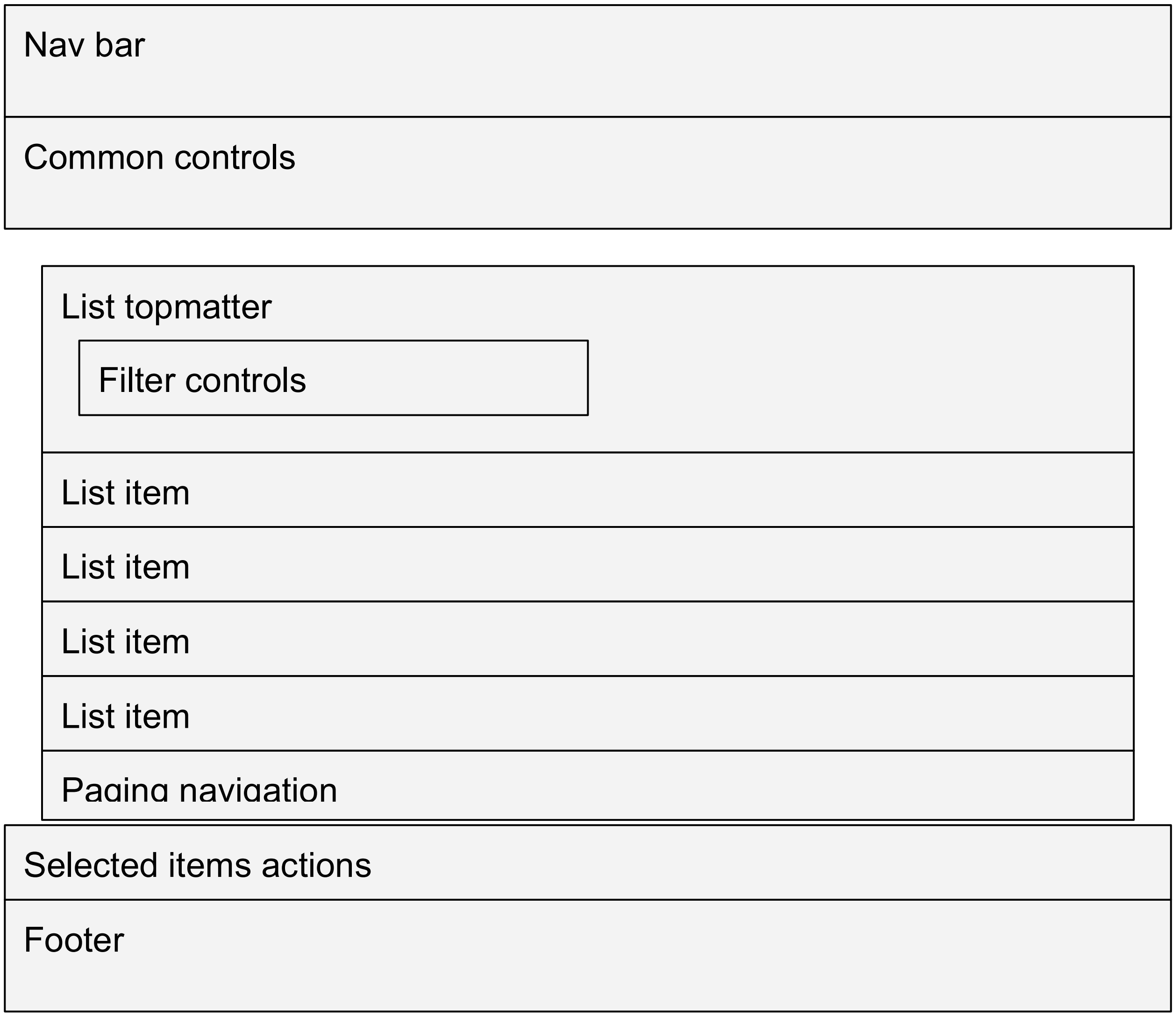
WGU OSMT

Acceptance Criteria for Patterns

Things common to multiple views

Patterns in a typical list view, for reference



The typical page is a stack of nested patterns:

1. Nav bar
2. Common controls

3. Content, in this example a list view consisting of
    a. List topmatter (describes the centerpiece of the view)
    b. Filter controls (operates on the centerpiece of the view)
    c. List items (the centerpiece of the view)
    d. Paging navigation (operates on the centerpiece of the view)
4. Selected items actions (operates on the data elements in the centerpiece of the view)
5. Footer

These patterns and others are described below.

