STACKABLE PANELS
MOCHI

2012 November 1 | **VERSION 02**

Design Lead
Esther Leong (esther.leong@palm.com)

Visual Design
Bryson Ahlstrom (bryson.ahlstrom@palm.com)
Liron Damir (liron.damir@palm.com)

Engineering Lead
Jeremy Thomas (jeremy.thomas@palm.com)
Kevin Schaaf (kevin.schaaf@palm.com)

TABLE OF CONTENTS

SUMMARY

Stacking Panels are sliding panes that slide in and stack on top of each other. Users can move between panels by sliding them left or right.

GOALS

- Make stacking panels scalable for different device sizes.

- Define panel behavior on different screen sizes (device size or orientation)

- To give examples of proper panel usage.

RECOMMENDATIONS

Stacking panels CAN be used for...

- Hierarchical navigation--navigation that only allows you to drill into a child through its parent (examples: file systems, email)

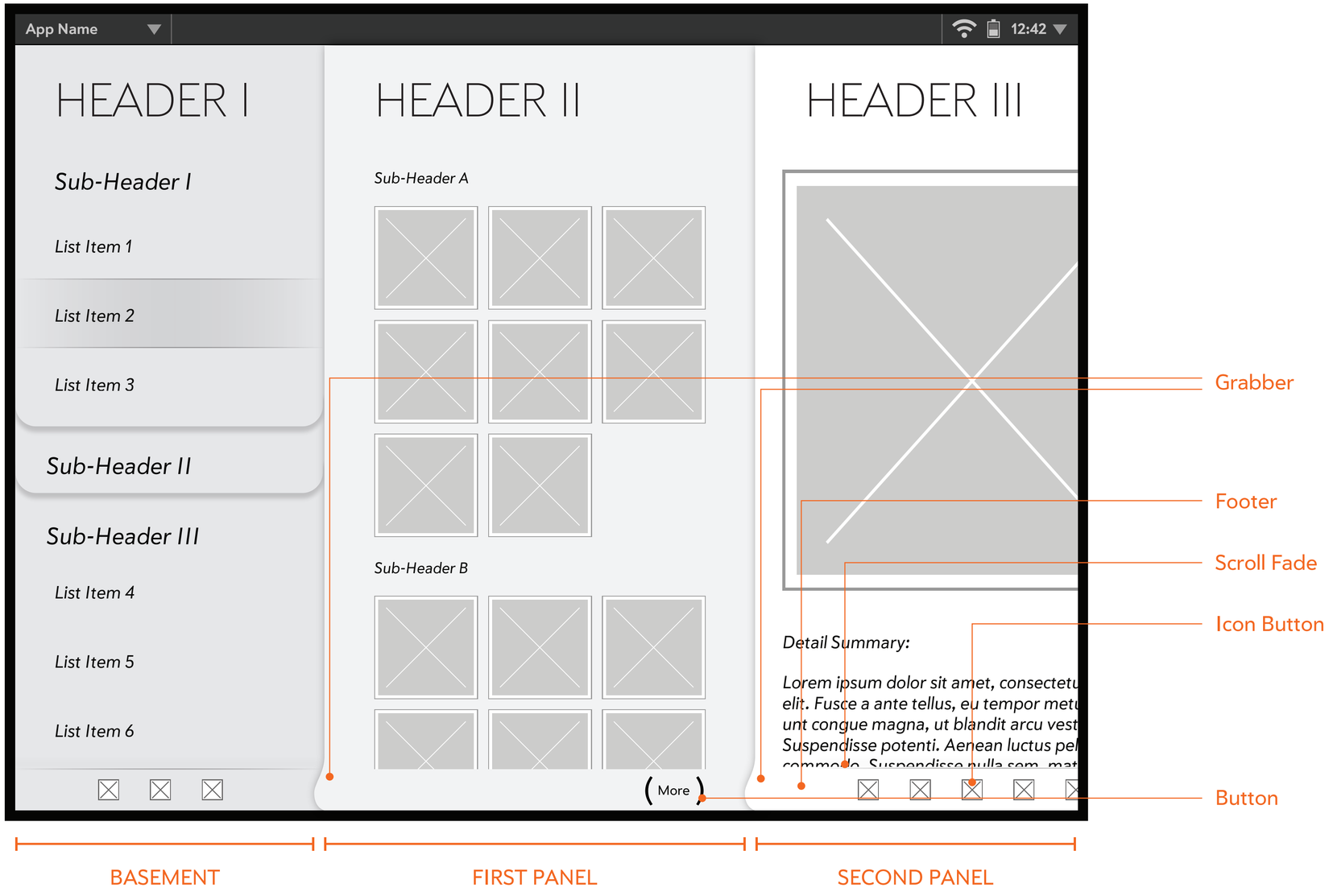
- Flat navigation--all panels are peers (examples: browser tabs, multiple open files in the same app)

Stacking panels should NOT be used for...

- Linear navigation--step by step navigation (examples: wizards, anything sequential)

STRUCTURE

STACKING PANELS EXAMPLE



Panels

Panels slide in and stack one on top of the other. Panels only move horizontally and must have a grabber.

Grabber

Visual indicators that can be used to manipulate a panel. They fold over when they are pushed against the edge of the screen.

Hit Areas

- Grabbers can accept tap, swipe, or drag.
- Areas without conflict can accept swipe and drag (this includes tappable objects such as buttons)
- Areas with conflict won't move a panel (e.g., swipe-able list items, zoomed-in image that can be panned)

Footer

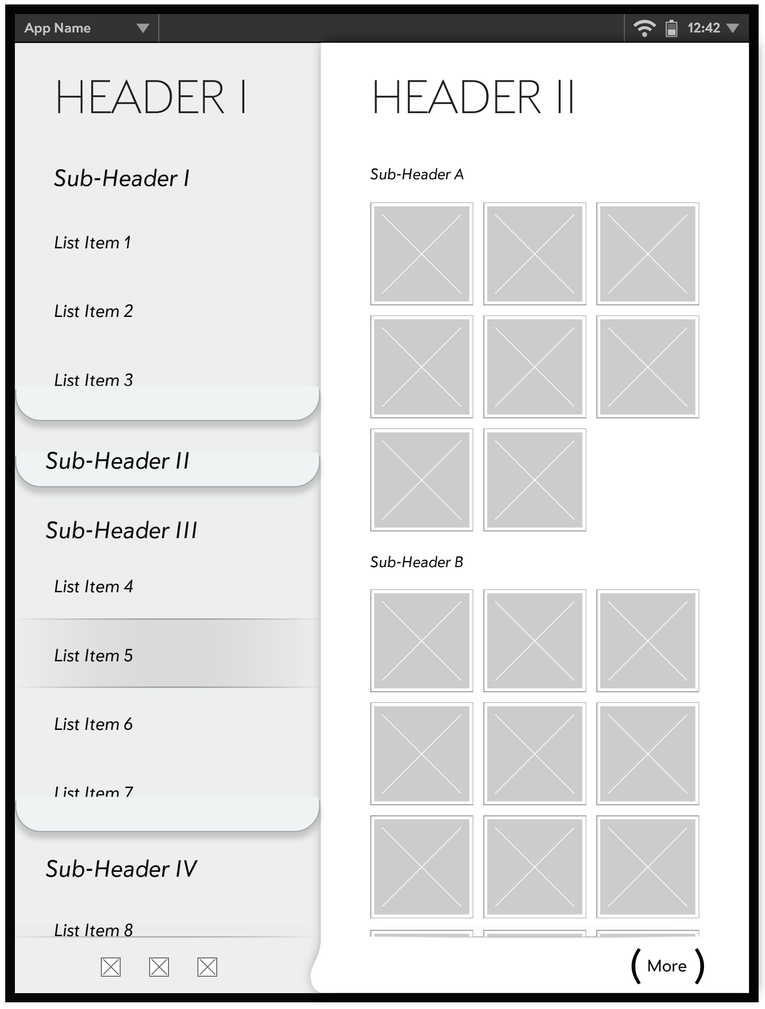
Can be placed at the bottom of the panel to hold buttons. Indicated by spacing and Scroll Fade, but isn't otherwise visible (unlike Chrome which is visible).

Basement

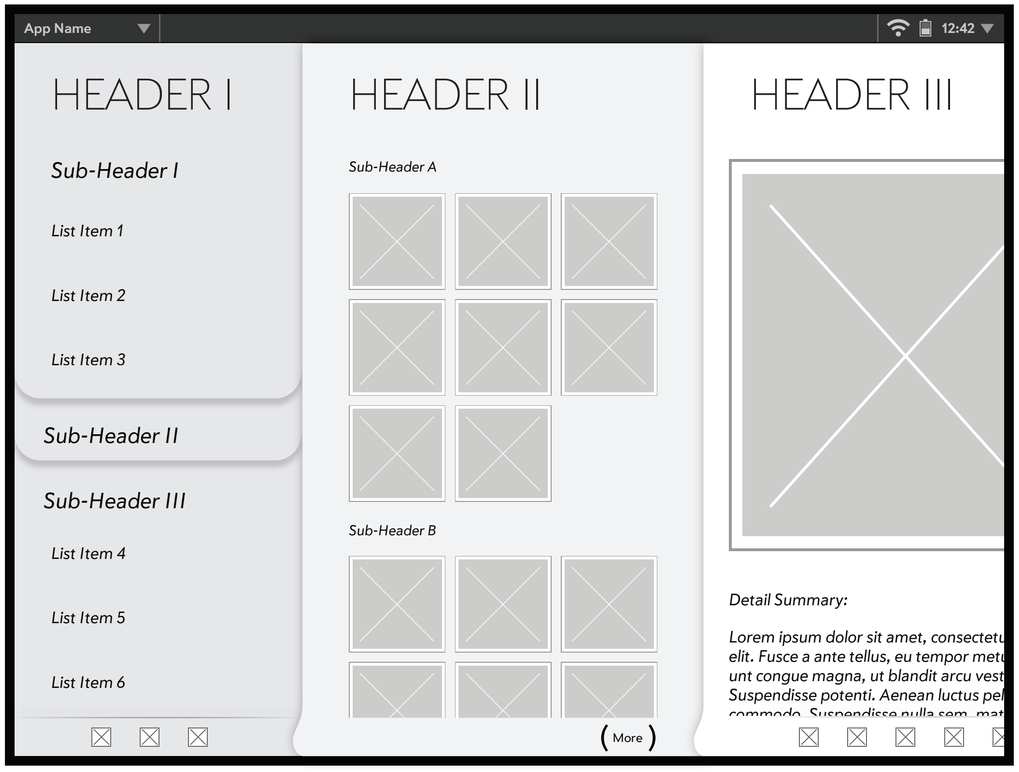
Bottom-most container, typically static. The basement should not move.

VARIATIONS: TABLET

TABLET PORTRAIT



TABLET LANDSCAPE

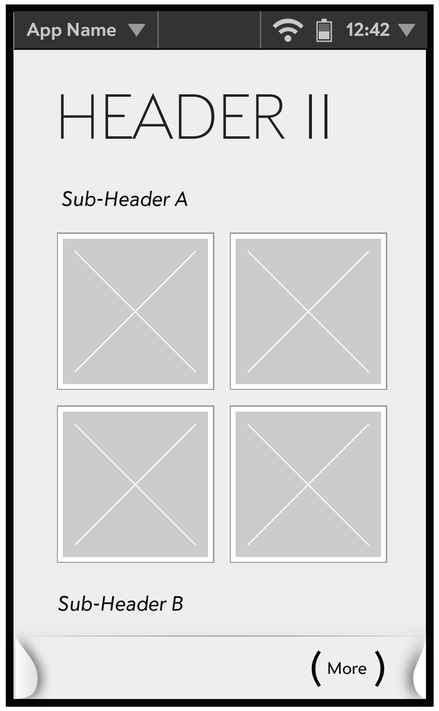


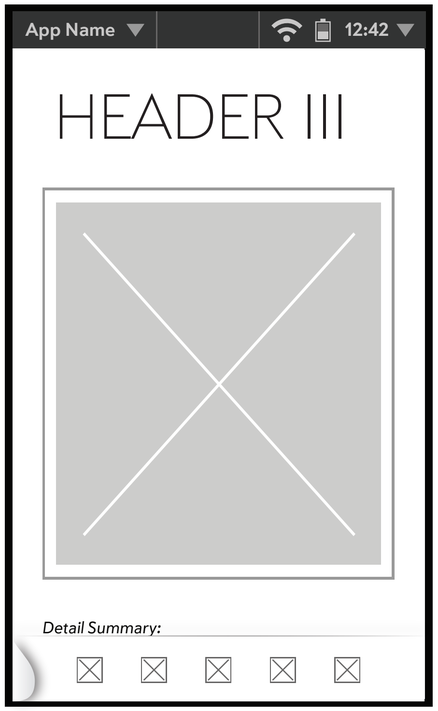
General Behavior on Tablet:

- Multiple panels can be shown at once
- Devs choose number of panels and panel width
- Devs choose whether panels and content resize or if content is hidden
- Devs choose which panels are given priority when changing screen orientation
- Devs choose auto-collapse behavior of panels

