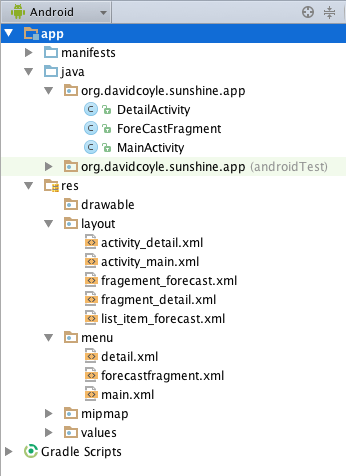
Weather app part 2

## Step 1:

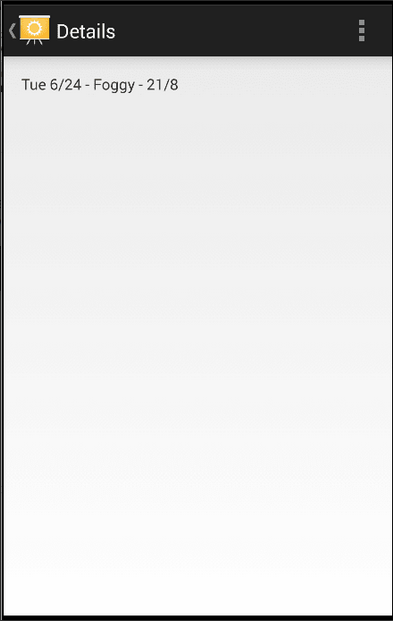
Download “Sunshine tutorial 2 - starter.zip” from the Moodle, expand and open in Android Studio.

This gives you an updated project with:

1. MainActivity.java, with an associated XML layout “activity\_main.xml” and a menu “main.xml”.
2. ForeCastFragment.java, with an associated XML layout “fragement\_forecast.xml” and a “menu forecastfragment.xml”. ForeCastFragment.java now includes an internal class FetchWeatherTask that extends AsyncThread and takes care of the connection to the openweathermap api.
3. An XML layout “list\_item\_forecast.xml”. This is used to display each day/item in the weather forecast list.
4. DetailActivity.java, with the associated XML layout “activity\_detail.xml” and menu “detail.xml”. These files will be used to display a more detailed forecast for an individual day.
5. DetailActivity.java includes an inner class called DetailFragment. This inner class has an associated XML layout “fragment\_detail.xml”.

The files in points 4 and 5 above are new. They were generated by adding a new “blank activity with fragment” to the Sunshine V2 project.

## Step 2

At the moment, when you click on a day/item in the forecast list it generates a Toast.

The aim in this step is to launch a new Activity instead and use this Activity to display a detailed forecast for the day clicked.

* In the file ForeCastFragment.java modify the listView.setOnItemClickListener. Instead of creating a Toast launch the Activity defined in DetailActivity.java and send it the weather information for the day clicked. You can use an explicit Intent to achieved this. See Lecture 4 on Intents for details. You can also check the Android Developers documentation[[1]](#footnote-1).
* Modify the file DetailActivity.java so that it accepts the Intent and displays the weather information in a TextView.
* When you finish run the app. Now when you click on a day in the weather forecast list you should get a view like the one on the right.
* Solution code is included at the end of this document, but please try to do it yourself first.

## Step 3

In this step we are going to add functionality to the Settings button in the ActiveBar.

* Start by creating a new file called SettingsActivity.java.
* Update the manifest file to declare the new SettingsActivity

|  |
| --- |
| <**activity  android:name=".SettingsActivity"  android:label="@string/action\_settings"  android:parentActivityName=".MainActivity"** >  <**meta-data  android:name="android.support.PARENT\_ACTIVITY"  android:value="org.davidcoyle.sunshine.app.MainActivity"** /> </**activity**> |

* Replace the code in this SettingsActivity.java with the following[[2]](#footnote-2):

