CSE 162 Mobile Computing Lab

Lab5b Media Recorder

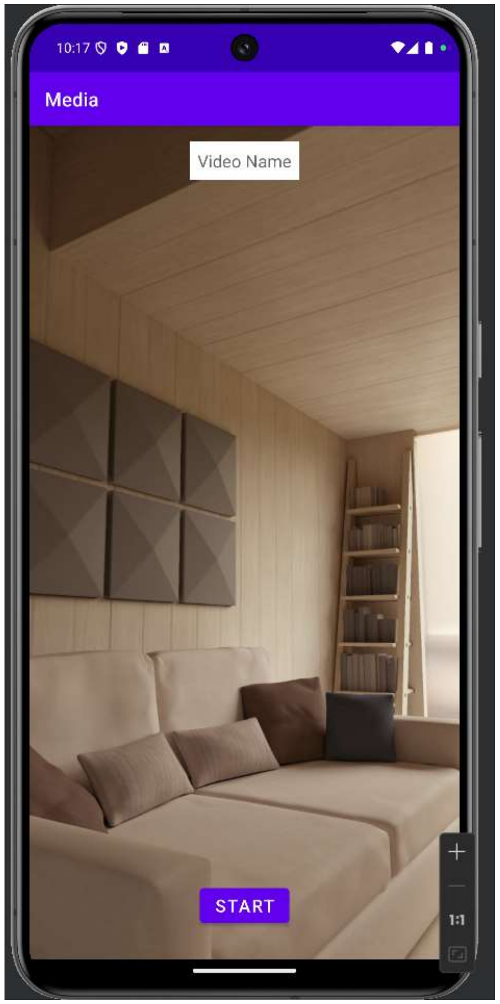
Department of Computer Science and Engineering
University of California, Merced, CA

Goal: achieve the following features

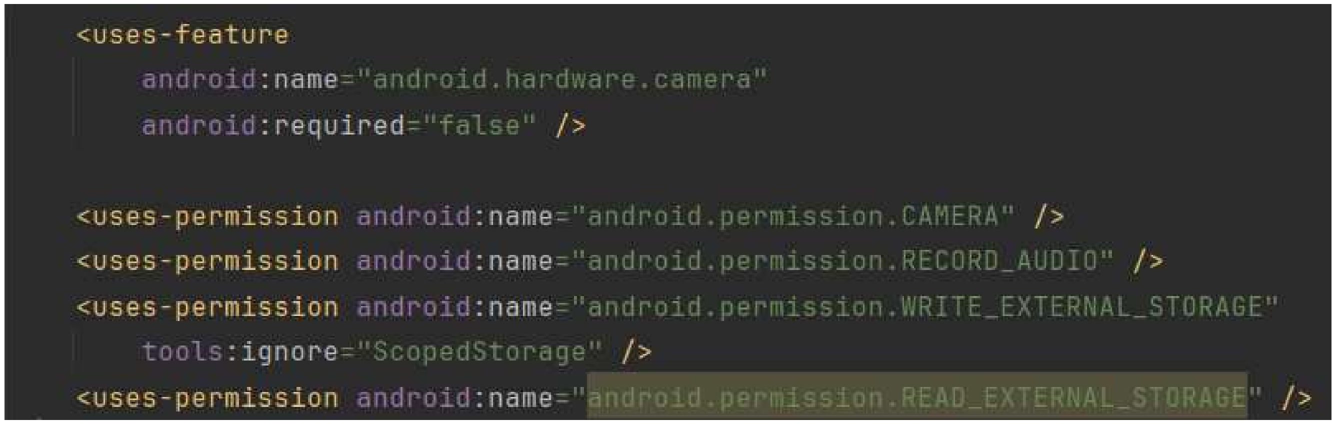
- Control the media recording capabilities
- Learn the MediaRecorder API
- Learn the Camera API

Outline

- create an app to shoot video
- Give it a name
- Push a button, the app begins to preview and record video
- Push the button again, save locally