White label

A few white label parameters are available to the organization building an instance of the OSMT

1. A *logo* URL
2. A *product name*
3. A single *brand color* hex value
4. A default *author* string
5. Whether or not the *author* field is editable
6. A *creator* URL
7. A *copyright* string
8. A *public skill view title* string
9. A *public collection view title* string
10. An *IdP logout redirect URL*

Design System

White Label

Behavior: *Author* visibility

The visibility/behavior implied by "whether or not the author field is editable" is covered in relevant views *new/edit skill form*, *new/edit collection form*, *skill view*, and *collection view*. But in general,

1. In forms, the author field is not shown if it is not editable.
2. In non-form single-item views, i.e. *single skill* and *collection*, author is visible since the view hearkens to the public view.

3. Author is always visible on the *public view of skill* and *public view of collection*.

Pattern: page <title>s (metadata)

Views

Every page should have a sensible and informative <title> element. This is the content that typically appears in window headers, browser tabs, bookmarks, and link text on other sites to help users identify these at a glance.

Elements and behavior

Internal Views

Detail Views Pattern

{RSD or collection name} | {page heading/H1*} | {tool name}

examples:
- Written Communications | Edit Rich Skill Descriptor | OSMT
- Information Technology Management – B.S. Business Administration | Collection | OSMT

Non-Detail Views Pattern

{page heading/H1*} | {tool name}

examples:
- RSD Library | OSMT
- Batch Import | OSMT
- Search Results | OSMT

*Notes About Headings

A few views do not have the typical page heading/H1. These are listed here with the text string to use in the <title> of each view.

View	**Text String to Use in <Title>**
view/manage RSD (detail view)	Rich Skill Descriptor

view/manage collection (detail view)	Collection
confirmation pages and guards	Heads Up
error states	Error
logged out of OSMT	Logged Out

Public Views

RSD View Pattern

{RSD name} | Rich Skill Descriptor | {tool name}

Collection View Pattern

{Collection name} | Collection | {tool name}

examples:

- Written Communications | Rich Skill Descriptor | OSMT
- Information Technology Management – B.S. Business Administration | Collection | OSMT

Pattern: nav bar

Views

The nav bar appears over all internal (logged in) views and does not appear over the *public skill* or *public collection* views.

Design System

Nav Bar

Elements and behavior

1. The *logo* from the whitelabel parameters
2. The *product name* from the whitelabel parameters
3. A hidden "skip to content" link (this can be found as part of the broader Header pattern)

4. A hidden “skip to search” link (this can be found as part of the broader Header pattern)
5. A link to the *skills library* view
6. A link to the *collections library* view
7. A link to log out

Pattern: public nav bar

Views

The public nav bar appears over public views, namely the *public skill* and *public collection* views.

Design System

Nav Bar - Public

Elements and behavior

8. The *logo* from the whitelabel parameters
9. A *title* from the whitelabel parameters
10. A hidden “skip to content” link

Pattern: footer

Views

The footer appears beneath all views, internal (logged in) or otherwise (public).