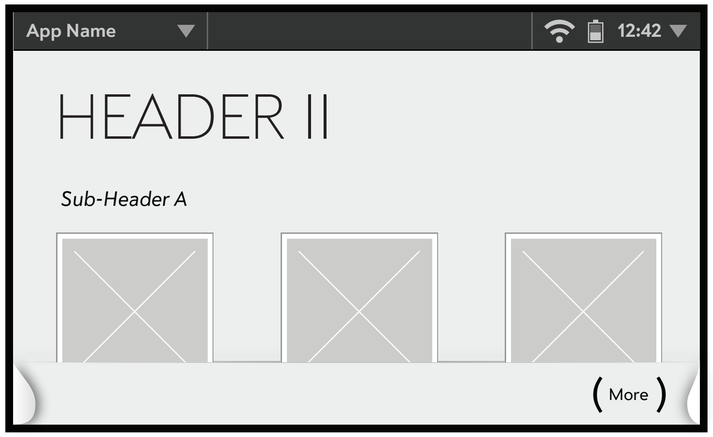
VARIATIONS: PHONE

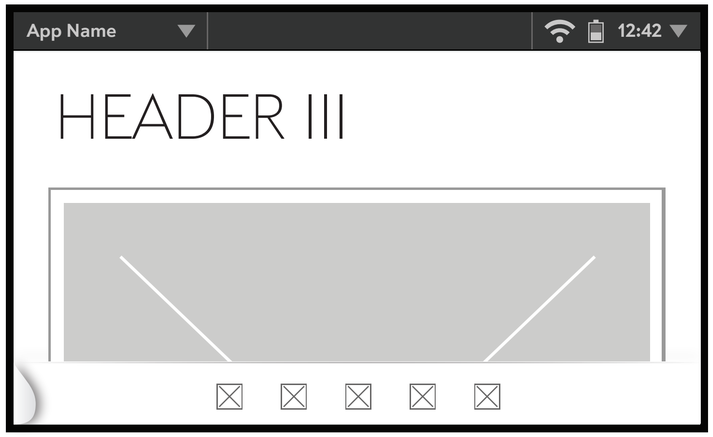
PHONE PORTRAIT





PHONE LANDSCAPE



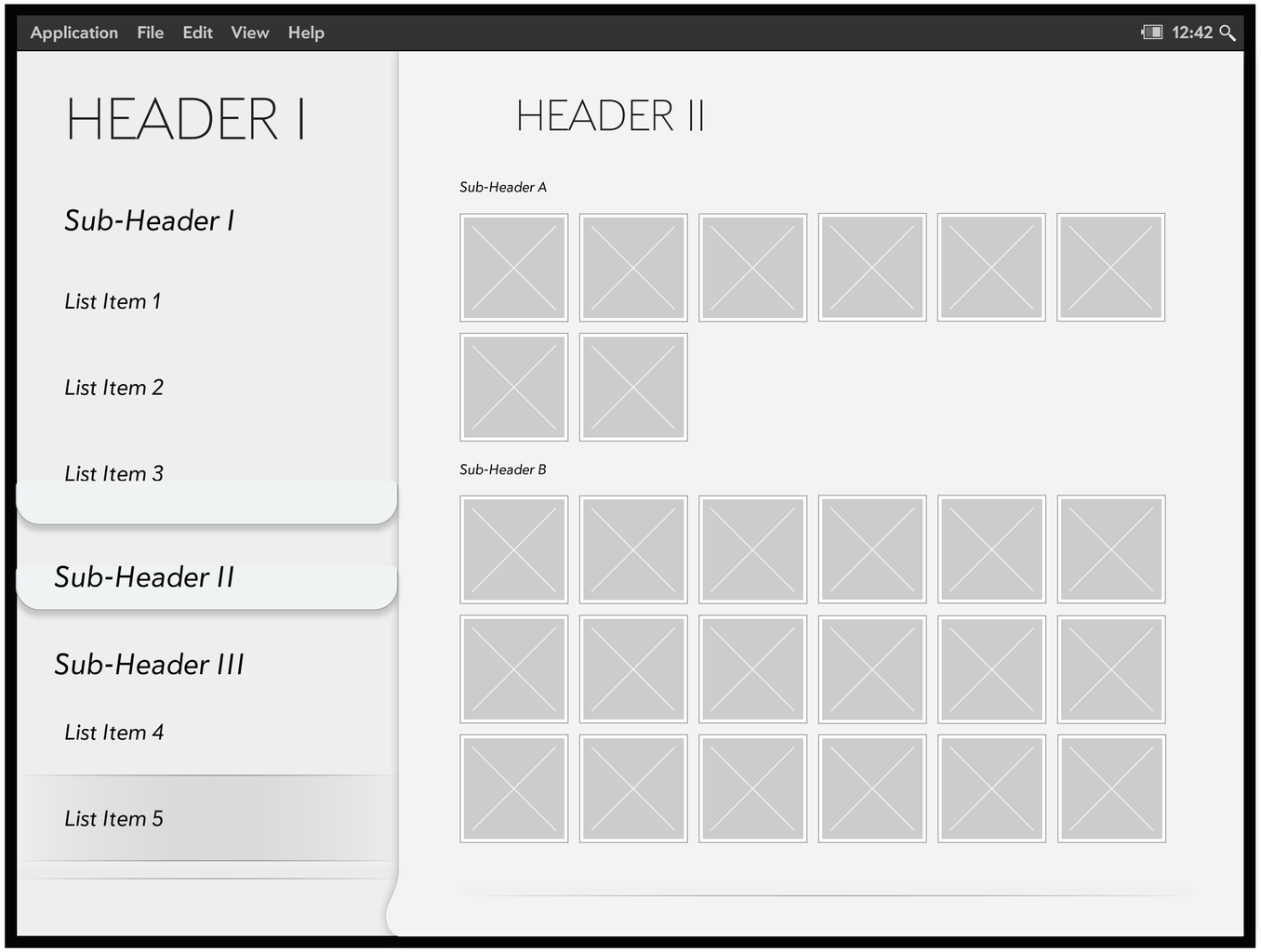


General Behavior on Phone:

- Only one panels is shown at a time (exception: peeking panels)
- Panel width is fixed (exception: peeking)

VARIATIONS: PC

PC

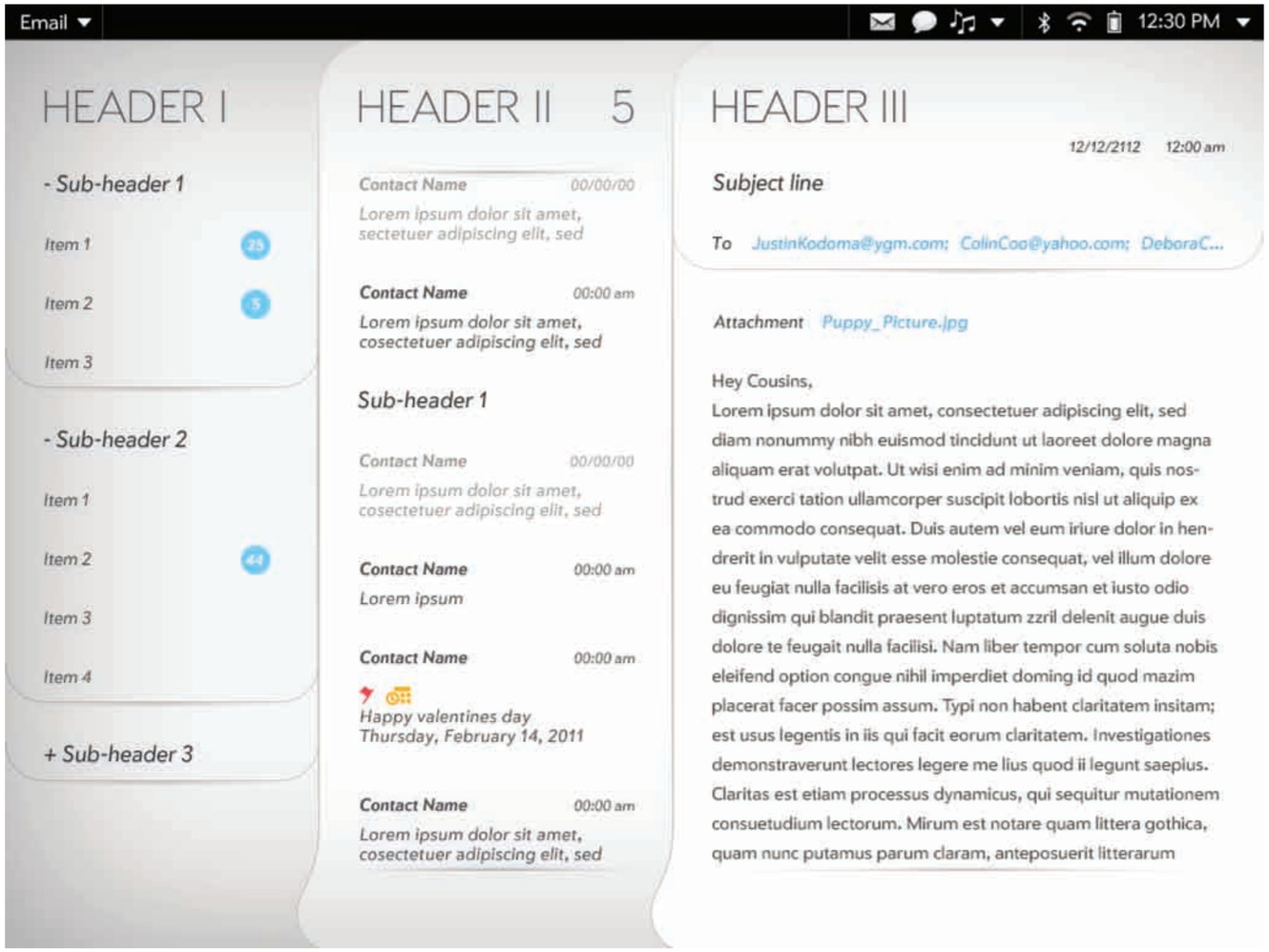


General Behavior on Tablet:

- Multiple panels can be shown at once
- Devs to choose number of panels and panel width
- Devs choose whether panels and content resize or if content is hidden
- Devs to choose which panels are given priority when changing window size
- Devs to choose auto-collapse behavior of panels

VISUAL DESIGN EXAMPLES

MULTIPLE PANELS

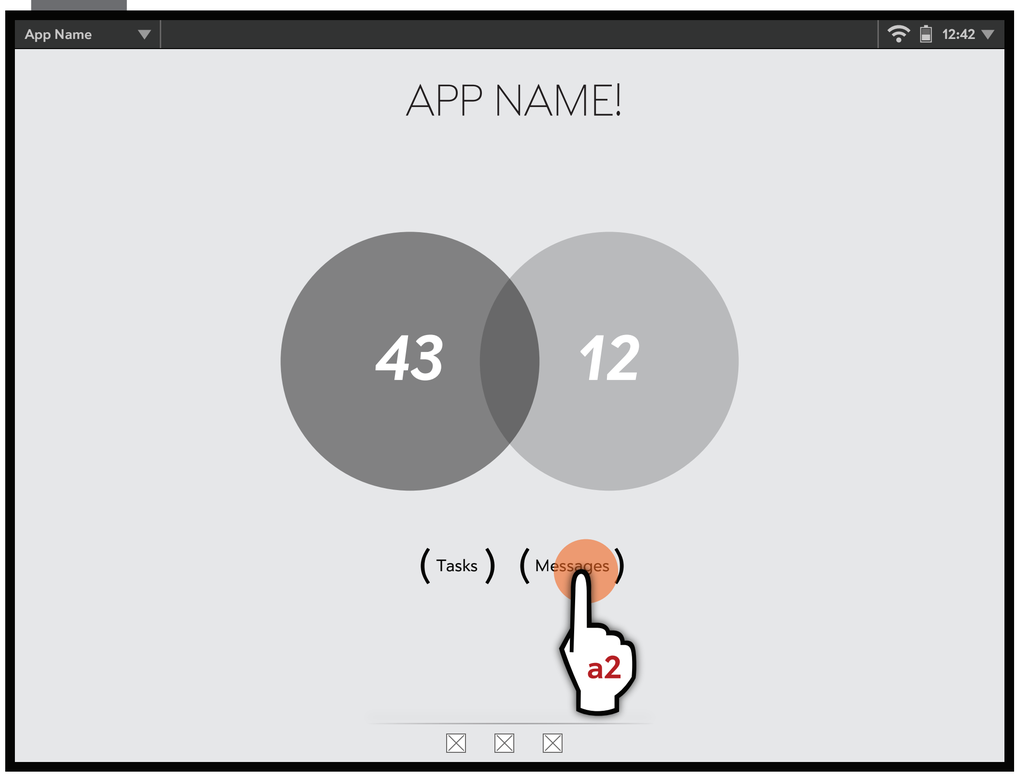


SINGLE PANEL

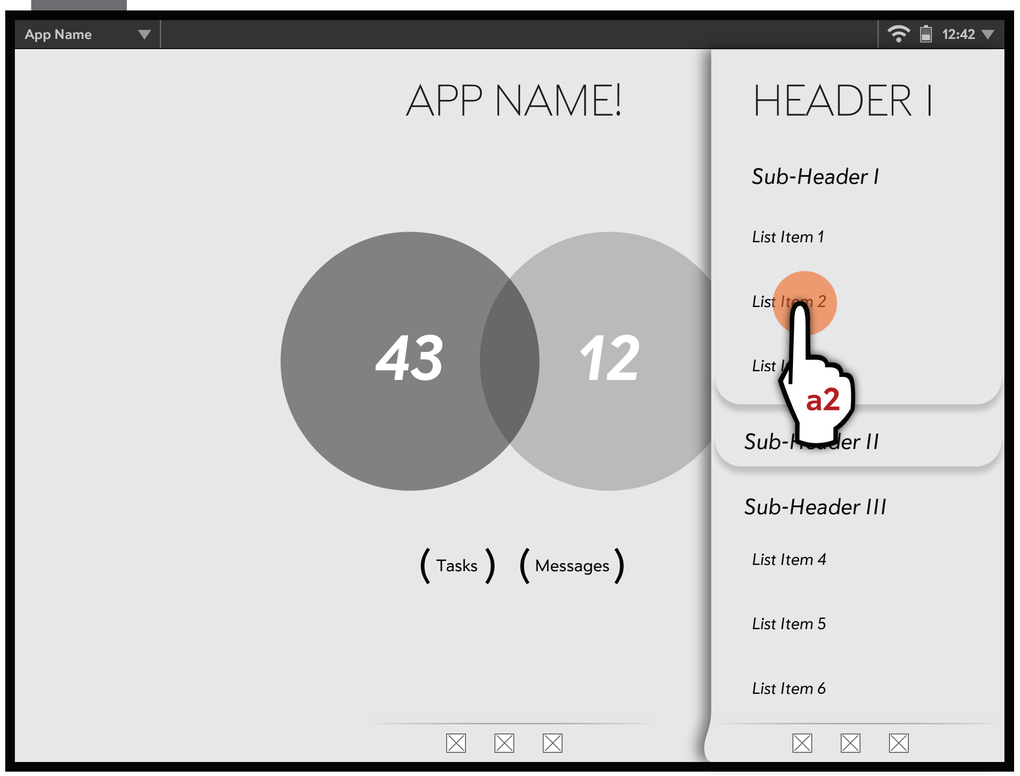
Please refer to visual design specs for animations or more details.

EXAMPLE TASK FLOW: FIRST LAUNCH

a1



a2



a1. App is launched. Fullscreen dashboard is displayed.