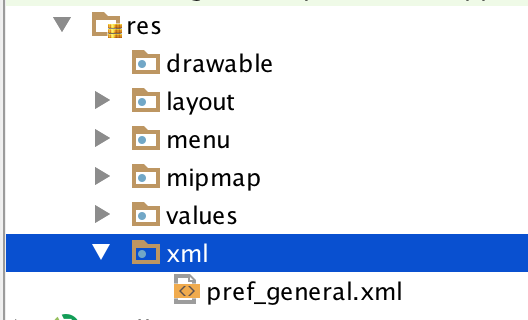
|  |
| --- |
| **package** org.davidcoyle.sunshine.app;  **import** android.os.Bundle; **import** android.preference.ListPreference; **import** android.preference.Preference; **import** android.preference.PreferenceActivity; **import** android.preference.PreferenceManager;  */\*\*  \* A {****@link*** *PreferenceActivity} that presents a set of application settings.  \* <p>  \* See <a href="http://developer.android.com/design/patterns/settings.html">  \* Android Design: Settings</a> for design guidelines and the <a  \* href="http://developer.android.com/guide/topics/ui/settings.html">Settings  \* API Guide</a> for more information on developing a Settings UI.  \*/* **public class** SettingsActivity **extends** PreferenceActivity  **implements** Preference.OnPreferenceChangeListener {   @Override  **public void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  *// Add 'general' preferences, defined in the XML file  //* ***TODO: Add preferences from XML*** *// For all preferences, attach an OnPreferenceChangeListener so the UI summary can be  // updated when the preference changes.  //* ***TODO: Add preferences*** }   */\*\*  \* Attaches a listener so the summary is always updated with the preference value.  \* Also fires the listener once, to initialize the summary (so it shows up before the value  \* is changed.)  \*/* **private void** bindPreferenceSummaryToValue(Preference preference) {  *// Set the listener to watch for value changes.* preference.setOnPreferenceChangeListener(**this**);   *// Trigger the listener immediately with the preference's  // current value.* onPreferenceChange(preference,  PreferenceManager  .*getDefaultSharedPreferences*(preference.getContext())  .getString(preference.getKey(), **""**));  }   @Override  **public boolean** onPreferenceChange(Preference preference, Object value) {  String stringValue = value.toString();   **if** (preference **instanceof** ListPreference) {  *// For list preferences, look up the correct display value in  // the preference's 'entries' list (since they have separate labels/values).* ListPreference listPreference = (ListPreference) preference;  **int** prefIndex = listPreference.findIndexOfValue(stringValue);  **if** (prefIndex >= 0) {  preference.setSummary(listPreference.getEntries()[prefIndex]);  }  } **else** {  *// For other preferences, set the summary to the value's simple string representation.* preference.setSummary(stringValue);  }  **return true**;  }  } |

* Finally, modify MainActivity.java and DetailsActivity.java so that the SettingsActivity is launched when the user click Settings in the ActionBar. Again you can use an explicit Intent to achieved this. The solution code is included at the end of this document, but please try it to do it yourself first.
* When you have made these changes run your app and check what happens when you click on Settings. Your app should show a blank screen, because we have not yet added a UI to the SettingsActivity.

## Step 4

At the moment the Settings screen is blank. In this step you will begin to add preferences that the user can configure. The final layout and flow the settings UI is shown below. This step walks through addition of the Location preference. In Step 5 you will add a Units preference yourself.



* Begin by creating a new resource folder and XML file for your preferences. Do this by adding a new folder called “xml” inside the res folder. In this new folder create a new XML file called “pref\_general.xml”.
* Copy the XML code below into “pref\_general.xml” file.

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"**>   <**EditTextPreference  android:title="@string/pref\_location\_label"  android:key="@string/pref\_location\_key"  android:defaultValue="@string/pref\_location\_default"  android:inputType="text"  android:singleLine="true"** />  </**PreferenceScreen**> |