Design System

Footer

Elements/data

1. The *copyright* string from the whitelabel parameters
2. A *powered by* string, currently “Powered by the Open Skills Network” that may include a link to an external site

Pattern: common controls

Views

The common controls module appears over all internal (logged in) views **except**

- *Batch import* process views
- *Guards* against publishing a collection with draft or archived skills

Design System

Common Controls

- Each action item can be either an anchor or a button depending on engineering needs. Buttons do something, anchors go somewhere. The class can be attached to either element
- Common controls appear alongside Nav Bar as part of the Header pattern

Elements and behavior

1. A *search* field with two actions
    a. Search skills, which leads to a *skills search results* view and is the default action of the form
    b. Search collections, which leads to a *collections search results* view
2. A link to *advanced search*
3. A *Create Skill* button that leads to the *new skill* form
4. A *Create Collection* button that leads to the *new collection* form
5. A *Batch Import Skills* button that leads to the beginning of the *batch import* process
6. Once a user has taken an action from the common controls, and the new page loads, screen reader focus would go to the new page header.

Pattern: skills list

Views

Included on views

1. *Skills library*

2. *Skills search results* from common controls search field
3. *Collection*
4. *Skills search results* within collection
5. *Skills search results* for outside-of-collection skills from a collection view

Does not include views

1. *Public view of collection*

Design System

- Skills List
- Skills List Empty

Elements/data

Surrounding the list

1. A caption that shows the number of skills returned, of the form “{x} of {y} skills in {context}” where {context} refers to the skills library or the specific collection in question, {x} refers to the number of filtered skills returned, and {y} refers to the unfiltered number of skills in that context.
2. A caption that shows the paginated range of collections displayed on the page "Viewing skills {m}–{n}” where {m} refers to the skill number displayed at the top of the page and {n} refers to the skill number displayed at the bottom of the page.
3. Filters that restrict the skills shown, discussed in *Module: Filters*
4. A checkbox that selects all items on all pages
    a. And is labeled with “select all” and the number of items that will be selected when the checkbox is operated
5. Group actions that operate on the one or more selected skills, appearing after the list of skills, discussed further in *Module: selected items actions*
6. Paging navigation

Of the list itself

7. Zero or more skills in a responsive table/card format, explained by *Pattern: skill as a list item*
8. An empty state message if no skills – this message differs by view
9. Default sort order is ascending alpha by *category*
10. 50 skills will be displayed on a single page

11. On narrow viewports, the sortable headers “Category” and “Skill” collapse to a single header “Skill” that reveals a menu to sort by category or skill name

Pattern: public skills list

Views

Included on *Public view of collection*

Elements/data

1. Zero or more skills in a responsive table/card format
2. A first column, headed “Category”, containing the category for each skill listed
3. A second column, headed “Rich Skill Descriptor”, containing for each skill listed
    a. The skill name, which is also a link to the public view of that skill
    b. The skill statement
    c. The skill’s archived status, if applicable
4. An empty state message if no skills – this message differs by view
5. Default sort order is ascending alpha by *category*
6. 50 skills will be displayed on a single page
7. On narrow viewports, the headers “Category” and “Skill” collapse to a single header “Skill”
8. **Stretch**: On narrow viewports, the sortable headers “Category” and “Skill” collapse to a single header “Skill” that reveals a menu to sort by category or skill name

The only clickable items in this pattern are the skill names.

Pattern: collections list

Views

Included on views

1. *Collections library*
2. *Collections search results* from common controls search field

Design System

Collection List

Elements/data

Surrounding the list

3. Filters that restrict the collections shown, discussed in *Module: Filters*
    a. By default, Draft and Published collections are selected, but Archived is NOT selected.
4. A caption that shows the number of collections returned, of the form “{x} of {y} collections in {context}” where {context} refers to the collections library, {x} refers to the number of collections returned, and {y} refers to the number of collections in that context
5. A caption that shows the paginated range of collections displayed on the page "Viewing collections {x}–{y}” where {x} refers to the collection number displayed at the top of the page and {y} refers to the collection number displayed at the bottom of the page. A group of 50 collections will be displayed on a single page.
6. A “select all” checkbox
    a. And is labeled with the number of items that will be selected when the checkbox is operated
7. Group actions that operate on the one or more selected collections, appearing after the list of collections, discussed further in *Module: selected items actions*
8. Table can be sorted by:
    a. Collection name A–Z or Z–A.
    b. Number of skills low–high or high–low.
    c. On mobile, the table sort is collapsed into a single drop down and the table header becomes “Sort”