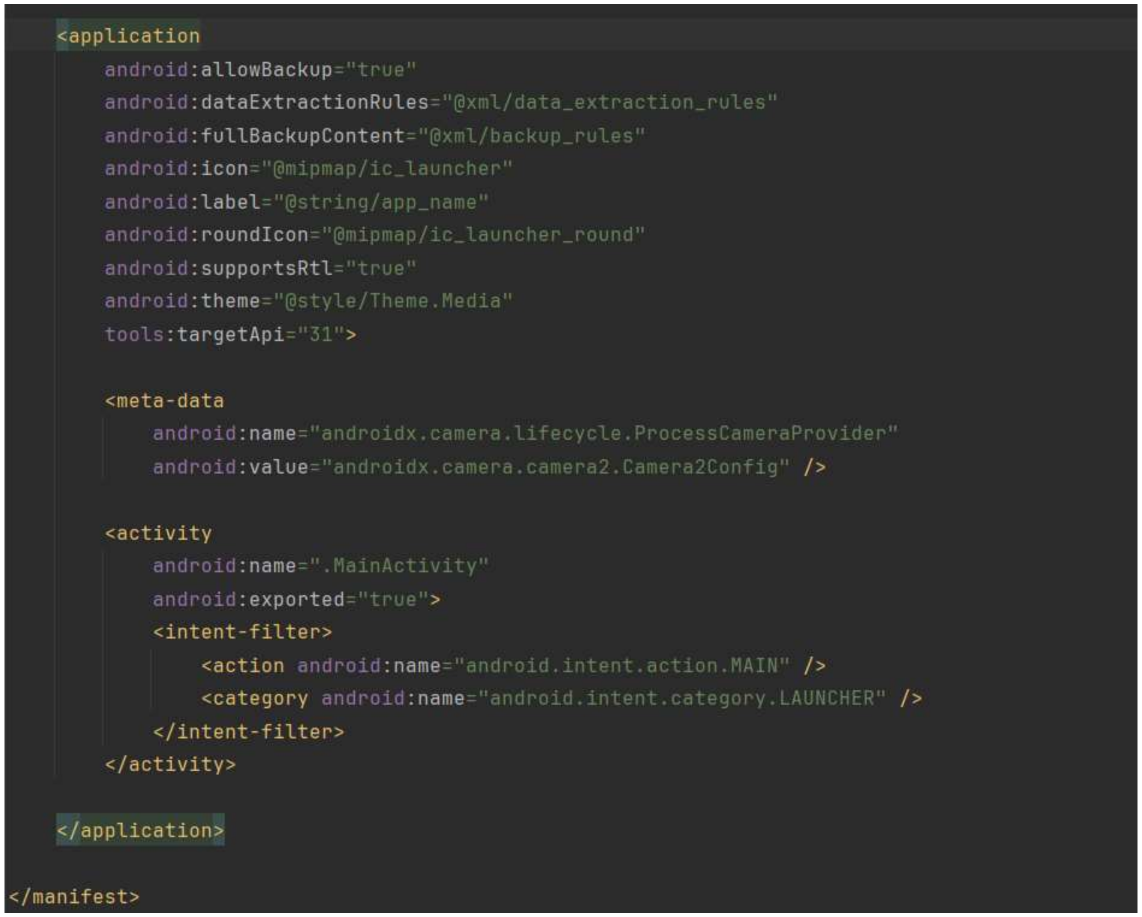


permission in the manifest file



```
<uses-feature
    android:name="android.hardware.camera"
    android:required="false" />

<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    tools:ignore="ScopedStorage" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

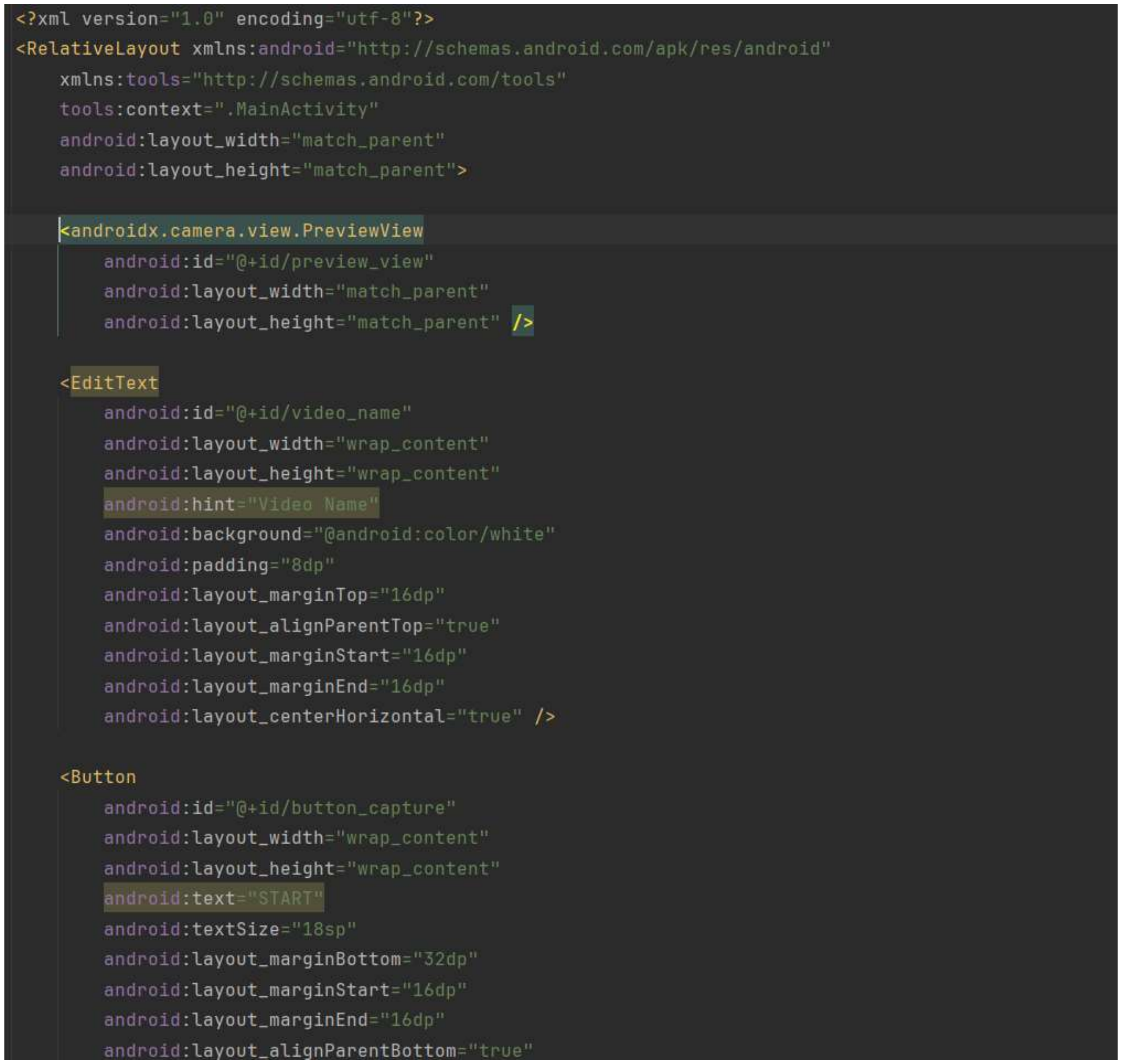
other parts of the manifest



```xml
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Media"
        tools:targetApi="31">

        <meta-data
            android:name="androidx.camera.lifecycle.ProcessCameraProvider"
            android:value="androidx.camera.camera2.Camera2Config" />

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

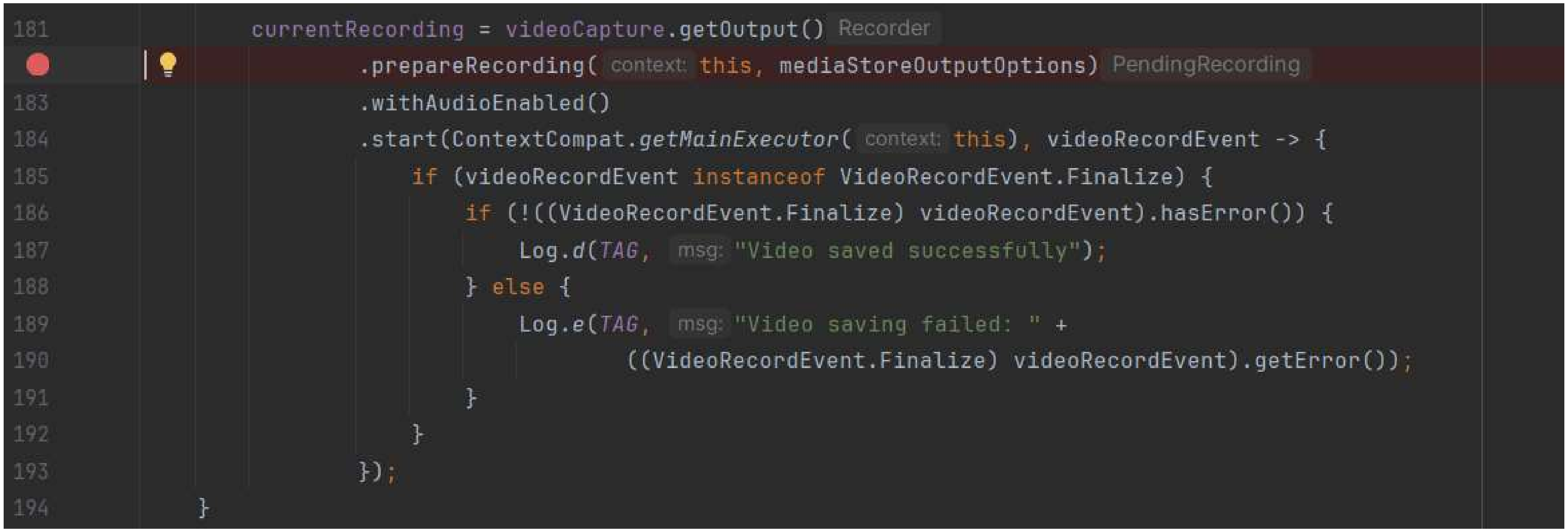
Prepare the UI

- in main_activity.xml, structured as follows
  - FrameLayout
    - Textureview
    - LinearLayout
      - EditText
      - Button



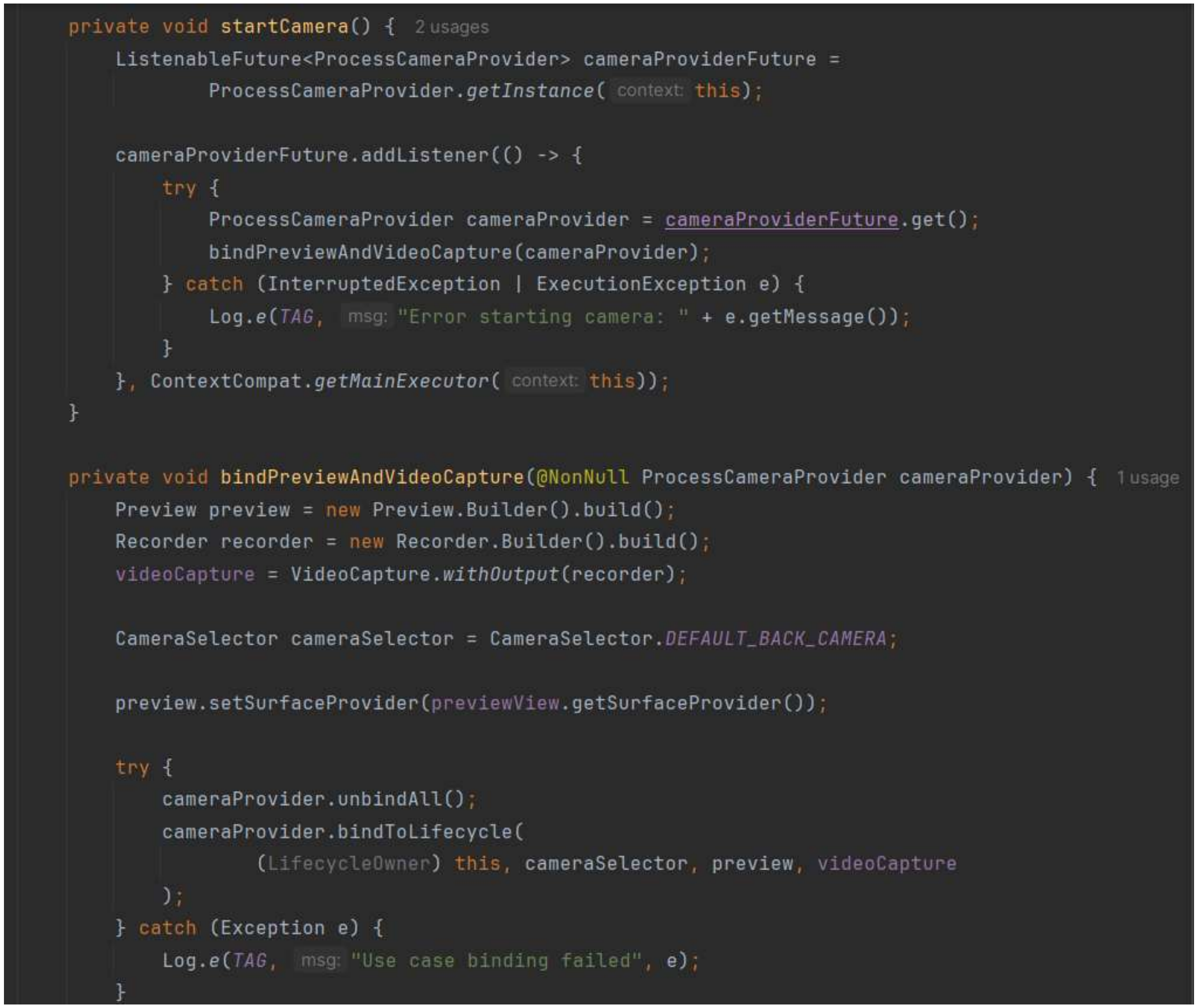
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.camera.view.PreviewView
        android:id="@+id/preview_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <EditText
        android:id="@+id/video_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Video Name"
        android:background="@android:color/white"
        android:padding="8dp"
        android:layout_marginTop="16dp"
        android:layout_alignParentTop="true"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_centerHorizontal="true" />

