Software Design Description (SDD)

Instructions: https://bjohnson.lmu.build/cmsi4071web/sddesign.html

6.1. Introduction

System Objectives

The **Momento** app aims to simplify the process of visual journaling by offering users a unified, aesthetically pleasing platform to capture daily moments, moods, and memories without the need for written entries. Development activities will include gathering requirements, designing the system, implementing key features, integrating relevant APIs, conducting testing, and deploying the application.

6.1.2 Hardware, Software, and Human Interfaces

Mobile app using swift as the primary language. The human interface aspect of the app plays into the images that the user can upload or click directly through the app.

6.2 Architectural Design

The architectural design consists of a) open screen with daily mood check: with camera app integration, b0) welcome/home page with actions option buttons, b1) button to capture a new journal entry: camera roll and camera app integration, b2) journal archive: log of all user's previous journal entries with dates, data stores in firebase, b3) monthly journal recaps: a monthly collection and mood assessment.

6.2.1 Major Software Components

Swift ui: various pages/views, buttons to lead to those views, database to store entries.

6.2.2 Major Software Interactions

Camera capture, photo upload, and optional written notes taken by the user.

6.2.3 Architectural Design Diagrams

6.3. CSC and CSU Descriptions

Memento App CSCI

Memento App CSCI: A photo-first journaling application designed for users to capture, store, and reflect on their daily moments through images and videos, focusing on aesthetic and emotional engagement.

Computer Software Components

1. Photo Capture CSC
    - Description: Manages all functionalities related to capturing and processing photos.
    - Components:
        - CameraInterface CSU: Handles camera functionalities, user permissions, and photo capture.
        - PhotoFilter CSU: Applies vintage filters to captured images.
        - PhotoStorage CSU: Organizes and stores photos by date in a clean timeline.
2. Mood Tracking CSC
    - Description: Facilitates mood check-ins and emotional tracking.
    - Components:
        - MoodCheckInInterface CSU: Manages the user interface for daily mood check-ins (morning and evening).
        - MoodStorage CSU: Records and associates mood entries with the respective daily photos.
        - MoodVisualization CSU: Creates visual representations of user moods over time.
3. Recap & Interaction CSC
    - Description: Handles monthly recaps and user interactions with their captured moments.
    - Components:
        - RecapDisplay CSU: Generates and displays monthly photo recaps as slideshows or collages.
        - InteractiveDetail CSU: Allows users to click on photos for additional details (dates, moods, tags).
        - TemplateManager CSU: Provides customization options for recap layouts and filters.
4. Social Sharing CSC
    - Description: Manages the functionality for sharing moments on social media.
    - Components:
        - ShareInterface CSU: Facilitates the sharing of photos and recaps to Instagram.
        - TemplateCreator CSU: Provides aesthetic templates and text overlay options for social media posts.

5. Tagging & Organization CSC
    - Description: Handles tagging and grouping of photos for better organization.
    - Components:
        - TagManager CSU: Allows users to add, edit, and manage tags for their photos.
        - CollectionViewer CSU: Enables users to view and filter their photos by tags and create themed collections.

Example of a Computer Software Unit (CSU) Description

PhotoCapture CSC

- CameraInterface CSU
    - Purpose: Manages camera functionalities, including permissions and photo capture.
    - Methods:
        - initializeCamera(): Initializes the camera interface.
        - capturePhoto(): Captures the photo and triggers the filter application.
- PhotoFilter CSU
    - Purpose: Applies a vintage disposable camera filter to captured images.
    - Methods:
        - applyFilter(image): Applies the vintage filter to the provided image.
        - getFilteredPhoto(): Returns the filtered photo for storage.
- PhotoStorage CSU
    - Purpose: Organizes and stores captured photos by date.
    - Methods:
        - savePhoto(photo, date): Saves the photo under the specified date.
        - getPhotosByDate(date): Retrieves photos for a specific date.

6.3.1 Class Descriptions

6.3.2 Detailed Interface Descriptions

1. Photo Capture CSC

CameraInterface CSU

- Purpose: Facilitates communication between the app and the device's camera hardware.
- Key Interfaces:
    - CaptureRequest Interface:
        - Method: requestCapture()
        - Description: Triggers the camera to capture a photo and notifies the UI when the capture is successful.

