󰀀 PHASE 1B: CUSTOMIZE COLUMNS IMPLEMENTATION PROMPT

Branch: `feature/phase-1b-customize-columns`
Timeline: 1 day
Status: Ready for Implementation
Depends on: Phase 1A Complete

󰀀 MISSION STATEMENT

Implement the Customize Columns feature for the Transaction Table.
Add column visibility toggle, column reordering via drag-drop, and persist preferences to localStorage.
Integrate customize icon (≡) to Action column header + slide form from LEFT side (consistent with Phase 1A design).

󰀀 PHASE 1B DELIVERABLES

Task 1: Create Customize Columns Hook

- [ ] Create `useColumnPreferences.ts` hook
- [ ] Store column visibility state (localStorage)
- [ ] Store column order state (localStorage)
- [ ] Provide methods: toggleColumn, reorderColumn, reset

Task 2: Create Customize Columns Slide (LEFT SIDE)

- [ ] Create `CustomizeColumnsSlide.tsx` component
- [ ] Slide opens from LEFT (not right, for consistency with current design)
- [ ] Display all available columns with toggle switches
- [ ] Support drag-drop column reordering
- [ ] Show column preview with changes
- [ ] Persist on Save button click

Task 3: Add Customize Icon to Action Column Header

- [ ] Add customize icon (≡) to Action column header
- [ ] Click icon → Opens CustomizeColumnsSlide
- [ ] Icon position: Next to "Action" text (not replacing)
- [ ] Icon style: Consistent with Pen/Duplicate/Trash icons

Task 4: Update Transaction Table

- [ ] Implement dynamic column rendering
- [ ] Use column order from useColumnPreferences
- [ ] Hide/show columns based on visibility state
- [ ] Update header with dynamic columns
- [ ] Ensure Action column always visible (cannot hide)

Task 5: Integration & Testing

- [ ] Hook up customize slide state to TransactionTable
- [ ] Test toggle column visibility
- [ ] Test drag-drop reordering (persist to localStorage)
- [ ] Test reset to default
- [ ] Verify no TypeScript errors
- [ ] Test localStorage persistence across page refresh

CODE STRUCTURE

```
src/
├─ components/
│ ├─ Transaction/
│ │ ├─ TransactionTable.tsx (MODIFY: dynamic columns)
│ │ ├─ TransactionSlides/
│ │ │ ├─ ... (Phase 1A components)
│ │ │ └─ CustomizeColumnsSlide.tsx (NEW)
│ │ └─ UI/
│ │ └─ DragDropList.tsx (NEW: for reordering)
│ └─ Icons/
│ └─ CustomizeIcon.tsx (use existing if available)
└─ hooks/
└─ useColumnPreferences.ts (NEW)
```

SLIDE SPECS (LEFT SIDE SLIDE)

Customize Columns Slide

```
┌─────────────────────────────────────┐
│ × Customize Columns [Action Btn] │
├─────────────────────────────────────┤
│ │
│ Available Columns: │
│ │
│ [✓] Date │
│ [✓] Amount │
│ [✓] Category │
│ [✓] Account │
│ [ ] Note │
│ [ ] Tags │
│ [✓] Action (locked) │
```

│ │
│ Reorder (Drag-Drop): │
│ ⟨ Date ⟩ │
│ ⟨ Amount ⟩ │
│ ⟨ Category ⟩ │
│ ⟨ Account ⟩ │
│ ⟨ Action ⟩ (locked) │
│ │
│ [Reset] [Cancel] [Save] │
│ │
└──┘

Features:

- Toggle switches for column visibility
- Action column locked (cannot hide/reorder)
- Drag-drop reordering of visible columns
- Real-time preview of column order
- Reset button → Default state
- Cancel button → Discard changes
- Save button → Persist to localStorage + Close slide