    <Button
        android:id="@+id/button_capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="START"
        android:textSize="18sp"
        android:layout_marginBottom="32dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_alignParentBottom="true"
```

prepare for video recording,
Prepare for the file storage,
videoCapture.getOutput()



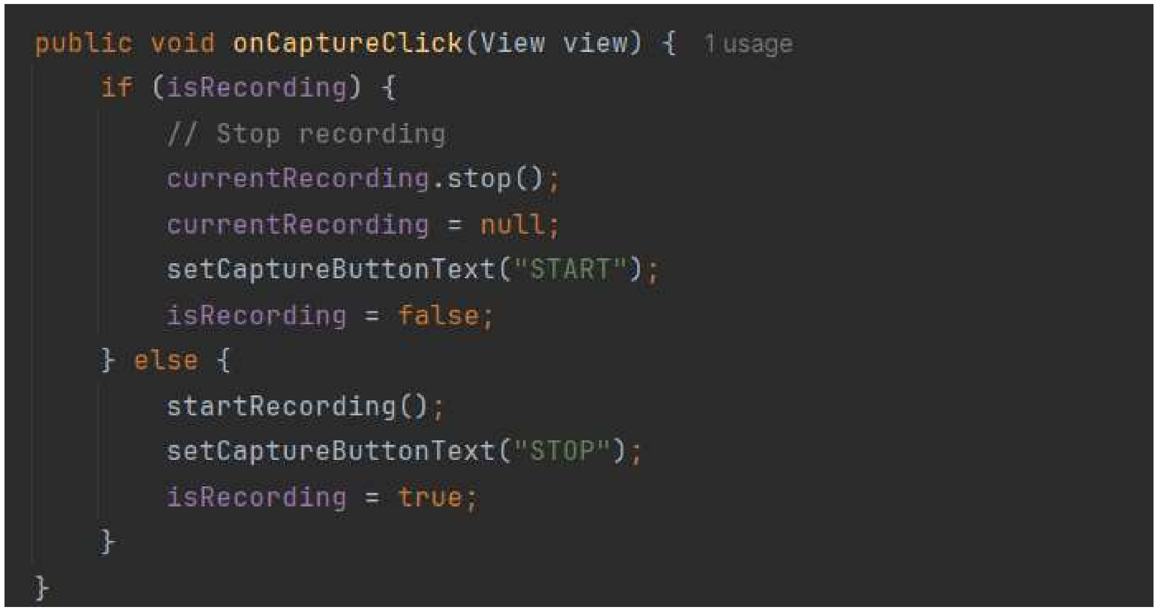
```
        currentRecording = videoCapture.getOutput() Recorder
                .prepareRecording( context: this, mediaStoreOutputOptions) PendingRecording
                .withAudioEnabled()
                .start(ContextCompat.getMainExecutor( context: this), videoRecordEvent -> {
                    if (videoRecordEvent instanceof VideoRecordEvent.Finalize) {
                        if (!((VideoRecordEvent.Finalize) videoRecordEvent).hasError()) {
                            Log.d(TAG,  msg: "Video saved successfully");
                        } else {
                            Log.e(TAG,  msg: "Video saving failed: " +
                                    ((VideoRecordEvent.Finalize) videoRecordEvent).getError());
                        }
                    }
                });
    }
```

Configure the camera



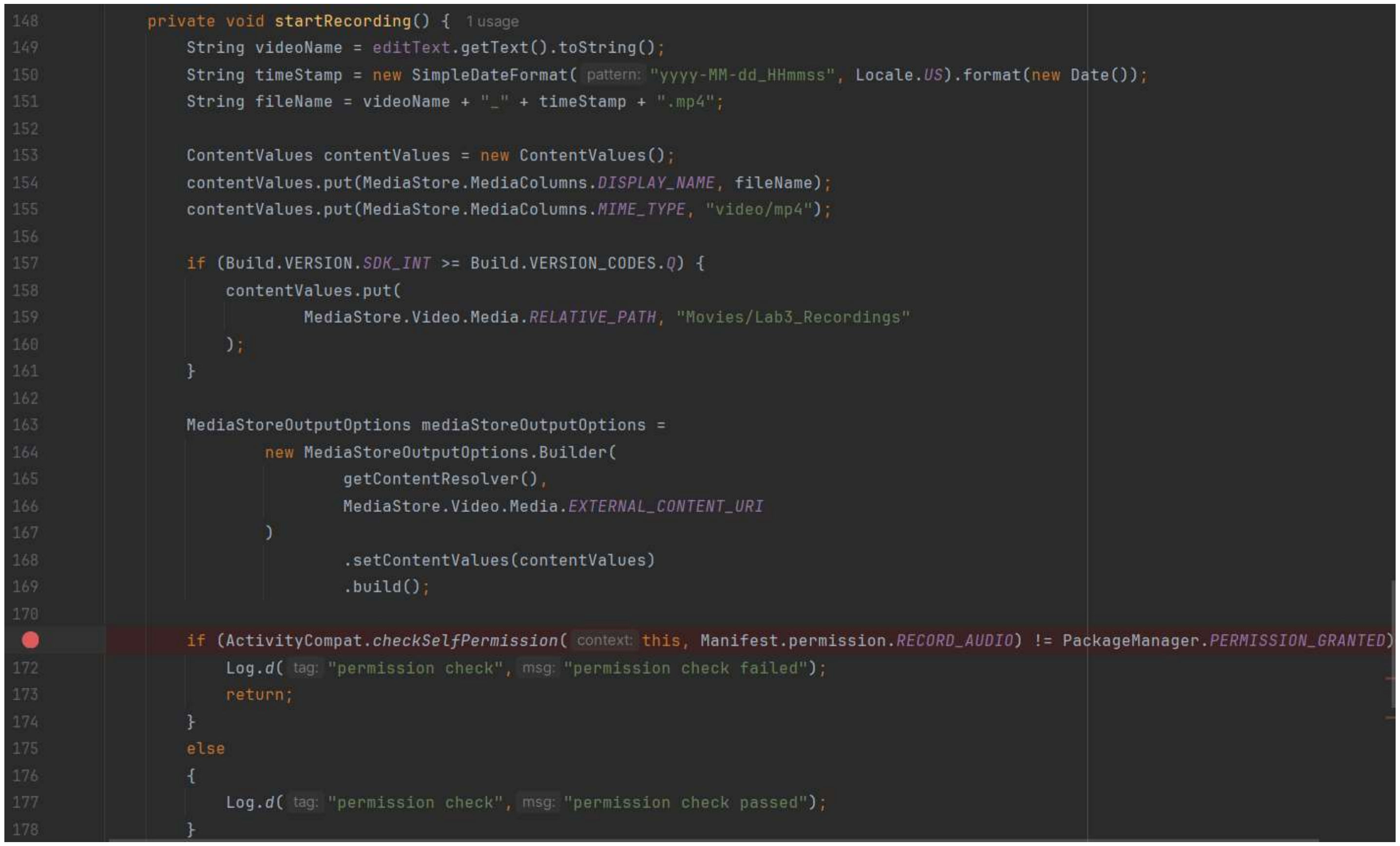
```
private void startCamera() {  2 usages
    ListenableFuture<ProcessCameraProvider> cameraProviderFuture =
            ProcessCameraProvider.getInstance( context: this);

    cameraProviderFuture.addListener(() -> {
        try {
            ProcessCameraProvider cameraProvider = cameraProviderFuture.get();
            bindPreviewAndVideoCapture(cameraProvider);
        } catch (InterruptedException | ExecutionException e) {
            Log.e(TAG,  msg: "Error starting camera: " + e.getMessage());
        }
    }, ContextCompat.getMainExecutor( context: this));
}

private void bindPreviewAndVideoCapture(@NonNull ProcessCameraProvider cameraProvider) {  1 usage
    Preview preview = new Preview.Builder().build();
    Recorder recorder = new Recorder.Builder().build();
    videoCapture = VideoCapture.withOutput(recorder);

