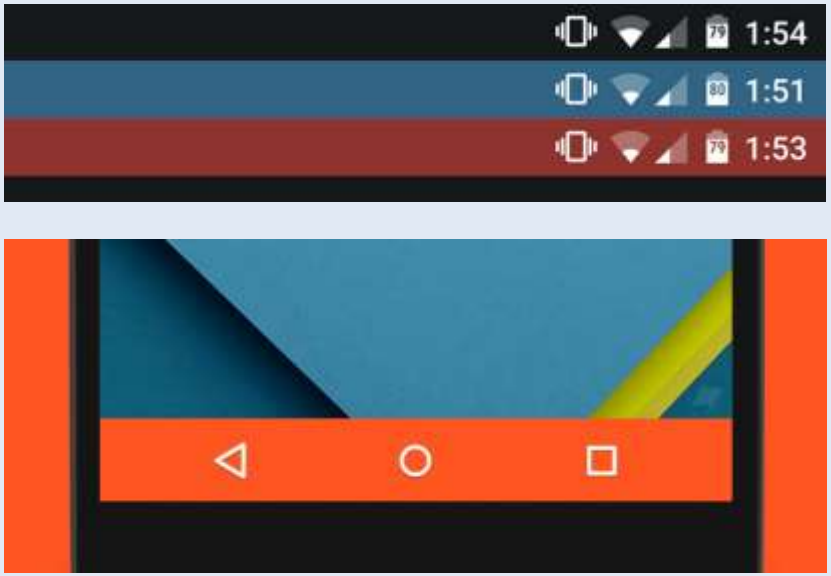
CSIS 3375 – 001

Priya Kandhadai

Building Android UI

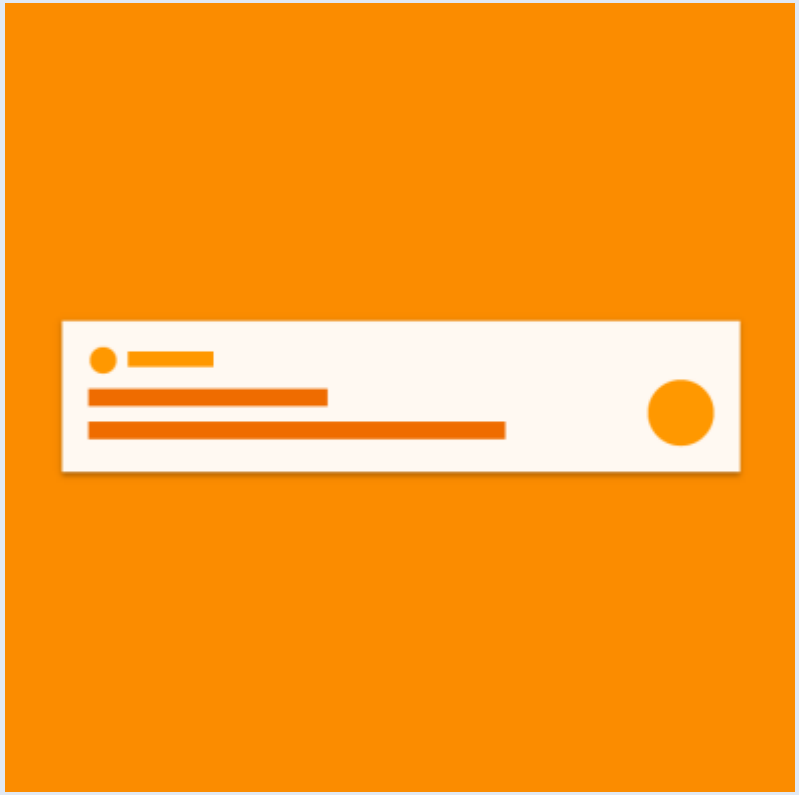
Android UI: System Bars

- System Bars:
  - Provide system relevant information
  - Android status bar: gives preview of notifications, phone settings information
    - Height: 24dp
  - Android bottom navigation bar
    - Height: 48dp



Android UI: Notifications

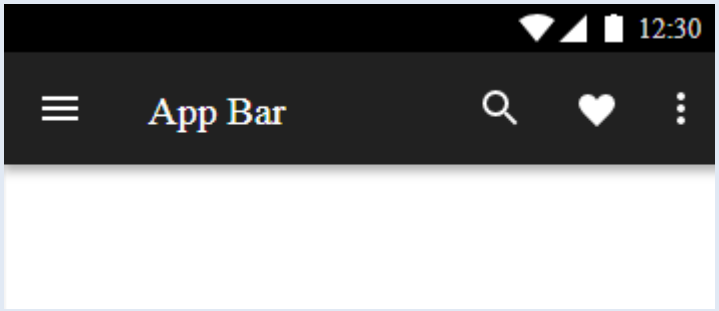
- Notifications: provide short, timely, and relevant information about your app when it's not in use
  - Background task progress
  - Communications from other users
  - App-relevant reminders (sparse use)
- Snack bars and toasts



