

Name: \_\_\_\_\_ ( ) Date: \_\_\_\_\_

### **Chapter 7: Advanced Internal System Components and APIs**

---

#### **7.1 Introduction to Chapter 7**

This chapter is an extension of Chapter 6 where this chapter will explore further on the application of intents, first extending the discussion on implicit intents in general to implementing implicit intents which require permissions. After which, advanced APIs like the speech-to-text (STT) and text-to-speech(TTS) and the Android Sensor Framework will be discussed. Optional sections such as the Google Map API and the Firebase API for useful components like user login and password authentications, databases and cloud storage.

Note that all examples in this chapter require the use of a proper working Android device to run the apps as the emulator will not be able to achieve the outcomes required out of the apps due to their nature of interactivity.

#### **7.2 Android Internal System Components That Require Permissions**

We have seen in Chapter 6 that there are some implicit intents which require permissions to run such as capturing images or even reading and writing files. Such permissions are put in place so that **user consent is captured** before personal data or actions which may invade the user's privacy will be enacted. **This security measure is put in place to protect both the user and the developer**, such that the user's privacy and data is protected and that the developer is protected as well as consent was enquired before use of the respective functionalities of the app.

Capturing the permissions programmatically and subsequently granting the permissions to the app will require quite a fair bit of work in terms of the code. The following shows an example of capturing images from the phone's camera and subsequently displaying it on the ImageView, or pulling an image from the phone's internal gallery and setting it on the app's ImageView.

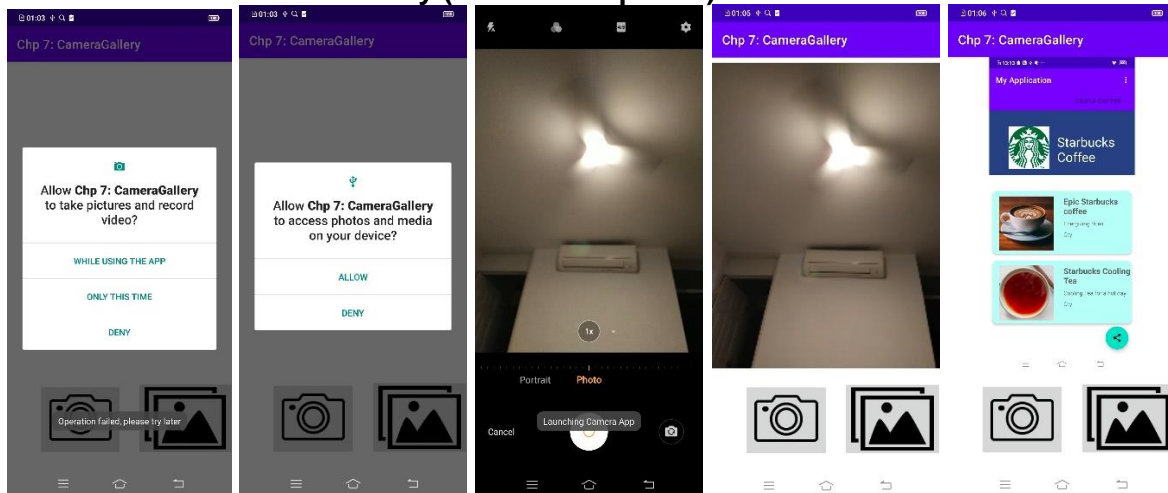
Refer to Chapter 6 for information on implicit intents which require permissions.

##### **Example 1: CameraGallery**

This app allows the user to pick from either capturing an image on the phone's camera then displaying it in the ImageView on top of the buttons OR pulling an image from the phone's internal storage or gallery and getting the ImageView to reference that image's URI.

Before being able to use the camera or retrieve files from the phone's storage, permissions are requested (if already not granted), to access the camera and access the phone's storage before the functionalities can be utilized.

## Screenshots of CameraGallery (from actual phone)



**You will need to declare the permissions needed for the app within the Android Manifest.** Since this app requires the camera and reading of the external storage, only those permissions are declared. It is important to keep the permission declarations concise as permissions that you declare in the Android Manifest may affect how your app is filtered within Google Play when the app is published for use. Of course, you may control how filtering is done on Google Play by using `<uses-feature>` instead to explicitly declare what kind of hardware the app uses, instead of letting Google Play guess the hardware use. You may read up more here:

<https://developer.android.com/guide/topics/manifest/uses-permission-element>

## AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="...">
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application ...
    ...
```

## strings.xml

```
<resources>
    <string name="app_name">Chp 7: CameraGallery</string>
    <string name="camera">Launching Camera App</string>
    <string name="gallery">Launching Gallery</string>
    <string name="fail">Operation failed...</string>
</resources>
```

## MainActivity.kt

```
1 import android.Manifest
2 import android.content.Intent
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.provider.MediaStore
6 import android.view.View
7 import android.widget.Toast
8 import androidx.core.app.ActivityCompat
9
10 import android.content.pm.PackageManager
11 import androidx.core.content.ContextCompat
12 import android.app.Activity
13 import android.widget.ImageView
```