    CameraSelector cameraSelector = CameraSelector.DEFAULT_BACK_CAMERA;

    preview.setSurfaceProvider(previewView.getSurfaceProvider());

    try {
        cameraProvider.unbindAll();
        cameraProvider.bindToLifecycle(
                (LifecycleOwner) this, cameraSelector, preview, videoCapture
        );
    } catch (Exception e) {
        Log.e(TAG,  msg: "Use case binding failed", e);
    }
```

Use the media recorder



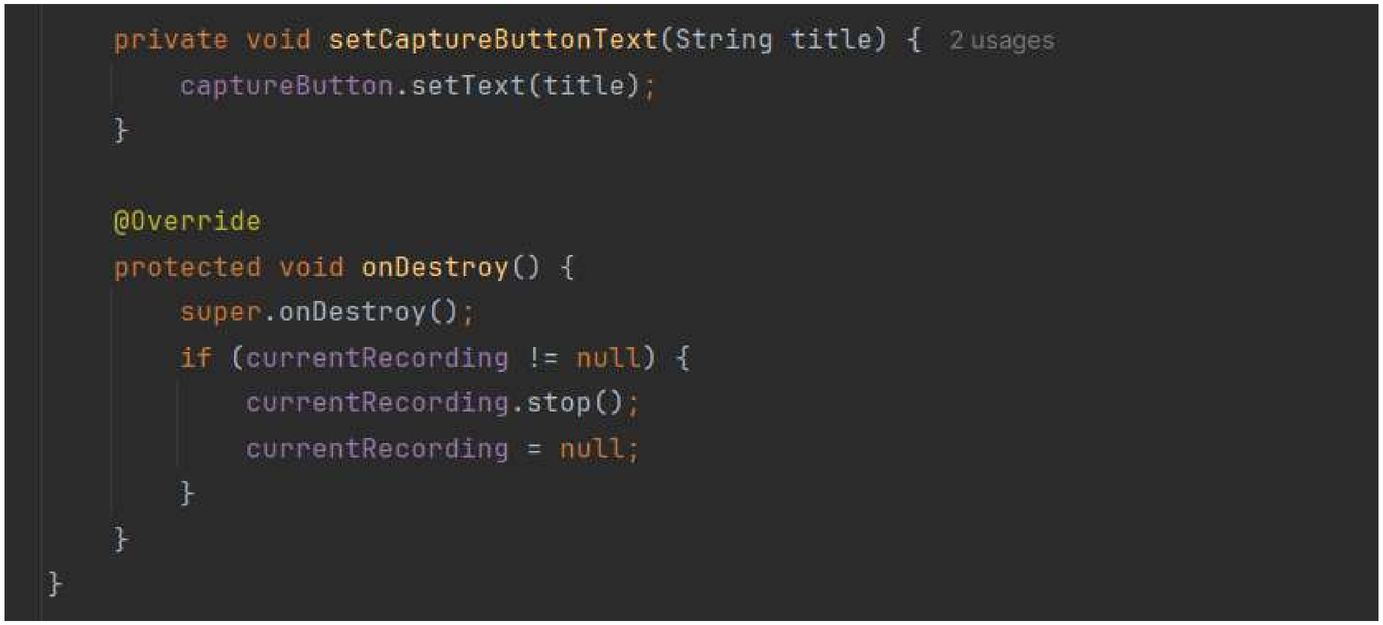
```
public void onCaptureClick(View view) {  1 usage
    if (isRecording) {
        // Stop recording
        currentRecording.stop();
        currentRecording = null;
        setCaptureButtonText("START");
        isRecording = false;
    } else {
        startRecording();
        setCaptureButtonText("STOP");
        isRecording = true;
    }
}
```

startRecording()



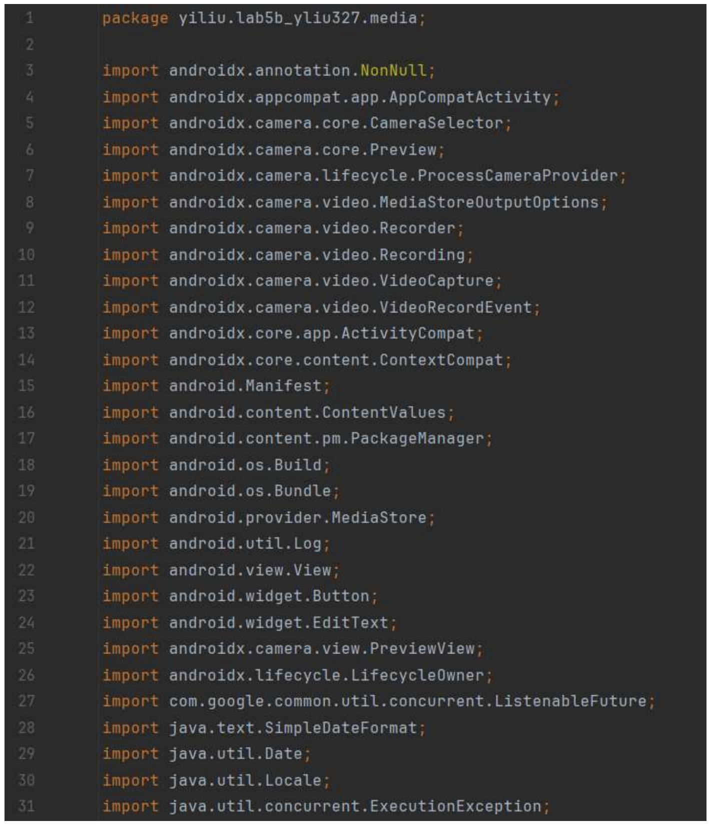
```java
private void startRecording() {  1 usage
    String videoName = editText.getText().toString();
    String timeStamp = new SimpleDateFormat( pattern: "yyyy-MM-dd_HHmmss", Locale.US).format(new Date());
    String fileName = videoName + "_" + timeStamp + ".mp4";

    ContentValues contentValues = new ContentValues();
    contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, fileName);
    contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "video/mp4");

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        contentValues.put(
                MediaStore.Video.Media.RELATIVE_PATH, "Movies/Lab3_Recordings"
        );
    }

    MediaStoreOutputOptions mediaStoreOutputOptions =
            new MediaStoreOutputOptions.Builder(
                    getContentResolver(),
                    MediaStore.Video.Media.EXTERNAL_CONTENT_URI
            )
                    .setContentValues(contentValues)
                    .build();

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.RECORD_AUDIO) != PackageManager.PERMISSION_GRANTED)
        Log.d( tag: "permission check", msg: "permission check failed");
        return;
    }
    else
    {
        Log.d( tag: "permission check", msg: "permission check passed");
    }
```

When the recording stops, release the camera and the mediarecorder



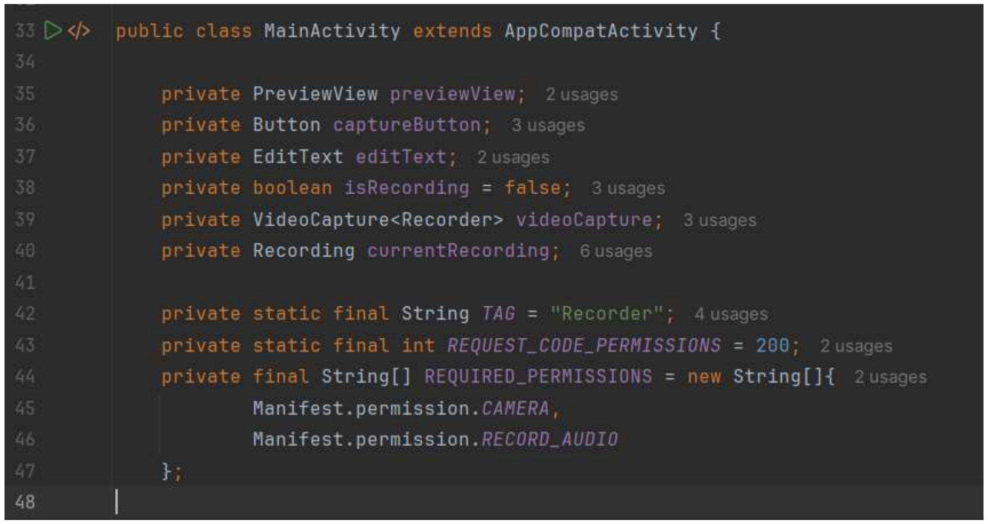
```
    private void setCaptureButtonText(String title) {  2 usages
        captureButton.setText(title);
    }

