Communications

Software Requirements Specification

Github: https://github.com/Ayoub-Mekkaoui/Communications-Project

Revision History

Date	Revision	Description	Author
9/12/2024	0.0.0	Created SRS “Skeleton”	GROUP
9/14/2024	0.0.1	Created GitHub repo for project	GROUP
9/15/2024	1.1.0	Wrote Purpose (1.1, 1.2, 1.3.)	GROUP
9/15/2024	1.2.0	Wrote General Description (2.1, 2.2, 2.3, 2.4, 2.5)	GROUP
9/15/2024	1.3.1	Wrote 5 Log-In Requirements (3.1.4.1 to 3.1.4.5)	Ayoub Mekkaoui
9/15/2024	1.3.2	Wrote 10 GUI Requirements (3.1.8.1 to 3.1.8.10)	Ayoub Mekkaoui
9/15/2024	1.3.3	Wrote 4 External Interface Requirements (3.2.1 to 3.2.4)	Ayoub Mekkaoui
9/15/2024	1.3.4	Wrote 3 Internal Interface Requirements (3.3.1 to 3.3.3)	Ayoub Mekkaoui
9/15/2024	1.4.1	Wrote 7 Non-Functional Requirements in 4.1, 4.2, 4.3	Ayoub Mekkaoui
9/15/2024	1.5.1	Created Use Case #1 UML	Ayoub Mekkaoui
9/15/2024	1.5.2	Drew Use Case Diagram #1 (Ms Paint)	Ayoub Mekkaoui
9/15/2024	1.5.3	Added reqs to 3.1.1-3.1.3, 3.1.5, 3.1.6	Ben Levy
9/26/2024	1.5.4	Added reqs to 3.1.1.3, 3.1.1.4	Van Nguyen
9/28/2024	1.5.5	Wrote 5 reqs to Message Handling Module (3.1.7.1-3.1.7.4)	Ayoub Mekkaoui
9/28/2024	1.5.6	Wrote 3 reqs to Storage Module (3.1.8.1-3.1.8.3)	Ayoub Mekkaoui
9/28/2024	1.6.0	Added 3 Use Cases (ID: 002, 003, 004)	Ayoub Mekkaoui
9/28/2024	1.6.1	Added 3 reqs to Security and Privacy and edited 3.1.5.1	Ben Levy
9/28/2024	1.6.2	Updated document outline(no visible changes)	Ben Levy
9/28/2024	1.6.3	Added use case “Create User”	Ben Levy
9/28/2024	1.6.4	Reformatted document	Ben Levy
9/19/2024	1.6.5	Added reqs to User Reqs 3.1.5.3-3.1.5.5, 3.1.6.3- 3.1.6.5	Van Nguyen
9/19/2024	1.6.6	Fix requirement for Common Req at 3.1.1.4	Van Nguyen
9/19/2024	1.7.1	Added 3 use cases(ID: 006, 007,008)	Van Nguyen
9/29/2024	1.7.2	Wrote requirements 3.1.5.5 & 3.1.6.8, Added Use Cases (ID:012, 13)	Ibraheem Fawal
9/29/2024	1.7.3	Added Class Diagram	Ayoub Mekkaoui
9/29/2024	1.7.4	Added use cases “Delete User” and “View Logs”	Ben Levy
9/29/2024	1.7.5	Added sequence diagram for “Create User”	Ben Levy
9/29/2024	1.7.6	Fixed Product Architecture	Akbar Hashimi
9/29/2024	1.7.7	Added Requirement 3.1.8.4	Akbar Hashimi
9/29/2024	1.7.8	Added Requirement 3.1.5.6	Akbar Hashimi
9/30/2024	1.7.9	Added Requirement 4.1.2	Akbar Hashimi
9/30/2024	1.8.0	Edited Use Case 10	Akbar Hashimi
9/30/2024	1.8.1	Added Requirement 3.1.6.7	Akbar Hashimi
9/30/2024	1.8.2	Added Use Case 11	Akbar Hashimi
9/30/2024	1.9.0	Added Sequence Diagram for Use Case ID: 001 and ID: 002	Ayoub Mekkaoui
9/30/2024	1.9.1	Added Sequence Diagram for “Delete User” and “View Logs”	Ben Levy

9/30/2024	1.9.2	Added Use Cases (ID: 014 - 017) and left comments for review	Ibraheem Fawal
09/30/2024	1.9.3	Added Use Case Diagram(ID: 006, 007)	Van Nguyen
09/30/24	1.9.4	Updated Use case 006 and revised Use Case 007	Van Nguyen
10/1/24	1.9.5	Added Use Case 018, added Sequence Diagram (012 - 013)	Ibraheem Fawal
10/1/24	1.9.6	Added Requirement 3.1.5.7	Akbar Hashimi
10/2/24	2.0.0	Updated Class Diagram	Ayoub Mekkaoui
10/2/24	2.0.1	Added Sequence Diagram for Use Case ID: 003 and ID: 004	Ayoub Mekkaoui
10/2/24	2.0.2	Added Use Cases 019, 020, 021	Nico Palma
10/2/24	2.0.3	Added Requirements 3.1.7.5, 3.1.7.6 & 3.1.9.11	Nico Palma
10/2/24	2.0.4	Removed Use Case 13	Akbar Hashimi
10/2/24	2.0.5	Add Admin Use Case Diagram	Van Nguyen
10/2/24	2.0.6	Added User Use Case Diagram	Akbar Hashimi
10/3/24	2.0.7	Added Sequence Diagram for Use Case 11	Akbar Hashimi
10/3/24	2.0.8	Added Sequence Diagram (ID: 014 and 017)	Ibraheem Fawal
10/13/24	2.0.9	Added Requirement 3.1.5.8	Akbar Hashimi
10/14/24	2.1.0	Added Sequence Diagram for Use Case 18 and 19	Nico Palma
10/15/24	2.1.1	Removed old table of contents	Ben Levy

Table of Content

1. Purpose

This document outlines the requirements for the Communications Messaging Application.

1.1 Scope

This document outlines the requirements for a Messaging Application based on Client Server Architecture over TCP/IP.

1.2 Definitions, Acronyms, Abbreviations

User Log In - This is where the user logs in.
Admin Log In - This is where the admin logs in.
Module: A common theme that umbrellas one or many classes
Multiple Connections: Multi-Communication system
Port: An Abstraction for ingress and egress
Interface: A pre-existing defined class which enforces certain behavior
Messages: Data
Verified Connection: An established channel of communication between two mediums
Flagged: Define a certain characteristic or behavior
Recorded: data stored in a text file
Ingress: incoming
Egress: outgoing
System Crash: shutting down of the software without any notification to the user

1.3 References

Use Case Specification Document
UML Use Case Diagrams Document
Class Diagrams
Sequence Diagrams

2. Overall Description

2.1 Product Perspective

The Communications System is an application that allows employees of a large organization to communicate via a chat interface. The system supports both synchronous and asynchronous communication, with features for private and group messaging. IT administrators have access to logs of all conversations for oversight purposes. The system operates with a client-server architecture using TCP/IP, and features a Java-based graphical user interface (GUI). The system is designed to ensure maximal transparency and to work efficiently within the constraints of the organization's network infrastructure.

2.2 Product Architecture

The system will be organized into 8 major modules: Server Module, Client Module, Log-In Module, User Module, Admin Module, GUI Module, Message Handling Module, and Storage Module.

2.3 Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

- Enables multiple users to join and communicate in real-time or via asynchronous messaging.
- Users can send private messages and participate in group chats.
- IT administrators can view logs of all communications.
- The system supports user authentication and ensures that only authorized users can access the application.

2.4 Constraints

- The application requires a stable internet connection for all users.
- Due to the simplistic nature of the project, basic security features for user authentication and data protection are included, but advanced encryption methods are not implemented.
- The system is designed for desktop environments and cannot be used for mobile devices.

2.5 Assumptions and Dependencies

- Assumes users have access to a stable internet connection.
- Assumes the server can handle multiple concurrent connections without significant latency.
- Assumes users will have Java Runtime Environment (JRE) installed on their devices.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Common Requirements:

3.1.1.1 Every module must ensure that actions are performed only by authenticated and authorized users.
3.1.1.2 All input data provided by users must be validated before being processed to ensure correctness and prevent errors.

3.1.2 Server Module Requirements:

3.1.2.1 Will respond to pings from the client.
3.1.2.2 Can determine client IP/port.
3.1.2.3 Keeps track of active clients.…
3.1.2.4 Relays new messages to all clients with relevant users.

3.1.3 Client Module Requirements:

3.1.3.1 Will respond to pings from the server.
3.1.3.2 Maintains separation of different users on the same client.
3.1.3.3 All clients can support normal and administrative users.
…

3.1.4 Log-in Module Requirements:

3.1.4.1 The client will request a username and password so the user may log in.
3.1.4.2 The log-in data for each user will be stored in a separate text file.
3.1.4.3 The log-in module will compare and validate the username and password to the contents of the file.
3.1.4.4 Each account will be flagged as either a regular user or an admin.
3.1.4.5 User sessions shall expire after a specified period of inactivity to ensure security.
3.1.4.6 User passwords must be at least 8 characters and may not be the same as their username. //
3.1.4.7 Usernames must be unique and may contain only letters, numbers, dashes, and apostrophes. //
3.1.4.8 Usernames can be reused after the previous account has been deleted.

3.1.5 User Module Requirements:

3.1.5.1 Users have unique internal identifiers which will never be reused.
3.1.5.2 Exists on both server and client.
3.1.5.3 Users can update their account details, such as password, and profile information.
3.1.5.4 Users can access and view their history of private chats and group chats
3.1.5.5 Users can view profile information of other users. Profile information will include: name, date of account creation, and biographical information.
3.1.5.6 User can create a group chat and private chat
3.1.5.7 Users may edit their profile information.
3.1.5.8 Chats created must have a name.

3.1.6 Admin Module Requirements:

3.1.6.1 Is an extension of the User Module.
3.1.6.2 Admins can add and remove other user accounts.
3.1.6.3 Admins can send notifications or messages to all users.
3.1.6.4 Admins can reset user passwords or unlock accounts when users encounter access issues.
3.1.6.5 Admins can view all logs.
3.1.6.6 Admins can add/remove users to/from any group chat.
3.1.6.7 Admins may flag messages as inappropriate, to be hidden from view of users
3.1.6.8 Admins can lock accounts to prevent entry to the System

3.1.7 Message Handling Module Requirements:

3.1.7.1 Message Routing - The system shall route messages between clients, ensuring that each message is delivered to its intended recipient(s).
3.1.7.2 Message Ordering - The system shall maintain the correct chronological order of messages, ensuring that messages appear in the same sequence for all participants in a conversation.
3.1.7.3 Message Delivery Acknowledgment - The system shall implement a mechanism to acknowledge the receipt of messages by the recipient(s) to ensure no message is lost in transit.
3.1.7.4 Message Archiving - The system shall support the archiving of messages, allowing admins to retrieve old messages on demand.
3.1.7.5 Forwarding Messages - The system shall allow users to forward messages to other recipient(s) while maintaining the original content and context.

