Epic 3.1: Form Molecules

Epic Overview

This epic creates sophisticated form molecule components by combining atomic form elements into complete, reusable form patterns with workspace context awareness and advanced functionality.

Priority: P0 (Critical)
Timeline: 4 weeks
Dependencies: Epic 2.1 (Input Components), Epic 2.2 (Display Components), Epic 2.3 (Layout Components)

Story 3.1.1: Form Field Components

Overview

Create comprehensive form field components that combine labels, inputs, helper text, and error messages into cohesive, accessible form interactions.

AI Developer Prompt

You are enhancing form field components for THE WHEEL design system. Building on the atomic components from Feature 2, you need to create sophisticated form field molecules that combine atoms into complete form interactions.

Context

- Complete atomic component system with workspace context
- Enhanced input components with validation and theming
- Typography system with workspace context
- Need to combine atoms into complete form field experiences
- Form fields are critical for all workspace applications

Requirements

1. Enhance FormField component with workspace validation:

- Combine Label, Input, HelperText, ErrorMessage atoms
- Workspace context styling throughout
- Validation state management and display

- Required field indication
- Accessibility improvements with proper ARIA relationships

2. Create consistent error handling across all fields:

- Standardized error message formatting
- Workspace context error styling
- Error state animations
- Multi-error handling
- Real-time validation feedback

3. Implement responsive form behavior:

- Mobile-first form field layouts
- Responsive label positioning
- Touch-friendly input sizing
- Keyboard navigation improvements
- Screen reader optimization

Specific Tasks

- [ ] Enhance FormField with workspace context
- [ ] Implement validation state management
- [ ] Create consistent error display system
- [ ] Add required field indicators
- [ ] Implement responsive form layouts
- [ ] Add comprehensive accessibility features
- [ ] Create form field composition patterns

Documentation Required

- Form field system architecture
- Validation implementation guide
- Accessibility best practices
- Error handling patterns
- Responsive form design guidelines
- Form field composition examples

Testing Requirements

- Form field validation tests
- Workspace context styling tests
- Accessibility compliance tests
- Responsive behavior tests
- Error handling tests
- Form field composition tests
- Cross-browser compatibility tests

Integration Points

- Integration with workspace context providers
- Validation system integration
- Error handling system integration
- Accessibility utilities integration
- Form state management integration

Deliverables

- Enhanced FormField component with workspace validation
- Consistent error handling system
- Responsive form field layouts
- Accessibility compliance validation
- Form field composition patterns
- Comprehensive Storybook stories

Component Specifications

typescript

```typescript
interface FormFieldProps {
  label?: string
  helperText?: string
  errorMessage?: string
  required?: boolean
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  validationState?: 'error' | 'warning' | 'success'
  layout?: 'vertical' | 'horizontal'
  responsive?: boolean
  children: React.ReactElement
  id?: string
  disabled?: boolean
}

interface FieldWrapperProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  validationState?: 'error' | 'warning' | 'success'
  focused?: boolean
  disabled?: boolean
  children: React.ReactNode
}

interface ErrorMessageProps {
  message?: string
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  multiple?: boolean
  errors?: string[]
  animate?: boolean
}
```

Story 3.1.2: Search & Filter Components

Overview

Create advanced search and filter components with workspace context awareness and sophisticated filtering capabilities.

AI Developer Prompt

You are enhancing search and filter components for THE WHEEL design system. Building on the form field components from Story 3.1.1, you need to create sophisticated search and filtering experiences

with workspace context awareness.

Context

- Enhanced form field components with workspace validation
- Need powerful search and filtering for workspace data
- Must support workspace scoping and permissions
- Search components are critical for data discovery
- Filter components handle complex business logic

Requirements

1. Enhance SearchBar with workspace scoping:

- Workspace-specific search contexts
- Autocomplete with workspace data
- Recent searches with workspace filtering
- Search suggestions based on workspace permissions
- Real-time search with debouncing

2. Enhance FilterPanel with workspace filters:

- Workspace-specific filter options
- Filter persistence across sessions
- Complex filter combinations
- Filter validation and error handling
- Responsive filter layouts

3. Add performance optimizations:

- Debounced search input
- Virtualized filter options
- Lazy loading for large datasets
- Search result caching
- Filter state optimization

Specific Tasks

- [ ] Enhance SearchBar with workspace scoping

- [ ] Add autocomplete functionality
- [ ] Enhance FilterPanel with workspace filters
- [ ] Implement filter persistence
- [ ] Add debounced search
- [ ] Create search result highlighting
- [ ] Implement performance optimizations

Documentation Required

- Search and filter system architecture
- Workspace scoping implementation
- Performance optimization techniques
- Search and filter patterns
- Accessibility implementation details
- Real-time search best practices

Testing Requirements

- Search functionality tests
- Filter logic tests
- Performance tests for large datasets
- Workspace scoping tests
- Accessibility compliance tests
- Real-time search behavior tests
- Cross-browser compatibility tests

