Customizable Dashboard Layout System Specification

Overview

The Financial Playbook application features a **fully customizable, drag-and-drop dashboard** that allows users to personalize their workspace by repositioning, resizing, and configuring widgets to match their workflow preferences.

This system provides elasticity and flexibility, ensuring that each user can create a dashboard layout that works best for their specific needs.

Core Features

1. Drag-and-Drop Repositioning

- Users can click and drag any widget to a new position on the dashboard
- Visual grid guides appear during drag operations
- Widgets snap to grid positions for clean alignment
- Other widgets automatically reflow to accommodate the moved widget
- Real-time preview shows where the widget will land

2. Resizable Widgets

- Users can resize widgets by dragging corner or edge handles
- Minimum and maximum size constraints prevent widgets from becoming unusable
- Widget content adapts responsively to new dimensions
- Charts and graphs automatically rescale to fit new widget size

3. Layout Persistence

- User’ s custom layout is automatically saved to the database
- Layout persists across sessions and devices
- Each user has their own unique layout (single-user for now, multi-user ready)

4. Widget Library

- Users can add new widgets from a widget catalog
- Users can remove widgets they don’ t need
- Widget visibility can be toggled on/off without deleting

5. Layout Presets

- Pre-built layout templates for different use cases
- “Executive View” - High-level summary with large charts
- “Detailed Analysis” - Multiple small widgets with granular data
- “Monitoring Mode” - Focus on alerts and real-time status
- Users can save their own custom layouts as presets

Technical Implementation

Frontend Library

Recommended: React Grid Layout (react-grid-layout)

- Industry-standard drag-and-drop grid system
- Responsive and mobile-friendly
- Supports resizing and repositioning
- Lightweight and performant

Alternative: React DnD Kit (dnd-kit)

- Modern, accessible drag-and-drop

- More flexible but requires more custom implementation

Grid System

- **Grid Columns:** 12 columns (standard responsive grid)
- **Row Height:** 60px base unit
- **Gap:** 16px between widgets
- **Breakpoints:**
  - Desktop: 12 columns (⩾1200px)
  - Tablet: 6 columns (768px-1199px)
  - Mobile: 1 column (<768px)

Database Schema

```
// User layout preferences table
export const userLayouts = mysqlTable("user_layouts", {
  id: int("id").autoincrement().primaryKey(),
  userId: int("user_id").notNull(), // Foreign key to users table
  layoutName: varchar("layout_name", { length: 100 }).default("Default"),
  isActive: boolean("is_active").default(true),
  layoutConfig: json("layout_config").notNull(), // Store grid positions
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().onUpdateNow().notNull(),
});

// Layout config JSON structure
interface LayoutConfig {
  widgets: Array<{
    id: string;           // Unique widget identifier (e.g., "alerts",
"cashflow")
    x: number;            // Grid column position (0-11)
    y: number;            // Grid row position (0-∞)
    w: number;            // Width in grid columns (1-12)
    h: number;            // Height in grid rows (1-∞)
    minW?: number;        // Minimum width
    minH?: number;        // Minimum height
    maxW?: number;        // Maximum width
    maxH?: number;        // Maximum height
    isVisible: boolean;   // Show/hide widget
    config?: object;      // Widget-specific configuration
  }>;
}
```

React Component Structure

```
// Main Dashboard Component
const Dashboard: React.FC = () => {
  const [layout, setLayout] = useState<LayoutConfig>();
  const { data: savedLayout } = trpc.dashboard.getLayout.useQuery();
  const saveLayout = trpc.dashboard.saveLayout.useMutation();

  const handleLayoutChange = (newLayout: Layout[]) => {
    // Update local state
    setLayout(newLayout);
    // Debounce save to database
    debouncedSave(newLayout);
  };

  return (
    <GridLayout
      layout={layout}
      onLayoutChange={handleLayoutChange}
      cols={12}
      rowHeight={60}
      width={1200}
      isDraggable={!isLayoutLocked}
      isResizable={!isLayoutLocked}
    >
      {layout.widgets.map(widget => (
        <div key={widget.id} data-grid={widget}>
          <Widget type={widget.id} config={widget.config} />
        </div>
      ))}
    </GridLayout>
  );
};
```

