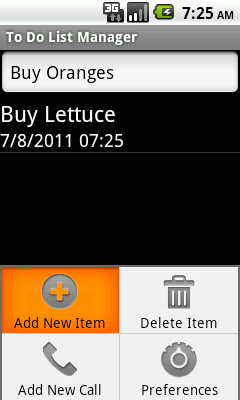
LAB: Module 3 – TODO List Manager

In this lab, you will create a TODO List Manager application. Your TODO List Manager will use the Android Contact Picker and Dialer for special “Call” TODO items.

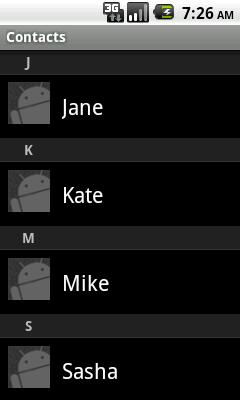
# Part 1

1. Create a new Android project named **ToDoListManager**.
2. Create a menu with three items: “Add New Item”, “Delete Item”, and “Add New Call”.
3. In the main activity, create a text editor for inputting new TODO items and a list of existing TODO items.
4. To add a new TODO item, the user will choose “Add New Item” from the menu, input its title into the text editor, and click the ENTER key.
   1. The text editor should become visible only after the “Add New Item” option is selected in the menu, and should become invisible after the item is added. (To control visibility, use the View.setVisibility() method.)
   2. The resulting TODO item should be stored in the list as is. I.e., the user’s text input should be the text displayed in the list.

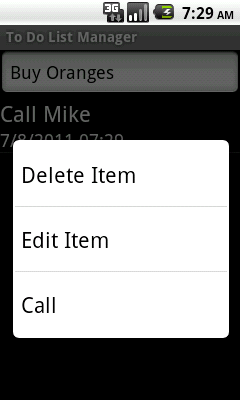
TIP: It is highly recommended that you use a class to store TODO item information, and override its toString() method to get the display text. The reason is that we will soon want to add more information to the TODO items, which will not be displayed in the UI.

# Part 2

1. To add a new “Call” TODO item, the user will choose “Add New Call” from the menu.
   1. When “Add New Call” is selected, you should launch the Android Contact Picker activity and receive its result in your main activity.
   2. The resulting TODO item should be stored in the list in the format “Call <name>”. If the contact has a phone number listed, you should store the phone number in your TODO item class, to be used later.

TIP: To use the contact picker and to receive the result, you should use code similar to the [following](#ContactPickerCode) (click to follow). This also requires adding the READ\_CONTACTS [permission](#PermissionCode) to your application manifest.

To save time, you can instead parse the user’s input – if the input is “Call 555-1212”, create a “Call” TODO item for the number 555-1212.

1. When the user chooses “Delete Item” from the menu, you should delete the currently selected item from the list. (If there is no selected item, do nothing.)
2. Create a context menu for the item list with two items: “Delete Item” and “Call”.
   1. When the “Delete Item” item is clicked in the context menu, you should delete the item from the list.
   2. The “Call” context menu item should be enabled only when “Call” TODO items are selected for the context menu. (Alternatively, you can create only the “Delete Item” menu item if the TODO item is not a “Call” TODO item.)

TIP: You can use the onPrepareContextMenu() notification to customize the menu just prior to displaying it.

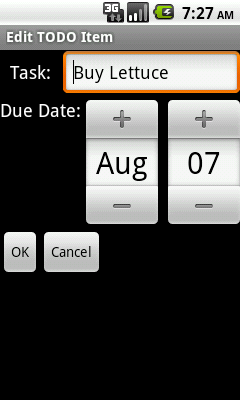
* 1. When the “Call” item is clicked in the context menu, you should open the Android Dialer with the phone number to call.

TIP: You can use Intent.ACTION\_DIAL or Intent.ACTION\_CALL to achieve this. Which is a better idea? Feel free to consult the online documentation.

# Part 3

1. Modify the list view so that it contains two TextViews for each row – one for the TODO item itself and another with the time and date when it was created.
   1. You can position the two views horizontally (TODO item on the left, date and time on the right) or vertically (TODO item at the top, date and time on the bottom).
   2. You should set the text sizes such that the date and time are smaller than the TODO item text.

TIP: To do this, you will have to derive from ArrayAdapter<T> and customize the view returned from its getView() method.

1. Create another activity named **EditToDoItemActivity**.
   1. The activity should contain text fields and edit fields for the TODO item’s text (or phone number in case of a “Call” TODO item) and for its due date (a new piece of information). Use a DatePicker for the due date control.
   2. The activity should have “OK” and “Cancel” buttons.

TIP: Don’t forget to register this activity in the application manifest.

1. Add another item to the list item context menu: “Edit Item”.
   1. When this item is clicked, the **EditToDoItemActivity** should open with the details of the task selected in the main list.
   2. If the user clicks “OK”, the changes should be reflected in the main UI. If the user clicks “Cancel”, no changes should occur.

# Part 4

1. Turn the **EditToDoItemActivity** into a dialog activity, using a dialog theme.

Code for contact picker:

**private** **void** addNewCall() {

Intent intent = **new** Intent(  
 Intent.*ACTION\_PICK*, ContactsContract.Contacts.*CONTENT\_URI*);

startActivityForResult(intent, *PICK\_CONTACT\_CODE*);

}

@Override

**protected** **void** onActivityResult(  
 **int** requestCode, **int** resultCode, Intent data) {

**super**.onActivityResult(requestCode, resultCode, data);

**if** (requestCode == *PICK\_CONTACT\_CODE* && resultCode == *RESULT\_OK*) {

Cursor cursor = managedQuery(  
 data.getData(), **null**, **null**, **null**, **null**);

cursor.moveToFirst();

String displayName = cursor.getString(  
 cursor.getColumnIndexOrThrow(  
 ContactsContract.Contacts.*DISPLAY\_NAME*));

**if** ("0".equals(cursor.getString(  
 cursor.getColumnIndexOrThrow(  
 ContactsContract.Contacts.*HAS\_PHONE\_NUMBER*)))) {

ToDoItem item = **new** ToDoItem("Call " + displayName);

items.add(item);

adapter.notifyDataSetChanged();

**return**;

}

String contactId = cursor.getString(  
 cursor.getColumnIndexOrThrow(  
 ContactsContract.Contacts.*\_ID*));

cursor.close();

cursor = managedQuery(Phone.*CONTENT\_URI*, **null**,   
 Phone.*CONTACT\_ID* + "=?", **new** String[] { contactId }, **null**);

cursor.moveToFirst();

String phoneNumber = cursor.getString(

cursor.getColumnIndexOrThrow(Phone.*NUMBER*));

cursor.close();

CallToDoItem item = **new** CallToDoItem(displayName, phoneNumber);

items.add(item);

adapter.notifyDataSetChanged();

}

}

Manifest modification to add the READ\_CONTACTS permission:

<uses-permission android:name=*"android.permission.READ\_CONTACTS"* />