



Andhra Pradesh State Skill Development Corporation



Andhra Pradesh State Skill Development Corporation



ANDROID APPLICATION DEVELOPMENT

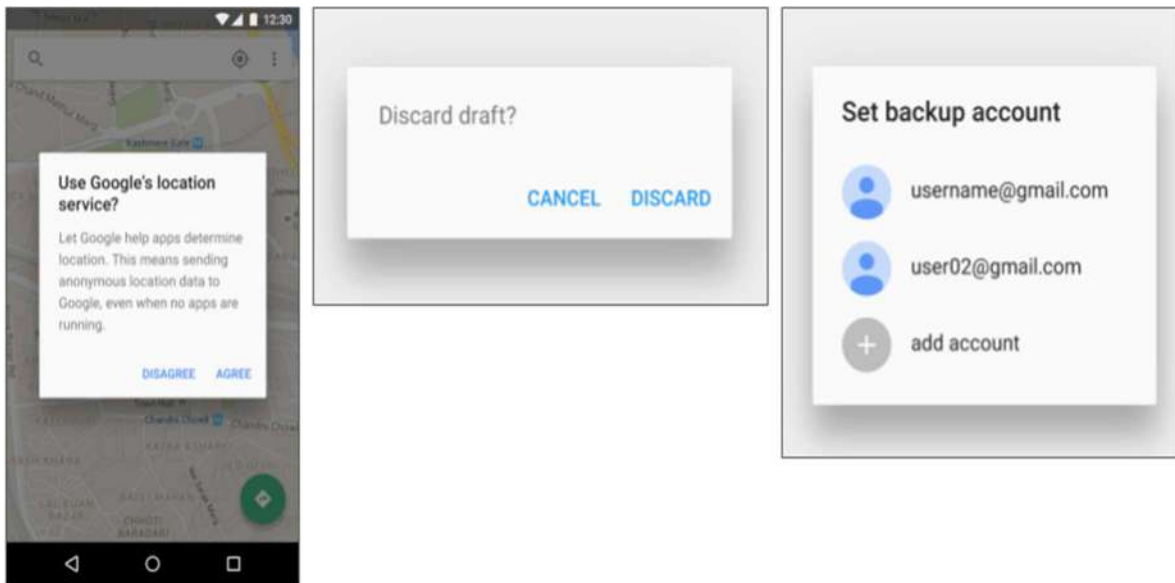
PICKERS _ DIALOG

Dialogs and pickers

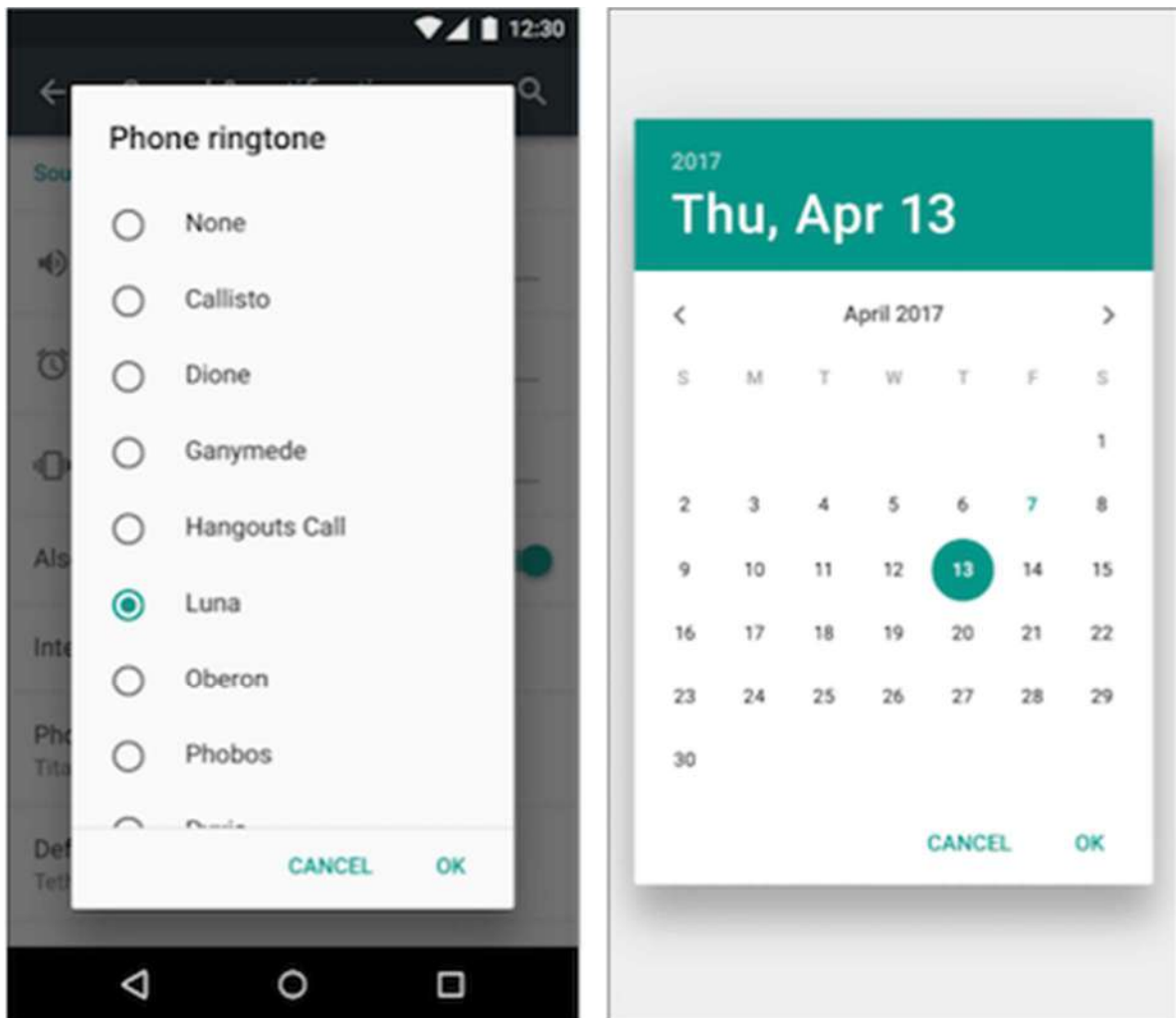
A dialog is a window that appears on top of the display or fills the display, interrupting the flow of *Activity*. Dialogs inform users about a specific task and may contain critical information, require decisions, or involve multiple tasks.

