ASUS MIS Android App. UI Design Guideline_V1.0

MIS / Usage Interface Design 2014.03.12

Table of content

Version Check

Version No.	Date	Authoe	Contents
V1.0	12.March 2014	Ivy_Wang	Document Create

Overview

此規範為針對 Android 裝置原生的 App 為主的規範定義。

目的為協助 ASUS IT 在針對 Android 裝置開發的原生 App，提供設計與流程規劃時設計方針，除保有本身系統不同的特性與精神，使 IT App 結構上進一步統一操作邏輯、且具有系統系的視覺呈現，確保跨部門整合開發的一致性，幫助開發流程順利進行。

我們最大的宗旨在於提供良好與愉悅的使用經驗與操作流程。提供使用者無負擔的學習應用，增加操作易用性與效率性，強化設計能力，讓使用者對 ASUS IT App 有識別度與忠誠度。

因此，請別被此規範限制您的創新思維，我們給的是一個方向而非產出規定，異中求同、同中求異將是您最大的挑戰與價值。

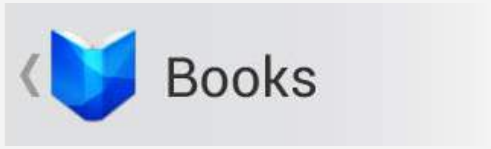
規範內容部份參考於 Android 開發規範
http://developer.android.com/design/patterns/index.html

Android Design Guideline

Navigation

Navigation with Back and Up

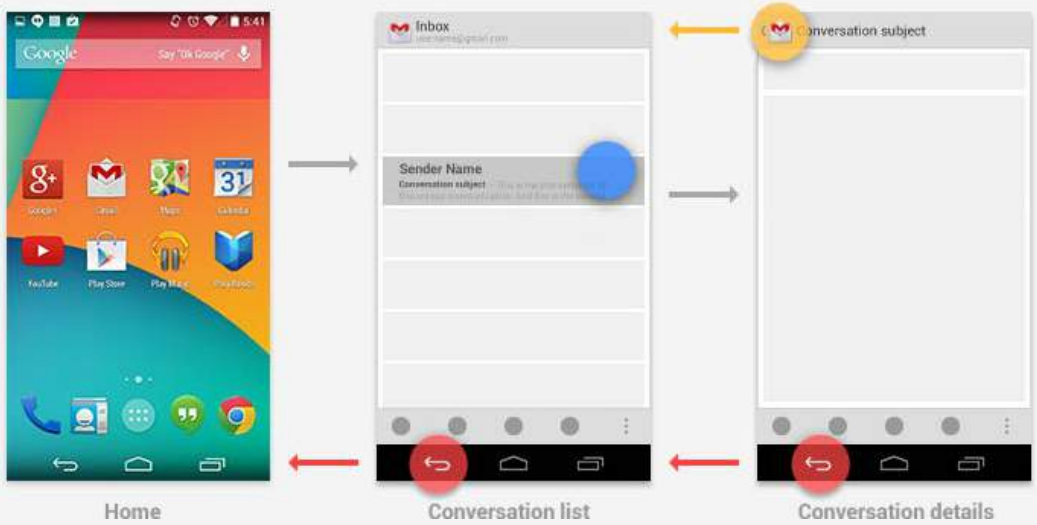
The System Back button for supporting navigation within an app.
The Up button, consisting of the app icon and a left-point caret.



Up vs. Back

The Up button is used to navigate within an app based on the hierarchical relationships between screens. For instance, if screen A displays a list of items, and selecting an item leads to screen B (which presents that item in more detail), then screen B should offer an Up button that returns to screen A.

The system Back button is used to navigate. It is generally based on the temporal relationships between screens, rather than the app's hierarchy.



The Back button also supports a few behaviors not directly tied to screen-to-screen navigation:

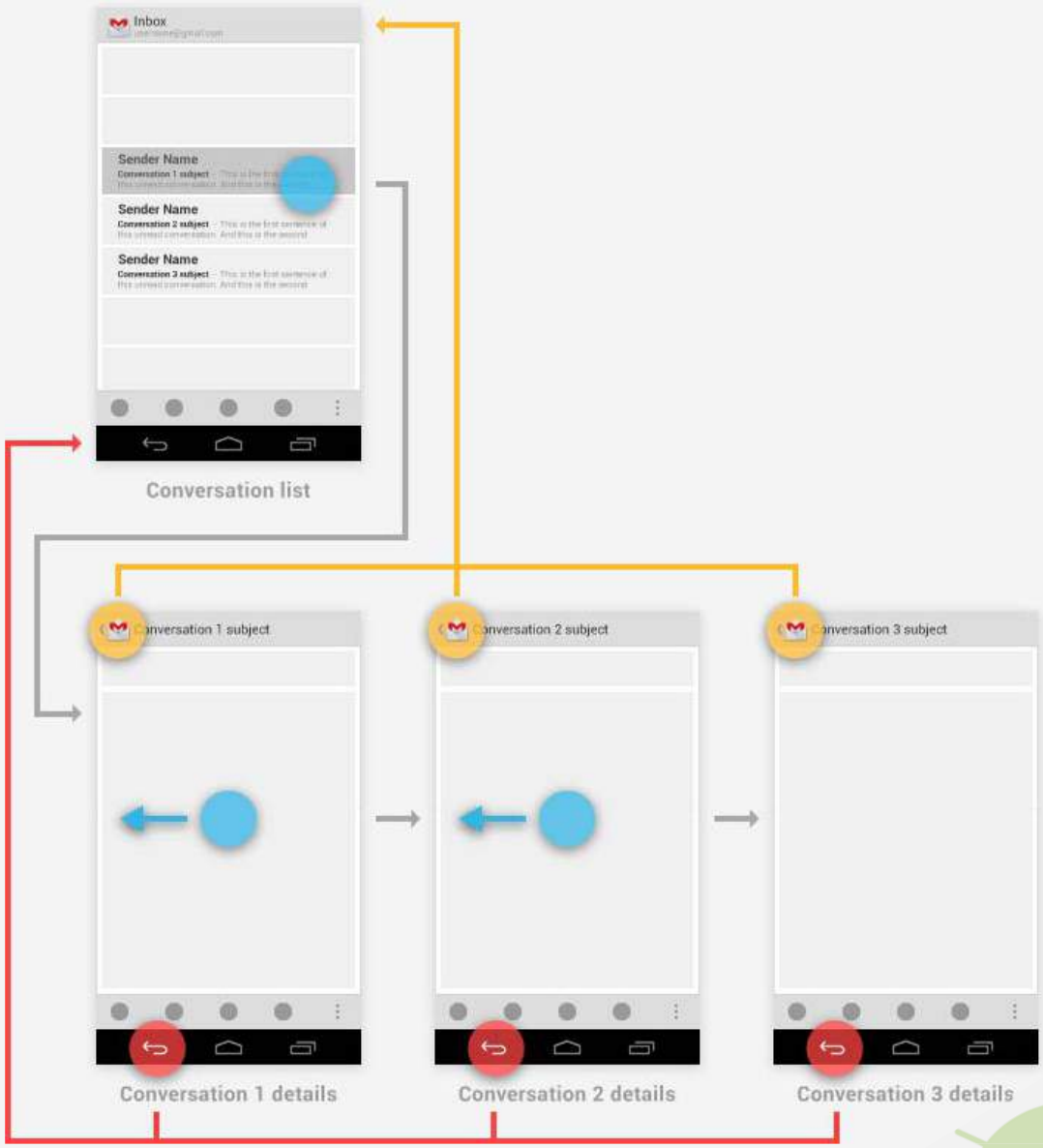
- Dismisses floating windows (dialogs, popups)
- Dismisses contextual action bars, and removes the highlight from the selected items

Navigation

Navigating between sibling screens

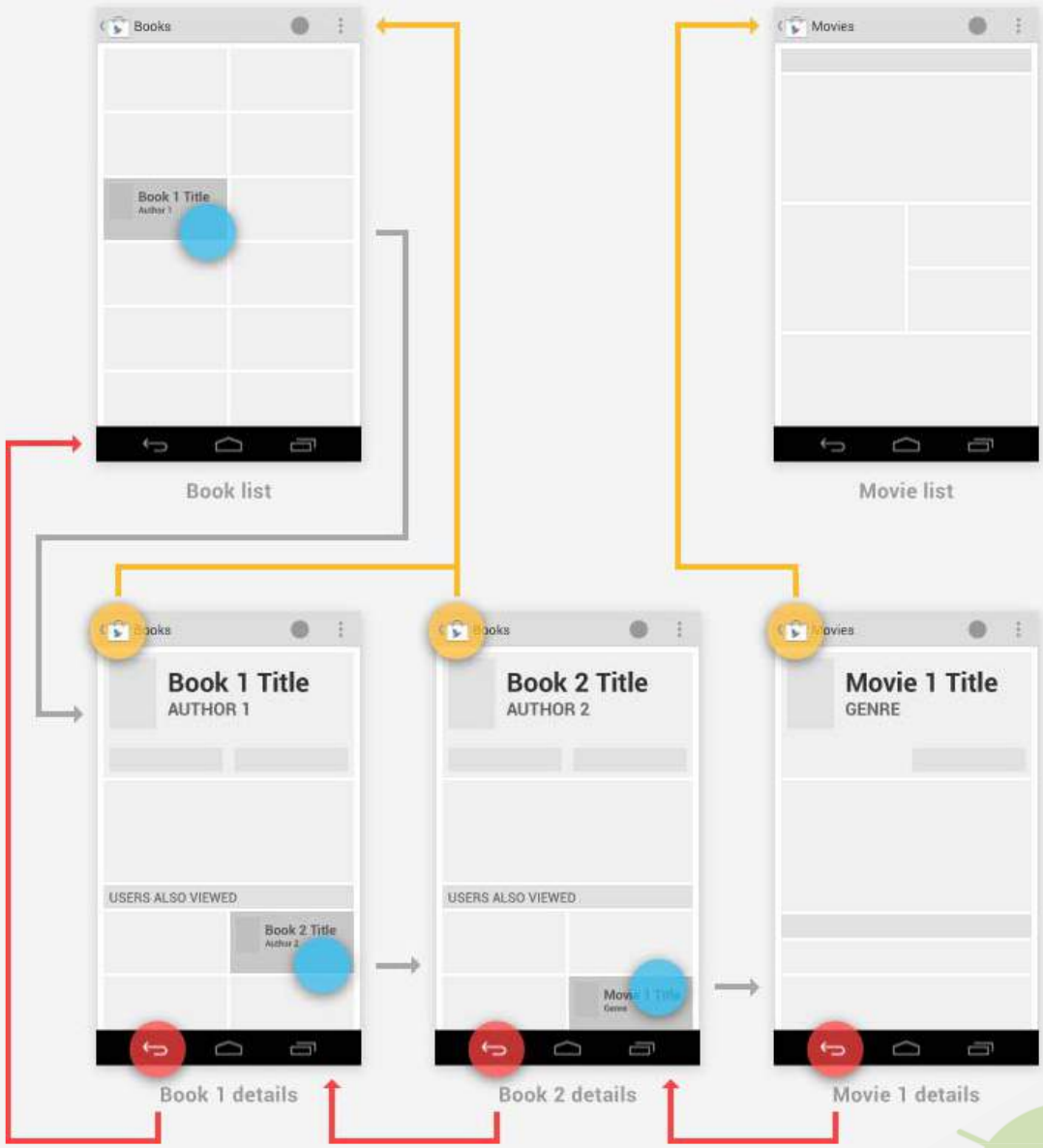
When your app supports navigation from a list of items to a detail view of one of those items, it's often desirable to support direction navigation from that item to another one which precedes or follows it in the list.

In Gmail, it's easy to swipe left or right from a conversation to view a newer or older one in the same Inbox. Just as when changing view within a screen, such navigation does not change the behavior of Up or Back.



Navigation

You have the ability to make the Up behavior even smarter based on your knowledge of detail view. Extending the Play Store example from above, imagine the user has navigated from the last Book viewed to the details for the Movie adaptation. In that case, Up can return to a container (Movies) which the user hasn't previously navigated through.

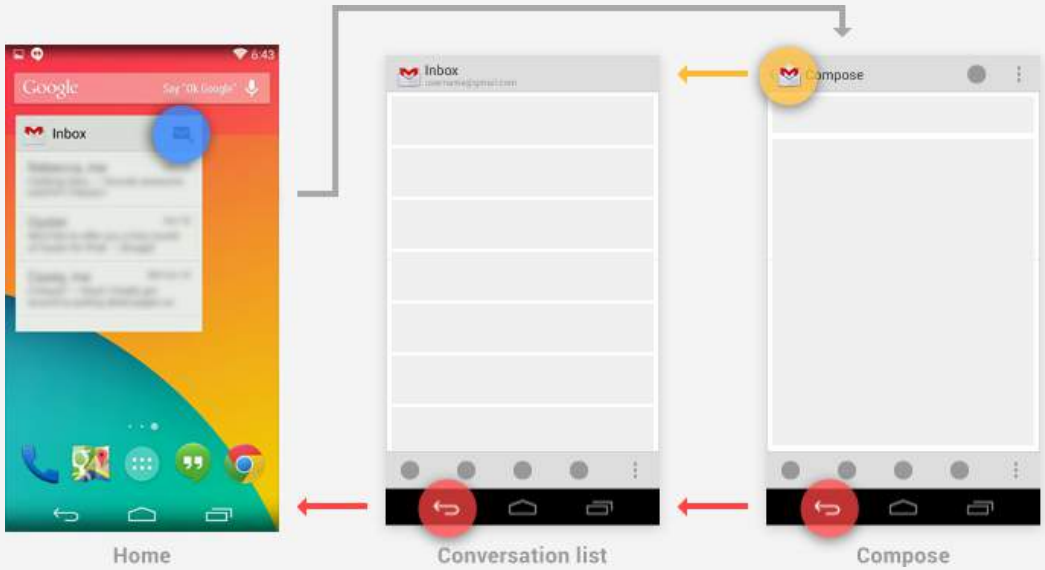


Navigation

Navigation into Your App via Home Screen Widgets and Notifications

Home screen widgets or notifications to help your users navigate directly to screens deep within your app's hierarchy.
For both of these cases, handle the Up button as follows:

- *If the destination screen is typically reached from one particular screen within your app*, Up should navigate to that screen.
- *Otherwise*, Up should navigate to the topmost ("Home") screen of your app.



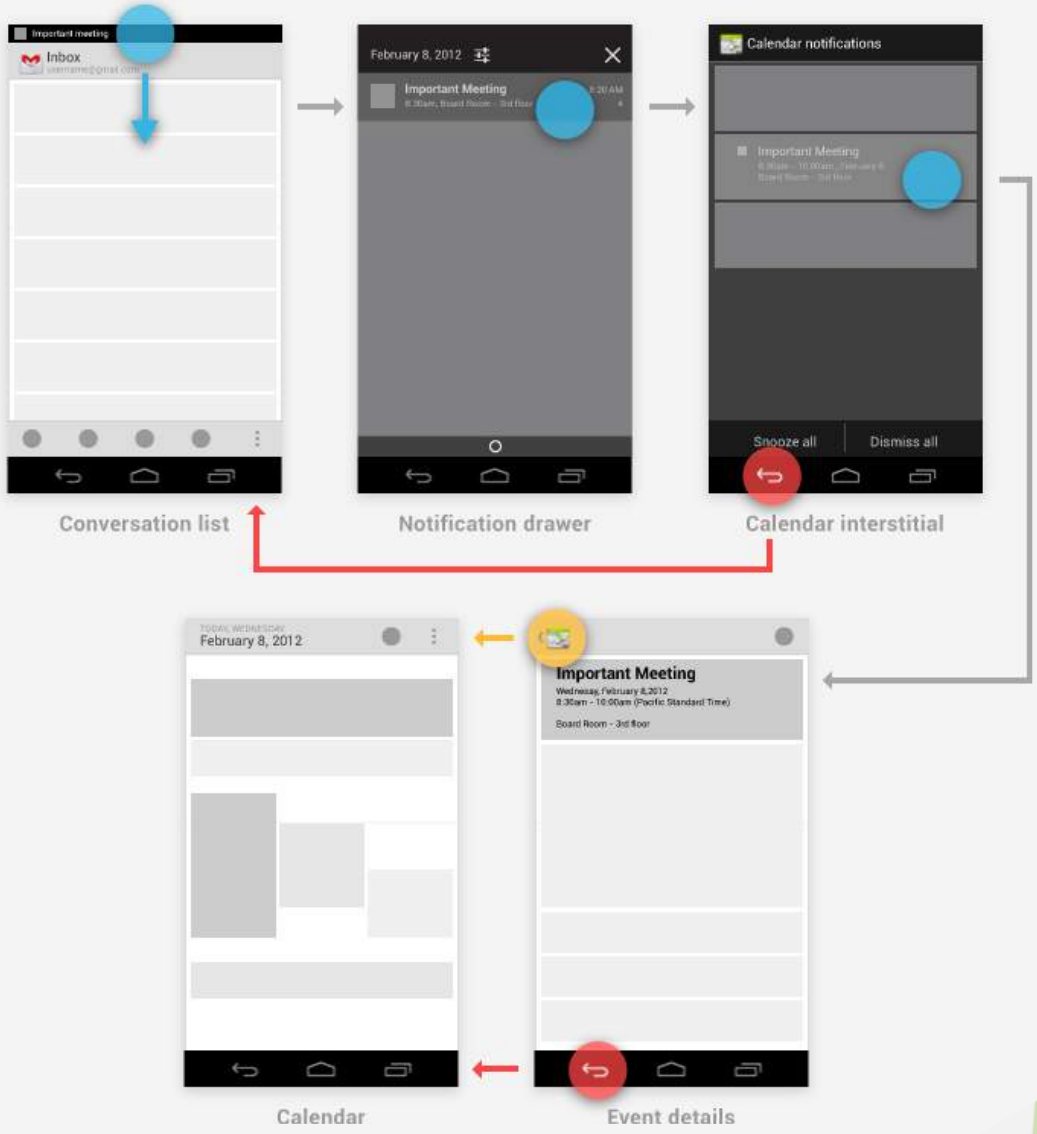
As an example, Gmail's Home screen widget has a button for diving directly to its compose screen. Up or Back from the compose screen would take the user to the Inbox, and from there the Back button continues to Home.

Navigation

Indirect notifications

When your app needs to present information about multiple events simultaneously, it can use a single notification that directs the user to an interstitial screen. This screen summarizes these events, and provides paths for the user to dive deeply into the app. Notifications of this style are called ***indirect notifications***.

For example, suppose a user in Gmail receives an indirect notification from Calendar. Touching this notification opens the interstitial screen, which displays reminders for several different events. Touching Back from the interstitial returns the user to Gmail. Touching on a particular event takes the user away from the interstitial and into the full Calendar app to display details of the event. From the event details, Up and Back navigate to the top-level view of Calendar.

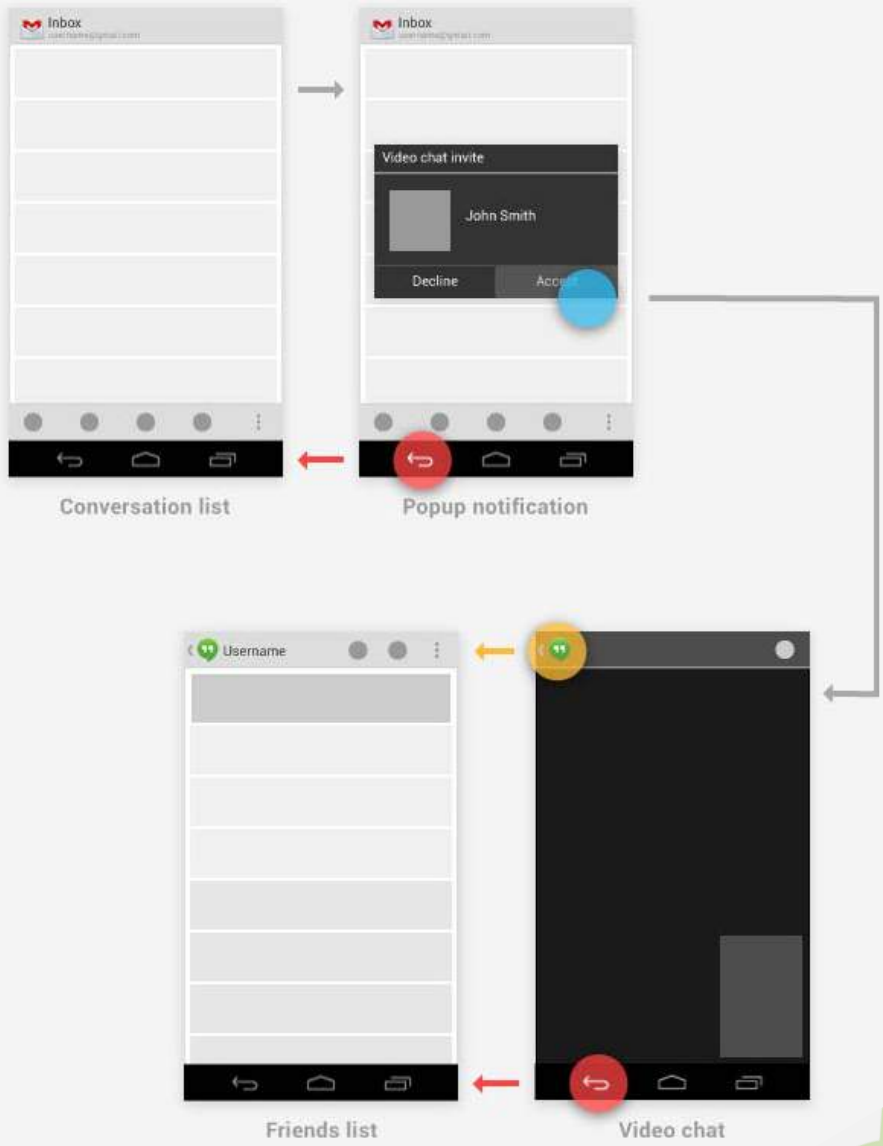


Navigation

Pop-up notifications

Pop-up notifications bypass the notification drawer, instead appearing directly in front of the user. They are rarely used, and should be reserved for occasions where a timely response is required and the interruption of the user's context is necessary.

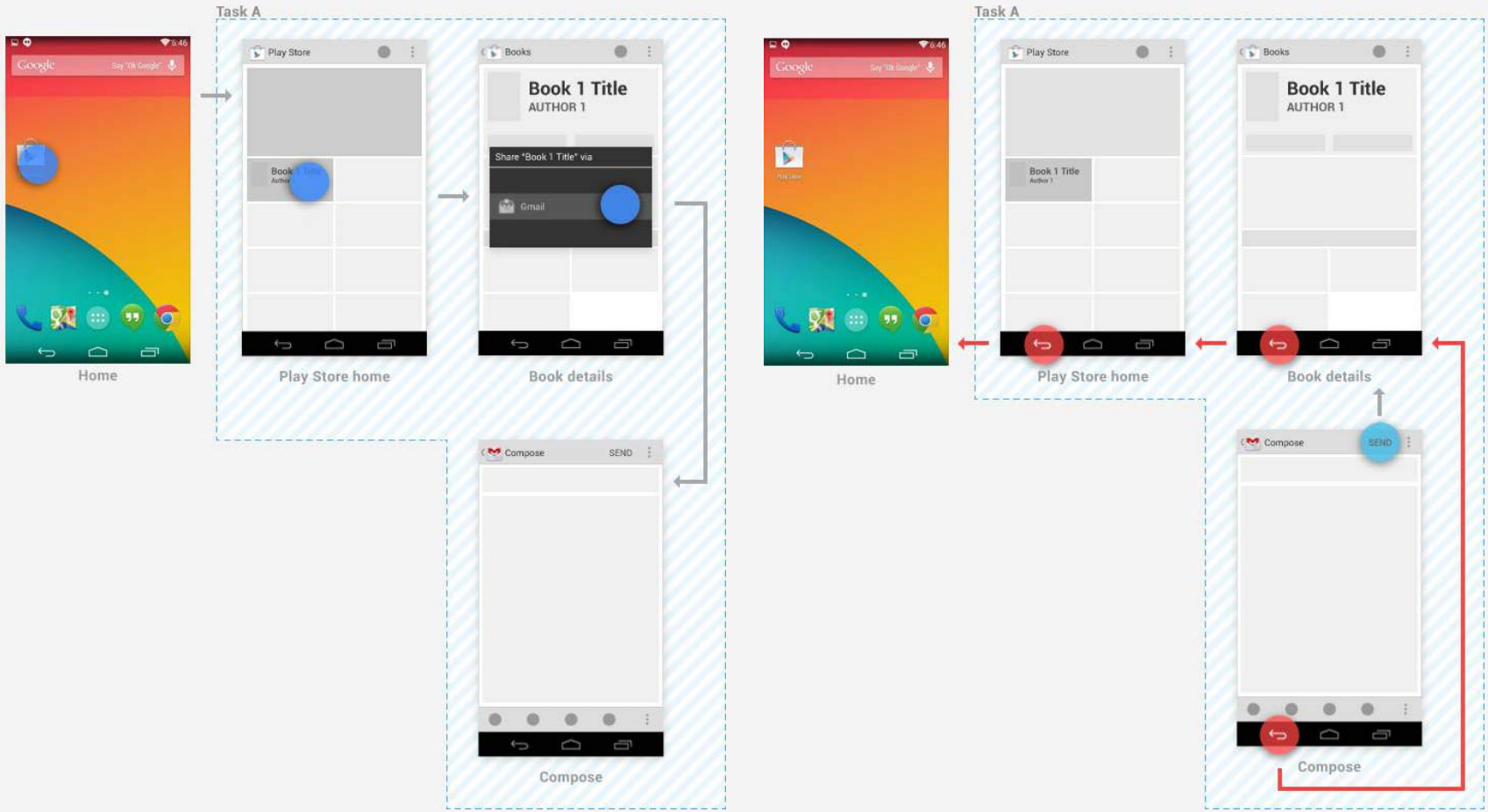
Talk uses this style to alert the user of an invitation from a friend to join a video chat, as this invitation will automatically expire after a few seconds.



