Software Requirements Specification

for

Context-Aware Keepsake Digitizer

Version 1.2

Prepared by Owen Cooke, Mahmud Jamal, Levi Thomas

University of Alberta - Faculty of Engineering

March 9th, 2025

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions **1**
1.3 Intended Audience and Reading Suggestions **1**
1.4 Product Scope **1**
1.5 References **2**
2. Overall Description **2**
2.1 Product Perspective 2
2.2 Product Functions **2**
2.3 User Classes and Characteristics **2**
2.4 Operating Environment **3**
2.5 Design and Implementation Constraints **3**
2.6 User Documentation **3**
2.7 Assumptions and Dependencies **3**
3. External Interface Requirements **4**
3.1 User Interfaces 4
3.2 Hardware Interfaces **5**
3.3 Software Interfaces **5**
3.4 Communications Interfaces **6**
4. System Features **7**
4.1 User Authentication **7**
4.2 View Collection 7
4.3 Image Access **8**
4.4 Image Metadata 9
4.5 Background Removal 9
4.6 Text Extraction 10
4.7 Filter Keepsakes 10
4.8 Create Keepsake 11
4.9 Bulk Create Keepsake 12
4.10 View Keepsake 13
4.11 Edit Keepsake 14
4.12 Delete Keepsake 14
4.13 Manual Create Collection 15
4.14 System-Recommended Create Collection 16
4.15 Edit Collection 16
4.16 Delete Collection 17
4.17 Collection Metadata 18
4.18 Map View 19
4.19 Collage Creation 19

4.20 Social Media Integration 19
4.21 Special Day Reminders 20
5. Other Nonfunctional Requirements 21
5.1 Performance Requirements 21
5.2 Safety Requirements 21
5.3 Security Requirements 21
5.4 Software Quality Attributes 21
5.5 Business Rules 21
6. Stakeholders 22
7. Other Requirements 26

Revision History

Name	Date	Reason For Changes	Version
ECE 493 Group 12	Jan 27, 2025	- Initial draft of functional requirements	0.1
ECE 493 Group 12	Feb 4, 2025	- Revised draft of functional requirements after stakeholder meeting - Added details regarding planned use of Expo for iOS and Android development	0.2
ECE 493 Group 12	Feb 7, 2025	- Approved completed document	1.0
ECE 493 Group 12	Feb 24. 2025	- Removed FR-3 - Removed FR-51 - Added View Collection system feature with FR-3	1.1
ECE 493 Group 12	Mar 9, 2025	- Removed FR-25 (view keepsake title) - Removed FR-29 (modify keepsake title)	1.2

1. Introduction

1.1 Purpose

This SRS document provides a complete description of the Context-Aware Keepsake Digitizer, the system function and its features. Context-Aware Keepsake Digitizer is an application designed to help users digitize, categorize, and manage keepsakes using contextual metadata such as dates, locations, text recognition, and image classification. This SRS describes the functional and non-functional requirements for the initial prototype of the system. It focuses on core features, including image submission, metadata extraction, organization, and retrieval. This SRS is a repository for all requirements that the Context-Aware Keepsake Digitizer should possess.

1.2 Document Conventions

This document follows IEEE 830-1998 guidelines for Software Requirements Specification. The following conventions are used:

- **Bold text** is used for section headings and functional requirement numbering.
- *Italic text* is used for emphasis where necessary.
- Functional requirements are numbered uniquely across system features.
- Priority levels follow a three-tier system: High, Medium, and Low.

1.3 Intended Audience and Reading Suggestions

This document is intended for the following stakeholders:

- Developers
- Project Managers & Project Owners
- Software Maintainers

Readers are encouraged to begin with *Product Scope (§1.4)* for an overview before diving into *System Features (§4)*. Developers should refer closely to *§3, §4, §5*, which outline system interfaces, features, constraints, and qualities.

1.4 Product Scope

The Context-Aware Keepsake Digitizer is a mobile application that allows users to store, categorize, and search keepsakes such as photographs, letters, or memorabilia. By leveraging image metadata, Optical Character Recognition (OCR), image classification, and customizable filters, the system enhances organization. The application's core objectives include:

- Simplifying keepsake digitization through image submission and background removal.
- Enhancing searchability using text recognition, image classification, and metadata-based filtering.
- Providing an intuitive user experience through collections, location-based views, and reminders.
- Facilitating easy sharing of collections by allowing users to generate collages from keepsakes and share them.

1.5 References

- IEEE 830-1998 Software Requirements Specification: https://standards.ieee.org/ieee/830/1222/
- Expo: https://expo.dev/
- React Native: https://reactnative.dev/
- Expo Go: https://expo.dev/go
- Expo Application Services (EAS): https://docs.expo.dev/eas/
- Supabase: https://supabase.com/
- FastAPI: https://fastapi.tiangolo.com/
- gluestack: https://gluestack.io/
- HeyAPI: https://heyapi.dev/

2. Overall Description

2.1 Product Perspective

Context-Aware Keepsake Digitizer is a mobile application that digitizes various user keepsakes, such as birthday cards, and stores them on the app. The app provides an interactive user interface, allowing users to store, categorize, and retrieve their keepsakes efficiently. The mobile application is implemented using TypeScript and React Native for cross-platform compatibility with iOS and Android. The backend is implemented using Python and integrates with Supabase.