Android UI: App bar

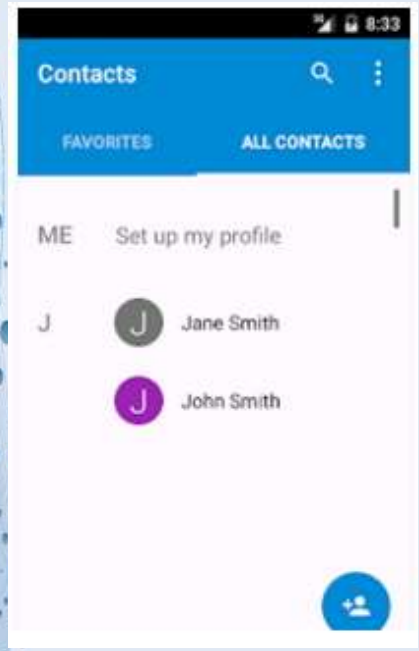
- App bar: A tool bar specific to the app that sits at the top of app
- Right below status bar
- Previously called action bar
- Standard height: 56dp on mobile devices
- May have another tool bar at the bottom (bottom tool bar)

- App bar may be hidden
  - Reading apps

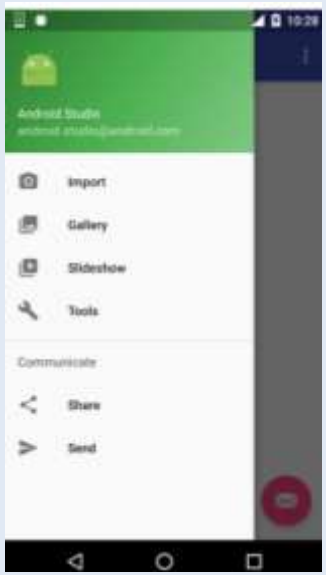


Android UI – Tabs and Navigation drawer

- Tabs: usually at top below app bar
  - No more than 2-3 tabs

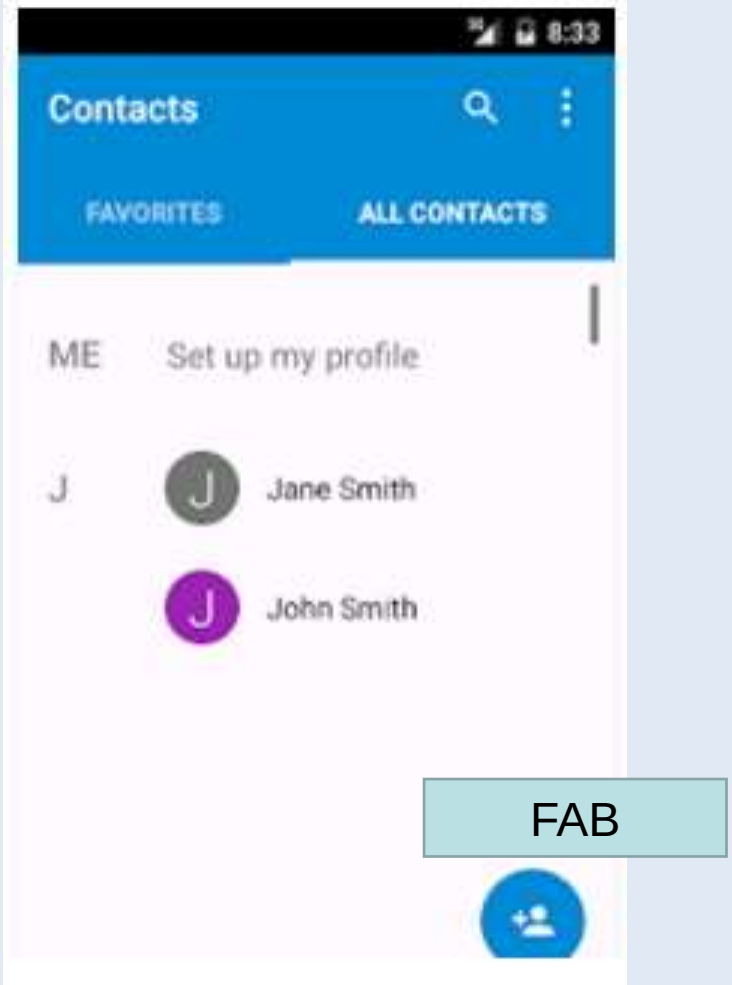


- Navigation drawer:
  - Several app sections
  - Full screen: sheet front of app bar

