

# Application Structure

Apps come in many varieties that address very different needs. For example:

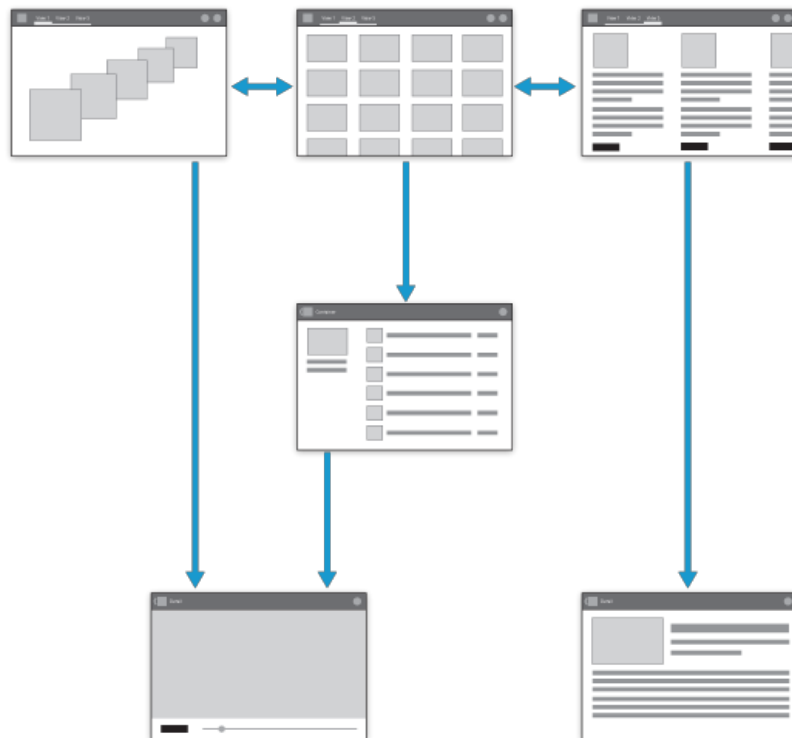
- Apps such as Calculator or Camera that are built around a single focused activity handled from a single screen
- Apps such as Phone whose main purpose is to switch between different activities without deeper navigation
- Apps such as Gmail or the Play Store that combine a broad set of data views with deep navigation

Your app's structure depends largely on the content and tasks you want to surface for your users.

## General Structure

---

A typical Android app consists of top level and detail/edit views. If the navigation hierarchy is deep and complex, category views connect top level and detail views.



### Top level views

The top level of the app typically consists of the different views that your app supports. The views either show different representations of the same data or expose an altogether different functional facet of your app.

### Category views

Category views allow you to drill deeper into your data.

### Detail/edit view

The detail/edit view is where you consume or create data.

## Top Level

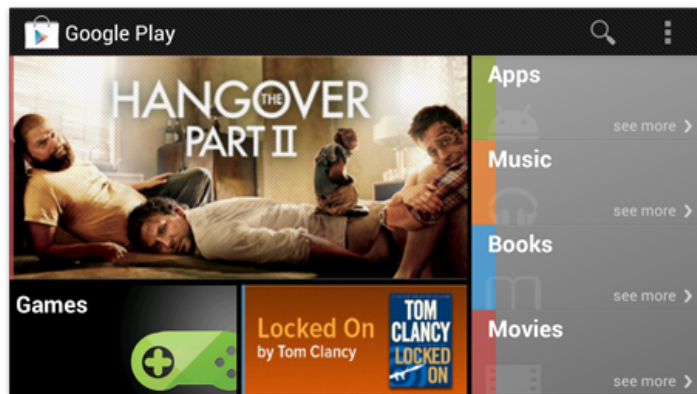
---

The layout of your start screen requires special attention. This is the first screen people see after launching your app, so it should be an equally rewarding experience for new and frequent visitors alike.

Ask yourself: "What are my typical users most likely going to want to do in my app?", and structure your start screen experience accordingly.

