Plan for the Course

Code always makes more sense when you understand what it's being used for. So, rather than just throw around some code snippets without context, we're going to walk through the process of building a sample application. That way, you'll be able to see how each feature fits in to the larger application, and how the code changes fit into the codebase.

It'll help a lot if we have some idea what kind of program we're trying to build. Since I'm the one writing the course, I get to choose what we're building :)

Inspiration for the Sample Application

I'm a big fan of the Battletech game universe, a game about big stompy robots with weapons that are used to fight wars a thousand years in the future. The game universe includes miniatures tabletop games, computer games, roleplaying games, fiction, and more.

There's a number of fan-built projects based in the Battletech universe. One of the most popular is Megamek, a cross-platform computer implementation of the Classic Battletech tabletop game. In addition, the Megamek authors have built a tool called **MekHQ**, a tool for managing an ongoing campaign for a fictional combat organization, such as a mercenary unit.

For this course, **we're going to build a miniature version of the MekHQ application**. I definitely do *not* intend to seriously rebuild all of MekHQ's functionality, but it serves as a useful inspiration and justification for most of the techniques I want to show off.

Every project needs a good name. I haven't come up with a good name yet, so we'll call this ***"Project Mini-Mek"*** .

Project Features

MekHQ is a very complex and in-depth application. The “About” page describes it this way:

> MekHQ is a Java program that allow users to manage units in-between actual games of MegaMek. It implements most of the rules in the “Maintenance, Repair, and Salvage” section of Strategic Operations, many of the rules and options from the Mercenary Field Manual, and various options that allow users to customize a mercenary, house, or clan campaign to their liking . Some of the features include:

> - Track XP and improve pilot skills and abilities
> - Integration with MegaMek and MegaMekLab
> - Planetary map and the plotting of jumpship travel
> - Organize a TO&E
> - Create and resolve missions and scenarios
> - Repair and salvage units
> - Equipment tracking
> - Financial tracking

For ***Project Mini-Mek***, we’re only going to deal with a couple of these, and at a very simple level. Here’s the list of features we’ll build in this course:

- A tabbed UI containing different panes, controlled by Redux state
- Load JSON data describing the pilots and BattleMechs in a combat force
- For both pilots and Battlemechs:
  - Show a list of all items
  - Allow selection of an item in the list, and show details of the selected item
- Edit the details for a pilot
- Deletion of pilots
- Modal dialogs that can be stacked on top of each other, and used as “pickers” for choosing values like colors

- Context menus for interactions such as commands for specific list items

We'll also look at many "under-the-hood" aspects of using React and Redux, such as ways to handle relational data in your Redux store, techniques to optimize performance for editing and form inputs, how to handle "draft data" when editing, and much more.