

# ITW202: Mobile Application

## Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology  
Royal University of Bhutan

May 7, 2021

# Mobile Application Development

## Contextual Menus

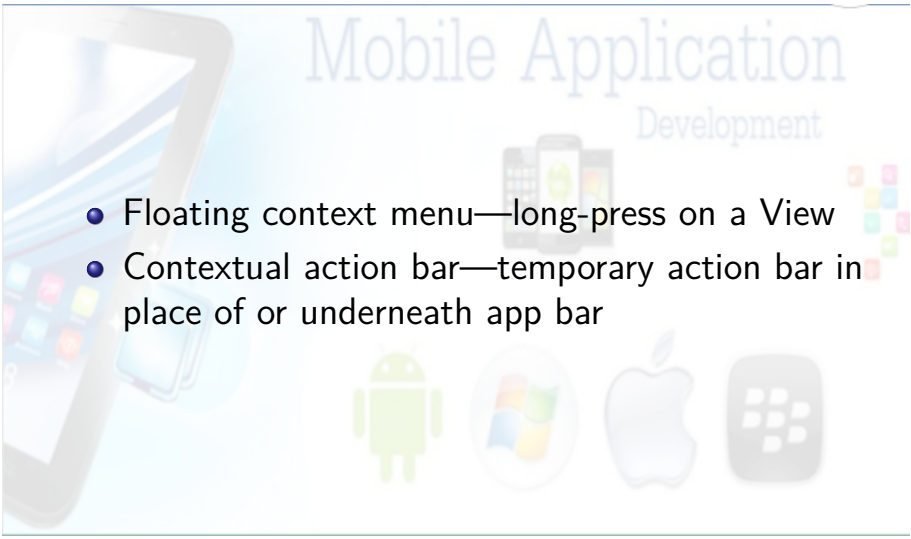


# Contextual Menus

## Mobile Application Development

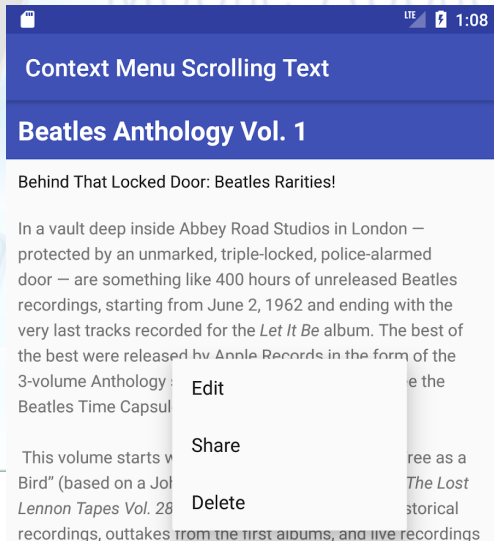
- Allows users to perform action on selected View
- Can be deployed on any View
- Most often used for items in RecyclerView, GridView, or other View collection

# Types of contextual menus

- 
- The background of the slide features a light blue gradient. On the left, there is a stylized illustration of a tablet and a smartphone. In the center, the text "Mobile Application Development" is written in a large, light blue, sans-serif font. Below this text, there are four icons: the Android robot, the Windows logo, the Apple logo, and the BlackBerry logo. To the right of the text, there is a small cluster of colorful squares.
- Floating context menu—long-press on a View
  - Contextual action bar—temporary action bar in place of or underneath app bar

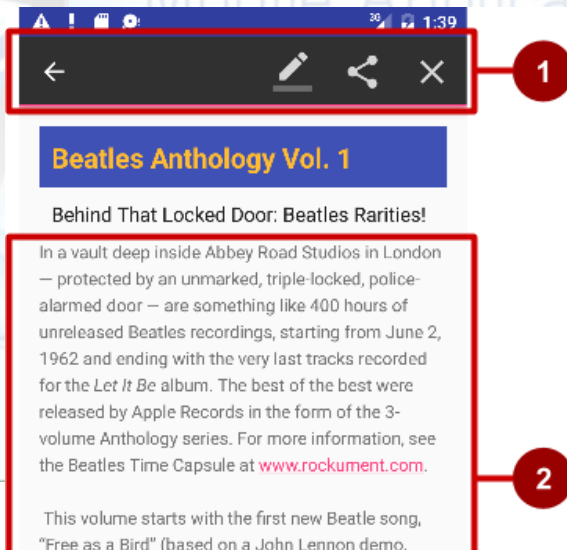
# Types of contextual menus

## Floating context menu.



# Types of contextual menus

## Contextual action bar



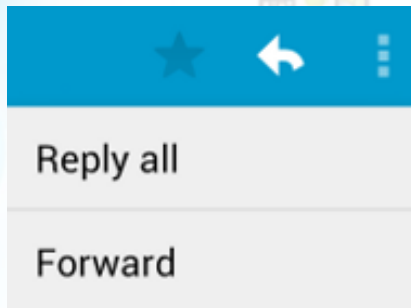
# Mobile Application Development

## Popup Menu



# What is a popup menu?

- Vertical list of items anchored to a view
- Typically anchored to a visible icon





# Mobile Application Development

## Dialogs



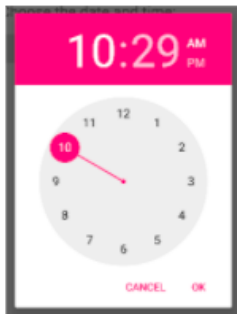
# Dialogs

## Mobile Application Development

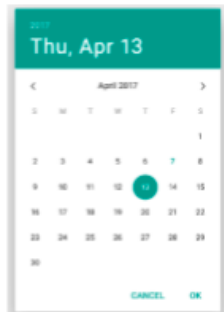
A **dialog** is a window that appears on top of the display or fills the display, interrupting the flow of Activity. Dialogs inform users about a specific task and may contain critical information, require decisions, or involve multiple tasks.