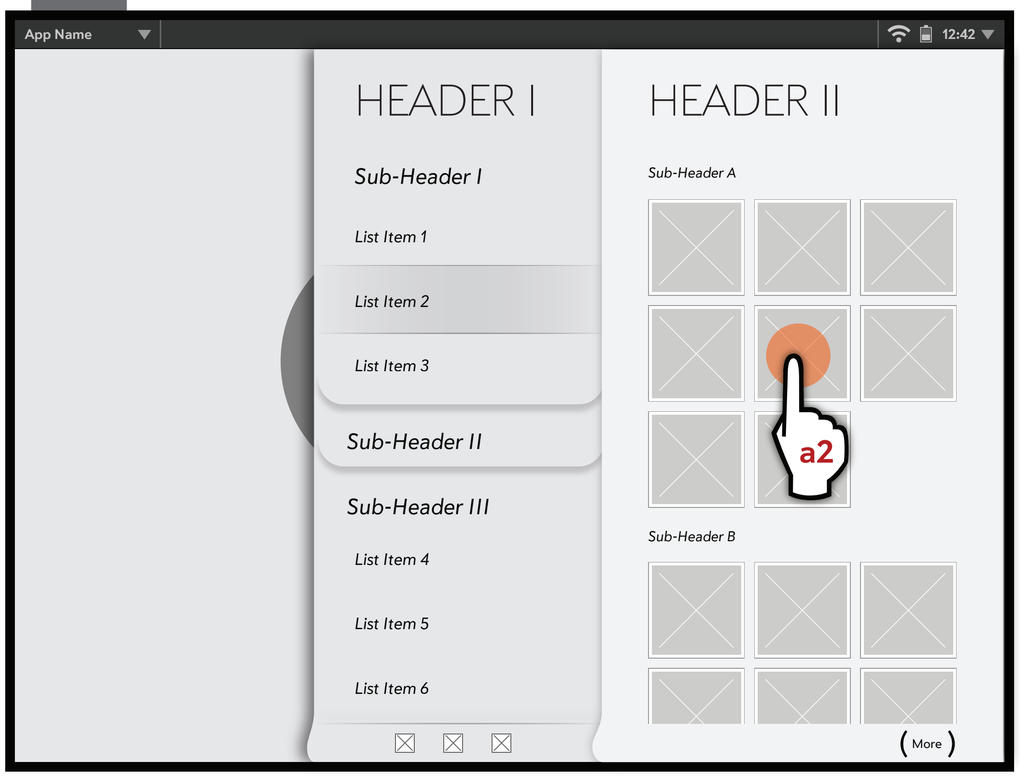
User taps a button that invokes a panel.

a2. Default behavior: the first panel slides in from the right and pins to the right. Developers may choose to left-align panels instead.

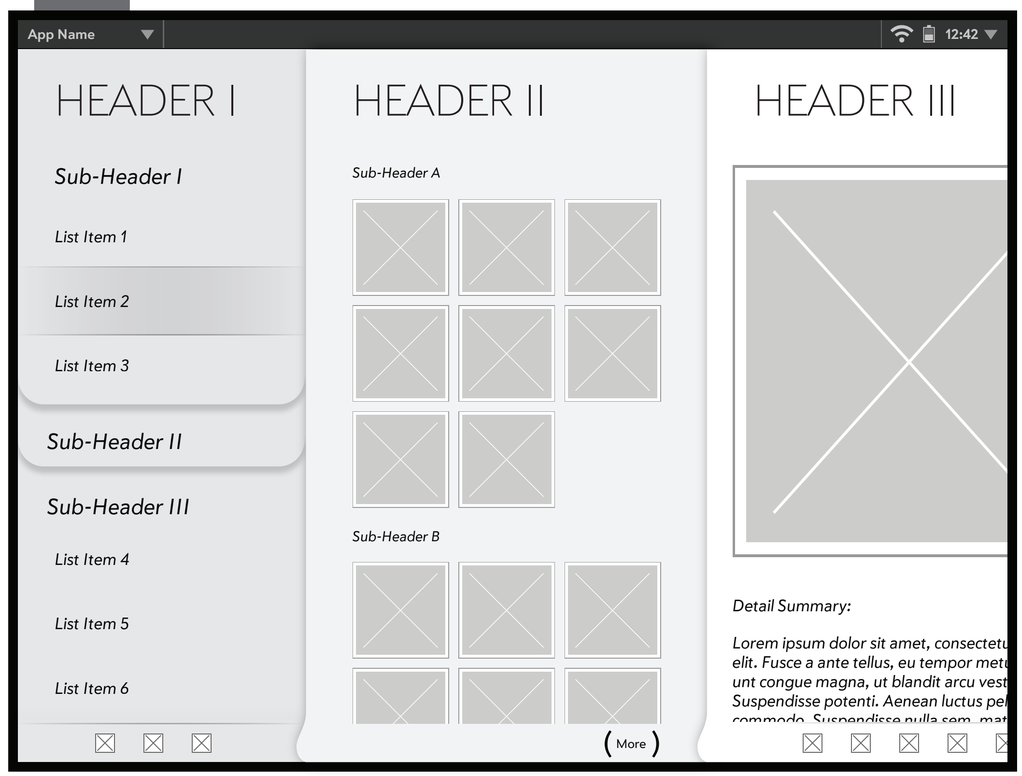
User invokes another panel.

EXAMPLE TASK FLOW: FIRST LAUNCH (CONT.)

a3



a4



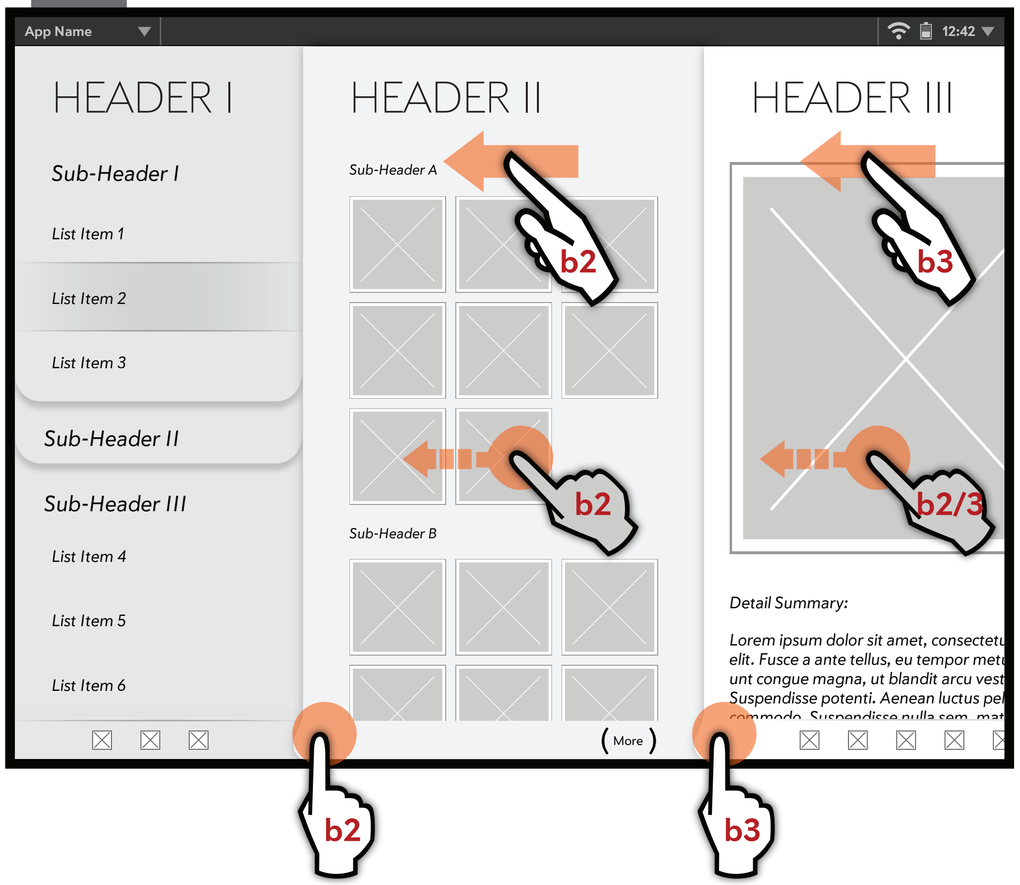
a3. A second panel slides in from the right and pushes the first panel with it. The right edge of the second panel should now be flush with the right side of the screen.

User invokes a third panel.

a4. The third panel slides in fom the right and pushes the other two panels left until the first panel is now flush with the left sign of the screen. Stacking panels should not fall off the left side of the screen; panels will lodge at the left side and consecutive panels will stack on top (see following pages).

EXAMPLE FLOW: DEFAULT BEHAVIOR

b1

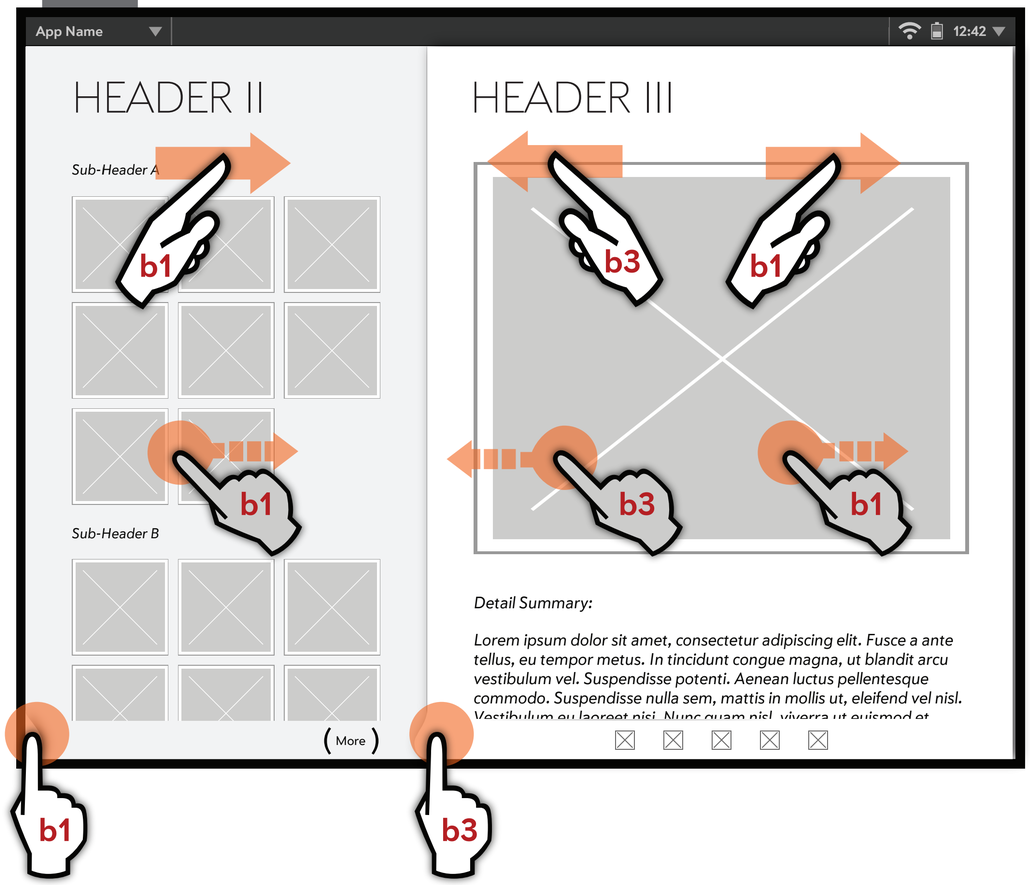


b1. This flow is the default behavior of an app with a traditional hierarchy.

Second panel: Swiping or dragging leftwards anywhere without a conflict will move the panel on top of the leftmost panel. Tapping on the tab of the panel will toggle views from **b1** to **b2.**

Third panel: Swiping left will go to **b2.** Dragging can go to **b2** or **b3** depending on where the user lifts his finger. Tapping the nub will toggle **b2** and **b3.**

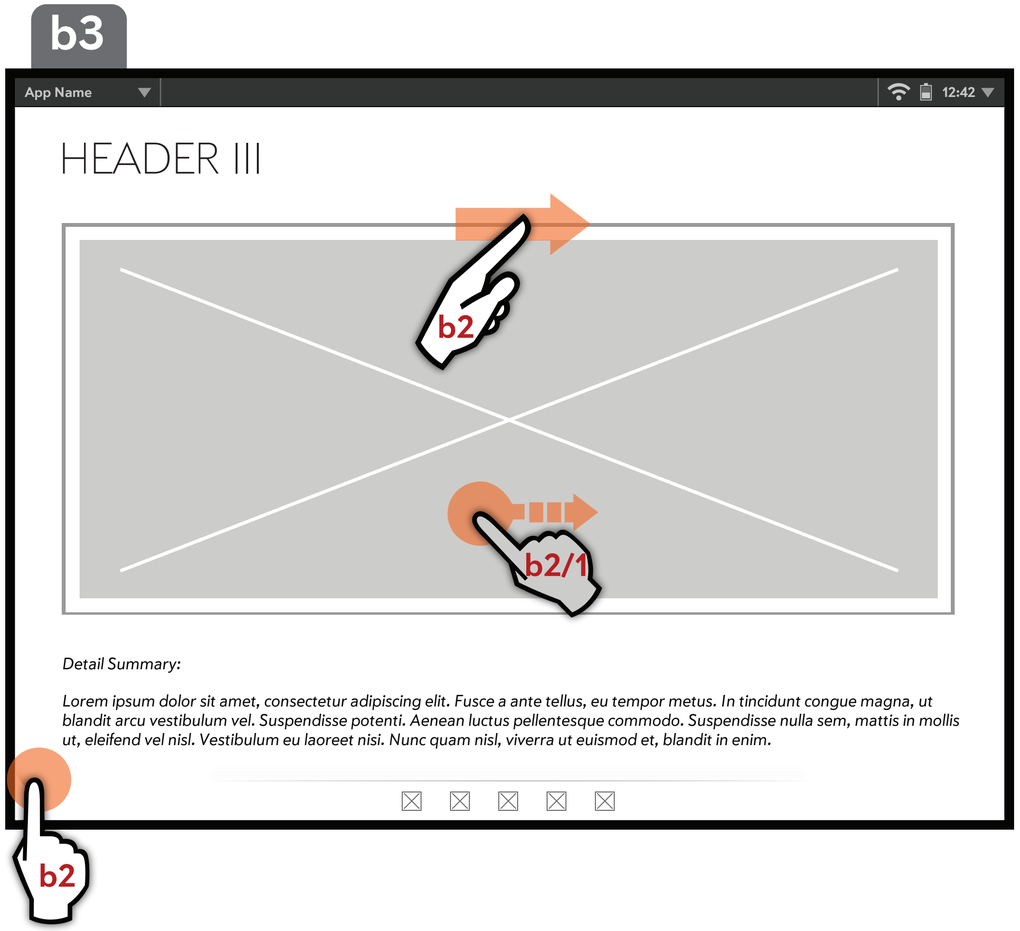
b2



b2. Second panel: Swiping or dragging right anywhere without a conflict will go back to **b1.** Tapping on the nub will toggle **b1** and **b2.**