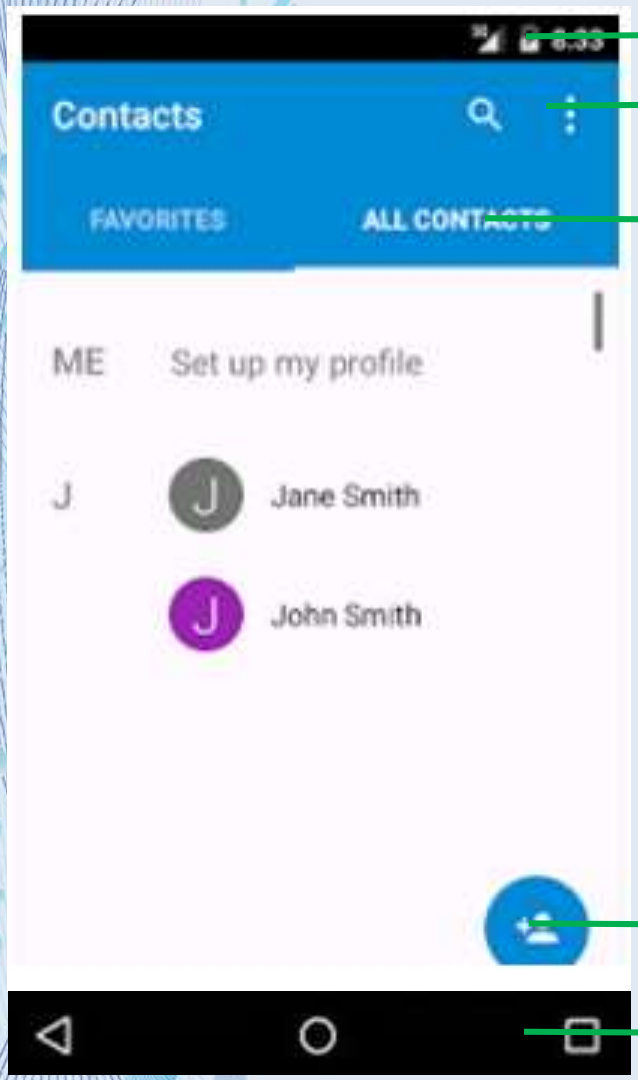


Android UI: FAB

- Floating Action Button
  - FAB
  - Typically accent color (secondary color)
  - Draw attention to important action
  - Eg: new event, new contact, new note in apps
  - Default: 56 X 56dp



Android UI



Status Bar

App Bar

Tabs

FAB

Navigation Bar

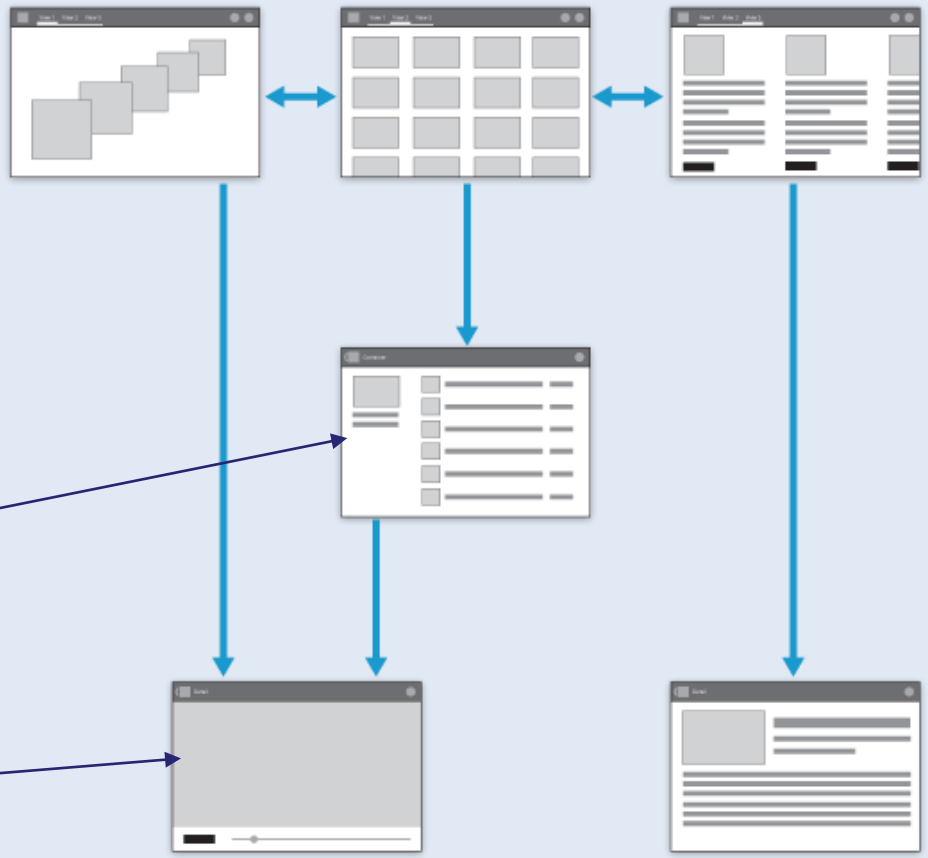
Newer Android Apps

- No Menu Button
- Long Press denotes selection rather than context menu (right-click)

