Implementing Pilot Deletion

We should now be able to dispatch an **ENTITY_DELETE** action to delete a given Pilot entry from the store. All we need to do now is add delete buttons to our **PilotsListRow** components, and hook them up to dispatch the action.

We'll add another column to the Pilots list, and show a red circular X button for each row. Clicking the button will delete the item.

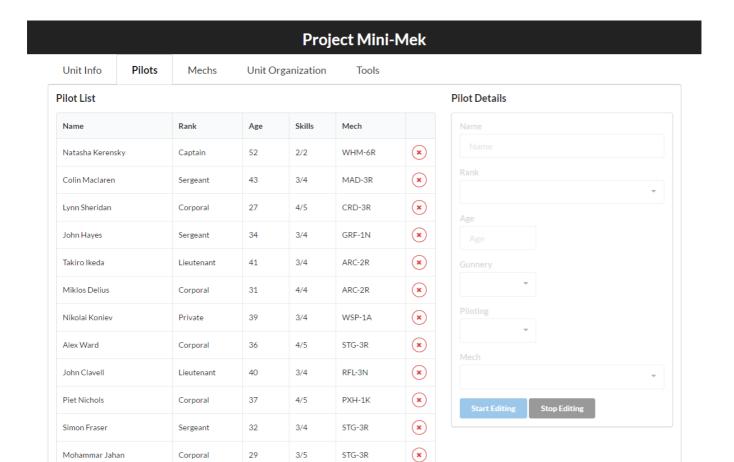
Commit 128b2ac: Add the ability to delete individual Pilot entries

features/pilots/PilotsList/PilotsListHeader.jsx

features/pilots/PilotsList/PilotsListRow.jsx

```
Entity}) => {
// Omit prop extraction
    const onDeleteClicked = () => deleteEntity("Pilot", id);
// Omit row cell rendering
            <Table.Cell>
                 {mechType}
            </Table.Cell>
             <Table.Cell>
+
                 <Button
+
+
                     compact
                     basic
+
                     circular
+
                     size="tiny"
+
                     color="red"
+
                     icon={<Icon name="delete" />}
+
                     onClick={onDeleteClicked}
+
                 </Button>
+
            </Table.Cell>
        </Table.Row>
```

That should give us a column of delete buttons for all our Pilot entries:



Clicking any of the delete buttons should remove the corresponding Pilot entry from the store, and that will result in the row being removed. If you play around with things a bit, you'll see there's also some interesting behavior around trying to delete any Pilot entry while a Pilot is being edited, whether it's the same Pilot or a different Pilot. Let's take a look at what's going on there.

Improving the Pilot Deletion Logic

Right now, *any* click on the pilots list will dispatch the PILOT_SELECT action, which also stops any active editing. Let's update the logic so that it only stops editing if the current pilot is deleted. There's probably a few different ways we could handle this. We're going to do it with two distinct changes.

First, we'll update the click handling in <PilotsListRow>. In the current code, a click on the delete button triggers the click handler for the button, but also triggers the click handler for the entire table row. We'll have the button click handler cancel the event, so that the row handler doesn't get run afterwards.

Second, we're going to have the pilots reducer listen for the **ENTITY_DELETE** action type. *If* the deleted item is a Pilot, and it's the currently selected pilot, we'll clear the selection (and stop editing too, just in case).

Commit 2ca4d62: Clear selection and stop editing if the current pilot is deleted

features/pilots/PilotsList/PilotsListRow.jsx

features/pilots/pilotsReducer.js

```
+import {
    ENTITY DELETE,
+} from "features/entities/entityConstants";
+export function stopEditingIfDeleted(state, payload) {
    const {itemType, itemID} = payload;
    const {isEditing, currentPilot} = state;
+
    if(itemType === "Pilot" && itemID === currentPilot) {
        return {
            ...state,
            isEditing : false,
            currentPilot : null,
        };
    }
+
    return state;
+}
export default createReducer(initialState, {
    [PILOT_SELECT] : selectPilot,
    [PILOT_EDIT_START] : startEditingPilot,
    [PILOT_EDIT_STOP] : stopEditingPilot,
    [ENTITY_DELETE] : stopEditingIfDeleted,
});
```

Notice that this is another example of multiple reducers in different slices responding to the same action.

Now if we edit one Pilot, and delete another, we'll stay in editing mode for the current pilot. Selecting a different pilot will still cancel editing.