# **Logo Builder Frontend - Requirements Specification**

## **Document Overview**

**Document Type:** Requirements Specification

**Target System:** Logo Builder Frontend Application

**Architecture:** React-based Single Page Application

**Optimization:** RAG (Retrieval-Augmented Generation) Compatible

**Version:** 1.0

**Last Updated:** 2024

## **Executive Summary**

The Logo Builder Frontend is a comprehensive web application designed to provide users with an intuitive, powerful platform for creating, editing, and managing custom logos. The system supports both authenticated and guest users, offering a range of logo creation tools, templates, and customization options.

## **Functional Requirements**

### **FR-001: User Authentication and Authorization**

#### **FR-001.1: User Login**

**Requirement:** Users must be able to authenticate using JWT-based authentication

**Implementation:** src/helpers/authentication-helper.js

**API Endpoint:** /v1.0/token/me

**Components:** FreeFlowAuthenticationModal

**Acceptance Criteria:**

Users can log in with valid credentials

JWT tokens are securely stored and managed

Session persistence across browser sessions

Automatic token refresh when expired

#### **FR-001.2: Guest User Support**

**Requirement:** Anonymous users must be able to create temporary logos

**Implementation:** src/clients/temporary-logo.client.js

**API Endpoint:** /v1.0/temporary-logos/save

**Flow:** Guest creation → Account signup → Logo transfer

**Acceptance Criteria:**

Temporary logos created without authentication

Seamless transfer to permanent account upon signup

Data persistence during guest session

### **FR-002: Logo Creation and Editing**

#### **FR-002.1: Logo Editor Interface**

**Requirement:** Comprehensive logo editing interface with real-time preview

**Implementation:** src/logomaker/index.js (387 lines)

**Components:** LogoMakerEditor, LogoArea, ControlsAccordion

**Features:**

Segment-based editing (name, slogan, symbol, backgrounds)

Real-time preview updates

Undo/redo functionality

Template-based starting points

#### **FR-002.2: Text Customization**

**Requirement:** Users must be able to customize text elements

**Implementation:** src/components/controls/name-controls.js, src/components/controls/slogan-controls.js

**Features:**

Font family selection from curated library

Font size, spacing, and styling adjustments

Text color customization

Text effects and decorations

#### **FR-002.3: Symbol and Icon Management**

**Requirement:** Extensive symbol library with customization options

**Implementation:** src/components/controls/symbol-controls.js

**Features:**

Searchable symbol library

Symbol positioning and sizing

Color and style modifications

Custom symbol upload (future enhancement)

#### **FR-002.4: Background and Layout Options**

**Requirement:** Flexible background and layout customization

**Implementation:** src/components/controls/background-controls.js

**Features:**

Solid colors, gradients, and patterns

Layout templates and arrangements

Card-based logo designs

Responsive layout adjustments

### **FR-003: Logo Management**

#### **FR-003.1: Logo Storage and Retrieval**

**Requirement:** Persistent logo storage with user access

**Implementation:** src/clients/logo.client.js

**API Endpoints:**

GET /v1.0/logos/list?include\_logo=true

GET /v1.0/logos/{logoId}/load

**Components:** MyLogosGrid, LogoTile

**Acceptance Criteria:**

Users can view all their created logos

Logo metadata and thumbnails displayed

Quick access to edit or download

#### **FR-003.2: Logo Operations**

**Requirement:** Complete CRUD operations for logos

**API Endpoints:**

POST /v1.0/logos/save - Create new logo

POST /v1.0/logos/{logoId}/save - Update existing

DELETE /v1.0/logos/{logoId}/delete - Delete logo

POST /v1.0/logos/{logoId}/duplicate - Duplicate logo

**Features:**

Save draft versions

Duplicate existing logos

Delete unwanted logos

Version history tracking

### **FR-004: Logo Export and Download**

#### **FR-004.1: Download Functionality**

**Requirement:** Multiple format export options

**Implementation:** src/logomaker/components/download-button.js

**Route:** /editor/:logoId/download

**Supported Formats:**

PNG (high resolution)

SVG (vector format)

PDF (print ready)

**Quality Options:**

Web resolution (72 DPI)

Print resolution (300 DPI)