### Put content forward

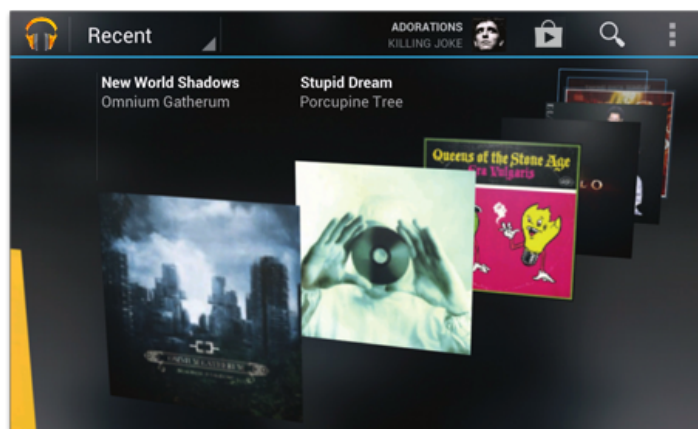
Many apps focus on the content display. Avoid navigation-only screens and instead let people get to the meat of your app right away by making content the centerpiece of your start screen. Choose layouts that are visually engaging and appropriate for the data type and screen size.



*The Play Store app's start screen primarily allows navigation into the stores for Apps, Music, Books, Movies, and Games. It is also enriched with tailored recommendations and promotions that surface content of interest to the user. Search is readily available from the action bar.*

### Create an identity for your app

Creating an identity for your app goes beyond the action bar. Your app communicates its identity through its data, the way that data is arranged, and how people interact with it. Especially for media-rich applications, try to create unique layouts that showcase your data and go beyond the monotony of simple list views.



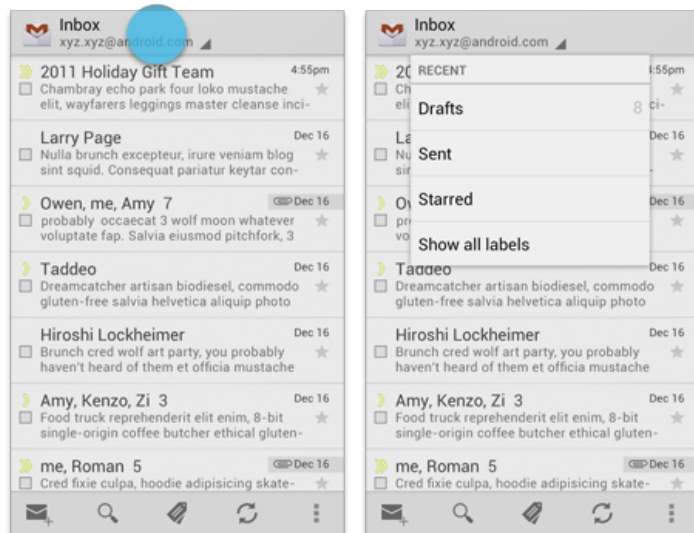
*The 3D carousel celebrates cover art and establishes a unique identity for the Music app. Defaulting to the Recent view keeps the focus on music the user has been listening to lately.*

### Set up action bars for navigation and actions

All screens in your app should display action bars to provide consistent navigation and surface important actions.

At the top level, special considerations apply to the action bar:

- Use the action bar to display your app's icon or title.
- If your top level consists of multiple views, make sure that it's easy for the user to navigate between them by adding view controls to your action bar.
- If your app allows people to create content, consider making the content accessible right from the top level.
- If your content is searchable, include the Search action in the action bar so people can cut through the navigation hierarchy.



*Email is about productivity, so an efficient, easy-to-skim list with higher data density works well. Navigation supports switching between accounts and recent labels. Icons for creating a new message or searching are prominent in the split action bar at the bottom.*

## Top Level Switching With View Controls

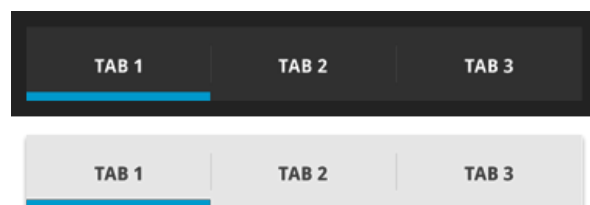
The top level communicates your app's capabilities by introducing the user to the major functional areas. In many cases the top level will consist of multiple views, and you need to make sure that the user can navigate between them efficiently. Android supports a number of view controls for this task. Use the control that best matches your app's navigation needs:

### Fixed tabs

*Fixed tabs* display top-level views concurrently and make it easy to explore and switch between them. They are always visible on the screen, and can't be moved out of the way like scrollable tabs. *Fixed tabs* should always allow the user to navigate between the views by swiping left or right on the content area.