2.2 Product Functions

Context-Aware Keepsake Digitizer will allow users to store, organize, and interact with their keepsakes:

- Authenticate using SSO
- View keepsakes and collections
- Upload keepsakes using the mobile device's camera or library
- Group keepsakes into collections manually or through recommended suggestions
- Edit keepsakes and collections
- Filter keepsakes
- Share keepsakes on social media
- Generate collage representations of keepsakes
- View collections on a map view

2.3 User Classes and Characteristics

The Context-Aware Keepsake Digitizer is meant to be used by anyone. There is only one type of user (*keepers*) who register using SSO and from there have access to the full functionality of the app.

2.4 Operating Environment

Context-Aware Keepsake Digitizer is a mobile application that can be run on Android 6+ and iOS 13.4+ devices. The app and database should be available for users 24 hours a day. The app should be available for download from the app store for both Android and iOS.

2.5 Design and Implementation Constraints

The app must adhere to the policies of the Google Play and Apple App Stores. Expo will be the core framework of the app. Database operations and machine learning features will rely on REST API endpoints called by the client. The app's bandwidth capacity and database size are set by Supabase; scaling to support high capacities will increase costs. SSO will rely on Google Identity Services. The implementation may be limited by the programming languages and frameworks listed in Table 2.7.1. The languages and frameworks are highly flexible, so no limitations are expected.

2.6 User Documentation

A tutorial video demonstrating the functionality of the app will be delivered along with the app.

2.7 Assumptions and Dependencies

Numerous software frameworks and libraries will be utilized as dependencies to build this project and are outlined in Table 2.7.1 below.

Table 2.7.1: External software packages required for application development, by tier

Tiers	Presentation	Application	Data
Dependencies	- expo - react-native - react-native-gesture-handler - react-native-reanimated - react-native-maps - gluestack-ui - @hey-api/client-fetch - @hey-api/openapi-ts - maestro - jest	- fastapi - pydantic - scikit-learn - pillow - rembg - pytesseract - keras - supabase-py - pytest	- Supabase - Google Cloud

It is assumed that the presentation tier's dependencies outlined above will be compatible with both Android and iOS devices through the Expo Go sandbox environment, which will be used for application development (outlined further in §3.2). Another assumption is that CPU or GPU-intensive libraries used in the application layer (such as keras) will be capable of running on the host machine used for development.

3. External Interface Requirements

3.1 User Interfaces

Upon opening the app for the first time, the user will be presented with a sign-in screen, which will display the app's logo and a button for signing in via an SSO provider. If the user is new, they will be presented with a screen to provide optional user information, such as their date of birth.

Once authenticated, the user is redirected to the main app layout. At the top of the view, the app logo and user's avatar are displayed. At the bottom of the view, a tab bar with icons is used for navigating through the main views in the app. The content of these tabs will be displayed in the middle of the main app layout, in between the top and bottom bars.

The first main view displays all the user's keepsakes in a grid view. The user can filter the keepsakes displayed in this view by selecting a filter icon that presents a sheet view to modify filters such as date, location, and keywords. A floating button at the bottom of the screen allows the user to create a new keepsake. A user can tap any of the keepsakes visible in the grid view and be brought to the individual keepsake screen.

On the individual keepsake screen, its associated images are displayed in a carousel-style view, with the specific details about the keepsake displayed below, such as caption and other metadata. A dropdown of action options includes editing, deleting, or sharing the keepsake with other apps.

When the user taps the create button, they are shown a sheet with a form for all the keepsake fields, such as image uploads, captions, and other metadata. An option for bulk creation allows the user to be directed to the bulk creation flow. Upon submission, the new item is added to the main keepsake grid view.

The second main view displays all the user's collections in a grid view. Similar to the keepsake view, it allows the user to create a collection via a floating button and they can tap on a specific collection to view it. An alternative map-view toggle exists, which upon user selection, displays all the collections as pins on a world map.

On the individual collection screen, the information associated with a collection such as its title, caption, location, and date is displayed. Keepsakes associated with the collection are presented in a grid view (similar to the main keepsake screen). Options for adding and removing one or more keepsakes from the collection are provided, as well as options to edit fields or delete the collection. An option to create a collage from the keepsakes present in the collection exists, which when clicked by the user presents them with a view of the generated collage and allows them to save or share the result to other platforms.

3.2 Hardware Interfaces

This application is intended to be run on mobile devices. The core development framework selected for this project is Expo, which is an open-source framework for building native mobile apps. The Expo SDK supports both Android 6+ and iOS 13.4+ as of January 31st, 2025.

Interactions with device-specific hardware will be handled through the Expo SDK, which provides generalized APIs to access device or hardware-specific features without writing native code. Most importantly, the expo-camera package will be used to interact with the device's camera and the expo-image-picker package will provide access to the device's system UI for selecting images.

