

# Contract Management System - Complete Workflow & Functionality Documentation

## Table of Contents

1. [Application Overview](#)
2. [Complete Workflow](#)
3. [Core Functionality](#)
4. [Data Models](#)
5. [Status Management System](#)
6. [User Interface Features](#)
7. [Technical Implementation Details](#)

## Application Overview

The Contract Management System is a web-based application that enables users to create reusable contract templates (Blueprints) and generate individual contracts from those templates. The system manages the complete lifecycle of contracts from creation through signing and locking.

### Key Technologies:

- Next.js 16 (React 19)
- Tailwind CSS v4
- Zustand (State Management with localStorage persistence)
- TypeScript

**Storage:** All data is persisted in browser's localStorage (client-side only)

# Complete Workflow

## Phase 1: Blueprint Creation

### Step 1: Access Blueprint Creation

- **Route:** /blueprints
- **Action:** User navigates to Blueprint Management page
- **UI:** Form interface for creating new blueprints

### Step 2: Define Blueprint Details

1. **Blueprint Name** (Required)
  - User enters a descriptive name (e.g., "Employment Contract", "Service Agreement")
2. **Description** (Optional)
  - User can add a brief description of the template's purpose
3. **Header Image URL** (Optional)
  - User can provide a URL for a logo/header image to display on contracts

### Step 3: Add Fields to Blueprint

- **Field Types Available:**
  - **Text Input:** Multi-line text fields for entering text content
  - **Date Picker:** Date selection fields
  - **Signature:** Signature capture fields
  - **Checkbox:** Boolean checkboxes
- **Field Configuration:**
  - User enters a field label
  - User selects field type from dropdown
  - Clicks "Add Field" button
  - Field is added to the blueprint with:
    - Unique ID ( `field-{uuid}` )
    - Position index (for ordering)
    - Required flag (defaults to true)
- **Field Management:**
  - Fields are displayed in a list
  - User can remove fields before submission
  - Minimum 1 field required to create blueprint

## Step 4: Submit Blueprint

- **Validation:** Blueprint name and at least one field required
- **Action:** Blueprint is saved to Zustand store
- **Storage:** Persisted to localStorage
- **Redirect:** User is redirected to /contracts page
- **Notification:** Success toast message displayed

## Phase 2: Contract Generation

### Step 1: Access Contract Creation

- **Route:** /contracts OR /contracts/new
- **Action:** User navigates to Contracts page

### Step 2: Select Blueprint

- **UI:** Dropdown showing all available blueprints
- **Display:** Shows blueprint name and field count
- **Validation:** At least one blueprint must exist

### Step 3: Enter Contract Details

- **Contract Name** (Required)
  - User enters specific contract name (e.g., "Employment Contract - John Doe")
  - This differentiates individual contracts from the template

### Step 4: Preview Blueprint

- **Display:** Shows selected blueprint details:
  - Blueprint name
  - Description (if available)
  - Number of fields
  - List of all fields with their types

### Step 5: Create Contract

- **Process:**
  - i. System creates a new contract with:
    - Unique ID ( contract-{uuid} )

- Contract name
- Reference to blueprint (blueprintId, blueprintName, blueprintDescription)
- Status: "created" (initial status)
- Deep copy of all blueprint fields (independent from blueprint)
- Empty fieldValues object
- Timestamps (createdAt, updatedAt)
- ii. Contract is saved to Zustand store
- iii. Data persisted to localStorage
- iv. User redirected to contract edit page ( /contracts/{id}?edit=true )

## Phase 3: Contract Editing

### Step 1: Access Contract Editor

- **Route:** /contracts/{id}?edit=true
- **Prerequisite:** Contract status must be "created"
- **UI:** Document renderer in edit mode

### Step 2: Fill Contract Fields

- **Field Types & Interactions:**

#### Text Fields:

- Multi-line textarea input
- User can enter any text content
- Changes tracked in local state (unsaved changes indicator)

#### Date Fields:

- HTML5 date picker
- User selects date from calendar
- Stored as string in ISO format