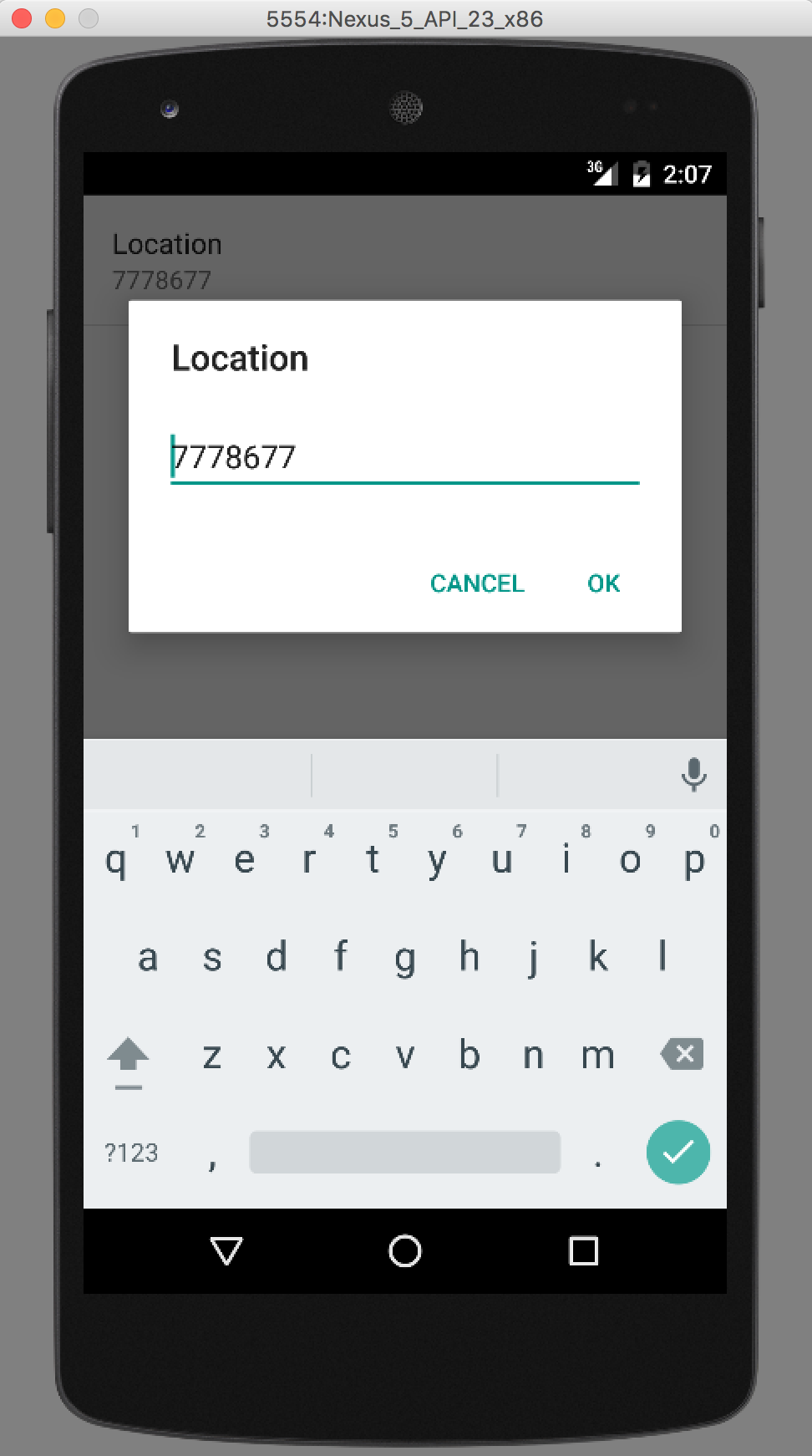
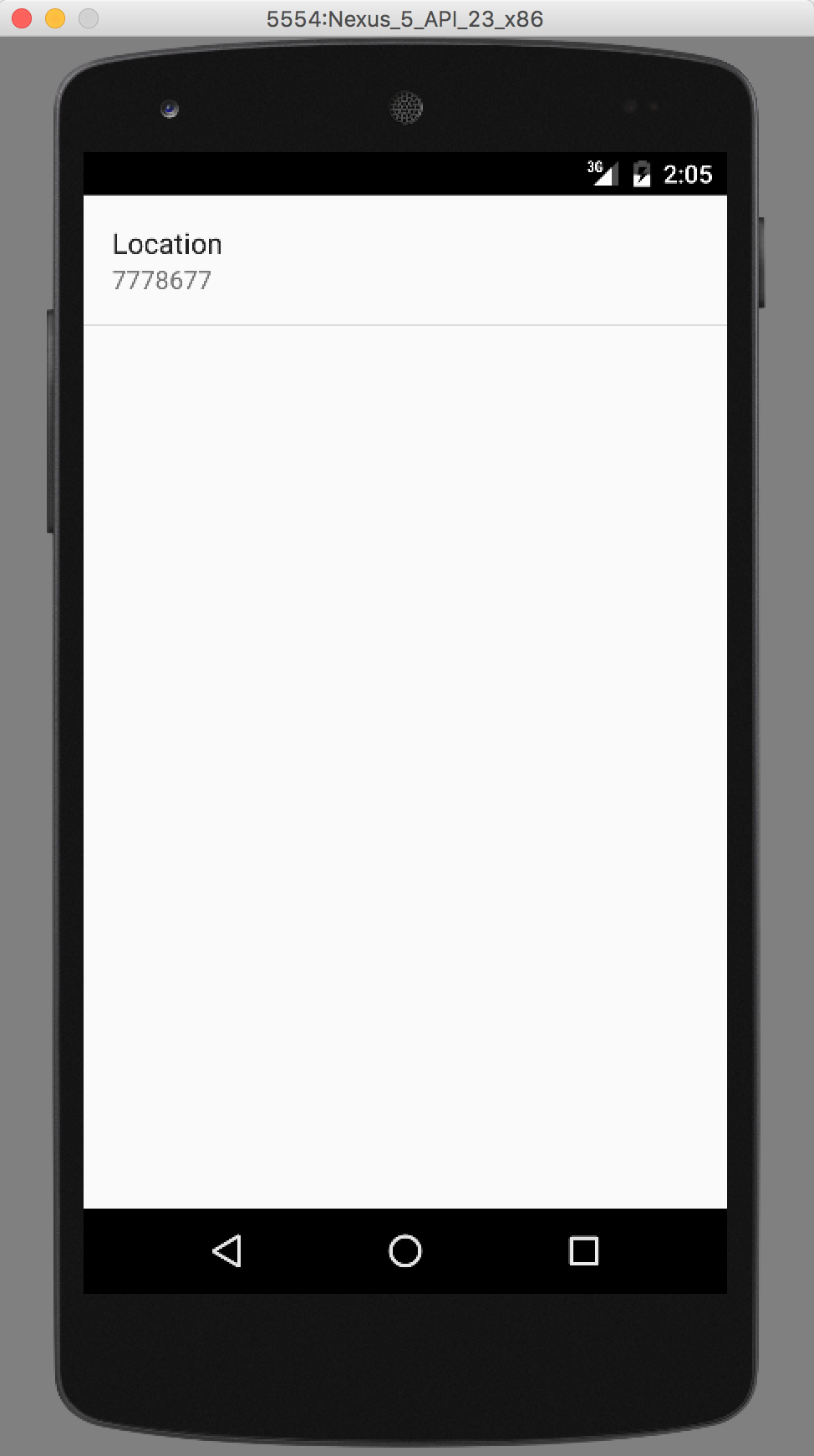
This code defines a new PreferenceScreen that includes a single EditTextPreference element. It also makes uses of some new String resources that you need to add to the strings.xml file. Your new strings.xml file should look something like this.

