CSE 162 Mobile Computing

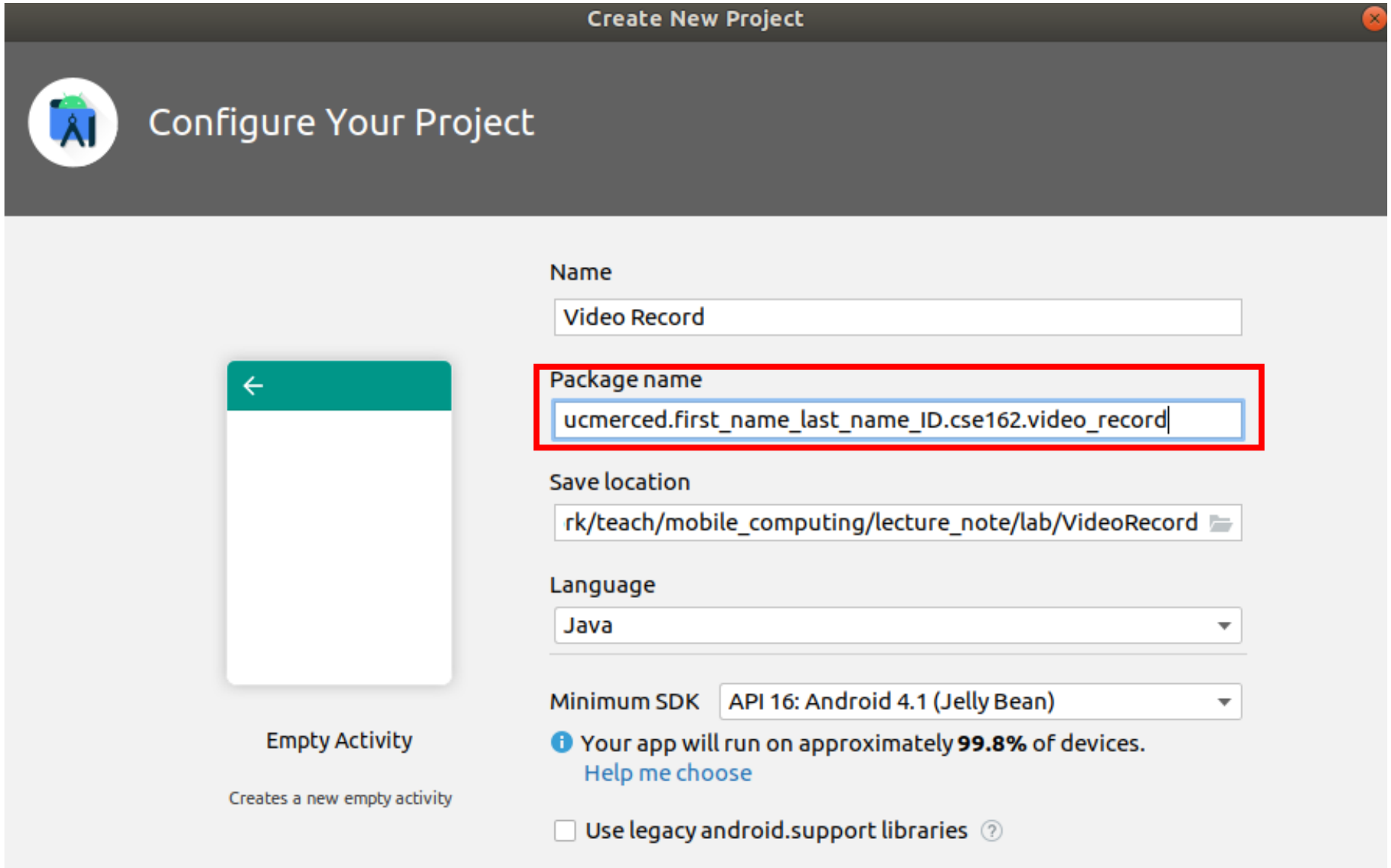
Lab 6 Media Recorder

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

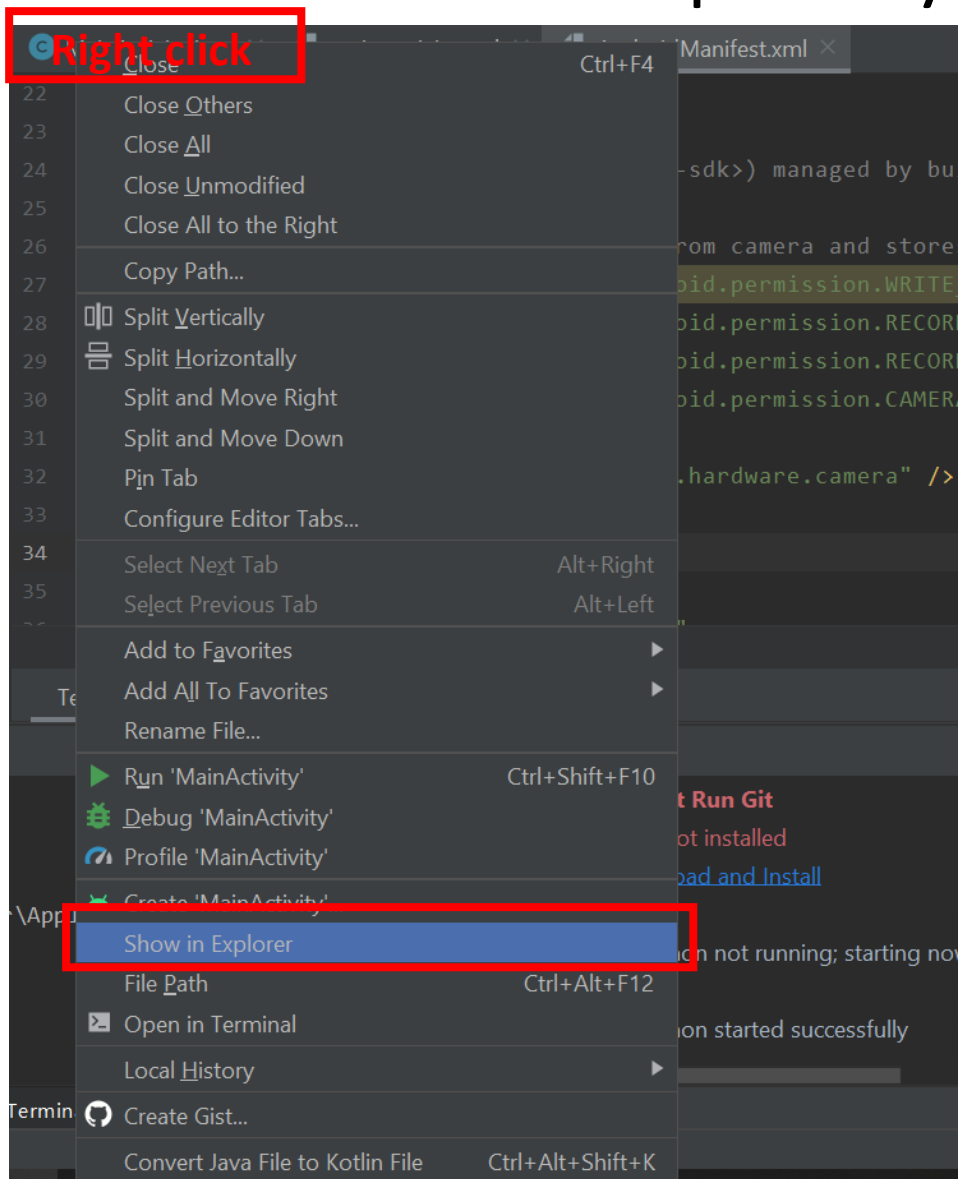
Submission policy

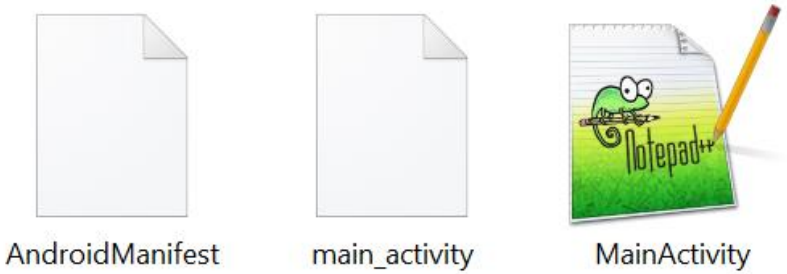
- Put your name in the app package when creating the app
  - ucmerced.first_name_last_name_ID.cse162.video_record
- Submit the code on CatCourse
  - Three separate files
  - AndroidManifest.xml, main_activity.xml, MainActivity.java

