ShutterSort

Version 0.1

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/10/2023 | 0.1 | The first iteration of the document. | Riley Meyerkorth |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 5

1.5 Overview 5

2. Overall Description 6

2.1 Product perspective 6

2.1.1 System Interfaces 6

2.1.2 User Interfaces 6

2.1.3 Hardware Interfaces 6

2.1.4 Memory Constraints 6

2.2 Product functions 6

2.3 User characteristics 7

2.4 Constraints 7

2.5 Assumptions and dependencies 8

2.6 Requirements subsets 8

3. Specific Requirements 9

3.1 Functionality 9

3.1.1 Automatic Live Photo Sorting 9

3.1.2 User Preferences and Customization 9

3.1.3 Machine Learning/Computer Vision Integration 9

3.1.4 Error Handling 9

3.1.5 Notifications and Tray Icon 9

3.2 Use-Case Specifications 9

3.2.1 Functional Requirement Use Cases 9

3.2.2 Non-Functional Requirement Use Cases 12

3.3 Supplementary Requirements 12

3.3.1 Unincluded Non-Functional Requirements 12

3.3.2 Unincluded Development Constraints 12

4. Classification of Functional Requirements 12

5. Appendices 13

# Introduction

## Purpose

The purpose of the Software Requirements Specification document is to provide a clear, complete, and concise description of the functionalities, behaviors, and constraints of the ShutterSort application. The SRS encompasses both the external behaviors exhibited by the application, such as interactions with users and interfaces, and internal functionalities, like metadata analysis and photo categorization. Beyond functional requirements, this document will also outline the non-functional requirements, design constraints, and other critical factors necessary to ensure the software meets its intended objectives and caters to its target audience effectively. The SRS seeks to guide the development team and stakeholders in the creation and evaluation of the ShutterSort application. As with any piece of documentation, the SRS may be appended to or modified over the course of ShutterSort’s development.

## Scope

The scope of this Software Requirements Specification (SRS) applies to the ShutterSort application, a state-of-the-art software solution designed to automate the sorting and categorization of photographs for professional and amateur photographers alike. ShutterSort leverages intricate algorithms to analyze photograph metadata, pixel data, and other pertinent information, facilitating the categorization of large volumes of photographs with accuracy and efficiency, through both algorithms and machine learning.

This SRS is associated with the use-case model that outlines the various interactions between users and the ShutterSort system, capturing the gamut of functionalities and behaviors the software exhibits. The document encompasses requirements for all major features of the software, including photograph uploading, metadata extraction and analysis, automatic sorting based on user-defined criteria, and user interface interactions.

Beyond the core functionalities of ShutterSort, this SRS also influences and informs tools, documentation, and supplementary applications developed in tandem with ShutterSort. This ensures that all associated components align with the primary objectives and specifications of the main application, guaranteeing cohesion and synergy across the entire ShutterSort ecosystem.

## Definitions, Acronyms, and Abbreviations

* ShutterSort: The name of the software application being developed, focused on automating the sorting and categorizing of photographs.
* Metadata: Data that provides information about other data. In the context of ShutterSort, it refers to details embedded in photographs, such as date, time, camera model, location, and other related information.
* SRS: Software Requirements Specification. A document that captures the complete software requirements for a system or subsystem.
* Use-Case Model: A type of software behavioral modeling that describes how potential users interact with a system to accomplish their goals.
* UI: User Interface. Refers to the space where user interactions with the software occur.
* UX: User Experience. Encompasses all aspects of a user's interaction with the software, particularly in terms of how easy and pleasing it is to use.
* API: Application Programming Interface. A set of tools and protocols that allows software applications to communicate with each other.
* OS: Operating System. System software that manages computer hardware, software resources, and provides various services for computer programs.

## References

The following documents and resources have been referenced in the creation of this Software Requirements Specification (SRS) for ShutterSort:

* ShutterSort Software Development Plan
  + Title: Software Development Plan for ShutterSort
  + Date: September 15, 2023
  + Publishing Organization: CRRANkS
  + Source: GitHub Repository
* Use-Case Model for ShutterSort
  + Title: ShutterSort Use-Case Model
  + Date: September 22, 2023
  + Publishing Organization: CRRANkS
  + Source: GitHub Repository
* Project Meeting Logs/Notes
  + Title: ShutterSort Project Meetings Compilation
  + Date: Ongoing (from project initiation)
  + Publishing Organization: CRRANkS
  + Source: GitHub Repository
* GitHub Repository
  + Title: ShutterSort
  + Date: Ongoing (from project initiation)
  + Publishing Organization: CRRANkS
  + Source: Canvas Group

These references provide additional information, detailed explanations, and foundational context to various aspects mentioned in the SRS. Any new documents or resources that are referenced during the project's progress will be duly added to this list.