```

14 import android.graphics.Bitmap
15
16 class MainActivity : AppCompatActivity() {
17     val REQUEST_ID_MULTIPLE_PERMISSIONS = 101
18     val REQUEST_CAMERA_CODE = 200
19     val REQUEST_GALLERY_CODE = 300
20     private lateinit var imageView: ImageView
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         setContentView(R.layout.activity_main)
25         imageView = findViewById(R.id.imageView)
26     }
27
28     fun processOnClicked(view: View) {
29         lateinit var intent: Intent
30         try {
31             val toastText : String
32             var request = 0
33             when (view.id) {
34                 R.id.cameraBtn -> {
35                     if (checkAndRequestPermissions(this@MainActivity)) {
36                         intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
37                         request = REQUEST_CAMERA_CODE
38                         toastText = getString(R.string.camera)
39                     } else {
40                         toastText = "Permissions not granted"
41                     }
42                 }
43                 R.id.galleryBtn -> {
44                     if (checkAndRequestPermissions(this@MainActivity)) {
45                         intent = Intent(Intent.ACTION_PICK,
46                             MediaStore.Images.Media.INTERNAL_CONTENT_URI)
47                         request = REQUEST_GALLERY_CODE
48                         toastText = getString(R.string.gallery)
49                     } else {
50                         toastText = "Permissions not granted"
51                     }
52                 }
53             } else -> {
54                 toastText = getString(R.string.fail)
55             }
56         }
57         Toast.makeText(this@MainActivity, toastText, Toast.LENGTH_SHORT).show()
58         if (intent.resolveActivity(packageManager) != null)
59             startActivityForResult(intent, request)
60     } catch (ex: Exception) {
61         Toast.makeText(this@MainActivity,
62             "Operation failed, please try later", Toast.LENGTH_SHORT).show()
63     }
64 }
65
66 fun checkAndRequestPermissions(context: Activity?): Boolean {
67     val ExtstorePermission = ContextCompat.checkSelfPermission(
68         context!!, Manifest.permission.READ_EXTERNAL_STORAGE)
69     val cameraPermission = ContextCompat.checkSelfPermission(
70         context, Manifest.permission.CAMERA)
71     val listPermissionsNeeded: MutableList<String> = ArrayList()
72
73     if (cameraPermission != PackageManager.PERMISSION_GRANTED)
74         listPermissionsNeeded.add(Manifest.permission.CAMERA)
75     if (ExtstorePermission != PackageManager.PERMISSION_GRANTED)
76         listPermissionsNeeded.add(Manifest.permission.READ_EXTERNAL_STORAGE)
77
78     if (listPermissionsNeeded.isNotEmpty()) {
79         ActivityCompat.requestPermissions(context, listPermissionsNeeded.toTypedArray(),

```

79	REQUEST_ID_MULTIPLE_PERMISSIONS)
80	return false
81	}
82	return true
83	}
84	override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String?>,
85	grantResults: IntArray) {
86	super.onRequestPermissionsResult(requestCode, permissions, grantResults)
87	when (requestCode) {
88	REQUEST_ID_MULTIPLE_PERMISSIONS -> {
89	if (ContextCompat.checkSelfPermission(this@MainActivity,
90	Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
91	Toast.makeText(applicationContext, "App Requires Access to Camara.",
92	Toast.LENGTH_SHORT).show()
93	} else if (ContextCompat.checkSelfPermission(this@MainActivity,
94	Manifest.permission.READ_EXTERNAL_STORAGE) !=
95	PackageManager.PERMISSION_GRANTED) {
96	Toast.makeText(applicationContext,
97	"App Requires Access to Your Storage.", Toast.LENGTH_SHORT).show()
98	}
99	}
100	}
101	}
102	override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
103	super.onActivityResult(requestCode, resultCode, data)
104	if (resultCode == Activity.RESULT_OK) {
105	if (requestCode == REQUEST_CAMERA_CODE && data != null) {
106	val selectedImage = data.extras!!.get("data") as Bitmap
107	imageView.setImageBitmap(selectedImage)
108	} else if (requestCode == REQUEST_GALLERY_CODE && data != null) {
109	val selectedImage = data.data
110	imageView.setImageURI(selectedImage)
111	}
112	}
113	}
114	}

## Explanation:

Line 34 to 52 Line 57 to 59 Line 102 to 113	<p>Selecting the options (either camera or gallery) will have to go through a check as to whether permissions have already been granted. The implicit intent is only passed if the permissions have already been granted or the user has newly given consent. Note that as the implicit intent is called, a request code is set up and passed into the overridden onActivityResult().</p> <p>Within the overridden onActivityResult(), if the result code is all good, based on the respective request code, put up either the captured image or the picture taken from gallery and put it into the ImageView. Note that if it is from the camera, the image can be simply obtained from the intent data and casted into a bitmap. Whereas, if it is obtained from the gallery, the intent instead contains the URI link to the image and the image is drawn from the URI link.</p>