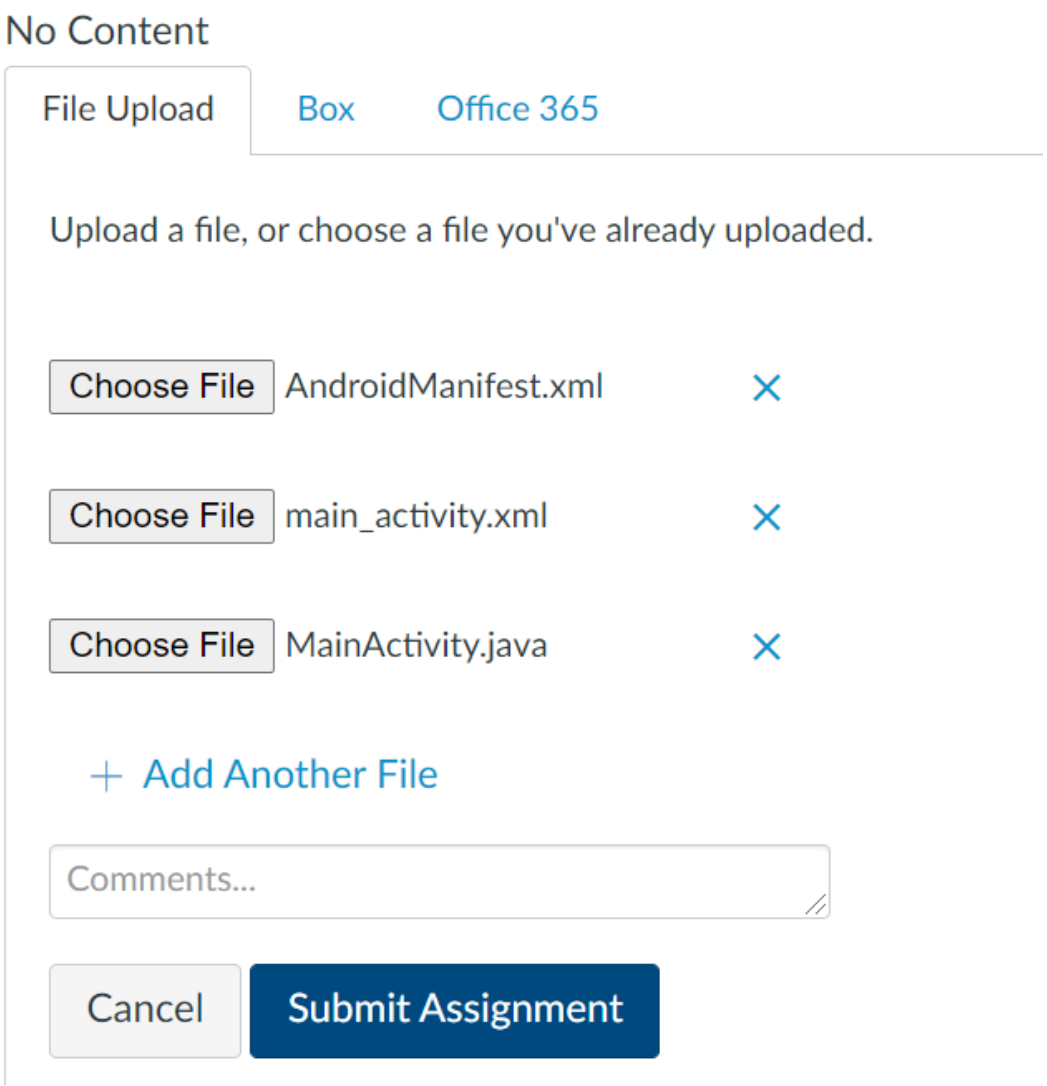
Submission policy



Submission policy





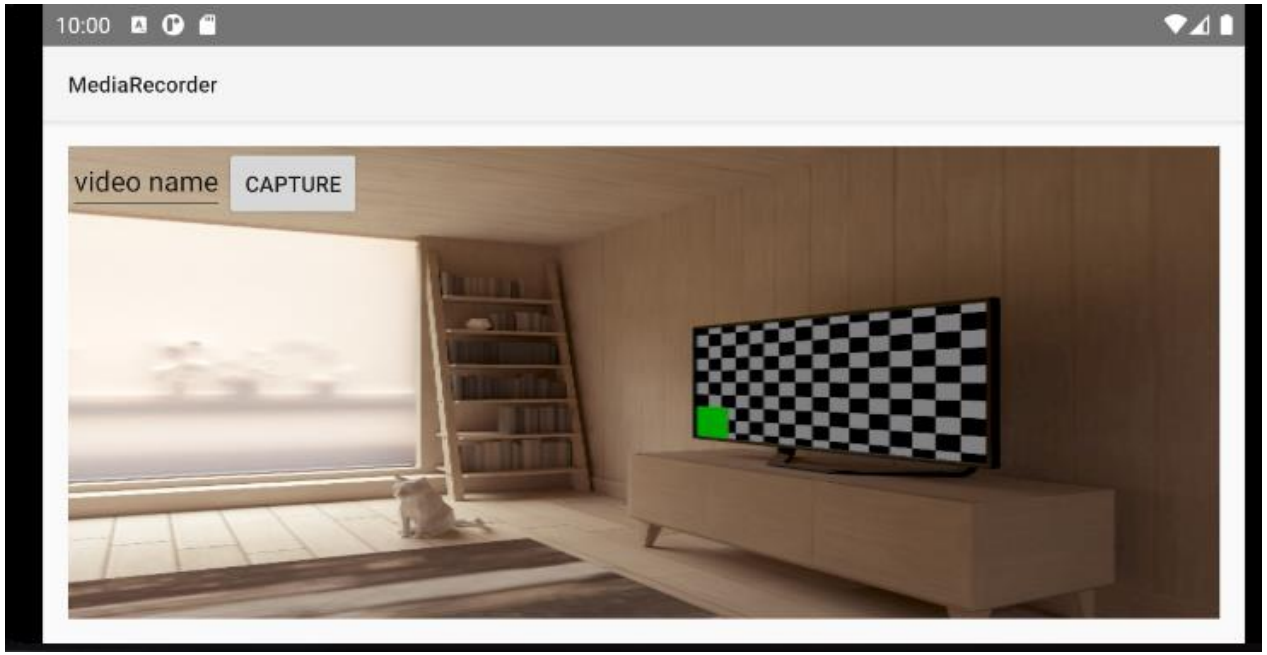


Goal: achieve the following features

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

Outline

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



permission in the manifest file

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CAMERA" />

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

other parts of the manifest

```
<application
    android:allowBackup="true"
    android:fullBackupContent="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.AppCompat.Light"
    tools:ignore="GoogleAppIndexingWarning">

    <activity
        android:name=".MainActivity"
        android:label="@string/app_name"
        android:screenOrientation="landscape">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Prepare the UI

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

```
<FrameLayout
  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="horizontal"
  android:paddingLeft="16dp"
  android:paddingTop="16dp"
  android:paddingRight="16dp"
  android:paddingBottom="16dp"
  tools:context=".MainActivity">
  <TextureView
    android:id="@+id/surface_view"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"  />
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <EditText
      android:id="@+id/video_name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="video_name"/>
    <Button
      android:id="@+id/button_capture"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="bottom"
      android:onClick="onCaptureClick"
      android:text="capture" />
  </LinearLayout>
</FrameLayout>
```

```
private Camera mCamera;
private TextureView mPreview;
private MediaRecorder mMediaRecorder;
private File mOutputFile;

private boolean isRecording = false;