Available Dashboard Widgets

1. Active Alerts Widget [QC: 209-211]

Default Size: 12 columns × 2 rows
Min Size: 6 columns × 1 row

Max Size: 12 columns × 4 rows
Content: List of current alerts with status indicators

2. Cash Flow Chart Widget [QC: 213]

Default Size: 12 columns × 4 rows
Min Size: 6 columns × 3 rows
Max Size: 12 columns × 8 rows
Content: Interactive line chart showing balance over time

3. Quick Stats Widget [QC: 214-217]

Default Size: 12 columns × 2 rows
Min Size: 4 columns × 2 rows
Max Size: 12 columns × 3 rows
Content: 3 stat cards (Total Balance, Monthly Burn, AR Outstanding)

4. Upcoming Bills Widget

Default Size: 6 columns × 3 rows
Min Size: 4 columns × 2 rows
Max Size: 12 columns × 6 rows
Content: List of bills due in next 30 days with status

5. Account Balances Widget

Default Size: 6 columns × 3 rows
Min Size: 4 columns × 2 rows
Max Size: 12 columns × 6 rows
Content: Current balances for all connected accounts

6. Recent Activity Widget

Default Size: 6 columns × 4 rows
Min Size: 4 columns × 3 rows
Max Size: 12 columns × 8 rows
Content: Recent transactions and changes from QuickBooks

7. Scenario Status Widget

Default Size: 6 columns × 2 rows
Min Size: 4 columns × 2 rows
Max Size: 12 columns × 4 rows
Content: Status of active scenarios with quick actions

8. Calendar Preview Widget

Default Size: 6 columns × 4 rows
Min Size: 4 columns × 3 rows
Max Size: 12 columns × 6 rows
Content: Mini calendar showing upcoming events

9. QuickBooks Sync Status Widget

Default Size: 4 columns × 2 rows
Min Size: 3 columns × 2 rows
Max Size: 6 columns × 3 rows
Content: Connection status, last sync time, sync button

User Interface Controls

Dashboard Toolbar [QC: 230-240]

Located at the top of the dashboard page:

```
┌──────────────────────────────────────────────────────────────┐
│ [230] Dashboard Layout                                       │
│ [231] [🔓 Unlock Layout] [232] [+ Add Widget] [233] [⚙️ Layout Settings]
│
└──────────────────────────────────────────────────────────────┘
```

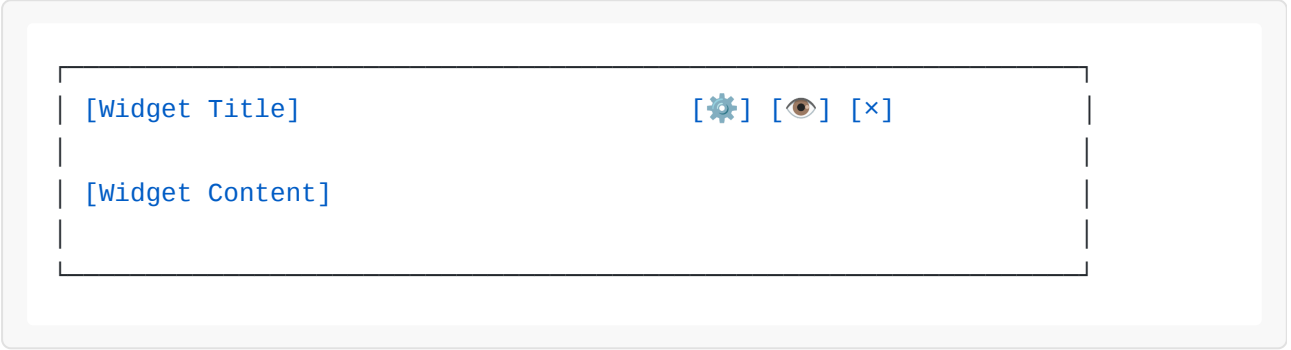
Controls:

- [231] **Lock/Unlock Toggle:** Prevents accidental changes when locked

- [232] **Add Widget Button:** Opens widget library modal
- [233] **Layout Settings:** Opens layout configuration panel