- Notifications: white or fully transparent icons.
  - Cross-app compatability especially important
- Avoid styles from other platforms
  - iOS and Android styles are drastically different

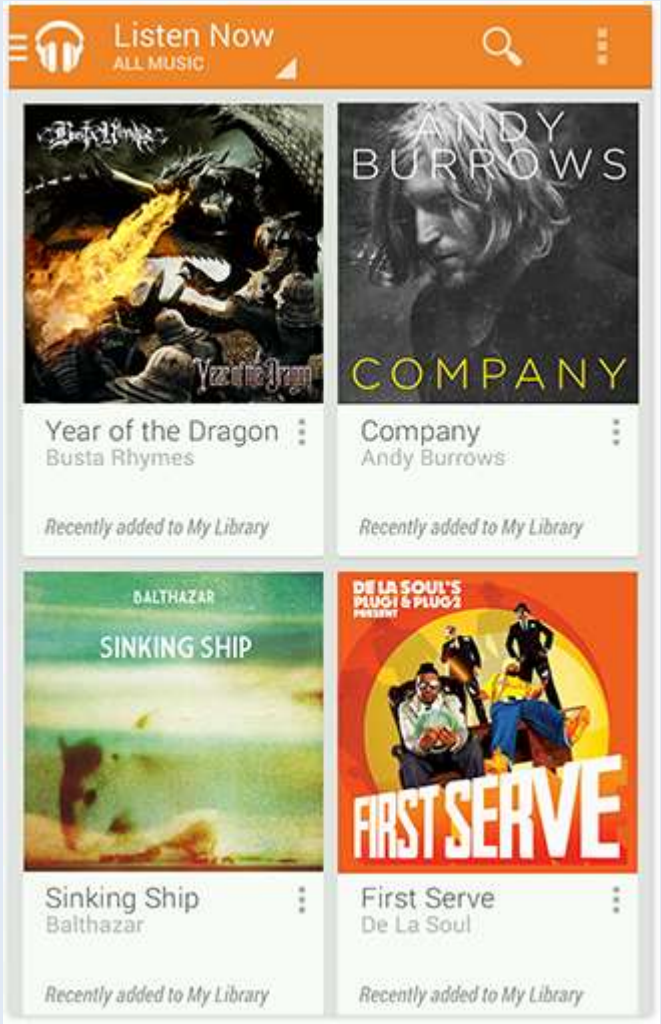
Basic Android Structure

- Top level views: different views that your app supports.
  - different representations of the same data
  - different functional facet of your app.
- Category views
  - drill deeper into your data.
- Detail/edit view: Consume or create data.



Top-Level

- Use bright and engaging layouts
- What do users of the app typically want to do?
- Identify different top-level views in your apps
- Add content to top-level view wherever possible
  - Engaging and fun!



App Bars

- App Bar: display your app's icon or title.
- If your top level consists of multiple views, add view controls to your app bar
- If your content is searchable, include the Search action in the app bar so people can cut through the navigation hierarchy.

- Identify functionality or utility of app bars in your app

Fixed Tabs

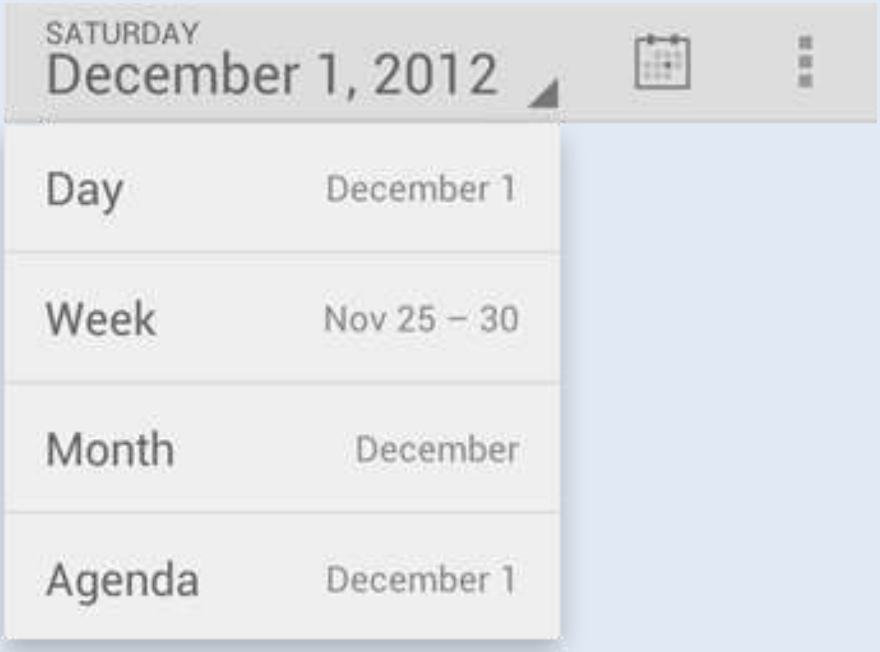
- Remains on the screen always at the top-level
- Allows multiple top-level views