Navigation

Navigation Between Apps

One of the fundamental strengths of the Android system is the ability for apps to activate each other, giving the user the ability to navigate directly from one app into another.



Multi-pane Layouts

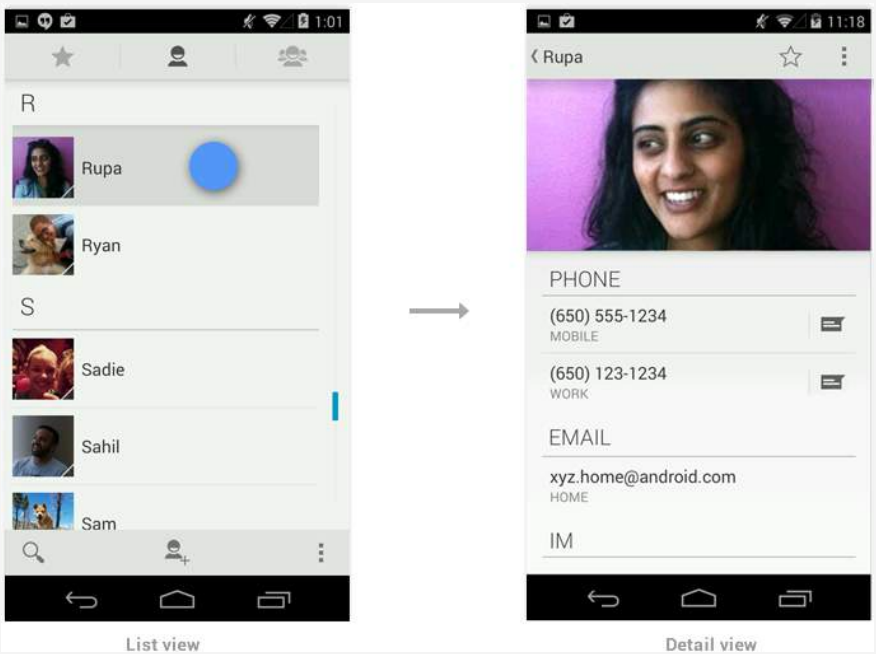
When writing an app for Android, keep in mind that Android devices come in many different screen sizes and types. Make sure that your app consistently provides a balanced and aesthetically pleasing layout by adjusting its content to varying screen sizes and orientations.

Combining Multiple Views Into One

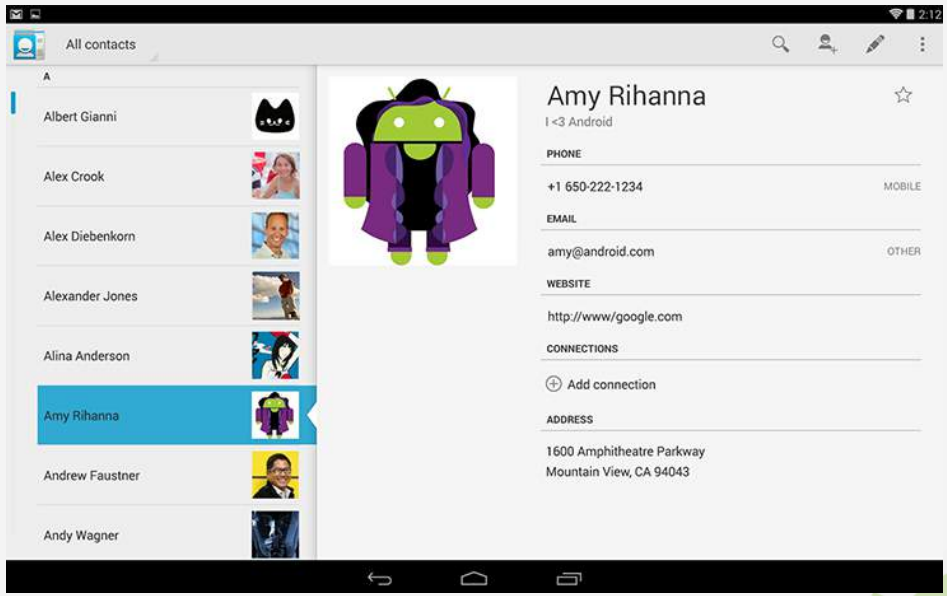
On smaller devices your content is typically divided into a master grid or list view and a detail view. Touching an item in the master view opens a different screen showing that item's detail information.

Because tablets have more screen real estate than phones, you can use panels to combine the related list and detail views into a single compound view. This uses the additional space more efficiently and makes navigating the app easier.

In general, use the pane on the right to present more information about the item you selected in the left pane. Make sure to keep the item in the left pane selected in order to establish the relationship between the panels.



List view Detail view



Multi-pane Layouts

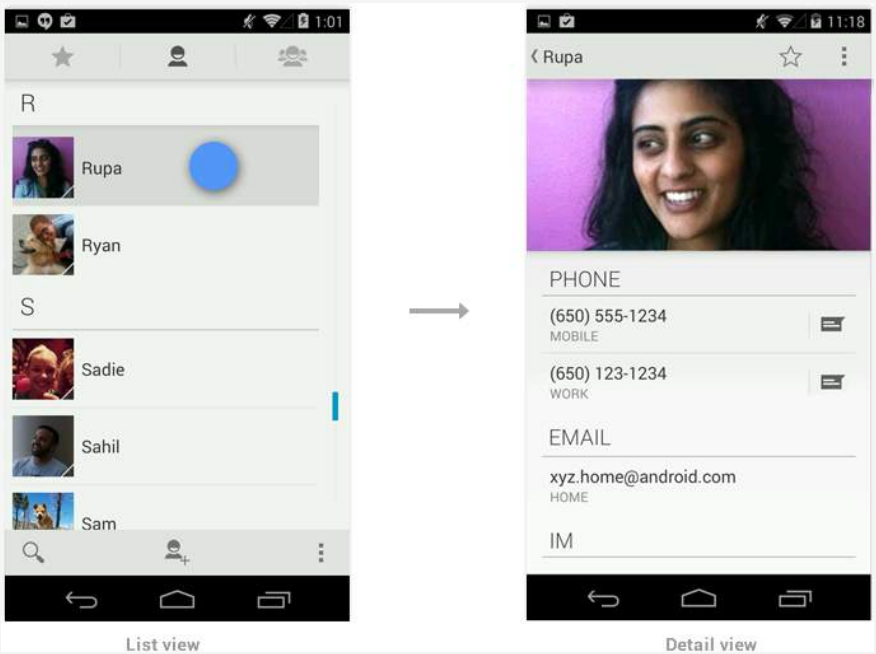
When writing an app for Android, keep in mind that Android devices come in many different screen sizes and types. Make sure that your app consistently provides a balanced and aesthetically pleasing layout by adjusting its content to varying screen sizes and orientations.

Combining Multiple Views Into One

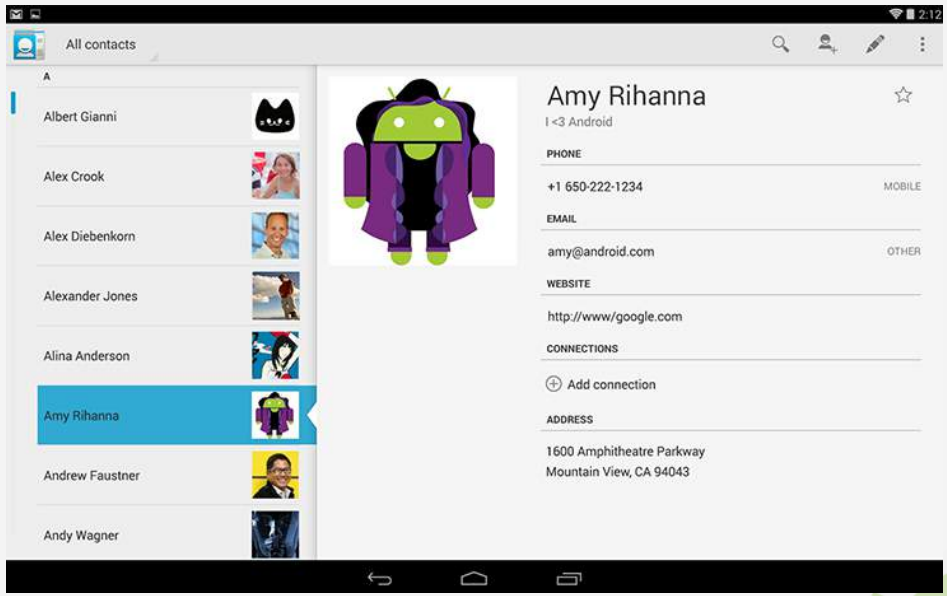
On smaller devices your content is typically divided into a master grid or list view and a detail view. Touching an item in the master view opens a different screen showing that item's detail information.

Because tablets have more screen real estate than phones, you can use panels to combine the related list and detail views into a single compound view. This uses the additional space more efficiently and makes navigating the app easier.

In general, use the pane on the right to present more information about the item you selected in the left pane. Make sure to keep the item in the left pane selected in order to establish the relationship between the panels.



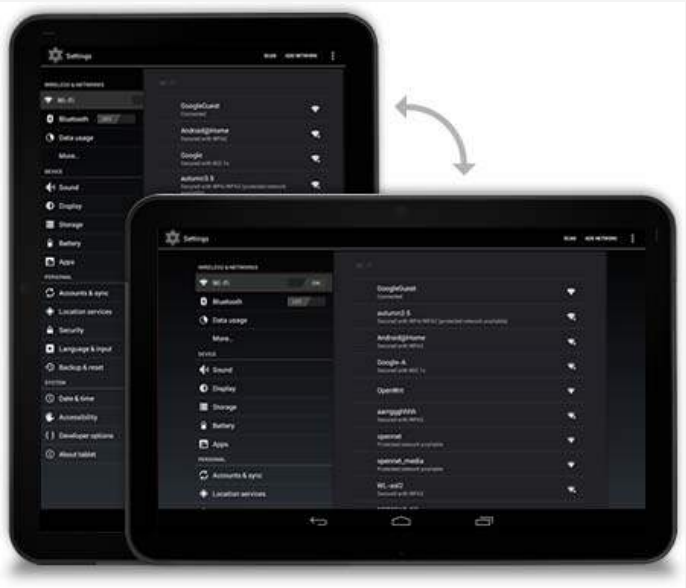
List view Detail view



Multi-pane Layouts

Compound Views and Orientation Changes

Screens should strive to have the same functionality regardless of orientation. If you use a compound view in one orientation, try not to split it up when the user rotates the screen. There are several techniques you can use to adjust the layout after orientation change while keeping functional parity intact.



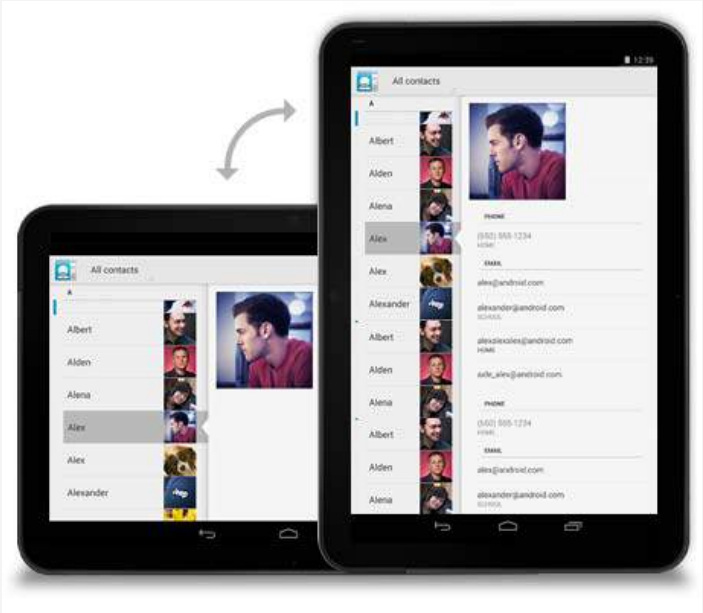
Stretch/compress

Adjust the column width of your left pane to achieve a balanced layout in both orientations.

Stack

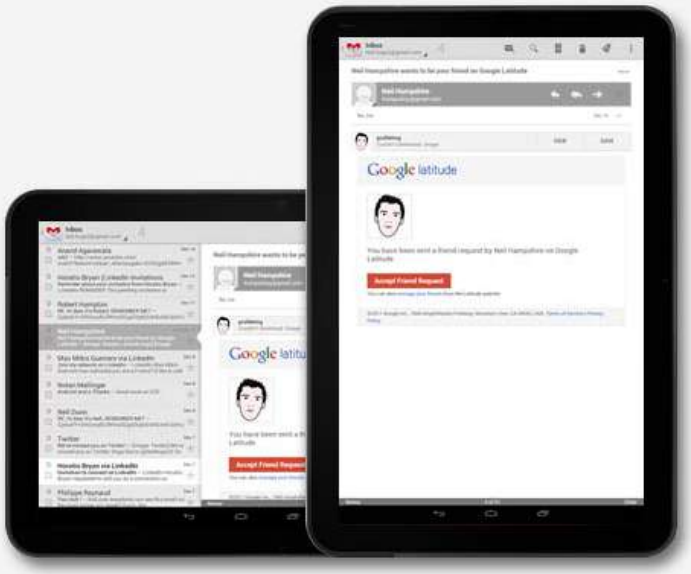
Rearrange the panels on your screen to match the orientation.

Multi-pane Layouts



Expand/collapse

When the device rotates, collapse the left pane view to only show the most important information.



Show/hide

If your screen cannot accommodate the compound view on rotation show the right pane in full screen view on rotation to portrait. Use the Up icon in action bar to show the parent screen.

Checklist

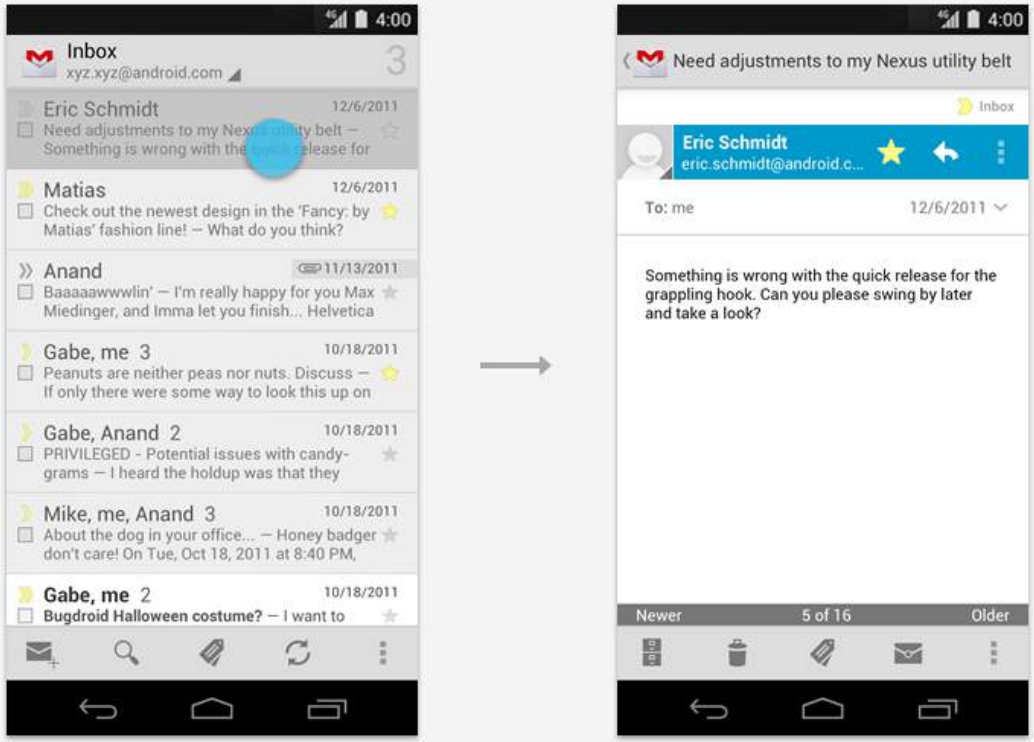
- Plan in advance on how your app scales to different screen sizes and screen orientations.
- Identify the most appropriate method for the panels in your compound views to reorganize themselves when screen orientation changes.
- Look for opportunities to consolidate your views into multi-panel compound views.
- Make sure that your screens try to provide functional parity after the screen orientation changes.

Swipe Views

Efficient navigation is one of the cornerstones of a well-designed app. While apps are generally built in a hierarchical fashion, there are instances where horizontal navigation can flatten vertical hierarchies and make access to related data items faster and more enjoyable. Swipe views allow the user to efficiently move from item to item using a simple gesture and thereby make browsing and consuming data a more fluent experience.

Swiping Between Detail Views

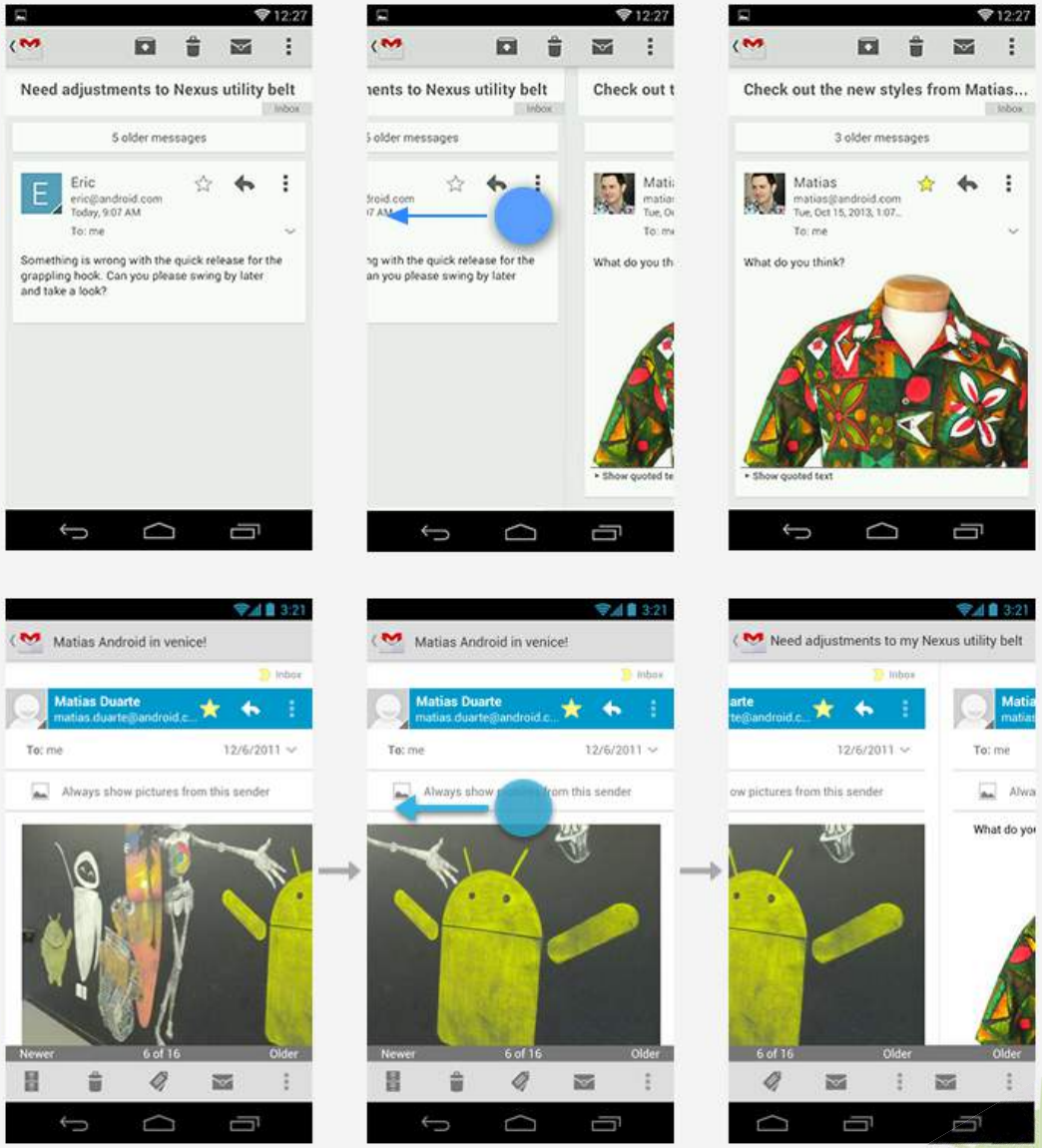
An app's data is often organized in a master/detail relationship: The user can view a list of related data items, such as images, chats, or emails, and then pick one of the items to see the detail contents in a separate screen.



Swipe Views

Navigating between consecutive email messages using the swipe gesture. If a view contains content that exceeds the width of the screen such as a wide email message, make sure the user's initial swipes will scroll horizontally within the view. Once the end of the content is reached, an additional swipe should navigate to the next view. In addition, support the use of edge swipes to immediately navigate between views when content scrolls horizontally.

Scrolling within a wide email message using the swipe gesture before navigating to the next message.



Notifications

New in Jelly Bean

In Jelly Bean, notifications received their most important structural and functional update since the beginning of Android.

- Notifications can include actions that enable the user to immediately act on a notification from the notification drawer.
- Notifications are now more flexible in size and layout. They can be expanded to show additional information details.
- A priority flag was introduced that helps to sort notifications by importance rather than time only.

Anatomy of a notification

Base Layout

At a minimum, all notifications consist of a base layout, including:

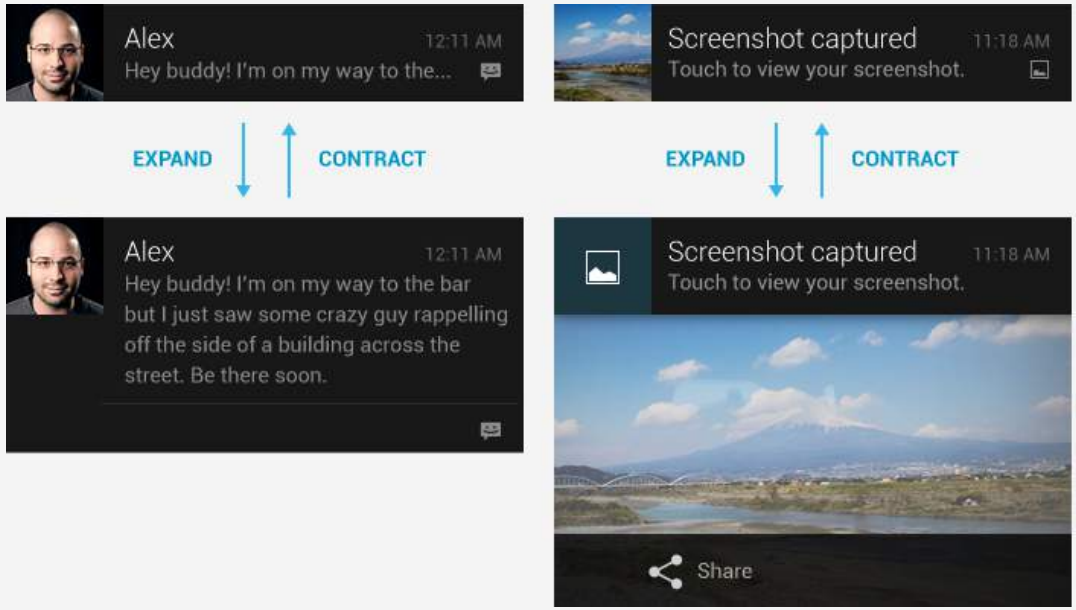
- the sending application's notification icon or the sender's photo
- a notification title and message
- a timestamp
- a secondary icon to identify the sending application when the senders image is shown for the main icon

The information arrangement of the base layout has not changed in Jelly Bean, so app notifications designed for versions earlier than Jelly Bean still look and work the same.

Notifications

Expanded layouts

With Jelly Bean you have the option to provide more event detail. You can use this to show the first few lines of a message or show a larger image preview. This provides the user with additional context, and - in some cases - may allow the user to read a message in its entirety. The user can pinch-zoom or two-finger glide in order to toggle between base and expanded layouts. For single event notifications, Android provides two expanded layout templates (text and image) for you to re-use in your application.