Use tabs if:

- You expect your app's users to switch views frequently.
- You have a limited number of up to three top-level views.
- You want the user to be highly aware of the alternate views.



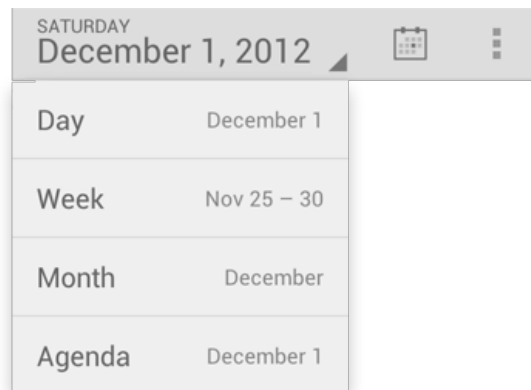
*Default fixed tabs shown in Holo Dark & Light.*

## Spinners

A *spinner* is a drop-down menu that allows users to switch between views of your app.

Use a spinner in the main action bar if:

- You don't want to give up the vertical screen real estate for a dedicated tab bar.
- The user is switching between views of the same data set (for example: calendar events viewed by day, week, or month) or data sets of the same type (such as content for two different accounts).



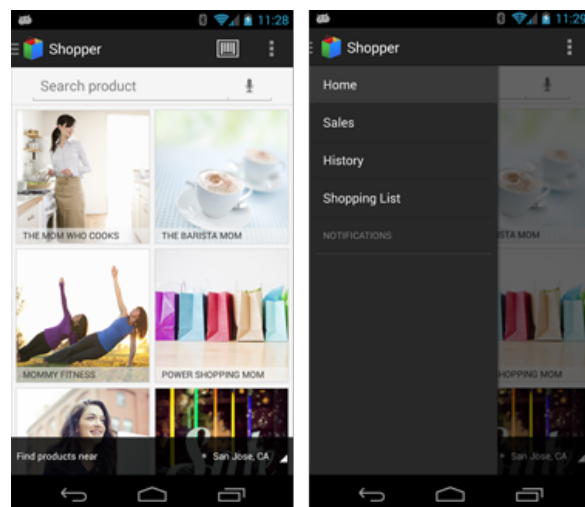
Action bar spinner from Calendar application.

## Navigation drawers

A *navigation drawer* is a slide-out menu that allows users to switch between views of your app. It can hold a large number of items and is accessible from anywhere in your app. Navigation drawers show your app's top-level views, but can also provide navigation to lower-level screens. This makes them particularly suitable for complex apps.

Use navigation drawers if:

- You don't want to give up the vertical screen real estate for a dedicated tab bar.
- You have a large number of top-level views.
- You want to provide direct access to screens on lower levels.
- You want to provide quick navigation to views which don't have direct relationships between each other.
- You have particularly deep navigation branches.



Navigation drawer from the Shopper app.

## Don't mix and match

After choosing the best top-level navigation for your app, don't mix and match patterns. For example, if you decide to use tabs for top-level switching, don't add a drawer, even if your navigation branches are deep. In this case, the navigation drawer would simply duplicate the information on the tabs, confusing your users.