CODE TEMPLATES

Template 1: useColumnPreferences Hook

// src/hooks/useColumnPreferences.ts

import { useEffect, useState } from 'react';

export interface ColumnConfig {
key: string;
label: string;
visible: boolean;
order: number;
locked?: boolean;
}

const DEFAULT_COLUMNS: ColumnConfig[] = [
{ key: 'date', label: 'Date', visible: true, order: 0 },
{ key: 'amount', label: 'Amount', visible: true, order: 1 },
{ key: 'category', label: 'Category', visible: true, order: 2 },
{ key: 'account', label: 'Account', visible: true, order: 3 },
{ key: 'note', label: 'Note', visible: false, order: 4 },
{ key: 'tags', label: 'Tags', visible: false, order: 5 },
{ key: 'action', label: 'Action', visible: true, order: 6, locked: true },
];

const STORAGE_KEY = 'transaction_column_preferences';

export function useColumnPreferences() {
const [columns, setColumns] = useState<ColumnConfig[]>(DEFAULT_COLUMNS);

```
const [isLoaded, setIsLoaded] = useState(false);

// Load from localStorage on mount
useEffect(() => {
const saved = localStorage.getItem(STORAGE_KEY);
if (saved) {
try {
const parsed = JSON.parse(saved);
setColumns(parsed);
} catch (error) {
console.error('Failed to parse column preferences:', error);
setColumns(DEFAULT_COLUMNS);
}
} else {
setColumns(DEFAULT_COLUMNS);
}
setIsLoaded(true);
}, []);

const toggleColumn = (key: string) => {
setColumns((prev) =>
prev.map((col) =>
col.key === key && !col.locked
? { ...col, visible: !col.visible }
: col
)
);
};

const reorderColumns = (newOrder: ColumnConfig[]) => {
setColumns(newOrder);
};

const savePreferences = () => {
localStorage.setItem(STORAGE_KEY, JSON.stringify(columns));
};

const resetPreferences = () => {
setColumns(DEFAULT_COLUMNS);
localStorage.removeItem(STORAGE_KEY);
};

const getVisibleColumns = () => {
return columns
.filter((col) => col.visible)
.sort((a, b) => a.order - b.order);
};

return {
columns,
isLoaded,
toggleColumn,
reorderColumns,
savePreferences,
resetPreferences,
getVisibleColumns,
};
}
```

Template 2: CustomizeColumnsSlide Component

```
// src/components/Transaction/TransactionSlides/CustomizeColumnsSlide.tsx

import React, { useState } from 'react';
import type { ColumnConfig } from '@/hooks/useColumnPreferences';
import { DragDropList } from '../UI/DragDropList';

interface CustomizeColumnsSlideProps {
columns: ColumnConfig[];
isOpen: boolean;
onToggle: (key: string) => void;
onReorder: (newOrder: ColumnConfig[]) => void;
onSave: () => void;
onCancel: () => void;
onReset: () => void;
}

export function CustomizeColumnsSlide({
columns,
isOpen,
onToggle,
onReorder,
onSave,
onCancel,
onReset,
}: CustomizeColumnsSlideProps) {
const [localColumns, setLocalColumns] = useState(columns);

const handleToggle = (key: string) => {
const updated = localColumns.map((col) =>
col.key === key && !col.locked
? { ...col, visible: !col.visible }
: col
);
setLocalColumns(updated);
onToggle(key);
};

const handleReorder = (newOrder: ColumnConfig[]) => {
setLocalColumns(newOrder);
onReorder(newOrder);
};
```

const handleSave = () => {
onSave();
};

const handleCancel = () => {
setLocalColumns(columns);
onCancel();
};

const handleReset = () => {
onReset();
};

if (!isOpen) return null;

return (
<div className="slide-overlay">
<div className="slide slide-from-left">
{/* Header */}

Customize Columns

×

```
{/* Body */}
<div className="slide-body">
  {/* Toggle Switches */}
  <div className="column-toggles">
    <h3>Column Visibility</h3>
    {localColumns.map((col) => (
      <div key={col.key} className="toggle-row">
        <label>
          <input
            type="checkbox"
            checked={col.visible}
            onChange={() => handleToggle(col.key)}
            disabled={col.locked}
          />
          {col.label}
        </label>
        {col.locked && <span className="lock-badge">Locked</span>}
      </div>
    ))}
  </div>
```

```
      {/* Drag-Drop Reordering */}
      <div className="column-reorder">
        <h3>Column Order</h3>
        <DragDropList
          items={localColumns}
          onReorder={handleReorder}
        />
      </div>
    </div>

    {/* Footer */}
    <div className="slide-footer">
      <button
        className="btn btn-secondary"
        onClick={handleReset}
      >
        Reset
      </button>
      <div className="button-group">
        <button
          className="btn btn-secondary"
          onClick={handleCancel}
        >
          Cancel
        </button>
        <button
          className="btn btn-primary"
          onClick={handleSave}
        >
          Save
        </button>
      </div>
    </div>
  </div>
</div>
```