    - PermissionCheck Interface:
        - Method: checkPermissions()
        - Description: Checks and requests camera permissions from the user.

PhotoFilter CSU

- Purpose: Manages the application of filters to photos.
- Key Interfaces:
    - FilterApplication Interface:
        - Method: applyFilter(photo)
        - Description: Receives a raw photo and returns the filtered version.
    - FilterSettings Interface:
        - Method: getFilterSettings()
        - Description: Provides access to the current filter settings used for photo processing.

PhotoStorage CSU

- Purpose: Handles storage and retrieval of photos.
- Key Interfaces:
    - StorageManagement Interface:
        - Method: savePhoto(photo, date)
        - Description: Saves the provided photo under the specified date.
    - RetrievalInterface:
        - Method: retrievePhotosByDate(date)
        - Description: Fetches all photos stored for a specific date.

---

2. Mood Tracking CSC

MoodCheckInInterface CSU

- Purpose: Manages user mood input and tracking.
- Key Interfaces:
    - MoodInput Interface:
        - Method: logMood(mood, time)
        - Description: Records the user's mood at a specific time and links it to the current photo.
    - MoodDisplay Interface:
        - Method: displayMoodOptions()
        - Description: Presents available mood options to the user for selection.

MoodStorage CSU

- Purpose: Organizes and manages stored mood data.
- Key Interfaces:
    - MoodDataManagement Interface:
        - Method: saveMoodEntry(moodEntry)
        - Description: Saves a mood entry associated with a photo.

    - MoodRetrieval Interface:
        - Method: getMoodEntriesByDate(date)
        - Description: Retrieves all mood entries for a given date.

MoodVisualization CSU

- Purpose: Creates visual representations of mood data.
- Key Interfaces:
    - MoodChart Interface:
        - Method: generateMoodChart(data)
        - Description: Accepts mood data and produces a visual chart representation.
    - TrendAnalysis Interface:
        - Method: analyzeMoodTrends()
        - Description: Analyzes and summarizes mood trends over a specified period.

---

3. Recap & Interaction CSC

RecapDisplay CSU

- Purpose: Manages the display of monthly recaps.
- Key Interfaces:
    - SlideshowInterface:
        - Method: startSlideshow(recapData)
        - Description: Initiates a slideshow of the user's monthly recap.
    - RecapInteraction Interface:
        - Method: onPhotoClick(photoId)
        - Description: Handles interactions when a user clicks on a specific photo in the recap.

InteractiveDetail CSU

- Purpose: Allows user interaction with recap details.
- Key Interfaces:
    - DetailRetrieval Interface:
        - Method: fetchPhotoDetails(photoId)
        - Description: Retrieves detailed information (date, mood, tags) for a specific photo.
    - UserFeedback Interface:
        - Method: submitFeedback(feedback)
        - Description: Collects user feedback on the recap experience.

TemplateManager CSU

- Purpose: Provides customization options for recap layouts.
- Key Interfaces:
    - TemplateSelection Interface:

        - Method: selectTemplate(templateId)
        - Description: Allows users to choose a layout template for their recap.
    - TemplatePreview Interface:
        - Method: previewTemplate(templateId)
        - Description: Displays a preview of the selected template.

---

4. Social Sharing CSC

ShareInterface CSU

- Purpose: Manages sharing of photos to social media.
- Key Interfaces:
    - ShareRequest Interface:
        - Method: initiateShare(photo, platform)
        - Description: Begins the sharing process for a selected photo on a specified platform.
    - ShareConfirmation Interface:
        - Method: confirmShare()
        - Description: Confirms that the photo has been successfully shared.

TemplateCreator CSU

- Purpose: Provides aesthetic templates for social media posts.
- Key Interfaces:
    - TemplateCustomization Interface:
        - Method: customizeTemplate(templateId, overlayText)
        - Description: Allows users to add custom text overlays to templates.
    - TemplateOutput Interface:
        - Method: getCustomizedTemplate()
        - Description: Returns the final version of the customized template for sharing.

---

5. Tagging & Organization CSC

TagManager CSU

- Purpose: Facilitates tagging and organization of photos.
- Key Interfaces:
    - TagCreation Interface:
        - Method: createTag(tagName)
        - Description: Creates a new tag for categorizing photos.
    - TagAssignment Interface:
        - Method: assignTag(photoId, tagName)
        - Description: Associates a tag with a specific photo.