#### Checkbox Fields:

- Boolean checkbox input
- User checks/unchecks
- Stored as boolean value

#### Signature Fields:

- Click-to-sign button
- On click, sets value to "signed"
- Visual indicator when signed

## Step 3: Reorder Fields (Optional)

- **Drag-and-Drop Functionality:**
  - Drag handle appears on hover (left side of field)
  - User drags field to new position
  - Visual feedback during drag (opacity change, border highlight)
  - Fields reordered by updating position property
  - Changes tracked in local state
- **Protection:**
  - Drag disabled when starting from input elements
  - Prevents accidental drags while typing

## Step 4: Save Changes

- **Action:** Click "Update" button
- **Process:**
  - i. Local field values and field order saved to contract
  - ii. Contract updated in store
  - iii. Changes persisted to localStorage
  - iv. Unsaved changes flag cleared
  - v. User redirected to dashboard
  - vi. Success notification displayed

## Step 5: Cancel Editing

- **Action:** Click "Cancel" button
- **Process:**
  - i. Local changes discarded
  - ii. Reverts to saved contract state
  - iii. Redirects to contract view page (read-only)
  - iv. Unsaved changes flag cleared

## Phase 4: Contract Status Management

### Status Flow Overview

The contract follows a linear progression through statuses:

Created → Approved → Sent → Signed → Locked

**Special Status:** Revoked (can be applied from Created or Sent )

## Status Transitions

### 1. Created Status (Initial)

- **Actions Available:**
  - Edit contract (fill fields, reorder)
  - Advance to "Approved"
  - Revoke contract

### 2. Approved Status

- **Actions Available:**
  - View contract (read-only)
  - Advance to "Sent"
  - Cannot edit or revoke

### 3. Sent Status

- **Actions Available:**
  - View contract (read-only)
  - Advance to "Signed"
  - Revoke contract

### 4. Signed Status

- **Actions Available:**
  - View contract (read-only)
  - Advance to "Locked"
  - Cannot revoke

### 5. Locked Status (Final)

- **Actions Available:**
  - View contract (read-only)
  - No further actions
  - Contract is immutable

### 6. Revoked Status (Terminal)

- **Actions Available:**
  - View contract (read-only)
  - No further actions
  - Contract cannot proceed

## Status Management Interface

- **Access:** Click "Manage Status" button on contract view page
- **Modal Display:**
  - Current status badge
  - Visual status flow diagram
  - Available action buttons
  - Revoke confirmation dialog (if applicable)

## Status Change Process

1. User clicks "Manage Status"
2. Modal opens showing current status and available transitions
3. User selects action (advance or revoke)
4. For revoke: Confirmation dialog appears
5. Status updated in store
6. Changes persisted to localStorage
7. Success notification displayed
8. Modal closes, page refreshes

## Phase 5: Contract Viewing & Management

### Read-Only View

- **Route:** `/contracts/{id}` (without `?edit=true` )
- **Display:**
  - Contract name as title
  - Blueprint name as subtitle
  - Current status badge
  - Document renderer in read-only mode
  - Creation date
  - Action buttons (Edit, Manage Status)

## Document Rendering

- **Components:**
  - Document title (centered, large)
  - Header image (if provided in blueprint)
  - Description (if available)
  - Sections (if defined in blueprint):
    - Section headers
    - Text content
    - Images
    - Fields (with values or "Not filled" placeholder)
  - Footer with generation date

## Field Display (Read-Only)

- **Text Fields:** Display entered text or "Not filled"
- **Date Fields:** Formatted date (e.g., "January 15, 2024") or "Not filled"
- **Checkbox Fields:** Checkmark (✓) if checked, empty if not
- **Signature Fields:** "✓ Signature captured" or "No signature"

## Phase 6: Dashboard & Analytics

### Dashboard Overview

- **Route:** /dashboard
- **Purpose:** Central hub for viewing all contracts and blueprints

### KPI Cards

Displays key metrics:

- **Total Contracts:** Count of all contracts
- **Total Blueprints:** Count of all blueprints
- **Active Contracts:** Contracts in "created", "approved", or "sent" status
- **Signed Contracts:** Contracts in "signed" or "locked" status

### Analytics Graphs

- **Contract Status Distribution:**
  - Pie chart showing breakdown by status

- Visual representation of contract lifecycle
- **Blueprint Usage:**
  - Bar chart showing how many contracts created from each blueprint
  - Identifies most-used templates

## Contracts List Section

- **View Toggle:** Switch between Contracts and Blueprints view
- **Search Functionality:**
  - Search by contract name or blueprint name
  - Real-time filtering
- **Status Filtering:**
  - Multi-select filter by contract status
  - Filter by: Created, Approved, Sent, Signed, Locked, Revoked
- **Field Type Filtering (Blueprints):**
  - Multi-select filter by field types
  - Filter by: Text, Date, Signature, Checkbox
- **Contract Cards Display:**
  - Contract name
  - Blueprint name
  - Status badge
  - Creation/update dates
  - Action buttons: View, Edit, Delete, Status Change (quick)
- **Blueprint Cards Display:**
  - Blueprint name
  - Description
  - Field count
  - Field types used
  - Action buttons: View, Edit, Delete
- **Sorting:**
  - Default: Most recently updated first
  - Automatic sorting by updatedAt timestamp

## Quick Actions from Dashboard

- **View:** Navigate to contract/blueprint detail page
- **Edit:** Navigate to edit mode (contracts only if status is "created")
- **Delete:** Opens confirmation dialog, then removes item
- **Status Change:** Quick status update dropdown (contracts only)

# Core Functionality

## 1. Blueprint Management

### Create Blueprint

- **Location:** /blueprints
- **Required Fields:**
  - Name (string)
  - At least one field
- **Optional Fields:**
  - Description (string)
  - Header Image URL (string)
- **Field Configuration:**
  - Add multiple fields of different types
  - Remove fields before submission
  - Fields assigned unique IDs and positions

### View Blueprint

- **Location:** /blueprints/{id}
- **Display:**
  - Blueprint details
  - Document preview (using DocumentRenderer)
  - Creation date
  - "Create Contract" button
  - "Edit" button

### Edit Blueprint

- **Location:** /blueprints/{id}?edit=true
- **Capabilities:**
  - Modify name, description, header image URL
  - Add new fields
  - Remove existing fields
  - Update field properties
- **Note:** Changes to blueprints do NOT affect existing contracts (deep copy architecture)

## Delete Blueprint

- **Location:** Dashboard or Blueprint list
- **Process:**
  - Confirmation dialog
  - Removes blueprint from store
  - Persisted to localStorage
  - Success notification

## 2. Contract Management

### Create Contract

- **Location:** /contracts OR /contracts/new
- **Process:**
  - i. Select blueprint from dropdown
  - ii. Enter contract name
  - iii. Preview blueprint details
  - iv. Click "Create Contract"
  - v. Redirected to contract editor

### Edit Contract

- **Location:** /contracts/{id}?edit=true
- **Prerequisites:**
  - Contract status must be "created"
  - Edit mode explicitly enabled via query parameter
- **Capabilities:**
  - Fill in field values
  - Reorder fields via drag-and-drop
  - Save changes
  - Cancel and discard changes

### View Contract

- **Location:** /contracts/{id}
- **Display:**
  - Read-only document view

- Current status
- All field values
- Action buttons (context-dependent)

## Delete Contract

- **Location:** Dashboard or Contract list
- **Process:**
  - Confirmation dialog
  - Removes contract from store
  - Persisted to localStorage
  - Success notification

## 3. Status Management

### Status Transition Rules

- **Linear Progression:** Can only advance one step at a time
- **Revoke:** Available from "created" or "sent" status
- **Locked:** Final state, no further changes
- **Revoked:** Terminal state, cannot proceed