Actions

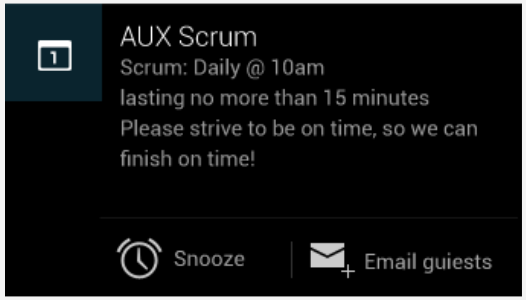
Starting with Jelly Bean, Android supports optional actions that are displayed at the bottom of the notification. With actions, users can handle the most common tasks for a particular notification from within the notification shade without having to open the originating application. This speeds up interaction and, in conjunction with "swipe-to-dismiss", helps users to streamline their notification triaging experience.

Good candidates for actions on notifications are actions that are:

- essential, frequent and typical for the content type you're displaying
- time-critical
- not overlapping with neighboring actions

Avoid actions that are:

- ambiguous
- duplicative of the default action of the notification (such as "Read" or "Open")

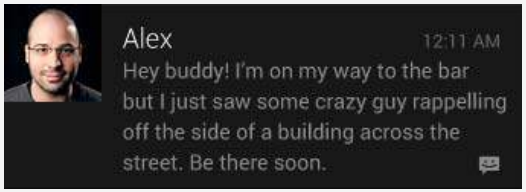


Notifications

Design guidelines

Make it personal

For notifications of items sent by another user (such as a message or status update), include that person's image.



Remember to include the app icon as a secondary icon in the notification, so that the user can still identify which app posted it.

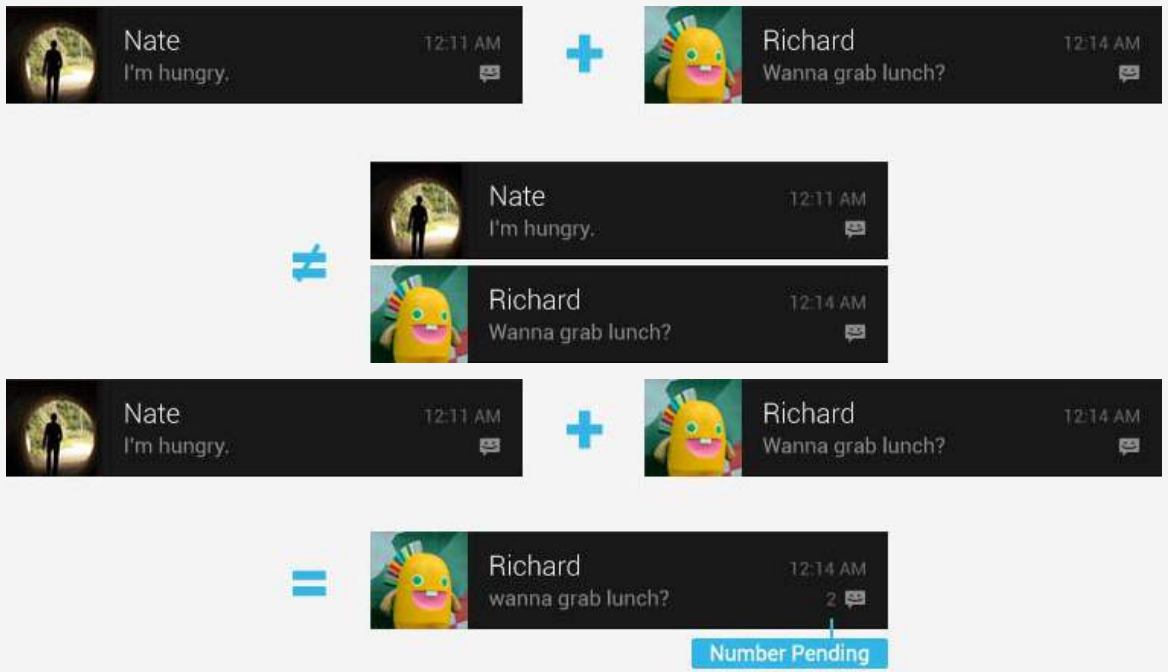
Correctly set and manage notification priority

Starting with Jelly Bean, Android now supports a priority flag for notifications. It allows you to influence where your notification will appear in comparison to other notifications and help to make sure that users always see their most important notifications first. You can choose from the following priority levels when posting a notification:

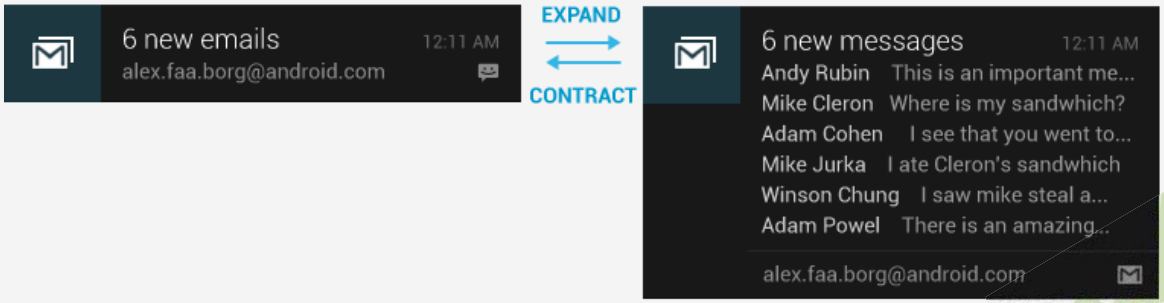
Notifications

Stack your notifications

If your app creates a notification while another of the same type is still pending, avoid creating an altogether new notification object. Instead, stack the notification.
A stacked notification builds a summary description and allows the user to understand how many notifications of a particular kind are pending.



You can provide more detail about the individual notifications that make up a stack by using the expanded digest layout. This allows users to gain a better sense of which notifications are pending and if they are interesting enough to be read in detail within the associated app.

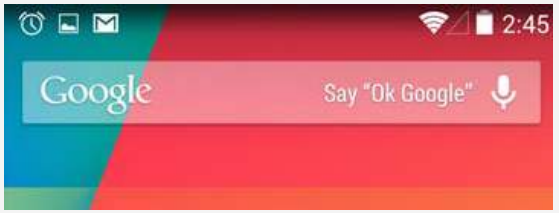


Notifications

Interacting With Notifications

Notifications are indicated by icons in the notification area and can be accessed by opening the notification drawer.

Inside the drawer, notifications are chronologically sorted with the latest one on top. Touching a notification opens the associated app to detailed content matching the notification. Swiping left or right on a notification removes it from the drawer.

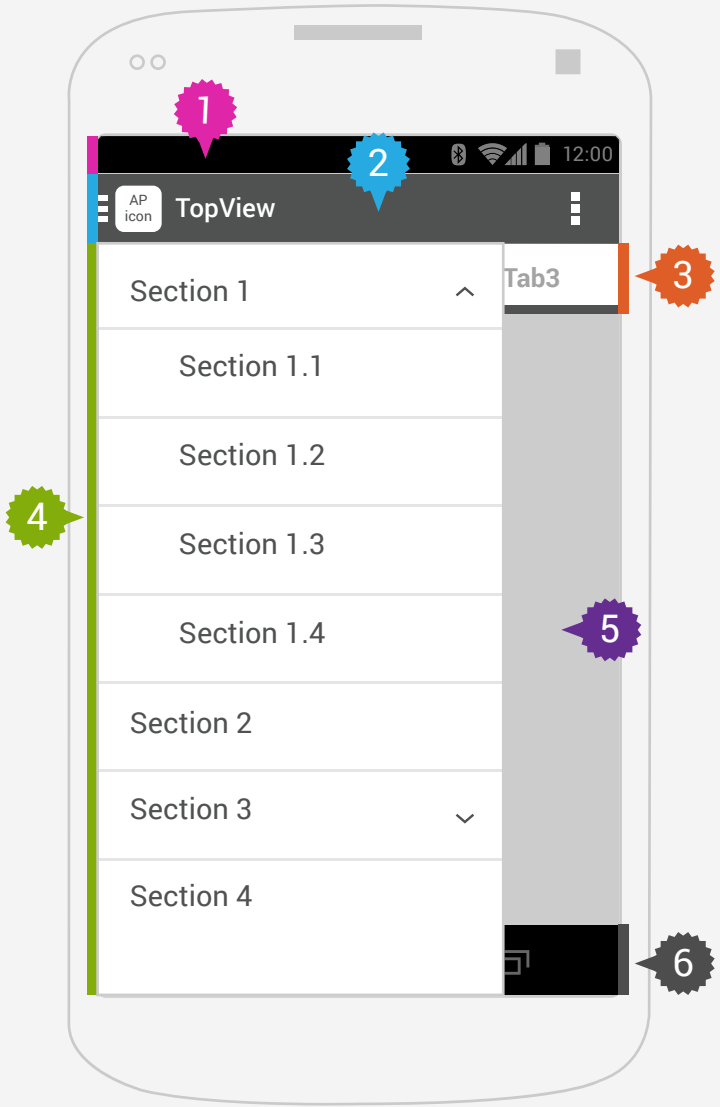


Ongoing notifications

Ongoing notifications keep users informed about an ongoing process in the background. For example, music players announce the currently playing track in the notification system and continue to do so until the user stops the playback. They can also be used to show the user feedback for longer tasks like downloading a file, or encoding a video. Ongoing notifications cannot be manually removed from the notification drawer.

ASUS MIS UI Patterns

Common App UI



#1. Status Bar

#2. Main Action Bar

The command and control center for your app. The action bar surfaces the most important actions for the current view, and may include simple controls for switching between views.

#3. Action Bar Tabs

#4. Navigation Drawer

If your app's structure is more complex, the navigation drawer can display the main navigation options. The navigation drawer expands from the left edge of the screen, overlaying the content & tabs area but not the action bar.

#5. Content Display

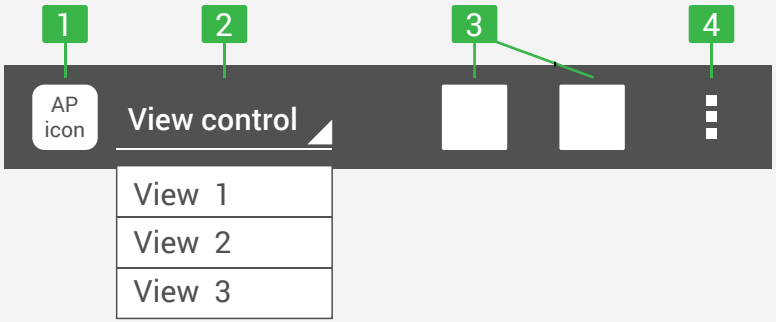
The space where the content of your app is displayed.

#6. Navigation Bar

* UI patterns should represent sections at the corrent level of navigation hierarchy. Is: 4 > 2 > 3 > 5

Action Bar

General Organization



1. App icon

The app icon establishes your app's identity. It can be replaced with a different logo or branding if you wish. Important: If the app is currently not displaying the top-level screen, be sure to display the Up caret to the left of the app icon, so the user can navigate up the hierarchy.

2. View control

If your app displays data in different views, this segment of the action bar allows users to switch views. Examples of view-switching controls are drop-down menus or tab controls.

If your app doesn't support different views, you can also use this space to display non-interactive content, such as an app title or longer branding information.

完整顯示

字數太長用省略符號...

3. Action buttons

Show the most important actions of your app in the actions section. Actions that don't fit in the action bar are moved automatically to the action overflow. Long-press on an icon to view the action's name.

4. Action overflow

Move less often used actions to the action overflow.

Action buttons order

From Left to Right:
Left area (AP icon →View control)
Right area (Search →Sorting →Function icon →Menu button)

Action Bar

Action buttons 使用原則

For guidance on prioritizing actions, use the F > I > T scheme.

F – Frequent 頻繁

I – Important 重要

T – Typical 典型

Action buttons 數量上限

小於360 dp = 2 icons

360-499 dp = 3 icons

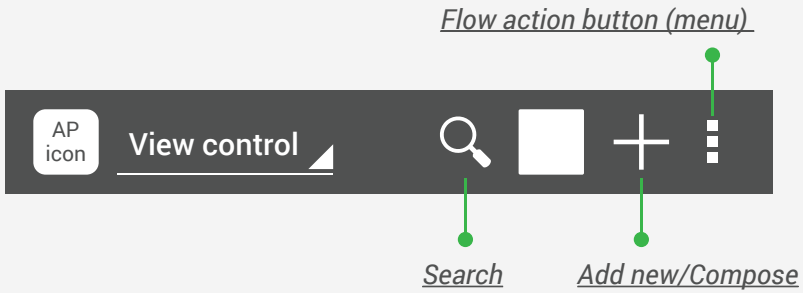
500-599 dp = 4 icons

600 dp 以上 = 5 icons

Device	Orientation	Horz. Dp	Icons	Example
Nexus S	Portrait	320	2	oo
Galaxy Nexus	Portrait	360	3	oo=
Nexus S	Landscape	534	4	oooo
7" Tablet	Portrait	600	5	oooo=
Galaxy Nexus	Landscape	640	5	oooo=
10" Tablet	Portrait	800	5	oooo=
7" Tablet	Landscape	1024	5	oooo=
10" Tablet	Landscape	1280	5	oooo=

上圖o代表Action , = 代表Overflow icon

Search / Add action buttons 位置



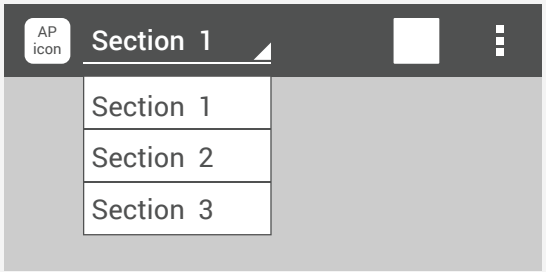
- *Search* is always the first left one action button on action bar.
- *Add new / Compose* is always on the left side of action flow(menu) button.

Action Bar

1. Spinners

Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one.

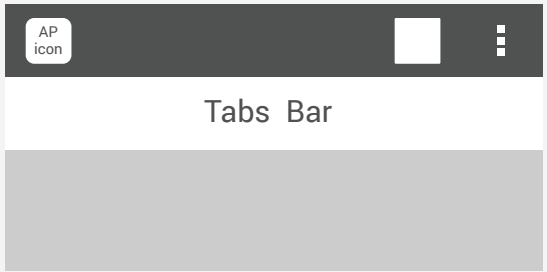
Action bar - Spinner



2. Tabs

Tabs in the action bar make it easy to explore and switch between different views or functional aspects of your app, or to browse categorized data sets.

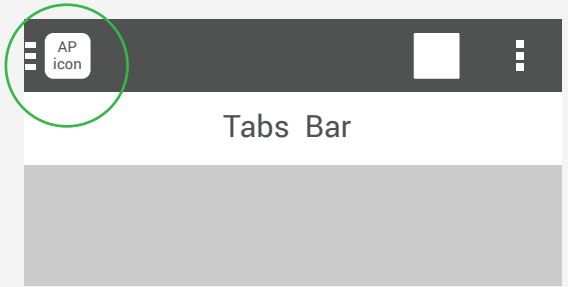
Action bar - Tabs



3. Drawers

The navigation drawer is a panel that transitions in from the left edge of the screen and displays the app's main navigation options.

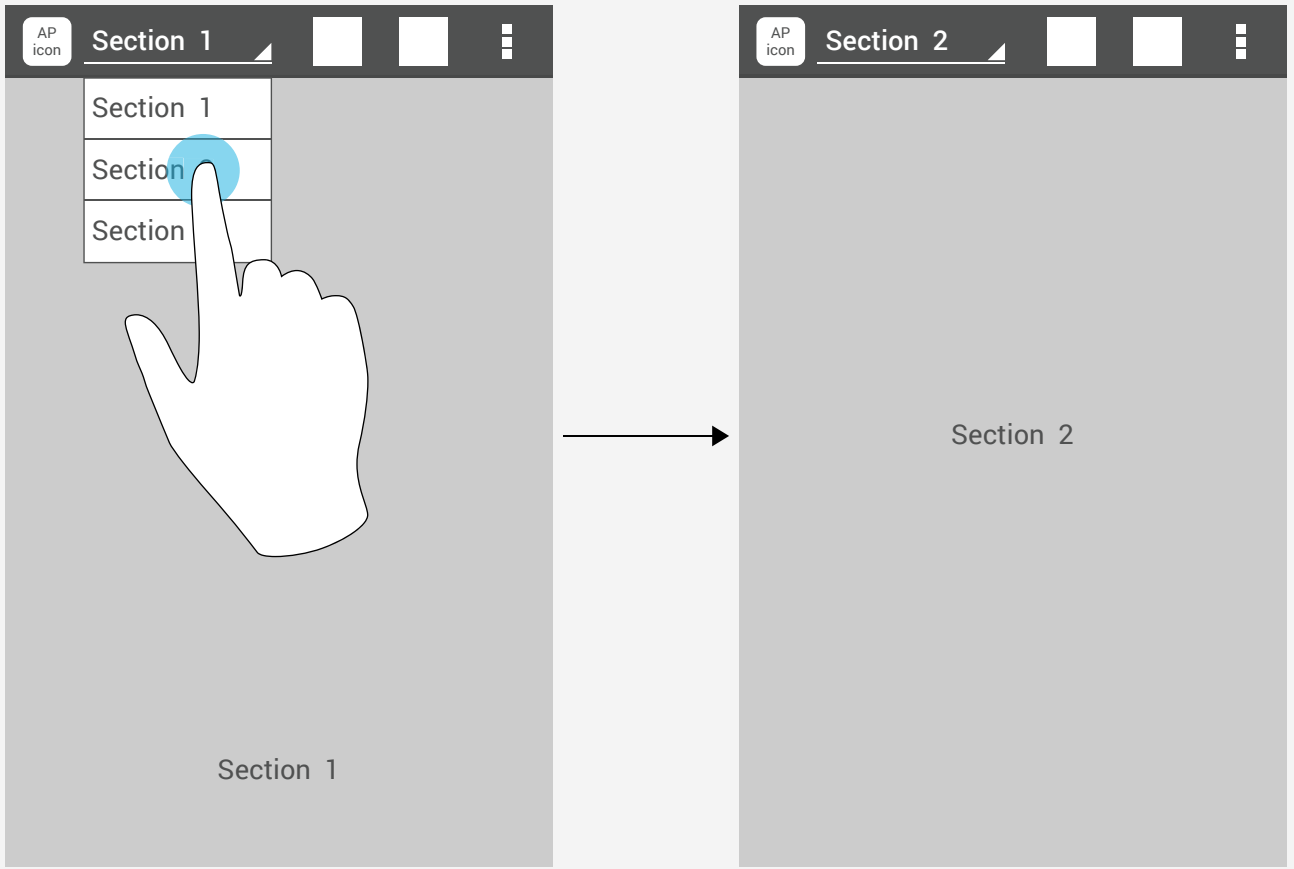
Action bar - drawer icon



Action Bar

View Controls_ 1.Spinners in action bars

Use spinners in action bars to switch views. For example, Gmail uses a spinner to permit switching between accounts or commonly used labels. Spinners are useful when changing the view is important to your app, but not necessarily a frequent occurrence. In cases where view switching is frequent, use tabs.



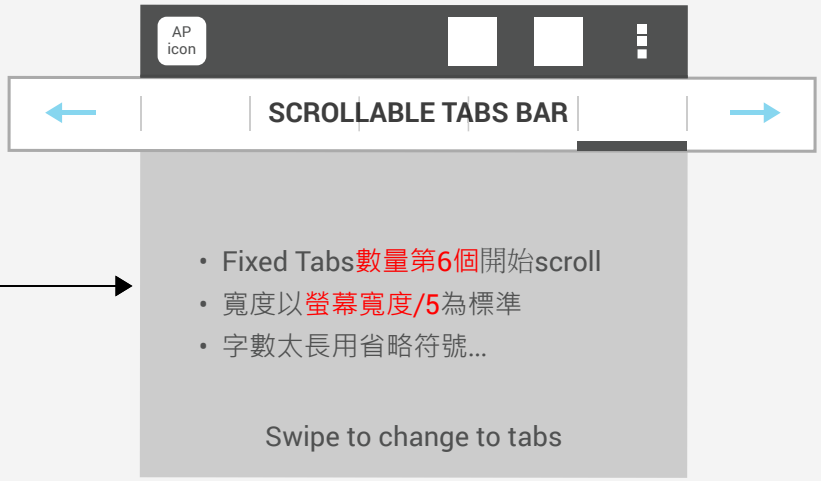
Action Bar

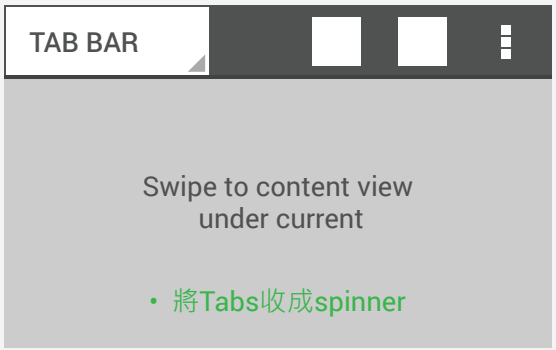
View Controls_ 2.Tabs

Fixed tabs display all items concurrently. To navigate to a different view, touch the tab, or swipe left or right.
Fixed tabs are displayed with equal width, based on the width of the widest tab label. If there is insufficient room to display all tabs, the tab labels themselves will be scrollable.

1. Fixed Tabs

2. Scrolling tab



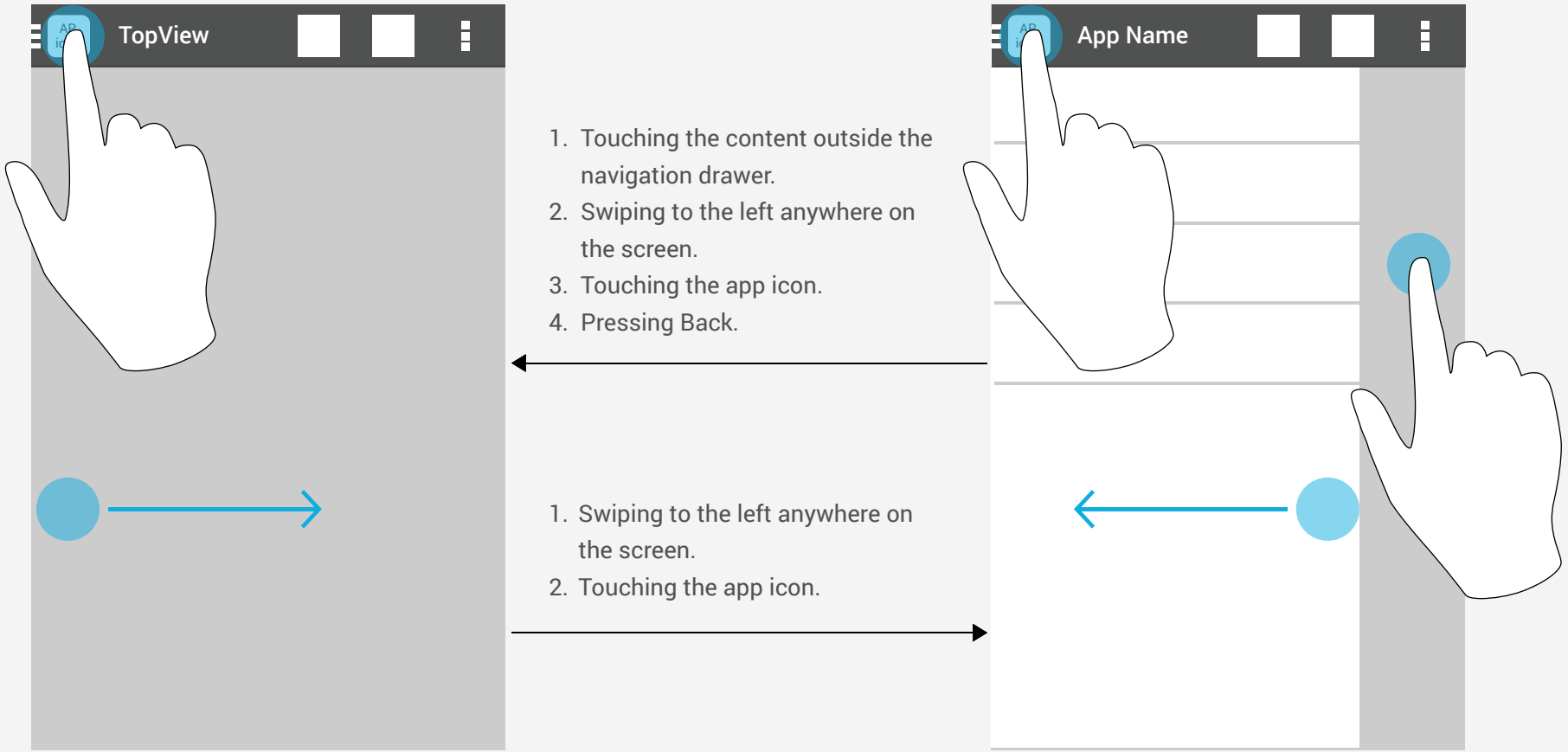


Action Bar

View Controls_ 3. Drawers

Here are some examples of where navigation drawers work best:

- More than 3 top-level views
- Cross-navigation from lower levels
- Deep navigation branches

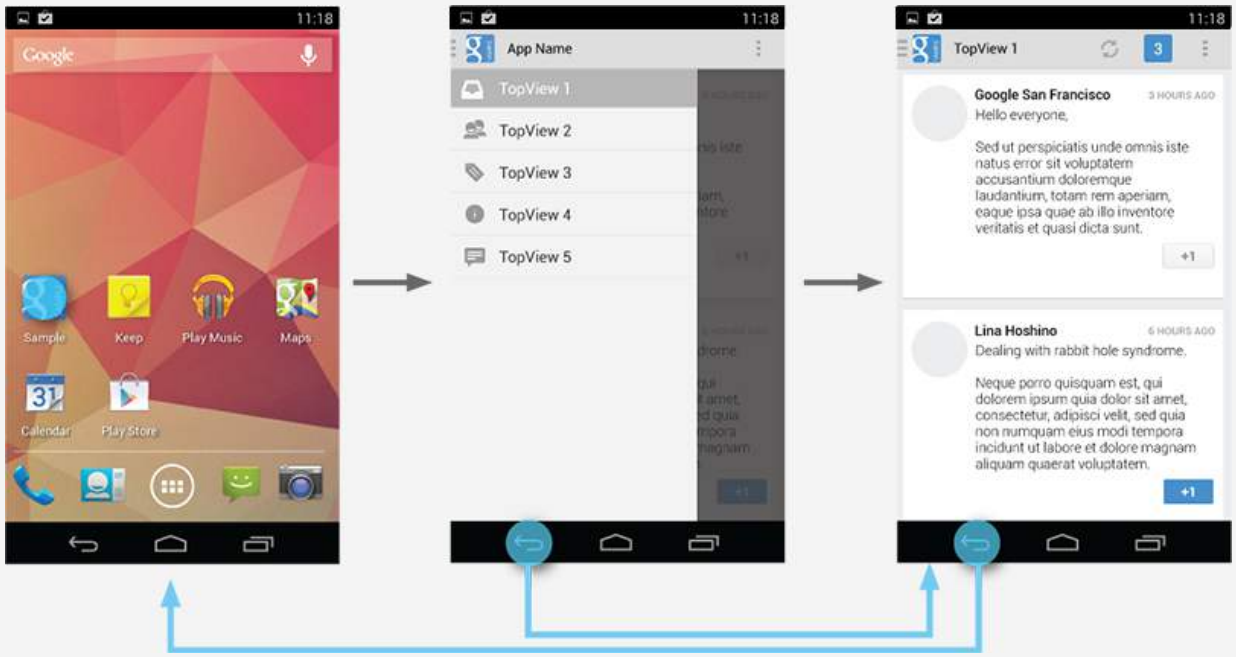


Action Bar

View Controls-3. Drawers

Introduce the user to the drawer at first use

Upon first launch of your app, introduce the user to the navigation drawer by automatically opening it. This ensures that users know about the navigation drawer and prompts them to learn about the structure of your app by exploring its content.

