Chapter - 4
Working with Menu and Internal Application

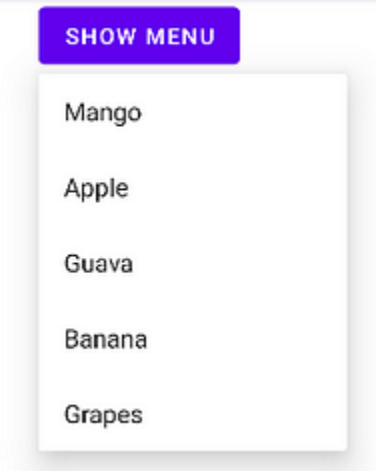
Prof. Devansh Parikh

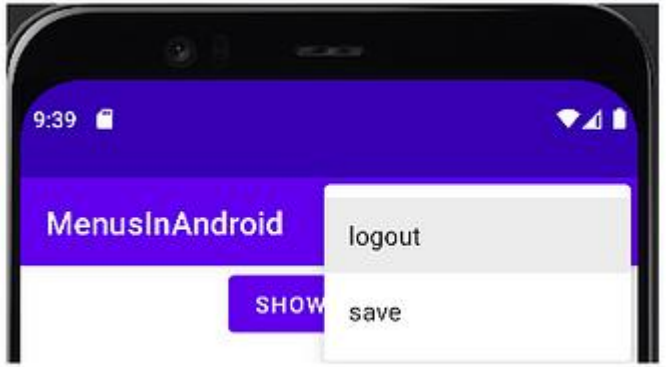
Table of Content.....

- Menus-(Option,Context, Popup)
- Images-ImageView
- AlertDialog,
- Alarm manager
- SMS,
- E-mail
- Media Player,
- Using camerarecording video
- Handling Telephony Manager

Menu

- Menus are common user interface in our Android applications that present user actions and other options in our activities.





Types of Menu

- Menus are created using Menu APIs. Menus can be used in three ways in our Android application.

1. **Options menu :** It contains the items that have the global impact on the application such as Bookmark, Search, Settings etc. Options menu is the primary collection of menu items.
2. **Context menu :** It is a floating menu that appears when the user performs a long-click on an element. A context menu contains the menu items that act on selected content or context frame.
3. **Popup menu :** It appears below the anchor view or above the view. It provides an overflow-style menu for actions that relate to specific content.

Working with Options Menu

- **Options Menu:** The Options Menu is a common menu used to provide actions and settings related to the current activity or fragment.
- It typically appears at the top of the screen when the user presses the menu button or the overflow icon. It allows you to include items such as "Settings," "Share," or "About.“
- Menus are created using XML format and can be updated by code at runtime. Menus are added to Activity within onCreateOptionsMenu() method.
- On click of menu items, actions are performed within onOptionsItemSelected() method in an Activity class.

Option menu (XML File)

- <menu> : Root node to define menu. This can contain <item> and <group> elements.
- <item> : Creates a menu item. This can contain <menu> element in order to create a submenu.
- <group> : An optional container for <item> elements. It is used to categorize menu items so they share properties such as active state and visibility.

Add Tool bar(activity_main.xml)

- <com.google.android.material.appbar.AppBarLayout
- android:layout_width="match_parent"
- android:layout_height="wrap_content"
- android:background="@color/material_on_surface_stroke"
- android:fitsSystemWindows="true">
- <com.google.android.material.appbar.MaterialToolbar
- android:id="@+id/toolbar"
- android:layout_width="match_parent"
- android:layout_height="?attr/actionBarSize" />
- </com.google.android.material.appbar.AppBarLayout>

