Module 3 : UI Design and Data storage

Table Layout

The TableLayout groups views into rows and columns. You use the <TableRow> element to designate a row in the table. Each row can contain one or more views. Each view you place within a row forms a cell. The width of each column is determined by the largest width of each cell in that column.

Consider the content of main.xml shown here:

```
<?xml version=”1.0” encoding=”utf-8”?>
<TableLayout
xmlns:android=”http://schemas.android.com/apk/res/android”
android:layout_height=”fill_parent”
android:layout_width=”fill_parent”
>
<TableRow>
<TextView
android:text=”User Name:”
android:width =”120px”
/>
<EditText
android:id=”@+id/txtUserName”
android:width=”200px” />
</TableRow>

<TableRow>
<TextView
android:text=”Password:”
/>
<EditText
android:id=”@+id/txtPassword”
android:password=”true”
/>
</TableRow>
<TableRow>
<TextView />
```

```
<CheckBox android:id="@+id/chkRememberPassword"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Remember Password"
/>
</TableRow>
<TableRow>
<Button
android:id="@+id/buttonSignIn"
android:text="Log In" />
</TableRow>
</TableLayout>
```

Figure 3-7 shows what the preceding looks like when rendered on the Android Emulator.

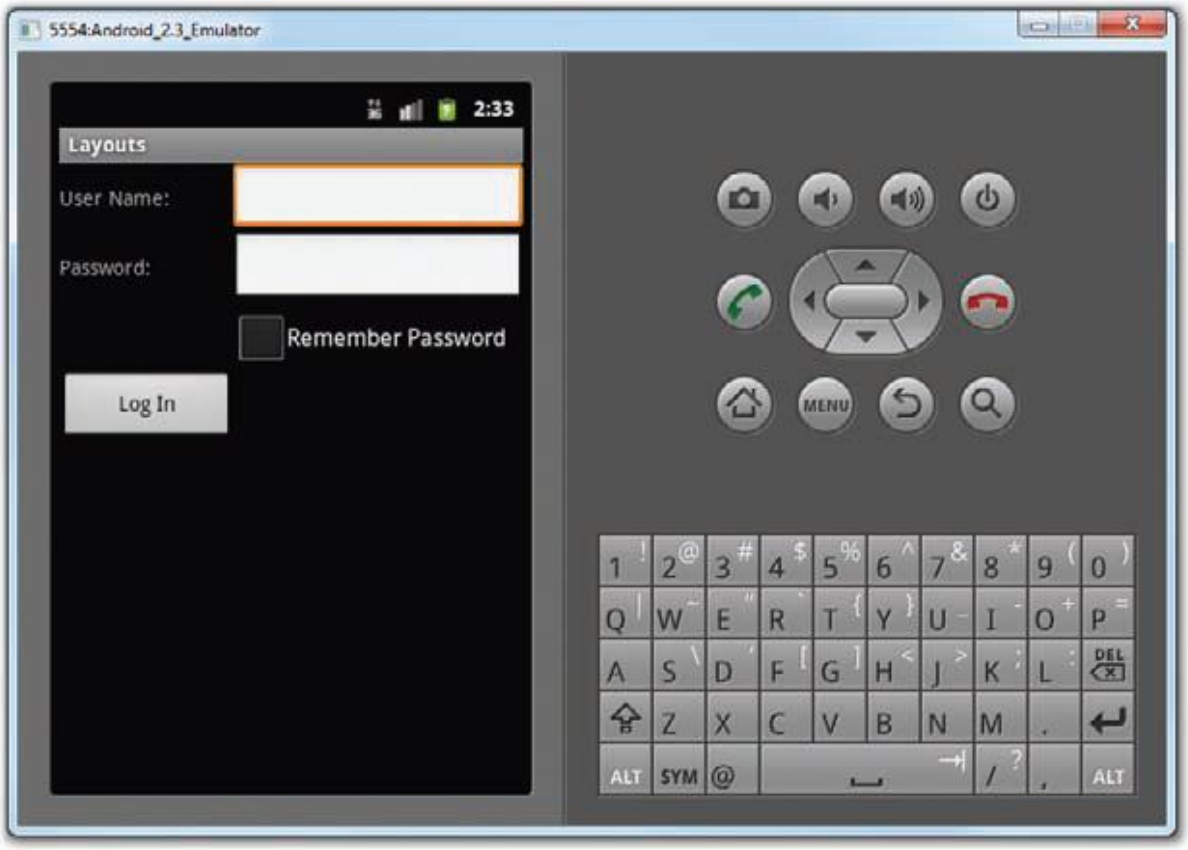

Figure 3-7

Note that in the preceding example, there are two columns and four rows in the TableLayout. The cell directly under the Password TextView is populated with an <TextView/> empty element. If you don't do this, the Remember Password checkbox will appear under the Password TextView, as shown in Figure 3-8.