CollectionViewer CSU

- Purpose: Allows users to view and filter tagged photos.

- Key Interfaces:
    - CollectionRetrieval Interface:
        - Method: getCollectionByTag(tagName)
        - Description: Retrieves all photos associated with a specific tag.
    - CollectionDisplay Interface:
        - Method: displayCollection(collectionData)
        - Description: Displays a collection of photos based on the selected tag.

6.3.3 Detailed Data Structure Descriptions

Photo Capture CSC

PhotoDataPacket

- Description: The PhotoDataPacket structure represents the data format used for transferring photo information between components of the application. It encapsulates details necessary for photo processing and storage.
- Fields:
    - photoId: A unique identifier for the photo (string).
    - imageData: The raw image data, typically in a byte array format.
    - dateTaken: The timestamp indicating when the photo was captured (datetime).
    - filterApplied: A boolean flag indicating whether a filter has been applied to the photo.
- Usage: This data structure is used when photos are captured, stored, or retrieved, ensuring that all necessary metadata accompanies the image.

Mood Tracking CSC

MoodEntryData

- Description: The MoodEntryData structure encapsulates the data related to mood entries, including emotional states and timestamps.
- Fields:
    - mood: A string representation of the mood (e.g., "Happy", "Anxious").
    - timestamp: The date and time when the mood was logged (datetime).
    - photoId: The ID of the photo associated with this mood entry (string).
- Usage: This structure is utilized when logging mood entries, allowing for easy association of mood data with specific photos.

MoodDataCollection

- Description: The MoodDataCollection structure is a collection of MoodEntryData objects, providing a way to store multiple mood entries for a specific date.
- Fields:

    - entries: An array or list of MoodEntryData objects.
- Usage: This structure is used to manage and retrieve mood entries for a particular date, enabling efficient data handling when displaying mood trends.

Recap & Interaction CSC
RecapData

- Description: The RecapData structure contains the necessary information to create and display monthly recaps for users.
- Fields:
    - month: The month and year of the recap (string).
    - photos: An array or list of PhotoDataPacket objects representing the photos taken during that month.
    - moods: An array or list of MoodEntryData objects linked to the photos.
- Usage: This structure facilitates the generation of monthly recaps, providing all relevant data in one package for easier processing and display.

Social Sharing CSC
ShareTemplate

- Description: The ShareTemplate structure defines the format of templates used for social media sharing, including customizable elements.
- Fields:
    - templateId: A unique identifier for the template (string).
    - backgroundImage: The background image used in the template (string or byte array).
    - overlayText: A string for text overlay, customizable by users.
    - dimensions: An object containing width and height for the template layout (object).
- Usage: This structure is used to define and manage templates for sharing posts, allowing users to customize their social media content.

