Epic 2.4: Icon & Asset System

Epic Overview

This epic establishes the comprehensive icon library and asset management system for THE WHEEL design system, providing scalable, accessible, and workspace-aware visual assets.

Priority: P0 (Critical)
Timeline: 3 weeks
Dependencies: Epic 1.1 (Monorepo Architecture Setup), Epic 2.2 (Display Components)

Story 2.4.1: Icon Library Architecture

Overview

Create the foundational icon library architecture with tree-shaking support, workspace context awareness, and comprehensive icon management.

AI Developer Prompt

You are creating the icon library architecture for THE WHEEL design system. This is a critical P0 component that will be used throughout all workspace contexts and component packages.

Context

- Monorepo structure with packages/ui/ for core components
- Sophisticated theming system with CSS variables
- Multiple workspace contexts requiring different icon styles
- Existing sophisticated component library with 85% completion
- Real-time collaboration features requiring status icons

Requirements

1. Create scalable icon library architecture:

- SVG-based icon system with tree-shaking support
- Icon variants for different workspace contexts
- Size variants (xs, sm, md, lg, xl) with consistent scaling
- Color variants integrated with theme system
- Accessibility compliance with proper ARIA attributes

2. Implement workspace context integration:

- Context-aware icon styling (consultant, client, admin, expert, tool creator, founder)
- Theme-based color application
- Dynamic icon selection based on workspace
- Permission-based icon visibility
- Real-time status icon updates

3. Create icon management system:

- Icon registration and discovery system
- TypeScript type generation for icons
- Icon optimization and compression
- Icon versioning and updates
- Icon usage analytics

Specific Tasks

- [ ] Create Icon base component with props interface
- [ ] Implement SVG icon loading and caching
- [ ] Set up icon variant system for workspace contexts
- [ ] Create icon registration system
- [ ] Implement icon optimization pipeline
- [ ] Set up TypeScript type generation for icons

Documentation Required

- Icon library architecture documentation
- Icon usage guidelines for different contexts
- Accessibility implementation guide
- Icon optimization best practices
- Icon contribution guidelines

Testing Requirements

- Icon rendering tests across all contexts
- Accessibility compliance tests
- Performance tests for icon loading

- Type safety tests for icon props
- Icon optimization validation tests

Integration Points

- Integration with existing theme system
- Compatibility with workspace context providers
- Support for real-time collaboration features
- Storybook integration for icon showcase
- Build system integration for optimization

Deliverables

- Complete icon library architecture
- Icon component with context awareness
- Icon optimization and build system
- TypeScript type definitions
- Comprehensive icon documentation

Performance Requirements

- Icon loading under 50ms
- Bundle size under 100KB for icon library
- Tree-shaking reduces unused icons by 80%+
- Icon switching under 16ms (60fps)
- Memory usage under 10MB for icon cache

Component Specifications

typescript

```typescript
interface IconProps extends React.SVGAttributes<SVGElement> {
  name: string
  size?: 'xs' | 'sm' | 'md' | 'lg' | 'xl'
  color?: 'primary' | 'secondary' | 'muted' | 'error' | 'warning' | 'success' | 'current'
  context?: 'consultant' | 'client' | 'admin' | 'expert' | 'tool-creator' | 'founder' | 'neutral'
  variant?: 'filled' | 'outlined' | 'two-tone'
  title?: string
  className?: string
  onClick?: () => void
}

interface IconRegistry {
  register(name: string, icon: IconDefinition): void
  get(name: string): IconDefinition | undefined
  list(): string[]
  getByCategory(category: string): IconDefinition[]
  search(query: string): IconDefinition[]
}

interface IconDefinition {
  name: string
  category: string
  tags: string[]
  variants: {
    filled?: string
    outlined?: string
    twoTone?: string
  }
  workspaceContexts?: string[]
  deprecated?: boolean
}
```

Story 2.4.2: SVG Icon Components

Overview

Implement comprehensive SVG icon components covering all workspace needs with proper optimization and accessibility.

