Epic 4.2: Data Display Organisms

Epic Overview

This epic creates sophisticated data display organism components that handle complex business data visualization, including tables, grids, timelines, and workspace-specific data displays with advanced features like filtering, sorting, and real-time updates.

Priority: P0 (Critical)
Timeline: 4 weeks
Dependencies: Epic 3.2 (Display Molecules), Epic 4.1 (Navigation Organisms)

Story 4.2.1: Table Components

Overview

Create advanced table components with workspace context support, featuring sorting, filtering, virtual scrolling, and permission-based data display.

AI Developer Prompt

You are enhancing table components for THE WHEEL design system. Building on the workspace-specific navigation from Story 4.1.3, you need to create sophisticated data display organisms that handle complex business data across workspace contexts.

Context

- Complete navigation system with workspace context
- Existing DataTable and DataGrid components
- Need advanced data display for consultant workspace applications
- Must support large datasets with performance optimization
- Tables are critical for business data management

Requirements

1. Enhance DataTable with workspace context:

- Workspace-specific column types and formatting
- Context-aware styling and theming
- Permission-based column visibility

- Workspace data scoping
- Advanced sorting and filtering

2. Enhance DataGrid with workspace variants:

- Grid customization for different workspace types
- Responsive grid layouts
- Virtual scrolling for large datasets
- Bulk actions with workspace permissions
- Export functionality with workspace data

3. Add advanced table features:

- Inline editing with workspace validation
- Row selection with permission checking
- Column reordering and resizing
- Table state persistence
- Real-time data updates

Specific Tasks

- [ ] Enhance DataTable with workspace context
- [ ] Add workspace-specific column types
- [ ] Enhance DataGrid with workspace variants
- [ ] Implement virtual scrolling for performance
- [ ] Add bulk actions with permissions
- [ ] Create inline editing functionality
- [ ] Add table state persistence

Documentation Required

- Table system architecture
- Workspace-specific table patterns
- Performance optimization techniques
- Bulk action implementation
- Table accessibility guidelines
- Data formatting and validation

Testing Requirements

- Table functionality tests
- Performance tests with large datasets
- Workspace context tests
- Permission-based feature tests
- Accessibility compliance tests
- Real-time data update tests
- Cross-browser compatibility tests

Integration Points

- Integration with workspace context providers
- Permission system integration
- Data service integration
- Real-time update system integration
- Export service integration

Deliverables

- Enhanced DataTable with workspace context
- DataGrid with workspace variants
- Virtual scrolling implementation
- Bulk action system
- Table state persistence
- Comprehensive Storybook stories

Component Specifications

typescript

```typescript
interface DataTableProps<T> {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  data: T[]
  columns: TableColumn<T>[]
  loading?: boolean
  pagination?: PaginationConfig
  sorting?: SortingConfig
  filtering?: FilteringConfig
  selection?: SelectionConfig
  bulkActions?: BulkAction[]
  permissions?: string[]
  onRowClick?: (row: T) => void
  onSelectionChange?: (selected: T[]) => void
  onBulkAction?: (action: string, selected: T[]) => void
  virtualScrolling?: boolean
  responsive?: boolean
  exportable?: boolean
}

interface TableColumn<T> {
  key: keyof T
  title: string
  width?: number
  sortable?: boolean
  filterable?: boolean
  resizable?: boolean
  permission?: string
  workspaceContext?: 'consultant' | 'client' | 'admin' | 'neutral'
  render?: (value: any, row: T) => React.ReactNode
  headerRender?: () => React.ReactNode
  editable?: boolean
  validator?: (value: any) => boolean
}

interface DataGridProps<T> {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  data: T[]
  columns: GridColumn[]
  loading?: boolean
  responsive?: boolean
  virtualScrolling?: boolean
  itemHeight?: number
  onItemClick?: (item: T) => void
  onItemDoubleClick?: (item: T) => void
  customization?: GridCustomization
  permissions?: string[]
}

interface BulkAction {
  id: string
  label: string
  icon?: string
  permission?: string
  workspaceContext?: 'consultant' | 'client' | 'admin' | 'neutral'
  onClick: (selected: any[]) => void
  disabled?: boolean
  variant?: 'primary' | 'secondary' | 'danger'
}
```

Story 4.2.2: List & Grid Components

Overview

Create flexible list and grid display organisms for various content types with support for infinite scrolling, real-time updates, and workspace-specific layouts.