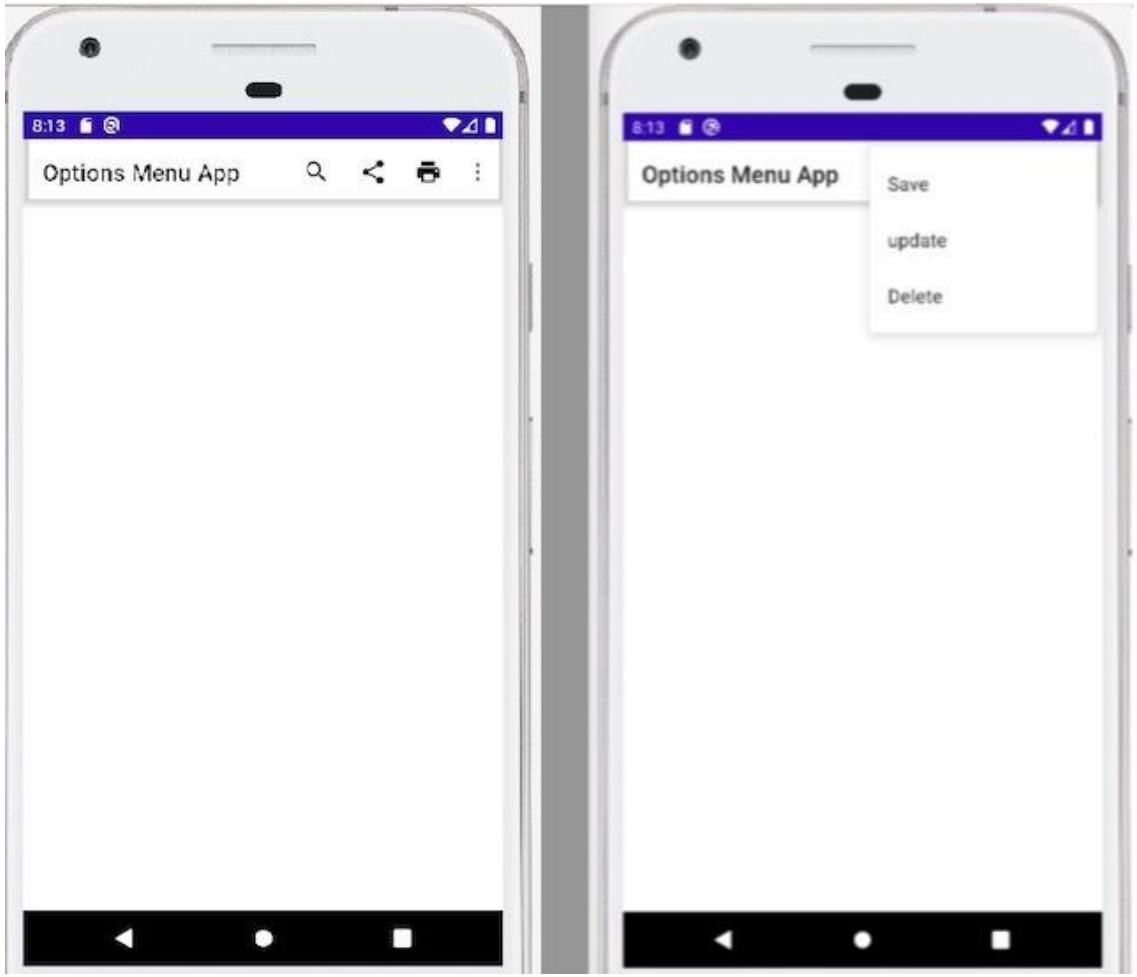
XML Code(Option Menu)

- <!—Code for Search Icon on top at tool bar -->
- <menu xmlns:android="http://schemas.android.com/apk/res/android"
- xmlns:app="http://schemas.android.com/apk/res-auto">
- <item android:id="@+id/search_menu"
- android:icon="@drawable/search"
- android:title="@string/search"
- app:showAsAction="always"/>
- </menu>

Option menu (Java File)

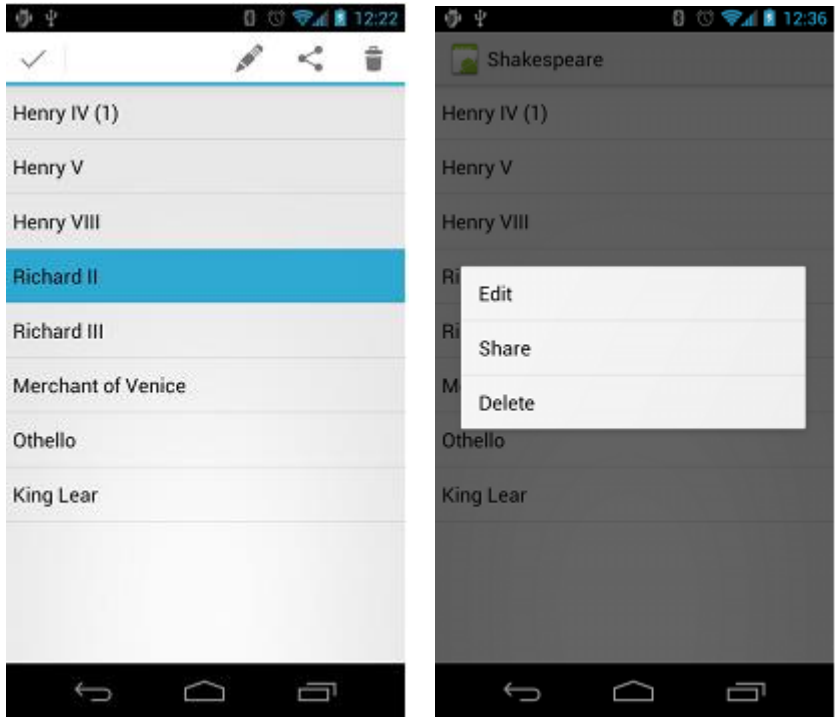
```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
//add on click listener
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Snackbar.make(binding.getRoot(), item.getTitle(), Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
        return true;
    }
```

