Experiment 9: Multimedia- Video Player

1. Objective

Create an app that can play video files. This will delve deeper into multimedia APIs and file handling in Android.

2. Steps to Complete the Experiment

1. Prepare Video Files:

   Place your video files in the res/raw folder of your project. Ensure they are in a supported format, such as MP4.

2. Design the UI:

   Use a VideoView in your activity's layout to display the video content.

   Include playback controls such as Play, Pause, and Stop buttons. Optionally, add a SeekBar for video progress and TextViews for displaying current time and total duration.

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

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/videoView"
    android:orientation="horizontal"
    android:gravity="center"
    android:layout_marginTop="16dp">

    <Button
        android:id="@+id/buttonPlay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Play" />

    <Button
        android:id="@+id/buttonPause"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pause" />
```

```
        <Button
            android:id="@+id/buttonStop"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Stop" />
    </LinearLayout>

    <!-- Optional SeekBar for video progress -->
    <SeekBar
        android:id="@+id/seekBarVideo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/linearLayout"
        android:layout_marginTop="16dp" />

</RelativeLayout>
```

3. Initialize VideoView:

   In MainActivity.java, obtain a reference to the VideoView and set the path of the video file using videoView.setVideoURI(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.your_video_file)).

4. Implement Playback Controls:

   Set OnClickListeners for the Play, Pause, and Stop buttons. Use methods like start(), pause(), and stopPlayback() on the VideoView to control playback.

5. Implement SeekBar Functionality (Optional):

If using a SeekBar, synchronize it with the video playback. This involves updating the SeekBar position as the video plays and seeking in the video when the user interacts with the SeekBar.

6. Handle Video Completion:

   Use setOnCompletionListener on the VideoView to handle actions to take when the video finishes playing, such as resetting the play position or updating UI elements.

```
package com.yourpackage.name; // Replace with your actual package name

import androidx.appcompat.app.AppCompatActivity;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

    private VideoView videoView;
    private Button playButton, pauseButton, stopButton;
    private SeekBar seekBarVideo;
    private Handler handler = new Handler();
```

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

        videoView = findViewById(R.id.videoView);
        playButton = findViewById(R.id.buttonPlay);
        pauseButton = findViewById(R.id.buttonPause);
        stopButton = findViewById(R.id.buttonStop);
        seekBarVideo = findViewById(R.id.seekBarVideo);

        // Set video URI (Replace R.raw.your_video with your
actual video file)
        Uri videoUri = Uri.parse("android.resource://" +
getPackageName() + "/" + R.raw.your_video);
        videoView.setVideoURI(videoUri);

        playButton.setOnClickListener(v -> videoView.start());
        pauseButton.setOnClickListener(v -> videoView.pause());
        stopButton.setOnClickListener(v ->
videoView.stopPlayback());

        videoView.setOnPreparedListener(mp -> {
            seekBarVideo.setMax(videoView.getDuration());
            updateSeekBar();
        });
```

```
        seekBarVideo.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {
                if (fromUser) {
                    videoView.seekTo(progress);
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                // Implementation not necessary for this example
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                // Implementation not necessary for this example
            }
        });
    }

    private void updateSeekBar() {

seekBarVideo.setProgress(videoView.getCurrentPosition());
```

```
        if (videoView.isPlaying()) {
            Runnable updater = this::updateSeekBar;
            handler.postDelayed(updater, 1000); // Delay 1
second
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (videoView != null) {
            videoView.stopPlayback();
        }
    }
}
```

7. Testing:

   Thoroughly test the video player functionality, ensuring that all controls work as expected and the video plays smoothly. Pay special attention to the synchronization between the SeekBar (if implemented) and the video playback.

3. Explanation

VideoView (videoView): The main component for displaying the video. It is set to match the parent's width and wrap its content for height, centered within the parent layout.

LinearLayout: A container for the playback control buttons, ensuring they are horizontally aligned and centered below the VideoView.

Play, Pause, Stop Buttons (buttonPlay, buttonPause, buttonStop): When clicked, these control the playback of the video using the VideoView.

SeekBar (seekBarVideo): Optionally included to allow users to see the video progress and seek to different positions within the video. It's placed below the control buttons.

Video Initialization: The VideoView is set up with a video from the res/raw folder using its URI. Replace R.raw.your_video with your actual video file's resource ID.

Playback Controls: The Play, Pause, and Stop buttons are linked to their respective actions on the VideoView. The start(), pause(), and stopPlayback() methods control the video playback.

SeekBar Synchronization: The SeekBar is synchronized with the video's current position. It's set to match the video's duration and updates as the video plays. User interaction with the SeekBar seeks the video to the corresponding position.

Video Preparation Listener: The setOnPreparedListener on the VideoView ensures that the SeekBar is only set up once the video is ready to play, preventing errors related to accessing video duration or current position prematurely.

Resource Cleanup: The onDestroy method ensures that video playback is stopped when the activity is destroyed, preventing potential resource leaks.