Project : To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Working with activity_main.xml file : Go to the app -> res -> layout -> activity_main.xml section and set the layout for the app. Below is the complete code for the activity_main.xml file.

Code for 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" >
    <Button
        android:id="@+id/btnClick"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Event"
        android:layout_marginTop="200dp" android:layout_marginLeft="130dp"/>
    <TextView
        android:id="@+id/txtResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:textColor="#86AD33"
        android:textSize="20dp"
        android:textStyle="bold"
        android:layout_marginTop="12dp"/>
</LinearLayout>

```

Step 3: Working with MainActivity.java file : Go to the app -> java -> com.example.(Package Name) -> MainActivity.java section. Below is the complete code for the MainActivity.java file.

Code for MainActivity.java :

```

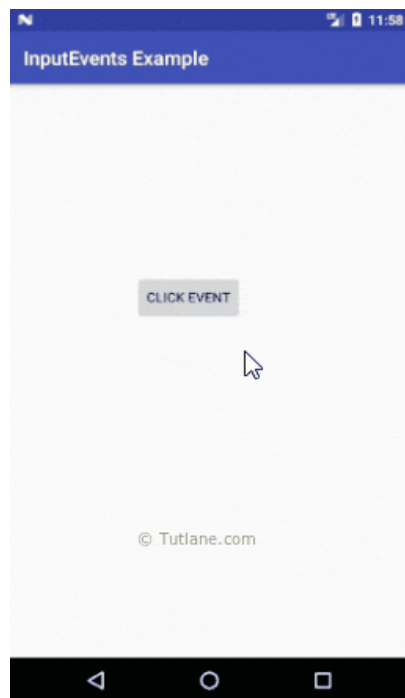
package com.tutlane.inputeventsexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button btn;
    TextView tView;
    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

```
setContentView(R.layout.activity_main);  
btn = (Button)findViewById(R.id.btnClick);  
textView = (TextView)findViewById(R.id.txtResult);  
btn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        textView.setText("You Clicked On Button");  
    }  
}); }
```

Output:



Conclusion:- In this Practical i learned to implement a program using activity class to show different events.

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 3

Date : 29 - 04 - 2021

Title : Write a program to send user from one application to another. (For example redirection to map).

Aim : To Implement a program to send user from one application to another. (For example redirection to map).

Theory:

Sending the user to another app : One of Android's most important features is an app's ability to send the user to another app based on an "action" it would like to perform. For example, if your app has the address of a business that you'd like to show on a map, you don't have to build an activity in your app that shows a map. Instead, you can create a request to view the address using an Intent. The Android system then starts an app that's able to show the address on a map.

Associate intent actions with data : Intents often also include data associated with the action, such as the address you want to view, or the email message you want to send. Depending on the intent you want to create, the data might be a Uri, one of several other data types, or the intent might not need data at all.

If your data is a Uri, there's a simple Intent() constructor you can use to define the action and data.

Complete example : Here's a complete example that shows how to create an intent to view a map, verify that an app exists to handle the intent, then start it:

Code :

```
// Build the intent.
```

```
val location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California")
```

```
val mapIntent = Intent(Intent.ACTION_VIEW, location)
```

```
// Try to invoke the intent.
```

```
try {
```

```
    startActivity(mapIntent)
```

```
} catch (e: ActivityNotFoundException) {
```

```
    // Define what your app should do if no activity can handle the intent.}
```


Program Steps :

Step 1: Create a New Project : To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Required Permission in AndroidManifest.xml

Code for AndroidManifest.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.com.mapexample">
    <!-- The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the 'MyLocation' functionality. -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <!-- The API key for Google Maps-based APIs is defined as a string resource. (See the file
            "res/values/google_maps_api.xml"). Note that the API key is linked to the encryption key used to sign the APK.
            You need a different API key for each encryption key, including the release key that is used to sign the APK for
            publishing. You can define the keys for the debug and release targets in src/debug/ and src/release/. -->

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps">
```

```

        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>

```

Step 3: Working with activity_main.xml file : Go to the app -> res -> layout -> activity_main.xml section and set the layout for the app. Below is the complete code for the activity_main.xml file.

Code for activity_main.xml :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <EditText
        android:layout_width="248dp"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_weight="0.5"
        android:inputType="textPersonName"
        android:hint="Search Location" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:onClick="searchLocation"
        android:text="Search" />
</LinearLayout>

```

Step 4: Working with MainActivity.java file : Go to the app -> java -> com.example.(Package Name) -> MainActivity.java section. Below is the complete code for the MainActivity.java file.

