Epic 3.2: Display Molecules

Epic Overview

This epic creates sophisticated display molecule components by combining atomic display elements into comprehensive information cards, progress indicators, and data visualization components with workspace awareness.

Priority: P0 (Critical)
Timeline: 4 weeks
Dependencies: Epic 2.2 (Display Components), Epic 2.3 (Layout Components), Epic 3.1 (Form Molecules)

Story 3.2.1: Information Display Cards

Overview

Create sophisticated information display cards that present user data, statistics, and status information consistently across workspace contexts.

AI Developer Prompt

You are enhancing information display card components for THE WHEEL design system. Building on the specialized input molecules from Story 3.1.3, you need to create sophisticated display cards that present information consistently across workspace contexts.

Context

- Complete specialized input molecule system with workspace integration
- Need information display cards for various data types
- Must support workspace-specific styling and content
- Display cards are critical for data presentation and user experience
- Need consistent patterns across all workspace contexts

Requirements

1. Enhance UserCard component with workspace context:

- Workspace-specific user information display
- Presence indicators and status
- Role-based information visibility

- Interactive user actions
- Responsive card layouts

2. Enhance StatCard component with workspace metrics:

- Workspace-specific KPI display
- Trend indicators and comparisons
- Interactive metric exploration
- Real-time data updates
- Context-aware formatting

3. Enhance StatusCard component with workspace statuses:

- Workspace-specific status types
- Status transition indicators
- Action buttons for status changes
- Historical status tracking
- Alert and notification integration

Specific Tasks

- [ ] Enhance UserCard with workspace context
- [ ] Add presence indicators and status
- [ ] Enhance StatCard with workspace metrics
- [ ] Add trend indicators and comparisons
- [ ] Enhance StatusCard with workspace statuses
- [ ] Add status transition logic
- [ ] Create consistent card API
- [ ] Add real-time updates

Documentation Required

- Information display card architecture
- Workspace-specific card patterns
- Real-time data integration
- Status management system
- Card interaction patterns
- Accessibility implementation

Testing Requirements

- Card display functionality tests
- Workspace context styling tests
- Real-time data update tests
- Status transition tests
- Accessibility compliance tests
- Performance tests for data updates

Integration Points

- Integration with workspace context providers
- Real-time data service integration
- Status management system integration
- User presence system integration
- Metrics and analytics integration

Deliverables

- Enhanced UserCard with workspace context
- StatCard with workspace metrics
- StatusCard with workspace statuses
- Real-time data update system
- Status transition management
- Comprehensive Storybook stories

Component Specifications

typescript

```
interface UserCardProps {
  user: User
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showPresence?: boolean
  showStatus?: boolean
  showActions?: boolean
  onUserClick?: (user: User) => void
  onActionClick?: (action: string, user: User) => void
  permissions?: string[]
  size?: 'sm' | 'md' | 'lg'
  layout?: 'horizontal' | 'vertical'
}

interface StatCardProps {
  title: string
  value: string | number
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  trend?: {
    value: number
    direction: 'up' | 'down' | 'stable'
    period: string
  }
  comparison?: {
    value: number
    label: string
    type: 'previous' | 'target' | 'average'
  }
  format?: 'currency' | 'percentage' | 'number' | 'custom'
  formatter?: (value: any) => string
  onClick?: () => void
  loading?: boolean
  size?: 'sm' | 'md' | 'lg'
}

interface StatusCardProps {
  title: string
  status: Status
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showHistory?: boolean
  showActions?: boolean
  onStatusChange?: (newStatus: Status) => void
  onActionClick?: (action: string) => void
  permissions?: string[]
```

```
  size?: 'sm' | 'md' | 'lg'
}
```

Story 3.2.2: Progress & Activity Components

Overview

Create progress tracking and activity display components that show progress, activities, and notifications across workspace contexts.

AI Developer Prompt

You are enhancing progress and activity components for THE WHEEL design system. Building on the information display cards from Story 3.2.1, you need to create components that track and display progress and activities across workspace contexts.

