Remove floating action button android

I'm trying to replace a third party FloatingActionButton with a native that's packed into the library com.android.support:design:22.2.0.the default look has a dark shadow around the image, How can I get rid of it? I know that the first one provides the setShadow method, but I just can't find a similar one from the latter. This is a related XML layout: And I've installed its background color zlt;android.support.design.widget.FloatingActionButton android:id'id/alarm_front android:layout_width'wrap_content android:layout_height'wrap_content android: src'@drawable/btn_icon_alarm_notset'lt;'android support.wrap_content.' mAlarmBtn.setBackground TintList (colorStateList.valueOf (getResources ().getColor (R.color.floatButtonColor); Google seeks to promote racial equality for black people. View specific initiatives. The Suspension Action Button (FAB) is a circular button that triggers the main action in the app interface. This page describes how to add a suspended action button to the layout, adjust some of the look of the button, and respond to the click of a button. To learn more about designing suspension action buttons in the app according to the Materials Design Guide, see buttons: Hover buttons. Figure 1. The Hover Action button Add the suspended action button to the layout The next code shows how FloatingActionButton should appear in the layout file: Default action button qlt;android.support.design.widget.FloatingActionButton android:id'id/fab android:layout_width'wrap_content android:layout_height wrap_content android: layout_gravity'end'bottom android:@drawable/ic_my_icon android:contentdescription'@string/submit android:layout_margin'16dp'gt;/android.support.design.widget.FloatingActionButton the pendant is painted with colorAccent property, that can be customized with a palette with a theme background. You can use the XML property or the appropriate method to customize other features of the suspended action button, as follows: Action of the response button then you can apply View.OnClickListener to handle the action of the action button. For example, the following code displays Snackbar when a user presses the suspended action button: val fab: View s findViewById (R.id.fab) fab.setOnClickListener's view --gt; Snackbar.make (view, here's the diner, Snackbar.LENGTH_LONG) .setAction (Action, null.show () fab.setOnClickListener (newView.OnClickListener (newView.onClickListener(s @Override public void onClick (View) s snackbar.make (view, here's the diner, Snackbar.LENGTH_LONG) .setAction (Action, null).show (); To learn more about the functionality of the suspension action button, see the content samples and the code on this page. Java is a registered Oracle and/or its affiliates. Last updated 2020-08-06 UTC. The floating action button is slightly different from the usual buttons. Floating action buttons are implemented in the user interface of the primary action application (encouraged). (encouraged). for users and actions under the floating button, actions are a priority for the developer. For example, activities such as adding an item to an existing list. Thus, this article has shown how to implement the Floating Action Button (FAB), and the buttons under the FAB are handled with a simple toast message. Please note that we are going to implement this project using java. Types of floating action buttons there are basically four types of floating action buttons available on Android. Normal/Regular Floating Action Buttons Mini Floating Action Buttons Advanced Floating Action Button Theming Floating Action Button In this article let's discuss normal/regular floating action buttons and mini floating action buttons with an example in Android. Normal/Regular Floating Action Buttons Regular FAB are FAB that are not extended and are regular sized. The following example shows a regular FAB with a plus icon. Step 1: Create a new project to create a new project in Android Studio, please refer to How to Create/Start a New Project in Android Studio. Please note that you choose Java as a programming language. Step 2: Add dependency on the App-level Gradle file. Here we use a floating action button that is designed and developed by Google Material Design Team. Add dependency in the build.gradle file (app) as: implementation 'com.google.android.material:material:1.3.0-alpha02' Make sure to add dependency to the app level Gradle file. After adding dependency, you need to click on the Sync Now button that appears in the top right corner of Android Studio IDE. When you press the Sync Now button, make sure you're connected to the network to download the files you need. Refer to the image below if you can't get the steps mentioned above: Step 3: Add FAB icons to drawble file for demonstration purposes will import 3 icons into the drawable folder, you can import icons of your choice. You can do this by clicking the right button to draw a folder - New - Check out the following image to import the vector icon. Now select the vector icon: Now open activity_main.xml and add floating action buttons. Step 4: Work with activity_main.xml in activity_main.xml add floating action buttons and trigger the following code. Now call the normal FAB button. Which has a radius of 56p. We chained the sub FAB buttons to the FAB parent button so they were in one key line. Comments are added inside the code to understand the code in more detail. ?xml version 1.0 codingutf-8?'gt; zlt;androidx.constraintlayout.widget.ConstraintLayout xmlns:android xmlns:app' андроид:layout_width match_parent match_parent инструменты:контекст». Инструменты MainActivity :игнорироватьHardcodedText> <com.google.android.material.floatingactionbutton.FloatingActionButton android:id=@+id/add_fab