Of the list itself

9. Zero or more collections in a responsive table/card format, explained by *Module: collection as a list item*
10. An empty state message if no collections – this message differs by view

Pattern: filter controls

Filters govern what skill or collection items are shown in a list of skills or collections. The filter options are always the same, but which are selected by default varies somewhat by view.

Design System

Filter Controls

Elements/data

1. The filters available are the same for all lists of skills and collections
   a. Draft – when on, show draft (unpublished) items in the list
   b. Published – when on, show published items in the list
   c. Archived – when on, show archived elements in the list
2. The filter controls are always visible above a list, except for public views
3. **Stretch**: The filter control labels show how many skills' visibility is governed by that filter
4. Which filters are selected by default differs per view and is discussed in the entry for each relevant view
5. There is a hidden "filter" label before the first filter.

Pattern: skill as a list item

Design System

Skill as a List Item

Elements/data

Each skill in a list of skills contains

1. The skill's *category*
2. The skill's *skill name*, which is required, and is also a link to the single-skill view of the skill
3. A hidden link to the next skill
4. The skill's *skill statement*, which is required

5. The skill’s keywords, if any, which may be truncated if longer than the width of the table cell/card
    a. If truncated, an indicator will appear to signify that there is hidden information
6. The skill’s *detailed occupations* (BLS detailed occupation job codes), if any, which may be truncated if longer than the width of the table cell/card
    a. If truncated, an indicator will appear to signify that there is hidden information
7. The skill’s *draft* status, if applicable
8. The skill’s *archived* status, if applicable
9. A checkbox that selects the skill
10. A hidden “jump to actions” link after each checkbox that brings the user to the actions in the action bar
11. An overflow action menu per skill – actions differ by view and by statuses of the skill. Possible actions include
    a. Archive skill
    b. Unarchive skill
    c. Publish skill
    d. Add to Collection
    e. If need be, these actions can spawn a blocking loader governed by *Pattern: blocking loader*
12. ~~An expand/collapse control that shows or hides additional information in the table row/card and reveals any truncated data~~
    a. ~~The un-truncated *keywords*, if any~~
    b. ~~A list of the skill’s higher-order BLS job codes, if any~~
    c. ~~The un-truncated BLS *detailed occupations*, if any~~
    d. ~~A list of the skill’s O*NET job codes, if any~~
    e. ~~A list of the skill’s *certifications*, if any~~
    f. ~~A list of the skill’s *employers*, if any~~
    g. ~~The skill’s *alignment* URL, if any~~
13. If a non-required field is empty, its label will also not appear in the list item.

Layout

Each skill as list item is responsive to the available width of the viewport; as the viewport becomes narrower each list item’s layout shifts from a somewhat tabular arrangement, with some data in columns and some stacked, to a fully stacked arrangement.

Pattern: collection as a list item

Design System

Collection as a List Item

Elements/data

Each collection in a list of collections contains

1. The collection's *collection name* and is also a link to the collection detail view
2. The number of skills within the collection
3. The collection's *draft* status, if applicable
4. The collection's *archived* status, if applicable
5. A checkbox that selects the collection
6. A hidden "jump to actions" link after each checkbox that goes to the sticky selected items actions bar.
7. An overflow action menu per collection – actions differ by view and by statuses of the collection. Possible actions include
    a. Archive collection
    b. Unarchive collection
    c. Publish collection
    d. If need be, these actions can spawn a blocking loader governed by *Pattern: blocking loader*

Layout

Each collection as list item is responsive to the available width of the viewport; as the viewport becomes narrower each list item's layout shifts from a somewhat tabular arrangement, with some data in columns and some stacked, to a fully stacked arrangement.