Context

- Enhanced information display cards with workspace context
- Need progress tracking and activity display components
- Must support workspace-specific progress types and activities
- Progress and activity components are critical for project management
- Need real-time updates and interactive features

Requirements

1. Enhance ProgressCard component with workspace progress:

- Workspace-specific progress types and metrics
- Visual progress indicators and milestones
- Interactive progress updates
- Progress history and trends
- Automated progress calculations

2. Build ActivityCard component for workspace activities:

- Workspace-scoped activity filtering
- Activity type categorization
- User-specific activity views
- Interactive activity actions

- Real-time activity updates

3. Enhance NotificationCard component with workspace context:

- Workspace-specific notification types
- Notification priority and urgency
- Interactive notification actions
- Notification history and tracking
- Bulk notification management

Specific Tasks

- [ ] Enhance ProgressCard with workspace progress
- [ ] Add milestone indicators and tracking
- [ ] Build ActivityCard component
- [ ] Add activity filtering and categorization
- [ ] Enhance NotificationCard with workspace context
- [ ] Add notification actions and management
- [ ] Create real-time update system
- [ ] Add progress calculation utilities

Documentation Required

- Progress and activity system architecture
- Workspace-specific progress patterns
- Activity filtering and categorization
- Real-time update implementation
- Notification management system
- Progress calculation methods

Testing Requirements

- Progress tracking functionality tests
- Activity filtering and display tests
- Notification management tests
- Real-time update tests
- Workspace context tests
- Performance tests for activity feeds

Integration Points

- Integration with workspace context providers
- Real-time activity streaming integration
- Progress calculation service integration
- Notification system integration
- User activity tracking integration

Deliverables

- Enhanced ProgressCard with workspace progress
- ActivityCard component with filtering
- NotificationCard with workspace context
- Real-time update system
- Progress calculation utilities
- Comprehensive Storybook stories

Component Specifications

```typescript
interface ProgressCardProps {
  title: string
  progress: Progress
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showMilestones?: boolean
  showTrend?: boolean
  onMilestoneClick?: (milestone: Milestone) => void
  onProgressUpdate?: (progress: Progress) => void
  size?: 'sm' | 'md' | 'lg'
  interactive?: boolean
}

interface ActivityCardProps {
  activities: Activity[]
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  filters?: ActivityFilter[]
  onActivityClick?: (activity: Activity) => void
  onFilterChange?: (filters: ActivityFilter[]) => void
  realTimeUpdates?: boolean
  maxItems?: number
  showLoadMore?: boolean
  onLoadMore?: () => void
}

interface NotificationCardProps {
  notification: Notification
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showActions?: boolean
  onActionClick?: (action: string) => void
  onDismiss?: () => void
  onMarkRead?: () => void
  compact?: boolean
}
```

Story 3.2.3: Workspace-Specific Display Cards

Overview

Build specialized display cards for different workspace contexts, including client cards, workspace overviews, project cards, and billing information.

AI Developer Prompt

You are building workspace-specific display cards for THE WHEEL design system. Building on the progress and activity components from Story 3.2.2, you need to create specialized display cards for different workspace contexts and business needs.

Context

- Complete progress and activity component system
- Need specialized display cards for workspace-specific data
- Must support different workspace contexts with unique requirements
- Display cards should adapt to user roles and permissions
- Need consistent patterns across all workspace types

Requirements

1. Build ClientCard component:

- Client information display with workspace context
- Project associations and status
- Communication history and actions
- Client-specific metrics and KPIs
- Permission-based information visibility

2. Build WorkspaceCard component:

- Workspace overview and summary
- Team member information
- Recent activity and updates
- Workspace-specific metrics
- Quick actions and navigation

3. Build ProjectCard component:

- Project information and status
- Progress tracking and milestones
- Team assignments and roles
- Project-specific actions

