LAMPIRAN

a. MainActivity.java

```
package com.example.texttospeech;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    public static final Integer RecordAudioRequestCode = 1;
    private SpeechRecognizer speechRecognizer;
    private EditText editText;
    private ImageButton micButton;

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

if(ContextCompat.checkSelfPermission(this,Manifest.permission.RECORD_A
UDIO) != PackageManager.PERMISSION_GRANTED){
            checkPermission();
        }

        editText = findViewById(R.id.text);
        micButton = findViewById(R.id.button);
```

```
        speechRecognizer =
SpeechRecognizer.createSpeechRecognizer(this);

        final Intent speechRecognizerIntent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
Locale.getDefault());

        speechRecognizer.setRecognitionListener(new
RecognitionListener() {
            @Override
            public void onReadyForSpeech(Bundle bundle) {

            }

            @Override
            public void onBeginningOfSpeech() {
                editText.setText("");
                editText.setHint("Listening...");
            }

            @Override
            public void onRmsChanged(float v) {

            }

            @Override
            public void onBufferReceived(byte[] bytes) {

            }

            @Override
            public void onEndOfSpeech() {

            }

            @Override
            public void onError(int i) {

            }

            @Override
            public void onResults(Bundle bundle) {

micButton.setImageResource(R.drawable.ic_mic_black_off);
                ArrayList<String> data =
bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
```

```
                    editText.setText(data.get(0));
                }

                @Override
                public void onPartialResults(Bundle bundle) {

                }

                @Override
                public void onEvent(int i, Bundle bundle) {

                }
            });

            micButton.setOnTouchListener(new View.OnTouchListener() {
                @Override
                public boolean onTouch(View view, MotionEvent motionEvent)
{
                    if (motionEvent.getAction() == MotionEvent.ACTION_UP){
                        speechRecognizer.stopListening();
                    }
                    if (motionEvent.getAction() ==
MotionEvent.ACTION_DOWN){

micButton.setImageResource(R.drawable.ic_mic_black_24dp);

speechRecognizer.startListening(speechRecognizerIntent);
                    }
                    return false;
                }
            });

    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        speechRecognizer.destroy();
    }

    private void checkPermission() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            ActivityCompat.requestPermissions(this,new
String[]{Manifest.permission.RECORD_AUDIO},RecordAudioRequestCode);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
```

```
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == RecordAudioRequestCode &&
grantResults.length > 0 ){
            if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
                Toast.makeText(this,"Permission
Granted",Toast.LENGTH_SHORT).show();
        }
    }
}

package com.example.texttospeech

import android.Manifest
import android.content.pm.PackageManager
import android.os.Build
import android.os.Bundle
import android.speech.SpeechRecognizer
import android.util.Log
import android.view.View
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.github.stephenvinouze.core.interfaces.RecognitionCallback
import
com.github.stephenvinouze.core.managers.KontinuousRecognitionManager
import com.github.stephenvinouze.core.models.RecognitionStatus
import kotlinx.android.synthetic.main.activity_main2.*


class MainActivity2 : AppCompatActivity(),RecognitionCallback  {
    companion object {
        /**
         * Put any keyword that will trigger the speech recognition
         */
        private const val ACTIVATION_KEYWORD = "halo semua"
        private const val RECORD_AUDIO_REQUEST_CODE = 101

    }

    private val recognitionManager: KontinuousRecognitionManager by
lazy {
        KontinuousRecognitionManager(this, activationKeyword =
ACTIVATION_KEYWORD, callback = this)
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)
```

```
        progressBar.visibility = View.INVISIBLE
        progressBar.max = 10

        recognitionManager.createRecognizer()

        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.RECORD_AUDIO) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.RECORD_AUDIO), RECORD_AUDIO_REQUEST_CODE)
        }
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.RECORD_AUDIO),
MainActivity2.RECORD_AUDIO_REQUEST_CODE)
        }
    }

    override fun onDestroy() {
        recognitionManager.destroyRecognizer()
        super.onDestroy()
    }

    override fun onResume() {
        super.onResume()

        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.RECORD_AUDIO) ==
PackageManager.PERMISSION_GRANTED) {
            startRecognition()
        }
    }