- User needs to switch between views frequently
- User needs to be made aware of the alternate views
- No more THAN three fixed tabs

Spinners

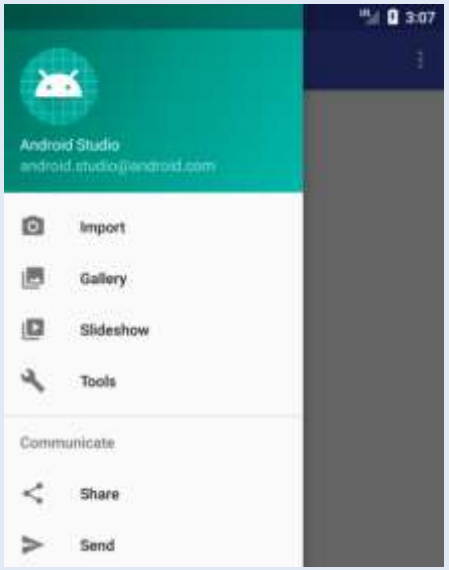
A spinner: drop-down menu
Use spinner in app bar if:

- Instead of dedicated fixed tab
- The user is switching between views
  - of the same data set: calendar events viewed by day, week, or month or
  - data sets of the same type: such as content for two different accounts



Navigation drawers

- **Use navigation drawers if**:
- You don’t want to waste navigation space
- 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.

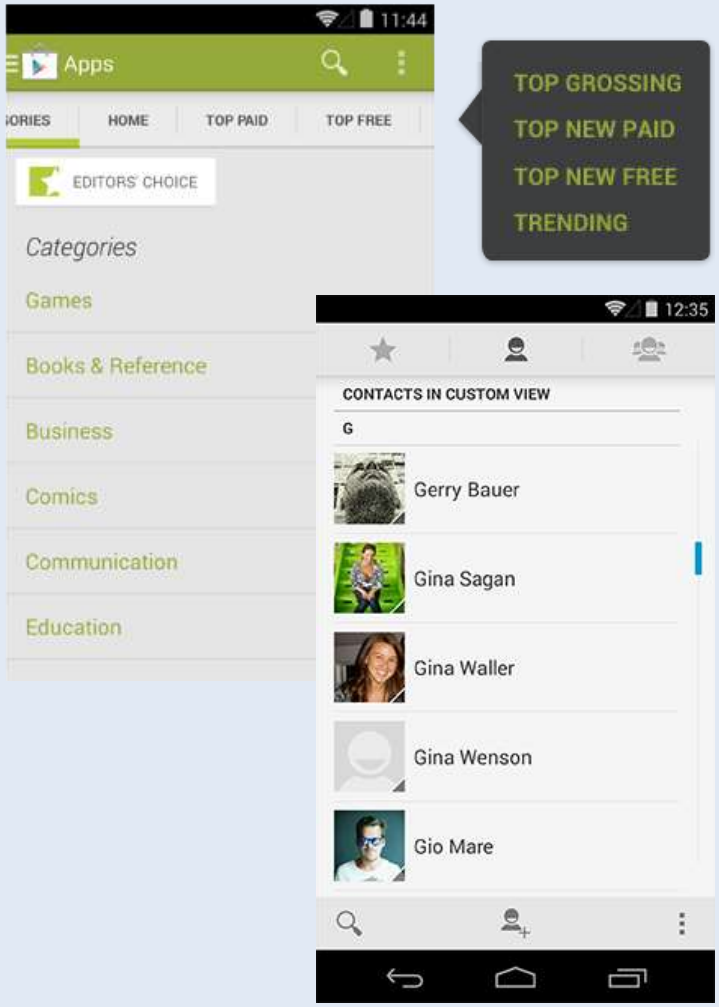
You have particularly deep navigation branches.