Code for MainActivity.java :

```
package example.com.mapexample;

import android.location.Address;
import android.location.Geocoder;
import android.os.Build;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.location.LocationServices;

import android.location.Location;
import android.Manifest;
import android.content.pm.PackageManager;
import android.support.v4.content.ContextCompat;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;

import java.io.IOException;
import java.util.List;
```

```

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
LocationListener, GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener{

private GoogleMap mMap;
Location mLastLocation;
Marker mCurrLocationMarker;
GoogleApiClient mGoogleApiClient;
LocationRequest mLocationRequest;

@Override protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_maps);
// Obtain the SupportMapFragment and get notified when the map is ready to be used.
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
.findFragmentById(R.id.map);
mapFragment.getMapAsync(this); }

@Override public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;

if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
buildGoogleApiClient();
mMap.setMyLocationEnabled(true); } }
else {
buildGoogleApiClient();
mMap.setMyLocationEnabled(true);
} }

protected synchronized void buildGoogleApiClient() {
mGoogleApiClient = new GoogleApiClient.Builder(this)

```

```
.addConnectionCallbacks(this)
.addOnConnectionFailedListener(this)
.addApi(LocationServices.API).build();
mGoogleApiClient.connect(); }
```

```
@Override public void onConnected(Bundle bundle) {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(1000);
    mLocationRequest.setFastestInterval(1000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest,
this);
    } }
```

```
@Override public void onConnectionSuspended(int i) { }
```

```
@Override public void onLocationChanged(Location location) {
    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }
    //Place current location marker
    LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(latLng);
    markerOptions.title("Current Position");
    markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
    mCurrLocationMarker = mMap.addMarker(markerOptions);

    //move map camera
    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
    mMap.animateCamera(CameraUpdateFactory.zoomTo(11));
}
```

```

//stop location updates
if (mGoogleApiClient != null) {
    LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
} }

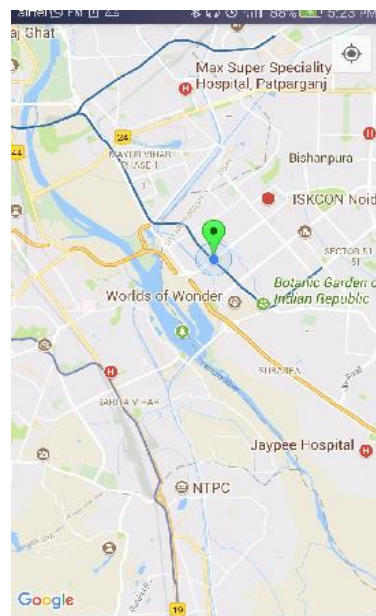
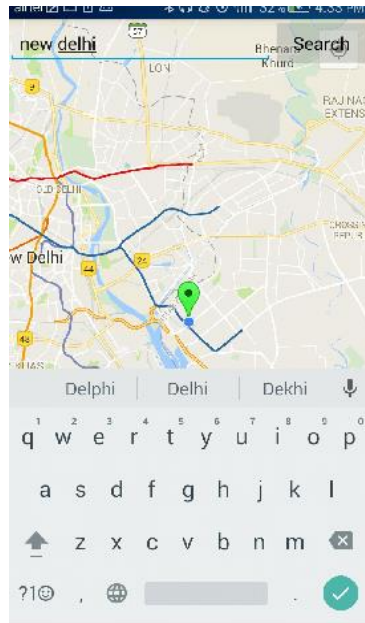
@Override public void onConnectionFailed(ConnectionResult connectionResult) { }

public void searchLocation(View view) {
    EditText locationSearch = (EditText) findViewById(R.id.editText);
    String location = locationSearch.getText().toString();
    List<Address> addressList = null;
    String uri = String.format(Locale.ENGLISH,
"https://www.google.com/maps/search/?api=1&query="+location");
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
    intent.setPackage("com.google.android.apps.maps");
    try
    {
        startActivity(intent);
    }
    catch(ActivityNotFoundException ex)
    {
        try
        {
            Intent unrestrictedIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
            startActivity(unrestrictedIntent);
        }
        catch(ActivityNotFoundException innerEx)
        {
            Toast.makeText(this, "Please install a maps application", Toast.LENGTH_LONG).show();
        }
    }
}

\Address address = addressList.get(0);
LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());
mMap.addMarker(new MarkerOptions().position(latLng).title(location));
mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
Toast.makeText(getApplicationContext(),address.getLatitude()+" "+address.getLongitude(),Toast.LENGTH_LONG).show(); } } }

```

Output:



Conclusion:- In this Practical i learned to implement a program to send user from one application to another. (For example redirection to map).

