CodeFreeze

DESIGN SYSTEM

Version 1.0 – Jul2019

CodeFreeze is a **living-breathing** design system. It's purpose is to be the smallest set of options that allow us to design and build everything we need.

Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away.

Antoine de Saint-Exupery

Principles and Guidelines

The principles, design language, and best practices in this document will allow developers to focus on logic, while allowing UX to focus on improving the user experience, interactions and workflows.

We strive to keep these guidelines top of mind as we make decisions. These principles are prioritized by importance.

Role Based

Support users' needs by only presenting data associated with their role and goal.

Consistent & Predictive

Create familiar experiences by strengthening intuition and applying the same solution to the same problem.

Timely

Support current needs for the users by displaying only relevant data as they need it.

Clean & Clear

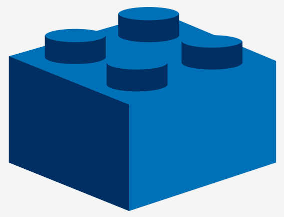
Build lovable experiences by providing actionable results that create value for the users.

The parts and the whole

CodeFreeze is based on Atomic Design methodology. This modularity allows greater flexibility and consistency, while reducing costs and time to market. *http://bradfrost.com/blog/post/atomic-web-design/*

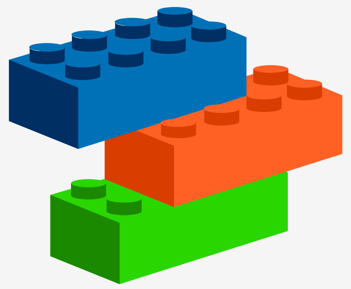
Theme

The theme is the **basic styling** of the application. It includes the colors, fonts, icons, and grid structure. Theming allows external purchasers to skin the application.



Components

Component are the **building blocks** of the application. Our design system utilizes Material Design's components. *https://material.io/guidelines/*



Patterns

A pattern is a simple, reusable **combination of multiple components** that function together as a single unit.

Theme

Theme / Color Palette

System

●	.primary	#0171B7	(1, 113, 183, 1)
●	.secondary	#002F63	(0, 47, 99, 1)
●	.success	#1B8900	(27, 137, 0, 1)
●	.alt	#DA3D00	(218, 61, 0, 1)
●	.danger	#BC0606	(188, 6, 6, 1)

UI Light Theme

●	.stage	#EFEFEF	(239, 239, 239, 1)
●	.board	#F6F6F6	(246, 246, 246, 1)
●	.card	#FFFFFF	(255, 255, 255, 1)

UI Dark Theme

●	.stage	#171621	(23, 22, 33, 1)
●	.board	#1C1C28	(28, 28, 40, 1)
●	.card	#232231	(35, 34, 49, 1)

✕ Do not add custom colors without consulting the UX team.

Neutrals

●	.black	#000000	(0, 0, 0, 1)
●	.mako	#4D5059	(77, 80, 89, 1)
●	.silver	#AAAAAA	(170, 170, 170, 1)
●	.alto	#E0E0E0	(244, 244, 244, 1)
●	.alabaster	#FAFAFA	(250, 250, 250, 1)

Theme / Typography

Font and Font Weights

Text is the primary way our users digest data. Help users complete their tasks by creating a clear visual hierarchy of the data.

Roboto *https://fonts.google.com/specimen/Roboto*

Aa	Aa	Aa	Aa
Light (300)	Regular (400)	Medium (500)	Bold (700)

Typeface styles *https://material.io/design/typography/the-type-system.html#*

H1 / Roboto Light

H2 / Roboto Light

H3 / Roboto Regular

H4 / Roboto Regular

H5 / Roboto Regular

H6 / Roboto Medium

Subtitle 1

Subtitle 2

Body 1

Body 2

BUTTON

Caption

OVERLINE

Color Contrast *http://webaim.org/resources/contrastchecker/*

The Web Content Accessibility Guidelines (WCAG) recommends a threshold ratio of 4.5:1. Background colors used are the two stage colors (light and dark theme).

Primary	Aa ✓	Aa ✗
Mako	Aa ✓	Aa ✗
Silver	Aa ✗	Aa ✓
Alto	Aa ✗	Aa ✓

Theme / Iconography

Material *https://material.io/icons/*

✓ Material icons are less cartoonish… for this reason, Material is preferred over Font Awesome.

✗ Icons often cause usability problems when they are used without consideration… use a text label and don't rely on a hover for clarification.

Font Awesome *http://fontawesome.io/icons/*

✓ Font Awesome icons should be used if a suitable Material icon isn't available.

ℹ Very few icons are universally recognizable by users. See the UX Team if you need help selecting an icon, or need one custom designed.

Label Placement *http://uxmyths.com/post/715009009/myth-icons-enhance-usability*

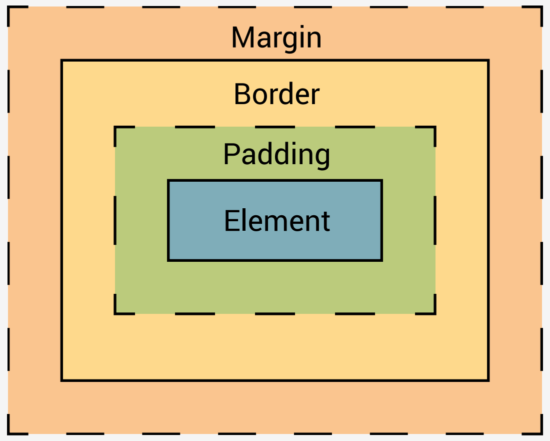
Delete

Settings

✓ Label are placed to the right, or under the icon.

8-Point Grid (Margin and Padding) *https://spec.fm/specifics/8-pt-grid*

Use multiples of 8 to define dimensions, padding, and margin of both block and inline elements. When all of your measurements follow the same rules, you automatically get a more consistent UI. By removing 7 of every 8 spacing options, it allows the developer to eyeball an 8pt increment instead of having to measure each time.



The Box Model

The Box Model is a way to describe an object's dimensions and spacing. It consists of 4 components: border, margin, padding, and the dimensions of the element itself.

Border: the thickness of the stroke around the edges of an element.

Padding: the space between the bounds of an element and its child elements

Margin: the space between the bounds of an element and neighboring objects

Naming

Class	px	rem
.none	0	0
.xs	8	.5
.sm	16	1
.med	24	1.5
.lg	32	2
.xl	40	2.5
.xxl	48	3