Top-level

- Use Tabs, Spinners and Navigation drawer appropriately
  - Don’t mix and match
  - Stick with one depending on your need

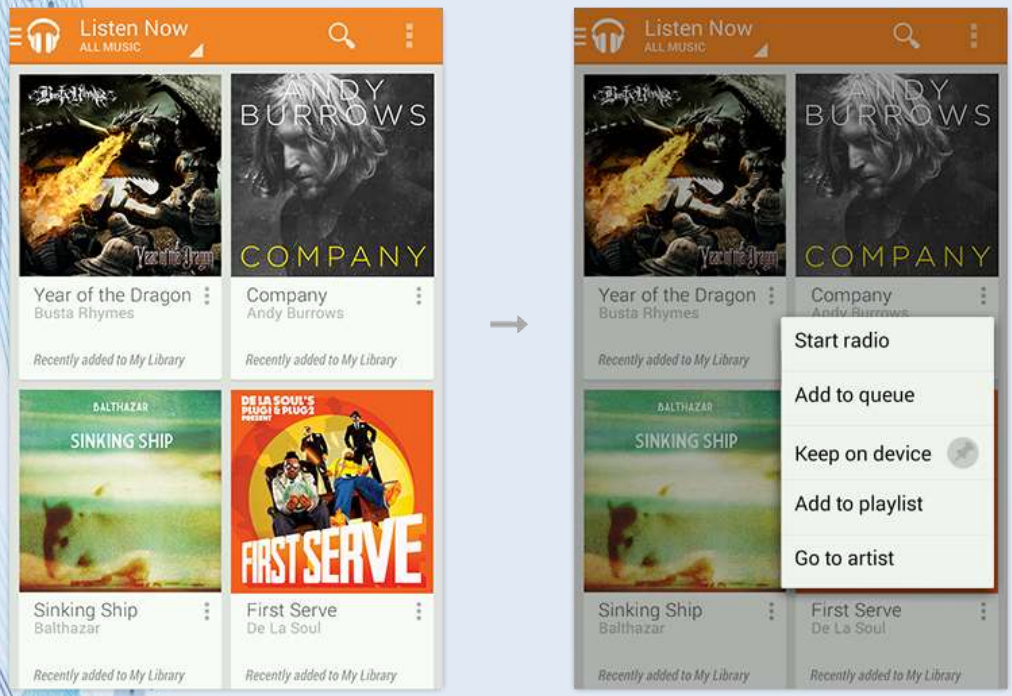
Category-View

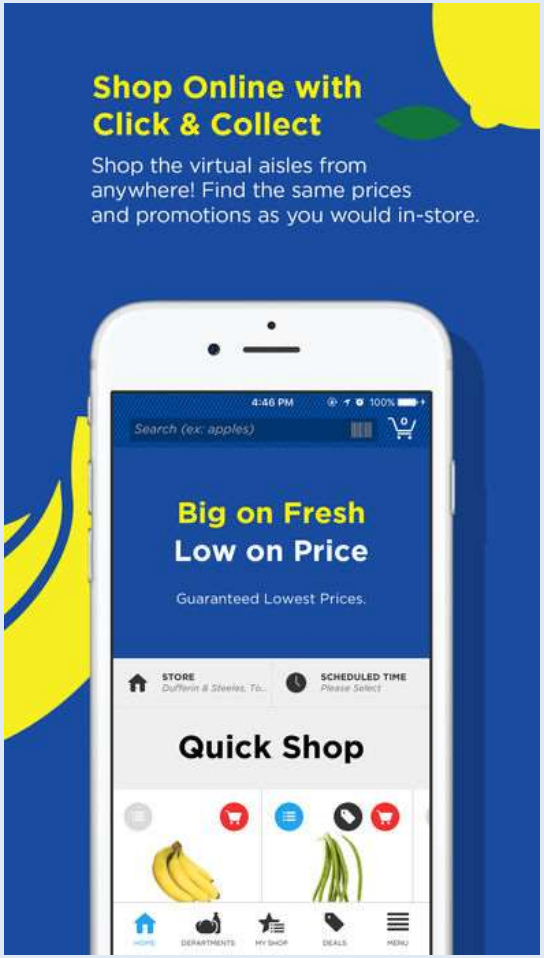
- Used in apps that have deep hierarchical structures
  - Use scrolling tabs for related categories
  - Use fixed tabs for unrelated categories