## Categories

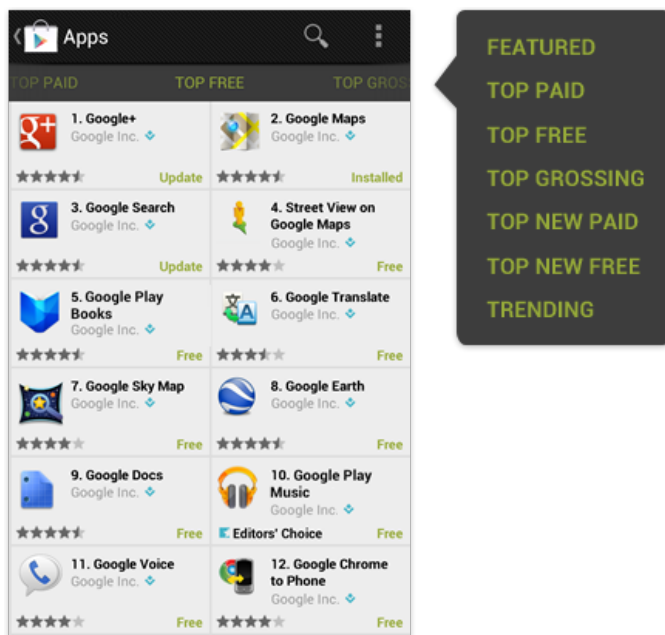
Generally, the purpose of a deep, data-driven app is to navigate through organizational categories to the detail level, where data can be viewed and managed. Minimize perceived navigation effort by keeping your apps shallow.

Even though the number of vertical navigation steps from the top level down to the detail views is typically dictated by the structure of your app's content, there are several ways you can cut down on the perception of onerous navigation.

### Use tabs to combine category selection and data display

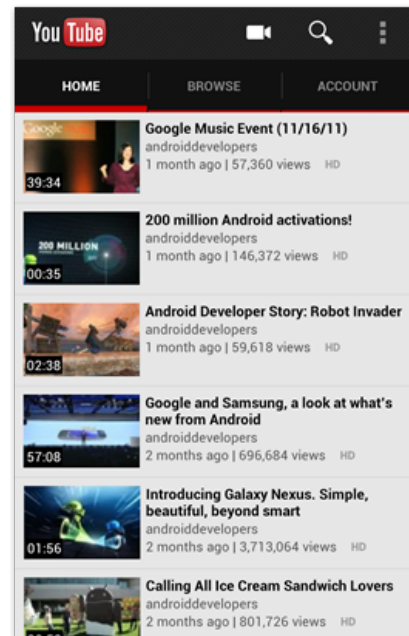
This can be successful if the categories are familiar or the number of categories is small. It has the advantage that a level of hierarchy is removed and data remains at the center of the user's attention. Navigating laterally between data-rich categories is more akin to a casual browsing experience than to an explicit navigation step.

If the categories are familiar, predictable, or closely related, use scrolling tabs (where not all items are in view simultaneously). Keep the number of scrolling tabs at a manageable level to minimize navigational effort. Rule of thumb: no more than 5–7 tabs.



*The Play Store app uses tabs to simultaneously show category choice and content. To navigate between categories, users can swipe left/right on the content.*

If the categories in the tabs are not closely related, favor fixed tabs, so that all categories are in view at the same time.