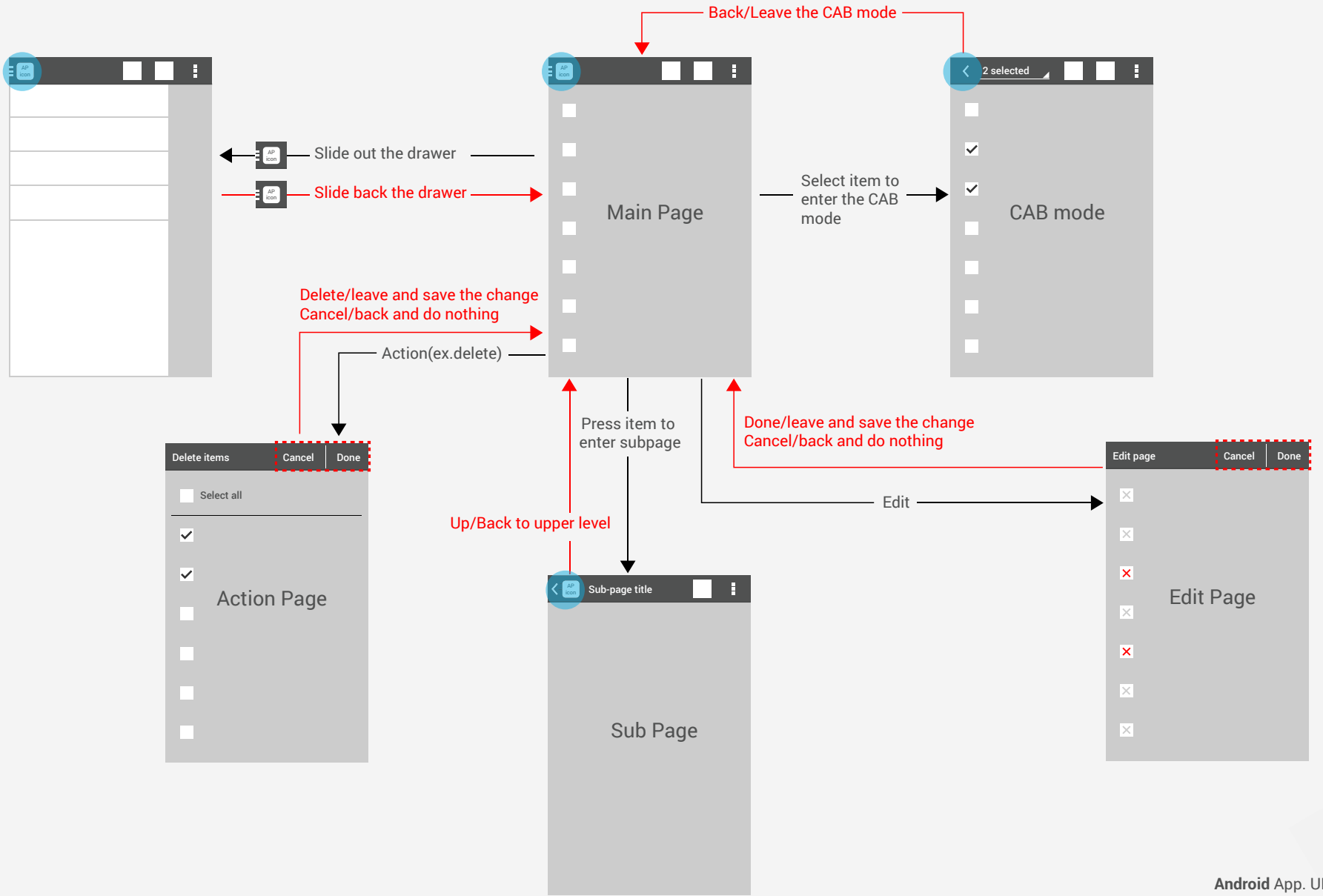


Keep Help and Settings in the overflow.

This also applies to common navigation targets, such as access to Help or the app's Settings. As per style guide convention Help and Settings are always located in the action overflow.

Action Bar

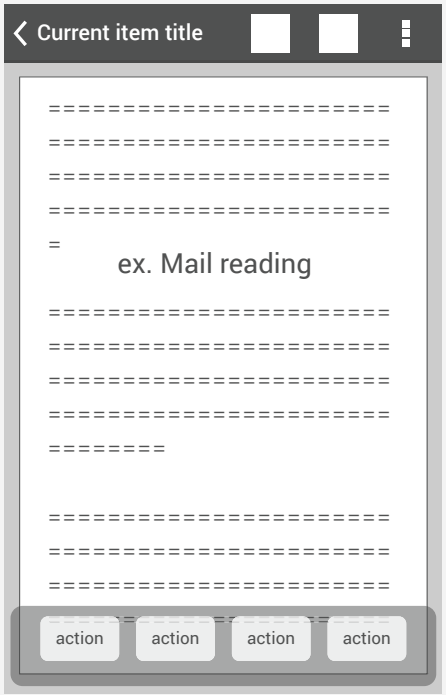
Overall of Navigator and confirmation

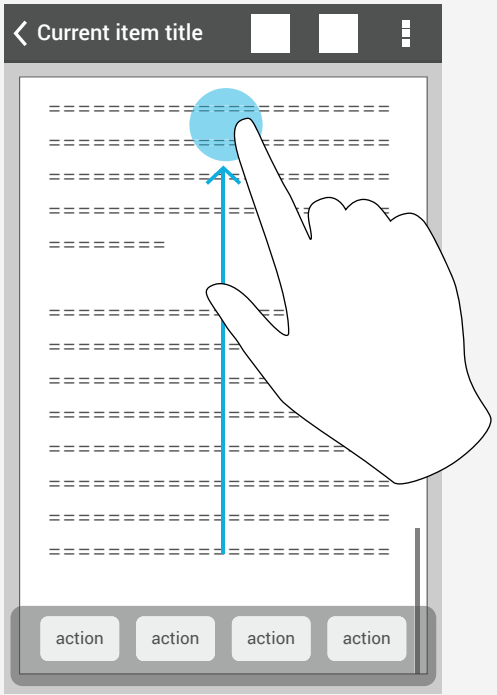


Action panel

Action Bar 與 Action Panel 所代表的層級不同

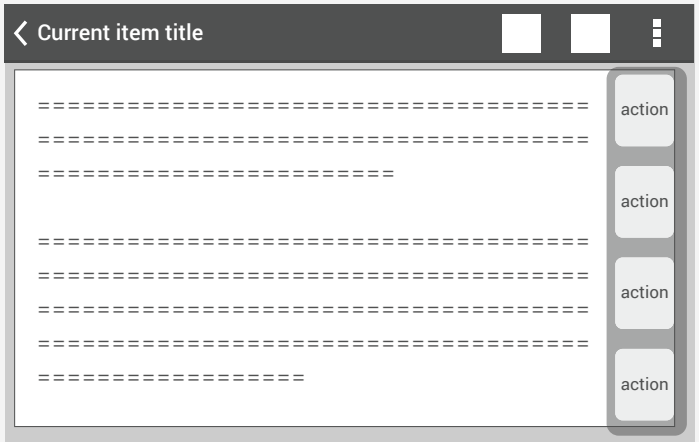
當UI的設計上需保留Action bar來處理上層動作，又必須提供對當下畫面內容物的獨立操作行為，後者則以Action panel處理。(EX. Email-mail reading page, Gallery-photo editing page)





Content view拉到底時bottom必須預留空間讓內容不被action panel遮蔽。

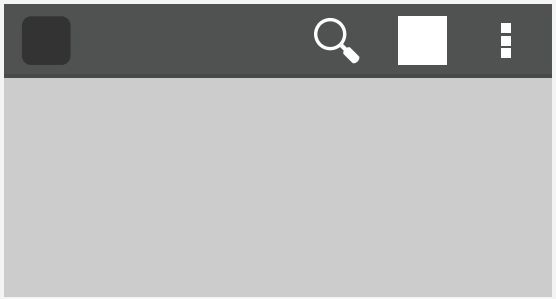
Landscape mode



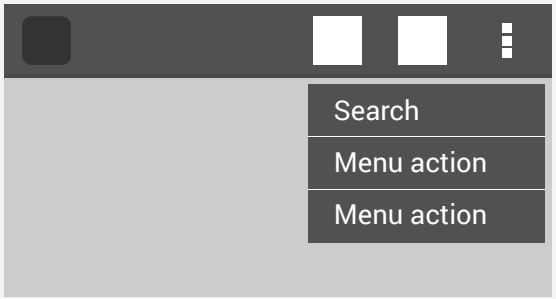
Search

Search type

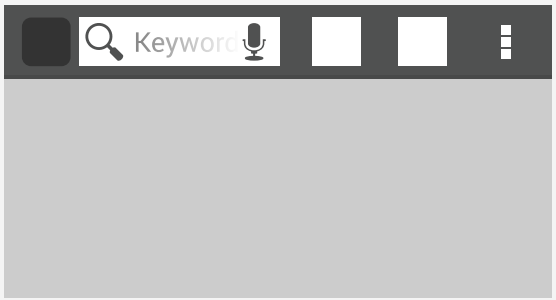
1. Search action icon



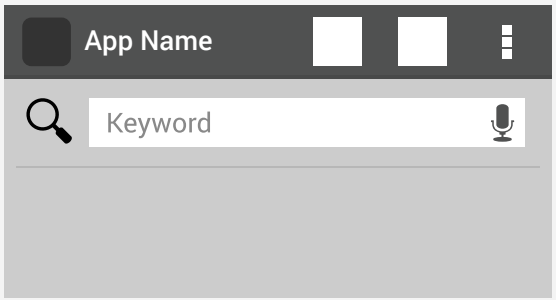
2. Search option in action menu



3. Search box in action bar

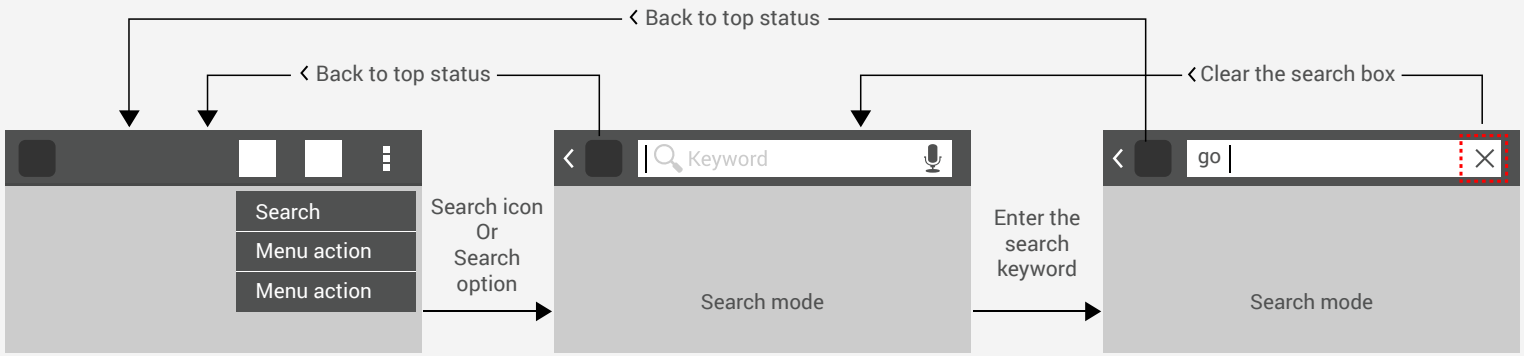


4. In line Search bar

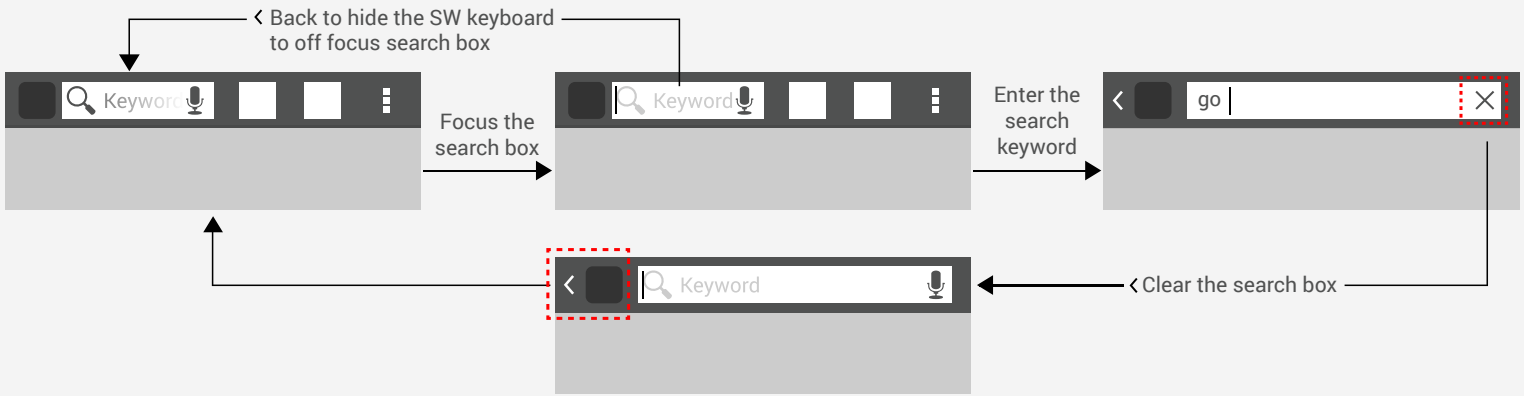


Search

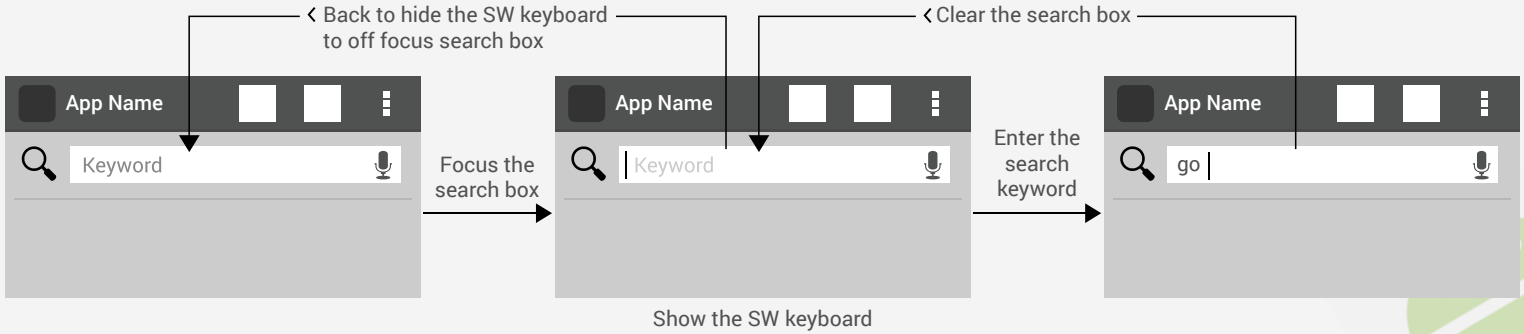
1. Search action icon and
2. Search option in action menu



3. Search box in action bar



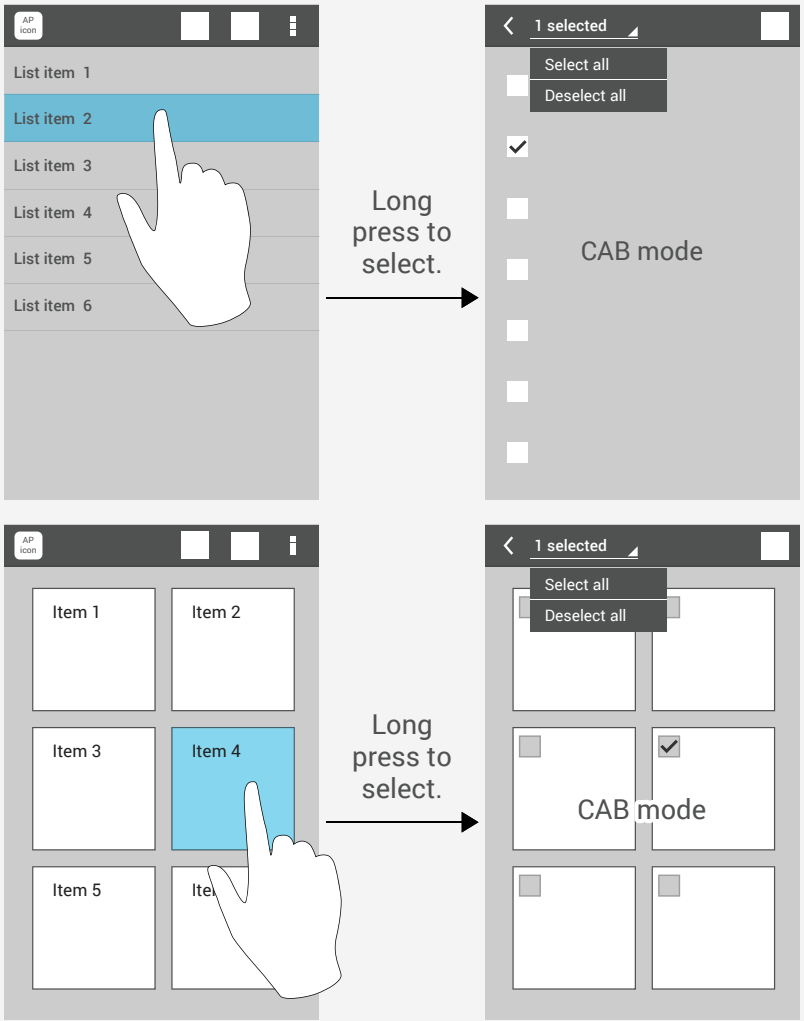
4. In line Search bar



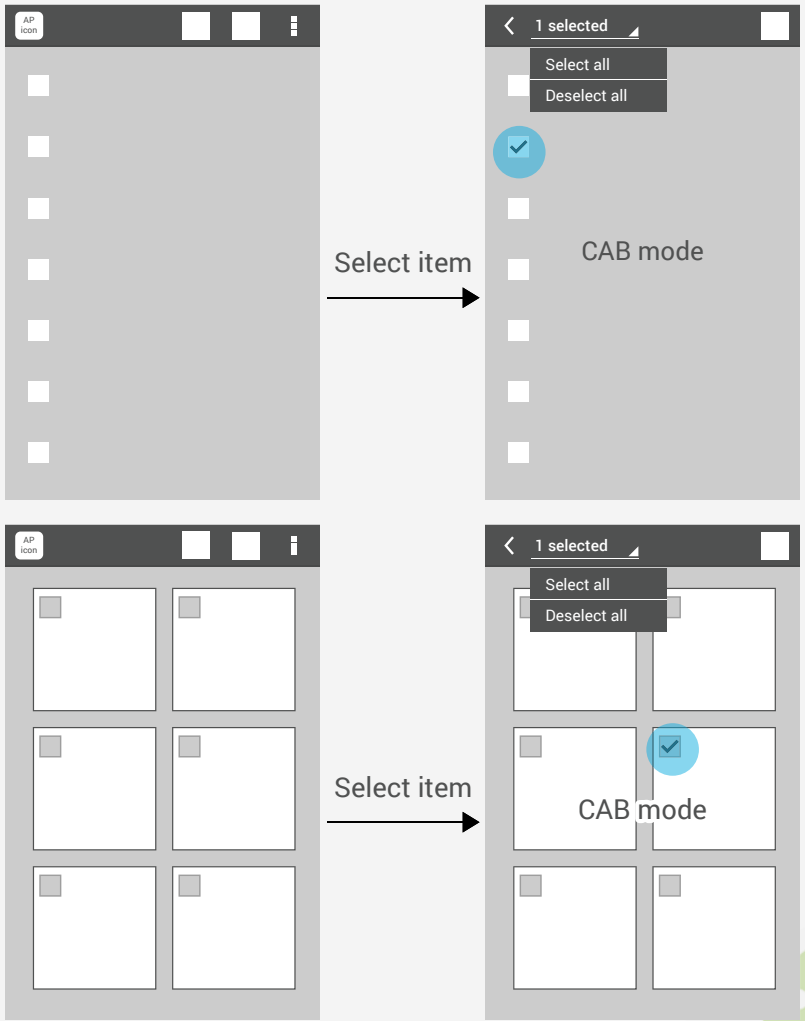
Selection

Type A. 以選取物件為始，在選擇目的動作

A-1. Long press item to select and enter CAB mode.