    override fun onPause() {
        stopRecognition()
        super.onPause()
    }

    private fun startRecognition() {
        progressBar.isIndeterminate = false
        progressBar.visibility = View.VISIBLE
        recognitionManager.startRecognition()
    }


    private fun stopRecognition() {
        progressBar.isIndeterminate = true
        progressBar.visibility = View.INVISIBLE
        recognitionManager.stopRecognition()
```

```
    }

    private fun getErrorText(errorCode: Int): String = when
(errorCode) {
        SpeechRecognizer.ERROR_AUDIO -> "Audio recording error"
        SpeechRecognizer.ERROR_CLIENT -> "Client side error"
        SpeechRecognizer.ERROR_INSUFFICIENT_PERMISSIONS ->
"Insufficient permissions"
        SpeechRecognizer.ERROR_NETWORK -> "Network error"
        SpeechRecognizer.ERROR_NETWORK_TIMEOUT -> "Network timeout"
        SpeechRecognizer.ERROR_NO_MATCH -> "No match"
        SpeechRecognizer.ERROR_RECOGNIZER_BUSY -> "RecognitionService
busy"
        SpeechRecognizer.ERROR_SERVER -> "Error from server"
        SpeechRecognizer.ERROR_SPEECH_TIMEOUT -> "No speech input"
        else -> "Didn't understand, please try again."
    }


    override fun onBeginningOfSpeech() {
        Log.i("Recognition", "onBeginningOfSpeech")
    }

    override fun onBufferReceived(buffer: ByteArray) {
        Log.i("Recognition", "onBufferReceived: $buffer")
    }

    override fun onEndOfSpeech() {
        Log.i("Recognition", "onEndOfSpeech")
    }

    override fun onError(errorCode: Int) {
        val errorMessage = getErrorText(errorCode)
        Log.i("Recognition", "onError: $errorMessage")
        textView.text = errorMessage
    }

    override fun onEvent(eventType: Int, params: Bundle) {
        Log.i("Recognition", "onEvent")
    }

    override fun onKeywordDetected() {
        Log.i("Recognition", "keyword detected !!!")
        textView.text = "Keyword detected"
    }

    override fun onPartialResults(results: List<String>) {
        TODO("Not yet implemented")
    }
```

```
    override fun onPrepared(status: RecognitionStatus) {
       when (status) {
           RecognitionStatus.SUCCESS -> {
               Log.i("Recognition", "onPrepared: Success")
               textView.text = "Recognition ready"
           }
           RecognitionStatus.UNAVAILABLE -> {
               Log.i("Recognition", "onPrepared: Failure or
unavailable")
               AlertDialog.Builder(this)
                       .setTitle("Speech Recognizer unavailable")
                       .setMessage("Your device does not support
Speech Recognition. Sorry!")
                       .setPositiveButton(android.R.string.ok, null)
                       .show()
           }
        }
    }

    override fun onReadyForSpeech(params: Bundle) {
        Log.i("Recognition", "onReadyForSpeech")
    }

    override fun onResults(results: List<String>, scores: FloatArray?)
{
        val text = results.joinToString(separator = "\n")
        Log.i("Recognition", "onResults : $text")
        textView.text = text
    }

    override fun onRmsChanged(rmsdB: Float) {
        progressBar.progress = rmsdB.toInt()
    }

    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        when (requestCode) {
            RECORD_AUDIO_REQUEST_CODE -> {
                if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    startRecognition()
                }
            }
        }
    }
}
```

b. Activity_main.xml

```
<?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">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_centerInParent="true"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_height="wrap_content">
        <EditText
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginEnd="15dp"
            android:layout_marginStart="15dp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            android:padding="10dp"
            android:hint="Tap to Speak"
            android:id="@+id/text"
            android:background="@drawable/gradient_blue"
            android:layout_centerInParent="true"
            />
       <!-- <ImageView
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:backgroundTint="#F9F8FA"
            android:paddingRight="10dp"
            android:src="@drawable/ic_mic_black_off"
            android:layout_alignParentRight="true"
            android:id="@+id/button"/>-->
        <androidx.appcompat.widget.AppCompatImageButton
            android:id="@+id/button"
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:layout_marginTop="16dp"
            app:layout_constraintStart_toStartOf="@id/text"
            app:layout_constraintEnd_toEndOf="@id/text"
            app:layout_constraintTop_toBottomOf="@id/text"
            android:src="@drawable/ic_mic_black_off"
            android:background="@drawable/circle_button_red"
```

```
            />

    </androidx.constraintlayout.widget.ConstraintLayout>

</RelativeLayout>
```

c. TextToSpeech.app.iml

```
<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id=":app"
external.linked.project.path="$MODULE_DIR$/../../../app"
external.root.project.path="$MODULE_DIR$/../../.."
