Communication Application

Software Requirements Specification

Revision History

Date	Revision	Description	Author
9/28/2025	1.0	Initial Version	Group 7

Table of Contents

1. Purpose

This document outlines the requirements for the Communication application

1.1. Scope

This document will catalog the user, system, and hardware requirements for the communication application. It will not, however, document how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

1.3. References

Use Case Specification Document – refer to pg. 9

UML Use Case Diagrams Document – refer to pg. 12

Class Diagrams – refer to pg. 13

Sequence Diagrams – refer to pg. 14

1.4. Overview

The application is a client-server messaging system designed to be scalable to a large number of users. The system supports both direct messaging between two users and group messaging with three or more participants. To ensure accountability and security, the application maintains immutable chat logs that record message content, sender and recipient IDs, and timestamps.

2. Overall Description

2.1. Product Perspective

2.2. Product Architecture

The system will be organized into 3 major modules: the User module, the Server module, and the Messaging module.

2.3. Product Functionality/Features

- **User Messaging**: Users can send and receive text-based messages in real time through the client application.
- **User Management**: The system supports account creation, login, logout, and session handling for secure access.
- **Notifications**: The client can display message alerts and status notifications to keep users informed of new activity.
- **Logging and Monitoring**: The server maintains logs of communication events, errors, and system status for monitoring and troubleshooting.
- **External Interfaces**: The system connects over TCP/IP, ensuring compatibility across different platforms and network environments.
- **Scalability and Reliability**: The architecture supports multiple concurrent clients and ensures reliable message delivery.
- **Security**: User data and communications are protected through authentication and encrypted transmission.

2.4. Constraints

The program should not include any databases, libraries, frameworks, or other technologies.

2.5. Assumptions and Dependencies

It is assumed that the server is running when a user wants to connect.

The Users device is dependent on have a Java Runtime Environment to run the Program.

It is assumed that the User device has a network interface card to connect to the server.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Privacy will be minimized in the system design where all administrators will have read access to all chat logs but must not be able to modify the logs.

3.1.1.2 The system should support both synchronous and asynchronous communication.

3.1.1.3 The application should be scalable to a large number of users with little or no difference in runtime.

3.1.1.4 All users should be able to interact with the application through a GUI implementation.

3.1.2. User Module Requirements:

3.1.2.1 Users should be allowed to log in using their issued ID and PIN, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.2.2 IT/Admins should be able to log in through the same system.

3.1.2.3 IT users should be able to create new users, picking a unique username, password, and ID.

3.1.2.4 Users should be notified of new messages.

3.1.2.5 Users will be assigned roles (IT or general user) upon successful login using credentials provided by IT

3.1.3. Server Module Requirements:

3.1.3.1 Chat logs should be immutable and cannot be deleted.

3.1.3.2 Chat logs should be accessible asynchronously.

3.1.3.3 Chat logs should only be accessible by users with IT/Admin privileges.

3.1.4. Messages Module Requirements:

3.1.4.1 Messages should only contain text supported by the UTF-8 standard.

3.1.4.2 Once added to a chat, users should not be able to leave the conversation.

3.1.4.3 Users should be able to exchange direct messages between two users or participate in group chats with three or more users.

3.1.4.4 Each message should be logged with a timestamp, sender ID, recipient ID(s), and message content.

3.1.4.5 Messages should be limited to a maximum of 1000 characters.

3.2. External Interface Requirements

3.2.1 The communication should happen over TCP/IP.

3.3. Internal Interface Requirements

3.3.1 Error handling and retry mechanisms should be defined to prevent message loss or corruption.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

Example:
4.1.1 The System must encrypt data being transmitted over the Internet.

4.2. Environmental Requirements

Example:
4.2.1 System cannot require that any software other than a web browser be installed on user computers.
4.2.2 System must make use of the University’s existing Oracle 9i implementation for its database.
4.2.3 System must be deployed on existing Linux-based server infrastructure.

4.3. Performance Requirements

Example:
4.3.1 System must render all UI pages in no more than 9 seconds for dynamic pages. Static pages (HTML-only) must be rendered in less than 3 seconds.