### Status Change Methods

#### 1. From Contract View Page:

- Click "Manage Status" button
- Modal opens with status flow
- Select action

#### 2. From Dashboard:

- Quick status change dropdown
- Direct status update

#### 3. From Status Page:

- Dedicated route: `/contracts/{id}/status`
- Full status management interface

## 4. Document Rendering

### DocumentRenderer Component

- **Purpose:** Unified component for displaying contracts/blueprints
- **Modes:**
  - **Read-Only:** Display values, no editing
  - **Editable:** Interactive form with drag-and-drop

### Document Structure

1. **Title Section:** Centered, large font
2. **Header Image:** Optional logo/image
3. **Description:** Optional text content
4. **Sections:** (If defined in blueprint)
  - Section headers
  - Text blocks
  - Images
  - Fields
5. **Fields:** (If no sections defined)
  - All fields rendered in order
6. **Footer:** Generation date

### Section Types

- **Section:** Major heading with optional content
- **Text:** Paragraph text block
- **Image:** Image display with URL
- **Field:** Form field reference

## 5. Drag-and-Drop Reordering

### Implementation

- **Technology:** HTML5 Drag and Drop API
- **Trigger:** Drag handle (grip icon) on left side of field
- **Visual Feedback:**
  - Opacity reduction while dragging
  - Border highlight on drop target

- Smooth transitions

## Protection Mechanisms

- Drag disabled when starting from input elements
- Prevents accidental drags during typing
- Only draggable via dedicated handle

## Position Management

- Fields use `position` property (number)
- Positions updated on reorder
- Persisted with contract data

# 6. Search & Filtering

## Search Functionality

- **Contracts:** Search by contract name or blueprint name
- **Blueprints:** Search by blueprint name or description
- **Real-time:** Filters as user types
- **Case-insensitive:** Matching

## Filtering

- **Contract Status Filter:**
  - Multi-select checkboxes
  - Filter by: Created, Approved, Sent, Signed, Locked, Revoked
  - Can select multiple statuses
- **Field Type Filter (Blueprints):**
  - Multi-select checkboxes
  - Filter by: Text, Date, Signature, Checkbox
  - Shows blueprints containing selected field types

## Clear Filters

- Button to reset all filters and search
- Returns to unfiltered view

# 7. Data Persistence

## Storage Mechanism

- **Technology:** Zustand with persist middleware
- **Storage:** Browser localStorage
- **Key:** contract-management-storage

## Data Serialization

- Dates converted to ISO strings for storage
- Dates rehydrated to Date objects on load
- Sections array ensured on rehydration

## Data Structure

```
{
  blueprints: Blueprint[],
  contracts: Contract[]
}
```

# Data Models

## Blueprint

```
{
  id: string;           // Unique identifier (bp-{uuid})
  name: string;          // Blueprint name
  description?: string;  // Optional description
  fields: Field[];       // Array of field definitions
  sections: DocumentSection[]; // Document structure
  headerImageUrl?: string; // Optional header image URL
  createdAt: Date;       // Creation timestamp
  updatedAt: Date;       // Last update timestamp
}
```

# Contract

```
{
  id: string; // Unique identifier (contract-{uuid})
  name: string; // Contract name
  blueprintId: string; // Reference to source blueprint
  blueprintName: string; // Cached blueprint name
  blueprintDescription?: string; // Cached blueprint description
  status: ContractStatus; // Current status
  fields: Field[]; // Deep copy of blueprint fields
  fieldValues: Record<string, string | boolean | Date | null>; // Field values
  createdAt: Date; // Creation timestamp
  updatedAt: Date; // Last update timestamp
}
```

# Field

```
{
  id: string; // Unique identifier (field-{uuid})
  type: FieldType; // "text" | "date" | "signature" | "checkbox"
  label: string; // Field label
  position: number; // Order position
  required?: boolean; // Required flag
}
```

