Component Documentation

Version: 0.1.0
Generated: 7/22/2025

Component Overview

This document provides detailed information about the key components used in the Blog Web Application. The application follows a component-based architecture with atomic design principles, organizing components into atoms, molecules, organisms, templates, and pages.

Component Hierarchy Diagram:

Component Organization

Components are organized following atomic design principles:

- Atoms: Basic building blocks like Button, Input, Icon, Typography
- Molecules: Simple combinations of atoms like SearchBar, FormField, Card
- Organisms: Complex UI sections like Header, Sidebar, CommentSection
- Templates: Page layouts that arrange organisms into a complete page structure
- Pages: Specific instances of templates that present actual content

Authentication Components

Login Component

Path: src/components/AccountIssue/Login/index.jsx
Purpose: Handles user login functionality with form validation and error handling
Key Features:

- Supports email/username and password authentication
- Integrates with OAuth providers (Google, Apple, Twitter)
- Implements client-side validation before submission
- Handles and displays server errors appropriately
- Supports "Remember me" functionality
- Includes password visibility toggle

```
  redirectPath="/dashboard"
  onLoginSuccess={() => trackAnalytics("user_login")}
  showRememberMe={true}
  showSocialLogin={true}
/>
```

SignUp Component

Path: src/components/AccountIssue/SignUp/index.jsx
Purpose: Handles new user registration with multi-step form process
Key Features:

- Multi-step registration flow (account details, profile setup, preferences)
- Progressive form validation at each step
- Password strength indicator
- Username availability checker
- Terms & conditions acceptance
- Email verification integration
- Profile setup guidance

Props:

```
{
  onSignupComplete: PropTypes.func, // Callback when signup is successful
  referralCode: PropTypes.string, // Optional referral code
  initialStep: PropTypes.number, // Which step to start on (default: 1)
  analyticsTracker: PropTypes.func // Function to track signup steps
}
```

State:

```
{
  currentStep: number,
  formValues: {
    username: string,
    email: string,
    password: string,
    confirmPassword: string,
    firstName: string,
    lastName: string,
    termsAccepted: boolean
  },
  formErrors: { ... }, // Validation errors
  isSubmitting: boolean,
  isUsernameTaken: boolean,
  passwordStrength: number, // 0-5 scale
  serverError: string
}
```

Animation Components

Aurora Component

Path: src/Animations/Aurora/Aurora.jsx
Purpose: Provides a dynamic Aurora Borealis effect as a background visual
Implementation: Uses Canvas API with WebGL for performant particle animations
Key Features:

- Responsive design that adapts to container dimensions
- Configurable colors, intensity, and animation speed
- Performance optimized with requestAnimationFrame
- Automatic pause when not in viewport to save resources
- Fallback static gradient for devices with WebGL disabled

Props:

```
{
  width: PropTypes.oneOfType([PropTypes.number, PropTypes.string]), // Container
width
  height: PropTypes.oneOfType([PropTypes.number, PropTypes.string]), // Container
height
  colorPalette: PropTypes.arrayOf(PropTypes.string), // Array of color hex codes
  particleCount: PropTypes.number, // Number of particles to render
  speed: PropTypes.number, // Animation speed multiplier
```

```
  intensity: PropTypes.number, // Color intensity (0.0-1.0)
  interactive: PropTypes.bool, // Whether mouse movement affects animation
  disableAnimation: PropTypes.bool // For performance-sensitive environments
}
```
Usage Example:
```
<Aurora
  width="100%"
  height="400px"
  colorPalette={["#1a2a6c", "#b21f1f", "#fdbb2d"]}
  particleCount={1000}
  speed={1.5}
  intensity={0.8}
  interactive={true}
/>
```

Iridescence Component

Path: src/Animations/Iridescence/Iridescence.jsx
Purpose: Creates a subtle iridescent shimmer effect on UI elements
Implementation: CSS-based animation using gradient overlays and CSS variables
Key Features:

- Light-weight CSS-only implementation for optimal performance
- Configurable gradient direction and shimmer speed
- Supports both light and dark mode with appropriate contrast
- Can be applied to buttons, cards, and section backgrounds
- Accessibility-friendly with reduced motion preference support

Props:
```
{
  width: PropTypes.oneOfType([PropTypes.number, PropTypes.string]), // Container
width
  height: PropTypes.oneOfType([PropTypes.number, PropTypes.string]), // Container
height
  direction: PropTypes.oneOf(['horizontal', 'vertical', 'diagonal']), // Shimmer
direction
  speed: PropTypes.oneOf(['slow', 'medium', 'fast']), // Animation speed
  intensity: PropTypes.number, // Effect intensity (0.0-1.0)
  children: PropTypes.node // Content to apply effect to
}
```

