App Components

**Activity**

1. An [Activity](http://developer.android.com/reference/android/app/Activity.html) represents a single screen in an app. You can start a new instance of an [Activity](http://developer.android.com/reference/android/app/Activity.html) by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to[startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)). The [Intent](http://developer.android.com/reference/android/content/Intent.html) describes the activity to start and carries any necessary data.
2. If you want to receive a result from the activity when it finishes, call [startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult(android.content.Intent, int)). Your activity receives the result as a separate [Intent](http://developer.android.com/reference/android/content/Intent.html) object in your activity's[onActivityResult()](http://developer.android.com/reference/android/app/Activity.html#onActivityResult(int, int, android.content.Intent)) callback. For more information, see the[Activities](http://developer.android.com/guide/components/activities.html) guide.

**Service**

1. A [Service](http://developer.android.com/reference/android/app/Service.html) is a component that performs operations in the background without a user interface.
2. A [Service](http://developer.android.com/reference/android/app/Service.html) is a component that performs operations in the background without a user interface. You can start a service to perform a one-time operation (such as download a file) by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to [startService()](http://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)). The [Intent](http://developer.android.com/reference/android/content/Intent.html) describes the service to start and carries any necessary data.
3. If the service is designed with a client-server interface, you can bind to the service from another component by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent, android.content.ServiceConnection, int)).

**Content** **Provider**

**Broadcast** **Reciever**

1. A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to [sendBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent)), [sendOrderedBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendOrderedBroadcast(android.content.Intent, java.lang.String)), or [sendStickyBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendStickyBroadcast(android.content.Intent)).

**Intent**

1. An implicit intent simply describes the type of action to perform (and, optionally, the data upon which you’d like to perform the action) and allows the system to find a component on the device that can perform the action and start it. If there are multiple components that can perform the action described by the intent, then the user selects which one to use.
2. An [Intent](http://developer.android.com/reference/android/content/Intent.html) is a messaging object you can use to request an action from another [app component](http://developer.android.com/guide/components/fundamentals.html#Components).
3. An [Activity](http://developer.android.com/reference/android/app/Activity.html) represents a single screen in an app. You can start a new instance of an [Activity](http://developer.android.com/reference/android/app/Activity.html) by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to[startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)). The [Intent](http://developer.android.com/reference/android/content/Intent.html) describes the activity to start and carries any necessary data.
4. If you want to receive a result from the activity when it finishes, call [startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult(android.content.Intent, int)). Your activity receives the result as a separate [Intent](http://developer.android.com/reference/android/content/Intent.html) object in your activity's[onActivityResult()](http://developer.android.com/reference/android/app/Activity.html#onActivityResult(int, int, android.content.Intent)) callback. For more information, see the[Activities](http://developer.android.com/guide/components/activities.html) guide.
5. A [Service](http://developer.android.com/reference/android/app/Service.html) is a component that performs operations in the background without a user interface. You can start a service to perform a one-time operation (such as download a file) by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to [startService()](http://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)). The [Intent](http://developer.android.com/reference/android/content/Intent.html) describes the service to start and carries any necessary data.
6. If the service is designed with a client-server interface, you can bind to the service from another component by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent, android.content.ServiceConnection, int)).
7. A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an [Intent](http://developer.android.com/reference/android/content/Intent.html) to [sendBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent)), [sendOrderedBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendOrderedBroadcast(android.content.Intent, java.lang.String)), or [sendStickyBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendStickyBroadcast(android.content.Intent)).
8. **Explicit intents** specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, start a new activity in response to a user action or start a service to download a file in the background.
9. **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.
10. When you create an explicit intent to start an activity or service, the system immediately starts the app component specified in the [Intent](http://developer.android.com/reference/android/content/Intent.html) object.
11. When you create an implicit intent, the Android system finds the appropriate component to start by comparing the contents of the intent to the intent filters declared in the [manifest file](http://developer.android.com/guide/topics/manifest/manifest-intro.html) of other apps on the device. If the intent matches an intent filter, the system starts that component and delivers it the[Intent](http://developer.android.com/reference/android/content/Intent.html) object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.
12. An intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive. For instance, by declaring an intent filter for an activity, you make it possible for other apps to directly start your activity with a certain kind of intent. Likewise, if you do notdeclare any intent filters for an activity, then it can be started only with an explicit intent.
13. **Caution:** To ensure your app is secure, always use an explicit intent when starting a [Service](http://developer.android.com/reference/android/app/Service.html) and do not declare intent filters for your services. Using an implicit intent to start a service is a security hazard because you cannot be certain what service will respond to the intent, and the user cannot see which service starts. Beginning with Android 5.0 (API level 21), the system throws an exception if you call [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent, android.content.ServiceConnection, int)) with an implicit intent.
14. Parameters for intent object
    1. **Component name**
       1. The name of the component to start.
       2. This is optional, but it's the critical piece of information that makes an intent **explicit**, meaning that the intent should be delivered only to the app component defined by the component name.
       3. Without a component name, the intent is **implicit** and the system decides which component should receive the intent based on the other intent information (such as the action, data, and category—described below).
       4. This field of the [Intent](http://developer.android.com/reference/android/content/Intent.html) is a [ComponentName](http://developer.android.com/reference/android/content/ComponentName.html) object, which you can specify using a fully qualified class name of the target component, including the package name of the app. For example,com.example.ExampleActivity. You can set the component name with [setComponent()](http://developer.android.com/reference/android/content/Intent.html#setComponent(android.content.ComponentName)), [setClass()](http://developer.android.com/reference/android/content/Intent.html#setClass(android.content.Context, java.lang.Class<?>)),[setClassName()](http://developer.android.com/reference/android/content/Intent.html#setClassName(java.lang.String, java.lang.String)), or with the [Intent](http://developer.android.com/reference/android/content/Intent.html) constructor.
    2. Action
       1. A string that specifies the generic action to perform (such as view or pick).
       2. you should usually use action constants defined by the [Intent](http://developer.android.com/reference/android/content/Intent.html) class or other framework classes.
       3. [ACTION\_VIEW](http://developer.android.com/reference/android/content/Intent.html#ACTION_VIEW) Use this action in an intent with [startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)) when you have some information that an activity can show to the user, such as a photo to view in a gallery app, or an address to view in a map app.
       4. [ACTION\_SEND](http://developer.android.com/reference/android/content/Intent.html#ACTION_SEND) Also known as the "share" intent, you should use this in an intent with [startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)) when you have some data that the user can share through another app, such as an email app or social sharing app.
       5. You can specify the action for an intent with [setAction()](http://developer.android.com/reference/android/content/Intent.html#setAction(java.lang.String)) or with an [Intent](http://developer.android.com/reference/android/content/Intent.html) constructor.