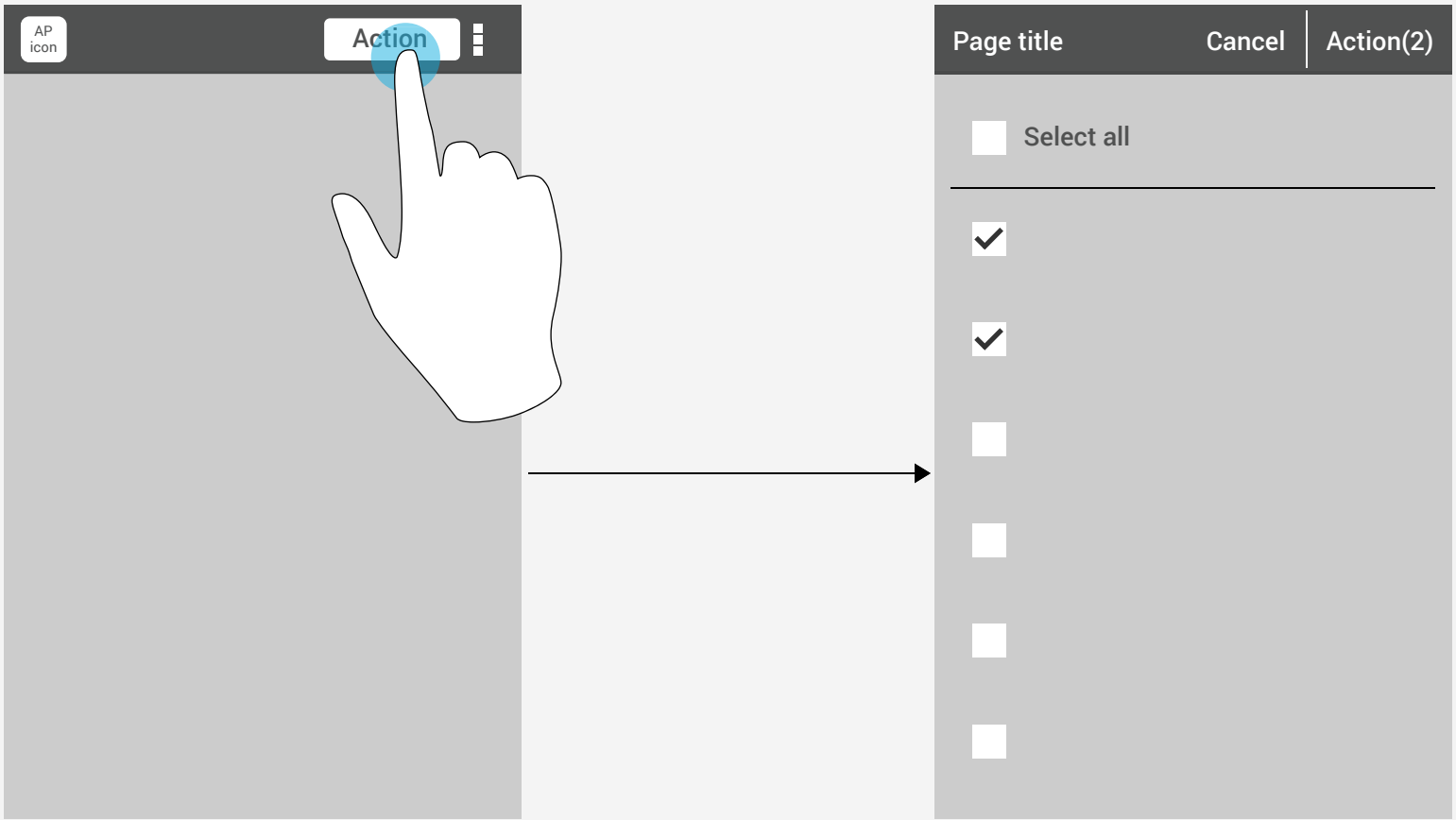
A-2. Mark checkbox to select and enter CAB mode.
(當此AP會大量選作用時，即是用此種狀況)



Selection

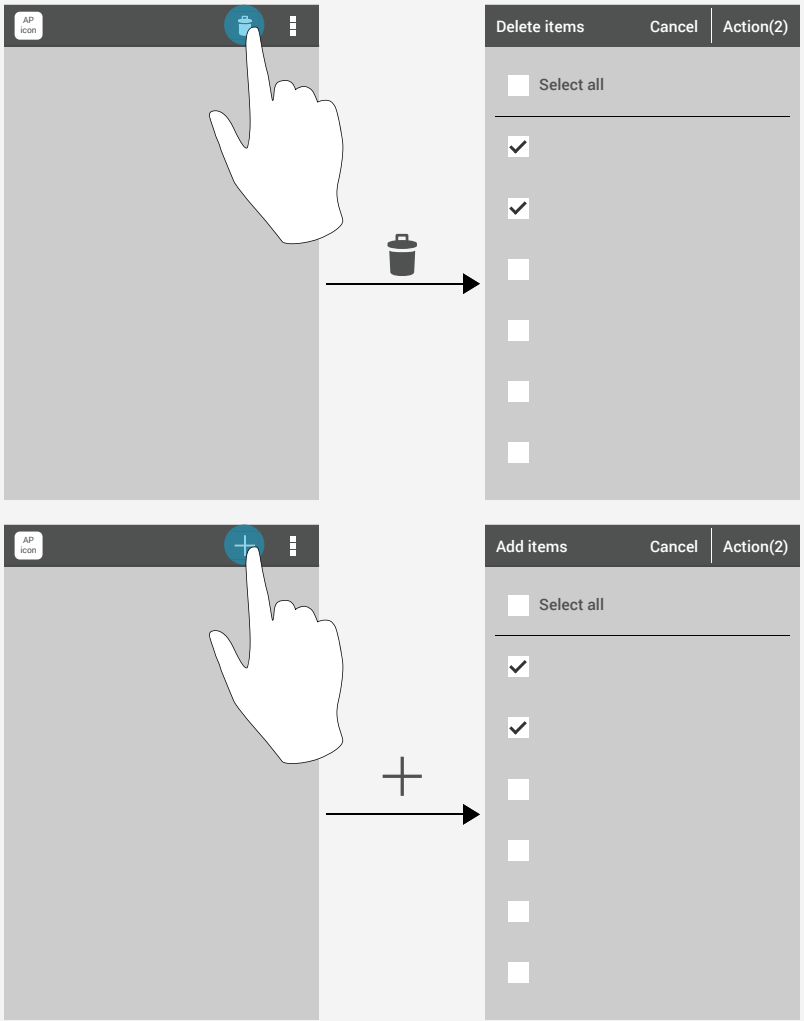
Type B. 以動作目的為始，在進入選擇頁面選擇對象

需保留 Page title 以提醒當下選去後動作目的.

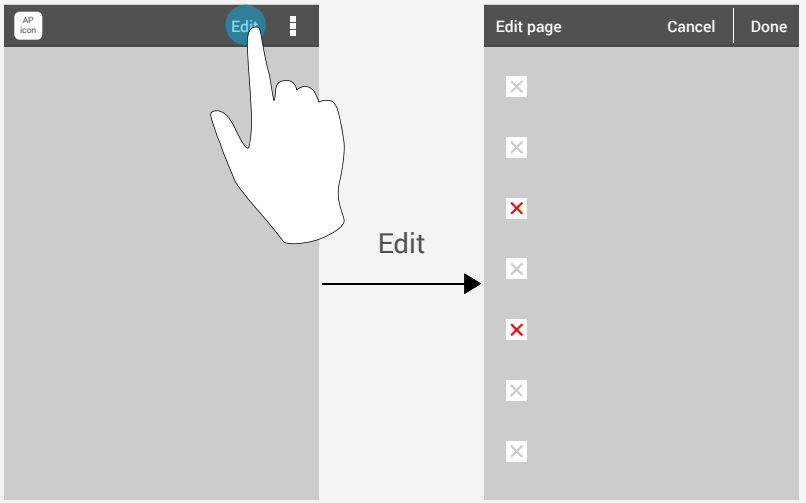


Selection

B-1. 以非常明確的單一動作目的進入選取模式(ex. Delete, add..etc)



B-2. 針對當下畫面以Edit進入編輯模式，內含兩個以上編輯行為
(ex. Delete, rearrange..etc)



- 套用有明確目的示意性的customized checkbox 來示意被選取物件在確認後的結果
- 確認的用字用Done來強調確認編輯結束

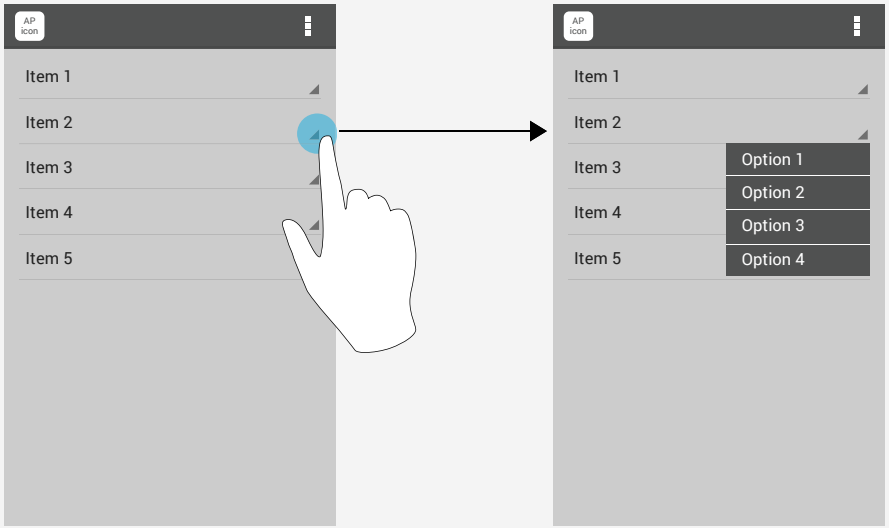
Search & Selection

Context Menu

1.

When long press behavior is used to enter
CAB selection, pop up context menu will be display by right corner arrow.

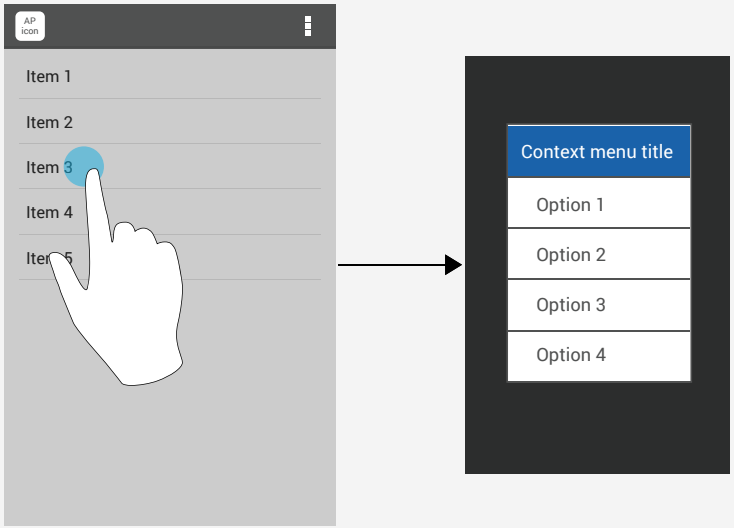
- Right low corner



2.

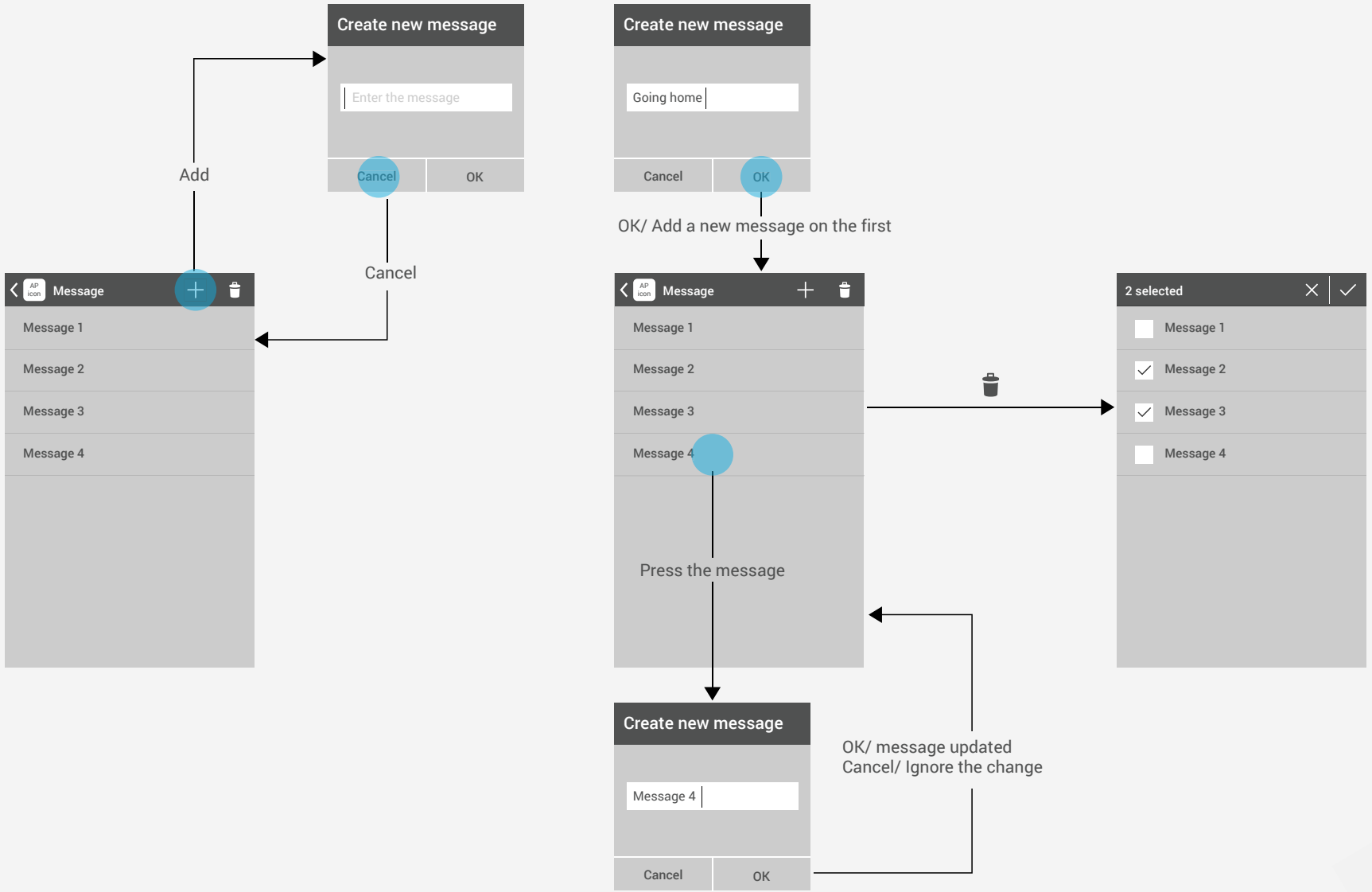
Other rule: Long press to opo up context menu.

- Long press to popup context menu



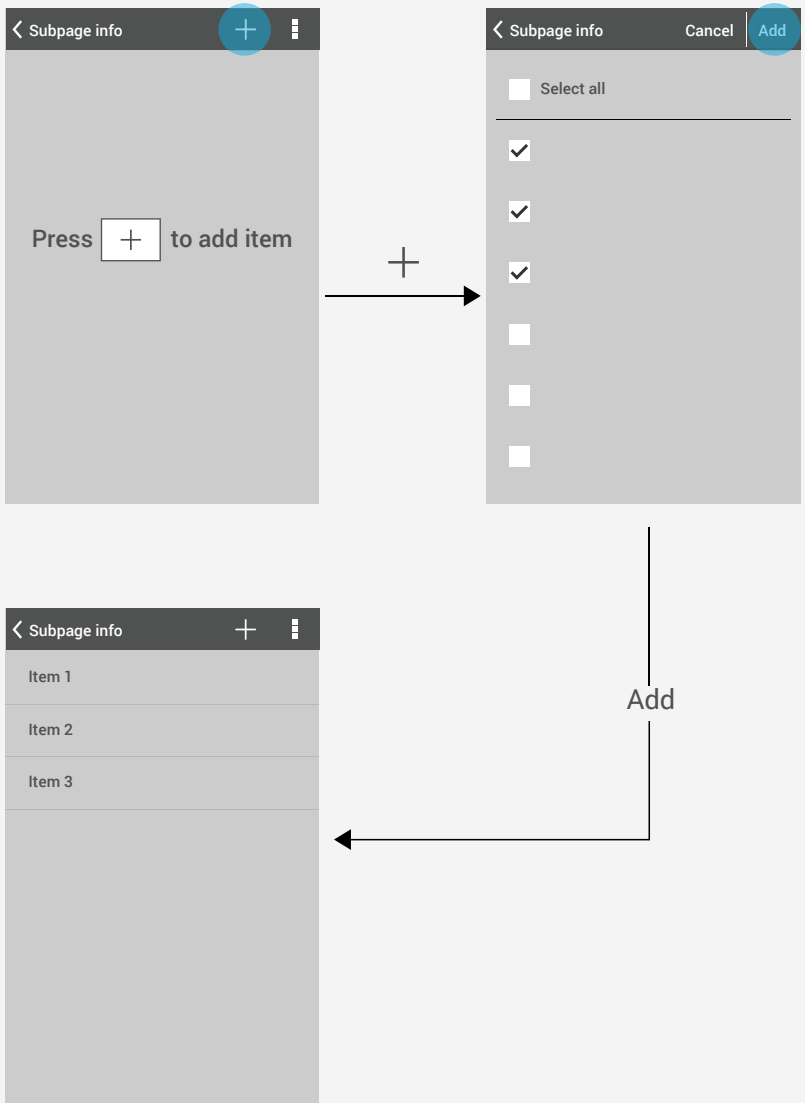
Create New

Sample Case - Create new message



Add

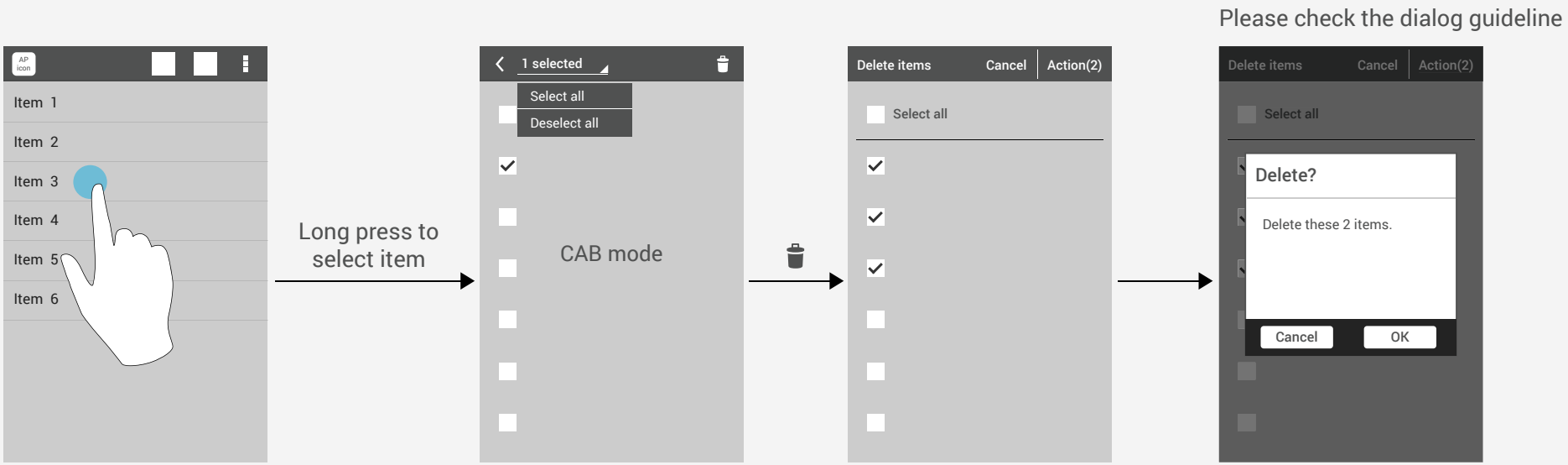
Sample Case - Add items



Delete

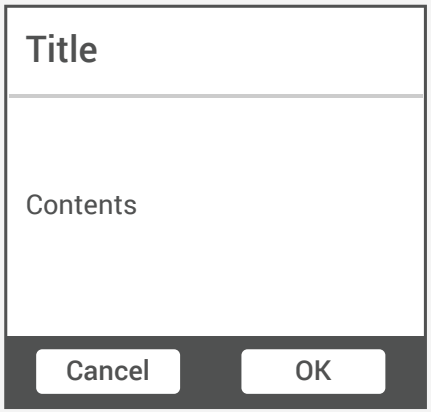
Delete behavior

1. Long press to enter CAB mode.
2. Checkbox list view to select to delete.
3. Select delete of action menu to enter selecting mode.
4. When the data will be deleted, it will show the confirmation dialog.

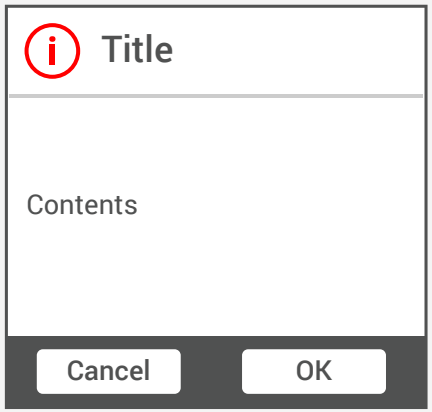


Dialog

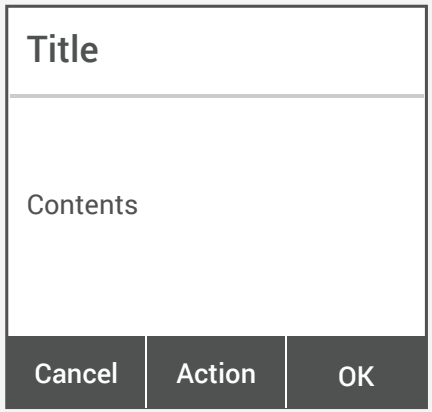
Dialog format Title/ Content/ Confirm Action



Warning dialog 才會有Icon.

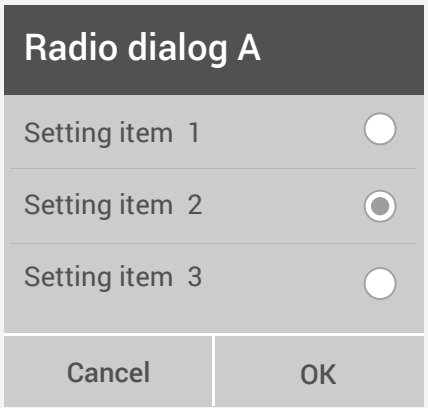


不只一個action時.