Line 65 to 100	<p>In the checkAndRequestPermissions function, the function does a check on whether the individual permissions have been granted <b>within the device for the app</b>. If any permissions are not already granted, a permission request will be called for each of these, which is where the permissions dialog appears. In the overridden onRequestPermissionsResult, it processes the permissions granted. If any of the permissions are not granted by the user, the app will</p>

	display a Toast to tell the user that a certain functionality cannot be carried out because permissions are required.
--	---

## activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toTopOf="@+id/cameraBtn"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
        android:id="@+id/cameraBtn"
        android:layout_width="133dp"
        android:layout_height="116dp"
        android:layout_marginBottom="24dp"
        android:onClick="processOnClicked"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.223"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        app:srcCompat="@drawable/camera" />

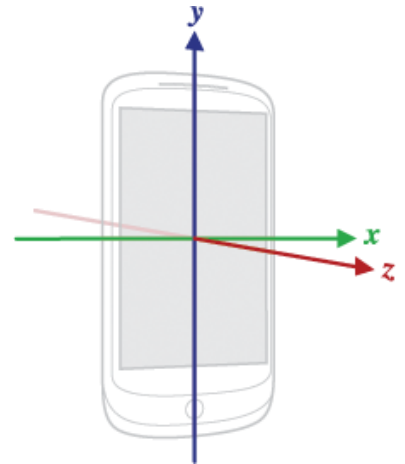
    <ImageButton
        android:id="@+id/galleryBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginBottom="24dp"
        android:onClick="processOnClicked"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toEndOf="@+id/cameraBtn"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        app:srcCompat="@drawable/gallery" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

### 7.3 Android Sensor Framework

Most Android devices have built-in sensors that measure motion, orientation, and various environmental conditions. Some of the sensors are hardware based and some are software based. Whatever the sensor is, Android allows us to get the raw data from these sensors and use it in our application. Android supports three broad categories of sensors:

- **Motion Sensors:** Responsible for measuring or identifying the shakes and tilts of your Android devices. These sensors measure the rotational forces along the three-axis. Some examples of motion sensors include, Gravity sensors and accelerometers.
- **Environmental Sensors:** Environmental properties like temperature, pressure, humidity, etc are identified with the help of environmental sensors. Some of the examples of Environmental sensors are thermometer, photometer, barometer, etc.
- **Position Sensors:** Responsible in determining the position of an Android device. Magnetometers, proximity sensors are some of the examples of position sensors.



Specific details on the available sensors are below:

Available Sensors	Hard/Soft	Description
<i>TYPE_ACCELEROMETER</i>	Hardware	Measures the acceleration force in m/s <sup>2</sup> that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.
<i>TYPE_AMBIENT_TEMPERATURE</i>	Hardware	Measures the ambient room temperature in degrees Celsius (°C).
<i>TYPE_GRAVITY</i>	Either	Measures the force of gravity in m/s <sup>2</sup> that is applied to a device on all three physical axes (x, y, z).
<i>TYPE_GYROSCOPE</i>	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, z).
<i>TYPE_LIGHT</i>	Hardware	Measures the ambient light level (illumination) in lx.
<i>TYPE_LINEAR_ACCELERATION</i>	Either	Measures the acceleration force in m/s <sup>2</sup> that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.
<i>TYPE_MAGNETIC_FIELD</i>	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in $\mu$ T.
<i>TYPE_ORIENTATION</i>	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z).
<i>TYPE_PRESSURE</i>	Hardware	Measures the ambient air pressure in hPa or mbar.
<i>TYPE_PROXIMITY</i>	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.
<i>TYPE_RELATIVE_HUMIDITY</i>	Hardware	Measures the relative ambient humidity in percent (%).
<i>TYPE_ROTATION_VECTOR</i>	Either	Measures the orientation of a device by providing the three elements of the device's rotation vector.

Access to the sensors and acquiring raw sensor data are done by using the Android sensor framework. The sensor framework is part of the `android.hardware` package and includes the following classes and interfaces:

- **SensorManager:** Creates an instance of the sensor service. It provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information. This class also provides several sensor constants used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

You may read more here:

<https://developer.android.com/reference/android/hardware/SensorManager>

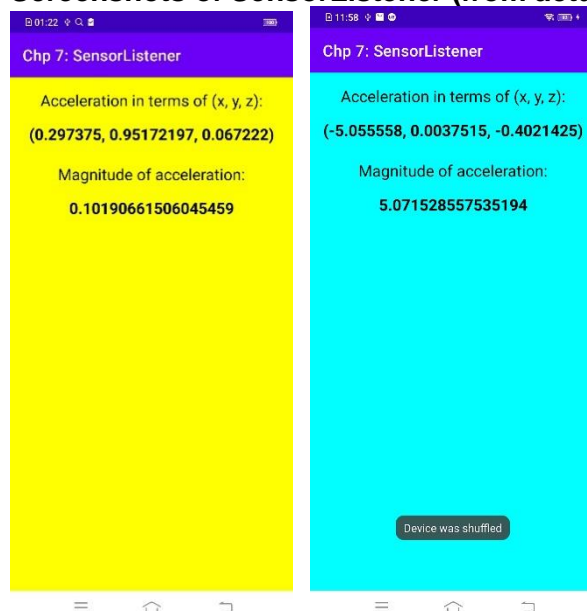
- **Sensor:** Create an instance of a specific sensor. This class provides various methods that let you determine a sensor's capabilities.
- **SensorEvent:** A sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, accuracy of the data, and the timestamp for the event.
- **SensorEventListener:** This interface creates two callback methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

The following example shows an application of the accelerometer tested with an actual phone.

### **Example 2: SensorListener**

This app will measure the acceleration of the device as the user shakes the phone. If the acceleration is large enough, the main layout of the app will change colour between cyan and yellow. Note that the acceleration values already compensates for the effect of acceleration due to gravity.

### **Screenshots of SensorListener (from actual phone)**



## strings.xml