Widget Controls

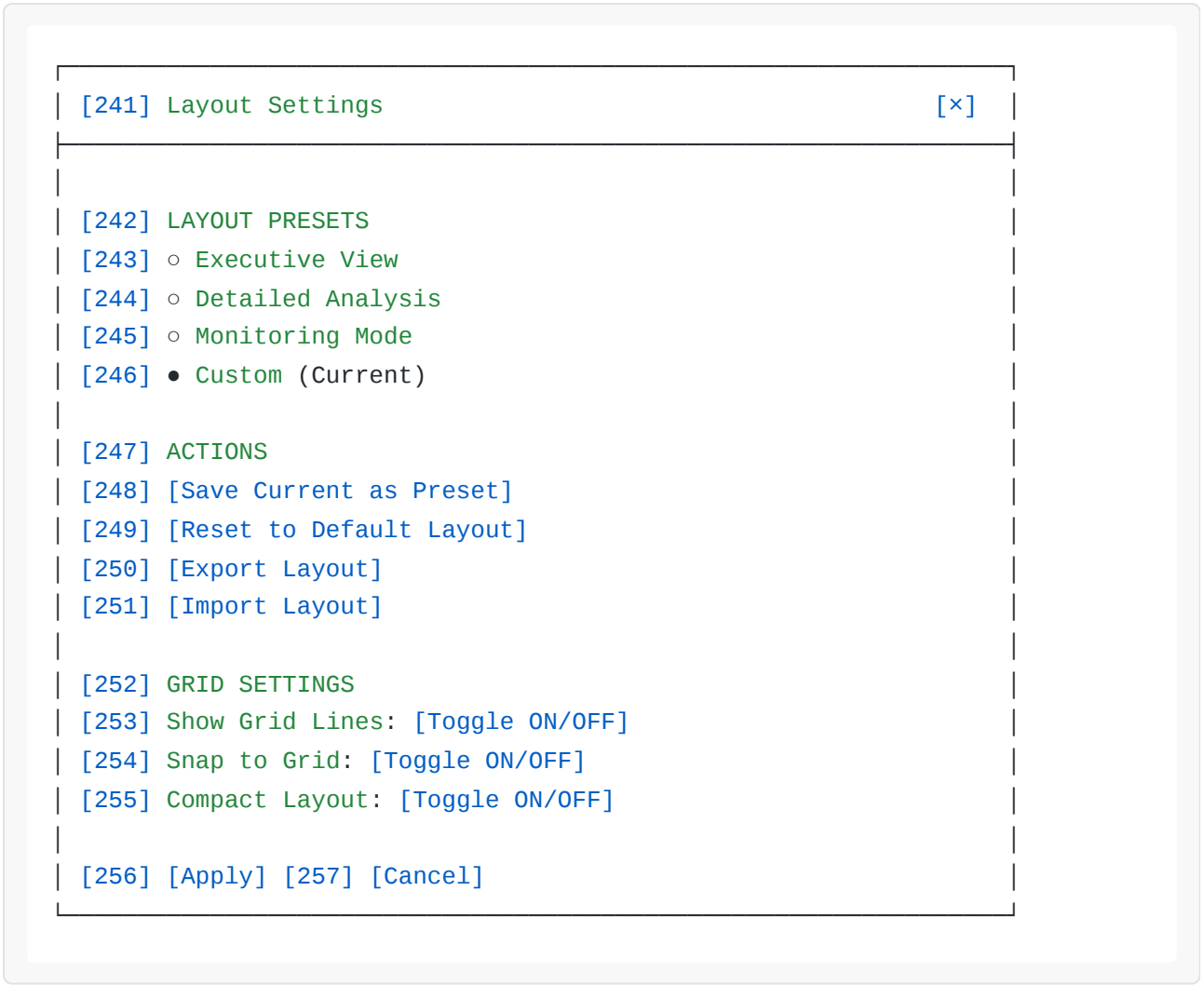
Each widget has a header with controls:



