Step 1: define user interactions

• Our ultimate goal is to specify the app’s screens and their relationships
• The first step is to specify the ways in which the user may interact with the app
  • An ERD diagram is one possible way
  • You may also create a simple list of possible user interactions
Step 2: create a screen map

- Define a screen list and the relationships between the screens
- An arrow from screen A to B implies that screen B should be directly reachable from screen A

Step 3: refine the screen map

The screen map should be refined based on Android UI design guidelines (and in particular Material Design)

Definition: section-related screens

- *Section-related* screens represent different sections of information about the parent
  - For example, one section may show textual information about an object
  - While another may provide a map of the object’s geographic location
- The number of section-related screens for a given parent is generally small
Section-related screens: basic navigation patterns

• Define a parent screen having
  • Buttons, or
  • Fixed-size list views, or
  • Text links (less recommended)
  • Tabs (recommended)

• Upon selecting one of these targets
  • The child screen is opened
  • Replacing the current parent entirely

Definition: collection-related screens

• Collection-related screens represent individual items in the collection represented by the parent

Note: buttons and other simple targets are rarely used for representing items in a collection
Collection-related screens: navigation patterns

- **Often, vertically scrolling lists** are the most straightforward and familiar kind of interface
  - Especially for textual information
- **For more visual or media-rich content items such as photos or videos, you may use**
  - Vertically scrolling grids of items
  - Horizontally scrolling lists (**carousels**)  

- **Avoid deep list-based navigation**, i.e., lists that lead to more lists which lead to even more lists
- **Larger screens** - vertical lists can lead to awkward user interactions and poor use of whitespace on larger screens. One solution is to **provide additional info** such as text summaries, that fills the available horizontal space. Another way is to provide additional information in a **separate horizontal pane**.

Lateral navigation

- So far, we have discussed **descendant navigation**
  - Allowing users to descend 'down' a screen hierarchy into a child screen
- **Lateral navigation** allows users to access sibling screens
Lateral navigation: tabs

- Using tabs is a very popular solution for lateral navigation
- Tabs are most appropriate for small sets (4 or fewer) of section-related screens
- Tabs should be located at the top of the screen, and should not be aligned to the bottom of the screen
- With tabs, users can navigate quickly between related screens, without needing to first revisit the parent

Lateral Navigation: horizontal paging (swipe views)

- This pattern applies best to collection-related sibling screens
- A single child screen is presented one at a time
- Users are able to navigate to sibling screens by touching and dragging the screen horizontally

Often, another UI element indicates the current page and available pages, to provide more context to the user
Multi-pane layouts

• Larger screens have much more available screen space and thus are able to present multiple panes of content
• In landscape, panes are usually ordered from left to right in increasing detail order
• Users are especially accustomed to multiple panes on larger screens
• Moreover, multiple panes are necessary to avoid too much whitespace

Multi-pane layouts in landscape work quite well. However, in portrait orientation, horizontal space is more limited, so you may need to design a separate layout for this orientation. This link presents possible portrait layouts

Putting it all together

• In the following links, see a possible result of applying the UI patterns to the initial screen map we saw earlier
  • http://developer.android.com/training/design-navigation/example-wireframe-phone.svg
  • http://developer.android.com/training/design-navigation/example-wireframe-tablet.svg

The following link explains how to implement the previously mentioned navigation patterns: http://developer.android.com/training/implementing-navigation/index.html

Note: this official guide was probably written before Material Design was introduced, so use it together with Material Design guidelines.