# Dialogs

## Mobile Application

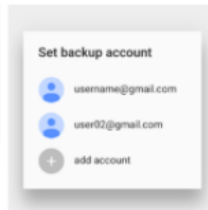
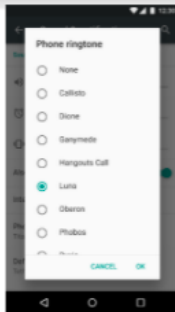
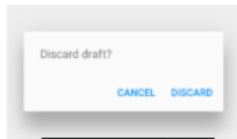
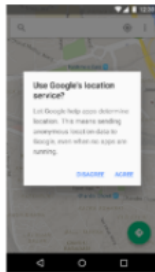


TimePickerDialog



DatePickerDialog

# Dialogs



## AlertDialog

# Dialogs

The **Dialog class** is the base class for dialogs. 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.

# AlertDialog

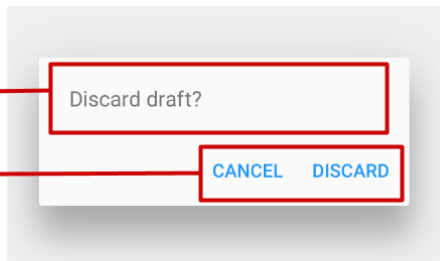
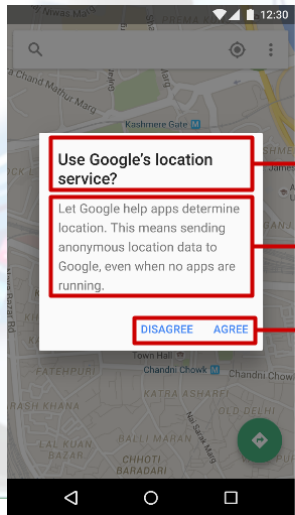
## Mobile Application Development

AlertDialog can show:

- 1 Title (optional)
- 2 Content area
- 3 Action buttons



# AlertDialog



# Build the AlertDialog

Use AlertDialog.Builder to build alert dialog and set attributes:

```
public void onClickShowAlert(View view) {  
    AlertDialog.Builder myAlertBuilder = new AlertDialog.Builder(  
        context: MainActivity.this);  
    myAlertBuilder.setTitle("Alert");  
    myAlertBuilder.setMessage("Click OK to continue, or Cancel to stop.");  
}
```



# Set the button actions

- `AlertDialog.setPositiveButton()`
- `AlertDialog.setNeutralButton()`
- `AlertDialog.setNegativeButton()`

# alertDialog code example

```
myAlertBuilder.setPositiveButton(text: "OK", new DialogInterface.OnClickListener(){  
    @Override  
    public void onClick(DialogInterface dialog, int which) {  
        Toast.makeText(getApplicationContext(), text: "Pressed OK",  
            Toast.LENGTH_SHORT).show();  
    }  
});
```

Same pattern for `setNegativeButton()` and `setNeutralButton()`

# Displaying the dialog

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

Mobile Application  
Development



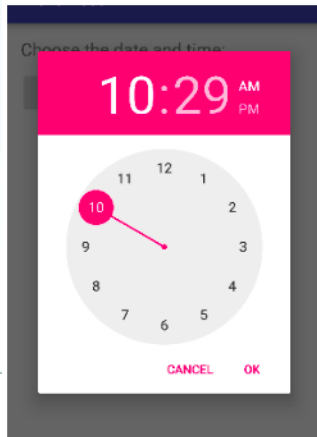
# Mobile Application Development

## Pickers



# Pickers use fragments

- Use DialogFragment to show a picker
- DialogFragment is a window that floats on top of Activity window



# Introduction to fragments

- A Fragment is like a mini-Activity within an Activity
  - Manages its own own lifecycle
  - Receives its own input events
- Can be added or removed while parent Activity is running
- Multiple fragments can be combined in a single Activity
- Can be reused in more than one Activity

# Creating a date picker dialog

- Add a blank Fragment that extends DialogFragment and implements DatePickerDialog.OnDateSetListener

```
public class DatePickerFragment extends DialogFragment  
    implements DatePickerDialog.OnDateSetListener {
```

# Creating a date picker dialog

- In `onCreateDialog()` initialize the date and return the dialog

```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    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);
    return new DatePickerDialog(getActivity(), listener: this, year, month, day);
}
```



# Creating a date picker dialog

- In `onDataSet()` handle the date

```
@Override
```

```
public void onDataSet(DatePicker view, int year, int month, int dayOfMonth) {  
    MainActivity activity = (MainActivity) getActivity();  
    activity.processDatePickerResult(year, month, dayOfMonth);  
}
```

# Creating a date picker dialog

- In Activity show the picker and add method to use date

```
public void showDatePicker(View view) {  
    DialogFragment newFragment = new DatePickerFragment();  
    newFragment.show(getSupportFragmentManager(), tag: "datePicker");  
}
```

# Creating a date picker dialog

- In Activity show the picker and add method to use date

```
public void processDatePickerResult(int year, int month, int day){  
    String month_string = Integer.toString(month + 1);  
    String day_string = Integer.toString(day);  
    String year_string = Integer.toString(year);  
  
    String date_message = (month_string + "/" + day_string + "/" + year_string);  
  
    Toast.makeText(context, this, text: "Date: " + date_message, Toast.LENGTH_SHORT).show()  
}
```

# Mobile Application Development

**THANK YOU**

