Group 03

Memoir
Use-Case Specification

Version <1.6>

Revision History

Date	Version	Description	Author
06/06/2025	1.0	2.1. Authentication 2.6. Location Management	Nguyễn Gia Huy
07/06/2025	1.1	2.7. Note upload and download 2.10. Admin feature	Bùi Hải Long
08/06/2025	1.2	2.3. Reminders and Notifications 2.4. Calendar 2.5. Relationship Visualization	Nguyễn Thị Trà My
09/06/2025	1.3	2.8 Note publish 2.9 Submit Feedback	Huỳnh Tuấn Phong
10/06/2025	1.4	2.2 Note Management	Phạm Nam Huyên
14/06/2025	1.5	Fix consistency issues	Bùi Hải Long
15/06/2025	1.6	Note recovery	Phạm Nam Huyên
04/07/2025	1.7	Removed use-case: Perform Server Maintenance Adjust basic flow of add event to calendar (2.4.1)	Bùi Hải Long
12/07/2025	1.8	Add view note list use case (2.2.1)	Phạm Nam Huyên
13/07/2025	1.9	Add delete uploaded note use case (2.7.2) Update use case model	Bùi Hải Long

Table of Contents

1. Use-case Model **5**

2. Use-case Specifications **6**

2.1 Use-case: User Authentication 6

2.1.1 User Registration 6

2.1.2 Log In 6

2.1.3 Log Out 7

2.1.4 Recover Forgotten Password 7

2.1.5 Change Password 8

2.2 Use-case: Note Management 9

2.2.1 View Note List/ Contact List 9

2.2.2 Create New Note/Contact 10

2.2.3 View or Edit Contact Note Content 10

2.2.4 Delete Contact Note 11

2.2.5 Tag a Contact Note 11

2.2.6 Attach Photos to a Contact Note 12

2.2.7 Filter and Sort Notes by Tag, Location or Date 12

2.3 Use-case: Reminders and Notifications 13

2.3.1 Receive Event Reminders 13

2.3.2 Receive System Notifications 14

2.4 Use-case: Calendar 14

2.4.1 Add Event to calendar 14

2.4.2 View Calendar of Events 15

2.5 Use-case: Relationship Visualization 16

2.5.1 Link Notes to Represent Relationships 16

2.5.2 View Relationship Graph 16

2.6 Use-case: Location Manager 17

2.6.1 Save Meeting Location for a Contact 17

2.6.2 View Contact Location on Map 17

2.7 Use-case: Note Upload and Download 18

2.7.1 Uploading Note 18

2.7.2 Delete uploaded Note 19

2.7.3 Downloading Note 19

2.8 Use-case: Note Publishing 20

2.8.1 Publish Note to Public Profile 20

2.8.2 View Public Notes as Visitor 21

2.8.3 Unpublished Note 21

2.9 Use-case: User Feedback 21

2.9.1 Submit Feedback or Report a Bug 21

2.10 Use-case: Admin Features 22

2.10.1 Manage User Accounts 22

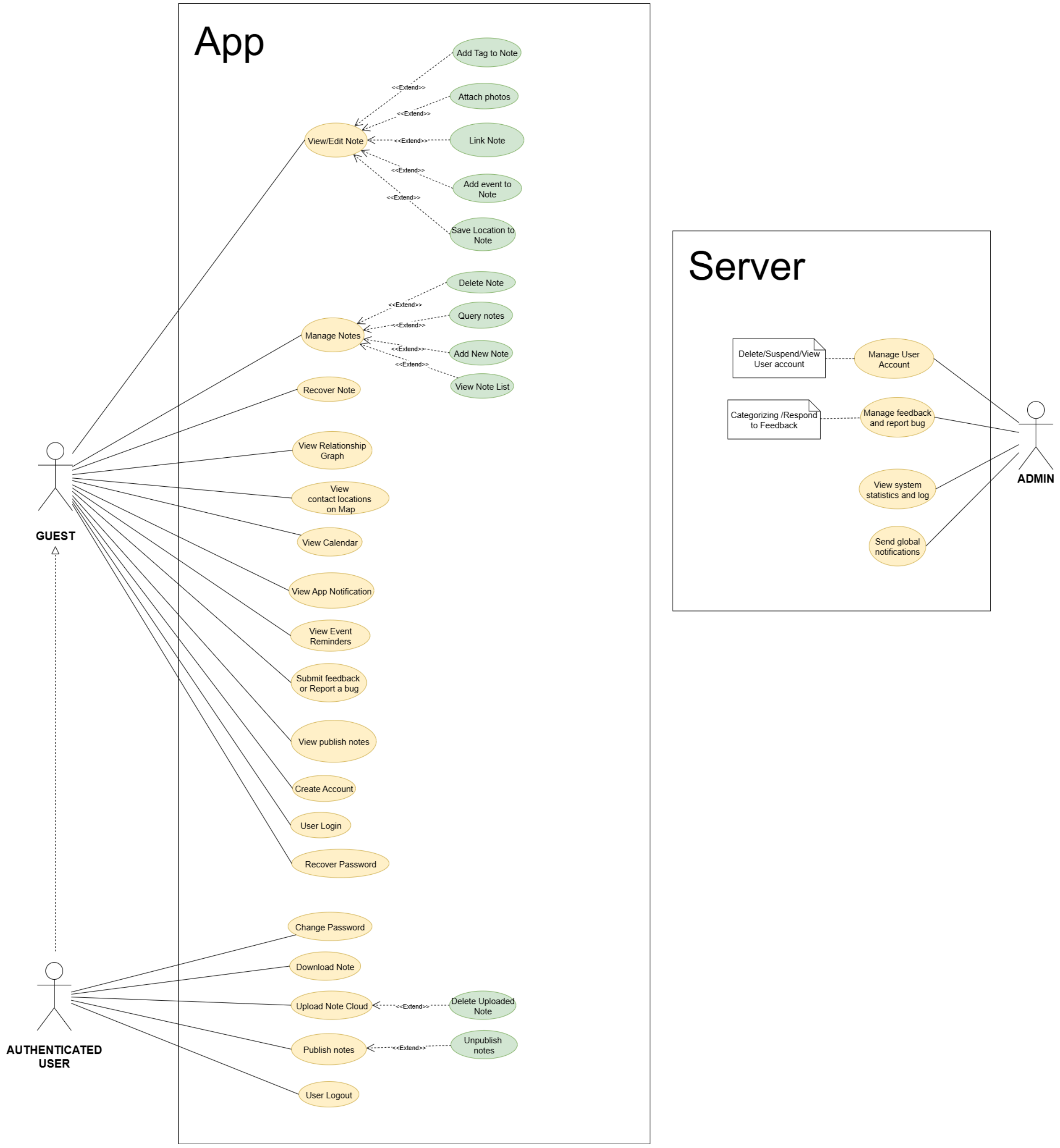
2.10.2 Manage User Feedback and Bug Reports 23