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 4

Date : 06 - 05 - 2021

Title : Write a program to play audio files.

Aim : To Implement a program to play audio files.

Theory:

Audio Player in Android : We can play and control the audio files in android by the help of MediaPlayer class. To play audio or video files in Android, the Android multimedia framework includes the support of MediaPlayer APIs. So, by using MediaPlayer APIs, you can play audio/video files from your Android filesystem or play files from your Application's resource file or even you can stream audio/video files just like Spotify.

Following are the operations that can be performed using MediaPlayer:

- **Prepare media file:** To play a media file, you need to first prepare it i.e. you need to load the file for playback. Methods used for doing this are prepare(), prepareAsync(), and setDataSource().
- **Start/Pause the playback:** After loading the media file, you can start playing the media file by using the start() method. Similarly, you can pause the playing media file by using the pause() method.
- **Stop the playback:** You can stop playback by using the reset() method. This method is used when we want to stop playback and then prepare another playback for playing.
- **Seek to a position:** You can directly skip to a particular position(in ms) in a media file by using the seekTo(position) method.
- **Length of song:** You can find the length of a particular song(in ms) by using the getDuration() method.
- **The current position of a song:** You can get the current position of playback by using the getCurrentPosition() method.
- **Resource deallocation:** You can deallocate the memory and resources used by the media player by using the release() method.

Program Steps :

Step 1: Create a New Project : To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Working with activity_main.xml file : Go to the app -> res -> layout -> activity_main.xml section and set the layout for the app. Below is the complete code for the activity_main.xml file.

Code for activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <!--Button for playing audio-->
    <Button
        android:id="@+id/idBtnPlay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Play Audio file"
        android:textAllCaps="false" />

    <!--Button for pausing the audio-->
    <Button
        android:id="@+id/idBtnPause"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/idBtnPlay"
        android:layout_centerInParent="true"
        android:text="Pause Audio"
        android:textAllCaps="false" />
```

</RelativeLayout>

Step 3: Working with MainActivity.java file : Go to the app -> java -> com.example.(Package Name) -> MainActivity.java section. Below is the complete code for the MainActivity.java file.

Code for MainActivity.java :

```
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {
    // creating a variable for button and media player
    Button playBtn, pauseBtn;
    MediaPlayer mediaPlayer;

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

        // initializing our buttons
        playBtn = findViewById(R.id.idBtnPlay);
        pauseBtn = findViewById(R.id.idBtnPause);

        // setting on click listener for our play and pause buttons.
        playBtn.setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View v) { playAudio(); // calling method to play audio.
            } });
        pauseBtn.setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View v) {
```

```

        // checking the media player if the audio is playing or not.
        if (mediaPlayer.isPlaying()) {
            //pausing the media player if media player is playing we are calling below line to stop our player
            mediaPlayer.stop();
            mediaPlayer.reset();
            mediaPlayer.release();
        }
        // below line is to display a message when media player is paused.
        Toast.makeText(MainActivity.this, "Audio has been paused", Toast.LENGTH_SHORT).show();
    } else { // this method is called when media player is not playing.

        Toast.makeText(MainActivity.this, "Audio has not played", Toast.LENGTH_SHORT).show();
    }
}

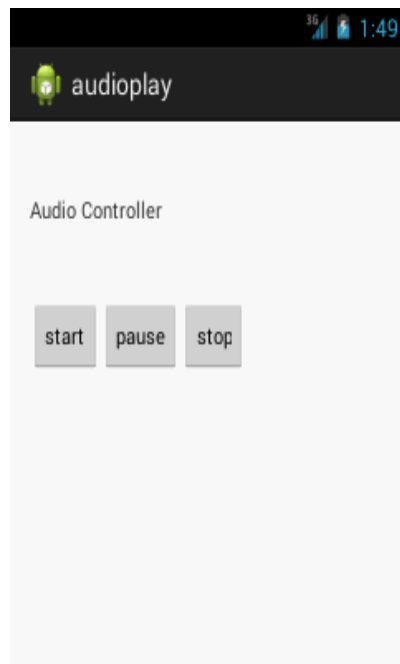
private void playAudio() {
    String audioUrl = "https://www.soundhelix.com/examples/mp3/SoundHelix-Song-1.mp3";
    // initializing media player
    mediaPlayer = new MediaPlayer();