AI Developer Prompt

You are implementing SVG icon components for THE WHEEL design system. Building on the icon library architecture from Story 2.4.1, you need to create production-ready SVG components.

Context

- Icon library architecture established with workspace context support
- Need comprehensive icon set for all workspace types
- Existing theming system with CSS variables
- Real-time collaboration features requiring status and notification icons
- Accessibility requirements for screen readers and keyboard navigation

Requirements

1. Create core SVG icon components:

- Business icons: briefcase, chart, client, project, invoice, time
- Navigation icons: menu, close, arrow, chevron, tab, breadcrumb
- Status icons: success, warning, error, info, loading, online/offline
- Action icons: edit, delete, copy, share, download, upload
- Communication icons: comment, chat, notification, email, call

2. Implement workspace-specific icon variants:

- Consultant icons: analytics, client management, project tracking
- Client icons: dashboard, progress, communication, documents
- Admin icons: settings, users, permissions, system health
- Expert icons: marketplace, services, ratings, expertise
- Tool Creator icons: development, deployment, analytics, API
- Founder icons: funding, team, growth, metrics

3. Create icon component system:

- Icon wrapper with consistent props interface
- Size and color variant system
- Animation support for status icons
- Accessibility attributes integration
- Performance optimization for rendering

Specific Tasks

- [ ] Create 50+ core SVG icon components
- [ ] Implement workspace-specific icon variants
- [ ] Set up icon animation system
- [ ] Create icon size and color variants
- [ ] Implement accessibility attributes
- [ ] Set up icon performance optimization

Documentation Required

- Icon component API documentation
- Icon usage guidelines per workspace
- Animation implementation guide
- Accessibility features documentation
- Icon customization guide

Testing Requirements

- Icon rendering tests for all variants
- Animation performance tests
- Accessibility compliance tests
- Color variant validation tests
- Size variant consistency tests

Integration Points

- Integration with icon library architecture
- Theme system color integration
- Workspace context integration
- Real-time status update integration
- Storybook story creation

Deliverables

- Complete SVG icon component library
- Workspace-specific icon variants
- Animation system for status icons

- Accessibility-compliant icon components
- Comprehensive icon documentation

Performance Requirements

- Icon rendering under 16ms (60fps)
- SVG optimization reduces size by 40%+
- Animation performance maintains 60fps
- Memory usage under 5MB for icon cache
- Loading performance under 100ms

Icon Categories and Examples

typescript

```typescript
// Business Icons
export const BusinessIcons = {
  briefcase: BriefcaseIcon,
  chart: ChartIcon,
  client: ClientIcon,
  project: ProjectIcon,
  invoice: InvoiceIcon,
  time: TimeIcon,
  revenue: RevenueIcon,
  contract: ContractIcon,
  meeting: MeetingIcon,
  report: ReportIcon
}

// Navigation Icons
export const NavigationIcons = {
  menu: MenuIcon,
  close: CloseIcon,
  arrowLeft: ArrowLeftIcon,
  arrowRight: ArrowRightIcon,
  chevronUp: ChevronUpIcon,
  chevronDown: ChevronDownIcon,
  tab: TabIcon,
  breadcrumb: BreadcrumbIcon,
  home: HomeIcon,
  back: BackIcon
}

// Status Icons
export const StatusIcons = {
  success: SuccessIcon,
  warning: WarningIcon,
  error: ErrorIcon,
  info: InfoIcon,
  loading: LoadingIcon,
  online: OnlineIcon,
  offline: OfflineIcon,
  busy: BusyIcon,
  away: AwayIcon,
  pending: PendingIcon
}
```

Story 2.4.3: Asset Management System

Overview

Create a comprehensive asset management system for images, illustrations, and brand assets with workspace-specific organization.

AI Developer Prompt

You are creating the asset management system for THE WHEEL design system. Building on the SVG icon components from Story 2.4.2, you need to create a comprehensive asset management solution.