If you define your own actions, be sure to include your app's package name as a prefix. For example: static final String ACTION\_TIMETRAVEL = "com.example.action.TIMETRAVEL";

* 1. **Data**
     1. The URI (a [Uri](http://developer.android.com/reference/android/net/Uri.html) object) that references the data to be acted on and/or the MIME type of that data.
     2. The type of data supplied is generally dictated by the intent's action. For example, if the action is[ACTION\_EDIT](http://developer.android.com/reference/android/content/Intent.html#ACTION_EDIT), the data should contain the URI of the document to edit.
     3. When creating an intent, it's often important to specify the type of data (its MIME type) in addition to its URI.
  2. **Category**
     1. A string containing additional information about the kind of component that should handle the intent. Any number of category descriptions can be placed in an intent, but most intents do not require a category.
     2. [CATEGORY\_BROWSABLE](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_BROWSABLE) The target activity allows itself to be started by a web browser to display data referenced by a link—such as an image or an e-mail message.
     3. [CATEGORY\_LAUNCHER](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_LAUNCHER) The activity is the initial activity of a task and is listed in the system's application launcher.
  3. **Extras**
     1. Key-value pairs that carry additional information required to accomplish the requested action. Just as some actions use particular kinds of data URIs, some actions also use particular extras.
     2. You can add extra data with various [putExtra()](http://developer.android.com/reference/android/content/Intent.html#putExtra(java.lang.String, android.os.Bundle)) methods, each accepting two parameters: the key name and the value. You can also create a [Bundle](http://developer.android.com/reference/android/os/Bundle.html) object with all the extra data, then insert the [Bundle](http://developer.android.com/reference/android/os/Bundle.html) in the[Intent](http://developer.android.com/reference/android/content/Intent.html) with [putExtras()](http://developer.android.com/reference/android/content/Intent.html#putExtras(android.content.Intent)).
     3. For example, when creating an intent to send an email with [ACTION\_SEND](http://developer.android.com/reference/android/content/Intent.html#ACTION_SEND), you can specify the "to" recipient with the [EXTRA\_EMAIL](http://developer.android.com/reference/android/content/Intent.html#EXTRA_EMAIL) key, and specify the "subject" with the [EXTRA\_SUBJECT](http://developer.android.com/reference/android/content/Intent.html#EXTRA_SUBJECT) key.