Third panel: Swiping or dragging left will go to **b2.** Swiping right or dragging right will go back to **b1.** Dragging right can go to Tapping the nub will toggle **b2** and **b3.**

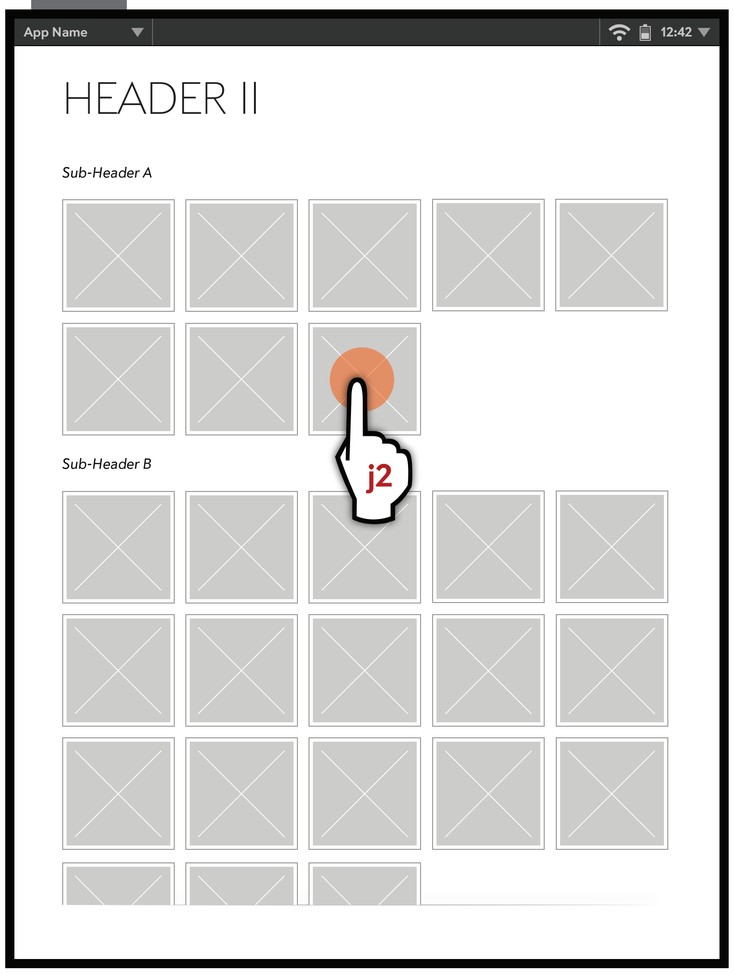
EXAMPLE TASK FLOW: DEFAULT BEHAVIOR (CONT.)



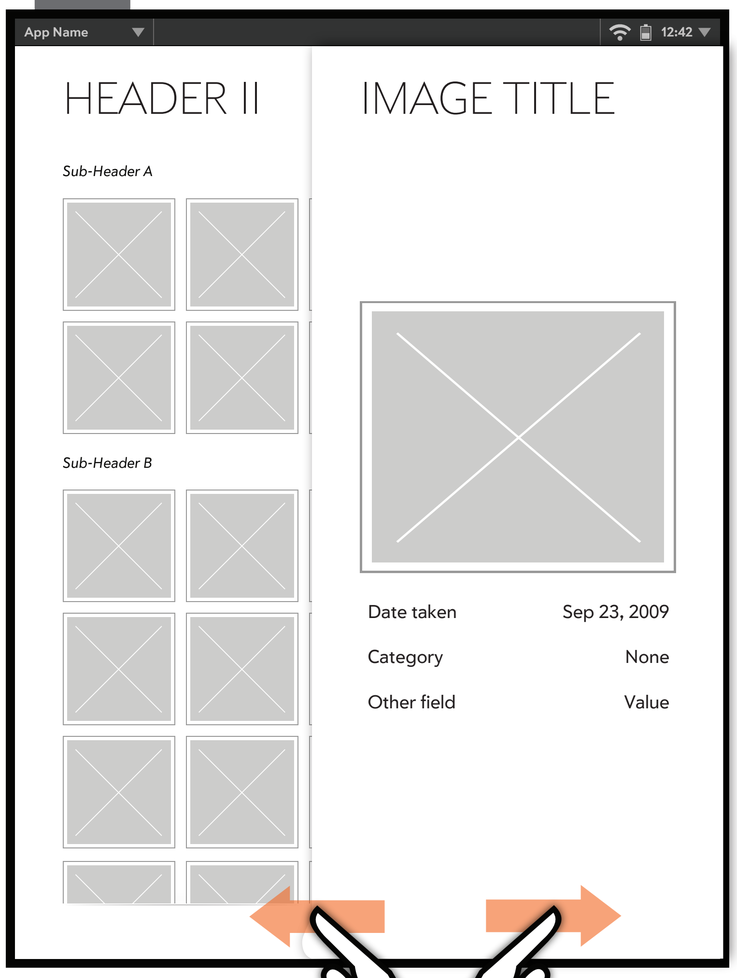
b3. Third panel**:** Swiping right will go to **b2.** Dragging can go to **b2** or **b1** depending on where the user lifts his finger**.** Tapping the nub will toggle **b2** and **b3.**

EXAMPLE TASK FLOW: NO REFLOW (DEFAULT)

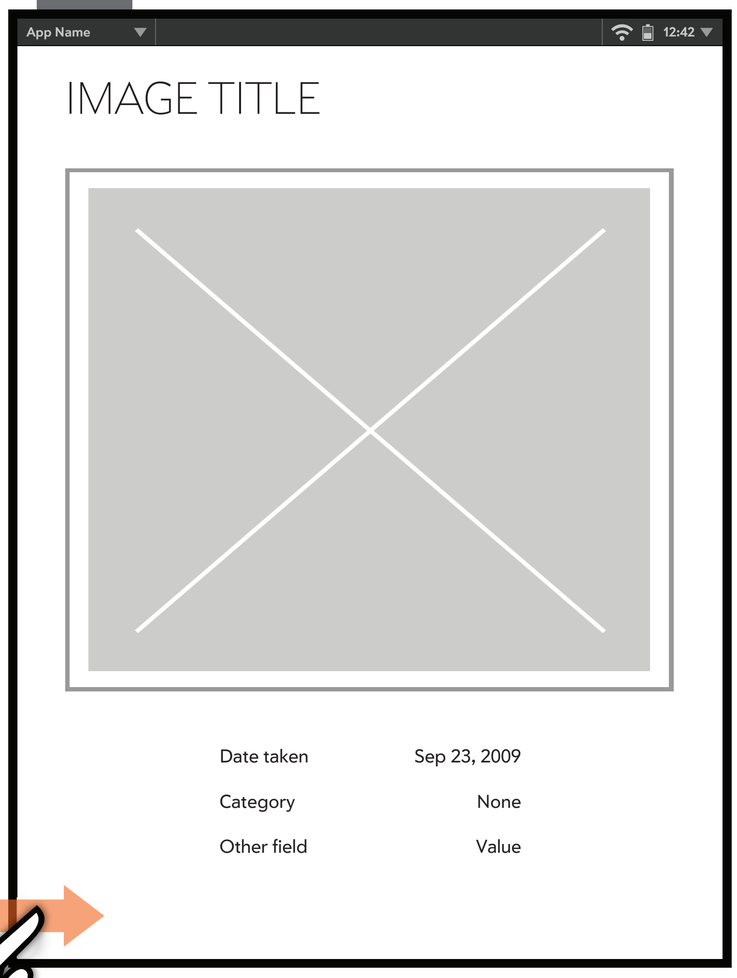
c1



c2



c3



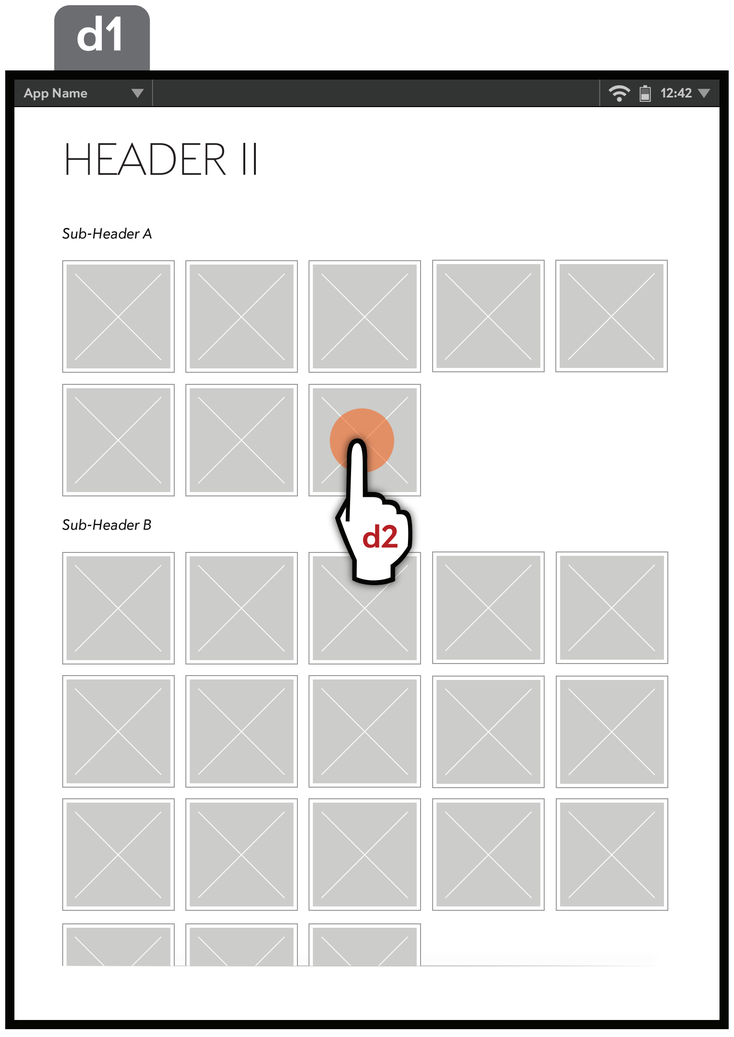
c1. User selects an object in a grid.

c2. New panel contain details slides in and stacks on top. Content is covered and does not reflow. User can swipe right to return to the previous view (**c1**) or left to expand the panel (**c3**).

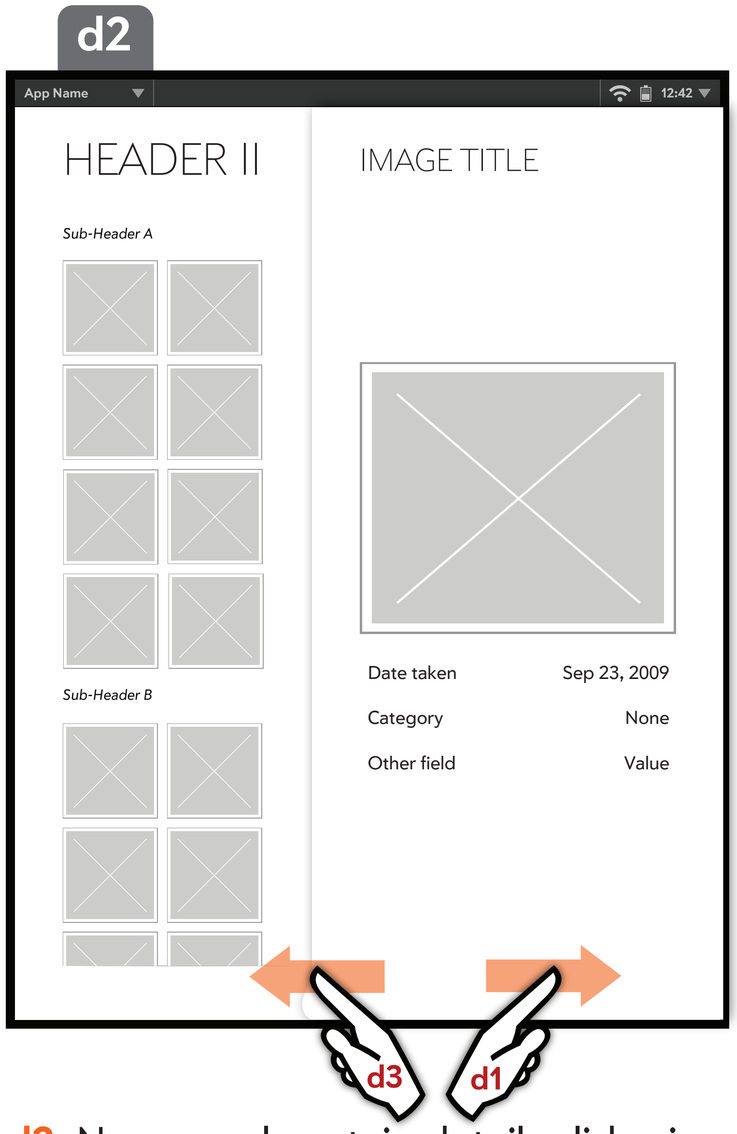
See **b1-3** *for more details on panel manipulation.*

c3. User can swipe right to return to the previous view (**c2**).

