Software Requirements Specification Template

Bartholomew BinkleBurg

Software Requirements Specification

Version 1.1

May 5th, 2025

Andrew, Mitch, Astrid

Software Engineer

Prepared for

CSCI4099-01

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| <date> | <Version 1> | <Your Name> | <First Revision> |
| 10/22/2024 | 1.1 | Andrew Booth |  |
| 5/5/2025 | 1.1 | Andrew Booth |  |
|  |  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | <Your Name> | Software Eng. |  |
| Andrew Booth | Andrew Booth | Student | 5/5/2025 |
|  |  |  |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

2. General Description 1

2.1 Product Perspective 1

2.2 Product Functions 1

2.3 Constraints, Assumptions and Dependencies 1

3. Specific Requirements 1

3.1 External Interface Requirements 2

3.1.1 Software Interfaces 2

3.2 Functional Requirements 2

3.2.1 <Functional Requirement or Feature #1> 2

3.2.2 <Functional Requirement or Feature #2> 2

3.3 Use Cases 2

3.3.1 Use Case #1 2

3.3.2 Use Case #2 2

3.4 Classes / Objects 2

3.4.1 <Class / Object #1> 2

3.4.2 <Class / Object #2> 2

3.5 Non-Functional Requirements 2

3.5.1 Performance 3

3.5.2 Reliability 3

3.5.3 Availability 3

3.5.4 Security 3

3.5.5 Maintainability 3

3.5.6 Portability 3

3.7 Logical Database Requirements 3

3.8 Other Requirements 3

4. Analysis Models 3

4.1 Use Case Diagrams 3

4.2 Class Diagrams 3

4.3 Sequence Diagrams 3

4.4 Activity Diagrams (DFD) 3

4.5 State-Transition Diagrams (STD) 3

5. Change Management Process 4

A. Appendices 4

# 1. Introduction

## *This Software Requirements Specification (SRS) describes the requirements for the Bartholomew Binkleburg web application, an AI-assisted image matching and contribution platform. The application allows users to upload fashion-related images and receive similar matches from a dataset, with optional search by item type and color. It includes user authentication, account management, and basic history tracking.*

## 1.1 Purpose

## *The purpose of this SRS is to define the functionality, constraints, and interactions of the Bartholomew Binkleburg system. The document is intended for developers, designers, testers, and project stakeholders involved in the development and deployment of the platform.*

## 1.2 Scope

*This subsection should:*

1. *Software product’s name*

*Bartholomew Binkleburg*

1. *What the software product will do and won’t do (just high level, no details)*

*Allows users to:*

*Allow users to log in, register, and log out*

*Enable users to search a dataset of images by selecting an item type (e.g., “jeans”) and an optional color (e.g., “red”)*

*Track and display search history per user*

*Does not include:*

*User profile editing*

*Advanced analytics or feedback mechanisms*

*Administrative moderation tools*

1. *Describe the application of the software being specified. Describe all relevant benefits, objectives, and goals as precisely as possible.*

*This web application provides a simplified interface for users to contribute fashion-related images and find matches using AI-driven techniques. It promotes crowd-sourced fashion image collection and intuitive search. Goals include enhancing user experience, supporting exploratory visual search, and contributing data to a growing dataset.*

## 1.3 Definitions, Acronyms, and Abbreviations

*This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.*

***SRS*** *– Software Requirements Specification*

***UI*** *– User Interface*

***AI*** *– Artificial Intelligence*

***Flask*** *– A lightweight Python web framework*

***CORS*** *– Cross-Origin Resource Sharing*

***.pkl*** *– Pickle format (Python object serialization)*

***RGB*** *– Red, Green, Blue color model*

# 2. General Description

*This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.*

## 2.1 Product Perspective

Bartholomew Binkleburg is a self-contained, local web application composed of:

* A Flask server for responding to item+color search requests (/search)
* A front-end interface with dropdown filters
* A simple PHP-based login/register system
* A static dataset of images annotated with item classes and color values stored in a .pkl file

## 2.2 Product Functions

## *The system provides the following high-level functions:*

## *User Authentication: Secure registration, login, and logout features*

## *Search by Filters: Dropdown selection of item type and color, followed by result display*

## *Search History: Retains and displays previous searches for logged-in users*

## 2.3 Constraints, Assumptions and Dependencies

# *Constraints:*

# *No image upload support in final version*

# *Search relies on static .pkl data loaded at server startup*

# *Requires localhost Flask server (localhost:5000) and PHP backend (/php/ routes)*

# *Assumptions:*

# *Users will use only the dropdowns for searching*

# *Search terms (item and color) correspond to known labels in the dataset*

# *Color filtering is approximate and based on average RGB values*

# *Dependencies:*

# *Flask, PIL, and Python for server logic*

# *JavaScript and Fetch API for front-end behavior*

# *PHP for handling user sessions*

# 3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 3.2. Overall, this section 3 will capture the requirements that are used to guide the project’s software design, implementation, and testing.