# DocumentSection

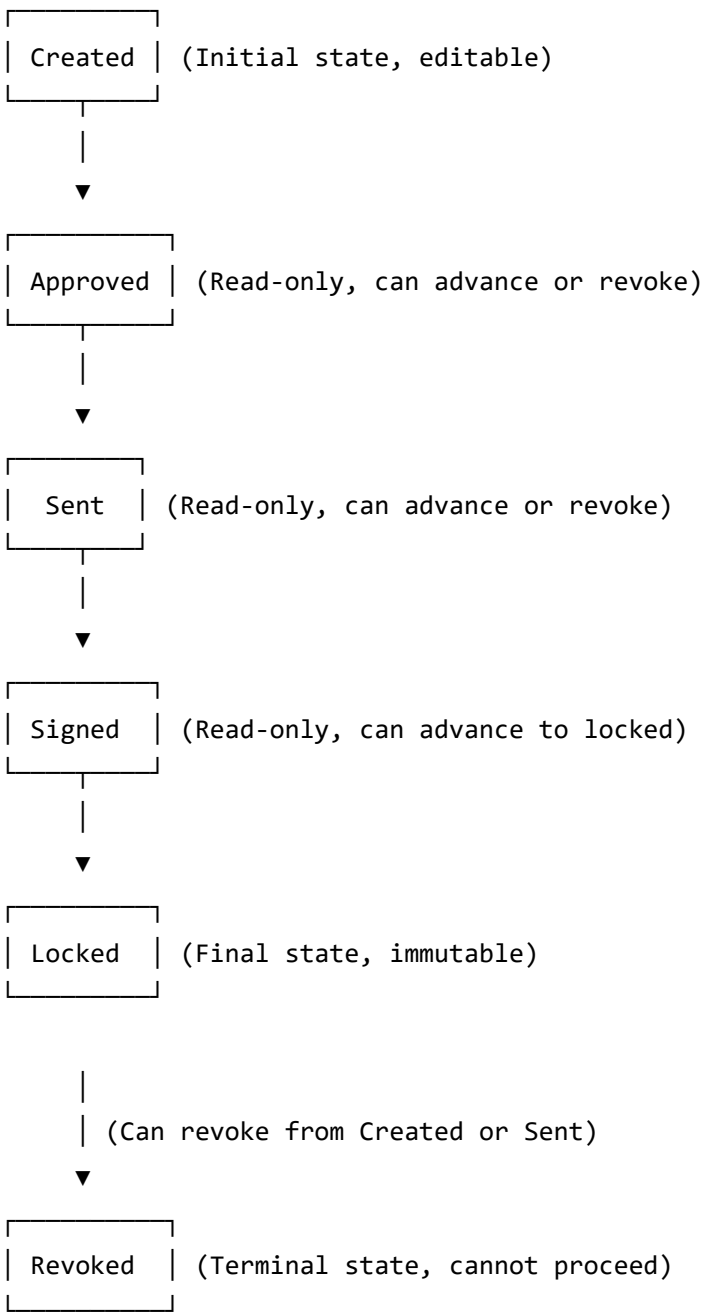
```
{
  id: string; // Unique identifier
  type: "section" | "text" | "image" | "field";
  title?: string; // Optional title
  content?: string; // Text content
  imageUrl?: string; // Image URL
  imageAlt?: string; // Image alt text
  fieldId?: string; // Reference to field (for field type)
  order: number; // Display order
}
```

# ContractStatus

"created" | "approved" | "sent" | "signed" | "locked" | "revoked"

## Status Management System

### Status Flow Diagram



# Status Transition Rules

## Valid Transitions

- created → approved ✓
- approved → sent ✓
- sent → signed ✓
- signed → locked ✓
- created → revoked ✓
- sent → revoked ✓

## Invalid Transitions

- Cannot skip statuses (e.g., created → sent X)
- Cannot go backwards (e.g., sent → approved X)
- Cannot revoke from approved , signed , or locked X
- Cannot transition from revoked or locked X

## Status Functions

- `canTransitionTo(current, target)` : Validates if transition is allowed
- `getNextStatus(current)` : Returns next status in flow
- `getStatusLabel(status)` : Returns human-readable label
- `getStatusGroup(status)` : Groups statuses (active/pending/signed)

# User Interface Features

## Navigation

- **Top Navigation Bar:**
  - Home (Landing page)
  - Dashboard
  - Blueprints
  - Contracts
  - Theme toggle (light/dark mode)