For example, an alert dialog might require the user to click **Continue** after reading it, or give the user a choice to agree with an action by clicking a positive button (such as **OK** or **Accept**), or to disagree by clicking a negative button (such as **Cancel**).

You can also use a dialog to provide choices in the style of radio buttons, as shown on the right side of the figure below.



The base class for all dialog components is a [Dialog](#). There are several useful *Dialog* subclasses for alerting the user on a condition, showing status or progress, displaying information on a secondary device, or selecting or confirming a choice, as shown on the left side of the figure below. The Android SDK also provides ready-to-use dialog subclasses such as pickers for picking a time or a date, as shown on the right side of the figure below. Pickers allow users to enter information in a predetermined, consistent format that reduces the chance for input error.



Dialogs always retain focus until dismissed or a required action has been taken.

Tip: Best practices recommend using dialogs sparingly as they interrupt the user's workflow. Read the [Dialogs design guide](#) for additional best design practices, and [Dialogs](#) in the Android developer documentation for code examples.

The [Dialog](#) class is the base class for dialogs, but you should avoid instantiating *Dialog* directly unless you are creating a custom dialog. For standard Android dialogs, use one of the following subclasses:

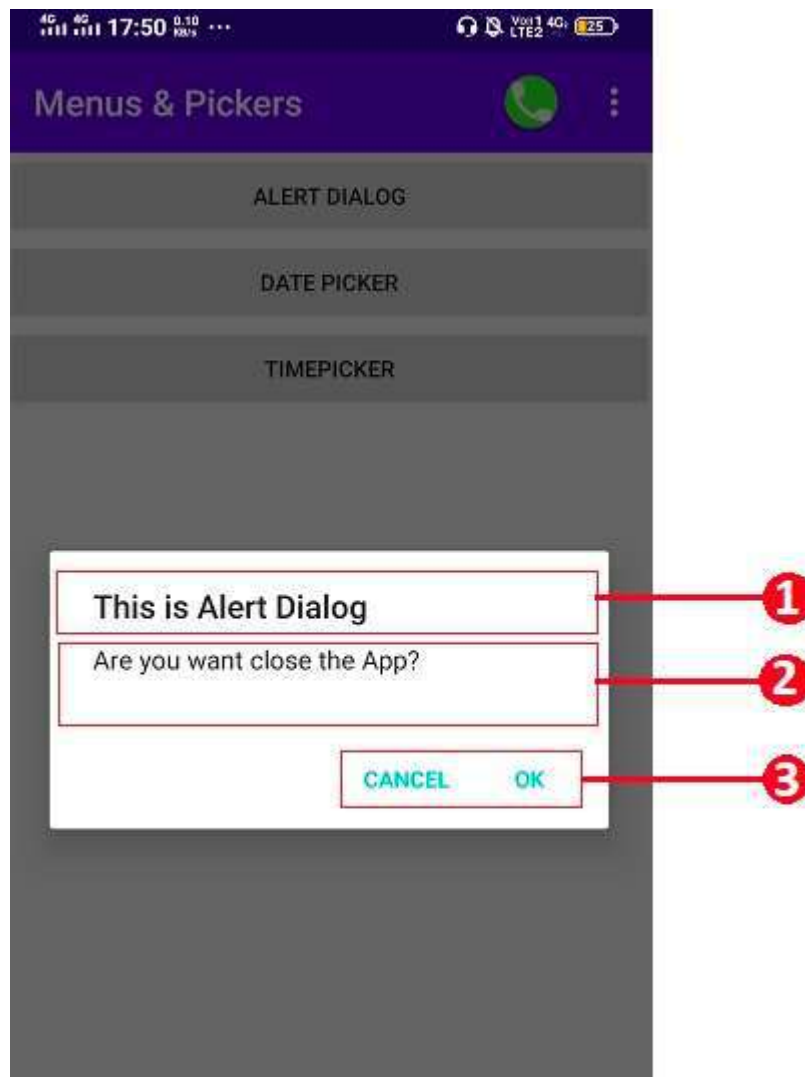
- **[AlertDialog](#):** A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.
- **[DatePickerDialog](#):** A dialog with a predefined UI that lets the user select a date.
- **[TimePickerDialog](#):** A dialog with a predefined UI that lets the user select a time.

Showing an alert dialog

Alerts are urgent interruptions, requiring acknowledgement or action, that inform the user about a situation as it occurs, or an action before it occurs (as in discarding a draft). You can provide buttons in an alert to make a decision.

For example, an alert dialog might require the user to click **Continue** after reading it, or give the user a choice to agree with an action by clicking a positive button (such as **OK** or **Accept**), or to disagree by clicking a negative button (such as **Disagree** or **Cancel**).

Use the [AlertDialog](#) subclass of the [Dialog](#) class to show a standard dialog for an alert. The *AlertDialog* class allows you to build a variety of dialog designs. An alert dialog can have the following regions (refer to the diagram below):