Integration Points

- Integration with workspace context providers
- Search service integration
- Filter persistence system integration
- Performance monitoring integration
- Accessibility utilities integration

Deliverables

- Enhanced SearchBar with workspace scoping
- FilterPanel with workspace filters

- Performance optimization implementation
- Search and filter persistence
- Real-time search functionality
- Comprehensive Storybook stories

Component Specifications

typescript

```typescript
interface SearchBarProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  placeholder?: string
  value?: string
  onChange?: (value: string) => void
  onSearch?: (query: string) => void
  suggestions?: string[]
  recentSearches?: string[]
  debounceMs?: number
  loading?: boolean
  workspaceScoped?: boolean
  autoComplete?: boolean
  showRecentSearches?: boolean
}

interface FilterPanelProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  filters: FilterOption[]
  selectedFilters: Record<string, any>
  onFilterChange: (filters: Record<string, any>) => void
  onReset?: () => void
  persistent?: boolean
  workspaceScoped?: boolean
  loading?: boolean
  layout?: 'vertical' | 'horizontal' | 'grid'
  collapsible?: boolean
}

interface FilterOption {
  key: string
  label: string
  type: 'select' | 'multiselect' | 'range' | 'date' | 'boolean'
  options?: Array<{value: any, label: string}>
  workspaceContext?: 'consultant' | 'client' | 'admin' | 'neutral'
  defaultValue?: any
  required?: boolean
}
```

Story 3.1.3: Specialized Input Molecules

Overview

Build specialized input molecules for complex business data entry with workspace-specific validation and formatting.

AI Developer Prompt

You are building specialized input molecules for THE WHEEL design system. Building on the search and filter components from Story 3.1.2, you need to create sophisticated input combinations for specific business use cases.

Context

- Enhanced search and filter components with workspace context
- Need specialized inputs for complex business data
- Must support workspace-specific validation and formatting
- Specialized inputs handle unique business requirements
- Need consistent API across all input types

Requirements

1. Enhance TagInput with workspace tags:

- Workspace-specific tag suggestions
- Tag validation and formatting
- Tag autocomplete with workspace data
- Tag categorization and grouping
- Workspace permission-based tag management

2. Enhance DateRangePicker with workspace timezones:

- Multiple timezone support
- Workspace default timezone settings
- Calendar integration with workspace schedules
- Date range validation
- Responsive date range selection

3. Build new workspace-specific input molecules:

- AddressInput with international formatting
- PhoneInput with international phone formatting

- CurrencyInput with workspace currency settings
- TimeRangeInput for scheduling
- DocumentInput for file handling with workspace permissions

Specific Tasks

- [ ] Enhance TagInput with workspace tags
- [ ] Add tag autocomplete and validation
- [ ] Enhance DateRangePicker with timezones
- [ ] Build AddressInput component
- [ ] Build PhoneInput component
- [ ] Build CurrencyInput component
- [ ] Build TimeRangeInput component
- [ ] Create consistent validation API

Documentation Required

- Specialized input system architecture
- Workspace-specific validation patterns
- International formatting guidelines
- Input molecule composition patterns
- Accessibility implementation details
- Business logic integration guide

Testing Requirements

- Tag input functionality tests
- Date range validation tests
- International formatting tests
- Workspace context tests
- Accessibility compliance tests
- Input validation tests
- Cross-browser compatibility tests

Integration Points

- Integration with workspace context providers
- Validation system integration

- International formatting services
- Calendar and scheduling integration
- File handling system integration

Deliverables

- Enhanced TagInput with workspace tags
- DateRangePicker with timezone support
- AddressInput with international formatting
- PhoneInput with international formatting
- CurrencyInput with workspace currencies
- TimeRangeInput for scheduling
- Comprehensive Storybook stories

Component Specifications

typescript

```typescript
interface TagInputProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  value?: string[]
  onChange?: (tags: string[]) => void
  suggestions?: string[]
  placeholder?: string
  maxTags?: number
  allowCustom?: boolean
  workspaceScoped?: boolean
  tagValidation?: (tag: string) => boolean
  tagFormatter?: (tag: string) => string
  categories?: string[]
}

interface DateRangePickerProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  value?: [Date, Date]
  onChange?: (range: [Date, Date]) => void
  timezone?: string
  format?: string
  minDate?: Date
  maxDate?: Date
  workspaceTimezone?: string
  calendarIntegration?: boolean
  validationState?: 'error' | 'warning' | 'success'
  helperText?: string
  errorMessage?: string
}

interface AddressInputProps {
  context?: 'consultant' | 'client' | 'admin' | 'neutral'
  value?: Address
  onChange?: (address: Address) => void
  countries?: string[]
  defaultCountry?: string
  validation?: boolean
  autocomplete?: boolean
  validationState?: 'error' | 'warning' | 'success'
  helperText?: string
  errorMessage?: string
}
```

Story 3.1.4: File Upload Components

Overview

Create sophisticated file upload components with drag-and-drop functionality, real-time progress tracking, and workspace-specific file handling.

