Buse Çarık 12/09/18

INTERRA R&D Material Design

Material Design

Chips

https://github.com/InterraMaterialDesign/Chips

**INDEX**

1. **INTRODUCTION….….….…….….……….………….……………………….……….3**
   1. **Purpose of this document…………………………………………………………...3**
   2. **Overview………………………….……….…….…………………………………...3**
2. **WHAT IS CHIP...…………...………...…………………………………………………4**
   1. **Listeners……………………………….………………….………………………….5**
   2. **Entry Chip……………………………………………….……….……….…………7**
   3. **Filter Chip…………….…….……….………………………………………………8**
   4. **Choice Chip………………………………………………………………….………9**
   5. **Action Chip…………………………………………………………………………10**
3. **CHIPGROUP.…………………………………………………….……………….……12**
4. **CHIPDRAWABLE……………………………….……….……………………………13**
5. **USING CHIPS IN AUTOCOMPLETE TEXT……………………………………….13**
6. **REQUIREMENTS FOR IMPLEMENTATION…….…………….…………………16**
7. **REFERENCES…………………………………….……………………………………17**

**INTRODUCTION**

**PURPOSE OF THIS DOCUMENT**

The aim of this project is analyzing chips, their types and chip group.

**OVERVIEW**

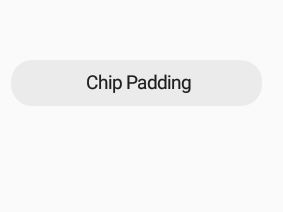
The project that is described in this document, includes chip’ description, types of the chip, chip group, styles, where and when it is used and how it should be designed according to Material Design.

**Chip**

Chips allow the user to make a choice, filter the content, take an input or trigger an action. They consist of a label and optional a close icon and chip icon and has a rounded background. They are usually used with ***ChipGroups*** which acts as a container of chips. Unlike buttons, the chips that are in the same area, are expected to perform the same actions.

Chips’ appearance can be customized with a bunch of attributes. Their minimum height can be set with ***android:chipMinHeight*** attribute. Its roundness of the corners can be set with ***android:chipCornerRadius*** attribute. Its color and thickness of the borders are changed with ***android:chipStrokeColor*** and ***android:chipStrokeWidth*** attributes. With ***android:chipBackgroundColor*** attribute, the background color of the chip can be changed but when this is changed, a feature is lost. By default, when a chip is selected, its color is become darker. If the color of the chip is customized, this feature should be re-implemented if it is desired. Chip has a ripple effect in black color by default. This is customized with ***android:rippleColor***.

Chip’s left and right padding is set with ***chipStartPadding*** and ***chipEndPadding***.



Start padding End padding

The text label also has left and right padding; ***textStartPadding*** and ***textEndPadding***.



textStartPadding textEndPadding

Chips are checkable which indicates whether the chip can be selected. This feature can be set with ***android:checkable=”true /false”*** attribute. And the chip can be set as checked when it is first declared with ***android:checked=”true/false”*** attribute. If the chip is checkable, check icon is set to the chip. This is optional. When chip is selected, this icon appears left of the label. This icon can be control with ***checkedIconEnabled***, ***checkIconVisible*** and ***checkedIcon*** attributes. With checkedIconEnabled attribute, the check icon can be removed or set. checkIconVisible attribute sets the visibility of the icon. And the icon can be changed with checkIcon attribute.

The close icon is found in some types of chips. It is placed the right of the text label by default. It can be used in all chips with ***android:closeIconEnabled=”true”*** attribute. The icon itself can be changed with ***android:closeIcon***. Its color and size also can be changed with ***android:closeIconTint*** and ***android:closeIconSize***. The visibility of the icon is set with ***android:closeIconVisible*** attribute. Its left and right padding are also set with ***android:closeIconStartPadding*** and ***closeIconEndPadding***.

Besides the close and check icons, an icon that is desired by the developer can be placed into the left of the text label. This icon is set with ***android:chipIcon***. This icon also has ***chipIconTint***(sets its color), ***chipIconSize***(sets its size), ***chipIconVisible***(sets its visibility), ***iconStartPadding*** and ***iconEndPadding*** (set the left and right padding of the icon).