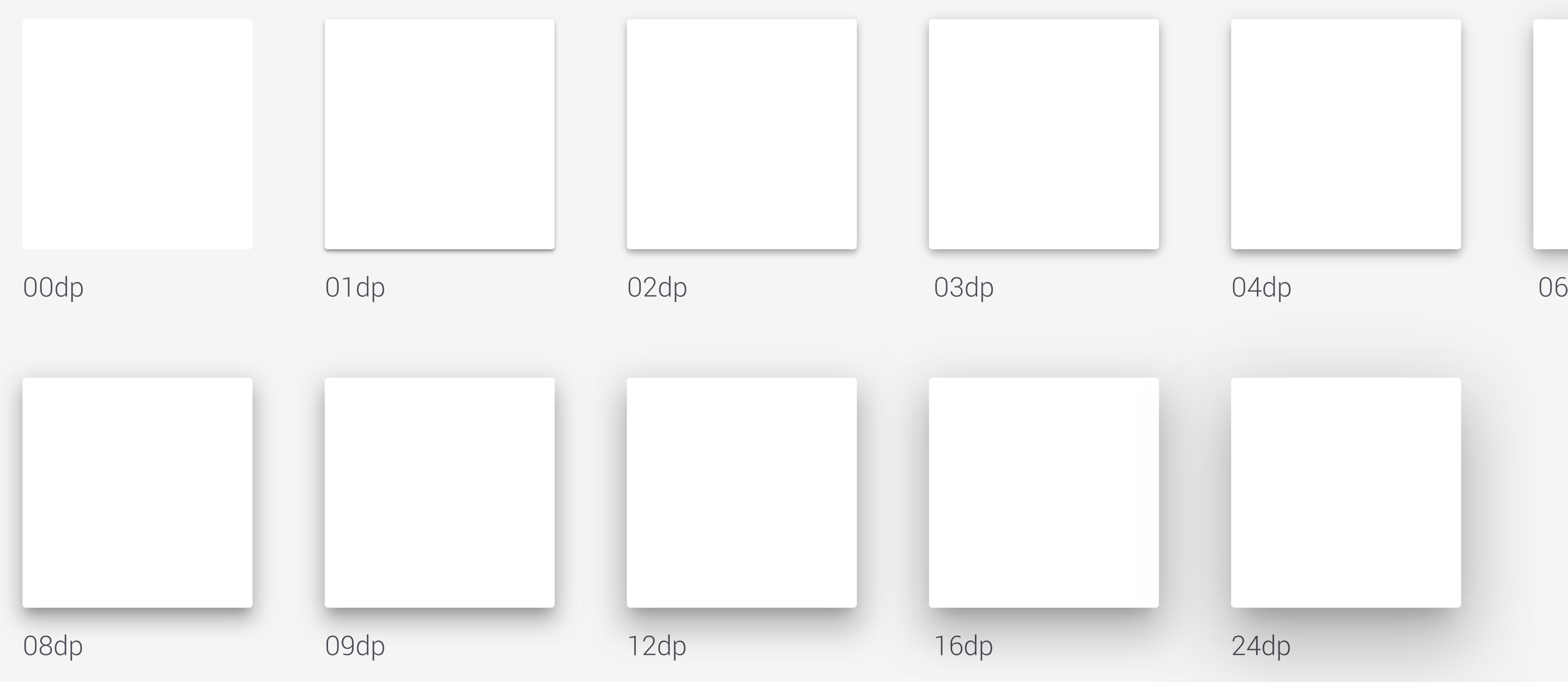
✓ Use these classes to properly size and position the components and patterns.

✗ Do not hard-code margin/padding on components.
Do not use numbers that break the 8-point grid.

Theme / Elevation and Shadows

Elevation and Shadows *https://material.io/guidelines/material-design/elevation-shadows.html#* *https://material-components-web.appspot.com/elevation.html*

Elevation provides important visual cues to users, helping them understand what actions are available. The higher an object’s elevation, the softer and larger its’ shadow becomes.

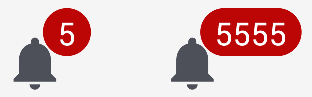


✓ Elevation should be used to create visual hierarchy. Objects with higher elevations are more prominent and should hold the most important information.

Components

Badges

A badge passively indicates unread/unseen content.



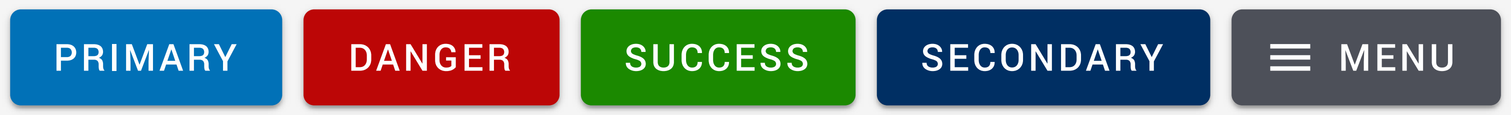
✓ Badges must be red and can only contain an integer. Badges are used when new/unread information is available for the user (comments, notes, etc). Update the integer as soon as the important content is viewed.

✗ Don’t send multiple notifications for the same thing. Badges are designed to be passive, and should not be used for critical information. Badge ≠ Count.

Buttons *https://material.io/guidelines/components/selection-controls.html#*

Buttons communicate what actions are available to the user.

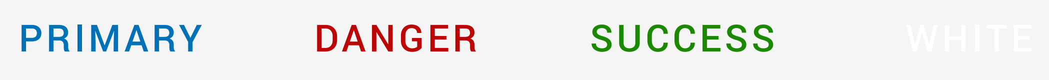
Raised Buttons



✓ This button is used to add dimension and emphasize important functions. Raised button representing the primary actions, and should be placed on the right of any secondary buttons.

✗ The text and color of the button should reinforce the action being taken, so don't rely on color and/or icon alone.

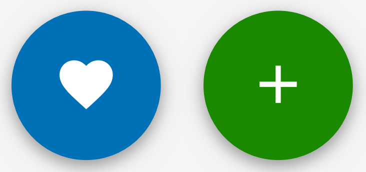
Basic Buttons



✓ This button is used for general functions and reduce the amount of layering on the screen, making it more readable.

✗ Never use more than two secondary buttons. If three or more secondary actions are available, consult the UX Team.

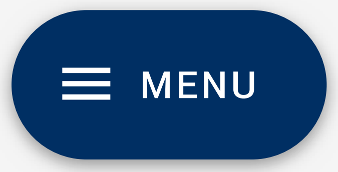
Floating Action Buttons (FAB) *https://material.io/design/components/buttons-floating-action-button.html#*



✓ Very few screens warrant a floating action button. FABs are only allowed to be placed in the bottom-right corner of the stage, and represent the primary application-wide action.

✗ Don't use icons that make the user interact with it to figure out what it does. No screen is permitted multiple FABs.

Extended FAB *https://material.io/design/components/buttons-floating-action-button.html#extended-fab*



✓ The text label of an extended FAB should describe its action.

✗ Avoid wrapping text... keep labels short.

Chips

https://material.io/guidelines/components/chips.html

A chip is a small block of supporting data such as a avatar, text or a status. Chips are placed to the right of the data it supports.

Default Primary Success Alt Danger Disabled

✓ Chips should only represent one chunk of data, but can be used in groups. Color can be used to help convey the data (ie. green for a good status, red for bad) but should be used sparingly.

✗ If the data isn't supporting the data directly to its left, a different component is needed. Avoid long, run-on text. Inactive Chips can only be used if the supporting data is inactive.

Deleteable Chips

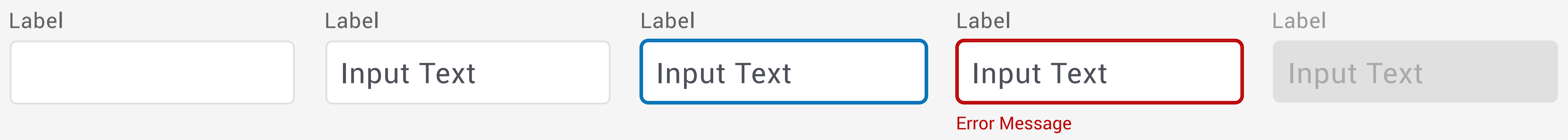
I can delete this ⓧ

✓ Deleteable Chips should only be used when the user added the Chip to the interface.

✗ Never allow users to remove an element from the UI without a way to add it back or undo the action.