2.10.3 View System Statistics and Logs 23

2.10.4 Send Global Notifications 24

2.11 Use-case: Recover notes/files 25

1. Use-case Model



2. Use-case Specifications

2.1 Use-case: User Authentication

2.1.1 User Registration

Use case Name	Register account
Brief description	This use case describes how the user can register a new account
Actors	Guest
Basic Flow	1. User navigates to the registration page 2. System displays the registration form 3. Users enter required information into the form (mail address, user name, password, etc…) 4. User submits the registration form 5. System validates the submitted data 6. If data is valid, system creates a new user account with the provided details and displays a success message to the user
Alternative Flows	**Alternative flow 1: Invalid or Incomplete Data** 1. From #5 in basic flow, if data is invalid or incomplete Data (missing required field, password mismatch, weak password, user name or email address already exits,... 2. System displays specific error messages next to the problematic fields or in a summary 3. System re-displays the registration form with the valid data retained 4. User corrects the errors and re-submits the form 5. Continue step #4 in the basic flow **Alternative flow 2: User Cancels Registration** 1. At any step before step #4, user clicking a ‘Cancel’ button 2. System discards any data entered by the user 3. Redirect users to offline mode
Pre-conditions	1. The server is operational 2. User has access to the registration page
Post-conditions	1. A new user account is created and stored in the system 2. User is informed of the successful registration

2.1.2 Log In

Use case Name	Log In
Brief description	This use case describes how a registered User can log in to the system
Actors	Guest
Basic Flow	1. User navigates to the login page 2. System displays the login form, typically requiring: *Username or Email Address, Password* 3. User enter information that system require and click ‘Sign in’ 4. System validates the submitted credentials 5. If credentials are valid: a. System establishes an authenticated session for the user. b. System redirects the user to their dashboard

Alternative Flows	**Alternative flow 1: Invalid Credentials** 1. From #4 of the basic flow, the username/email does not exist or the password does not match 2. System displays an error message 3. System re-displays the login form, clearing the informations are entered before 4. Continue step #3 in the basic flow **Alternative flow 3: User Clicks "Forgot Password?"** 1. At any step, user clicks a "Forgot Password?" 2. System redirects the user to the Password Recovery (2.1.4)
Pre-conditions	1. System is operational 2. User is on the login page 3. User has a registered and active account
Post-conditions	1. **On successful:** User is redirected to their account area 2. **On failure:** User remains on the login page or is informed of account status

2.1.3 Log Out

Use case Name	Log Out
Brief description	This use case describes how a logged-in User can log out of the system
Actors	Authenticated User
Basic Flow	1. User click on ‘Log out’ button 2. System redirects the user to offline note mode 3. System displays a message confirming successful logout
Alternative Flows	**No have alternative flow**
Pre-conditions	1. System is operational 2. User is currently logged into the system
Post-conditions	User successfully logged out and switched to offline note mode

2.1.4 Recover Forgotten Password

Use case Name	Recover Forgotten Password
Brief description	This use case describes how the User can recover access to their account if they have forgotten their password
Actors	Guest, Authenticated User
Basic Flow	1. User navigates to the password recovery page 2. System displays a form requesting the User's registered email address 3. User enters their email address and submits the form 4. System checks if the email address exists in the user database 5. If the email exists, system sent a verify code to user’s mail 6. User enter the code and submit 7. If the code is valid, system displays a form for the User to enter a new password and confirm the new password 8. User enters their new password and confirms it 9. User submits the new password form 10. System validates the new password 11. System updates the user's account with the new password 12. System displays a success message