## 

## 3.1 External Interface Requirements

### 3.1.1 Software Interfaces

**Front-End**: HTML/CSS/JavaScript interface served through a browser

**Back-End**:

* Python Flask server providing /search endpoint for filtered item lookups
* PHP scripts for login (login.php), registration (register.php), logout (logout.php), and user history (get\_history.php)

**Data Source**: Local .pkl file (image\_content\_data\_with\_color.pkl) contains image metadata (bounding boxes, item classes, color values)

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 <User Registration and Login#1>

**Description**: The system shall allow users to create accounts, log in securely, and log out. Sessions shall persist across visits until manual logout.

### 3.2.2 <Item + Color Search #2>

## Description: The system shall allow users to search for items by selecting a category (e.g., hoodie, jeans) and an optional color (e.g., red, blue). The search result will include up to 3 matched images and bounding box coordinates.

### 3.2.3 <Display Search Results #3>

**Description**: The system shall display matching image thumbnails with bounding box details in a styled format. If no matches are found, the system shall show a default message.

### 3.2.4 <View Account Info and Search History #4>

**Description**: The system shall display the currently logged-in username and show a list of previous search queries made during the session.

### 3.2.5 <Error Handling #5>

**Description**: The system shall catch errors such as missing item selection, invalid color filters, or back-end unavailability and present appropriate feedback to the user.

## 3.3 Use Cases

### 3.3.1 Use Case #1

**Filter-Based Search**

**Actors**: Logged-in User

**Trigger**: User selects an item and clicks “Search by Color”

**Preconditions**: User is logged in

**Normal Flow**:

1. User selects item type and optionally a color
2. User clicks the search button
3. Flask server returns matching images
4. Front-end displays images

**Alternative Flow**: If no results, display “No matches found.”

**Exceptions**: Network error or invalid filter → show alert

### 3.3.2 Use Case #2

## Login

**Actors**: New or Returning User

**Trigger**: User submits login credentials

**Preconditions**: User exists in database

**Normal Flow**:

1. User enters credentials
2. System verifies against MySQL
3. Session is created and redirected to homepage

**Exceptions**: Wrong credentials → show alert

## 3.4 Classes / Objects

### 3.4.1 <SearchHandler>

3.4.1.1 Attributes

class\_name

color\_filter

image\_content\_data

* + - 1. Methods

search\_by\_class\_and\_color()

### 3.4.2 <UserManager (PHP-side)>

3.4.1.1 Attributes

username

password\_hash

* + - 1. Methods

register()

login()

logout()

### 3.4.2 <HistoryManager>

3.4.1.1 Attributes

user\_id

search\_terms

* + - 1. Methods

save\_history()

get\_history()

## 3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

### 3.5.1 Performance

95% of search queries should return results within 2 seconds on localhost

### 3.5.2 Reliability

The Flask server should handle up to 20 concurrent users without crashing

### 3.5.3 Availability

The application should be usable 24/7 on a local server environment with 99% uptime during demos

### 3.5.4 Security

User sessions and passwords must be stored securely with hashing

### 3.5.5 Maintainability

Modular code structure in Flask and PHP to allow future replacement of the .pkl dataset or database schema