);
}

Template 3: DragDropList Component

// src/components/Transaction/UI/DragDropList.tsx

import React, { useState } from 'react';
import type { ColumnConfig } from '@/hooks/useColumnPreferences';

interface DragDropListProps {
items: ColumnConfig[];
onReorder: (newOrder: ColumnConfig[]) => void;
}

export function DragDropList({ items, onReorder }: DragDropListProps) {
const [draggedIndex, setDraggedIndex] = useState<number | null>(null);
const [order, setOrder] = useState(items);

const handleDragStart = (index: number) => {
setDraggedIndex(index);
};

const handleDragOver = (e: React.DragEvent<HTMLDivElement>) => {
e.preventDefault();
};

const handleDrop = (dropIndex: number) => {
if (draggedIndex === null) return;

```
const newOrder = [...order];
const [draggedItem] = newOrder.splice(draggedIndex, 1);
newOrder.splice(dropIndex, 0, draggedItem);

// Update order numbers
const reorderedItems = newOrder.map((item, idx) => ({
  ...item,
  order: idx,
}));

setOrder(reorderedItems);
onReorder(reorderedItems);
setDraggedIndex(null);
```

};

```
const handleDragEnd = () => {
setDraggedIndex(null);
};

return (

{order.map((item, index) => (
<div
key={item.key}
draggable={!item.locked}
onDragStart={() => handleDragStart(index)}
onDragOver={handleDragOver}
onDrop={() => handleDrop(index)}
onDragEnd={handleDragEnd}
className={drag-item ${item.locked ? 'locked' : ''} ${ draggedIndex === index ? 'dragging' : ''
}}
>
⟨ ⟩ {item.label}
{item.locked && ▯}

))}
</div>
);
}
```

Template 4: Updated TransactionTable Integration

```
// src/components/Transaction/TransactionTable.tsx (MODIFY)

import { useColumnPreferences } from '@/hooks/useColumnPreferences';
import { CustomizeColumnsSlide } from './TransactionSlides/CustomizeColumnsSlide';
import { useTransactionSlideState } from '@/hooks/useTransactionSlideState';

export function TransactionTable() {
const {
columns,
isLoaded,
toggleColumn,
reorderColumns,
savePreferences,
resetPreferences,
getVisibleColumns,
} = useColumnPreferences();

const {
state: slideState,
openAddSlide,
closeSlide,
} = useTransactionSlideState();

const [customizeOpen, setCustomizeOpen] = React.useState(false);
const handleCustomizeOpen = () => {
setCustomizeOpen(true);
};

const handleCustomizeClose = () => {
setCustomizeOpen(false);
};

const handleCustomizeSave = () => {
savePreferences();
handleCustomizeClose();
};

const handleReset = () => {
resetPreferences();
};

if (!isLoaded) return
Loading...
;

const visibleColumns = getVisibleColumns();

return (
<>

{/* Table /}

{visibleColumns.map((col) => (

))}

{/ Render rows with dynamic columns /}
{transactions.map((txn) => (

{visibleColumns.map((col) => (
<td key={${txn.id}-${col.key}}>
{col.key === 'action' ? (

{/ Pen, Duplicate, Trash icons */}

) : (
renderCell(txn, col.key)
)}
</td>
))}
```

))}

{col.label}
{col.key === 'action' && (
 	
≡
 	
)}

```
  {/* Customize Columns Slide */}
  <CustomizeColumnsSlide
    columns={columns}
    isOpen={customizeOpen}
    onToggle={toggleColumn}
    onReorder={reorderColumns}
    onSave={handleCustomizeSave}
    onCancel={handleCustomizeClose}
    onReset={handleReset}
  />

  {/* Other slides from Phase 1A */}
  {/* ... */}
</>
```