Behavior

Screen readers will read out the table column headers before each data point. For example: "Collection Name: Health Education Development, Status: Draft, Skills: 20"

Pattern: selected items actions

Views

Any list view has selectable list items and one or more actions that can be taken on selected items. These actions differ by view.

Design System

Selected Items Actions

Elements/layout

The actions appear in a sticky bar at the bottom of the view/end of the list of items. In narrow views, these are stacked. In very narrow views, the actions may also stack.

1. In general, lists of skills offer some of the following actions, depending on view
    a. Publish skills
    b. Archive skills
    c. Unarchive skills
    d. Add skills to collection
    e. Remove skills from collection
2. In general, lists of collections offer some of the following actions, depending on view
    a. Publish collections
    b. Archive collections
    c. Unarchive collections

Behavior

1. When no list items are selected, no actions are available. This is shown by disabling the controls that fire these actions.
2. When an action is initiated, the action temporarily changes to an inert, indeterminate “processing” state.
    a. If the process will take more than 500ms (a proxy measure maybe chosen for this such as the number of affected records), an indeterminate loading indicator will appear visibly over the page content and be announced via aria-live. This is detailed in *Pattern: blocking loader*
3. When one or more list items are selected, the actions are enabled. It’s possible that some of the available actions may not *seem* appropriate, such as *publish*

skill when only an already-published skill is selected. But publishing again would have no effect, so it is safe to have such a control enabled.

4. Users can select list items of different states. But since actions change a binary state, it is safe to select heterogenous items and then apply a single action that may not necessarily be appropriate for all items; for the “inappropriate” items there will be no effect. These tables demonstrate:

↓on what action→ effect ↘	**Archive skill**	**Unarchive skill**
Skills	Archive the skills, ~~not~~ confirmed, fire success toast a la “Archived 27 skills.”	~~No effect, not confirmed, fire success toast a la “The 27 skills were already not archived.”~~ Action is disabled
Archived skills	~~No effect, confirmed, fire success toast a la “The 27 skills were already archived.”~~ Action is disabled	Unarchive the skills, not confirmed, fire success toast a la “Un-archived 27 skills.”
Mix of skills and archived skills	Archive the unarchived skills, ~~not~~ confirmed, no effect on the others, fire success toast explaining overall effect	Archive the unarchived skills, not confirmed, no effect on the others, fire success toast explaining overall effect

↓on what action→ effect ↘	**Archive collection**	**Unarchive collection**
Collections	Archive the collections, not confirmed, fire success toast explaining overall effect	No effect, not confirmed, fire success toast explaining overall effect
Archived collections	~~No effect, not confirmed, fire success toast explaining overall effect~~ Action is disabled	Unarchive the collections, not confirmed, fire success toast explaining overall effect
Mix of collections and archived collections	Archive the unarchived collections, not confirmed,	Unarchive the archived collections, not confirmed,

	no effect on the others, fire success toast explaining overall effect	no effect on the others, fire success toast explaining overall effect

↓on what action→ effect ↘	**Publish skills**
Draft skills	Confirm “are you sure you want to publish”, publish the skills, fire success toast explaining overall effect
Draft and archived skills	Confirm “are you sure you want to publish” ~~while offering unarchiving of the archived skills~~, publish the skills, fire success toast explaining overall effect
Published skills	~~No effect, not confirmed, fire success toast explaining overall effect~~ Publish action disabled
Published, archived skills	~~No effect, not confirmed, fire success toast explaining overall effect~~ Publish action disabled
Mix of any of these	If any draft skills, confirm; if any draft, archived skills, confirm while offer unarchiving of those skills; publish; fire success toast explaining overall effect