3.1.8 Storage Module Requirements:

3.1.8.1 Secure Message Storage - The system shall store all user messages and logs in a manner that ensures data integrity and easy retrieval by authorized users.
3.1.8.2 Data Backup and Restoration - The system shall support regular backups of stored data and provide a mechanism for restoring data in the event of corruption or loss.
3.1.8.3 Storage Management - The system shall monitor storage usage and alert administrators when storage limits are approaching, allowing for proactive management of storage resources.
3.1.8.4 Storage of Profile Data for all Users - The system must have a file dedicated to each user which will be updated as the user change
…

3.1.9 GUI Requirements:

3.1.9.1 The designed interface shall be intuitive, allowing for easy navigation and interaction with the application.
3.1.9.2 The interface shall display error messages for invalid operations or inputs to guide users towards correct actions.
3.1.9.3 Display user status indicators (online, offline, away) clearly within the chat window.
3.1.9.4 User accounts will have different button options than Admin accounts.
3.1.9.5 The chat window shall show timestamps for all messages to indicate when messages were sent or received.
3.1.9.6 The interface will allow users to search for messages using a keyword search bar.
3.1.9.7 Admin users will have additional buttons to manage users, delete messages from active chats, and monitor conversation logs.
3.1.9.8 Users will be able to toggle between private and group chats using clearly defined tabs or buttons.
3.1.9.9 The GUI shall include visual notifications (e.g., flashing tabs or pop-ups) for new messages in chats the user is part of.
3.1.9.10 The system shall include a settings menu accessible via the GUI where users can adjust basic preferences (e.g., notification settings, theme colors, etc.).
3.1.9.11 Mute Notifications - The user is able to mute/unmute notifications. Visual feedback such as a changed notification icon, shall indicate when the notifications are muted.

3.2 External Interface Requirements

3.2.1 Use TCP/IP sockets for establishing and managing connections between clients and the server, detailing protocols for message formatting and handling.
3.2.2 Define the methods for serializing data for transmission over sockets, specifying formats for encoding messaging, and state information.
3.2.3 Implement a message queue or a similar mechanism to handle asynchronous communications and callbacks between modules, ensuring non-blocking operations.

3.2.4 Define a standardized access layer for interacting with the file system or local databases, supporting CRUD operations (Create, Read, Update, Delete) for message state, user profiles, and messaging logs.

3.3 Internal Interface Requirements

3.3.1 Simple Communication Between Modules - Modules communicate using straightforward method calls and return basic data types or simple objects.

3.3.2 Use simple file operations (open, read, write, close) for handling messaging data and user information with plain text files.

3.3.3 Minimal Logging - Implement a basic logging function that records errors and key events to a text file, without detailed configuration.

4. Non-Functional Requirements

4.1 Security and Privacy Requirements:

4.1.1 For certain features such as message logging, user data is recorded.

4.1.2 Profile information is only accessible by Admins and between users who share a chat,it being private or group chat.

4.2 Environmental Requirements:

4.2.1 The System should be compatible for Java development.

4.2.2 A connection between server and client has to be established and communication to take place within the network.

4.2.3 The flow of data between the server and client is handled by inbuilt Java classes such as ServerSocket and Socket Class.

4.3 Performance Requirements

4.3.1 The connection between the client and the server will be through an ingress and egress port designated by the server.

4.3.2 The file sharing mechanism within the system will be handled by inbuilt Java I/O class.

4.3.3 If there are errors that will prevent the system from running or errors that cause the system to crash, those errors need to be caught using the try/catch method and notify the user of the error. Specifics of the error need not be shown to the players.

Communications

Design Document

Use Case Description

Actors

1. ***User:*** A user who interacts with the system to message other users. This includes regular employees who can send and receive private or group messages.
2. ***Admin:*** An admin that manages user accounts, monitors chat logs, and has the ability to lock/unlock user accounts.
3. ***System:*** Represents the backend logic that handles message sessions, user authentication, and data persistence. The system also manages the connection between client and server using the appropriate networking protocols.

Use Case ID: 001
Use Case Name: User Login
Relevant Requirements: 3.1.4 Log-in Module Requirements
Primary Actor: User
Pre-conditions: User has an account registered with the system.
Post-conditions: User is logged into their account and can access the messaging interface.
Basic Flow or Main Scenario: 1) User opens the application and selects the login option. **2)** User enters their username and password. **3)** System verifies the credentials against the stored data. **4)** User is granted access to the messaging interface.
Extensions or Alternate Flows: 3b) If the credentials are incorrect, the user is informed and prompted to try again.
Related Use Cases: Account Management, Message User

Use Case ID: 002
Use Case Name: Send Message
Relevant Requirements: 3.1.7 Message Handling Module Requirements
Primary Actor: User
Pre-conditions: User is logged into the system and is part of a chat (either private or group).
Post-conditions: The message is delivered to the intended recipient(s) and is visible in the chat interface.
Basic Flow or Main Scenario: 1) User types a message into the chat input box. **2)** User clicks the "Send" button or presses "Enter." **3)** The system routes the message to the intended recipient(s) (3.1.7.1). **4)** The system ensures the message is displayed in the correct chronological order (3.1.7.2). **5)**The system acknowledges receipt of the message and updates the chat interface to show the sent message (3.1.7.3). **6)** The message is archived for future retrieval (3.1.7.4).
Extensions or Alternate Flows: 3b) If the user's message fails to send due to a network issue, the system notifies the user and attempts to resend the message. **3c)** If the recipient(s) are offline, the message is queued and delivered when they come online.
Related Use Cases: Message Retrieval, Group Chat

Use Case ID: 003
Use Case Name: Search Messages
Relevant Requirements: 3.1.8 Storage Module Requirements, 3.1.9 GUI Requirements
Primary Actor: User
Pre-conditions: User is logged into the system and has previously sent or received messages in a chat.
Post-Conditions: The system displays the search results, highlighting the relevant messages in the chat interface.
Basic Flow or Main Scenario: 1) User enters a keyword or phrase in the search bar within the chat interface (3.1.9.6). **2)** User clicks the "Search" button. **3)** The system queries the archived messages for matches (3.1.7.4, 3.1.8.1). **4)** The system displays the search results in the chat interface, highlighting the matching terms. **5)** The user selects a specific result to jump to that message in the chat history.
Extensions or Alternate Flows: 3b) If no matches are found, the system displays a "No results found" notification.
3c) If the search query fails, the system notifies the user and prompts them to try again later.
Related Use Cases: View Message History, Send Message

Use Case ID: 004
Use Case Name: View Message History
Relevant Requirements: 3.1.7 Message Handling Module Requirements, 3.1.8 Storage Module Requirements
Primary Actor: User
Pre-conditions: User is logged into the system and is part of a chat (either private or group). The system has archived messages available for the user to view.
Post-Conditions: The user successfully views the message history for the selected chat.

Basic Flow or Main Scenario: 1) User selects the "View Message History" option from the chat interface. **2)** The system retrieves the archived messages from storage (3.1.8.1). **3)** The system displays the retrieved messages in the chat interface in chronological order (3.1.7.2). **4)** The user scrolls through the message history as needed.
Extensions or Alternate Flows: 2b) If the message history retrieval fails, the system notifies the user and prompts them to try again later. **2c)** If no messages are available, the system displays a "No message history available" notification.
Related Use Cases: Send Message, Search Messages

Use Case ID: 005
Use Case Name: Create User
Relevant Requirements: 3.1.6.2 Admin Module; 3.1.5.1, 3.1.4.6-3.1.4.8 User Module
Primary Actor: Admin
Pre-conditions: The server is running and an admin is logged in.
Post-Conditions: A new user is created.
Basic Flow or Main Scenario: 1) Admin selects “Create new user” from the admin controls menu. **2)** Admin enters a valid name and password. **3)** A new user is generated.
Extensions or Alternate Flows: If an invalid username or password is entered, the field will not clear and the admin will be able to edit the entry.
Related Use Cases: User Login

Use Case ID: 006
Use Case Name: Create a group chat
Relevant Requirements: 3.1.5.3 User Module, 3.1.6.2 Admin module
Primary Actor: User
Pre-conditions: No existing group chat has the same name.
Post-Conditions: Group chat is successfully created.
Basic Flow or Main Scenario: 1) User creates a group chat. 2) The user adds other users into group chat 3)Group is successfully created. 4)Group members can start sending and receiving messages in the group chat.
Extensions or Alternate Flows: Some users may choose to leave the group chat after it is created.
Related Use Cases: Send Message, Search Messages

Use Case ID: 007
Use Case Name: Reset Password/Unlock User Account
Relevant Requirements: 3.1.5 User Module,3.1.6.4 Admin Module
Primary Actor: Admin
Pre-conditions: The user account must be locked or the user has requested a password reset.
Post-Conditions: The user’s password is reset, or their account is unlocked.
Basic Flow or Main Scenario: 1) Admin logs into the system and accesses the user management interface. 2) Admin selects the user whose account needs to be unlocked or whose password needs to be reset. 3) Admin performs the reset password or unlocks the account action 4)The system updates the user's account.
Extensions or Alternate Flows: If the user’s account cannot be found: The system informs the admin that the account does not exist, prompting the admin to search again.
Related Use Cases: User Login

Use Case ID: 008
Use Case Name: Log Out
Relevant Requirements: 3.1.4.5 Log-in Module,
Primary Actor: User
Pre-conditions: The user must log in
Post-Conditions: The session is terminated, and the user is no longer able to access chat functions, but still can receive messages.
Basic Flow or Main Scenario: 1)After logging into the system, the user selects the log-out option 2) The system terminates the user’s session.
Extensions or Alternate Flows: After log out from the system, users are still able to receive messages but can’t view or send messages.
Related Use Cases: Login, Message User.

Use Case ID: 009
Use Case Name: Delete User

Relevant Requirements: 3.1.6.2 Admin Module, 3.1.5.1 User Module, 4.1.2-4.1.4 Security and Privacy
Primary Actor: Admin
Pre-conditions: The server is running, an admin is logged in, and at least one user account exists.
Post-Conditions: The user account is "deleted" meaning that the user can no longer access their account. User information is retained for recordkeeping.
Basic Flow or Main Scenario: 1) Admin selects "Delete user" from the admin controls menu. **2)** Admin enters an existing username **3)** The user is deleted.
Extensions or Alternate Flows: If an invalid username is entered, the field will not clear and the admin will be able to edit the entry.
Related Use Cases: User Login

Use Case ID: 010
Use Case Name: View Logs
Relevant Requirements: 3.1.6.5 Admin
Primary Actor: Admin
Pre-conditions: The server is running, an admin is logged in, and at least one chat exists.
Post-Conditions: The logs for one chat are displayed.
Basic Flow or Main Scenario: 1) The admin selects "View logs" from the admin controls menu. **2)** The admin selects a chat. **3)** The logs for the selected chat are displayed.
Extensions or Alternate Flows:
If no logs exist, a message appears stating "No Logs Exist".