AI Developer Prompt

You are enhancing list and grid components for THE WHEEL design system. Building on the table components from Story 4.2.1, you need to create flexible data display organisms that work with various content types and workspace contexts.

Context

- Enhanced table components with workspace context
- Need flexible list and grid displays for different content types
- Must support responsive layouts and infinite scrolling
- Lists and grids are used throughout workspace applications
- Need real-time updates and filtering capabilities

Requirements

1. Enhance CardGrid with workspace cards:

- Workspace-specific card variants

- Responsive grid layouts
- Card customization for different data types
- Grid state persistence
- Infinite scrolling support

2. Enhance Timeline with workspace events:

- Workspace-specific event types
- Timeline filtering and grouping
- Real-time event updates
- Interactive timeline navigation
- Responsive timeline behavior

3. Enhance ActivityFeed with workspace activities:

- Workspace-scoped activity filtering
- Activity type customization
- Real-time activity updates
- Activity grouping and pagination
- User-specific activity views

Specific Tasks

- [ ] Enhance CardGrid with workspace cards
- [ ] Add responsive grid behavior
- [ ] Enhance Timeline with workspace events
- [ ] Add timeline filtering and grouping
- [ ] Enhance ActivityFeed with workspace activities
- [ ] Implement infinite scrolling
- [ ] Add real-time updates

Documentation Required

- List and grid system architecture
- Workspace-specific display patterns
- Infinite scrolling implementation
- Real-time update handling
- Responsive layout guidelines

- Activity feed customization

Testing Requirements

- Grid layout tests
- Timeline functionality tests
- Activity feed tests
- Infinite scrolling tests
- Real-time update tests
- Responsive layout tests
- Cross-browser compatibility tests

Integration Points

- Integration with workspace context providers
- Real-time update system integration
- Data service integration
- Infinite scrolling service integration
- Activity tracking integration

Deliverables

- Enhanced CardGrid with workspace cards
- Timeline with workspace events
- ActivityFeed with workspace activities
- Infinite scrolling implementation
- Real-time update system
- Comprehensive Storybook stories

Component Specifications

typescript

```
interface CardGridProps<T> {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  data: T[]
  cardComponent: React.ComponentType<{item: T, context?: string}>
  loading?: boolean
  columns?: number | 'auto'
  gap?: 'sm' | 'md' | 'lg'
  responsive?: boolean
  infiniteScroll?: boolean
  onLoadMore?: () => void
  onItemClick?: (item: T) => void
  onItemDoubleClick?: (item: T) => void
  filtering?: FilteringConfig
  sorting?: SortingConfig
  permissions?: string[]
}

interface TimelineProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  events: TimelineEvent[]
  loading?: boolean
  groupBy?: 'date' | 'type' | 'user'
  filtering?: TimelineFilter[]
  onEventClick?: (event: TimelineEvent) => void
  onFilterChange?: (filters: TimelineFilter[]) => void
  realTimeUpdates?: boolean
  responsive?: boolean
  maxHeight?: number
}

interface ActivityFeedProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  activities: Activity[]
  loading?: boolean
  grouped?: boolean
  onActivityClick?: (activity: Activity) => void
  onUserClick?: (user: User) => void
  realTimeUpdates?: boolean
  userFilters?: User[]
  typeFilters?: string[]
  onFilterChange?: (filters: ActivityFilter) => void
  infiniteScroll?: boolean
  onLoadMore?: () => void
}

interface TimelineEvent {
  id: string
  type: string
  title: string
  description?: string
  timestamp: Date
  user?: User
  workspaceContext?: 'consultant' | 'client' | 'admin' | 'neutral'
  metadata?: Record<string, any>
  icon?: string
  color?: string
}
```

Story 4.2.3: Workspace-Specific Data Display

Overview

Build specialized data display organisms for different workspace contexts, including dashboards and overview components tailored to specific user roles.

AI Developer Prompt

You are building workspace-specific data display organisms for THE WHEEL design system. Building on the list and grid components from Story 4.2.2, you need to create specialized data display components for different workspace contexts.