EXAMPLE TASK FLOW: WITH REFLOW*

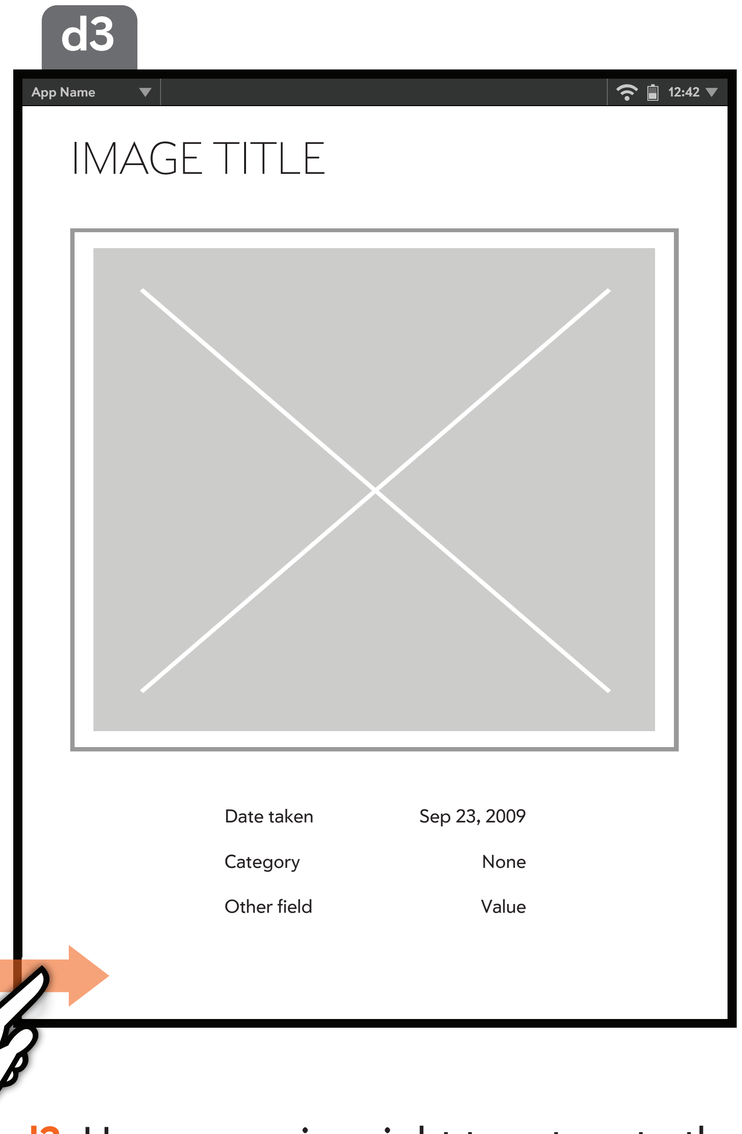


d1. User selects an object in a grid.



d2. New panel contain details slides in and resizing the first panel. Content reflows. User can swipe right to return to the previous view (**d1**) or left to expand the panel (**d3**).

We'll need to decide on an animation for the transition.

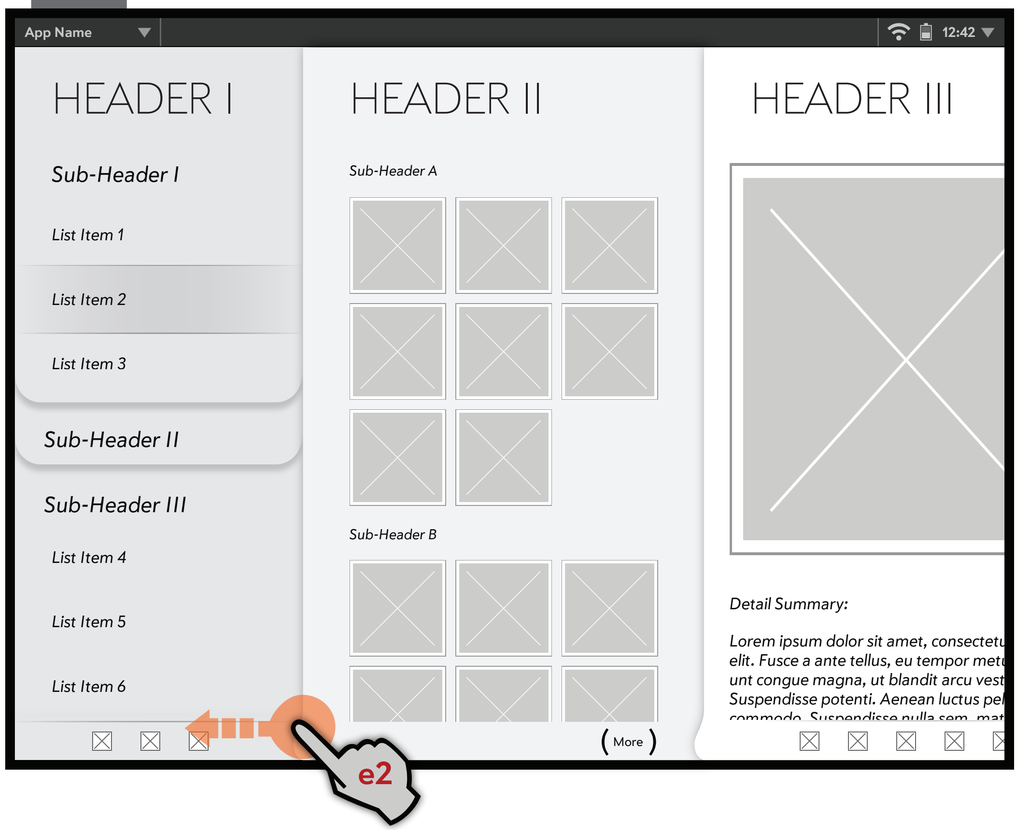


d3. User can swipe right to return to the previous view (**d2**).

See **d1-3** *for more details on panel manipulation.*

EXAMPLE TASK FLOW: RESIZING PANELS

e1

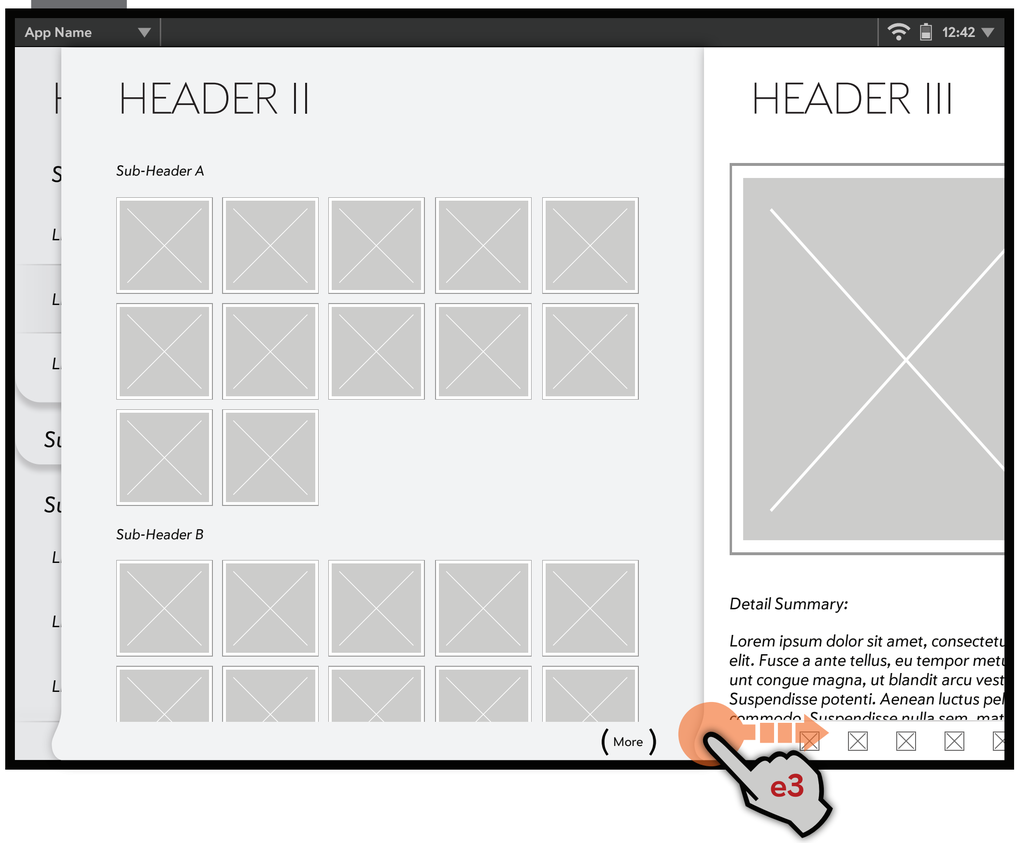


e1. Devs may choose to make panels resizeable. A minimum and maximum size is required. If the min or max size is reached, panels should move as in **b1-3.**

User can drag the nub of panels to resize. To widen the panel leftwards, user must drag the nub of the panel towards the left.

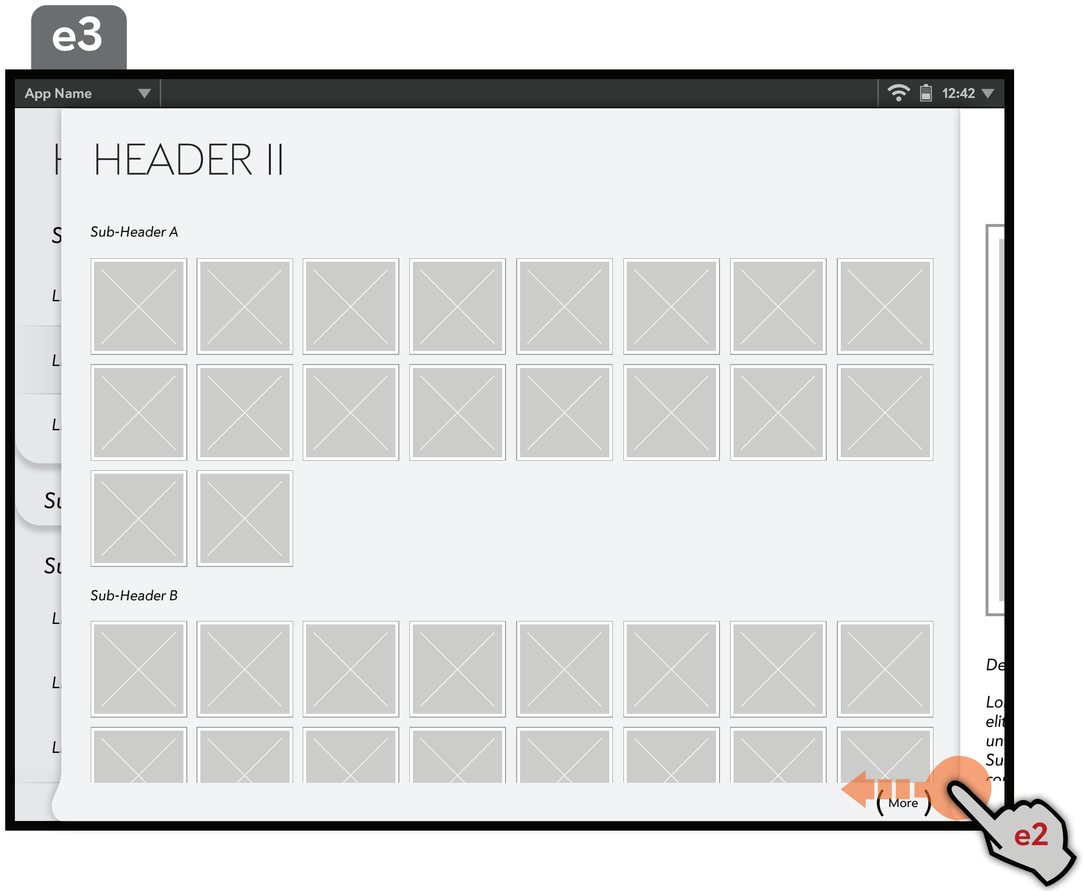
Tapping and dragging will retain behavior as in **b1-3.**

e2



e2. To resize rightwards, the user must drag the nub of the adjacent panel towards the right.

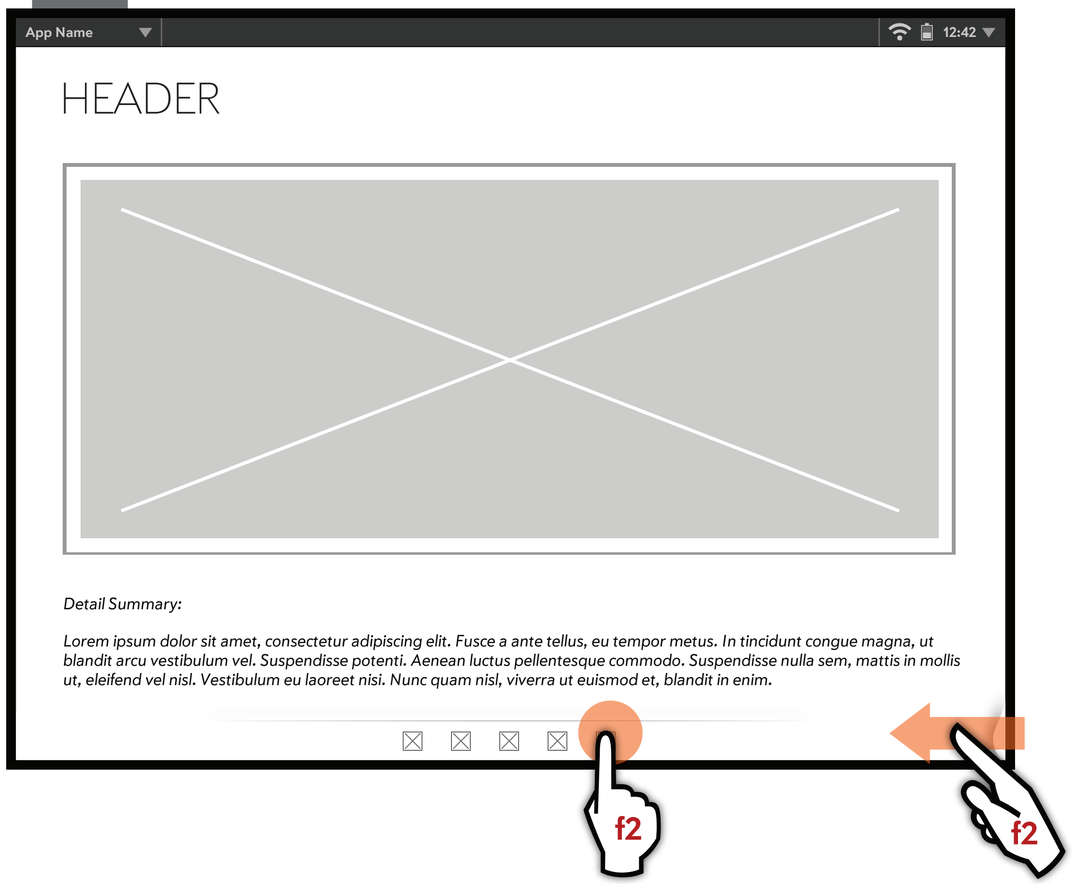
EXAMPLE TASK FLOW: RESIZING PANELS (CONT.)



e3. Users may shrink the panel by dragging the nub of the panel right or the nub of the adjacent panel towards the left.