1. **Title:** A title is optional. Most alerts don't need titles. If you can summarize a decision in a sentence or two by either asking a question (such as, "Discard draft?") or making a statement related to the action buttons (such as, "Click OK to continue"), don't bother with a title. Use a title if the situation is high-risk, such as the potential loss of connectivity or data, and the content area is occupied by a detailed message, a list, or custom layout.
2. **Content area:** The content area can display a message, a list, or other custom layout.
3. **Action buttons:** You should use no more than three action buttons in a dialog, and most have only two.

Building the AlertDialog

The [AlertDialog.Builder](#) class uses the builder design pattern, which makes it easy to create an object from a class that has a lot of required and optional attributes and would therefore require

a lot of parameters to build. Without this pattern, you would have to create constructors for combinations of required and optional attributes; with this pattern, the code is easier to read and maintain. For more information about the builder design pattern, see [Builder pattern](#).

Use [AlertDialog.Builder](#) to build a standard alert dialog, with [setTitle\(\)](#) to set its title, [setMessage\(\)](#) to set its message, and [setPositiveButton\(\)](#) and [setNegativeButton\(\)](#) to set its buttons. If [AlertDialog.Builder](#) is not recognized as you enter it, you may need to add the following *import* statements to the *Activity*:

```
import android.content.DialogInterface;
import android.support.v7.app.AlertDialog;
```

The following creates the dialog object (*alertDialog*) and sets the title (the string resource *alerttitle_*) and message (the string resource *alertmessage_*):

```
AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
alertDialog.setTitle("This is Alert Dialog");
alertDialog.setMessage("Are you want close the App?");
```

Setting the button actions for the alert dialog

Use the [setPositiveButton\(\)](#) and [setNegativeButton\(\)](#) methods to set the button actions for the alert dialog. These methods require a title for the button and the [DialogInterface.OnClickListener](#) class that defines the action to take when the user presses the button:

```
alertDialog.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // User clicked OK button.
        // ... Action to take when OK is clicked.
    }
});
alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // User clicked the CANCEL button.
        // ... Action to take when CANCEL is clicked.
    }
});
```

You can add only one of each button type to an *AlertDialog*. For example, you can't have more than one "positive" button.

Tip: You can also set a "neutral" button with [setNeutralButton\(\)](#). The neutral button appears between the positive and negative buttons. Use a neutral button, such as **Remind me later**, if you want the user to be able to dismiss the dialog and decide later.