↓on what action→ effect ↘	**Publish collection**
Draft collections	Confirm “are you sure you want to publish”, publish the collections, fire success toast explaining overall effect
Draft, archived collections	Confirm “are you sure you want to publish” ~~while offering to unarchive the collections~~, publish the collections, fire success toast explaining overall effect
Published collections	~~No effect, fire success toast explaining overall effect~~
Published, archived collections	~~No effect, fire success toast explaining overall effect~~
Mix of any of these	If any draft collections, confirm; if any draft, archived collections, offer to unarchive the collections; publish; fire success toast explaining overall effect

5. Some of these actions are further guarded – that is to say that we go above and beyond to confirm the action and offer alternatives. These actions are guarded:
    a. Publishing a collection that contains draft skills – the user must publish the draft skills before proceeding, and is offered the opportunity to do so
    b. Publishing a collection that contains archived skills – the user is offered the opportunity to unarchive the archived skills before proceeding, but is not required to do so

Pattern: success toast message

Toast messages are non-dismissible success notifications that appear after a user has taken an action.

Views

See Pattern: selected items actions for some situations when a toast message would appear.

Design System

Toast module and Toast layout

Behavior

After a user has taken an action and the action was successful, toast messages appear at the top middle of the viewport for a duration of 5 seconds and then disappear. A screen reader will read the toast as a passive announcement (aria-live) after page load. A toast message should be specific to the completed action, for example: “Success! {x} has been removed from {y}”.

Pattern: error message

Error messages are inline messages that can appear after the user has taken an action or when the server is experiencing a problem.

Views

Any page can have an error message displayed.

Design System

- Icon Message
- Error Message

Behavior

Error messages appear on the page after the common controls and before the page title. A screen reader will read the error message in the same order. An error message should be specific to the problem, if known, for example: “Your skill was not saved. {Remedy}” or “Oops! Something happened. Please try again.”

Pattern: paging navigation

Internal (logged in) list views (skills, collections) are paged. A page is 50 records.

Views

Paging nav appears on list views whenever the number of returned records exceeds one page.

Design System

Paging Navigation

Elements

1. A control that takes the user to the prior page of results
2. A control that takes the user to each specific page of results, numbered by page
3. A control that takes the user to the next page of results

Behavior

1. If there is only one page of results, paging navigation will not appear.
2. If a user is on the first page, the prior page control is disabled
3. If a user is on the last page, the next page control is disabled

4. The specific page control for the current page is ~~disabled and visible~~ shown as clearly active
5. Should there be more than five pages of results, the paging controls are filtered
    a. The specific page control for the first page of results is shown
    b. The specific page control for the last page of results is shown
    c. The specific page control for the current page of results is shown
    d. The two specific page controls to either side of the current page, if any
    e. Ellipses wherever one or more specific page controls are not shown via the above rules

Pattern: search-assisted entry

(aka "Search & Multi-Select" in Badgr)

Search-assisted entry helps users find and enter information in forms by returning prior or expected entries as they type and allowing them to select one or more of these entries to save with the form. It's applicable to structured and pre-determined data sets, such as job codes, as well as user-generated data sets such as keywords.

It is intended to speed data entry by making it easier to choose data that has been used before or is known to be formatted correctly, to assist with keyboard accessibility, and to assist with screen reader accessibility.

Views

Search-assisted entry is used on the create and edit skills forms. It'll initially be used to help enter job codes, but can be applied to keywords, categories, standards, certifications, and employers.

Design System

[Add example]

Major elements

1. A text field that serves as the *search field*
2. A *results list* that appears after the search field once the user has begun typing in the search field, containing search result elements if any results are returned
3. *Selected items* that appear after the search field, representing the data the user has chosen to save with the form

Behavior

See the demo at https://patternlibrary.badgr.com/examples/search-multi-select.html#page-ca2229a62f639ccbbddec37085cfd394 for a very similar (but not the same!) capability.

