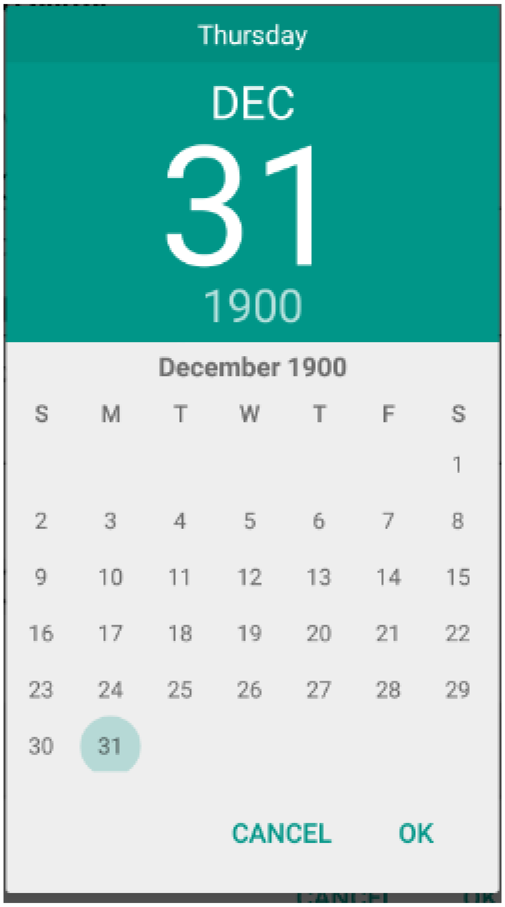
Using DatePickerDialog in Android

The `DatePickerDialog` in Android is a convenient way for users to select a date from a visual calendar interface. Here’s a comprehensive guide on how to implement and use `DatePickerDialog` in your Android application.

Overview

`DatePickerDialog` is a pre-built dialog that allows the user to pick a date. It is part of the Android framework and can be used to simplify date selection in your apps.



December 1900

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Implementation Steps

1. **Setup the Layout**
2. **Show the DatePickerDialog**
3. **Handle Date Selection**

Step-by-Step Implementation

1. Setup the Layout

First, define a layout with a `TextView` and a `Button` to show the selected date and to trigger the `DatePickerDialog` respectively.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/layout/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">

   <TextView
       android:id="@+id/tvSelectedDate"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Selected Date"
       android:textSize="18sp" />

   <Button
       android:id="@+id/btnSelectDate"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Select Date"
       android:layout_marginTop="16dp" />
</LinearLayout>
```

2. Show the DatePickerDialog

In your activity, set up the Button to show the DatePickerDialog when clicked.

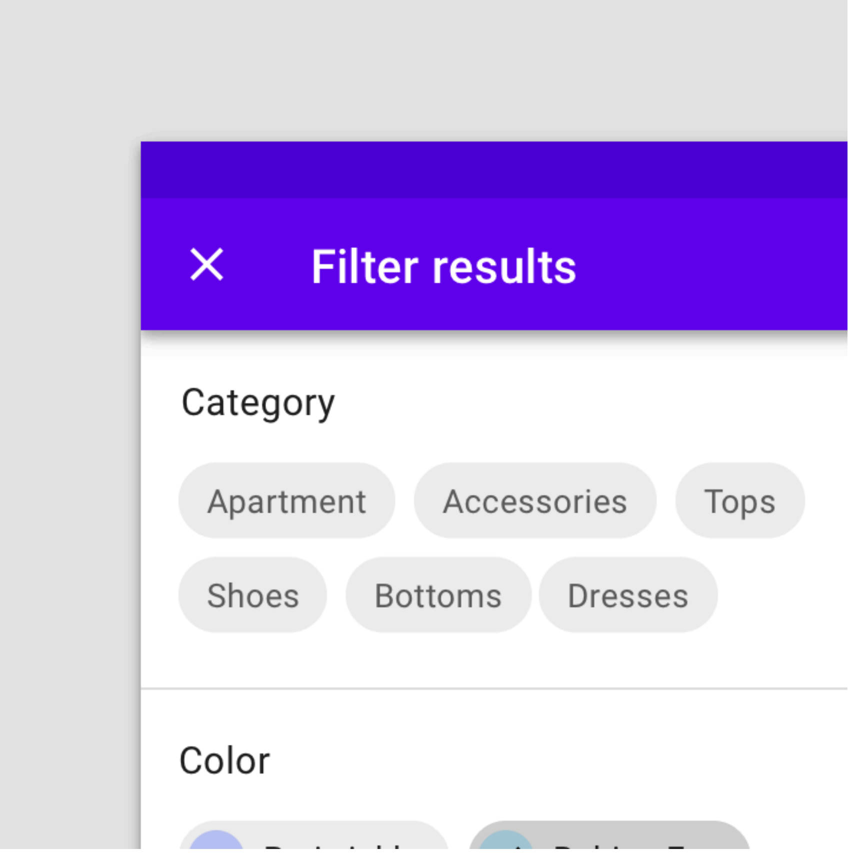
```
// MainActivity.java
import android.app.DatePickerDialog;
import android.os.Bundle;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private TextView tvSelectedDate;
    private Button btnSelectDate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvSelectedDate = findViewById(R.id.tvSelectedDate);
        btnSelectDate = findViewById(R.id.btnSelectDate);

        btnSelectDate.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    showDatePickerDialog();
                }
            });