Alternative Flows	**Alternative flow 1: Email Address Not Found** 1. From #4 in basic flow, the entered email address does not exist in the system 2. System display a error message and a clear form 3. Continue #3 in the basic flow **Alternative flow 2: Confirm New Passwords Do Not Match** 1. Form #8 in basic flow, the new password and confirms it not match 2. System displays specific error messages 3. System re-displays the new password and confirms field 4. Use-case backtrack #8 **Alternative flow 3: User Does Not Receive Verify Code** 1. From #5 in basic flow, user doesn't receive verify code 2. User clicking a 'Resent the code' 3. Continue #4ai
Pre-conditions	1. The server is operational 2. User has an existing, active account 3. User is on the password recovery page
Post-conditions	1. **On successful:** User's password has been changed in the system and informed of the successful password change, can now log in with the new password 2. **On failure:** User's password is not changed and informed of the reason for failure

2.1.5 Change Password

Use case Name	Change Password
Brief description	This use case describes how the a logged-in User can change their current password
Actors	Authenticated User
Basic Flow	1. User navigates to the "Change Password" page 2. System displays a form requiring: Current Password, New Password, Confirm New Password 3. User fills in all three password fields and click submit button 4. System validates the submitted data a. Verifies that the "Current Password" entered matches the user's actual current password b. Checks that the "New Password" meets complexity requirements c. Checks that "New Password" and "Confirm New Password" fields match d. Checks that the "New Password" is different from the "Current Password" 5. If all validations pass, system updates the user's account with the new password and displays a success message 6. System may send a notification email to the user about the password change

Alternative Flows	**Alternative flow 1: Incorrect Current Password** 1. From #4 password does not follow the rules 2. System displays an error message 3. System re-displays the change password form, clearing password fields 4. Returns to #3 of the basic flow **Alternative flow 2: User Cancels** 1. At any point before submitting the form, the user decides to cancel 2. User navigates away from the form **Alternative flow 3: User Clicks "Forgot Password?"** 1. At any point before submitting the form, user clicks ‘Forgot Password?’ link 2. The system redirects the user to the password recovery page
Pre-conditions	1. The server is operational 2. User is on the "Change Password" page 3. User is currently logged into the system
Post-conditions	1. **On successful:** User's password has been changed in the system and informed of the successful password change 2. **On failure**: User's password is not changed and informed of the reason for failure

2.2 Use-case: Note Management

2.2.1 View Note List/ Contact List

Use case Name	View Note List
Brief description	Users can view a list of notes or contacts that have been stored in the system.
Actors	User
Basic Flow	1. **The user accesses the list screen** The user opens the application and navigates to the "contacts" tab in the bottom navigation bar **2. The system displays the contact list/ note list** **a. Contact List:** Each Contact includes: avatar icon, contact name, contacts tags Total number of contacts **b. Notes List:** Each Note includes: note title, contact that owned the note, date of note creation and modification, tags attached to the note, partial preview of note content Total number of notes in current contact **3. User can scroll to see contact/notes list**
Alternative Flows	**Alternative flow 1: No Contacts or no notes** From #1: If the contact list is empty, the system displays a message “no contacts” or “no notes”, suggest the user to use the "+" button to add a new contact/note
Pre-conditions	The user has opened the application
Post-conditions	The user has seen the contact list on the screen The status of the contacts in the system has not changed

2.2.2 Create New Note/Contact

Use case Name	Create new note
Brief description	User creates a new note related to a saved contact, or anything else
Actors	Guest, Authenticated User
Basic Flow	1. Click the "Add New Note" button (in Note List), or "Add New Contact" button (in Contact List) 2. Enter the note’s title 3. Enter the note's content 4. (Optional) Enter the tag, date, location or attach a photo 5. Clicks "Save" 6. The note is saved and displayed in the list
Alternative Flows	**Alternative flow 1: Empty note title** From #3: There is no note title → the app set the default note’s title as “Untitled Note” **Alternative flow 2: Exit without saving** From #5: User exits without saving → warns user that content being created could be lost **Alternative flow 3: Duplicate note name** From #3: New note’s name is similar with an existed note → give user two choices: 1. Rename the note 2. Overwrite note name *(RandomName → RandomName(1))*
Pre-conditions	**For normal notes:** None **For contact notes:** At least one contact exist
Post-conditions	**On success:** The note is saved to the database and associated with the contact **On failure:** No note is saved, an error message is displayed

2.2.3 View or Edit Contact Note Content

Use case Name	View or edit Contact note content
Brief description	View or edit the note content
Actors	Guest, Authenticated User
Basic Flow	1. User selects a note from the list 2. The interface displays the note details 3. Click the "Edit" button 4. Update the content or other fields 5. Click "Save" 6. The note is updated
Alternative Flows	**Alternative flow 1: Exit without saving** From #2: The user leaves without saving → the system asks “Do you want to save before exiting?”
Pre-conditions	The note already exists

Post-conditions	**On success:** The note is updated **On failure:** The data does not change, displaying errors if any

2.2.4 Delete Contact Note

Use case Name	Delete Contact Note
Brief description	Delete a note from contact's list note
Actors	Guest, Authenticated User
Basic Flow	1. User accesses the note 2. Clicks the "Delete" button 3. The system displays a confirmation message "Are you sure to delete <contact_name>: <note_name> ?" 4. User clicks "Agree" 5. The note is deleted from the system
Alternative Flows	**Alternative flow 1: Deletion canceled** From #3: User selects "Cancel" → operation is canceled, return to previous interface **Alternative flow 2: Note has been deleted before or does not exist** From #5: Note has been deleted before or does not exist → system reports error
Pre-conditions	Note exists, user has permission to delete
Post-conditions	**On success:** Note is deleted **On failure:** Note still exists, reports error

