Stakeholders:

1. Christina Bankhead
2. Kennedy Porter
3. Josh Powell

Sprint 1:

* Dsp-294
  + Have column table that references the item table (don't actually delete the item, just remove the item from the column)
* DSP-298
  + Add new column in mongo and change in one place in the code
  + New build column after the demand column
* DSP-299 / DSP-300 creating blank page for metrics and history
* DSP-302
  + Take current date and last date that the item was moved and create an aging field next to the assigned user
  + Time it was updated (look at events array and most recent event that it was updated)
  + Any movement recorded in the events
* DSP-303
  + Function lead filter
* DSP-304
  + OCM tag filters

**Meeting with Kennedy 4/8:**

* Custom filters for bobcat export (more difficult)
  + Filter by premise, functional lead, status
* Making filters query parameters (in effect)
* Login call back functionality (unsigned in and someone sends you a filtered link it takes you to the unfiltered view) tweak login logic to take you to filtered query page
* Role based access control ticket (RBAC talk to Varun about that) need to make it scalable
* Slam dunk functionality could be confetti going from intake to build (on demand confetti button --> make it a celebration emoji) websocket triggers confetti across entire site

**What I Learned:**

* **Memoization** is an optimization technique where the result of a function is cached so that if the function is called again with the same inputs, the cached result is returned instead of recalculating it.
* **Flattening** here refers to the process of taking a complex, nested structure (like an object with nested fields or arrays) and converting it into a simpler, single-layer structure.
  + This approach is useful for implementing search functionality. Instead of searching through the original, nested [data.tasks](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) structure, you can search through the [flattenedTasks](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) object, which is simpler and optimized for keyword matching.

**Questions:**

* Should I make users fill out the additional questions form before moving to build column??? Will this cause an issue (test in dev)

**Worked on:**

* Confetti feature
  + Had to overcome double scroll issues with dragging a component across a scrollable screen because of the library
  + Decided it would be best to add a button to the item to automatically take it from intake grooming to build then shoot confetti based off that
* Deleting tasks (keeping in DB in case of recovery)
* DSP 334 - make pervasive login query parameter URL
* DSP-306 RBAC for admin deletion function <https://www.axelerant.com/blog/how-to-setup-role-based-access-control-with-next-auth>
  + Using RBAC and next-auth the key to effectively implementing RBAC authentication is to use sessions and middleware to protect routes based on this role data
  + Middleware file: create protected path for pages
  + **QUESTION FOR VARUN:** how do we set the roles for users in their sessions? Is this determined in AD group access?