**Software Requirements Specification (SRS) for Image Searching Software**

## 1. Introduction

### 1.1 Purpose

The Image Searching Software is designed to help users find specific images efficiently. It enables users to search for images based on metadata, visual similarity, or text-based queries. This software aims to improve image retrieval by leveraging a structured database and advanced search algorithms.

### 1.2 Scope

The system will allow users to:

* Upload and index images into a database.
* Search images using keywords, metadata, and image similarity.
* Retrieve images quickly with optimized search algorithms.
* Store and manage image metadata using SQL databases.
* Provide a user-friendly interface for searching and filtering images.

This software will be developed using Python, SQL, and C++.

### 1.3 Intended Audience and Use

The intended users of this software include:

* Photographers and designers searching for visual references.
* Researchers needing structured image retrieval.
* General users looking for specific images within large datasets.

### 1.4 Definitions and Acronyms

* **SQL (Structured Query Language)** - A programming language for managing relational databases.
* **Image Indexing** - The process of storing and structuring images for efficient retrieval.
* **Metadata** - Data that provides information about an image (e.g., tags, descriptions, formats).
* **Search Algorithm** - A method used to find relevant images based on user input.

## 2. System and Functional Requirements

### 2.1 Functional Requirements

1. **User Authentication:**
   * Users must be able to create an account and log in securely.
2. **Image Uploading & Indexing:**
   * Users can upload images, which are then processed and indexed in the database.
3. **Search Functionality:**
   * Keyword-based search (search by tags, filename, or descriptions).
   * Image similarity search (search using an existing image).
4. **Metadata Management:**
   * Users can add, edit, and delete metadata associated with images.
5. **Result Display & Filtering:**
   * Users can filter search results based on metadata, image size, or upload date.
6. **Database Management:**
   * The system must store image references efficiently using an SQL database.

### 2.2 System Requirements

* **Operating System:** Windows, Linux, macOS
* **Programming Languages:** Python, C++
* **Database:** SQL-based relational database
* **Image Processing Libraries:** OpenCV, PIL (Python Imaging Library)

## 3. External Interface Requirements

### 3.1 User Interfaces

* A graphical user interface (GUI) with an intuitive search bar and image results page.
* Users should be able to upload images via drag-and-drop or file selection.

### 3.2 Hardware Interfaces

* The software should support storage devices such as SSDs and HDDs for storing indexed images.

### 3.3 Software Interfaces

* Integration with OpenCV for image processing.
* Use of SQL databases for metadata storage.

### 3.4 Communication Interfaces

* API for external applications to request image search results.

## 4. Non-Functional Requirements

### 4.1 Performance Requirements

* The system should return search results within 2 seconds for a database of up to 100,000 images.
* Image indexing should take no longer than 5 seconds per image.

### 4.2 Security Requirements

* Encrypted database storage for user credentials.
* Access control to restrict unauthorized modifications.

### 4.3 Usability Requirements

* The user interface should be designed for easy navigation and fast interaction.
* The search system should allow auto-suggestions for quick searching.

### 4.4 Scalability Requirements

* The system should support database expansion up to 1 million images without significant performance degradation.

### 4.5 Maintainability Requirements

* The codebase should be modular for easy upgrades and debugging.

## 5. Appendix

This document serves as a guideline for the development of the Image Searching Software. The team will follow an incremental development approach, ensuring each functionality is tested before moving to the next phase.