**Listeners**

Chip has ***setOnClickListener*** callback. When the chip is clicked, this callback is invoked. Instead of setOnClickListener, ***setOnCheckedChangeListener*** can be used. This callback is invoked when the chip’s state changes (checked/unchecked).

Chip chip = findViewById(R.id.chip);

chip.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

}

});

Another callback method is ***setOnCloseIconClickListener***. The close icon in the chip is clicked, this callback method is invoked.

Chip chip = findViewById(R.id.chip).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

}

});

Note: setting the layout direction of the chip as ***LAYOUT\_DIRECTION\_LOCALE*** ensures that the padding of the text label properly when the direction of the text is not the LTR direction.

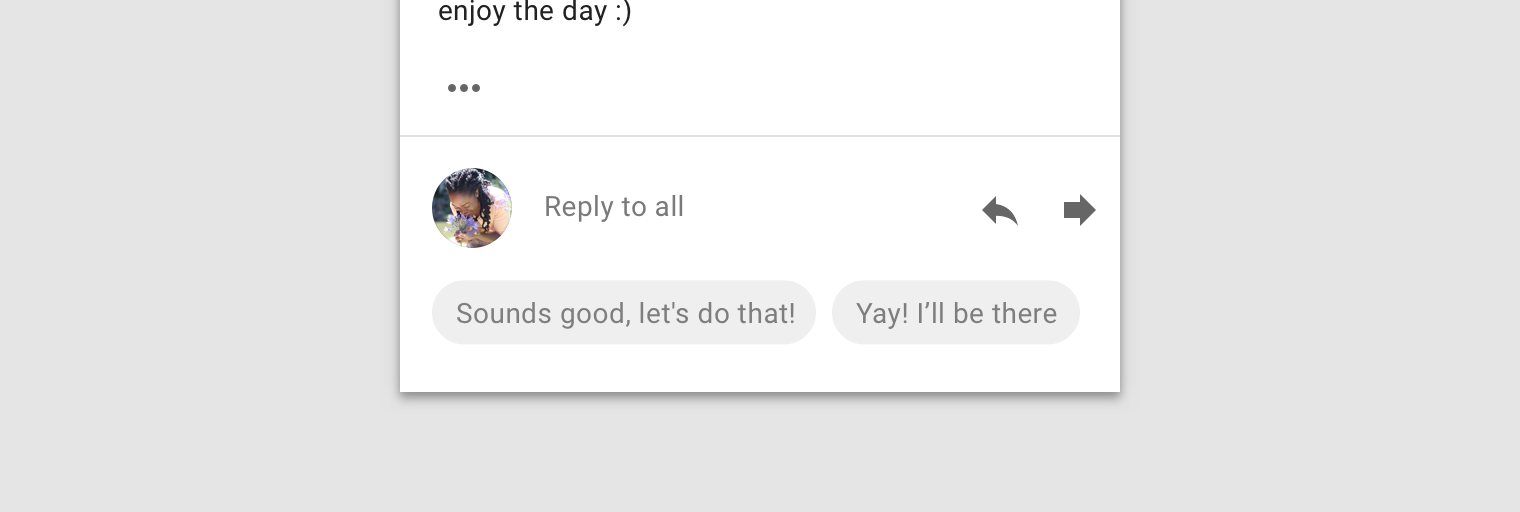
Chip chip = findViewById(R.id.chip);

chip.setLayoutDirection(View.LAYOUT\_DIRECTION\_LOCALE);

Chip has 4 types which are ***Entry*** Chips, ***Filter*** Chips, ***Choice*** Chips and ***Action*** Chips. These types are applied with Material styles.

**Entry Chips**

The main usage of this type of chip is to display complex information in a compact form like information about person or place. They can be used for taking inputs like in email. The text that is entered by the user can be transferred to the input chips. Or they can be used as suggested answers under the mail.