android:layout_width=wrap_content android:layout_height=wrap_content android:layout_gravity=end android:layout_marginend=16dp android:layout_marginbottom=16dp android:src=@drawable/ic_add_black_24dp app:fabsize=normal app:layout_constraintbottom_tobottomof=parent app:layout_constraintend_toendof=parent></com.google.android.material.floatingactionbutton.FloatingActionButton> <com.google.android.material.floatingactionbutton.FloatingActionButton android:id=@+id/add_alarm_fab android:layout_width=wrap_content android:layout_height=wrap_content android:layout_marginbottom=24dp app:fabsize=normal app:layout_constraintbottom_totopof=@+id/add_fab app:layout_constraintend_toendof=@+id/add_fab app:layout_constraintstart_tostartof=@+id/add_fab app:srccompat=@drawable/ic_add_alarm_black_24dp></com.google.android.material.floatingactionbutton.FloatingActionButton> <TextView android:id=@+id/add_alarm_action_text android :layout_width=wrap_content android:layout_height=wrap_content android:layout_marginend=8dp android:text=Add Alarm app:layout_constraintbottom_tobottomof=@+id/add_alarm_fab app:layout_constraintend_tostartof=@+id/add_alarm_fab app:layout_constrainttop_totopof=@+id/add_alarm_fab></TextView> <com.google.android.material.floatingactionbutton.FloatingActionButton android:id=@+id/add_person_fab android:layout_width=wrap_content android:layout_height=wrap_content android:layout_marginbottom=24dp app:fabsize=normal app:layout_constraintbottom_totopof=@+id/add_alarm_fab app:layout_constraintend_toendof=@+id/add_alarm_fab app:layout_constraintstart_tostartof=@+id/add_alarm_fab app:srccompat=@drawable/ic_person_add_black_24dp></com.google.android.material.floatingactionbutton.FloatingActionButton> <TextView android:id=@+id/add_person_action_text android:layout_width=wrap_content android :layout_height=wrap_content android:layout_marginend=8dp android:text=Add Person app:layout_constraintbottom_tobottomof=@+id/add_person_fab app:layout_constraintend_tostartof=@+id/add_person_fab app:layout_constrainttop_totopof=@+id/add_person_fab></TextView> The user interface is made as: Step 5: Working with the MainActivity.java file Now handle all these FAB buttons using the setOnClickListener method () you can refer to The Tap Event Processing in the Android button. Now the next code is called to MainActivity.java to process them. Read the comments under the code for better understanding. This code has shown that when sub FABs should be visible from onClickListener. Comments are added inside the code to understand the code in more detail. Import android.os.Bundle; Import android.view.View; Import android.widget.TextView; import android.widget.Toast; import androidx.appcompat.app.AppCompatActivity; import com.google.android.material.floatingactionbutton.FloatingActionButton; MainActivity Expands AppCompatActivity - FloatingActionButton mAddFab, mAddAlarmFab, mAddPersonFab; TextView addAlarmActionText, addPersonActionText; Boolean isAllFabsVisible; @Override protected void onCreate (Bundle savedInstanceState) - super.onCreate (savedInstanceState); setContentView (R.layout.activity_main); mAddFab - findViewById (R.id.add_fab); mAddAlarmFab - findViewById (R.id.add_alarm_fab); mAddPersonFab - findViewById (R.id.add_person_fab); addAlarmActionText - findViewById (R.id.add_alarm_action_text); addPersonActionText - findViewById (R.id.add_person_action_text); mAddAlarmFab.setVisibility (View.GONE); mAddPersonFab.setVisibility (View.GONE); addAlarmActionText.setVisibility (View.GONE); addPersonActionText.setVisibility (View.GONE); isAllFabsVisible - false; mAddFab.setOnClickListener (new View.OnClickListener.) - @Override public void on Click (View) - if (!isAllFabsVisible) - mAddAlarmFab.show (); mAddPersonFab.show(); addAlarmActionText.setVisibility (View.VISIBLE); addPersonActionText.setVisibility (View.VISIBLE); isAllFabsVisible - true; - more mAddAlarmFab.hide(); mAddPersonFab.hide(); addAlarmActionText.setVisibility (View.GONE); isAllFabsVisible - false; } } }); mAddPersonFab.setOnClickListener (new View.OnClickListener)- @Override public void on Click (View) - Toast.makeText (MainActivity.this, Person Added, Toast.LENGTH_SHORT).) } }); mAddAlarmFab.setOnClickListener (new View.OnClickListener)- @Override public void on Click (View) - Toast.makeText (MainActivity.this, Alarm Added, Toast.LENGTH_SHORT).) } }); Exit: Running on the emulator mini-FAB are used on small screens. Mini-FAB can also be used to create visual continuity with other screen elements. The following example shows a mini-FAB with a plus icon. Create a Mini FAB Now to trigger the FAB mini button. That has a radius of 40 d/s. All other things rename the same. Only one attribute needs to be changed, which: app:fabSizemini After making changes output user interface looks like this: Note: The FAB button can also be made automatic resizable by changing the attribute fabSize to

automatic. This will make the size of the FAB automatically mini and normal depending on the size of the window. Appendix:fabsize auto attention reader! Don't stop learning now. Get all the important Java and Collections concepts with the Java Collection Basics and Java course at a cost-friendly price for students and get ready to work in the industry. Recommended messages: If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or send your article by mail contribute@geeksforgeeks.org. See your article by appearing on the GeeksforGeeks Homepage and help other Geeks.Please improve this article if you find anything wrong by clicking on the Improve article below. Below. remove floating action button android studio. android floating action button remove shadow. android floating action button remove padding. android floating action button remove border