MODAL PANELS: RIGHT SIDE

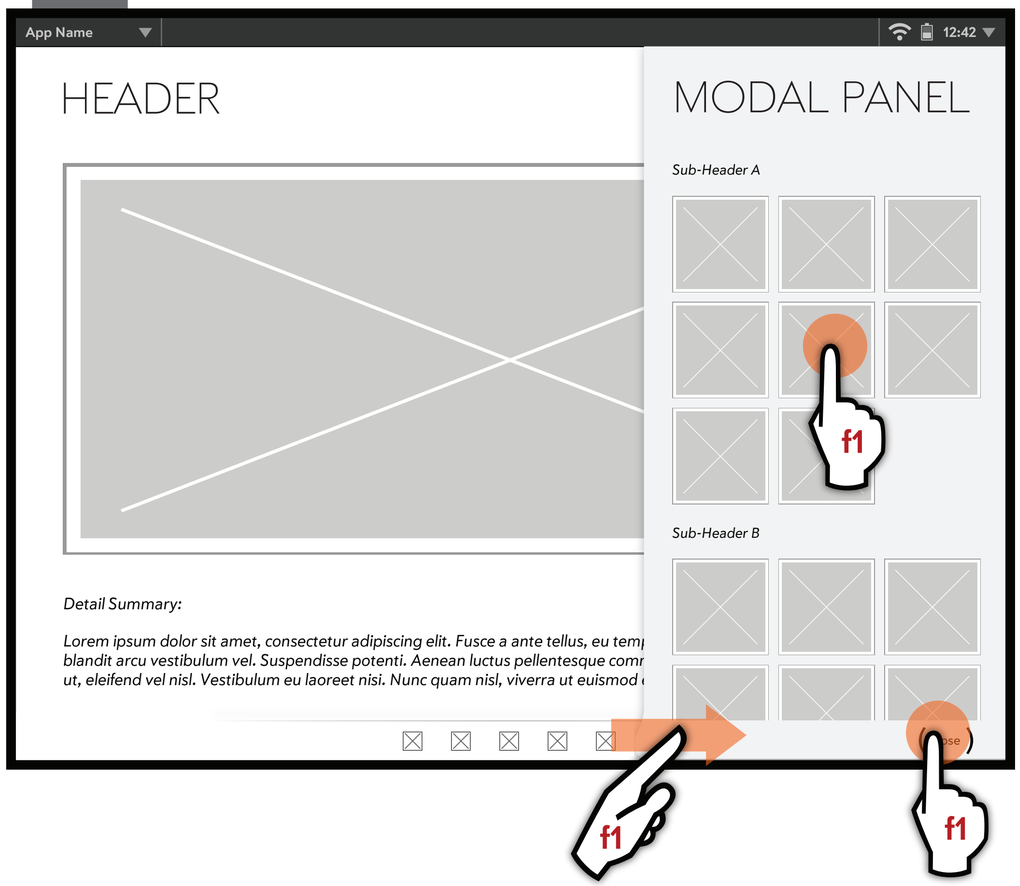
f1



f1. Developers can choose to make panels modal. Panel size is determined by the developer. If modal panels are right-aligned they should appear as a panel from the right and on top of current content.

Examples of how a modal panel can be invoked:
- via button tap
- by swiping the nub of the panel when it is in view (this is recommended only if there are no more hierarchical panels to the right)
- tapping or swiping peeking panels (see **i1-2**)
- swiping left from off screen to onscreen.

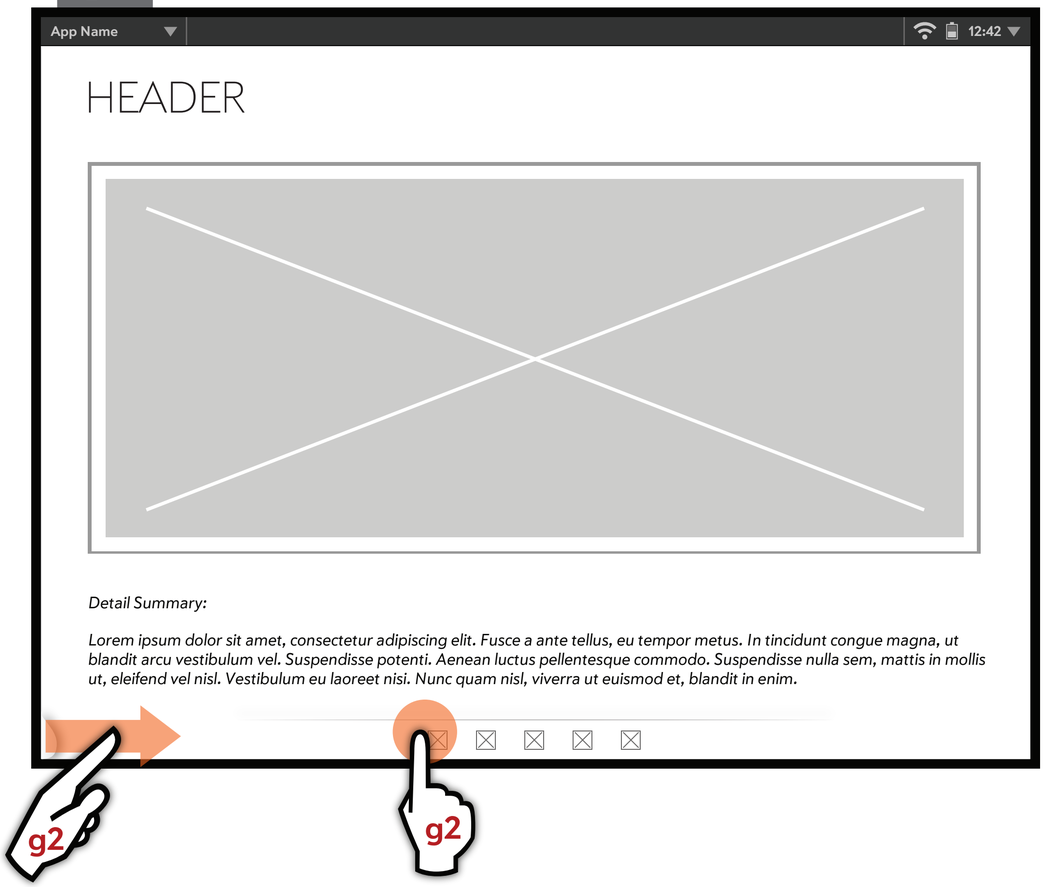
f2



f1. Examples of how a modal panel can be closed:
- by making a selection in the panel
- by swiping the nub of the panel when it is in view
- tapping or swiping peeking panels (see **i1-2**)

MODAL PANELS: LEFT SIDE

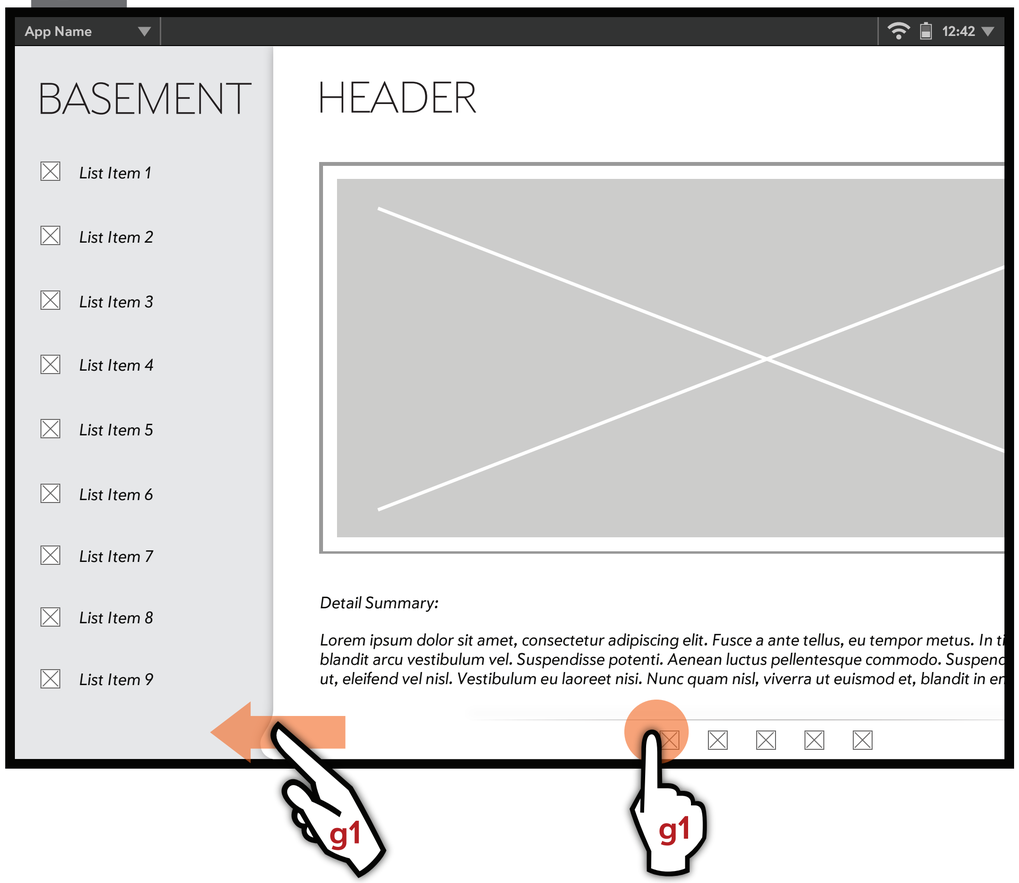
g1



g1. If modal panels are left-aligned and the app is at the top level of the hierarchy, they should appear as a basement beneath the panel from which it was invoked.

Examples of how a modal panel can be invoked:
- via button tap
- by swiping the nub of the panel from which it is invoked.
- tapping or swiping peeking panels (see **i1-2**)
- swiping left from off screen to onscreen.

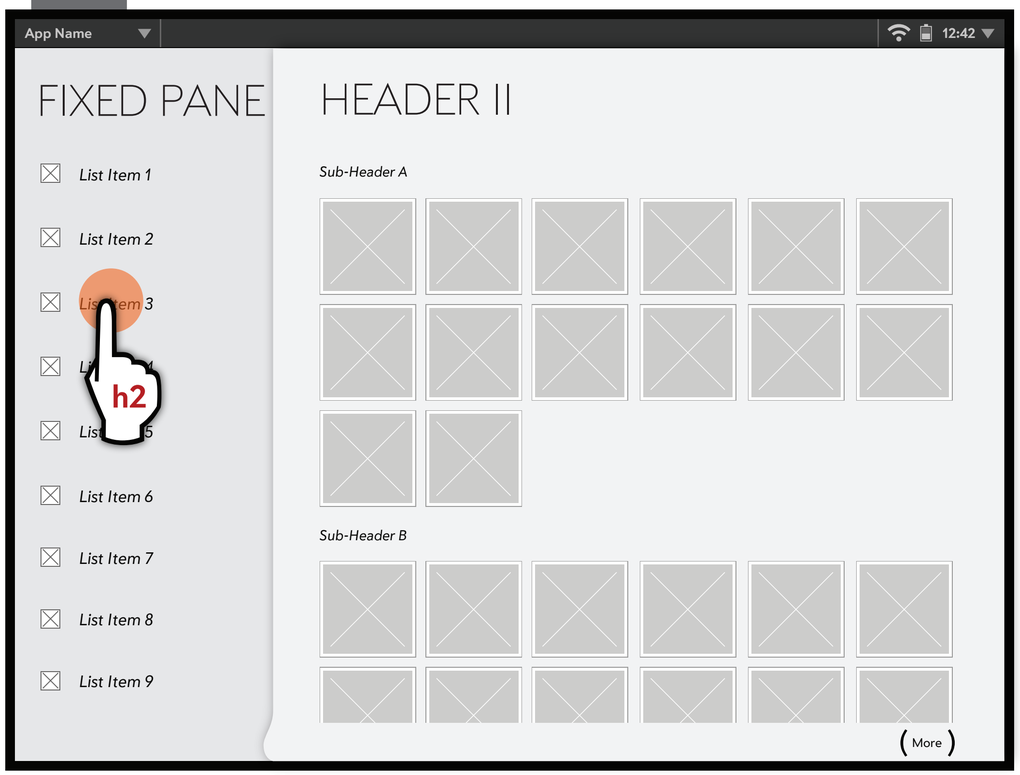
g2



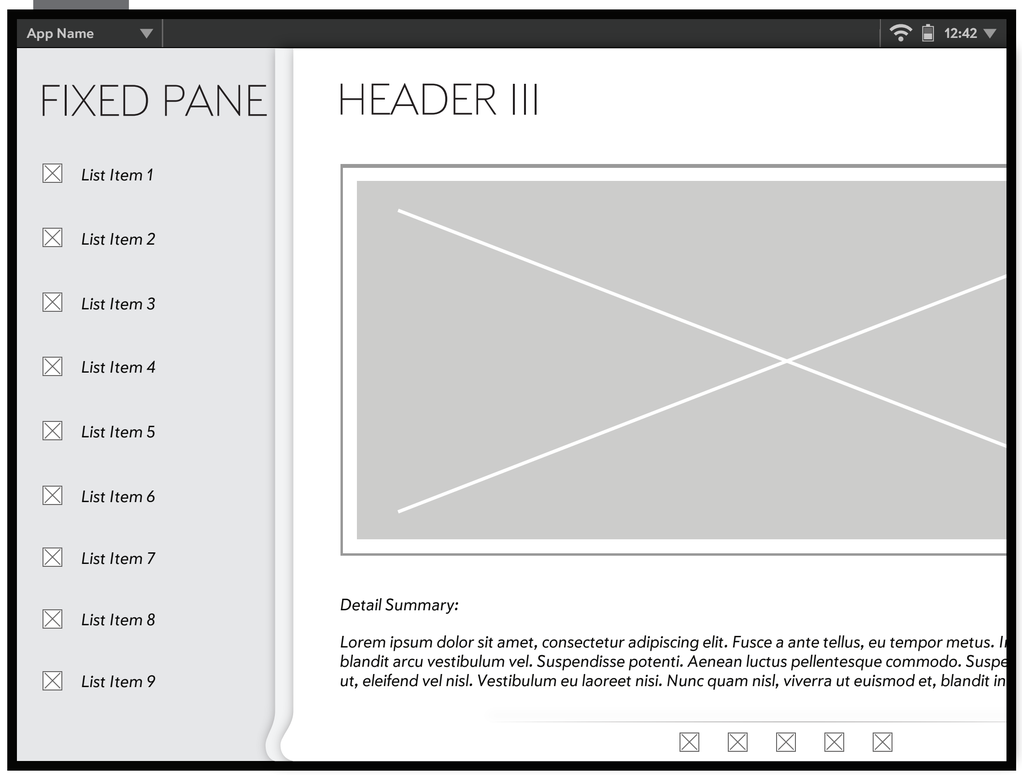
g2. Examples of how a modal panel can be closed:
- by making a selection in the basement
- by swiping the nub of an adjacent panel
- tapping or swiping peeking panels (see **i1-2**)

PANELS WITH A FIXED PANE

h1



h2



h1. It is recommended that any fixed panes remain to the left of panels. Panels stack when they are flush with the right edge of the fixed pane.

h2. When panels stack, they can stack so that only the top panel can be seen (recommended for historical navigation) or they can peek as in the example above (recommended for flat navigation).