```

<resources>
    <string name="app_name">Chp 7: SensorListener</string>
    <string name="accel">Acceleration in terms of (x, y, z):</string>
    <string name="shake">Shake the device!</string>
    <string name="accelMag">Magnitude of acceleration:</string>
    <string name="shuffle">Device was shuffled</string>
</resources>

```

## MainActivity.kt

```

1  import android.graphics.Color
2  import android.hardware.Sensor
3  import android.hardware.SensorEvent
4  import android.hardware.SensorEventListener
5  import android.hardware.SensorManager
6  import androidx.appcompat.app.AppCompatActivity
7  import android.os.Bundle
8  import android.widget.TextView
9  import android.widget.Toast
10 import androidx.constraintlayout.widget.ConstraintLayout
11 import kotlin.math.sqrt
12
13 class MainActivity : AppCompatActivity() {
14     private lateinit var accelVec : TextView
15     private lateinit var accelMag : TextView
16     private lateinit var mainLayout : ConstraintLayout
17     private lateinit var sensorM : SensorManager
18     private lateinit var sensorList : List<Sensor>
19     private lateinit var sensorListener : SensorEventListener
20     private var isColor = false
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         setContentView(R.layout.activity_main)
25         sensorM = getSystemService(SENSOR_SERVICE) as SensorManager
26         sensorList = sensorM.getSensorList(Sensor.TYPE_ACCELEROMETER)
27
28         accelVec = findViewById(R.id.accelVector)
29         accelMag = findViewById(R.id.accelMag)
30         mainLayout = findViewById(R.id.mainLayout)
31         mainLayout.setBackgroundColor(Color.YELLOW)
32
33         var actualTime : Long
34         var lastUpdate = System.currentTimeMillis()
35         sensorListener = object : SensorEventListener {
36             override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {}
37             override fun onSensorChanged(event: SensorEvent) {
38                 val coords = event.values
39                 val g = SensorManager.GRAVITY_EARTH
40                 accelVec.text = "${coords[0]/g}, ${coords[1]/g}, ${coords[2]/g}"
41                 val accelMagVal = sqrt((coords[0]*coords[0] + coords[1]*coords[1] +
42                     coords[2]*coords[2]).toDouble())/g
43                 actualTime = event.timestamp
44                 accelMag.text = "${accelMagVal}"
45                 if(accelMagVal >= 2){
46                     if (actualTime - lastUpdate >= 200){
47                         lastUpdate = actualTime
48                         Toast.makeText(applicationContext, getString(R.string.shuffle),
49                             Toast.LENGTH_SHORT).show()
50                         if (isColor)
51                             mainLayout.setBackgroundColor(Color.YELLOW)
52                         else
53                             mainLayout.setBackgroundColor(Color.CYAN)
54                         isColor = !isColor

```



55	}
56	}
57	}
58	}
59	if (sensorList.isNotEmpty())
60	sensorM.registerListener(sensorListener, sensorList.get(0) as Sensor,
61	SensorManager.SENSOR_DELAY_NORMAL)
62	else
63	Toast.makeText(this, "Error: No Accelerometer.", Toast.LENGTH_LONG).show()
64	}
65	override fun onResume() {
66	super.onResume()
67	// register this class as a listener for the orientation and accelerometer sensors
68	sensorM.registerListener(this.sensorListener,
69	sensorM.getDefaultSensor(Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_NORMAL)
70	}
71	override fun onStop() {
72	if (sensorList.isNotEmpty()) {
73	sensorM.unregisterListener(sensorListener)
74	}
75	super.onStop()
76	}
77	}

## Explanation:

Line 25 to 26 Line 33 to 64	<p>The SensorManager will get the accelerometer for use in this app and a sensor listener is set up which will detect changes in movement so that the sensor will update the values. The accelerometer will compute the acceleration vector as well as its magnitude and display it in the app.</p> <p>When the acceleration is fast enough (which will normally be when you do a fast hard shuffle on the phone), the app will toggle between cyan and yellow, and a Toast will be output to indicate that a shuffle happened. The code also keeps track of the timestamp of events such that a shuffle is only counted if it is 0.2 milliseconds from the last update.</p> <p>Note that compensation from gravity is just division by the scalar <math>g = 9.81</math> (or the constant <code>SensorManager.GRAVITY_EARTH</code>).</p>
Line 65 to 76	<p>Within the activity's onResume, the MainActivity class is registered as the main listener itself for changes in the acceleration. Note that you can also change rate of data retrieval from the sensor. The current setting of <code>SensorManager.SENSOR_DELAY_NORMAL</code> is the normal rate for just detecting normal changes in orientations and movement of the phone. There are rates available for games or something even faster.</p> <p>Within the activity's onStop, the MainActivity class will be deregistered as the main listener to ensure any possible conflicts with other apps or system configurations will not occur.</p>

## activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/title1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/accel"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/accelVector"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/shake"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/title1" />

    <TextView
        android:id="@+id/title2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/accelMag"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/accelVector"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/accelMag"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/shake"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/title2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 7.4 Android Speech-To-Text and Text-To-Speech

The speech-to-text (STT) and text-to-speech (TTS) features in Android is part of the Android Speech API which provides recognition, background services, intents and support for multiple languages. Speech recognition is supported by the **RecognizerIntent class**, supporting speech recognition intents with customisations available in the form of **actions** to support the most appropriate language model for recognition, and **SpeechRecognizer class**, supporting recognizing the speech and translating it into text. **On the topic of the user experience, this feature is very useful for users who may be either hard of sight or hard of hearing.**

Note that Google now supports a wide range of languages, ranging from English, and even Mandarin and Japanese.

### Example 3: TTS-STT-Example

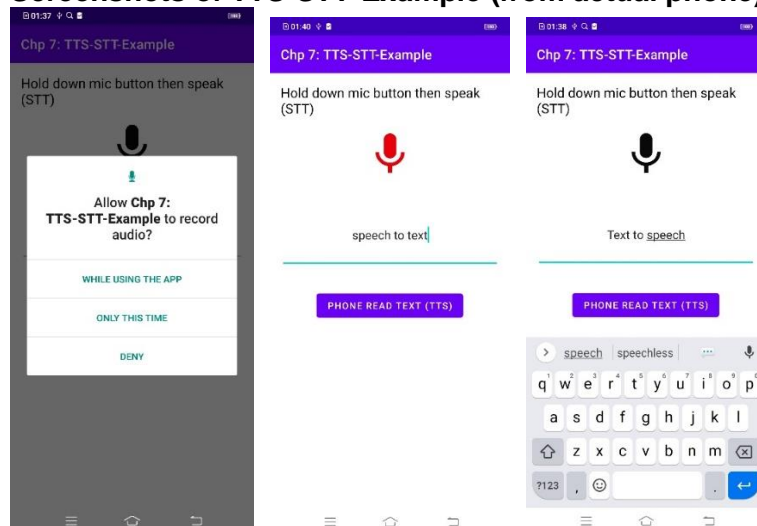
This app showcases both the STT and TTS functionalities. The STT functionality is shown through the holding down the microphone icon and speaking into the phone. The app will then display the recognized text in the EditText. The TTS functionality is shown through typing the text into the same EditText and subsequently pressing the button to read text, and an audio will be output reading the text.

Note that permissions need to be granted to record audio for recognition before the app's functionalities can be used and we will need to activate the recognition service query for STT.

#### AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package=...
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <queries>
        <intent>
            <action android:name="android.speech.RecognitionService" />
        </intent>
    </queries>
    <application ...
    ...
```

#### Screenshots of TTS-STT-Example (from actual phone)



## strings.xml

```

<resources>
    <string name="app_name">Chp 7: SensorListener</string>
    <string name="accel">Acceleration in terms of (x, y, z):</string>
    <string name="shake">Shake the device!</string>
    <string name="accelMag">Magnitude of acceleration:</string>
    <string name="shuffle">Device was shuffled</string>
</resources>