## Overview

The rest of the SRS contains the following sections:

* Overall Description
  + Defines the general factors which will affect both ShutterSort and the requirements.
* Specific Requirements
  + Details each requirement at a level of detail sufficient enough to help the developers of the software understand what is needed.
* Classification of Functional Requirements
  + A list of all the functional requirements of ShutterSort and their type.
* Appendices
  + Any appended file that could or could not be considered part of the requirements.

# Overall Description

## Product perspective

### System Interfaces

ShutterSort is designed to interact with the OS filesystem to move files and read file data.

* Storage Interfaces
  + Description: ShutterSort will interface directly with storage devices specifically designated by the user.
  + Data: Metadata from files; directory modification; file deletion, creation, and duplication

### User Interfaces

The User Interface for ShutterSort will be thoughtfully designed with ease of use in the forefront.

* Dashboard
  + Description: The primary landing page when the user opens the application. It provides a quick overview of sorting methods, the sorting order, and a few buttons to change simple user settings so that the user can mold the application to their needs.
  + Features: Navigation buttons, sorting order stack, sorting methods selector, sorting method modifier
* Settings Page
  + Description: A page where the user can change settings directly relating to the *application* portion of ShutterSort
  + Features: Toggle automatically start on boot, toggle live-sorting in the folder, color themes(?)
* Desktop Addons
  + Description: Desktop addons allow for more clarity to the user if something goes wrong with the software during sorting, provides clarity to the user on where to find their sorted photos, ability to re-open the GUI, and allows users to start/stop the sorting process at will
  + Features: Desktop notifications, taskbar icon
* Anchor Point
  + The folder which will be the main point of contact between the user and the software. This will be interacted with much more than the main GUI in practical use.

### Hardware Interfaces

* Processors and Memory:
  + Description: Efficient sorting and rapid processing of high-resolution images require robust computational power.
  + Specifications: Optimized for multi-core processors and utilizes RAM for temporary data storage during active sorting sessions.

### Memory Constraints

Given the early nature of the development process, we have not been able to test a benchmark for the program’s efficiency on different computer set-ups, processors, and RAM amounts. In general, however, ShutterSort is designed to be a very lightweight program that still can perform strenuous sorting tasks.

## Product functions

ShutterSort boasts a suite of core functions designed to provide a seamless and efficient photo sorting experience. Here's a description of the main product functions:

* Automatic Live Photo Sorting:
  + Description: ShutterSort's primary function, where it analyzes the metadata of images and employs predefined or user-defined criteria to categorize them in real time.
  + Functionalities: Metadata extraction, categorization algorithm, user-preference adjustments.
* User Preferences and Customization:
  + Description: Allows users to tweak and modify the sorting criteria based on their specific needs or preferences.
  + Functionalities: Adding new sorting criteria, adjusting existing ones, setting priorities for sorting.
* Machine Learning/Computer Vision
  + Description: ShutterSort can utilize machine learning and computer vision to sort by recognizing specific objects or places or people within the images it sorts.
  + Functionalities: sort by object, sort by structure, sort by person
* Error Handling and Notifications:
  + Description: In the event of software hiccups, incorrect photo formats, or other anomalies, ShutterSort promptly informs the user.
  + Functionalities: Error detection, user alerts, troubleshooting suggestions.
* Anchor Point Operations:
  + Description: The main folder or directory where all the sorting actions occur, serving as the primary touchpoint for user interaction.
  + Functionalities: Direct photo upload, manual sorting adjustments, view sorted/unsorted photos.
* Help and Support:
  + Description: A dedicated section offering guides, FAQs, and assistance to users.
  + Functionalities: FAQ search, tutorial access, direct support communication.

By offering these core functions, ShutterSort aims to address the diverse needs of its user base, ensuring flexibility, user control, and automated efficiency in the photo sorting process.

## User characteristics

* Photographers
  + Description: Professional and amateur photographers who have a considerable volume of photographs and need a quick, efficient method to categorize and sort them.
* Digital Artists
  + Description: Individuals working with digital media who accumulate a large number of image files over time and need organized storage.
* Archivists and Digital Librarians
  + Description: Individuals responsible for maintaining large digital archives, such as in museums, libraries, or research institutions.
* General Users
  + Description: Everyday users who, in the digital age, accumulate photos from various sources like smartphones, downloadable content, and screenshots.