Tagging & Organization CSC
TagData

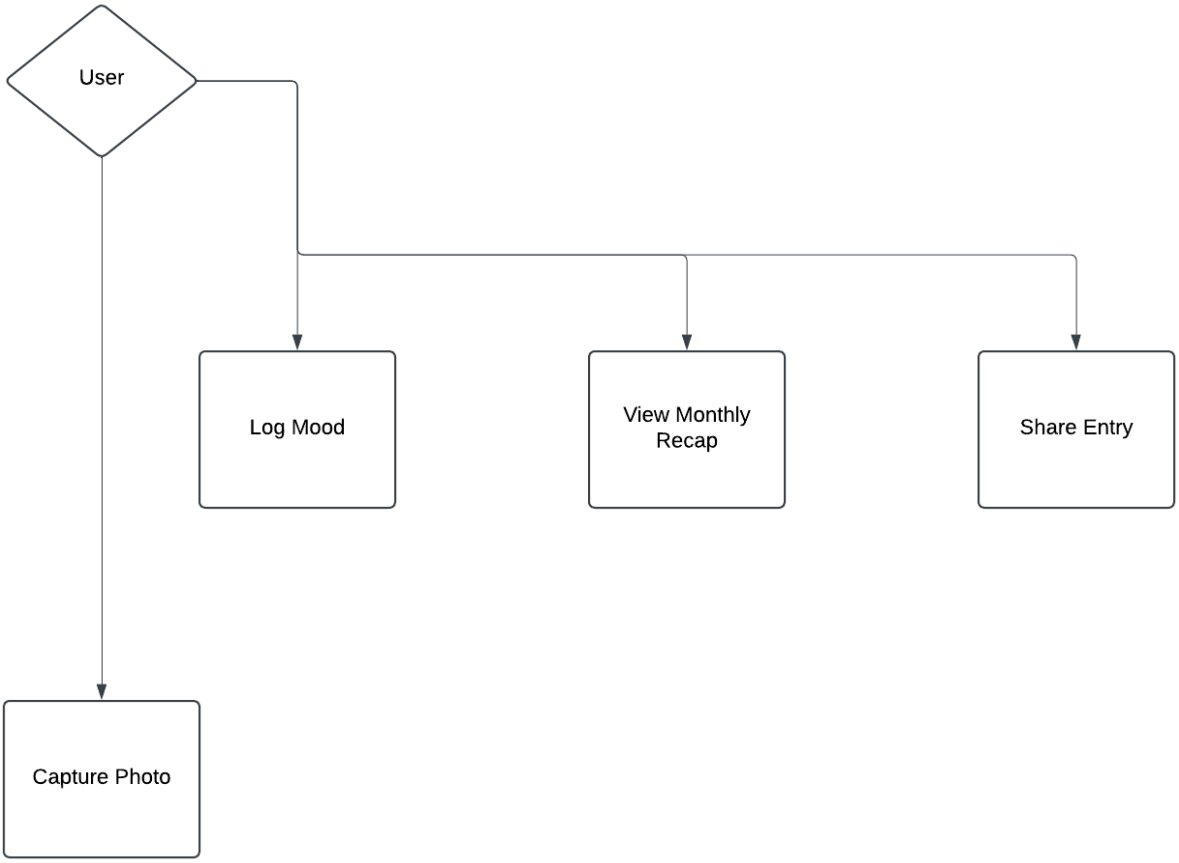
- Description: The TagData structure represents the information related to a tag used for categorizing photos.
- Fields:
    - tagId: A unique identifier for the tag (string).
    - tagName: The name of the tag (string).
    - associatedPhotos: An array or list of photo IDs associated with this tag.
