File - MainActivity.java

```
 1 package com.example.alarm.alarmapp;
 2
 3 import android.Manifest;
 4 import android.content.Intent;
 5 import android.content.pm.PackageManager;
 6 import android.media.MediaPlayer;
 7 import android.os.Bundle;
 8 import android.os.Handler;
 9 import android.os.Message;
10 import android.support.annotation.NonNull;
11 import android.support.v4.app.ActivityCompat;
12 import android.support.v4.content.ContextCompat;
13 import android.support.v7.app.AppCompatActivity;
14 import android.util.Log;
15 import android.view.SurfaceView;
16 import android.view.View;
17 import android.view.WindowManager;
18 import android.widget.Button;
19 import android.widget.Switch;
20 import android.widget.TextView;
21 import android.widget.ToggleButton;
22
23 import com.example.alarm.alarmapp.views.AlarmCameraView;
24
25 import org.opencv.android.BaseLoaderCallback;
26 import org.opencv.android.LoaderCallbackInterface;
27 import org.opencv.android.OpenCVLoader;
28
29 /**
30  * This is the MainActivity. It handles all the ui related stuff like
   buttons and controls the AlarmCameraView.
31  */
32 public class MainActivity extends AppCompatActivity implements Handler
   .Callback, View.OnClickListener, AlarmCameraView.IAlarmCameraListener
   {
33     private static final String TAG = "MainActivity";
34
35     private static final int TIMEOUT_START = 10000;
36
37     private AlarmCameraView mCameraView;
38     private ToggleButton mTbtnStartStop;
39     private Switch mSwSound;
40     private TextView mTvAlarmTriggered, mTvState;
41     private boolean mHasPermission = false;
42
43     private Runnable mRunnableStartTimer = () -> onCommand(Command.
   START_TIMER);
44
45     private MediaPlayer mAlarmPlayer;
46     private Handler mUiHandler;
47
48     private State mState = State.IDLE;
49
50     @Override
```

File - MainActivity.java

```
51      public void onAlarm() {
52          Log.d(TAG, "onAlarm()");
53          onCommand(Command.ALARM);
54      }
55
56      @Override
57      public void onCalibrating() {
58          Log.d(TAG, "onCalibrating()");
59          onCommand(Command.CALIBRATING);
60      }
61
62      @Override
63      public void onRun() {
64          Log.d(TAG, "onRun()");
65          onCommand(Command.RUNNING);
66      }
67
68      @Override
69      public Handler getHandler() {
70          return mUiHandler;
71      }
72
73      private enum State {
74          IDLE, WAITING_TO_START, CALIBRATING, RUNNING
75      }
76
77      private enum Command {
78          BUTTON_START_STOP, START_TIMER, ALARM, CALIBRATING, RUNNING,
    ON_PAUSE, RESET_ALARM_TEXT
79      }
80
81      @Override
82      protected void onCreate(Bundle savedInstanceState) {
83          super.onCreate(savedInstanceState);
84
85          //check for camera permission, if it is not granted invoke a
    layout that displays an error
86          mHasPermission = ContextCompat.checkSelfPermission(this,
    Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED;
87          if (!mHasPermission) {
88              setContentView(R.layout.activity_main_permission_missing)
    ;
89              ActivityCompat.requestPermissions(this, new String[] {
    Manifest.permission.CAMERA}, 0);
90              return; //cancel remaining init. To start the real app we
     need permission from the start.
91          }
92
93          setContentView(R.layout.activity_main);
94
95          getWindow().addFlags(WindowManager.LayoutParams.
    FLAG_KEEP_SCREEN_ON);
96
97          mUiHandler = new Handler(this);
```

```
 98          mAlarmPlayer = MediaPlayer.create(this, R.raw.sound_alarm);
 99
100          mCameraView = findViewById(R.id.cameraView);
101          mTvAlarmTriggered = findViewById(R.id.tvAlarmTriggered);
102          mTvState = findViewById(R.id.tvState);
103          mSwSound = findViewById(R.id.swAlarmSound);
104          mTbtnStartStop = findViewById(R.id.tbtnStartStop);
105
106          mTbtnStartStop.setOnClickListener(this);
107
108          mCameraView.setVisibility(SurfaceView.VISIBLE);
109          mCameraView.setAlarmListener(this);
110      }
111
112      @Override
113      public void onRequestPermissionsResult(int requestCode, @NonNull
     String[] permissions, @NonNull int[] grantResults) {
114          switch (requestCode) {
115              case 0:
116                  //restart app if we got the permission
117                  if (grantResults[0] == PackageManager.