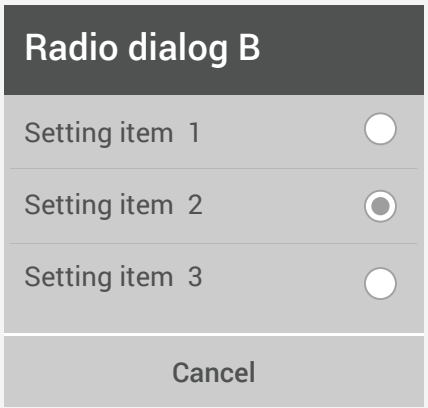


避免超出畫面可容納的選項

Rodio Button Dialog 使用規則



- 需要confirm時

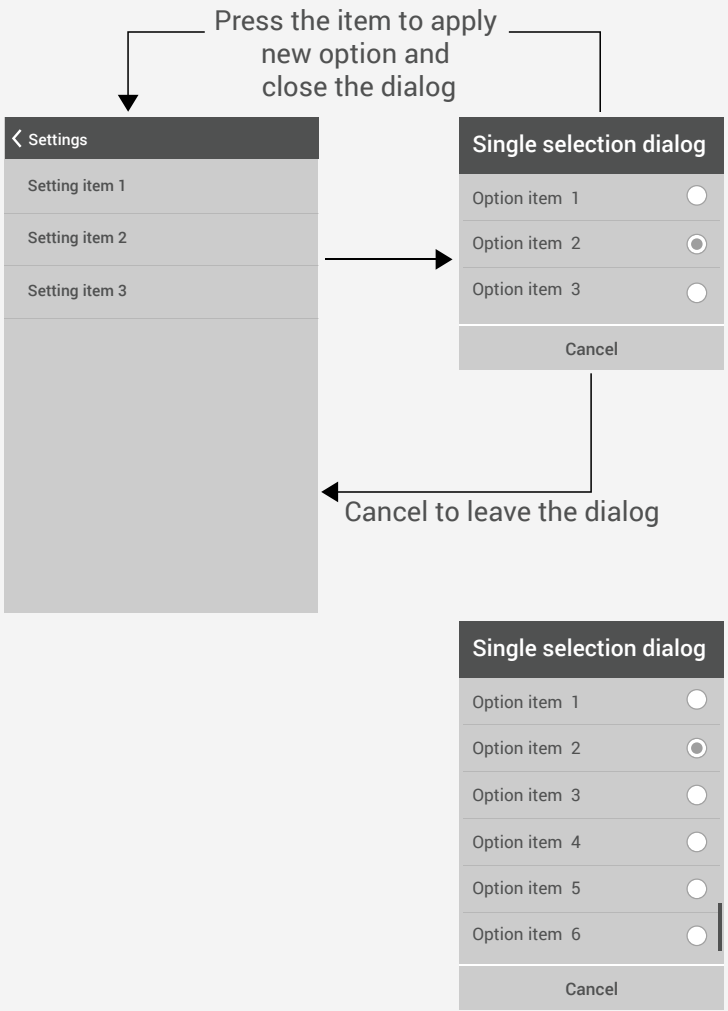


- 不需要confirm時
- 點選即離開並套用
- Back/ Dialog out side/ Cancel並離開

Settings

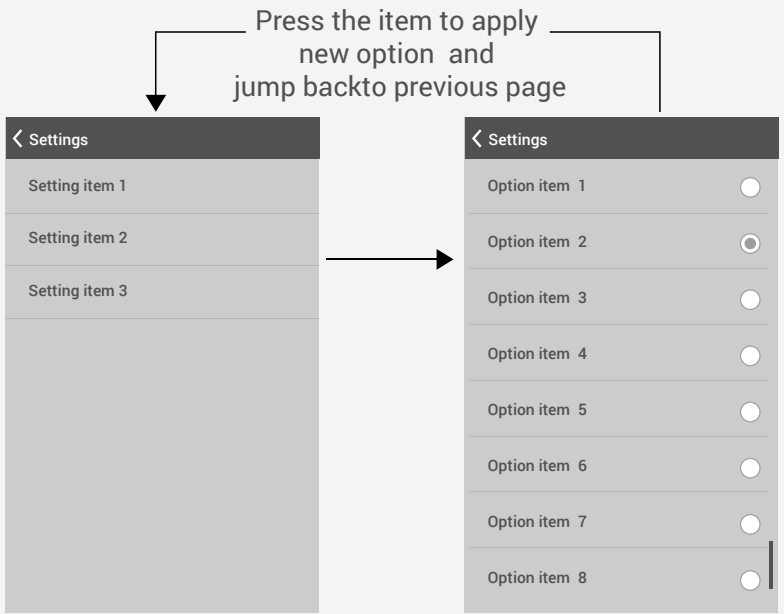
No confirm single selecting

A. Options list dialog



- Scrollable dialog

B. Full page list

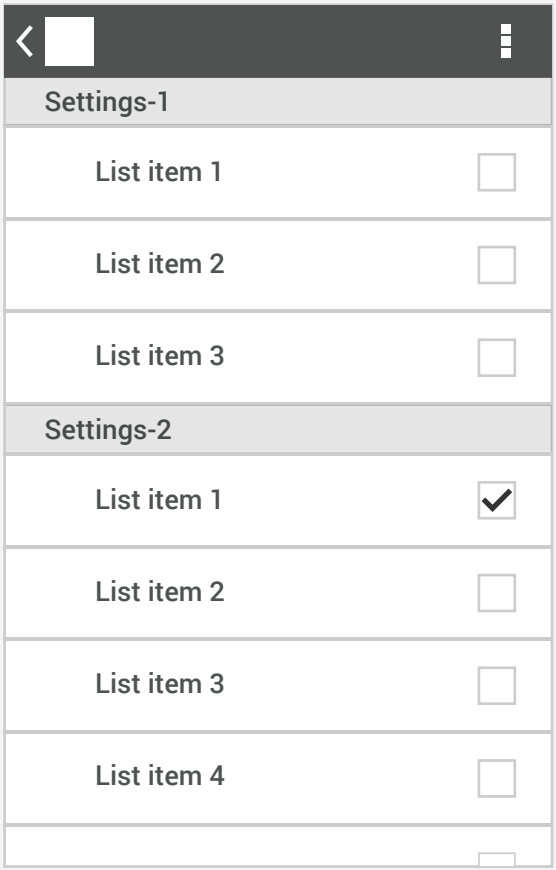


Checkbox

Postion of Checkbox

Selection function: Alight to Left

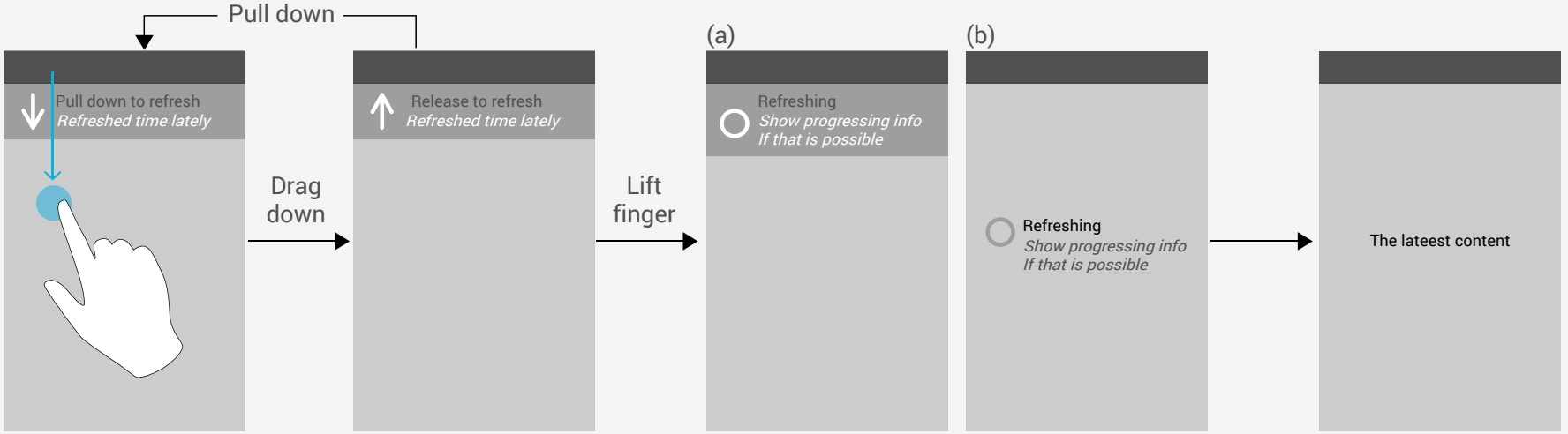
Swith function: Alight to Right



Refresh

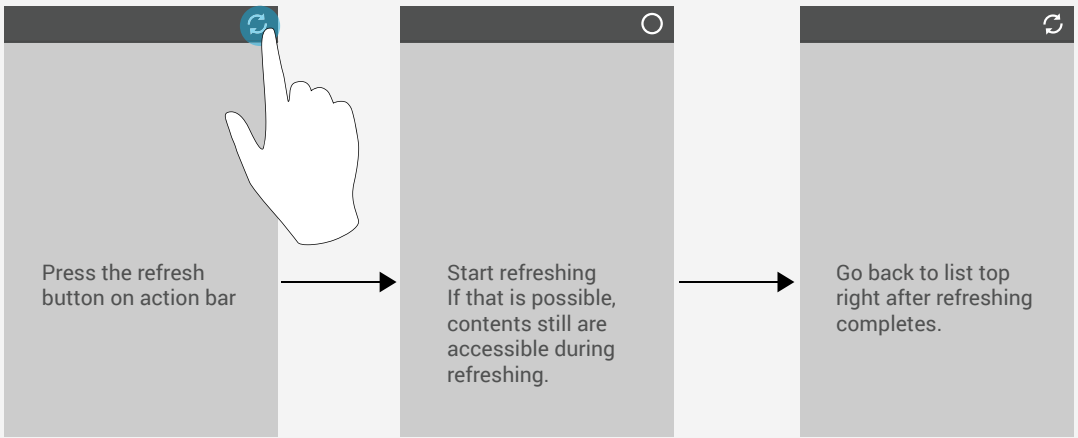
1. Using “Pull down refresh” if

- The latest content stacks on the top the screen.
- Refreshing is not critical enough to renew immediately.
- The content list isn’t too long generally.



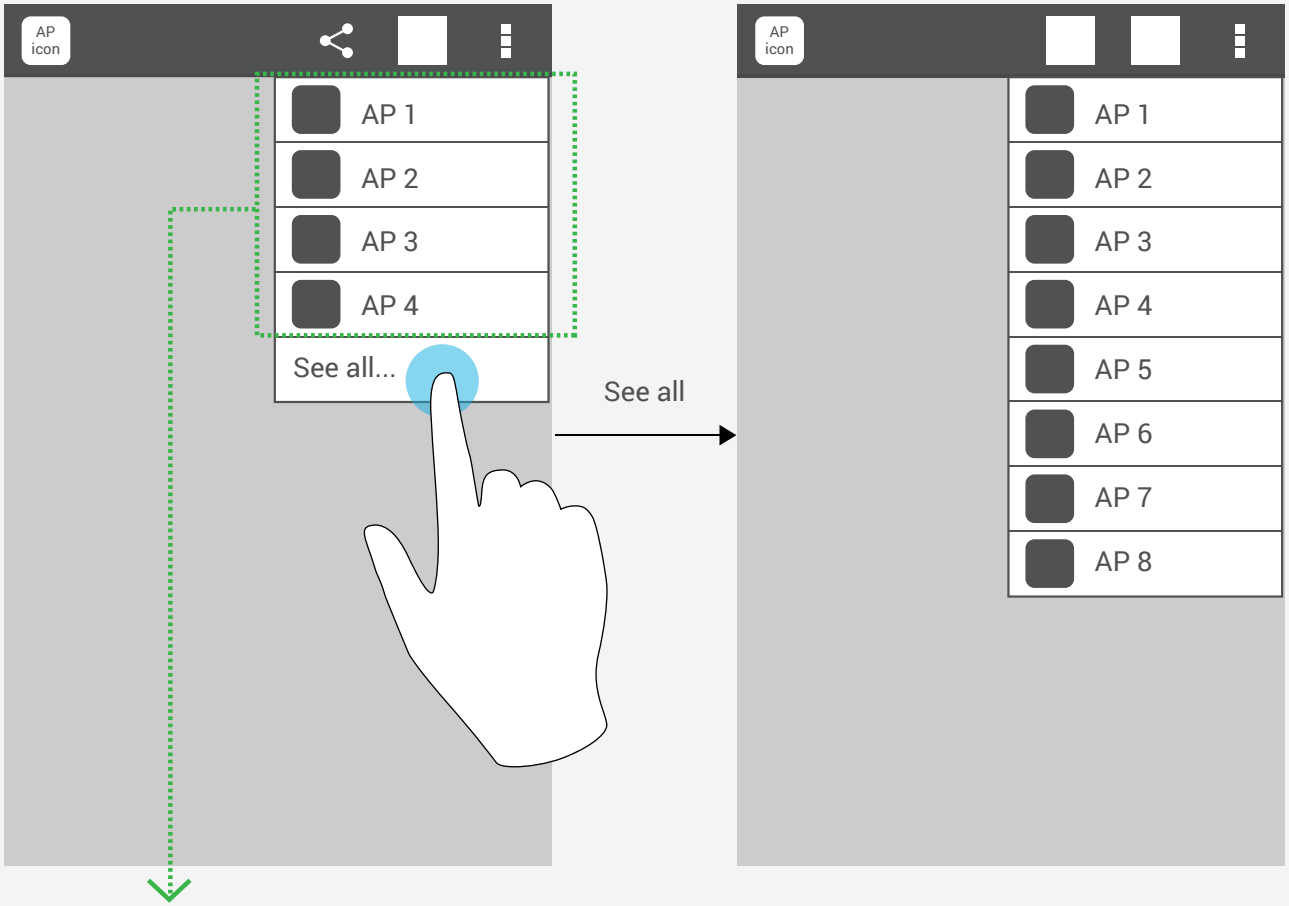
- Back to normal status if releasing on this stage.
- If that is possible, content still are accessible during refreshing(a).
- Go to the top of the screen.

2. Press refresh button



Share via app activity

A. Share option to popup activity controller.
B. Share button with drop down menu.



Record user pattern, list the most used share App in the first level.

ASUS MIS UI Style

About 行動使用者的頁面排版指引

- 行動裝置真正可利用的空間珍貴，要避免佔用空間、卻沒有任何特殊目的的橫幅、長條、照片和圖像，這種使用目的可能是間接的，ex.傳達階層或架構，但設計師往往要能解釋原因。
- 盡量使用條列式資訊，而非表格。
- 元素編排的層級為 Postion →Size →Shape →Contrast →Color →Form，最重要的項目較大、較高、較亮等。
- 規劃內容時，將重要的資訊放置於內容區域的左上角。
- 為快速、效率和安心設計，移除不必要的欄位，以及把頁面和步驟減少到最少。
- 頁數、標題、表頭、頁腳、頁籤、連結等，可以用來辨識使用者當前的位置。
- 太長文字不易閱讀，建議大約30～35個字元為最長。
- 標題用來描述頁面、頁面裡的元素和內容，使用上要一致，並遵循易讀性和可讀性法則。
- 標籤文字使用小的字體時，必須跟背景有顏色上的反差，文字的明度和背景色的明度至少要 3：1。
- 使用顏色來呈現細節時，明度的對比是最重要的原則。

Our App.

Colors

採用一個主色(ex.藍色)，運用於UI元件，例如進度條、按鈕、捲軸、頁籤...，並用不同的透明度來表示手指觸發的狀態。可在另外搭配其他次要輔助色(橘色、灰色...)使用。

States

Normal	REMAINS STATIC
Pressed	DARKENS SLIGHTLY (LIGHTENS ON DARK THEMES)
Focused	ILLUMINATES WITH COLOR
Disabled	DRAWS 30% OF THE NORMAL STATE
Disabled & focused	DRAWS 30% OF THE FOCUSED STATE

Logo

可以自由設計您的系統Iocn，並可以在Action Bar中強調凸顯。

ASUS APP
SYSTEM DESIGN

ITRS+

或是用字體設計。

Icons

可以自由設計您的系統Iocn，但要注意Android系統的基本樣式，不要畫為iOS的樣式。

× Don't ✓ Do

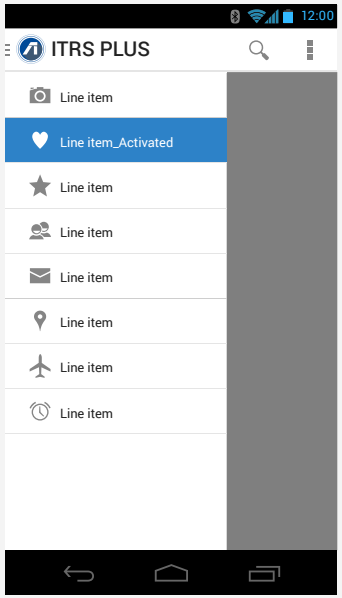
Our App.

Our UI Reference

企業標準色

01
#004A97 #2D82C8

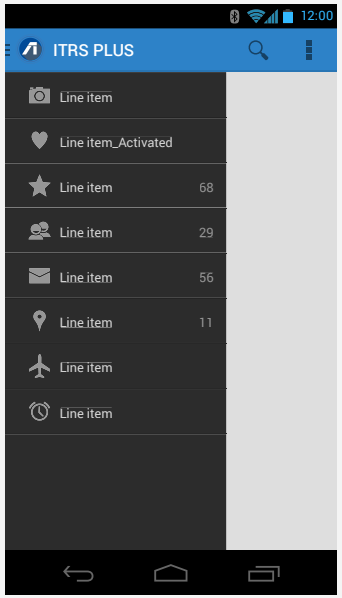
Action Bar White + Drawers White



60% Icon Colors: #333333
Enabled: 60% opacity
Disabled: 30% opacity

02
#2D82C8

Action Bar Blue + Drawers Dark Gray



50% Icon Colors: #000000
Enabled: 50% opacity
Disabled:

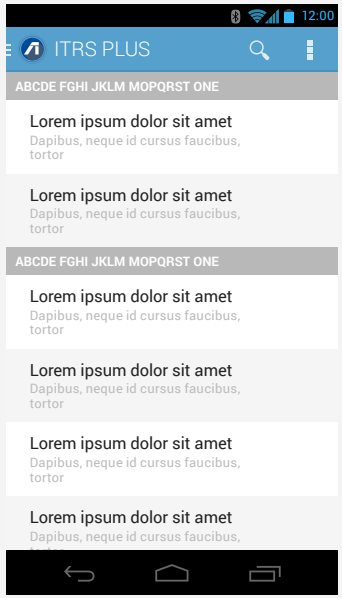
03
#2D82C8

Action Bar Blue + Tab White

80% Icon Colors: #FFFFFF
Enabled: 80% opacity
Disabled:

04
#55A0CC

Action Bar Blue + Gray分層

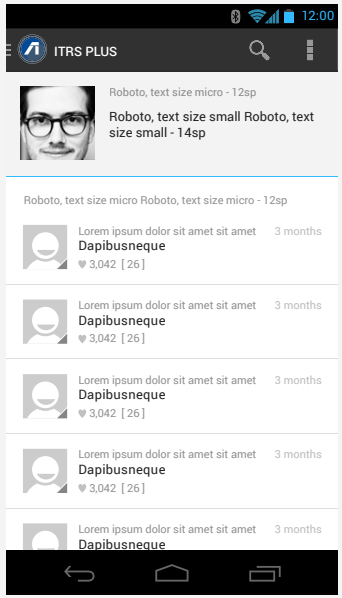


80% Icon Colors: #FFFFFF
Enabled: 80% opacity
Disabled:

官網Link色

05
#33B9FF

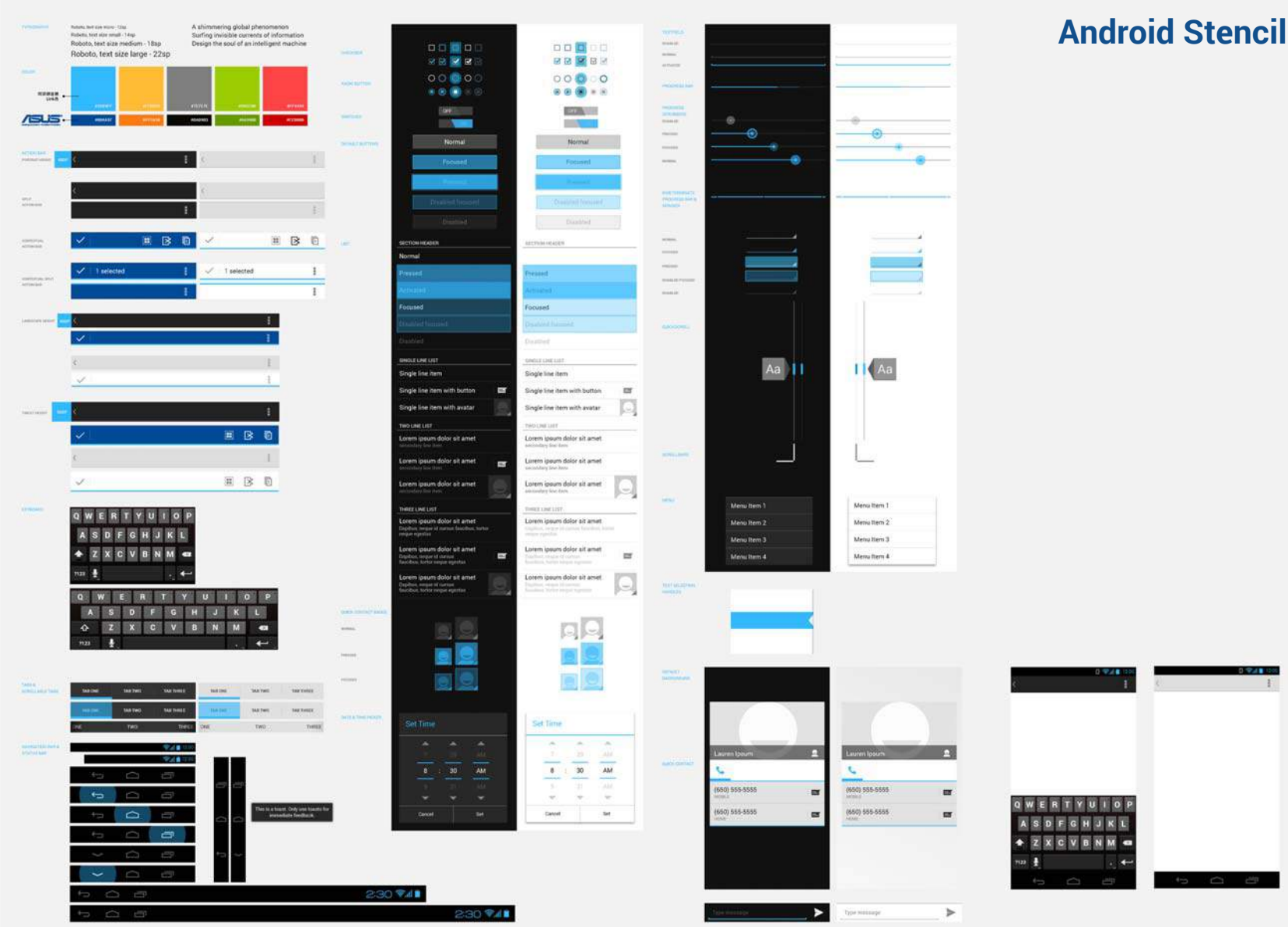
Action Bar Dark Black + Gray Content



50% Icon Colors: #FFFFFF
Enabled: 50% opacity
Disabled:

Our App.

Android Stencil



Typography

A To support such use of typography, Ice Cream Sandwich introduced a new type family named Roboto, created specifically for the requirements of UI and high-resolution screens.

Roboto Thin

Roboto Light

Roboto Regular

Roboto Medium

Roboto Bold

Roboto Black

Roboto Condensed Light

Roboto Condensed

Roboto Condensed Bold

Typographic Scale

Android framework uses the following limited set of type sizes:

Text Size Micro	12sp
Text Size Small	14sp
Text Size Medium	18sp
Text Size Large	22sp

Default type colors

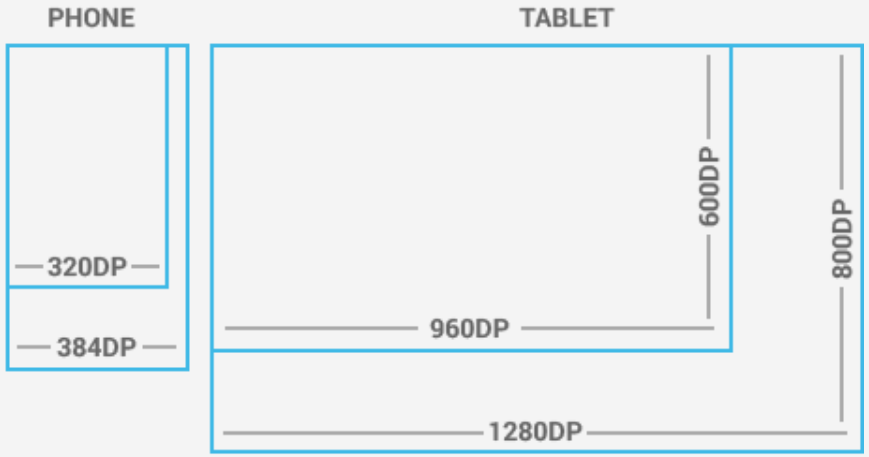
The Android UI uses the following default color styles:

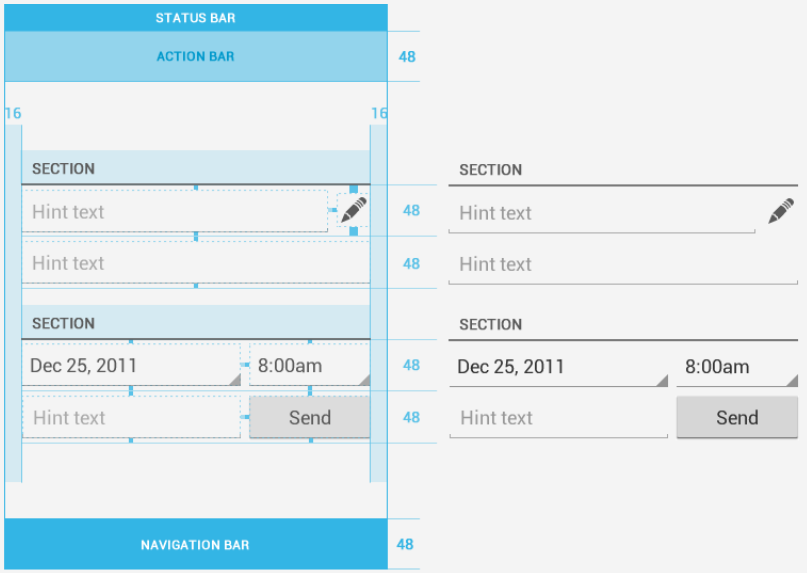
Text Color Primary Dark
Text Color Secondary Dark

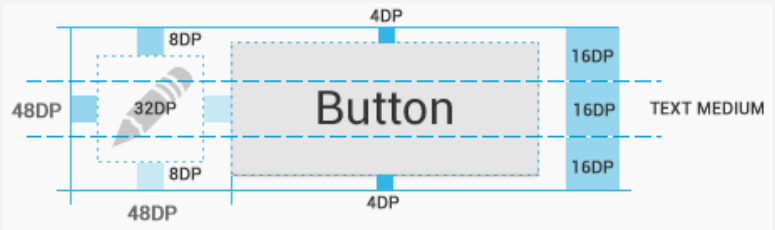
Text Color Primary Light
Text Color Secondary Light

Metrics and Grids

Android不僅設備大小與螢幕解析(DPI)眾多，我們可以假想只有兩種設備：小於600dp的手機；大於600dp的平板。
DPI只有：LDPI(120dpi), MDPI(120dpi), HDPI(160dpi), and XHDPI(320dpi)幾種，選取這幾種進行設計優化佈局。







48dp Rhythm

Touchable UI components are generally laid out along 48dp units.
48dp轉化為物理單位為9mm，一般7-10mm是最準確並適合手指觸控的區域大小。

Mind the gaps

Spacing between each UI element is 8dp.

Devices and Displays

Android設備眾多，螢幕尺寸不一，請確保您的App內容可以適應個各尺寸(包含橫式)，以達到一致性的美觀與平衡。

1. Be flexible
擴展和縮小您的布局來適應不同的高度與寬度。

2. Optimize layouts
在大設備上，利用額外的螢幕版面，創建復合視圖，展示更多的內容更便捷的選單。

3. Assets for all
為不同螢幕解析度提供資源，保證您的App在任何設備上都漂亮。

Strategies

方法一：從Baseline(MDPI)開始，然後放大縮小，以適應到其它尺寸。
方法二：從螢幕的最大尺寸開始，然後縮小，並適應到您需要的最小螢幕尺寸。

進一步Multi-pane Layouts詳細請見官方介紹：
http://developer.android.com/design/patterns/multi-pane-layouts.html

Iconography

1. Launcher

The launcher icon is the visual representation of your app on the Home or All Apps screen. Since the user can change the Home screen's wallpaper, make sure that your launcher icon is clearly visible on any type of background.

Size Full asset, 48x48 dp

Style

Use a distinct silhouette. Three-dimensional, front view, with a slight perspective as if viewed from above, so that users perceive some depth.

2. Action Bar

Action bar icons are graphic buttons that represent the most important actions people can take within your app. Each one should employ a simple metaphor representing a single concept that most people can grasp at a glance.