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**resources**>   *<!-- Used in Action Bar, and in AndroidManifest to tell the device the name of this app.  It's best to keep this short, so your launcher icon isn't displayed with "The greatest Wea"  or something similar. -->* <**string name="app\_name"**>Sunshine v2</**string**>   *<!-- By convention, "action" denotes that this String will be used as the label for an Action,  typically from the action bar. The ActionBar is limited real estate, so shorter is better. -->* <**string name="action\_settings"**>Settings</**string**>   *<!-- Menu label to fetch updated weather info from the server -->* <**string name="action\_refresh" translatable="false"**>Refresh</**string**>  <**string name="title\_activity\_detail"**>Details</**string**>  <**string name="hello\_world"**>Hello world!</**string**>   *<!-- Label for the location preference [CHAR LIMIT=30] -->* <**string name="pref\_location\_label"**>Location</**string**>   *<!-- Key name for storing location in SharedPreferences [CHAR LIMIT=NONE] -->* <**string name="pref\_location\_key" translatable="false"**>location</**string**>   *<!-- Default postal code for location preference [CHAR LIMIT=NONE] -->* <**string name="pref\_location\_default" translatable="false"**>7778677</**string**>  </**resources**> |

Finally, update the onCreate() method in SettingsActivity.java as follows:

|  |
| --- |
| @Override **public void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  *// Add 'general' preferences, defined in the XML file* addPreferencesFromResource(R.xml.***pref\_general***);   *// For all preferences, attach an OnPreferenceChangeListener so the UI summary can be  // updated when the preference changes.* bindPreferenceSummaryToValue(findPreference(getString(R.string.***pref\_location\_key***)));  } |

After you have made these changes run you app and test the new settings screen, which should look like image 1 below. When you click on the Location preference it should open a dialogue as shown in image 2, where you can edit the location.



