# Software Requirements Specification (SRS) for Blog Web Application

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to define the requirements for a blog web application. This application enables users to create, read, search, and subscribe to newsletters. The frontend is developed using HTML, CSS, and JavaScript, while the backend is powered by Django with SQLite as the database.

### 1.2 Scope

The blog web application provides users with the ability to sign up, log in, create blog posts, search for articles, and subscribe to a newsletter. It ensures a seamless experience for managing personal blogs and accessing content.

### 1.3 Definitions and Acronyms

* **SRS**: Software Requirements Specification
* **UI**: User Interface
* **Django**: A Python-based web framework
* **SQLite**: A lightweight database engine

## 2. Overall Description

### 2.1 Product Perspective

This blog web application is a standalone system that leverages Django for backend functionality and SQLite for data storage. The frontend is built using HTML, CSS, and JavaScript, ensuring responsiveness and interactivity.

### 2.2 Product Features

1. **Authentication System**: Users can sign up, log in, and log out securely.
2. **Blog Management**: Users can create blog posts with a title and body. The author’s name is automatically set to the username.
3. **Search Functionality**: Users can search for articles by keywords.
4. **Newsletter Subscription**: Users can subscribe to receive newsletters.

### 2.3 User Classes and Characteristics

* **Registered Users**: Can log in, create blog posts, search articles, and subscribe to newsletters.

### 2.4 Operating Environment

* **Frontend**: HTML, CSS, JavaScript (browser-based interface).
* **Backend**: Django (Python framework).
* **Database**: SQLite.

### 2.5 Constraints

* Unique usernames for each user.
* Responsive design for accessibility on multiple devices.

## 3. Functional Requirements

### 3.1 Authentication

1. Users can sign up with a unique username and password.
2. Users can log in with valid credentials.
3. Users can log out using a signout button.

### 3.2 Blog Management

1. Users can add a new blog with the following fields:
   * Title
   * Body
2. The author’s name is automatically set to the username of the logged-in user.

### 3.3 Search Functionality

1. Users can search for articles by entering keywords in the search bar.
2. The system displays a list of matching articles.

### 3.4 Newsletter Subscription

1. Users can subscribe to a newsletter by providing their email address.
2. The system stores subscriber information for newsletter distribution.

## 4. Non-Functional Requirements

### 4.1 Performance Requirements

* The system should handle up to 100 concurrent users.
* Search results should be displayed within 5 seconds.

### 4.2 Security Requirements

* Passwords must be securely hashed.
* Authentication tokens should be used for managing user sessions.

### 4.3 Usability Requirements

* The interface should be intuitive and easy to navigate.
* Forms should include validation to help users avoid errors.

### 4.4 Maintainability Requirements

* Code should be well-documented and follow standard practices.
* The code base should be modular to support future enhancements.

## 5. System Design Constraints

### 5.1 Frontend

* The UI should be designed using responsive web design principles.

### 5.2 Backend

* The backend should use Django’s built-in authentication system for user management.

### 5.3 Database

* SQLite should be used for storing user data, blogs, and subscriber information.