- Usage: This structure is used to manage and organize tags within the application, enabling users to filter and categorize their photos effectively.

TagCollection

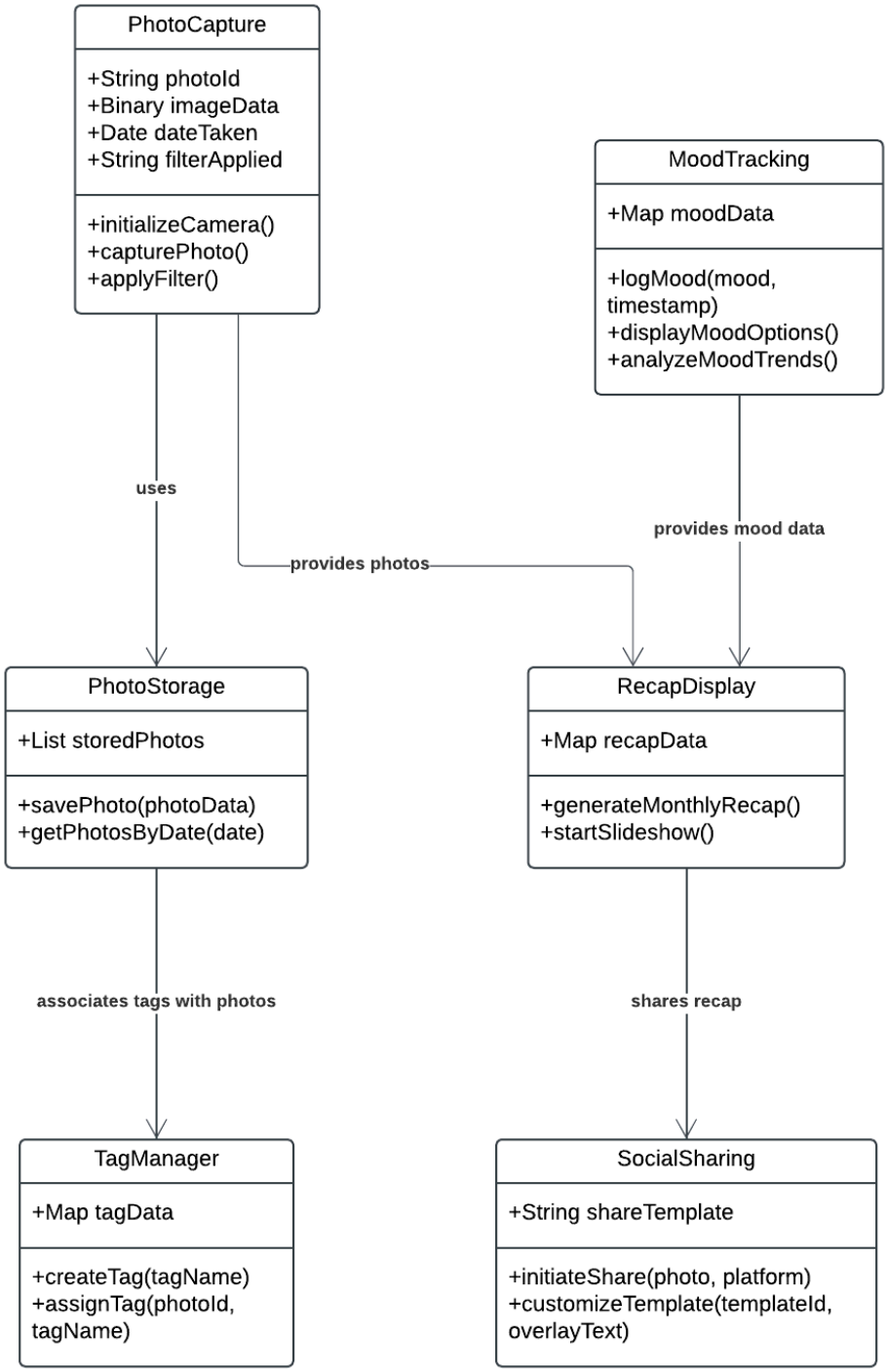
- Description: The TagCollection structure is a collection of TagData objects, facilitating bulk operations on multiple tags.