 Google Material Design

This type of chip can contain of a text label, an optional close icon, optional chip icon and optional checkmark. By default, they are checkable. The close icon and checkmark(when the chip is clicked) is present by default. They are usually used with ***ChipDrawable***(explained below). An entry chip’s text can be editable. Until the action is performed, the text can be changed. It can be editable, when it is clicked. Multiple entry chips can be placed in a single field. They can be aligned and scrolled horizontally. Or they can be transferred to new line. When an entry chip is tapped, it can expand to display more information. They can be placed where the input cursor is located, in a horizontal list or in a stacked list.

To create an entry chip, Material’s default entry style should be applied to the Chip. All attributes of Material Chips can be used and customized also in that type.

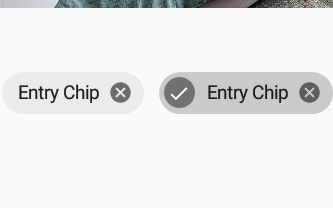
<android.support.design.chip.Chip

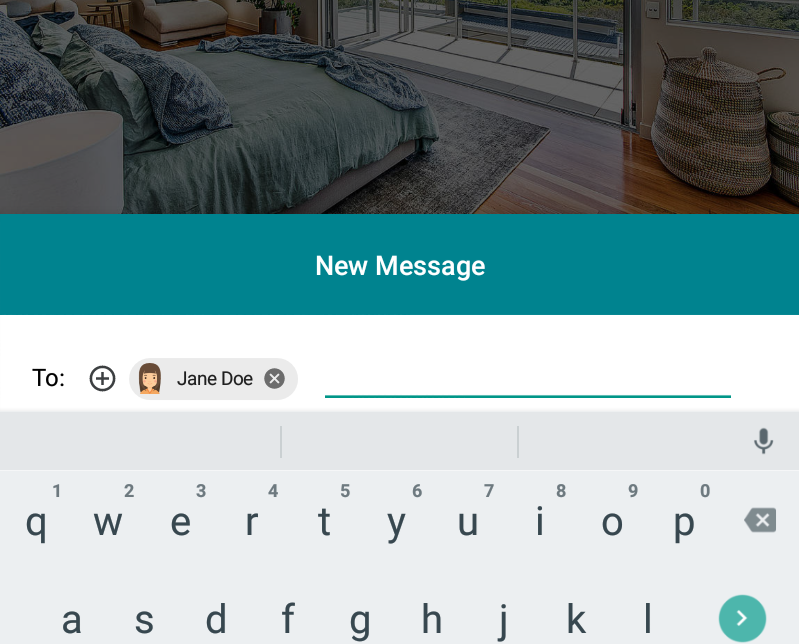
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

style="@style/Widget.MaterialComponents.Chip.Entry"

android:text="@string/entry\_chip"/>





**Filter Chips**

The main usage of this type of chip is to filter a collection. They contain a descriptive word from a group. It displays multiple options in a region. Multiple filter chips can be selected and deselected. It should not display only one option. They should be usually used with ChipGroup. It is an alternative to checkbox and toggle buttons. By default, they are checkable and contain a checkmark. When a filter chip is selected, check icon is appeared left of the text label. They can be chosen dynamically. An optional chip icon and optional close icon can be placed into the chip. This type is aligned horizontally and can be scrolled. They can be pass to the new line but if more than two lines will be occured, they should be scrolled horizontally. That type of chips can be listed under a search field.

To create a filter chip, Material’s default filter style should be applied to the Chip. All attributes of Material Chips can be used and customized also in that type.