Allow cutting through hierarchies

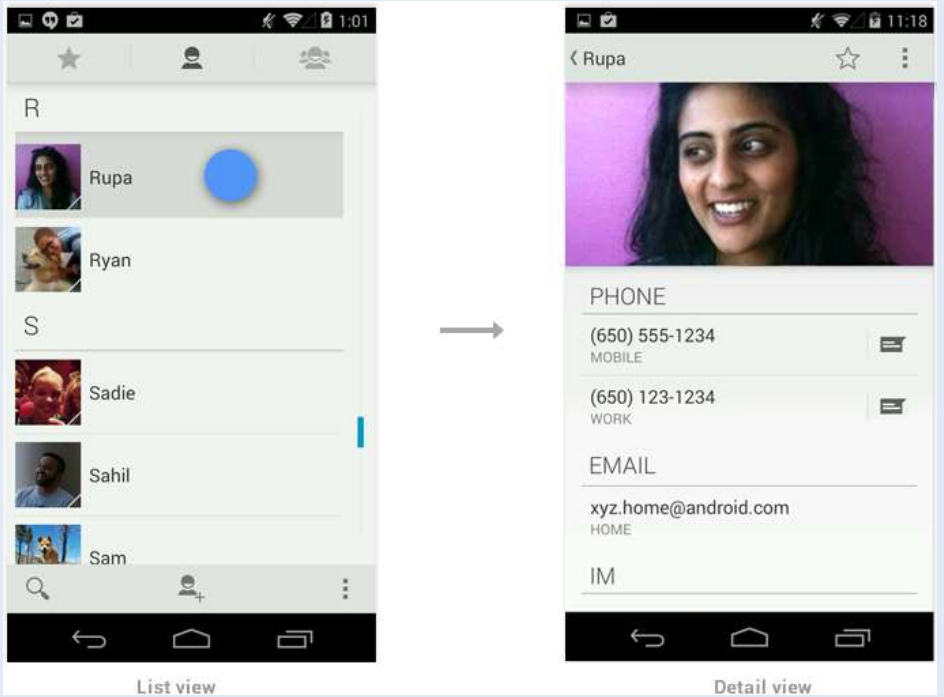
- Display prominent actions directly on the list items





Detail View

- Less is more
- Think about order of content processing to arrange layout
- Allow navigation between multiple detail views
  - Swipe to retrieve next item



Overall Android App Structure

- Find ways to display useful content on your start screen.
- Use app bars to provide consistent navigation.
- Maintain shallow hierarchies: use 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.

Android UI

- Basic Android UI components
  - What? When to use?
- Android UI structure
  - Overall layout
  - Three levels of Views
- Top-level vs. Category vs. Detail Views
  - What? When to use?
- Use of tabs, spinners, navigation drawers
- Design consideration for Android UI structure

Research Activity: Identify these in TWO apps of your choice

- Basic Android UI components: system bar, app bar, FAB, tabs, navigation drawer, notifications, android navigation bar, bottom navigation
- Android UI structure: Identify top-level, category and detail views
- Use of tabs, spinners, navigation drawers

Fixed Tabs

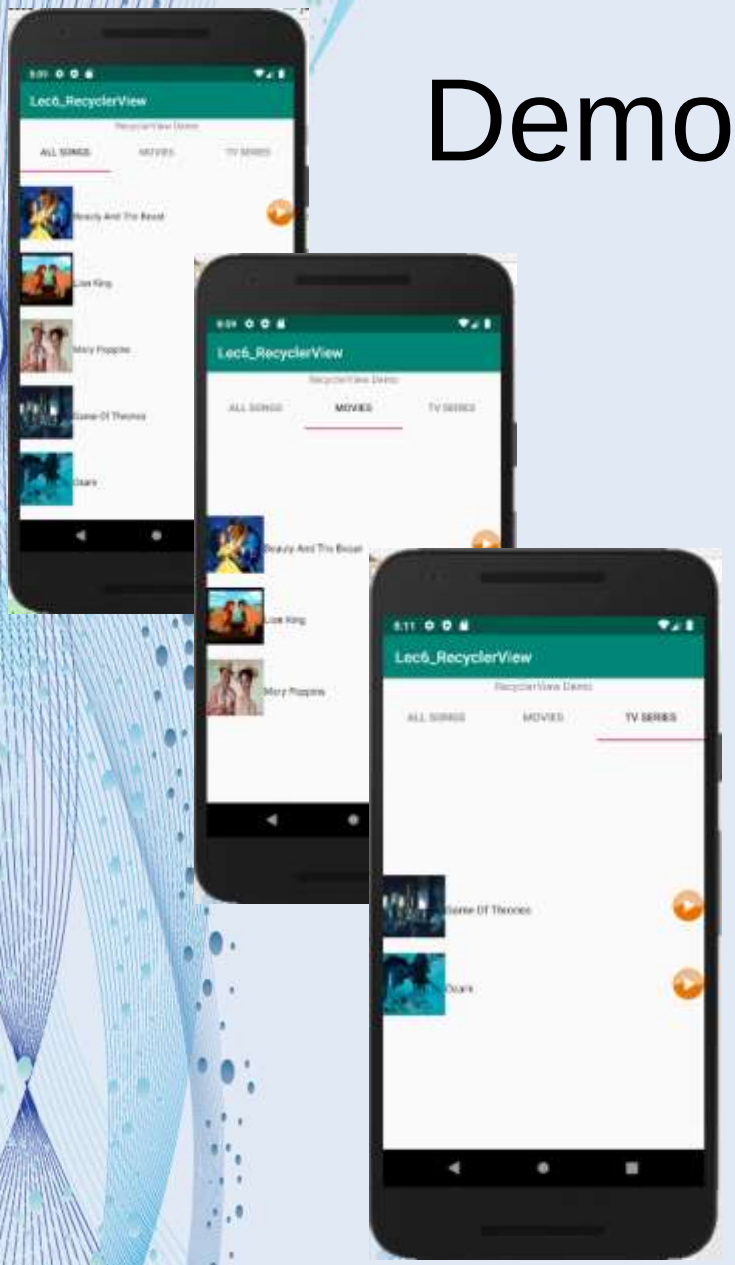
- Remains on the screen always at the top-level
- Allows multiple top-level views