Output


8:13
Options Menu App
8:13
Options Menu App
Save
update
Delete

Context Menu

- A context menu, shown on the left side in the figure, appears as a floating list of menu items when the user performs a long tap on a View. It is typically used to modify the View or use it in some fashion.
- For Example:
  - a context menu might include Edit to edit the contents of a View, Delete to delete a View, and Share to share a View over social media. Users can perform a contextual action on one selected View at a time.



Steps for Creating Context Menu

1. Create an XML menu resource file for the menu items. Assign appearance and position attributes as described in the previous section for the options menu.
2. Register a View to the context menu using the registerForContextMenu() method of the Activity class.
3. Implement the onCreateContextMenu() method in your Activity to inflate the menu.
4. Implement the onContextItemSelected() method in your Activity to handle menu-item clicks.
5. Create a method to perform an action for each context menu item.

Creating the XML resource file

- <item
- android:id="@+id/context_edit"
- android:title="Edit"
- android:orderInCategory="10"/>
- <item
- android:id="@+id/context_edit"
- android:title=“Delete"
- android:orderInCategory="10"/>

Registering a View to the context menu

- To register a View to the context menu, call the registerForContextMenu() method with the View.
- Registering a context menu for a view sets the View.OnCreateContextMenuListener on the View to this activity, so that onCreateContextMenu() is called when it's time to show the context menu.
- Then implement onCreateContextMenu in the next after onCreate method.

add registerForContextMenu()

- // Registering the context menu to the TextView of the article.
  - TextView article_text = findViewById(R.id.article);
  - registerForContextMenu(article_text);

Implementing the onCreateContextMenu() method

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.menu_context, menu);
}
```

Implementing the onContextItemSelected() method

```
@Override
public boolean onContextItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.context_edit:
      editNote();
      return true;
    case R.id.context_share:
      shareNote();
      return true;
    default:
      return super.onContextItemSelected(item);
  } }
```

Popup Menu

- In android, Popup Menu displays a list of items in a modal popup window that is anchored to the view. The popup menu will appear below the view if there is a room or above the view in case if there is no space and it will be closed automatically when we touch outside of the popup.
- The android Popup Menu provides an overflow style menu for actions that are related to specific content.
- Following is the pictorial representation of using Popup Menu in our android applications.

XML file (menu_example.xml).

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/mail"
      android:icon="@drawable/ic_mail"
      android:title="@string/mail" />
   <item android:id="@+id/upload"
      android:icon="@drawable/ic_upload"
      android:title="@string/upload"
      android:showAsAction="ifRoom" />
   <item android:id="@+id/share"
      android:icon="@drawable/ic_share"
      android:title="@string/share" />
</menu>
```

Add Button

```
<Button
   android:id="@+id/btnShow"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Show Popup Menu"
   android:onClick="showPopup" />
```

Load Android Popup Menu from an Activity

```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.menu_example, popup.getMenu());
    popup.show();
}
```

Handle Android Popup Menu Click Events

```
public void showMenu(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    popup.setOnMenuItemClickListener(this);
    popup.inflate(R.menu.actions);
    popup.show();
}
```

```
@Override
public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.archive:
            archive(item);
            return true;
        case R.id.delete:
            delete(item);
            return true;
        default:
            return false;
    }
}
```

Images-ImageView

- In Android, the ImageView class is used to display an image file in an application.
- Image files are easy to use but hard to master due to the various screen sizes across Android devices.
- Android provides rich UI design widgets that help developers create visually appealing and attractive applications.