6.3.4 Detailed Design Diagrams

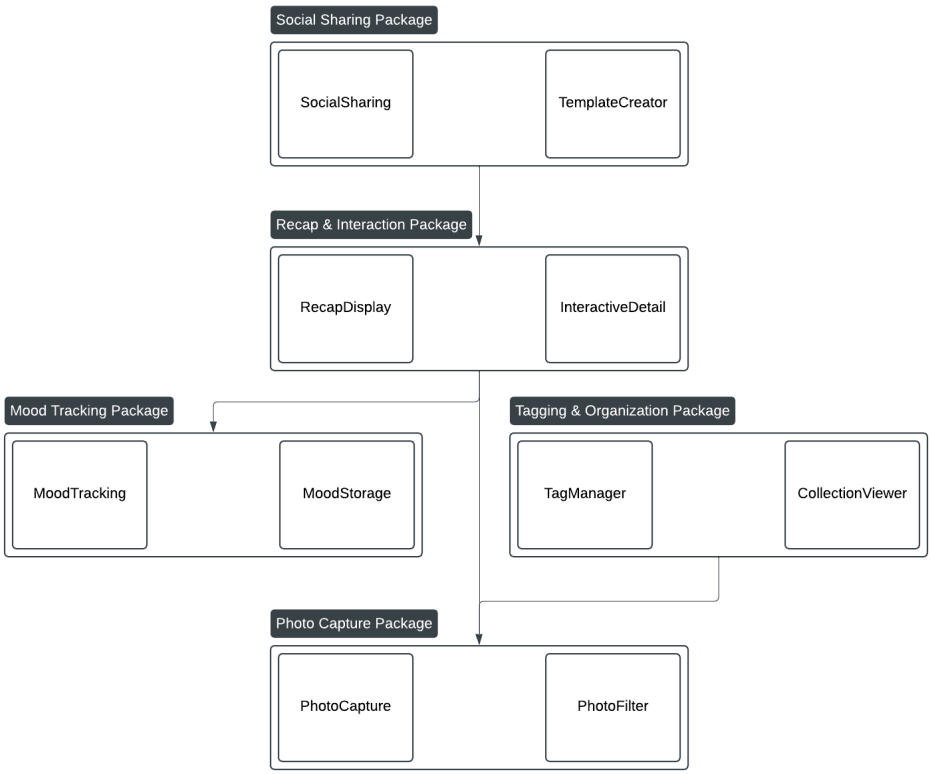
1. **Use Case Diagram**



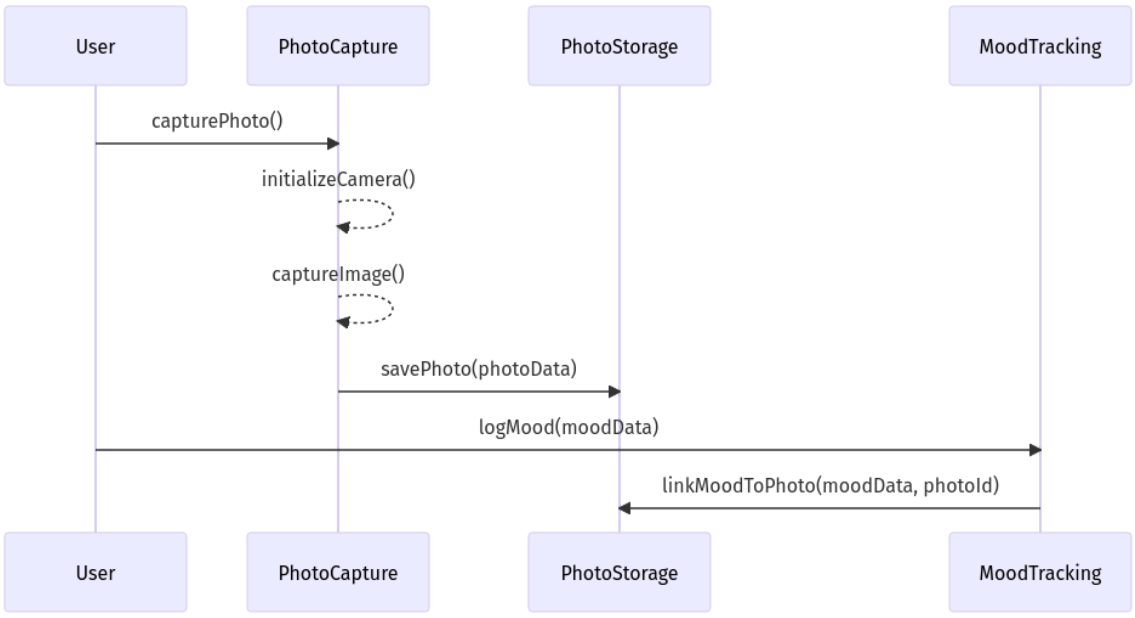
2. **Class Diagram**



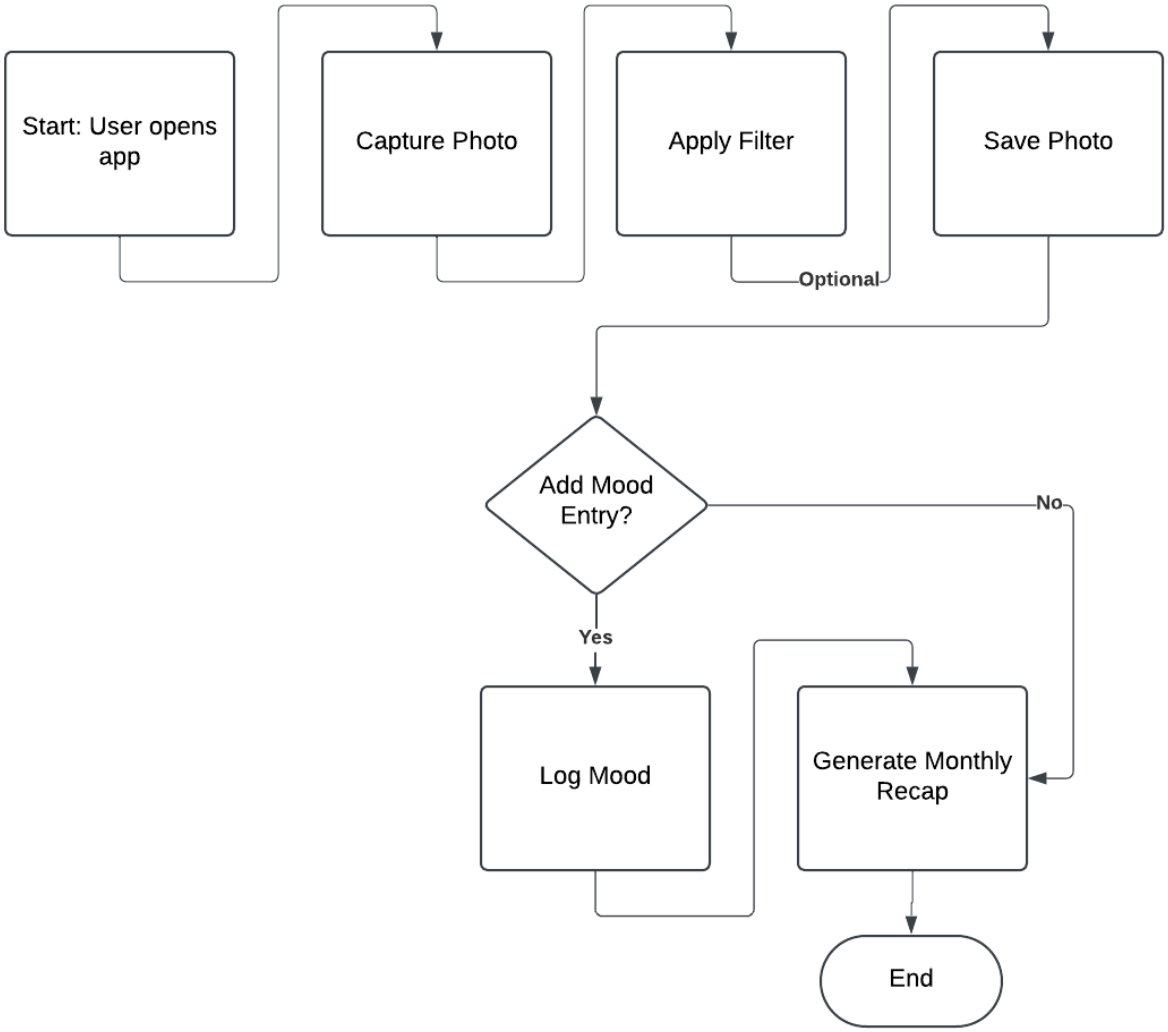
3. Package Diagram



4. Sequence Diagram

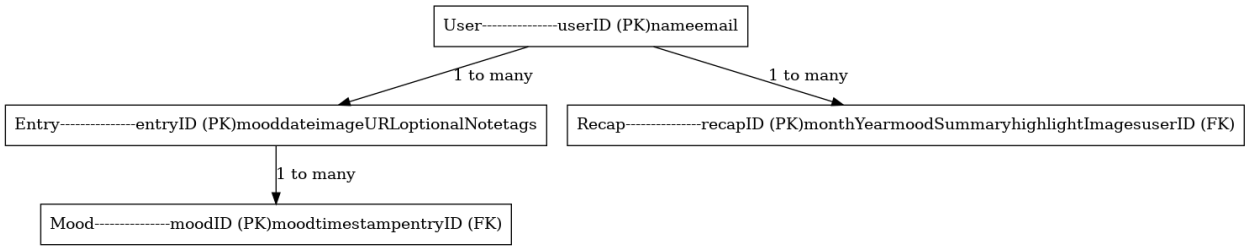


5. Activity Diagram



6.4 database design and description section

6.4.1 database design ER diagram section



6.4.2 Database access section

The Momento app relies on Firebase Firestore to store and manage user data efficiently and securely. Firestore's real-time data syncing allows users to save their journal entries, moods, and photos instantly, so any new moment they capture is available across their devices immediately. Each user has their own unique data space within the database, keeping entries private and easily accessible only to the individual user.

The app uses Firestore's flexible querying features to retrieve data quickly and intuitively. For example, users could filter their journal archive by date or tags to find specific entries without scrolling endlessly. Additionally, to create the monthly mood recaps, the app aggregates mood data over each month, giving users a clear picture of their mood trends over time. These query capabilities make interacting with past moments and visualizing personal patterns simple and seamless, enriching the overall journaling experience.

6.4.3 Database security

Protecting user data is central to the Momento app, ensuring that each person's moments and moods stay private. To achieve this, we use Firebase Authentication to require user login before any data access. This setup means that only signed-in users can create, view, or edit their journal entries, adding a personal layer of security to the journaling experience. By keeping authentication simple but effective, we help users feel confident that their entries are kept private and secure.