2.2.5 Tag a Contact Note

Use case Name	Tag a contact note
Brief description	Tag the note for easy classification
Actors	Guest, Authenticated User
Basic Flow	1. User accesses the note 2. Click "Add tag" 3. Select or enter tag (eg: #family, #friends…) 4. Click "Save" 5. Note is tagged
Alternative Flows	**Alternative flow 1: Invalid characters** From #3: Tag contains invalid characters → display error **Alternative flow 2: Tag already exists** From #3: Tag already exists → system notifies and does not create new one
Pre-conditions	Note has been created, user has editing rights
Post-conditions	**On success:** Note displays new tag **On failure:** No new tag is assigned

2.2.6 Attach Photos to a Contact Note

Use case Name	Attach Photos to a Contact Note
Brief description	User adds an illustration to the note
Actors	Guest, Authenticated User
Basic Flow	1. User selects a note 2. Click "Add Image" 3. Select an image file from the device (if it's the first time the user use this function, ask the device for permission) 4. Click "Save" 5. The image is attached to the note
Alternative Flows	**Alternative flow 1: Wrong format image** From #3: The image file is not in the correct format (.exe, .pdf) → an error is displayed **Alternative flow 2: Too large image** From #3: Image file > size limit → warning "Image too large"
Pre-conditions	The note exists
Post-conditions	**On success:** The image is displayed in the note interface **On failure:** The note does not change, the image is not attached

2.2.7 Filter and Sort Notes by Tag, Location or Date

Use case Name	Filter and Sort Notes by Tag, Location, Date
Brief description	Search and sort notes by tag, location or date: 1. Tag: enter a keyword (ex: #family, #business), the system displays suggested tags (autocomplete), then filters out notes with matching tags. 2. Location: enter a location (ex: "Ho Chi Minh"), the system suggests previously saved locations, then filters notes by matching locations. 3. Date: allows you to select a. a specific date b. date range (ex: from 01/01 to 30/03) c. sort by time (Newest → Oldest or Oldest → Newest, A-Z, Z-A)
Actors	Guest, Authenticated User
Basic Flow	1. User opens the list of notes 2. Enter filter conditions in Search Bar: a. Enter #tag_name b. Enter the location c. Select a date or date range. (Optional) Select the sort method: newest, oldest, A-Z 3. Click "Apply" button 4. The list of notes is updated according to the filter.
Alternative Flows	**Alternative flow 1:** From #2a, no notes match → display "No notes found". **Alternative flow 2:** From#2, User enters nothing and press "Apply" → system displays all notes. **Alternative flow 3:**

	From#2, User selects multiple filter criteria → the system processes the combination
Pre-conditions	At least one note is saved.
Post-conditions	The list is displayed correctly according to the filter request.

2.3 Use-case: Reminders and Notifications

2.3.1 Receive Event Reminders

Use case Name	Receive Event Reminders
Brief description	This use case describes how the user receives timely notifications for upcoming events recorded in the application, such as birthdays, appointments,...
Actors	Guest, Authenticated User
Basic Flow	1. The system identifies events that have been previously created and saved by the user in the application. 2. The system continuously monitors these upcoming events and their configured reminder times. 3. When the reminder time for an event is reached, the system generates and sends a notification to the user's device. 4. The user receives the notification, which alerts them about the upcoming event. 5. The user taps on the notification to open the application and view the full details of the event.
Alternative Flows	**Alternative Flow 1: User snoozes the reminder** 1. The system sends a notification at the scheduled reminder time. 2. The user receives the notification and selects the "Snooze" option. 3. The system postpones the notification and resends it after a predefined snooze period (e.g., 10 minutes). 4. The user receives the rescheduled notification. **Alternative Flow 3: Notification fails to deliver due to system permissions** 1. The system attempts to send a notification. 2. The device blocks the notification because the application lacks permission to send notifications. 3. The system logs the failure and may alert the user within the app to enable notifications. **Alternative Flow 4: Event is deleted before the reminder time** 1. The event associated with the reminder is deleted by the user before the scheduled reminder. 2. The system cancels the scheduled notification for the deleted event. 3. No notification is sent at the reminder time.
Pre-conditions	1. The user has previously created and saved at least one event with a reminder set in the application. 2. The application has permission to send notifications on the user's device. 3. The device is powered on and connected to the system

Post-conditions	1. The user successfully receives timely notifications for upcoming events. 2. The system updates the status of the reminder to indicate it has been sent. 3. If the user interacts with the notification, the application opens the corresponding event details. 4. If the event is dismissed or snoozed, the system records the user's choice accordingly.

2.3.2 Receive System Notifications

Use case Name	Receive System Notifications
Brief description	This use case describes how the user receives and views notifications sent by the system administrator. These notifications may include announcements such as maintenance schedules, app updates, policy changes, or other important system-wide information.
Actors	Guest, Authenticated User
Basic Flow	1. The user is currently using the app. 2. The user navigates to the system notification section. 3. The app displays a list of system notifications previously sent by the administrator. 4. The user selects a notification to view its full content. 5. The system displays the complete message within the app interface.
Alternative Flows	**Alternative flow 1: User receives notification outside the app and opens it** 2. A system notification appears on the user's device as a push notification. 3. The user taps on the notification. 4. The app opens and navigates directly to the full content of the notification. 5. The system records that the notification was opened. **Alternative Flow 2 – User receives notification outside the app and dismisses it** 1. A system notification appears on the user's device as a push notification. 2. The user ignores or dismisses the notification. 3. The notification is not opened, but remains available in the app's notification center for later viewing. 4. The system records that the notification was dismissed or not interacted with.
Pre-conditions	1. A system notification has already been sent. 2. The user has the app installed and has enabled notification permissions.
Post-conditions	1. The user is aware of the content of the system notification. 2. The system records whether the notification was opened or ignored.