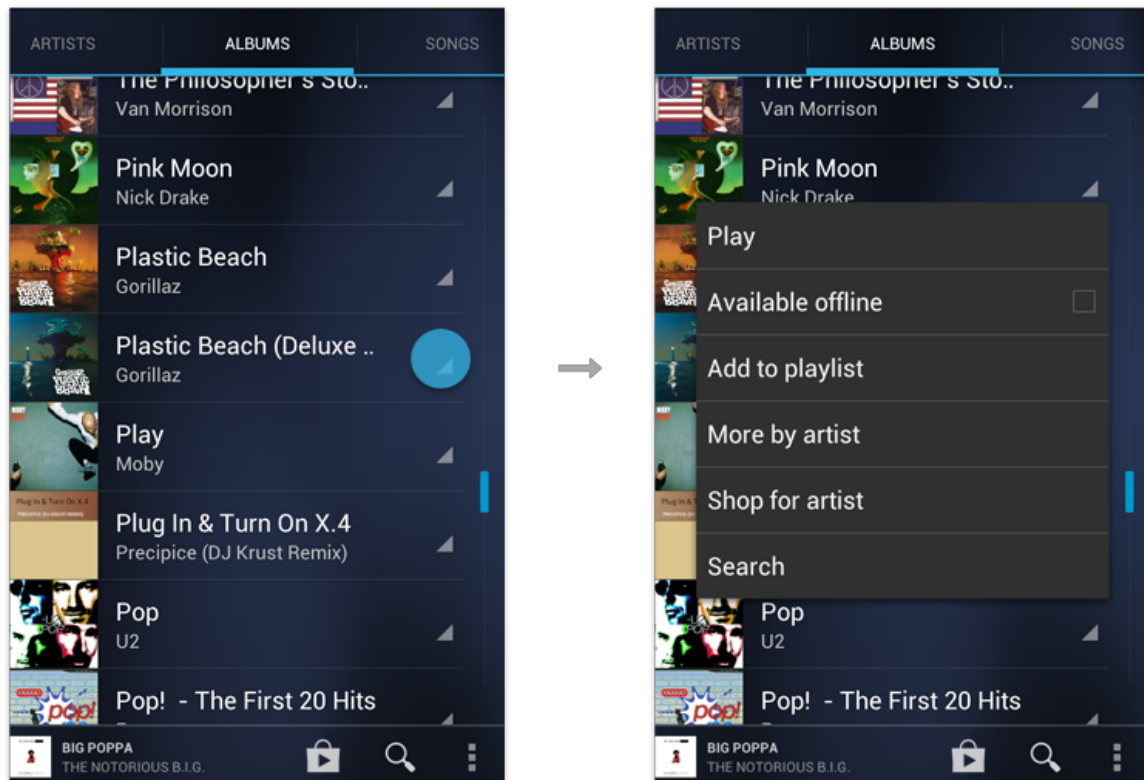


*YouTube uses fixed tabs to switch between different, relatively unrelated functional areas.*

For more discussion, see the [Tabs \(/design/building-blocks/tabs.html\)](https://design/building-blocks/tabs.html) design guide.

### Allow cutting through hierarchies

Take advantage of shortcuts that allow people to reach their goals quicker. To allow top-level invocation of actions for a data item from within list or grid views, display prominent actions directly on list view items using drop-downs or split list items. This lets people invoke actions on data without having to navigate all the way down the hierarchy.



Music allows the user to act upon a data item (song) from within the category view (album), thereby removing the need to navigate all the way down to the song's detail view.

### Acting upon multiple data items

Even though category views mostly serve to guide people to content detail, keep in mind that there are often good reasons to act on collections of data as well.

For example, if you allow people to delete an item in a detail view, you should also allow them to delete multiple items in the category view. Analyze which detail view actions are applicable to collections of items. Then use multi-select to allow application of those actions to multiple items in a category view.

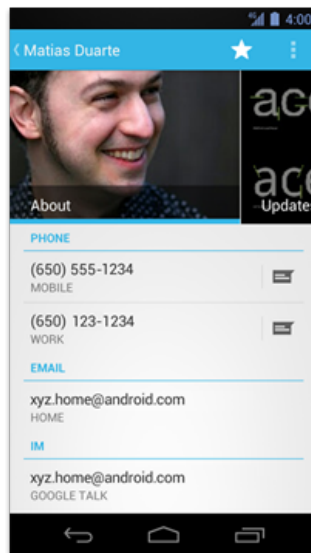
For more discussion, see the [Selection \(/design/patterns/selection.html\)](https://design.google.com/patterns/selection.html) design guide.

## Details

The detail view allows you to view and act on your data. The layout of the detail view depends on the data type being displayed, and therefore differs widely among apps.

## Layout

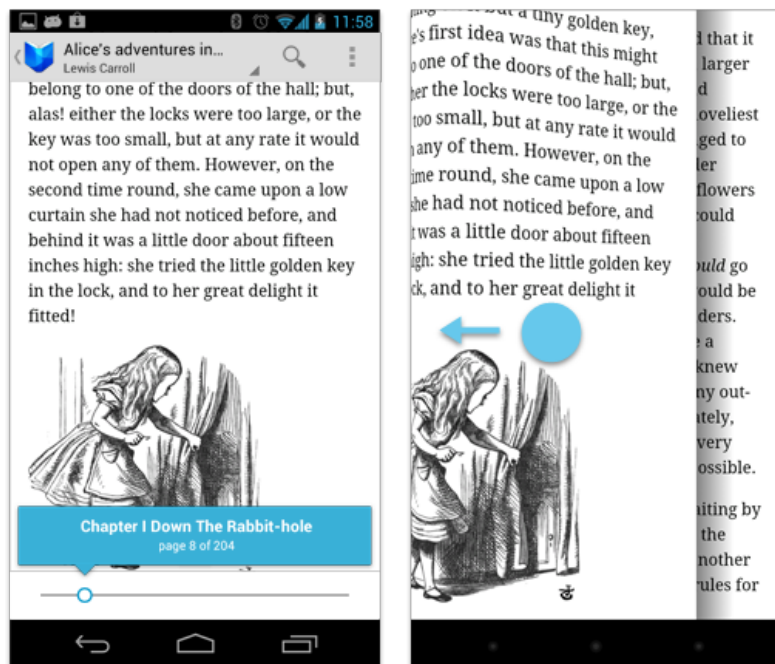
Consider the activities people will perform in the detail view and arrange the layout accordingly.