```

## MainActivity.kt

```

1  import androidx.core.content.ContextCompat
2  import android.Manifest
3  import android.content.Context
4  import android.content.Intent
5  import android.content.pm.PackageManager
6  import androidx.appcompat.app.AppCompatActivity
7  import android.os.Bundle
8  import android.speech.RecognitionListener
9  import android.speech.RecognizerIntent
10 import android.speech.tts.TextToSpeech
11 import android.view.MotionEvent
12 import android.view.inputmethod.InputMethodManager
13 import android.widget.Button
14 import android.widget.EditText
15 import android.widget.Toast
16 import java.util.*
17 import android.speech.SpeechRecognizer
18 import android.speech.tts.Voice
19 import android.widget.ImageView
20 import androidx.core.app.ActivityCompat
21
22 class MainActivity : AppCompatActivity() {
23     val RecordAudioRequestCode = 1
24     private var listening = false
25     private lateinit var btnSTT : ImageView
26     private lateinit var btnTTS : Button
27     private lateinit var editText : EditText
28     private lateinit var speechRecognizer: SpeechRecognizer
29     private lateinit var textToSpeech : TextToSpeech
30
31     override fun onCreate(savedInstanceState: Bundle?) {
32         super.onCreate(savedInstanceState)
33         setContentView(R.layout.activity_main)
34         if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO)
35             != PackageManager.PERMISSION_GRANTED ) {
36             ActivityCompat.requestPermissions(this,
37                 arrayOf(Manifest.permission.RECORD_AUDIO), RecordAudioRequestCode )
38         }
39         btnSTT = findViewById(R.id.btn_stt)
40         btnTTS = findViewById(R.id.btn_tts)
41         editText = findViewById(R.id.et_text_input)
42         textToSpeech = TextToSpeech(this){ status ->
43             if (status == TextToSpeech.SUCCESS) {
44                 textToSpeech.language = Locale.UK
45             }
46         }
47         val voiceObj = Voice("en-us-x-sfg#male_1-local", Locale.getDefault(),
48             1, 1, false, null)
49         textToSpeech.voice = voiceObj
50
51         val sttIntent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
52         sttIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
53             RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)
54         sttIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())

```

```

55
56     speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this)
57     speechRecognizer.setRecognitionListener(object : RecognitionListener {
58         override fun onReadyForSpeech(bundle: Bundle) {}
59         override fun onBeginningOfSpeech() {
60             editText.setText("")
61             editText.hint = "Listening..."
62         }
63         override fun onRmsChanged(v: Float) {}
64         override fun onBufferReceived(bytes: ByteArray) {}
65         override fun onEndOfSpeech() {}
66         override fun onError(i: Int) {}
67         override fun onResults(bundle: Bundle) {
68             val data = bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION)
69             if (data != null) {
70                 editText.setText(data[0])
71             } else {
72                 editText.setText(getString(R.string.errorSTT))
73             }
74         }
75         override fun onPartialResults(bundle: Bundle) {}
76         override fun onEvent(i: Int, bundle: Bundle) {}
77     })
78     btnSTT.setOnTouchListener { view, motionEvent ->
79         when (motionEvent.action) {
80             MotionEvent.ACTION_UP -> {
81                 speechRecognizer.stopListening()
82                 btnSTT.setImageResource(R.drawable.ic_mic_black_off)
83                 listening = false
84             }
85             MotionEvent.ACTION_DOWN -> {
86                 btnSTT.setImageResource(R.drawable.ic_mic_black_24dp)
87                 speechRecognizer.startListening(sttIntent)
88                 listening = true
89             }
90         }
91         false
92     }
93     btnTTS.setOnClickListener {
94         val text = editText.text.toString().trim()
95         if (text.isNotEmpty()) {
96             textToSpeech.speak(text, TextToSpeech.QUEUE_FLUSH, null, "tts1")
97         } else
98             Toast.makeText(this, getString(R.string.errorTTS), Toast.LENGTH_LONG).show()
99     }
100 }
101 override fun onPause() {
102     textToSpeech.stop()
103     super.onPause()
104 }
105 override fun onDestroy() {
106     textToSpeech.shutdown()
107     super.onDestroy()
108     speechRecognizer.destroy()
109 }
110 // The method will force the touch keyboard to hide when user touches anywhere on screen
111 override fun onTouchEvent(event: MotionEvent?): Boolean {
112     val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
113     if(imm.isAcceptingText) imm.hideSoftInputFromWindow(currentFocus!!.windowToken, 0)
114     return true
115 }
116 override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String?>,
117     grantResults: IntArray) {
118     super.onRequestPermissionsResult(requestCode, permissions, grantResults)
119     if (requestCode == RecordAudioRequestCode && grantResults.isNotEmpty()) {

```

120	if (grantResults[0] == PackageManager.PERMISSION_GRANTED)
121	Toast.makeText(this, "Permission Granted", Toast.LENGTH_SHORT).show()
122	}
123	}
124	}

**Explanation:**

Line 42 to 49 Line 93 to 100	<p>The TTS code is more straightforward such that the TextToSpeech class can be directly instantiated and the voice Locale can be set so that the accent is as easily understandable by the user as possible (for us, the Locale.UK is good enough).</p> <p>When the button is clicked, the speak method is called. The QUEUE_FLUSH parameter is meant to “flush out” any previous input to start the TTS in a clean slate. Alternatively, you may use QUEUE_ADD to append the current text to speech.</p> <p>Note that before speak() is called, you can set both the pitch (textToSpeech.pitch) and the speed of speech (textToSpeech.speechRate) by setting the values to be different from 1.0, which is the default setting for both.</p>
Line 51 to 92	<p>The STT code is more complex such that you will need to go through a series of steps. First the Recognizer intent needs to be set up to recognize speech (ACTION_RECOGNIZE_SPEECH) and flags need to be set where:</p> <ul style="list-style-type: none"> <li>• LANGUAGE_MODEL_FREE_FORM: Considers input in free form English</li> <li>• Locale.getDefault(): Also gets the locale default for the language</li> </ul> <p>The SpeechRecognizer will also need to be set up with a listener to such that upon some speech, the recognizer will start listening (line 87) and the intent is received by itself for processing, which is where lines 68 to 74 come in, where the data is obtained from the intent and placed into the EditText.</p> <p>Note that from lines 78 to 92, a touch listener is set on the microphone icon to change colour upon the user’s tap and hold onto the icon.</p>
Line 101 to 109	<p>In the lifecycle, it is important to ensure the TTS feature is temporarily paused when the app is inactive to ensure the processing in the TTS does not conflict with any other apps.</p> <p>When the app stops, it is good practice to properly destroy the TTS and STT functionalities so that there is no unintentional recording of speech or processing of text to speech.</p>

## activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/speakTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/speak"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/btn_stt"
        android:layout_width="75dp"
        android:layout_height="79dp"
        android:layout_marginTop="16dp"
        android:src="@drawable/ic_mic_black_off"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/speakTextView"
        app:layout_constraintVertical_bias="0.0" />

    <EditText
        android:id="@+id/et_text_input"
        android:layout_width="0dp"
        android:layout_height="100dp"
        android:layout_marginTop="32dp"
        android:layout_weight="1"
        android:gravity="center"
        android:hint="@string/editText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btn_stt" />

    <Button
        android:id="@+id/btn_tts"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/listen"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_text_input"
        app:layout_constraintVertical_bias="0.0" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

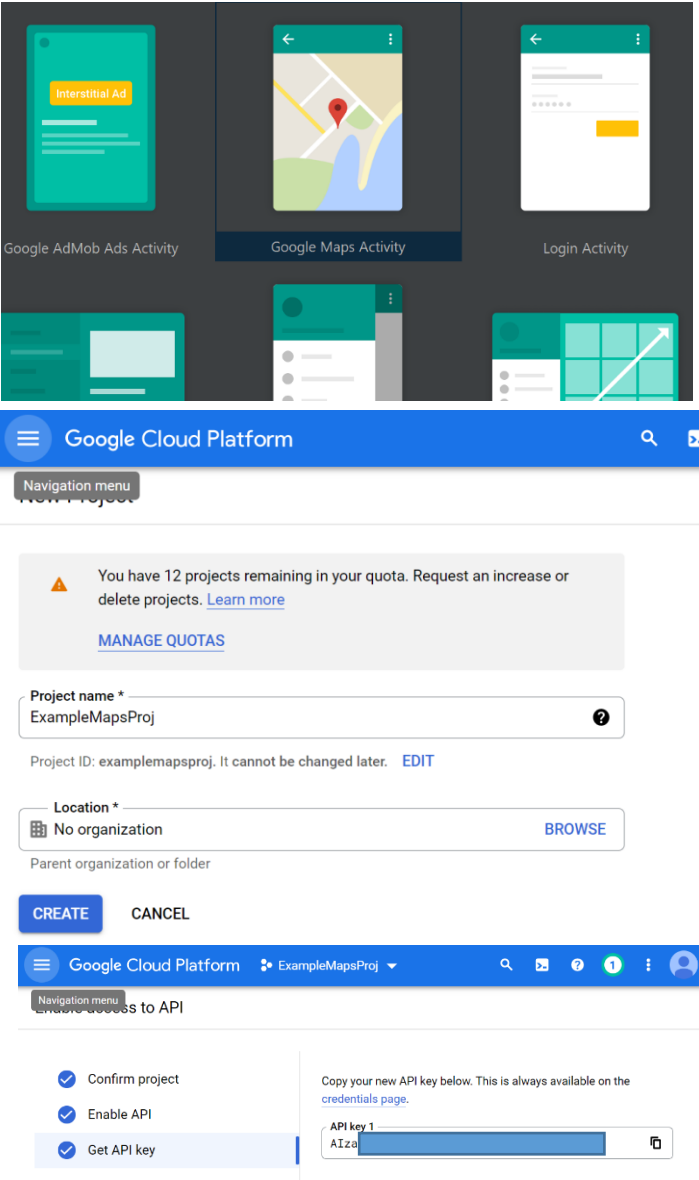
## 7.5 [Optional] Google Maps API

The Google Maps API for Android allows you to add a Google Map into your app. This section will cover both on how to add the API into the app as well as to programmatically add features into the map which can be seen in the example. For this you will have to open a new Maps Activity in Android Studio.

After the template Google Maps activity starts, to start using the Google Maps functionalities, you will need an API key inside the `google_maps_api.xml`. Follow the link given and you will reach the following page:

You need to give a project name and organization if applicable, then **create** the new project. Note that you will need a Google Account to even access the page. Upon clicking on **CREATE**, you will have to ensure everything in the checklist of the page is satisfied, only then will your API key given be working. Note that the API key will always start with `AIza`.

This API key will have to be then copy-pasted into the `google_maps_api.xml`:



```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
```

After this, your Maps Activity is now good to go.

You may look at the following documentation which includes a tutorial on how to set up the Google Maps API:

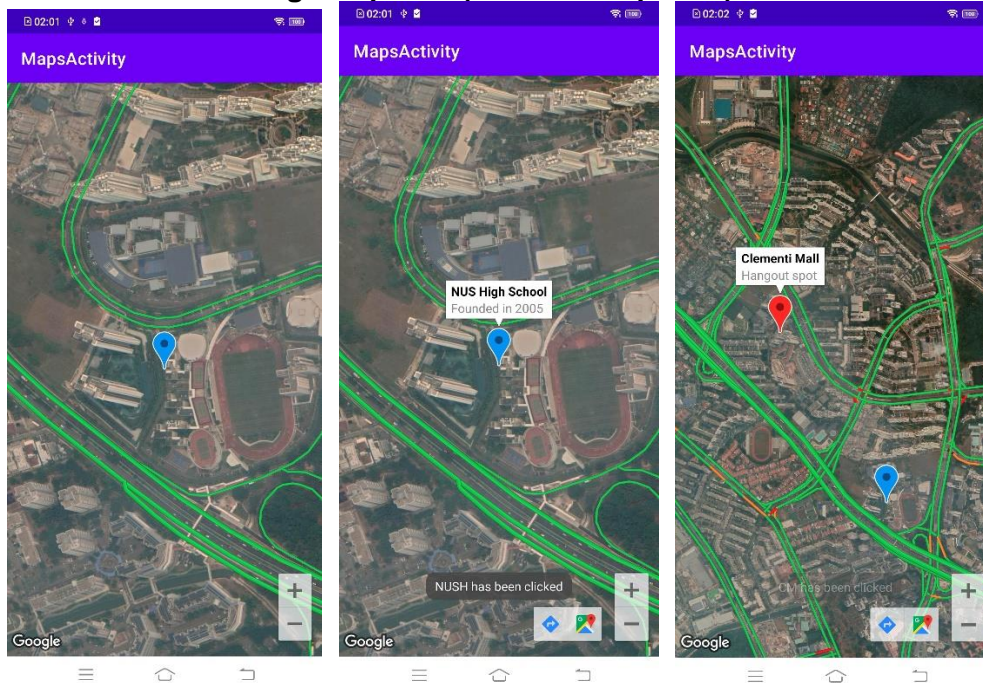
<https://developers.google.com/maps/documentation/android-sdk/overview?section=tutorials>