Form Field *https://material.io/design/components/text-fields.html*

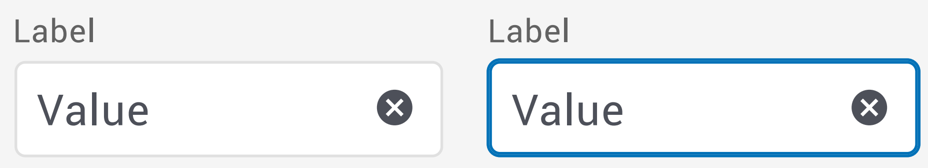
Form Fields allow users to input text and usually appear in forms. Users may enter text, numbers, or mixed-format types of input.



✓ Label must be descriptive and short. Use good defaults, and autocomplete when at all possible.

✕ Avoid really long or wrapping labels. Don't repeat section headers with the same label.

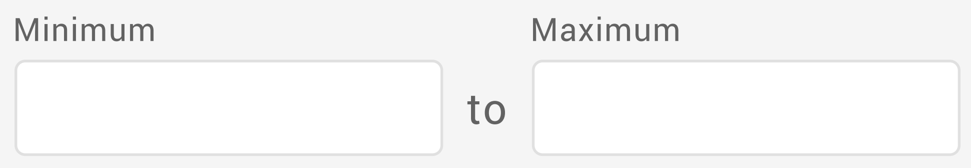
Clear Input



✓ Displays only after characters have been entered.

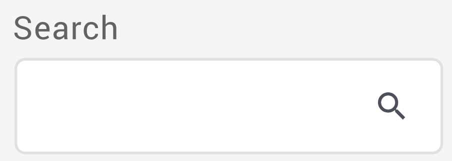
✕ Only use on fields where it makes sense for the user to empty the input field.

Range Input



✓ The minimum is on the left, maximum on right.

Search Input

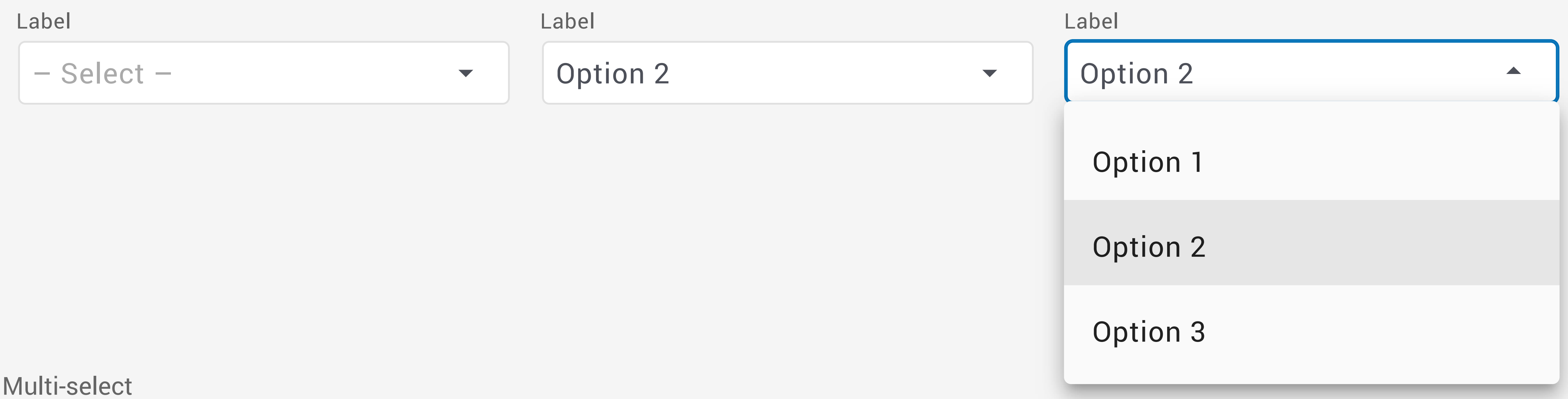


✓ Only use when searching on a specific field.

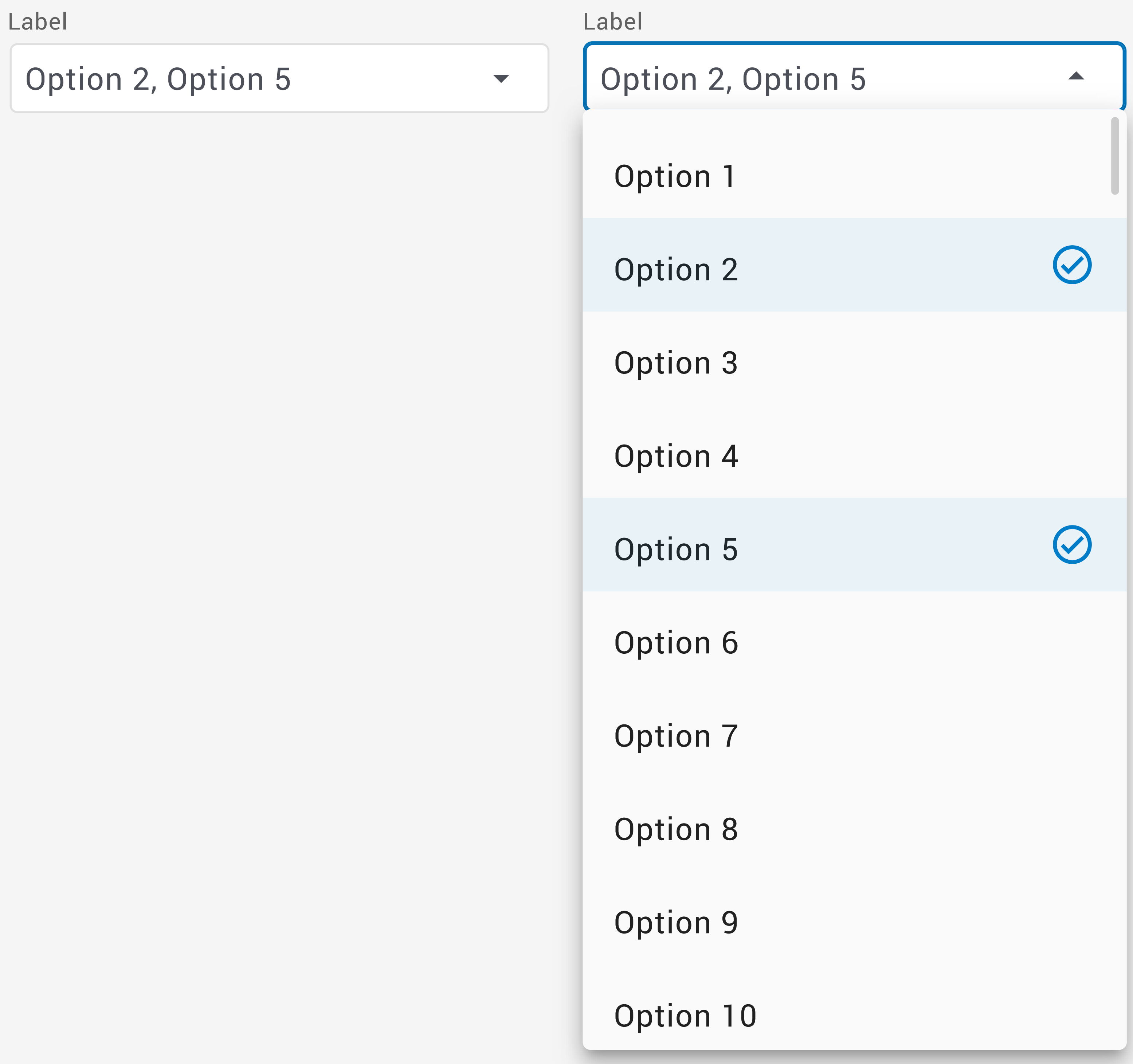
✕ Don't use for large, app-wide searches… it lacks the affordance of a large-scale search.

