# SaaS Landing Page - Technical Documentation

## Overview

This documentation provides a comprehensive guide to the AI SaaS Platform landing page, including architecture, components, styling, and deployment information.

## Architecture

### Technology Stack

- **Frontend Framework:** React 18.x with JSX
- **Build Tool:** Vite 6.x for fast development and optimized builds
- **Styling:** Tailwind CSS with custom CSS for specialized components
- **Animations:** Framer Motion for smooth transitions and interactions
- **Icons:** Lucide React for consistent iconography
- **Deployment:** Static site deployment optimized for CDN delivery

### Project Structure

Plain Text

```
src/
├── components/           # Reusable React components
│   ├── CustomCursor.jsx     # Custom cursor implementation
│   ├── LoadingScreen.jsx    # Initial loading animation
│   ├── Navigation.jsx       # Main navigation bar
│   ├── Footer.jsx          # Site footer
│   ├── HeroSection.jsx     # Landing page hero
│   ├── FeaturesSection.jsx # Features showcase
│   ├── PricingCalculator.jsx # Interactive pricing tool
│   └── BlogSection.jsx     # Blog/articles section
├── assets/              # Static assets (images, etc.)
├── App.jsx             # Main application component
├── App.css             # Global styles and theme definitions
├── index.css            # Base styles
└── main.jsx             # Application entry point
```

## Component Documentation

### CustomCursor Component

**Purpose:** Implements a custom elastic cursor with a red control point.

**Features:**

- Tracks mouse movement across the entire viewport
- Scales up on hover over interactive elements
- Smooth transitions using CSS transforms
- Z-index management to stay above all content

**Implementation:**

JSX

```jsx
const CustomCursor = () => {
  const [position, setPosition] = useState({ x: 0, y: 0 });
  const [isHovering, setIsHovering] = useState(false);
  // ... implementation details
};
```

### LoadingScreen Component

**Purpose:** Displays an animated loading screen with bouncing vertical lines.

**Features:**

- Four animated vertical lines with staggered timing
- Automatic dismissal after 2 seconds
- Smooth fade-out transition

- Callback function for completion handling

## Navigation Component

**Purpose:** Responsive navigation bar with theme toggle and page routing.

**Features:**

- Curved corner navigation buttons with hover effects
- Active page indication with underline and glow
- Theme toggle with smooth animation
- Responsive design for mobile devices
- Backdrop blur effect for modern appearance

## PricingCalculator Component

**Purpose:** Interactive pricing calculator with real-time updates.

**Features:**

- User count slider (1-100 users)
- Plan type selection (Starter, Standard, Premium)
- Billing cycle toggle (Monthly/Yearly with 20% discount)
- Real-time price calculation
- Animated price updates
- Plan comparison cards

**Pricing Logic:**

JavaScript

```javascript
const calculatePrice = () => {
  const basePrice = basePrices[features][billing];
  const userCost = users * userMultipliers[features];
  const total = basePrice + userCost;
  return billing === 'yearly' ? total * 0.8 : total;
};
```

### BlogSection Component

**Purpose:** Displays blog articles with AI-generated thumbnails and demo section.

**Features:**

- Featured demo section with presenter image
- Three blog article cards with metadata
- Responsive grid layout
- Hover animations and transitions
- Read time and publication date display

## Styling System

### Color Scheme

The application uses a light red and white theme with the following color variables:

**Light Mode:**

- Primary: `oklch(0.65 0.15 15)` (Light red)
- Background: `oklch(0.98 0.01 15)` (Off-white)
- Foreground: `oklch(0.145 0 0)` (Dark text)

**Dark Mode:**

- Primary: `oklch(0.75 0.15 15)` (Brighter red)
- Background: `oklch(0.08 0.01 15)` (Dark background)
- Foreground: `oklch(0.985 0 0)` (Light text)

## Custom CSS Classes

### Cursor Styles

CSS

```css
.custom-cursor {
  position: fixed;
  width: 40px;
  height: 40px;
  background: rgba(239, 68, 68, 0.1);
  border: 2px solid #ef4444;
  border-radius: 50%;
  pointer-events: none;
  z-index: 9999;
  transition: transform 0.1s ease;
}
```