## Responsive Design

- Mobile-first approach
- Breakpoints: sm, md, lg, xl
- Adaptive layouts for all screen sizes
- Touch-friendly interactions

## Toast Notifications

- Success notifications (green)
- Error notifications (red)
- Warning notifications (yellow)
- Auto-dismiss after timeout

## Confirmation Dialogs

- Delete confirmation for contracts and blueprints
- Revoke confirmation for contracts
- Prevents accidental deletions

## Loading States

- Suspense boundaries for async routes
- Loading indicators during data operations

## Error Handling

- 404 pages for missing contracts/blueprints
- Validation error messages
- User-friendly error displays

## Technical Implementation Details

### State Management

- **Store:** Zustand with TypeScript
- **Persistence:** localStorage via persist middleware

- **Actions:**
  - Blueprint CRUD operations
  - Contract CRUD operations
  - Status updates
  - Queries (get by ID, filter by status)

## Routing

- **Framework:** Next.js App Router
- **Dynamic Routes:**
  - /blueprints/[id] - Blueprint detail/edit
  - /contracts/[id] - Contract detail/edit
  - /contracts/[id]/status - Status management
- **Query Parameters:**
  - ?edit=true - Enable edit mode
  - ?blueprintId={id} - Pre-select blueprint

## Component Architecture

- **Reusable Components:**
  - DocumentRenderer (core document engine)
  - UI components (Button, Card, Input, etc.)
  - Shared components (DeleteDialog, SearchFilter)
- **Feature Components:**
  - BlueprintForm
  - ContractCard
  - BlueprintCard
  - Dashboard sections

## Data Flow

1. **User Action** → Component Event Handler
2. **Handler** → Zustand Store Action
3. **Store** → Update State + Persist to localStorage
4. **Store** → Trigger Re-render
5. **Component** → Display Updated State

# Deep Copy Architecture

- Contracts maintain independent copy of fields
- Blueprint changes don't affect existing contracts
- Ensures data integrity and isolation

## Field Position System

- Fields use explicit `position` property
- Not dependent on array index
- Enables reliable drag-and-drop
- Persists across sessions

# Complete User Journey Example

## Scenario: Creating and Managing an Employment Contract

### 1. Create Blueprint:

- Navigate to `/blueprints`
- Enter name: "Employment Contract"
- Add description: "Standard employment agreement template"
- Add fields:
  - Employee Name (text)
  - Start Date (date)
  - Salary (text)
  - Benefits (checkbox)
  - Employee Signature (signature)
- Submit blueprint

### 2. Generate Contract:

- Navigate to `/contracts`
- Select "Employment Contract" blueprint
- Enter contract name: "Employment Contract - John Doe"
- Click "Create Contract"
- Redirected to editor

### 3. Fill Contract:

- Enter "John Doe" in Employee Name

- Select "2024-01-15" for Start Date
- Enter "\$75,000" in Salary
- Check Benefits checkbox
- Click "Click to Sign" for signature
- Click "Update" to save

#### 4. **Manage Status:**

- View contract (status: "created")
- Click "Manage Status"
- Advance to "approved"
- Advance to "sent"
- Advance to "signed"
- Advance to "locked"
- Contract is now immutable

#### 5. **View on Dashboard:**

- Navigate to /dashboard
- See contract in "Signed Contracts" KPI
- View in contracts list with "Locked" status
- Search and filter capabilities available

## Summary

This Contract Management System provides a complete workflow from template creation to contract finalization. The system emphasizes:

- **Template Reusability:** Create blueprints once, use many times
- **Data Independence:** Contracts are isolated from blueprint changes
- **Status Control:** Enforced workflow with clear status transitions
- **User Experience:** Intuitive interface with drag-and-drop, search, and filtering
- **Data Persistence:** All data saved locally in browser
- **Flexibility:** Support for multiple field types and document structures

The application is designed as a single-user, client-side tool that requires no backend infrastructure, making it easy to deploy and use.