*The purpose of the People app's detail view is to surface communication options. The list view allows for efficient scanning and quick access of phone numbers, email addresses and other information items. Split items are used to combine calling and messaging into one compact line item.*

## Lights-out mode

Immersive content like media and games is best experienced full screen without distractions. But that doesn't mean you can't also offer actions on the content like sharing, commenting, or searching. If the user hasn't interacted with any of the controls after a short period of time, automatically fade away the action bar and all system UI affordances so the user can lean back and enjoy the content. We call this lights-out mode. Later, if the user wants to take some action, they can touch anywhere on the screen to exit lights-out mode and bring back the controls.

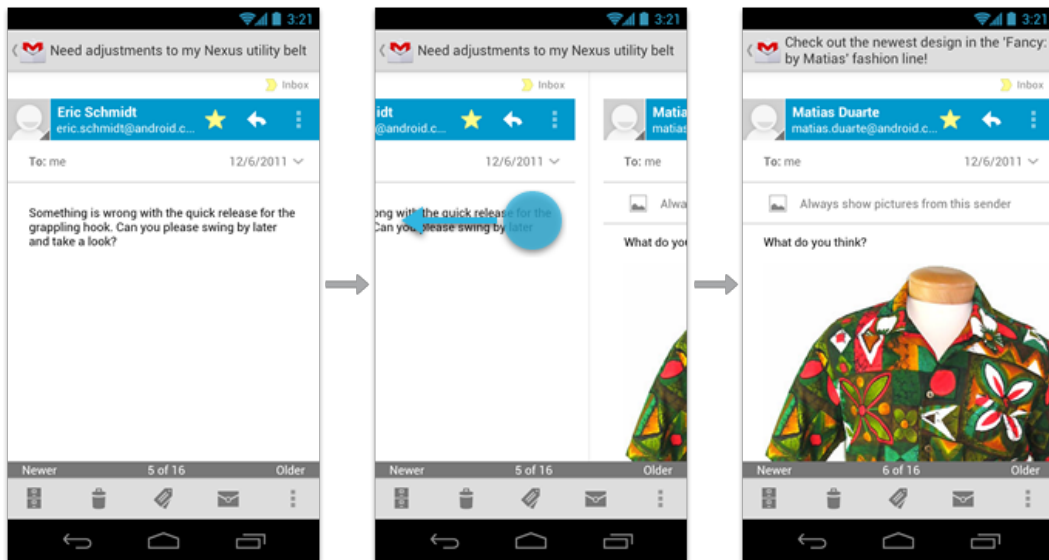


*Google Books' detail view replicates the immersive experience of reading an actual book through lights-out mode and a page-flip animation.*

## Make navigation between detail views efficient

If your users are likely to want to look at multiple items in sequence, allow them to navigate between items from within the detail view. Use swipe views or other techniques, such as thumbnail view controls, to achieve this.





*Gmail using swipe views to navigate from detail view to detail view.*



*In addition to supporting swipe gestures to move left or right through pages, Magazines provides a thumbnail view control that lets people quickly jump to specific pages.*

For more discussion, see the [Swipe Views \(//design/patterns/swipe-views.html\)](http://design/patterns/swipe-views.html) design guide.

## Checklist

- Find ways to display useful content on your start screen.
- Use action bars to provide consistent navigation.
- Keep your hierarchies shallow by using horizontal navigation and shortcuts.
- Use multi-select to allow the user to act on collections of data.
- Allow for quick navigation between detail items with swipe views.