# Multi-Select Picklist Component

## 1. Introduction

**Purpose:** Define a reusable Lightning Web Component (LWC) that presents a searchable, multi-select picklist. The component must fetch its options dynamically via a query or fall back to a provided default list. It should allow users to select or deselect multiple items, display the current selection as pills, support “Select All” and “Clear All” actions, enforce a maximum selection limit, and broadcast selection changes to its parent.

**Scope:** This document describes functional and non-functional requirements, user interface behavior, data structures, events, error handling, and acceptance criteria. No implementation details (e.g. variable names or code structure) are included, so developers may choose their own architectural approach.

## 2. Component Overview

* **Name:** Multi-Select Picklist
* **Description:** A composite control combining a search field, a scrollable list of checkbox options, “Select All” / “Clear All” links, and a pill-style summary of selected items.
* **Use Case:** Parent components bind the picklist to a specific data field, supply a query to load values, set a title, and optionally define a maximum number of selections.

## 3. API Parameters (Public Properties)

1. **title** (string, required)
   * Label for the search box and dropdown.
2. **queryString** (string, optional)
   * Query expression used to fetch picklist values from a server. If omitted, only the default list is used.
3. **defaultOptions** (array of strings, required)
   * Fallback list of option labels to display when no query is provided or data loading fails.
4. **fieldIdentifier** (string, required)
   * Logical name of the field; used in dynamic queries and in event payloads.
5. **maxSelections** (integer, optional, default = 1000)
   * Maximum number of items a user may select. Exceeding this limit triggers a validation error.

## 4. Events

* **selectionchange**
  + **Detail Payload:**
    - selectedItems – Array of objects, each with { id, label } for the currently selected options.
    - field – The fieldIdentifier value.
  + **When Fired:** Whenever the selection set changes (add, remove, clear all, select all).

## 5. Functional Requirements

### 5.1 Data Loading

* **Initial Load:**
  1. If queryString is provided, fire a server call to retrieve records, each with a single field value.
  2. Extract the values in display order and build an internal list of option objects { id, label }.
  3. If the server call fails or returns no records, fall back immediately to defaultOptions.
* **Dynamic Filtering:**
  + On each character typed into the search box:
    - Filter the in-memory list of options by case-insensitive substring match.
    - If the in-memory list length equals the total possible values and the search term length ≥ 2, regenerate queryString to include a server-side filter by the search term, invoke the query again, and update the options list.

### 5.2 User Interface Behavior

#### 5.2.1 Search Input

* Displays title as its label.
* Placeholder displays the summary of selected items (e.g. “3 options selected” or “None Selected”).
* onfocus: Show the dropdown list.
* onblur or mouseleave after focus loss: Hide the dropdown.
* Support keyboard navigation (Up/Down arrows to move, Enter to toggle, Esc to close).

#### 5.2.2 Dropdown List

* Appears below the search input when open.
* Contains up to 100 filtered entries to preserve performance and readability.
* Each entry shows:
  + A checkbox reflecting the selected state.
  + The option label.
* Scrollable when entries exceed visible area.
* Two links at the top/right when the list is open: “Select All” and “Clear All.”

#### 5.2.3 Selection Summary (Pills)

* Rendered beneath the search input.
* One pill per selected item, showing the label and an “×” remove icon.
* Clicking the remove icon deselects the item immediately.

### 5.3 Selection Logic

* **Selecting an Item:**
  + Add it to the internal selectedItems array if under maxSelections.
  + If the limit is reached, prevent further additions and display a validation error on the search input.
* **Deselecting an Item:**
  + Remove it from selectedItems.
  + Clear any validation error if the selection count falls below maxSelections.
* **Select All:**
  + Add every currently filtered option to selectedItems, up to maxSelections.
  + If the number of filtered options exceeds the remaining capacity to maxSelections, only select as many as permitted, then show a validation error.
* **Clear All:**
  + Remove every item from selectedItems.
  + Reset validation state.

### 5.4 Error Handling & Validation

* If selectedItems.length > maxSelections, display a custom error message on the search input:  
  “Maximum of {maxSelections} items can be selected.”
* Any server call failure logs an error to the console and uses the default options.
* All UI controls must remain responsive and recover gracefully from errors.

## 6. Non-Functional Requirements

1. **Performance:**
   * Limit displayed list to 100 items.
   * Debounce user keystrokes (e.g. 200 ms) before triggering server-side filtering.
2. **Accessibility (a11y):**
   * All interactive elements must be keyboard-operable.
   * Use appropriate ARIA roles for listbox, options, and dialog semantics.
   * Provide screen-reader labels for the “Select All” and “Clear All” links.
3. **Styling:**
   * Follow Salesforce Lightning Design System (SLDS) standards for inputs, checkboxes, pills, and layout.
   * Ensure the component is responsive (mobile and desktop).
4. **Internationalization (i18n):**
   * Support translation of all static text (“Select All,” “Clear All,” error messages, “None Selected,” etc.).
5. **Extensibility:**
   * Developers must be able to override default behavior (e.g. inject a custom query processor) by extending or decorating the component.

## 7. Data Model

| Property | Type | Description |
| --- | --- | --- |
| id | string | Unique key for each option |
| label | string | Display text for the option |
| isChecked | boolean | Whether this option is checked |
| selectedItems | array of {id,label} | Current selection set |
| allOptions | array of {id,label} | Complete list before filtering |
| filteredOptions | array of {id,label} | Options matching the search term |

## 8. Acceptance Criteria

1. **Initial Display:**
   * Load and display defaultOptions when no query is provided.
   * If a query is provided, fetch data and display server results.
2. **Search Filtering:**
   * Typing in the search input filters the dropdown instantly up to 100 matches.
   * At ≥ 2 characters with full dataset loaded, the component re-queries the server for matching options.
3. **Selection:**
   * Clicking a checkbox adds/removes the corresponding pill and updates the placeholder count.
   * Exceeding maxSelections prevents new selections and shows an error.
   * Clicking a pill’s remove icon deselects that item.
4. **Select All / Clear All:**
   * “Select All” adds every currently visible option up to maxSelections.
   * “Clear All” empties the selection set and clears the error.
5. **Event Dispatch:**
   * Every change to the selection (add, remove, select all, clear all) fires selectionchange with the correct payload.
6. **Accessibility & Responsiveness:**
   * All interactions available via keyboard.
   * Layout adjusts correctly on narrow viewports.

**End of Requirements**