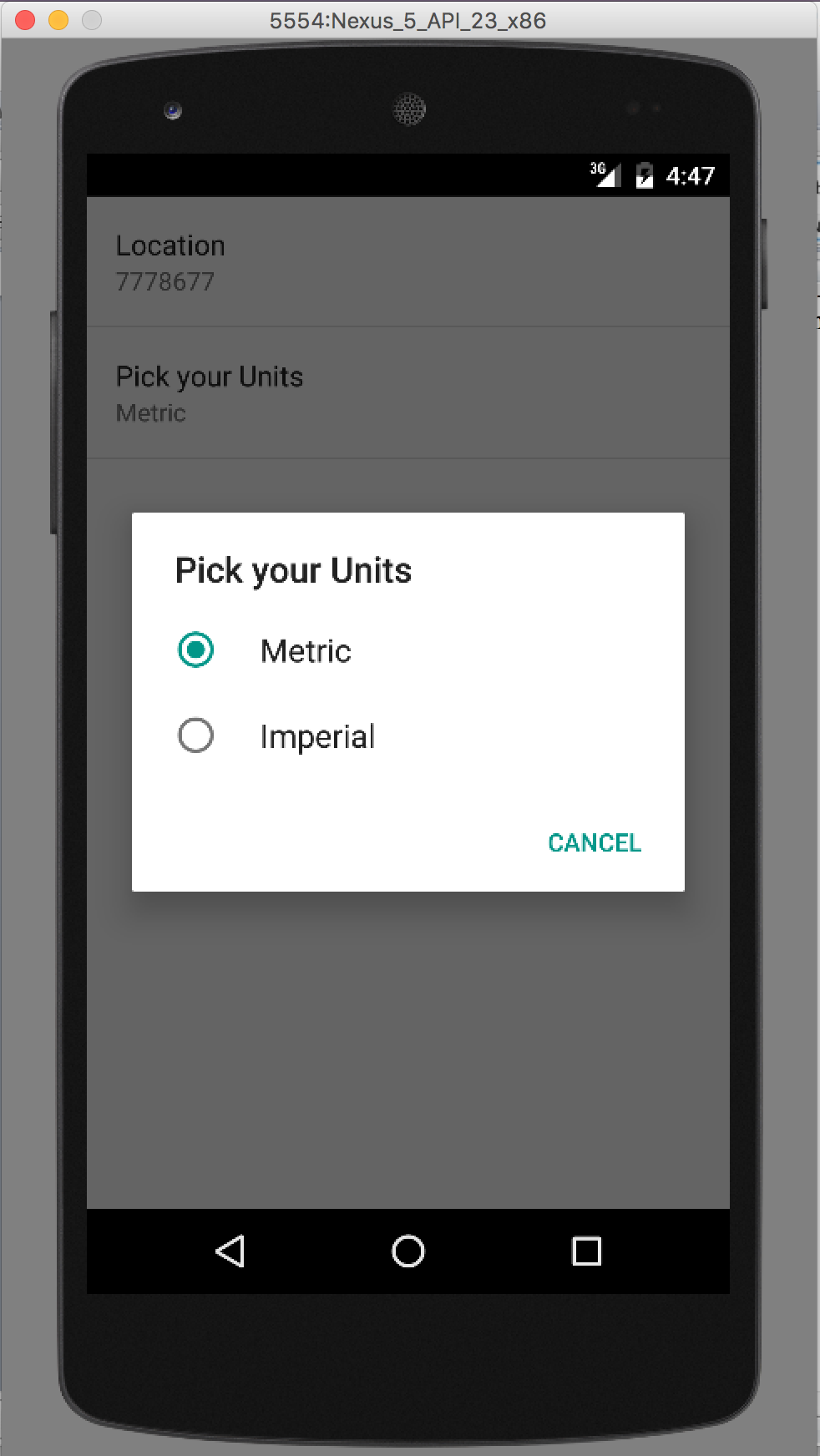
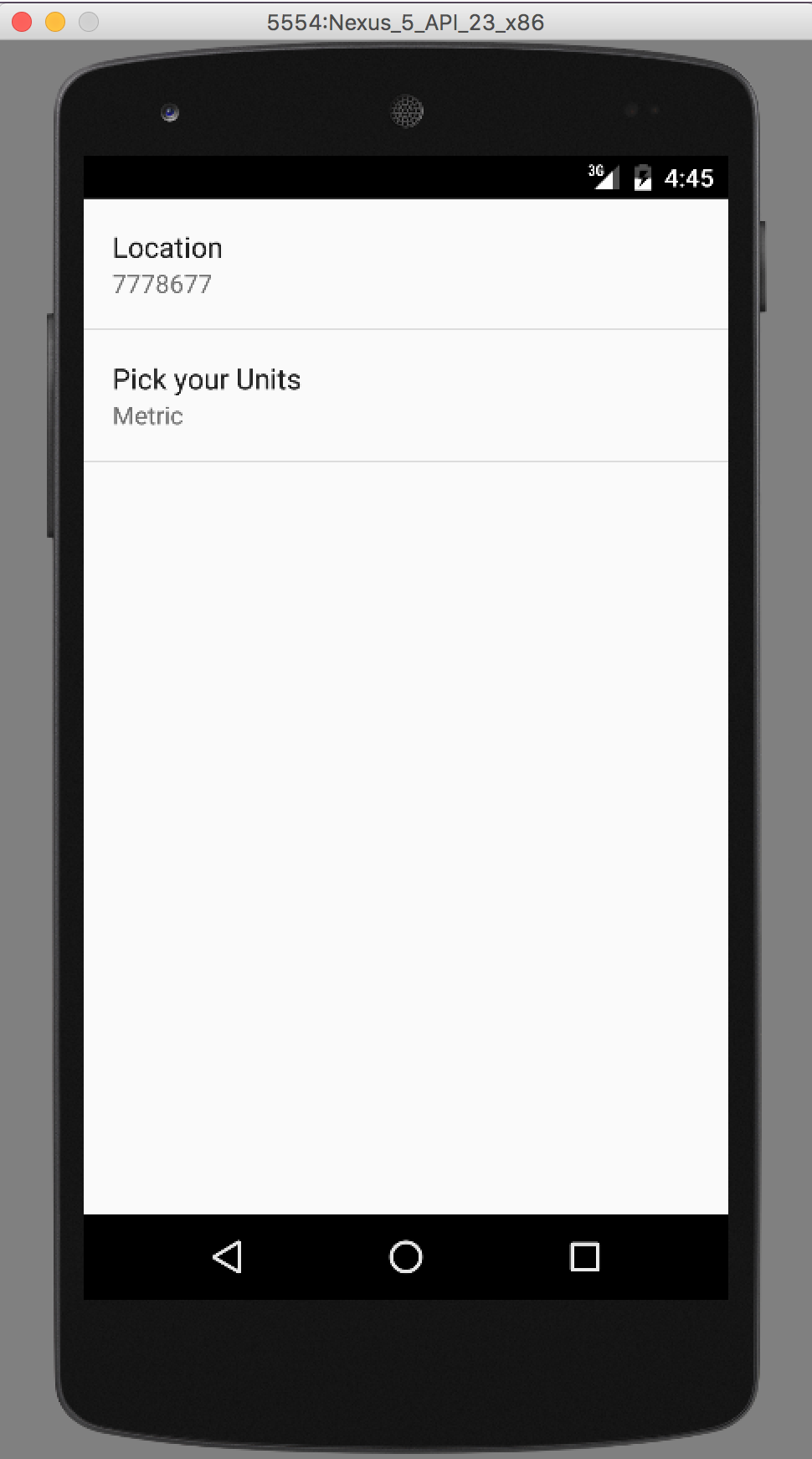
## Step 5