external.system.id="GRADLE"
external.system.module.group="TextToSpeech"
external.system.module.version="unspecified" type="JAVA_MODULE"
version="4">
  <component name="FacetManager">
    <facet type="android-gradle" name="Android-Gradle">
      <configuration>
        <option name="GRADLE_PROJECT_PATH" value=":app" />
        <option name="LAST_SUCCESSFUL_SYNC_AGP_VERSION" value="4.1.3"
/>
        <option name="LAST_KNOWN_AGP_VERSION" value="4.1.3" />
      </configuration>
    </facet>
    <facet type="android" name="Android">
      <configuration>
        <option name="SELECTED_BUILD_VARIANT" value="debug" />
        <option name="ASSEMBLE_TASK_NAME" value="assembleDebug" />
        <option name="COMPILE_JAVA_TASK_NAME"
value="compileDebugSources" />
        <afterSyncTasks>
          <task>generateDebugSources</task>
        </afterSyncTasks>
        <option name="ALLOW_USER_CONFIGURATION" value="false" />
        <option name="MANIFEST_FILE_RELATIVE_PATH"
value="/src/main/AndroidManifest.xml" />
        <option name="RES_FOLDER_RELATIVE_PATH" value="/src/main/res"
/>
        <option name="RES_FOLDERS_RELATIVE_PATH"
value="file://$MODULE_DIR$/../../../app/src/main/res;file://$MODULE_DI
R$/../../../app/src/debug/res;file://$MODULE_DIR$/../../../app/build/g
enerated/res/rs/debug;file://$MODULE_DIR$/../../../app/build/generated
/res/resValues/debug" />
        <option name="TEST_RES_FOLDERS_RELATIVE_PATH"
value="file://$MODULE_DIR$/../../../app/src/androidTest/res;file://$MO
DULE_DIR$/../../../app/src/androidTestDebug/res;file://$MODULE_DIR$/..
/../../app/build/generated/res/rs/androidTest/debug;file://$MODULE_DIR
$/../../../app/build/generated/res/resValues/androidTest/debug" />
        <option name="ASSETS_FOLDER_RELATIVE_PATH"
value="/src/main/assets" />
      </configuration>
    </facet>
```

```
    <facet type="kotlin-language" name="Kotlin">
      <configuration version="3" platform="JVM 1.6" allPlatforms="JVM
[1.6]" useProjectSettings="false">
        <compilerSettings />
        <compilerArguments>
          <option name="languageVersion" value="1.4" />
          <option name="apiVersion" value="1.4" />
          <option name="pluginOptions">
            <array>
              <option
value="plugin:org.jetbrains.kotlin.android:experimental=false" />
              <option
value="plugin:org.jetbrains.kotlin.android:enabled=true" />
              <option
value="plugin:org.jetbrains.kotlin.android:defaultCacheImplementation=
hashMap" />
            </array>
          </option>
          <option name="pluginClasspaths">
            <array />
          </option>
        </compilerArguments>
      </configuration>
    </facet>
  </component>
  <component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_8">
    <output
url="file://$MODULE_DIR$/../../../app/build/intermediates/javac/debug/
classes" />
    <output-test
url="file://$MODULE_DIR$/../../../app/build/intermediates/javac/debugU
nitTest/classes" />
    <exclude-output />
    <content url="file://$MODULE_DIR$/../../../app">
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/src/main/java"
isTestSource="false" />
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/src/androidTest/java"
isTestSource="true" />
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/src/test/java"
isTestSource="true" />
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/src/main/res" type="java-
resource" />
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/build/generated/ap_generated_sou
rces/debug/out" isTestSource="false" generated="true" />
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/build/generated/source/buildConf
ig/debug" isTestSource="false" generated="true" />
```

```
      <sourceFolder
url="file://$MODULE_DIR$/../../../app/build/generated/res/resValues/de
bug" type="java-resource" />
      <excludeFolder url="file://$MODULE_DIR$/../../../app/.gradle" />
      <excludeFolder url="file://$MODULE_DIR$/../../../app/build" />
    </content>
    <orderEntry type="jdk" jdkName="Android API 30 Platform"
jdkType="Android SDK" />
    <orderEntry type="sourceFolder" forTests="false" />
    <orderEntry type="library" name="Gradle:
org.jetbrains.kotlin:kotlin-android-extensions-runtime:1.5.20-RC"
level="project" />
    <orderEntry type="library" name="Gradle:
org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.5.20-RC" level="project" />
    <orderEntry type="library" name="Gradle:
org.jetbrains.kotlin:kotlin-stdlib:1.5.20-RC" level="project" />
    <orderEntry type="library" name="Gradle:
org.jetbrains:annotations:13.0" level="project" />
    <orderEntry type="library" name="Gradle:
org.jetbrains.kotlin:kotlin-stdlib-common:1.5.20-RC" level="project"
/>
    <orderEntry type="library" name="Gradle:
androidx.collection:collection:1.1.0" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.lifecycle:lifecycle-common:2.1.0" level="project" />
    <orderEntry type="library" name="Gradle: androidx.arch.core:core-
common:2.1.0" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.annotation:annotation:1.2.0" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.constraintlayout:constraintlayout-solver:1.1.3"
level="project" />
    <orderEntry type="library" name="Gradle: androidx.core:core-
ktx:1.7.0-alpha01@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.appcompat:appcompat:1.1.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.fragment:fragment:1.1.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.appcompat:appcompat-resources:1.1.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.drawerlayout:drawerlayout:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.viewpager:viewpager:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.loader:loader:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.activity:activity:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.vectordrawable:vectordrawable-animated:1.1.0@aar"
level="project" />
    <orderEntry type="library" name="Gradle:
androidx.vectordrawable:vectordrawable:1.1.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.customview:customview:1.0.0@aar" level="project" />
```

```
    <orderEntry type="library" name="Gradle: androidx.core:core:1.7.0-
alpha01@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.cursoradapter:cursoradapter:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.versionedparcelable:versionedparcelable:1.1.1@aar"
level="project" />
    <orderEntry type="library" name="Gradle:
androidx.lifecycle:lifecycle-runtime:2.1.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.lifecycle:lifecycle-viewmodel:2.1.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.savedstate:savedstate:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.lifecycle:lifecycle-livedata:2.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.lifecycle:lifecycle-livedata-core:2.0.0@aar" level="project"
/>
    <orderEntry type="library" name="Gradle: androidx.arch.core:core-
runtime:2.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.interpolator:interpolator:1.0.0@aar" level="project" />
    <orderEntry type="library" name="Gradle:
androidx.annotation:annotation-experimental:1.1.0@aar" level="project"
/>
    <orderEntry type="library" name="Gradle:
androidx.constraintlayout:constraintlayout:1.1.3@aar" level="project"
/>
    <orderEntry type="library" name="Gradle:
com.github.StephenVinouze:KontinuousSpeechRecognizer:1.1.0@aar"
level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
junit:junit:4.12" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
org.hamcrest:hamcrest-integration:1.3" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
org.hamcrest:hamcrest-library:1.3" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
org.hamcrest:hamcrest-core:1.3" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
com.squareup:javawriter:2.1.1" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
javax.inject:javax.inject:1" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
com.google.code.findbugs:jsr305:2.0.1" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
androidx.test.ext:junit:1.1.1@aar" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
androidx.test.espresso:espresso-core:3.3.0@aar" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
androidx.test:runner:1.3.0@aar" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
androidx.test:core:1.2.0@aar" level="project" />
    <orderEntry type="library" scope="TEST" name="Gradle:
androidx.test:monitor:1.3.0@aar" level="project" />
```

```
    <orderEntry type="library" scope="TEST" name="Gradle:
androidx.test.espresso:espresso-idling-resource:3.3.0@aar"
level="project" />
  </component>
</module>
```

d. TextToSpeech.iml

```
<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id="TextToSpeech"
external.linked.project.path="$MODULE_DIR$/../.."
external.root.project.path="$MODULE_DIR$/../.."