2.4 Use-case: Calendar

2.4.1 Add Event to calendar

Use case Name	Add event to calendar
Brief description	This use case describes how the system automatically adds a new event to the user's calendar based on a note that has a time setting. The event includes relevant details (time, date, and reminder) as configured in the note.
Actors	Guest, Authenticated User
Basic Flow	1. The user sets a time on a note.

	2. The user set the reminder option for the event 3. The system detects that the note includes a scheduled time. 4. The system creates a corresponding event in the user’s calendar. 5. The event includes the time, date, and reminder based on the note’s settings. 6. The system notifies the user that the event has been successfully added to the calendar.
Alternative Flows	**Alternative flow 1: Note time is removed after event creation** 1. The user removes or clears the time setting in the note. 2. The system detects the change. 3. The system removes the corresponding event from the calendar. **Alternative flow 2: Missing required fields** 1. The user leaves required fields empty (e.g., date or time). 2. The system displays an error message prompting the user to complete all necessary information. 3. The user corrects the input and proceeds.
Pre-conditions	1. The user has access to the calendar feature. 2. A note with a set time has been created. 3. The application is functioning normally.
Post-conditions	1. An event is added to the user’s calendar using the note’s time and reminders. 2. If the note’s time is removed, the corresponding calendar event is deleted.

2.4.2 View Calendar of Events

Use case Name	View Calendar of Events
Brief description	This use case describes how the user accesses and views their scheduled events within the calendar interface of the application.
Actors	Guest, Authenticated User
Basic Flow	1. The user opens the calendar feature in the application. 2. The system loads and displays the list of scheduled events in a fixed format. 3. The user scrolls through the list to view events on different days. 4. The user taps on a specific event to view its details.
Alternative Flows	**Alternative flow 1: No events found** 1. The user opens the calendar. 2. The system finds no events to display. 3. The system shows a message such as “No events scheduled”.
Pre-conditions	1. The user has accessed the calendar screen.

	2. At least one event has been previously added to the calendar.
Post-conditions	1. The user can see and interact with the list of scheduled events. 2. The system has displayed all events in the predefined layout.

2.5 Use-case: Relationship Visualization

2.5.1 Link Notes to Represent Relationships

Use case Name	Link Notes to Represent Relationships
Brief description	This use case describes how the user creates or edits links between notes to represent relationships, enabling the system to build a graph of connected notes based on these links.
Actors	Guest, Authenticated User
Basic Flow	1. The user selects a note to edit or creates a new note. 2. The user inserts a markdown link by selecting another existing note from their local notes to reference within the current note. 3. The system saves the note along with the embedded markdown link. 4. The system updates the relationships data to include this new link between notes.
Alternative Flows	**Alternative flow 1: Invalid markdown link format** 1. The user inserts a markdown link in an incorrect format. 2. The system detects the invalid format and prompts the user to fix the link before saving.
Pre-conditions	1. At least one note exists locally in the system to link to. 2. The user has permission to edit or create notes.
Post-conditions	1. The note is saved with the new or updated markdown link(s). 2. The system's relationship data reflects the updated connections between notes.

2.5.2 View Relationship Graph

Use case Name	View Relationship Graph
Brief description	This use case describes how the user views a visual graph representing the connections between notes, where links embedded within notes (markdown links) are displayed as edges between note nodes in the graph.
Actors	Guest, Authenticated User
Basic Flow	1. The user opens the Relationship Graph feature in the application. 2. The system retrieves notes data along with the internal links between notes. 3. The system generates a graph where each node represents a note, and edges represent links from one note to another. 4. The system displays the graph visually, showing the network of connected notes. 5. The user can interact with the graph. 6. When a node is selected, the system displays detailed information about that note and its connected notes.

Alternative Flows	**Alternative flow 1: No linked notes available** 1. The system detects that there are no notes with internal links. 2. The system shows a message informing the user that no relationship graph can be displayed. **Alternative flow 2: Data loading error** 1. The system fails to load notes or link data due to an error. 2. The system notifies the user of the problem and suggests retrying later.
Pre-conditions	1. There are notes containing internal links to other notes in the system. 2. The application has access to the stored notes and their link metadata.
Post-conditions	1. The user has viewed the visual graph of notes and their interconnections. 2. The system maintains correct visualization and interaction state within the graph view.