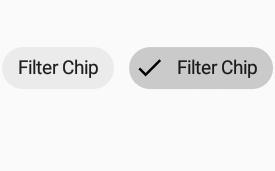
<android.support.design.chip.Chip

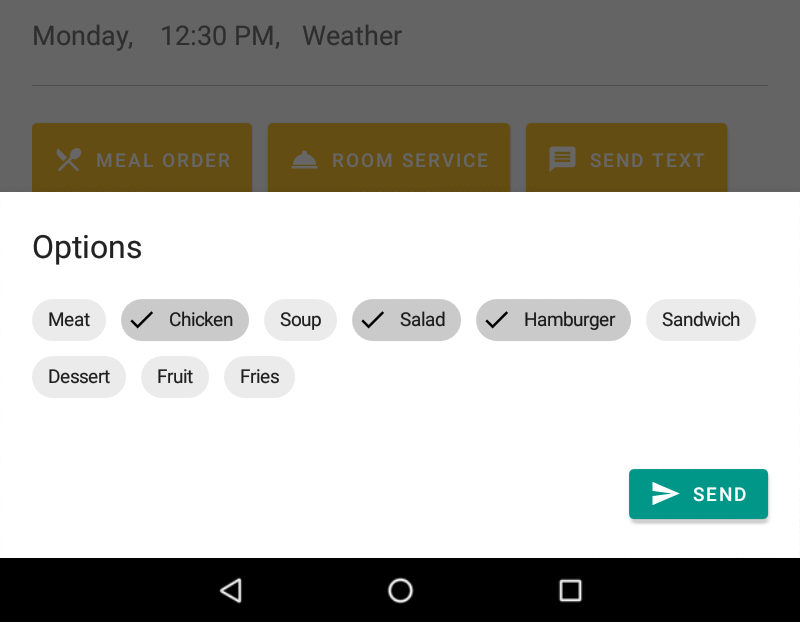
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

style="@style/Widget.MaterialComponents.Chip.Filter"

android:text="@string/filter\_chip"/>





**Choice Chips**

This type of chips is used for making a choice from a bunch of choice chips. They are checkable by default. An optional chip icon can be placed into the chip. It is an alternative to radio button, toggle button and single select menus. Choice chips save space rather than radio buttons and in choice chips, options are visible without any action unlike menus. They are used in the ***ChipGroups*** and at least two options should be displayed. Usually, they are aligned horizontally. The chosen one becomes darker. According to Material Design, only one of them can be selected in set of choice chips. When a chip is selected, if one of them was selected before, previous chip should be deselected. To ensure the single selection, ***android:singleSelection=”true”*** attribute is applied to the ChipGroup.

To create a choice chip, Material’s default choice style should be applied to the Chip. All attributes of Material Chips can be used and customized also in that type.

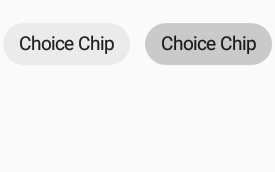
<android.support.design.chip.Chip

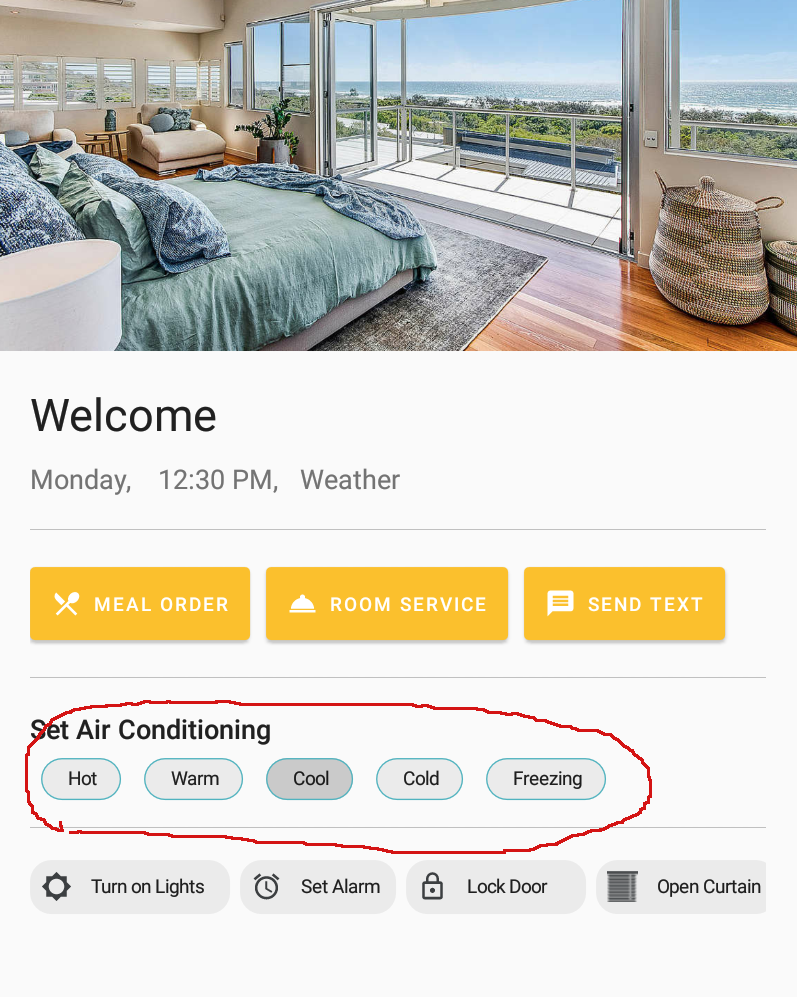
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/choice\_chip"