## Constraints

* Image Recognition Accuracy
  + Description: While computer vision has made very large advances in the past decade, there is still a margin of error that is present.
  + Constraints: The accuracy of the recognition depends on the training size, training time, etc. which all will take more space offline.
* Data Privacy and Security
  + Description: Ensuring user data, especially images, remains private and is not misused.
  + Constraints: ShutterSort must adhere to strict data handling protocols, ensuring no image data is stored or accessed beyond the application's primary purpose.
* Resource Consumption
  + Description: Processing large batches of photos, especially with machine learning functionalities, can be resource-intensive.
  + Constraints: Users with limited system resources might experience slower processing times. The software should be optimized to reduce resource consumption.
* Internet Connectivity
  + Description: Future versions of ShutterSort might incorporate cloud functionalities or API functionalities.
  + Constraints: Such features would require users to have stable internet connectivity.
* Storage Limitations
  + Description: If ShutterSort wants to remain a completely offline program, the machine learning and computer vision components, if implemented, could demand additional storage space for model data.
  + Constraints: Users might need to ensure they have enough storage space on their device for optimal software functionality.

## Assumptions and dependencies

**Assumptions**

* Standardized Metadata
  + The software assumes that all photos fed to it will have the standard metadata tags.
* User Knowledge
  + ShutterSort assumes that the user knows how to use basic functions of a computer, such as dragging and dropping files and clicking buttons.

**Dependencies**

* External Libraries and Frameworks
  + Machine learning datasets depend on massive external libraries in order to save space on the user’s storage device.
* API
  + If APIs are used in the development of ShutterSort, their continued support will be necessary for ShutterSort to continue running without refactoring.

## Requirements subsets

1. Basic Algorithmic Photo Sorting (Phase 1)
   1. Description: This subset covers the primary function of ShutterSort, which is the categorization of photographs based on their metadata.
   2. Key Requirements: Metadata extraction, categorization algorithm, user-interface for viewing sorted photos.
2. Graphical User Interface Connectivity (Phase 2)
   1. Description: The connectivity between the GUI and the actual functionality of the sorting algorithms.
   2. Key Requirements: GUI windows, ergonomic UI design, multiple pages
3. Advanced User Customization (Phase 3)
   1. Description: Advanced features allowing users to set detailed sorting criteria and preferences.
   2. Key Requirements: Advanced sorting criteria settings, sorting profiles, customizable UI themes, saving and loading of user-defined sorting presets.
4. Machine Learning/Computer Vision Integration (Phase 4)
   1. Description: Integration of machine learning capabilities to enhance sorting accuracy and recognize specific objects or scenes.
   2. Key Requirements: ML model training and integration, object/scene recognition, feedback loop for user corrections.
5. Help and Extensive Documentation (Phase 5)
   1. Description: Features designed to support and engage the user community, including detailed tutorials, FAQ sections, and possibly a forum or community section.
   2. Key Requirements: In-app help, FAQ, community forum (if pursued), feedback/support ticketing system.

# Specific Requirements

## Functionality

The main pieces of functionality for ShutterSort are:

### Automatic Live Photo Sorting

* **Description:** Automatically categorizes images based on their metadata, offering real-time sorting.
* **Requirements:**
  + Ability to read metadata from a wide range of image formats.
  + Robust algorithm to categorize images based on predefined and user-defined criteria.
  + Notification system to alert users once sorting is complete or if there are issues

### User Preferences and Customization

* **Description:** Provides users with the flexibility to modify sorting criteria and the application based on specific needs.
* **Requirements:**
  + User-friendly interface for setting and modifying sorting criteria.
  + Save and load functionality for user-defined presets.
  + Option to prioritize certain metadata tags over others.
  + Ability to change application-specific settings

### Machine Learning/Computer Vision Integration

* **Description:** Machine learning and computer vision can categorize images based on the objects within them, the amount of a particular object present, a specific person, and more depending on the training data.
* **Requirements:**
  + Ability to access large set of training material
  + Ability to identify objects and various other targets within images

### Error Handling

* **Description:** Ensures user file safety in the event of an unprecedented error or interruption.
* **Requirements:**
  + Identify and categorize common errors during the sorting process.
  + Log errors for further troubleshooting
  + Move backup of affected photos to the output folder to avoid file damage or corruption

### Notifications and Tray Icon

* **Description:** Provides clarity to users about the status of sorting and the ability quickly to pause/resume sorting or open the GUI
* **Requirements:**
  + Show specific data relating to sorting with clear messages
  + Allow very basic control over sorting and the program quickly and easily
  + Ability to show the user that the program is running in the background, and allow them to open the GUI to make changes

## Use-Case Specifications

### Functional Requirement Use Cases

#### Open the GUI

**Actor(s):**

* User

**Description:**

* The user opens the application, revealing the main dashboard interface. From there, they can reorganize the sorting parameters and change other various settings.

