By using just one form context to handle both editing and adding items, you completely avoid the conflict. There's only one form provider wrapping everything, so components never try to access a non-existent context. the Single Form Context Solution would be easier to implement if you're already using React Hook Form. You'd essentially:

- Move your form declaration to the parent component

- Wrap everything in a single FormProvider

- Have your dialog components use the same form context but operate on different parts of the form state

This would immediately resolve the error you're seeing while preserving your desired UX of allowing users to edit and add items simultaneously with real-time budget feedback.

Key Changes in the Redesign:

- Inline "Add Item" Form - Instead of having a separate modal dialog for adding items, I've integrated the add functionality directly into the main table as an inline form row. This keeps everything within a single form context.

- Simplified Workflow - The instructions now mention adjusting quantities OR adding new items in the same step, reinforcing that they're part of the same process.

- Streamlined UI - I've removed the separate "Add Item" button since adding is now integrated into the main form. This avoids the context switching that was causing the technical issues.

- Live Feedback - As users edit quantities or add new items, the Budget Usage and Remaining Balance cards would update immediately, providing the real-time feedback you want.

Implementation Notes

In this approach, you would:

Have a single React Hook Form context at the parent component level

Define a form structure that contains both existing items and a "new item" field

When a user selects an item from the dropdown and enters a quantity, add it to the items array

The table would render all existing items plus the "add new" row at the bottom

This solves your technical issues because:

There's only one form context

All form operations happen within that single context

No competing contexts means no null useFormContext() errors

The user still gets to simulate different budget scenarios by adjusting quantities and adding items while seeing immediate feedback on their budget status.