As part of the development cycle, we will use Expo Go, a mobile application available on Android and iOS that provides a sandbox environment for Expo projects. By simply scanning a QR code generated by the Expo development server, you can run your project's code on a physical device without requiring a production build. It is important to note that some features are not supported by Expo Go, such as custom native code. However, JavaScript source code and all packages from the Expo SDK are supported.

Additionally, Expo Application Services (EAS) – a cloud platform that supports compiling builds and uploading apps to Google Play or the Apple App Store – has a free plan that could be used to create production builds of the application for both iOS and Android devices.

3.3 Software Interfaces

This application follows a three-tier architecture comprised of presentation, application, and data tiers. The presentation tier consists of a mobile client that serves as the user interface and provides access to mobile device features necessary for implementation. The data tier is responsible for storing and managing images, their related data, and user information. The application tier, implemented through a backend server, performs computationally expensive operations and facilitates communication between the presentation and data tiers.

The client, a cross-platform mobile application, will be powered by Expo - a React Native framework. Expo provides access to every device API through JavaScript/TypeScript wrappers, as well as defining a file-based routing system. Expo provides some cross-platform navigation components such as tabs and stacks, however, to minimize time spent styling components we will be using gluestack-ui to build the user interfaces described in §3.1. Gestures and motions (e.g., drag and drop) will be used for interactive actions within the app, facilitated by the react-native-gesture-handler and react-native-reanimated libraries. Additionally, to implement the map-based view for collections described in §4.17, we will utilize the react-native-maps library.

The mobile client will connect to the application tier through a Python3 backend server using the FastAPI framework. This server will perform computationally expensive operations (such as image processing) and will facilitate communication between the mobile client and the database. To ensure the consistency and ease of interaction between the client and server, the libraries: client-fetch and openapi-ts, provided by HeyAPI will be used to generate TypeScript types and methods for calling backend API's.

The Python libraries that will be leveraged to implement system features include:

- pydantic: for data validation, modelling database schemas, and specifying API responses
- scikit-learn: used to implement the system-recommended collection algorithm
- pillow: the Python Imaging Library, which enables image processing and manipulation

- rembg: uses machine learning to remove backgrounds from images
- pytesseract: for performing optical character recognition (OCR)
- keras: for performing image classification using pre-trained models and weights

The data tier will be provided by Supabase, a cloud provider for relational databases and object storage, to store user data such as keepsakes and collections. The supabase-py library will be used to connect the backend Python server to the cloud provider.

For authentication, the system will integrate with Google Identity Services to provide SSO capabilities using a user's existing Google Workspace account. The system will integrate Supabase and Google SSO to associate a user's SSO session with their information in the database.

Lastly, several software testing frameworks will be used to write test suites to validate this application's implementation and verify the specified functional requirements were met. For white-box testing, Jest will be used for the presentation tier, while Pytest will be used for the application tier. For black-box testing, the application's functionality will be tested end-to-end using Maestro, a UI testing framework for Expo and React Native.

3.4 Communications Interfaces

The backend server will provide multiple REST API endpoints that can be called by clients for CRUD database operations, as well as to perform image processing and give collection recommendations. The mobile client(s) will use the HTTP communication protocol to communicate with the server.

The message format for the body of the majority of the HTTP requests will be JSON. However, some APIs may choose to use multipart/form-data or image specifications for the Content-Type header in the HTTP request, particularly when handling image uploads or transformations.

For authentication, the OAuth 2.0 protocol will be utilized via Google Identity Services over HTTP to validate a user's credentials.

4. System Features

This section specifies the features of the system, described in detail as a scenario. Functional software requirements are extracted from each one. The requirements will be used for designing, implementing and testing the system.

4.1 User Authentication

4.1.1 Description and Priority

Users can register a new account after downloading the app by filling out a form. The system will use Single Sign-On (SSO) as a secure authentication method. Additional user information not provided by SSO (ex: birthday) will be collected in the registration form. Users need to be

registered to access the rest of the system. SSO will also enable subsequent logins to previously registered accounts, on the same or different mobile devices.

Priority: High

4.1.2 Stimulus/Response Sequences

Stimulus: User requests to create an account by filling out the registration form.

Response: System redirects the user to the SSO authentication page. Once SSO authenticates the user, the system stores user information provided by the user and SSO provider in the database. Otherwise, it displays an error message indicating failure.

Stimulus: User requests to login to an existing account.

Response: System redirects the user to the SSO authentication page. Once SSO authenticates the user, the user is redirected to the system's home page and the login session is stored on the device. Otherwise, an error message is displayed.

4.1.3 Functional Requirements

FR-1: The system shall allow users to register a new account using SSO.
FR-2: The system shall allow users to log in to an existing account using SSO.

4.2 View Collection

4.2.1 Description and Priority

Users can view their created collections in the app. The system shall provide a list of the users' collections, each collection in the list will provide a preview consisting of: a thumbnail, location if present, and date if present. If the user selects a collection from the list, the system will present the collections's information including all associated keepsakes in a carousel, title, date, and location. The user may select any keepsake from the collection to view it.

Priority: High

4.2.2 Stimulus/Response Sequences

Stimulus: The user selects a collection to view.

Response: The system retrieves and displays the collections's details, including the title, description, metadata, and associated keepsakes.