The [Intent](http://developer.android.com/reference/android/content/Intent.html) class specifies many EXTRA\_\* constants for standardized data types. If you need to declare your own extra keys (for intents that your app receives), be sure to include your app's package name as a prefix. For example: static final String EXTRA\_GIGAWATTS = "com.example.EXTRA\_GIGAWATTS";

* 1. **Flags**
     1. Flags defined in the [Intent](http://developer.android.com/reference/android/content/Intent.html) class that function as metadata for the intent. The flags may instruct the Android system how to launch an activity (for example, which [task](http://developer.android.com/guide/components/tasks-and-back-stack.html) the activity should belong to) and how to treat it after it's launched (for example, whether it belongs in the list of recent activities).
     2. see the [setFlags()](http://developer.android.com/reference/android/content/Intent.html#setFlags(int)) method.

1. These properties listed above (component name, action, data, and category) represent the defining characteristics of an intent. By reading these properties, the Android system is able to resolve which app component it should start.

**Using** **Different hardware** **or software** **feature**

1. In order for you to manage your app’s availability based on device features, Android defines feature IDs for any hardware or software feature that may not be available on all devices. For instance, the feature ID for the compass sensor is [FEATURE\_SENSOR\_COMPASS](http://developer.android.com/reference/android/content/pm/PackageManager.html#FEATURE_SENSOR_COMPASS) and the feature ID for app widgets is [FEATURE\_APP\_WIDGETS](http://developer.android.com/reference/android/content/pm/PackageManager.html#FEATURE_APP_WIDGETS).
2. If necessary, you can prevent users from installing your app when their devices don't provide a given feature by declaring it with a [<uses-feature>](http://developer.android.com/guide/topics/manifest/uses-feature-element.html) element in your app's [manifest file](http://developer.android.com/guide/topics/manifest/manifest-intro.html).
3. <manifest ... >  
       <uses-feature android:name="android.hardware.sensor.compass"  
                     android:required="true" />  
       ...  
   </manifest>

**Exceptional** **points**

1. Activities, services, and content providers that you include in your source but do not declare in the manifest are not visible to the system and, consequently, can never run. However, broadcast receivers can be either declared in the manifest or created dynamically in code (as [BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html) objects) and registered with the system by calling [registerReceiver()](http://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter)).
2. For every resource that you include in your Android project, the SDK build tools define a unique integer ID, which you can use to reference the resource from your app code or from other resources defined in XML. For example, if your app contains an image file named logo.png (saved in the res/drawable/ directory), the SDK tools generate a resource ID named R.drawable.logo, which you can use to reference the image and insert it in your user interface.
3. **Caution:** To ensure your app is secure, always use an explicit intent when starting a [Service](http://developer.android.com/reference/android/app/Service.html) and do not declare intent filters for your services. Using an implicit intent to start a service is a security hazard because you cannot be certain what service will respond to the intent, and the user cannot see which service starts. Beginning with Android 5.0 (API level 21), the system throws an exception if you call [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent, android.content.ServiceConnection, int)) with an implicit intent.
4. **Note:** When starting a [Service](http://developer.android.com/reference/android/app/Service.html), you should **always specify the component name**. Otherwise, you cannot be certain what service will respond to the intent, and the user cannot see which service starts.
5. **Caution:** It's possible that a user won't have any apps that handle the implicit intent you send to[startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)). If that happens, the call will fail and your app will crash. To verify that an activity will receive the intent, call [resolveActivity()](http://developer.android.com/reference/android/content/Intent.html#resolveActivity(android.content.pm.PackageManager)) on your [Intent](http://developer.android.com/reference/android/content/Intent.html) object. If the result is non-null, then there is at least one app that can handle the intent and it's safe to call [startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)). If the result is null, you should not use the intent and, if possible, you should disable the feature that issues the intent.
6. When [startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)) is called, the system examines all of the installed apps to determine which ones can handle this kind of intent (an intent with the [ACTION\_SEND](http://developer.android.com/reference/android/content/Intent.html#ACTION_SEND) action and that carries "text/plain" data). If there's only one app that can handle it, that app opens immediately and is given the intent. If multiple activities accept the intent, the system displays a dialog so the user can pick which app to use..