### Example 4: GoogleMapsAPI

This app focuses on where NUS High is located on the Google Maps and NUS High will be indicated by a blue pin. When the blue pin is clicked on, it will show some details as well as a Toast message on what is being clicked on. If you zoom out of the map, you will see Clementi Mall, as well as NUS marked by different coloured pins on the map as well. Note that the map type used is an satellite aerial photo instead of your conventional Google Map.



## Screenshots of GoogleMapsAPI (from actual phone)



## MapsActivity.kt

```

1  import androidx.appcompat.app.AppCompatActivity
2  import android.os.Bundle
3  import android.widget.Toast
4
5  import com.google.android.gms.maps.CameraUpdateFactory
6  import com.google.android.gms.maps.GoogleMap
7  import com.google.android.gms.maps.OnMapReadyCallback
8  import com.google.android.gms.maps.SupportMapFragment
9  import com.google.android.gms.maps.model.LatLng
10 import com.google.android.gms.maps.model.MarkerOptions
11 import com.example.googlemapsapi.databinding.ActivityMapsBinding
12 import com.google.android.gms.maps.model.BitmapDescriptorFactory
13 import com.google.android.gms.maps.model.Marker
14
15 class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
16     private lateinit var mMap: GoogleMap
17     private lateinit var binding: ActivityMapsBinding
18     private val NUSHIGH = LatLng(1.307, 103.769)
19     private val CLEMENTIMALL = LatLng(1.315, 103.764)
20     private val NUS = LatLng(1.297, 103.774)
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24
25         binding = ActivityMapsBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27
28         // Obtain the SupportMapFragment and get notified when the map is ready to be used.
29         val mapFragment = supportFragmentManager
30             .findFragmentById(R.id.map) as SupportMapFragment
31         mapFragment.getMapAsync(this)
32     }
33
34     /**
35      * Manipulates the map once available.
36      * This callback is triggered when the map is ready to be used.
37      * This is where we can add markers or lines, add listeners or move the camera.

```

```

38      * If Google Play services is not installed on the device, the user will be prompted to
39      install
40      * it inside the SupportMapFragment. This method will only be triggered once the user has
41      * installed Google Play services and returned to the app.
42      */
43      override fun onMapReady(googleMap: GoogleMap) {
44          mMap = googleMap
45          mMap.mapType = GoogleMap.MAP_TYPE_SATELLITE
46          mMap.isTrafficEnabled = true
47          mMap.uiSettings.isZoomControlsEnabled = true
48          mMap.setOnMarkerClickListener(object: GoogleMap.OnMarkerClickListener {
49              override fun onMarkerClick(marker: Marker): Boolean {
50                  Toast.makeText(applicationContext, "${marker.tag} has been clicked",
51                      Toast.LENGTH_SHORT).show()
52                  /* Return false to indicate that we have not consumed the event and wish
53                     for default behaviour to occur */
54                  return false
55              }
56          })
57
58          val mNUSHigh = mMap.addMarker(MarkerOptions().position(NUSHIGH)
59              .title("NUS High School")
60              .snippet("Founded in 2005")
61              .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)))
62          mNUSHigh.tag = ("NUSH")
63
64          val mClementiMall = mMap.addMarker(MarkerOptions().position(CLEMENTIMALL)
65              .title("Clementi Mall").snippet("Hangout spot"))
66          mClementiMall.tag = ("CM")
67
68          val mNUS = mMap.addMarker(MarkerOptions().position(NUS).title("NUS")
69              .snippet("Top University in Asia")
70              .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)))
71          mNUS.tag = ("NUS")
72
73          mMap.moveCamera(CameraUpdateFactory.newLatLng(NUSHIGH))
74          mMap.animateCamera(CameraUpdateFactory.zoomTo( 17.0f ))
75      }
76  }

```

## Explanation:

Line 18 to 20 Line 44 to 47	<p>Much of the code is already given and all that is needed is to manipulate onMapReady(). In lines 18 to 20, the latitude and longitude are tagged to the variables NUSHIGH, CLEMENTIMALL and NUS for the respective places so that they can be used to mark out the locations. Note that the latitudes and longitudes have to be given to <b>at least 3 decimal places</b>, otherwise, the app may not function properly.</p> <p>Some of the features of the Google Map are set, such as MAP_TYPE_SATELLITE (other options include MAP_TYPE_NORMAL, MAP_TYPE_HYBRID, MAP_TYPE_TERRAIN, MAP_TYPE_NONE), enabling traffic view and enabling zoom options.</p>
Line 48 to 71	<p>This code sets up the markers for the three locations. A listener is in place so that the moment the markers are clicked on, a Toast will be generated indicated what was clicked on. For each marker, some descriptions and a tag is defined and the colour of the marker is also set.</p>

Line 73 to 75	<p>The code represents the camera options such that the moveCamera() method will focus its initial attention to NUS High, such that NUS High is at the centre of the map. The animateCamera is set to a zoom of 17.0f, which is the usual standard and recommended for street views.</p> <p>You may read more about the Google Maps camera here:  <a href="https://developers.google.com/maps/documentation/android-sdk/views">https://developers.google.com/maps/documentation/android-sdk/views</a></p>
---------------	---

google\_maps\_api.xml

```
<resources>
    <string name="google_maps_key" templateMergeStrategy="preserve"
        translatable="false">AIza...</string> // Use your own API key.
</resources>
```

## 7.6 [Optional] Firebase API

The Firebase API is a mobile platform which gives the tools and infrastructure from Google to help develop and grow apps through features such as measuring user activity and engagement with analytics, cloud storage and cloud messaging, email and password authentication, as well as connecting all users of the app to a realtime database or a cloud database. Firebase is even beta-testing (as of 2021) their Machine Learning module for use which can provide AI features to your apps.