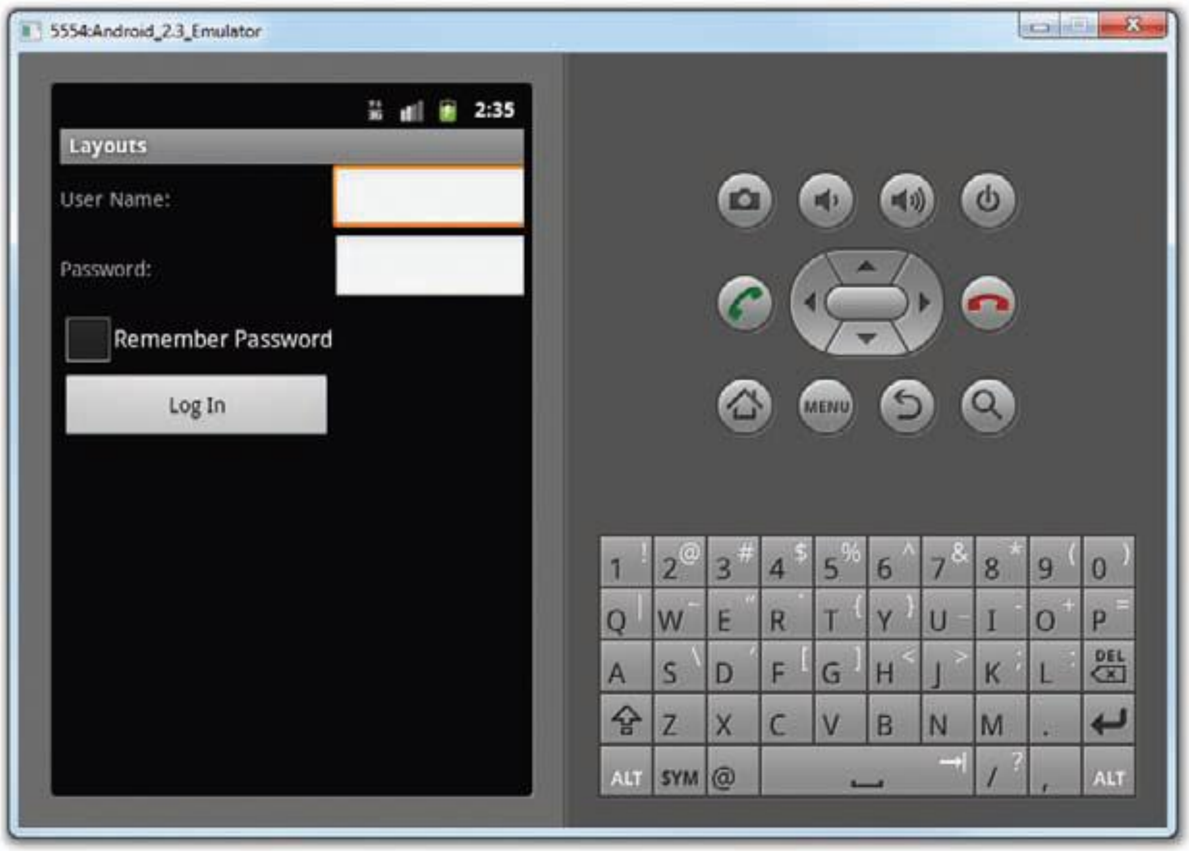

Figure 3-8

RelativeLayout

The RelativeLayout enables you to specify how child views are positioned relative to each other.