Custom dimensions

#### **FR-004.2: Preview and Sharing**

**Requirement:** Logo preview and sharing capabilities

**Implementation:** src/modules/editor/containers/logo-preview.js

**Route:** /editor/:logoId/preview

**Features:**

Full-screen preview mode

Social media format previews

Shareable preview links

Merchandise mockups

### **FR-005: Template and Variation System**

#### **FR-005.1: Template Library**

**Requirement:** Curated template collection for quick starts

**Implementation:** src/core/components/grid.js

**Features:**

Industry-specific templates

Style-based categorization

Template search and filtering

Customizable starting points

#### **FR-005.2: Logo Variations**

**Requirement:** Automatic generation of logo variations

**Implementation:** src/components/variations-area/variations-area.js

**Features:**

Color scheme variations

Layout alternatives

Style modifications

A/B testing support

### **FR-006: User Onboarding**

#### **FR-006.1: Guided Onboarding Flow**

**Requirement:** Step-by-step user guidance for first-time users

**Implementation:** src/modules/onboarding/containers/stateful-onboarding.js

**Route:** /onboarding/:currentStep/:logoId

**Components:** BottomNavWithProgress, Selection

**Steps:**

Business information collection

Style preference selection

Template recommendation

Initial customization

Account creation (optional)

#### **FR-006.2: Progressive Disclosure**

**Requirement:** Gradual feature introduction to prevent overwhelm

**Implementation:** Stepper component with context providers

**Features:**

Step-by-step progression

Progress indicators

Skip options for experienced users

Help and guidance tooltips

### **FR-007: Account Limitations and Restrictions**

#### **FR-007.1: Subscription-Based Features**

**Requirement:** Feature gating based on account type

**Implementation:** src/clients/limitations.client.js

**API Endpoint:** /v2.0/limitation

**Limitations:**

Number of logos per account

Download quality restrictions

Advanced feature access

Commercial usage rights

#### **FR-007.2: Upgrade Prompts**

**Requirement:** Contextual upgrade suggestions

**Implementation:** src/logomaker/components/pricing-plan-modal.js

**Features:**

Feature-specific upgrade prompts

Pricing plan comparisons

Seamless upgrade flow

Trial period management

## **Non-Functional Requirements**

### **NFR-001: Performance Requirements**

#### **NFR-001.1: Page Load Performance**

**Requirement:** Initial page load under 3 seconds

**Implementation:** Code splitting with React.lazy()

**Metrics:**

First Contentful Paint < 1.5s

Largest Contentful Paint < 2.5s

Time to Interactive < 3s

**Optimization Techniques:**

Webpack bundle optimization

Image lazy loading

CDN asset delivery

#### **NFR-001.2: Editor Responsiveness**

**Requirement:** Real-time editing with minimal latency

**Implementation:** Optimized rendering with React.memo

**Metrics:**

Edit action response < 100ms

Preview update < 200ms

Smooth 60fps animations

**Techniques:**

Canvas-based rendering

Debounced updates

Virtual scrolling for large lists

### **NFR-002: Scalability Requirements**

#### **NFR-002.1: User Concurrency**

**Requirement:** Support for 10,000+ concurrent users

**Implementation:** Stateless frontend with API-based backend

**Architecture:**

Micro-frontend modules for independent scaling

CDN distribution for static assets

API rate limiting and caching

#### **NFR-002.2: Data Volume**

**Requirement:** Handle large logo libraries and templates

**Implementation:** Pagination and virtual scrolling

**Features:**

Infinite scroll for logo grids

Lazy loading of template data

Efficient caching strategies

### **NFR-003: Security Requirements**

#### **NFR-003.1: Data Protection**

**Requirement:** Secure handling of user data and logos

**Implementation:** HTTPS communication, JWT authentication

**Security Measures:**

End-to-end encryption for API calls

Secure token storage

Input sanitization and validation

CSRF protection

#### **NFR-003.2: Access Control**

**Requirement:** Proper authorization for user resources

**Implementation:** Resource-level permissions

**Features:**

User-specific logo access

Role-based feature access

Secure file upload/download

### **NFR-004: Usability Requirements**

#### **NFR-004.1: Responsive Design**

**Requirement:** Optimal experience across all device types

**Implementation:** Material-UI responsive components

**Breakpoints:**

Mobile: < 768px