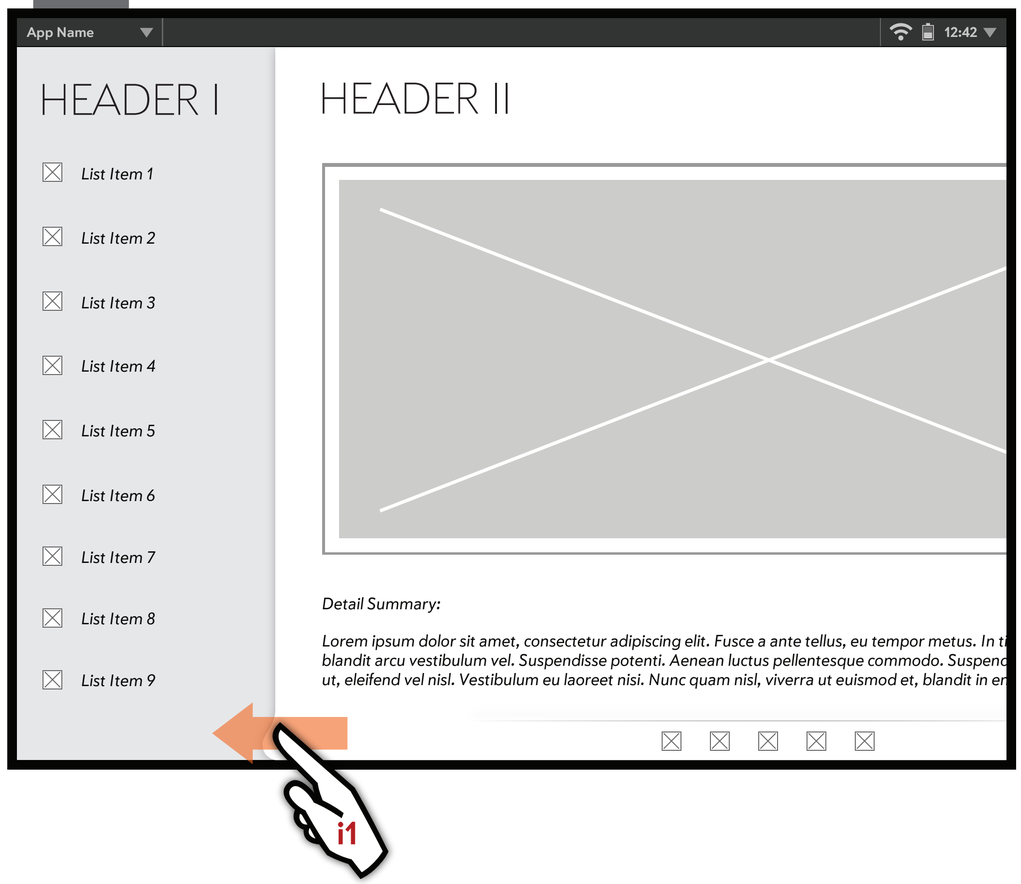
EXAMPLE TASK FLOW: PARTIAL VIEW OF PANEL

i1

i1. Panels can be given more than one locking point. Views can be the following: full width of panel, partial width of panel, panel is covered. This flow is an example of a partial to full view. Content does not reflow. (See **c1-3.**)

The user is still able to interact with the panel when it is only partially shown (e.g., making selections). Swiping/dragging right on the right panel (no-conflict areas) or tapping the nub will expose the entire left panel.

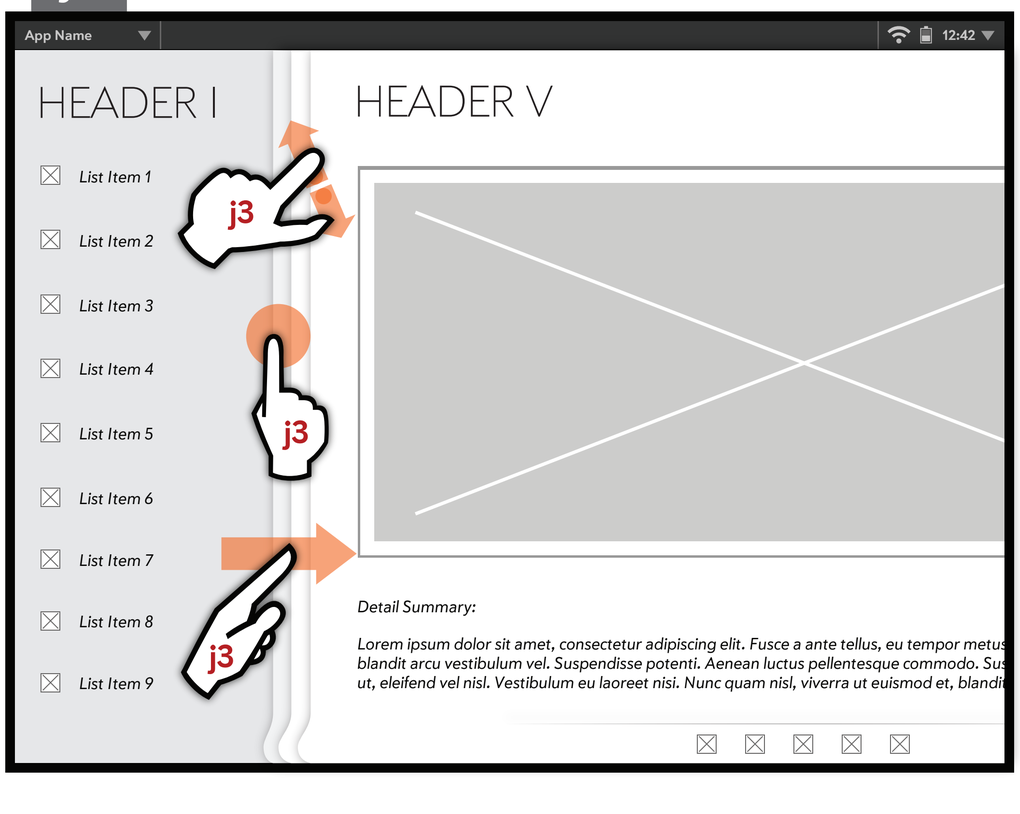
i2



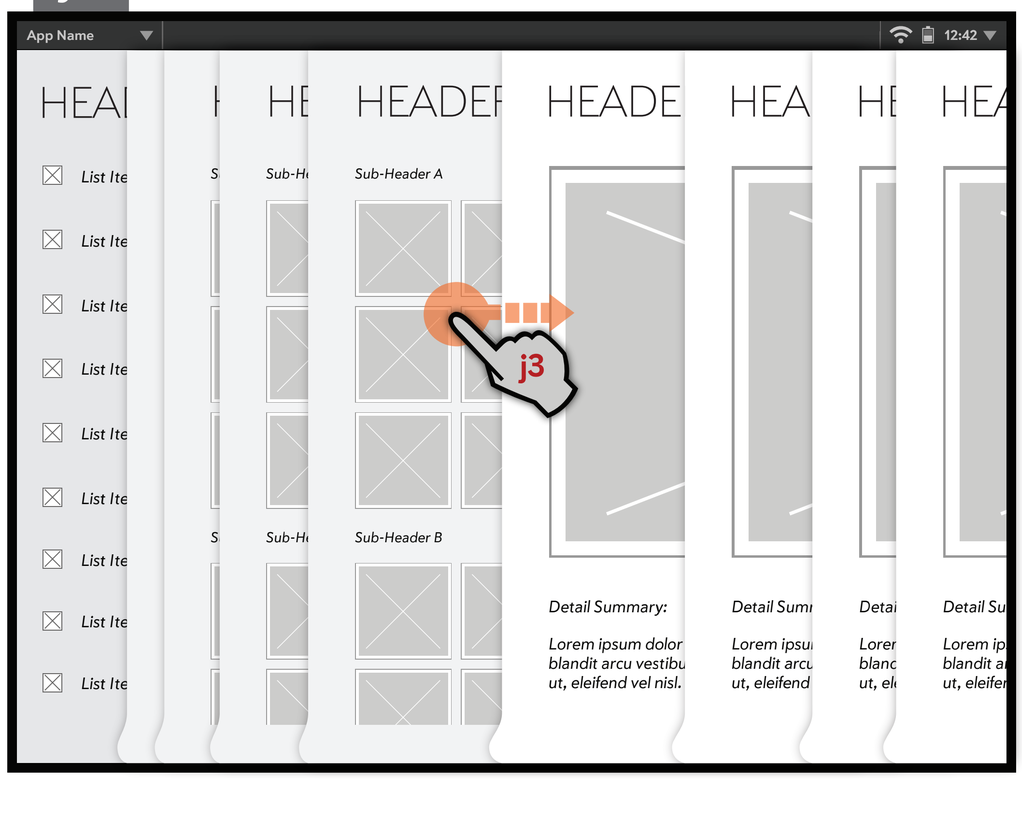
i2. Swiping left on the right panel (no-conflict areas) or tapping the nub will move it over the left panel so that it is partially shown again. At this point if the dev has it enabled, the user may continue dragging the panel or swipe the panel again to cover the left panel entirely.

EXAMPLE TASK FLOW: PEEKING PANELS

j1



j2

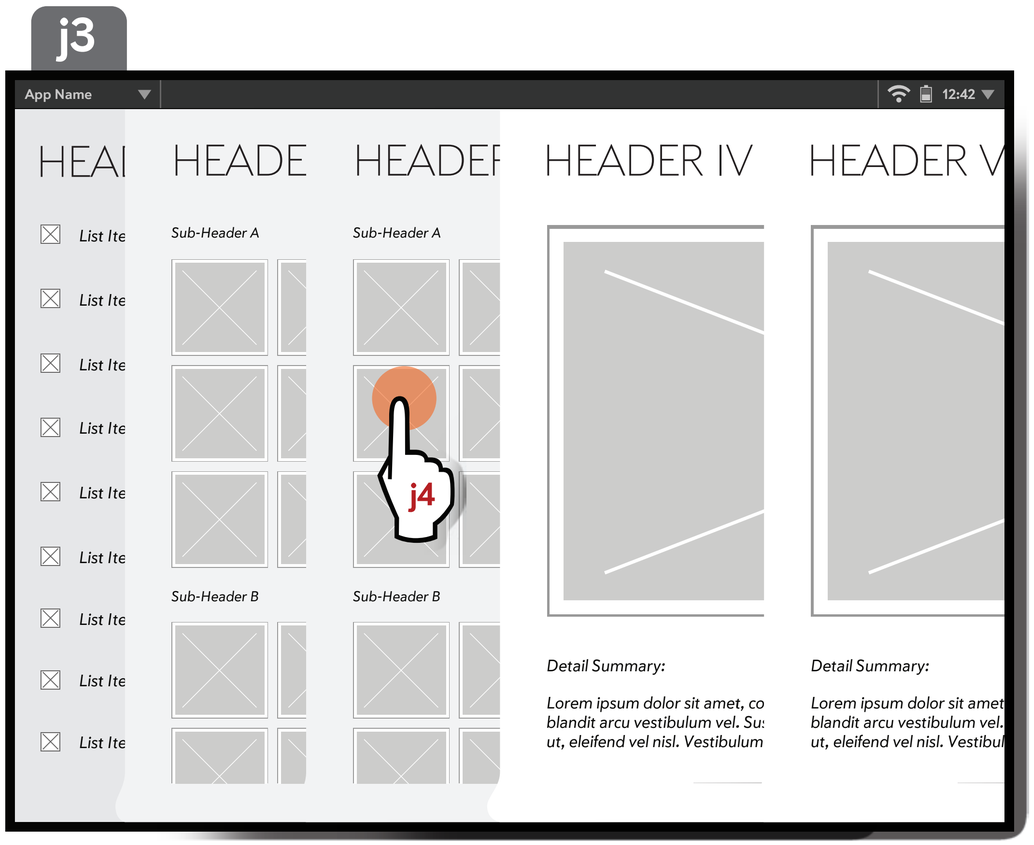


j1. Peeking panels are a modal state. They are a visual representation that informs the user that there are more panels, but may not represent the actual number of panels.