Consider the following main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
android:id="@+id/RLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/lblComments"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Comments"
```

```
android:layout_alignParentTop=”true”
android:layout_alignParentLeft=”true”
/>
<EditText
android:id=”@+id/txtComments”
android:layout_width=”fill_parent”
android:layout_height=”170px”
android:textSize=”18sp”
android:layout_alignLeft=”@+id/lblComments”
android:layout_below=”@+id/lblComments”
android:layout_centerHorizontal=”true”
/>
<Button
android:id=”@+id/btnSave”

android:layout_width=”125px”
android:layout_height=”wrap_content”
android:text=”Save”
android:layout_below=”@+id/txtComments”
android:layout_alignRight=”@+id/txtComments”
/>
<Button
android:id=”@+id/btnCancel”
android:layout_width=”124px”
android:layout_height=”wrap_content”
android:text=”Cancel”
android:layout_below=”@+id/txtComments”
android:layout_alignLeft=”@+id/txtComments”
/>
</RelativeLayout>
```

Notice that each view embedded within the RelativeLayout has attributes that enable it to align
with another view. These attributes are as follows:

➢➢ layout_alignParentTop

- layout_alignParentLeft
- layout_alignLeft
- layout_alignRight
- layout_below
- layout_centerHorizontal

The value for each of these attributes is the ID for the view that you are referencing. The preceding XML UI creates the screen shown in Figure 3-9.

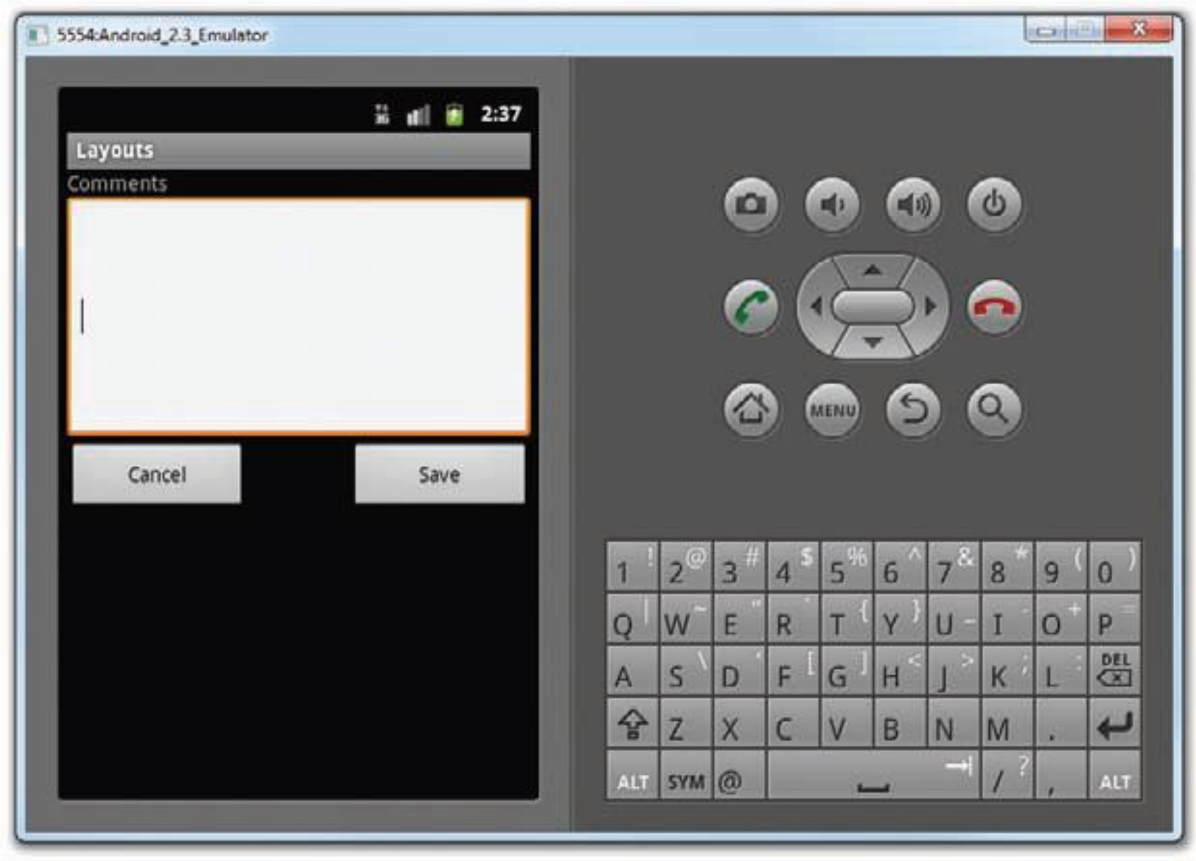

Figure 3-9