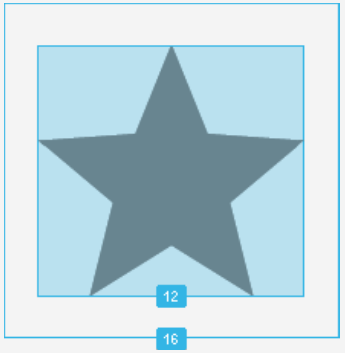
Size Full asset, 32x32 dp
Optical square, 24x24 dp

Style

Pictographic, flat, not too detailed, with smooth curves or sharp shapes. If the graphic is thin, rotate it 45° left or right to fill the focal space. The thickness of the strokes and negative spaces should be a minimum of 2 dp.

3. Small / Contextual Icons

Within the body of your app, use small icons to surface actions and/or provide status for specific items. For example, in the Gmail app, each message has a star icon that marks the message as important.



Size Full asset, 16x16 dp
Optical square, 12x12 dp

Style

Neutral, flat, and simple. Filled shapes are easier to see than thin strokes. Use a single visual metaphor so that a user can easily recognize and understand its purpose.

Iconography

4. Notification Icons

If your app generates notifications, provide an icon that the system can display in the status bar whenever a new notification is available.

Size Full asset, 24x24 dp
Optical square, 22x22 dp

Style

Keep the style flat and simple, using the same single, visual metaphor as your launcher icon.

Colors

Notification icons must be entirely white. Also, the system may scale down and/or darken the icons.

Icons nameing 切圖命名原則

檔名只能有以下字元，否則會無法編譯：

- 小寫a~z
- 數字0~9 (如果是流水號的話，要從 0 開始編號)
- 不能有空白，把所有的 - 改成 _ 底線
- 不要把數字或符號當成檔名的開頭

Asset Type	Prefix	Example
Icons	ic_	ic_ic_star.png
Launcher icons	ic_launcher	ic_launcher_calendar.png
Menu icons and Action Bar icons	ic_menu	ic_menu_archive.png
Status bar icons	ic_stat_notify	ic_stat_notify_msg.png
Tab icons	ic_tab	ic_tab_recent.png
Dialog icons	ic_dialog	ic_dialog_info.png

其他參考：

- 不能點擊(disabled)：btn-xxx-d.png
- 圖標：ic-xxx.png
- 圖片：pic-xxx.png
- 照片：pho-xxx.png
- 背景：bg-xxx.png
- 按鈕：btn-xxx-n.png

String Guideline

App寫句子時請注意下列幾點

1. Concise

從限制使用30各字符（包含空格）開始，除非比要絕對不增加字符。

2. Simple

假裝您正在與一個精明能幹的人說話，但他不懂技術語言、英文不是很好。

- Put top news first, Put the user's goal first.
前10個字應該包涵一個最重要的資訊，如果不是請重新開始。

3. Friendly

直接使用第二人稱(你)，如果這不是你休閒對話時會用的句子，那大概也不適合用於此。

- Avoid being confusing or annoying.
不要試圖解釋細節的差別。

其他

- 逗號(,)和句號(.) 冒號(:) 跟前面的字中間不要空格，但和後面的字中間要空一格
- Confirmation button: OK要大寫
- Dialog titled文法: 目的動作名詞型態+行為動作形容詞型態(ex. Editing finished)
- 不要使用重複的前最字。

Words to avoid

更多詳細請見官方介紹
http://developer.android.com/design/style/writing.html

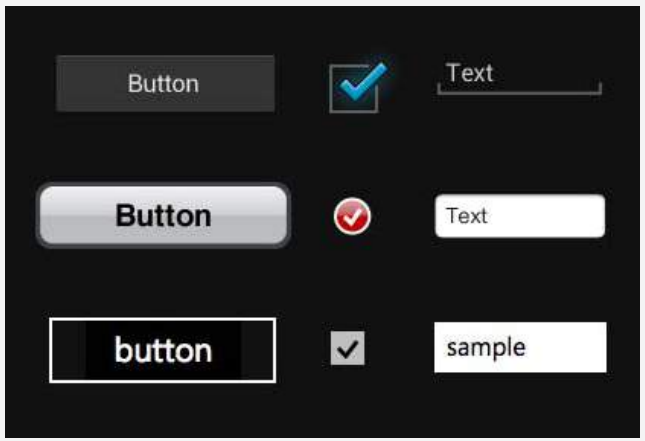
✖ Don't use	✔ Use
one, two, three, four, ...	1, 2, 3, 4, ...
application	app
cannot, could not, do not, did not will not, you will	*Contractions:* can't, couldn't, don't, didn't won't, you'll, and so on
okay, ok	OK
please, sorry, thank you	*Attempts at politeness can annoy the user, especially in messages that say something has gone wrong. Exception: In Japanese, "please" is mandatory and imperative verbs should be localized accordingly (turn on -> please turn on).*
there is, there are, it is *and other "disappeared" subjects (grammatical expletives)*	*Use a noun as the subject*
abort, kill, terminate	stop, cancel, end, exit
fail, failed, *negative language*	*In general, use positive phrasing (for example, "do" rather than "don't," except in cases such as "Don't show again," "Can't connect," and so on.)*
me, I, my, mine	you, your, yours
Are you sure? Warning!	*Tell user the consequence instead, for example, "You'll lose all photos and media"*

Pure Android

大多數的開發App都是跨平台的，請記住平台間規則和習慣的差異。採用同一個版本走遍全部平台，會造成與平台本身的的不一致，很可能會讓您疏遠使用者。
所以請遵循以下一些思路，避免常見的錯誤。

Don't mimic UI elements from other platforms

不要照搬其他平台的UI，如一些平台按鈕鼓勵用圓角，一些鼓勵標題欄使漸層。如果要自訂自己的主題UI元素，要注意參考您系統的特性，而不是參考別的平台的習慣。



Don't carry over platform-specific icons

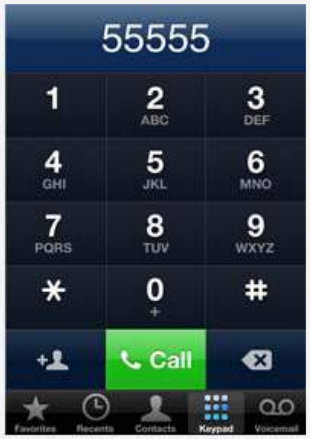
每個平台通常都有自己一套基本Icons，如分享、新建、刪除等。當您在設計Android時，請使用Android的Icons。
您可以在找到各種各樣的Icons。
http://developer.android.com/design/downloads/index.html

Pure Android

Don't use bottom tab bars

iOS透過底部tab切換頁面；Android的慣例是，在頂部放置切換tab，底部為工具欄區域。您應該像其他App一樣避免非一致性的體驗，避免混淆actions and view switching。

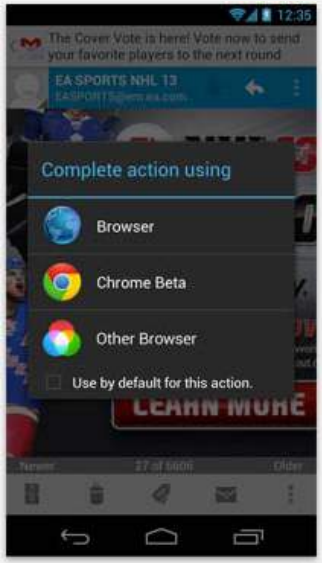
Android



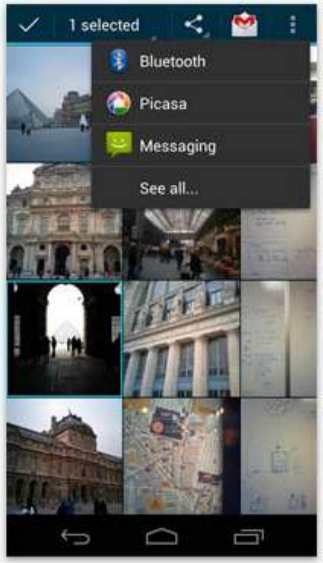
iOS

Don't hardcode links to other apps

不要強制連接到期他App。在需要使用其他App完成某種功能如分享，瀏覽網頁等，不要使用hardcodeed到特定App，而使用Androidn自帶的intent API加載activity chooser，讓使用者選擇使用那個App。



Android



iOS

Pure Android

Don't use labeled back buttons on action bars

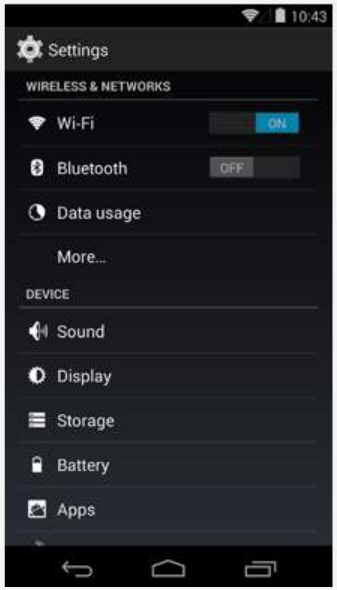
iOS習慣頂部操作欄使用〈返回按鈕〉，讓使用者回到上一層級的頁面；但Android分為兩個概念，在navigation bar是〈返回上一個紀錄頁面〉，而底部實體按鈕為〈返回上一級〉。

Android action bar with up caret

iOS labeled Back button.

Don't use right-pointing carets on line items

iOS用right-pointing carets，表示有下一級；但Android不適用此標誌，要避免使用，要避免使用者誤猜。



Android settings

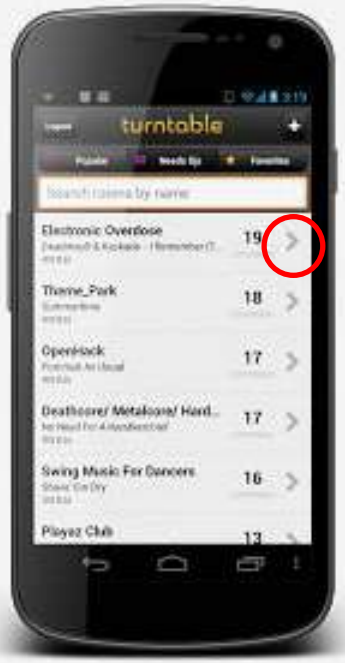
iOS settings

Common Mistake

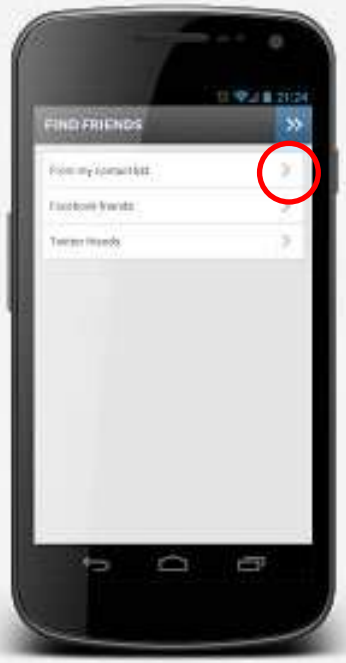
1. Right caret in lists

Android Design guidelines clearly states that lists should not use right caret.

× Don't



× Don't



Ex: Turntable.fm and Instagram both break this guideline.

2. The state of the drawer navigation

Overlaying the content area but not the action bar.

× Don't

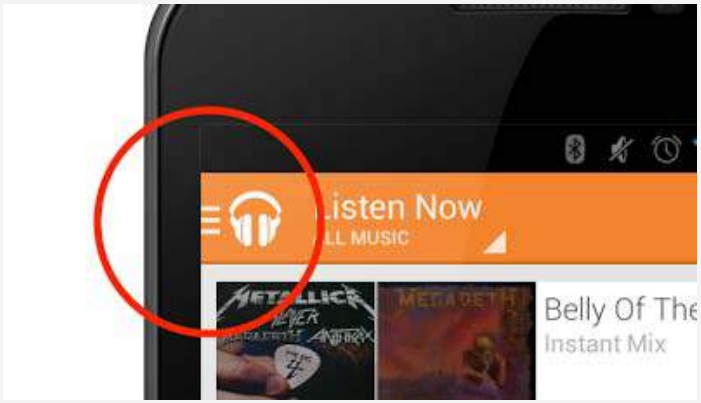
✓ Do

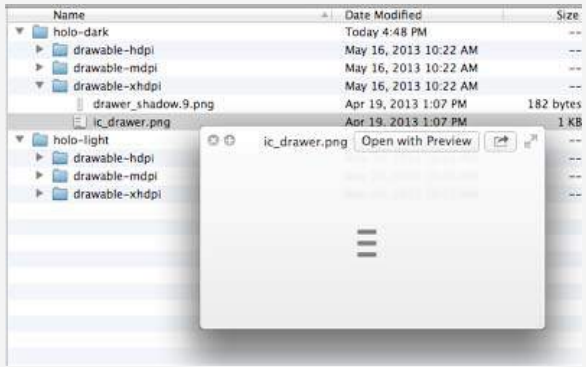
Common Mistake

3. Drawer navigation icon

Android Design guidelines clearly states that lists should not use right caret.

✓ Do

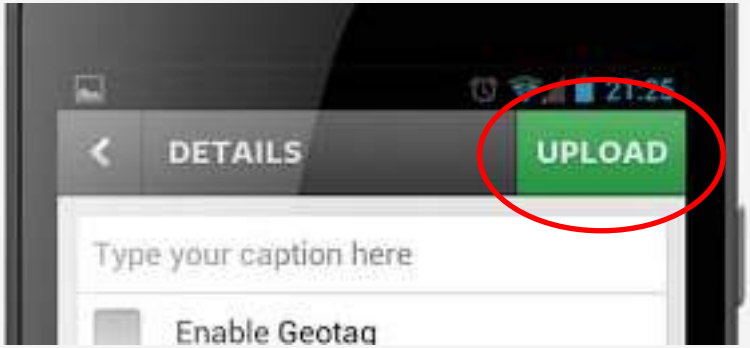


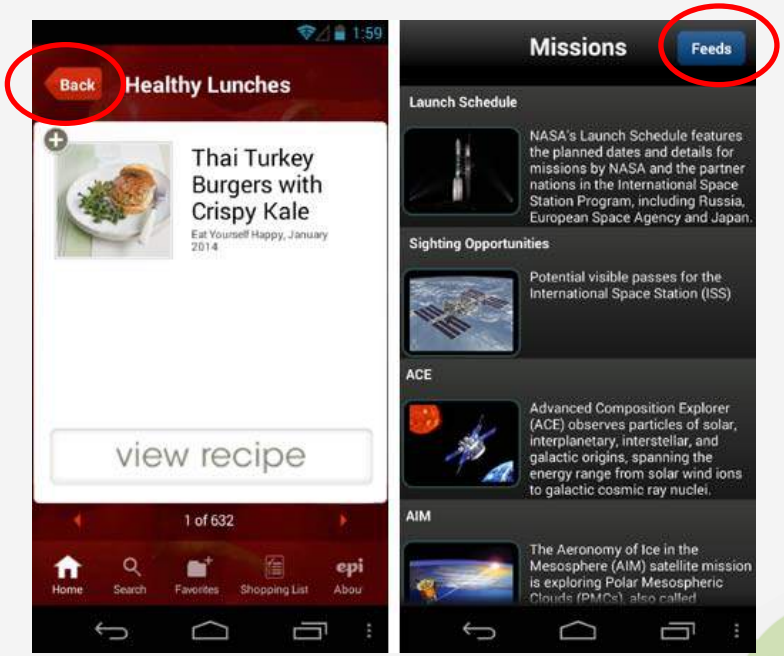


4. On sereen back button

Fortunately, this is less common nowadays but this is critical. Android apps should not have an on-screen back button.

× Don't



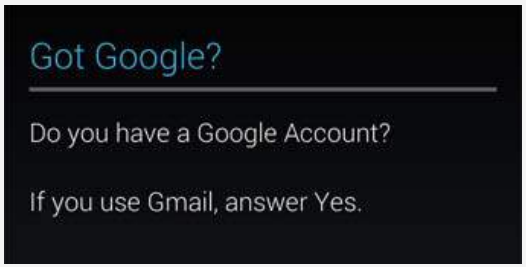


Design Principles

Simplify User Life

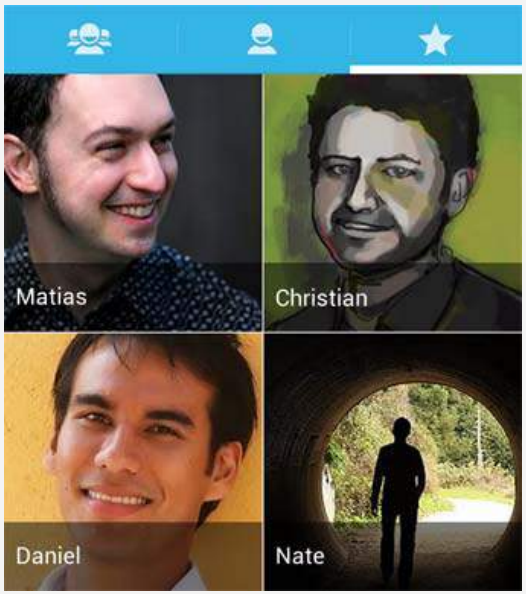
1. Keep it brief

Use short phrases with simple words. People are likely to skip sentences if they're long.



2. Pictures are faster than words

Consider using pictures to explain ideas. They get people's attention and can be much more efficient than words.

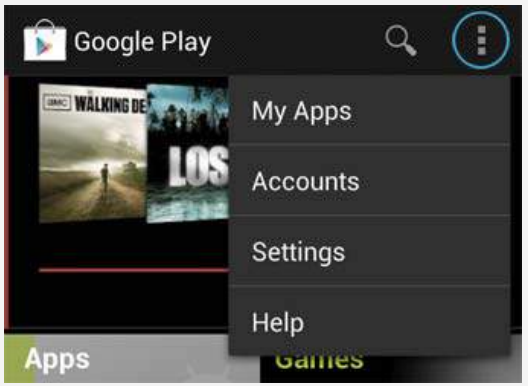


1. Decide for me but let me have the final say

Take your best guess and act rather than asking first. Too many choices and decisions make people unhappy. Just in case you get it wrong, allow for 'undo'.

4. Only show what I need when I need it

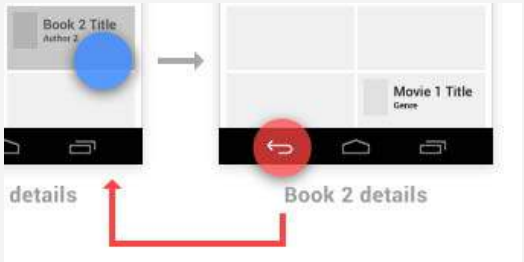
People get overwhelmed when they see too much at once. Break tasks and information into small, digestible chunks. Hide options that aren't essential at the moment, and teach people as they go.



Simplify User Life

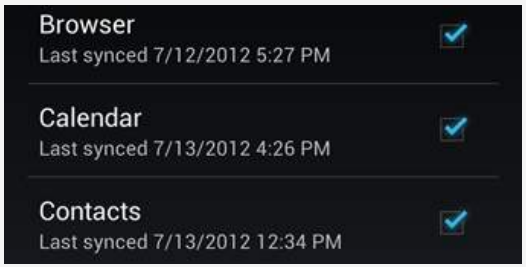
5. I should always know where I am

Give people confidence that they know their way around. Make places in your app look distinct and use transitions to show relationships among screens. Provide feedback on tasks in progress.



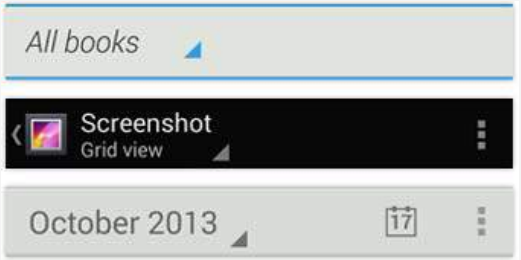
6. Never lose my stuff

Save what people took time to create and let them access it from anywhere. Remember settings, personal touches, and creations across phones, tablets, and computers. It makes upgrading the easiest thing in the world.



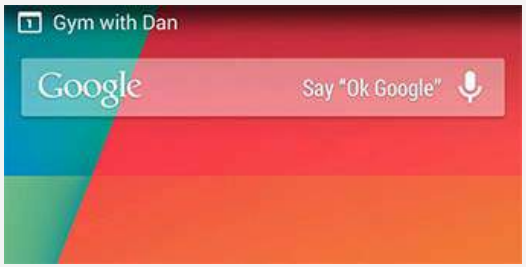
7. If it looks the same, it should act the same

Help people discern functional differences by making them visually distinct rather than subtle. Avoid modes, which are places that look similar but act differently on the same input.



8. Only interrupt me if it's important

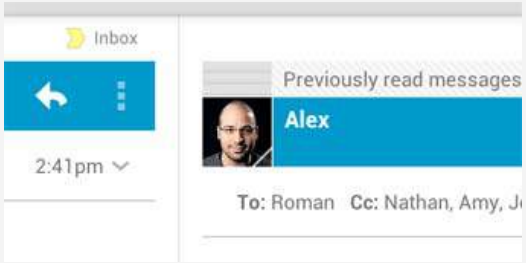
Like a good personal assistant, shield people from unimportant minutiae. People want to stay focused, and unless it's critical and time-sensitive, an interruption can be taxing and frustrating.



Make User Amazing

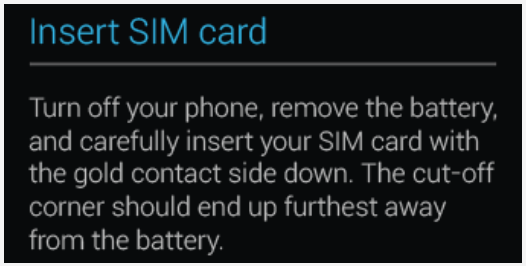
1. Give me tricks that work everywhere

People feel great when they figure things out for themselves. Make your app easier to learn by leveraging visual patterns and muscle memory from other Android apps. For example, the swipe gesture may be a good navigational shortcut.



2. It's not my fault

Be gentle in how you prompt people to make corrections. They want to feel smart when they use your app. If something goes wrong, give clear recovery instructions but spare them the technical details. If you can fix it behind the scenes, even better.

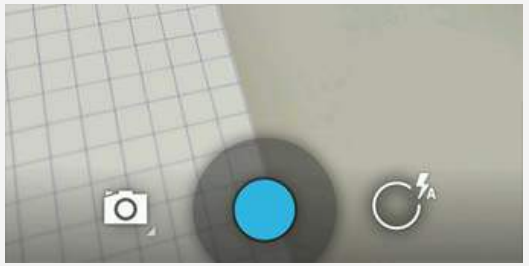


3. Do the heavy lifting for me

Make novices feel like experts by enabling them to do things they never thought they could. For example, shortcuts that combine multiple photo effects can make amateur photographs look amazing in only a few steps.

4. Make important things fast

Not all actions are equal. Decide what's most important in your app and make it easy to find and fast to use, like the shutter button in a camera, or the pause button in a music player.



附加資源

Mobile Design Pattern Gallery : UI Patterns for iOS, Android and More
http://www.mobiledesignpatterngallery.com/

Mobile Design Pattern Gallery
http://www.flickr.com/photos/mobiledesignpatterngallery/collections/

Interaction Pattern
http://www.androidpatterns.com/

Library Pattern
http://patterntap.com/

Android Market
http://www.appbrain.com/

Android Patterns
http://androidpttrns.com/

建議參考書籍

行動介面設計模式_歐萊禮 (Designing Mobile Interfaces)
- 行動介面元件說明書

行動介面設計模式圖鑑_歐萊禮 (Mobile Design Pattern Gallery: UI Patterns for Mobile Applications)
- 以視覺範例學習有效的介面設計模式，關於iOS、Android和其他系統的介面設計模式

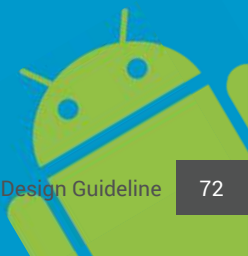
Check List For Android

1

P72-88

Navigation Anti-Patterns

http://youtu.be/Sww4omntVjs 詳細說明影片_英文



ANTI
PATTERN
1

Navigation in overflow

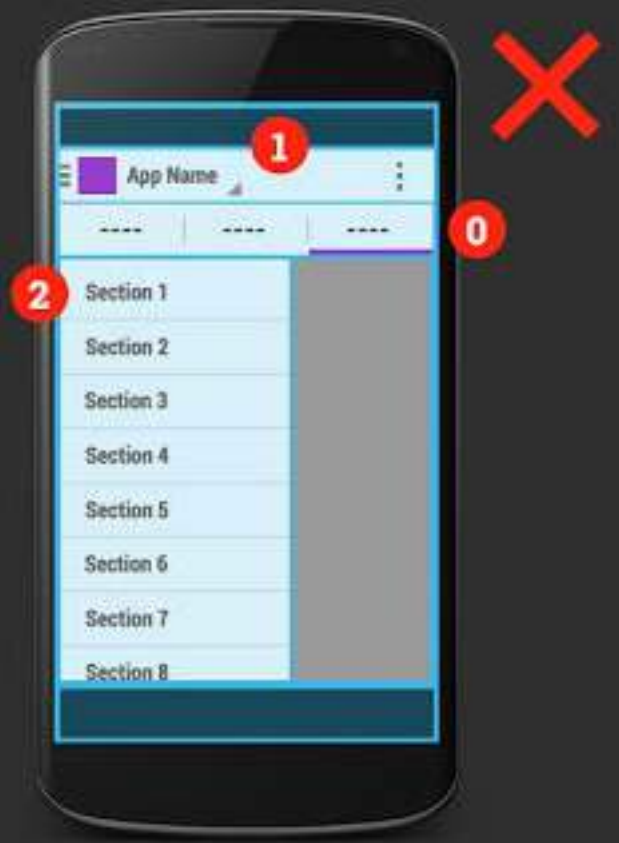
The action bar overflow is for secondary action, not core app navigation!