Here you are going to create a Units preference to the Settings screen. This will allow users to choose whether forecast data is displayed using metric or imperial units. When you have completed this step your settings screen should look like image 1 below. When the user presses “Pick your Units” a dialogue similar to that shown in image 2 should appear.

**Clue:** you will need to add a ListPreference to “pref\_general.xml” and also modify “strings.xml” and “SettingsActivity.java”.

See here[[3]](#footnote-3) for details of ListPreference. You will also need to declare some StringArrays[[4]](#footnote-4).

Again, the solution code for this step is included at the end of this document, but try it yourself first.



## Step 6

Finally make the necessary changes to your app such that the weather forecasts on the details and overview screens are updated based on changes users make to their preferences, e.g. changes to their location.

No solution is provided for this step, so please try this yourself.

Solutions

## Step 2 code:

First modify the onItemClick() method in ForeCastFragment.java. This method is inside the OnItemClickListener with the onCreate() method.

|  |
| --- |
| **public void** onItemClick(AdapterView<?> adapterView, View view, **int** position, **long** l) {  CharSequence forecast = **mForecastAdapter**.getItem(position);  Intent newIntent = **new** Intent(getActivity(), DetailActivity.**class**);  newIntent.putExtra(Intent.***EXTRA\_TEXT***, forecast);  startActivity(newIntent);  } |

Next modify the onCreateView() method in the DetailFragment class within DetailActivity.java.

|  |
| --- |
| **public** View onCreateView(LayoutInflater inflater, ViewGroup container,  Bundle savedInstanceState) {   View rootView = inflater.inflate(R.layout.***fragment\_detail***, container, **false**);  *// The detail Activity called via intent. Inspect the intent for forecast data.* Intent intent = getActivity().getIntent();  **if** (intent != **null** && intent.hasExtra(Intent.***EXTRA\_TEXT***)) {  String forecastStr = intent.getStringExtra(Intent.***EXTRA\_TEXT***);  ((TextView) rootView.findViewById(R.id.***detail\_text***)).setText(forecastStr);  }   **return** rootView; } |

## Step 3 code

For the final part of step three modify the onOptionsItemSelected() method in both “MainActivity.java” and “DetailActivity.java” to include the following:

|  |
| --- |
| **int** id = item.getItemId();  *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {   Intent settingsIntent = **new** Intent(**this**, SettingsActivity.**class**);  startActivity(settingsIntent);  **return true**; }  **return super**.onOptionsItemSelected(item); |

When the settings button has been pressed this code creates a new Intent to launch SettingsActivity and starts the Actitity.

## Step 5

Add the following code to “pref\_general.xml”

|  |
| --- |
| <**ListPreference  android:title="@string/pref\_units\_label"  android:key="@string/pref\_units\_key"  android:dialogTitle="@string/pref\_units\_dialog\_title"  android:entries="@array/pref\_units\_options"  android:entryValues="@array/pref\_units\_values"  android:defaultValue="@string/pref\_units\_metric"** /> |

Add the following to strings.xml

|  |
| --- |
| *<!-- Label for the units preference [CHAR LIMIT=30] -->* <**string name="pref\_units\_label"**>Pick your Units</**string**>  *<!-- Label for the units dialogue box [CHAR LIMIT=30] -->* <**string name="pref\_units\_dialog\_title"**>Pick your Units</**string**>  *<!-- Key name for storing units in SharedPreferences [CHAR LIMIT=NONE] -->* <**string name="pref\_units\_key" translatable="false"**>units</**string**>  *<!-- Default unit preference [CHAR LIMIT=NONE] -->* <**string name="pref\_units\_metric" translatable="false"**>Metric</**string**>  <**string-array  name="pref\_units\_options"**>  <**item**>Metric</**item**>  <**item**>Imperial</**item**> </**string-array**>  <**string-array  name="pref\_units\_values"**>  <**item**>true</**item**>  <**item**>false</**item**> </**string-array**> |

Finally add the following line to the onCreate() method in SettingsActivity.java.

|  |
| --- |
| bindPreferenceSummaryToValue(findPreference(getString(R.string.***pref\_units\_key***))); |

## Step 6

You’re on your own here, for the time being!

1. <https://developer.android.com/guide/components/intents-filters.html> [↑](#footnote-ref-1)
2. See here for more detail on dealing with settings <https://developer.android.com/guide/topics/ui/settings.html> [↑](#footnote-ref-2)
3. <https://developer.android.com/guide/topics/ui/settings.html> [↑](#footnote-ref-3)
4. <https://developer.android.com/guide/topics/resources/string-resource.html#StringArray> [↑](#footnote-ref-4)