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

Overview of MainActivity



```java
package yiliu.lab5b_yliu327.media;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.camera.core.CameraSelector;
import androidx.camera.core.Preview;
import androidx.camera.lifecycle.ProcessCameraProvider;
import androidx.camera.video.MediaStoreOutputOptions;
import androidx.camera.video.Recorder;
import androidx.camera.video.Recording;
import androidx.camera.video.VideoCapture;
import androidx.camera.video.VideoRecordEvent;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.ContentValues;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import androidx.camera.view.PreviewView;
import androidx.lifecycle.LifecycleOwner;
import com.google.common.util.concurrent.ListenableFuture;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.concurrent.ExecutionException;
```

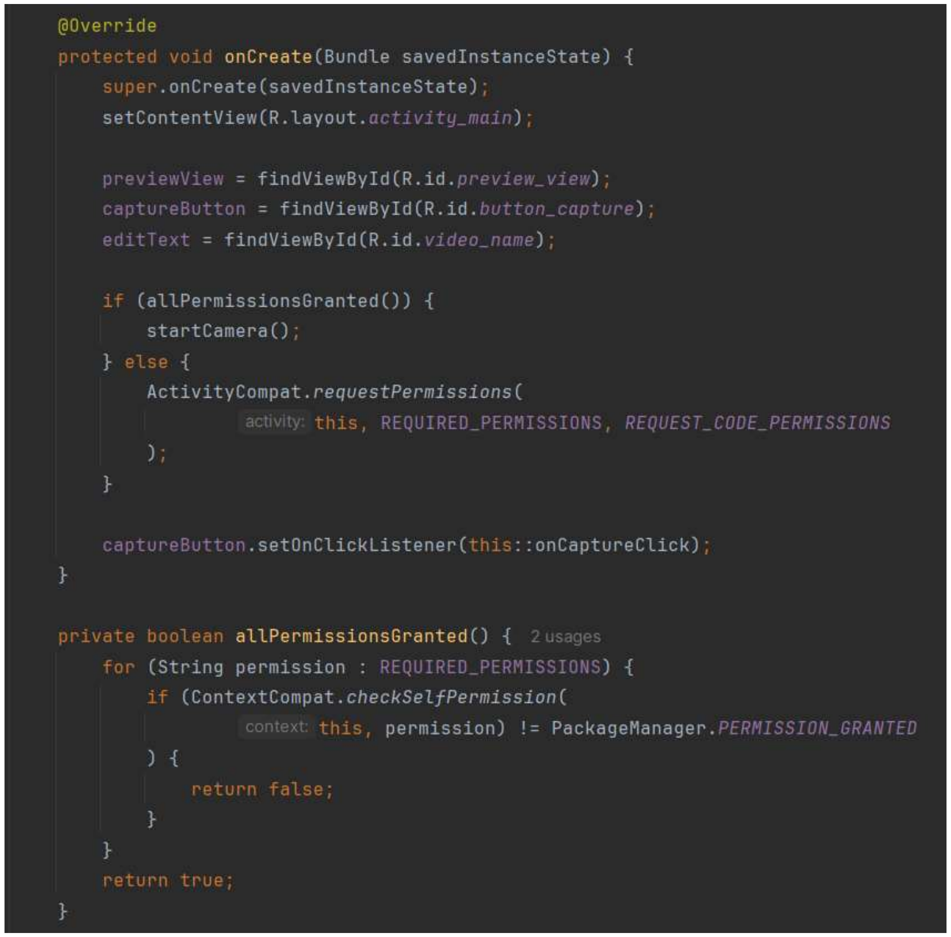
Import Section



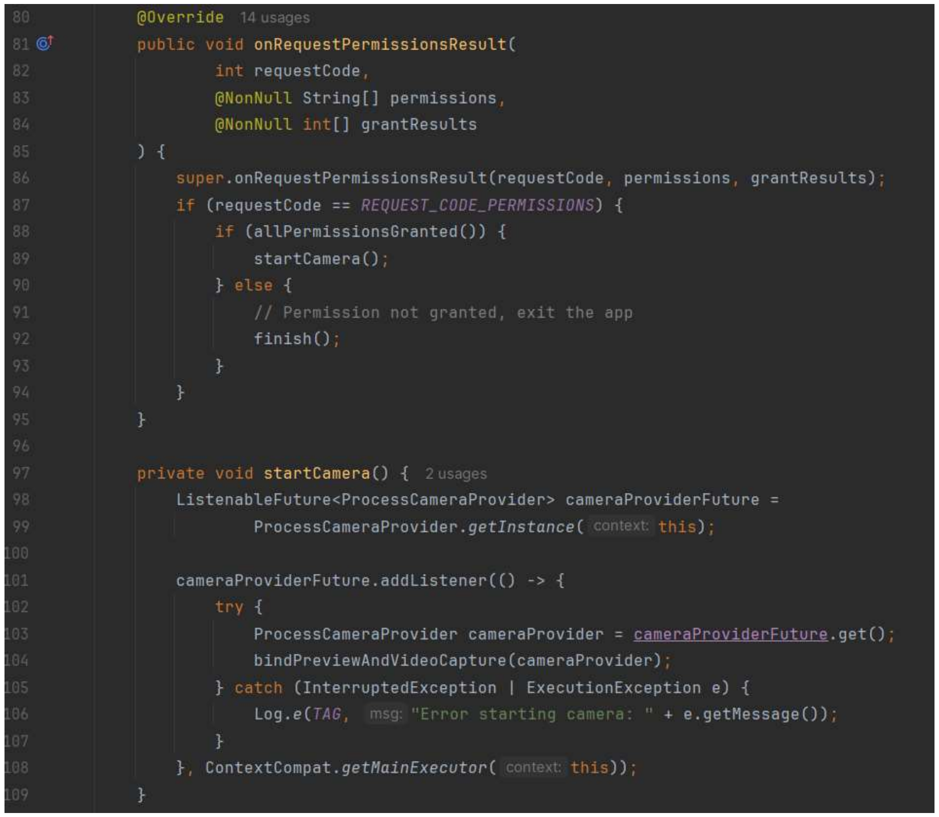
```java
public class MainActivity extends AppCompatActivity {

    private PreviewView previewView;  2 usages
    private Button captureButton;  3 usages
    private EditText editText;  2 usages
    private boolean isRecording = false;  3 usages
    private VideoCapture<Recorder> videoCapture;  3 usages
    private Recording currentRecording;  6 usages

    private static final String TAG = "Recorder";  4 usages
    private static final int REQUEST_CODE_PERMISSIONS = 200;  2 usages
    private final String[] REQUIRED_PERMISSIONS = new String[]{  2 usages
            Manifest.permission.CAMERA,
            Manifest.permission.RECORD_AUDIO
    };

```

Declare your variables

MainActivity (Continued)