Related Use Cases: User Login

Use Case ID: 011
Use Case Name: View Profile
Relevant Requirements: 3.1.5.6 User
Primary Actor: User/Admin
Pre-conditions: User: User is logged in, within a chat (group chat or private), and at least one chat exists.
Post-Conditions: The profile of a user is displayed with their information.
Basic Flow or Main Scenario: 1) User selects "View Message History" **2)** The User selects a username from the right end of the screen 3) Profile of selected user is displayed
Extensions or Alternate Flows: If the user whose profile has been selected does not exist(deleted user), the message "Profile/User does not exist" is shown.
Related Use Cases: View Message History

Use Case ID: 012
Use Case Name: Edit Profile
Relevant Requirements: 3.1.5 User Module. 3.1.8.4 Storage Module
Primary Actor: User
Pre-conditions: User is logged in.
Post-Conditions: User's profile information is updated
Basic Flow or Main Scenario: 1) User selects "Edit Profile" option **2)** User updates their name, bio, and/or password **3)** System authenticates the input **4)** System saves the updated information and will display the new information from now on
Extensions or Alternate Flows: 2b) if the input fails authentication, the system will prompt the user to enter valid input
Related Use Cases: View Profile, Create User

Use Case ID: 013
Use Case Name: Lock User Account
Relevant Requirements: 3.1.6 Admin Module Requirements
Primary Actor: Admin
Pre-conditions: Admin is logged in and there's a regular user account
Post-Conditions: User's account is locked and cannot log in
Basic Flow or Main Scenario: 1) Admin selects a user from the list of users **2)** Admin selects "Lock Account" option **3)** System locks the account and prevents the user from logging in
Extensions or Alternate Flows:
Related Use Cases: Delete User

Use Case ID: 014
Use Case Name: Flag/Hide Messages
Relevant Requirements: 3.1.6.7 Admin Module, 3.1.7 Message Handling Module, 3.1.9.7 GUI Module
Primary Actor: Admin
Pre-conditions: Admin is logged in and there is an inappropriate message
Post-Conditions: The flagged message is hidden from regular users chats and removed from chat logs
Basic Flow or Main Scenario: 1) Admin selects a chat to view **2)** Admin finds an inappropriate message **3)** Admin selects "flag message" option **4)** the system removes the message from the live chat and chat history without removing from logs
Extensions or Alternate Flows:
Related Use Cases: View Message History

Use Case ID: 015
Use Case Name: Sending Notifications
Relevant Requirements: 3.1.9 GUI Requirements
Primary Actor: Admin/System
Pre-conditions: Admin is logged in
Post-Conditions: System sends notification to users
Basic Flow or Main Scenario: 1) Admin selects notification type for certain user **2)** System sends notification to the user
Extensions or Alternate Flows:
Related Use Cases: Receiving Notifications

Use Case ID: 016
Use Case Name: Receiving Notifications
Relevant Requirements: 3.1.9 GUI Requirements, 3.1.2.4 Server Module Requirements
Primary Actor: User/System
Pre-conditions: User is logged in
Post-Conditions: User receives notification
Basic Flow or Main Scenario: 1) User receives notification from System **2)** user views new alert **3)** notification is removed
Extensions or Alternate Flows: 2b) user never views the alert **3b)** notification is not removed
Related Use Cases: Sending Notifications

Use Case ID: 017
Use Case Name: Add/Remove User from group chats
Relevant Requirements: 3.1.6.6 Admin Module
Primary Actor: Admin
Pre-conditions: Admin is logged in and there's a group chat with members
Post-Conditions: User(s) are added or removed from the group chat
Basic Flow or Main Scenario: 1) Admin selects the group chat **2)** Admin chooses user from list of users and adds them to the group
Extensions or Alternate Flows: 2b) Admin chooses user from group chat and removes them from the group
Related Use Cases:

Use Case ID: 018
Use Case Name: Mute Notifications
Relevant Requirements: 3.1.9 GUI Requirements
Primary Actor: User
Pre-conditions: The user is logged into the system and is part of active chat
Post-Conditions: Notifications for the selected chat are disabled, and the user will no longer receive notifications until unmuted
Basic Flow or Main Scenario: 1) The user chooses the chat they want to mute **2)** The user selects the "Mute Notifications" options from the chat settings **3)** The chat remains active, but the user won't receive any notifications until they manually unmute the chat
Extensions or Alternate Flows: 4b) The user can go back to the chat settings and select "Unmute Notifications" to receive notifications again
Related Use Cases: Receiving Notifications

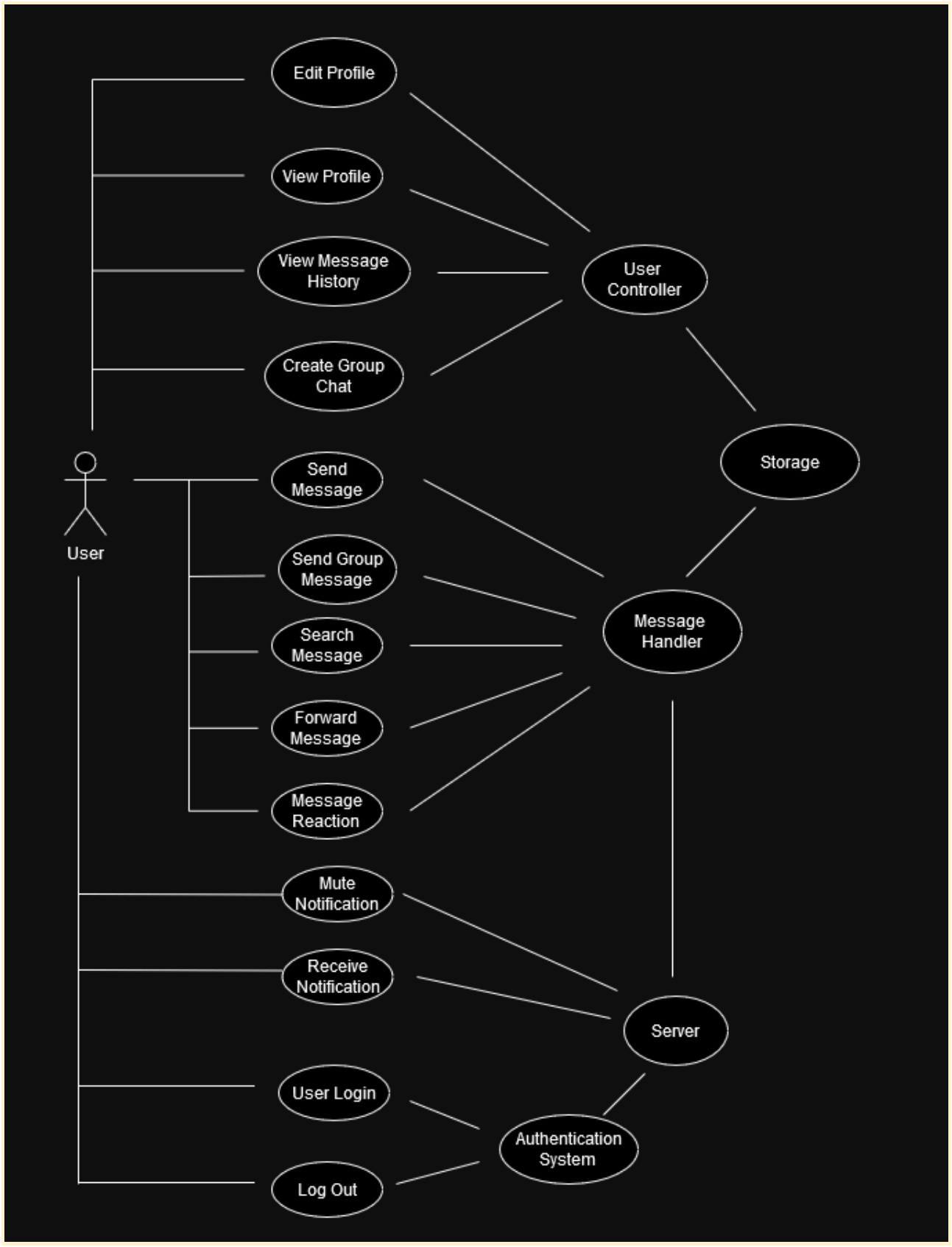
Use Case ID: 019
Use Case Name: Forward Message
Relevant Requirements: 3.1.7 Message Handling Module Requirements
Primary Actor: User

Pre-conditions: The user is logged in and has a message to forward
Post-Conditions: The message is sent to the selected recipients(s)
Basic Flow or Main Scenario: 1) User selects a message to forward **2)** User chooses the “Forward” option **3)** User chooses one or more recipients **4)** System sends the message to the selected recipients **5)** System confirms the message was forwarded
Extensions or Alternate Flows: 3b) The message is not forwarded, if the user cancels it **4b)** If forwarding fails due to network issues or the recipient(s) being unavailable, it will notify the user
Related Use Cases: Send Message, View Message History