- Timeline and deadline tracking

4. Build BillingCard component:

- Billing information and status
- Payment history and tracking
- Invoice generation and management
- Revenue metrics and trends
- Billing-specific actions

Specific Tasks

- [ ] Build ClientCard component
- [ ] Add client status and metrics
- [ ] Build WorkspaceCard component
- [ ] Add workspace overview features
- [ ] Build ProjectCard component
- [ ] Add project progress tracking
- [ ] Build BillingCard component
- [ ] Add billing management features
- [ ] Create consistent card patterns
- [ ] Add workspace-specific actions

Documentation Required

- Workspace-specific card architecture
- Client management patterns
- Project tracking implementation
- Billing management system
- Card interaction patterns
- Permission-based display logic

Testing Requirements

- Client card functionality tests
- Workspace card display tests
- Project card progress tests
- Billing card management tests

- Permission-based visibility tests
- Workspace context tests

Integration Points

- Integration with workspace context providers
- Client management system integration
- Project tracking system integration
- Billing management system integration
- Permission system integration

Deliverables

- ClientCard component with workspace context
- WorkspaceCard component with overview
- ProjectCard component with progress tracking
- BillingCard component with management
- Consistent card pattern library
- Comprehensive Storybook stories

Component Specifications

typescript

```
interface ClientCardProps {
  client: Client
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showProjects?: boolean
  showMetrics?: boolean
  showActions?: boolean
  onClientClick?: (client: Client) => void
  onProjectClick?: (project: Project) => void
  onActionClick?: (action: string, client: Client) => void
  permissions?: string[]
  size?: 'sm' | 'md' | 'lg'
}

interface WorkspaceCardProps {
  workspace: Workspace
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showMembers?: boolean
  showActivity?: boolean
  showMetrics?: boolean
  onWorkspaceClick?: (workspace: Workspace) => void
  onMemberClick?: (member: User) => void
  onActivityClick?: (activity: Activity) => void
  permissions?: string[]
  size?: 'sm' | 'md' | 'lg'
}

interface ProjectCardProps {
  project: Project
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showProgress?: boolean
  showTeam?: boolean
  showActions?: boolean
  onProjectClick?: (project: Project) => void
  onTeamMemberClick?: (member: User) => void
  onActionClick?: (action: string, project: Project) => void
  permissions?: string[]
  size?: 'sm' | 'md' | 'lg'
}

interface BillingCardProps {
  billing: BillingInfo
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  showPaymentHistory?: boolean
```

```
  showInvoices?: boolean
  showActions?: boolean
  onBillingClick?: (billing: BillingInfo) => void
  onInvoiceClick?: (invoice: Invoice) => void
  onActionClick?: (action: string, billing: BillingInfo) => void
  permissions?: string[]
  size?: 'sm' | 'md' | 'lg'
}
```

Story 3.2.4: Media Player Components

Overview

Create media player components for video and audio content, including training materials, presentations, and collaborative media viewing.

AI Developer Prompt

You are creating media player components for THE WHEEL design system. Building on the existing display molecules, you need to create video and audio players for training and presentation content.

Context

- Existing display molecule system with workspace context
- Need for training videos and presentation content
- Multiple workspace contexts requiring different media experiences
- Real-time collaboration features for shared viewing
- Integration with existing asset management system

Requirements

1. Create media player components:

- Video player with custom controls
- Audio player with waveform visualization
- Live streaming player for real-time content
- Presentation player with slide navigation
- Interactive media with annotations

2. Implement workspace context features:

- Context-specific player theming
- Permission-based media access
- Workspace-specific player controls
- Brand-aware player interface
- Context-aware media recommendations

3. Create collaborative features:

- Synchronized viewing across users
- Real-time comments and annotations
- Shared playback controls
- Viewing analytics and tracking
- Interactive media sessions

Specific Tasks

- [ ] Create VideoPlayer component
- [ ] Implement AudioPlayer component
- [ ] Set up live streaming support
- [ ] Create presentation player
- [ ] Implement synchronized viewing
- [ ] Set up media analytics