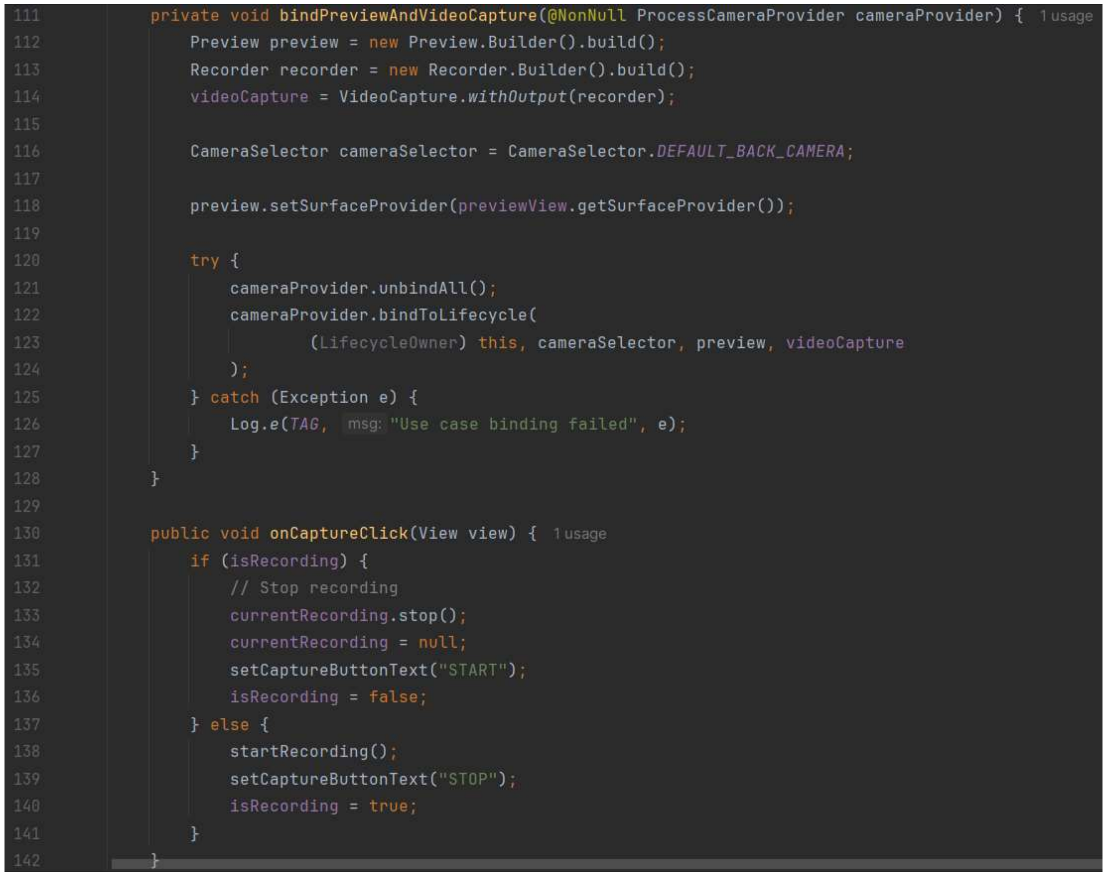


```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    previewView = findViewById(R.id.preview_view);
    captureButton = findViewById(R.id.button_capture);
    editText = findViewById(R.id.video_name);

    if (allPermissionsGranted()) {
        startCamera();
    } else {
        ActivityCompat.requestPermissions(
                activity: this, REQUIRED_PERMISSIONS, REQUEST_CODE_PERMISSIONS
        );
    }

    captureButton.setOnClickListener(this::onCaptureClick);
}

private boolean allPermissionsGranted() {  2 usages
    for (String permission : REQUIRED_PERMISSIONS) {
        if (ContextCompat.checkSelfPermission(
                context: this, permission) != PackageManager.PERMISSION_GRANTED
        ) {
            return false;
        }
    }
    return true;
}
```



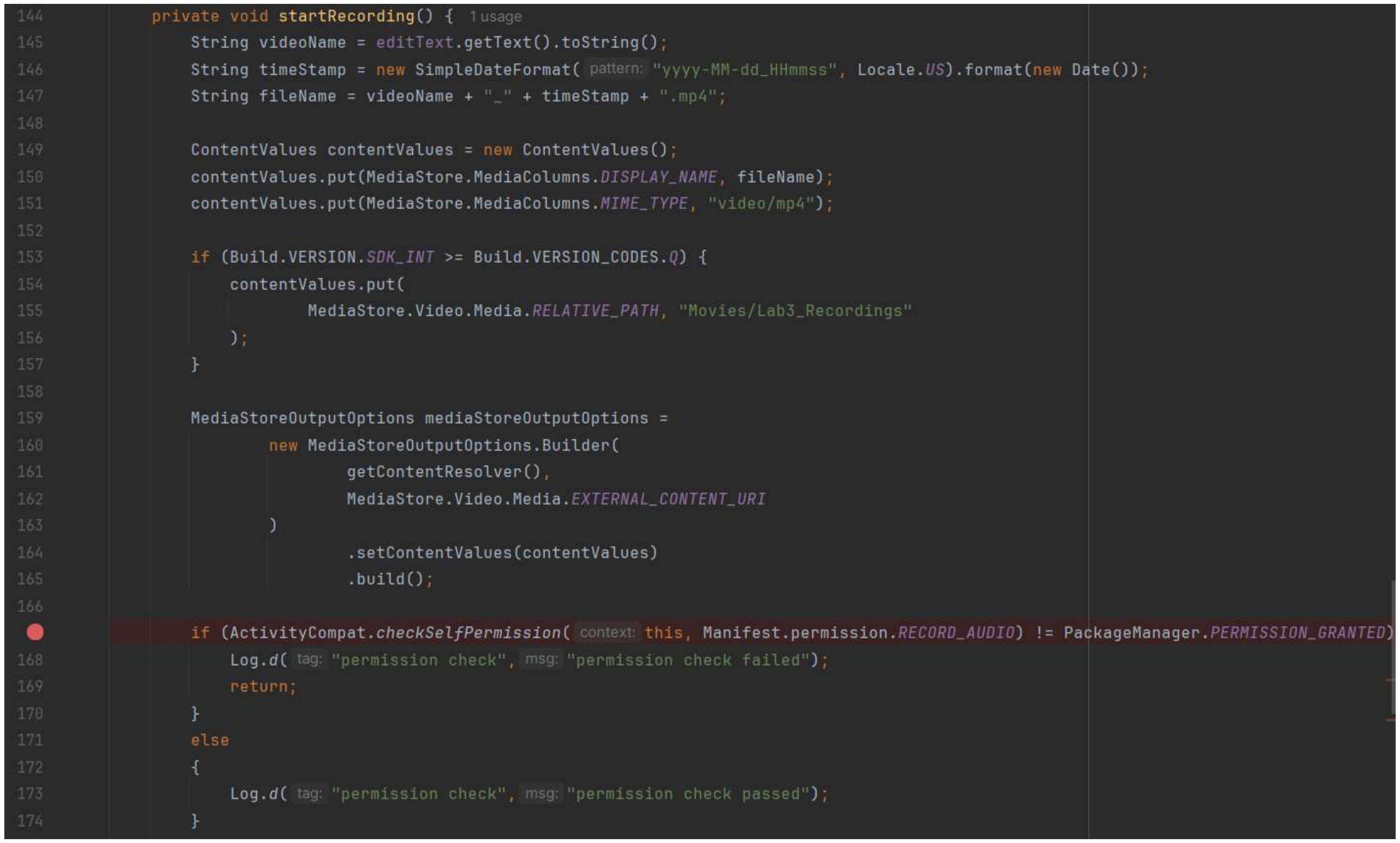
```java
@Override  14 usages
public void onRequestPermissionsResult(
        int requestCode,
        @NonNull String[] permissions,
        @NonNull int[] grantResults
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CODE_PERMISSIONS) {
        if (allPermissionsGranted()) {
            startCamera();
        } else {
            // Permission not granted, exit the app
            finish();
        }
    }
}

private void startCamera() {  2 usages
    ListenableFuture<ProcessCameraProvider> cameraProviderFuture =
            ProcessCameraProvider.getInstance( context: this);

    cameraProviderFuture.addListener(() -> {
        try {
            ProcessCameraProvider cameraProvider = cameraProviderFuture.get();
            bindPreviewAndVideoCapture(cameraProvider);
        } catch (InterruptedException | ExecutionException e) {
            Log.e(TAG,  msg: "Error starting camera: " + e.getMessage());
        }
    }, ContextCompat.getMainExecutor( context: this));
}
```

MainActivity (Continued)