Image View - Attributes

Attribute Name	Description
id	Used to uniquely identify an ImageView in Android.
src	Sets the source file (image) for the ImageView to enhance the layout's appearance.
background	Sets the background of an ImageView, which can be a color or a drawable.
padding	Defines padding from the left, right, top, or bottom of the ImageView.
scaleType	Controls how the image should be resized or moved to fit the ImageView. Possible values include fit_xy, center_crop, fitStart, etc.

Send SMS: To send an SMS using EditText

- Add SMS Permission in AndroidManifest.xml
  - <uses-permission android:name="android.permission.SEND_SMS"/>

activity_main.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">

   <EditText
      android:id="@+id/etPhoneNumber"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Enter phone number"
      android:inputType="phone" />
```

```
<EditText
      android:id="@+id/etMessage"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Enter message"
      android:inputType="textMultiLine" />

   <Button
      android:id="@+id/btnSend"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Send SMS"/>
</LinearLayout>
```

MainActivity.java

```
private static final int PERMISSION_REQUEST_CODE = 1;
EditText etPhoneNumber, etMessage;
Button btnSend;

etPhoneNumber = findViewById(R.id.etPhoneNumber);
etMessage = findViewById(R.id.etMessage);
btnSend = findViewById(R.id.btnSend);
```

Conti...(Button Click Event)

```
btnSend.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           if (ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.SEND_SMS)
              != PackageManager.PERMISSION_GRANTED) {
              ActivityCompat.requestPermissions(MainActivity.this,
                    new String[]{Manifest.permission.SEND_SMS}, PERMISSION_REQUEST_CODE);
           } else {
              sendSMS();
           }
        }
     });
```

Conti...(Send SMS)

```
private void sendSMS() {
        String phoneNumber = etPhoneNumber.getText().toString().trim();
        String message = etMessage.getText().toString().trim();

        if (!phoneNumber.isEmpty() && !message.isEmpty()) {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(phoneNumber, null, message, null, null);
            Toast.makeText(this, "SMS Sent!", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Please enter both fields", Toast.LENGTH_SHORT).show();
        }
    }
```

Conti...(Allow Permission)

```
@Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)
{
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                sendSMS();
            } else {
                Toast.makeText(this, "Permission denied!", Toast.LENGTH_SHORT).show();
            }
        }
    }
```

Send EMAIL: To send an Email using EditTexts

- Add SMS Permission in AndroidManifest.xml
  - <uses-permission android:name="android.permission.SEND_EMAIL"/>

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <EditText
        android:id="@+id/etEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter recipient's email"
        android:inputType="textEmailAddress"/>
    <EditText
        android:id="@+id/etSubject"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter subject"
        android:inputType="text"/>
```

```
<EditText
        android:id="@+id/etMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter message"
        android:inputType="textMultiLine"
        android:minLines="3"/>

    <Button
        android:id="@+id/btnSendEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Email"/>
</LinearLayout>
```

MainActivity.java

```
EditText etEmail, etSubject, etMessage;
Button btnSendEmail;


etEmail = findViewById(R.id.etEmail);
etSubject = findViewById(R.id.etSubject);
etMessage = findViewById(R.id.etMessage);
btnSendEmail = findViewById(R.id.btnSendEmail);
```

Conti...(Button Click)

```
btnSendEmail.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            sendEmail();
        }
    });
```

Conti..(Send Email)

```
private void sendEmail() {
        String email = etEmail.getText().toString().trim();
        String subject = etSubject.getText().toString().trim();
        String message = etMessage.getText().toString().trim();

        if (!email.isEmpty() && !subject.isEmpty() && !message.isEmpty()) {
            Intent intent = new Intent(Intent.ACTION_SENDTO);
            intent.setData(Uri.parse("mailto:")); // Only email apps should handle this
            intent.putExtra(Intent.EXTRA_EMAIL, new String[]{email});
            intent.putExtra(Intent.EXTRA_SUBJECT, subject);
            intent.putExtra(Intent.EXTRA_TEXT, message);
```

Conti..(Send Email)

```
try {
            startActivity(Intent.createChooser(intent, "Choose an email client"));
        } catch (Exception e) {
            Toast.makeText(this, "No email client installed",
Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show();
    }
  }
```