    }

    private void showDatePickerDialog() {
        final Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(
                MainActivity.this,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker view, int selectedYear, int
selectedMonth, int dayOfMonth) {
                        String selectedDate = selectedYear + "-" +
(selectedMonth + 1) + "-" + dayOfMonth;
                        tvSelectedDate.setText(selectedDate);

                    }
                },
                year, month, day
        );

        datePickerDialog.show();
    }
}
```

ChipGroup

The ChipGroup in Android is a flexible and efficient way to display a group of Chip components, often used for selections, filtering, or tagging. It provides a more user-friendly and visually appealing alternative to traditional checkboxes and radio buttons.

Overview

A ChipGroup is a container that holds multiple Chip elements. It supports single or multiple selection modes and can be styled to fit the look and feel of your application.



Overview

In this example, we'll:

1. Define a ChipGroup in the XML layout.
2. Add Chip components to the ChipGroup programmatically.
3. Handle chip removal using the close icon click listener.

Step-by-Step Implementation

1. Setup the Layout

First, define a ChipGroup in your XML layout file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">



   <com.google.android.material.chip.ChipGroup
       android:id="@+id/chipGroup"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       app:singleSelection="true" />  <!-- Use single or multiple selection
mode -->
   <Button
       android:layout_marginTop="16dp"
       android:id="@+id/btnAddChip"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Add Chip"
       android:layout_marginBottom="16dp" />
</LinearLayout>
```

2. Initialize ChipGroup and Add Chips Programmatically

In your activity, set up the ChipGroup, add chips dynamically, and handle chip removal using a click listener.

```
// MainActivity.java
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.material.chip.Chip;
import com.google.android.material.chip.ChipGroup;

public class MainActivity extends AppCompatActivity {

   private ChipGroup chipGroup;
```

```
    private Button btnAddChip;
    private int chipCount = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        chipGroup = findViewById(R.id.chipGroup);
        btnAddChip = findViewById(R.id.btnAddChip);

        // Set button click listener to add chips
        btnAddChip.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addChip("Dynamic Chip " + (++chipCount));
            }
        });
    }

    // Method to add a chip dynamically
    private void addChip(String text) {
        Chip chip = new Chip(this);
        chip.setText(text);
        chip.setCloseIconVisible(true);  // Enable close icon
        chip.setClickable(true);
        chip.setCheckable(false);

        // Set close icon click listener
        chip.setOnCloseIconClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                chipGroup.removeView(chip);
            }
        });

        chipGroup.addView(chip);
    }
}
```

Adding Chips

```
chipGroup.setSingleSelection(true);
```

```
Chip firstChip = null;
for (int i = 0; i < categoryArrayList.size(); i++) {
   Chip chip = new Chip(chipGroup.getContext());
   chip.setText(categoryArrayList.get(i).getName());
   chip.setId(View.generateViewId());
   chip.setTag(categoryArrayList.get(i));
   chip.setCheckable(true);
   chipGroup.addView(chip);
   if (i == 0) {
       firstChip = chip;
   }
}
firstChip.setChecked(true);
chipGroup.setSingleSelection(true);
```

Get Data From chip Selected

```
Chip selectedChip=findViewById(chipGroup.getCheckedChipId());
Category category= (Category) selectedChip.getTag();
```

Chip Remove Icon Listener

```
btnAdd.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       if(validateFields())
       {
           long
count=dbHelper.insertCategory(etName.getText().toString().trim());
           if(count>0)
           {
               Chip chip = new Chip(chipGroup.getContext());
               chip.setText(etName.getText().toString().trim());
               chip.setTag(new Category((int)
count,etName.getText().toString().trim()));
               chip.setCheckable(true);
               chip.setCloseIconVisible(true);
               chip.setId(ViewCompat.generateViewId());
               chip.setCloseIconVisible(true); // Enable close icon
               chip.setOnCloseIconClickListener(new View.OnClickListener() {
                   @Override
                   public void onClick(View v) {
                       Category category= (Category) chip.getTag();
                       int count=dbHelper.deleteCategory(category.getId());
                       if(count>0)
                       {
                           chipGroup.removeView(chip);
                           chipGroup.invalidate();
                       }

                   }
               });
               chipGroup.addView(chip);
           }
       }
       }

});
```

===

Remove Chips From List

```
ArrayList<Category> categoryArrayList = dbHelper.getAllCategories();
for (int i = 0; i < categoryArrayList.size(); i++) {
   Chip chip = new Chip(chipGroup.getContext());
   chip.setText(categoryArrayList.get(i).getName());
   chip.setTag(categoryArrayList.get(i));
   chip.setCheckable(false);
   chip.setId(ViewCompat.generateViewId());
   if(i==0){
   }else{
       chip.setCloseIconVisible(true); // Enable close icon
       chip.setOnCloseIconClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               Category category= (Category) chip.getTag();
               int count=dbHelper.deleteCategory(category.getId());
               if(count>0)
               {
                   chipGroup.removeView(chip);
               }
```

```
                }
            });
        }

        chipGroup.addView(chip);
    }
```