    // below line is use to set the audio stream type for our media player.
    mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);

    // below line is use to set our url to our media player.
    try {
        mediaPlayer.setDataSource(audioUrl);
        // below line is use to prepare
        // and start our media player.
        mediaPlayer.prepare();
        mediaPlayer.start();
    } catch (IOException e) { e.printStackTrace(); }
    // below line is use to display a toast message.
    Toast.makeText(this, "Audio started playing..", Toast.LENGTH_SHORT).show();
}
}

```

Output:



Conclusion:- In this Practical i learned to implement a program to play audio files.

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 5

Date : 13 - 05 - 2021

Title : Write a program to play video files

Aim : To Implement a program to play video files

Theory:

Android Video Player : By the help of MediaController and VideoView classes, we can play the video files in android.

1) MediaController class : The android.widget.MediaController is a view that contains media controls like play/pause, previous, next, fast-forward, rewind etc.

2) VideoView class : The android.widget.VideoView class provides methods to play and control the video player. The commonly used methods of VideoView class are as follows:

Program Steps :

Step 1: Create a New Project : To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Working with activity_main.xml file : Go to the app -> res -> layout -> activity_main.xml section and set the layout for the app. Below is the complete code for the activity_main.xml file.

Code for activity_main.xml :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <VideoView
        android:id="@+id/videoView1"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true" />
</RelativeLayout>
```

Step 3: Working with MainActivity.java file : Go to the app -> java -> com.example.(Package Name) -> MainActivity.java section. Below is the complete code for the MainActivity.java file.

Code for MainActivity.java :

```
package com.example.video1;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends Activity {
    @Override protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        VideoView videoView =(VideoView)findViewById(R.id.videoView1);

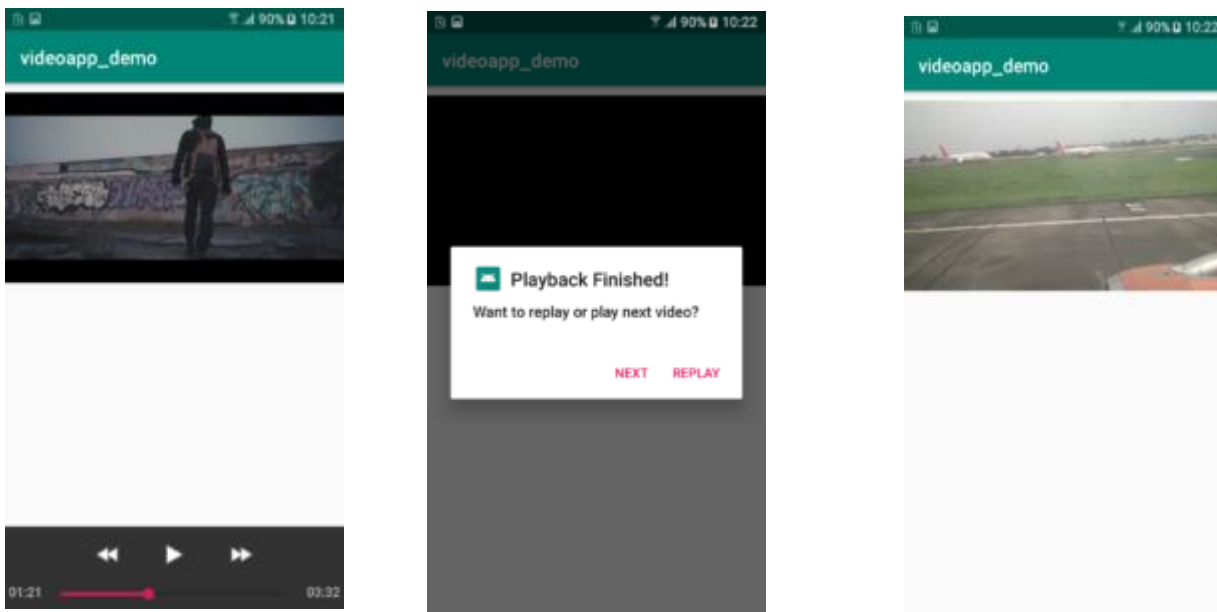
        //Creating MediaController
        MediaController mediaController= new MediaController(this);
        mediaController.setAnchorView(videoView);

        //specify the location of media file
        Uri uri=Uri.parse(Environment.getExternalStorageDirectory().getPath()+"/media/1.mp4");

        //Setting MediaController and URI, then starting the videoView
        videoView.setMediaController(mediaController);
        videoView.setVideoURI(uri);
        videoView.requestFocus();
        videoView.start();
    }
}
```

```
@Override public boolean onCreateOptionsMenu(Menu menu)
{ // Inflate the menu; this adds items to the action bar if it is present.
  getMenuInflater().inflate(R.menu.activity_main, menu);
  return true;
} }
```