ANTI PATTERN

2

Wrong nav hierarchy

UI patterns should represent sections at the correct level of navigation hierarchy.



ANTI
PATTERN
3

Tabs that don’t swipe

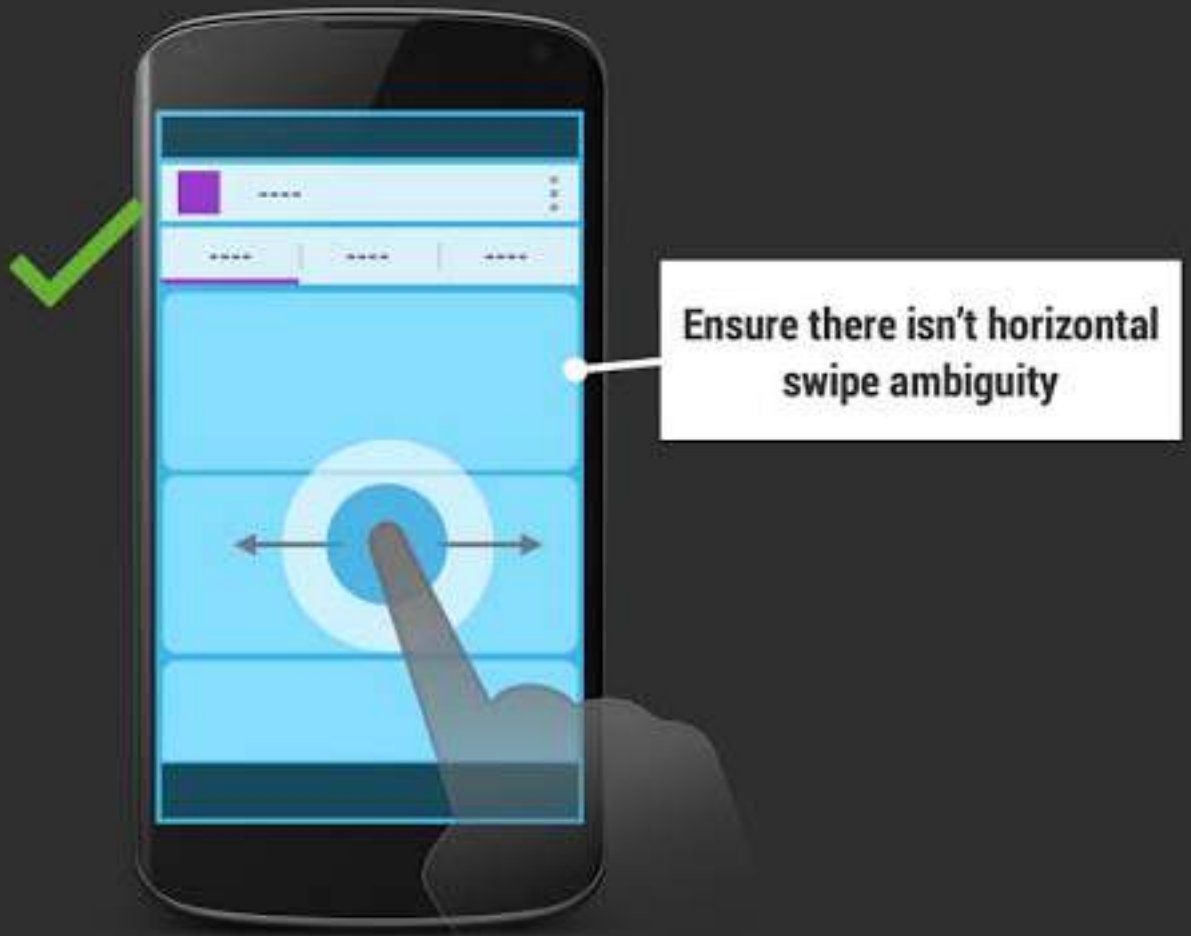
Any time you see tabs, you should be able to swipe their contents horizontally to switch tabs.

ANTI PATTERN

3

Tabs that don’t swipe

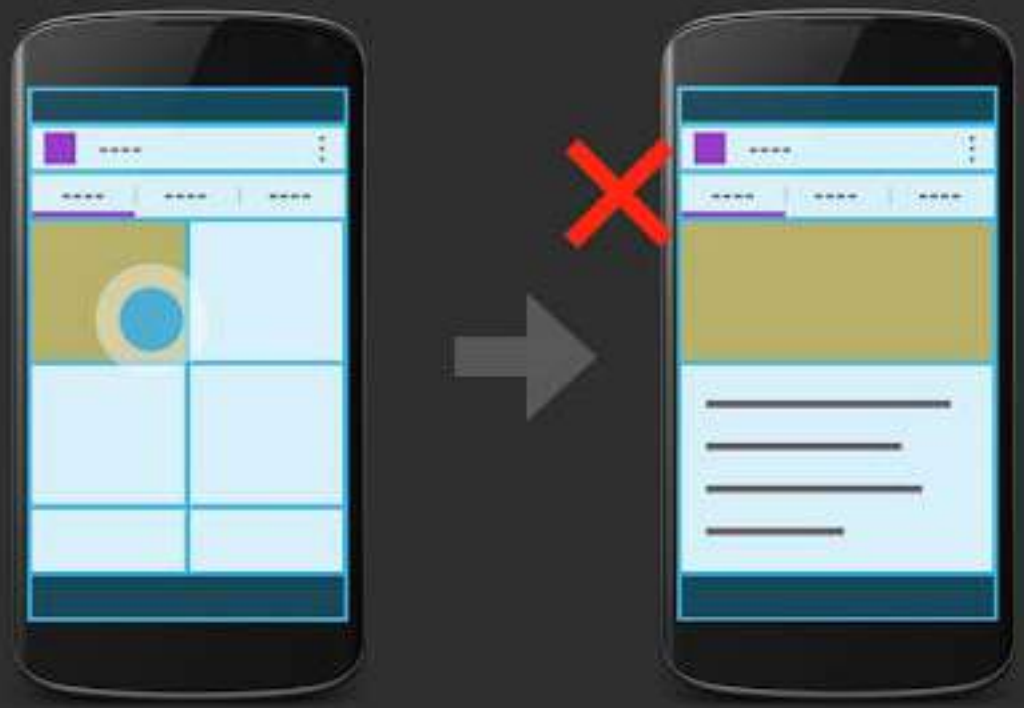
Any time you see tabs, you should be able to swipe their contents horizontally to switch tabs.



ANTI
PATTERN
4

Deep or persistent tabs

Tabs should be shallow, don't allow descendant navigation within a tab; don't keep global tabs around throughout the app.



ANTI
PATTERN
4

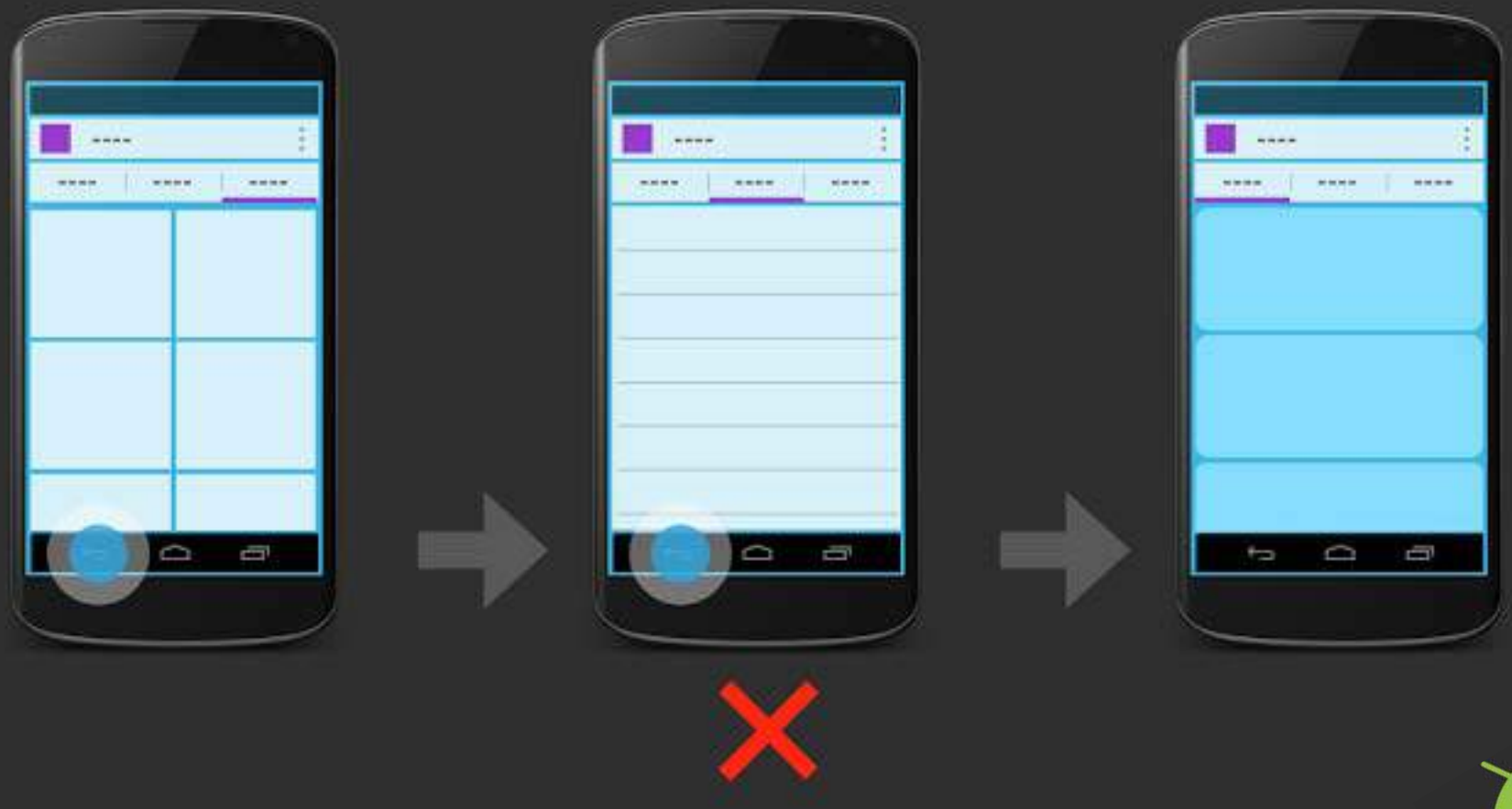
Deep or persistent tabs

Tabs should be shallow, don't allow descendant navigation within a tab; don't keep global tabs around throughout the app.

ANTI
PATTERN
5

Back traverses tabs

Pressing Back should take you to the previous screen, not the previous tab; tabs are shallow!



ANTI PATTERN

6

Back traverses drawer

Selecting a section in the drawer should reset your task back stack.

ANTI
PATTERN
7

Deep navigation drawers

The only depth in navigation drawers should be one optional level of collapsible sub-navigation.

ANTI PATTERN

7

Deep navigation drawers

The only depth in navigation drawers should be one optional level of collapsible sub-navigation.

ANTI PATTERN

7

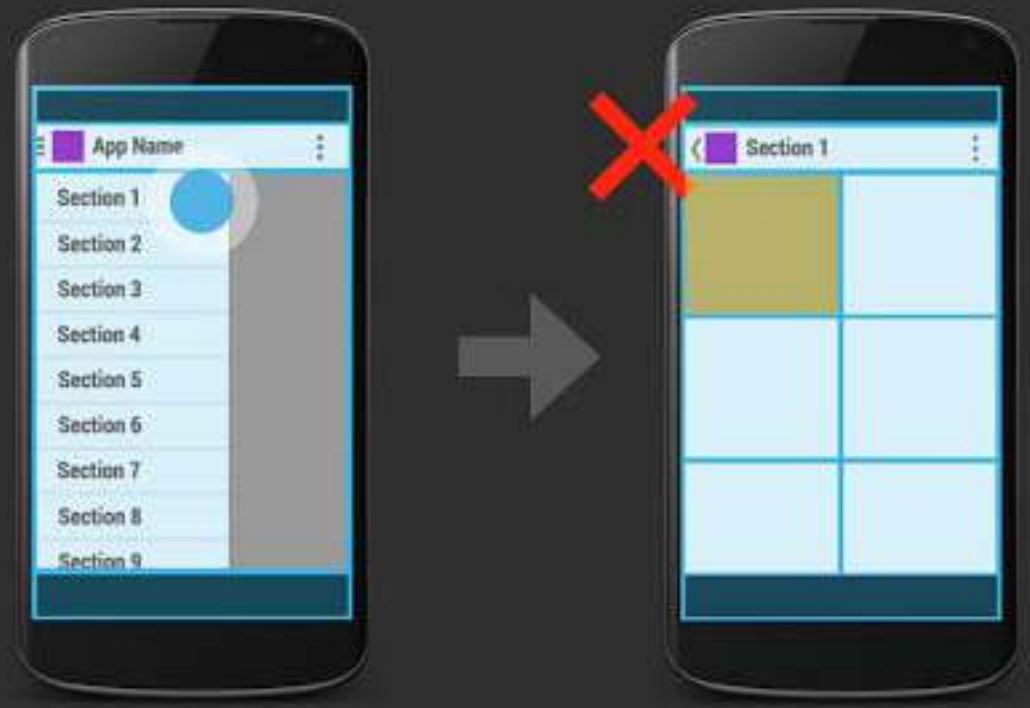
Deep navigation drawers

The only depth in navigation drawers should be one optional level of collapsible sub-navigation.

ANTI
PATTERN
8

Bad drawer transitions

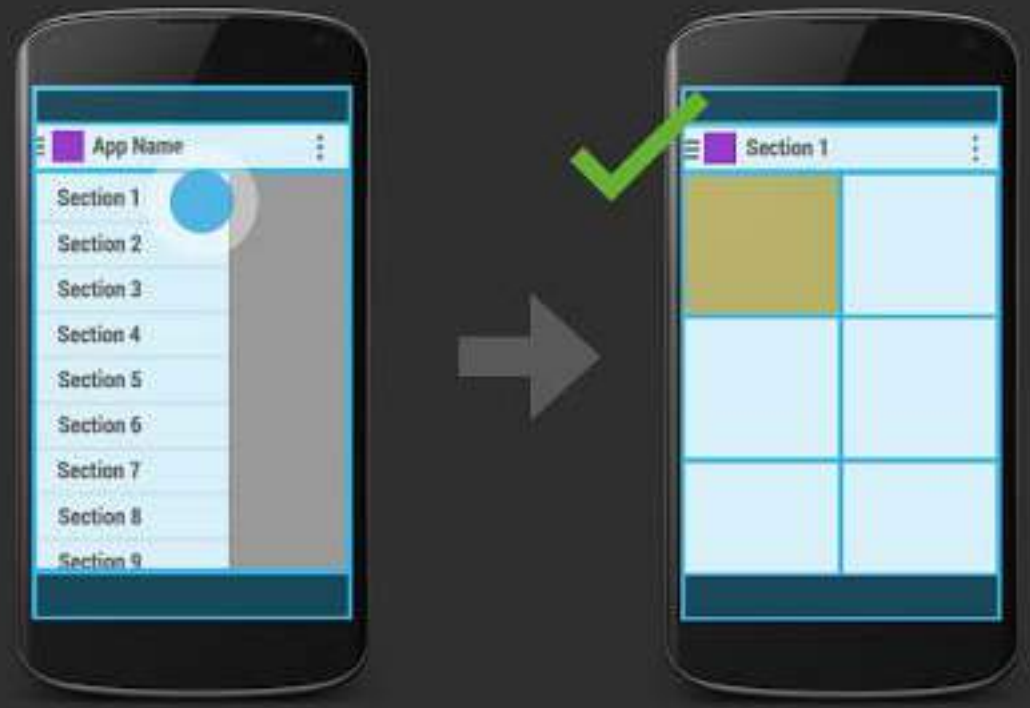
Screens listed in the drawer should have the drawer icon.
Avoid "new scree" transitions when selecting drawer items.



ANTI
PATTERN
8

Bad drawer transitions

Screens listed in the drawer should have the drawer icon.
Avoid "new scree" transitions when selecting drawer items.



ANTI
PATTERN
9

Never showing Up caret

The drawer indicator should only be shown on screens represented in the drawer; otherwise, show Up caret.

ANTI PATTERN

9

Never showing Up caret

The drawer indicator should only be shown on screens represented in the drawer; otherwise, show Up caret.

ANTI
PATTERN
10

Right-side navigation

Navigation and master-detail relationships should read left-to-right on Android.

P89-98

2 UX Anti-Patterns

UX ANTI PATTERN

1

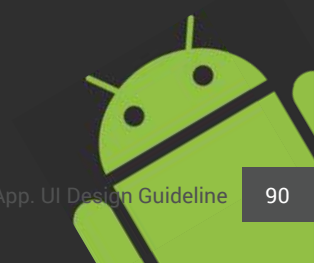
Can’t touch this !

Small touch targets

Should be at least 32dp tall, ideally 48dp.

No touch feedback

All touchable elements should have pressed and focused states.



UX ANTI PATTERN

2

Design ≠ Photoshop

Poor visual design decisions can make your app feel dated and even worse-amateurish.

Too much or too heavry chrome

Users aren't installing your app for the pretty buttons.

Visual stying inCOn$istEncY

Half holo, half widebeest?

Poor attention to detail

Metri c s/spacing, color, typography

UX ANTI PATTERN

3

Living in the past

If your app is living in 2009, your users are gonna have bed time.

Legacy menu button

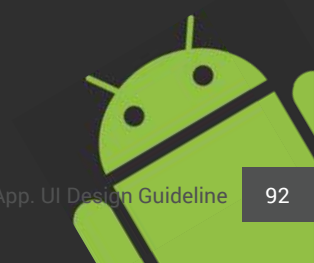
There's an action bar for that.

targetSdkVersion < 16

You know about API 16. Why doesn't your app?

Gingerbread-era UI elements

You should be partying like 2014.



UX ANTI PATTERN

4

Pure Android

Don't deviate from the Pure Android guidelines.

Right-pointing carets

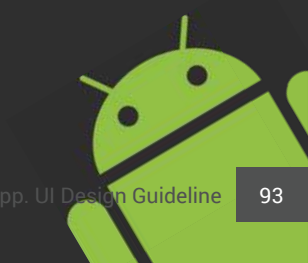
Descendant navigation has no horizontal spatial mapping on Android.

Bottom tab bars

Tabs belong at the top on Android.

"Borrowing" visual styling from other platforms

Android users expect Android apps.



UX ANTI PATTERN

5

The navigation is my brand

Don't reinvent the wheel.App navigatin is well-defined on Android.

Custom top bars

And poorly implemented action bar clones

DEVS ActionBarCompat

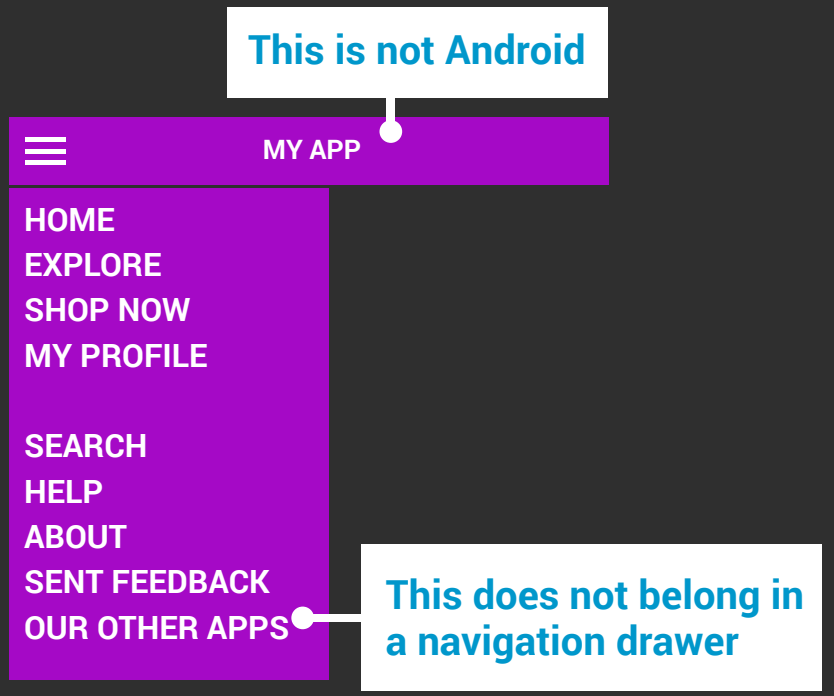
Bad navigation drawers

Navigation drawer is for primary app navigation

Search and settings belong in the overflow

Don't slide the action abr or screen content

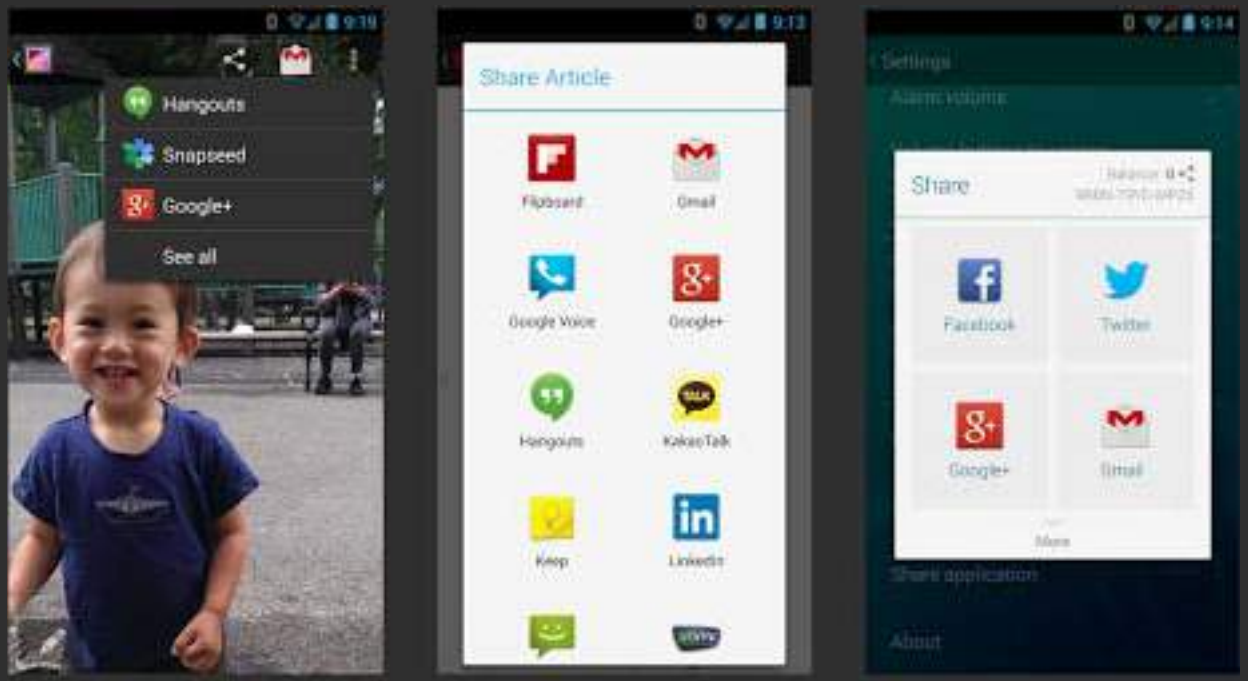
DEVS DrawerLayout + ActionBarDrawerToggle



UX ANTI PATTERN 6

Custom non-Android sharing

This violates Pure Android guideline.



"Share" functionality to specific networks should at most complement built-in ACTION_SEND-based sharing.

UX ANTI PATTERN

7

Mimicking native w/WebView

Core UI and content should be rendered w/the Android framework.

It's not genuine. Kinda like lying to your users.

It's results in poor performance.

Don't package up a web app as an APK just to "be on Android".

Just be a web app that lives on the web...
we love web apps!

UX ANTI PATTERN

8

Poor onbaarding UX

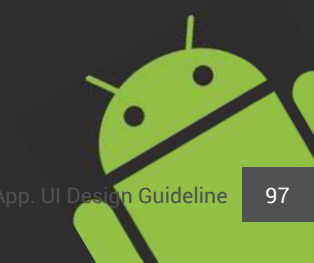
Up and running in 5 seconds or your uninstall on the house!

Don’t require registation up front.

Don’t use splash screens.

Do integrate with popular login providers.

Do make your app’s value proposition clear.
why do I need your app?



UX ANTI PATTERN

9

Android = Phones in portrait

Support tablets and phones in both portrait and landscape.

Poor tablet experiences reflect negatively on your brand.

Only an hour or two of XML can get you halfway there.

Poor tablet experiences reflect negatively on your brand.

UX ANTI PATTERN

10

“Thou must load this...or else!”

Modal loading dialogs are awful. Especially those that don’t respect the Back button.Epic sadface.

Thanks!

MIS / Usage Interface Design