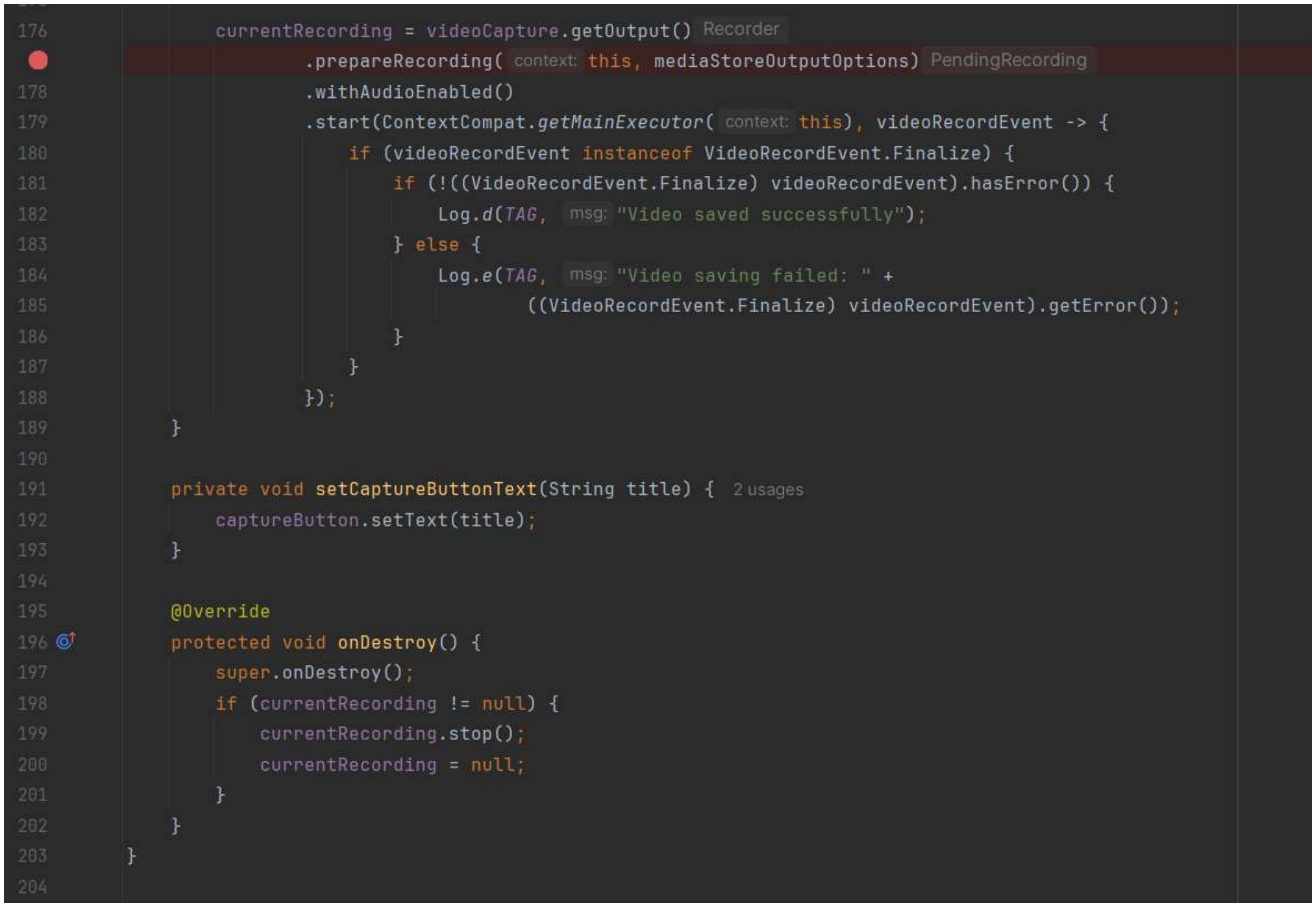
```java
    private void bindPreviewAndVideoCapture(@NonNull ProcessCameraProvider cameraProvider) {  1 usage
        Preview preview = new Preview.Builder().build();
        Recorder recorder = new Recorder.Builder().build();
        videoCapture = VideoCapture.withOutput(recorder);

        CameraSelector cameraSelector = CameraSelector.DEFAULT_BACK_CAMERA;

        preview.setSurfaceProvider(previewView.getSurfaceProvider());

        try {
            cameraProvider.unbindAll();
            cameraProvider.bindToLifecycle(
                    (LifecycleOwner) this, cameraSelector, preview, videoCapture
            );
        } catch (Exception e) {
            Log.e(TAG,  msg: "Use case binding failed", e);
        }
    }

    public void onCaptureClick(View view) {  1 usage
        if (isRecording) {
            // Stop recording
            currentRecording.stop();
            currentRecording = null;
            setCaptureButtonText("START");
            isRecording = false;
        } else {
            startRecording();
            setCaptureButtonText("STOP");
            isRecording = true;
        }
    }
```

MainActivity (Continued)



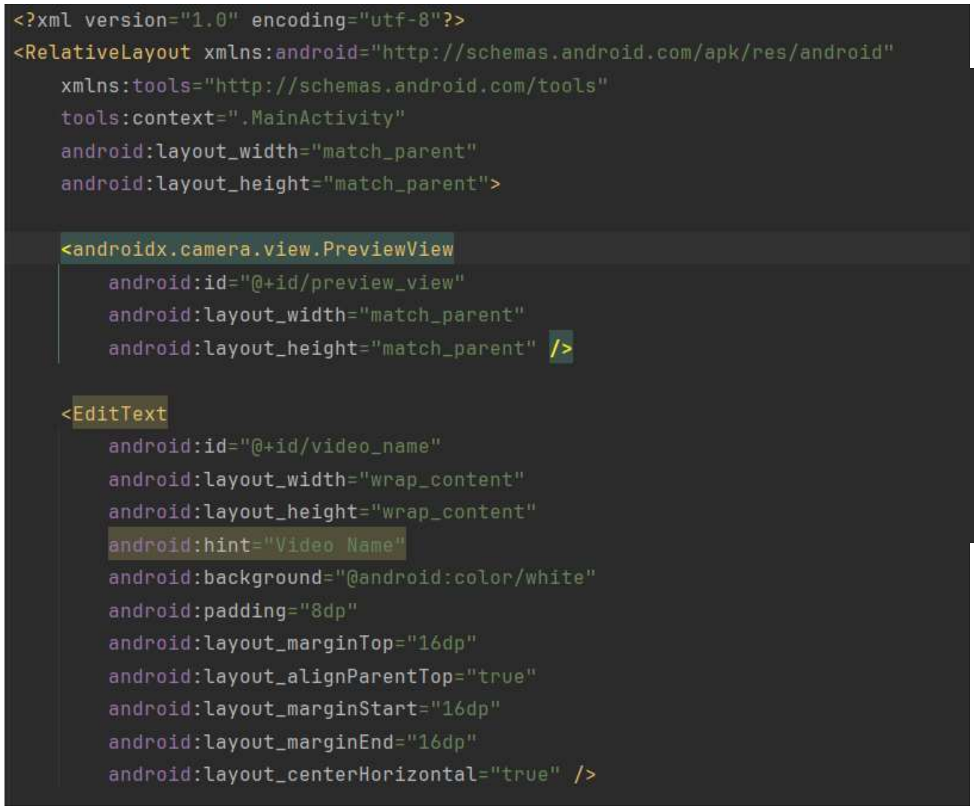
```
    private void startRecording() {  1 usage
        String videoName = editText.getText().toString();
        String timeStamp = new SimpleDateFormat( pattern: "yyyy-MM-dd_HHmmss", Locale.US).format(new Date());
        String fileName = videoName + "_" + timeStamp + ".mp4";

        ContentValues contentValues = new ContentValues();
        contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, fileName);
        contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "video/mp4");

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            contentValues.put(
                    MediaStore.Video.Media.RELATIVE_PATH, "Movies/Lab3_Recordings"
            );
        }

        MediaStoreOutputOptions mediaStoreOutputOptions =
                new MediaStoreOutputOptions.Builder(
                        getContentResolver(),
                        MediaStore.Video.Media.EXTERNAL_CONTENT_URI
                )
                        .setContentValues(contentValues)
                        .build();

        if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.RECORD_AUDIO) != PackageManager.PERMISSION_GRANTED)
            Log.d( tag: "permission check", msg: "permission check failed");
            return;
        }
        else
        {
            Log.d( tag: "permission check", msg: "permission check passed");
        }
```

MainActivity (Continued)