AI Developer Prompt

You are creating file upload components for THE WHEEL design system. Building on the existing form molecules, you need to create drag-and-drop file upload with real-time features.

Context

- Existing form molecule system with validation
- Real-time collaboration requiring file sharing
- Multiple workspace contexts with different file requirements
- Need for secure file upload with virus scanning
- Integration with existing asset management system

Requirements

1. Create file upload components:

- Drag-and-drop file upload area
- Multi-file upload with progress tracking
- Image upload with preview and cropping
- Document upload with format validation
- Bulk file upload with batch processing

2. Implement workspace context features:

- Context-specific file type restrictions
- Permission-based upload limits
- Workspace-specific file organization
- Brand-aware upload interface styling
- Context-aware file processing

3. Create advanced upload features:

- Resume interrupted uploads
- Real-time upload progress
- File compression and optimization
- Virus scanning integration
- Cloud storage integration

Specific Tasks

- [ ] Create FileUpload component
- [ ] Implement drag-and-drop functionality
- [ ] Set up progress tracking
- [ ] Create image upload with preview
- [ ] Implement file validation
- [ ] Set up real-time progress updates

Documentation Required

- File upload component API
- Security implementation guide
- File type and size restrictions
- Progress tracking documentation
- Error handling guide

Testing Requirements

- File upload functionality tests
- Drag-and-drop interaction tests
- Progress tracking tests
- Security validation tests
- Performance and large file tests

Integration Points

- Integration with asset management system
- Workspace context integration
- Real-time collaboration integration
- Security and virus scanning integration
- Cloud storage integration

Deliverables

- Complete file upload system
- Drag-and-drop functionality
- Progress tracking and resumption
- Security and validation features
- Comprehensive upload documentation

Performance Requirements

- Upload progress updates under 100ms
- File processing under 2 seconds
- Memory usage under 100MB
- Large file handling up to 1GB
- Upload speed optimization

Story 3.1.5: Multi-Step Form Components

Overview

Create multi-step form components with wizard-style navigation, progress tracking, and workspace-specific form flows.

AI Developer Prompt

You are creating multi-step form components for THE WHEEL design system. Building on the existing form molecules, you need to create wizard-style forms for complex workflows.

Context

- Existing form validation and input components
- Need for complex onboarding and setup workflows
- Multiple workspace contexts requiring different form flows
- Real-time collaboration for form completion
- Integration with existing form validation system

Requirements

1. Create multi-step form components:

- Step wizard with progress indication
- Form step validation and navigation
- Conditional step routing
- Form data persistence across steps
- Step completion validation

2. Implement workspace context features:

- Context-specific form flows
- Permission-based step access
- Workspace-specific validation rules
- Brand-aware form styling
- Context-aware form completion

3. Create advanced form features:

- Real-time form collaboration
- Auto-save and recovery
- Form branching and conditional logic
- Progress persistence across sessions
- Form analytics and completion tracking

Specific Tasks

- [ ] Create MultiStepForm component
- [ ] Implement StepWizard component
- [ ] Set up form step validation
- [ ] Create conditional routing
- [ ] Implement data persistence
- [ ] Set up progress tracking

Documentation Required

- Multi-step form implementation guide
- Step validation documentation
- Conditional logic configuration
- Data persistence guide
- Form analytics integration

Testing Requirements

- Multi-step form navigation tests
- Step validation tests
- Data persistence tests
- Conditional routing tests
- Form completion tests

Integration Points

- Integration with form validation system
- Workspace context integration
- Real-time collaboration integration
- Analytics and tracking integration
- Data persistence integration

Deliverables

- Complete multi-step form system
- Step wizard with progress tracking
- Conditional routing and validation
- Data persistence and recovery
- Comprehensive form documentation

Performance Requirements

- Step navigation under 200ms
- Form validation under 100ms
- Data persistence under 500ms
- Memory usage under 30MB
- Form completion tracking real-time

Timeline and Dependencies

Timeline

- Week 1: Story 3.1.1 - Form Field Components

- Week 1-2: Story 3.1.2 - Search & Filter Components
- Week 2-3: Story 3.1.3 - Specialized Input Molecules
- Week 3-4: Story 3.1.4 - File Upload Components
- Week 4: Story 3.1.5 - Multi-Step Form Components

Dependencies

- Epic 2.1 (Input Components) - Complete
- Epic 2.2 (Display Components) - Complete
- Epic 2.3 (Layout Components) - Complete
- Workspace context system operational
- Validation system established

Success Metrics

- All form molecules support workspace contexts
- 100% accessibility compliance (WCAG 2.1 AA)
- Form validation response under 100ms
- Complete test coverage (90%+ for all components)
- Comprehensive documentation for all patterns
- Zero regression in form functionality

Risk Mitigation

- Incremental form component development
- Thorough validation testing at each phase
- Regular accessibility audits
- Performance monitoring for complex forms
- Clear migration guides for existing forms
- User testing for multi-step workflows