Select Fields *https://material.io/design/components/menus.html#dropdown-menu*

A dropdown menu is a compact way of displaying multiple choices. It appears upon interaction with an element.



Multi-select



✓ Follow the guidelines set forth in the Text Field section.

✗ Don't allow the window to overflow the stage… the entire window should be visible.

Components / Date Picker

Date Picker

https://material.io/design/components/pickers.html

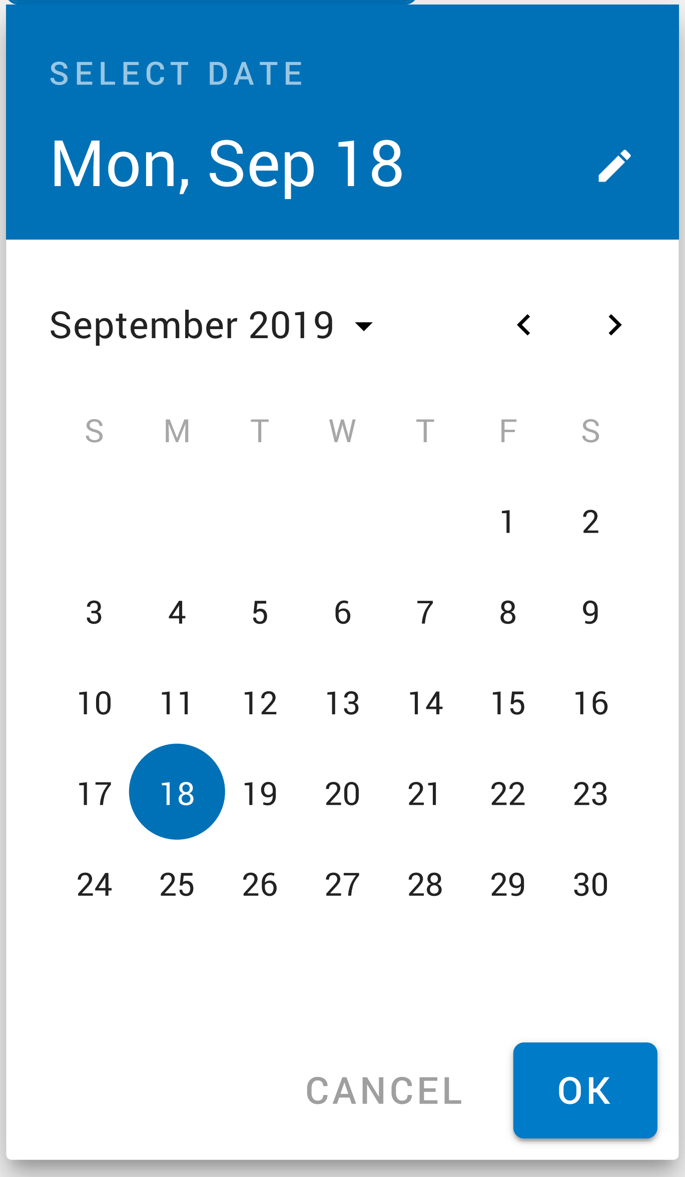
A control used for selecting a single date.

Date

dd-mmm-yyyy

Date

18-Sep-2019



✓ Default picker is Inline Container with AutoOK.

✗ Don’t include the word “Date” in the label. The calendar icon and date (when filled in) are adequate affordance.

Selection Controls *https://material.io/guidelines/components/selection-controls.html#*

Selection controls allow the user to select options.

Checkbox

- ☑ On
- ☐ Off
- ☑ Disabled On
- ☐ Disabled Off

✓ For selecting multiple options from one set.

✗ Not to be used as a single on/off switch. Use a toggle in such cases.

Radio Button

- ◉ On
- ○ Off
- ◉ Disabled On
- ○ Disabled Off

✓ For selecting a single option from a set. Default to the most likely selection (when possible).

✗ Avoid long lists of options. After five options, consider a dropdown.

Switches

- On
- Off
- Disabled On
- Disabled Off

✓ Use when a single settings is either True/False.

Toggle Button *https://material.io/design/components/buttons.html#toggle-button*

Toggle buttons can be used to group related options. To emphasize groups of related toggle buttons, a group should share a common container.

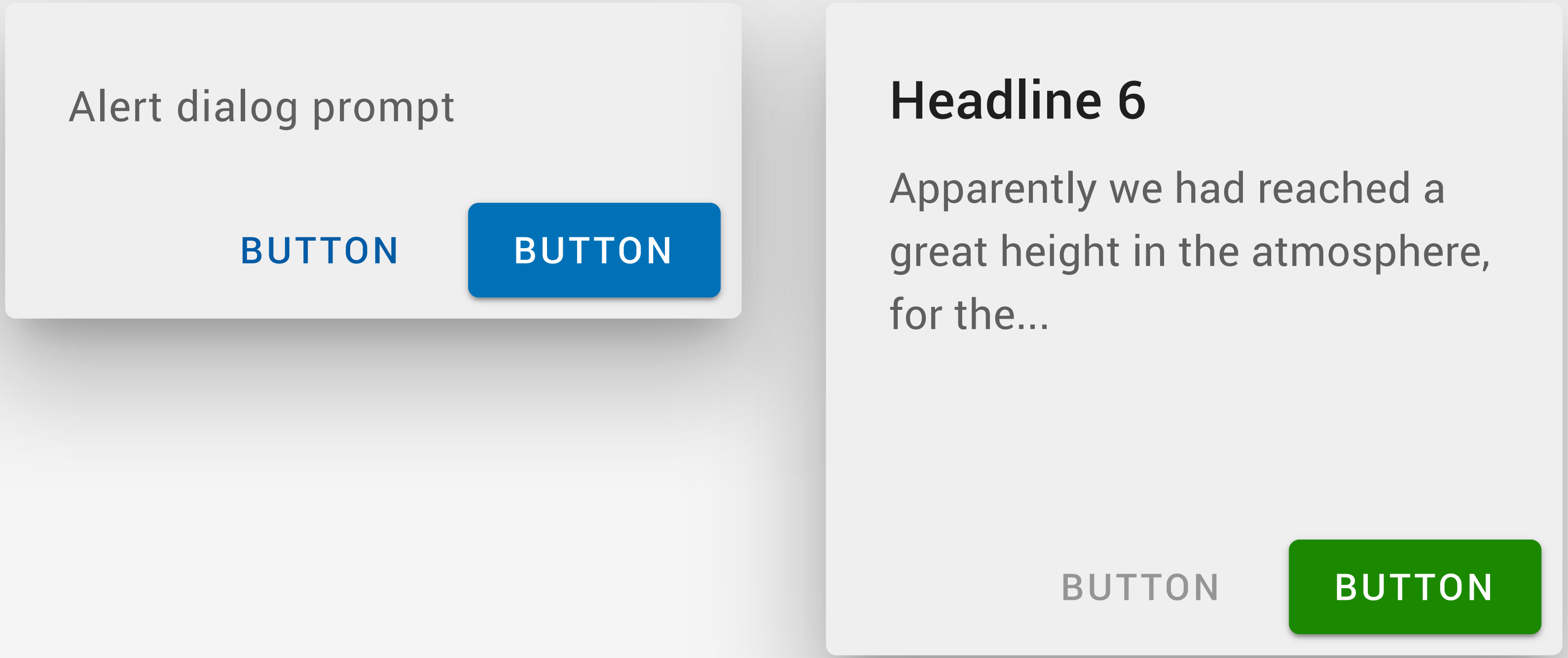
♥ 👁 🛒

✓ Toggle buttons behave as radio buttons by default.

