Telephony & SMS

- Dialing vs. Calling
- Dialing tutorial
- Calling overview
- SMS Overview
- Built in SMS App
- SmsManager
Background and Use Cases

- SMS and Calls used everywhere to communicate
- Useful in social apps or business apps to communicate with individuals or organizations
- Very widespread
- Enhances communication, solves problem of user manually dialing numbers
Dialing vs. Calling

- **Dial** - use the phone’s native dialing app
  - Preferred technique
  - Don't need to monitor the phone's state.
  - Can change number in dialer

- **Call** - place call from within the app
  - Request the user's permission
  - Make a phone call from within the app
  - Ability to monitor the phone's state.
  - Enables phone calls if the phone app has been disabled in Settings.
Formatting a number

- App must prepare a Uniform Resource Identifier (URI) for the phone number
- URI - string prefixed by "tel:", tel:14155551212
- Hard-code phone number or provide an EditText field
- The PhoneNumberUtils class provides utility methods for normalizing and formatting phone number strings.
- normalizeNumber() can remove extraneous characters (dashes or parentheses)
Display

- Use a button to let the user start the call.
- When the user taps the button, the click handler initiates the call, either dialing or calling
Call dialNumber() method upon tapping button

```java
public void dialNumber() {
    TextView textView = (TextView) findViewById(R.id.number_to_call);
    // Use format with "tel:" and phone number to create phoneNumber.
    String phoneNumber = String.format("tel: %s",
                                        textView.getText().toString());
    // Create the intent.
    Intent dialIntent = new Intent(Intent.ACTION_DIAL);
    // Set the data for the intent as the phone number.
    dialIntent.setData(Uri.parse(phoneNumber));
    // If package resolves to an app, send intent.
    if (dialIntent.resolveActivity(getPackageManager()) != null) {
        startActivity(dialIntent);
    } else {
        Log.e(TAG, "Can't resolve app for ACTION_DIAL Intent.");
    }
}
```
Calling

- Add permissions to enable making a call
- Check if telephony is enabled; if not, disable the phone feature.
- Check if the user grants permission, request permission if needed.
- Extend PhoneStateListener, register the listener using the TelephonyManager class.
- Use an implicit intent with ACTION_CALL to make the phone call.
Two ways of sending a text message:

- Built-in SMS Application
- SmsManager API
SMS Permissions

- Both methods require send-SMS permission

```xml
<uses-permission
    android:name="android.permission.SEND_SMS" />
```
Sending SMS – Built in SMS App

To call the default SMS application from phone:

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);
```
Sending SMS – SmsManager Api

- Call getDefault() to get the SmsManager instance with the default subscription ID
- Call divideMessage() to make sure the message is shorter than the text message length limit
- Iterate through the array of message strings, call sendTextMessage() to send them
Sending SMS – SmsManager Api

sendMessage.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String msg = message.getText().toString();
        String phoneNumber = phone.getText().toString();
        if (!TextUtils.isEmpty(msg) && !TextUtils.isEmpty(phoneNumber)) {
            if (checkPermission(Manifest.permission.SEND_SMS)) {
                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage(phoneNumber, null, msg, null, null);
                Toast.makeText(MainActivity.this, "Message Sent!", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(MainActivity.this, "Permission denied!", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(MainActivity.this, "Enter a message and a phone number!", Toast.LENGTH_SHORT).show();
        }
    }
});
public void sendTextMessage (  
        String destinationAddress,  
        String scAddress,  
        String text,  
        PendingIntent sentIntent,  
        PendingIntent deliveryIntent)

The destination and text are two not-null parameter fields.  
The rest fields can be null in certain scenarios.
References

- http://www.vlfeat.org/matconvnet/
- https://towardsdatascience.com/is-google-tensorflow-object-detection-api-the-easiest-way-to-implement-image-recognition-a8bd1f500ea0
- https://medium.com/salesforce-developers/continuous-integration-for-salesforce-lightning-development-a59adf1a21be
References

- Google Developer Training, “Android Apps: Phone Calls and SMS”, https://google-developer-training.gitbooks.io/android-developer-phone-sms-course/content/