Context

- SVG icon components established with workspace variants
- Need to manage images, illustrations, and brand assets
- Multiple workspace contexts requiring different asset sets
- Existing theming system for brand customization
- Real-time collaboration requiring shared asset access

Requirements

1. Create asset management architecture:

- Asset storage and retrieval system
- Asset optimization and compression
- Asset versioning and updates
- Asset caching and performance optimization
- Asset security and access control

2. Implement workspace asset organization:

- Brand assets per workspace context
- Illustration sets for different industries
- Image placeholders and defaults
- Logo management system
- Asset categorization and tagging

3. Create asset component system:

- Image component with optimization
- Illustration component with variants
- Logo component with brand integration
- Asset placeholder system
- Lazy loading and performance optimization

Specific Tasks

- [ ] Create asset management service
- [ ] Implement asset optimization pipeline
- [ ] Set up asset caching system
- [ ] Create asset component library
- [ ] Implement asset security and access control
- [ ] Set up asset versioning system

Documentation Required

- Asset management system documentation
- Asset optimization guidelines
- Asset security implementation
- Asset contribution guidelines
- Asset usage best practices

Testing Requirements

- Asset loading performance tests
- Asset optimization validation tests
- Security and access control tests
- Asset versioning tests
- Cache performance tests

Integration Points

- Integration with existing theme system
- Workspace context integration
- Real-time asset sharing integration
- Build system integration
- CDN integration for asset delivery

Deliverables

- Complete asset management system
- Asset optimization pipeline
- Asset component library
- Security and access control system
- Asset management documentation

Performance Requirements

- Asset loading under 200ms
- Asset optimization reduces size by 60%+
- Cache hit rate above 90%
- Memory usage under 50MB for asset cache
- Asset security response under 50ms

Asset Management Specifications

typescript

```
interface AssetManagerConfig {
  storage: StorageProvider
  optimization: OptimizationConfig
  cache: CacheConfig
  security: SecurityConfig
  cdn?: CDNConfig
}

interface Asset {
  id: string
  name: string
  type: 'image' | 'illustration' | 'logo' | 'document'
  category: string
  tags: string[]
  workspaceContexts: string[]
  versions: AssetVersion[]
  metadata: AssetMetadata
  permissions: AssetPermissions
}

interface AssetVersion {
  id: string
  version: string
  url: string
  size: number
  format: string
  optimized: boolean
  created: Date
  checksum: string
}

interface AssetComponent {
  Image: React.FC<ImageAssetProps>
  Illustration: React.FC<IllustrationAssetProps>
  Logo: React.FC<LogoAssetProps>
  Placeholder: React.FC<PlaceholderProps>
}

interface ImageAssetProps {
  assetId: string
  alt: string
  context?: WorkspaceContext
  size?: 'thumbnail' | 'small' | 'medium' | 'large' | 'full'
  lazy?: boolean
  placeholder?: boolean
  onLoad?: () => void
  onError?: () => void
}
```

Timeline and Dependencies

Timeline

- Week 1: Story 2.4.1 - Icon Library Architecture
- Week 1-2: Story 2.4.2 - SVG Icon Components
- Week 2-3: Story 2.4.3 - Asset Management System

Dependencies

- Epic 1.1 (Monorepo Architecture Setup) - Complete
- Epic 2.2 (Display Components) - Should be complete
- Theme system operational
- Build system configured

Success Metrics

- Complete icon library with 100+ icons
- All icons support workspace contexts
- Asset optimization reduces file sizes by 50%+
- 100% accessibility compliance for icons
- Performance benchmarks met (60fps animations)
- Complete test coverage (90%+ for all components)

Risk Mitigation

- Incremental icon development and release
- Performance monitoring during development
- Regular accessibility audits
- Cross-browser testing for SVG rendering
- Clear icon contribution guidelines
- Automated optimization pipeline