2.6 Use-case: Location Manager

2.6.1 Save Meeting Location for a Contact

Use case Name	Save Meeting Location
Brief description	This use case describes how Guest save a meeting location for a contact
Actors	Guest, Authenticated User
Basic Flow	1. The User navigates to their contact list and selects a specific contact 2. User selects the option to add or edit a location for the contact 3. System displays a form with a text field for the address and/or an interactive map 4. User enters a location address 5. User confirms the selection by clicking a "Save" button 6. System saves the location data and associates it with the selected contact 7. System displays a confirmation message, "Location saved successfully."
Alternative Flows	**Alternative flow 1: User Cancels the Action** 1. At any point before #5, the user clicks "Cancel" 2. System returns the user to the previous screen
Pre-conditions	A contact exists in the user's contact list
Post-conditions	**On Success:** The contact's record is updated with the new or modified location data **Failure:** The contact's location data remains unchanged, and the user is notified of the failure

2.6.2 View Contact Location on Map

Use case Name	View Contact Location on Map
Brief description	This use case describes how a guest can view contact location on map
Actors	Guest, Authenticated User
Basic Flow	1. User navigates to the contact details page for a specific contact 2. User clicks on the "View on Map" button 3. System retrieves the geo coordinates for the saved location 4. System displays an interactive map centered on the contact's location, with a pin marking the specific address

Alternative Flows	**No Alternative Flow**
Pre-conditions	1. A location has been previously saved for the selected contact 2. The device has an active internet connection
Post-conditions	**On Success:** The user is shown a map displaying the contact's location **On Failure:** The map is not displayed, and the user is notified of the reason for the failure

2.7 Use-case: Note Upload and Download

2.7.1 Uploading Note

Use case Name	Uploading Note
Brief description	Allows users to upload their locally stored notes and associated contact data to the cloud for backup and synchronization across devices.
Actors	Authenticated User
Basic Flow	1. User selects the "Upload" option from the settings or data menu. 2. System authenticates the user session. 3. System displays a list of available data (notes, contacts, media) for upload. 4. User selects specific items or data to upload. 5. System begins uploading data to the cloud and shows progress. 6. Upon completion, the system confirms successful upload to the user.
Alternative Flows	**Alternative Flow 1: Select All** 1. From #3 users choose to upload all available data with one click. 2. System pre-selects all items and proceeds to Step #4. **Alternative Flow 2: Unauthenticated User** 1. From #2, User haven't sign-in 2. Notify user to sign-in **Alternative Flow 3: Upload Fails (Network Error)**: 1. From #5, Network System notifies users of failure and prompts them to retry. 2. System shows an error message indicating failure and provides the user with options to: a. Retry the upload, repeat step #5 b. Cancel the upload **Alternative Flow 4: Duplicate Conflict**: 1. From step #5, A note or contact being uploaded already exists in the cloud. 2. System prompts the user with conflict resolution options: a. **Overwrite**: Replace the cloud version with the local version b. **Skip**: Do not upload the conflicting item c. **Keep Both**: Save the new item with a different timestamp or version label
Pre-conditions	1. User is logged in and has local notes or contact data available. 2. Device has an active internet connection.
Post-conditions	**On successful:** 1. Selected notes and data are saved to the user's cloud storage. 2. System marks the local data as "synced". **On Failure:** 1. Unuploaded items remain unsynced on the device. 2. User is notified of what cause the failure

2.7.2 Delete uploaded Note

Use case Name	Delete Uploaded Note
Brief description	Allows authenticated users to delete previously uploaded notes from their cloud storage. This helps manage cloud storage space and remove outdated or unnecessary notes.
Actors	Authenticated User
Basic Flow	1. User navigates to the cloud-synced notes section. 2. System authenticates the user session. 3. System displays a list of uploaded notes. 4. User selects one or more notes to delete. 5. System prompts the user for confirmation. 6. User confirms deletion. 7. System deletes the selected notes from the cloud storage. 8. System notifies the user of successful deletion.
Alternative Flows	**Alternative Flow 1: Select All** 1. From Step #3: User chooses to delete all uploaded notes at once. 2. System pre-selects all items and proceeds to Step #5. **Alternative Flow 2: Unauthenticated User** 1. From Step #2: If user is not signed in 2. System prompts user to sign in before proceeding. **Alternative Flow 3: Deletion Cancelled by User** 1. From Step #6: User cancels the confirmation dialog. 2. System aborts deletion and returns to the uploaded notes list. **Alternative Flow 4: Deletion Error (Network or Server Failure)** 1. From Step #7: If deletion fails due to network/server issues: 2. System displays an error message and provides options to: a. Retry the deletion (return to Step #7) b. Cancel the operation
Pre-conditions	1. User is signed in. 2. Notes have already been uploaded and exist in cloud storage. 3. Device has an active internet connection.
Post-conditions	**On Success:** ● Selected notes are permanently removed from cloud storage. ● Local copy (if exists) remains unaffected unless separately deleted. **On Failure:** ● Notes remain in cloud storage. ● User is notified with an appropriate error message.

2.7.3 Downloading Note

Use case Name	Downloading Note
Brief description	Users retrieve previously uploaded notes, contacts, and media from their cloud account to restore or sync with the local device.
Actors	Authenticated User
Basic Flow	1. User opens settings or data menu. 2. System checks if the user is authenticated.

	3. System retrieves metadata of available cloud data. 4. System displays a list of downloadable notes and contacts. 5. User selects items to download. 6. System downloads and integrates data into local storage. 7. System confirms completion.
Alternative Flows	**Alternative Flow 1: Select All** 1. From #4, the user selects “Download All”. 2. System pre-selects all available items and proceeds to Step #5. **Alternative Flow 2: Unauthenticated User** 1. From #2, the user hasn’t signed in. 2. Notify users to sign in before continuing. **Alternative Flow 3: Download Fails (Network Error)** 1. From #6, the network system notifies users of failure and prompts them to retry. 2. System shows an error message indicating failure and provides the user with options: a. Retry the download (repeat Step #6) b. Cancel the download **Alternative Flow 4: Duplicate Conflict** 1. From Step #6, a local item already exists with the same ID or timestamp. 2. System prompts the user with conflict resolution options: a. Overwrite: Replace local version with the cloud version b. Skip: Do not download the conflicting item c. Keep Both: Save the downloaded item with a version label
Pre-conditions	1. User is logged in and has existing data in their cloud account. 2. Device has an active internet connection.
Post-conditions	**On Success:** 1. Selected notes and data are saved to the user’s cloud storage. 2. System marks the local data as “synced”. **On Failure:** 1. Unuploaded items remain unsynced on the device. 2. User is notified of what caused the failure.