Context

- Enhanced list and grid components with workspace context
- Need specialized displays for different workspace types
- Must support workspace-specific data models and workflows
- Data displays should adapt to user roles and permissions
- Each workspace type has unique data visualization needs

Requirements

1. Build ClientDashboard component:

- Client-specific widgets and metrics
- Project progress visualization

- Communication timeline
- Document access and status
- Simplified, client-friendly interface

2. Build ConsultantDashboard component:

- Consultant-specific metrics and KPIs
- Multi-client overview
- Time tracking and billing summaries
- Task and project management
- Advanced analytics and reporting

3. Build WorkspaceOverview component:

- Workspace summary and statistics
- Team member activity
- Recent changes and updates
- Workspace health indicators
- Customizable widget layout

Specific Tasks

- [ ] Build ClientDashboard component
- [ ] Add client-specific widgets
- [ ] Build ConsultantDashboard component
- [ ] Add consultant-specific metrics
- [ ] Build WorkspaceOverview component
- [ ] Add workspace summary features
- [ ] Implement customizable layouts
- [ ] Add real-time data updates

Documentation Required

- Workspace-specific dashboard architecture
- Widget system implementation
- Dashboard customization patterns
- Real-time data integration
- Performance optimization techniques

- Accessibility guidelines

Testing Requirements

- Dashboard functionality tests
- Widget system tests
- Real-time data update tests
- Customization tests
- Performance tests
- Accessibility compliance tests
- Cross-browser compatibility tests

Integration Points

- Integration with workspace context providers
- Real-time data service integration
- Widget system integration
- Customization service integration
- Analytics integration

Deliverables

- ClientDashboard component with client-specific features
- ConsultantDashboard component with consultant features
- WorkspaceOverview component with summary features
- Widget system implementation
- Dashboard customization system
- Comprehensive Storybook stories

Component Specifications

typescript

```typescript
interface ClientDashboardProps {
  client: Client
  projects: Project[]
  currentProject?: Project
  onProjectChange?: (project: Project) => void
  widgets: DashboardWidget[]
  onWidgetInteraction?: (widget: DashboardWidget, action: string) => void
  realTimeUpdates?: boolean
  responsive?: boolean
  customizable?: boolean
}

interface ConsultantDashboardProps {
  consultant: Consultant
  clients: Client[]
  workspaces: Workspace[]
  widgets: DashboardWidget[]
  onWidgetInteraction?: (widget: DashboardWidget, action: string) => void
  onClientSelect?: (client: Client) => void
  onWorkspaceSelect?: (workspace: Workspace) => void
  realTimeUpdates?: boolean
  responsive?: boolean
  customizable?: boolean
}

interface WorkspaceOverviewProps {
  workspace: Workspace
  members: User[]
  activities: Activity[]
  metrics: WorkspaceMetrics
  widgets: DashboardWidget[]
  onWidgetInteraction?: (widget: DashboardWidget, action: string) => void
  onMemberClick?: (member: User) => void
  realTimeUpdates?: boolean
  responsive?: boolean
  customizable?: boolean
}

interface DashboardWidget {
  id: string
  title: string
  type: 'metric' | 'chart' | 'list' | 'table' | 'activity' | 'custom'
  size: 'small' | 'medium' | 'large'
  position: {x: number, y: number}
  data?: any
  config?: WidgetConfig
  permissions?: string[]
  workspaceContext?: 'consultant' | 'client' | 'admin' | 'neutral'
  refreshInterval?: number
  actions?: WidgetAction[]
}
```

Story 4.2.4: Advanced Filter Components

Overview

Create sophisticated filter components for complex data queries with support for multi-criteria filtering, saved filters, and workspace-specific filter options.

AI Developer Prompt

You are creating advanced filter components for THE WHEEL design system. Building on the existing data display organisms, you need to create sophisticated filtering interfaces for complex data.