Stimulus: The user navigates through the keepsakes.

Stimulus: The user selects a keepsake.

Response: The system displays the selected keepsake in a keepsake view.

4.2.3 Functional Requirements

FR-3: The system shall allow the user to view all of their collections and select a collection to view its contents.

4.3 Image Access

4.3.1 Description and Priority

Users can submit images to the system using their mobile device, either by taking a new image with the device's camera or by uploading an existing image from the device's built-in photo library. The system shall also provide a reference overlay for where to position the camera when taking a new photo, to assist the user in capturing the intended keepsake.

Priority: High

4.3.2 Stimulus/Response Sequences

Stimulus: User selects option to take a new photo.

Response: System opens the mobile device's native camera app and allows the user to take a new picture, which is made available to the system.

Stimulus: User selects option to upload their existing photos.

Response: System opens the mobile device's native photos library and allows the user to select one or more existing pictures, which are made available to the system.

4.3.3 Functional Requirements

FR-4: The system shall allow users to submit an image by taking a new photo using the device camera.
FR-5: The system shall display a reference square overlay on the camera when taking a new photo.

FR-6: The system shall allow users to submit image(s) by selecting one or more existing photos from the device's library.

4.4 Image Metadata

4.4.1 Description and Priority

Images uploaded using Exchangeable Image File Format (EXIF) include additional metadata tags, such as date and time, location, and descriptions. These metadata fields will be extracted from the images uploaded by the user and used by the system to populate default information fields associated with each keepsake. The system will also use image classification to determine additional information about the keepsake.

Priority: High

4.4.2 Stimulus/Response Sequences

Stimulus: User uploads an image to the system.

Response: System performs image classification on the image. The results of the classification are combined with relevant EXIF metadata fields obtained from the image and used by the system to populate the default fields for a new keepsake.

4.4.3 Functional Requirements

FR-7: The system shall access metadata properties (EXIF, location) from submitted photos.
FR-8: The system shall perform image classification on a submitted photo and add the result to the image metadata.
FR-9: The system shall use a photo's metadata to populate the associated keepsake's fields.

4.5 Background Removal

4.5.1 Description and Priority

When a user uploads a photo of their keepsake, the image will likely include more than just the intended keepsake, such as items in the background or the surface it is placed on. To emphasize the specific keepsake, the system will automatically remove the background of uploaded images. The user should be able to accept or reject the result if they prefer to upload their original image.

Priority: Medium

4.5.2 Stimulus/Response Sequences

Stimulus: User uploads an image to the system.

Response: System performs background removal on the image. Upon successful removal, it displays the result to the user and allows them to confirm whether they would like to save the result or use the original image. If an error occurs, the original image is used.

4.5.3 Functional Requirements

FR-10: The system shall automatically remove the background of an image once uploaded and allow the user to specify if they want to use it or the original image.

4.6 Text Extraction

4.6.1 Description and Priority

Keepsakes may contain writing or words that can be used as an identifier. The system will attempt to extract any text from the uploaded image and store it in the database associated with the keepsake.

Priority: Low

4.6.2 Stimulus/Response Sequences

Stimulus: User creates a new keepsake with at least one associated image.

Response: For each image associated with the keepsake, the system performs image-based text extraction using an algorithm (ex: OCR). If successful, the results are stored in the database and associated with the keepsake. If the extraction fails, the result is ignored.

4.6.3 Functional Requirements

FR-11: The system shall extract text from user-submitted images and associate it with the keepsake.

4.7 Filter Keepsakes

4.7.1 Description and Priority

The user can view their keepsakes and apply filters through the app to restrict the viewable results. The system will provide multiple filter options that use database fields associated with the keepsakes to determine whether or not a keepsake should be included in the results presented to the user. Filters can be combined. The system should allow the user to easily clear previous filters.

Priority: Medium

4.7.2 Stimulus/Response Sequences

Stimulus: User selects a filter option and inputs a specific value.

Response: The system removes any keepsakes from the current view that do not match the filter requirements.

Stimulus: User removes a filter.

Response: The system adds keepsakes that didn’t match the previous filters back to the view.

4.7.3 Functional Requirements

FR-12: The system shall allow users to filter keepsakes by date.
FR-13: The system shall allow users to filter keepsakes by location.
FR-14: The system shall allow users to filter keepsakes by keywords derived from OCR results.
FR-15: The system shall allow users to filter keepsakes by keywords derived from user annotations/descriptions.
FR-16: The system shall allow users to filter keepsakes by image classification results.

4.8 Create Keepsake

4.8.1 Description and Priority

Users can create new keepsakes within the app. The system will provide a form for the user to add specific details related to the keepsake, including image uploading, a textual caption for describing

the keepsake's purpose or meaning, thumbnail selection and metadata properties such as location, date, and keywords. Once a user has filled out the form, they can submit it to add a new keepsake to their account.

Priority: High

4.8.2 Stimulus/Response Sequences

Stimulus: User requests to create a new keepsake.

Response: System redirects the user to the keepsake creation form.

Stimulus: User fills out the keepsake creation form and clicks submit.