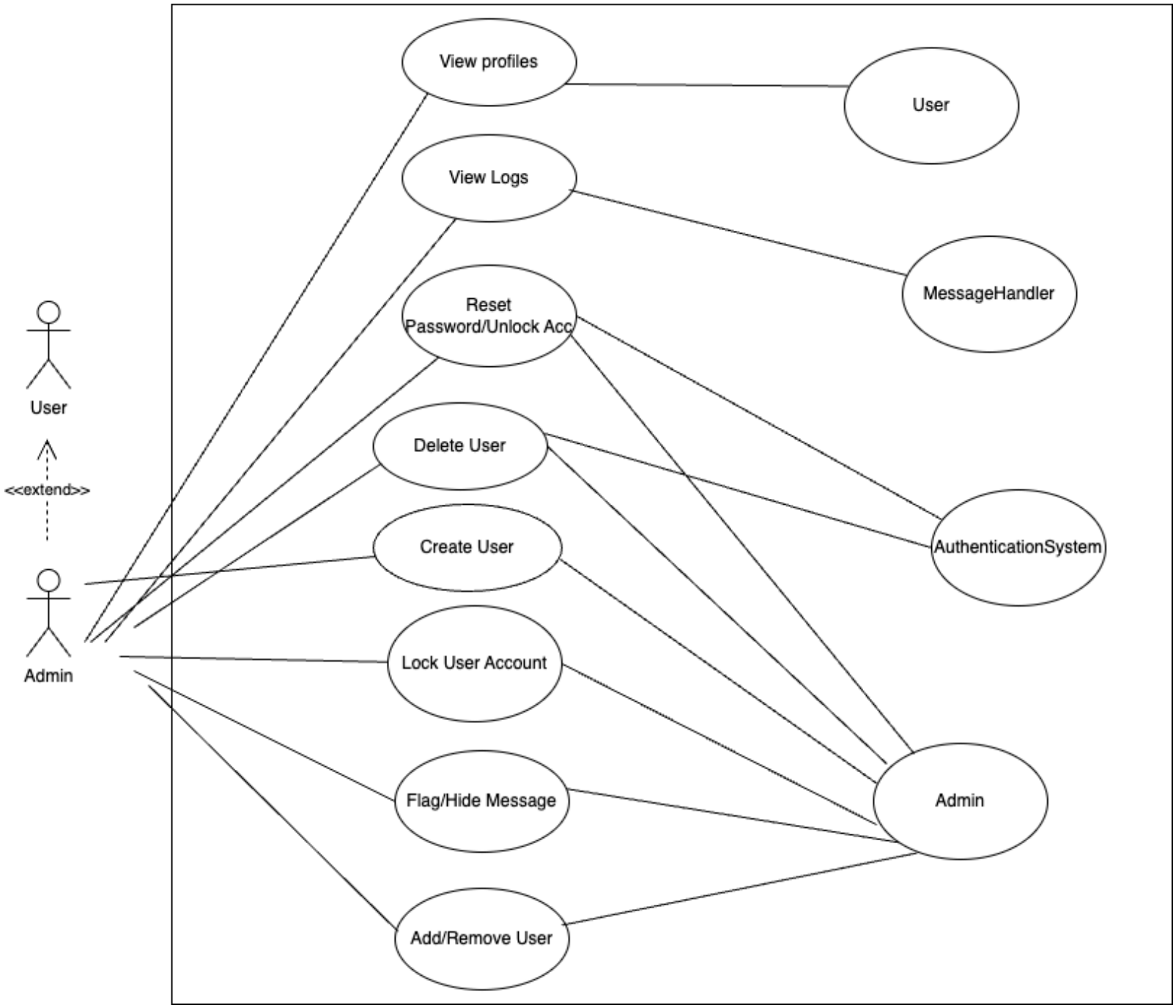
Use Case ID: 020
Use Case Name: React to Message
Relevant Requirements: 3.1.7 Message Handling Requirements
Primary Actor: User
Pre-conditions: User is logged in and has messages in the chat
Post-Conditions: The user’s reaction is recorded and displayed with the message
Basic Flow or Main Scenario: 1) The user views a message in the chat **2)** The user selects a reaction (e.g., thumbs up, thumbs down, heart) **3)** System updates the messages to show the selected reaction **4)** System confirms the reaction was added
Extensions or Alternate Flows: If the user removes the reaction, the system updates the message to reflect the change
Related Use Cases: Send Message, View Message History