Use Case Specification

Use Case ID: UC-01

Use Case Name: Login
Relevant Requirements: Device is connected to wifi, and has typing/mouse capabilities. Minimal hardware requirements.
Primary Actor: General user (does not vary per level of staff) or IT
Preconditions: User account exists; system is online
Postconditions: User authenticated; role assigned (IT/regular user); chat dashboard displayed
Basic Flow:

1. User enters credentials.
2. System verifies credentials.
3. System assigns role (IT or regular user).
4. Dashboard is displayed. If IT role, additional abilities are given in addition.
   Extensions/Alternate Flows: Invalid credentials → error/warning shown; device lockout after 3 failed attempts. Can be manually allowed to login via IT, or accounts can be locked.

Use Case ID: UC-02

Use Case Name: Viewing logs
Relevant Requirements: User was properly assigned IT role and access in UC-01
Primary Actor: IT, System
Preconditions: IT user exists and properly logged into system that is properly connected to device
Postconditions: IT user gets *TEXT or System.out.println()?* Of chosen user's logs, either with another user or in general?
Basic Flow: The user IT types into a prompt is chosen, system accesses the logs of the user, and then prints/displays logged chats.

1. The user IT types into a prompt is chosen

2. System accesses chat logs of user
3. Computer prints accessed chat logs
   Extensions/Alternate Flows:

Use Case ID: UC-03

Use Case Name: Create group
Relevant Requirements: User properly logged in and given user role, and has typing/mouse capabilities
Primary Actor: User, including IT
Preconditions: Properly able to interact with interface, and has other user ID's of other group chat members
Postconditions: a chat system that includes and sends/receives messages to all members of the group chat
Basic Flow:

1. Click "create group" button
2. Types multiple users to add to group chat and click "create"

3. Group chat is properly initialized
   Extensions/Alternate Flows: Invalid user→ error/user ID not found

Use Case ID: UC-04

Use Case Name: Create new user
Relevant Requirements: IT user, and has enough info on new user to create them
Primary Actor: IT
Preconditions: Ready user id/password
Postconditions: New user with the given user ID/password, and potentially any metadata and whether they are also IT
Basic Flow:

1. IT creates user with information given
2. User is created along with metadata and roles given.
3.
4.
   Extensions/Alternate Flows: Existing user credentials already, does not create a new user and warns IT that attempted creating user.

Use Case ID: UC-05

Use Case Name: Send Message
Relevant Requirements: Only IT can create users. System supports role assignment and password.
Primary Actor: User, including IT
Preconditions: Sender logged in; recipient (user or group) resolvable. Sender has permission to message recipient.
Postconditions: Message persisted and timestamped. Delivery enqueued to recipient(s); sender sees "sent" status. Event available to logging subsystem (UC-07).
Basic Flow:

1. User types message and sends
2. Message is sent through system to targetted users and logged in UC-07
   Extensions/Alternate Flows: Network/server error → "failed to send", retry option

Use Case ID: UC-06

Use Case Name: Receive message
Relevant Requirements: Real-time delivery (websocket/push). User is online or offline with queued delivery.
Primary Actor: User or IT
Preconditions: Recipient account exists and is addressable. Message previously sent (UC-05) and accepted by system.
Postconditions: Message appears in recipient's chat view. Delivery/read receipts updated as applicable. Notification triggered (toast/badge/sound) per user settings.
Basic Flow:

1. Message is sent to server, and sent to recipient user

2. Recipient user receives notification that a new message from sender user ID
3. Recipient can read message
   Extensions/Alternate Flows: Log store unavailable warning of system issue to user; retry later and alert IT. Storage near limit, throttle non-critical logs and alert IT.

