

Name and B-number

COM413 Workbook

Contents

COM413 Workbook	1
COM413 Workbook Practical 1	2
Workbook 1 Basics:	2
Workbook 1 Advanced:	3
Workbook 1 Additional:	6
COM413 Workbook Practical 2	8

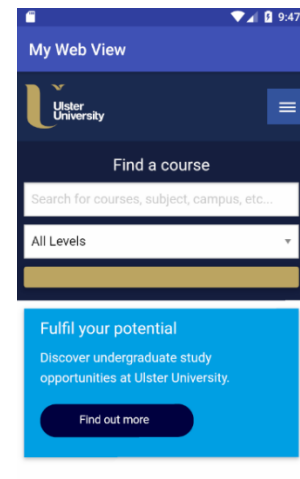
COM413 Workbook Practical 1

Workbook 1 Basics:

This app has one main feature, allowing the user to view web pages via the app. This can be handled in two ways. The first is to hardcode the URL and display this on the webview.

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

    WebView webview = new WebView(context, this);
    setContentView(webview);
    webview.loadUrl("https://www.ulster.ac.uk/");
}
```

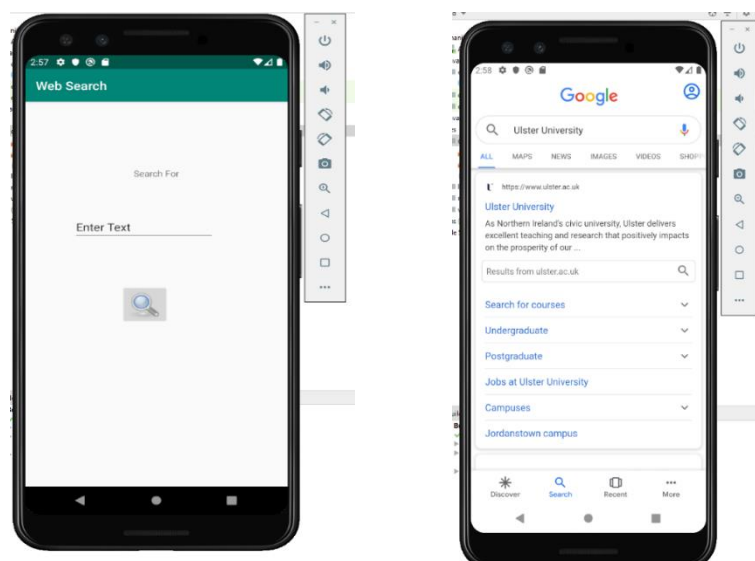


The second way of displaying a web page is to include a search bar. This search bar works by converting the text to a string and using an image button to perform a search. The app sends the string using an Intent, and the Android OS uses the best app to perform and display the result i.e. Google Chrome app.

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

public void webSearch(View view) {
    String searchFor = ((EditText) findViewById(R.id.editText)).getText().toString();
    Intent viewSearch = new Intent(Intent.ACTION_WEB_SEARCH);
    viewSearch.putExtra(SearchManager.QUERY, searchFor);
    startActivity(viewSearch);
}
```

Each element will be shown on the emulator below:



Workbook 1 Advanced:

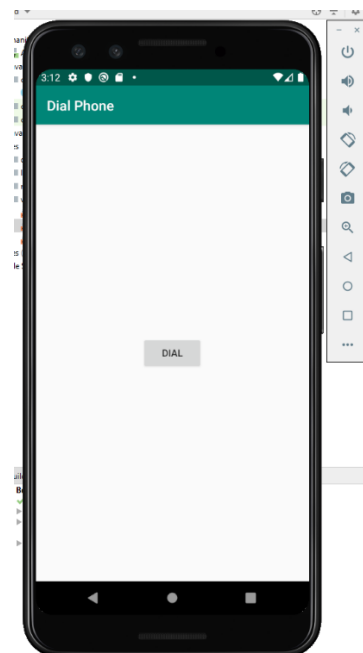
The advanced section consists of creating two buttons in an app to allow access to other features. A dial button and a way of sending SMS. I used two separate apps to build each function and then combined them for the advanced section. The important part here is to open the Android Manifest file to allow permissions.