Content Components

BlogPostEditor Component

Path: src/components/Contents/TextEditor/index.jsx
Purpose: Rich text editor for creating and editing blog posts
Implementation: Built on top of TipTap editor with custom extensions
Key Features:

- WYSIWYG editing experience with formatting toolbar
- Support for images, videos, code blocks, and embeds
- Markdown shortcuts for power users (# for headings, ** for bold, etc.)
- Auto-save functionality to prevent data loss
- Draft versioning with restore capability
- Word count and reading time estimation
- SEO optimization suggestions
- Custom block components (callouts, tables, galleries)

Props:
```
{
  initialContent: PropTypes.string, // HTML or markdown content to initialize
editor
  onSave: PropTypes.func.isRequired, // Callback when content is saved
  onPublish: PropTypes.func, // Callback when publish button is clicked
  autoSaveInterval: PropTypes.number, // Time in ms between auto-saves
  toolbarConfig: PropTypes.shape({ // Configure which toolbar items to show
    basic: PropTypes.bool, // bold, italic, underline
    headings: PropTypes.bool,
    lists: PropTypes.bool,
```

```
    media: PropTypes.bool,
    advanced: PropTypes.bool // code, tables, etc.
  }),
  maxLength: PropTypes.number, // Maximum character count
  readOnly: PropTypes.bool, // Whether editor is in read-only mode
  placeholder: PropTypes.string, // Placeholder text when editor is empty
}
```

BlogPostCard Component

Path: src/components/Contents/Item/Article/index.jsx
Purpose: Displays a preview card for blog posts in listings and search results
Implementation: Responsive card component with multiple display variants
Variants:

- Standard: Image, title, excerpt, author, timestamp, engagement metrics
- Compact: Title, author, timestamp only (for sidebar listings)
- Featured: Larger image, title, excerpt, author with background image
- List: Horizontal layout for search results and feed views
- Minimal: Just title and timestamp for notification contexts

Props:

```
{
  post: PropTypes.shape({ // Post data object
    id: PropTypes.string.isRequired,
    title: PropTypes.string.isRequired,
    excerpt: PropTypes.string,
    coverImage: PropTypes.string,
    createdAt: PropTypes.string,
    readTime: PropTypes.number,
    category: PropTypes.string,
    tags: PropTypes.arrayOf(PropTypes.string),
    author: PropTypes.shape({
      id: PropTypes.string,
      name: PropTypes.string,
      avatar: PropTypes.string
    }),
    metrics: PropTypes.shape({
      likes: PropTypes.number,
      comments: PropTypes.number,
      shares: PropTypes.number
    })
  }).isRequired,
  variant: PropTypes.oneOf(['standard', 'compact', 'featured', 'list',
'minimal']),
  onClick: PropTypes.func, // Click handler for the card
  showAuthor: PropTypes.bool, // Whether to show author info
  showMetrics: PropTypes.bool, // Whether to show engagement metrics
  isBookmarked: PropTypes.bool, // Whether post is bookmarked by user
  onBookmarkToggle: PropTypes.func // Handler for bookmark toggle
}
```

Layout Components

AppLayout Component

Path: src/components/Layout/AppLayout.jsx
Purpose: Main application layout wrapper that provides common UI elements
Features:

- Responsive layout with mobile, tablet, and desktop breakpoints
- Header with navigation, search, and user menu
- Optional sidebar with customizable content
- Footer with site links and information
- Toast notification container
- Theme switching capability
- Authentication state awareness

Props:

```
{
```

```
  children: PropTypes.node.isRequired, // Page content
  title: PropTypes.string, // Page title for SEO and browser tab
  description: PropTypes.string, // Meta description
  showHeader: PropTypes.bool, // Whether to show header
  showFooter: PropTypes.bool, // Whether to show footer
  showSidebar: PropTypes.bool, // Whether to show sidebar
  sidebarContent: PropTypes.node, // Custom sidebar content
  headerTransparent: PropTypes.bool, // Whether header has transparent background
  containerClassName: PropTypes.string, // Additional CSS class for container
  requireAuth: PropTypes.bool, // Whether page requires authentication
  fullWidth: PropTypes.bool // Whether to use full width layout
}
```