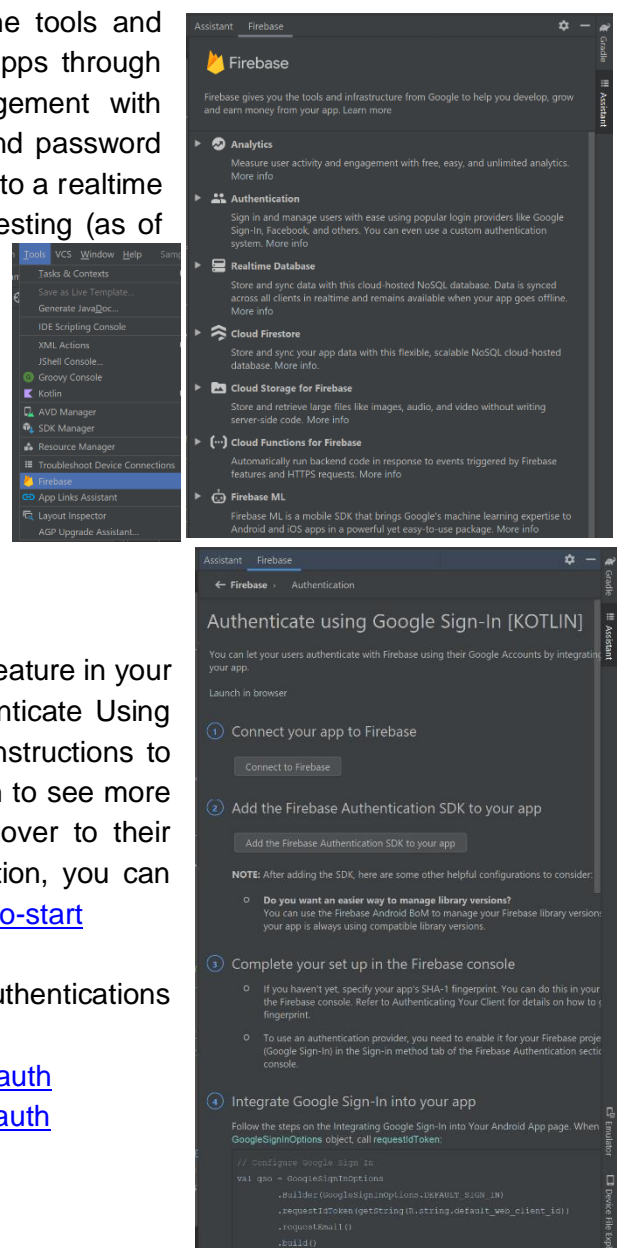
To add firebase into your project, you can go to Tools → Firebase. From here, a side panel will open which will guide you through the setup of Firebase into your app. Each of the options will guide you through what you will need to do and what code to enter to utilize that Firebase functionality.

For instance, if you wish to implement a Google Sign-In feature in your app, you can click on Authentication then select Authenticate Using Google Sign-In [KOTLIN], and it will then feature the instructions to implement the Google Sign-In authentication. If you wish to see more details in the implementations, you can always head over to their documentation to find out more. Specific to authentication, you can head here: <https://firebase.google.com/docs/auth/where-to-start>

Specific implementations with email link and password authentications can be found below.

<https://firebase.google.com/docs/auth/android/password-auth>

<https://firebase.google.com/docs/auth/android/email-link-auth>



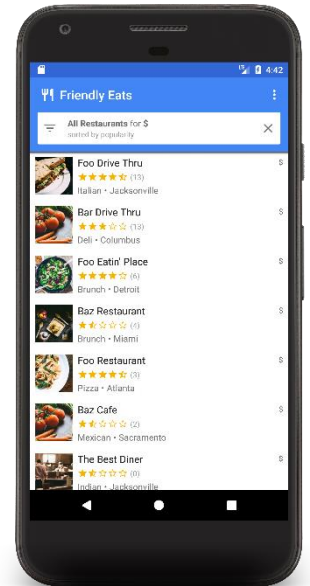
Another typical implementation of Firebase will be the implementation of a database to store and retrieve information, which you can see in the following links:

<https://firebase.google.com/docs/database/android/start>

<https://firebase.google.com/docs/storage/android/start>

Lastly while this section does not have an example, you may go to their open-source example to see how Firebase is implemented in the app context:

<https://firebaseopensource.com/projects/firebase/friendlyeats-android/>



### [Reference]

- [1] Android Permissions: <https://developer.android.com/guide/topics/permissions/overview>
- [2] Android App Manifest Permissions:  
<https://developer.android.com/reference/android/Manifest.permission>
- [3] Android Sensors Overview  
[https://developer.android.com/guide/topics/sensors/sensors\\_overview](https://developer.android.com/guide/topics/sensors/sensors_overview)
- [4] Android RecognizerIntent class:  
<https://developer.android.com/reference/android/speech/RecognizerIntent>
- [5] Android SpeechRecognizer class:  
<https://developer.android.com/reference/android/speech/SpeechRecognizer>
- [6] Android TextToSpeech class:  
<https://developer.android.com/reference/android/speech/tts/TextToSpeech>
- [7] GoogleMap class:  
<https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap>
- [8] FireBase Documentation: <https://firebase.google.com/docs/>
- [9] Connect to Firebase Using Android Studio:  
<https://developer.android.com/studio/write/firebase>