Output:



Conclusion:- In this Practical i learned to implement a program to play video files

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 6

Date : 27 - 05 - 2021

Title : Write a program to capture image using built in camera.

Aim : To Implement a program to capture image using built in camera.

Theory:

Android Camera : Camera is mainly used to capture picture and video. We can control the camera by using methods of camera api. Android provides the facility to work on camera by 2 ways:

- By Camera Intent (Built-in Camera)
- By Camera API

Understanding basic classes of Camera Intent and API : There are mainly four classes that we are going to discuss.

1. Intent : By the help of 2 constants of MediaStore class, we can capture picture and video without using the instance of Camera class.
 - ACTION_IMAGE_CAPTURE
 - ACTION_VIDEO_CAPTURE
2. Camera : It is main class of camera api, that can be used to take picture and video.
3. SurfaceView : It represents a surface view ore preview of live camera.
4. MediaRecorder : It is used to record video using camera. It can also be used to record audio files as we have seen in the previous example of media framework.

Program Steps :

Step 1: Create a New Project : To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Working with activity_main.xml file : Go to the app -> res -> layout -> activity_main.xml section and set the layout for the app. Below is the complete code for the activity_main.xml file.

Code for activity_main.xml :


```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="Take a Photo" >

    </Button>

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_above="@+id/button1"
        android:layout_alignParentTop="true"
        android:src="@drawable/ic_launcher" >

    </ImageView>

</RelativeLayout>

```

Step 3: Working with MainActivity.java file : Go to the app -> java -> com.example.(Package Name) -> MainActivity.java section. Below is the complete code for the MainActivity.java file.

Code for MainActivity.java :

```

package com.example.simplecamera;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;

```

```

import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends Activity {
    private static final int CAMERA_REQUEST = 1888;
    ImageView imageView;

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView = (ImageView) this.findViewById(R.id.imageView1);
        Button photoButton = (Button) this.findViewById(R.id.button1);

        photoButton.setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View v) {
                Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(cameraIntent, CAMERA_REQUEST);
            } });

        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
            if (requestCode == CAMERA_REQUEST) {
                Bitmap photo = (Bitmap) data.getExtras().get("data");
                imageView.setImageBitmap(photo);
            }

            @Override public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.activity_main, menu);
                return true;
            }
        }
    }
}

```

Output:



Conclusion:- In this Practical i learned to implement a program to capture image using built in camera.

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 7

Date : 03 - 06 - 2021

Title : Write a program to send SMS.

Aim : To Implement a program to send SMS.

Theory:

Android - Sending SMS : In Android, you can use SmsManager API or devices Built-in SMS application to send SMS's. **SMSManager** class manages operations like sending a text message, data message, and multimedia messages (MMS). For sending a text message method `sendTextMessage()` is used likewise for multimedia message `sendMultimediaMessage()` and for data message `sendDataMessage()` method is used. Below is an example of a basic application that sends a text message.

Program Code :

Step 1: Create a new Android Application.

Step 2: Go to AndroidManifest.xml : app->Manifest->AndroidManifest.xml

Step 3: In AndroidManifest.xml add the permission to send SMS. It will permit an android application to send SMS : `<uses-permission android:name="android.permission.SEND_SMS" />`

Code for AndroidManifest.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:dist="http://schemas.android.com/apk/distribution"
    package="com.example.gfg">
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <dist:module dist:instant="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
```

```

        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
</application>

</manifest>

```

Step 4: Open activity_main.xml from the following address and add the below code. Here, in a Linear Layout, two edittext for taking phone number and a text message and a button for sending the message is added. : app->res->layout->activitymain.xml

Code for activitymain.xml :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_marginTop="140dp"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter number"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter message"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="60dp"
        android:layout_marginRight="60dp"
        android:text="SEND" />
</LinearLayout>

```

Step 5: Open MainActivity.java : app->java->com.example.gfg->MainActivity

Create objects for views used i.e., editTexts and button. In the onCreate method find all the views using the findViewById() method. Since the Send button is for sending the message so onClickListener is added with the button. Now create two string variables and store the value of editText phone number and message into them using method getText() (before assigning convert them to a string using toString() method). Now in try block create an instance of SmsManager class and get the SmsManager associated with the default subscription id. Now call method sendTextMessage() to send message.

```
SmsManager smsManager=SmsManager.getDefault();
smsManager.sendTextMessage(number,null,msg,null,null);
```

Then display the toast message as “Message sent ” and close the try block. At last in catch block display a toast message because the message was not sent if the compiler executes this block of the code.