Use Case ID: UC-07

Use Case Name: Log message
Relevant Requirements: Device that UC-05 is connected to system and internet
Primary Actor: System, user (IT included)
Preconditions: A message is sent/received (UC-05/UC-06). Logging feature enabled and reachable.
Postconditions: Message metadata written to log store with timestamps and IDs. Logs stored in an array for search (UC-02). Failures are captured and alerted to IT.
Basic Flow:

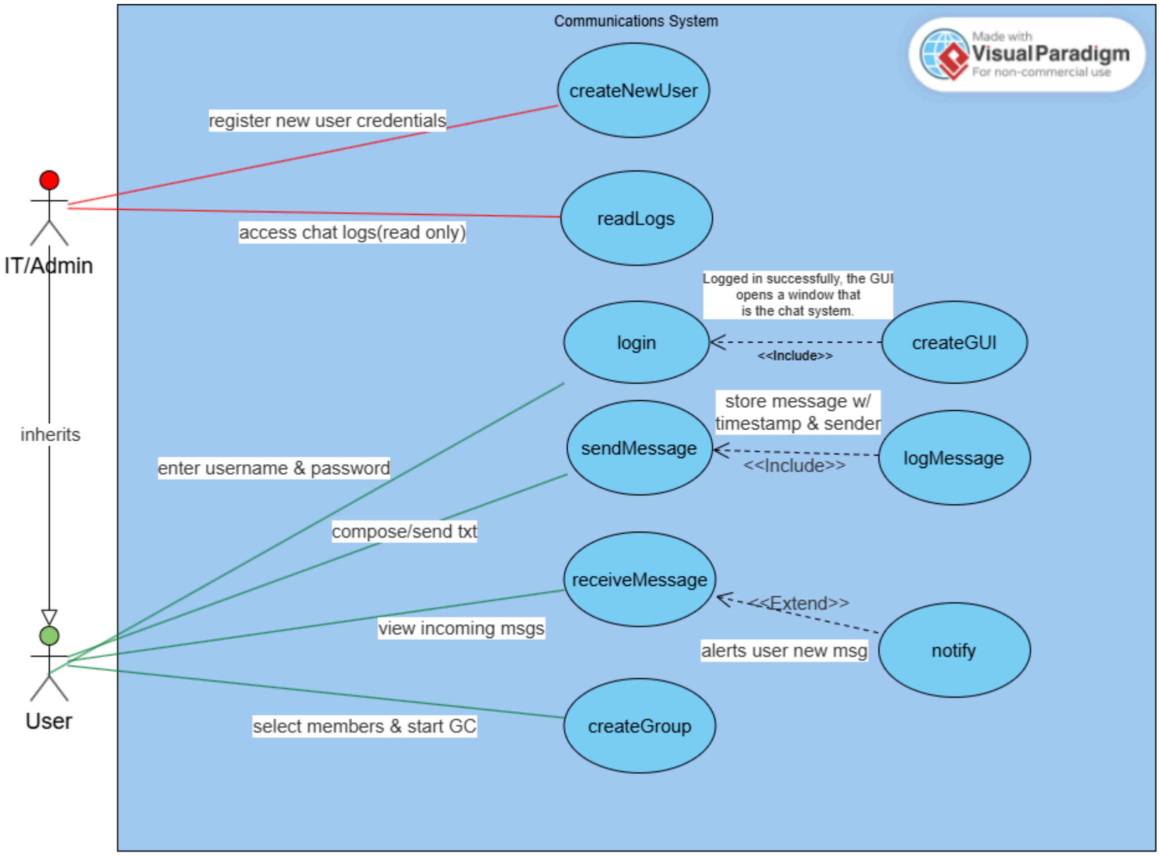
1. Message successfully sent and received, and sent to be logged
2. Message is appended to array to be later retrieved.
   Extensions/Alternate Flows: No memory available, IT warned and future messaging suspended