✗ Don't rely on icons alone... this example is strictly for demo purposes.

Dialog *https://material.io/design/components/dialogs.html*

Dialogs inform users about a task and can contain critical information, require decisions, or involve multiple tasks. Dialog Buttons should help users make those decisions.


Alert dialog prompt

Headline 6

Apparently we had reached a great height in the atmosphere, for the...

✓ Use dialogs sparingly because they are interruptive. Not every choice, setting, or detail warrants interruption.

✗ Don't open a dialog from within a dialog. Avoid scrolling in dialogs.

Snackbar *https://material.io/design/components/snackbars.html*

Snackbars provide brief messages about app processes at the bottom of the screen. Snackbars can contain a single action.

Default

Your template was published.

✓ The default should contain a single line of text directly related to the operation performed.

✗ Don’t stack snackbars… only one can be open at a time.

Color and Action

Your template was published. UNDO

✓ The snackbar can have a background color of Success or Danger to convey the result of an operation (example: “Save Successful”). A single action button can be included (if needed to complete the operation).

Multi-line and Long Text Button Example

Greyhound divisively hello coldly wonderfully marginally far upon excluding.

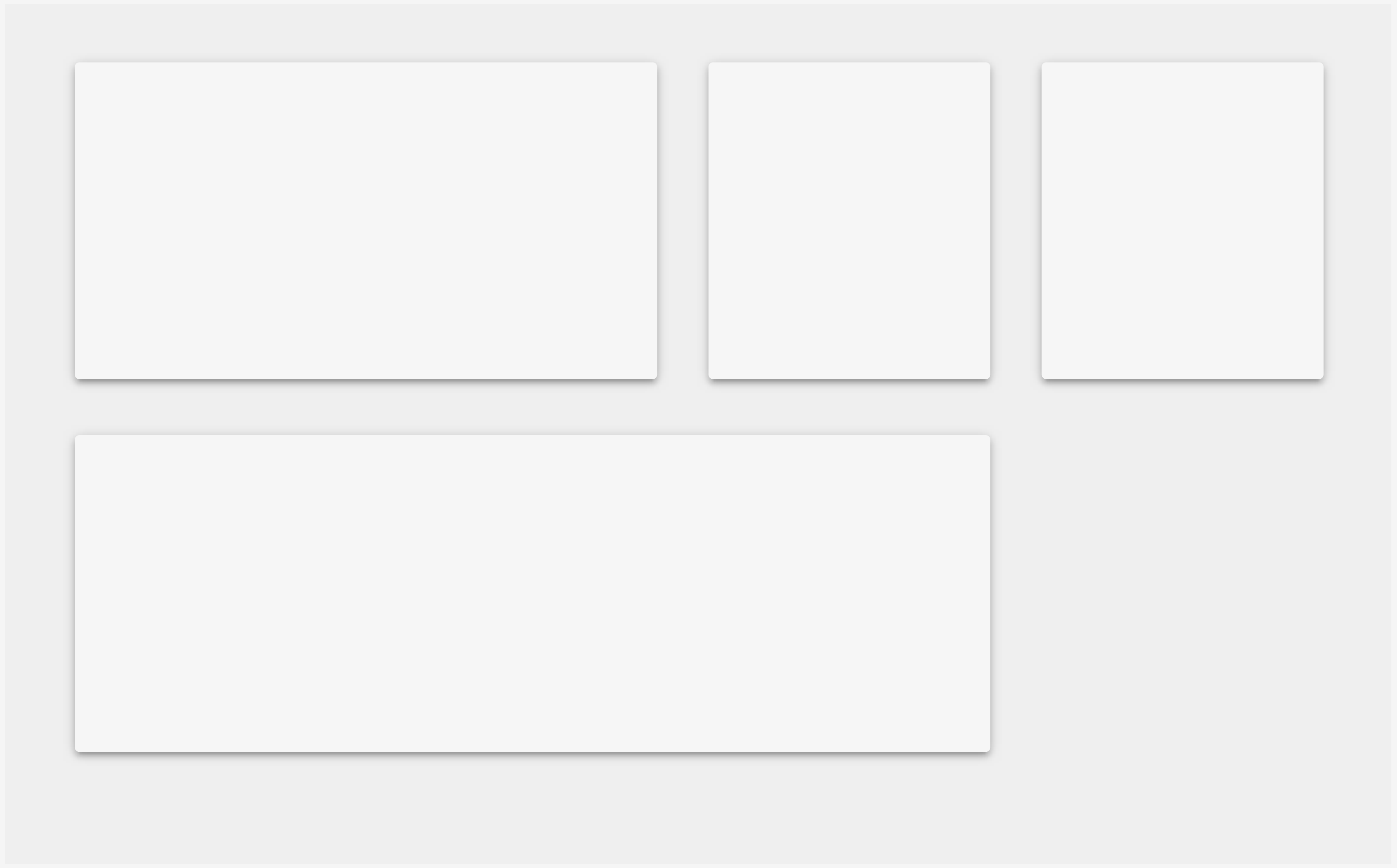
LONG TEXT BUTTON

✗ Snackbars don’t require user input to disappear… don't use the button to make users close a snackbar.

Sheets

https://material.io/guidelines/components/cards.html#

A sheet are essentially an empty card used to group data into logical chunks and to create visual hierarchy on the work area.