Context

- Existing data display organisms with basic filtering
- Need for complex filtering in analytics and reporting
- Multiple workspace contexts requiring different filter types
- Real-time collaboration requiring shared filter states
- Integration with existing data management systems

Requirements

1. Create advanced filter components:

- Multi-criteria filter builder
- Date range and time-based filters
- Numeric range and comparison filters
- Text search with fuzzy matching
- Saved filter presets and templates

2. Implement workspace context features:

- Context-specific filter options
- Permission-based filter access
- Workspace-specific filter presets
- Role-based filter capabilities
- Context-aware filter suggestions

3. Create collaborative filtering features:

- Shared filter states across users
- Filter collaboration and sharing
- Filter analytics and usage tracking
- Real-time filter updates
- Filter export and import

Specific Tasks

- [ ] Create FilterBuilder component
- [ ] Implement DateRangeFilter component
- [ ] Set up NumericRangeFilter component
- [ ] Create TextSearchFilter component
- [ ] Implement filter presets
- [ ] Set up filter collaboration

Documentation Required

- Advanced filter API documentation
- Filter builder configuration
- Collaboration features guide
- Performance optimization
- Filter pattern guidelines

Testing Requirements

- Filter functionality tests
- Multi-criteria filter tests
- Collaboration feature tests
- Performance and scalability tests
- Filter accuracy tests

Integration Points

- Integration with data display organisms
- Workspace context integration
- Data management integration
- Real-time collaboration integration
- Analytics and tracking integration

Deliverables

- Complete advanced filter system
- Multi-criteria filter builder
- Collaborative filtering features
- Performance-optimized filtering
- Comprehensive filter documentation

Performance Requirements

- Filter processing under 500ms
- Real-time updates under 200ms
- Memory usage under 60MB
- Filter search under 300ms
- Collaboration sync under 100ms

Story 4.2.5: Data Export Components

Overview

Create comprehensive data export components with support for multiple formats, scheduled exports, and workspace-specific export permissions.

AI Developer Prompt

You are creating data export components for THE WHEEL design system. Building on the advanced filter components from Story 4.2.4, you need to create comprehensive data export functionality.

Context

- Advanced filter system established with multi-criteria filtering

- Need for data export in various formats
- Multiple workspace contexts requiring different export permissions
- Real-time collaboration requiring export coordination
- Integration with existing data management and analytics

Requirements

1. Create data export components:

- Export wizard with format selection
- Batch export for large datasets
- Scheduled export automation
- Custom export templates
- Export progress tracking

2. Implement workspace context features:

- Context-specific export permissions
- Role-based export capabilities
- Workspace-specific export formats
- Brand-aware export styling
- Context-aware export validation

3. Create advanced export features:

- Real-time export status updates
- Export history and tracking
- Export sharing and collaboration
- Export analytics and usage
- Export security and encryption

Specific Tasks

- [ ] Create ExportWizard component
- [ ] Implement BatchExport component
- [ ] Set up ScheduledExport component
- [ ] Create ExportProgress component
- [ ] Implement export templates

- [ ] Set up export security

Documentation Required

- Data export API documentation
- Export format specifications
- Security implementation guide
- Automation configuration
- Performance optimization

Testing Requirements

- Export functionality tests
- Batch export performance tests
- Security and encryption tests
- Format validation tests
- Export accuracy tests

Integration Points

- Integration with advanced filter system
- Workspace context integration
- Data management integration
- Security and encryption integration
- Analytics and tracking integration

Deliverables

- Complete data export system
- Multiple export formats support
- Security and encryption features
- Automation and scheduling
- Comprehensive export documentation

Performance Requirements

- Export processing under 5 seconds
- Batch export under 30 seconds

- Memory usage under 200MB
- Export generation under 10 seconds
- Real-time updates under 100ms

Timeline and Dependencies

Timeline

- Week 1: Story 4.2.1 - Table Components
- Week 1-2: Story 4.2.2 - List & Grid Components
- Week 2-3: Story 4.2.3 - Workspace-Specific Data Display
- Week 3: Story 4.2.4 - Advanced Filter Components
- Week 4: Story 4.2.5 - Data Export Components

Dependencies

- Epic 3.2 (Display Molecules) - Complete
- Epic 4.1 (Navigation Organisms) - Complete
- Data service infrastructure available
- Real-time update system operational
- Permission system established

Success Metrics

- All data display components support workspace contexts
- Virtual scrolling handles 10,000+ rows smoothly
- Real-time updates latency under 200ms
- Export functionality supports all major formats
- 100% accessibility compliance
- Complete test coverage (90%+ for all components)

Risk Mitigation

- Performance testing with large datasets
- Progressive enhancement for complex features
- Regular accessibility audits
- Cross-browser compatibility testing

- User testing for data visualization
- Clear documentation for all patterns