Use Case ID: UC-08

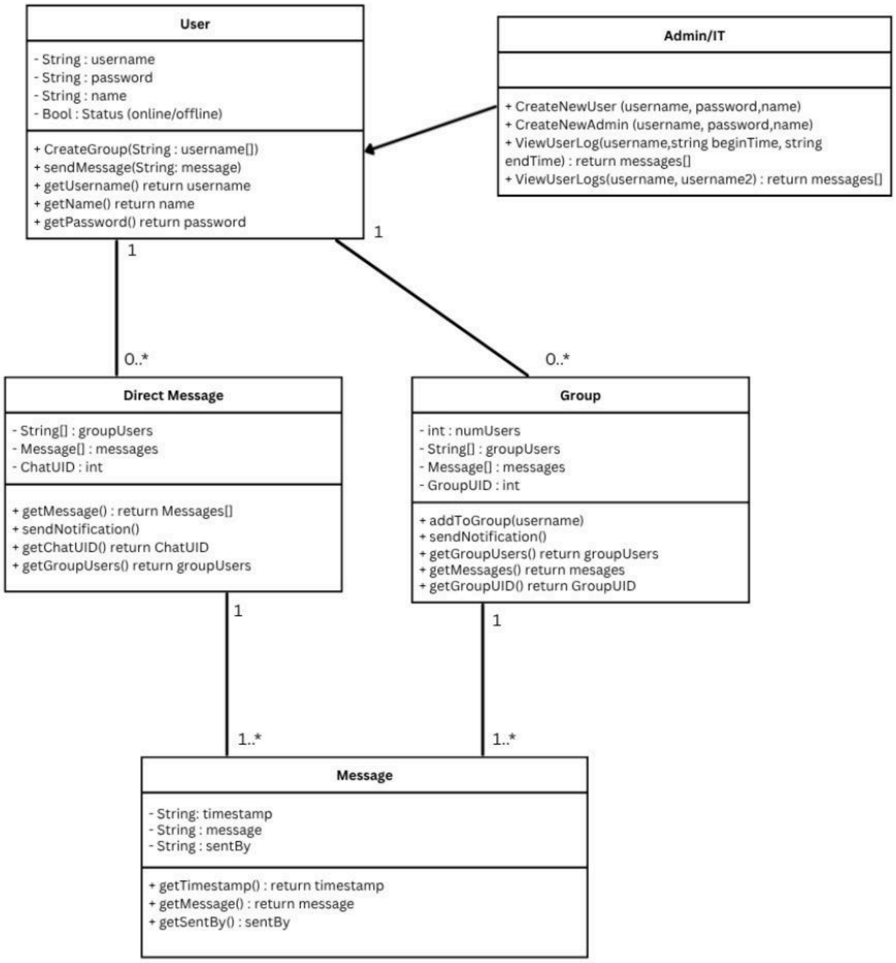
Use Case Name: Create GUI
Relevant Requirements: Used device has the Swing/JavaFX libraries in its java version installed
Primary Actor: Software, upon successful UC-01, with extra if given role is IT
Preconditions: UC-01 properly logged in, and software properly starts up.
Postconditions: Interface to chat properly begins, and interface for IT is also given, to pull up a user's cha
Basic Flow:

1. Logged in successfully, the GUI opens a window that is the chat system.
2. Depending on IT or not, allowed additional features for chat log access and user creation