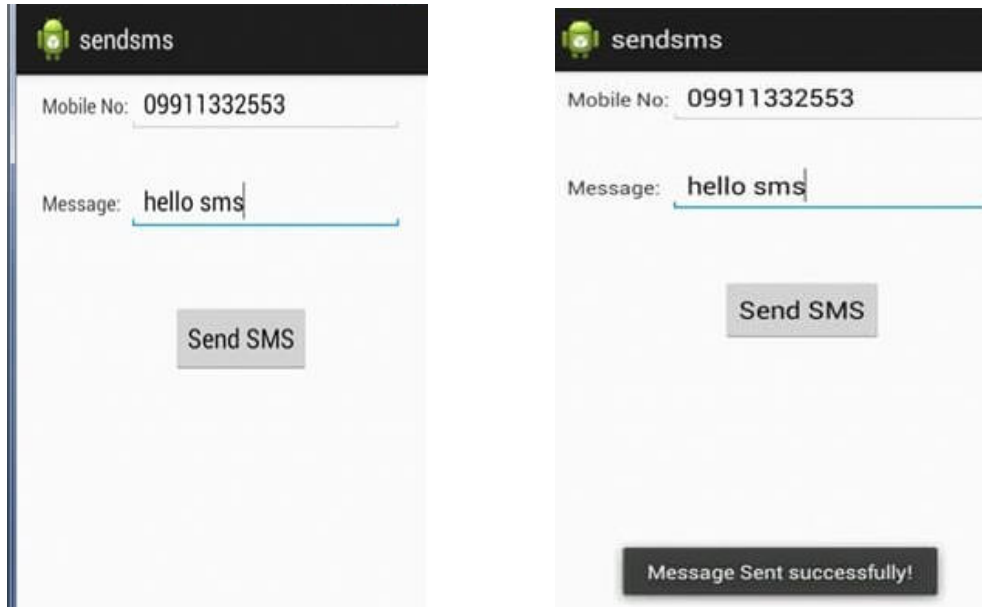
Code for MainActivity.java :

```
package com.example.gfg;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText phonenumber,message;
    Button send;
    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        send=findViewById(R.id.button);
        phonenumber=findViewById(R.id.editText);
        message=findViewById(R.id.editText2);
        send.setOnClickListener(new View.OnClickListener() {

            public void onClick(View view) {
                String number=phonenumber.getText().toString();
                String msg=message.getText().toString();
                try {
                    SmsManager smsManager=SmsManager.getDefault();
                    smsManager.sendTextMessage(number,null,msg,null,null);
                    Toast.makeText(getApplicationContext(),"MessageSent",Toast.LENGTH_LONG).show();
                }catch (Exception e)
                {
                    Toast.makeText(getApplicationContext(),"Some fiels is Empty",Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

Output:



Conclusion:- In this Practical i learned to implement a program to send SMS.

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class : T. Y. Btech. (Comp.)

Practical No. : 8

Date : 17 - 06 - 2021

Title : Write a program to convert text to speech.

Aim : To Implement a program to convert text to speech.

Theory:

Text to Speech in Android : Android allows you convert your text into voice. Not only you can convert it but it also allows you to speak text in variety of different languages. Android provides TextToSpeech class for this purpose. In order to use this class, you need to instantiate an object of this class and also specify the initListener. Its syntax is given below –

private EditText write;

```
ttobj=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener(){  
@Override public void onInit(int status) {}  
});
```

Text to Speech App converts the text written on the screen to speech like you have written “Hello World” on the screen and when you press the button it will speak “Hello World”. Text-to-speech is commonly used as an accessibility feature to help people who have trouble reading on-screen text, but it’s also convenient for those who want to be read too. This feature has come out to be a very common and useful feature for the users.

Program Steps for Converting Text to Speech in Android :

Step 1: Create a New Project : To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Working with activity_main.xml file : Go to the app -> res -> layout -> activity_main.xml section and set the layout for the app. In this file add an EditText to input the text from the user, a Button, so whenever the user clicks on the Button then it’s converted to speech. Below is the complete code for the activity_main.xml file.

Code for activity_main.xml :


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="30dp"
    tools:context=".MainActivity">
```

```
<!--To add text in the app-->
```