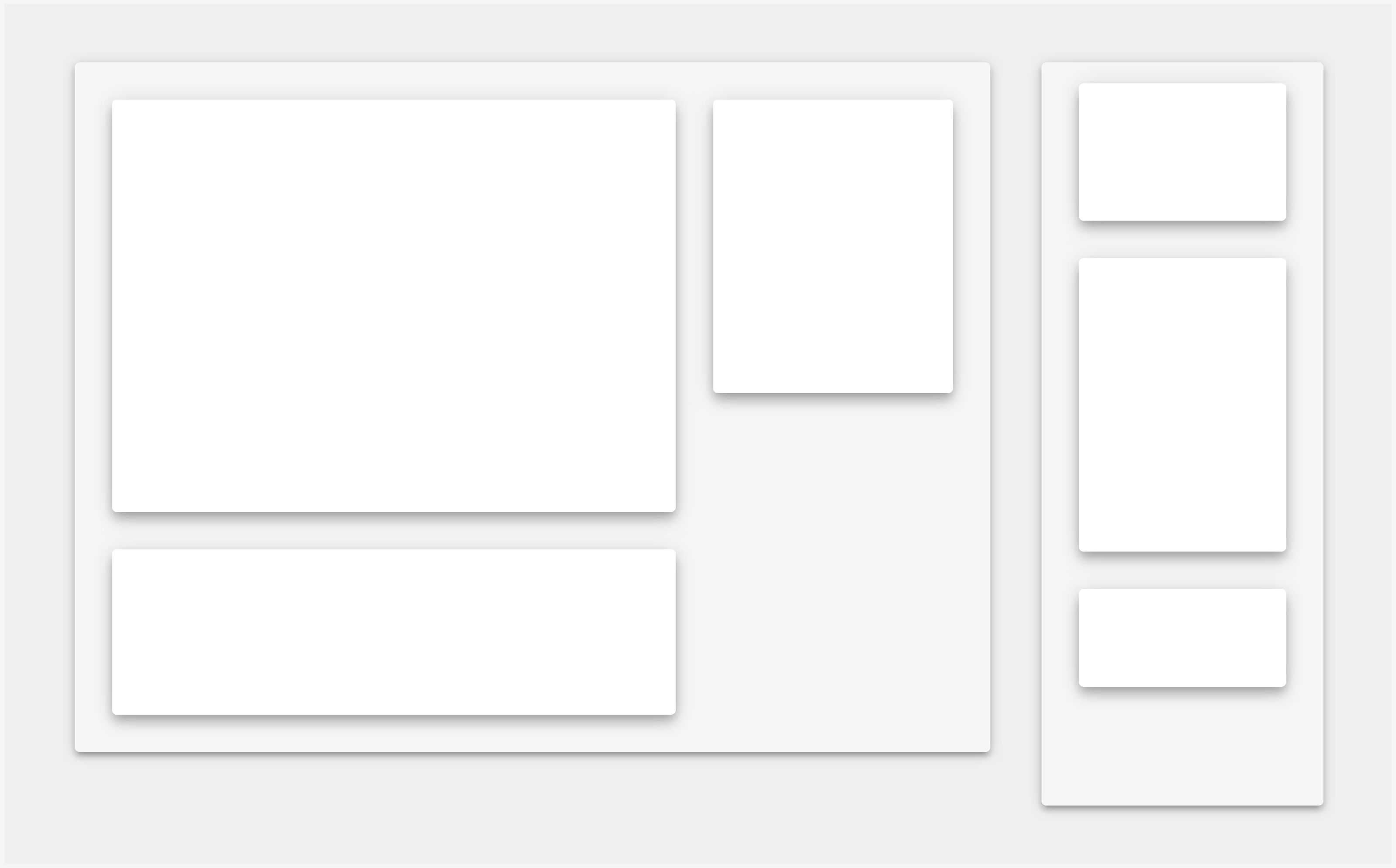


✓ Sheets are always the same color and elevation (2dp). Sheets can be tabbed when the data and/or workflow necessitate it. Layouts are created using Flexbox and should never overflow-x (causing horizontal scrolling).

✗ Sheets can not overlap because they are on the same elevation.

Cards https://material.io/guidelines/components/cards.html#

Cards contain content and actions about a single subject. Cards allow for another level of grouping data into chunks.

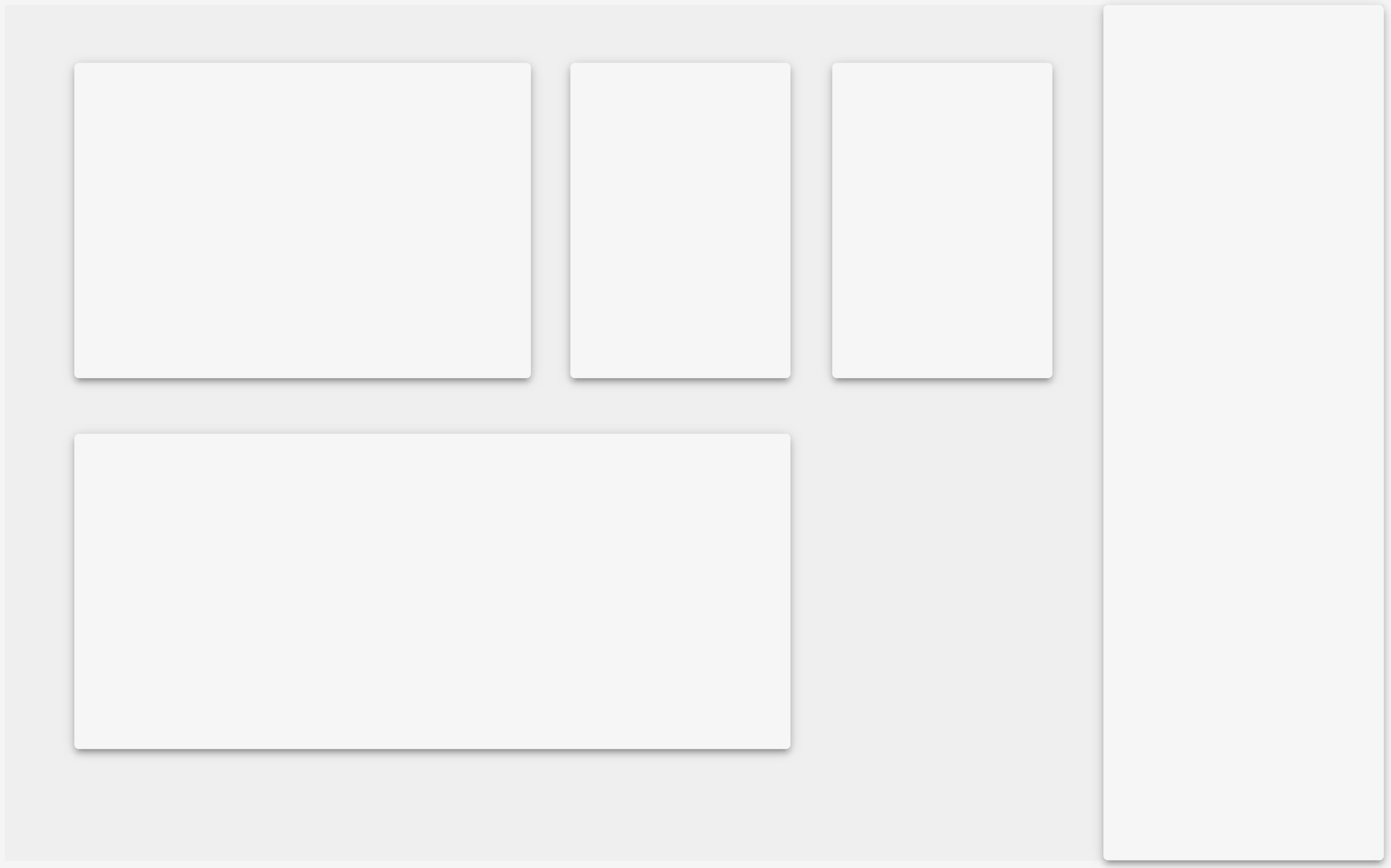


✓ Cards are always the same color and elevation (4dp). Cards are never tabbed. Card layouts are created using Flexbox and should never overflow-x (causing horizontal scrolling) the sheet containing

✗ Cards are not to be used as a design element… they are only used to group data within a board.

Side Sheets https://material.io/design/components/sheets-side.html#

Side sheets are surfaces containing supplementary content that are anchored to the left or right edge of the screen.



✓ The primary usage will be to display supporting information or metadata. Details that can help users complete their task without navigating away for additional information.

✗ Don't ask users to open additional elements to find the data they need to complete their task.

Tables

https://material.io/design/components/data-tables.html

Data tables display information in a way that's easy to scan, so that users can look for patterns and insights.

Header	↓ Sorted Header	Left align text	Right align numbers	Flex columns to data size
Value	Value	Value	5.00	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Value	Value	Value	55.00	How to truncate if necessary. Lorem ipsum consectetu…
Value	Value	Value	555.00	Lorem ipsum dolor sit amet, consectetur.

✓ For tabular data only. Use the default Material table but add zebra stripping

✗ Don't create custom sorts or filters for tables… use the Material defaults when needed.

Lists

https://material.io/design/components/lists.html#

Lists are container components that wrap and format a series of line items. As the base list component, it provides Material Design styling, but no behavior of its own.