Tablet: 768px - 1024px

Desktop: > 1024px

**Features:**

Touch-optimized controls

Adaptive layouts

Mobile-specific navigation

#### **NFR-004.2: Accessibility**

**Requirement:** WCAG 2.1 AA compliance

**Implementation:** Semantic HTML, ARIA labels

**Features:**

Keyboard navigation support

Screen reader compatibility

High contrast mode

Focus management

### **NFR-005: Reliability Requirements**

#### **NFR-005.1: Availability**

**Requirement:** 99.9% uptime availability

**Implementation:** Robust error handling and fallbacks

**Features:**

Graceful degradation

Offline capability (limited)

Error boundaries

Retry mechanisms

#### **NFR-005.2: Data Integrity**

**Requirement:** Prevent data loss during editing

**Implementation:** Auto-save and version control

**Features:**

Periodic auto-save

Conflict resolution

Backup and recovery

Transaction rollback

## **Technical Requirements**

### **TR-001: Browser Compatibility**

#### **TR-001.1: Supported Browsers**

**Chrome:** Version 90+

**Firefox:** Version 88+

**Safari:** Version 14+

**Edge:** Version 90+

**Mobile Safari:** iOS 14+

**Chrome Mobile:** Android 10+

#### **TR-001.2: Progressive Enhancement**

**Core Functionality:** Works without JavaScript (limited)

**Enhanced Experience:** Full features with modern browsers

**Polyfills:** Support for older browser features

### **TR-002: Integration Requirements**

#### **TR-002.1: API Integration**

**Logo Management API:** RESTful API for CRUD operations

**Authentication API:** JWT-based authentication service

**Storage API:** File upload and download services

**Analytics API:** User behavior tracking

#### **TR-002.2: Third-Party Services**

**CDN:** Asset delivery and caching

**Analytics:** Application Insights integration

**Error Tracking:** Automated error reporting

**Payment Processing:** Subscription management

### **TR-003: Development Requirements**

#### **TR-003.1: Code Quality**

**ESLint:** Code linting and style enforcement

**Prettier:** Code formatting

**TypeScript:** Type safety (gradual adoption)

**Testing:** Jest, React Testing Library

#### **TR-003.2: Build and Deployment**

**Webpack:** Module bundling and optimization

**Babel:** JavaScript transpilation

**CI/CD:** Bitbucket Pipelines

**Environment Management:** Multi-stage deployments

## **User Stories and Acceptance Criteria**

### **Epic: Logo Creation**

**As a user, I want to create professional logos easily so that I can establish my brand identity.**

#### **Story: LOGO-001 - Basic Logo Creation**

**As a new user, I want to create my first logo using templates so that I can get started quickly.**

**Acceptance Criteria:**

Given I am on the homepage

When I click "Create Logo"

Then I should see template options

And I can select a template to start editing

And the editor loads with my selected template

**Implementation Files:**

src/modules/onboarding/containers/stateful-onboarding.js

src/core/components/grid.js

src/logomaker/index.js

#### **Story: LOGO-002 - Text Customization**

**As a user, I want to customize the text in my logo so that it reflects my brand name.**

**Acceptance Criteria:**

Given I am in the logo editor

When I click on text elements

Then I should see text editing controls

And I can change font, size, color, and effects

And changes appear in real-time

**Implementation Files:**

src/components/controls/name-controls.js

src/components/controls/font-controls.js

src/logomaker/components/controls-accordion.js

### **Epic: Logo Management**

**As a user, I want to manage my logo collection so that I can organize and access my designs.**

#### **Story: LOGO-003 - My Logos Dashboard**

**As a returning user, I want to see all my saved logos so that I can continue working on them.**

**Acceptance Criteria:**

Given I am logged in

When I visit the dashboard

Then I should see all my saved logos

And I can click to edit, duplicate, or delete

And I can search and filter my logos

**Implementation Files:**

src/modules/my-logos/components/my-logos-page.js

src/components/logo-tile/logo-tile.js

src/clients/logo.client.js

## **Data Models and Schemas**

### **Logo Data Model**

{  
 id: "string",  
 name: "string",  
 logo: "string", // JSON serialized logo template  
 userId: "string",  
 createdAt: "datetime",  
 updatedAt: "datetime",  
 isPublic: "boolean",  
 downloadCount: "number",  
 metadata: {  
 industry: "string",  
 style: "string",  
 colors: ["string"],  
 fonts: ["string"]  
 }  
}