### Navigation Styles

CSS

```css
.nav-button {
  position: relative;
  padding: 8px 16px;
  border-radius: 20px;
  transition: all 0.3s ease;
}

.nav-button:hover {
  background: rgba(239, 68, 68, 0.1);
  border-color: #ef4444;
  box-shadow: 0 0 20px rgba(239, 68, 68, 0.3);
}
```

### Button Styles

CSS

```css
.cta-button {
  background: linear-gradient(135deg, #ef4444, #dc2626);
  color: white;
  padding: 12px 24px;
  border-radius: 25px;
  transition: all 0.3s ease;
  box-shadow: 0 4px 15px rgba(239, 68, 68, 0.3);
}
```

## Animation System

### Framer Motion Implementation

The application uses Framer Motion for smooth animations and transitions:

#### Page Entry Animation

```jsx
const containerVariants = {
  hidden: { opacity: 0 },
  visible: {
    opacity: 1,
    transition: {
      delayChildren: 0.3,
      staggerChildren: 0.2
    }
  }
};
```

#### Hover Animations

```jsx
whileHover={{ scale: 1.05 }}
whileTap={{ scale: 0.95 }}
transition={{ type: "spring", stiffness: 400, damping: 10 }}
```

### CSS Animations

#### Loading Animation

```css
```

```
@keyframes bounce {
  0%, 80%, 100% {
    transform: scaleY(0.4);
  }
  40% {
    transform: scaleY(1);
  }
}
```

## State Management

### Theme Management

JSX

```jsx
const [isDark, setIsDark] = useState(false);

const handleThemeToggle = () => {
  setIsDark(!isDark);
  document.documentElement.classList.toggle('dark');
};
```

### Page Navigation

JSX

```jsx
const [currentPage, setCurrentPage] = useState('home');

const renderPage = () => {
  switch (currentPage) {
    case 'home': return <HomePage />;
    case 'features': return <FeaturesSection />;
    // ... other cases
  }
};
```

## Performance Optimizations

### Image Optimization

- All images are optimized during the build process
- Responsive image loading with appropriate sizes
- Lazy loading for images below the fold

### Code Splitting

- Components are organized for optimal bundling
- Dynamic imports where appropriate
- Tree shaking enabled for unused code elimination

### CSS Optimization

- Tailwind CSS purging removes unused styles
- Critical CSS inlined for faster initial render
- Custom CSS minimized and compressed

## Responsive Design

### Breakpoints

- Mobile: `< 768px`
- Tablet: `768px - 1024px`
- Desktop: `> 1024px`

### Grid System

JSX

```jsx
// Example responsive grid
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
```

### Typography Scale

```css
/* Mobile-first approach */
.text-4xl { font-size: 2.25rem; }
.md:text-5xl { font-size: 3rem; }
.lg:text-6xl { font-size: 3.75rem; }
```

## Deployment

### Build Process

1. **Development:** `npm run dev` - Starts Vite dev server
2. **Build:** `npm run build` - Creates optimized production build
3. **Preview:** `npm run preview` - Preview production build locally

### Production Optimizations

- Asset compression and minification
- CSS purging and optimization
- JavaScript bundling and tree shaking
- Image optimization and format conversion

### Deployment Checklist

- [ ] All images optimized and compressed
- [ ] CSS purged of unused styles
- [ ] JavaScript minified and bundled
- [ ] Meta tags and SEO optimized
- [ ] Performance metrics validated

- [ ] Cross-browser compatibility tested
- [ ] Mobile responsiveness verified

## Browser Support

- **Modern Browsers:** Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- **Mobile Browsers:** iOS Safari 14+, Chrome Mobile 90+
- **Features Used:** CSS Grid, Flexbox, CSS Custom Properties, ES6+ JavaScript

## Accessibility

### ARIA Labels

```jsx
<button aria-label="Toggle theme">
<img alt="AI Platform Demo Presenter" />
```

### Keyboard Navigation

- All interactive elements are keyboard accessible
- Focus indicators visible and styled
- Tab order logical and intuitive

### Screen Reader Support

- Semantic HTML structure
- Descriptive alt text for images
- Proper heading hierarchy

## Performance Metrics

### Target Metrics

- **First Contentful Paint:** < 1.5s
- **Largest Contentful Paint:** < 2.5s
- **Cumulative Layout Shift:** < 0.1
- **First Input Delay:** < 100ms

### Optimization Techniques

- Critical CSS inlining
- Image lazy loading
- Component code splitting
- Asset preloading for above-the-fold content