Archilogic Software Requirements Specification (SRS)

Version 1.0

1. Introduction

1.1. Purpose

This document provides a detailed description of the requirements for the "Technical Blog Platform." It is intended for developers, project managers, and testers to understand the system's functionalities, features, and constraints. The platform will serve as a personal-to-multi-admin technical blog, allowing administrators to publish and manage content for registered users to read and interact with.

1.2. Scope

The initial scope of the project is to build a fully functional web application that includes:

- An admin panel for content creation, management, and user administration.
- A public-facing interface for users to read blogs, comment, and react.
- A robust backend built with Java (Spring Boot) and a frontend with React.
- A local file storage system for images and thumbnails, organized by blog title.

Future versions will build upon this foundation, introducing new features in a structured manner.

1.3. Definitions, Acronyms, and Abbreviations

- **Admin:** A user with elevated privileges to create, manage, and publish content and other users.
- **User:** A registered user with standard privileges to read and interact with content.
- **SRS:** Software Requirements Specification.
- **SDD:** Software Design Document.
- **UI:** User Interface.
- **API:** Application Programming Interface.
- **JWT:** JSON Web Token.

2. Overall Description

2.1. Product Perspective

The platform is a self-contained web application. The initial product will be a monolithic application (which can be scaled) with a clear separation between the backend API and the frontend client. The system is designed for growth, allowing the initial admin to promote other users to admin status, evolving from a personal blog to a multi-author platform.

2.2. Product Functions

Admin Functions:

- **Authentication:** Secure login/logout.
- **Dashboard:** View analytics (likes, dislikes, comments per blog), user list.
- **Blog Management:**
    - Create blogs using a rich-text editor.
    - Upload a thumbnail for each blog.
    - Upload images within the blog content.
    - Edit existing blogs.
    - Enable/Disable blogs to control public visibility.
    - Preview blogs before publishing.
    - Delete blogs.
- **Comment Management:**
    - View all comments on their blogs.
    - Reply to comments.
- **User Management:**
    - View a list of registered users.
    - Promote a standard user to an Admin role.

User Functions:

- **Authentication:** Register for an account, secure login/logout.
- **Blog Interaction:**
    - View all enabled/published blogs.
    - Read individual blog posts.
    - Like or Dislike a blog post (can only choose one).
    - View the like/dislike count.
- **Comment Interaction:**
    - Post comments on blogs.
    - View comments from other users and replies from admins.

2.3. User Characteristics

- **Admins:** Technically proficient individuals (initially the owner) comfortable with content creation and system management.
- **Users:** Individuals interested in technical topics like system design, software engineering, etc. They are readers and community participants.

2.4. Constraints

- **Technology Stack:** The backend must be built with Java/Spring Boot, the frontend with React, and the database must be MySQL.
- **Image Storage:** All image files (thumbnails, blog images) must be stored on the local file system of the server where the application is deployed. The directory structure must be programmatically managed based on blog titles.
- **Deployment:** The system must be deployable and scalable.

3. Specific Requirements

3.1. Functional Requirements

Authentication & Authorization (FUNC-AUTH)

- **FUNC-AUTH-001:** The system shall allow users to register with a unique email and password.
- **FUNC-AUTH-002:** The system shall allow users and admins to log in using their credentials.
- **FUNC-AUTH-003:** The system shall use role-based access control (Admin, User).
- **FUNC-AUTH-004:** Admins shall have access to all admin-level functionalities. Users shall be restricted to user-level functionalities.

Blog Management (FUNC-BLOG)

- **FUNC-BLOG-001:** An Admin shall be able to create a blog post with a title, content, and a thumbnail image.
- **FUNC-BLOG-002:** The blog content editor shall allow for rich-text formatting and inline image uploads.
- **FUNC-BLOG-003:** An Admin shall be able to save a blog as a draft or publish it.
- **FUNC-BLOG-004:** An Admin shall be able to disable a published blog, making it invisible to Users.
- **FUNC-BLOG-005:** An Admin shall be able to edit and update any blog post they have created.
- **FUNC-BLOG-006:** An Admin shall be able to preview the blog post as it would appear to a User.
- **FUNC-BLOG-007:** When a User views a blog, all content, including inline images, must be displayed in the correct order as defined by the Admin.

File Management (FUNC-FILE)

- **FUNC-FILE-001:** When a thumbnail is uploaded for a blog titled "My Blog Title", it shall be saved in the path: <base_dir>/thumbnails/my-blog-title/thumbnail.<ext>.
- **FUNC-FILE-002:** When an inline image is uploaded to the same blog, it shall be saved in the path: <base_dir>/blog-images/my-blog-title/1.<ext>, with the number incrementing for each subsequent image.
- **FUNC-FILE-003:** The system must handle special characters and spaces in blog titles to create valid directory names (slugification).

Interaction (FUNC-INT)

- **FUNC-INT-001:** A logged-in User shall be able to "Like" or "Dislike" a blog post once. They can change their vote.
- **FUNC-INT-002:** The total count of likes and dislikes shall be visible to all Users and Admins.
- **FUNC-INT-003:** A logged-in User shall be able to post a text-based comment on a blog.
- **FUNC-INT-004:** An Admin shall be able to post a reply to any comment. Replies should be visually distinct from primary comments.

Admin Panel (FUNC-ADMIN)

- **FUNC-ADMIN-001:** The admin dashboard shall display key analytics for each blog: view count, like count, dislike count, and comment count.
- **FUNC-ADMIN-002:** An Admin shall be able to view a list of all registered users and their roles.
- **FUNC-ADMIN-003:** An Admin shall be able to change a User's role to Admin.

3.2. Non-Functional Requirements

- **Performance (NFR-PERF-001):** The system should be scalable to handle a growing number of users and posts. API response times for reading blogs should be under 500ms.
- **Security (NFR-SEC-001):** Passwords must be hashed. The system must be protected against common web vulnerabilities (XSS, CSRF, SQL Injection).
- **Reliability (NFR-REL-001):** The application should have high availability, with minimal downtime.
- **Maintainability (NFR-MAIN-001):** The system shall be versioned (e.g., API v1, v2) to allow for the addition of new features without breaking existing functionality.