Use Case Diagrams

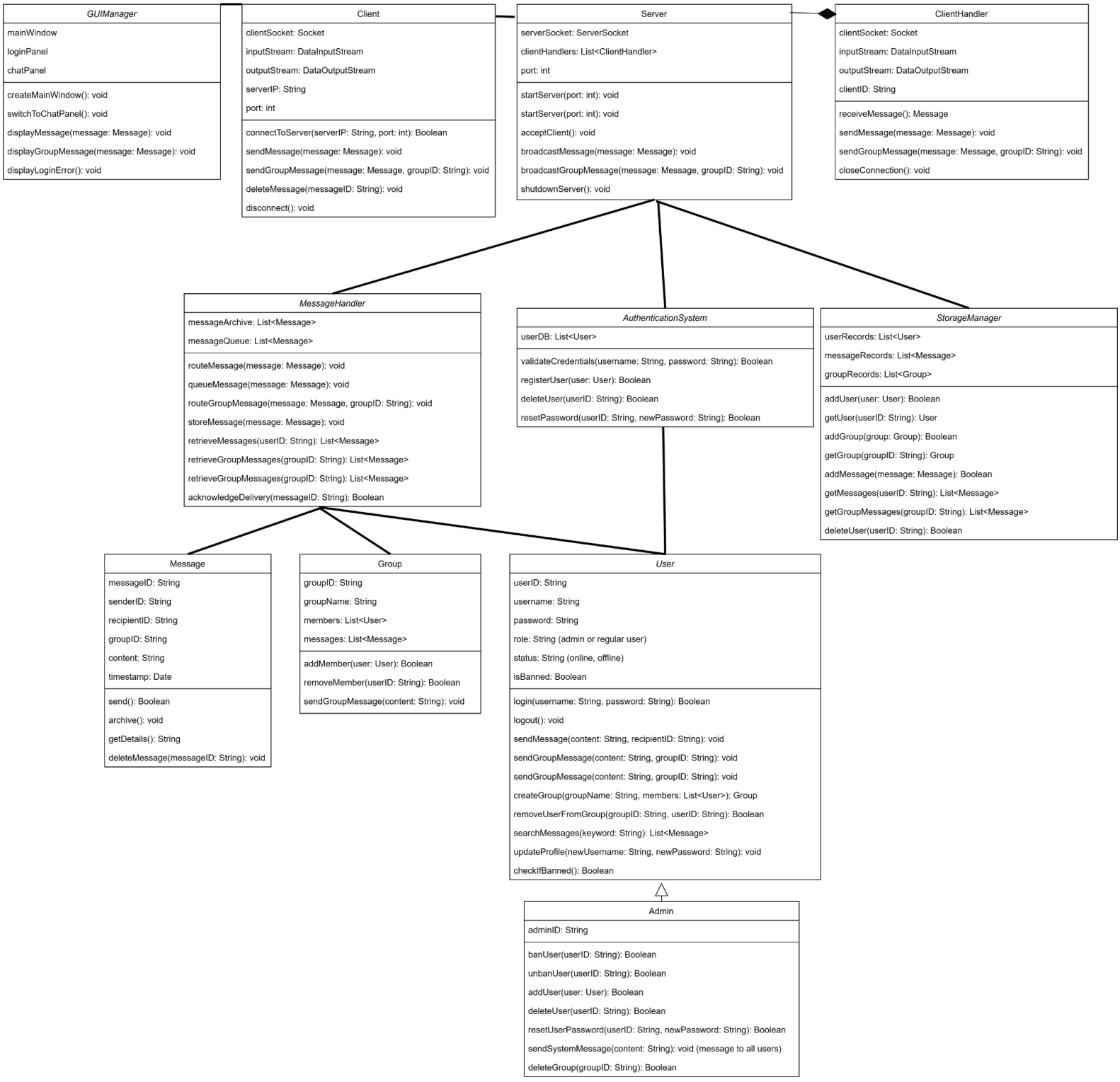
User Use-Case Diagram



Admin Use-Case Diagram

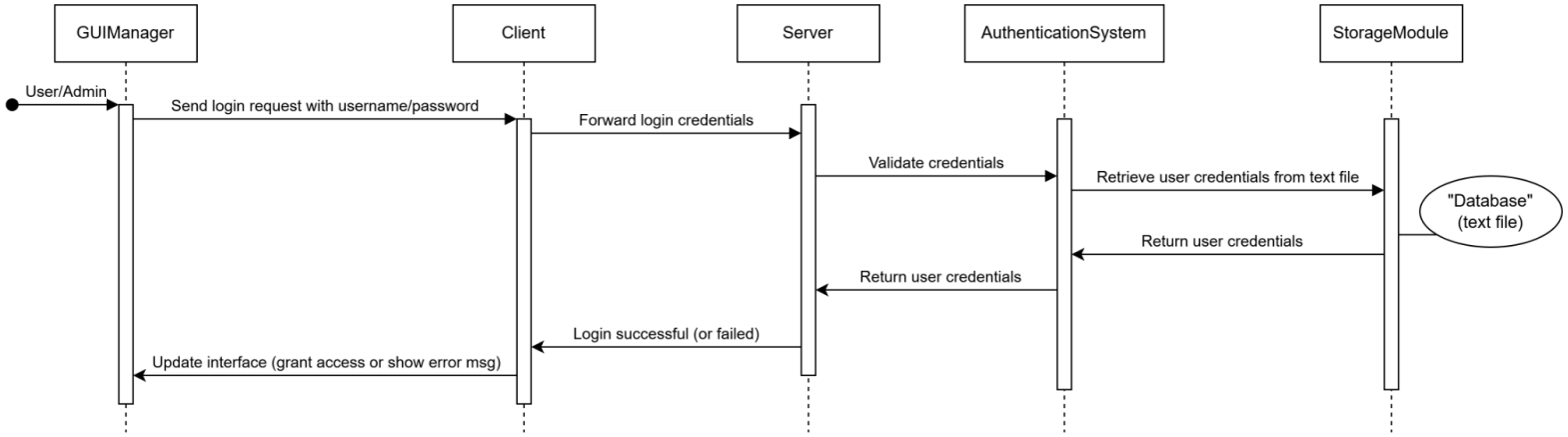


Class Diagram

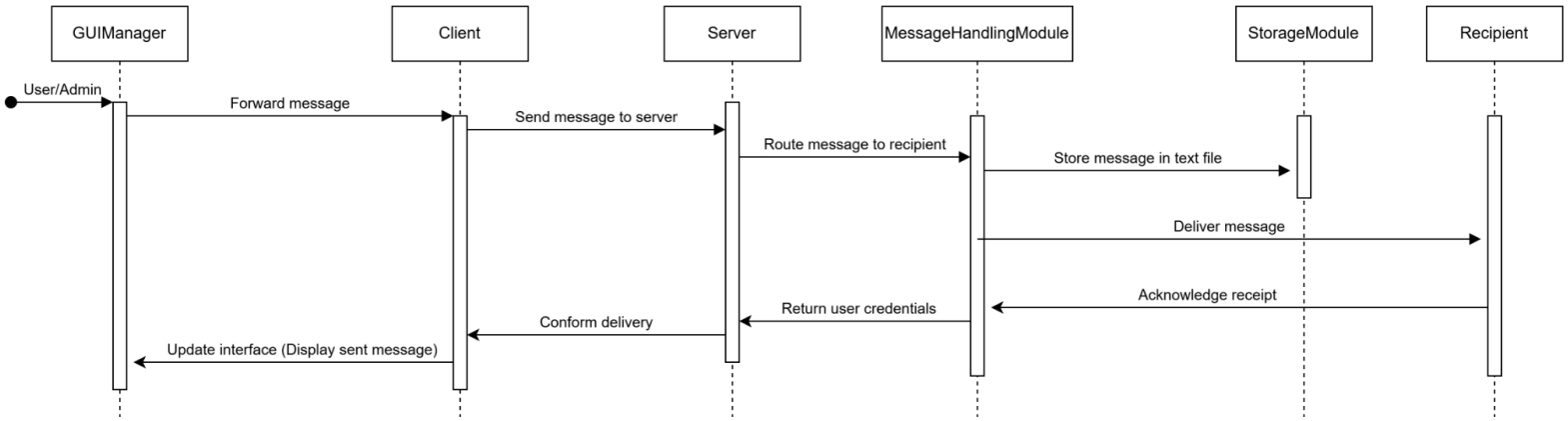


Sequence Diagrams

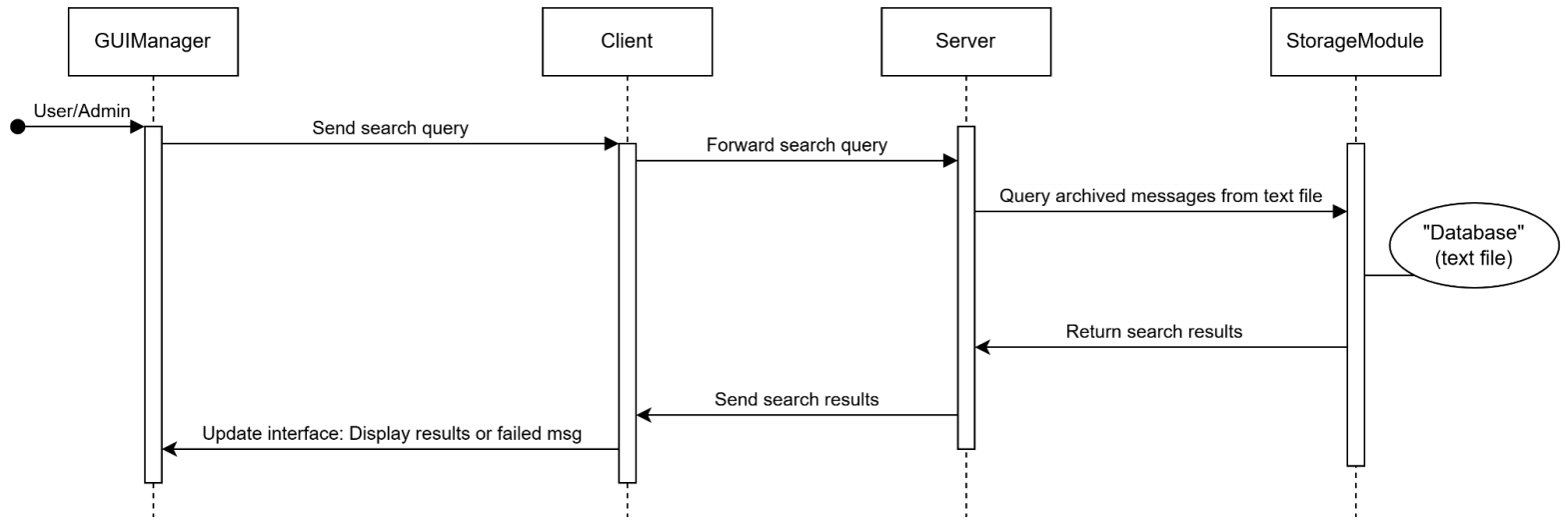
Use Case ID 001: User Login



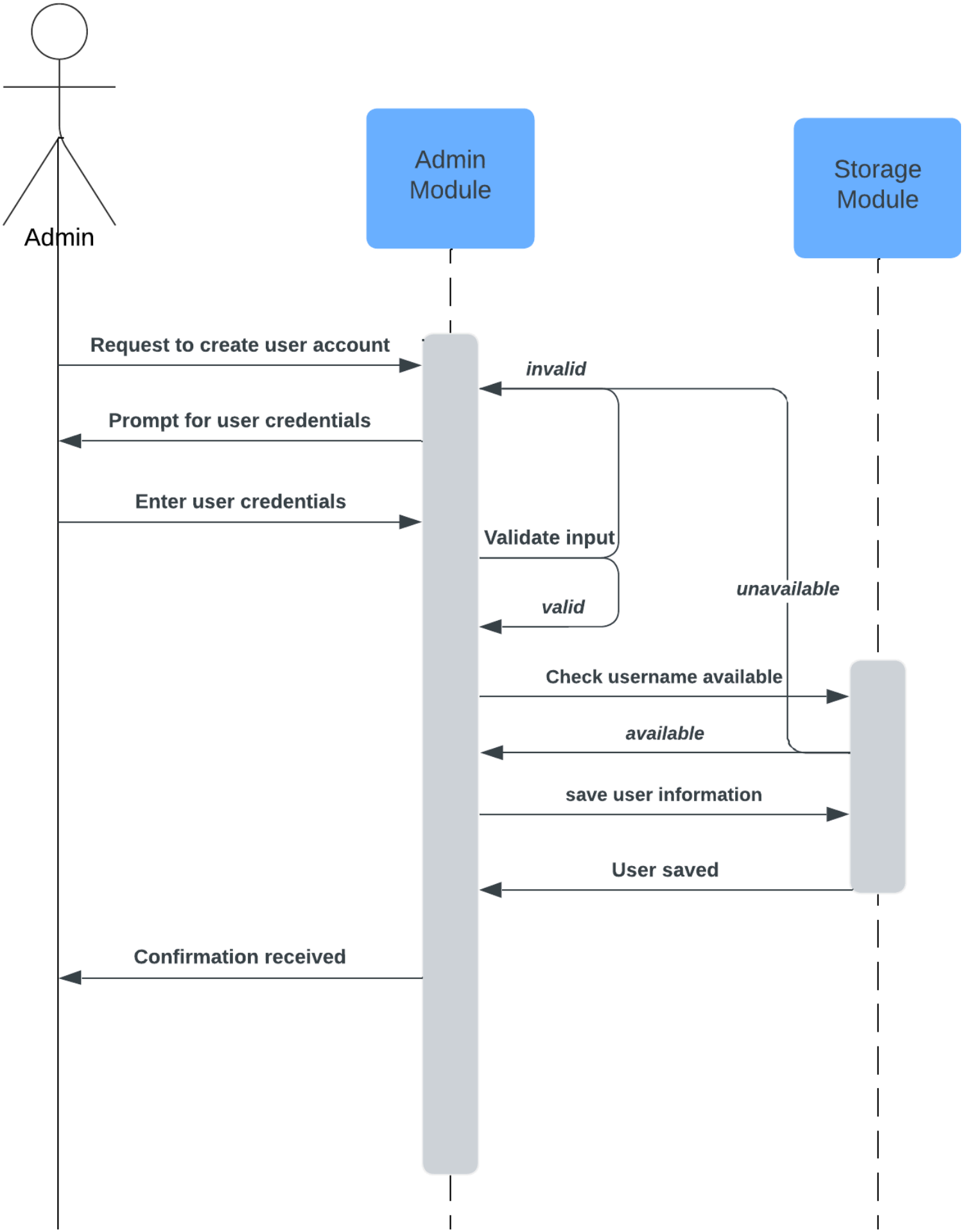
Use Case ID 002: Send Message



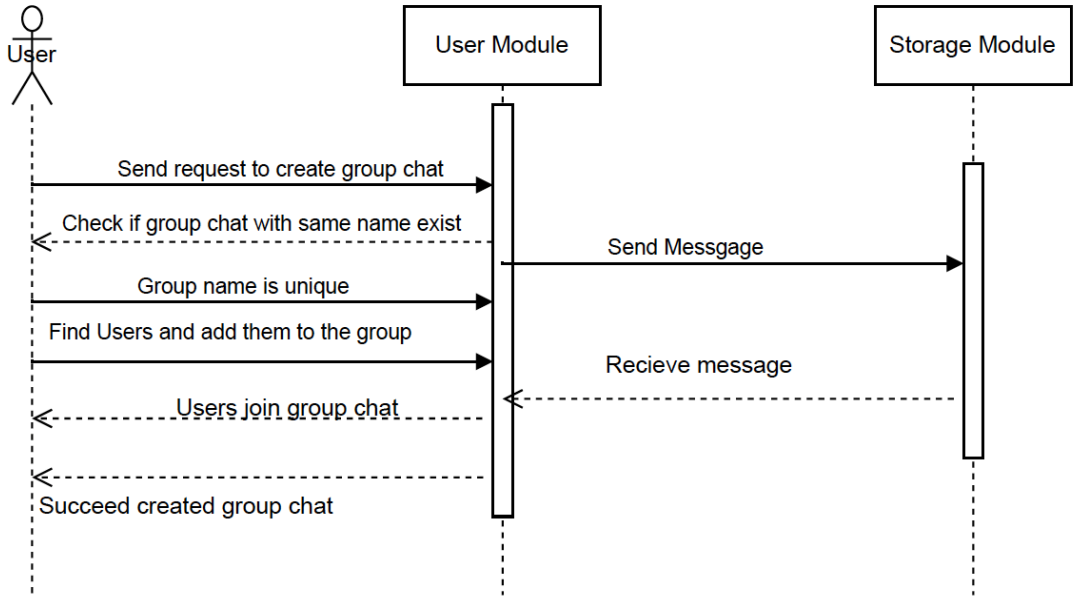
Use Case ID 003 AND 004: Search Message / View History