Response: System validates that all form fields have been filled out properly. After validation, the system uploads the associated image(s) to object-based storage and stores the remaining keepsake fields in the database. If the keepsake creation is successful, the system shows the new keepsake in the user's app view. Otherwise, an error message is presented indicating why the validation or creation failed.

4.8.3 Functional Requirements

FR-17: The system shall allow users to add a textual caption to a keepsake.
FR-18: The system shall process the user-added textual caption into keywords and add them to the keepsake metadata.
FR-19: The system shall allow users to edit or add properties to a keepsake's metadata.
FR-20: The system shall allow users to add at least one image to a keepsake.
FR-21: The system shall allow users to specify the thumbnail for a keepsake, defaulting to the first image.

4.9 Bulk Create Keepsake

4.9.1 Description and Priority

Users may wish to digitize multiple keepsakes at once; however, the process of clicking create for each keepsake would be time-consuming. Therefore, users will have the option of bulk-creating multiple keepsakes at the same time. Users will be able to take or select multiple photos from their device and be able to sort them into groups. Each group represents one new keepsake, and the system will default to assigning one photo per group. Once the photos are grouped, the users will be able to specify details such as captions or other metadata for each group. Once the user is finished, the system will add each keepsake to the database in bulk.

Priority: Medium

4.9.2 Stimulus/Response Sequences

Stimulus: User requests to create multiple new keepsakes in bulk.

Response: System presents the user with the option to select multiple images from their device's photo library.

Stimulus: User uploads multiple images from their device.

Response: System displays a group view for new keepsakes, where each image defaults to an individual new keepsake.

Stimulus: User drags an image into a different group.

Response: The system updates the group view with the selected image added to the specified group. If the old group contains no other images, it is removed from the group view.

Stimulus: User taps a keepsake group.

Response: The system presents the keepsake details form, where the user can edit keepsake fields such as caption and metadata.

Stimulus: User taps the submit button on the bulk create page.

Response: The system validates the inputs for each new keepsake group. Once validated, it stores the keepsakes into the database in bulk. On success, the system prompts the user to add all new keepsakes to a new collection, which they can accept or reject. If the bulk creation fails, an error message displays the reason for invalid input or creation error.

4.9.3 Functional Requirements

FR-22: The system shall allow users to group specific images into multi-image keepsakes, defaulting to one image per keepsake.
FR-23: The system shall allow users to specify captions and metadata for each keepsake group.
FR-24: The system shall allow users to optionally add bulk-created keepsakes to a collection.

4.10 View Keepsake

4.10.1 Description and Priority

Users can view their created keepsakes in the app. The system shall provide a list of the users' keepsakes, each keepsake in the list will provide a preview consisting of: a thumbnail, location if present, and date if present. If the user selects a keepsake from the list, the system will present the keepsake's information including all associated images in a carousel format, textual caption, date, and location. The user may select any image from the keepsake to view it in fullscreen.

Priority: High

4.10.2 Stimulus/Response Sequences

Stimulus: The user selects a keepsake to view.

Response: The system retrieves and displays the keepsake's details, including the title, description, and metadata.

Stimulus: The user navigates through the keepsake's photos.

Response: The system presents the image(s) in a carousel view, allowing the user to scroll through them.

Stimulus: The user selects an image from the carousel view.

Response: The system displays the selected image in an enlarged image view.

4.10.3 Functional Requirements

FR-26: The system shall allow users to view keepsake's description.
FR-27: The system shall allow users to view keepsake's metadata.
FR-28: The system shall allow users to view photos associated with a keepsake in a carousel view.

4.11 Edit Keepsake

4.11.1 Description and Priority

Users may wish to modify or remove various attributes of a keepsake, including its title, description, images, thumbnail, and metadata. This functionality ensures that users can keep their stored keepsakes up-to-date and relevant to their needs.

Priority: High

4.11.2 Stimulus/Response Sequences

Stimulus: The user selects a keepsake and chooses to edit it.

Response: The system displays the current keepsake details and provides editing options.

Stimulus: The user modifies or removes the keepsake's title.

Response: The system updates the title and saves the changes.

Stimulus: The user modifies or removes the keepsake's description/annotation.

Response: The system updates the description and reflects the changes.

Stimulus: The user adds, modifies, or removes one or more images from the keepsake.

Response: The system updates the keepsake's image list accordingly.

Stimulus: The user changes the keepsake's thumbnail.

Response: The system updates the keepsake with the new thumbnail.

Stimulus: The user modifies or removes metadata associated with the keepsake.

Response: The system updates the metadata and saves the changes.

Stimulus: The user confirms and saves all edits.

Response: The system applies the changes and updates the keepsake in storage.

4.11.3 Functional Requirements

FR-30: The system shall allow users to modify or remove a keepsake's description/annotation.
FR-31: The system shall allow users to modify or remove images from a keepsake.
FR-32: The system shall allow users to change a keepsake's thumbnail.
FR-33: The system shall allow users to modify or remove a keepsake's metadata.

4.12 Delete Keepsake

4.12.1 Description and Priority

The user can permanently remove a keepsake from the system. Once deleted, all associated data, including the title, description, metadata, and images, will be erased and cannot be recovered.