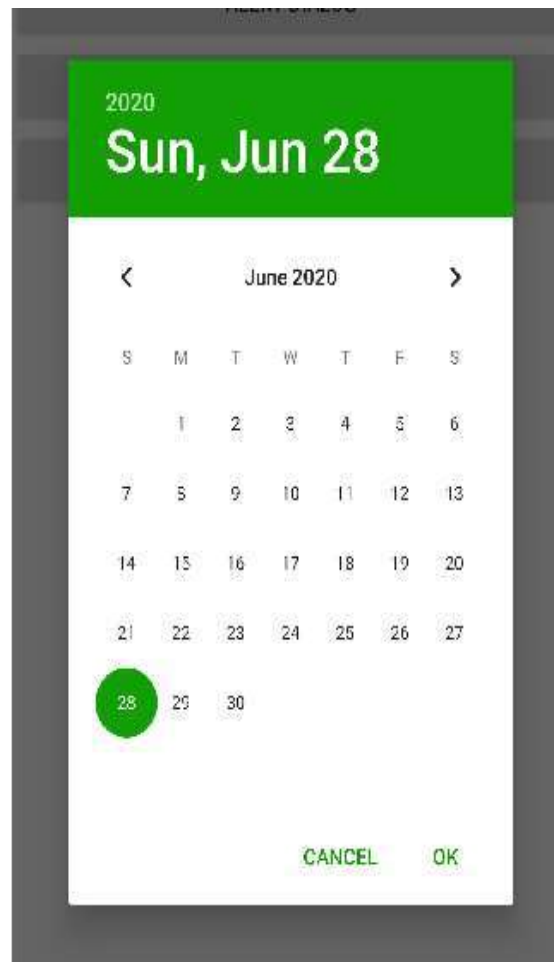
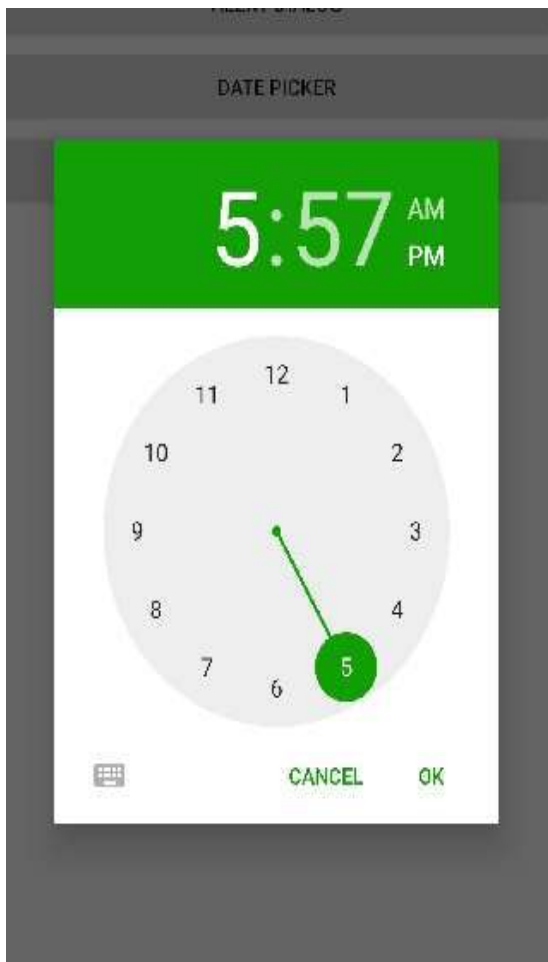
Displaying the dialog

To display the dialog, call its `show()` method:

AlertDialog.show();

Date and Time pickers

Android provides ready-to-use dialogs, called *pickers*, for picking a time or a date. Use them to ensure that your users pick a valid time or date that is formatted correctly and adjusted to the user's locale. Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).



When showing a picker, you should use an instance of `DatePickerDialog`, which displays a dialog window floating on top of its Activity window.

Tip: Here you can use fragments instead of for the pickers is that you can implement different layout configurations, such as a basic dialog on handset-sized displays or an embedded part of a layout on large displays.