1. *Search field*
    a. Initially, the search field contains the placeholder text "Search"
    b. There is helper text for screen reader users above the field that explains that if they search here, results will appear as the next item after they exit the search field.
    c. If the field does not accept novel input, it includes a search icon with the screen reader caption "search"
    d. If the user types into the search field, an X control appears that allows the user to clear the contents of the field
        i. The X control is labeled "clear" for screen readers
    e. As the user types, results are searched for
        i. Each subsequent keypress refines the results.
            1. Aria-live will tell the screen reader user the results after the user stops typing.
        ii. While results are being fetched, an indefinite progress indicator is shown alongside the X control. It is removed once results are present. This is likely derivative of the spinner component of the *non-blocking loader* pattern mentioned below.
    f. From here the user may
        i. Hit enter to accept their input as a selected item and clear the search field, after which focus remains on the search field
            1. If the input is valid for the field in question, the input is accepted
            2. If the input is not valid for the field in question, a validation failure message is shown
        ii. Tab or use the down key to move focus to the search results.
            1. While focus is in the list, additional typing brings focus to the search field and is reflected there
            2. While focus is in the list, the left/right keys move the cursor in the search field
        iii. Tab away from the area entirely, leaving their input in place in the search field. This input will not be evaluated until the form is submitted

   g. The search field CANNOT accept multiple search terms at once. Anything entered there is considered a single search term.

2. *Results list*
   a. The results list is a scrolling list of entries found by searching either the body of allowable entries (as in occupations) or prior entries (as in *employers, standards, keywords*, etc.)
   b. Typing in the search field causes the results list to be made visible. The search results are not visible when the search field is empty
   c. This results list is made up of one or more of:
      i. Empty state copy if the user has begun typing but no results are found
      ii. Active entries (those that are not also members of *selected items*), which contain
         1. Hidden helper text for screen readers reading "click this entry to choose"
         2. The text of a found item
      iii. Inactive entries (those that are already members of *selected items*) are greyed out to signal that they are inactive, and contain
         1. The text of a found item
         2. A caption indicating that the item is already selected
   d. The search results can be navigated by mouse or keyboard up/down
   e. Clicking an active entry, or hitting enter while it has focus, or selecting it via screen reader, will cause
      i. the entry content to be added to the *selected items*
      ii. the entry to become inactive
      iii. an aria-live message pointing out that the selection was added
      iv. focus to remain in the search results so the user can select additional items
   f. The search results also has a hidden close control that collapses the search results, revealing the *selected items* if any
3. *Selected items*
   a. Zero or more selected items may be present
      i. In some cases, a field might be limited to zero or one selected item
   b. If one or more selected items is present, they are preceded by a hidden skip link allowing the user to skip past the selected items
   c. Each selected item shows its text and an X control
      i. The X control is labeled "remove" for screen readers
      ii. Operating the X control removes the item from the list of selected items

    d. There is no empty state text or graphic for this area, as the empty search field is a sufficient empty state signal

Variants

There are a few conditions that govern this pattern's acceptance of typed input and alter the above general behavior.

1. **The data type may or may not accept novel input**. For example, the *occupations* field cannot accept novel input, while the *keywords* field can
    a. If the data type accepts novel input, typed input that has not been seen by the system before can be accepted as a new selected item and saved for later searches
    b. If the data type does not accept novel input
        i. ~~Novel input causes a validation failure~~
        ii. The search results would display "No results found"
        iii. The search field contains a search icon, which reads as the word "search" to a screen reader
2. **The data type may or may not accept more than one item**. For example, the *employers* field can accept multiple strings but the *category* field accepts only a single string
    a. If the data type accepts only a single input, any selected item replaces the previous content of the field
    b. If the data type accepts multiple inputs, any selected item adds to the previous contents of the field

Copy

- Hidden helper text before search field: "Enter text in the Search field, then select from the suggestions that appear in the list that follows. Repeat the process to add multiple selections. Use the Escape key to close the list and reset your search terms."
- Placeholder text in search field: "Search"
- Hidden label for X control to clear search field: "Clear Search field"
- Hidden helper text in active entries: "Click to select"
- Caption indicating that the item is already selected: "Already selected"
- Aria-live message pointing out that the selection was added: "Selection added"
- Hidden label for close control on search results: "Close list"
- Hidden link to skip selected items: "Skip selected items"
- Hidden label for X control on a selected item: "Remove selection"

Pattern: non-blocking loader

A non-blocking loader appears on a page while a process occurs in the background, but does not prevent other user action. All loaders are indeterminate: They are not tied to a particular duration, percentage complete, or other measurable increment.