Priority: Medium

4.12.2 Stimulus/Response Sequences

Stimulus: The user selects a keepsake and chooses the delete option.

Response: The system prompts the user for confirmation before proceeding.

Stimulus: The user confirms the deletion.

Response: The system permanently removes the keepsake and all its associated data.

Stimulus: The user cancels the deletion.

Response: The system retains the keepsake and no changes are made.

4.12.3 Functional Requirements

FR-34: The system shall allow users to permanently delete a keepsake.

4.13 Manual Create Collection

4.13.1 Description and Priority

The user can create a new collection to organize their keepsakes. Each collection requires a title, and users can optionally add a caption to provide additional context.

Priority: High

4.13.2 Stimulus/Response Sequences

Stimulus: The user initiates the creation of a new collection.

Response: The system presents a form for entering collection details.

Stimulus: The user enters a title for the collection.

Response: The system validates the input and ensures a title is provided.

Stimulus: The user optionally enters a caption.

Response: The system accepts and stores the caption.

Stimulus: The user confirms and saves the new collection.

Response: The system creates the collection and makes it available for use.

4.13.3 Functional Requirements

FR-35: The system shall allow users to create a new collection.
FR-36: The system shall require a title for the collection.
FR-37: The system shall allow users to add an optional caption to a collection.

4.14 System-Recommended Create Collection

4.14.1 Description and Priority

The system utilizes an algorithm(s) to analyze existing keepsakes and suggest relevant collections to the user. Users are notified of these recommendations and have the option to accept or reject them.

Priority: High

4.14.2 Stimulus/Response Sequences

Stimulus: The system analyzes existing keepsakes using its recommendation algorithm.

Response: The system generates suggested collections based on keepsake metadata.

Stimulus: A recommended collection is generated

Response: The system notifies the user of the suggested collection.

Stimulus: The user reviews the recommended collection.

Response: The system displays the proposed collection and the included keepsakes.

Stimulus: The user accepts the recommended collection.

Response: The system creates the collection and saves it in the user's library.

Stimulus: The user rejects the recommended collection.

Response: The system discards the recommendation without making any changes.

4.14.3 Functional Requirements

FR-38: The system shall use an algorithm to recommend collections of existing keepsakes.
FR-39: The system shall notify users of recommended collections.
FR-40: The system shall allow users to accept or reject a recommended collection.

4.15 Edit Collection

4.14.1 Description and Priority

The user may wish to modify an existing collection by adding or removing keepsakes, updating the title, and modifying or deleting the collection's caption and metadata. This functionality extends upon the *Manual Create Collection* feature, allowing the user to add their keepsakes and modify the collection metadata used for the *Map View* and *Collage Creation* features.

Priority: High

4.15.2 Stimulus/Response Sequences

Stimulus: The user selects a collection to edit.

Response: The system displays the selected collection's details and available editing options.

Stimulus: The user adds one or more existing keepsakes to the collection.

Response: The system updates the collection to include the selected keepsakes.

Stimulus: The user removes one or more keepsakes from the collection.

Response: The system updates the collection and removes the specified keepsakes.

Stimulus: The user modifies the collection's title.

Response: The system updates the collection with the new or removed caption.

Stimulus: The user modifies or removes the collection's metadata.

Response: The system updates the metadata and applies the changes.

Stimulus: The user saves the changes.

Response: The system applies the modifications and updates the collection.

4.15.3 Functional Requirements

FR-41: The system shall allow users to add one or more existing keepsakes to a collection.
FR-42: The system shall allow users to remove one or more keepsakes from a collection.
FR-43: The system shall allow users to modify a collection's title.
FR-44: The system shall allow users to modify or remove a collection's caption/description.
FR-45: The system shall allow users to modify or remove a collection's metadata.

4.16 Delete Collection

4.16.1 Description and Priority

Users can delete collections. After deletion, the keepsakes within the collection will be moved to the main view and will not be part of a collection.
Priority: High

4.16.2 Stimulus/Response Sequences

Stimulus: The user requests to delete a collection.

Response: The system moves the keepsakes from the collection to the main view and deletes the collection.

4.16.3 Functional Requirements

FR-46: The system shall allow users to permanently delete a collection without deleting the keepsakes contained within it.

4.17 Collection Metadata

4.17.1 Description and Priority

The system should examine the metadata of keepsakes in a collection to automatically set the collection's location and date. The user can update the location, date, and other metadata of the collection or remove its metadata.
Priority: Medium

4.17.2 Stimulus/Response Sequences

Stimulus: The user creates a collection.

Response: The system examines the keepsakes' metadata to set the collection's date.

Response: The system examines the keepsakes' metadata to set the collection's location.

Stimulus: The user requests to modify a collection's location.

Response: The system presents an option to modify the collection's location.

Stimulus: The user edits the collection's location.

Response: The system updates the collection's location.

Stimulus: The user requests to modify a collection's time/date.

Response: The system presents an option to modify the collection's time/date.

Stimulus: The user edits the collection's time/date.

Response: The system updates the collection's time/date.

Stimulus: The user requests to modify or remove a collection's metadata.