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/Text"
    android:layout_marginBottom="20dp"
    android:hint="Enter Any Sentence"
    android:gravity="center"
    android:textSize="16dp"/>
```

```
<!--when you press this button it will
convert text into speech-->
```

```
<Button
    android:layout_width="wrap_content"
    android:id="@+id/btnText"
    android:layout_height="wrap_content"
    android:text="Click Here"
    android:layout_gravity="center"/>
```

```
<!--To display the name of GeeksForGeeks -->
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:layout_marginTop="70dp"
android:gravity="center_horizontal"
android:text="Shweta Patil"
android:textColor="@android:color/holo_green_dark"
android:textSize="36sp" />
```

```
</LinearLayout>
```

Step 3: Working with MainActivity.java file : Go to the app -> java -> com.example.(Package Name) -> MainActivity.java section. Now join the Button and EditText to Java code and comments are added inside code to understand the code easily. Below is the complete code for the MainActivity.java file.

Code for MainActivity.java :

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    EditText Text;
    Button btnText;
    TextToSpeech textToSpeech;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Text = findViewById(R.id.Text);
        btnText = findViewById(R.id.btnText);

        // create an object textToSpeech and adding features into it
        textToSpeech = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
```

```

@Override public void onInit(int i) {

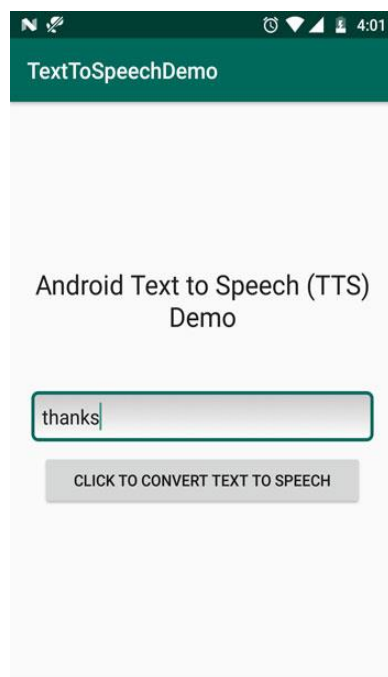
    // if No error is found then only it will run
    if(i!=TextToSpeech.ERROR){
        // To Choose language of speech
        textToSpeech.setLanguage(Locale.UK);
    }
}

});

// Adding OnClickListener
btnText.setOnClickListener(new View.OnClickListener() {
    @Override public void onClick(View view) {
        textToSpeech.speak(Text.getText().toString(),TextToSpeech.QUEUE_FLUSH,null);
    } });
}

```

Output:



Conclusion:- In this Practical i learned to implement a program to convert text to speech.

Godavari College Of Engineering, Jalgaon.

Subject Name: MAD

Class: T. Y. Btech. (Comp.)

Practical No. : 9

Date : 19 - 06 - 2021

Title : Write a program to call a number.

Aim : To Implement a program to call a number.

Theory:

Intent Object : We are able to make a phone call in android via intent. You need to write only three lines of code to make a phone call.

```
Intent callIntent = new Intent(Intent.ACTION_CALL);  
callIntent.setData(Uri.parse("tel:"+8802177690));//change the number  
startActivity(callIntent);
```

Basically **Intent** is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers and services, here use to make phone call. This application basically contain one activity with edit text to write phone number on which you want to make a call and button to call that number.

You will use **ACTION_CALL** action to trigger built-in phone call functionality available in Android device. Following is simple syntax to create an intent with **ACTION_CALL** action. You can use **ACTION_DIAL** action instead of **ACTION_CALL**, in that case you will have option to modify hardcoded phone number before making a call instead of making a direct call.

```
Intent phoneIntent = new Intent(Intent.ACTION_CALL);
```

To make a phone call at a given number 91-000-000-0000, you need to specify tel: as URI using setData() method as follows. The interesting point is that, to make a phone call, you do not need to specify any extra data or data type.

```
phoneIntent.setData(Uri.parse("tel:91-000-000-0000"));
```