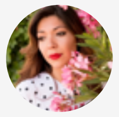
Expandable Lists ⌄

Expandable Lists ⌄

Selectable Lists ☑

Selectable Lists ☐

Avatar List with Menu ≡


Avatar List with Menu ≡

List Item Name
Short description / text
Meta Data

List Item Name
Short description / text
Meta Data

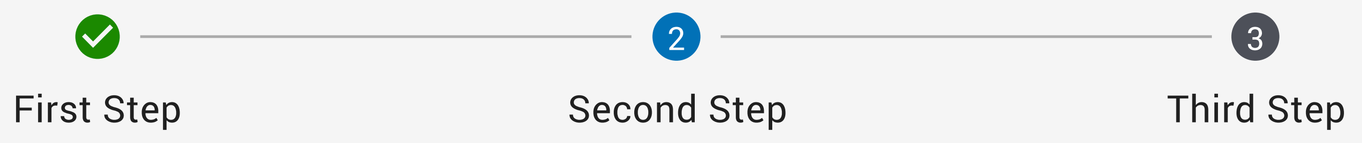
✓ Lists present content in a way that makes it easy to identify a specific item in a collection and act on it.

✗ Lists should direct users to an action… or more information… don't overload a list item with data.

Components / Stepper

Stepper https://material.angular.io/components/stepper/overview

The stepper provides a wizard-like workflow by dividing content into logical steps.

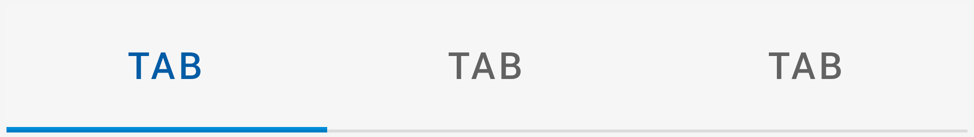


✓ Steps should be clear… show as much detail as possible… and allow users to see how much work remains.

✗ Don't use as the only way to navigate through the steps… provide some sort of Next/Back options. Wizards should be reserved for workflows that are not performed regularly.

Tabs *https://material.io/design/components/tabs.html*

Tabs organize content across different screens, data sets, and other interactions.



BUTTON
BUTTON
BUTTON

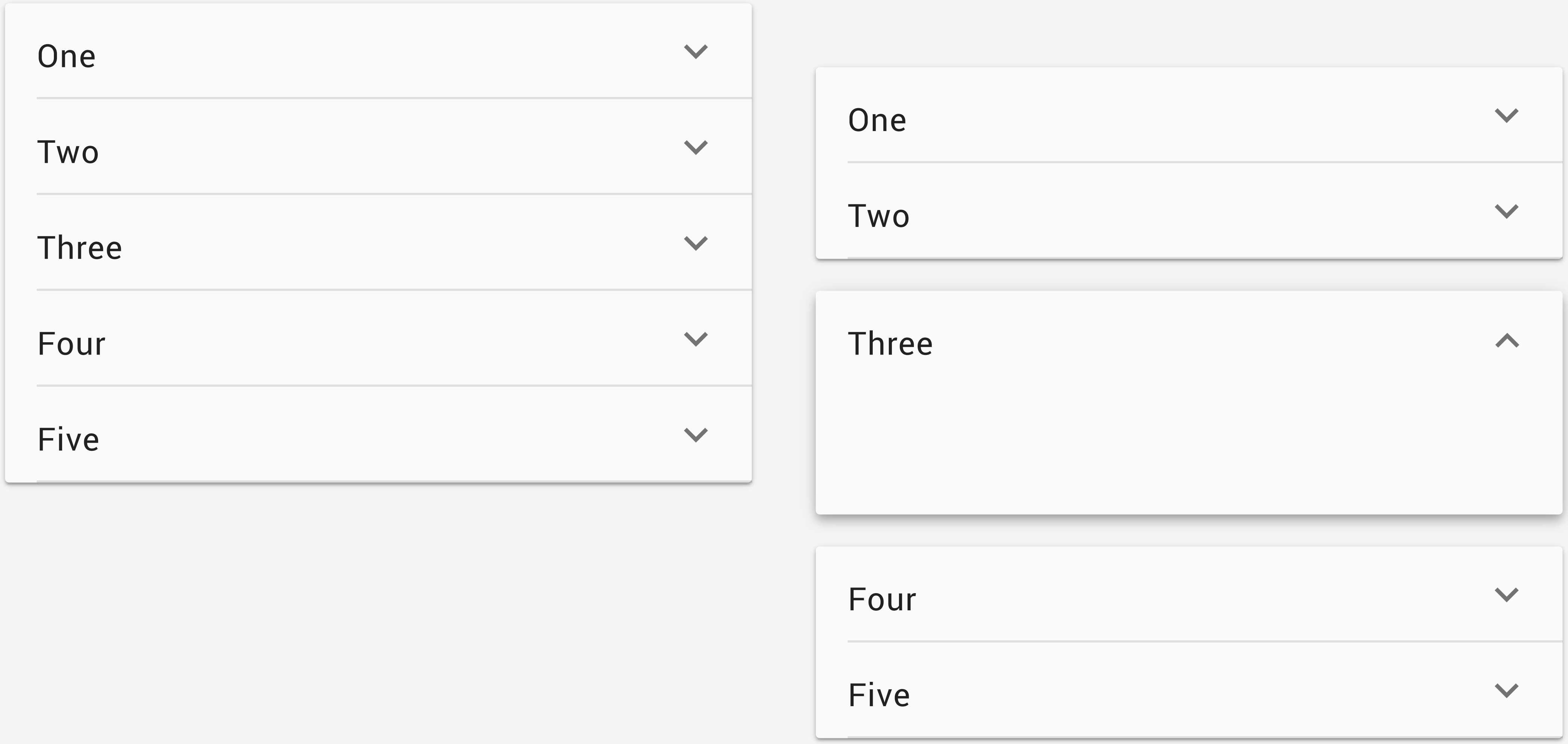
✓ Tabs organize content into categories to help users easily find different types of information.

✗ Cards and Sheets can be tabbed… use the background color of the component it sits on. Don't make tabs colorful to stand out.

Expansion Panels

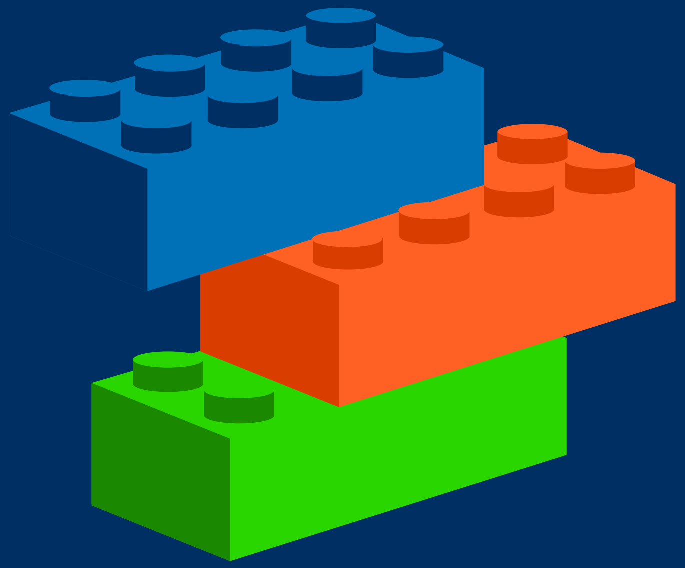
https://material.angular.io/components/expansion/overview

Expansion Panels allow content to be placed within expandable sections.



✓ A collapsed panel displays summary information of the data it contains. Expanded panels, like cards, are a blank canvas. They can contain a variety of data.