Create DatePickerDialog

1. Implement `DatePickerDialog.OnDateSetListener` to create a standard date picker with a listener.

```
DatePickerDialog datePicker= new DatePickerDialog(this, new
DatePickerDialog.OnDateSetListener() {
```

// Implement OnDateSetListener

});

- Click the red bulb icon and choose **Implement methods** from the popup menu. A dialog appears with `onDateSet()` already selected and the **Insert @Override** option selected. Click **OK** to create the empty `onDateSet()` method. This method will be called when the user sets the date.

import android.widget.DatePicker;

The `onDateSet()` parameters should be `int i`, `int i1`, and `int i2`. Change the names of these parameters to ones that are more readable:

public void onDateSet(DatePicker datePicker, **int** year, **int** month, **int** day)

- You use your version of the callback method to initialize the *year*, *month*, and *day* for the date picker. For example, you can add the following code to `onCreateDialog()` to initialize the *year*, *month*, and *day* from `Calendar`, and return the dialog and these values to the *Activity*. As you enter `Calendar.getInstance()`, specify the import to be `java.util.Calendar`.

// Use the current date as the default date in the picker.

```
final Calendar c = Calendar.getInstance();  
int year = c.get(Calendar.YEAR);  
int month = c.get(Calendar.MONTH);  
int day = c.get(Calendar.DAY_OF_MONTH);
```

The `Calendar` class sets the default date as the current date—it converts between a specific instant in time and a set of calendar fields such as *YEAR*, *MONTH*, *DAYOF_MONTH*, and *HOURL*. `Calendar` is locale-sensitive. The `Calendar getInstance()` method returns a `Calendar` whose fields are initialized with the current date and time.

Displaying the Picker

To display the `DatePickerDialog`, call its `show()` method:

```
datePicker.show();
```

Implementation of DatePickerDialog

```
public void dpd(View view) {  
    int c_date,c_month,c_year;  
    Calendar c=Calendar.getInstance();  
    c_year=c.get(Calendar.YEAR);  
    c_date=c.get(Calendar.DATE);  
    c_month=c.get(Calendar.MONTH);  
    DatePickerDialog datePicker= new DatePickerDialog(this, new DatePickerDialog.OnDateS  
etListener() {  
        @Override  
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {  
            Toast.makeText(MainActivity.this, dayOfMonth+"-"+(month+1)+"-"+year, Toast.LEN  
GTH_SHORT).show();  
        }  
    },c_year,c_month,c_date);  
    datePicker.show();  
}
```

```
}
```

Time Picker Dialog:

Follow the same procedures outlined above for a date picker, with the following differences:

1. Implement `TimePickerDialog.OnTimeSetListener` to create a standard date picker with a listener.

```
TimePickerDialog pickerDialog=new TimePickerDialog(this, new TimePickerDialog.OnTimeSetListener() {  
    //Implement OnTimeset Listener  
});
```

2. Click the red bulb icon and choose **Implement methods** from the popup menu. A dialog appears with `onTimeSet()` already selected and the **Insert @Override** option selected. Click **OK** to create the empty `onTimeSet()` method. This method will be called when the user sets the date.

```
import android.widget.TimePicker;
```

The `onTimeSet()` parameters should be *TimePicker* view, *int hourOfDay*, and *int minute*. Change the names of these parameters to ones that are more readable:

```
public void onTimeSet(TimePicker view, int hourOfDay, int minute)
```

3. You use your version of the callback method to initialize the *year*, *month*, and *day* for the date picker. For example, you can add the following code to `onCreateDialog()` to initialize the *year*, *month*, and *day* from `Calendar`, and return the dialog and these values to the *Activity*. As you enter `Calendar.getInstance()`, specify the import to be `java.util.Calendar`.

```
// Use the current date as the default date in the picker.
```

```
final Calendar c = Calendar.getInstance();  
int hours = c.get(Calendar.HOUR_OF_DAY);  
int minute = c.get(Calendar.MINUTE);
```

The `Calendar` class sets the default time as the current time—it converts between a specific instant in time and a set of calendar fields such as *hourOfDay*, *minute*, and *HOURLY*. `Calendar` is locale-sensitive. The `Calendar.getInstance()` method returns a `Calendar` whose fields are initialized with the current date and time.

Displaying the Picker

To display the `TimePickerDialog`, call its `show()` method:

```
timePicker.show();
```

Implementation of TimePickerDialog

```
public void tpd(View view) {  
    int c_hours,c_minute;  
    Calendar c=Calendar.getInstance();  
    c_hours=c.get(Calendar.HOUR_OF_DAY);  
    c_minute=c.get(Calendar.MINUTE);  
    TimePickerDialog timePicker=new TimePickerDialog(this, new TimePickerDialog.OnTimeSetListener() {
```




```
@Override
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    Toast.makeText(MainActivity.this, hourOfDay+":"+minute, Toast.LENGTH_SHORT)
        .show();
}
}, c_hours, c_minute, false);
timePicker.show();
}
```