Program:-

Step 1. Permission code in Android-Manifest.xml file

You need to take permission from user for phone call and for that **CALL_PHONE** permission is added in manifest file. Here is code of manifest file:

Code :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.geeksforgeeks.phonecall"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />
    <!--permission for phone call-->
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/gfg"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.geeksforgeeks.phonecall.MainActivity"
            android:label="@string/gfg" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Step 2. activity_main.xml

activity_main.xml contains a [Relative Layout](#) which contains Edit text to write phone number on which you want to make phone call and button for starting intent or making call :

code :

```
<?xml version="1.0" encoding="utf-8"?>
<!--Relative Layout-->
```

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <!--Edit text for phone number-->
    <EditText
        android:id="@+id/editText"
        android:layout_marginTop="30dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
    />
    <!--Button to make call-->
    <Button
        android:id="@+id/button"
        android:layout_marginTop="115dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Make Call!!"
        android:padding="5dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"    />
</RelativeLayout>

```

Step 3. MainActivity.java

In Main activity Intent object is created to redirect activity to call manager and action attribute of intent is set as ACTION_CALL. Phone number input by user is parsed through Uri and that is passed as data in Intent object which is then used to call that phone number. **setOnClickListener** is attached to button with intent object in it to make intent with action as ACTION_CALL to make phone call. Here is complete code.

code :

```
package com.geeksforgeeks.phonecall;
```

```

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.content.Intent;
import android.widget.EditText;
import android.view.View;
import android.view.View.OnClickListener;
import android.net.Uri;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    // define objects for edit text and button
    EditText edittext;
    Button button;

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

        // Getting instance of edittext and button
        button = findViewById(R.id.button);
        edittext = findViewById(R.id.editText);

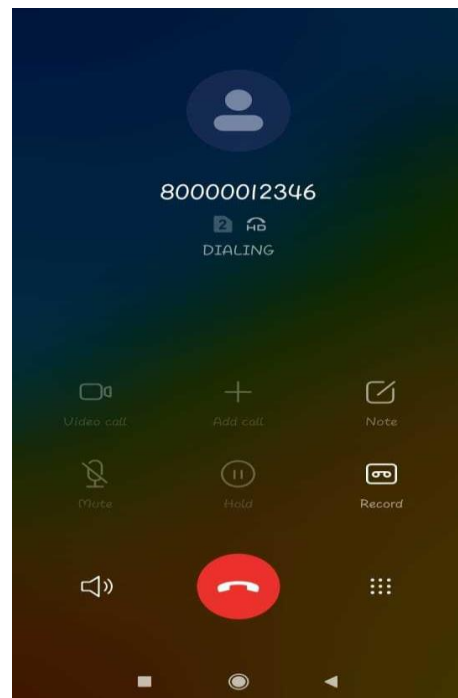
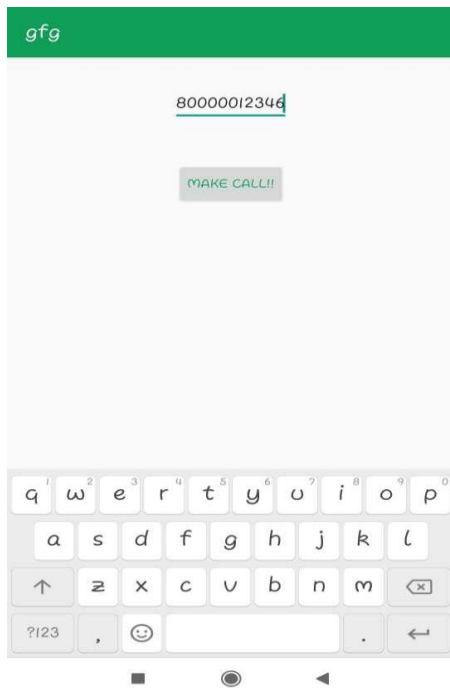
        // Attach set on click listener to the button for initiating intent
        button.setOnClickListener(new OnClickListener() {
            @Override public void onClick(View arg)
            {
                // getting phone number from edit text and changing it to String
                String phone_number= edittext.getText().toString();

                // Getting instance of Intent with action as ACTION_CALL
                Intent phone_intent= new Intent(Intent.ACTION_CALL);

                // Set data of Intent through Uri by parsing phone number
                phone_intent.setData(Uri.parse("tel:"+ phone_number));
                startActivity(phone_intent); // start Intent
            }
        });
    }
}

```

Output:



Conclusion:- In this Practical i learned to implement a program to call a number.