private static final String TAG = "Recorder";
private Button captureButton;

private EditText editText;
```

<u>Make sure you import the hardware camera (not graphics)</u>

```
import android.hardware.Camera;
```

prepare the UI

- Oncreate()

Obtain the views

```
mPreview = findViewById(R.id.surface_view);
captureButton = findViewById(R.id.button_capture);
editText = findViewById(R.id.video_name);
```

- obtain the permissions

```
String[] perms = {"android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.VIBRATE",
"android.permission.RECORD_AUDIO", "android.permission.BODY_SENSORS","android.permission.CAMERA"};

    int permsRequestCode = 200;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        requestPermissions(perms, permsRequestCode);
    }
```

prepare for video recording prepareVideoRecorder()

- Set the sizes of the video frame

```
private boolean prepareVideoRecorder() {
    // BEGIN_INCLUDE (configure_preview)
    mCamera= Camera.open();
    Camera.Parameters parameters = mCamera.getParameters();
    // Use the same size for recording profile.
    CamcorderProfile profile = CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH);

    List<Camera.Size> mSupportedPreviewSizes = parameters.getSupportedPreviewSizes();
    profile.videoFrameWidth = mSupportedPreviewSizes.get(0).width;
    profile.videoFrameHeight = mSupportedPreviewSizes.get(0).height;
    // likewise for the camera object itself.
    parameters.setPreviewSize(profile.videoFrameWidth, profile.videoFrameHeight);
    mCamera.setParameters(parameters);
    try {
      mCamera.setPreviewTexture(mPreview.getSurfaceTexture());
    } catch (IOException e) {
      Log.e(TAG, "Surface texture is unavailable or unsuitable" + e.getMessage());
      return false;
    }
```

prepareVideoRecorder()

- configure the MediaRecorder