Views

The non-blocking loader is used in “processing” views in batch import, surrounded by other components of the page.

Design System

Non-Blocking Loader

Elements

A non-blocking loader consists of

1. a seamlessly repeating animated element
2. an optional caption explaining what is happening, detailed in the relevant acceptance criteria for the view

Behavior

The non-blocking loader appears when the user initiates the associated process and disappears when that process is complete. For some completed processes, a new page or new data will display as well.

If there is no caption, an aria-live attribute announces “processing” when the loader appears and “processing complete” when it disappears.

If there is a caption, aria-live attribute reads the caption when the loader appears and “processing complete” when it disappears.

Copy

Since the caption is optional, it will be defined on a view or story basis.

Pattern: Loading indicator on “checking similarity” messages

Elements and Views

A loading spinner, derivative of the non-blocking loader but much smaller, appears alongside interstitial “checking similarity” captions that appear in batch import and in skill edit forms.

The same loading spinner may also be used in *search-assisted entry* above.

Design System

Similarity Checker

Behavior

The loading indicator appears when the system begins checking for similar skill statements, concurrent with a caption change. It is later replaced when the caption is changed again to indicate that similarity was ok or not ok.

Copy

Copy is defined on a story basis.

Pattern: blocking loader

A blocking loader prevents further action on a page or explains stale data while a DOM replacement process occurs. All such loaders are indeterminate.

Views

Any view where the user can take an action on one or more elements, such as archiving several skills or publishing several collections, will show the blocking loader while the process completes.

Design System

Blocking Loader

Elements

The blocking loader is essentially

1. A subtler derivative of the non-blocking loader mentioned in *Pattern: non-blocking loader*, but
2. appearing on a pale scrim that covers the content to be replaced. It is important that the scrim be subtle as it may come and go rapidly and we don't want to produce a strobe effect. Its purpose is to show that the data is stale and will be replaced imminently.

It does not have a caption.

Behavior

The non-blocking loader appears when the user initiates an action and disappears when that process is complete. For some completed processes, a new page or new data will display as well. The non-blocking loader will cover the old data until the new data is available.

An aria-live attribute announces "processing" when the loader appears and "processing complete" when it disappears. **This announcement should NOT occur when changing views, for example from a list of skills to an individual skill.**

Ideally the loader will *cover rather than replace* the DOM elements it is meant to obscure so that page elements do not jump. Rather, data that will be replaced should seem to dim until fresh data appears in its stead. If this is not possible, padding values will need to be carefully chosen to try to hold page areas comfortably open to minimize yoyoing.

Pattern: loader on action button

Views

Multi-unit actions likely to take a bit may benefit from a loading state on the button itself, close to the user's gaze. This can be implemented at the developers' discretion.

Design System

Button Is Loading (A state on Button)

Elements

1. An icon-sized spinner, derivative of that for *non-blocking loader* mentioned above, and
2. a dimmed form of the button’s style

Behavior

When a person operates an action button we’d like to suggest that the system heard them and is taking action, so it seems appropriate to disable the action, give the action a dimmed appearance, and swap that action’s icon, if any, for an indeterminate loading indicator.

Neighboring actions (such as Cancel when saving a skill form,or neighboring sticky action bar actions) should also take on a disabled state.

Once the process completes, if the selected action is not replaced as part of the triggered process, we’d reverse the appearance changes.

Ideally we would not replace the entire control, but modify or augment its appearance so that page elements don’t jump around.