Unit-3: Android Manifest File and its common settings

3

Unit Structure

3.0 Learning Objectives

1.3 Introduction

1.4 Components of Manifest file

1.5 Package name and application ID

1.6 App components

1.7 Permissions

1.8 Device compatibility

1.9 File conventions

1.10 Manifest elements reference

1.11 Example of Manifest file

1.12 Let us sum up

1.13 Check your Progress: Possible Answers

1.14 Further Reading

1.15 Assignment

1.16 Activities

3.0 Learning Objectives

After studying this unit students should be able to

- What is AndroidManifest.xml file
- Understand various components of Manifest file
- Specify package, activity, permission, device configuration in Manifest file
- Understand File convention used in Manifest file
- Modify the default Manifest file as per requirement

3.1 Introduction

Each and every android app project must have an AndroidManifest.xml file in the root of your project. The manifest file describes important information about your app. The manifest file declares the following:

- The app's package name
- The components of the app such as activities, services, broadcast receivers, and content providers.
- Which device configurations it can handle
- Intent filters that describe how the component can be started.
- Permissions required by the app
- The hardware and software features the app requires

Android Studio generally builds the manifest file for you when you create a project. For a simple application with a single activity and nothing else, the auto-generated manifest will work fine with little or no modifications.

3.2 Component of Manifest file

Android manifest file is global application description file which defines your application’s capabilities and permissions and how it runs. This topic describes some of the most important characteristics of your app which is stored in the manifest file.

3.3 Package name and application ID

The manifest file's root element requires an attribute for your app's package name, For example, the following snippet shows the root <manifest> element with the package name "in.edu.baou.databasedemo":

```
<? xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="in.edu.baou.databasedemo"
   android:versionCode="1"
   android:versionName="1.0" >
   ...
</manifest>
```

While building your app into the final APK, the Android build tools use the package attribute for two things:

It applies this name as the namespace for your app's generated R.java class. With the above manifest, the R class is created at in.edu.baou.databasedemo.R.

Android manifest file uses package this name to resolve any relative class names that are declared in the manifest file.

If, an activity declared as <activity android:name=".MainActivity"> is resolved to be in.edu.baou.databasedemo.MainActivity.

You should keep in mind that once the APK is compiled, the package attribute also represents your app's universally unique application ID. After the build tools perform the above tasks based on the package name, they replace the package value with the value given to the applicationId property in your project's build.gradle file.

3.4 App Components

For each app component that you create in your app, you must declare a corresponding XML element in the manifest file so that the system can start it.

For each subclass of Activity, we have <activity>
For each subclass of Service, we have <service>
For each subclass of BroadcastReceiver we have <receiver>.
For each subclass of ContentProvider, we have <provider>

The name of your subclass must be specified with the name attribute, using the full package designation, e.g. an Activity subclass can be declared as follows

```
<manifest package=" in.edu.baou.databasedemo" ... >
   <application ... >
      <activity android:name=".SQLiteDBActivity " ... >
         ...
      </activity>
   </application>
</manifest>
```

In above example, the activity name is resolved to
"in.edu.baou.databasedemo.SQLiteDBActivity "

App activities, services, and broadcast receivers are activated by intents. It is an asynchronous messaging mechanism to match task requests with the appropriate Activity.

When an app issues intent to the system, the system locates an app component that can handle the intent based on intent filter declarations in each app's manifest file. The system launches an instance of the matching component and passes the Intent object to that component. If more than one app can handle the intent, then the user can select which app to use. An intent filters is defined with the <intent-filter> element as shown below.

```
<activity
        android:name=".SQLiteDBActivity" >
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
```

A number of manifest elements have icon and label attributes for displaying a small icon and a text label, respectively, to users for the corresponding app component.

For example, the icon and label that are set in the <application> element are the default icon and label for each of the app's components.

The icon and label that are set in a component's <intent-filter> are shown to the user whenever that component is presented as an option to fulfill intent.

3.5 Permissions

Android apps must request permission to access personnel user data such as contacts, SMS, camera, files, internet etc. Each permission is identified by a unique label. For example, an app that needs to send and receive SMS messages must have the following line in the manifest:

```
<manifest ... >
   <uses-permission android:name="android.permission.SEND_SMS" />
   <uses-permission android:name="android.permission.RECEIVE_SMS" />
   ...
</manifest>
```

From API level 23, the user can approve or reject some app permissions at runtime. You must declare all permission requests with a <uses-permission> element in the manifest. If the permission is granted, the app is able to use the protected features. If

not, its attempts to access those features fail. A new permission is declared with the <permission> element.

3.6 Device Compatibility

In manifest file is you can declare what types of hardware or software features your app requires and types of devices with which your app is compatible. It can't be installed on devices that don't provide the features or system version that your app requires. The following table shows the most common tags for specifying device compatibility.