Documentation Required

- Media player component API
- Streaming integration guide
- Collaboration features documentation
- Analytics implementation
- Performance optimization guide

Testing Requirements

- Media player functionality tests
- Streaming performance tests
- Collaboration feature tests
- Cross-browser compatibility tests
- Mobile device tests

Integration Points

- Integration with asset management system
- Workspace context integration
- Real-time collaboration integration
- Analytics and tracking integration
- Streaming service integration

Deliverables

- Complete media player system
- Streaming and live content support
- Collaborative viewing features
- Analytics and tracking
- Comprehensive media documentation

Performance Requirements

- Media loading under 2 seconds
- Streaming startup under 3 seconds
- Memory usage under 200MB
- CPU usage under 30%
- Synchronization accuracy within 100ms

Story 3.2.5: Chart & Visualization Components

Overview

Create interactive chart and visualization components for data analytics and reporting with workspace-specific styling and real-time updates.

AI Developer Prompt

You are creating chart and visualization components for THE WHEEL design system. Building on the existing display molecules, you need to create interactive data visualization for analytics and reporting.

Context

- Existing display molecule system with workspace context

- Need for business analytics and reporting
- Multiple workspace contexts requiring different chart types
- Real-time collaboration features for shared analysis
- Integration with existing data and analytics systems

Requirements

1. Create chart components:

- Line charts for time series data
- Bar charts for categorical data
- Pie charts for proportional data
- Scatter plots for correlation analysis
- Area charts for cumulative data

2. Implement workspace context features:

- Context-specific chart themes
- Permission-based data access
- Workspace-specific chart types
- Brand-aware visualization styling
- Context-aware data filtering

3. Create interactive features:

- Real-time data updates
- Interactive tooltips and legends
- Chart zooming and panning
- Data export and sharing
- Collaborative chart annotation

Specific Tasks

- [ ] Create LineChart component
- [ ] Implement BarChart component
- [ ] Set up PieChart component
- [ ] Create ScatterPlot component
- [ ] Implement interactive features

- [ ] Set up real-time data updates

Documentation Required

- Chart component API documentation
- Data visualization guidelines
- Interactive features guide
- Performance optimization
- Accessibility implementation

Testing Requirements

- Chart rendering tests
- Interactive feature tests
- Data update tests
- Performance and memory tests
- Accessibility compliance tests

Integration Points

- Integration with data analytics system
- Workspace context integration
- Real-time data integration
- Theme system integration
- Export and sharing integration

Deliverables

- Complete chart visualization library
- Interactive chart features
- Real-time data updates
- Accessibility-compliant charts
- Comprehensive visualization documentation

Performance Requirements

- Chart rendering under 1 second
- Data update under 500ms

- Memory usage under 100MB
- Animation performance maintains 60fps
- Data processing under 2 seconds

Timeline and Dependencies

Timeline

- Week 1: Story 3.2.1 - Information Display Cards
- Week 1-2: Story 3.2.2 - Progress & Activity Components
- Week 2-3: Story 3.2.3 - Workspace-Specific Display Cards
- Week 3: Story 3.2.4 - Media Player Components
- Week 4: Story 3.2.5 - Chart & Visualization Components

Dependencies

- Epic 2.2 (Display Components) - Complete
- Epic 2.3 (Layout Components) - Complete
- Epic 3.1 (Form Molecules) - Should be complete
- Workspace context system operational
- Real-time collaboration infrastructure

Success Metrics

- All display molecules support workspace contexts
- Real-time data updates working across components
- 100% accessibility compliance for all components
- Performance benchmarks met (60fps interactions)
- Complete test coverage (90%+ for all components)
- Consistent visual language across all cards

Risk Mitigation

- Progressive enhancement for complex visualizations
- Performance monitoring during development
- Regular accessibility audits
- Cross-browser testing for media players

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