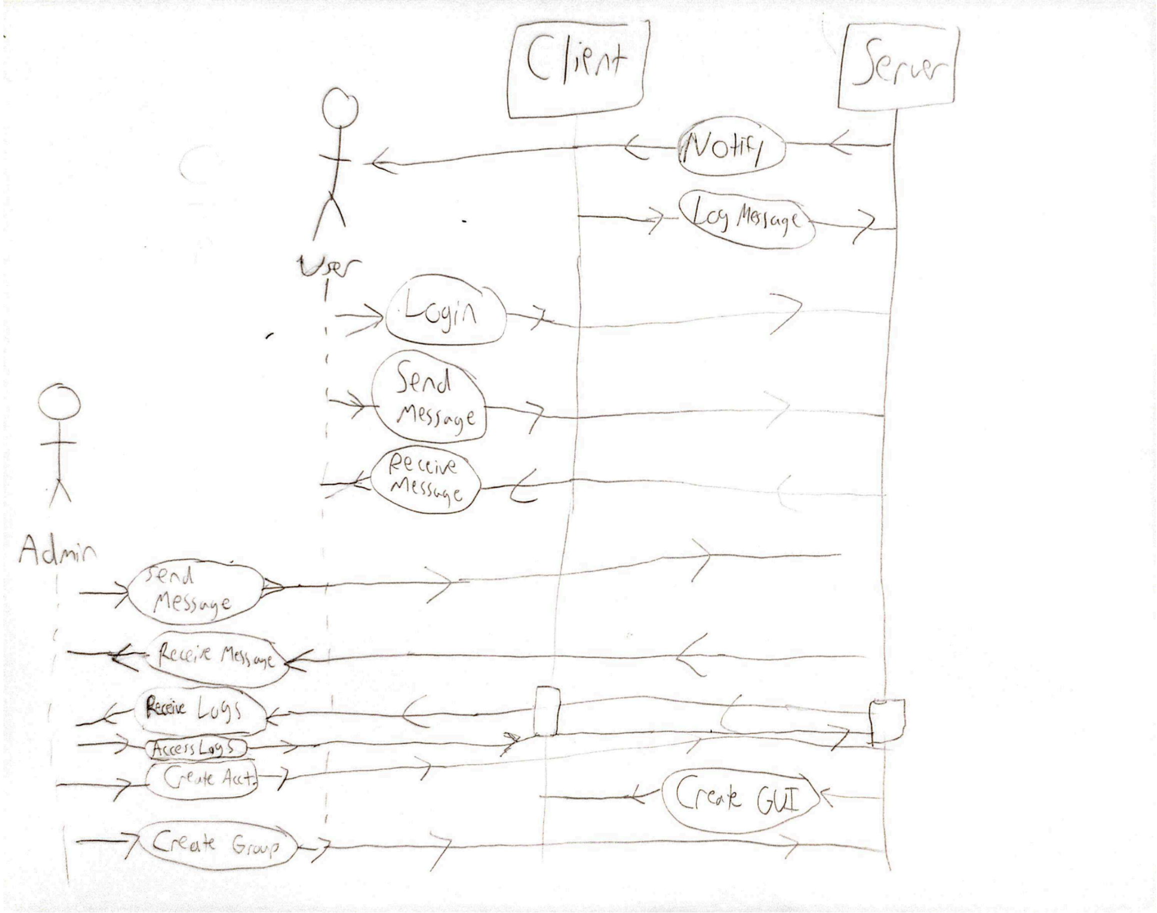
Use Case Diagrams


Communications System
Made with Visual Paradigm For non-commercial use
createNewUser
register new user credentials
readLogs
access chat logs(read only)
IT/Admin
Logged in successfully, the GUI opens a window that is the chat system.
login
<<Include>>
createGUI
inherits
store message w/ timestamp & sender
sendMessage
<<Include>>
logMessage
enter username & password
compose/send txt
receiveMessage
<<Extend>>
view incoming msgs
alerts user new msg
notify
User
select members & start GC
createGroup

Class Diagrams


User
- String : username
- String : password
- String : name
- Bool : Status (online/offline)
+ CreateGroup(String : username[])
+ sendMessage(String: message)
+ getUsername() return username
+ getName() return name
+ getPassword() return password
Admin/IT
+ CreateNewUser (username, password,name)
+ CreateNewAdmin (username, password,name)
+ ViewUserLog(username,string beginTime, string endTime) : return messages[]
+ ViewUserLogs(username, username2) : return messages[]
1
1
0..*
0..*
Direct Message
- String[] : groupUsers
- Message[] : messages
- ChatUID : int
+ getMessage() : return Messages[]
+ sendNotification()
+ getChatUID() return ChatUID
+ getGroupUsers() return groupUsers
Group
- int : numUsers
- String[] : groupUsers
- Message[] : messages
- GroupUID : int
+ addToGroup(username)
+ sendNotification()
+ getGroupUsers() return groupUsers
+ getMessages() return mesages
+ getGroupUID() return GroupUID
1
1
1..*
1..*
Message
- String: timestamp
- String : message
- String : sentBy
+ getTimestamp() : return timestamp
+ getMessage() : return message
+ getSentBy() : sentBy

Sequence Diagrams


Client
Server
User
Notify
Log Message
Login
Send Message
Receive Message
Admin
Send Message
Receive Message
Receive Logs
Access Logs
Create Acct.
Create GUI
Create Group