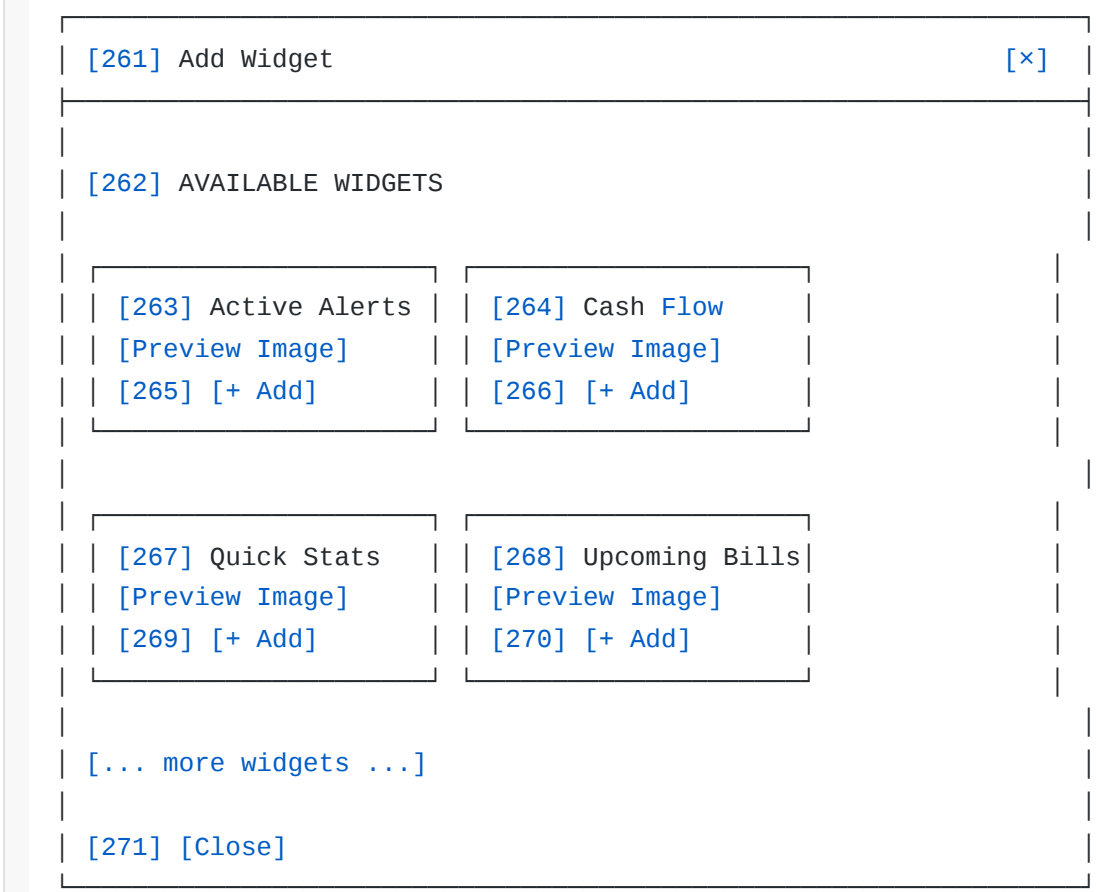
Widget Header Controls:

- ⚙️ **Configure:** Widget-specific settings (e.g., date range, data filters)
- 👁️ **Hide:** Temporarily hide widget without removing
- × **Remove:** Remove widget from dashboard
- ⋮⋮ **Drag Handle:** Click and drag to reposition (visible when unlocked)
- **Resize Handles:** Corner and edge handles for resizing (visible when unlocked)

Layout Settings Panel [QC: 241-260]



Widget Library Modal [QC: 261-280]



Default Layout Configurations

Executive View (Default)

```
{
  widgets: [
    { id: "alerts", x: 0, y: 0, w: 12, h: 2 },
    { id: "cashflow", x: 0, y: 2, w: 12, h: 4 },
    { id: "quickstats", x: 0, y: 6, w: 12, h: 2 },
  ]
}
```