```
mMediaRecorder = new MediaRecorder();
    // Step 1: Unlock and set camera to MediaRecorder
    mCamera.unlock();
    mMediaRecorder.setCamera(mCamera);
    // Step 2: Set sources
    mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
    mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    // Step 3: Set a CamcorderProfile (requires API Level 8 or higher)
    mMediaRecorder.setProfile(profile);
    // Step 4: Set output file
    mOutputFile = getOutputMediaFile();
    if (mOutputFile == null) {
        return false;
    }
    mMediaRecorder.setOutputFile(mOutputFile.getPath());
    // END_INCLUDE (configure_media_recorder)
    // Step 5: Prepare configured MediaRecorder
    try {
        mMediaRecorder.prepare();
    } catch (IllegalStateException e) {
        Log.d(TAG, "IllegalStateException preparing MediaRecorder: " + e.getMessage());
        releaseMediaRecorder();
        return false;
    } catch (IOException e) {
        Log.d(TAG, "IOException preparing MediaRecorder: " + e.getMessage());
        releaseMediaRecorder();
        return false;
    }
    return true;
}
```

Prepare for the file storage File getOutputMediaFile()

- permission, get the path, etc

```
public File getOutputMediaFile(){
    // To be safe, you should check that the SDCard is mounted
    // using Environment.getExternalStorageState() before doing this.
    if (!Environment.getExternalStorageState().equalsIgnoreCase(Environment.MEDIA_MOUNTED)) {
      return  null;
    }

    File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
          Environment.DIRECTORY_PICTURES), "CameraSample");
    // This location works best if you want the created images to be shared
    // between applications and persist after your app has been uninstalled.

    // Create the storage directory if it does not exist
    if (! mediaStorageDir.exists()){
      if (! mediaStorageDir.mkdirs()) {
          Log.d("CameraSample", "failed to create directory");
          return null;
      }
    }
```

create the media file

```
// Create a media file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.US).format(new Date());
    File mediaFile;

    Editable video_name=editText.getText();

    mediaFile = new File(mediaStorageDir.getPath() + File.separator +
        video_name.toString()+"_"+ timeStamp + ".mp4");


    return mediaFile;
```

Use the media recorder

```
public void onCaptureClick(View view) {
    if (isRecording) {
        // BEGIN_INCLUDE(stop_release_media_recorder)
        // stop recording and release camera
        try {
            mMediaRecorder.stop();  // stop the recording
        } catch (RuntimeException e) {
            // RuntimeException is thrown when stop() is called immediately after start().
            // In this case the output file is not properly constructed ans should be deleted.
            Log.d(TAG, "RuntimeException: stop() is called immediately after start()");
            //noinspection ResultOfMethodCallIgnored
            mOutputFile.delete();
        }
        releaseMediaRecorder(); // release the MediaRecorder object
        mCamera.lock();         // take camera access back from MediaRecorder
        // inform the user that recording has stopped
        captureButton.setText("Capture");
        isRecording = false;
        releaseCamera();
    } else {

        if (prepareVideoRecorder()) {
            // Camera is available and unlocked, MediaRecorder is prepared,
            // now you can start recording
            mMediaRecorder.start();
            isRecording = true;
        } else {
            // prepare didn't work, release the camera
            releaseMediaRecorder();
        }
        // END_INCLUDE(prepare_start_media_recorder)
    }
}
```

When the recording stops, release the camera and the mediarecorder

```
private void releaseMediaRecorder() {
    if (mMediaRecorder != null) {
        // clear recorder configuration
        mMediaRecorder.reset();
        // release the recorder object
        mMediaRecorder.release();
        mMediaRecorder = null;
        // Lock camera for later use i.e taking it back from MediaRecorder.
        // MediaRecorder doesn't need it anymore and we will release it if the activity pauses.
        mCamera.lock();
    }
}

private void releaseCamera() {
    if (mCamera != null) {
        // release the camera for other applications
        mCamera.release();
        mCamera = null;
    }
}
```

```
@Override
  protected void onPause() {
      super.onPause();
      // if we are using MediaRecorder, release it first
      releaseMediaRecorder();
      // release the camera immediately on pause event
      releaseCamera();
  }
```

Extra credit

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