Response: The system presents an option to modify or remove the collection's metadata.

Stimulus: The user selects the option to modify or remove the collection's metadata.

Response: The system updates the collection's metadata or removes it.

4.17.3 Functional Requirements

FR-47: The system shall automatically derive collection date from keepsakes' metadata contained within.

FR-48: The system shall automatically derive collection location from keepsakes' metadata contained within.
FR-49: The system shall allow users to modify a collection's location.
FR-50: The system shall allow users to modify a collection's time/date.

4.18 Map View

4.18.1 Description and Priority

Users can view their collections displayed on a map. The collections' metadata will be used to determine their location. Users can click on a collection on the map to view the collection.
Priority: Medium

4.18.2 Stimulus/Response Sequences

Stimulus: The user requests to visualize the collections on a map.

Response: The system opens a map and displays the collections on it.

Stimulus: The user clicks a collection on the map.

Response: The system opens the collection, so the user can view it.

4.18.3 Functional Requirements

FR-51: The system shall visually depict collections with location metadata on a geographical map.

FR-52: The system shall allow users to click on a collection's location marker to view the collection.

4.19 Collage Creation

4.19.1 Description and Priority

Users have the option to generate a collage representing a collection. The collage will be composed of some or all of the keepsakes in the collection.
Priority: Medium

4.19.2 Stimulus/Response Sequences

Stimulus: The user selects a collage and requests to create a collage representation.

Response: The system generates a collage representation of some or all of the keepsakes.

4.19.3 Functional Requirements

FR-53: The system shall allow users to select a collection and have the system generate a collage representation of some keepsakes in the collection.

4.20 Social Media Integration

4.20.1 Description and Priority

Users have the option to share their keepsakes and collages on social media platforms. An action sheet should appear containing social media platforms where the user can share their keepsakes and collages.
Priority: Medium

4.20.2 Stimulus/Response Sequences

Stimulus: The user requests to share a keepsake.

Response: The system displays a window to share the keepsake to social media platforms.

Stimulus: The user selects a platform.

Response: The system shares the keepsake on the platform.

Stimulus: The user requests to share a collage.

Response: The system displays a window to share the collage to social media platforms.

Stimulus: The user selects a platform.

Response: The system shares the collage on the platform.

4.20.3 Functional Requirements

FR-54: The system shall allow users to share a keepsake to social media platforms
FR-55: The system shall allow users to share a generated collage for a collection to social media platforms.

4.21 Special Day Reminders

4.21.1 Description and Priority

Users can receive a push notification on their device on noteworthy days, with suggestions to use the application.

Priority: Low

4.21.2 Stimulus/Response Sequences

Stimulus: The system checks if the current day is special (ex. birthdays).

Response: The system pushes a notification to the user.

4.21.3 Functional Requirements

FR-56: The system shall notify the user on special days to remind users to save their keepsakes.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system should provide a smooth and responsive experience for users while interacting with keepsakes. Image uploads and metadata extraction should be completed within a few seconds under normal network conditions. Background removal should process images in a reasonable time, aiming for an average of five seconds per image. Searching/filtering keepsakes should return results within two seconds for most queries. The first version of the system will focus performance optimizations on core functionality rather than scalability, but the system should be able to handle at least one hundred concurrent users without significant slowdowns.

5.2 Safety Requirements

To prevent accidental data loss, the system should include confirmation prompts before a user permanently deletes a keepsake or collection. While there will be no automated data recovery feature, a soft deletion may be considered, where deleted keepsakes remain recoverable for a short period before permanent removal. The system will not edit the image metadata directly, nor will it destructively modify the uploaded images.

5.3 Security Requirements

User authentication should be handled through Single-Sign-On (SSO) to ensure a secure login process, without requiring the system to store passwords. User-submitted data, including images and metadata, should be transmitted securely using HTTPS. Basic access controls should be implemented to ensure that only authenticated users can manage their own keepsakes and collections. The first version of the system will follow security best practices to prevent common vulnerabilities, however, more advanced features will not be prioritized.

5.4 Software Quality Attributes

The system should be intuitive and user-friendly, allowing new users to upload and organize keepsakes without extensive guidance. Reliability is important, but given the scope of the first version of this project, availability and failover-esque mechanisms will not be a priority. Maintainability is a key concern, and the software should be modular and well-documented to allow future improvements and feature expansions. While scalability is not a focus, the architecture should not impose unnecessary constraints that will hinder future growth. Interoperability with common mobile devices should be ensured, but support for legacy systems will not be required.

5.5 Business Rules

Users must be authenticated to create, modify, or delete keepsakes and collections. A keepsake must contain at least one image to be saved, and collections must have a title. The system may suggest collections based on metadata, but users will have the ability to accept or reject these suggestions.