     PERMISSION_GRANTED) {
118                      Intent newActivity = new Intent(this,
     MainActivity.class);
119                      startActivity(newActivity);
120                      finish();
121                  }
122                  break;
123          }
124      }
125
126      private void onCommand(Command cmd) {
127          Log.d(TAG, "Command: " + cmd.toString() + " received in state
     : " + mState.toString());
128          if (cmd == Command.RESET_ALARM_TEXT) mTvAlarmTriggered.
     setText("false");
129          switch (mState) {
130              case IDLE:
131                  switch (cmd) {
132                      case BUTTON_START_STOP:
133                          mState = State.WAITING_TO_START;
134                          mUiHandler.postDelayed(mRunnableStartTimer,
     TIMEOUT_START);
135                          break;
136                      default:
137                          Log.e(TAG, "invalid command in " + mState +
     ": " + cmd);
138                  }
139                  break;
140              case WAITING_TO_START:
141                  switch (cmd) {
142                      case START_TIMER:
143                          mCameraView.startAlarm();
144                          break;
```

```
145                     case CALIBRATING:
146                         mState = State.CALIBRATING;
147                         break;
148                     case BUTTON_START_STOP:
149                         mState = State.IDLE;
150                         mUiHandler.removeCallbacks(
    mRunnableStartTimer);
151                         break;
152                     case ON_PAUSE:
153                         mState = State.IDLE;
154                         mUiHandler.removeCallbacks(
    mRunnableStartTimer);
155                         break;
156                     default:
157                     Log.e(TAG, "invalid command in " + mState + ": "
    + cmd);
158                 }
159                 break;
160             case CALIBRATING:
161                 switch (cmd) {
162                     case BUTTON_START_STOP:
163                     case ON_PAUSE:
164                         mState = State.IDLE;
165                         mCameraView.stopAlarm();
166                         break;
167                     case RUNNING:
168                         mState = State.RUNNING;
169                         break;
170                     default:
171                         Log.e(TAG, "invalid command in " + mState +
    ": " + cmd);
172                 }
173                 break;
174             case RUNNING:
175                 switch (cmd) {
176                     case BUTTON_START_STOP:
177                     case ON_PAUSE:
178                         mState = State.IDLE;
179                         mCameraView.stopAlarm();
180                         break;
181                     case ALARM:
182                         if(mSwSound.isChecked()) playAlarmSound();
183                         mTvAlarmTriggered.setText("ALAAAAAARM!");
184                         mUiHandler.postDelayed(() -> onCommand(
    Command.RESET_ALARM_TEXT), 2000);
185                         break;
186                     default:
187                         Log.e(TAG, "invalid command in " + mState +
    ": " + cmd);
188                 }
189                 break;
190             default:
191                 Log.e(TAG, "State " + mState + " not implemented.");
192         }
```

```
193         Log.d(TAG, "State after command execution: " + mState.
    toString());
194         mTvState.setText(String.format(getString(R.string.state_val),
     mState.toString()));
195         //set start stop button to correct rendering for the current
    state
196         mTbtnStartStop.setChecked(mState == State.RUNNING || mState
    == State.CALIBRATING || mState == State.WAITING_TO_START);
197     }
198
199     //region lifecycle
200
201     @Override
202     protected void onResume() {
203         super.onResume();
204         if (mHasPermission) {
205             if (!OpenCVLoader.initDebug()) {
206                 Log.d(TAG, "Internal OpenCV library not found. Using
    OpenCV Manager for initialization");
207                 OpenCVLoader.initAsync(OpenCVLoader.
    OPENCV_VERSION_2_4_2, this, mLoaderCallback);
208             } else {
209                 Log.d(TAG, "OpenCV library found inside package.
    Using it!");
210                 mLoaderCallback.onManagerConnected(
    LoaderCallbackInterface.SUCCESS);
211             }
212         }
213     }
214
215     @Override
216     protected void onStart() {
217         super.onStart();
218     }
219
220     @Override
221     protected void onPause() {
222         super.onPause();
223         if (mHasPermission) {
224             if (mCameraView != null) {
225                 mCameraView.disableView();
226                 onCommand(Command.ON_PAUSE);
227             }
228         }
229     }
230
231     //endregion
232
233     private void playAlarmSound() {
234         if(!mAlarmPlayer.isPlaying()){
235             mAlarmPlayer.seekTo(0);
236             mAlarmPlayer.start();
237         }
238     }
```

```
239
240     private BaseLoaderCallback mLoaderCallback = new
    BaseLoaderCallback(this) {
241         @Override
242         public void onManagerConnected(int status) {
243             switch (status) {
244                 case LoaderCallbackInterface.SUCCESS:
245                 {
246                     Log.i(TAG, "OpenCV loaded successfully");
247                     mCameraView.enableView();
248
249                 } break;
250                 default:
251                 {
252                     super.onManagerConnected(status);
253                 } break;
254             }
255         }
256     };
257
258     @Override
259     public boolean handleMessage(Message msg) {
260         return false;
261     }
262
263     @Override
264     public void onClick(View v) {
265         switch (v.getId()) {
266             case R.id.tbtnStartStop:
267                 onCommand(Command.BUTTON_START_STOP);
268                 break;
269         }
270     }
271 }
272
```