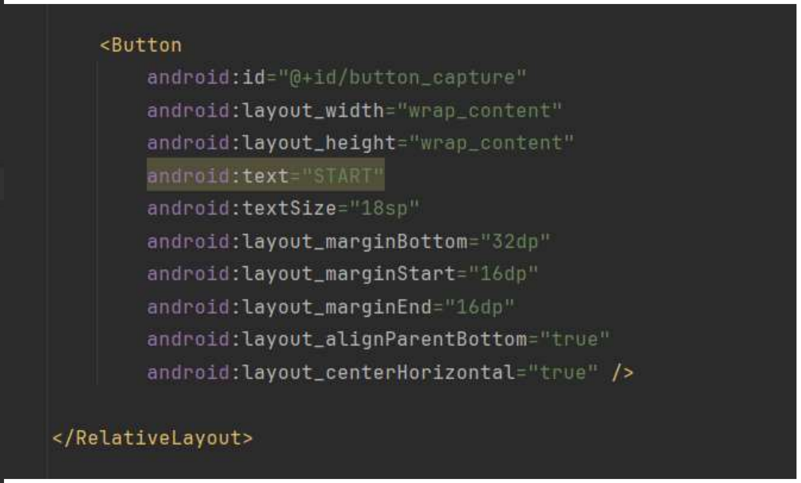


```
        currentRecording = videoCapture.getOutput() Recorder
                .prepareRecording( context: this, mediaStoreOutputOptions) PendingRecording
                .withAudioEnabled()
                .start(ContextCompat.getMainExecutor( context: this), videoRecordEvent -> {
                    if (videoRecordEvent instanceof VideoRecordEvent.Finalize) {
                        if (!((VideoRecordEvent.Finalize) videoRecordEvent).hasError()) {
                            Log.d(TAG,  msg: "Video saved successfully");
                        } else {
                            Log.e(TAG,  msg: "Video saving failed: " +
                                    ((VideoRecordEvent.Finalize) videoRecordEvent).getError());
                        }
                    }
                });
    }

    private void setCaptureButtonText(String title) {  2 usages
        captureButton.setText(title);
    }

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

activity_main.xml

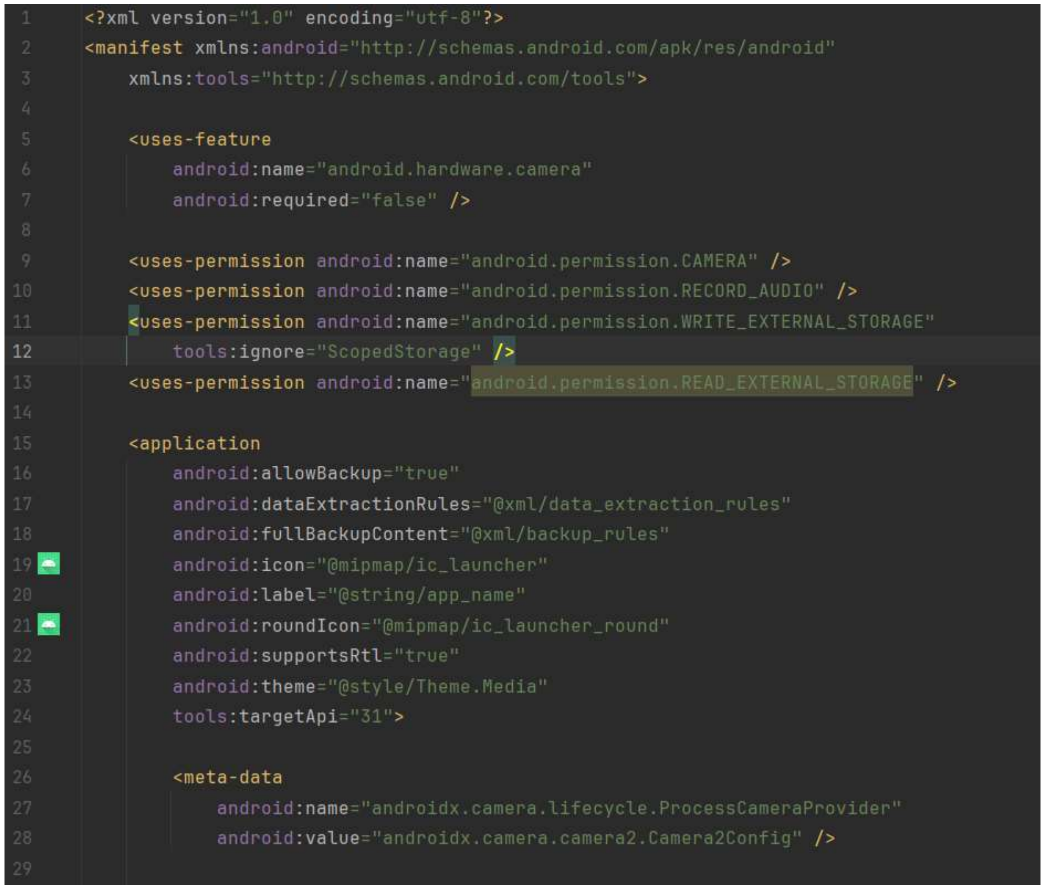


```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.camera.view.PreviewView
        android:id="@+id/preview_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <EditText
        android:id="@+id/video_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Video Name"
        android:background="@android:color/white"
        android:padding="8dp"
        android:layout_marginTop="16dp"
        android:layout_alignParentTop="true"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_centerHorizontal="true" />
```

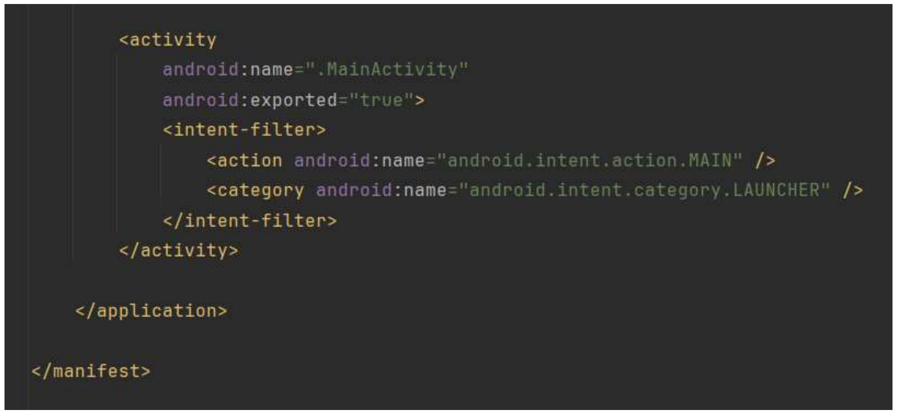


```xml
    <Button
        android:id="@+id/button_capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="START"
        android:textSize="18sp"
        android:layout_marginBottom="32dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Manifest

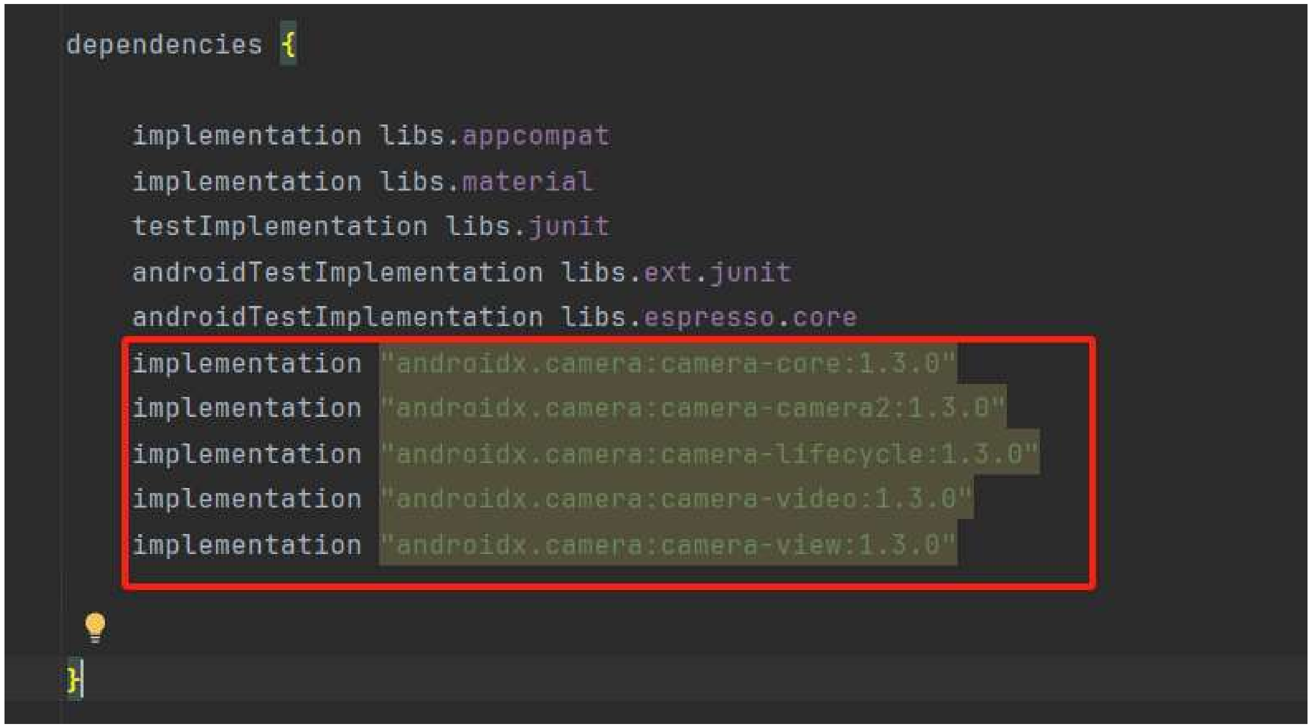


```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        tools:ignore="ScopedStorage" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Media"
        tools:targetApi="31">

        <meta-data
            android:name="androidx.camera.lifecycle.ProcessCameraProvider"
            android:value="androidx.camera.camera2.Camera2Config" />

```



```xml
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Gradle (app level) dependencies



```
dependencies {

    implementation libs.appcompat
    implementation libs.material
    testImplementation libs.junit
    androidTestImplementation libs.ext.junit
    androidTestImplementation libs.espresso.core
    implementation "androidx.camera:camera-core:1.3.0"
    implementation "androidx.camera:camera-camera2:1.3.0"
    implementation "androidx.camera:camera-lifecycle:1.3.0"
    implementation "androidx.camera:camera-video:1.3.0"
    implementation "androidx.camera:camera-view:1.3.0"

}
```

Extra credit

- Add one more button on the UI. Click and play the most recently recorded video in the preview.