);
}
function renderCell(txn: Transaction, key: string) {
switch (key) {
case 'date':
return new Date(txn.date).toLocaleDateString();
case 'amount':
return `$${txn.amount.toFixed(2)}`;
case 'category':
return txn.category;
case 'account':
return txn.account;
case 'note':
return txn.note || '-';
case 'tags':
return txn.tags?.join(', ') || '-';
default:
return '-';

}
}

□ TEST CASES

Test Suite: Customize Columns

✓ Customize icon visible in Action column header
✓ Click icon opens CustomizeColumnsSlide from LEFT
✓ All columns shown with toggle switches
✓ Action column marked as locked (no toggle)
✓ Toggle visibility → Column disappears from table
✓ Drag-drop reorders columns
✓ Dragging locked column does nothing
✓ Reset button → Default state
✓ Cancel button → Discard changes (undo)
✓ Save button → Persist to localStorage
✓ Page refresh → Column preferences persist
✓ Multiple toggles work correctly
✓ Column order persists across refreshes
✓ Visible columns rendered in correct order
✓ No console errors

□ DATA FLOW

Customize Columns Flow

User clicks customize icon (≡)
↓
OpenCustomizeColumnsSlide (LEFT side)
↓
Display all columns with:
• Toggle switches
• Drag-drop list
• Current preferences from localStorage
↓
User toggles column visibility
↓
Update local state + preview in table
↓
User reorders columns (drag-drop)
↓
Update order, preview in table
↓
User clicks "Save"
↓
Save preferences to localStorage
↓
Close slide
↓

Table updates with new columns/order
↓
User clicks "Reset"
↓
Reset to DEFAULT_COLUMNS
↓
Clear localStorage
↓
Reload table with defaults
↓
User clicks "Cancel"
↓
Discard changes (undo to original)
↓
Close slide without saving

IMPLEMENTATION CHECKLIST

Step 1: Create Hook

- *[ ] Create useColumnPreferences.ts*
- *[ ] Define DEFAULT_COLUMNS array*
- *[ ] Implement localStorage load/save*
- *[ ] Export all methods: toggle, reorder, save, reset, getVisibleColumns*

Step 2: Create Slide Component

- *[ ] Create CustomizeColumnsSlide.tsx*
- *[ ] Add toggle switch UI for each column*
- *[ ] Mark locked columns (Action)*
- *[ ] Add Reset, Cancel, Save buttons*
- *[ ] Test slide opens/closes*

Step 3: Create DragDropList

- *[ ] Create DragDropList.tsx component*
- *[ ] Implement drag-drop handlers*
- *[ ] Lock Action column from reordering*
- *[ ] Update order numbers on drop*
- *[ ] Test drag-drop works*

Step 4: Update TransactionTable

- *[ ] Import useColumnPreferences hook*
- *[ ] Import CustomizeColumnsSlide component*
- *[ ] Use getVisibleColumns() for rendering*
- *[ ] Add customize icon to Action header*
- *[ ] Wire icon click to open slide*
- *[ ] Update table to render dynamic columns*
- *[ ] Test column visibility toggle*

Step 5: Styling & Polish

- *[ ] Add CSS for slide-from-left animation*
- *[ ] Style toggle switches*
- *[ ] Style drag-drop list items*
- *[ ] Style customize icon button*
- *[ ] Test responsive design*

Step 6: Testing

- *[ ] Test toggle column visibility*
- *[ ] Test drag-drop reordering*
- *[ ] Test localStorage persistence*
- *[ ] Test reset to default*
- *[ ] Test multiple toggles*
- *[ ] Test page refresh (persistence)*
- *[ ] Check TypeScript errors*
- *[ ] Check console for warnings*

