Absolutely—you could piece this together almost entirely with off-the-shelf plugins, but you'd trade off some of the fine-grained control, performance, and ease of future tweaks that you have with a custom solution. Here's roughly how it might look:

---

1. ACF Front-End Forms

- **Plugin(s):** Advanced Custom Fields Pro's Frontend Forms, Gravity Forms with the ACF Add-On, or Ninja Forms + ACF bridge.
- **What you get:** A way to render and save your ACF fields on the front end without writing the low-level AJAX handlers.
- **What's missing / must-code:**
    - The bulk-edit logic (iterating over multiple posts, merging values, excluding fields on clear) would likely need custom PHP snippets hooked into the form-submit process.
    - Any File/Media handling in the form—especially "remove file" checkboxes—powers you implemented manually.

2. Faceted Search & Filtering

- **Plugin(s):** Search & Filter Pro or FacetWP.
- **What you get:** Powerful UI for multi-select checkboxes, "Match Any / Match All / is None Of / is Unknown" logic, AJAX pagination, result counts, and even templating.
- **What's missing / must-code:**
    - Styling the facet controls exactly like your Materialize design—there's always a bit of CSS/JS glue to match their markup.
    - Integrating your custom "Not specified" (empty meta) download filter—FacetWP can do "exists / not exists," but you'd need to configure it or write a small hook for your three-range + blank combination.

3. Front-End Modals

- **Plugin(s):** A jQuery Modal plugin or a page-builder modal widget (Elementor, Beaver Builder, etc.).
- **What you get:** Triggered popups without hand-crafting the `<div>` markup and the `M.Modal` initializations.
- **What's missing / must-code:**
  - Tightly coupling the modal's "Save" button to your field group and doing a custom AJAX call. You'd still need a little JS or PHP hook to take the form data and call `wp_update_post()` / `update_post_meta()`.

4. Bulk-Editing

- **Plugin(s):** There are some front-end bulk-edit plugins (e.g., WP Sheet Editor, Admin Columns Pro with Frontend Editing), but they're rare.
- **What you get:** A spreadsheet-style interface without building your own table + checkboxes.
- **What's missing / must-code:**
  - Custom field groups: mapping each column in the sheet to your ACF field group.
  - Your "clear field" links and the logic to only apply changes to checked items.

What You'd Compromise

1. **Performance & Bloat:** Every plugin loads its own assets, settings screens, and queries—even for features you don't use.
2. **Tight UI/UX Control:** Your current solution matches your Materialize-based design pixel-for-pixel. A plugin's markup/CSS will require overrides.
3. **Complex Logic (e.g., "is unknown," "not specified"):** Those custom meta-query combinations often need little code snippets anyway.
4. **Future Maintainability:** If you ever want to tweak that modal's animation or the bulk-logic edge-case, a bespoke setup is faster to iterate on than wrestling with plugin

filters.

What You'd Still Have to Code

- **Meta-Query Filters:** Even Search & Filter Pro needs you to register your custom field keys and sometimes drop in a PHP filter to tweak "NOT EXISTS" behavior.
- **Modal Save Hooks:** Grabbing the front-end form data and routing it to `wp_ajax` or a Gravity Forms hook.
- **Bulk-Update Handler:** Looping through selected posts, sanitizing inputs, and calling `update_post_meta()`—that stays custom.