# Software Requirements Specification (SRS) for Blog Web Application

## 1. Introduction

### 1.1 Purpose

This SRS document outlines the requirements for a blog web application, designed to provide users with a platform to read, interact with, and subscribe to content. It includes functionality for users to like, comment on, and subscribe to a newsletter, while providing administrators with tools to manage content through a simple interface.

### 1.2 Scope

The blog web application will be built using Django as the backend and HTML, CSS, and JavaScript for the front end. Users will have various interactive features, while admins will have content creation and management tools. The application is designed to be modular and extendable, allowing for future enhancements.

### 1.3 Definitions, Acronyms, and Abbreviations

* **SRS**: Software Requirements Specification
* **Django**: Python-based web framework
* **Frontend**: Client-side components (HTML, CSS, JavaScript)
* **Backend**: Server-side components (Django)

## 2. Overall Description

### 2.1 Product Perspective

This application is a standalone web application that will run on a web server. It uses Django for backend processes, including data storage, authentication, and content management, with HTML, CSS, and JavaScript for frontend design and interactivity.

### 2.2 Product Functions

* **User functionalities**: Register, log in, view articles, like articles, comment on articles, subscribe to/unsubscribe from the newsletter.
* **Admin functionalities**: Create and manage articles, upload images, view and moderate comments, view subscribers list.

### 2.3 User Characteristics

* **End-users**: Individuals who view, interact with, and subscribe to content.
* **Admins**: Individuals responsible for content creation and management.

## 3. Functional Requirements

### 3.1 User Features

#### 3.1.1 Article Browsing

* Users can browse a list of articles.
* Users can view individual articles on separate pages.

#### 3.1.2 Article Interactions

* **Like Article**: Users can like/unlike articles. The system will maintain a count of likes per article.
* **Comment on Article**: Users can add comments to articles.
* **View Comments**: Users can view other comments associated with an article.

#### 3.1.3 Newsletter Subscription

* Users can subscribe to receive newsletter updates.
* Users can unsubscribe from the newsletter at any time.

### 3.2 Admin Features

#### 3.2.1 Admin Login

* Admins can log in using a secure, separate login page with authentication.

#### 3.2.2 Article Management

* **Create Article**: Admins can create new articles by entering a title, content, and uploading an image.
* **Edit Article**: Admins can edit existing articles.
* **Delete Article**: Admins can delete articles.

#### 3.2.3 Comment Moderation

* Admins can view all comments on articles.
* Admins can delete comments if necessary.

#### 3.2.4 Newsletter Management

* Admins can view a list of newsletter subscribers.
* Admins can export the list of subscribers.

## 4. Non-Functional Requirements

### 4.1 Performance Requirements

* The system should handle up to 10,000 users with minimal latency.
* Article loading should take no longer than 2 seconds on average.

### 4.2 Security Requirements

* Only authenticated admins can access content management features.

### 4.3 Usability Requirements

* The interface should be intuitive and responsive across devices.
* Forms should include validation to help users avoid errors.

### 4.4 Reliability Requirements

* The system should be available 99.9% of the time.
* The system should include error-handling and logging for admin reviews.

### 4.5 Maintainability Requirements

* The codebase should be modular to support future enhancements.
* The system should have thorough documentation for developers.

## 5. Interface Requirements

### 5.1 User Interfaces

* **Home Page**: Lists recent articles with options to like and view.
* **Article Page**: Displays the article’s content, comments, like button, and a form to add comments.
* **Admin Panel**: Accessible only to admins, containing options to create/edit/delete articles and manage comments.

### 5.2 Software Interfaces

* **Database**: SQL lite for storing data related to users, articles, comments, likes, and subscriptions.
* **Email Service**: For sending newsletters to subscribers.

## 6. Future Enhancements

* **User Analytics**: Track user engagement metrics such as views, likes, and comments.
* **Social Media Integration**: Allow users to share articles on social media.
* **Advanced Comment Moderation**: Implement automated tools to filter spam.