external.system.id="GRADLE" external.system.module.group=""
external.system.module.version="unspecified" type="JAVA_MODULE"
version="4">
  <component name="FacetManager">
    <facet type="java-gradle" name="Java-Gradle">
      <configuration>
        <option name="BUILD_FOLDER_PATH"
value="$MODULE_DIR$/../../build" />
        <option name="BUILDABLE" value="false" />
      </configuration>
    </facet>
  </component>
  <component name="NewModuleRootManager" inherit-compiler-
output="true">
    <exclude-output />
    <content url="file://$MODULE_DIR$/../..">
      <excludeFolder url="file://$MODULE_DIR$/../../.gradle" />
      <excludeFolder url="file://$MODULE_DIR$/../../build" />
    </content>
    <orderEntry type="inheritedJdk" />
    <orderEntry type="sourceFolder" forTests="false" />
  </component>
</module>
```

e. Recognation

```
init {
   recognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
   RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH)

   recognizerIntent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
   context.packageName)

   recognizerIntent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 3)

   recognizerIntent.putExtra(RecognizerIntent.EXTRA_PREFER_OFFLINE,
   true)

            initializeRecognizer()
       }

       private fun initializeRecognizer() {
```

```
        if (SpeechRecognizer.isRecognitionAvailable(context)) {
            speech =
SpeechRecognizer.createSpeechRecognizer(context)
            speech?.setRecognitionListener(this)
            callback?.onPrepared(if (speech != null)
RecognitionStatus.SUCCESS else RecognitionStatus.FAILURE)
        } else {
            callback?.onPrepared(RecognitionStatus.UNAVAILABLE)
        }
    }

    fun destroyRecognizer() {
        muteRecognition(false)
        speech?.destroy()
    }

    fun startRecognition() {
        if (!isListening) {
            isListening = true
            speech?.startListening(recognizerIntent)
        }
    }

    fun stopRecognition() {
        speech?.stopListening()
    }

    fun cancelRecognition() {
        speech?.cancel()
    }

    private fun muteRecognition(mute: Boolean) {
        val flag = if (mute) AudioManager.ADJUST_MUTE else
AudioManager.ADJUST_UNMUTE
        audioManager.adjustStreamVolume(AudioManager.STREAM_MUSIC,
flag, 0)
        audioManager.adjustStreamVolume(AudioManager.STREAM_SYSTEM,
flag, 0)
    }

    override fun onBeginningOfSpeech() {
        callback?.onBeginningOfSpeech()
    }

    override fun onReadyForSpeech(params: Bundle) {
        muteRecognition((shouldMute != null && shouldMute) ||
!isActivated)
        callback?.onReadyForSpeech(params)
    }

    override fun onBufferReceived(buffer: ByteArray) {
        callback?.onBufferReceived(buffer)
    }

    override fun onRmsChanged(rmsdB: Float) {
        callback?.onRmsChanged(rmsdB)
    }
```

```
    override fun onEndOfSpeech() {
        callback?.onEndOfSpeech()
    }

    override fun onError(errorCode: Int) {
        if (isActivated) {
            callback?.onError(errorCode)
        }
        isActivated = false
        isListening = false

        when (errorCode) {
            SpeechRecognizer.ERROR_RECOGNIZER_BUSY ->
cancelRecognition()
            SpeechRecognizer.ERROR_SPEECH_TIMEOUT -> {
                destroyRecognizer()
                initializeRecognizer()
            }
        }

        startRecognition()
    }
    override fun onEvent(eventType: Int, params: Bundle) {
        callback?.onEvent(eventType, params)
    }
    override fun onPartialResults(partialResults: Bundle) {
        val matches =
partialResults.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITI
ON)
        if (isActivated && matches != null) {
            callback?.onPartialResults(matches)
        }
    }
    override fun onResults(results: Bundle) {
        val matches =
results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION)
        val scores =
results.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES)
        if (matches != null) {
            if (isActivated) {
                isActivated = false
                callback?.onResults(matches, scores)
            } else {
                matches.forEach {
                    if (it.contains(other = activationKeyword,
ignoreCase = true)) {
                        isActivated = true
                        callback?.onKeywordDetected()
                        return@forEach
                    }
                }
            }
        }

        isListening = false
        startRecognition()
    }
}
```