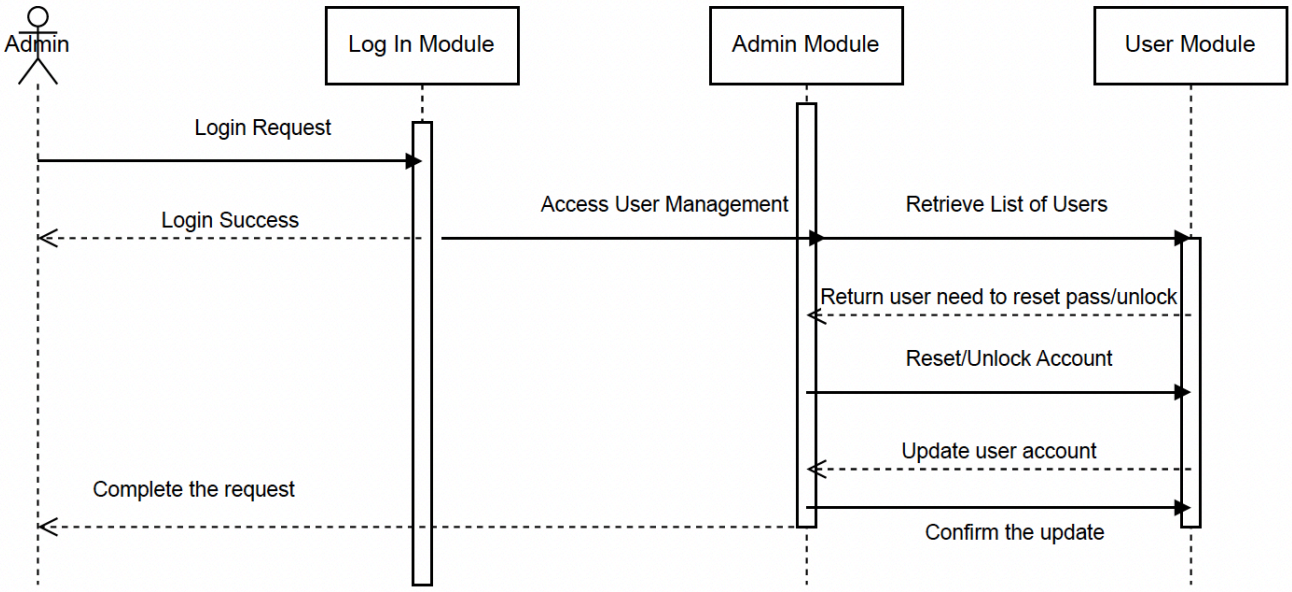
Use Case ID 005: Create User



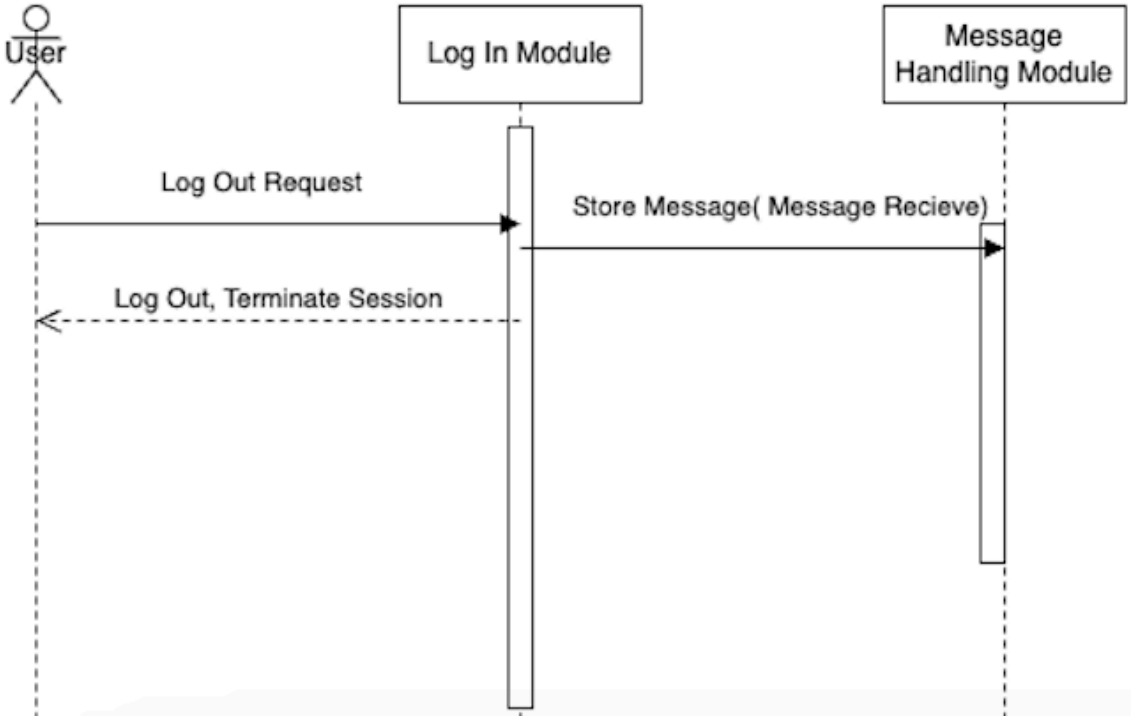
Use Case ID: 006



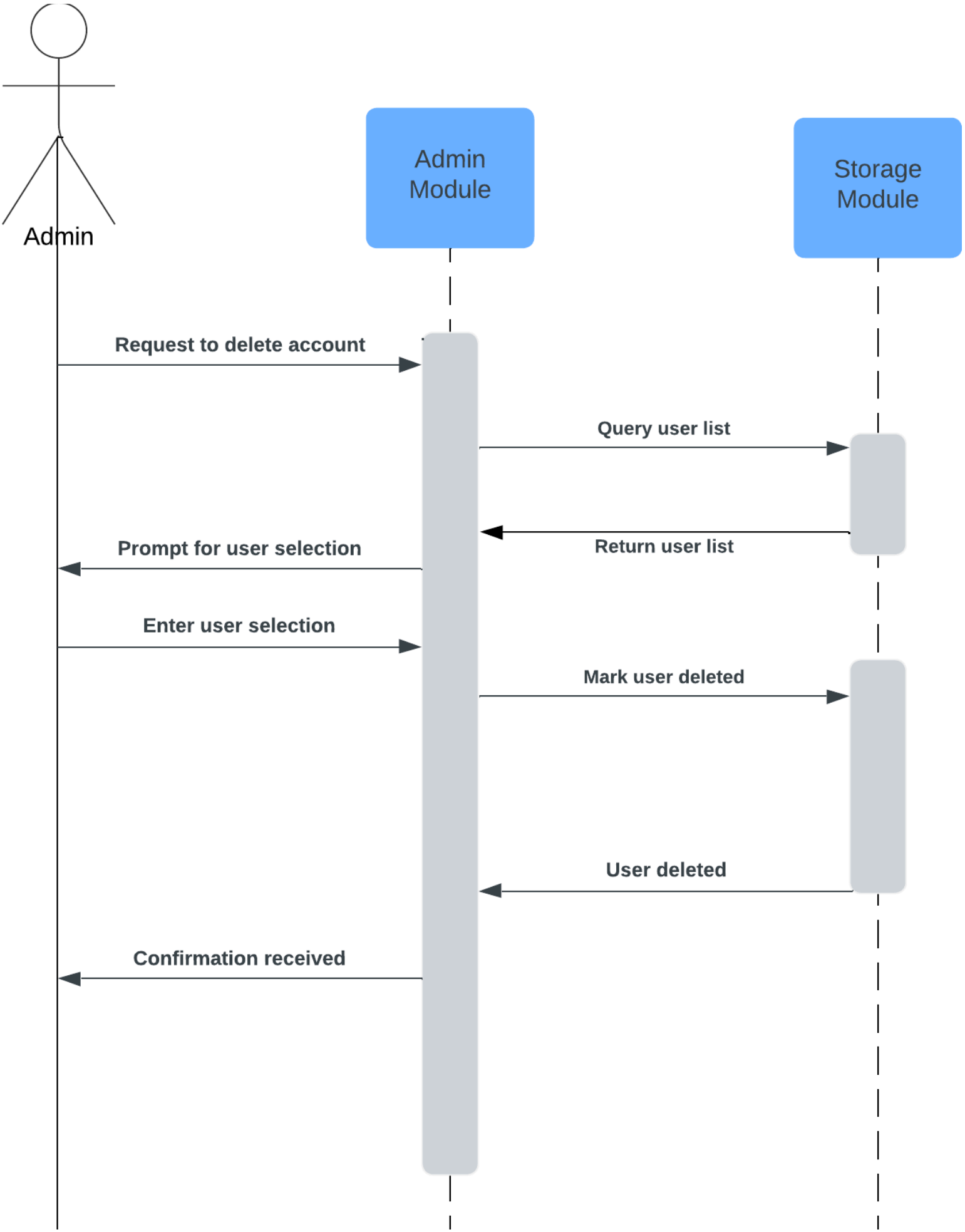
Use Case ID: 007



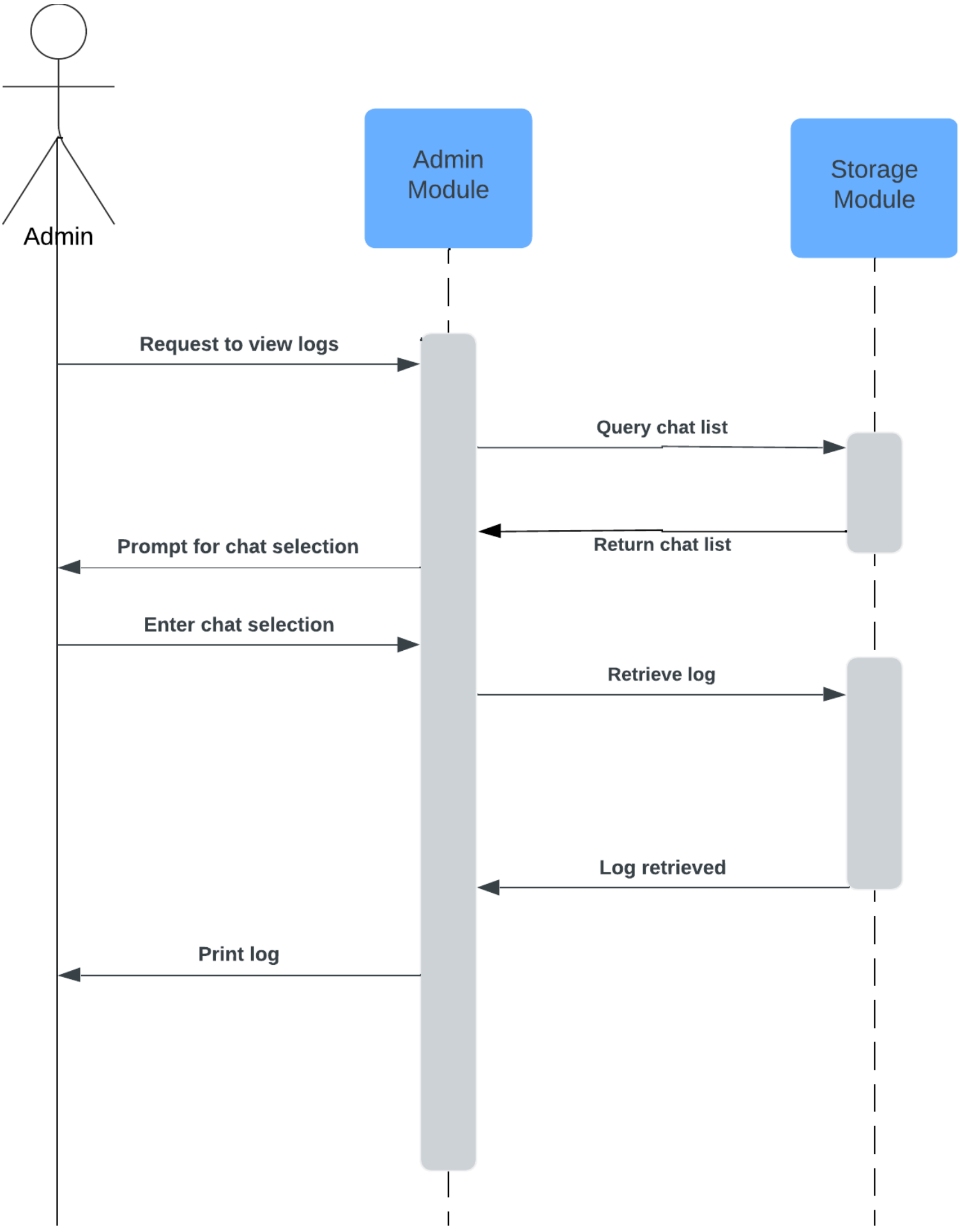
Use Case ID: 008



