Implementation Plan: Unified Budget Management Form

Overview

Create a single form context solution for the "Budget Items and Usage" section that:

* Replaces the current implementation
* Maintains existing budget validation rules (under/over budget)
* Preserves all current calculations
* Affects only the "Budget Items and Usage" component
* Follows updated development guidelines

Files to Create/Modify

1. New Files

* client/src/components/budget/UnifiedBudgetManager.tsx (main component)
* client/src/components/budget/BudgetItemRow.tsx (subcomponent for item rows)
* client/src/components/budget/BudgetCatalogSelector.tsx (item selector component)
* client/src/components/budget/BudgetValidation.tsx (validation helper component)

2. Files to Modify

* client/src/components/budget/BudgetPlanDetailsIntegrated.tsx (replace implementation)

Implementation Steps

Step 1: Create Form Schema

Define a comprehensive schema for the unified budget form:

// Schema for the unified budget form

export const unifiedBudgetFormSchema = z.object({

items: z.array(z.object({

id: z.number().optional(), // Optional for new items being added

itemCode: z.string(),

description: z.string(),

quantity: z.number().min(0),

unitPrice: z.number().min(0),

total: z.number(), // Calculated field

isNew: z.boolean().optional() // Flag to track newly added items

})),

// Meta fields for validation

totalBudget: z.number(),

totalAllocated: z.number(),

remainingBudget: z.number()

});

export type UnifiedBudgetFormValues = z.infer<typeof unifiedBudgetFormSchema>;

Step 2: Create the UnifiedBudgetManager Component

The main component will:

1. Load existing budget items
2. Set up a single form context with useFieldArray
3. Handle all budget calculations
4. Provide unified validation
5. Manage adding/editing/removing items

// UnifiedBudgetManager.tsx (core implementation)

export function UnifiedBudgetManager({ plan, budgetItems, catalogItems }) {

// Single form setup with useFieldArray

// Validation logic

// Add/remove/edit functionality

// Submit handler with mutation

}

Step 3: Create Supporting Components

BudgetItemRow Component

Handles individual item display and editing.

BudgetCatalogSelector Component

Provides catalog item selection without introducing a new form context.

BudgetValidation Component

Shows validation messages and budget status.

Step 4: Update BudgetPlanDetailsIntegrated

Replace current implementation with new unified approach.

Detailed Implementation Plan

1. Form Structure

* Use useForm and useFieldArray in a single parent component
* Pre-populate with existing budget items
* Add computed fields for budget totals and validation

2. Item Management

* Existing items are loaded into the form's initial state
* Edit functionality modifies items directly in the form array
* Adding items appends to the form array with a temporary "isNew" flag
* Deleting items removes them from the array (with confirmation for existing items)

3. Catalog Selection

* Replace modal dialog with an inline catalog selector
* Use a dropdown or popover component (not a separate Dialog)
* Selection directly adds to the form array

4. Validation Logic

* Maintain $375 budget constraint
* Real-time validation as quantities change
* Visual indicators for over/under budget status

5. Submission Process

* Batch all changes (edits, additions, deletions) into a single transaction
* Differentiate between updates to existing items and creation of new items
* Use TanStack Query mutations with optimistic updates

6. User Experience

* Clear edit mode indicators
* Inline adding capability
* Real-time budget tracking
* Confirmation for potentially destructive actions

Code Structure

UnifiedBudgetManager Component

export function UnifiedBudgetManager({

plan,

budgetItems: initialBudgetItems,

catalogItems

}) {

// Form setup

const form = useForm<UnifiedBudgetFormValues>({

resolver: zodResolver(unifiedBudgetFormSchema),

defaultValues: {

items: initialBudgetItems.map(item => ({

id: item.id,

itemCode: item.itemCode,

description: item.description,

quantity: item.quantity,

unitPrice: item.unitPrice,

total: item.unitPrice \* item.quantity

})),

totalBudget: 375, // Fixed budget amount

totalAllocated: calculateTotal(initialBudgetItems),

remainingBudget: 375 - calculateTotal(initialBudgetItems)

}

});

// Field array for dynamic items management

const { fields, append, remove, update } = useFieldArray({

name: "items",

control: form.control

});

// Watch values for real-time calculation

const items = form.watch("items");

const totalAllocated = useMemo(() =>

items.reduce((sum, item) => sum + (item.quantity \* item.unitPrice), 0),

[items]

);

// Rest of implementation...

}

Implementation Timeline

Phase 1: Core Structure

1. Create the main component with form setup
2. Implement read-only view of existing items
3. Set up basic calculations

Phase 2: Editing Capability

1. Enable quantity editing for existing items
2. Implement delete functionality
3. Add real-time validation

Phase 3: Adding Items

1. Create the catalog selector component
2. Implement item addition flow
3. Ensure budget constraints are respected

Phase 4: Submission & Integration

1. Implement the submission logic
2. Connect with API mutations
3. Replace the component in BudgetPlanDetailsIntegrated
4. Thorough testing of all scenarios

Testing Scenarios

1. Loading existing budget items correctly
2. Editing quantities and seeing totals update
3. Adding new items and validating budget constraints
4. Deleting items and confirming budget updates
5. Handling edge cases (max budget, zero quantities)
6. Validating the complete submission flow

Success Criteria

* Single form context managing all budget operations
* No form context conflicts/errors
* Maintaining all existing validation rules
* Fully functional add/edit/delete operations
* User-friendly interface with clear feedback
* Successful API integration for persistence

This plan provides a comprehensive approach to implementing the unified budget management solution while ensuring we maintain all existing functionality and focus only on the "Budget Items and Usage" section.