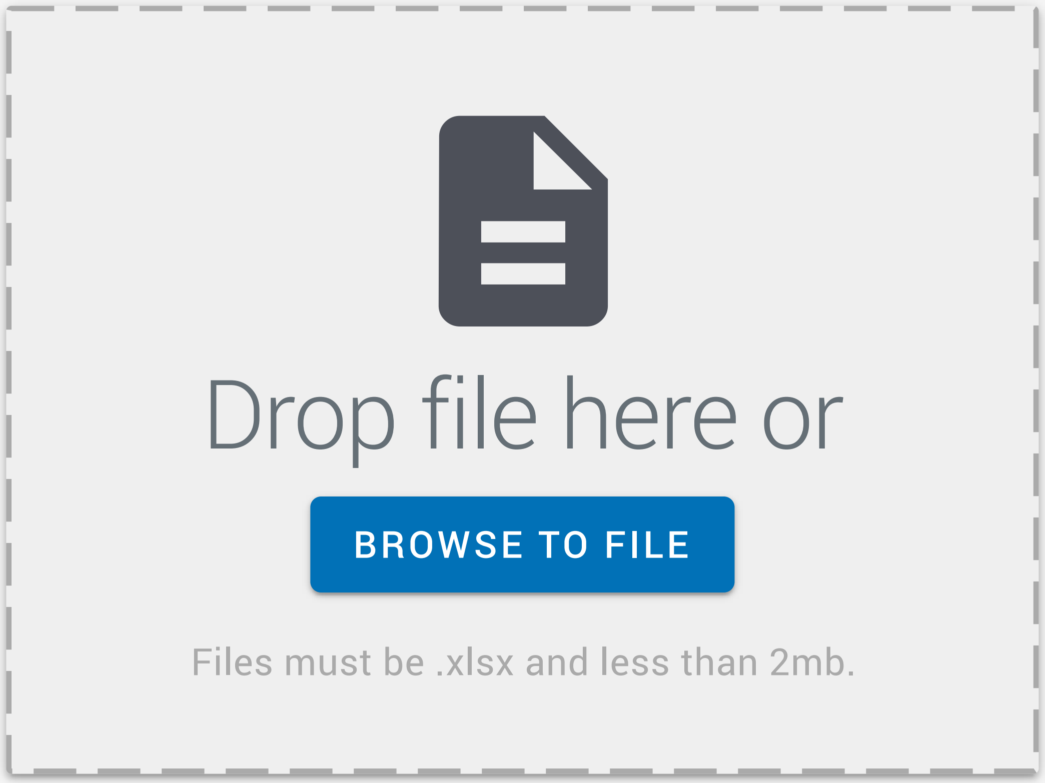
✗ Never hide pertinent information. Give the user what they need and allow them to get more (if requested).



Patterns

Uploads

This pattern allows users to upload information into the system.



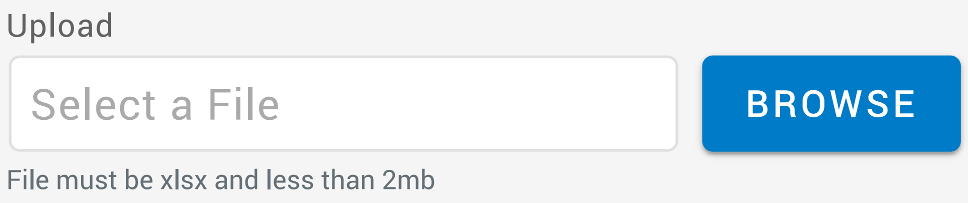
Drop file here or

BROWSE TO FILE

Files must be .xlsx and less than 2mb.

✓ This large drop zone with browse feature is the preferred option (when viable).

Inline / Form



Upload

Select a File

BROWSE

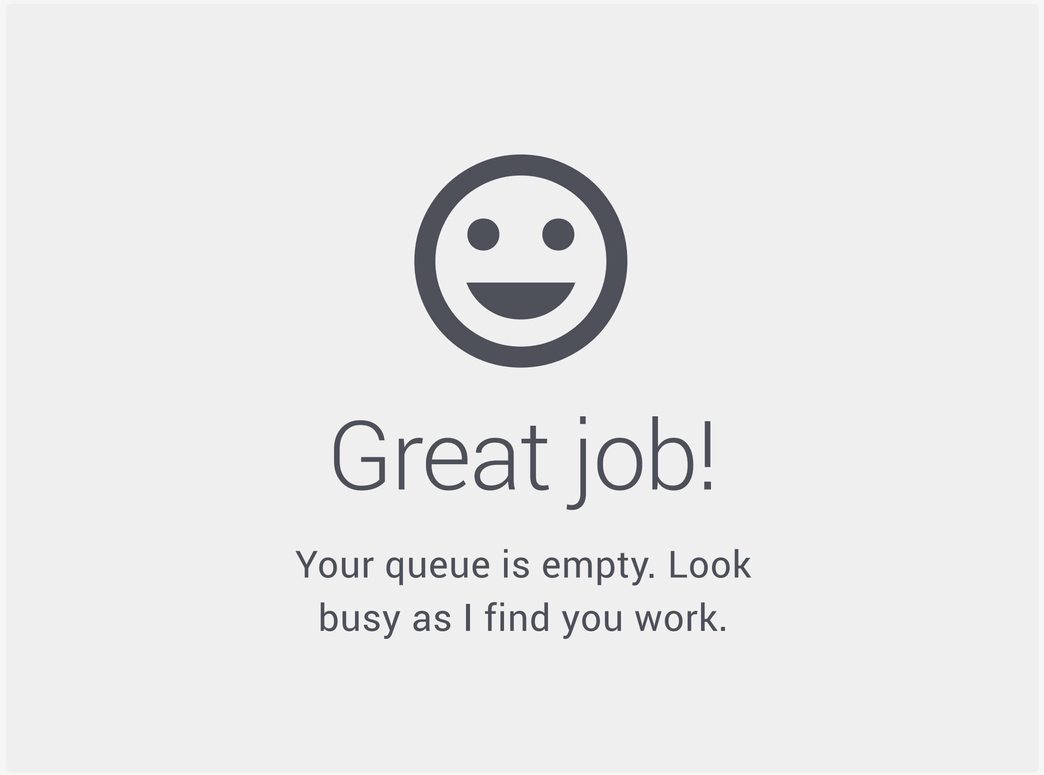
File must be xlsx and less than 2mb

✓ Use this pattern when users are restricted to a single file.

✗ Don't stack this pattern to allow multiple files. If users can upload multipe files use the drop zone.

Empty State

An empty state, or zero-data state, notifies users when an item’s content can’t be shown.



Great job!

Your queue is empty. Look busy as I find you work.

✓ Think of this empty state as a mini landing page. While minimal in design, a successful empty state will explain a specific feature and then compel the user to take the next step.

✗ Empty states are not to be used for system errors.

Progress and Loading

Progress and activity indicators are visual indications of an app loading content.

Spinner *https://material.angular.io/components/progress-spinner/overview*

✓ Use the spinner (indeterminate) as the default progress indicator.

Bar *https://material.angular.io/components/progress-bar/overview*

Importing (58%)

✓ When progress can be accurately calculated, use a progress bar and include the progress in the label.

✗ Avoid for long processes where the bar appears to stop. Use the spinner so users see they system isn't frozen.

Displaying Data

How data is displayed greatly impacts the users ability to accomplish their task. Be mindful of what task the user is trying to accomplish when displaying data.

Vertical

Study

T400-T19

✓ The default display of data pairs is vertically. Use short, descriptive labels.

✗ Don't mix displays and text fields because of their similar styling. Avoid clustering too many displays.

Horizontal

Study	MT400-T19
Sex	F
Group	Vehicle Control
Time	DAY 90

✓ A zebra-striped display can be used if the data is easier to digest/compare. Inline edits are displayed as links.

Something missing?

If your story or feature cannot be solved with existing components or pattern, consult the UX Team.