Below is the XML Manifest and Java:

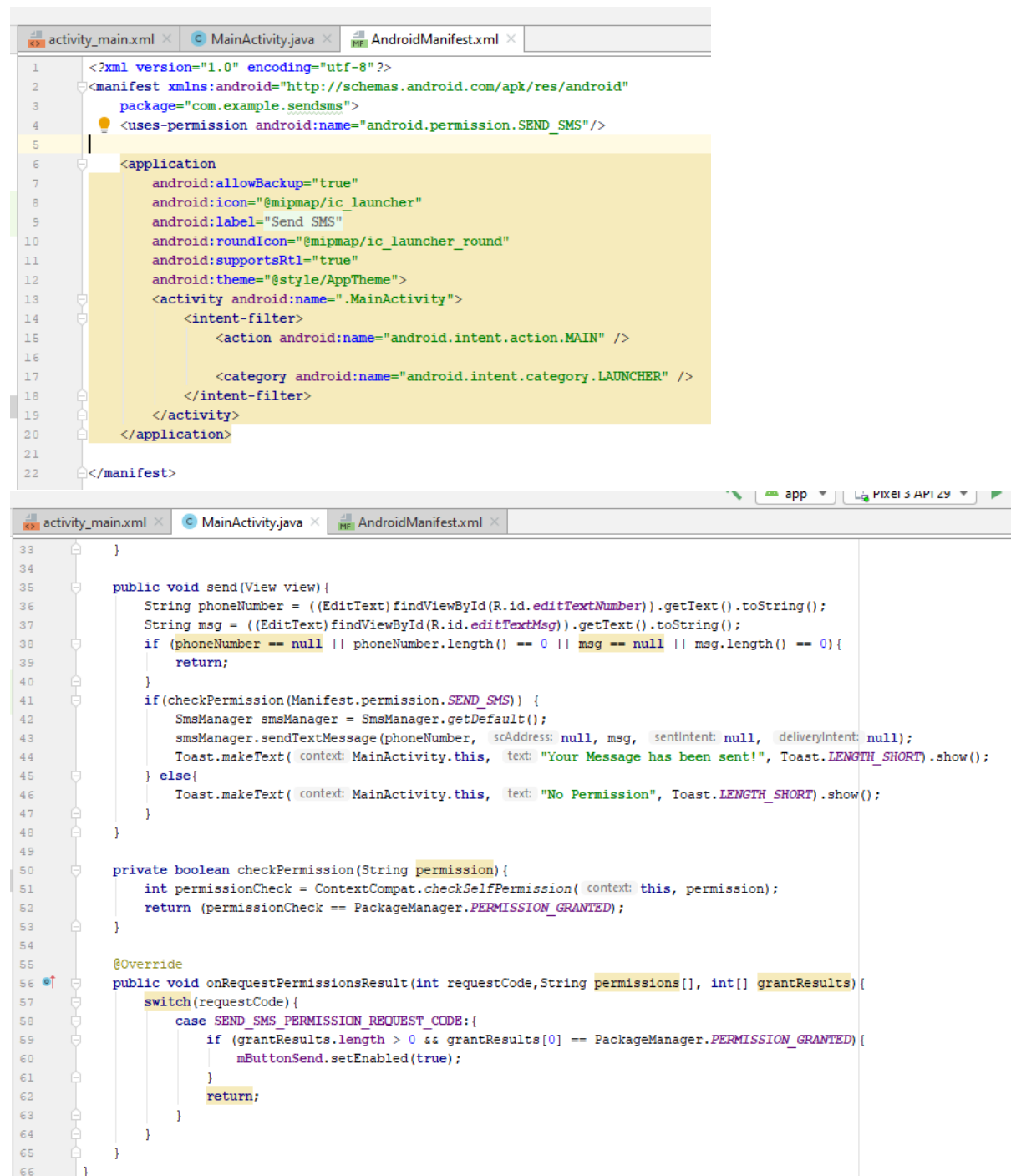
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CALL_PHONE" />
```

The dial button is the clicked and passes a number as a string.

```
public void dialPhone(View view){
    String number = "123456789";
    if (checkPermission("android.permission.CALL_PHONE")==false) {
        Toast toast = Toast.makeText(context, this, text: "You do not have the permissions", Toast.LENGTH_SHORT);
        toast.show();
    }
    else{
        Intent intent = new Intent(Intent.ACTION_CALL);
        intent.setData(Uri.parse("tel: " + number));
        startActivity(intent);
    }
}
```



The SendSMS is more complicated and requires, first of all the permissions in the manifest and second of all, it needs a lot more functionality. Below is the Manifest and MainActivity.



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3  package="com.example.sendsms">
4  <uses-permission android:name="android.permission.SEND_SMS"/>
5
6  <application
7      android:allowBackup="true"
8      android:icon="@mipmap/ic_launcher"
9      android:label="Send SMS"
10     android:roundIcon="@mipmap/ic_launcher_round"
11     android:supportsRtl="true"
12     android:theme="@style/AppTheme">
13     <activity android:name=".MainActivity">
14         <intent-filter>
15             <action android:name="android.intent.action.MAIN" />
16
17             <category android:name="android.intent.category.LAUNCHER" />
18         </intent-filter>
19     </activity>
20 </application>
21
22 </manifest>