style="@style/Widget.MaterialComponents.Chip.Choice"/>





Choice Chips

**Action Chips**

This type of chips triggers the primary user actions. This is the default chip type. They are not checkable; therefore, they do not contain checkmark. Optional chip icon can be placed into the chip. They have only ripple effect; they do not become darker, when they are checked. They are an alternative to buttons. Main area of usages of that type of chip are displaying progress, confirmation or triggering an action. Action chips are placed under the primary content. This can be the bottom the Material Card or bottom of the screen. This type of chips should be displayed as a group and can scroll horizontally.

To create an action chip, Material’s default action style should be applied to the Chip. All attributes of Material Chips can be used and customized also in that type.

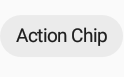
<android.support.design.chip.Chip

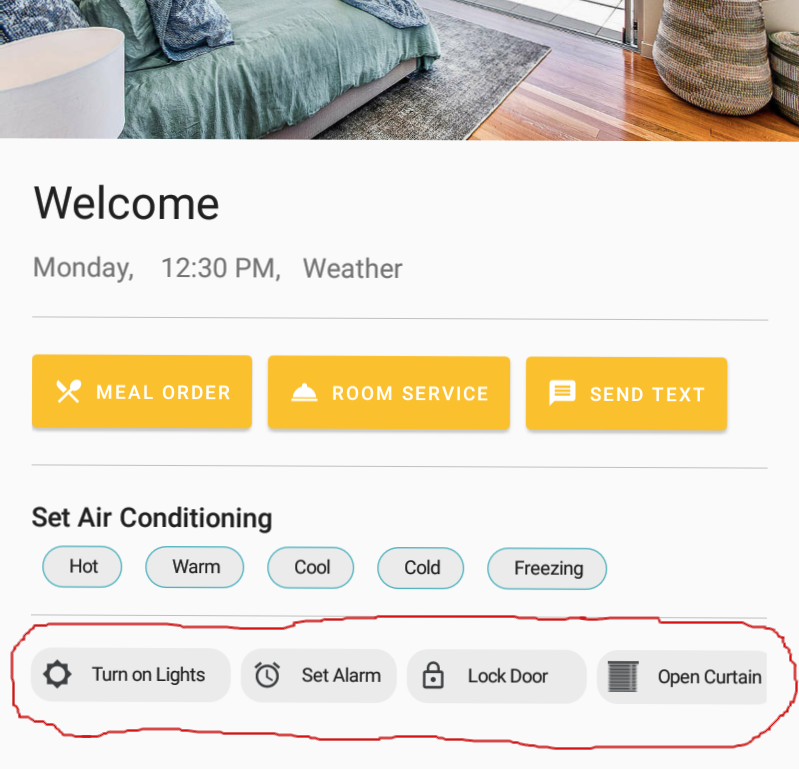
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/action\_chip"

style="@style/Widget.MaterialComponents.Chip.Action"/>





Action chips

**ChipGroup**

ChipGroup is a container for Chips. Chips are placed into the ChipGroups which are holds multiple chips. It can align the chips horizontally in a single line or transfer to the new line. By default, it transfers to the new line. But with ***app:singleLine=”true”*** attribute, all chips are displayed in a single line. To show all the chips, ChipGroup should wrap with ***HorizontalScrollView***. Another important feature of the ChipGroup is that enable the single selection in group of chips. If the ***app:singleSelection=”true”*** is applied to the ChipGroup, only one chip can be selected. When a chip is selected, if a selection was made before, selected chip’s selection is canceled. ChipGroup also provides space between chips. By default, it leaves 8dp spaces between chips. This area can be customized with ***chipSpacing***, ***chipSpacingHorizontal*** and ***chipSpacingVertical***. chipSpacing leaves space both in horizontal and vertical. chipSpacingHorizontal leaves only in horizontal. chipSpacingVertical leaves only in vertical. The other attribute is ***app:checkedChip***. This attribute specifies that which chip is selected by default in a ChipGroup.