CSS HINTS (Add to your stylesheet)

```
/* Slide from left animation */
.slide-from-left {
animation: slideInFromLeft 300ms ease-out;
}
@keyframes slideInFromLeft {
from {
transform: translateX(-100%);
opacity: 0;
}
to {
transform: translateX(0);
opacity: 1;
}
}
/* Toggle row styling */
.toggle-row {
display: flex;
align-items: center;
justify-content: space-between;
padding: 12px 0;
border-bottom: 1px solid var(--color-border);
}
.toggle-row label {
display: flex;
align-items: center;
gap: 12px;
cursor: pointer;
}
.toggle-row input[type="checkbox"]:disabled {
cursor: not-allowed;
```

```
opacity: 0.5;
}
.lock-badge {
font-size: 12px;
color: var(--color-text-muted);
background: var(--color-bg-muted);
padding: 4px 8px;
border-radius: 4px;
}
/* Drag-drop list */
.drag-drop-list {
display: flex;
flex-direction: column;
gap: 8px;
}
.drag-item {
display: flex;
align-items: center;
gap: 12px;
padding: 12px;
background: var(--color-bg-surface);
border: 1px solid var(--color-border);
border-radius: 6px;
cursor: move;
transition: all 200ms ease;
}
.drag-item.locked {
cursor: not-allowed;
opacity: 0.6;
}
.drag-item.dragging {
opacity: 0.5;
transform: scale(0.95);
}
.drag-item:hover:not(.locked) {
background: var(--color-bg-muted);
border-color: var(--color-primary);
}
.drag-handle {
color: var(--color-text-muted);
user-select: none;
}
.column-name {
flex: 1;
font-weight: 500;
}
.lock-icon {
font-size: 12px;
}
/* Customize icon button */
.customize-icon-btn {
```

```
background: none;
border: none;
padding: 4px 8px;
cursor: pointer;
color: var(--color-text-secondary);
font-size: 18px;
line-height: 1;
transition: color 200ms ease;
}
.customize-icon-btn:hover {
color: var(--color-primary);
}
/* Slide footer buttons */
.slide-footer {
display: flex;
gap: 12px;
padding: 16px;
border-top: 1px solid var(--color-border);
}
.button-group {
display: flex;
gap: 12px;
margin-left: auto;
}
```

❓FAQ & TROUBLESHOOTING

Q: Drag-drop not working?

A: Ensure DragDropList component is imported correctly
Check draggable={!item.locked} condition
Verify handleDrop updates state
Log draggedIndex to debug

Q: localStorage not persisting?

A: Check STORAGE_KEY is consistent
Verify savePreferences() called on Save button
Check browser localStorage is enabled
Test JSON.stringify/parse in console

Q: Customize icon not appearing?

A: Check customize icon exists in Action column header
Verify icon styling (CSS)
Check onClick handler is wired
Inspect HTML in DevTools

Q: Columns not reordering?

A: Check order numbers updated correctly
Verify reorderColumns() called
Check component re-renders
Log newOrder in handleDrop

Q: Action column hidden?

A: Ensure action column has locked: true
Check toggle handler respects locked flag
Verify filter in getVisibleColumns()

IMPLEMENTATION TIMELINE

Hour 1: Setup

- *[ ] Create useColumnPreferences hook*
- *[ ] Create CustomizeColumnsSlide component*
- *[ ] Create DragDropList component*

Hour 2: Integration

- *[ ] Add customize icon to Action header*
- *[ ] Wire icon click to open slide*
- *[ ] Update TransactionTable for dynamic columns*

Hour 3: Testing & Polish

- *[ ] Test all functionality*
- *[ ] Add CSS animations and styling*
- *[ ] Test localStorage persistence*
- *[ ] Fix bugs and TypeScript errors*

QUICK REFERENCE

***Default Columns**:*
- *Date (visible)*
- *Amount (visible)*
- *Category (visible)*
- *Account (visible)*
- *Note (hidden)*
- *Tags (hidden)*
- *Action (visible, locked)*

***localStorage Key**: `transaction_column_preferences`*
***Locked Column**: Action (cannot hide or reorder)*
***Slide Direction**: LEFT (for consistency)*
***Button Layout**: [Reset] ... [Cancel] [Save]*

End of Phase 1B Prompt
Created: January 18, 2026

Status: Ready for Implementation
Dependency: Phase 1A Complete