6. Stakeholders

Stakeholder	Role	Influence and Power	Risk Level	Interest Level	Importance
Users (aka "Keepers")	Keepers are registered users who can upload and manage keepsakes on the app.	High Keepers are external stakeholders whose needs must be satisfied on the app to encourage them to use the app.	Medium Keepers' actions on the app are private and have no risk on the app's reputation. But, they are the primary audience for the app and their usage will dictate the project's success.	High The app allows keepers to save keepsakes that take up space but are too valuable to throw away. Keepers will be interested in the app as it solves an important problem.	High Keepers are the main and only users of the app. The entire app is based on being useful to keepers.
Developers	The current capstone team members (Mahmud, Levi, and Owen). Responsible for implementing and testing the software necessary to complete the project.	High The developers are internal stakeholders. The resulting app depends entirely on their implementation and ability to meet the product requirements.	High Missing or delayed features can cause negative feedback from users, which could lead to churn or negative app reviews.	High The developers share the goal of wanting to build a useful product. They also want to use the app for themselves.	High Without developers, there will be no application features for users to use.
Product Owners	Product owners define the vision, strategy and priorities for	High They are internal stakeholders	Medium If requirements are not well-	High They are deeply invested in the	High The owner defines the product use

	the application. They ensure the project aligns with business goals and user needs.	who make key decisions on features, budget and timelines, affecting the overall project direction.	defined, scope creep or misalignment of user needs can occur.	success of the product and the application is designed to satisfy the owner's requirements.	cases and features for the product, and their decisions impact the viability and functionality of the product.
Software Maintainers	Maintainers are future developers who will continue working on the project (to add new features and fix issues with old ones).	Medium As external stakeholders, maintainers may have difficulty understanding the system without proper documentation and code structure. This influences how current developers build the app.	High Future maintainers may not be able to continue work on the existing system if the codebase and documentation are done poorly. This could waste a large amount of project resources.	Low Maintainers may be interested in the project's goals and improving its usability.	High Without performing maintenance on the software, future user needs and fixes will go unresolved, causing churn.
Office of the Privacy Commissioner of Canada	The Office of the Privacy Commissioner (OPC) is an agency of the Government of Canada responsible for the protection of personal information.	Low As an external stakeholder, they influence the rules the app must adhere to regarding the storage of personal information.	Low The OPC will not be contributing to the project. Personal data storage rules have been set and the project is in a prototype phase with minimal users.	High The OPC is heavily invested in ensuring that personal information is protected.	High The rules set by the OPC will ensure that the personal data stored on the project is safe.

Supabase (Cloud Provider)	Provides managed PostgreSQL database services and cloud object storage for storing user keepsakes and metadata.	Medium Supabase is an external stakeholder that controls data storage, availability, and security, but the app's logical components remain under developer control.	Medium Service outages, API limitations, or pricing model changes could affect data availability and performance.	Low Their primary interest is in maintaining reliable services and customer retention. They do not directly care about the application's success.	High The application's ability to store, receive, and manage user keepsakes depends on their database services.
App Stores	App Stores are responsible for advertising the final app, enabling users to download it to their device, and collecting reviews.	Low As an external stakeholder, they provide rules and criteria that the app must adhere to, during both development and release.	High Apps must be approved by the App Store admins before they can be released. Users will not be able to download the app or not trust the app if it is not on their device's app store.	Low Currently, there are 8.93 million apps worldwide. The App Stores do not care whether this new project launches.	Medium App Stores help make the software accessible to users and encourage them to download if reviews are high.
Open Source Library Maintainers	Develops and maintains one or more of the libraries that the project depends on.	Medium The maintainer, an external stakeholder, controls updates, bug fixes, feature additions, and feature deprecations	Medium If the maintainer stops maintaining the library or introduces breaking changes, it could impact the project.	Low If issues are reported or our developers are active in their community, they may be interested in this project. But,	High The project depends on their library. Any significant changes to the library could affect project stability and performance.

		for the library, but they do not directly dictate the project's direction.	However, open-source alternatives or forking the project may mitigate this risk.	considering the prototype scope, it's unlikely they will engage with the project.	
Google Cloud	Facilitates Single Sign-On (SSO) to the application by using a user's pre-existing Google Workspace credentials.	Low Many alternative external stakeholders provide SSO (Apple, Facebook, Microsoft, etc.) or other authentication methods. So they are a low-influence external stakeholder.	Low As mentioned previously, if Google SSO is incompatible or too difficult to integrate, an alternative provider can be used for authentication	Low If the application scales to many users, they will be able to profit; but in the current prototype state with no users, their interest is low.	Medium The ability to use the application's features requires the user to be authenticated. Data cannot be stored, associated, or retrieved without a user identifier.
Expo Application Services (EAS)	Provides cloud-based build, testing, and deployment services for the project.	Medium As an external stakeholder, EAS controls infrastructure for running E2E tests and deploying the app to app stores. Service disruptions or policy changes directly impact the development of the project.	Medium Disruptions in EAS could delay testing. However, alternative CI/CD solutions exist that could be used instead if necessary.	Low If the project moves beyond a prototype state, EAS may make a profit off of more frequent production builds and service usage.	Low Currently, EAS is only to enable E2E testing, not for compiling or submitting production builds.

		Currently, deployment to app stores is not a priority, reducing their influence.			

7. Other Requirements

Given the current scope of the project, there are no additional requirements at this time. In the future, if the project is launched and made available to the public, considerations for legal requirements and privacy concerns should be addressed in a subsequent version of this document.