```

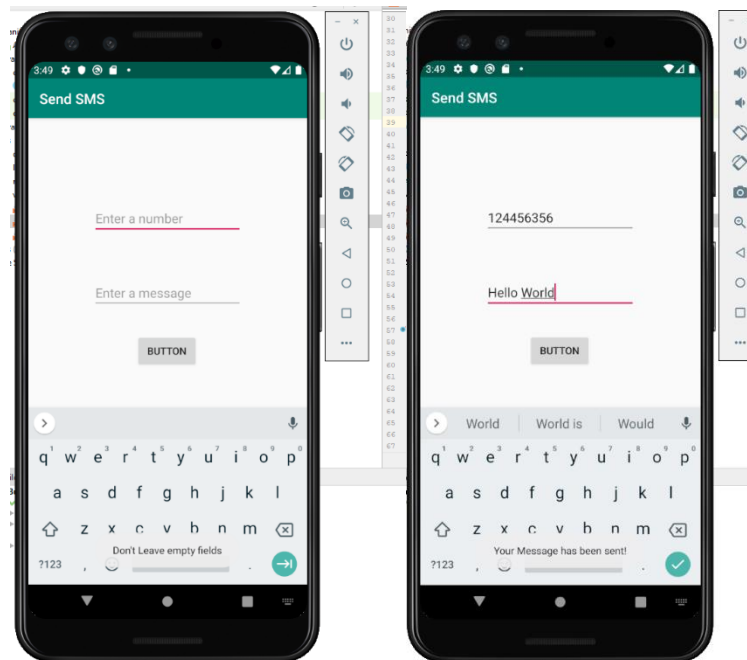
```

33
34
35 public void send(View view) {
36     String phoneNumber = ((EditText)findViewById(R.id.editTextNumber)).getText().toString();
37     String msg = ((EditText)findViewById(R.id.editTextMsg)).getText().toString();
38     if (phoneNumber == null || phoneNumber.length() == 0 || msg == null || msg.length() == 0) {
39         return;
40     }
41     if (checkPermission(Manifest.permission.SEND_SMS)) {
42         SmsManager smsManager = SmsManager.getDefault();
43         smsManager.sendTextMessage(phoneNumber, scAddress: null, msg, sentIntent: null, deliveryIntent: null);
44         Toast.makeText(context: MainActivity.this, text: "Your Message has been sent!", Toast.LENGTH_SHORT).show();
45     } else {
46         Toast.makeText(context: MainActivity.this, text: "No Permission", Toast.LENGTH_SHORT).show();
47     }
48 }
49
50 private boolean checkPermission(String permission) {
51     int permissionCheck = ContextCompat.checkSelfPermission(context: this, permission);
52     return (permissionCheck == PackageManager.PERMISSION_GRANTED);
53 }
54
55 @Override
56 public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
57     switch(requestCode) {
58         case SEND_SMS_PERMISSION_REQUEST_CODE: {
59             if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
60                 mButtonSend.setEnabled(true);
61             }
62             return;
63         }
64     }
65 }
66 }

```

This app has an Edit Text for allowing the user to input a number and an Edit Text for allowing the user to input a message to send.

The app will ask the user if they have enabled the permissions on their device, if they haven't the send button will not be available to the user. It will also not send if either of the fields are empty, this can be seen below on the emulator:

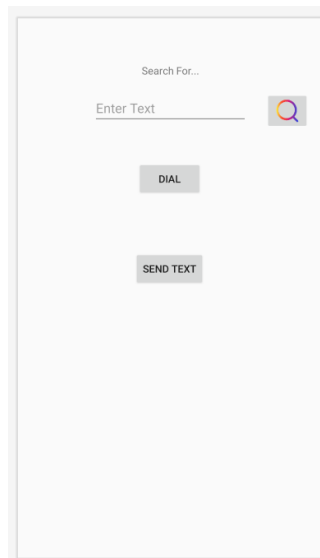


Workbook 1 Additional:

This section requires the developer to combine the elements of the app and create a splashscreen.

As seen below, I have combined all the basic and advanced tasks.

However, there was no room to combine all elements and have the SendSMS working. So, I created a second activity, this is main XML layout:



When the user presses the Send Text button a second activity is created. Using a local Intent to open a new activity a new xml layout is created and the SendSMS functionality from the advanced task is used allow the app to send messages:

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

    Button btntxt = (Button) findViewById(R.id.btn_txt);
    btntxt.setOnClickListener((v) -> {
        Intent i = new Intent( packageContext MainActivity.this, sendsms_activity.class);
        startActivity(i);
    });
}
```

A mobile application form interface. It features a light gray background with a thin blue border. At the top, there is a text input field with the placeholder text "Number". Below it is another text input field with the placeholder text "Message". At the bottom center, there is a gray rectangular button with the text "BUTTON" in white capital letters.

The next part was to create a splashscreen for my app...

COM413 Workbook Practical 2

This Practical allows applications to use databases for storing usernames and passwords for an application...