**Layout**

1. All user interface elements in an Android app are built using [View](http://developer.android.com/reference/android/view/View.html) and [ViewGroup](http://developer.android.com/reference/android/view/ViewGroup.html) objects.
2. A [View](http://developer.android.com/reference/android/view/View.html)is an object that draws something on the screen that the user can interact with.
3. A [ViewGroup](http://developer.android.com/reference/android/view/ViewGroup.html) is an object that holds other [View](http://developer.android.com/reference/android/view/View.html) (and [ViewGroup](http://developer.android.com/reference/android/view/ViewGroup.html)) objects in order to define the layout of the interface.
4. Each view group is an invisible container that organizes child views, while the child views may be input controls or other widgets that draw some part of the UI.
5. The name of an XML element for a view is respective to the Android class it represents. So a <TextView>element creates a [TextView](http://developer.android.com/reference/android/widget/TextView.html) widget in your UI, and a <LinearLayout> element creates a [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) view group.
6. When you load a layout resource in your app, Android initializes each node of the layout into a runtime object you can use to define additional behaviors, query the object state, or modify the layout.
7. A **View** is an object that draws something on the screen that the user can interact with and a **ViewGroup** is an object that holds other View (and ViewGroup) objects in order to define the layout of the user interface.

**Linear Layout**

1. [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the [android:orientation](http://developer.android.com/reference/android/widget/LinearLayout.html#attr_android:orientation) attribute.
2. All children of a [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) are stacked one after the other, so a vertical list will only have one child per row, no matter how wide they are,
3. horizontal list will only be one row high (the height of the tallest child, plus padding).
4. A [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) respects margins between children and the gravity (right, center, or left alignment) of each child.
5. [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) also supports assigning a weight to individual children with the [android:layout\_weight](http://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html#attr_android:layout_weight) attribute.
6. To create a linear layout in which each child uses the same amount of space on the screen, set the[android:layout\_height](http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html#attr_android:layout_height) of each view to "0dp" (for a vertical layout) or the [android:layout\_width](http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html#attr_android:layout_width) of each view to "0dp" (for a horizontal layout). Then set the[android:layout\_weight](http://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html#attr_android:layout_weight) of each view to "1".

**Linear Layout**

1. [RelativeLayout](http://developer.android.com/reference/android/widget/RelativeLayout.html) is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent [RelativeLayout](http://developer.android.com/reference/android/widget/RelativeLayout.html) area (such as aligned to the bottom, left or center).
2. By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from [RelativeLayout.LayoutParams](http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html).
3. [android:layout\_alignParentTop](http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_alignParentTop)
4. [android:layout\_centerVertical](http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_centerVertical)
5. [android:layout\_below](http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_below)
6. [android:layout\_toRightOf](http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_toRightOf)

# Table

1. [TableLayout](https://developer.android.com/reference/android/widget/TableLayout.html) is a [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) that displays child [View](https://developer.android.com/reference/android/view/View.html)elements in rows and columns.
2. [TableLayout](https://developer.android.com/reference/android/widget/TableLayout.html) positions its children into rows and columns. TableLayout containers do not display border lines for their rows, columns, or cells. The table will have as many columns as the row with the most cells. A table can leave cells empty, but cells cannot span columns, as they can in HTML.

ScrollView

1. Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display. A ScrollView is a [FrameLayout](http://developer.android.com/reference/android/widget/FrameLayout.html), meaning you should place one child in it containing the entire contents to scroll; this child may itself be a layout manager with a complex hierarchy of objects. A child that is often used is a [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) in a vertical orientation, presenting a vertical array of top-level items that the user can scroll through.
2. You should never use a ScrollView with a [ListView](http://developer.android.com/reference/android/widget/ListView.html), because ListView takes care of its own vertical scrolling.
3. ScrollView only supports vertical scrolling. For horizontal scrolling, use [HorizontalScrollView](http://developer.android.com/reference/android/widget/HorizontalScrollView.html).

**Step 11: Reading Input from EditText Controls**

Now that your form is designed and the controls have been implemented, you next need to collect the form data from the individual fields when the Button control is clicked.

For an EditText control, you use the getText() method.

final EditText nameField = (EditText) findViewById(R.id.EditTextName);

String name = nameField.getText().toString();

final EditText emailField = (EditText) findViewById(R.id.EditTextEmail);

String email = emailField.getText().toString();

final EditText feedbackField = (EditText) findViewById(R.id.EditTextFeedbackBody);

String feedback = feedbackField.getText().toString();

**Step 12: Reading Input From Spinner Controls**

Your form included a Spinner control. You use the getSelectedItem() method to read the data from this form control.

final Spinner feedbackSpinner = (Spinner) findViewById(R.id.SpinnerFeedbackType);

String feedbackType = feedbackSpinner.getSelectedItem().toString();

In this case, the selected item in the Spinner control is the String chosen by the user of the selected item.

**Step 13: Reading Input from CheckBox Controls**

Finally, your form included a CheckBox control. In this case, the result is just a flag to tell your application if the box was checked or not.

final CheckBox responseCheckbox = (CheckBox) findViewById(R.id.CheckBoxResponse);

boolean bRequiresResponse = responseCheckbox.isChecked();

You can use this Boolean value however you want in your app.