Detailed Analysis

```
{
  widgets: [
    { id: "alerts", x: 0, y: 0, w: 6, h: 2 },
    { id: "qbsync", x: 6, y: 0, w: 6, h: 2 },
    { id: "cashflow", x: 0, y: 2, w: 8, h: 4 },
    { id: "calendar", x: 8, y: 2, w: 4, h: 4 },
    { id: "quickstats", x: 0, y: 6, w: 12, h: 2 },
    { id: "upcomingbills", x: 0, y: 8, w: 6, h: 3 },
    { id: "accountbalances", x: 6, y: 8, w: 6, h: 3 },
    { id: "recentactivity", x: 0, y: 11, w: 12, h: 4 },
  ]
}
```

Monitoring Mode

```
{
  widgets: [
    { id: "alerts", x: 0, y: 0, w: 8, h: 3 },
    { id: "qbsync", x: 8, y: 0, w: 4, h: 3 },
    { id: "upcomingbills", x: 0, y: 3, w: 6, h: 4 },
    { id: "accountbalances", x: 6, y: 3, w: 6, h: 4 },
    { id: "recentactivity", x: 0, y: 7, w: 12, h: 5 },
  ]
}
```

Responsive Behavior

Desktop (≥1200px)

- Full 12-column grid
- All widgets visible and resizable
- Drag-and-drop enabled

Tablet (768px-1199px)

- 6-column grid
- Widgets automatically reflow to fit
- Some widgets may stack vertically
- Drag-and-drop enabled with adjusted snap points

Mobile (<768px)

- Single column layout
- Widgets stack vertically in priority order
- Drag-and-drop disabled (use “Reorder” mode instead)
- Widgets are full-width, height-only resizing

User Experience Flow

Example: User Customizes Dashboard

1. User logs in and sees default “Executive View” layout
2. User clicks [🔓 Unlock Layout] button
3. Visual grid guides appear on dashboard
4. Resize handles appear on all widgets
5. User drags “Quick Stats” widget to top of page
6. Other widgets automatically reflow
7. User resizes “Cash Flow Chart” to be larger (8 rows instead of 4)
8. User clicks [+ Add Widget] button
9. Widget library modal opens
10. User clicks [+ Add] on “Calendar Preview” widget
11. Calendar widget appears at bottom of dashboard
12. User drags calendar to desired position

13. User clicks [🔒 Lock Layout] to prevent accidental changes
14. Layout is automatically saved to database
15. User can switch to “Detailed Analysis” preset anytime via Layout Settings

---

Implementation Priority

Phase 1 (MVP)

- [ ] Basic grid layout with fixed positions
- [ ] Drag-and-drop repositioning
- [ ] Layout persistence to database
- [ ] Lock/unlock toggle

Phase 2

- [ ] Widget resizing
- [ ] Widget library modal
- [ ] Add/remove widgets
- [ ] Layout presets

Phase 3

- [ ] Custom preset saving
- [ ] Layout export/import
- [ ] Advanced grid settings
- [ ] Undo/redo functionality

Technical Considerations

Performance

- Debounce layout saves (500ms delay) to avoid excessive database writes
- Use React.memo() for widget components to prevent unnecessary re-renders
- Lazy load widget content when not visible
- Virtualize long lists in widgets (e.g., Recent Activity)

Accessibility

- Keyboard navigation for drag-and-drop (arrow keys to move, +/- to resize)
- Screen reader announcements for layout changes
- Focus management when adding/removing widgets
- High contrast mode support for grid guides

Browser Compatibility

- Test on Chrome, Firefox, Safari, Edge
- Fallback to static layout on older browsers
- Touch support for tablet devices

Testing Checklist

- [ ] Widgets can be dragged to new positions
- [ ] Widgets can be resized by dragging handles
- [ ] Layout persists after page refresh
- [ ] Layout persists across different sessions
- [ ] Lock/unlock toggle works correctly
- [ ] Widget library modal opens and closes
- [ ] New widgets can be added from library

- [ ] Widgets can be removed
- [ ] Layout presets can be applied
- [ ] Custom layouts can be saved as presets
- [ ] Reset to default works correctly
- [ ] Responsive behavior works on tablet
- [ ] Responsive behavior works on mobile
- [ ] Keyboard navigation works
- [ ] Screen reader compatibility

Document Version: 1.0
Last Updated: November 23, 2025
Related: Financial Playbook Complete Documentation v8.0