2.8 Use-case: Note Publishing

2.8.1 Publish Note to Public Profile

Use case Name	Publish Note to Public Profile
Brief description	User publishes a chosen note to the world wide web for people to view.
Actors	Authenticated User
Basic Flow	1. User moves a group of notes to the publish section. 2. User confirms the action. 3. Server hosts the published note on the web.
Alternative Flows	**Alternative flow 1: User cancels the action.** 1. Everything remains unchanged.
Pre-conditions	User is authenticated

Post-conditions	**On Success:** 1. The server executes the intended publishing action. 2. User is notified and receives a link or QR code to the page. **On Failure:** 1. User sees an error message and a solution to try again.

2.8.2 View Public Notes as Visitor

Use case Name	View Public Notes as Visitor
Brief description	Any world wide web user can view note if they have link and is authorized
Actors	Guest, Authenticated User
Basic Flow	1. Visitor goes to the public note by link or QR scanning.
Alternative Flows	
Pre-conditions	Visitor have access to internet
Post-conditions	Visitor can read public notes

2.8.3 Unpublished Note

Use case Name	Unpublished Note
Brief description	User unpublishes an already published note.
Actors	Authenticated User
Basic Flow	1. User moves a group of notes out of the publish section. 2. User confirms the action. 3. Server hosts the removes the published note from the web.
Alternative Flows	**Alternative flow 1: User cancels the action.** 1. Everything remains unchanged.
Pre-conditions	User is authenticated
Post-conditions	**On Success:** 1. The server executes the intended unpublishing action. 2. User is notified. **On Failure:** 1. User sees an error message and a solution to try again.

2.9 Use-case: User Feedback

2.9.1 Submit Feedback or Report a Bug

Use case Name	Submit Feedback or Report a Bug
Brief description	User sends feedback or bug report to admin.
Actors	Authenticated User
Basic Flow	1. User goes to the Feedback page. 2. User writes feedback or bug report. 3. User confirms the action.
Alternative Flows	**Alternative flow 1: User cancels the action.** 1. Everything remains unchanged.
Pre-conditions	User is authenticated
Post-conditions	**On Success:** 1. The server executes the intended action. 2. User is notified.

	On Failure: 1. User sees an error message and a solution to try again.

2.10 Use-case: Admin Features

2.10.1 Manage User Accounts

Use case Name	Manage User Accounts
Brief description	Admin can view, edit, disable, or delete user accounts for moderation, security, or maintenance purposes.
Actors	Admin
Basic Flow	1. Admin logs in with administrator privileges. 2. Admin accesses the "User Management" panel. 3. System retrieves and displays the list of registered users. 4. Admin selects a specific user. 5. Admin performs one of the following actions: a. View account details b. Disable or suspend account c. Delete account permanently 6. System applies changes and confirms success.
Alternative Flows	**Alternative Flow 1: Search and Filter Users** 1. From #3, admin filters users by name, email, status, or role. 2. System displays filtered list for easier selection. **Alternative Flow 2: Unauthorized Access** 1. From #1, the user is not an admin. 2. System blocks access and redirects to the login screen or shows a permission error. **Alternative Flow 3: Operation Fails (e.g., Database Error)** 1. From #6, the system encounters an error while saving changes. 2. System notifies admin and provides options: a. Retry b. Cancel and return to user list **Alternative Flow 4: Delete Confirmation** 1. From #5, when deleting an account, the system prompts a confirmation dialog. 2. Admin must confirm deletion before proceeding.
Pre-conditions	1. Admin is authenticated. 2. Database is accessible.
Post-conditions	**On Success:** 1. Selected changes to user accounts are applied and saved. 2. System updates logs for audit purposes. **On Failure:** 1. No changes are saved. 2. Admin is notified of failure reason (e.g., insufficient permissions, server error).

2.10.2 Manage User Feedback and Bug Reports

Use case Name	Manage User Feedback and Bug Reports
Brief description	Admin reviews, categorizes, and responds to feedback or bug reports submitted by users to improve the application or fix issues.
Actors	Admin
Basic Flow	1. Admin logs into the admin dashboard. 2. Admin navigates to the "Feedback & Bug Reports" section. 3. System displays a list of submitted feedback and bug reports. 4. Admin selects an item to review. 5. Admin takes one or more of the following actions: a. Mark as “Resolved”, “In Progress”, or “Ignored” b. Respond to user (if needed) 6. System saves the action and updates the status.
Alternative Flows	**Alternative Flow 1: Filter and Sort Submissions** 1. From #3, Admin filters by type (feedback or bug), status, or date. 2. System displays only relevant submissions. **Alternative Flow 2: No Submissions Found** 1. From #3, the system finds no matching submissions. 2. Displays a message: “No feedback or reports found.” **Alternative Flow 3: Feedback Linked to Deleted User** 1. From #4, the report is linked to an account that no longer exists. 2. System shows basic details and flags the item as "Anonymous". **Alternative Flow 4: Save Fails** 1. From #6, the system fails to save status due to a server issue. 2. System notifies the admin and offers retry or cancel.
Pre-conditions	1. Admin is authenticated. 2. Feedback and report system is operational.
Post-conditions	**On Success:** 1. Feedback/report is updated with a new status and optionally a response. 2. Changes are logged for audit. **On Failure:** 1. No updates are made. 2. Admin is informed of the issue (e.g., system error, connectivity problem).