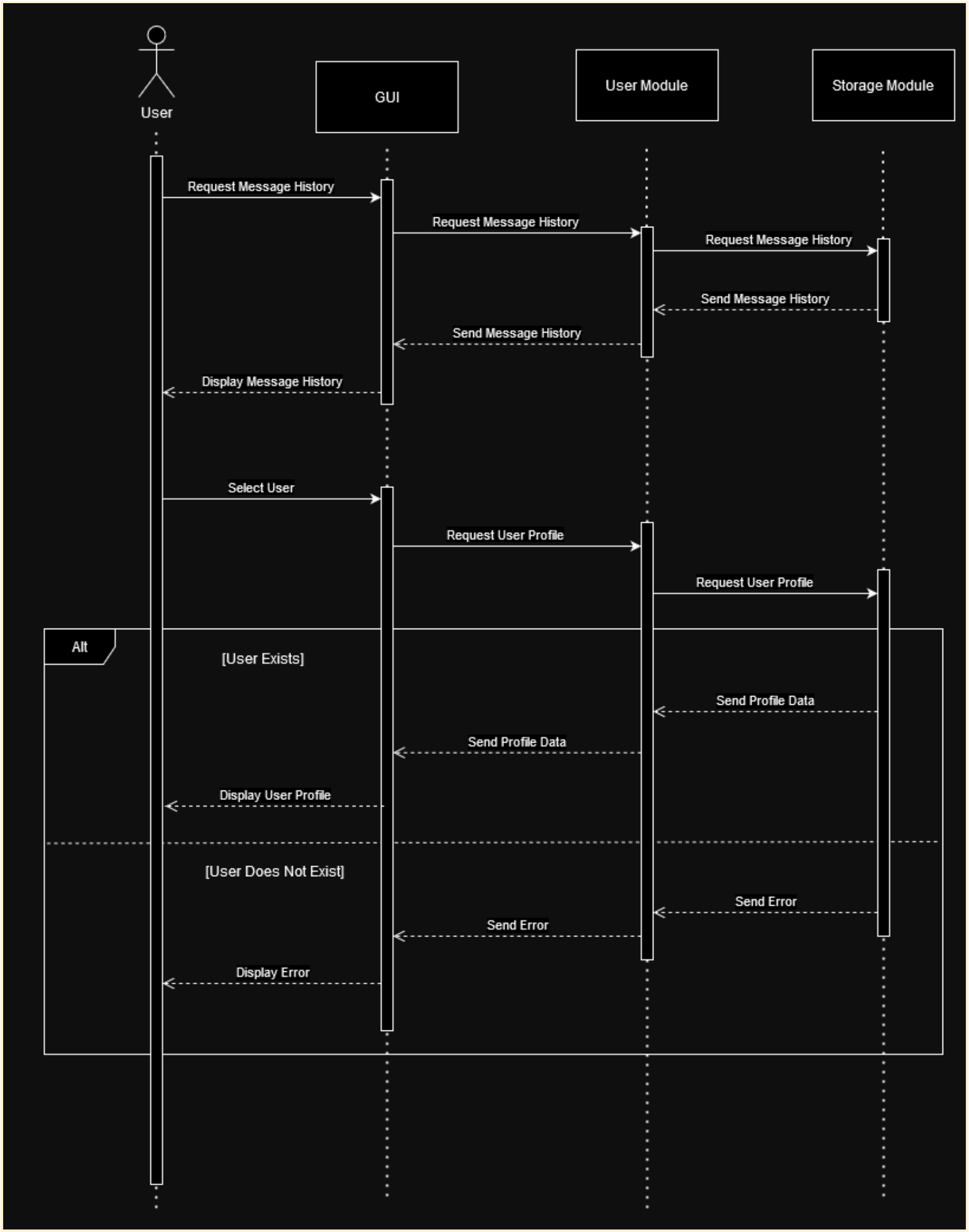
Use Case ID 009: Delete User



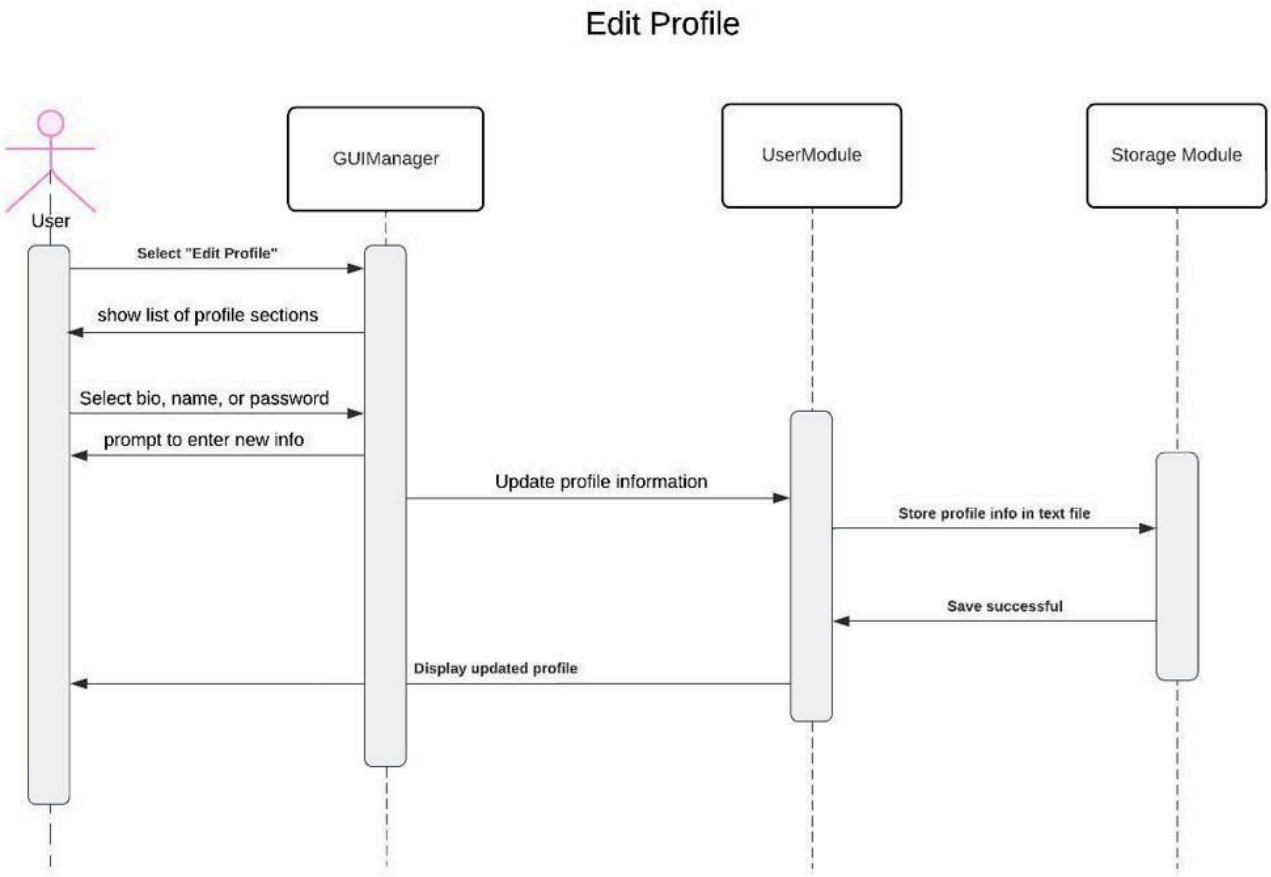
Use Case ID 010: View Logs



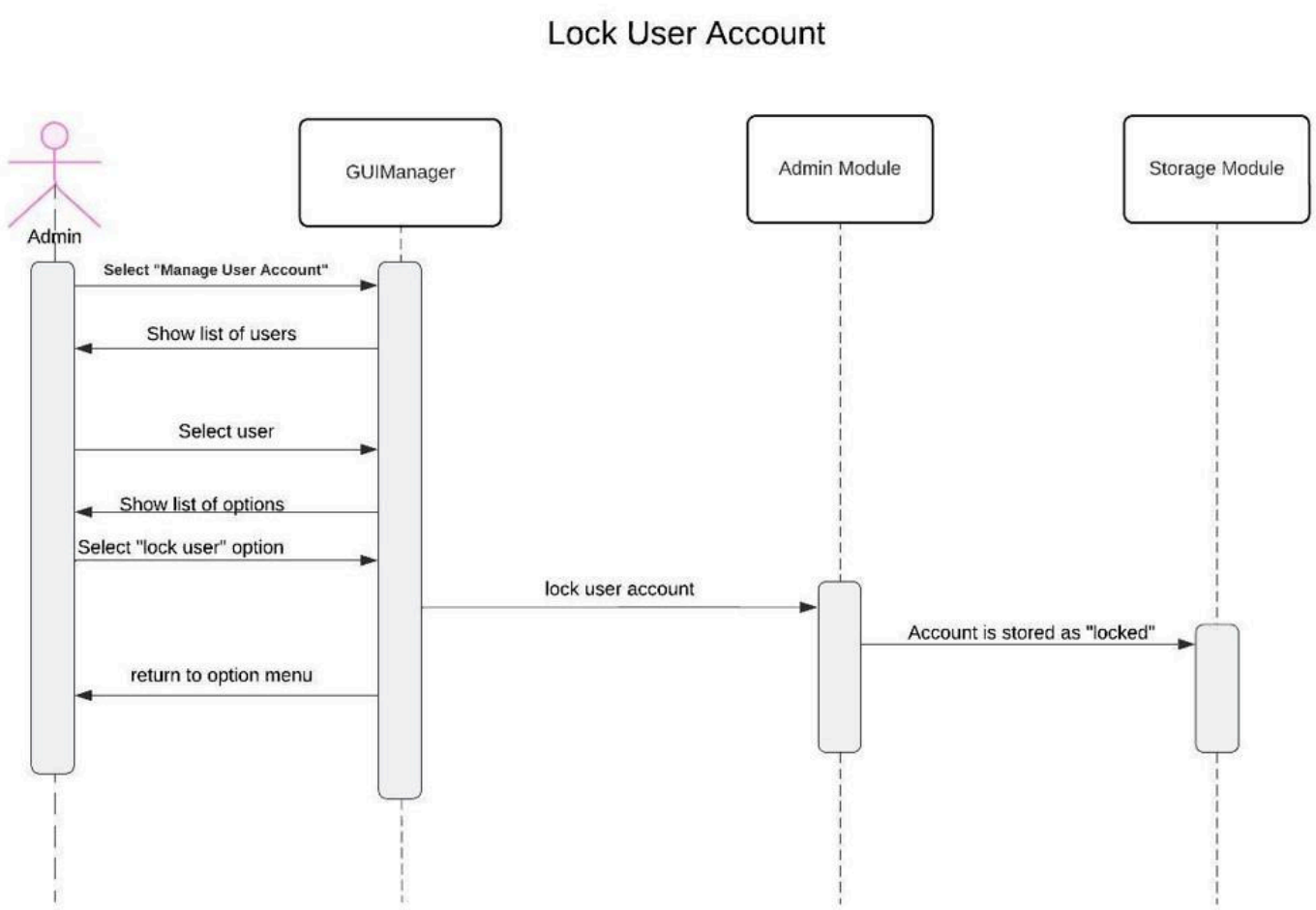
Use Case ID 011: View Profile



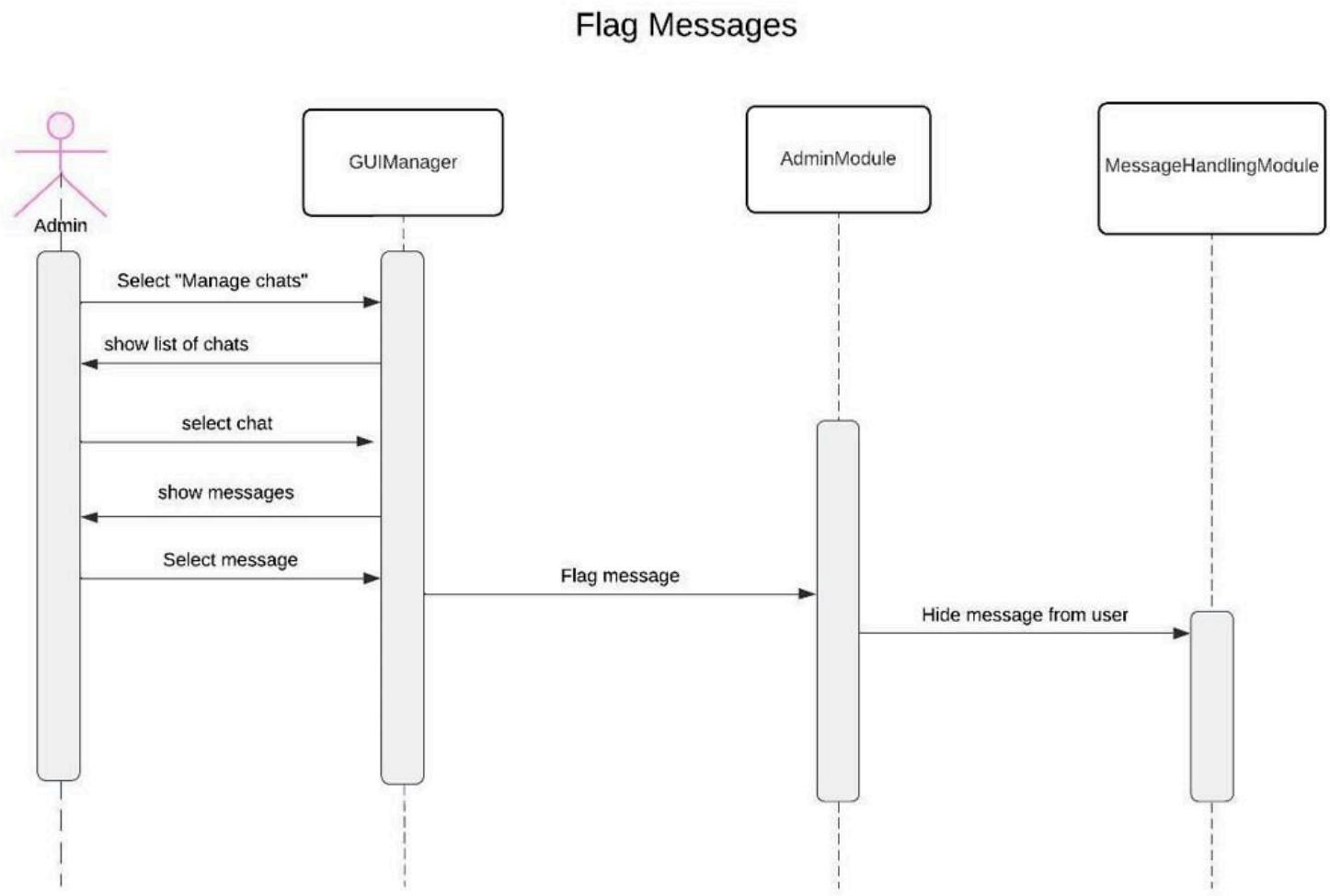