Tag	Description
<uses-feature>	It allows you to declare hardware and software features your app needs Example <manifest ... > <uses-feature android:name="android.hardware.sensor.compass" android:required="true" /> ... </manifest>
<uses-sdk>	It indicates the minimum version with which your app is compatible element are overridden by corresponding properties in the build.gradle file. <manifest> <uses-sdk android:minSdkVersion="5" /> ... </manifest>

Table-9

3.7 File conventions

Following are the conventions and rules that generally apply to all elements and attributes in the manifest file.

- Only the <manifest> and <application> elements are required. They each must occur only once, other elements can occur zero or more times.
- Elements at the same level are generally not ordered hence elements can be placed in any order
- All attributes are optional but attributes must be specified so that an element can serve its purpose. If attributes are not provided then it indicates the default value
- Except for some attributes of the root <manifest> element, all attribute names begin with an android: prefix.

3.8 Manifest elements reference

The following table provides links to reference documents for all valid elements in the AndroidManifest.xml file.

Element	Description
<action>	It is used to add an action to an intent filter.
<activity>	It is used to declare an activity component.
<activity-alias>	It is used to declare an alias for an activity.
<application>	It is used to declare the application.
<category>	It is used to add category name to an intent filter.
<compatible-screens>	It is used to specifies each screen configuration with which the application is compatible.
<data>	Adds a data specification to an intent filter.
<grant-uri-permission>	Specifies the subsets of app data that the parent content provider has permission to access.
<instrumentation>	Declares an Instrumentation class that enables you to monitor an application's interaction with the system.
<intent-filter>	Specifies the types of intents that an activity, service, or broadcast receiver can respond to.
<manifest>	The root element of the AndroidManifest.xml file.
<meta-data>	A name-value pair for an item of additional, arbitrary data that can be supplied to the parent component.
<path-permission>	Defines the path and required permissions for a specific subset of data within a content provider.
<permission>	Declares a security permission that can be used to limit access to specific components or features of this or other applications.

<permission-group>	Declares a name for a logical grouping of related permissions.
<permission-tree>	Declares the base name for a tree of permissions.
<provider>	Declares a content provider component.
<receiver>	Declares a broadcast receiver component.
<service>	Declares a service component.
<supports-gl-texture>	Declares a single GL texture compression format that the app supports.
<supports-screens>	Declares the screen sizes your app supports and enables screen compatibility mode for screens larger than what your app supports.
<uses-configuration>	Indicates specific input features the application requires.
<uses-feature>	Declares a single hardware or software feature that is used by the application.
<uses-library>	Specifies a shared library that the application must be linked against.
<uses-permission>	Specifies a system permission that the user must grant in order for the app to operate correctly.
<uses-sdk>	Lets you express an application's compatibility with one or more versions of the Android platform, by means of an API level integer.

Table-10

3.9 Example of Manifest file

The XML below is a simple example AndroidManifest.xml that declares two activities for the app.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="in.edu.baou.listnameactivity"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="22" />
    <application
```

```
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="in.edu.baou.listnameactivity.NameDisplayActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="in.edu.baou.listnameactivity.MultipleChoiceActivity"
            android:label="@string/title_activity_multiple_choice" >
        </activity>
    </application>
</manifest>
```

Check your progress-1

a) Android Studio generally builds the manifest file for you when you create a project. (True/False)

b) <uses-feature> indicates specific input features the application requires. (True/False)

c) _____ declares a security permission that can be used to limit access to specific components or features of this or other applications.

d) In manifest file is you can declare _____ features your app requires

(i) Hardware feature (ii) Software feature

(iii) Device compatibility (iv) All of these

e) In Manifest file _____ tag is compulsory

(i) <manifest> (ii) <application>

(iii) Both (i) and (ii) (iv) None of these

3.10 Let us sum up

The AndroidManifest.xml file contains important information regarding package, and components of the application such as activities, services, broadcast receivers, content providers etc. It performs some other tasks such as to protect the application to access any protected parts by providing the permissions. It also declares the android api that the application is going to use. It lists the instrumentation classes. The instrumentation classes provide profiling and other information. This information is removed just before the application is published etc. This is the required xml file for all the android application and located inside the root directory.

Each Android application has a specially formatted XML file called AndroidManifest.xml.This file describes the application's identity in great detail. Some information you must define within the Android manifest file includes the application's name and version information, what application components it contains, which device configurations it requires, and what permissions it needs to run.The Android manifest file is used by the Android operating system to install, upgrade, and run the application package.

3.11 Check your Progress: Possible Answers

1-a) True

1-b) False

1-c) <permission>

1-d) (iv) All of these

1-e) (iii) Both (i) and (ii)

3.12 Further Reading

- https://developer.android.com/guide/topics/manifest/manifest-intro
- http://www.androiddocs.com/guide/topics/manifest/manifest-intro.html
- https://www.javatpoint.com/AndroidManifest-xml-file-in-android

3.13 Assignment

- Write detailed note on AndroidManifest.xml file
- How can we specify permission in Android Manifest file?
- Explain with example how can we specify device compatibility in Android Manifest file.

3.14 Activities

- Create Android Studio Project and study the default AndroidManifest.xml file created for you and try to modified as per your requirement