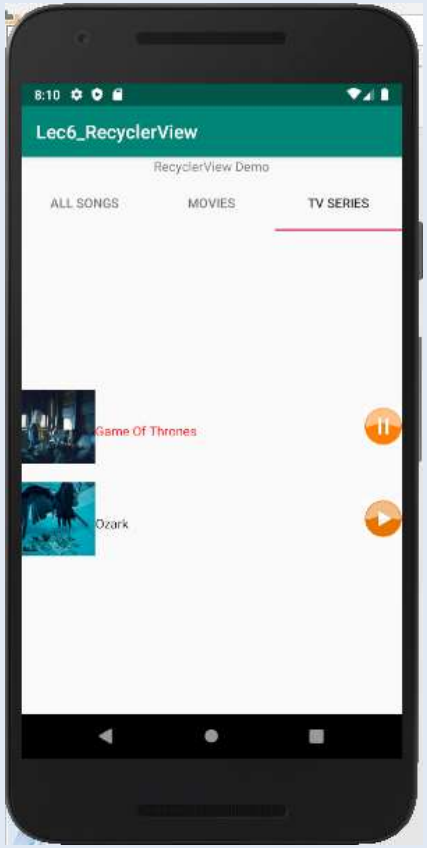
- User needs to switch between views frequently
- User needs to be made aware of the alternate views
- No more THAN three fixed tabs

Fixed Tabs with RecyclerView

- Add three fixed tabs
  - All Items
  - Movies
  - TV Shows
- When each tab is clicked
  - Create a list of entries for each
- Use RecyclerView to implement the list using an external layout
- When each item in the list is clicked, it should toggle text color between red and dark gray

Demo SnapShot

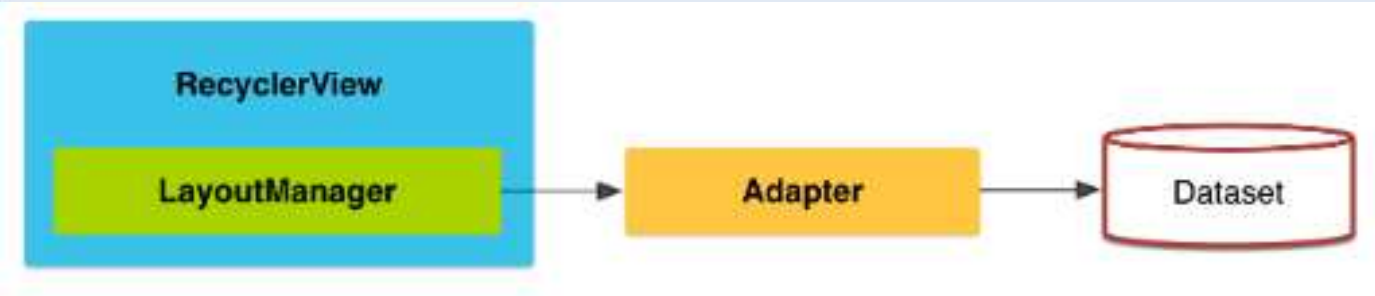




Click Event

RecyclerView

- Needs custom Adapter
- Required ViewHolder in Adapter
- Has no built-in click listeners – offers full flexibility on click events
- Customizable Item Layouts
- Item Animations



Components of RecyclerView

- LayoutManagers: used for arranging items in recycler view
  - `Linear Layout Manager:` items in a vertical or horizontal scrolling list.
  - `GridLayoutManager`: items in a grid.
  - `StaggeredGridLayoutManager`: items in a staggered grid.

Components of RecycerView

- RecyclerView.Adapter:
  - Links Data source to the adapter: at the constructor or other methods
  - Inflate external Layout, create a ViewHolder class to hold the recycled view from this inflated layout
    - Add click events to the views if needed inside the view holder constructor
    - Instantiate this ViewHolder
  - Use the ViewHolder instance to bind new data to the recycled view
  - Use getCount() to figure out how many times the view is recycled

RecyclerView User Events

- User Event that changes the UI within an item
- User Event that changes the UI within the recycler view
- User Event that changes the UI outside the recycler view