XML declaration of ChipGroup:

<HorizontalScrollView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<android.support.design.chip.ChipGroup

android:id="@+id/chipAction"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:chipSpacing="18dp"

app:checkedChip="@id/chip2"

app:singleLine="true"

app:singleSelection="true">

<android.support.design.chip.Chip

style="@style/Widget.MaterialComponents.Chip.Choice"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/chip1" />

<android.support.design.chip.Chip

style="@style/Widget.MaterialComponents.Chip.Choice"

android:id="@+id/chip2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/chip2" />

<android.support.design.chip.Chip

style="@style/Widget.MaterialComponents.Chip.Choice"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/chip3" />

</android.support.design.chip.ChipGroup>

</HorizontalScrollView>

**ChipDrawable**

ChipDrawable can be used when a drawable is needed instead of Chip. It is usually used in autocomplete texts to place ChipDrawable using span into the text field. When creating a ChipDrawable, first an xml directory should be created under the res file. <chip tag must used in the file. The default type of chip is Entry for ChipDrawable, but other styles can be applied with style attribute. Also, other attributes of chip can be applied to the ChipDrawable.

<chip

xmlns:app="http://schemas.android.com/apk/res-auto"

style="@style/Widget.MaterialComponents.Chip.Entry"

android:text="@string/chip"/>

After creating the xml file, it should be inflated. Then, it acts like other Drawables.

ChipDrawable chip = ChipDrawable.createFromResource(getContext(), R.xml.chip);

chip.setBounds(left, right, top, bottom);

ImageSpan span = new ImageSpan(chip);

Editable text = editText.getText();

text.setSpan(span, 0, text.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

**Using Chips in Autocomplete Text**

In autocomplete texts, chips can be used. One of the alternatives is adding ChipGroup in front of the AutoCompleteTextView in XML file.

<android.support.design.chip.ChipGroup

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/chipGroup"

android:layout\_margin="8dp"

app:singleLine="true">

</android.support.design.chip.ChipGroup>

<AutoCompleteTextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/editText"/>

Then, dynamically add chips into the ChipGroup when the inputs are matched.

final Chip chipJoe = new Chip(view.getContext());

chip.setText(R.string.chip);

chip.setChipIconResource(R.drawable.chip\_icon);

final Chip chipJane = new Chip(view.getContext());

chip.setText(R.string.chip);

chip.setChipIconResource(R.drawable.chip\_icon);

final ChipGroup chipGroup = view.findViewById(R.id.chipGroup);

final AutoCompleteTextView editText = view.findViewById(R.id.editText);

final ArrayList<String> person = new ArrayList<>();

person.add("Jane Doe");

person.add("Joe Doe");

final ArrayAdapter<String> adapter = new ArrayAdapter<>(view.getContext(), android.R.layout.simple\_list\_item\_1, person);

editText.setAdapter(adapter);

editText.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

if("Jane Doe".equals(parent.getItemAtPosition(position))) {

chipGroup.addView(chipJane);