### 3.5.6 Portability

System must work in Chrome and Firefox on both Windows and macOS

## 3.6 Logical Database Requirements

## *MySQL database with a users table containing:*

## *id (INT, PK)*

## *username (VARCHAR, UNIQUE)*

## *password\_hash (VARCHAR)*

## *created\_at (TIMESTAMP)*

## *History tracking via a simple file or table (search\_history) storing:*

## *user\_id*

## *search\_terms*

## *timestamp*

## 3.7 Other Requirements

# *The system must load the .pkl file at startup*

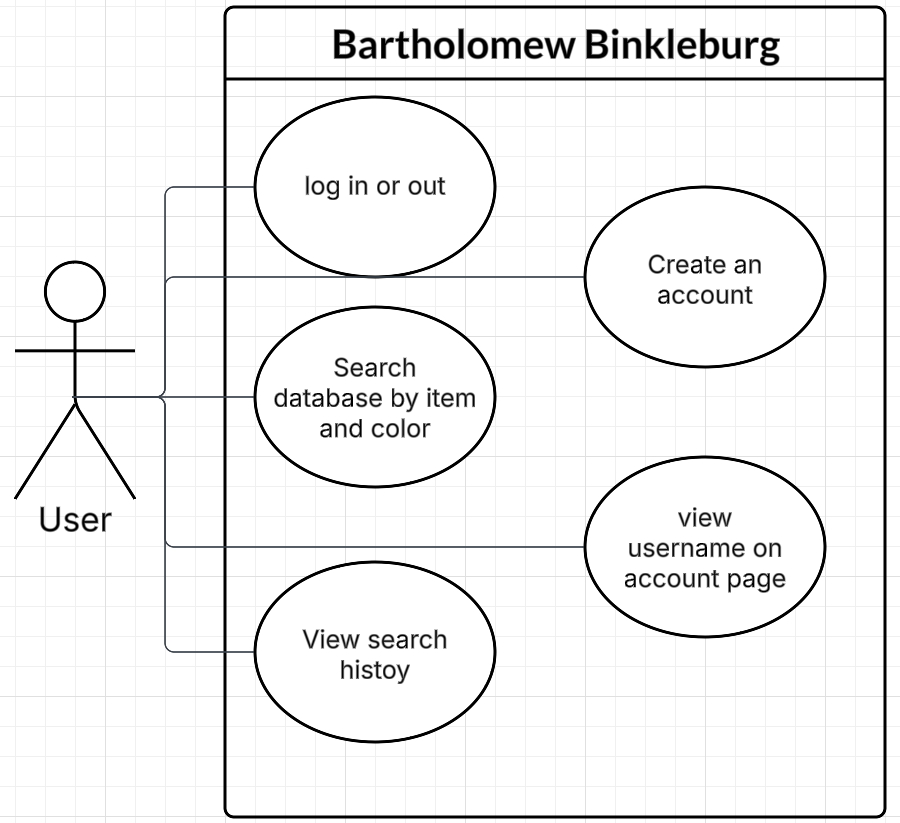
# *Color filters are approximated using RGB range matching*

# *The search interface must be responsive on screens >768px*

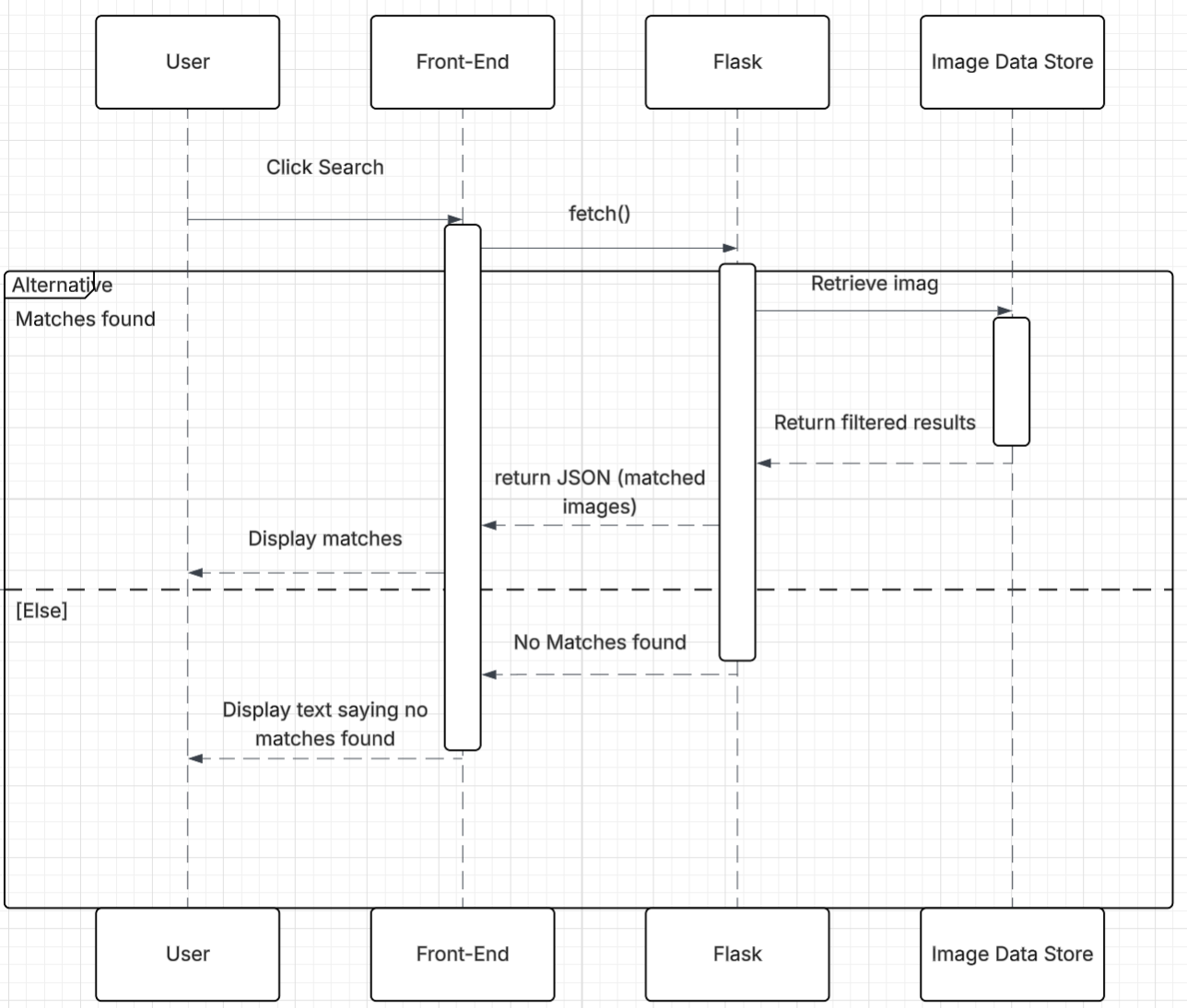
# 4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