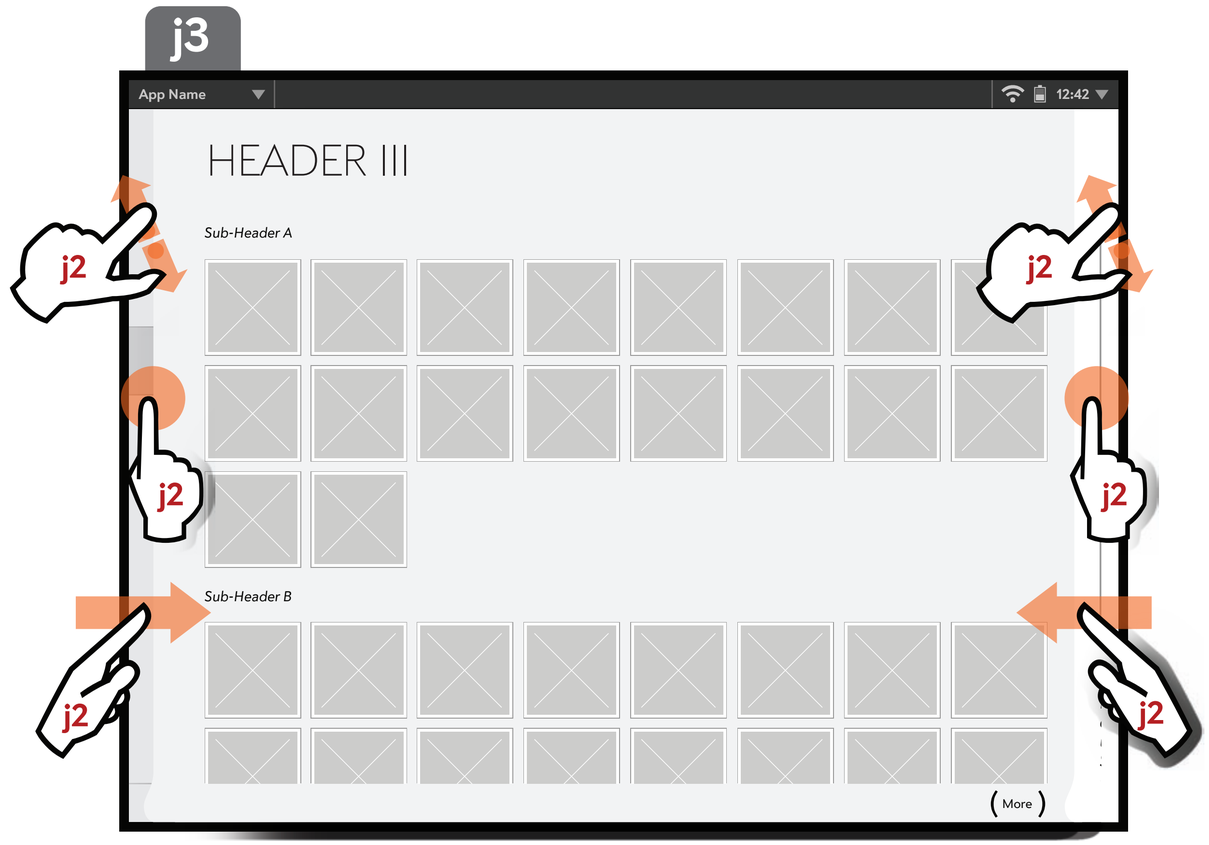
Users can spread, tap, swipe, or drag to spread apart peeking panels.

j2. If possible, animate panels out so that all panels are equally visible on the screen. If number of panels exceed available screen real estate, expose the panels in the center a little more and allow the user to pan through the panels.

EXAMPLE TASK FLOW: PEEKING PANELS (CONT.)

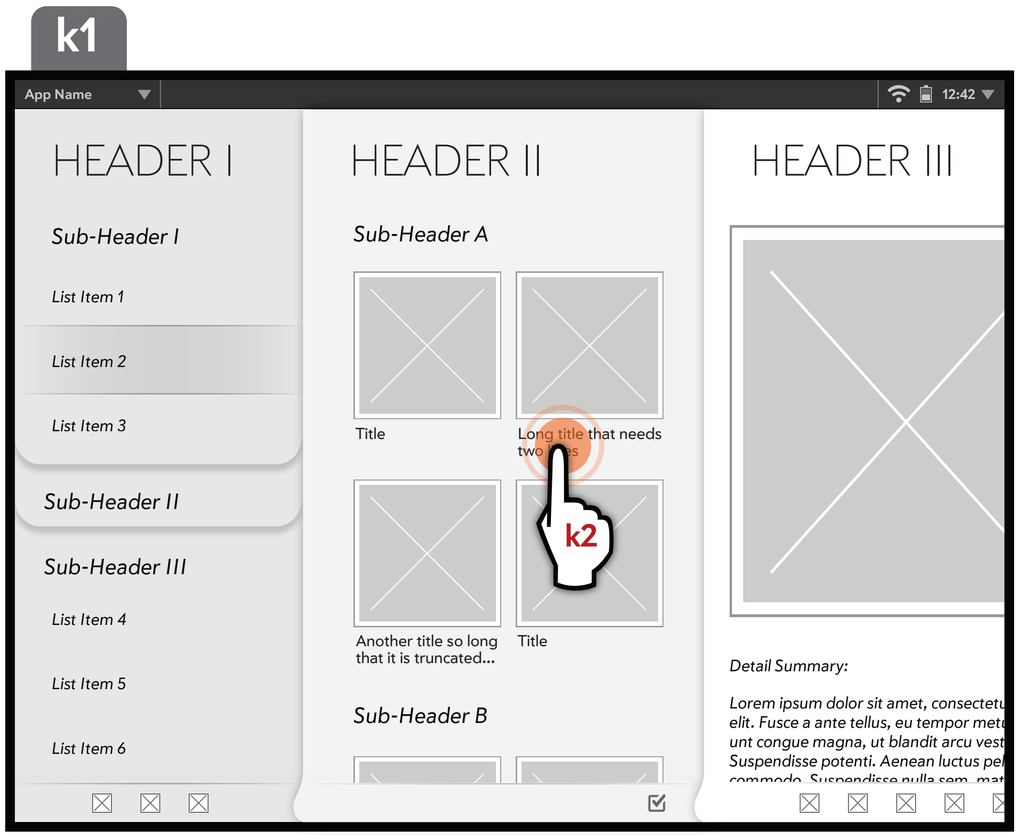


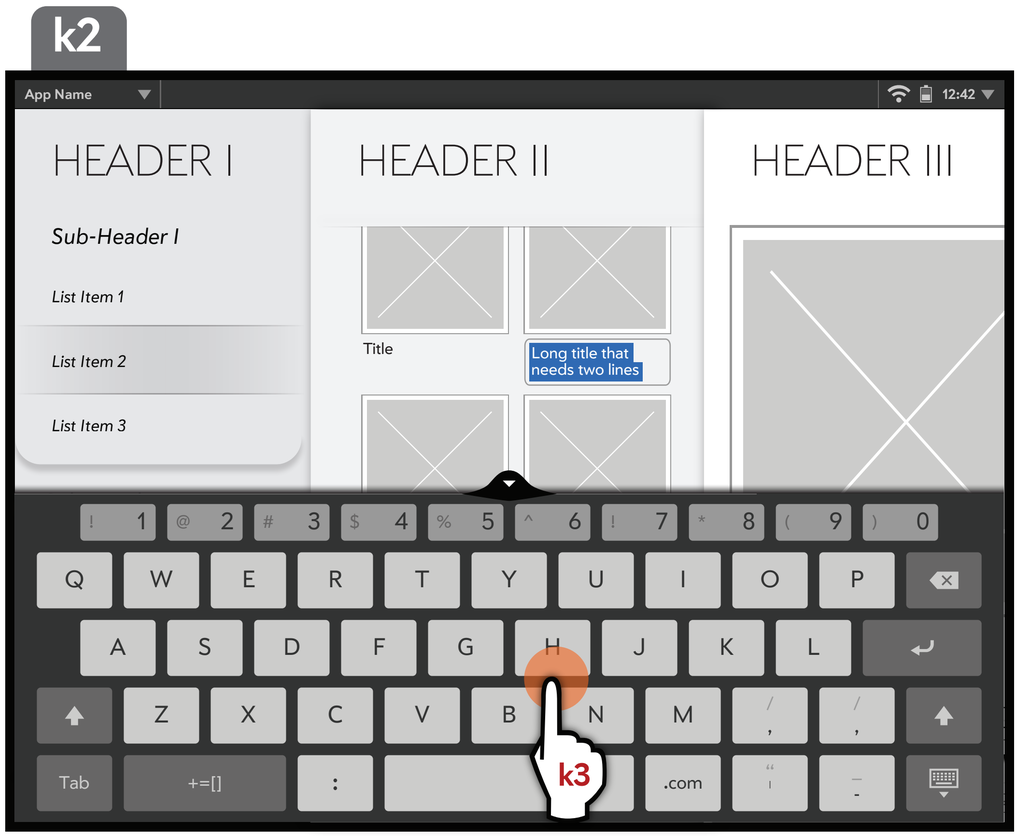
j3. While in this modal state, users should not be able to interact with content on the panels. User must select a panel in order to interact with its content.



j3. When the user selects a panel, expand that panel. If there are panels to the left, peek to the left. If there are panels to the right, peek to the right. Tapping, spreading, swiping, or dragging either side will expand all panels.

EXAMPLE TASK FLOW: PANELS WITH A KEYBOARD





k1. User taps an input field or double taps editable text.

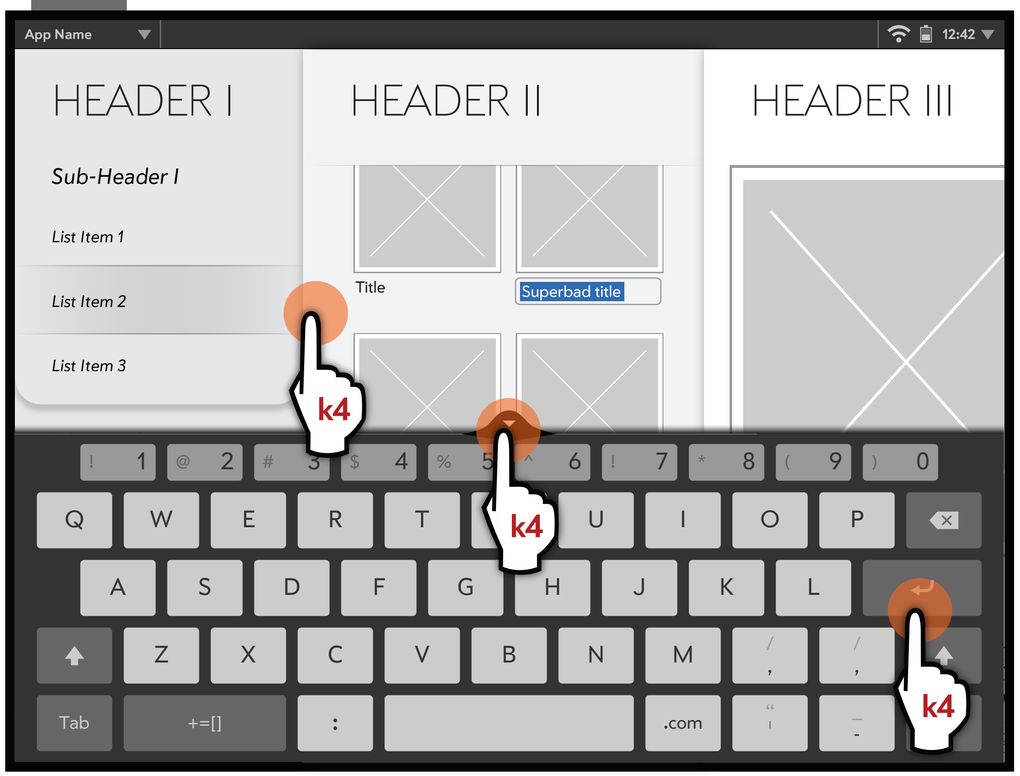
For details on input fields, please refer to documentation on Input. For details on keyboards, please refer to Keyboard-Behavior documentation.

k2. Virtual keyboard is invoked; field is focused and cursor appears. Panels do not make any movement. If necessary, scroll input field into view. Existing text is selected.

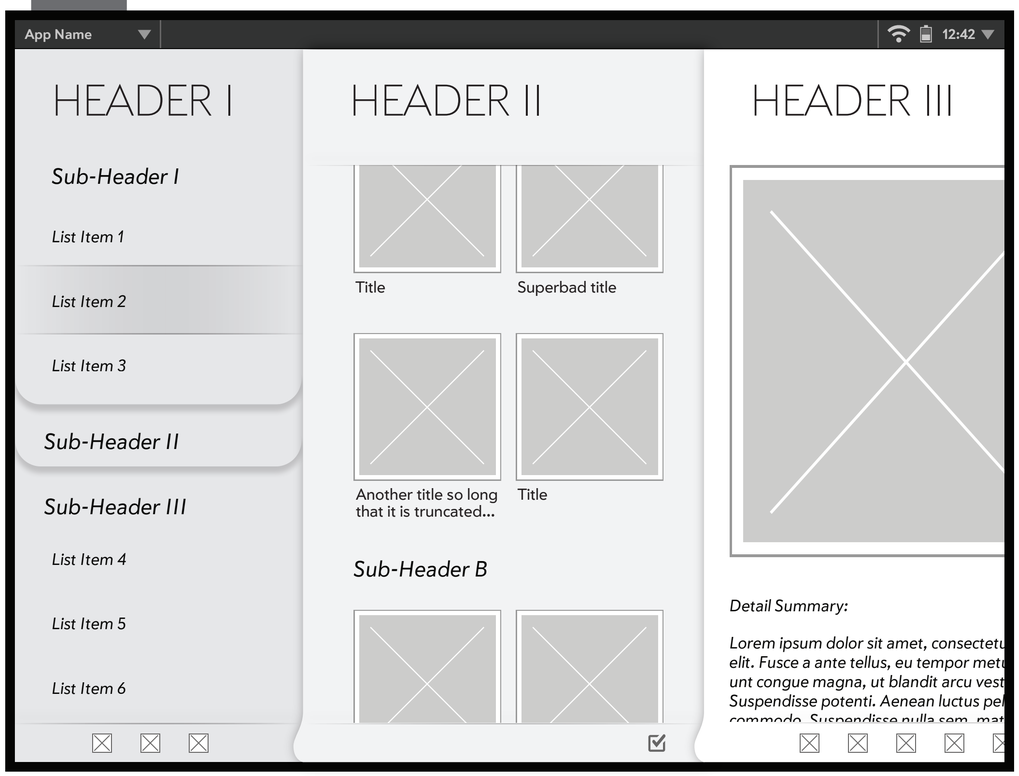
If user taps out, selects RETURN, or dismisses the keyboard without editing, do not make any changes.

EXAMPLE TASK FLOW: PANELS WITH A KEYBOARD (CONT.)

k3



k4



k3. If user has made a change and taps out, hits RETURN, or dismisses the keyboard, save the changes.

k4. Edits are finalized.

CHANGE HISTORY

10/25/2012
- First Draft

11/2/2012
- First launch flows adjusted so that panels are initially right aligned.
- Partial View of Panels flow added
- Peeking Panels flow added
- Panels with Keyboard flow added
- Moving and Dismissing flows removed till further notice.