2.10.3 View System Statistics and Logs

Use case Name	View System Statistics and Logs
Brief description	Admin views application statistics (e.g., user activity, system usage) and system logs (e.g., login records, error logs) for monitoring and diagnostics.
Actors	Admin
Basic Flow	1. Admin logs into the admin dashboard. 2. Admin navigates to the “System Monitoring” section. 3. System displays a dashboard of analytics and log categories. 4. Admin selects the desired category (e.g., error logs, login history, user activity, database capacity). 5. System shows detailed data entries or charts.

	6. Admin may apply filters (e.g., by date, severity, user ID). 7. Admin optionally exports or saves logs/statistics.
Alternative Flows	**Alternative Flow 1: No Data Available** 1. From #3, the system finds no data in the selected category. 2. Shows message: “No data available for the selected time range.” **Alternative Flow 2: Export Logs/Stats** 1. From #7, Admin selects “Export”. 2. System prompts to choose file format (e.g., CSV, PDF) and downloads the report. **Alternative Flow 3: System Error** 1. From #5 or #6, system fails to fetch logs/statistics due to backend issue. 2. Displays an error and offers retry or alternative options.
Pre-conditions	1. Admin is authenticated. 2. System has stored log/statistics data.
Post-conditions	**On Success:** 1. Admin views and optionally exports up-to-date system stats and logs. 2. Accesses insight for diagnostics or audits. **On Failure:** 1. Logs or stats are not displayed. 2. System informs the admin of the error and recommends next steps.

2.10.4 Send Global Notifications

Use case Name	Send Global Notifications
Brief description	Admin composes and sends announcements or alert messages to all users (e.g., for maintenance, updates, or system-wide messages).
Actors	Admin
Basic Flow	1. Admin logs into the admin dashboard. 2. Admin navigates to the “Notifications” section. 3. Admin clicks “Create New Notification”. 4. Admin enters the title, message content, and optional expiry time. 5. Admin previews and confirms the message. 6. System sends the notification to all users (in-app and/or via email/push). 7. Users receive and view the notification on their devices.
Alternative Flows	**Alternative Flow 1: Schedule Notification** 1. From #4, Admin sets a future date/time for the message to be sent. 2. System stores the message and sends it at the scheduled time. **Alternative Flow 2: Cancel Notification Before Sending** 1. From #5, Admin chooses to cancel or go back instead of sending. 2. System discards the draft. **Alternative Flow 3: Delivery Failure** 1. From #6, some users fail to receive the notification due to connectivity issues. 2. System logs failed deliveries and retries based on system policy.
Pre-conditions	Admin is authenticated
Post-conditions	**On Success:**

	1. Notification is delivered to all active users. 2. System logs the delivery event. **On Failure:** 1. Message is not sent or only partially delivered. 2. Admin is notified, and failed deliveries are logged.

2.11 Use-case: Recover notes/files

Use case Name	Recover Deleted File (Restore or Permanently Delete)
Brief description	Allows the authenticated user to manage deleted notes or files in the **Recycle Bin**, including: - Restoring them to their original state - Permanently deleting them
Actors	Authenticated users, guest
Basic Flow	1. The user navigates to the **Recycle Bin** screen 2. The user selects one or more deleted notes 3. The user clicks the **1 of 2** following buttons: ○ **Restore** ○ **Permanently Delete** 4. The system shows a confirmation dialog 5. If the user confirms: ○ If Restore: ■ Restore the note to its original folder or a default fallback location ■ If it's a contact-note: re-link to the original contact (if exists) ○ If Permanently Delete: ■ Remove note from database and storage ■ Delete all associated files (images, location…) ○ System shows a success message

Alternative Flows	**Alternative Flow 1: User Restore/Permanently Delete All Notes** ● From #2, The user selects **“Select All”** to delete all notes in the Recycle Bin, then Recycle Bin is empty **Alternative Flow 2: User Cancel Restore/Permanently Delete** ● From #4, the user clicks **“Cancel”** ⇒ returns to Recycle Bin without making changes **Alternative Flow 3: Restore ⇒ Note name conflicts** ● From #5, a note has the same name as an existing active note ⇒ system prompts: ○ Overwrite the existing note ○ Create a duplicate (ex: “Note (1)”) ○ Skip restoring that specific note **Alternative Flow 4: Missing contact link** ● From #5, the original contact of the notes no longer exists ○ System warning ○ Create new Empty Contact note name “SomebodyYouUsedToKnow_id” ○ Restores the note into that new Contact note **Alternative Flow 5: Delete ⇒ Notes not found** ● From #5, File not found when deleting (race condition or backend error), system warns: “Some items no longer exist.” and continues processing remaining items
Pre-conditions	1. User is logged in 2. Notes to be restored are in Recycle Bin (in soft-deleted state)
Post-conditions	**On Success:** ● If restored: Notes return to usable state, visible in the app. ● If deleted: Notes are permanently removed from both database and storage. **On Failure:** ● No changes are made if canceled or if a restore error occurs ● Partial failures are logged; system continues remaining actions where possible