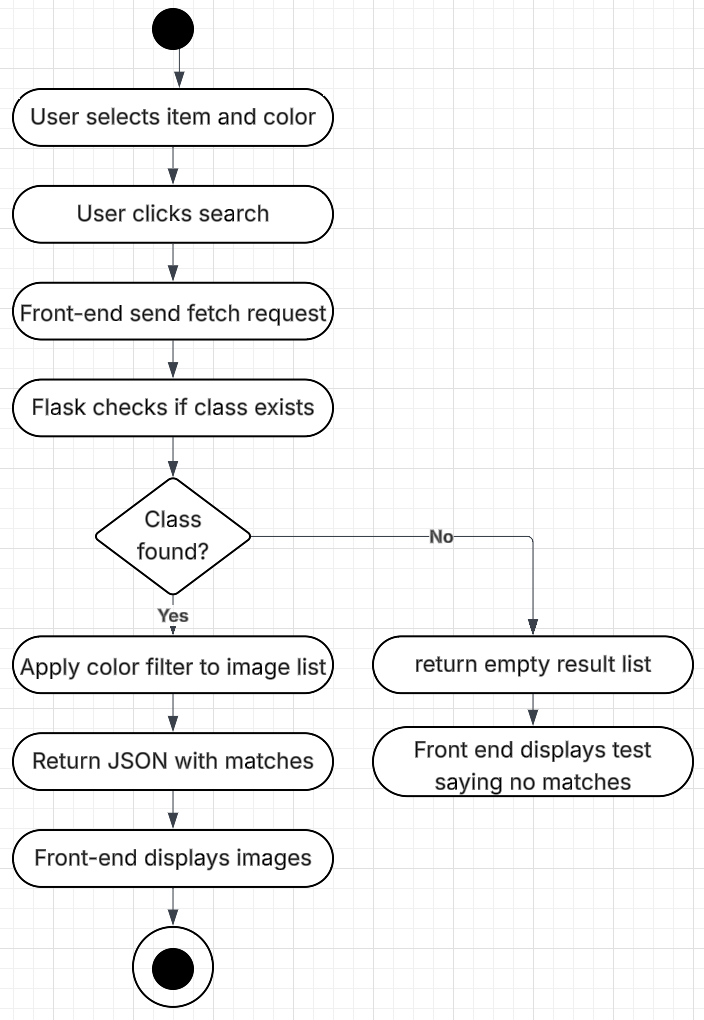
## 4.1 Use Case Diagrams



## 4.3 Sequence Diagrams



## 4.4 Activity Diagrams



# 5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

# A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS’s overall set of requirements.

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*