### **Template Data Model**

{  
 logoVersion: "number",  
 text: {  
 brandName: "string",  
 slogan: "string",  
 initials: "string"  
 },  
 font: {  
 brand1: {  
 id: "string",  
 size: "number",  
 lineSpacing: "number",  
 letterSpacing: "number",  
 color: "string"  
 }  
 },  
 layout: {  
 decoration: "object",  
 brand: "object",  
 slogan: "object",  
 symbol: "object"  
 },  
 symbol: "object",  
 colors: ["string"],  
 dimensions: {  
 width: "number",  
 height: "number"  
 }  
}

## **API Specifications**

### **Logo Management Endpoints**

#### **GET /v1.0/logos/list**

**Purpose:** Retrieve user's logo collection

**Authentication:** Required

**Parameters:**

include\_logo (boolean) - Include full logo data

page (number) - Pagination page

limit (number) - Items per page

**Response:**

{  
 "logos": [  
 {  
 "id": "logo-123",  
 "name": "My Company Logo",  
 "logo": "{...}",  
 "createdAt": "2024-01-01T00:00:00Z",  
 "updatedAt": "2024-01-01T00:00:00Z"  
 }  
 ],  
 "total": 25,  
 "page": 1,  
 "limit": 10  
}

#### **POST /v1.0/logos/save**

**Purpose:** Create new logo

**Authentication:** Required

**Request Body:**

{  
 "name": "My New Logo",  
 "logo": "{...}",  
 "metadata": {  
 "industry": "technology",  
 "style": "modern"  
 }  
}

**Response:**

{  
 "id": "logo-456",  
 "name": "My New Logo",  
 "createdAt": "2024-01-01T00:00:00Z"  
}

## **Error Handling Specifications**

### **Error Categories**

#### **Client Errors (4xx)**

**400 Bad Request:** Invalid logo data format

**401 Unauthorized:** Missing or invalid authentication

**403 Forbidden:** Account limitations exceeded

**404 Not Found:** Logo not found

**429 Too Many Requests:** Rate limit exceeded

#### **Server Errors (5xx)**

**500 Internal Server Error:** Unexpected server error

**502 Bad Gateway:** Upstream service unavailable

**503 Service Unavailable:** Temporary service outage

### **Error Response Format**

{  
 "error": {  
 "code": "LOGO\_NOT\_FOUND",  
 "message": "The requested logo could not be found",  
 "details": {  
 "logoId": "logo-123",  
 "userId": "user-456"  
 },  
 "timestamp": "2024-01-01T00:00:00Z"  
 }  
}

## **Testing Requirements**

### **Unit Testing**

**Coverage Target:** 80% code coverage

**Framework:** Jest with React Testing Library

**Focus Areas:**

Component rendering

User interactions

State management

Utility functions

### **Integration Testing**

**API Integration:** Mock API responses

**Component Integration:** Multi-component workflows

**State Integration:** Redux store interactions

### **End-to-End Testing**

**User Workflows:** Complete user journeys

**Cross-Browser:** Major browser compatibility

**Performance:** Load time and responsiveness

## **Deployment and Infrastructure**

### **Environment Configuration**

**Development:** Local development with hot reload

**UAT:** User acceptance testing environment

**Production:** Live production environment

### **Build Pipeline**

**Code Quality:** ESLint, Prettier, type checking

**Testing:** Unit, integration, and E2E tests

**Building:** Webpack optimization and bundling

**Deployment:** Automated deployment to target environment

### **Monitoring and Analytics**

**Performance Monitoring:** Application Insights

**Error Tracking:** Automated error reporting

**User Analytics:** Usage patterns and behavior

**Business Metrics:** Conversion rates and engagement

## **Conclusion**

This requirements specification provides a comprehensive foundation for the Logo Builder Frontend application, optimized for RAG system integration. The document covers all functional and non-functional requirements, technical specifications, and implementation details necessary for development, testing, and maintenance of the application.

The modular structure and detailed cross-references enable efficient information retrieval and support both human developers and AI-assisted development workflows.