} else{

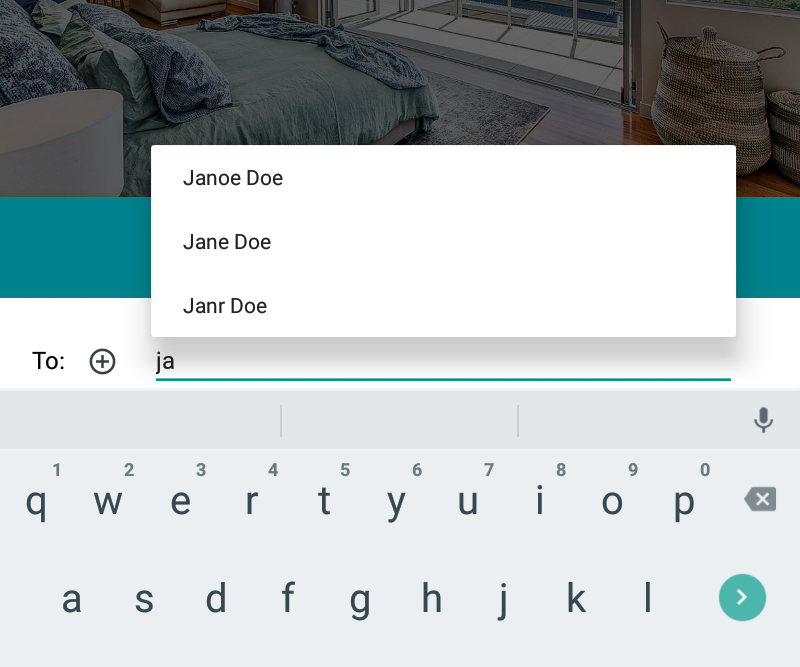
chipGroup.addView(chipJoe);

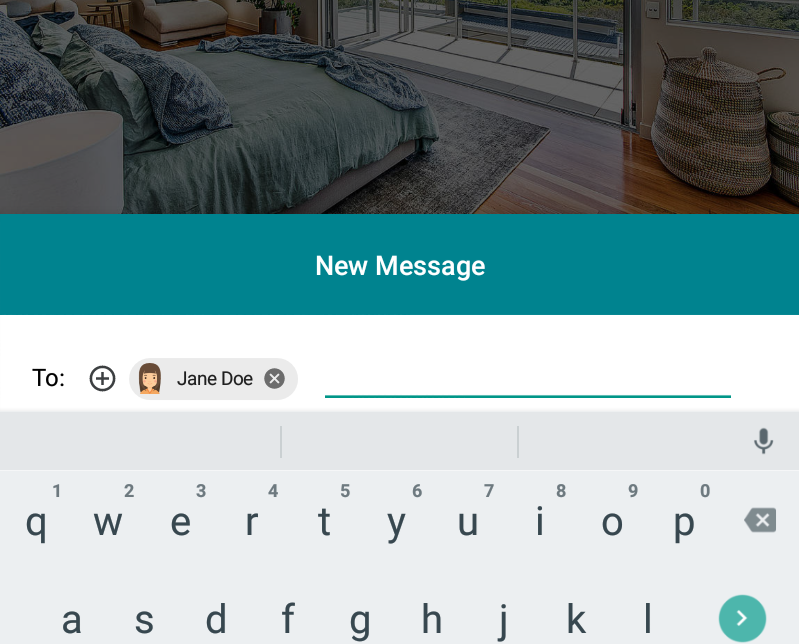
}

editText.setText("");

}

});





**Requirements for Implementation**

Chips are part of android design library. They are added in version 28. To use chips, you must add the following requirements to the build.gradle file in the project:

dependencies {  
    implementation ' com.android.support:appcompat-v7:28.0.0 -rc02'  
 implementation 'com.android.support: design:28.0.0 -rc02’ }

XML declaration of a simple chip:

<android.support.design.chip.Chip

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/text" />

**References**

(2018, May 14). Android P - What's New in Material Design – Noteworthy - The Journal Blog. Retrieved September 17, 2018, from https://blog.usejournal.com/android-p-changes-in-new-material-design-ffb61e6112f9

Android Developers. (n.d.). Retrieved June 27, 2018, from https://developer.android.com/

Bottom Sheet with Android Design Support Library. (2017, March 16). Retrieved August 16, 2018, from https://blog.iamsuleiman.com/bottom-sheet-android-design-support-library/

Homepage. (n.d.). Retrieved August 14, 2018, from https://material.io/

Karababa, F. (2018, July 22). Chips: Material Components for Android – Material Design in Action – Medium. Retrieved September 12, 2018, from https://medium.com/material-design-in-action/chips-material-components-for-android-46001664a40f