Use Case 012: Edit Profile



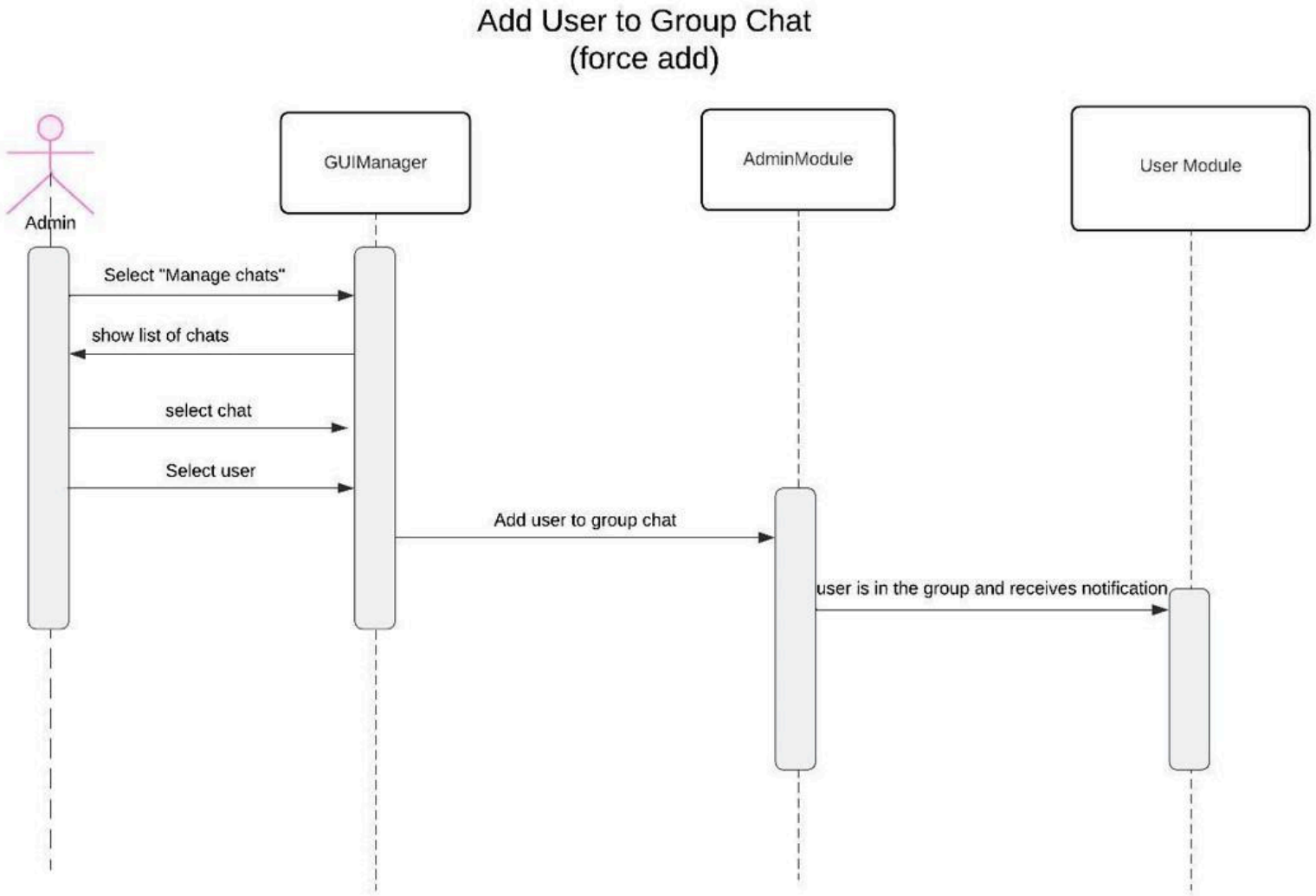
Use Case 013: Lock User Account



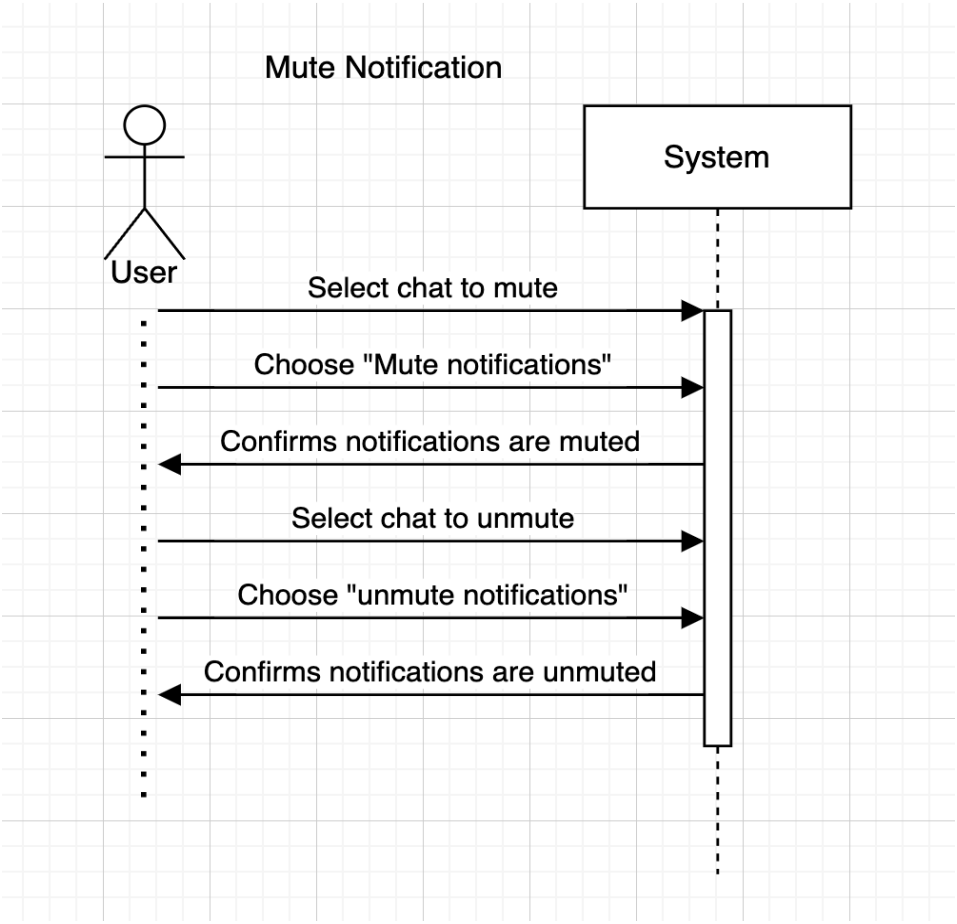
Use Case 014: Flag/Hide Messages



Use Case 017: Force Add User



Use Case 018: Mute Notification



Use Case 019: Forward Message