**Precondition(s):**

* ShutterSort application is installed.

#### Open app settings page

**Actor(s):**

* User

**Description:**

* User accesses the application's settings to modify or view configurations.

**Precondition(s):**

* GUI is open.

#### Toggle auto-run

**Actor(s):**

* User

**Description:**

* User toggles the auto-run feature, determining if the application starts on system boot.

**Precondition(s):**

* User is on the settings page.

#### Toggle live-sorting

**Actor(s):**

* User

**Description:**

* User enables or disables the real-time sorting of incoming photos.

**Precondition(s):**

* User is on the settings page.

#### Set Anchor Point

**Actor(s):**

* User

**Description:**

* User sets a primary directory as the main sorting location.

**Precondition(s):**

* GUI is open.

#### Create new folder

**Actor(s):**

* User

**Description:**

* User creates a new folder for the Anchor Point.

**Precondition(s):**

* GUI is open.

#### Choose existing folder

**Actor(s):**

* User

**Description:**

* User designates an existing folder as the Anchor Point.

**Precondition(s):**

* GUI is open.

#### Configure sorting methods

**Actor(s):**

* User

**Description:**

* User adjusts the list of specific methods by which photos are sorted.

**Precondition(s):**

* GUI is open.

#### Change the priority/order of sorting methods

**Actor(s):**

* User

**Description:**

* User rearranges the priority of the applied sorting methods.

**Precondition(s):**

* User is configuring the sorting methods

#### Enable/disable specific sorting methods

**Actor(s):**

* User

**Description:**

* User can activate or deactivate particular sorting methods from the sorting list.

**Precondition(s):**

* User is configuring the sorting methods

#### Modify data ranges of specific sorting methods

**Actor(s):**

* User

**Description:**

* User modifies the specific parameters or thresholds of a sorting method.

**Precondition(s):**

* User is configuring the sorting methods

#### Change/create output folder for specific sorting methods

**Actor(s):**

* User

**Description:**

* User specifies or changes the destination folder for a sorting type. If the user does not specify a specific destination folder, a folder is automatically created and the user is noted of it’s position.

**Precondition(s):**

* User is configuring the sorting methods

#### Add/remove specific sorting method from list

**Actor(s):**

* User

**Description:**

* User adds a new sorting method or removes an existing one.

**Precondition(s):**

* User is configuring the sorting methods

#### Drop images into Anchor Point

**Actor(s):**

* User

**Description:**

* User drags and drops photos into the Anchor Point for sorting.

**Precondition(s):**

* Anchor Point is set.

#### Sort images based on desired parameters

**Actor(s):**

* System

**Description:**

* Program automatically checks the Anchor Point directory for changes. If there are new files added, they are sorted according to the set parameters.

**Precondition(s):**

* Anchor Point is set

#### Sort by metadata

**Actor(s):**

* System

**Description:**

* Application categorizes photos using embedded metadata. There may be multiple forms of metadata that can be sorted, and thus a database of metadata sorting functions will be created.

**Precondition(s):**

* Photos with metadata are in the Anchor Point.

#### Sort by machine learning

**Actor(s):**

* System

**Description:**

* The application employs machine learning to sort photos.

**Precondition(s):**

* Photos suitable for machine learning are in the Anchor Point and machine learning parameters are set.

#### Sort by pixel data

**Actor(s):**

* System

**Description:**

* The application categorizes photos based on their pixel content.

**Precondition(s):**

* Photos are present within the Anchor Point and pixel data sorting is enabled and set

### Non-Functional Requirement Use Cases

N/A

## Supplementary Requirements

### Unincluded Non-Functional Requirements

N/A

### Unincluded Development Constraints

N/A

# Classification of Functional Requirements

|  |  |
| --- | --- |
| **Functionality** | **Type** |
| Automatic Live Photo Sorting | Essential |
| User Preferences and Customization | Essential |
| Error Handling | Essential |
| Machine Learning/Computer Vision Integration | Desirable |
| Notifications and Tray Icon | Optional |

# Appendices

N/A